Timelines provides perspectives on HCI history, glancing back at a road that sometimes took unexpected branches and turns. History is not a dry list of events; it is about points of view and differing interpretations.

Jonathan Grudin, Editor

Design Case Study: The Bravo Text Editor

William Newman
University College, London | wmn@pobox.com

William Newman's definitive and engaging article reveals that something we take for granted now was once one possibility of many. Recognizing this can provide a deeper perspective on design choices we face today. Newman is a meticulous student of history who created some of that history himself, working with other pioneers at the University of Utah, Xerox PARC, and other institutions. He contributed to several groundbreaking systems and co-authored the highly influential Principles of Interactive Graphics, *published in* 1973.—Jonathan Grudin

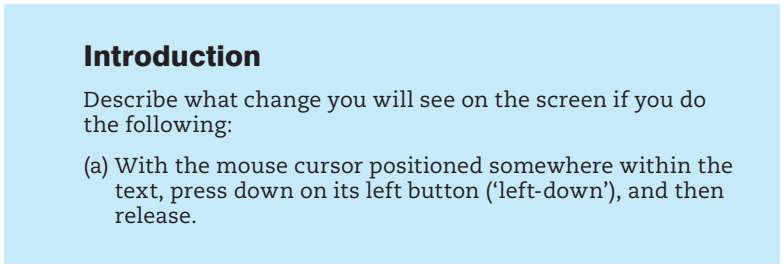
With few exceptions, today's screen-based text editors adhere to a common set of user-interface conventions. For example, they allow users to select a *position* where typed text will be added, by either pointing to the position and clicking, or using the arrow keys to move the insertion point vertically or horizontally. They allow the *selection* of a sequence of characters, by clicking down and dragging across the text, or a sequence of words by double-clicking and dragging. Once selected, text can be *deleted* by pressing the Delete key, or it can be *moved* to a new position by clicking down on it and dragging it. Conventions like these underpin the *standard text-editing user interface* found in today's computer applications. The wide adoption of this standard ensures that users can move easily from one editing program to another, without needing to learn from scratch how to use each one.

The origins of the text-editing standard can be traced back to pioneering work carried out in the 1960s and 1970s, much of it by research groups at Stanford University, SRI, and Xerox PARC. This work contributed to the development of the Xerox Star workstation, which in 1981 became the first available product to offer the emerging standard editing interface. Subsequently the standard was adopted by major companies such as Apple, Microsoft, and IBM. The development of the standard editing interface, whose story is told here, has been one of the computer industry's major success stories.

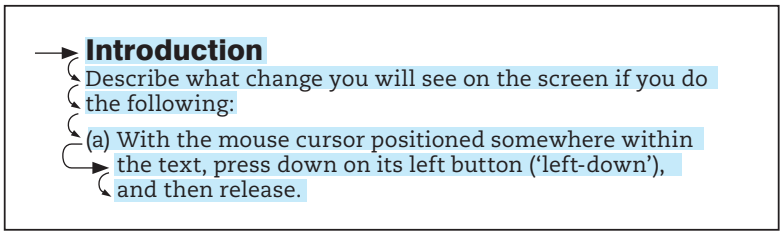
Xerox PARC and the Alto

The histories of Xerox PARC and its Alto personal computer have been thoroughly documented [1,2,3] and need not be retold here in full. Certain aspects of the Alto's design deserve mentioning, however, because they had a strong influence on the design of early text editors. From the outset, PARC's management bought into the arguments of its senior scientists, notably Butler Lampson and Alan Kay, that Xerox's future lay with personal computers and with the applications they could make available to every office worker. The key application domain, argued Lampson and others, would be document preparation, and the first step should be to develop a powerful personal computer, capable of displaying and manipulating lengthy documents involving proportionally spaced, variable-size typefaces [4]. It was also clear, early on, that there would be problems in designing such a text editor, given that the Alto had only 128K bytes of main memory, and nearly half of this was required to store a full-page image for display on the bit-mapped screen.

