TEXT TECHNOLOGY

AD DOCENDUM o INVESTIGANDUM o SERVIENDUM
WRIGHT STATE UNIVERSITY 1967

Volume 1 Number 6
November 1991

In this issue . . .

EDITOR'S CHOICE

Jim Schwartz

Don't Forget to Write

Oops, I forgot!

[illegible]

This publication is funded in part by a research grant from the *Wright State University President's Club.*

TEXT Technology. Volume 1, Number 6. Copyright ©1991 by Wright State University. All rights reserved. ISSN: 1053-900X. Also indexed by ERIC, COMPUTER AND CONTROL ABSTRACTS (INSPEC), COMPUTER LITERATURE INDEX, LITERARY CRITICISM REGISTER, and SOFTWARE REVIEWS ON FILE. *TEXT Technology* is published bimonthly by Wright State University—Lake Campus, 7600 State Route 703, Celina, Ohio, USA 45822-2921; (419) 586-2365. Postage paid at Celina, Ohio. SUBSCRIPTION RATES US funds: $20 (US); $27 (Canada); $35 (Foreign). Address all subscription inquiries and manuscripts to the Editors, *TEXT Technology*, Wright State University—Lake Campus, 7600 State Route 703, Celina, Ohio, USA 45822-2921. Note that the Editors are not responsible for any unsolicited manuscripts. Allow 4 to 6 weeks for subscription processing. Selected back issues are also available; contact the Editors for more details. POSTMASTER: Send address changes to *TEXT Technology*, Wright State University—Lake Campus, 7600 State Route 703, Celina, Ohio, USA 45822-2921.

WRIGHT STATE LAKE CAMPUS

TEXT Technology is composed on a Northgate Elegance 486-25 computer and printed using an IBM Personal Page Printer II Postscript printer. Images and text are scanned with a Hewlett-Packard ScanJet Plus scanner. Software used includes Microsoft *Windows*, Aldus *PageMaker*, *Caere Omnipage 386*, *WordPerfect*, *HP Scan Gallery* and *Paintbrush*, Micrografx *Designer*, *CorelDRAW!*, Lotus *Ami Professional*, Microsoft *Windows Write*, and Symantec's *Q&A*.

COLUMN ONE

Eric Johnson

A Bright Future for *BASIC*

Recently at a conference, I asked a professor of classical languages who had created a computer application for processing Greek texts what computer language he had used. He told me that his program was written in *COBOL*. I politely suggested that it would be much easier for him to use a language designed for text processing: *Icon* or *SNOBOL4*. He bristled. "I can't be bothered to learn every new-fangled language that comes along," he said.

We had a spirited discussion. I pointed out that whatever virtues *COBOL* had for producing business reports, it had clumsy facilities for manipulating character strings and graphics. Moreover, there were few, if any, sources he could turn to for help with programming in *COBOL* for the humanities. Nevertheless, he remained convinced that he could get along very well, thank you, without learning another programming language.

Until recently, I might have had a similar conversation with someone who programmed for the humanities in *BASIC*, although *BASIC* is better adapted to the humanities than *COBOL*. However, now there are so many excellent versions of *BASIC* and there is so much information about how to use them, as well as third-party support, that the programmer who started programming in *BASIC* and who stayed with it might justifiably feel quite smug.

Not long ago, in the IBM-compatible microcomputer world, the *BASIC* programmer had only GW*BASIC* and *BASICA*. They, of course, are nearly identical, the chief difference being that *BASICA* requires IBM-brand hardware. Today, one company, Microsoft, offers four kinds of *BASIC*.

QuickBASIC, developed in the mid-1980s, offers at least two kinds of advantages over GW*BASIC* and *BASICA*. First, it runs much faster—by as much as a factor of ten—because it is a compiler rather than an interpreter. Second, *QuickBASIC* contains a number of additional features, including control structures that allow programmers to organize their programs more clearly; line numbers and "gotos" are not needed.

QBASIC, which should not be confused with *QuickBASIC*, is similar to *QuickBASIC* except that it is a smaller implementation. It is an interpreter, and thus slow, and it is free. It is distributed with DOS starting with version 5.0, and it is included with at least one textbook on *BASIC* programming.

QBASIC will almost certainly drastically change the way *BASIC* is taught in schools. Because *QBASIC* contains powerful control structures and does not require line numbers and "gotos", students can be taught to use *QBASIC* to write programs that look very much like programs in Pascal or other structured languages. (It is increasingly difficult to determine what computer language is being used by simply glancing at the code on a programmer's screen.)

Microsoft's monumental *BASIC* Professional Development System (*BASIC* PDS) is intended for professional programmers who write large applications that may require huge ISAM files. It includes *QuickBasic* Extended (QBX) which, as its name suggests, is an enhancement of *QuickBASIC*. It offers multiple editing environments and elaborate debugging facilities. *BASIC* PDS, especially with QBX, can be of considerable value to the humanist programmer who can afford it.

The hottest implementation of *BASIC* currently is Microsoft's *VisualBASIC* (VB) for *Windows* 3.0. VB allows ordinary mortals to write programs that display *icons* and buttons, open and resize *windows*, and respond when a mouse is moved or clicked. VB is a must for humanists who want to write computer applications that run under *Windows* 3.0.

Microsoft deserves praise and support for its development and marketing of these four *BASIC*s, but there are solid versions of *BASIC* produced by others too: GFA-*BASIC*, *PowerBASIC* (formerly *TurboBASIC*), *TrueBASIC*, and *ZBASIC*. Moreover, there are products similar to VB that allow for the quick creation of *Windows* 3.0 applications.

There are, of course, also numerous versions of *BASIC* available for the Macintosh and for other computers.

