Vndblt (Vanderbilt University): H. Denson Burnum, 615-322-2357
WashStU (Washington State University): Dean Guenther, 509-335-0411
Wzmn (Weizmann Institute, Rehovot, Israel): Malka Cymbalista, 08-482443
Yale: Bill Gropp, 203-436-3761

Index to Sample Output from Various Devices

Camera copy for the following items in this issue of TUGboat was prepared on the devices indicated, and can be taken as representative of the output produced by those devices. Some items (noted below) were received as copy larger than 100%; these were reduced photographically using the PMT process. The bulk of this issue, as usual, has been prepared (all with TEX82) on the DEC 2060 and Alphatype CRS at the American Mathematical Society.

- Apple LaserWriter (300 dpi):
  Textset advertisement, p. 103.
- Canon CX (300 dpi):
  Metafoundry advertisement, p. 100.
- Epson LQ1500 (180 dpi):
  Norman Naugle, *An elementary sum*, p. 70; TI/PC running PC TEX.
- QMS Lasergrafix 800 (300 dpi):
  Norman Naugle and Tomas Rokicki, `\output= . . . \random`, p. 71; TI/PC with PC TEX.
  Gregory Marriott, *A TEX82 implementation on the HP9000 Series 500*, p. 80.
  MicroTEX advertisement (Addison-Wesley), p. 102; IBM PC using MicroTEX.
- QMS Lasergrafix 1200 (300 dpi):
  Michael J. Ferguson, *Multilingual TEX*, p. 57; VAX 11/780 (VMS).
- Toshiba P351 (180 dpi):
  PC TEX advertisement, p. 104; IBM PC/XT using PC TEX.
- Versatec (200 dpi): Hans Riesel, *Report on experience with TEX80*, p. 76; reduced from 130%; TEX80, DEC-20.
- Xerox Dover (384 dpi): Amy Hendrickson, *Some diagonal line hacks*, p. 83.

GRAPHICS COMMANDS FOR TEX DISCUSSION IN TEXHAX CONFERENCE

Alan Spragens
Stanford Linear Accelerator Center

The TEXhax network conference carried a number of comments concerning graphics and TEX during a period from about a year ago until about six months ago. Then the discussion petered out, presumably because no consensus was reached. My file of mail items comprising this discussion runs to 53 printed pages.

I wrote the following description of parts of that discussion as a memo to a committee at SLAC investigating how we might best create merged text and graphics on our computer systems. Although we have been creating such documents experimentally for some time in a variety of ways, it has required hacking. We're on the track of methods applicable to a variety of systems and devices, usable by our community of hundreds of physicists who do their own papers. I tried to give a flavor of the discussion and mention some ideas that seemed important to me rather than a summary, thinking that more interested parties should get hold of the actual material that came over the wire. Accordingly, I don't include here mention of important contributions from some of the main participants in the discussion, such as Todd Allen and William LeFebvre, and I hope they'll pardon the omission.

The TEX Project's "party line" on why the TEX language and `DVI` (TEX "device-independent" output) format lack graphics commands was stated by David Fuchs a year ago: (1) there is no way to provide the capabilities in a device-independent manner, and (2) the world lacks a standard, comprehensive, accepted language for describing computer graphics. Dave mentioned that TEX's designers recognized the need for graphics capabilities in a language specifying the appearance of a printed page, so they included the `\special` command for extending the language for just such a purpose. He exhorted people to consider the long range view, beyond present technologies, rather than dash off a "standard" that would be unsatisfactory in a couple of years, *e.g.*, consider shading, halftones, splines, color, *etc.* Since this "party line" message came over the net a year ago, I called David last week to ask if anything had changed. He said nothing had changed, that they had hoped that "Adobe would take over the world by now," but it hadn't. He also mentioned that a number of sites, including Stanford, had implemented various graphics languages

via \special that were device-dependent and site-specific. He suggested not limiting ourselves to a particular language, but allow for inclusion of a standard (*e.g.*, put a "tag" such as slac into our commands). One suggestion that came up in the TEXhax discussion was to establish a "registry" of graphics commands under the ægis of the TEX Users Group to avoid incompatibilities, at least within the TEX community. Dave said such a registry had not been established to his knowledge.

The TEXhax discussion centered around commands for producing line graphics. There was little discussion of commands for inserting external files or of the more esoteric graphics functions such as color and shading. I think I can fairly say that the following line-graphic operators were generally agreed to be desirable: **line**, **circle**, **ellipse**, **arc**, and **spline**.

