# Software

## News from the VORTEX Project

Michael A. Harrison*
Computer Science Division
University of California, Berkeley

### Introduction

This is a report on the current state of the VORTEX project as it relates to the TEX community. The project has been running for over three years. Our research funding has been extended, but we will be evolving in non TEX-compatible ways. We expect to publish traditional research papers on the novel aspects of our work. Cf. [7] for a summary of a number of issues underlying this work. Design issues are discussed in [6].

This brief summary will tell you what software we will have available for use by the research community; it can be licensed commercially as well. By the time you read this report, all of the systems discussed should be available.

The original goal of the VORTEX project was to create an integrated document preparation environment capable of producing high quality technical documents which involve mathematics, text, and graphics. In VORTEX, both source and target representations of a document are maintained and presented. The source representation refers to a TEX document in its original unformatted form; the target representation presents its formatted result. The user can edit both representations using a text editor and what is called a proof editor, respectively. Our editor is an Emacs-like editor written in our own VLisp (VORTEX Lisp). Changes made to one representation propagate to the other version automatically. It is easy to support source modifications that cause the target representation to change, but a major challenge was the transformation from a target version back to the source version. The original design ideas are described in [6]. The system reformats a document and redisplays it on the screen incrementally. Only the part of the document or the subregion of the screen that is affected by recent changes is reprocessed. The document environment has support for automatic production of indexes for books [8], support for bibliographic material [3], spelling checkers, and other document-related utilities. Cf. [5].

Our current system uses workstations with bitmap displays running under the UNIX operating system and the X window system [12]. Some subsystems also support SUNVIEW. In order to provide device independent high quality graphics, we use PostScript [1,2] and have written a PostScript interpreter. We are developing methods to interpolate PostScript into our displays. Future research plans include supporting composite objects, symbolic mathematics, hypertext documents, audio, live video, etc. A key element will be access to a persistent object base which is important for many applications.

The VORTEX prototype is running and did pass the traditional Trip test [10], as well as the more demanding Trip* test required of incremental document processors [9]. Because the TEX processor is based on Pat Monardo's C-TEX, the system is fast. Some timing data are reported in [9]. The most challenging part of VORTEX is the reverse mapping mechanism. To handle some of the semantic difficulties, functions in VLisp are used to implement reverse mapping. Only a few are implemented in the current prototype.

The VORTEX prototype will be available on the distribution sometime in January 1989. As the system currently stands, it will require a great deal of work to make into an industrial-strength system. It served our purposes admirably because we found out what we did right (and wrong) and this will influence the design of its successor.

We have produced another system called INCTEX, derived from VORTEX, which we think will be useful immediately to the TEX community. INCTEX is an incremental TEX processor.[1] It is very fast and is more efficient in its use of memory than VORTEX. It supports quiescence checking, convergence testing, etc. (cf. [9] for definitions). It creates a DVI file and checkpoints some state information for each page on the first run. On subsequent runs, only those parts of the document which have changed are reprocessed. When running LATEX incrementally, no unnecessary passes are used. INCTEX, unlike VORTEX, is editor independent and is designed to avoid parts of the operating system which would hamper porting.

INCTEX has not passed the Trip* test as of this writing, but works well enough to process many research papers and a 200-page dissertation. INCTEX

*Sponsored by the Defense Advanced Research Projects Agency (DoD), monitored by Space and Naval Warfare Systems Command, under Contract No. N00039-88-C-0292. Additional funding came from the California MICRO program (Grant 88-083 in conjunction with IBM).

[1] Any flavor of TEX you want, not just plain TEX.

should be a very valuable extension to the family of TEX processors.

### Graphics

PostScript was chosen as the language for specifying graphics [2,1]. We have a PostScript interpreter which runs under X10(R2), and under SUNVIEW. This interpreter is available for distribution. The system involves a base interpreter and separate programs for interface to various window systems. Some of the algorithms used are original and will be published. While the interpreter is still only a prototype, it has been used in some commercial implementations, and can be extended and optimized. Perhaps some Berkeley MS students can be induced to do the remaining tasks (like porting to X11(R3)), but we have neither the resources nor the mission to develop commercial-level software. Several vendors have more robust PostScript systems, but we will distribute our source code and encourage you to improve on the interpreter.