The objections that have been traditionally raised to using *BASIC* are no longer sound. It was argued that its programs were slow to execute. Well, some versions of *BASIC* are not too fast, but applications in some other languages are even slower, and, if speed is supremely important, *BASIC* PDS can be used; its programs run about as quickly as those of any high-level language. Moreover, *BASIC* programs can often be written more quickly than similar programs in other languages—few would care that a program runs three seconds faster if it took three weeks longer to develop.

The most damning complaint about *BASIC* has always been that it is not a structured language, and that it encourages a kind of spaghetti code that is impossible to maintain. In the first place, clear, modular programming is produced by disciplined programmers who are determined to write maintainable code; such programmers can write good code with any version of *BASIC*, and undisciplined hackers will propagate unintelligible code in even the most structured language. In any case, the *BASIC* implementations mentioned in this article contain control structures similar to those of other languages; there is no reason that the code produced by these implementations cannot be as clear and maintainable as that of any language.

An enormous amount has ben published about programming in *BASIC*. In addition to manuals for each *BASIC* implementation, there are many trade volumes and dozens of textbooks about programming in *BASIC*. Almost every computer publication that discusses programming gives examples in *BASIC*, and there is a new magazine devoted solely to *BASIC*.

There are good books about programming for the humanities in three computer languages: in *SNOBOL4* (several books), in *Icon*, and in (of all things) standard Pascal. But, at least as far as this writer knows, there is no book devoted exclusively to programming for the humanities in *BASIC*, and that is odd.

There is, however, excellent third-party support for *BASIC*. There are companies that can provide large, reliable libraries of routines and various kinds of toolkits that allow the *BASIC* programmer to use fast, sophisticated functions without the exhausting coding. Some implementations (such as *BASIC* PDS) include libraries of math and graphics functions.

BASIC has always been the most commonly used language on microcomputers, and it appears that it will continue to dominate. Certainly, for the *BASIC* programmer, this is an interesting and exciting time to work with the language.

* * * * * * * * * *

Eric Johnson is Professor of English and Dean of the College of Liberal Arts at Dakota State University, Madison, SD 57042. He is the Director of the International Conference on Symbolic and Logical Computing, and he has published more than fifty articles and reviews about computers, writing, and literary study. His BITNET address is ERIC@SDNET.

Articles & Reviews Welcome

TEXT Technology welcomes submissions that pertain to wordprocessing, text-analysis, and research applications in professional writing, either corporate or academic. Also, hardware and software reviews are encouraged, but please contact Jim Schwartz before submitting them—either call Jim at 419-586-2365 or send him a note at the following BITNET address

JSCHWARTZ@DESIRE.WRIGHT.EDU

Manuscripts may be submitted on MS-DOS 5¼" or 3½" floppy disks, through BITNET (in ASCII), or in hardcopy format.

How We Produced the College Catalog on Campus and Lived to Tell About It (so far)

Michael Dobberstein & Ruth MacDonald

With the arrival of desktop publishing labs on college campuses and the stringent fiscal climate of most colleges and universities in the 1990s, some campuses, like ours, have decided to use their desktop publishing facilities to produce the college catalog. Technologically, this is a straightforward job. It means using desktop publishing to design and typeset the catalog, hiring a service bureau to provide typeset-quality output, and sending camera-ready copy off to the printer.

A great deal of work is involved if the catalog must be converted to wordprocessing files first (necessary for use with page-composition software), but the real complications come when a total redesign is called for. Our experience shows that redesigning and producing the catalog on campus using desktop publishing requires equal parts design, technical, and political expertise.

ADVANTAGES

To be sure, there are advantages to taking on the task: using desktop publishing technology makes it easier to update and keep up-to-date the information in the catalog, and makes that same information available for other uses, such as online access to the catalog, printing pages for advising manuals, student handbooks, and other academic departmental uses.

Desktop publishing keeps design decisions on campus, not in the hands of outsiders, and makes redesign and massive changes a cheaper, more timely undertaking. Galleys can be eliminated, and multiple reviews of page proofs can be done quickly and cheaply. Finally, the project can advertise the presence of the technological platform on your campus and is a good promotion of both the desktop-publishing lab and desktop-publishing courses.

DISADVANTAGES

Our experience suggests that junior faculty members, who are likely to be those with the computer expertise, may find an inordinate amount of time, which might otherwise go toward teaching and research, diverted into this project. The problem here is one shared by many faculty in the liberal arts who have done innovative work with computers: because such work does not fit easily into traditional patterns of research and publication, evaluating it for tenure and promotion can be difficult. In our opinion, on-campus production of the catalog is a significant contribution to the university. However, we would be less than realistic to believe that all faculty shared our view.

In addition, junior faculty members may be at a disadvantage in working with the more senior faculty and administrators, especially department heads, whose prose may need heavy editing for intelligibility. Even with the backing of the loftiest of campus officials, cooperation may be hard to get without someone of at least department-head rank using interpersonal skills and institutional might.

Finally, there were aspects of the project that were just beyond our abilities—like the photography needed for the cover. Desktop publishing may not be able to do it all.

OUR PLAN

After some discussion, we decided against using the makeover of the catalog as a student project. Although we've been teaching desktop publishing for a year or so, it is still very new on our campus. The course introduces students to *Ventura Publisher* 3.0, but most undergraduates are unlikely to master this software well enough to be productive in desktop publishing a book-length manuscript. Also, undergraduates usually do not have the experience with academic prose, document design, or even simple copy-editing to be able to take on such tasks.

The actual work on our project was accomplished with a staff of three, two English faculty members (one of whom is head of the department) and a graduate student. Using others on campus as consultants, we produced a brand-new catalog in about six months, from initial planning to the printing of camera-ready copy.

Once we had our staff, our production plan took shape. This consisted of two parts: converting the catalog text to Microsoft *Word* files (*Word* is the wordprocessing standard on our campus), then building a catalog using *Ventura Publisher*. Though we have a Dest PC 1000 scanner in our desktop publishing lab, we quickly discovered that there would be no shortcut to converting catalog copy to machine-readable text. The Dest OCR software, *Text Pac* 3.0, simply could not read the typefaces in the catalog.

