Technology and the Blind Musician

Draft 1999 Peter Elsea elsea@cats.ucsc.edu

A Promise Unfulfilled

The advance of technology is creating major changes in music. The jobs of recording engineer, composer and even performer are undergoing a continual revolution as technology makes the transition from analog to digital systems. In the early days, it appeared that blind musicians would benefit from technical advances: basic tools were becoming affordable, computer systems were beginning to speak, and several companies were formed to provide accessible software.

However, the promise of the early days was only partially realized. Commercial equipment has certainly become affordable and powerful, but the interfaces have become so dependent on vision that blind users are shut out. Computer systems speak better than ever, but applications built around complex graphical interfaces leave them with nothing to say. The access companies are hard pressed to keep up with the constantly changing landscape of mainstream applications and can devote very little attention to the marginal market of blind musicians.

Nonetheless, many blind musicians are finding ways to cope with the technical onslaught, and their strategies, successes and failures provide important lessons for those who would follow in their footsteps and those interested in helping tear down some of the barriers.

Speech in Computer Systems

The development of computer speech should be familiar to readers of this journal; suffice it to say that a variety of strategies make even modest computers capable of fluent, understandable speech in nearly any language. In fact, computer generated speech is a commonplace part of ordinary life, from voice mail systems to video recorders that coach the user through the setup process. However, giving a computer speech and getting it to say the right things are different matters.

It is a reasonable expectation of a blind computer user that a speech equipped computer should be able to read what is appearing on the screen. This was a straightforward problem when text was generated at an ASCII terminal or by the DOS BIOS; a concurrent program known as a screen reader would trap text calls and build a private map of the screen display. It would also trap keyboard input, and read sections of the screen when prompted by special key combinations.

Reading within a windowing system is much more complex. The screen reader must keep models of all windows, and keep track of which windows are visible and which applications own them. The text is no longer ASCII code, but complex fonts which may be customized in unpredictable ways by the applications. Further, the text is becoming only a minor part of the information on the screen. The interface of the word processing application being used to write this document includes 81 graphic objects scattered around the screen. Many of these have alternate appearances indicating active, selectable, non selectable, or inactive. Keep in mind that the inclusion or positioning of most of these elements can be changed either by the program or by the user.

Without going further into the details, I will simply state that blind users find they typically can read only 30% to 80% of what is on the screen, and that the screen reader has to be customized for each application.

In addition, the current economics of the software industry seem to demand constant revisions of the applications and the operating systems. The screen reader companies, all of which are small enterprises, are quite hard pressed to keep up with the mainstream products. Any effectiveness for music applications is more or less accidental, and requires weeks of unpaid labor by the end users to develop custom scripts. It is not unusual for someone to spend months creating a script for a popular program, only to discover he can not share the fruits of his labor with others because the version of the program he worked from is no longer current.

Embedded Systems

In the modern music studio, only one or two of the computers are big boxes with CRTs attached. In fact all but the simplest pieces of gear have embedded microcomputers with elaborate user interfaces. The most recent trends in these interfaces are threatening to further reduce the ability of blind engineers to work in the studio.

As an example, I'll compare a popular analog mixing console built last year with its new digital successor. The old console has faders dedicated to specific functions, knobs with limited range of rotation and tactile markings, and pushbuttons that stay down to route signals. With the addition of a relatively inexpensive audible level meter, this desk is usable by the blind, and in fact is being used by many blind engineers across the country.

The new model has faders whose function depends on which of four modes the desk is in. (The mode is indicated by a light) The knobs have no markings of any kind and rotate freely with the value of the associated parameter indicated by lights in the panel. The functions of these knobs are also variable, with a dozen possible meanings. Routing is accomplished by pressing buttons in a specific order, and the same buttons are used for many other functions. The state of the board is so unclear from the controls that the manufacturers have included a CRT display with a picture of a standard mixing console to indicate status. The computer system underlying the unit is proprietary, so only the manufacturer is in a position to add speech. [Potential graphic- Mackie 8 bus and D8b side by side- can be taken from ad copy]

I'm not going to claim that the new model is unusable by a blind engineer, but the difficulties are obviously significant. This mixer is not an isolated example, but merely the latest of a trend. Devices with controls and operating protocols that are readily manageable without sight are becoming quite rare.

