Nonvisual Presentation of Graphical User Interfaces: Contrasting Two Approaches

Elizabeth D. Mynatt
Graphics, Visualization and Usability Center
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
404-894-3658
beth@cc.gatech.edu

Gerhard Weber
Universitäe Stuttgart, Institut füe Informatik,
and F.H.Papenmeier GmbH&Co, KG, Schwerte
Institut füe Informatik, Breitwiesenstr.
20-22, 70565 Stuttgart, F.R. Germany
weber@dia.informatik.uni-stuttgart.de

ABSTRACT
Users who are blind currently have limited access to graphical user interfaces based on MS Windows or X Windows. Past access strategies have used speech synthesizers and braille displays to present text-based interfaces. Providing access to graphical applications creates new human interface design challenges which must be addressed to build intuitive and efficient nonvisual interfaces. Two contrasting designs have been developed and implemented in the projects Mercator and GUIB. These systems differ dramatically in their approaches to providing nonvisual interfaces to GUIs. This paper discusses four main interface design issues for access systems, and describes how the Mercator and GUIB designs have addressed these issues. It is hoped that the exploration of these interfaces will lead to better nonvisual interfaces used in low visibility and visually overloaded environments.

KEYWORDS Nonvisual HCI, blind users, graphical user interfaces, auditory interfaces, tactile interfaces

INTRODUCTION

Alan Newell's plenary address at INTERCHI '93, titled "CHI for Everyone," argued that by extending our vision of interface design to encompass extraordinary users, we would not be limiting the applicability of our work. Instead, we would discover and refine new interaction techniques which would be of use to the general user community. A current interface design challenge is developing interfaces which provide access to graphical user interfaces for people who are blind [1]. Even the goal itself sounds like an oxymoron. The design issues in translating an interactive, spatially presented, visually dense interface into an efficient, intuitive and non-intrusive nonvisual interface are numerous. Moreover, practical concerns such as using affordable hardware while providing access to many application interfaces transparently to the graphical applications adds to the complexity of the task [18].

The typical scenario to providing access to a graphical interface is as follows: While an unmodified graphical application is running, an outside agent (or screen reader) collects information about the application interface by watching objects drawn to the screen and by monitoring the application behavior. This screen reader then translates the graphical interface into a nonvisual interface, not only translating the graphical presentation into an nonvisual presentation, but providing different user input mechanisms as well.

This paper presents two contrasting approaches to GUI access. The GUIB (Textual and Graphical User Interfaces for Blind People) Project is a consortium of partners from six European countries which is developing a prototype implementation of an access system for MS Windows and X Windows [17]. The GUIB design is based on translating the screen contents into a tactile presentation which is based on the spatial organization of the graphical interface. GUIB uses a new input/output device, called GUIDE, which integrates vertical and horizontal braille displays, two loudspeakers and a touch-sensitive tablet. This device allows blind users to experiment with direct manipulation and 2D spatial sound presentation.

The Mercator project is an interdisciplinary research effort, at the Georgia Institute of Technology which is exploring different access strategies for X Windows [10]. Mercator replaces the spatial graphical display with a hierarchial auditory interface. Minimizing the use of special-purpose hardware, Mercator only adds a speech synthesis system to the standard desktop configuration. Mercator extensively uses nonspeech auditory cues to convey iconic information presented in the graphical user interface.

Both approaches recognize many important points. First, access to graphical user interfaces is a critical concern for the visually-impaired community. In the past, command-line, text-based interfaces were accessible through the use of screen reader software which transmitted text to a speech synthesizer or a braille display. Now graphical user interfaces create new barriers to access through the use of spatially separated windows, icons, and direct manipulation. Second, both projects recognize the need to design a complete solution which takes into account not only the presentation of graphical information, but also interaction with the nonvisual interface. Previous access system simply focused on translating the on-screen graphics without providing new interaction techniques appropriate to the nonvisual presentation [13].

The designs of Mercator and GUIB are also radically different. Some of the differences are the result of varying preferences between U.S. and European users. In North America, the use of speech synthesizers speaking English is the de facto standard. In Europe, due to the lack of affordable speech synthesizers for many languages as well as the notational imprecision of speech, educational and rehabilitation efforts are focusing on braille as a standard notation. Likewise, the two approaches differ on the inclusion of pointing and direct manipulation in the nonvisual interface.