Thus, all text had to be manually entered into *Word*, a tedious job even for a good typist. Our typist felt better about the project when major editing was done first, so that changes after the text was entered were minimized. In our case, it helped to have a student typist who could give us a student's point of view on the text entered. One comment we heard was that some course descriptions were so detailed that a student might as well read the description as go to the class described—there was no brevity in what was supposed to be a summary. Once again, editing beforehand might have helped, though certain campuses have Byzantine processes for changing course descriptions. After all, brevity may not be possible.

THE REDESIGN

Redesigning the catalog went through several stages, from initial planning to convincing department heads that our format served the primary purpose of informing students about degree requirements. The limitations of the old catalog provided the impetus for our redesign. Because the old catalog was small and crowded (6" X 9"; 273 pp.), we designed a book that was larger and contained more white space, with fewer pages (8 1/2" X 11"; 237 pp.). We kept margins wide and used a vertical rule to separate the text column from the margin.

We also wanted a catalog in which finding information was easier, so we changed the organization. We took all course listings from departmental sections and grouped them alphabetically in a separate section. From the department's point of view, however, the most important changes we made were in how degree requirements were to be presented. A hodge-podge of styles prevailed in the old catalog, but we believed the new catalog should have a consistent format for listing degree requirements and for providing easier reading.

There was some resistance to changing the old formats, and in some cases departments had to re-think their degree requirements to make sure that accurate lists of necessary courses were provided.

Developing a readable, logical, consistent format for all degree requirements was the most daunting design challenge we faced. Since such requirements mean long lists of course numbers and names, we used the table edit feature in *Ventura* 3.0 to save space. Using table edit, we were able to create 2-column lists wherever we needed them without having to create separate frames for each list. We used 9-point type for table text and did not break tables across pages. Even at 9 points, some course titles had to be shortened; nevertheless, we tried to keep abbreviations to an absolute minimum.

Our chief design goal was ease of readability, but some sections forced us to use type as small as 6 points. Because of this, and because we wanted the whole book to look as good as possible, we decided against using our 300 dpi laser printer for the final, camera-ready copy. While using the laser printer may have been the cheapest thing for the university to do, we believed that it would result in a final product that looked like a class project rather than a class act. We used a service bureau to produce Linotronic copy (at $4.65 a page), since the money to buy better printing technology was not available on campus.

We tried to be careful, and as thorough as we knew how, in choosing both our service bureau and our printer. Before deciding which service bureau to use, we asked for samples of work, and we checked on references for the printer. The company we chose—a specialist in college catalogs—gave us a bid that was significantly lower than the price the university had previously paid a local company to print the catalog. We knew that at least the first time through our credibility—and desktop publishing's—was on the line. We worked with the printer as early as possible to make sure that we were preparing camera-ready copy in a way that was acceptable to their technology.

SIGNIFICANT OTHERS ON CAMPUS AND THE DESKTOP PUBLISHER

Of course, many, many people on campus were responsible for providing us with information and for reviewing, at every step, our progress. Our experience here suggests the following:

1. Make sure you keep relations among the team members light and easy—this is a difficult, tedious job that requires that you all work together well over what will seem a very long haul. It helps to have at least one uncontrollable optimist on the team at any given moment—we found the role passed among us.

2. Archive drafts and correspondence against complaints of inaccuracy. Don't throw drafts or edited copy away; you may need to defend yourself against the inevitable griping about errors in fact or in fiction. CYA files are the best revenge.

3. Scheduling will be difficult, especially as academics don't like to abide by rigorous schedules. Make sure you keep records of who was sent what copy when, deadlines for return, and when it was actually returned. As scheduling is so problematic, establish a timetable that includes lots of time for proofreaders to turn copy around, and time for you to pressure them gently after they miss deadlines.

4. Most academics do not enjoy proofreading, perhaps even do not know how to proofread. We have learned by sad experience that oral instructions do not get remembered; we would advise a precise set of written instructions, right down to using a red pen or pencil. Such instructions should include directions on how to proofread and what to proofread for, and should be sent to all persons who are responsible for copy or who want to be responsible for copy—those academic busybodies who must have a say in every word that issues forth in print from anywhere on campus.

5. There will be those editors who think they need not devote real time to the project until they see page proofs—even though that seems like the last minute to you. They will claim, and in their minds they're right, that they've never seen the copy before even though you know they may have been asked to read typescripts and galleys previously. For them, it's not real until it looks like a page. This explanation does not excuse their desultory proofing earlier—once again, an insight into the quirky academic mind.

6. Be sure you know what the graphic standards are from your office of publications or whatever it is called on your campus. Following the standards avoids last-minute changes and lets you take them into account when redesigning or rewording becomes an issue.

7. Be sure you know the legal issues involved. Even if the disclaimer in the current catalog is deadly dull, change it only after review by the university attorney. Student consumer information legislation has a bearing on what the financial aid office must say, what veterans affairs must say, etc. The various officers on campus in charge of these activities usually know what they must do legally; it may be that you can get them to use plain English, but sometimes that's not possible. The courts have ruled that some college catalogs are contracts between the student and the institution; accuracy as well as the disclaimer become crucial in such a legal climate.

8. While accuracy with names and degrees is not a legal issue, it is a sensitive one, especially with those whose names are misspelled and whose degrees are misattributed. Take special care here.

9. Beware the people on campus who globalize your errors or who may not understand what a draft copy is. These are the people who think that because there is one error, major, minor, or otherwise, the whole document is called into question for its accuracy. Especially for those in the sciences and other fields that emphasize precision, who may not understand writing as process, one error is intolerable. You may not be able to convert such nay-sayers, but at least you may understand their point of view, if not excuse it.