The Transcription Problem.

One problem common to all blind musicians that would seem to be easily solved by computers is the issue of music transcription. In the past, blind musicians would use Braille notation for their own study and rely on sighted assistants to transcribe music to and from print as needed. It would seem almost trivial to write a program that would afford Braille formatted entry and produce usable standard notation, at least at the undergraduate music theory level, but no such program seems to exist. The only commercially available notation program that allows direct keyboard entry is SCORE, and that is expensive, difficult to learn, and requires sighted proofing. Some other common programs allow entry from a MIDI keyboard, but sighted assistance is still required for proofing and printing.

There is a program, GOODFEEL, that automatically translates LIME format scores into Braille, but LIME itself is not accessible, and translation the other way is not yet available.

Working Musicians: Some Case Histories

To determine the software requirements of blind musicians, I have interviewed a series of active professionals, discussing their experiences, the equipment they are using, the problems they are finding, and their concerns for the future.

I found blind people working in all branches of the audio industry, including radio stations, recording studios and tech support for synthesizer manufacturers. Many others have private facilities of various levels of sophistication and either provide contract services or work on their own projects. While the common concept that blind people have innately superior hearing is a myth, it is certainly true that many blind musicians have honed their aural abilities to a degree rare in sighted people, and that when it comes to critical listening they are at least on an equal footing with others.

The following short sketches will give an idea of the types of work blind people are doing, and how the changing technical scene is threatening their opportunities.

Recording Engineer

RW records radio jingles and demos for aspiring performers. He works in a manner typical of most recording engineers: he does not have a studio of his own, but contracts time in four or five different facilities. For more than twenty years he has been able to do this by carrying an audible VU meter from job to job, working the board and tape decks by touch. Recently, the better studios in his area have begun modernizing, converting to digital equipment and computer based hard disk recording systems. These systems offer many advantages in quality and efficiency, but the most popular do not work well with screen readers. The problems with digital consoles have been discussed above, and even the ubiquitous ADAT raises difficulties, because there is no reliable way to know which tracks are armed for recording. RW is now forced to work mostly in second tier rooms that are less attractive to clients.

Radio Reporter

VP is a feature reporter for a public radio station. She typically gathers interviews and audio material on a portable cassette recorder, transfers the material to reel to reel tape and edits it with a razor blade. She does not have her own reel to reel deck, but depends on the aging units at the station. Eventually there will be none available, as the station managers have announced plans to convert to computer based editing.

Record Producer

MM is a well known and successful composer with a modest studio in an urban apartment, where he produces advertising jingles and backgrounds for television. By owning his equipment, he can ensure a place to work, but he still must compete for customers within a tight market. He would like to install a digital workstation, but cannot find a system that is appropriate and usable. There are several on the market, but each has some fatal flaw. For instance, in the most popular application, the record function cannot be armed with anything but the mouse.

MIDI Performer

GK is a performer and composer who works primarily in popular music. He has a project studio where he uses an integrated MIDI and audio sequencer to create backing tracks for his live act. He also produces tracks and arrangements for other musicians. He has spent many hours scripting the screen reader to work with the sequencer, and has achieved nearly 90% functionality. Although he is very good at making the system do what he wants, the process is appallingly slow. In a recent demonstration, a simple task that a sighted user could accomplish with five mouse clicks took nearly twenty minutes. He performs with a complex suite of MIDI synthesizers, which he programs with the only speech friendly computer based editor, a Windows 3 application that supports few modern instruments.

MIDI Composer and Publisher

VE also owns a private studio, where she produces CDs of her songs with synthesizer accompaniment. She uses an obsolete DOS based sequencer that has limited functions and low reliability, but better than average accessibility. She records on digital modular multitracks using custom written software for control of the decks and mixer. Creating custom software is a reasonable approach for simple tasks, but not cost effective for elaborate applications.

Graduate Student

SO is finishing a doctorate in computer music at one of the world's most prestigious research facilities. Although she is surrounded by the literal cutting edge of technology, she still utilizes startlingly primitive apparatus. She uses a dos-based terminal program to access a unix-based mainframe for various music applications which she operates from the command line.

