THE MERCATOR PROJECT
A NON-VISUAL INTERFACE TO THE X WINDOW SYSTEM

W. Keith Edwards, Elizabeth D. Mynatt, and Tom Rodriguez

EDITORIAL NOTE
This is an ascii text version of a Framemaker document. Although the original document contained figures and screenshots which cannot be reproduced in this version, there is sufficient text descriptions of the graphical material so that there should be no net loss of information.

ABSTRACT

This paper describes work to provide mappings between X-based graphical user 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 user interfaces for users who are blind or severely visually impaired. We describe the features of an auditory interface which simulates many of the characteristics of graphical interfaces. We then describe the architecture we have built to model and transform graphical interfaces. We present some of the difficulties encountered in building such a system on top of X. Finally, we conclude with some indications of future work.

Keith Edwards is a Research Assistant at the Georgia Tech Multimedia Computing Group. His research interests focus on computer-supported cooperative work. His email address is keith@cc.gatech.edu.

Beth Mynatt is a Research Scientist at Georgia Tech where she manages the Multimedia Computing Group. Her interests include auditory interfaces and novel interaction techniques. Her email address is beth@cc.gatech.edu.

Tom Rodriguez is a Research Assistant at the Multimedia Computing Group. His interests include window systems and digital video. His email address is jack@cc.gatech.edu.

Introduction

A common design principle for building computer applications is to separate the design and implementation of the application functionality from the application interface. The reasons for this separation are clear. If the application interface, or more precisely, the presentation of the application interface is independent of the application behavior then the interface can be easily modified to suit the needs of a wide range of users.

The graphical user interface is at this time an extremely common vehicle for presenting a human- computer interface. There are many times, however, when a graphical user interface is inappropriate or unusable. One example is when the task requires that the user's visual attention is directed somewhere other than the computer screen. Another example is when the computer user is blind or visually-impaired. Unfortunately, graphical user interfaces, or GUIs, have disenfranchised this portion 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]. The motivation for this legislation is clear. As more organizations move to a standard graphical environment, computer users who are visually-impaired may lose employment. Although this legislation has yet to be tested in the courts, both vendors and consumers of computer equipment and software are beginning to seriously address accessibility concerns.

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 Project, is to provide transparent access to X Windows applications for computer users who are blind or severely visually-impaired [ME92a][ME92b]. In order to achieve this goal, we need to solve two major problems. First, in order to provide transparent access to applications, we need to build a framework which will allow us to monitor, model and translate graphical interfaces of X Window System applications without modifying the applications. Second, given these application models, we need to develop a methodology for translating graphical interfaces into nonvisual interfaces.

In this paper, we describe the steps we have taken to solve these two problems. In the following section, we describe the design for the Mercator interface. We introduce the concept of audio GUIs and the abstract components of auditory interfaces. We also detail some of the techniques we are using to convey a range of interface attribute information via the auditory channel.

Next we describe the architecture we have constructed to provide this interface transparently for X applications. We detail the requirements and goals of the system, the individual components of the architecture, and how those components interoperate to provide a translation of a graphical interface into an auditory interface.

Auditory Interfaces

The primary human-interface 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, we discuss the major design considerations for these Audio GUIs. We then describe the presentation of common interface objects such as buttons, windows, and menus, and detail the navigation paradigm for Mercator interfaces.

Design Considerations

There are several design decisions we had to make when constructing our nonvisual interface. A 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. The user could move a mouse over a screen and hear the objects announced as the cursor touched the object. In this type of system the user 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, the system could 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.

The primary question here is determining what aspects of the graphical interface contribute to the user's model of the application and what aspects of the graphical interface are simply visual artifacts of its graphical presentation. Retaining the user's model of the application interface across presentations is necessary to support collaboration between sighted and non-sighted users. It is also important to remove visual artifacts of the graphical presentation which do not make sense in the auditory domain. For example, scrollbars serve two purposes in graphical interfaces. First they conserve limited screen real estate by presenting only a portion of a list. Second, they provide a mechanism for the user to quickly search the list. The first use of the scrollbar most likely does not need to be conveyed in the auditory interface since there is no limited display real estate. The second use does need to be transferred to the auditory domain. But the movement of the scrollbar (up or down so many items) would need to be based on the auditory presentation of the scrollable object not its visual presentation.

Essentially, we have chosen a compromise between the two extremes outlined above. To ensure compatibility between visual and nonvisual interfaces, we are translating the interface at the level of the interface objects which form the user's model of the application interface. 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.

Another design 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][Edwa89]. 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.

Interface Components

As stated in the last section, the auditory presentation will be based on the user's model of the application interface. This model will be composed of the objects that make up the graphical presentation. The constructs of the user's model can be thought of in terms of common

interface objects such as menus, buttons, and dialog boxes or in terms of the functions afforded by these objects such as "selection from a set" or "containing heterogenous types of information". In X Windows applications, these objects roughly 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.

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. The Mercator Model Manager stores the widget hierarchy for the application interface and any attribute information associated with the widgets. Using a set of rules and widget templates, these widgets are combined to form abstract interface objects.

As with graphical interface components, there is not always a one-to-one