ERIC BOHLMAN

DEVELOPMENT OF A POWERFUL AND AFFORDABLE SCREEN READER

Tinytalk is a powerful, memory-efficient, and inexpensive screen reader program for blind or otherwise print-handicapped users of IBM-compatible computers. It was developed to meet needs expressed by blind consumers, who participated in all stages of the design and implementation process and played a key role in providing recommendations and feedback. The program is marketed as shareware, allowing potential users to determine whether it meets their needs before making a commitment to purchase it. The development of Tinytalk illustrates how small private-sector developers can produce high-quality and inexpensive adaptive software by involving consumers in the development process, avoiding stereotypical preconceptions about the nature of people with disabilities and incrementally improving existing technology.

INTRODUCTION

A screen reader is a program that lets a blind or dyslexic person use a speech synthesizer to access the text shown on a computer's video display. It stays in the background while the user runs application software such as a word processor or database manager. Screen readers are adaptive software (system programs that modify a computer's user interface to enable a user with a disability to use other programs) rather than therapeutic or prosthetic software (application programs that enable the user to perform tasks that an able-bodied person could perform without a computer).

A screen reader lets a blind user select and read the sections of the display he or she wants to hear. For example, sighted users stop reading a menu when they come to the item they want; blind users use the screen reader to silence the synthesizer's output after they have heard it. Screen-reader users consider the ability to filter out irrelevant information as important as the ability to hear relevant information.

Screen readers would be unnecessary if all application software programs were specifically designed to use speech output. In the early days of speech access, publishers sold application programs specially designed for blind users, but this approach was unsatisfactory because it did not give blind users access to most of the programs that sighted people could use. Since blind people use computers for the same purposes as sighted users, not as therapy or compensation for their blindness, they need access to the entire range of available software.

Computers represent information as electrical signals that nobody can directly perceive. Sighted users need technology (a video monitor) to translate this information into a perceptible form. A screen reader/voice synthesizer simply performs the same function for a blind user.

A BRIEF HISTORY OF SCREEN READERS

Video displays, in the form of serially interfaced terminals, were first used with computers in the early 1960s, but did not become common until the early 1970s. Early terminals used a printer rather than a video screen for output; consequently, application programs output a one-dimensional stream of scrolling text that could be fed into a speech synthesizer connected between the computer and the terminal. The only problem was that the user had no way to reread text that had already been spoken; this shortcoming led to the development of the talking terminal, a video terminal equipped with a voice synthesizer and some extra control logic that let the user move an audio cursor around the screen display and hear the text.[1]

The first personal computers developed in the late 1970s used serial terminals, but manufacturers soon began building keyboard and video-display logic into the computers themselves. Those advances changed the way programs produced output; instead of writing a stream of characters to an output device, they wrote the characters into memory locations corresponding to the rows and columns of the display. Since the computer could now generate a video signal rather than a stream of character codes, no "tap-in" point was available to insert a speech synthesizer. Programmers started taking advantage of the two-dimensional nature of the display by writing full-screen rather than line-oriented programs.

One way to access the displayed characters was to add extra circuitry to shadow the display memory and provide talking-terminal functions. The problem with that approach was the expense of the extra circuitry and the lack of standardized interfaces for equipment that needed access to internal memory. Manufacturers would need to produce separate units for each computer model, and installation would require special modifications to the computer. A few manufacturers developed computers designed specifically for blind users, but they were unsatisfactory for the same reason that special talking software was unsatisfactory: blind users could not use mainstream computers.

The second approach was to extend the computer's operating system with code to monitor the display memory and send the text to the speech synthesizer. In the late 1970s and early 1980s, Maggs and others[2] developed screen-reader programs for computers such as the Apple II, the Radio Shack TRS-80, and the many computers that used CP/M operating systems that existed at the time. Most of them offered few facilities because they had to share the very limited memory available at the time with application software. When I developed a screen-reader program for the Commodore 64 (the 64Reader, marketed by OMS Development in the mid-1980s), the entire code had to fit into 8 kilobytes.

The introduction of the IBM personal computer in 1982 removed many of the constraints that had kept screen readers in a primitive state by providing a standard interface for connecting external devices, a standard operating system, and the possibility of installing much more memory. By the late 1980s, several screen-reader programs were commercially available; examples included Enable Reader, Soft Vert, and the Enhanced PC Talking Program.