Her school offers a multitude of sophisticated tools for synthesizing sound, but she finds the quickest way to generate her own sound synthesis examples is to write scripts for Matlab that use its signal processing toolboxes. Like VE, SO uses many custom-designed pieces of software, most of which she writes herself. Despite the current proliferation of Integrated Development Environments for writing software, she uses a simple text editor and makefiles to develop and debug code. She is often required to work on colaborative projects where code is shared between many developers and a specific compiler must be used. Recently, she had to withdraw from a project because the compiler she was using was updated and the update failed to comply with acessibility standards and thus could not be used with a screen reader.

Interviews with other blind musicians of various levels of accomplishment show the same trends. Reliance on old, limited but dependable hardware and software, or frustration with the current generation of products are the universal complaints.

The Specialized vs. General Dilemma

One of the underlying conundrums of the situation is a basic point of philosophy: would effort be best expended creating programs that directly address the needs of the blind, or is it more efficient in the long term to make existing software, in all its profusion, accessible by modifications to the operating system and/or helper applications.

Generalized Accessibility

The commercial software industry is clearly favoring the latter approach. There are two or three companies that produce software expressly for Braille conversion; those companies have discovered that the best market is the blind services agencies where (sighted) operators convert printed text or music to Braille. The fact that such agencies work under government or charitable support allows the companies to charge enough to keep going. Some of their programs are also usable by the blind, but many blind users are discouraged by the cost of the software; the purchasing power of blind musicians is very low compared to government budgets.

There are more companies that produce general accessibility software, screen readers and Braille display systems with the goal of allowing blind users access to mainstream programs such as word processors, email, and databases. The philosophy is that an employer would prefer to have all workers use the same software, and that the only concession necessary to a blind employee is the installation of a screen reader or Braille display. The cost of such helper applications is again high, but a single purchase (in theory) enables the user to take advantage of a host of low cost applications written for the general market.

Several of these companies are successful, and have done a remarkable job of keeping up with the constant flood of revisions in standard applications and operating systems. Unfortunately, music applications can get little attention by the reader companies because the market is very small, and the problems presented by music applications are especially difficult.

Accessibility Roadblocks

The problem the screen reader companies struggle with the most is the detection and interpretation of graphic elements. Any time a programmer invents a new way to put a button on the screen, he has probably come up with another way to circumvent the screen readers. If the program uses a non standard cursor, the blind user will not know where it is.

The operational design of the program can also be an obstacle. If there is no way to accomplish a task besides clicking on an object with the mouse, it will take a blind user a very long time to find the object, route the mouse pointer to it and click. If that object is invisible to the screen reader the task cannot be done.

With some programs, the nature of the process presents inherent difficulties. It is hard to imagine how a photo editing application could be made accessible, for instance. This is partially true of music scoring programs perhaps, but there is no reason a MIDI sequencer or audio editor should be inaccessible. The graphic features of these programs exist as an aid to the sighted user, to allow them a quick way of finding silences in a recording, or to show subtle timing relationships between notes. It is only when these visual aids become the only way of operating the program that the blind user is shut out.

How To Make Your Software Accessible

There are several sets of guidelines for writing accessible programs published on line [Microsoft, AFB], and these should be consulted by all application authors before starting program design. The issues to watch for include such commonsense principles as using standard controls and cursors, providing keyboard versions of all actions, and using text labels, as well as some subtle issues such as avoiding timeout situations and allowing the user to customize colors. Accessibility has to extend to program installation and configuration as well as routine operation.

Programmers working in Windows should consult the Microsoft developers web site for specifications for Active Accessibility. This is a system Microsoft is putting in place to enhance the capabilities of screen readers. It is not a panacea, and is not yet trouble free, but promises to improve the situation considerably in the near future.

Accessibility for Embedded Systems

Despite the fad for talking appliances in the late 80's, there are very few devices with speech that are really useful to blind people. That's because the speech was seen as a marketing device and not an access method. The result was items like clocks that speak the time when a button is pushed, but not when they are being set.

Introduction of music and audio equipment with internal speech is very unlikely, but the designers of such systems can take certain steps that would ensure some level of usability by blind customers.

