Auditory Presentation of Graphical User Interfaces

Elizabeth D. Mynatt
Graphics, Visualization & Usability Center
College of Computing
Georgia Institute of Technology
beth@cc.gatech.edu

ABSTRACT

This paper describes work to provide mappings between X-based graphical interfaces and auditory interfaces. In our system, dubbed Mercator, this mapping is transparent to applications. The primary motivation for this work is to provide accessibility to graphical applications for users who are blind or visually impaired. In this paper, I describe the design of an auditory interface which simulates many of the features of graphical interfaces. Some of these features have been evaluated in a small user study. I will present lessons learned from this study as well as discuss areas for further work.

KEYWORDS: Auditory interfaces, GUIs, X, visual impairment, multimodal interfaces.

INTRODUCTION

The goal of human-computer interfaces is to provide a communication pathway between computer software and human users. The history of human-computer interfaces can be interpreted as the struggle to provide more meaningful and efficient communication between computers and humans. One important breakthrough in HCI was the development of graphical user interfaces. These interfaces provide graphical representations for system objects such as disks and files, interface objects such as buttons and scrollbars, and computing concepts such as multi-tasking. Unfortunately, these graphical user interfaces, or GUIs, have disenfranchised a percentage of the computing population. Presently, graphical user interfaces are all but completely inaccessible for computer users who are blind or severely visually-disabled [BBV90][Bux86][Yor89]. This critical problem has been recognized and addressed in recent legislation (Title 508 of the Rehabilitation Act of 1986, 1990 Americans with Disabilities Act) which mandates that computer suppliers ensure the accessibility of their systems and that employers must provide accessible equipment [Lad88].

Our work on this project began with a simple question, how could we provide access to X Windows applications for blind computer users. Historically, blind computer users had little trouble accessing standard ASCII terminals. The line-oriented textual output displayed on the screen was stored in the computer's framebuffer. An access program could simply copy the contents of the framebuffer to a speech synthesizer, a Braille terminal or a Braille printer. Conversely, the contents of the framebuffer for a graphical interface are simple pixel values. To provide access to GUIs, it is necessary to intercept application output before it reaches the screen. This intercepted application output becomes the basis for an off-screen model of the application interface.The information in the off-screen model is then used to create alternative, accessible interfaces.

The goal of this work, called the Mercator[1] Project, is to provide *transparent* access to X Windows applications for computer users who are blind or severely visually-impaired [ME92]. In order to achieve this goal, we needed to solve two major problems. First, in order to provide transparent access to applications, we needed to build a framework which would allow us to monitor, model and translate graphical interfaces of X Windows applications without modifying the applications. Second, given these application models, we needed to develop a methodology for translating graphical interfaces into nonvisual interfaces. This methodology is essentially the implementation of a *hear-and-feel* standard for Mercator interfaces. Like a look-and-feel standard for graphical interfaces, a hear-and feel standard provides a systematic presentation of nonvisual interfaces across applications.

In this paper, I describe the design of Mercator interfaces which are based on the Mercator off-screen model of application GUI interfaces. I introduce the concepts of audio GUIs and the abstract components of auditory interfaces. I describe the use of auditory icons and filtears to convey a range of information about interface components. I will also present some lessons learned from a user study of Mercator interfaces. In conclusion I will present some ideas for further work in nonvisual interfaces.

1. Named for Gerhardus Mercator, a cartographer who devised a way of projecting the spherical Earth's surface onto a flat surface with straight-line bearings. The Mercator Projection is a mapping between a three-dimensional presentation and a two-dimensional presentation of the same information. The Mercator Environment provides a mapping from a two-dimensional graphical display to a three-dimensional auditory display of the same user interface.

MERCATOR OFF-SCREEN MODELS

The first goal of the Mercator Project is to build a framework which can monitor, model and translate application GUI interfaces transparently to the application. The details of this framework can be found in [ME92a]. The resulting off-screen models are based on the X widgets which are used to implement the X interface. The application widget hierarchy is stored along with each widget's attributes, resources, and properties.