HOW TINYTALK CAME ABOUT

The screen readers of the late 1980s still fell short of providing blind users appropriate access to screen displays. The trend toward full-screen applications had continued, and programmers were using attributes such as color and reverse video to convey information. Modern application programs were using pop-up windows, light-bar menus, and fill-in-the-form data-entry fields, but screen readers could not automatically read these features; users had to stop what they were doing and enter a review mode to navigate around the screen and find out what had been displayed. As more memory became available, both screen readers and application programs got bigger, and some large application programs could not be used with speech.

Screen-reader suppliers often acted as if third parties such as rehabilitation and special education agencies, rather than blind computer users, were their ultimate customers. Their products often seemed to be designed to impress sighted rehabilitation counselors rather than to be of use to blind computer users, and their marketing efforts were directed at administrators who often had little computer knowledge. That focus kept them from discovering what features the ultimate users actually wanted and needed. The assumption that third parties would pay for the software kept prices too high for individual users; although computer prices were dropping rapidly, screen readers typically cost more than $400 per copy, and the combination of screen reader and speech synthesizer often cost more than the computer itself.

By late 1989, many blind computer users who knew of my earlier work with the Commodore 64 were urging me to develop a screen reader for IBM-compatible computers. Although I was hesitant to do so because of the number of screen readers already on the market, conversations with these users and messages posted on computer bulletin boards convinced me that an unserved market still existed for a powerful, low-cost, and memory-efficient screen reader. I learned that many users were struggling with outdated software, running illegal copies of commercial screen readers, or trying to make do with limited demonstration versions of the newer programs. After a little more prodding (including a donation of an Echo GP voice synthesizer by a particularly impatient user), I began to develop Tinytalk.

I first concentrated on implementing a minimal set of screen-reader features using as little memory as possible. As the project gained momentum and positive comments were received from early testers, all of whom were experienced blind computer users, I started adding experimental features such as automatic monitoring of multiple areas of the screen (windows), ways to specify not only when to read a window but how to read it (e.g., reading only text in a certain color or only lines that scrolled into the area), and reading field prompts and values as the user moved through a data-entry screen. Throughout this process, I was in constant contact with the testers, who kept me on the right track whenever I introduced a feature that was awkward to use or omitted some function that they needed. I broadened the base of testers by making test copies of Tinytalk available for the asking on computer bulletin boards (the program was presented as-is as an experimental program with no guarantees of support).

By late 1990, it was apparent that Tinytalk would be commercially viable, and I started developing versions for synthesizers other than the Echo. I set a target of mid-1991 for release and decided to market Tinytalk using the shareware method, which would avoid large up-front advertising and distribution costs on my end and encourage potential customers to try the program. Tinytalk became commercially available in April 1991 and continues to evolve.

THE FEATURES OF TINYTALK

Tinytalk is a memory-resident program that uses less than 27 kilobytes of computer memory. A separate version of the program is run for each voice synthesizer; this arrangement keeps memory usage to a minimum because the program does not have to store command tables for synthesizers not being used. Tinytalk provides the user with all of the standard features that screen-reader users have come to expect, as well as advanced features that let the user make application programs speak as naturally as possible.

Tinytalk automatically reads teletype-style output written through the operating system, echoes keystrokes as either words or letters according to the user's preference, and provides a review/control mode where the user can freeze the application program, read selected portions of the screen, and change synthesizer settings such as speed and pitch. Since most application programs write directly to screen memory rather than through the operating system, Tinytalk lets the user set up areas of the screen that will be read whenever they change.

Tinytalk was one of the first screen readers to search the screen for box-enclosed pop-up windows and read them as they appear, and can track light-bar menus even if the application program does not move the actual display cursor. The user can specify how much text to read

when using the arrow keys to move the cursor in the application program; options include reading whole lines, characters, words, or sections of columns. Tinytalk stores up to thirty configurations (lists of operating modes, synthesizer parameters, and window definitions) in memory at one time, and switches to the appropriate one when the user runs an application program. The user can have separate configurations for each screen within an application program; Tinytalk switches between the screens on the basis of the presence of identifying text.