Provide alternate display output

There was some attempt a few years ago in the UK to produce a device that would read the common 80 cell LCD display. That effort may still come to fruition just as most manufacturers convert to bitmapped plasma screens. A few MIDI instruments (such as the Emu Proteus 2000) feature a system exclusive command that produces a screen dump. If the dump is in ASCII, a simple program gives the blind user access to all of the instrument's features. If the dump is a bitmap, an access program has to be customized for each screen on each instrument, but it is feasible. There is a movement underway to request the

MMA set a standard for such dumps, and providing them is probably the most cost effective way of ensuring accessibility to hardware.

Provide Limited or Resetable Controls

The endlessly spinning data dial is the bane of blind users. A normal knob or slider has defined values at the end points and is easily used by anybody. There are some instruments that provide a way of getting to a known value, either by a fast spin of the knob or by pressing two buttons at once.

Provide Limited Or Homing Cursors

Cursors within data screens are a problem if they appear in unpredictable locations. The best strategy is have one parameter per screen, but a cursor that always is in the upper left when the screen opens or that can be sent there by hitting the enter key is nearly as good.

Provide Complete System Exclusive Implementation and Documentation

System exclusive access to a device's settings is attractive to all users. It is expensive to implement, but does have a positive impact on sales when third parties write software editors and remote controllers. Inclusion of sysex is only half the process. The documentation must be available to any who want to try writing assistive software.

Specialized Programs

There seem to be no music programs in general circulation that assume that the user is blind. Some individual musicians have commissioned software for their private use, and are willing to share it on an "as is" basis, but do not have the resources necessary to provide fully debugged applications with appropriate technical support. Blind user specific programs would probably have to be funded by grants, possibly written as graduate projects. Such projects could possibly take one of the following forms:

Basic MIDI Sequencing

A basic MIDI sequencing program would look very much like an old fashioned command line application. All program responses would appear in a simple console view for the screen reader to pick up. The command set can be fairly sparse, and editing could be done in a list of some kind. The user would specify what appears in the list and use the arrow keys to step through it. Timing can be specified in the common system of measures, beats and ticks, but the placing of barlines should be done in an intelligent rather than a metrically rigid manner. The user should be able to send immediate program changes as well as enter them into the tracks. This program could work directly with standard MIDI files.

Advanced MIDI Sequencing

An advanced sequencer would add searching and grouping capabilities, among other things. The user would be able to search for chords and melodic patterns, and construct complex events such as controller movements with simple commands. Tempo tracks, subsequences, and numbered looping would be useful features. Editing would still be list based, but should include actions like scaling by percent. Some capability for dealing with system exclusive messages would be desirable. This type of program would probably require a custom file format to include markers and complex action specifiers.

Audio Recording and Editing

A basic recording and editing program should support standard file formats, importing from audio CDs, and simultaneous playback and record. The basic editing paradigm should be the placing of markers during playback. The user would then be able to audition the track before and after each marker, perhaps hearing a beep at the marker location. They could then select the range between any two markers for editing. Find silence and find clips commands should be included, as well as amplitude calculation for any region. Non destructive playlist playback would be a nice touch.

Multitrack Audio Recording and Mixdown

An advanced editing application would add the ability to overlay tracks and script mixdowns as well as process the audio with standardized plugins. Eventually this should be combined with the advanced MIDI sequencer to create a comprehensive program.

Production of Print Music

Working from Braille music or DARMS type entry, a user should be able to create correctly formatted readable printed music. It would be acceptable to simply produce NIF files for printout by another application. This is not necessarily publication quality output, just something to get a student through music theory courses.

Control of standard studio equipment and instruments

This is a grab bag of modest to complex applications. There are many devices that have extensive system exclusive implementations that could be made accessible by very simple programs. These can be as simple as a compiled MAX patcher such as the author's EOSTalk, that does no more than access and speak the disk catalog features of Emu brand samplers. This single page of code makes it possible for blind owners of these instruments to use hundreds of sample discs that are available on CDs. A more ambitious example is ADATTalk, a 450 kilobyte application that gives control of any MIDI Machine Control compatible recording device.

What to Say in a Blind Friendly Application

Designing an interface that targets blind users requires just as much thought and testing as creating a GUI. The fundamental tasks for the interface are the same: provide a usable view of the data, display program status, and provide direct means of manipulating either. However, the different mediums of communication require different metaphors. Whereas a word processor usually emulates the action of a typewriter, a blind user is better served by the paradigm of the secretary taking dictation.

