VTEX Enhancements to the TEX Language

Michael Vulis
MicroPress, Inc., 67-30 Clyde Str., Forest Hills, NY 11375
718-575-1816. Bitnet: cscmlv@ccnyvme

Abstract

VTEX enhances TEX by providing support for scalable fonts and thus achieving true device independence. VTEX turns TEX into a compact system (less than 10% of the size of traditional TEX), supports a printer driver definition language, supplements the TEX system with a number of new high-quality scalable typefaces, implements a variety of font effects (compression, shade, outline, shadow). Support of scalable fonts necessitated certain changes to the TEX program, syntax, and fonts; this article describes some of these changes. Since it is likely that other scalable TEX implementations will follow, the author hopes that a scalable TEX standard can be defined before appearance of a conflicting set of definitions.

The Aim of the VTEX System

Device-independence. From the beginning, TEX was designed as a device- and resolution-independent document processor; however, because of its reliance on raster fonts, TEX has never achieved this aim. In a typical TEX implementation, a user is confined to particular output devices and particular resolutions by the mere necessity to maintain a large volume of raster fonts. A typical TEX with a reasonable collection of fonts requires between 10 and 15 MB of storage; support for a second device, or just another magnification step, adds a few megabytes of storage.

The VTEX system is based on vector, rather than raster fonts. Instead of maintaining multiple copies of raster fonts at each resolution, VTEX keeps only one version of each font. The algorithmic encoding of characters is expanded into raster images at run time, allowing instant access to any needed magnification. Use of vector fonts not only greatly increases the number of available fonts — it also shrinks the size of the TEX system to under a megabyte.

Changes in TEX Syntax

Dynamic fonts in VTEX necessitated certain changes in TEX syntax, most importantly in the `\font` command. These changes do not interfere with the TRIP test (so VTEX is still a TEX), and provide much greater flexibility in choosing fonts.

`scaled` and `at`. The `scaled` and `at` parameters work as in standard TEX, with an important exception: *any* acceptable value that follows these keywords is fully supported by all device drivers.

`slant`. The `slant` keyword is used to specify the amount of slant in the font. Any VTEX font can be dynamically slanted. The number that follows the keyword is the slant coefficient, multiplied by 1000. The `slant` keyword makes the existence of `cmsl`, `cmssi` and `cmti` fonts unnecessary, since they can be obtained from other fonts. For instance, `\font\sl=cmr10 slant 167` can be used to declare a *slanted* roman font.

`aspect`. The `aspect` keyword specifies the aspect ratio of the font. `aspect 1000` is the default, `aspect 500` defines a half-height font, while `aspect 2000` defines a font that is twice as high as the default font. The `aspect` keyword, in particular, makes `cmdunh` unnecessary, since `cmdunh` is fairly close to an "aspected" `cmr`.

`smallcaps`. The `smallcaps` keyword defines a font of caps and smallcaps. Any font can be used with the `smallcaps` option. This keyword makes `cmcsc` and `cmtcsc` unnecessary.

`outline`, `shadow`, `gray`, and `fillpattern`. These keywords implement standard font effects: outline, drop shadow, gray, and pattern shading. The width of the outline and the length and direction of shadow can be specified in resolution-independent units. The gray option is followed by the percentage of gray; gray is reasonably device-independent. The fillpattern option implements non-uniform shading, such as horizontal or vertical stripes; fillpattern is inherently device- and implementation-dependent. (The current implementation simply enumerates the available fill patterns).

Changes in TEX Internals

TEX program. Additional features supported by the `\font` command forced deep changes in the VTEX program. Character dimensions (widths, height, depth, and italic correction) are computed dynamically whenever a character is used. The dimensions are adjusted when the font has `slant`, `aspect`, and `smallcaps`; but no adjustment is made for `shade`, `outline`, or `shadow`. The changes to the TEX program generally fall into two groups: the font dimension computations and the `\aliasfont` primitive. The precise specifications for changes will be made available to interested TEX users.

TFM format. VTEX retains the structure of `.tfm` files but alters the meaning of some fields. For instance, the italic correction values in VTEX `.tfm` files are stored for fonts slanted 45° (`slant 1000`). The actual italic correction is dynamically computed by VTEX.