10. Sing your own praises. While your name need not appear on the title page, you can include a statement there about the catalog being produced in your campus lab.

11. You will, when all is said and done, receive some praises--archive those, too, for your next personnel evaluation. If no one seems to notice your work, you can always send a memo on a wide distribution thanking all of those who helped you and thereby call attention to your accomplishments.

CONCLUDING CAVEATS

Redesigning the catalog is a natural suggestion especially when a new technology will be used for the catalog copy. Such a redesign may have tangible benefits, such as an improved appearance and improved readability for your document. It also has real drawbacks, such as inevitable complaints when anything on campus changes, even for the better. Redesigning also takes longer than simply updating the current format using a different technology. If you opt for redesign, give yourself plenty of time—as much as six months (as we did) or even a year, which, in academic time, may be considered hasty. Try changing the size of the catalog if you have redesign in mind—it makes a statement that the product is "different," even without any other changes.

Once you've succeeded in your effort, especially if it's a total redesign, you may be expected to come up with a new design every couple of years. Maybe you can pull this off once, but beyond that, the burden may be too great on your creativity to be successful consistently. The administrative time it takes to process a new design with multiple editings and revisions makes such changes more than most institutions and individuals can bear. A new design once a decade is more than most institutions see; once every five or six years would be truly revolutionary.

For the next time through, leave instructions on how to do the operation again. With any luck, you'll be nowhere to be found when the job comes around for the second time, but the process will need to be recreated from whatever archaeological materials you leave behind. Copies of diskettes, including at least one backup copy, need to be archived in someone's file drawer along with changes made periodically by a small group of responsible people. Allowing too many hands access opens the possibility for unauthorized changes and can cause problems with data integrity.

People will begin pointing out your errors almost immediately. Thank them and keep notes on what to fix next time. Perhaps you'll want to keep a copy of the new catalog devoted only to the copy-editing comments of others.

Even if you find yourself in the same position next time the catalog needs to be reissued, don't rely on your memory. Leave yourself notes on what to do differently next time. Human beings and computers can be mighty forgetful.

* * * * * * * * * * *

Michael Dobberstein and Ruth MacDonald teach in the Department of English and Philosophy at Purdue University Calumet, Hammond, IN 46323; ph. (219) 989-2259

New Subscriber Information

To academic and corporate writers and teachers of writing, *TEXT Technology* brings analyses of microcomputer hardware and software, discussions of programming techniques (both in languages and in applications), book reviews, updates of significant events in computing around the world, bibliographic citations, and much more.

TEXT Technology, created by the editor of the *Research in Wordprocessing Newsletter*, also will become an information clearinghouse for subscribers' opinions and queries about personal computing during the 90s—and beyond.

Subscription rates for one year (6 bi-monthly issues—16 pages) of *TEXT Technology* are as follows:

US	$20
Canada	$27
Foreign	$35

All prices are in US funds

Please make and fill out a copy of the form below and send it with your check or purchase order to

Subscriptions Department
TEXT Technology
Wright State University—Lake Campus
7600 State Rte. 703
Celina, Ohio, USA 45822-2921

- -

Name ______________________

College or Corporation ______________________

Street ______________________

City ______________________

State or Province ______________________

Zip Code or Postal Code ______________________

BITNET ______________________

Affiliation: ACH______ ALC______ ALLC______ 11/01/91

"Hey! Where Did My Term Paper Go?" and Other Problems for *WordPerfect* Students

James Guthrie

Having taught several beginning desktop publishing classes the fundamentals of *WordPerfect* 5.1, I keep thinking I have seen every kind of mistake a novice can possibly make. So much for reckless optimism. With each new class, a new and unpredictable error appears. Usually I can figure out what happened and engineer a quick fix, but occasionally I have had to blurt out, quite unprofessionally, "I don't know what the heck happened. Reboot."

With the passage of time, however, I have become interested not just in fixing errors, but in understanding how those errors were made in the first place. What did my student see on the screen that made him or her veer so wildly off course? Is there anything *WordPerfect* could do to make the software easier to use and understand?

Many of the errors result from mutual misunderstandings between the student and *WordPerfect*, so that a sort of collaboration to go wrong occurs between user and software. Not that it's hard to go wrong: *WordPerfect*'s labyrinthine menu and submenu user interface makes it a relatively complicated application to use for a wordprocessing package. Like most best-selling, long-lived software applications, *WordPerfect* has piled bell upon whistle until the range of choices has become overwhelming. Because features do not always work analogously, users in a quandary over what to do next cannot depend upon consistency within the interface to give them hints.

Version 5.1's new drop-down menus and mouse compatibility do speed up the process of creating documents, but they haven't made *WordPerfect* any more lucid (and even the mouse seems only relatively useful to an unreconstructed function key pusher like myself). The trick to using *WordPerfect* successfully still lies in interpreting the short, rather cryptic messages and submenus appearing at the bottom of the screen, and I propose to look at a few of those that can cause unwary students to stumble.

Because I teach *WordPerfect* 5.1 for the PC, students who enter the class without at least a rudimentary grasp of DOS are already at a distinct disadvantage, particularly in regard to knowing how directories work. For example, students who claim they cannot find WP files saved on their disks have often merely forgotten which directory they stored them under, and since the "List Files" function in *WordPerfect* displays the contents of only one directory at a time, students may jump to the conclusion that what they don't see simply doesn't exist.

But knowing where a file went doesn't present as much of a problem for *WordPerfect* beginners as knowing where one will go, because designating a default directory in *WordPerfect* is not an intuitively obvious procedure. From the "List Files" submenu the user must select "7--Other Directory" and then type in the name of the directory to be used as the default. Confusing matters even further, *WordPerfect* responds to "7--Other Directory" with the message, "New Directory =", when the default directory may not be new at all. Finally, even when the default directory has been designated, the edit screen itself does not show a pathname until the file has been named; so the novice user won't have a clue for a while whether a default directory has been successfully assigned or not. *WordPerfect* programmers could solve the first problem by rewriting Option 7 as "7--Other Directory/ Default Directory". Then, they could also add this message: "If you choose to name this document, it will have the pathname C:\WP51\, etc."