Essentially this off-screen model provides information about the types of interface objects used in the application interface as well as attributes of these objects. The widget hierarchy provides some information about the relationships between these objects but this information is incomplete. For example, a menu button may be the child of the box containing the menu buttons but the actual menu associated with the menu button may be located in an unrelated portion of the widget hierarchy. On the whole, the widget specification of the application interface is too low-level for a straight-forward translation from graphical objects to Mercator objects. Many widgets used in an X application interface are essentially invisible during the visual presentation of the interface and should likewise be invisible in Mercator interfaces as well. The following sections will describe how the information in the off-screen models are translated into Mercator interfaces.

AUDIO GUIs

The primary design question to be addressed in this work is, given a model for a graphical application interface, what corresponding interface do we present for blind computer users. In this portion of the paper, I will discuss two major design considerations for these interfaces. I then describe the presentation of common interface objects such as buttons, windows, and menus and detail the navigation paradigm for Mercator interfaces. Next I will describe a small user study which evaluated one implementation of Mercator interfaces and provide a summary of the lessons learned from that study.

Design Considerations

There are several design decisions we had to make when constructing our nonvisual interface. One consideration is which nonvisual interface modality to use. The obvious choices are auditory and tactile. We are currently basing our design on previous work in auditory interfaces which has demonstrated that complex auditory interfaces are usable [BGB91]. Another factor that we considered is that a significant portion of people who are blind also suffer from diabetes which may cause a reduction in their sensitivity to tactile stimuli [HTAP90]. Nevertheless our system will eventually have tactile components as well. For example, a braille terminal provides an alternate means for conveying textual information which may be preferred to speech synthesis. Other possible uses of tactile output are described later in this paper.

A second major design question for building access systems for visually-impaired users is deciding the degree to which the new system will mimic the existing visual interface. At one extreme the system can model every aspect of the visual interface. For example, in Berkeley System's Outspoken,$^{\mathbf{tm}}$ which provides access to the Macintosh, visually-impaired users use a mouse to search the Macintosh screen [Van89]. When the mouse moves over an interface object, Outspoken reads the label for the object. In these systems, visually-impaired users must contend with several characteristics of graphical systems which may be undesirable in an auditory presentation, such as mouse navigation and occluded windows.

At the other extreme, access systems can provide a completely different interface which bears little to no resemblance to the existing visual interface. For example, a menu-based graphical interface could be transformed into an auditory command line interface.

Both approaches have advantages and disadvantages. The goal of the first approach is to ensure compatibility between different interfaces for the same application. This compatibility is necessary to support collaboration between sighted and non-sighted users. Yet if these systems are too visually-based they often fail to model the inherent advantages of graphical user interfaces such as the ability to work with multiple sources of information simultaneously. The second approach attempts to produce auditory interfaces which are best suited to their medium. [Edw89]

We believe that there 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 an auditory domain. If we consider the GUI screen to be in many regards a limitation, rather than something to be modeled exactly, then we are free to use the characteristics of auditory presentation which make it more desirable in some ways than graphical presentation.

We have chosen a compromise between the two approaches outlined above. To ensure compatibility between visual and nonvisual interfaces, we are translating the interface at the level of the 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 the interface objects will vary.

By performing the translation at the level of interface objects, rather than at a pixel-by-pixel level (like Outspoken) we can escape from some of the limitations of modeling the graphical interface exactly. We only model the structural features of the application interface, rather than its pixel representation on screen.

Interface Components

Graphical user interfaces are made up of a variety of interface components such as windows, buttons, menus and so on. In X Windows applications, these components roughly

TABLE 1. Using Filtears to convey AIC attributes.