Two "systems" of defining these commands emerged, which I'll describe shortly. There was some discussion as to how the graphics commands should be implemented in TEX, that is, how macros to resolve them into \special's would work, which I'm leaving out. (That information would be of interest to macro writers, however—contact me if you want it.) A suggestion that new TEX commands (instead of \special) be implemented for graphics was generally rejected.

Paul Grosso described a system wherein the different graphics capabilities of different devices are resolved by "tags" in the graphics language, *e.g.*, \special{FOO:ABC}\special{BAR:XYZ} where FOO and BAR are devices and ABC and XYZ are differing graphics commands for the two devices. This caused a number of complaints that TEX's device independent philosophy was thereby violated. It was suggested that DVI-to-device translator programs should *ignore* graphics commands unacceptable to target devices, rather than treat them as errors.

There was some discussion of *pen* specification; generally the people concerned with pens seemed to want them defined in terms of shape and size, and they wanted to be able to store pen definitions, perhaps by name or number, and recall them later. A suggestion that METAFONT-style pen definitions be adopted was favorably commented upon by several people. Such a definition might look like \special{pen 5 ellipse 2pt 1pt} for an elliptical pen shape whose width is 2 points, whose height is 1 point, and whose definition is stored as pen 5. Some people advocated that pen shapes be characters in a font; I didn't quite grasp the significance of this suggestion.

Regarding the set of graphic operators to be accepted, two distinct systems emerged in the discussion. One was called the *delta* system; the other was called the *join* system. I believe the salient characteristic of the *delta* system, as proposed by Pierre MacKay, is that graphics coördinates are given in units relative to the "current" position on the page. In the *join* system, coördinates are absolute. To illustrate, conceivable *delta* system commands are shown in Table 1, more or less as presented by Pierre MacKay working from TiTROFF specifications of Brian Kernighan.

`line dh dv`	draw line from current position by `dh dv`
`circle d`	draw circle of diameter `d` with left side here
`ellipse d1 d2`	draw ellipse of diameters `d1 d2`
`arc dh1 dv1 dh2 dv2`	draw arc from current to `dh1+dh2 dv1+dv2`, center at `dh1 dv1` from current position
`spline dh1 dv1 dh2 dv2 ...`	draw B-spline from current position to `dh1 dv1`, then to `dh2 dv2`, then to ...

Table 1. Graphics Commands of the *delta* System

In all of the above examples, `dh dv` is an increment on the current horizontal and vertical position, with down and right positive. The exact syntax of the commands could be somewhat different, depending on how the TEX macros implementing the \special were written. The point is to note that the *delta* system depends on relative coördinates. The *join* system, proposed by Howard Trickey and based on work of Ignacio Zabala, depends on absolute coordinates. This sounds less desirable until we see an important additional command that is proposed: `point` n, where n is an integer in the range, say, between 0 and 255. That is, the current cursor position is saved and associated with the numeric name. Thereafter, its absolute coördinates may be used in a set of line graphics commands similar to those of the *delta* system. An additional command of the *join* system would be a `join` command which would draw a line from one previously named point to another, perhaps using a "pen" of specified shape and size which could also have been defined and named. The usefulness of the *join* approach may be seen from the TEX code in Example 1, contributed to the discussion by Howard Trickey.

```
\def\p#1{\special{point #1}}
\def\j#1#2{\special{join 4 #1 #2}}
\tabskip=15pt
\halign{&&\hfil#\hfil\cr
John& Harry& Alexander\cr
\p1&  \p2&   \p3\cr
\noalign{\vskip 30pt}
\p4&  \p5&   \p6\cr
Helen& Janet& Amy\cr}
\j15\j24\j36
```

Example 1. The *join* Graphics System in TEX

This TEX code is a table specification which, were the `point` and `join` commands implemented, would create a table with three columns and draw lines from points centered under John, Harry, and Alexander to points centered over Janet, Helen, and Amy, respectively. The obvious advantage is that the user needn't know where the points `p1`–`p6` will be placed on the paper. It was noted that points in one system could be converted to points in the other with some TEXhackery. There was discussion of such questions as whether points and pens would be remembered across page boundaries. Pierre MacKay pointed out that the *join* system would be preferable if a graphic were closely joined with text, as in the example, but that the *delta* system would probably be better for graphic objects created independently of text.

Some discussion about how to treat the ends of splines escaped me, presumably because of my small experience drawing them. There seemed to be some agreement that options on spline-drawing commands should allow one or both of the end points ends to be "hidden" or "visible."