The process of saving files is itself a bit perilous for novices. *WordPerfect*'s choice of F10 or F7 often confuses students, initially, but even having grasped the difference between the two, they still may not understand exactly how to respond to *WordPerfect*'s prompts during the save procedure. Pressing F7 elicits the message, "Document to be saved:", which many students interpret as a question, answering "Y." If they do so, they will be surprised when the edit screen sets up a new page for text, leaving no more opportunities to name the just-finished file. When students complain that work they had saved has disappeared, I now know to look around in their directories for a new file titled "Y," often found lurking off-screen near the bottom of the alphabetical listing, hoping nobody would scroll down and find it there. The folks in Orem, Utah, could help lessen the likelihood of this happening by rephrasing the prompt as a question: "What do wish to name this document?"

Retrieving a file can puzzle students even more than saving one. The most common method of retrieving a file in *WordPerfect* is to use the cursor to select it from the displayed list of files within a directory and then put it on the screen by pressing "1 Retrieve." (Students could also use Shift+F10, which will cause WP to prompt them for the document's name, but who ever remembers what he or she called a file written way back when?)

Novice users have to resist their initial impulse to press ENTER rather than "1 Retrieve", for pressing ENTER causes *WordPerfect* to perform the same function elicited by pressing "6 Look". Then, even though the user can see the file, he or she cannot edit it; further, the "Next doc" and "Prev doc" messages at the bottom of the screen will make no sense to someone who didn't understand the difference between "Look" and "Retrieve" in the first place. I think ENTER would better serve users as a redundant feature for retrieving files, rather than providing another way merely of looking at them—an activity which, in my experience, is seldom necessary.

More sophisticated WordPerfect *users make more sophisticated mistakes.*

Students who don't understand the process of retrieving files often accidentally append files to each other. In some cases, students turn up with documents that have file after file accidentally appended. This predicament arises when, while still in the midst of editing a file, students use F5 to list files. Then, if they try to retrieve a file from the list, *WordPerfect* will display the message, "Retrieve into current document?" Students may interpret "current document" to mean "on my screen" rather than "document currently open." "Of course I want you to put the document on my screen!" students mutter to themselves. "Why do you think I pressed `Retrieve'?" Then, they respond with "Y" and the retrieved file is immediately appended to the open one, a fact that students may not realize until they happen to look at their document's final page. I think this problem would occur less frequently if the word "current" in WP's prompt were replaced by "open."

More sophisticated *WordPerfect* users make more sophisticated mistakes. For example, knowledgeable novices can get themselves thoroughly confused while creating headers (or footers). For one thing, students who wish to edit headers they have created may not realize that, they must place their cursor to the right of the header code—otherwise, *WordPerfect* cannot find the header to be edited. In fact, the whole concept of *WordPerfect*'s processing of codes from left to right is crucial to understanding how the product works.

But *WordPerfect* itself doesn't always work strictly from left to right; for example, although the cursor must be placed to the right of a header for that header to be edited, any Text Box or Figure Box can be edited regardless of the cursor's location. Headers (and footers) are idiosyncratic in other ways, too—such as in their preference for being placed at the very beginning of a document, virtually before any other codes. Placing them anywhere else invites them not to appear on their designated pages or to conduct themselves in other equally inappropriate ways.

Even though you can edit a text box or a figure box from anywhere in the document, those two features (and their ilk, under "Graphics") pose problems of their own. Figure boxes, for example, can be created with alarming ease: all you have to do is to invoke the graphics function, Alt + F9, choose figure, and then select "Create". As soon as the "Definition: Figure" submenu appears, a figure box using the default dimensions has already been created.

Users returning to the edit screen may be surprised to see the text on their screens unceremoniously shoved aside by the newly created figure box; worse, when they use the Reveal Codes function to find the offending figure and delete it, they may not find it where their cursor was last located. Instead, *WordPerfect* inserted it at the last position in the document where the default dimensions could be logically applied.

Such spontaneous creativity is not a problem in the case of most other *WordPerfect* features. When a user checks the top/bottom margin settings submenu under Page/Format, for example, those margin settings are not re-inserted by the mere act of looking at them. Graphical elements and even tab settings, however, are so easily invoked that students' documents may become infested with accidentally created codes.

Sometimes *WordPerfect* 5.1 can place a user in a no-win situation. Look at what happens when a student creates a box 2 inches wide by 2 inches high, and then tries to cut and paste into it a page-length paragraph. *WordPerfect* responds by displaying the message, ERROR: Too Much Text. Then, despite the text box edit screen's direction to use EXIT (F7) when he or she is done, the hapless student cannot use that usually dependable means of escape to get out of the edit screen and enlarge the box's dimensions under the "Text Box Definition" submenu.

The only solution is to delete text until the remnant fits--and that deleted text cannot be recovered. It seems to me that the *WordPerfect* people back at headquarters in Orem, Utah, could have worked it out so that the EXIT key still functioned, perhaps by having *WordPerfect* hold the offending text in clipboard memory until the box dimensions were made large enough to accommodate it.

Editing an existing graphical element can pose a problem for the novice user as well, chiefly because the menu options involved repeat themselves confusingly. After having chosen the figure or text box to be modified, the student chooses "Edit" from a submenu; then, within the "Definition" submenu, "Edit" must be chosen again. Furthermore, the word "Edit" means two different things for text boxes and for figures.