Fortunately, however, Lampson and Charles Thacker had devised a way to reduce the demands on memory for displaying text [5]. There were many blank regions in a page image, such as margins and between-line gaps (Figure 1a). But the same image could be constructed as a set of linked horizontal *bands* of variable height and width, as shown in Figure 1b. Each band was allocated just enough memory to accommodate the characters of one text line; the spaces between the bands were automatically displayed in background color, consuming no memory. The resulting screen image was therefore indistinguishable from



(a) using a single, full-width bitmap



(b) using a display list containing a set of minimum-size bands, which in this example consumes only a third of the memory required for the full-width bitmap

▸ Figure 1. Memory required for text display on the Alto: The amount used is represented by the colored areas.

a display of the same text as a single bitmapped image, yet required only a fraction of the memory.

Bravo

In 1973, PARC hired Charles Simonyi, who had recently gained his Ph.D. in computer science and was deeply interested in software development methods. On joining PARC, Simonyi set about creating a "software factory" in which to test new approaches such as metaprogramming [6]. He pursued this goal via a series of experimental design projects, using the Alto as a test-bed. An enthusiast for all forms of manned flight, he drew names for his projects from the pilots' phonetic alphabet: Alpha, Bravo, Charlie, etc.

Lampson was meanwhile realizing that building an Alto-based text editor could be an interesting project for Simonyi's software factory, particularly if it were to incorporate a couple of ideas for improving the editor's performance. One of these was a method for minimizing rewrites of the document file, using a *piece table* to keep track of changes to the document (see sidebar). This method is now used in several leading word processors, including Microsoft Word. Another idea offered a way to speed up screen updates by reusing parts of the existing screen image.

Lampson proposed the editor project to Simonyi and described his ideas for implementing it. It became the software factory's second project and therefore received the name Bravo. During the summer of 1974, Simonyi hired a programmer, Tom Malloy, to help him with Bravo's implementation. Others at PARC lent a hand as the program, and its user interface, gradually took shape.

Bravo's user interface. In designing Bravo's user interface, Lampson and Simonyi took a relatively low-risk approach, using existing techniques from other editors in preference to novel, untested ideas. There were, unsurprisingly, many such ideas circulating around PARC at the time. Prominent among these were the recommendations made by Larry Tesler and Jeff Rulifson in their *OGDEN Report* [7]; these included:

- A cursor should be displayed showing where the next character typed will appear.
- A command-last (postfix) language is preferable to a command-first (prefix) language.
- All keys on the terminal that look like typewriter keys should do what typewriter keys do.
- To move text, the source text should be "cut" out of the document, the destination signified, and the material "pasted" in at that point.

Although Lampson and Simonyi would have much preferred to adhere to these recommendations, they could not afford the extra design and implementation effort. They didn't expect Bravo to be widely used and assumed the software factory would soon be moving on to its next project, Charlie. As it turned out, neither of these expectations panned out, but some invaluable lessons were learned from building Bravo's user interface.

Selecting and typing text. Like existing editors at the time, Bravo followed the convention of allowing the user to make a *selection* consisting of one or more contiguous characters, to which the user could apply a chosen editing operation. Clicking the left mouse button resulted in selecting just one character—the nearest character to the mouse pointer. There were two methods for extending this selection as far as another character: by right-clicking on this character, or by dragging to it while keeping the left button down. Both of these methods are now universal standards [8]. In all cases, when a new selection was made, the previously selected text was deselected.

Bravo used underlining to highlight the currently selected characters. While simple to implement, this highlighting was sometimes difficult for the user to find amid a

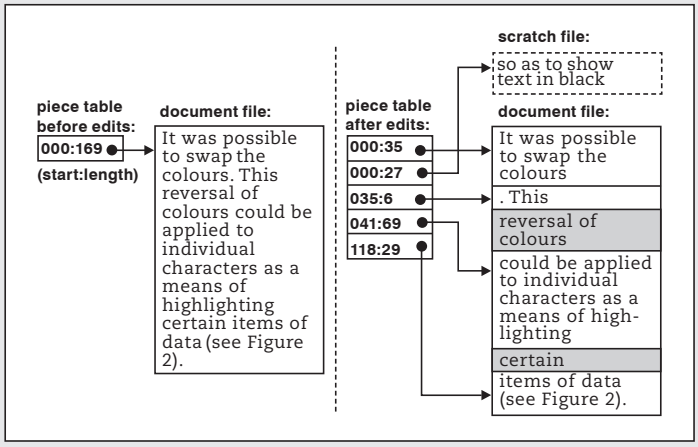

ESC
TAB
CTRL
SHIFT
LOCK

BRAVO'S PIECE TABLE

The piece table, invented by Butler Lampson, solved the problem of achieving a rapid response to editing large text files on the Alto. Most documents were too long to fit in the Alto's main memory and therefore had to be stored as disk files. Each change to the displayed text could necessitate a lengthy file update, during which the user would be blocked from editing.