Attribute	AIC	Filtear	Description
selected	all buttons	animation	Produces a more lively sound by accenting frequency variations
unavailable	all buttons	muffled	A low pass filter produces a duller sound
has sub-menu	menu buttons	inflection	Adding an upward inflection at the end of an auditory icon suggests more information
relative location	lists, menus	pitch	Map frequency (pitch) to relative location (high to low)
complexity	containers	pitch, reverberation	Map frequency and reverberation to complexity. Low to large, complex AICs and high to small, simple AICs

correspond to widgets. There does not always exist a one-to-one mapping between graphical interface components and X widgets. For example, a menu is made up of many widgets including lists, shells (a type of container), and several types of buttons.

Mercator provides auditory interface objects which mimic some of the attributes of graphical interface objects. In Mercator, we call the objects in our auditory presentation Auditory Interface Components, or AICs. The translation from graphical interface components to AICs occurs at the widget level. As with graphical interface components, there is not always a one-to-one mapping between X widgets and AICs. AICs may also be composed of many widgets. Additionally, many visual aspects of widgets need not be modeled in AICs. For example, many widgets serve only to control screen layout of sub-widgets. In an environment where there is no screen, there is no reason to model a widget which performs screen layout. For many widgets there *will* be a one-to-one mapping to AICs. As an example, push buttons (interface objects which perform some single function when activated) exist in both interface domains. In other cases, many widgets may map to a single AIC. For example, a text window with scrollbars may map to one text display AIC. Scrollbars exist largely because of the need to display a large amount of text in a limited area. A text display AIC may have its own interaction techniques for scanning text.

There are two types of information to convey for each AIC: the type of the AIC and the various attributes associated with the AIC. In our system, the type of the AIC is conveyed with an auditory icon. Auditory icons are sounds which are designed to trigger associations with everyday objects, just as graphical icons resemble everyday objects [Gav89]. This mapping is easy for interface components such as trashcan icons but is less straight-forward for components such as menus and dialog boxes, which are abstract notions and have no innate sound associated with them. As an example of some of our auditory icons, touching a window sounds like tapping on a glass pane, searching through a menu creates a series of shutter sounds, a variety of push button sounds are used for radio buttons, toggle buttons, and generic push button AICs, and a touching a text field sounds like a old fashioned typewriter.

AICs can have many defining attributes. Most AICs have text labels which can be read by a speech synthesizer upon request. Many attributes can be conveyed by employing so-called *filtears* to the auditory icon for that AIC. Filtears provide a just-noticeable, systematic manipulation of an auditory signal to convey information[LPC90][LC91]. Table 1 details how filtears are used to convey some AIC attributes.

Navigation

The navigation paradigm for Mercator interfaces must support two main activities. First, it must allow the user to quickly "scan" the interface in the same way as sighted users visually scan a graphical interface. Second, it must allow the user to operate on the interface objects, push buttons, enter text and so on.

In order to support both of these activities, the user must be able to quickly move through the interface in a structured manner. Standard mouse navigation is unsuitable since the granularity of the movement is in terms of graphic pixels. Auditory navigation should have a much larger granularity where each movement positions the user at a different auditory interface object. To support navigation from one AIC to another, we map the user interface into a tree structure which breaks the user interface down into smaller and smaller AICs. This tree structure is related to application's widget hierarchy but there is not a one-to-one mapping between the widget hierarchy and the interface tree structure. As discussed earlier, there is sometimes a many-to-one mapping between widgets and AICs. Additionally, an AIC may conceptually be a child of another AIC but the widgets corresponding to these AICs may be unrelated. For example, a push button may cause a dialog box to appear. These AICs are related (the dialog box is a child of the push button) but the widget structure does not reflect the same relationship. Figure 1 shows a screen-shot of the graphical interface for xmh, an X-based mail application. Figures 2a and 2b show a portion of the xmh widget hierarchy and the corresponding interface tree structure, respectively.