For a text box, "Edit" means that a user can add or delete text, while for a figure box it means that the user can manipulate the figure, not add text (since we don't normally think in terms of "editing" an image, perhaps "Modify" would have been a more felicitous choice of words for figure boxes). Finally, the "Edit" screen for figures includes, peculiarly, the commands "Move", "Scale", and "Rotate" twice each, with no indication that *WordPerfect* provides one set of commands as a shortcut for the other.

Repetitive and unclear onscreen instructions can also bedevil the student wishing to copy text from one place to another. Having blocked the text to be copied, the student selects Control + F4, which produces a line submenu reading: "Move: 1 Block; 2 Tabular Column; 3 Rectangle." "But I don't want to move anything!" says the student. "I just want to copy the block of text!" No problem—but first he or she has to be brave enough to select "Move: 1 Block." Then, another line submenu appears: "1 Move; 2 Copy; 3 Delete; 4 Append." Why "Move" should have to appear twice is anybody's guess.

But how much hand-holding should a computer software application be expected to do? At what point may its designers fairly credit a user with enough sense to deduce what must be done next? As a programmer, you don't want to make the screen look too busy, for in a user interface that has become too solicitous, unnecessary reminders and prompts can gum up the screen to the point of obscuring the work area itself.

One rule that software developers might follow is to completely redesign the user interface of any application that has passed through three successive versions. In that way, what a computer application looks like can be made to conform more comfortably, accurately, and attractively to what it does.

WordPerfect has achieved success in the market thus far by offering users more speed and flexibility than other wordprocessing packages offer, but at some point in the near future, the company should take another long, hard look from the vantage point of the driver's seat.

To the student driver, at least, *WordPerfect*'s controls are beginning to look too complicated to use—and the danger in that is that soon, students may become too discouraged even to try.

* * * * * * * * * * *

James Guthrie is a member of the Department of English at Wright State University in Dayton, OH 45435. He may be reached either by voice at 513-873-2472 or via e-mail at JGUTHRIE@WRIGHT.EDU

SOFTWARE REVIEW

George Kren

German Assistant

Program:	*German Assistant*
Company:	MicroTac Software
Address:	4655 Cass Street, Suite 214 San Diego, CA 92109
Phone:	800-366-4170
FAX:	619 272-9734
Price:	$79.95 + $4.00 s/h
Includes:	three 5.25" disks, two 3.5" disks, and user's manual
System:	IBM or compatible computer, DOS 2.0 or above, 512K RAM, hard disk (uses approx. 1.2 megabytes of space)
Other:	Memory residents reference tools require approx. 130K of RAM.

Years ago one could identify engineering students because to a man (a woman engi–neering student was almost nonexistent twenty years ago) they had slide rules attached to their belts. I doubt that any current engineering student would even know how to use one.

The slide rule has been replaced by the electronic calculator. Historians and others in the human–ities, having no mechanical or electronic instru–ment, learn languages which enable them to read sources. Four programs (for German, Spanish, Italian, and French) from Microtac Software now bring the computer to the task of translating. These programs will not eliminate the need for learning a language. In physics one may establish an exact equivalence for different standards of measurement. The metric equivalent of one inch can be calculated to any required numbers of decimal places. No such equivalence exists for words and phrases between languages. Not only that, but the grammatical structures differ widely.

The emotional overtones of words do not readily translate. ***Heimat*** is not identical to the English dictionary rendering of "country." Hence, the sentence by sentence translations of this program are, though an interesting beginning, not yet functional. It can handle simple sentences well enough, but as soon as these become more complex, the program makes a mess of things. the program does not claim to do more than translate simple senctences from English to German.

The sentence **The program does translate simple sentences** is translated as:

> ***Das programm übersetzt einfache Sätze.***

However, as soon as the sentence becomes more complex, difficulties develop. Thus:

> **This program has the capacity of translating simple sentences from German to English, but makes a mess of complex ones,**

becomes

> ***Dieser Programm hat die Fähigkeit übersetzen einfache Sätze aus Deutsch Englisch, aber macht ein Schlamassel komplexen eine.***

The meaning of the sentence is garbled and there are problems with grammar, gender, and word order. The case for **this** is incorrect. It should be ***Dieses***. The word order needs to be changed to

einfache Sätze zu übersetzen, and the last phrase requires reorganization.

Jack and Jill went up the hill, to fetch a pail of water

becomes

Buben und Jill ist gegangen hinauf der Hügel, einen Eimer Wassers zu holen.

The fault for translating Jack as ***Buben*** lies with the dictionary, which translates Jack as ***Wagenheber*** (automobile Jack and Jack, as in Jack of spades) but does not list it as a first name. The word order as well as the grammatical constructions contain errors. The sentence should read ***Jack*** (or ***Hans***) ***sind auf den Hügel gegangen, um*** (um translates the "to" with the sense of in order to) ***einem Eimer Wasser*** (for some reason the program added an s to ***Wasser***) ***zu holen.***

The greater the complexity, the more the problems encountered by the program.

A year ago Charles Beard contended that the ideal of impartiality was an impossibility.

becomes

Ein Jahr vor Charles Bart hat behauptet, da das Ideal Unparteilichkeit war eine Unmöglichkeit.

That **Beard** is translated as **beard** is not serious, though it should be possible to program a recognition of proper names. The inability to "render a year ago" is due to a problem with word order and structure and destroys the meaning of the sentence. The significant divergences in word order be–tween English and German can confuse even simple sentences:

I must show her the book.

becomes

Ich muß zeigen ihr das Buch.

Obviously, the verb ***zeigen*** should be at the end of the sentence. The program's rendering is awkward, but the sense is still recognizable.

Given the complexities of translation, this program marks an interesting beginning, but clearly substantially more work is required before we can call such a translation program adequate. This program also contains some language tools. The *Random House Dictionary*, which is part of this program, contains some 40,000 entries.