Tinytalk has "hot keys" that let the user review parts of the screen without having to go into review mode; these functions can be assigned to whatever keystroke combinations the user prefers. The user can also assign labels that are spoken whenever a certain key is pressed, and can save and restore configurations to and from disk files.

Tinytalk supports most commercially available voice synthesizers, including some very inexpensive ones. It can even generate speech (albeit of very low quality) through the computer's internal speaker. Tinytalk was written using Borland International's Turbo C++ compiler and TASM assembler.

THE SHAREWARE MARKETING METHOD

Tinytalk is marketed as shareware by OMS Development; a user can evaluate a fully functional copy of the program before committing to buy it. The user is legally obligated to purchase (register) the program for $75 if he or she decides to continue using it after evaluation; this process is handled on the honor system. The evaluation copy is not crippled in any way; the only difference from the registered copy is that it makes the user listen to a thirty-second commercial when it is loaded.

Although the shareware method has become a popular way of marketing all kinds of mainstream software, it has seldom been used for adaptive or other special needs programs. We decided to use it as an effective way to market Tinytalk directly to end users without incurring large up-front expenses for packaging materials, advertisements, presentations at conventions, or dealer promotions (these costs are often cited to explain why adaptive software has been so expensive). Customers also benefit because they know that when they commit to buy Tinytalk they will be getting a screen reader that meets their needs, since they have already been using it. The user can pass the evaluation copy along to others, so we get the benefit of word-of-mouth recommendations for our software.

THE FUTURE OF TINYTALK

By its nature, a screen reader is never finished; it always has to catch up with new developments in application programs and operating systems. At OMS Development, we are continuing to develop Tinytalk by adding new features, improving old ones, and supporting new voice synthesizers as they come on the market. We are committed to enabling Tinytalk to use the new multimedia sound devices for speech output; as mass-produced consumer products rather than special-needs items, they are widely available at low prices. Tinytalk already supports two such devices, the SoundBlaster from Creative Labs and the Speech Thing from Covox.

We intend to expand the market for Tinytalk beyond blind users. Many dyslexic users, for example, could use its screen-reading facilities. One person who has difficulty telling when she has applied enough pressure to a key has used Tinytalk's keystroke echo facility; a $400, 100-kilobyte program would be overkill for this kind of application, but Tinytalk's low price and compactness make it feasible. We also intend to develop screen readers that can work in graphics-mode operating environments such as Microsoft Windows, which have only recently become accessible by speech.

BARRIERS TO THE DEVELOPMENT OF ADAPTIVE SOFTWARE

I developed Tinytalk to fill a market void created by the scarce, expensive, and unsophisticated nature of much of the screen-reading software available at the time. Adaptive software in general has had many of these problems, and I believe that most of them have been caused by attitudinal barriers rather than by technological challenges. One such barrier is the view that the lives of the disabled center around their disabilities. This belief has led software developers to regard adaptive software as being therapeutic or life-supporting, needing to be customized for each user on the basis of diagnosis or etiology.

This model has distorted the marketplace for adaptive software by fostering the notion that it can be developed and marketed only by special companies that must recover their start-up and operating expenses from a small number of customers. The result is that individual users cannot afford the software, so the companies are forced to deal mainly with institutional customers. In doing so, companies incur additional costs and end up basing product designs on what professionals think their clients need rather than on the functional needs of the end users.

Under the resulting system, potential users often need to obtain funding from third parties such as governmental agencies or charitable organizations to purchase adaptive software. These agencies usually can purchase only programs directly related to immediate and narrowly defined vocational or educational goals, and even then funding is difficult to obtain. Instead of the user evaluating possible tools and purchasing the one that best fits his or her needs, the agencies evaluate the user and attempt to prescribe a treatment. Thus, the attitude that adaptive software is somehow special discourages the development of affordable standard-product solutions to common problems and cuts the real customer out of the product development process.

A second barrier to the development of adaptive technology is the belief that progress consists only of major breakthroughs rather than continuous, small improvements of existing technology. Although this attitude affects all technological development, it has a particular impact on adaptive technology because the field is perceived as esoteric. Small companies and individual de-

velopers often believe that they do not have the resources to develop computer access tools, and that only heavily funded research groups in large corporations and universities can achieve progress in computer access.