In this paper, we summarize the interface design issues addressed by these systems while noting the overall goals which influenced our design decisions as well as the tools and techniques used to implement our designs. Next, we step through the major design issues and compare our two approaches. Where possible, we also include data gathered from formal and informal user evaluations. We conclude by summarizing the status of our individual efforts as well as speculating on possible advantages to be gained by combining our two designs.

DESIGN ISSUES FOR GUI ACCESS

Creating a nonvisual interface which allows a blind user to interact with a graphical application raises a number of challenging HCI design issues [3][14]. These concerns are briefly introduced below. The remainder of this paper compares how Mercator and GUIB have handled these issues in their respective designs.

- Coherence between visual and nonvisual interfaces

  An overriding concern in access systems is maintaining coherent, parallel visual and nonvisual interfaces. The primary reason for this goal is supporting collaboration between sighted and nonsighted co-workers. For example, a naive access system could simply translate a graphical interface into a command line interface. But the nonsighted user's mental model of the interface would be substantially different from the mental model of a sighted user. Therefore colleagues would not be able to communicate how to use an application to accomplish a joint goal.

  Not only must the visual and nonvisual interfaces support the same mental model of the application interface, their respective presentations must be synchronized to support joint operation. For example, a sighted user should be able to "watch" a blind user work with a nonvisual interface and be able to interact with the visual interface as well. Likewise a blind user should be able to listen to a sighted user working with the graphical interface.

- Exploration in a nonvisual interface

  A significant advantage of graphical user interfaces is the constant presentation of the interface with sufficient resolution to display a great deal of information. No nonvisual media (audio, tactile displays) are able to convey as much information due to limited resolution possibly combined with a serial, dynamic presentation. Therefore additional functionality is required to support efficient exploration of the nonvisual interface.

- Conveying graphical information in a nonvisual interface

  The use of iconic information in graphical user interfaces is both a boon to sighted users and a barrier to blind users. The first step in accessing these symbols is to develop methods to convey interface objects which are partially identified by their appearance such as buttons and menus. Next, attributes of these objects, such as selection state (which would be conveyed graphically through highlighting, for example), must be presented in the nonvisual interface. More difficult to translate are abstract concepts such as sliders and scrollbars which depend on direct manipulation. A further challenge remains in translating pure graphics, animations and multimedia interfaces.

- Interaction in a nonvisual interface

  The only common denominators between interacting with graphical and nonvisual interfaces is the use of the keyboard and generic audio output. All other forms of interaction with graphical interfaces (visual changes, pointing, dragging and clicking) need to be replaced or modified for the nonvisual interface.

Both the GUIB and Mercator projects have addressed these issues, at times resulting in substantially different designs. Two major reasons for these contrasting designs are the prioritization of different interface design goals as well as the selection of tools and established techniques to implement the designs.

Many important interface design goals have influenced both approaches to GUI access. An important criterion for any access system is the scope of applications made accessible. Providing access to applications at a per-application basis would be a frustrating and somewhat useless approach as new applications appear everyday. Therefore systems must devise methods to provide access to entire sets of applications such as all X Windows or Macintosh applications. The generality and implicit restrictions of these methods often limits the information available to the nonvisual interface [4]. Cost is also an important practical consideration. Special purpose devices expand the scope of interfaces which can be explored, but are often too costly for most users.

The learnability of graphical user interfaces must also be addressed from the perspective of the blind user. Sighted users and designers of application software have had more than a decade time to learn about graphical user interfaces, while blind users were excluded from this development. The introduction of nonvisual user interfaces for GUIs can be successful only if an upgrade path from existing technology and metaphors is provided through the user interface.

A number of tools and techniques are available for the creation of nonvisual interfaces. For output, speech synthesizers offer various options for controlling the presentation of speech. Both braille devices for active and passive reading are often used. Large braille displays are popular since the user reads the braille by actively moving their finger against the braille pins as opposed to passively feeling pins change under their finger. These displays are often equipped with braille keys to let the user point at individual characters. Unfortunately, large braille devices are often expensive. Conversely, the widespread inclusion of audio hardware in most computers now facilitates the use of nonspeech auditory cues. Additionally, current research in the generation of spatial audio provides a new medium for the creation of nonvisual interfaces[2].