Lampson realized it wasn't necessary to update the document file after each deletion. Instead the file could be treated as a set of pieces, and a piece table could maintain a record of which pieces had been deleted and which had not. Bravo could then show just the undeleted pieces of text and could ignore the deleted pieces (shaded gray in the diagram).

When a document file was first created by Bravo, therefore, it was treated as a single piece. Whenever the user made an insertion or deletion, the affected piece was split into two pieces at the point of change, one piece before the deletion or insertion, and the other piece following it. If text was inserted, it was appended to a temporary "scratch file" (shown with a dashed boundary in the figure), and a record of it was made in the piece table. As a result, the document file could be left unchanged until editing was complete, and could then be updated by writing out the contents of all of the pieces in the piece table.



▸ Using a piece table to manage edits to a document file.

page of text, particularly when just a single character had been selected. If the text had already been underlined by the user, it now acquired two underlines (Figure 2). Simonyi had experimented with showing the selection by inverting the text to white on a black background, similar to today's standard. On the Alto, however, this took noticeably longer to appear than underlining, and it also made the text less legible. Later, when Xerox developed a higher-resolution screen for the Star workstation, the now-standard practice of inverting the selected text took hold.

As now, the user's purpose in making a selection was typically to indicate a position for entering text. However, since the selection always included at least one character, the user needed the choice of whether to *insert* text to the left of the selection, or *append* text to the right of it. Bravo therefore provided two different one-letter commands for text entry: I to insert, and A to append. An *insertion point*, in the shape of an inverted V, then appeared before or after the selection. The user could type new text and then press the ESC key to complete the operation.

Type-in mode and command mode. The decision to use keyboard characters for issuing commands required the Bravo user interface to have two basic modes. The program always started in *command mode*, in which certain alphabetic keys were interpreted as commands. If the I or A commands were given, Bravo switched into *type-in mode* during which alphabetic keys generated text, and the concluding ESC switched the program back to command mode. Other commands, some of which caused a switch to type-in mode, included:

D *delete selection*
E *select everything*
F *find text*
G *get file*
P *put file*
Q *quit Bravo*
S *substitute text for text*
U *undo*

For the user of Bravo, with its two modes, there was always the risk of starting to enter text when the program was in command mode, causing the entered characters to be treated as commands. The outcome could be quite unhelpful. For example, typing the word "edit" when in command mode would select everything, i.e., the entire text, then delete the selection, leaving the document file empty, then start to insert text, and finally add the letter "t" to a now empty document. Although Bravo provided an

undo command, it applied only to the most recent command, so in this instance only the "insert t" command could be reversed, and all the text was now lost [9].

Many of Bravo's commands caused a mode change because they required input from the user. For example, the selected text could be *replaced* by first typing R and then selecting the text that should replace it. The *look* command, for formatting the currently selected text, required a one-letter formatting parameter to follow it, e.g., B for boldface or I for italic.

Scrolling the text. By the mid-1970s, scrolling was already supported by most text editors, usually by means of single-keystroke commands that, for example, stepped the text up or down by one line [10] or by a half-screen [11]. With a long document, several such steps would often be needed to scroll the text to a precise position.

Simonyi and Lampson realized these multiple steps could be reduced to a single step by using the mouse to control what was made visible. Pointing and clicking the mouse within the actual text was to be avoided, because this would be interpreted as selecting a word or character. To scroll, therefore, the user would have to point in a region where there was no text, and where vertical movement could be specified. The two options available were therefore to use either the left or the right margins.