To navigate the user interface, the user simply traverses the interface tree structure. Currently the numeric keypad is used to control navigation. Small jumps in the tree structure are controlled with the arrow keys. Other keys can be mapped to make large jumps in the tree structure. For example, one key

Figure 1: An Example X Application (XMH)

on the numeric keypad moves the user to the top of the tree structure. It is worth noting that existing application keyboard short-cuts should work within this structure as well.

Navigating the interface via these control mechanisms does not cause any actions to occur except making new AICs "visible." To cause a selection action to occur, the user must hit the Enter key while on that object. This separation of control and navigation allows the user to safely scan the interface without activating interface controls. Table 2 summarizes the navigation controls.

TABLE 2. Summary of Navigation Controls

Key	Action
8/up arrow	Move to parent
2/down arrow	Move to first child
6/right arrow	Move to right sibling
4/left arrow	Move to left sibling
0/Ins	Move to top of tree
Enter	Activate selection action
= / * -	Can be mapped to other movements

USER STUDY

I conducted a simple user study to evaluate the one possible presentation of Mercator interfaces. Both sighted and non-sighted participants were used in the study. Sighted users were able to see the corresponding visual presentation of xmh although they only interacted with the interface via Mercator controls The study consisted of four parts:

Auditory Icon Identification Test

Participants were asked to play 7 sounds and select what they thought the sound represented. The possible choices for each sound were:

- Menu Button: A button that you would select to make a menu appear.
- Command Button: A simple push button.