COHERENT VISUAL AND NONVISUAL INTERFACES

Supporting collaboration and interaction between sighted and nonsighted users motivates the need for coherence between visual and nonvisual interfaces. First, the users' mental models of the visual and nonvisual interface must be reasonably similar to support discussions about an application interface. Second, the visual and nonvisual interfaces must be kept "in sync" so that simultaneous interaction is possible.

Creating A Textual Display

GUIB and Mercator use different interface metaphors as the basis for the nonvisual presentation. GUIB continues the use of the spatial metaphor as presented in the graphical interface. In this design, the organization of the interface is based on the spatial location of the objects on the tactile pad and braille display. This design supports the notion of translating the graphical interface into a textual display which can be translated into braille.

Coherence between visual and nonvisual presentation is ensured by filtering the data generated by the user or by the user interface. This data is made up of user interface events such as keyboard input or refreshing the screen. Filtering takes place on both the lexical level and the syntactical level of the graphical user interface. For text retrieval, at the lexical level a *virtual screen copy* describes, for every pixel on the screen, the character being displayed to the sighted user. On the syntactical level, an *off-screen model* is implemented through a tree of interaction objects which includes the virtual screen copy whenever the syntactical level provides insufficient data. The screen reader is written in an event-response language, called GUIB-ERL [7]. Rules transform the hierarchical off-screen model into textual output and, where appropriate, into acoustic media.

Creating A Hierarchical Model

The rationale behind the Mercator design is based on the premise that they are many features of graphical interfaces which do not need to be modeled in an auditory interface. Many of these features are artifacts of the relatively small, two-dimensional display surfaces typically available to GUIs and do not add richness to the interaction in the auditory domain. For example occluded windows and other space saving techniques can be considered an artifact of the small display and not an inherent part of the application interface.

Mercator ensures compatibility between the visual and nonvisual interfaces by translating the interface at the level of interface components. For example, if the visual interface presents menus, dialog boxes and push buttons then the corresponding auditory interface will also present menus, dialog boxes and push buttons. Only the presentation of these objects will vary.

By performing the translation at the level of interface objects, rather than at a pixel-by-pixel level, the auditory interface can be unencumbered by limitations of modeling the graphical interface exactly. Likewise, the user of the auditory interface is not confused by the inclusion of information which is simply an artifact of visual presentation.

Synchronizing The Visual and Nonvisual Interfaces

Both systems ensure that the visual and nonvisual interfaces are synchronized for parallel interaction. Synchronization is also achieved in an object oriented manner. For example, selecting an object in the nonvisual interface results in warping either the mouse or cursor to that object in the visual interface.

The GUIB interface also addresses synchronization of the braille display. Since braille is a fixed-width font, it cannot present the original layout of characters. As a consequence, synchronization of mouse movements is required. For example, if the mouse cursor is moved on the braille display for a selection within a menu, then the visual presentation centers the mouse cursor within the selection, and the nonvisual presentation centers it in turn. This feature also helps when a frame is to be selected. A frame of a window is one character wide in braille, and three pixels on the screen. Knowing the frame size of the window, the screen reader corrects positioning of the mouse cursor to ensure the window's frame is hit on the visual presentation.

Feedback for mouse movements initiated by a sighted colleague using the normal mouse is recognized by the blind user as the braille display tracks the mouse movements and displays it as caret. Cooperation is thus ensured in both directions, independent of which interface generated the mouse input.

EXPLORING THE NONVISUAL INTERFACE

Various techniques can be used to compensate for the lack of overview information. In the GUIB project, a tactile display is used to provide an overview of large screen objects such as windows. In the Mercator project, the interface structure is mapped onto a hierarchical tree-structure which replaces the spatial organization with a logical organization.