Notice that if a font is defined with `slant 1000`, no adjustments are made to the italic correction specified in the `.tfm` file. This is used with those Symbol fonts (e.g., math extensions) that should never be slanted. These fonts should be defined with `slant 1000`; a flag in the font file informs the drivers that slant should be ignored.

DVI format. The DVI format has been enhanced to support vector fonts. To retain as much compatibility as feasible, extra font information is passed as a `\special` that immediately follows a `fontdef` command. Thus, drivers that use raster fonts can handle enhanced DVI files by ignoring the "tail" of the `fontdef` command.

Font names. As mentioned above, VTEX does not support those Computer Modern fonts that can be obtained as attributes of other fonts. To retain source compatibility with raster TEX sources, VTEX implements the `\aliasfont` primitive. This command maps font names: specifying

```
\aliasfont cmsl10=mvr10 slant 167
```

forces VTEX to treat all references to the `cmsl10` font as references to the `mvr10` font with the `slant 167` parameter.

Fonts

VTEX supports vector analogs of Computer Modern fonts. For compatibility, VTEX uses the same metric files as raster-based TEX. VTEX fonts were developed with the InstaFont program. InstaFont is a combination of the optimized METAFONT algorithms with a user-friendly interface. InstaFont makes font development a relatively trivial task—a complete font can be designed by a novice in less than a week. (For comparison, the Euler font project sponsored by AMS featured about character-per-day performance.) About one hundred fonts have been developed, including look-alikes of such traditional typefaces as Times Roman, Helvetica, and Avant Garde.

Font Layouts

Several changes were made to the font layouts, primarily to eliminate some inconsistencies in the original TEX layouts. In particular:

- VTEX fonts contain all printable ASCII characters, including the greater and less signs and braces. The original TEX approach of borrowing these characters from symbol fonts works poorly with bold or non-CM fonts.
- The " character (`\char34`) is a straight double quote, not the closing double quote. Closing double quote is still available as the ligature ''.
- Straight quote is available, since it may be preferred to the opening quote in some abbreviations.
- VTEX does not "cross" L or l. Instead crossed L and l are included as separate characters. In professional typefaces, the crossing line in these letters is not a straight line.
- VTEX fonts include a number of additional characters, that are essential for professional typesetting. Among them are the section, paragraph and dagger signs, pound and yen, single and double guillemets, crossed D and d. Ordinary TEX lacks some of these characters, the others are "borrowed" from symbol fonts. This works poorly even with many CM fonts: the usual section sign does not blend well with CM Bold Sans-Serif text (§§**1.1**).

The changes in the layout are mostly transparent, since they are compensated for by changes in PLAIN macros. Thus, as long as the user does not unnecessary refer to characters by their position and does not use the double quote character for closing quotes, VTEX stays compatible.

Printer and Screen support

In order to support the maximum number of printers without having to manufacture many different device drivers, VTEX uses printer definition files. These files are written in a brief Pascal-like language that is sufficiently flexible to describe most graphics printers. A Printer Information Compiler (PINC) compiles the definitions to a pseudo-code, which is interpreted by device drivers during printing. Since compiled printer information files (.PIN) are very small (100 - 200 bytes each), VTEX supports many printers in minimal space. On most printers more than one resolution is supported. For instance, on the NEC P-series, the output can be written at 120×240, 240×240, 180×180, 360×180, or 360×360 dots per inch. The variety of PIN files allows the user to choose the optimal tradeoff between time and quality. PINs for non-standard printers can be easily designed by the end-user with PINC.

In a similar fashion, VTEX supports Screen Information files (.SIN). The "open architecture" approach of VTEX resulted in many PINs and SINs developed by VTEX users.

Performance

Runtime scaling used by VTEX drivers does not seriously slow down printing or previewing. The following factors contributed to the performance:

- VTEX scaling algorithms are very fast (up to 500 times faster than those used by METAFONT).
- Similar to PostScript, VTEX drivers maintain a font cache and reuse characters.
- VTEX allows one to pre-generate commonly used fonts. If raster fonts are available, they would be used instead of vectors. Preliminary timing experiments show that in most cases it does not pay to pre-generate more than two or three of the most common fonts. On 386 class computers, pre-generation is completely unneeded.

