Transforming Graphical Interfaces into Auditory Interfaces for Blind Users

Elizabeth D. Mynatt

Xerox Palo Alto Research Center

ABSTRACT

While graphical interfaces have provided a host of advantages to the majority of computer users, they have created a significant barrier to blind computer users. To meet the needs of these users, a methodology for transforming graphical interfaces into nonvisual interfaces has been developed. In this design, the salient components of graphical interfaces are transformed into auditory interfaces. Based on a hierarchical model of the graphical interface, the auditory interface utilizes auditory icons to convey interface objects. Users navigate the interface by traversing its hierarchical structure. This design results in a usable interface that meets the needs of blind users while providing many of the benefits of graphical interfaces.

1. INTRODUCTION

The problem addressed by this research can be simply stated, "What kind of interface would you design for a blind person using a graphical user interface?" The requirements of blind users demand a *general* mechanism for *transforming* graphical interfaces into nonvisual interfaces. Additionally, blind users would like to enjoy the benefits of graphical user interfaces. The requirements of blind users coupled with the benefits of graphical interfaces form a set of goals for the interface transformations. We will see that commercial software for blind users fails to meet many of these goals. Specifically, the reliance on a spatial model of the graphical interface impacts the usability of the resulting nonvisual interface.

The first step in transforming graphical interfaces into nonvisual interfaces is determining the contents of the transformation. What is being converted from the graphical modality into the nonvisual modality? Graphical interfaces are composed of groups of interface objects that are presented spatially on a two dimensional display. These objects are characterized by a number of attributes that help convey their intended functionality. The objects making up the graphical interface, their attributes and the relationships between the objects comprise the contents of the interface transformation.

Next, it is necessary to model the contents of the transformation so that the model both captures the critical characteristics of the graphical interfaces and provides the basis for an intuitive nonvisual interface. Different models are possible. After comparing spatial, hierarchical and conversational models, we argue for utilizing a hierarchical model because it best captures the underlying structure of the graphical interface without requiring application, domain-specific knowledge.

Given a hierarchical model of the graphical interface, the next step is designing the nonvisual interface. In this design, we have focused on conveying the contents of the user interface, supporting navigation, and providing controls for manipulating the interface. By using auditory cues akin to sound effects heard in real world environments, we attempt to provide the benefits of iconic representations. These auditory icons (see Gaver, 1988) convey the type of interface objects as well as attributes of the objects such as their size, selection state, and spatial location. For example, a muffled, light switch sound conveys a greyed-out toggle button.

Users move from object to object based on the hierarchical model of the interface. Interruptability, navigation shortcuts, and previews help alleviate the potential tedium of traversing a large structure. Auditory feedback also helps users perceive changes in the interface based on their input or application events. For example, rising and falling whistling sounds accompany the appearance and disappearance of pop-up windows.

Assessments of this design are based on over four years of feedback from blind computer users as well as controlled experiments with sighted users. We conclude this paper by evaluating this design against the goals that we outlined for interface transformations. Although lacking in its ability to present information spatially, this design results in a usable interface that provides many of the critical characteristics of graphical interfaces.

2. BACKGROUND

2.1 The GUI Problem

In the paper “The Graphical User Interface: Crisis, Danger and Opportunity,” Boyd et al. (1990), summarized an overwhelming concern expressed by the blind community: a new type of visual interface threatened to erase the progress made by the innovators of screen reader software. Such software (as the name implies) could read the contents of a computer screen, allowing blind computer users equal access to the tools used by their sighted colleagues. Whereas character-based screens were easily accessible, new graphical interfaces presented a host of technological challenges. The contents of the screen were mere pixel values, the on or off “dots” which form the basis of any bit-mapped display.

The goal for screen reader providers was to develop new methods for bringing the meaning of these picture-based interfaces to users who could not see them.

The crisis was imminent. Graphical user interfaces were quickly adopted by the sighted community as a more intuitive interface. Ironically, these interfaces were deemed more accessible by the sighted population because they seemed approachable for novice computer users. The danger was tangible in the forms of lost jobs, barriers to education, and the simple frustration of being left behind by the computer industry.

Much has changed since that article was published. Commercial screen reader interfaces now exist for two of the three main graphical environments. But many blind users still do not view graphical interfaces as a new opportunity. Screen readers designers, faced with the task of translating a complex, visual interface into auditory or tactile output, have attempted to create one-to-one translations of the spatially arranged graphical interfaces. Blind users have responded with difficulties in using these visually-oriented interfaces.

The Mercator Project at Georgia Tech addressed two untouched areas of work in the screen reader community. First, no one had designed a screen reader for X Window applications, such as Motif applications used in research, business and educational settings. Second, there was little work in alternate representations of graphical interfaces that were not based on speech output and spatial organizations.

The implementation of Mercator is described in (Edwards, W.K. 1993, 1994, 1995). Briefly, the system provides the infrastructure to monitor and model unmodified X applications while they are running. A collection of "hooks"[1] in the Xlib and Xt Intrinsics libraries of the X Window System trap interesting events such as the creation of a push button or the appearance of a window. The information gleaned from these hooks is transmitted to

Mercator which creates a model of the graphical interface based on the application's widget hierarchy. Mercator also provides facilities for creating interfaces to replace or augment the graphical interface. Interface behavior can be specified in an interpreted language supporting prototyping and end-user customization.

An underlying assumption in the design of Mercator interfaces is the dominant use of auditory output. Researchers have experienced limited success with tactile devices with the exception of braille output. Additionally, a significant portion of people who are blind also suffer from diabetes which reduces their sensitivity to tactile stimuli (Humanware et al. 1990). Nevertheless Mercator includes a tactile component as well. For example, since speech synthesizers are notoriously bad at reading source code, Mercator provides a Braille terminal as an alternate means for presenting textual information

2.2 Requirements for GUI Access by Blind Users

The requirements for the auditory interface are driven by the need for blind users to work with their sighted colleagues employing the same graphical applications. Without knowledge of the application domain, the screen reader system must transform the contents of the graphical interface into a usable auditory interface. By monitoring the execution of a graphical application, the screen reader creates a model of the application interface and derives a complimentary auditory interface. The user's interaction with the auditory interface is forwarded to the graphical interface and the process continues as shown in the following figure.

1. The "hooks" are now part of the standard X Windows System since X11R6. The protocol used to transmit information trapped by the hooks to an external program is under consideration by the X Consortium.

FIGURE 1 ABOUT HERE

The foremost critical requirement is *transparent transformations* of graphical interfaces. Modifying individual applications does not address problems of providing access to a set of graphical applications. A general mechanism for transforming any X Window application is needed. The ideal scenario is that blind users running the screen reader on their systems should be able to use any X application without needing to specially tailor the application interfaces.

To address this need, Mercator automatically transforms text-based, X Windows applications while they are running, providing an auditory interface. This requirement for transparent transformations impacts the design in two critical ways. First, there is no domain knowledge to inform the creation of the auditory interface. Mercator is unaware of the functionality of the application, such as whether it is a word processor or electronic mail tool, but is only aware of how the graphical interface is constructed. Second, this transformation is done in "real-time". There is no off-line processing or analysis of the graphical interface.

An implicit requirement for screen reader systems is that they *facilitate collaboration between sighted and blind colleagues*. Blind users do not work in isolation from their sighted counterparts. Therefore it is imperative that blind and sighted users be able to communicate about their use of application interfaces.

In addition to reinforcing the need for transparent access, this requirement constrains the design of the auditory interface. While an auditory interface to an application may be quite intuitive and usable, if it does not express interface concepts similar to the graphical interface, it does not solve the collaboration need by the blind computer user. Ideally a blind user should be able to ask a sighted user how to do something with an application interface and be able to utilize directions expressed in terms of the graphical interface. This ideal scenario is difficult to achieve, but the design goals of designing for collaboration versus designing for intuitive auditory interaction conflict in interesting ways.