Spatial Exploration

The presentation of GUIB's off-screen model is guided by a spatial metaphor since the emphasis for output is on a large braille display equipped with acoustic facilities for sound and speech generation and a 25 line by 80 braille characters canvas. This canvas is sufficiently large enough to display a 640 by 480 pixel screen. Pilot users have preferred a mode switching feature: either the complete screen or the current application is shown. In the later case, space requirements are reduced. The display has two keys to explore lines sequentially upwards or downwards. Additionally, a vertical tactile display shows four braille pins in a row for each line. These pins also correspond to braille keys. Detecting a different line is therefore easier for standard situations as the location of window frames and icons is indicated through two of the braille pins. A click on the braille key next to the vertical indicator activates the presentation of the corresponding line and thus accelerates exploration of the screen.

The spatial metaphor combined with pointing allows efficient use of dialog boxes. While keyboard-based interaction could require each element to be visited in a step by step fashion, a typical dialogue box can be presented on 3 to five lines of braille. These lines are scanned by the user sequentially, and as soon as the desired element is found, it can be selected (pointing) by clicking a braille key with the reading finger. A second click on the same braille key simulates a mouse click and this completes this frequent action immediately. Pilot users have requested a one-handed operation for all mouse operations (point, click, double-click, drag) which is now tested [16]. A double click is a sequence of one click and a double click on the braille keys. For one-hand operation, dragging is initiated by a double click generated at the spot of the mouse (button down). This mode is reported by acoustic and braille cues. Practice shows that users then point at the destination of the dragging operation. A second double click simulates the mouse movement and the mouse button release. Thereby, common operations like resizing a window or drag&drop can be performed effectively.

Tree-Based Exploration

Since Mercator interfaces are derived from an object-based representation of the user interface, the mechanisms for exploring the user interface are also based on the same object model. The goal of this approach is to ensure that each movement positions the user at a new interface object. With this approach, there is no dead space, and no sense of falling between objects in the user interface. Essentially, the graphical user interface is mapped onto a tree structure which breaks the user interface down into smaller and smaller auditory objects. This tree structure is derived from the conceptual model of the application interface which is partially determined by the X widget hierarchy. The tree structure represents hierarchical relationships (this object is a child of, or contained by, a parent object) as well as dynamic relationships (selecting this object moves the user to this object).

To explore the user interface, the user simply traverses the interface tree structure. When entering an application interface, a breadth first search exposes the main interface objects. Conversely, depth-first searches expose more levels of detail for a portion of the interface. It is worth noting that existing keyboard shortcuts work within this structure. Likewise this scheme is easily extended to accessing multiple applications.

CONVEYING SYMBOLIC INFORMATION

A significant benefit of graphical interfaces is the use of icons to convey symbolic information. A basic problem in providing access to graphical applications is providing intuitive translations for these graphical icons. Mercator incorporates a combination of techniques used to convey information via nonspeech auditory cues [9]. GUIB also uses nonspeech auditory cues, but relies heavily on translating symbolic information directly into braille. These approaches are partially summarized in Table 1: Nonvisual Presentation of Interaction Objects

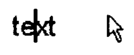
Using Nonspeech Auditory Cues

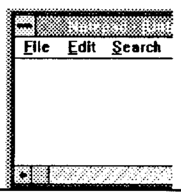
Mercator uses three levels of nonspeech auditory cues to convey symbolic information. The first level addresses the question of "What is this object?" In Mercator, the type of an interface object is conveyed with an *auditory icon*. Auditory icons are sounds designed to trigger associations with everyday objects [5]. This mapping is easy for interface components such as trashcan icons, but is less straightforward for components such as menus and dialog boxes which are abstract notions and have no innate sound associated with them. As examples of some of the auditory icons used in Mercator: touching a window sounds like tapping on a piece of glass, container objects sound like a wooden box opening with a creaky hinge, a variety of push button sounds are used for radio buttons, push buttons and toggle buttons, and editable text fields sound like an old fashioned typewriter.

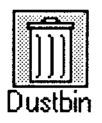
Auditory icons are not limited to simply denoting categories of events and objects, but can be parameterized to reflect their relevant dimensions as well [6]. In Mercator, auditory icons are parameterized to convey icon specific attributes such as the length of a menu. Often these attributes are not explicitly presented in the graphical interface, but are simply part of the graphical presentation.