### Fonts

As part of the PostScript interpreter project, we needed outline fonts. It was possible to write a PostScript program which calculates the coordinates of the points on the outline curves for any font which is native to or may be downloaded into a PostScript printer. From this data, one can automatically construct outline fonts using well known spline techniques. Thus, we have available 35 outline fonts which are on the distribution, together with the supporting software. These fonts use our own outlines but are *congruent* to some well known fonts. Table 1 lists equivalents for the more interesting outline fonts.

Having outline fonts is important, not only for the use of the interpreter. Of course, you can do illustrations like Figure 1. The A's in Figure 1 are from our t-rom font.

We can generate a full set of `pk` fonts for Table 1 for use with our previewers. We have an `awk` script which converts our outlines into the `.cf` format used by the TYPO editor [13]. This editor, written by Jakob Gonczarowski and marketed by Typographics, Inc., is very useful. Among other features, it converts between various descriptions and font formats. Thus we can automatically convert from our outlines into Metafont files and hence into your favorite format.

### Other Systems

There are many other subsystems available and a brief outline of some of them is given below.

**Table 1**: Font Equivalents

| Berkeley Fonts | Old Favorites |
|---|---|
| ag-book | AvantGarde-Book |
| ag-bookobl | AvantGarde-BookOblique |
| ag-demi | AvantGarde-Demi |
| ag-demiobl | AvantGarde-DemiOblique |
| b-demi | Bookman-Demi |
| b-demiita | Bookman-DemiItalic |
| b-lig | Bookman-Light |
| b-ligita | Bookman-LightItalic |
| h-bol | Helvetica-Bold |
| h-bolobl | Helvetica-BoldOblique |
| h-med | Helvetica |
| h-obl | Helvetica-Oblique |
| ncs-bol | NewCenturySchlbk-Bold |
| ncs-bolita | NewCenturySchlbk-BoldItalic |
| ncs-ita | NewCenturySchlbk-Italic |
| ncs-rom | NewCenturySchlbk-Roman |
| p-bol | Palatino-Bold |
| p-bolita | Palatino-BoldItalic |
| p-bolobl | Palatino-BoldOblique |
| p-ita | Palatino-Italic |
| p-obl | Palatino-Oblique |
| p-rom | Palatino-Roman |
| t-bol | Times-Bold |
| t-bolita | Times-BoldItalic |
| t-bolobl | Times-BoldOblique |
| t-ita | Times-Italic |
| t-itaun | Times-ItalicUnslanted |
| t-mathita | Times-MathItalic |
| t-obl | Times-Oblique |
| t-rom | Times-Roman |
| zc-medita | ZapfChancery-MediumItalic |
| zd | ZapfDingbats |

### Screen Previewers

`DVItool` is a previewer for `DVI` files which runs on the SUN workstation. This system is very robust, handles arbitrary `DVI` files, and provides a great many features. It is a full tool in the sense of the SUN window system and can be adjusted to any size the user finds appropriate. It is possible to keep a small window on the screen for previewing at the same time a source window is present. This is extremely valuable in debugging. Changing the view you have of a page is instantaneous. All magnifications are supported. Forward and reverse searching for strings in the `DVI` file is implemented as well as the ability to select a character and display its font The previewer may be customized.