Within the range of graphical interfaces, this work focuses on the transformation of text-oriented graphical interfaces such as electronic mail programs, word processors and spreadsheets. By choosing this area, we are focusing on interfaces in which text is the primary object of interest and where text is manipulated through the use of graphical controls. In contrast, applications such as drawing programs where graphics are the primary objects of interest are not addressed in this work. This restriction is due to the additional difficulty of representing pure graphical information in the auditory modality. Nevertheless the chosen application set is sufficiently interesting since it represents applications that are commonly used.

Expressed as a general requirement, an additional need by blind computer users is to *experience the benefits of graphical interfaces* enjoyed by their sighted counterparts, such as iconic representation and direct manipulation. Below, we present goals for screen reader interface design based on the benefits of GUIs:

- **Access to functionality**

  At a minimum, the user must be able to use the functions represented by the graphical interface. For example, in a word processor where pull-down menus support operations for loading and saving files, users would need an interface to this functionality. Some software vendors maintain that their graphical applications are accessible to blind users because they provide a separate command-line interface that can be read by older screen readers. Simply providing access to the same functionality likely breaks the goal of supporting collaboration between blind and sighted users since they use a distinctly different interface.

- **Iconic representations of interface objects**

  Graphical icons, from trashcans to push buttons, help the user assess the capabilities of an interface by leveraging knowledge of the physical world. Visual attributes of interface objects such as size and highlighting also convey information to the user.

- **Direct manipulation**

  Closely coupled with the benefit of iconic representation, is the benefit of direct manipulation. According to Hutchins et al. (1986), this benefit is achieved when the user is able to directly interact with objects of interest to the task at hand, and output in the interface is expressed via these objects.

- **Spatial arrangement**

  Graphical interfaces allow the user to organize information in a 2 1/2D space. Contrast organizing a desktop by maintaining lists of objects and categories of lists. Another benefit of spatial arrangement is that it can leverage knowledge of the physical world. Sliders that support viewing portions of a document capitalize on moving sheets of paper sideways and front-to-back in a stack.

- **Constant presentation**

  A benefit of visual interfaces is that they exist in physical space that can be reviewed over time. This advantage of the visual sensory system is capitalized in graphical interfaces. These displays serve as a surrogate short-term memory for recalling the contents of the user interfaces.

We will see that these benefits are ordered from easiest to hardest for a screen reader system to provide. In the following section, we briefly evaluate screen reader systems that allow blind users to interact with representations of graphical interfaces.

2.3 Evaluation of Screen Reader Interfaces

There are two general classes of commercial screen readers that provide auditory interfaces for graphical interfaces. The first class is dominated by a product called OutSpoken (1989). The primary characteristic of this class is that the structure of the auditory interface is based primarily on the spatial layout of the graphical interface. Users navigate the screen using the mouse or keyboard shortcuts. The interface uses synthesized speech almost exclusively. At the basic level, the user moves the mouse cursor across the screen, and when the cursor intersects a graphical object the speech synthesizer reads information about that object. An auditory cue is used to convey moving across a window boundary.

Since OutSpoken relies heavily on optical character recognition (OCR) algorithms that are extended to recognize graphical icons, this interface does not group icons as one might expect. Two examples of OutSpoken's interface reveal its usability limitations.

In a grouping of controls, such as these in the following figure, the users must access the controls in terms of their visual layout. For example to move from "Row" to "Selection", the user must move down twice. There is little information conveyed by this spatial layout, but the arrangement was chosen because it fit well within the dialog box. The user could as easily move to the right twice. This interaction style requires blind users to memorize visual layouts that conveys little meaning about the interface.

FIGURE 2 ABOUT HERE

The conceptual model that underlies the OutSpoken interface is the arrangement of information in a row-column format. This model was chosen because it is similar to previous text-oriented screen reader interfaces. Because the OutSpoken interface imposes little hierarchy (windows are the only grouping mechanism), moving through the objects in the above dialog would result in this order of spoken output: Row, Insert, Column, Delete, Selection and so on. Users are confused by this interaction since the semantic groupings that are obvious in the visual interface are not conveyed in the auditory interface

ScreenReader II by IBM, WindowBridge by SynthAVoice, and ProTalk by Hinter Joyce are products that provide access to the Microsoft Windows environment. As representatives of the second class of screen readers, these interfaces require the use of existing keyboard shortcuts provided by the Windows environment. Like OutSpoken, these products use only synthesized speech and braille output.

The reliance on the Windows keyboard shortcuts creates most of the usability problem with these screen readers. First, while the shortcuts provide more structural information than OutSpoken, they are designed to be augmented with the information in the visual display.

A more troublesome problem is that the shortcuts only allow the user to navigate to graphical objects that accept user input. Areas such as greyed out buttons and message bars are “invisible.” In order to access read-only information, the user must define view areas by a row-column position per application. The user can then create keyboard macros to read the information at a particular view area. These view areas are defined in a separate file or application profile, and this process requires the assistance of a sighted person.

The following list highlights difficulties that blind users had when working with these interfaces according to Edwards, A.D.N. (1989, 1991, 1992). This evaluation is primarily based on informal observation as well as discussions with users and developers.

- **Up, Down, Left or Right?**

  Users must understand the interface in terms of its spatial presentation. The reliance on the visual layout seems to be consistently problematic for blind users. Users describe their interactions with a graphical interface by first discussing the contents and structure of the interface, later augmenting their description by discussing the spatial layout. It appears that users are keeping two mental representations of the user interface, one based on structural information which is the primary model for the capabilities of the interface, and one based on spatial information, which is required to use the screen reader interface.

- **Dead Space, Occluded Space and Iconified Space**

  The blank space of the background, occluded portions of interfaces and desktop icons (minimized application windows) are confusing since these concepts are based on the limited visual display and do not map well to the limitless space of auditory interfaces. One interesting observation is that sighted users iconify application windows to de-clutter the visual display while blind users simply cycle through a list of active applications. Blind users typically complained about “falling in between objects." Whereas the dead space in visual interfaces helps users to identify individual objects, it is confusing for blind users who want to move from object to object.

- **Object Relationships**

  For a number of reasons, blind users have difficulty recognizing objects and relationships between objects in the same way as their sighted counterparts. The worst case is OutSpoken's interface where the label for a graphical icon, for example, a file folder, is treated as a separate object from the icon itself. Since users are often unaware of bounding boxes and other visual notation used to group objects, they do not perceive object groups.

- **Invisible Spots**

  Since most of the screen-readers rely on keyboard shortcuts, portions of the interface cannot be accessed by the blind user and are essentially "invisible." Keyboard shortcuts are designed for sighted users; therefore there is no need to have a keystroke for portions of the screen that do not accept user input.

- **Synthesized Speech Overload**

  Most screen readers rely on either speech or braille output. Many blind users report being confused by the use of speech for interface information ("Menu Button", "Window") and application information ("File", "Edit", "Microsoft Word") even when different voices are used.

- **Visual Characteristics Not Conveyed**

  Despite a reliance on the spatial layout, most screen readers fail to convey a great deal of visual information. For example, the relative size of an object, such as a menu or text field, is not conveyed in any of the screen readers discussed here.

The design of these screen reader interfaces are based on a number of design decision on how to model the graphical interface. For example, these interfaces often maintain the overall spatial organization of the interface, but do not convey visual attributes such as size or structural attributes used for grouping objects. The next section addresses these design decisions:

- What are the contents of the interface transformation?
- What is the underlying model for the interface transformation?

3. MODELING GRAPHICAL INTERFACES

3.1 Determining the Contents of the Interface Transformation

To motivate creating a model of a graphical interface, we examine a typical graphical interface as shown in the following figure. The question to be answered at this stage is*:*

> *"What are the characteristics and components of this interface that are critical to its use?"*

In contrast to the previous discussion on the advantages of graphical interfaces, during this section we need to categorize information about graphical interfaces that will be stored in our model.