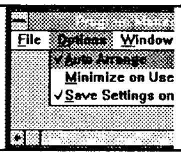
Another interesting question is how to convey attributes which are common across different types of interface objects. For example, the concept of highlighting and greying-out interface objects is common across push buttons, generic icons, and windows as well. Ludwig, Pincever and Cohen [8] suggest that various sound effects or *filtears* can be used to systematically manipulate an auditory cue without losing the identifiability of the original auditory cue. For example, an animation filtear produces a more lively sound by accenting frequency variations while a muffle filtear produces a duller sound by using a linear, low-pass filter. The animation filtear can be used to auditorially highlight auditory icons while the muffle filtear can be to grey-out auditory icons.

A user study was conducted to evaluate the use of nonspeech auditory cues in Mercator interfaces [11]. Both sighted and blind subjects participated in a three part series of tests which evaluated the identifiability, conceptual mapping and usability of a set of auditory icons and filtears. Although the auditory icons were readily learned, the initial use and overall intuitive nature of the interface suffered from the subjects' frustration with identifying the auditory cues.

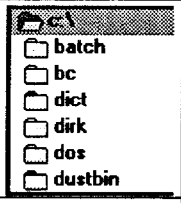
interaction object	**example**	**braille-based presentation**	**example**	**speech-based presentation**	**example**	**nonspeech-based presentation**	**example**
caret, mouse pointer		one braille character	`t■xt    ■`	text around caret is spoken	"e"	audio "click" at caret	t e click x t
text, text attributes	sample	braille, attributes through dots 7 and 8 or on request	`sample`	text is spoken, attributes are verbalized (??)	"sample"	pitch of speech is modified for attributes	

window		window frame in braille
		name is spoken
		pop-up, move, size by sound
			`+[-] Notepad-(Unt [↓][↑]`
			`|F|File E|Edit S|Search`
			`|`
			`+-----------------------`
				name is spoken	"notepad"	auditory icon for window object, modified for popup, iconify, or focus	tapping on glass sound

icon		name in braille	`[Icon]`
			`Dustbin`
					"dustbin"	auditory icons	sound of dropping something in a trashcan

menu		all items in braille vertical or horizontal layout	`F|File O|Options W|Wi`
			`|*A■Auto Arrang`
			`| M|Minimize on`
			`|*S|Save Settin`
			`+-------------`
				new selection is spoken	"auto arrange"	auditory icon for menu-button, pitch is modified relative to location in menu	flipping sound at a high pitch

scroll bar		in braille	`■-□-------------■`	status is verbalized	"slider at zero percent"	auditory icon for scollbar, location conveyed with pitch	slide whistle sound, low pitch
edit field	winwod.exe	in braille, selection with dots 7 and 8	`|winword.exe    |`	text is spoken	"winword dot exe"	auditary icon for editable text field	sound of an old-fashioned typewriter

list box	 c:\ batch bc dict dirk dos dustbin	all items in braille	`+--------------+`
			`| c:\`
			`|  batch`
			`|  bc`
			`|  dict`
			`|  dirk`
			`|  dos`
			`|  dustbin`
			`+--------------+`
				selection is spoken	"c colon backslash"	auditory icon for list, pitch is modified relative to location in list	line-printer sound at a high pitch

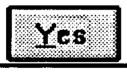
button	 Yes	in braille	`[Y|Yes]`		"yes"	auditory icon for push button	sound of pushing an old elevator button

Table 1: Nonvisual Presentation of Interaction Objects

Using Braille Notation

In GUIB, symbolic information is integrated into the braille code if possible. Each interaction object, in the off-screen model, is characterized by special characters, e.g. brackets around push button names. The shape of the braille characters resembles the graphical appearance wherever possible. Underlining in braille notation (which is more a form of strike-through) supports the use of markings. Thereby text attributes (font, color changes) can be conveyed to the user. Nevertheless some icons are not suitable for braille-based presentation. For example, sounds and speech are used for rapidly changing icons. The shape of the mouse cursor is recognized in the filtering process and when it changes, it is verbalized as well as sonified (an hour glass becomes "please wait" accompanied by several clock ticks). In this sense the desktop (which is empty space in braille) is announced through repetitive sounds as for example a "jungle" sound [15].