A larger dictionary would be helpful. Even with its limitations it is useful to be able to call up the dictionaryfrom within a wordprocessing program. Clearly significant improvements in translation programs will require the use of artificial intelligence.

Several useful tools which can be made memory resident (they conflict with *Sidekick*, but then what doesn't) provide declensions of nouns and contain an eminently useful verb conjugator. With a touch of an arrow key a verb can be changed into the subjunctive, into the future, into the past perfect, etc. A minor but convenient feature is the ability to include words with accent marks (***Umlauts***, sharp *s*'s) (*ß*) into the text. The accent entry utility provides a quick and useful way of adding diacritical marks or sharp *s*'s in German) (*ß*) into the text. However, even after translating a *German Assistant* file into ASCII, my printer (a Toshiba P321) would not accept the ***Umlaut***'s.

As Samuel Johnson remarked about a dog walking on its hind legs, one is so surprised that he can do it, that one does not ask how well. Given the immense difficulties of computer translation, this program marks, with all its flaws, a welcome beginning; and the auxiliary tools are useful.

* * * * * * * * * *

Professor George M. Kren teaches history at Kansas State University in Manhattan, Kansas 66506 and is the editor of the internationally known *History Microcomputer Review*.

TEXTechography

Arthur A. Molitierno

Abbreviations:

Countries

AU	Austria
CZ	Czechoslovakia
GR	Germany
JP	Japan
NL	Netherlannds
SP	Spain
SW	Switzerland
UK	United Kingdom

Terms

DTP	Desktop Publishing
np	no page
n.	number

Months

JA	January
FE	February
MR	March
AP	April
MY	May
JE	June
JL	July
AU	August
SE	September
OC	October
NO	November
DE	December

[*Editor's Note:* TEXT Technology *welcomes bibliographic items of interest from the entire community of text producers. Send items to Arthur A. Molitierno.*

Each issue will feature a bibliographic form employed by a specific orgainzation, association, or journal. This issue features the format for the Publication of the Modern Language Association.

Although there will be some notable differences from the original style (italicizing the names of software or journals, for instance), "TEXTechography" will present the featured style as close as possible to the original's appearance.]

Anand, V., reviewer. "Review of Richard Friedhoff and William Benzon's *Visualization: the second computer revolution.*" *Choice* 27 (FE 1990): 977. [180 word review indicates text is well written, containing over 200 color and black and white illustrations; text depicts the basic issues surrounding visualization which are both technological and philosophical]

Assadi, B. "Review of Full Write Professional 1.5." *Infoworld* (FE 1990): 64. [program is intended for Macs; combines wordprocessing, page layout, draw graphics in WYSIWYG format; Full Write Professional's strong layout features support nine columns per pages; while speed is not substantial, a new feature allows the moving and resizing of sidebars—independently-formatted documents—on the page without accessing a dialog box]

Bernhardt, S.; et al. "Teaching college composition with computers: a time observation study." *Written Communication* 7.3 (JL 1990): 342-74.

Blechman, F. "Review of Express Publisher 2.0 [DTP]." *Computer Shopper* (MY 1991): 441-443. [comfortable GUI—graphical user interface; the number of functions and features requires keeping the manual nearby; contains new feature "TextEffects" for special typesetting operations]

Cavuoto, S.; Cavuoto, J. "Desktop publishing for the design firm." *Civil Engineering* 60 (NO 1990): 63-64.

Eckhouse, R. "Quality displays [Adobe Type Manager]." *Computer* 24 (FE 1991): 89.

Egol, L. "Color publishing systems boast a richer palette." *Chemical Engineering* 98 (FE 1991): 169-70+.

English, A. "Review of Express Publisher 2.0 [DTP]." *Personal Publishing* (MY 1991): 47-50. [easy to use and suited for flyers, brochures, and documents up to 32 pages; interface—Windows similar—could use improvement; program needs to gray our inactive menu items; poses some difficulty when using a Microsoft serial mouse]

Garza, V.; Kvitka, A.; Zittle, T. Eva, E. "Review of TextPert, 3.0 for Mac and 1.1 for IBM." *Infoworld* (OC 1990): 79, 83, 87. [indicates that program's ability to distinguish and separate text and graphics along with searching for text strings is a major plus but overall performance is disappointing, with a poor rate for accuracy]

Gruman, G.; Lombardi, J.; Azinger, E.; Exkert, J. "Review of Full Write Professional 1.5." *Infoworld* (AU 1990): 59-69. [strong points—program offers footnotes, endnotes, and bibliographic formatting; does not allow for enlarging text in editing mode as in PageMaker 4.0; reviewer finds Microsoft offers more while Wordperfect offers less]

Holzinger, A. "Review of Grammatik IV." *Nation's Business* (SE 1990): 39. [program identifies misspellings, split infinitives, incorrect punctuation, and subject-verb agreement; easy to install and use; operates on the fly wwith popular wordprocessors: WordPerfect, Microsoft Word, Wordstar, Professional Write, MultiMate, and XyWrite]

James, M. "Review of Express Publisher 2.0 [DTP]." *PC Today* [UK] (AP 1991): 93-94. [useful on-disk introduction and a complete tutorial in the manual; notable features include "Align"—for lining up two objects—and "Equate"—for making two objects the same size]

Kleinholz, L. "Review of American Heritage Electronic Dictionary 1.0." *Home-Office Computing* (AP 1991): 66-67. [finds the thesaurus a most welcomed addition to the speller of wordprocessing programs; toll-free telephone support is good; program superior to Definition Plus! since American Heritage gives full definitions]

LaBadie, H. "Review of TextPert, 3.0 for Mac and 1.1 for IBM." *Computer Shopper* (DE 1990): 413-414. [TextPert, an OCR—Optical Character Recognition—program designed to read any type of text; reviewer finds a significant problem in program's misidentification and the inability of user to correct the problem; compares this program to other page recognition programs such as WordScan Plus, OmniPage, AccuScan, ReadRight 2.0, and Recognize! 2.0]