FIGURE 3 ABOUT HERE

What are the *objects*?

A fundamental notion behind graphical interfaces is that the user directly interacts with *things*: objects or interactors that can be manipulated by the user in a set number of ways. In the example, there are a number of objects such as windows, radio buttons, push buttons, scroll bars, editable text areas, and read-only text areas such as message bars. These objects form the basis for how we conceptualize a graphical interface.

What are *attributes* of the objects?

Most interface objects are characterized by a number of visual attributes that help clarify their use. For example, many interactors can be highlighted (often indicating a current selection) and greyed out (indicating that the object is unavailable for use).

The relative sizes of different objects may be informative. In the screenshot, one of the text areas is much smaller than the others. This size indicates the type of text presented in this space, in this case, short diagnostic messages. Other size attributes are related to collections of objects. For example, the four radio buttons take up less space than the eighteen push buttons in the sample interface. Because sighted users quickly scan this information, they internalize that the radio buttons represent a smaller set of choices.

Other attributes are related to the spatial distribution of objects. For example, the sample interface is meant to be *read* from top to bottom, left to right, following Western reading conventions.

What are the *affordances* of the objects?

Objects in graphical interfaces can be categorized by their basic functionality. Many objects provide a means to group other objects. In this example, both windows and boxes collect objects into meaningful groups.

Users interact with objects in graphical interfaces in a set of predetermined ways. For example, a host of button objects support different forms of selection.

Text objects support the entering and manipulation of textual information, although the behavior of a text object can be constrained to indicate information about its contents. For example, the text in the first, large text object of the pictured interface, supports selecting a line of text since those lines represent email message headers. Some text objects support reading, but not editing of their contents.

What are the *relationships* between objects?

Another important question is how does the user perceive that objects are related to each other. Commonly, relationships are expressed via grouping. For example, the four radio buttons are not randomly scattered throughout the interface but are grouped together. In general, hierarchical relationships among objects inform us about the structure of the application interface. As already discussed, windows, boxes and white space convey groups of related objects.

Another primary relationship is cause and effect. In the example, selecting the push button "reply" causes the dialog box to be popped up. Selecting a message header causes the message to be displayed. If the interface response time is short, the user will *associate* these objects as having a cause-and-effect relationship.

What are the *names* of objects?

Although not visually depicted, many objects in the graphical interface have common names associated with them such as window, push button, and scroll bar. Since sighted and blind users will need to communicate with each other about application interfaces, it is necessary to retain naming conventions.

Completeness?

Is our model of the graphical interface complete? Depending on how we represent the model of the graphical interface with auditory cues, it may appear that we are discarding information in the graphical presentation. For example, we may not attempt to present objects at specific x,y location, but we may use the x,y coordinates to help determine the order of objects in the auditory interface. Likewise, we may not convey the amount of overlap between partially occluded windows, but we will likely support the notion of *focus* in the auditory interface.

There will likely be information in the graphical interface not conveyed in the auditory interface such as line width or color when these attributes do not convey information. The difficulty is determining, given a generic transformation, when the visual attribute is not meaningful.

Additionally, there are characteristics of the graphical interface that are difficult, if not impossible, to convey in an auditory interface, such as a persistent overview of the interface. How information about the graphical interface is utilized is determined by the underlying representation of the interface. In the next section, we compare three potential classes of models that could be used to represent the information about the graphical interface. This model provides the basis for the auditory interface.

3.2 Modeling the GUI

The next step in the design process is determining a model for graphical user interfaces. Since the model impacts both the user interface as well as the system design and implementation, it is necessary for us to consider the following questions when evaluating possible models:

- How well does this model capture important GUI characteristics?
- What kinds of auditory interfaces could be based on this model?

As an extreme example of a possible model, we could attempt to represent the GUI interface with musical notation. Although it would be easy to create an auditory interface based on musical notation, it would be quite difficult, if not impossible, to represent GUI characteristics with musical notation.

During this discussion, we compare three types of potential conceptual models. These are:

- **Spatial Models**

  The graphical interface is modeled as a 2 1/2 dimensional projection in space easily capturing aspects of the GUI such as the spatial distribution of objects. This model is primarily used by commercial screen readers.

- **Hierarchical Models**

  The graphical interface is modeled as a hierarchical structure, such as a tree or outline easily capturing parent-child grouping relationships. Most phone-based auditory interfaces utilize hierarchical interfaces implemented with menus.

- **Conversational Models**

  The graphical interface is modeled as a dialogue where the user can converse with the auditory interfaces. Natural language understanding coupled with voice recognition systems are used to implement these interfaces.

FIGURE 4 ABOUT HERE

Assessing Spatial Models

Using spatial models to represent graphical interfaces is attractive since graphical interfaces are presented using a spatial metaphor. The sighted user is presented with a spatially arranged picture of interface objects that can stack on top of each other in a 2 1/2D fashion. Many of the advantages of graphical interfaces, discussed previously stem from their static, spatial presentation. Obviously representing the graphical interface with a spatial model is not difficult, so the remaining question is what type of auditory interface could be based on this model.

The major difficulty with spatial models is that auditory interfaces are limited in their ability to present information spatially. Since 3D spatial sound systems cannot be used to produce a one-to-one mapping of the visual space to an auditory space, it would be necessary to present an abstraction of the graphical display. Just as maps serve as abstractions for physical space, the goal would be to create an auditory abstraction for the graphical space. Although this approach is feasible and worth future investigation, the analysis of existing screen readers points to two problems with this approach.

First, as discussed during the review of screen reader interfaces, blind users find it difficult to work with spatially arranged user interfaces. Many users conceptualize the interface based on its logical structure and then attempt to memorize the spatial presentation. Although it is clear that blind people can successfully navigate physical spaces such as their home, one user likened working with a graphical interface with trying to navigate a large, unknown room where it is “easy to get lost and become disoriented (Day, 1995).”

Second, although one could argue that existing screen reader interfaces have not provided the right spatial abstraction for a graphical interface, finding such an abstraction is difficult because graphical interfaces have been optimized for visual presentation. The need to fit the graphical interface into a limited visual space results in spatial layouts that are not informative, but are the result of conserving screen real estate. In a generic analysis of an X Windows graphical interface, it is impossible to determine *when* spatial layouts are informative. Although current screen reader interfaces have attempted to provide the benefits of a spatial organization, their users more often are confused by spatial arrangements that convey no meaning.

Assessing Hierarchical Models

Many auditory interfaces are based on hierarchical models (see Ly, 1993 & Schmandt, 1993). For example, interfaces for voice mail allow the user to navigate through a hierarchy of choices for listening to and deleting messages. Hierarchical models are used because they can abstractly represent groups. It is also relatively easy to navigate these auditory interfaces using keypad or voice input, although the requisite path from one object to another may be lengthy. Since hierarchical structures represent discrete, as opposed to continuous, values, they are well suited for conveying discrete objects.

Given that there are previous examples of complex, hierarchical auditory interfaces, the primary question is how well graphical interfaces can be modeled using a hierarchical structure. A tree-structure representation of the graphical interface in Figure 3 is shown in the following figure. The tree structure lends itself to representing the objects in their interface, as well as the parent-child relationships between those objects. Cause-and-effect relationships can be modeled as additional links in the structure. In the example, pushing the reply button causes the pop-up dialogue to appear.

FIGURE 5 ABOUT HERE

A significant limitation of this model is that it does not capture visual attributes of the graphical interface. Some representations of visual cues are possible. For instance, the ordering of objects in the structure can be based on their spatial arrangement in the graphical interface. Likewise the size of grouping objects, such as windows, is partially represented as the number of children. Nevertheless, this model suffers from its inability to explicitly represent all the visual characteristics of the graphical interface, These characteristics can be stored as attributes of the objects in the hierarchical structure. The auditory interface would be responsible for conveying these attributes, in addition to conveying the underlying model.