The utility of the GUI to a sighted user is based on the very high bandwidth of a visual display. The screen constantly presents dozens of chunks of unrelated data which the user can immediately access by simply looking at objects on the screen. The screen display can also give an overview of the data, affording the opportunity make large jumps from place to place. The mouse or other pointer is the user's means of telling the computer where he has focused his attention and affords a streamlined way of issuing certain commands.

Speech presentation of information does not have this bandwidth and focusability— attention is restricted to what is currently being said, and only one thing can be said at a time. A speaking interface should provide only the information the user requests and the format of the presentation should vary according to the context of the request. The metaphor of the cluttered desktop must give way to that of an agent, an entity that has immediate access to all of the data and will read aloud whatever is desired.

Data Readback

No matter what the nature of the data in a document, it must be possible to navigate through the data in a variety of ways. In the most common model, there are two cursors, one for action, and one for reading. Either can be moved incrementally or in large jumps, and each can be moved to the other. Reading occurs in blocks of varying size, in a manner appropriate to the task at hand. In a text processor, commands for reading back the last word, sentence or paragraph are usually more useful than reading whatever happens to fit on one line, but line by line reading is essential when formatting for final publication. Reading must be stoppable at any point. It should be possible to skim through the data by reading the first few words of each paragraph, or to spell a single word. The location of the action cursor is announced if it is encountered, and selected data can be indicated by a change in voice, or simply by interjecting 'begin selection'.

There must be a flexible system of bookmarks. These can be numbered by default, but should be namable. The action cursor can be moved to a mark by a *goto* command and the reading cursor by *readfrom*. Selection of data for editing can be made from mark to mark or from action cursor to reading cursor.

The manner of reading the data is dependent on the type of application. It's pretty clear what should be read with a text processor, but a music program should have varying levels of verbosity. The typical sequencer presentation of onset time, pitch, velocity and duration (in ticks) is tiresome to work with. All the user usually needs to hear is pitch and duration, or maybe just pitch, enough to identify the note as he finds his way through the score. The duration should be in familiar terms of quarters, eighths and so forth. A chord should be read from bottom up regardless of the actual attack order of the notes. When the user wants to examine some notes more closely, then detailed information such as velocity, offset from the beat and precise duration can be read.

The verbosity can vary with context. There is a convention in Braille that the octave number is placed before the pitch name after a jump of a fourth or more. This is useful in reading, as is the convention of dropping durations when they stay the same on successive notes. When stepping through a measure of eighth notes, the user may hear '3 B flat eighth note, A eighth, C, D, E flat'. Since speech synthesizers can be difficult to understand, phonetic spelling (able, baker etc.) should be available when needed. Extensive reading preferences must be available; for instance, the user should be able to choose either the Braille or the MIDI octave numbering system.

Program Status

If the user wishes to be reminded of some setting of the program, he should post a query that is answered in a conversational manner, such as 'The paragraph style is normal'. Responses can range from terse to verbose according to how often they are used and how ambiguous the reply would be. For instance, a query for the font at the cursor might just say 'Geneva', but a query for the default font should say 'The default font is Geneva'. Font size should not be quite so terse. The word 'Geneva' is clearly a font, but the number 12 could refer to a variety of things.

Commands

The commands that are used most often must be the most efficient- a key combination is greatly preferable to a command line entry, and a command with a couple of arguments is preferable to a dialog. A command with time value (such as "stop reading") must be a key combination, and any operation that requires no more data should be. You can be quite clever in key commands- for instance hitting the "read previous word" key three times might read the previous three words. The keys chosen for commands should be selected for location rather than mnemonics. It is well and good to reserve 'S' for save, but you can’t go far down that road before encountering irresolvable conflicts. The commands issued most often should be the easiest to type.

In a word processor, when any direct typing has meaning, command line operations should be avoided, but they could be useful in a MIDI sequencer. If the command set is so rich that all of the comfortable key combinations are in use, a command window becomes attractive, especially for operations that take a single argument, such as goto or find. When the command window is open, there must be access to help, and the user must be able to review and edit the command string.

All command actions must be verbally acknowledged with enough detail that the user is confident the proper action has occurred. When a deletion occurs, the data should not just go away; the program should announce something like "Four measures deleted".