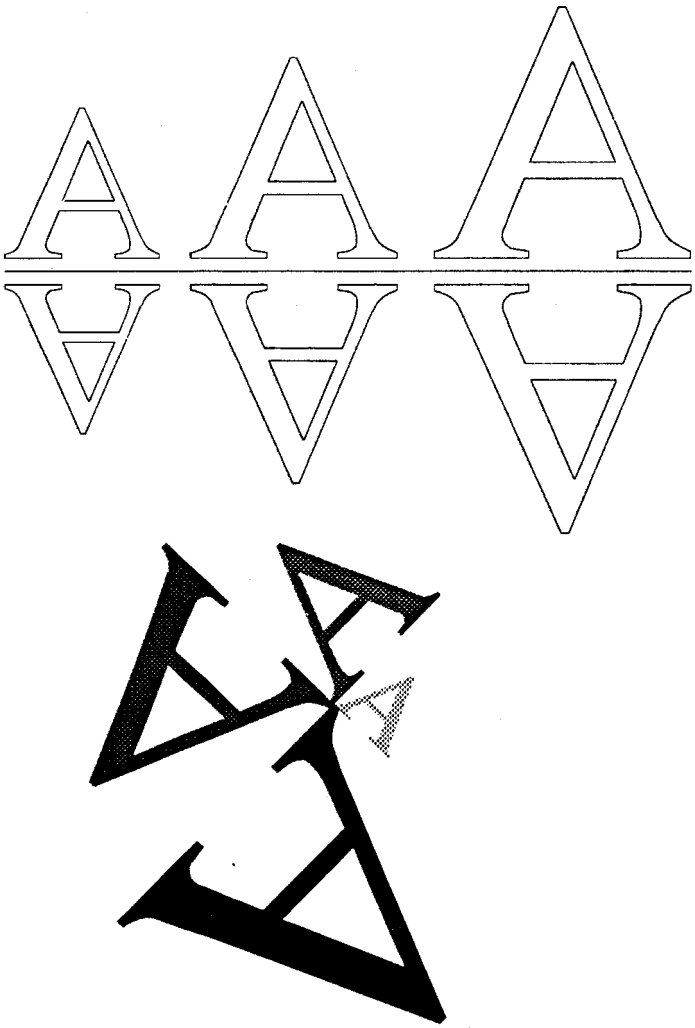

**Figure 1**: Some advantages of outline fonts.

DVI2x is a previewer for DVI files which runs under the X10 window system. It has a number of the same features as DVItool such as customizable key bindings. There are user definable functions, rulers, and side-by-side display of two pages. This program is quite fast.

DVI2X11 is a previewer for DVI files which runs under the X11(R3) window system. It is similar to DVI2x but is more than just a simple adaptation. The graphics have been improved and the scroll bars are draggable. This version is now faster than texx.

### Environments

We distribute a large LISP library for use with emacs. We work with the gnuemacs version. This is intended for use with TEX, LATEX and BIBTEX. In addition to the most elaborate TEX mode known [4], the user is assisted in creating .bib files, and given forms-based assistance in filling in bibliographic entries [3]. Functions exist for copying and duplicating fields from previous entries, etc. One particularly useful option is preparing a draft bibliography which includes numerical references, symbolic references and a formatted version of the entries. Another of the options allows previewing on the SUN or printing on any of your local printers. This system interfaces nicely with the DVI display programs mentioned above. It takes many pages of single-spaced text to explain all the options. Cf. [4,3,8]. There is also support for dealing with indices and for online-reference inspection. [11]

### Licensing

Since we are a research project, we publish our work in scholarly publications and distribute our research software at a nominal charge. We have established site licenses of various kinds to simplify acquisition of the software.

We do ask that researchers who have found bugs or make significant extensions let us know about them.

In the past, commercial licenses have been negotiated separately, but this has proven to be too costly for us in terms of time. To simplify matters, we will now license the entire distribution for a fixed fee without royalties

We are adding the new software to the distribution at this time and hope that it will be complete by the end of January 1989.

### Credits

The following people have written or made major contributions to the software being distributed: Pehong Chen, John Coker, Michael A. Harrison, Paul N. Hilfinger, Dan Hydar, Jeffrey W. McCarrell, Ikuo Minakata, Pat Monardo, and Steven Procter.



### References

[1] Adobe Systems, Inc. *PostScript Language Manual.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1985. ISBN 0-201-10174-2.

[2] Adobe Systems, Inc. *PostScript Language Manual: Tutorial and Cookbook.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1985. ISBN 0-201-10179-3.

[3] Pehong Chen. Gnu emacs BIBTEX-*mode*. Technical Report 87/317, Computer Science Division, University of California, Berkeley, California, 1986.