Lewis, D. "Review of Publish It! Easy 2.01 [DTP]." *Macuser* (SE 1990): 61, 63. [indicates menus and dialog boxes are cleaner than earlier version; speed and ease of use is a strong point; recommend purchasing if more powerful programs such as PageMaker are not needed]

Litten, C. H. *The effects of wordprocessing and peer review on the revision process of freshman composition students.* PHD dissertation. University of Maryland, College Park, 1989. [the study attempts to 1) analyze the relationship between teacher and peer criticism in the revision process and 2) describe the effect of wordprocessing and non wordprocessing tools on the revision process; the study is of the same population of 107 students and concludes that wordprocessing has no effect on the final outcome of the quantity or quality of revised texts]

Lynch, T. "The DocuTech [Xerox] Product Publisher." *Design News* 46 (DE 1990): 56-61.

McCarthy, J. A. *Freshman writers and word processors: case studies in the use and effects of computers in the revision practices of college composition students*. PHD dissertation. Miami University, 1989. [the study concludes that while the computer can alter writing and revising strategies, the major factors in altering outcomes through computer-assisted writing are 1) familiarity and 2) reaction to the machine]

Moad, J. "Xerox's enterprisewide printer." *Datamation* 37 (JA 1991): 79-80.

Morrison, P., reviewer. "Review of Richard Friedhoff and William Benzon's *Visualization: the second computer revolution*." *Scientific American* 262 (JE 1990): 136. [850 word review which indicates the text is meant for the general public; comprehensively reviews techniques in visualization; includes essay on visual perception]

Nielsen, L. "Review of American Heritage Electronic Dictionary 1.0." *Technology & Learning* (FE 1991): 6. [program is easy to install, convenient to run with word processors, broad range of features; finds fault with search procedure which when accomplished destroys the search criteria; size of program—3 megabytes—may hamper some users with small hard drives but may be especially useful for ESL classes—English as a second language—and journalism classes]

Parker, J. "Review of TextPert, 3.0 for Mac and 1.1 for IBM." *PCM* (AP 1991): 60. [considers this program a major addition to Optical Character Recognition—OCR—programs; indicates program is fast, friendly, and powerful as well as easily usable within Windows; speed of recognition is listed as 5,000 characters per minute]

Rasmus, D. "Review of Full Write Professional 1.5." *Macuser* (AP 1991): 83. [easy to use but lacking in essential features such as smaller graphic sizing and direct support for More II and 3.0 outlines; conversion from popular wordprocessors is not convenient]

Rohan, R. "Review of American Heritage Electronic Dictionary 1.0." *Computer Shopper* (JA 1991): 394, 402-403. [indicates ease of installation, ease of use and menu; considers features such as SearchText and Anagram worthwhile; while package is strong, it still has room to grow]

Rosenbluth, G. S. *The affects of writing process-based instruction and wordprocessing on remedial and accelerated eleventh-graders*. EDD dissertation. West Virginia University, 1990. [this 16-week study of 38 remedial students and 29 accelerated students indicates that while computers motivate remedial and accelerated students, the impact of computer-assisted writing is still questionable in terms of the quality of the outcomes]

Siciliano, R. J. *The effects of wordprocessing instruction on the writing styles and achievement in higher-level English courses of beginning college writers*. EDD dissertation. The George Washington University, 1990. [comparison of computer-assisted composition and traditional composition to determine the outcome for students; while students using computers have a positive view of using such devices, the study concludes that there is no great difference in writing styles between those using computers and those not; those using computers, however, became less spontaneous but more linear and solitary in their writing]

Sirc, J.; Reynolds, T. "The face of collaboration in the networked writing classroom." *Computers and Composition* 7 (AP 1990): 53-70. [this special issue includes papers from the 5th Computers and Writing Conference]

Smith, C. A. *The effects of handwriting versus wordprocessing on learning-disabled students' written compositions*. EDD dissertation. Columbia University Teachers College, 1989. [this study of 24 secondary students with learning disabilities concludes that although systematic instruction in both typing and wordprocessing was supplied to the students, the students' writing showed improvement in only the area of fluency and spelling when compared to those students how employed handwriting; no indication of improvement in quality is indicated in this study]

Somerville, D. "Microsoft Works 2.00." *Computer* 24 (FE 1991): 86-87.

Storm, B. "Review of Grammatik IV." *Computers in Accounting* (SE 1990): 73. [corrects grammatical mistakes faster than Correct Grammar and Right-Writer; sometimes reformats a document when using Microsoft's Word]

Svacina, J. M. *The computer as a tool in teaching basic writing to college freshmen*. PHD. dissertation. University of Illinois at Urbana-Champaign, 1988. [this study of 93 college students concludes that computer-assisted instruction in basic writing (which emphasizes grammar and lower-order writing skills) does not significantly enhance the beginning or developmental writer]

Thompson, T. "Create and lay out documents with Taste [MAC application]." *BYTE* 16 (JA 1991): 130.

Vaughn, F. "Color WYSIWYG comes of age." *BYTE* 15 (DE 1990): 275-77+.

Vogt, C. "Powerful wordprocessing from Samna's Ami." *Design News* 46 (DE 1990): 162.

Vovcsko, J. "Review of Express Publisher 2.0 [DTP]." *PCM* (AP 1991): 66-67. [ne feature "TextEffects" allows user to fill polygons with text or wrap letters around curves; Zoom-in feature doubles the size of document for easy editing of small print; excellent text reproduction for dot-matrix printers]

Widman, J. "Review of Publish It! Easy 2.01 [DTP]." *Publish* (SE 1990): 112, 114. [this program offers more features than Springboard Publisher II; documents easy to create and sample templates are included; advanced features are not easy to use]