Assessing Conversational Models

Another class of auditory interfaces commonly uses conversational dialogues as the basis for the user interfaces. For example, both Stifelman's Conversational VoiceNotes (1993) and Yankelovich's SpeechActs (1994, 1995) utilize voice recognition technology as the primary means of input to an auditory interface. VoiceNotes provides an interface to a hand-held notes organizer while SpeechActs provides an interface to desktop applications such as email and calendar. Typical user input phrases are:

List notes for July 12th

New appointment with Jim Foley this Friday at 3pm

Both of these interfaces are replicating functionality that can be found in a graphical interface. SpeechActs is actually a front-end for graphical desktop programs. Given that it is possible to create useful auditory interfaces using conversational models, the remaining question is how does this model work with our goal of modeling graphical user interfaces. There are two problems with using these models for our task:

- Requires Domain Knowledge

  The example input phrases above illustrate that these interfaces rely on understanding the domain of the application interfaces. In our automatic analysis of graphical interfaces, it is unlikely that we will obtain sufficient information to build a domain-dependent dialogue.

- Interaction Significantly Different than Graphical Interface

  Sighted users and blind users will not have the same building blocks for discussing how to operate an interface since the conversational interface hides components of the GUI such as menus and buttons.

Choosing a Model

We have based our representation of the graphical interface on a **hierarchical** conceptual model since best captures the underlying structure of the graphical interface without requiring domain-specific knowledge of the graphical application. The primary relationship represented in the hierarchical model is the parent-child relationship between interface objects. These relationships appear to be the basis for how blind users conceptualize graphical interfaces. In many ways, they are likely to be the basis for how all users conceptualize graphical interfaces given the importance of structural information in this class of interfaces. Spatial organizations are problematic since graphical interfaces typically generate spatial layouts based on space-conserving constraints that are often confusing for blind users. Conversational models require domain knowledge to capture the functionality specific to the application interface. This information would be extremely difficult, if not impossible, to obtain from a generic X Windows application.

One important limitation of the hierarchical model is that it does not effectively capture the power of a visual, spatial presentation. Two advantages of the visual interface are that the user can quickly recognize interface objects from the bit mapped pictures on the screen, and that the user can quickly scan the collection of onscreen objects. Therefore, two critical requirements of the design are that the user can quickly recognize interface objects and that the user can quickly survey the contents of the interface.

4. MERCATOR INTERFACE DESIGN

In this section, we describe the basic interface design for Mercator. The primary question that we address is:

> *Given the hierarchical model of the graphical interface, what auditory interface do we present for a blind user?*

The inherent disadvantage of all auditory interfaces is they are largely invisible. For this reason, a significant portion of this design will focus on **conveying the contents** of the auditory interface. Users must be able to determine the identity and attributes of the various objects that make up the auditory interface.

In addition to recognizing individual objects, the user must be able to navigate the space of the interface. The controls for **navigation** must support the user's mental model of the auditory interface. For example, moving the mouse cursor across a graphical screen supports the notion of the interface as a picture in 2D space. Navigation must be safe so that navigation is orthogonal to manipulating the user interface.

After users are able to navigate the auditory interface and identify the objects within it, they need the ability to **manipulate** those objects to accomplish their tasks. The most common manipulation is the ability to select an object whether it is a menu button or a text field. In the graphical interface, selection is generally accomplished by clicking on a mouse button. When we manipulate an interface, changes in the interface convey **feedback** as to the ramifications of our actions. The auditory interface must also provide conventions for manipulating the interface and providing feedback to the user.

4.1 Conveying Auditory Objects

To convey the contents of the auditory interface, it is necessary to convey the types of objects in the interface as well as attributes and affordances of those objects.

Conveying Object Identity

Numerous strategies for conveying objects in auditory interfaces have already been suggested by previous work. Possible strategies include using speech, pure tones, earcons, or auditory icons. For example, an auditory cue to convey a text-entry field could be:

- A synthesized voice saying “text-entry”
- A pure tone such as G-sharp (~ 415.3047 Hz)
- A musical timbre of a violin
- The sound of an manual typewriter

Each of these approaches has advantages and disadvantages. The speech message is unambiguous and reasonably efficient, but may be confused with other speech messages, i.e. reading the label on the field. A pure tone is easy to produce and takes minimal time to hear, but may be confused with other pure tones. Also the mapping of the note G-sharp to a text-entry field would be difficult to remember. Various musical timbres would also be easy to produce, and would be easier to discriminate than pure tones, but again, the mapping from violin to text-entry is hardly intuitive.

This design is based on the premise that auditory icons (see Gaver 1988, 1994) offer the most promise for producing discriminable, intuitive mappings. In the previous example, the sound of an old-fashioned typewriter maps easily to a text-entry field. The user is reminded of typing or entering text. In general, the use of auditory icons mimics how information is conveyed in graphical interfaces. We recognize many objects in graphical interfaces by their physical appearance. Sometimes concrete representations are used such as the picture of a trashcan. Abstract icons also leverage our understanding of the physical world. Although Motif push buttons do not look like button controls in the physical world, they look pushable. Likewise, an auditory icon may not sound like a real push button, but the sound may indicate an object that can be pushed.

Two alternate design strategies that were considered and discarded were using speech or earcons. Synthesized speech is required for presenting textual information in the graphical interface. This information is domain-dependent, such as the text in an electronic mail message or the labels on a pull down menu. By relegating speech to domain-dependent information, and respectively relegating nonspeech cues to domain-independent information, the user can more easily separate these classes of information[1]. The structured combinations of musical sounds employed in earcons by Blattner and others (1991, 1992, 1994), have been successfully used in providing access to mathematical equations for blind users (see Stevens, 1995). That use of earcons was especially compelling since the natural prosody for reading mathematical equations mapped well to the rhythm of presenting successive earcons. In Mercator, the primary role of the auditory cues is to convey the types of objects in the graphical interface. We concluded that iconic, everyday sounds would be more intuitive than abstract, musical sounds.

In Mercator, we use a set of auditory icons to convey the identity of various interface objects. Some auditory icons are fairly concrete like the typewriter and the printer, while the sounds for various buttons are more abstract. The following table provides a listing of some of the auditory icons used in Mercator. The selection of sounds was based on a series of experiments exploring how people describe sounds and how they map concepts in graphical interfaces to sounds. These experiments are discussed in (Mynatt, 1994, 1995).

FIGURE 6 ABOUT HERE

Conveying Object Characteristics

From our model of the graphical interface, we know there are many characteristics of the interface objects that we need to convey to the user. The use of auditory icons often serves to convey the affordances of the objects as well. For example, the typewriter sound should convey the affordance of entering text just as the push button sound helps convey the notion of pushing. But there are other attributes of objects we need to convey such as its label, whether it is greyed out, and its relative size.

---

1. Since speech sounds are often less ambiguous than nonspeech, everyday sounds, speech output plays a role in supporting first-time users. We use redundant speech output to help users learn the meaning of the different nonspeech cues. The design of user levels is discussed later in this paper.

Text-based attributes can be presented via synthesized speech. For example the auditory icon for a push button can be presented simultaneously with its text label. Other attributes can be presented by modifying the base auditory icon.

Auditory icons are not limited to simply reflecting categories of events and objects, but can be *parameterized* to reflect their relevant dimensions as well. For example, the auditory icon for a file can be manipulated to convey the size of the file. Gaver's (1994) techniques for parameterizing auditory icons are similar to the *filtears* described by Ludwig, Pincever and Cohen (1990,1991). We used the following filtears because they could process sounds in real-time[1]:

- Muffling

  High frequency energy in the auditory cue is removed, causing the cue to sound deeper in pitch with reduced intensity.

- Thinning

  Low frequency energy in the auditory cue is removed, causing the cue to sound higher in pitch with increased intensity.

By combining these filtears with modifying the overall intensity of the sound, we can create the impression of an auditory object being selected or greyed out.