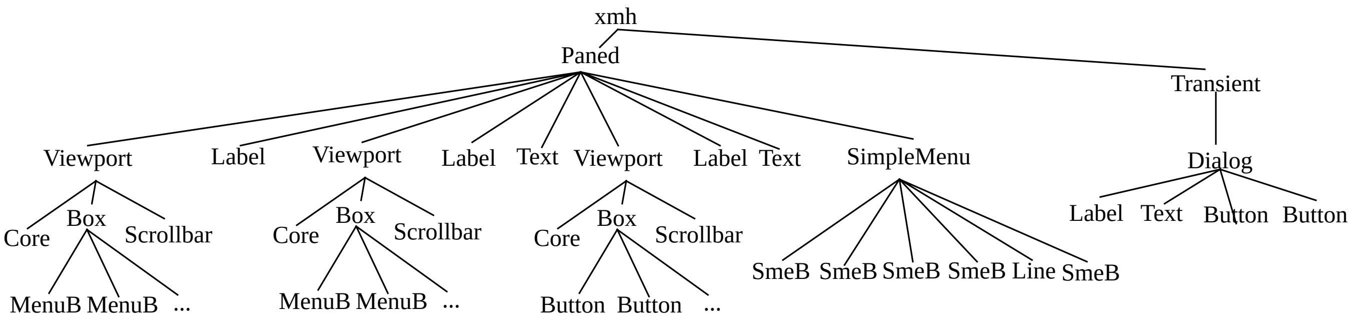

Figure 2a: XMH Widget Hierarchy

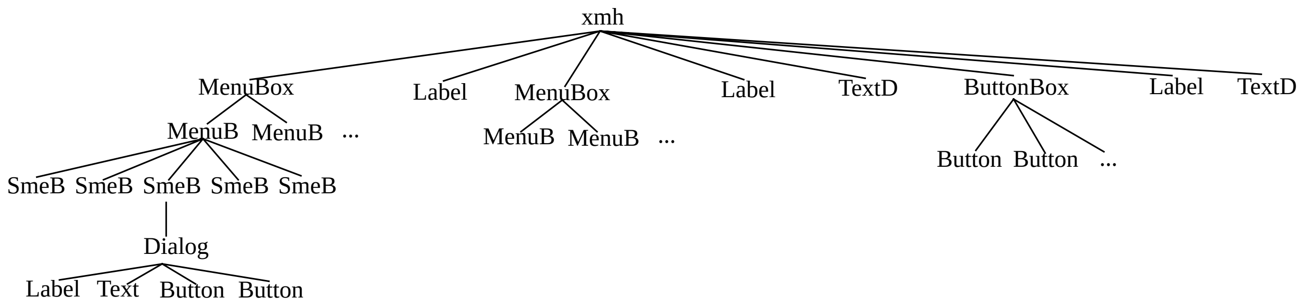

Figure 2b: Corresponding XMH Auditory Interface Component Hierarchy

- Box (container): An object which contains other objects such as a box of push buttons.
- Text Area: An area where you would read and/or enter text.
- Label: A simple label (read-only text).
- Navigation Error: The sound made when a user attempts to navigate out of the bounds of the available area.
- Window Mapped: The sound made when a window appears (pops-up) on the screen.

Participants were then asked to play two related sounds which represented the same interface object and choose what information was conveyed by modifying the first sound to produce the second sound. The second sound was the result of running a low pass filter over the first sound. The possible choices were:

- the object is already selected (highlighted)
- the object is unavailable (greyed out)
- the object has a sub-object (i.e. a cascading menu)

Mercator Navigation Task

Participants were asked to experiment navigating a Mercator interface for the X application xmh, a mail reading tool. Participants heard combinations of speech and non-speech audio to convey information about different interface objects. The non-speech audio cues were the same as those presented in the Auditory Icon Identification Test.

Written Feedback

Participants were asked to record their impressions of the navigation strategy used by Mercator, the auditory icons used by Mercator and any other comments.

Retake Auditory Icon Identification Test

Participants were asked to re-take the same test they had taken earlier.

USER STUDY - LESSONS LEARNED

The number of participants precludes the use of formal statistical analysis of the results. The following section details observations from test results, written comments, watching the participants use Mercator and later discussions with the participants.

Auditory Icons

Although participants fared better when re-taking the auditory icon identification test, the overall test results did not support the goal of designing intuitive auditory icons. Several factors seemed to affect the results. First, most participants judged the 8kHz sound quality as harsh and the low frequency auditory icons too “industrial sounding.” Second, many of the auditory icons were synthesized not sampled. This fact seemed to increase confusion as to what the actual sound was to begin with, independent of what the sound represented in the interface. For example, a synthesized sound of a typewriter was used to represent text widgets. Participants who did not recognize the sound as a typewriter were hard pressed to associate that sound with a text widget. Third, again due to the poor sound quality, some participants relied on previous experience with other poor sound quality devices such as video games to provide clues to identifying the sound.

Navigation

Overall impressions of the navigation strategy were favorable. Participants rarely tried to move outside of the interface structure. Some confusion for sighted participants was caused by contradictions in navigation controls and the visual presentation. For example, moving across (right) the interface tree often corresponding to the pointer moving down the visual presentation. Non-sighted users found the navigation scheme easy to use. Both sets of users requested navigation controls that would provide larger jumps in the interface tree such as moving to the last object in a particular direction as well as the ability to set markers as navigation points. Both of these features were not implemented in the version of Mercator used in this study.

FUTURE DIRECTIONS

In this section I present some ideas for increasing the power of the auditory interfaces provided by Mercator as well as adding tactile presentation to Mercator interfaces.

Spatial organization via spatialized sound

One of the cognitive benefits of graphical desktop environments is the ability to organize objects in a 2D space. We have been investigating uses of spatialized sound to create a 3D auditory environment. We have done some preliminary design work on a system called Audio Rooms which would be a 3D auditory presentation of the Rooms metaphor introduced by researchers at PARC[HC86][ME91]. Due to the costs of commercial spatialized sound systems such as the Convolvotron, we have built a low cost implementation of spatialized sound based on the algorithms implemented in the Convolvotron which runs on a standard DSP 56000[Bur92].

