## A Screen Previewer for VM/CMS

Don Hosek

A good previewer is a useful tool for working with TEX, but unfortunately, there are very few available. For users of TEX under the IBM VM/CMS system, the only choice available used to be DVI82, a Versatec driver that, as an added feature, allowed previewing on IBM 3279 and 3179-G terminals.

To deal with this situation, I wrote DVIview, a TEX previewer that displays its output on VT640-compatible displays connected to an IBM mainframe via either a 3705 controller or a Series-1/7171 protocol converter. In addition, the output routines are modularized enough that it should be a fairly simple task to modify the program to drive any graphics terminal connected to the mainframe. (I have plans to include support for GDDM-driven displays in the near future.)

DVIview is a lengthy WEB program that interprets the instructions in a DVI file and displays them on the user's screen as determined by commands typed at the keyboard. The entire page may be viewed with block outlines of the characters, or smaller portions of the page may be selected and viewed using the actual shapes of the TEX fonts. Font information is read from PK files. (I cannot recommend the PK format enough to people writing new device drivers; the fonts take roughly half the space of GF files and about a third the space of PXL files. And PK readers are easier to write!)

The DVIview distribution includes two manuals: "Previewing TEX Output With DVIview" is the users' guide and explains how to use the program from a user's standpoint. Also included is "Installing and Customizing DVIview", intended for the systems person who installs DVIview. Instructions are given for installing DVIview as is, as well as instructions on adding changes to the file and a "Hitchhikers' Guide to WEB" (for those who don't care how they get where they're going as long as they don't have to ride the bus).

Due to the size of the program, it cannot be distributed over the networks. To obtain a copy of DVIview and its documentation, send $30 (to defray duplication costs), a blank tape, and a return mailer to:

Don Hosek
Platt Campus Center
Harvey Mudd College
Claremont, CA 91711

The program is public domain, so feel free to give it away. However, since it is still a young program, I'd like to keep track of who has copies for purposes of distributing updates.

This article first appeared in TEXmag, 1987#7.

## Why TEX Should NOT Output PostScript — Yet

Shane Dunne
University of Western Ontario

In a recent TUGboat issue [1], Leslie Lamport suggested that since PostScript is becoming accepted as a standard page description language, perhaps TEX could be modified to output PostScript instead of DVI code. This is a good idea, but it should *not* be done yet for the following reason: At the moment, the available PostScript literature does not state precisely how drawn objects are to be rendered on the output raster. As I will show in this article, such a specification of PostScript's semantics is urgently needed to allow precision application programs such as TEX to properly use the language. I have written to PostScript's developers, Adobe Systems Inc. of Palo Alto, California, to draw their attention to this problem, and suggested that it be resolved publicly using TUGboat as a forum for discussion.

For readers unfamiliar with the PostScript language, a few words of explanation are in order. PostScript is a language designed specifically for specifying the output of raster printing devices. The language is interpreted, with the interpreter usually resident in the printer itself. It was intended to be human-readable, and hence uses only printable ASCII characters, but to simplify parsing it uses a rather cryptic postfix syntax. This is justified on the grounds that most PostScript programs will be written automatically as the output of other applications. PostScript incorporates a sophisticated device-independent drawing model in which a single transformation matrix (called the *current transformation matrix* or CTM) specifies the correspondence between the user and device coordinate systems. User coordinates are floating-point numbers with essentially infinite resolution; device coordinates are normally integers.

The incompleteness of the current PostScript semantic definition is apparent from the following example. Assume that the CTM of a PostScript device is set so that one unit in user space corresponds