INTERACTION IN A NONVISUAL INTERFACE

Many of the interaction mechanisms in graphical interfaces do not translate directly to nonvisual interfaces. Especially problematic is the use of the mouse. Access researchers disagree on whether mouse input should be part of a nonvisual interface. In Mercator, mouse input is completely replaced by keyboard interaction. Although a sighted user can use the mouse while working with a blind colleague, it is not expected that the blind user will use the mouse as well. Conversely, one motivation for the GUIDE display was supporting direct manipulation tasks in the nonvisual interface [12].

Another important issue is deciding how, and when, to signal the user when changes on the screen occur. Mercator provides rules which govern how different objects behave. These rules specify when changes are presented to the user via speech and / or nonspeech auditory cues. GUIB provides additional tactile cursors which mark portions of the screen which have changed.

Eliminating Mouse Input

In Mercator, the need for mouse input has been eliminated. As discussed previously, exploration of the interface can be accomplished by walking a tree structure representation of the interface model. These actions are performed through keyboard input. Likewise, keyboard substitutes for mouse button input are provided. One advantage of this feature is that the selection mechanism can be the same for all interface objects independent of what type of mouse input (single click, double click) is expected.

For GUIDE, two substitutes for the mouse have been developed [16]. Users can point at every braille character or touch a pressure-sensitive touch tablet. The touch sensitive tablet allows the blind users to explore typical direct manipulation tasks such as pointing and dragging.

Signaling The User

An important question is how to notify the user of changes on the screen such as the appearance of a dialog box or the change of button label. Many types of interaction can cause rapid, and possibly, large changes on the screen. Such changes are small in case of text editing as only the caret moves, but switching between applications can cause a large portion of the screen to change.

Mercator provides rule templates for each type of interface object in X Windows applications. These templates guide the behavior of the objects in the nonvisual interface, such as the manipulation of the nonspeech auditory cues associated with that type of object. The rules also specify if, and how, changes to the object should be presented to the user. For example, a message window in an application interface will have at least three modes of operation:

- Always present new information via an auditory cue and synthesized speech
- Signal new information via an auditory cue
- Do not signal the presentation of new information

These modes of operation can be combined in various ways depending on whether the application is the current focus. Cues from applications which are not the current focus are preceded by a cue (speech or nonspeech) which identifies the sending application. The presentation rules for a particular object, or set of objects, can also be modified by the user on a per-application or per-user basis. This flexibility gives the user a large degree of control over the nonvisual interface.

In GUIB, additional cursors mark changes caused asynchronously by the application or by the user's input. A focus indicator marks the active element of a dialogue box or the active document of a multi-window editor. A menu cursor marks the selected option. The GUIDE display then moves the cursor to the object that changed last. Thereby, if the depicted interaction object is small, the user is always informed about any changes on one line of braille.

In case of vertical or horizontal scroll operations, and in general for window operations, only one line can be presented. Therefore, for window operations, the window title is announced verbally through speech output in addition to the movement of the braille line to the object having the focus.

CONCLUSIONS

The two projects discussed here have been driven by their initial assumptions and requirements: Mercator as a low-cost, audio-oriented system; GUIB as a tactile-oriented system with added specialized I/O devices. As a result, the different techniques used in designing and implementing these systems have resulted in dramatically different nonvisual interfaces.

No one interface medium or tool can satisfy all potential users. Both of these systems provide much-needed experience in the area of translating a graphical interface into a nonvisual medium. We believe that it is possible to learn from both of these systems, and to select interface characteristics from each that can be used effectively in future nonvisual interfaces. In the future, it is likely that GUIB will use a greater number of nonspeech auditory cues.

Likewise, Mercator will be adding a braille output component as well.

At a somewhat more abstract level, both Mercator and GUIB are exploring the space of methodologies for translation of graphical user interfaces. While both of these projects are targeted at translating GUIs into nonvisual interfaces to provide access to visually-impaired users, access is merely one application of "retargetting," or translating interface to a new medium.