Since muffling or thinning a sound affects our perceived pitch of the sound, we use these filtears to modify other auditory icons where the pitch of the auditory icon can be associated with its size or spatial location. If we strike two metal bins where one bin is

1. Facilities for muffling and thinning audio samples, as well as for playing, mixing and interrupting sounds was provided by NetAudio II, a tool developed by David Burgess (1993).

much larger than the other, the sound of the larger bin will have a lower perceived pitch. Containers are objects in Mercator that group other objects, such as a collection of push buttons. The auditory icon for a container is an opening door. We modify this sound to indicate the number of items in the container. We use the same technique for text areas, so that the perceived pitch of the typewriter is based on the number of lines.

Sometimes it is helpful to convey the spatial location of an object or its position in a serial order. We modify the cursor sound to indicate how many lines down the cursor is in a textual list. We also slightly modify the sounds of grouped buttons indicating a button's location in the serial order. The modification is slight because extreme modifications are reserved for conveying the selection state of a button (selected, normal, greyed out).

FIGURE 7 ABOUT HERE

4.2 Navigation

In addition to recognizing individual objects, the user must be able to navigate the space of the interface. The controls for navigation must support the user's mental model of the auditory interface. For example, moving the mouse cursor across a graphical screen supports the notion of the interface as a picture in 2D space. Since the conceptual model of the auditory interface is a hierarchical structure, the navigation controls should map to moving throughout that structure. For the controls to feel *automatic,* it is necessary for the meanings of the controls to be consistent throughout the interface, just as moving a mouse is consistent across the screen. This design is in contrast with typical phone interfaces where the meaning of the control (*Press 1 to do this*) is often context dependent.

Another comparison to mouse navigation is that navigation must be safe in the sense that it is orthogonal to manipulating the user interface. When users move a mouse across the screen, the interface may respond to give more information, but users are generally safe from triggering potentially harmful events such as stopping or starting an application. To support this separation, the navigation controls need to be distinct from the controls used to select or otherwise manipulate objects.

In Mercator, at the simplest level, the user uses the arrow keys on the numeric keypad to navigate a tree structure that corresponds to the condensed, hierarchical model of the graphical interface. The user presses up and down to move in and out of groups of objects and presses left and right to move within groups of objects. When the user moves to an object, they hear the auditory icon (possibly filtered) for that object. If the user attempts to move in a direction where no object exists, e.g. moving right when you are at the end of a cluster of push buttons, they hear a simple error sound of a ball bouncing against a wall. The premise is that the users reinforce their mental model of the auditory interface since the navigation is explicitly based on the hierarchical structure.

For the graphical interface pictured in Figure 3 whose respective tree structure in shown below, a user navigating from the top of the structure to the push button "delete" would hear:

FIGURE 8 HERE

This technique would be tedious if the user had to listen to the entire auditory icon each time they moved to an object. Although the auditory icons are short, average of one second, they are interruptible within approximately 50 ms. This set-up allows the user to quickly move throughout the interface. Also, it is important to remember that the navigation controls are consistent throughout the interface. Expert users seem to exhibit a form of muscle memory where they quickly press a sequence of keys to jump to parts of the interface. Some users even orient themselves by quickly moving to an “edge” in the tree structure, hearing the out-of-bounds sound, and then proceeding. Overall the feel of the navigation is quick and responsive.

Navigation Shortcuts

The persistent image of the graphical interface coupled with mouse input allows sighted users to quickly move from one portion of the interface to another. Even though Mercator users can quickly move throughout the interface, it is beneficial to provide keyboard shortcuts for expert users. One useful shortcut is the ability to move to the beginning, or end, of a group of objects. This jump is accomplished by hitting the 1 and 3 key, respectively, on the numeric keypad. The user hears quick snippets of the auditory icons for the objects that are “passed over” by using the short-cut.

Although the user could switch between applications by navigating to the “top” of an application and over to the next application, the user can press the right or left arrow key coupled with the Shift or Alt key to switch between applications. When the Alt key is used, the user is moved to the “top” of the next application. When the Shift key is used, the user is moved to their last location in that application saved from the last time they used that application. This control helps the user recover their working context within an application. When the user switches between applications, they hear a paper flipping sound that should remind the user of switching between tasks, as well as the windows that are popping to the front of the screen contents. The new application name is announced with a message, such as, “Framemaker is the current application.”

The user can also set hot keys for the row of keys above the numeric keypad. These keys can be used to move to a designated spot in an application interface that is specific to that application. The numeric keypad, annotated with the navigation controls, is pictured in the following Figure 9.

FIGURE 9 ABOUT HERE

Although the navigation short-cuts assist the user in moving quickly throughout the hierarchy, they still do not afford the same freedom as quickly moving the mouse from one part of the screen to another. Different interaction styles not explored in this research include using a spatial model for the interface where the user could operate the mouse to move from one portion to another. Likewise, a tactile interface representing the tree structure could be used to provide a persistent overview as well as a medium for large jumps in the interface.

Auditory Preview

One limitation of auditory interfaces is the difficulty in presenting an overview of the interface contents. When sighted users look at a graphical interface, their eyes can quickly scan the interface to get a rough determination of its contents. Sometimes they can tell if they are where they want to be by the visual features of the interface. This technique applies to reading text as well. Robert Steven's (1995) design of an auditory preview of mathematical equations can be applied to previewing portions of the Mercator interface. An auditory preview is simply short snippets of auditory icons that are played in quick succession. By the overall length and diversity of sounds in the preview, the user gets a rough sense of the contents. The user can ask for previews of any group of objects, for example, objects grouped in a container or in a pop-up dialogue.

Sometimes the user does not need an auditory preview, but simply needs a reminder about the current object. In a visual interface, we can look away and then look back, regaining our visual focus. A user of an auditory interface may also need to regain the auditory focus. By pressing the 5 key, the user hears the auditory cue (may be a combination of nonspeech and speech output) for the current object. The inclusion of this feature is a simple example of learning from user feedback. The first Mercator interface did not include this control, and users (including us) would navigate away from and then back to the current object to regain the auditory context.

4.3 Manipulating the Interface and User Feedback

Up to this point, the description of the user interface has focused on the user perceiving and navigating the contents of the auditory interface. The next step is allowing the user to manipulate the interface. In this section, we describe how the users manipulate Mercator interfaces, as well as the feedback that the user receives from Mercator. The auditory feedback cues used in Mercator are summarized in Figure 10.

Selection

A principal action that users perform with graphical interfaces is selection. The action is typically accomplished by clicking (or double-clicking) on an object with the mouse. Since the Mercator user is working with the keyboard and not with the mouse, mapping selection to a keystroke reduces the distance that the users's hand must move. In Mercator, pressing the Enter key on the numeric keypad is mapped to selecting an object. Mercator can determine what mouse events the application expects (e.g. single or double click) and then simulate those events for the application.

Given the limitations of manipulating sampled sounds, creating pairs of sounds for {this is a push button, you just pushed a push button} was too difficult given the set of auditory icons used in Mercator. If the user successfully selects an object, the user will hear a short sound akin to someone ripping a batch of papers. This sound was chosen because it seemed to imply that something was happening, indicating to the user that the selection event had taken place. Since few users could actually identify the sound as ripping papers, they did not express any negative connotations about the sound. A longer discussion of the action-oriented content of sounds is presented in (Mynatt, 1994, 1995).

Pop-up Windows

Pop-up windows are an interesting case of the content, structure and focus of the interface changing almost instantaneously. When a pop-up window appears, the space of the interface (its content) is now augmented with the contents of the pop-up window. Likewise the structure of the interface, and our hierarchical model, is augmented by the structure of the pop-up. Often the input focus of the interface is moved to the pop-up as well, for example, modal pop-ups that require users to confirm or cancel an action before proceeding.