This led to the concept of the *scroll bar*, as Lampson and Simonyi called it. Bravo's display provided no indications of where its scroll bars were or that they even existed. Adding the necessary graphics to each band of the display would have consumed considerably more display memory, so Simonyi opted for an *invisible scroll bar*, in the left margin, detectable only by changes in the shape of the mouse cursor. When it was moved into the scroll bar, the cursor changed to a double arrow (Figure 3b), and when it was moved out it returned to its normal shape (Figure 3a). As shown in Figures 3c and 3d, if the left or right mouse button was pressed while the cursor was within the scroll bar, its shape changed to a single up- or down-arrow. When the button was released, the adjacent line moved up to the top of the screen (left button), or the top line moved down to the cursor position (right button). This had the advantage that a scrolling operation could be reversed by clicking the other button.

Pressing the middle button on the mouse changed the cursor to the *thumbing* symbol, a striped right-pointing arrow (Figure 3e). On the button's release, the document would scroll to a position in the document proportional to the cursor's position in the scroll bar. Today's scroll bars in Windows provide the same thumbing effect when the elevator is moved.

The line bar. Simonyi also provided a second invisible bar, the *line bar*, to the right of the scroll bar; when the cursor was within this bar, it changed to point toward the text (see Figure 3f). By left-clicking while pointing within the line bar, the user could select the whole of the adjacent line, and this selection could be extended by right-clicking opposite to another line. Clicking the middle button selected the whole paragraph. A similar line bar, also invisible, is provided in many of today's word processors.

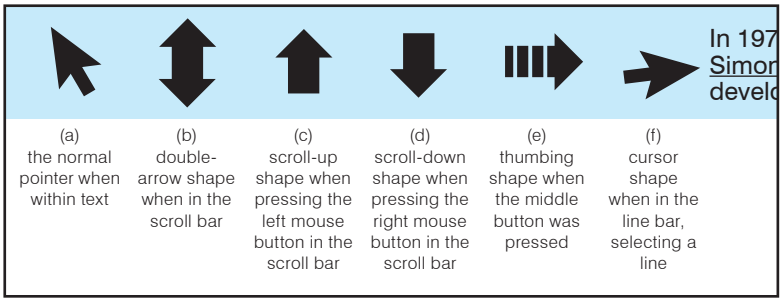
Bravo users generally had no difficulty positioning the cursor in the correct vertical slice of the screen. What they lacked was context—where were they in the document as a whole? In today's editors, a visible scroll bar offers that context.

Bravo's deployment. The development of Bravo was followed with great interest by PARC's researchers. At that time they were still using relatively slow and awkward systems for creating documents, including document compilers such as PUB [12] and line editors like QED [13]. When released late in 1974, Bravo transformed out of all recognition the task of preparing a document. Within a few months it was in use not only by PARC's computer scientists, but also by other Xerox researchers and by a growing number of administrative staff.

It was some years before Bravo was made public. In 1975 Xerox

selected just a single character, but it worked well otherwise. When underlined text was selected, it acquired two underlines.

▸ Figure 2. Underlining and selection in Bravo: Here, underlining has been applied to the word "two," and the words "two underlines" have been selected.



▸ Figure 3. Bravo's cursor shapes.

began the development of the Star product, which was to incorporate many features of Bravo when it finally launched in 1981. Prior to that, Bravo, although kept largely under wraps, was included in the donation by Xerox of 50 Altos to select American universities. Meanwhile it was shown to a number of potential customers, including a congressional committee exploring new technologies, who visited PARC in 1975.

The congressional visit led to a purchase of Alto systems and laser printers in 1978 for use by the U.S. Congress, the White House, and the U.S. vice president's office [14]. These prestigious organizations were far from ideal as test sites, for they required extensive support—thousands of miles from PARC—and they tended to use the Alto in bizarre ways. For example, President Jimmy Carter's staff continued to have their typing pool prepare drafts of documents; Bravo was used only to retype and print the final text once it had been approved. By 1980 the White House had almost given up using the Alto system, and in 1981 they were about to get rid of it. It was put back into service, however, because it alone could print Ronald Reagan's speeches in a large typeface that he could read in public without wearing spectacles.

Conclusion

PARC was a powerhouse of innovation during the 1970s, and Bravo was one of its most influential programs. Among its many firsts:

- Bravo was the first general-purpose editor to support on-screen, multifont, variable-size text editing. Its designers foresaw that this style of editing, and the personal computers to support it, would render all other styles obsolete.
- Bravo was the first WYSIWYG [15] editor, and indeed it was responsible for introducing this term to interface design. Previous editors rarely made any attempt to match the display to the printed page; this meant that users could not check whether they had formatted the document correctly, except by printing it out.
- Bravo was the first text editor whose speed of response to commands was largely unaffected by the size of the document. This was achieved through the use of internal structures—principally the piece table—that to this day remain an essential part of word processors.