A dialog in a blind user's application is exactly that: when the dialog is triggered, the computer starts talking and the user responds. The first announcement should be the name of the dialog, followed by the first item and its current value. Full editing capability should be available in the value fields. Although it is not necessary to allow the user to move randomly around the items of a dialog, it must be possible to revisit items or repeat them, and escape harmlessly from any point. Tab for forward, shift tab for reverse and escape to dismiss the dialog are widely used key combinations. If the enter key is used to complete individual entries, the OK button or equivalent must be in the tab cycle. Dialogs must be scripted carefully and tested thoroughly. They must be designed with a particular task in mind and contain all of the information needed to do that task, even if that means some parameters appear in more than one dialog.

Help

Because manuals in Braille or on cassette are difficult to navigate for quick reference, any application targeted for blind users must have better than average help features. Everything that the program is capable of should be documented in the help files. Help could be in a separate file rather than directly incorporated into the program, but if so it should be a plain text document. PDF or other formatted text files are not very readable.

The ideal help system is somewhat context sensitive, in that it opens in an appropriate place in a complete help space. Since the user probably does not know the documented name for the information needed, a list of topics is more useful than a find feature, but both must be provided. The list should be organized as in a tutorial, with operations documented in the order the user will encounter them. If the list is quite long, a nested tree structure should be used, grouping related topics under larger headings. When a topic is selected, the user then hears a concise but complete description of related commands and operations. A hypertext structure is desirable, but the ability to return instantly to the topic list is essential. The user should be able to add his own annotations to the help file, including inserting new topics into the contents tree.

Defaults and Templates

A designer should be aware that many blind users will not have certain concepts sighted people take for granted. These may include a sense of spatial relationships and knowledge of some common notational conventions. Therefore, anything that produces printed output should have default settings that give acceptable results without tweaking. In a scoring program, all conventions of stemming and beaming should be automatic, and there must be a collision detection mechanism for slurs and other symbols.

When there are several possibilities for page layout, appropriate templates should be provided, along with the ability to convert an existing document into a template.

Testing

The need and methods for user testing is well stated throughout the literature, and it should go without saying that an application for blind users should be tested with blind users. As an interim test, the programmer may throw a blanket over the screen and try the program out, but only a real user can give the feedback necessary for validating the design. They should be able to install the application, perform all operations, and produce finished output with no assistance.

Conclusions

For a variety of reasons, blind musicians are not well served by the current state of computer software. Foremost among these is the fact that the trend toward rich graphical displays limits access to data and control for users of screen reading software. Progress toward a solution may take two paths; developers of general market programs can follow a set of guidelines to reduce the obstacles to use by the blind, or the needs of the blind musician can be addressed directly by software especially fitted to their use.

[sidebar: How to set up an accessible computer workstation.

Many readers of this journal will eventually be asked to install an accessible computer workstation for a blind music student. It should be understood that such an installation will not be a complete replacement for sighted assistants, particularly if the student has little experience with computer use.

The underlying CPU should be a reasonably powerful Windows machine with a hardware speech synthesizer and compatible screen reader as well as a scanner and Braille printer. A good word processor and OCR scanning software designed for blind users will round out the basic system. This package should be provided and configured by a consultant who specializes in such systems, because the options are constantly changing and there are many hidden incompatibilities.

Music programs are constantly changing in features and accessibility, and you should research the field before making a final choice, but the best options as of this writing are:

Goodfeel by Dancing Dots software, for conversion of music (Lime format or standard Midi Files) to Braille.

Cakewalk by Twelve Tone Technologies, for MIDI operations and possibly recording.

Finale to convert Standard MIDI files to print (Sighted assistance will be required.)

Cool Edit for audio recording.

An Alesis QS6 keyboard for music entry and MIDI playback.

The best way to research the current options is to ask blind musicians. These can be found through on line lists such as MIDI Mag, and through associations such as the American Council of the Blind.]

References:

Microsoft Guidelines for Program Accessibility; published on line, http://www.microsoft.com/enable/default.htm , Microsoft Corporation, Redmond, WA, U.S.A.

Creating Applications Accessible to People Who Are Visually Impaired; Published on line, http://www.afb.org/technology/accessapp.html American Foundation for the Blind, New York, NY, USA