In graphical interfaces, pop-up windows capture the user’s attention by being drawn on top of the other windows. In Mercator, whistling sounds are used to notify the appearance or disappearance of a pop-up window. A whistling sound with a rising pitch indicates that a pop-up has appeared, while a descending pitch indicates that a pop-up has disappeared. If the input focus is shifted to the pop-up, the user is moved to that location in the application tree structure. This move is indicated by the auditory icon for the pop-up window, a springy sound. There is a deliberate attempt to reinforce the terminology of pop-up window with these sounds. Both the whistling and spring sounds help form the illusion of something popping up in front of you.

When the user dismisses a pop-up, they are placed in their original location, where they were before the pop-up appeared. For example, in Figure 3, when the user presses the reply push button, they hear the following sounds as the pop-up appears on the screen:

"Rip" the selection is successful

"Whistle-up" the pop-up appears on the screen

"Spring" the user is moved to the top of the popup structure

If the user navigated to the cancel button, selecting that button and thereby dismissing the pop-up, they would hear:

"Rip" the selection is successful

"Whistle-down" the pop-up disappear from the screen

"Ca-chunk" "Reply" the user is moved back to the reply push button

Pop-up windows are stored in the interface model as descendants of the uppermost node of the application tree structure since they are perceived as separate windows on the screen. When the pop-ups are not modal, the user can navigate up out of the pop-up and back to the main application structure. If the pop-up is modal, such as requiring a confirm or cancel operation, the user is not allowed to navigate out of the pop-up, retaining the semantics of the interface.

FIGURE 10 ABOUT HERE

Interacting with Text Objects

Screen readers for text-based interfaces, such as the command line interface to DOS, have existed for many years. These interfaces have formed a set of standard requirements for reading and manipulating text areas[1]. One requirement is support for two "cursors," an edit cursor that is located at the insert position in the text, and a review cursor that can be moved independently to read portions of the text. Operations for moving and synchronizing the cursors are coupled with operations for reading text by character, word, line, sentence and paragraph. Different filters are used to parse and pronounce the text based on the current task requirements. For example, a Unix filter can be used so that the command:

more dissertation.text | grep Mercator

would be read as:

more dissertation dot text pipe grep Mercator

To review a text area, the user is required to enter "text mode" by pressing the ./Del key. For example, when the user navigates to a text area (hearing the typewriter sound), they then press the ./Del key to enter text mode. This operation is accompanied with a rolling/ rocking sound to indicate moving into a different state. While in text mode, the keys on the numeric keypad are mapped to operations for reviewing text. The users can return to navigating the interface by exiting text mode, again pressing the ./Del key and hearing the rolling sound.

Some of the commands provided in Mercator for reading and manipulating text are summarized in the following figure.

FIGURE 11 ABOUT HERE

1. Although there is not a paper detailing requirements for text-based screen readers, we were able to determine the needed functionality by examining existing screen readers and talking with blind computer users.

User Levels

Based on experience with demonstrating and evaluating Mercator, it became clear that the interface could be modified to support the transition from a novice to expert user. The tcl interface code was easily extended to support three user levels (Novice, Intermediate, Advanced). The primary modifications focused on information presented to the user when they navigated to an object, and when they requested information about an object. Based on observations of people using Mercator, three stages of learning became apparent.

- **Recognizing auditory icons**

  The users learned the sounds for push buttons, text areas and so on.

- **Parameterized auditory icons**

  The users learned how the auditory icons are manipulated to convey attributes of objects such as a push button being greyed out.

- **Understanding modes**

  Users learned that they have to enter "text mode" to review the contents of a text area.

The current user level determined the amount of redundant speech information. What the user would hear, per user level, after navigating to a greyed out push button labeled undelete, is shown in the following table.

FIGURE 12 ABOUT HERE

When the user asks for information about an object by pressing the 5 key, they hear the information corresponding to one level less experienced than their current setting. This strategy helps users transition between levels. For example, a user can switch to operating as an Intermediate, but still get additional information for objects that they have forgotten or have not encountered.

5. ASSESSING MERCATOR'S INTERFACE

During the course of this research we have utilized many methods for assessing Mercator's design including discussing our design with users and other designers, observing people using Mercator as well as observing how people teach others to use Mercator, and measuring the performance of people conducting specific tasks. To collect quantitative data on the learnability of Mercator, we measured how quickly sighted users reached peak performance in a specific task of reading and replying to email messages using a graphical email application. One motivation for using sighted people in this experiment is that we also examined the effects of transitioning between using the graphical and auditory versions of the same application.

We have also compiled reactions by blind users that we have received over the past three years. We did not perform controlled experiments with blind users for two reasons. First, previous experience with computers appears to be an overriding factor in how well blind users perform with screen readers for graphical interfaces. It would have been difficult to control previous experience so that performance data would be meaningful across subjects. Second, the available sample of blind users in the Atlanta area generally had no computer experience. In contrast, users attending conferences for assistive technology, generally had comparable experience with computers and were motivated to use graphical interfaces. We discussed Mercator's design with potential users at over ten conferences that included an emphasis on assistive technologies. At three of the these conferences, Mercator was available for use over multiple days among the product exhibits. From these experiences, we have summarized favorable and critical assessments of Mercator interfaces.

5.1 Measuring Performance with Mercator

Having already observed that blind users could learn to use Mercator, we wanted to assess how well sighted users performed with Mercator for two reasons. First, we needed a controlled setting in which we could measure the time needed to learn to use Mercator. Based on demonstrations with blind and sighted users, it appeared that computer literate sighted users took longer to learn the interface than computer literate blind users, but that the stages of learning were the same.

Second, we wanted users to contrast their use of a graphical interface and the Mercator-derived auditory interface. One hypothesis was that experience with the graphical interface would be beneficial in using the auditory interface since the two interfaces share the same structure.

In order to assess users' performance with the auditory interface, as well as determine the effects of prior experience with the graphical interface, test subjects worked with graphical and auditory versions of the application xmailtool. A screen shot of the graphical interface is shown in Figure 3.

Quantitative data was calculated from analyzing activity logs. The logs indicated each time an event had occurred in the interface, such as moving to a new object, entering or exiting text mode, or selecting an object. With the subjects' consent, we videotaped the sessions including training and debriefing in addition to the test trials.

Seventeen subjects worked with combinations of the graphical and auditory interfaces. The subjects were randomly divided into four groups as shown in the following table. The group designation determined which interfaces they used, and in what order. For example, in Group 2, the subjects started with the graphical interface, but switched to the auditory interface after four trials. The subjects in Group 3 only used the auditory interface.

FIGURE 13 ABOUT HERE

In each trial, the subject selected, read and replied to three specified email messages.

The training for the experiment was conducted in three stages.

- **Description of the Task**

  I told the subjects the details of the task they were to perform, namely that they were to locate, read and reply to three specified email messages. I explained that in each message was a test phrase that they would need to type into their reply.

- **Description of the Interface**

  At this point, I either described the graphical interface or the auditory interface. I explained how to navigate the interface and how to select objects.

  I also showed the subjects a diagram of the common structure of the graphical and auditory interfaces similar to the diagram in Figure 5.

- **Demonstration of the Task**

  I demonstrated replying to one email message. I went through all of the steps including writing and sending the reply.

Learning the Auditory Interface

The most promising result of the experiment is that all of the subjects were able to learn how to use the auditory interface. The average time to complete a trial sharply decreased after one trial with peak performance achieved around the fourth trial (see Figure 14). Another important result is that the variance in time taken sharply decreased after one trial (average of 401.33 to average of 88.96).

FIGURE 14 ABOUT HERE

Stages of Learning

So what did the subjects learn? From observation and inspection of the data, it appears that four concepts were acquired in the approximate order:

1. Basic Auditory Icons

Since the subjects had only heard a brief demonstration, they needed to spend some time recognizing and learning the auditory icons. Although ten sounds were needed in this interface, the subjects never asked what a sound meant. They seemed to use the 5-Info key to hear the auditory icon coupled with redundant speech information until they learned the meaning of the auditory icon.