Research projects often focus on the technology rather than the functional needs of the users, resulting in programs and devices that are technically elegant but cannot be practically produced at prices users can afford. They frequently concentrate on areas that translate into marginal product features; for example, many of the developers of early reading machines put more effort toward the marginal goal of making the speech output sound as human as possible than they put toward the essential goal of getting reliable character recognition (personal communication with Harvey Lauer, 1985).

The third barrier is the lack of dissemination of information on adaptive technology. Consumers frequently do not know what technology is available, because no system is in place for getting the information to them. Companies develop products that merely duplicate existing ones with no improvement because they are not aware of what others have done and do not find out what the real customers want.

A CUSTOMER-DRIVEN STRATEGY FOR DEVELOPING QUALITY ADAPTIVE SOFTWARE

An alternative to the medical/institutional model is the message of the disability-rights movement: a person with a disability is first of all a person, one who is more like than unlike other people. This view leads to the idea that adaptive software is more like than unlike other software: it should be developed by the same kind of companies that develop other software, marketed like general software, and priced so that customers can afford to buy it. Consumers should be routinely involved in all phases of product development, just as they increasingly are when other products are developed. In short, since people with disabilities are people first, products used by people with disabilities should be products first.

Such a view of adaptive software should encourage software producers to include adaptive programs in their product lines, allowing the cost of administration and development tools to be amortized over a larger market and resulting in prices that customers can afford without outside funding. If developers see adaptive programs as commodity products rather than custom engineering efforts, they will be more likely to produce—rather than hundred-bladed Swiss Army knives—tools that do one job, do it well, and work in conjunction with other tools.

Entrepreneurs without the resources to develop expensive new technology can make significant contributions to computer access by refining and improving existing technology. Reducing the memory usage of screen readers, designing a compact and inexpensive track-ball–controlled keyboard emulator, developing text-magnification software for popular word processors, and writing a word predictor that works well with other companies' alternative input devices are all examples of incremental improvements that small software developers have accomplished without large financial resources. These developments have enhanced computer access for people with disabilities as much as many highly funded research projects have.

Computerized communication networks such as bulletin board systems and commercial on-line services can play an important role in breaking down information barriers and bringing developers, consumers, and third parties together. Consumers can share information about what works well and what does not, and can find out what is available. Developers can quickly learn what current or potential customers want, and counselors, educators, and other third parties can keep up-to-date on the latest advances.

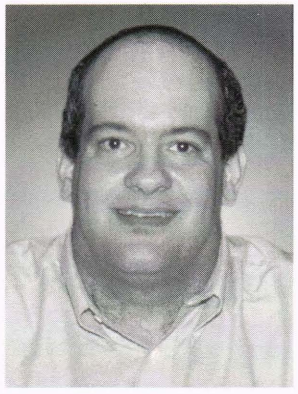
Tinytalk has been successful because its development followed a customer-driven strategy. At OMS Development, we can profitably sell it for $75 because adaptive software is not our only business; our capital expenses are spread over a broader customer base. Tinytalk is not bloated with features such as alarm clocks or phone dialers, because it works well with existing general-purpose programs that can perform these functions. By building on and improving existing concepts rather than reinventing the wheel, we were able to develop Tinytalk within our means. We hope that other small software firms will realize that adaptive software development is not as arcane as is commonly believed and will produce simple, low-cost solutions to common computer access needs if the proper strategy is used.

REFERENCES

[1]Stoffel, D., "Talking Terminals," *Byte* **7**(9), 218–227 (Sep 1982).
[2]Maggs, P. B., "Adapting Personal Computers for Blind and Speech-Handicapped Users," *Johns Hopkins APL Tech. Dig.* **3**(3), 258–261 (1982).

ACKNOWLEDGMENTS: Most of the credit for the success of Tinytalk goes to the users who requested its development and put in the time and effort to test and evaluate the many interim versions that I developed. I also wish to thank the Committee on Personal Computers and the Handicapped (COPH-2) for kindling my interest in adaptive technology.

THE AUTHOR



ERIC BOHLMAN is the owner of OMS Development, a software development firm based in Wilmette, Illinois. He received his B.A. in mathematics and computer science from Knox College in 1980. Mr. Bohlman is a member of the Committee on Personal Computers and the Handicapped and administered its national computer bulletin board service for several years.