Bravo's user interface was not without its faults, including modes that could result in user errors. These were rectified, first in Larry Tesler's Gypsy editor, which used much of Bravo's internal design [16], and later in a Bravo redesign led by Simonyi, resulting in a system called BravoX that was included in the congressional purchase. Efforts by Xerox to commercialize BravoX came to nothing, however, and in 1981 Simonyi joined Microsoft, then a small company producing operating systems. He led the development of Word, basing the design heavily on BravoX, and the product was released in October 1983.

Acknowledgements

I am deeply grateful to Bill Moggridge, whose book Designing Interactions *revealed to me the insights to be gained from interviewing designers, and led eventually to my embarking on this investigation. I am also most grateful to Butler Lampson and Charles Simonyi for their willingness to be interviewed at length about Bravo's history, and then to read the drafts of this case study and provide suggestions and corrections. Without their help, this study could not have been written. I would like also to thank Xerox PARC's Sally Peters for her tireless help in locating essential reference documents.*

ENDNOTES:

1. Brand, S. *Two Cybernetic Frontiers*. Random House, New York, 1974.

2. Thacker, C.P., McCreight, E.M., Lampson, B.W., Sproull, R.F., and Boggs, D.R. Alto: A personal computer. In *Computer Structures: Principles and Examples, second edition.*

3. Hiltzik, M.A. *Dealers of Lightning: Xerox PARC and the Dawn of the Computer Age.* HarperCollins, New York, 1999.

4. Lampson, B.W. Why Alto? Xerox Inter-Office Memorandum, 1972; http://www.digibarn.com/friends/butler-lampson/

5. Fraser, C.W. and Krishnamurthy, B. Live text. *Software: Practice and Experience 20,* 8 (1990), 851-858.

6. Simonyi, C. Meta-programming: A software production model. Xerox PARC Technical Report CSL-76-7, 1976.

7. Tesler, L.G. and Rulifson, J.F. *OGDEN: An Overly General Display Editor for Non-Programmers.* Xerox PARC, Palo Alto, 1973.

8. The use of double clicking for word selection was added to Bravo later, after its invention by Tim Mott. See Moggridge, W. *Designing Interactions.* MIT Press, Cambridge, MA, 2007, 69.

9. A more forgiving Undo facility was later provided. It reloaded the file and replayed all of the user's subsequent edits.

10. Bolski, M.I. *The vi User's Handbook.* AT&T Bell Laboratories, 1984.

11. Wiseman, N.E. A scope text editor for the PDP7/340. Paper presented at the DECUS European Spring Seminar, 1966.

12. Tesler, L.G. PUB: The document compiler. Stanford AI Laboratory Operating Note 70, 1972; http://www.nomodes.com/pub_manual.html.

13. Deutsch, L.P. and Lampson, B.W. An online editor. *Comm. ACM 10,* 12 (1967), 793-799.

14. Smith, D.K. and Alexander, R.C. *Fumbling the Future.* Morrow, New York, 1988.

15. "What you see is what you get," a catch phrase made popular by Flip Wilson, who played the character Geraldine on a 1960s American TV show, *Rowan and Martin's Laugh-In.* Jim Morris adopted the phrase and the acronym to describe editors that presented an accurate display ("what you see") of the document as printed ("what you get"), and it was soon in widespread use around PARC.

16. Tesler, L.G. and Mott, T. *GYPSY: The Ginn Typescript System.* Xerox PARC, Palo Alto, 1975.

ABOUT THE AUTHOR

William Newman gained his Ph.D. in computer science at Imperial College, London. With Robert Sproull he wrote the seminal 1973 textbook, *Principles of Interactive Computer Graphics.* During the 1970s and 1990s he worked at Xerox PARC and then at Xerox Research Centre Europe. He is currently a visiting professor at University College, London, engaged in documenting the design history of today's interactive desktop.

DOI: 10.1145/2065327.2065342
© 2012 ACM 1072-5220/12/01 $10.00