2. Navigation

The biggest hurdle in using the interface is understanding the hierarchical navigation scheme. Subjects who had never seen the graphical interface had to learn that the down and up keys took them in and out of groups of objects. Improved navigation times greatly contribute to overall improved performance times. In part, improved navigation times seemed to be impacted by how safe the users felt. As users realized that they could navigate the interface without causing unwelcome consequences, they increased their rate of input, “bouncing off the walls” when they went too far in any direction.

3. Parameterized Auditory Icons

As the subjects continued using the interface, it became apparent that they were learning to listen for the pitch differences between auditory icons of the same class. For example, the reply push button is in the container with 18 children. This container has a deeper sound than other containers in the interface. Likewise the text areas with the headers and message are much larger than the text area with diagnostic output. Subjects learned to listen for the container and text areas with a lower pitch helping them locate these objects faster and more reliably.

4. Text Mode

A common guideline in human-computer interaction is to avoid modes in the interface. Mercator has one mode and it proved problematic. When users navigate to a text area, they need to enter text-mode so that the numeric keypad can then be used to navigate the text as opposed to navigating the rest of the interface. Often a subject would reach a text area, but not remember to switch into text mode. Common guesses were selecting the text area and trying to navigate down into the text area (not a bad idea!).

Transitioning between the Graphical and Auditory Interfaces

One way to demonstrate that the auditory interface captures critical characteristics of the graphical interface is to look for a *transfer effect* when the user transitions from using the interface in one modality to using the interface in the other modality. For example, if the user has experience with the graphical interface, this experience should help the user learn the auditory interface. Unfortunately quantitative measurements did not demonstrate that such an effect took place. There are two reasons related to the experimental design that may help explain why the transfer effect was not evidenced:

- **Exposure to Interface Structure**

  During the training, I showed all of the subjects a diagram of the interface structure. Users of the graphical interface paid little attention to the diagram. In contrast, users of the auditory interface studied the diagram and indicated that they would have preferred to consult the diagram during the task. The information in that diagram is in essence the information that should cause a transfer effect. Experience with the graphical interface should give the user information about the structure of the interface. By showing the diagram to the users of the auditory interface, I accidently gave those users the same information that the transfer effect is based on. Therefore the effect was hidden by the improved performance of the users of the auditory interface.

- **Graphical Task Too Easy**

  Performing the task with the graphical interface required little cognitive effort. The performance times increase over the trials is likely due to boredom. The subjects spent most of their time trying to determine what I was actually testing them on. One subject asked me if I was manipulating the lights in the room. Since they did not have to think about the task, they internalized little information about the content of the graphical interface.

During the debriefing, subjects who first used the graphical interface and then used the auditory interface made three interesting observations:

- **Exposure to Graphical Interface Helped in Using Auditory Interface**

  Although not evident in the quantitative analysis, the subjects reported that their experience with the graphical interface was helpful in understanding the auditory interface. Aspects of the graphical interface that were helpful included knowing the objects in the interface, the spatial ordering of the objects, and the relative sizes of objects.

- **Subjects Needed to Update their Simple Model of the Interface**

  Although subjects reported that their exposure to the graphical interface was helpful, they remarked that they needed to form a more complex model of the interface when working with the auditory interface. They did not describe forming a new model, but augmenting their simple model with more information. For example, in the graphical interface, the subjects could easily ignore a number of objects in the interface. Since they had to navigate past these objects in the auditory interface, they needed to augment the interface model with these objects.

- **Initial Transition Between Spatial and Hierarchical Was Difficult**

  Since the subjects had a fresh visual image of the graphical interface in their minds when they began working with the auditory interface, they typically tried to navigate the interface based on the spatial layout. Most of the interface objects are arranged top to bottom in the graphical interface, but since they are sibling objects, they are accessed by moving left and right in the auditory interface. Navigation errors from trying to move in spatial directions generally disappeared during the first trial.

5.2 Observations by Blind Users

Reactions to Nonspeech Auditory Cues

Blind users have expressed an overwhelming positive response to the use of everyday sounds in screen reader interfaces. As noted previously, users of current screen readers have difficulty separating interface information, such as “Push Button’” from application information, such as “Edit,” when both types of information are presented with speech or braille. When blind users were able to work with and listen to the Mercator interface, they remained impressed with the use of everyday sounds. In addition to particularly liking the typewriter and whistle sounds, in contrast to sighted users, blind users liked the container sound and were rarely confused about its use. Users commented that the filtering of the auditory icons was subtle, noting that many designers unnecessarily exaggerate changes in audio.

Reactions to Hierarchical Interface Structure

In contrast to the use of everyday sounds, blind users were skeptical about hierarchical navigation schemes as presented during design briefings. The general consensus was that they needed to “know what was on the screen” since that was what their sighted counterparts used. Only after using Mercator, did users express their preference for this scheme.

Users have requested that Mercator allow them to print out information about the structure (object hierarchy) of an interface using a braille printer. Users experimenting with this strategy refer to consulting the constant tactile image while exploring the auditory interface. The tactile image seems to provide some of the functionality that the constant visual image provides to sighted user.

A new screen reader also uses an underlying hierarchical model. The system, called Virgo, transforms Microsoft Windows interfaces into braille interfaces. Instead of using auditory icons, the first two braille cells contain a code that represents the type of objects, and the remaining braille cells contain the label and highlighting information.

5.3 Meeting Goals for Screen Reader Design

After discussing the benefits that graphical interfaces provide for sighted users, we outlined six goals for transforming graphical interfaces into auditory interfaces. Given the auditory interface design discussed in this paper, how well does Mercator meet those goals?

- **Access to functionality**

  By providing general strategies for representing graphical interfaces with auditory interaction techniques, Mercator provides transparent access to applications for word processing, electronic mail, calendars and so on. In these text-based interfaces, spatial information in the interface is generally mapped to structural information. One exception is when domain specific information such as the text in a document is searched spatially using the controls for manipulating and reading text.

  One advantage of Mercator is that all objects in the interface are treated as first class objects. In contrast to current screen readers, users do not have to define special view areas to access portions of the interface that do not accept user input such as message bars.

- **Iconic representations of interface objects**

  When possible, interface objects are grouped into the same discrete objects that sighted users perceive. The identify and attributes of these objects are conveyed with auditory icons. Like their graphical counterparts, auditory icons leverage knowledge of the real world in presenting interface output.

- **Structural organization**

  A central motivation in Mercator's design is to convey the underlying structure in graphical interfaces. The groupings of objects, conveyed with visual cues in the graphical interface, are made evident as the user navigates into, within, and out of object groups. These groups help clarify the functionality of individual objects.

- **Direct manipulation**

  As in graphical interfaces, users directly interact with objects in the interface and interface output is conveyed via the objects. In as much as the graphical interface provides objects that match how user's conceptualize tasks with the application, Mercator provides a direct manipulation interface.

- **Spatial arrangement**

  A primary difference between Mercator and commercial screen readers is that Mercator is based on a hierarchical model of the graphical interface as opposed to a spatial model. Mercator, however, does provide information about the spatial attributes and layout of the graphical interface. The relative sizes of objects are conveyed by manipulating their base auditory icons. Information about the layout of text is conveyed by modifying the sound of the edit cursor as it is moved throughout the text. The layout of objects helps determine their ordering in the auditory interface.

  Nevertheless, information about the layout of the interface is lost in this representation. Likewise, users are not able to arrange application windows along spatial dimensions. This design trade-off was made to offset existing usability problems with commercial screen readers.

- **Persistent presentation**

  A benefit of visual interfaces is that they exist in physical space and can be reviewed over time creating a surrogate short-term memory for recalling the contents of the user interfaces. This type of persistent presentation is difficult to achieve in a complex auditory interface where multiple continuous sounds are confusing and distracting. To improve the user's scanning capabilities, we provide the preview facility. The short snippets of the auditory cues help convey portions of the interface, and is sufficiently succinct to confirm the user's location in the interface.