[4] Pehong Chen. Gnu emacs *TEX-mode*. Technical Report 87/316, Computer Science Division, University of California, Berkeley, California, 1986.

[5] Pehong Chen, J. Coker, Michael A. Harrison, J. McCarrell, and S. Procter. An Improved User Environment for TEX. In J. Desarménien, editor, *TEX for Scientific Documentation, Proc. 2nd European Conf. Strasbourg, France. June 19–21*, 1986. Lecture Notes in Computer Science, No. 236, pp. 32–44 (1986).

[6] Pehong Chen, John L. Coker, Michael A. Harrison, Jeffrey W. McCarrell, and Steven J. Procter. The VORTEX document preparation environment. In J. Desarménien, editor, *TEX for Scientific Documentation, Proc. 2nd European Conf. Strasbourg, France. June 19–21*, 1986. Lecture Notes in Computer Science, No. 236, pp. 45–54 (1986).

[7] Pehong Chen and Michael A. Harrison. Multiple representation document development. *IEEE Computer*, 21(1):15–31, January 1987.

[8] Pehong Chen and Michael A. Harrison. Index preparation and processing. *Software, Practice and Experience*, 18(9):897–915, September 1988.

[9] Pehong Chen, Michael A. Harrison, and Ikuo Minakata. Incremental document processing. In *Proceedings of the ACM Conference on Document Processing*, New York, December 1988. Association for Computing Machinery.

[10] Donald E. Knuth. A torture test for TEX. Technical Report STAN-CS-84-1027, Computer Science Department, Stanford University, Stanford, California, November 1984.

[11] Ethan V. Munson. Enhancements to Bibliography Management. Master's thesis, Computer Science Div., University of California at Berkeley, 1988.

[12] Robert W. Scheifler and Jim Gettys. The X window system. *ACM Transactions on Graphics*, 5(2):79–109, April 1986.

[13] Typographics, Ltd., Jerusalem. *TYPO Font Editing System, Reference Manual*, 1988.

## An Enhanced TEX-Editor Interface for VMS

Adrian F. Clark
Essex University

The author described various enhancements to the Stanford distribution of TEX under VAX/VMS in a couple of previous articles. The most significant of these was to allow an editor to be invoked (by typing 'e' or 'E') when TEX spotted an error in its input file. The editor which was interfaced to TEX at that time was TPU. Since then, a number of requests have been made for interfaces to other editors, usually EDT. The author is pleased to announce that this has now been done, in a way which is fully compatible with the previous editor interface. This note discusses the interfaces to the editors usually encountered under VMS and shows how other editors can also be used.

VMS is supplied with a number of editors: EDT and, latterly, TPU are the most widely used, but Real Programmers and people who have used other DEC systems sometimes prefer TECO; those of us who remember the bad old days under VMS 1 will probably be familiar with SOS (which, although unsupported, can still be acquired through DECUS). There are also layered products which provide editors, such as LSE (language-sensitive editor) and a UNIX-compatible ed in DECshell. And, of course, there are third party and public-domain products — for example, Gnu Emacs and STE, the *Software Tools* editor.

The basic strategy for invoking an editor from TEX is (in VMS jargon) to spawn a sub-process which executes a DCL command to edit the erroneous TEX input file. When the corrections have been made, the editor is exited, the sub-process deleted and control returned to the TEX session. However, sub-process creation and deletion is rather slow under VMS, so invoking editors in this way can give an unsatisfactory response on systems with a significant load. For this reason, both TPU and EDT are *callable*, i.e. they can be invoked as procedures from TEX; this gives a much better response.

To decide which editor to use, TEX looks at the logical name TEX$EDIT (by analogy with MAIL and other utilities). This should translate to the name of the editor to be used (see below). If the logical name is not defined, TPU is selected, for compatibility with the previous version of the editor interface. Following selection of the editor, TEX translates the logical name TEX$EDIT_INIT, which should give the initialisation file for the appropriate editor. (For TPU, if TEX$EDIT_INIT does not exist, TEX translates TEX$TPU_INI, again for compatibility with the