For example, a graphical application's interface could be retargetted to a completely new medium, such as a touch-tone telephone. This technique would provide access to graphical applications for otherwise "normal" users who happen to be in disabling circumstances (such as not being present at their graphical monitors). We believe that exploration of nonvisual media, along with investigation of different translation techniques, will be crucial to the development of next-generation toolkits and UIMS's that provide true, native interface retargetting.

ACKNOWLEDGEMENTS
The GUIB Project is supported by the Commission of the European Communities DG XIII under the TIDE Pilot Action (Technology Initiative for the Disabled and Elderly Persons).

The Mercator project is a joint effort by the Georgia Tech Multimedia Computing Group (a part of the Graphics, Visualization, and Usability Center) and the Center for Rehabilitation Technology. This work has been sponsored by the NASA Marshall Space Flight Center (Research Grant NAG8-194) and Sun Microsystems Laboratories.

REFERENCES

1. L.H. Boyd, W.L. Boyd, and G.C. Vanderheiden. The graphical user interface: Crisis, danger and opportunity. *Journal of Visual Impairment and Blindness*, pages 496-502, December 1990.
2. David Burgess. Low Cost Sound Spatialization. In *UIST '92: The Fifth Annual Symposium on User Interface Software and Technology Conference Proceedings*, November 1992.
3. Alistair D. N. Edwards. Modeling blind users' interactions with an auditory computer interface. *International Journal of Man-Machine Studies*, pages 575-589, 1989.
4. W. Keith Edwards, Elizabeth D. Mynatt and Tom Rodriguez. The Mercator Project: A Nonvisual Interface to the X Window System. *The X Resource*. O'Reilly & Associates, Inc. April 1993.
5. W. W. Gaver. The SonicFinder: An interface that uses auditory icons. *Human Computer Interaction*, 4:67-94, 1989.
6. W. W. Gaver. Synthesizing Auditory Icons. *Proceedings of the 1992 International Conference on Auditory Display.* Addison-Wesley Publishing Company.
7. Hill, R. D. Supporting concurrency, communication and synchronization in Human - Computer Interaction - the Sassafras UIMS, *ACM Transactions on Graphics*, Vol. 5, No. 3 (1986) 179 - 210.
8. Lester F. Ludwig, Natalio Pincever, and Michael Cohen. Extending the notion of a window system to audio. *Computer*, pages 66-72, August 1990.
9. Elizabeth Mynatt and Keith Edwards. New metaphors for nonvisual interfaces. In *Extraordinary Human-Computer Interaction*, 1991. Draft chapter accepted for upcoming book.
10. Elizabeth Mynatt and W. Keith Edwards. Mapping GUIs to Auditory Interfaces. In *UIST '92: The Fifth Annual Symposium on User Interface Software and Technology Conference Proceedings*, November 1992.
11. Elizabeth D. Mynatt. Auditory Presentation of Graphical User Interfaces. Auditory Presentation of Graphical User Interfaces. *Proceedings of the 1992 International Conference on Auditory Display.* Addison-Wesley Publishing Company.
12. Petrie, H; Heinila, J; Ekola, H. (1993) A comparative evaluation of computer input devices for blind user, in *Proceedings of ECART 2*, Stockholm, May 26-28, 1993, pp. P-II
13. Schwerdtfeger, Richard S.: Making the GUI talk, *BYTE*, Dec 1991, 118-128.
14. Vanderheiden, G. C.; Boyd, W.; Mendenhall, J.H.; Ford, K.: Development of a multisensory nonvisual interface to computers for blind users, in *Proceedings of the Human Factors Society 35th Annual Meeting* 1991, pp. 315 - 318
15. Weber, G.: Adapting graphical interaction objects for blind users by integrating braille and speech, in Zagler, W. (Ed.) *Computers for Handicapped Persons*, Oldenburg: Wien, 1992
16. Weber, G.(1993) Adapting direct manipulation for blind users, in Ashlund, Stacey et.al (eds.): *Adjunct Proceedings of INTERCHI '93*, Addison Wesley, pp. 21-22
17. Weber, G. (1993) Access by blind people to interaction objects in MS Windows, in *Proceedings of ECART 2*, Stockholm, May 26-28, 1993, pp. 2.2
18. Bryant W. York, editor. *Final Report of the Boston University Workshop on Computers and Persons with Disabilities*, 1989.