  Some blind users have experimented with using braille printouts of the interface structure. The users refer to this constant tactile image while exploring the auditory interface.

6. FUTURE WORK

This design is effective for blind users working with text-oriented applications such as word processors, electronic mail and other menu and form-based interfaces. The challenge of providing access to more graphical applications such as drawing programs remains. One area of future research is incorporating the use of a tactile display. The auditory and tactile displays could be used to create complementary presentation of the graphical interfaces. The tactile display would help offset some of the limitations of the auditory interface by providing a constant presentation of the interface as well as supporting large moves across the space of the interface.

In many ways, this research addressed an important, but overly constraining problem of transparent access to graphical applications. Once the constraint of transparency is relaxed, one could imagine combining aspects of spatial and conversational interfaces into the default hierarchical interface to leverage domain-dependent interaction. The potential of adding voice interaction is especially compelling. We have extended the Mercator architecture to include voice as a potential input source, but we have not furthered explored its use. The inclusion of a spatial model could aid in providing access to a broader range of applications that inherently include spatial content such as drawing and map-based tasks.

Sighted computer users could also benefit from auditory representations of graphical interfaces while performing eyes-busy tasks such as driving, performing maintenance on a airplane, or inspecting a manufacturing plant. The needs of these users will be different however. For instance, supporting mobility will likely be a key requirements. In these cases, improving the flexibility of conversational interfaces may provide the most promise.

Background. This article is based on the Ph.D. thesis of the author. Elizabeth Mynatt joined Xerox PARC in 1995 from the Georgia Institute of Technology. At Georgia Tech, she completed her doctorate in computer science by developing a method for transforming graphical user interfaces into auditory user interfaces. One application of this work is providing access to graphical interfaces for blind users.

Acknowledgements. The Mercator project represents a multi-year, multi-person effort. Thanks go to Keith Edwards, Tom Rodriguez, Kathryn Stockton, Ian Smith, Sue

Liebeskind, Sue Long, Kevin Chen, Will Luo and Stacy Ann Johnson for their design and implementation contributions.

Support. This research was supported by Sun Microsystems, the NASA Marshall Space Flight Center, the National Security Agency, and the Georgia Institute of Technology.

Author Address. The author can be reached at: Xerox PARC, 3333 Coyote Hill Road, Palo Alto, CA 94304, USA. Email: `mynatt@parc.xerox.com`

HCI Editorial Record.

REFERENCES

(1989). outSPOKEN, The Talking Macintosh Interface. User Manual. Berkeley Systems.

Blattner, M., Glinert, E. P., and Papp, A. L., III. (1994). Sonic Enhancements for 2-D Graphic Displays: In G. Kramer (Ed.), *Auditory Display: Sonification, Audification and Auditory Interfaces* (pp. 447-470). SFI Studies in the Sciences of Complexity Proc. Vol. XVIII, Addison-Wesley.

Blattner, M. M. & Greenberg, R. M. (1992). Communicating and Learning Through Non-Speech Audio: In A. Edwards & S. Holland (Eds.) *Multimedia Interface Design in Education* (pp. 133-143). Springer-Verlag, NATO ASI Series F.

Blattner, M. M., Sumikawa, D. A, & Greenberg, R. M. (1991). Earcons and Icons: Their Structure and Common Design Principles. *Human-Computer Interaction* 4(1), 11-44.

Boyd, L.H., Boyd, W.L. & Vanderheiden, G.C. (1990). The graphical user interface: Crisis, danger and opportunity. *Journal of Visual Impairment and Blindness*, December. 496–502.

Burgess, D. (1993). *The NA3 Audio Server.* Final report to Sun Microsystems.

Day, G. (1995). Personal communication.

Edwards, A. D. N. (1989). Modeling blind users' interactions with an auditory computer interface. *International Journal of Man-Machine Studies*, 575-589.

Edwards, A. D. N. (1991). *Evaluation of Outspoken software for blind users* (Technical Report YCS150) University of York, Department of Computer Science.

Edwards, A. D. N. (1992). Graphical User Interfaces and Blind People. *Proceedings 3rd International Conference on Computers for Handicapped Persons*, 114-119. Vienna.

Edwards, W.K., Liebeskind, S. H., Mynatt, E.D & Walker, W.D. (1995). A Remote Access Protocol for the X Window System. *Proceedings of the 9th Annual X Technical Conference*, O'Reilly & Associates, Inc.

Edwards, W.K. & Mynatt, E.D. (1994). An Architecture for Transforming Graphical Interfaces. *Proceedings of UIST'94: User Interface Software and Technology Symposium*, 39-47, New York: ACM.

Edwards, W. K. & Rodriguez, T. (1993). Runtime Translation of X Interfaces to Support Visually-Impaired Users. *Proceedings of the 7th Annual X Technical Conference*, 229-238, O'Reilly & Associates, Inc.

Gaver, W. W. (1988). *Everyday listening and auditory icons*. Unpublished doctoral Dissertation, University of California, San Diego.

Gaver, W.W. (1994). Using and Creating Auditory Icons. In G. Kramer (Ed.), *Auditory Display: Sonification, Audification and Auditory Interfaces* (pp. 417-446). SFI Studies in the Sciences of Complexity Proc. Vol. XVIII, Addison-Wesley.

Glinert, E.P. & York, B.W. (1992). Computers and People with Disabilities. *Communication of the ACM*, 35(5), 32-35.

HumanWare, Artic Technologies, ADHOC, & The Reader Project. (1990). Making good decisions on technology: Access solutions for blindness and low vision. *Proceedings of the Closing the Gap Conference*.

Hutchins, E.L., Hollan, J.D. & Norman, D. A. (1986). Direct Manipulation Interfaces: In Norman, D.A. & Draper, S.W(Eds.), *User Centered System Design (*pp. 87-124). Lawrence Erlbaum Associates.

Ludwig, L. L. and Cohen, M. (1991). Multidimensional audio window management. *International Journal of Man-Machine Studies*, 34(3) 319-336.

Ludwig, L. L., Pincever, N. & Cohen, M. (1990). Extending the notion of a window system to audio. *Computer*, August, 66-72.

Ly, E. (1993). *Chatter: A Conversational Telephone Agent*. Master's Thesis, Program in Media Arts and Sciences, MIT.

Mynatt, E. (1994). Designing Auditory Icons. *Proceedings of the Second International Conference of Auditory Display, ICAD '94*, 109-120, Sante Fe Institute.

Mynatt, E. (1995). *Transforming Graphical Interfaces into Auditory Interfaces*. Doctoral Dissertation, Georgia Institute of Technology.

Mynatt, E. & Weber, G. (1994). Nonvisual Presentation of Graphical User Interfaces: Contrasting Two Approaches, *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 166-172, New York: ACM.

Norman, D. A. (1988). *The Psychology of Everyday Things*. New York: Basic Books.

Schmandt, C. (1993). Phoneshell: the Telephone in Computer Terminal. *Proceedings of ACM Multimedia Conference,* 373-382, New York: ACM.

Stevens, R., Brewster, S., Wright, P. C., & Edwards, A. D. N. (1995). Design and Evaluation of an Auditory Glance at Algebra for Blind Readers. *Proceedings of the Second International Conference of Auditory Display, ICAD '94*, 21-30, Sante Fe Institute.

Stifelman, L. J., Arons, B., Schmandt, C. & Hulteen, E. A. (1993). VoiceNotes: A Speech Interface for a Hand-Held Voice Notetaker. *Proceedings of INTERCHI '93*, 179-186, New York: ACM.

Yankelovich, N. (1994) SpeechActs & The Design of Speech Interfaces. *Adjunct Proceedings of the 1994 ACM Conference on Human Factors and Computing Systems*, New York: ACM.

Yankelovich, N., Levow, G., and Marx, M. (1995) Designing Speech Acts: Issues in Speech Interfaces. *Proceedings of CHI '95*, 369-376, New York: ACM.