Periphery information via auditory contexts

Graphical user interfaces allow users to not only see the current object of interest but the users generally also see other objects which are related to the current object. This ability allows the users to use peripheral information to understand the object of focus. In contrast, most auditory interfaces play only one sound at a time. A more powerful scheme would be to presents auditory cues for peripheral information as well as auditory cues for the current focus. These secondary cues could be mixed with the primary cues or presented spatially around the primary cue. In either case, the user would be presented with an auditory context which would represent the current area of focus not just the current object of focus.

Possible uses of tactile output

The strategy of translating graphical interfaces into auditory interfaces begins to break down as the visual interfaces become more like pictures and use fewer standard representations for interface objects. The extreme case is a picture such as a free-hand drawing or a scanned-in image. Tactile pads which allow users to feel a picture provide some possibilities. Combinations of auditory and tactile output might allow a user to hear different colors or could be used to inform the user of changes in the tactile presentation.

ACKNOWLEGEMENTS

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.

Also, a number of people have contributed a great deal of time and energy to the development of the Mercator Application Manager; without their hard work this system would not have been possible. I would like to thank Gerry Higgins (NASA), Earl Johnson (Sun) and Keith Edwards, Dave Burgess, Tom Rodriguez, and Ian Smith, and John Goldthwaite (Georgia Tech).

REFERENCES

[BBV90] 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.

[BGB91] Bill Buxton, Bill Gaver, and Sara Bly. The Use of Non-SpeechAudio at the Interface. *Tutorial Presented at CHI'91*. April 1991.

[Bur92] David Burgess. Low Cost Sound Spatilization. To appear in *UIST '92: The Fifth Annual Symposium on User Interface Software and Technology and Technology*, November 1992.

[Bux86] William Buxton. Human interface design and the handicapped user. In *CHI'86 Conference Proceedings*, pages 291–297, 1986.

[HC86] Jr. 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.

[Edw89] Allstair D. N. Edwards. Modeling blind users' interactions with an auditory computer interface. *International Journal of Man-Machine Studies*, pages 575–589, 1989.

[Gav89] William W. Gaver. The sonicfinder: An interface that uses auditory icons. *Human Computer Interaction*, 4:67–94, 1989.

[HTAP90] HumanWare, Artic Technologies, ADHOC, and The Reader Project. Making good decisions on technology: Access solutions for blindness and low vision. In *Closing the Gap Conference*, October 1990. Industry Experts Panel Discussion.

[Lad88] Richard E. Ladner. Public law 99–506, section 508, electronic equipment accessibility for disabled workers. In *CHI'88 Conference Proceedings*, pages 219–222, 1988.

[LC91] Lester F. Ludwig and Michael Cohen. Multidimensional audio window management. *International Journal of Man-Machine Studies*, Volume 34, Number 3, pages 319-336, March 1991.

[LPC90] Lester F. Ludwig, Natalio Pincever, and Michael Cohen. Extending the notion of a window system to audio. *Computer*, pages 66–72, August 1990.

[ME91] Elizabeth Mynatt and Keith Edwards. New metaphors for nonvisual interfaces. In *Extraordinary Human-Computer Interaction*, 1991. Draft chapter accepted for upcoming book.

[ME92] Elizabeth Mynatt and W. Keith Edwards. The Mercator Environment: A Nonvisual Interface to X Windows and Unix Workstations. *GVU Tech Report GIT-GVU-92-05*. February 1992.

[ME92a] Elizabeth Mynatt and W. Keith Edwrds/ Mapping GUIs to Auditory Interfaces. To appear in *UIST '92: The Fifth Annual Symposium on User Interface Software and Technology and Technology*, November 1992.

[Van89] G.C. Vanderheiden. Nonvisual alternative display techniques for output from graphics-based computers. *Journal of Visual Impairment and Blindness*, 1989.

[Yor89] Bryant W. York, editor. Final Report of the Boston University Workshop on Computers and Persons with Disabilities, 1989.