One entry in TEXhax mentioned the ANSI standard called GKS (Graphical Kernel Standard) which is being proposed. Copies of the committee's working papers can be obtained for $35.00 from

X3 Secretariat/CBEMA
311 First St. NW, Suite 500
Washington, D.C. 20001

Phil Andrews described the part of the standard called a "Virtual Device Metafile" (VDM) which establishes a set of primitives device drivers would be expected to handle. They include: **polyline, polygon, circle, arc, arc close** (pie or chord), and **cell array** (an array of colored points). More complicated figures such as splines are supposed to be drawn before the VDM is written.

Leslie Lamport's LATEX system was mentioned for its graphics capabilities. The interesting thing about LATEX's graphics is that they are generated by TEX itself, rather than at the DVI-device level; they require nothing more of the device and DVI translator program than is already present if TEX is working—the capability to place characters at coordinates as specified in the DVI. The LATEX graphics work by typesetting line segments, straight and curved, from special fonts supplied with the macros. Circles and arcs of various sizes are available as are lines at various slopes (TEX draws vertical and horizontal lines itself). Certainly, the LATEX graphics capabilities are quite limited, but they suffice for many purposes and require no enhancements to the TEX system and its friends. They do require a fat version of TEX, and I suspect some printer peculiarities may cause broken lines.

John Aspinall from MIT recommended two books on splines in response to a query about where they came from: *A Practical Guide to Splines* by Carl de Boor (Springer-Verlag, 1978) and "Local Control of Bias and Tension in Beta-Splines" by Brian A. Barsky and John C. Beatty in *ACM Transactions on Graphics*, **2**(2) April 1983. He noted that, traditionally, the spline was a draftsman's tool, a long, flexible piece of wood used to draw a smooth curve through a series of points. The spline was held in place by weights, called "ducks," a term which did not make the transition into mathematical jargon.

A number of rancorous exchanges were made debating the proper position of point `0,0` on the physical device page. This question was settled by David Fuchs who declared that point `0,0` is 1 inch down and 1 inch to the right of the top left corner of the actual output page. He then said, "the Great and Powerful Oz has spoken." That is the TEX standard; it is what DVI expects.

Finally, Charles Karney proposed three new `\special` commands: (1) to specify landscape/-portrait page orientation, (2) to print portions of text in a arbitrary rotation, and (3) to position text correctly with respect to a figure. I believe the meaning and implications of the first two proposed commands are fairly obvious. The third is more complicated and was described as follows (more or less). The idea is to allow TEXed labels (or callouts or nomenclature) on figures that are contained in separate graphics files. (Heard this idea before?) The graphics files don't know anything about TEX, and the DVI doesn't know anything about where the labels should go. Basically, Karney proposes that TEX typeset the labels and the graphic file specify where they should be placed. The two are coordinated by a tag given to each label. This proposal requires that the DVI-reading

program read the graphics file and pull out the label specifications, storing their positions and rotations in a table. This information would then be inserted into the label specifications in the DVI so that the DVI-reading program could set the labels in their correct positions, possibly using the capability in Karney's proposal (2). Presumably the graphics-generating program would need to be able to generate the described label material; perhaps a pre- or post-processor could pick out the information and put it somewhere for the DVI-reading program, but the positioning coördinates, it seems, would need to come from the graphic generating program. As another respondent to the proposal asked, "Do most device-independent graphics packages offer a reasonable way of inserting a '`put label n here`' control sequence in their output stream?"

Miscellaneous activity at Texas A&M

Norman W. Naugle and Tomas G. Rokicki

The following three pages illustrate the output from several devices interfaced to TEX at Texas A&M, as well as announcing the availability of a C version of TEX82.

"An Elementary Sum" was output on an Epson LQ1500 (180dpi × 180dpi) using 200dpi fonts (180dpi fonts are not yet available). It was TEXed on a TI/PC running PCTEX (it also works on MicroTEX), and used an output driver and screen preview system written by Tomas Rokicki, which will soon be available on most MS/DOS machines.

We have drivers that work, or can quickly be made to work, on almost any reasonable dot matrix printer or screen display device (currently: TI-855 printer, TI/PC screen, LQ1500, and Printronix P-300). Almost any new driver can be written in a matter of two days. These drivers are written in `WEB` and translated into C, and then, in some cases, modified in machine language. They can be supplied for VAX/VMS, VAX/Unix, Unix in general, MS/DOS, Prime and DG (soon others), as well, of course, as for the QMS-800, 1200, 2400, and soon the Smartwriter. We are also considering the Postscript problem, but no actual work has begun. Some of the drivers suffer from the lack of fonts in the correct size (for example the TI screen), but most have a set of Almost (Computer) Modern fonts.

Our port of TEX to C is aimed at the Unix world, even the large machines, although our main interest is the small systems. The main advantage to C is its portability.

All of these will be available from the Texas A&M TEX Users Group. Write or call for information.