The current implementation limits font scaling to approximately 120 - 150 point fonts (at 300 dpi). This is because VTEX font effects require drivers to actually generate the entire bitmaps in memory. By the time this paper appears, we expect to raise the limits to about 1000 points (for a 386 CPU).

A Dirty Trick: Find-a-Font

The availability of fonts at any size makes the following example meaningful: Assume that you want to fill a box of given width and height with a given text set in a given font. It is simple to write a generic TEX macro that will return the required magnification of the font. In a raster-based TEX this macro would not be especially helpful — the chances are it will return a magnification that is not available. With VTEX, however, the computed magnification (and, if desired, the aspect ratio) can be used immediately for actually building the box.

The same approach can be used to build an adjustable `\hat` macro. TEX provides the `\hat` in just three sizes. It is, however, possible to compute the dimensions of a `\hat` that would cover a given expression, and then construct a font that would contain a `\hat` of the correct size. The same approach can also work with extendable delimiters.

Not Yet Implemented

Single character scaling. The examples given above will work, as long as you do not use too many individually scaled characters at once. Defining an entire font to scale a single character is overkill, since there are limits on the number of fonts and the amount of font memory available. A possible mechanism would be to allow individual scaled characters with a syntax similar to

```
\char'\^ xscaled 4000 yscaled 1200
```

Such an extension would not be difficult and may be implemented in VTEX in the future.

Rotation. Vector font representation, used in VTEX, allows easy rotation of fonts. A 90° rotation may be particularly useful. A possible syntax for such an extension would be the `rotated` keyword on `\hbox` and `\vbox` commands. Again, changes to the TEX program would not be too difficult.

Much more exciting is a possibility of incorporating an arbitrary-degree rotation. The algorithmic font representation used by VTEX makes generation of rotated fonts relatively simple. On the other hand, A. Hoenig demonstrated in a recent *TUGboat* article (volume 11, number 2, pages 183 - 190) a set of TEX macros that position text along slanted lines and a circle. Hoenig's approach relies on pre-generation of a large number of fonts via METAFONT; if a 50-letter sentence is to be positioned along a circle, 50 different fonts are to be generated. Combining his ideas with VTEX would remove the font pre-generation element. However, an unmodified TEX engine will still have to allocate 50 fonts, which will be extremely memory consuming. Thus, additional changes will have to be made to TEX internals to make Hoenig's techniques truly useful.

Graphics. A natural complement to scalable fonts would be scalable graphics. We are currently evaluating several possible approaches and would very much appreciate input from the TEX community.

Accent Positioning. VTEX currently follows TEX's centering approach to accent positioning. While experimenting with CM and non-CM fonts, we have discovered many glitches in positioning (L̄ for one). Ideally, every font should include a correction table that specifies the amount an accent should be shifted from the default position. This type of positioning is rather similar to kerning and, in fact, can and should be specified as part of the `.tfm` kern table. Changes to the TEX program will be minimal.

Support for older implementations

VTEX typefaces can be used with raster-based TEX. Doing this defeats the main goals of VTEX since with older TEX implementations adding more typefaces makes the system much more bulky. However, this would be a solution for those outside of MS-DOS world (VTEX currently runs only under DOS), and those who are not yet ready to accept scalable fonts.

To install VTEX typefaces in other TEX's one would use the `PXLGEN` font scaling utility that creates raster fonts in `PXL` or `GF` format. `PXLGEN` supports most of font attributes described above. `PXLGEN` uses the same scaling kernel as VTEX, so scaling is extremely fast. For instance, generating a `cmr10` variant at 300 dpi takes under 10 seconds on 386/16mhz machines.

`PXLGEN`'s companion program `TFMGEN` adjusts VTEX `.tfm` files for use with non-scalable TEXs. This is necessary, since non-scalable TEXs require different `.tfm` files each setting of `slant`, `aspect`, and `smallcaps` parameters.

Availability. If you are interested in trying `PXL-GEN`, send a self-addressed stamped disk mailer to MicroPress.

Is VTEX a TEX?

The question whether VTEX is TEX or a different program came up during the discussion of this paper at the TUGboat Annual Meeting. While the ultimate judgement lies with the TEX community, I would like to begin the dicussion with a few remarks:

- VTEX is TEX since it supports a compatibility mode, where all enhancements are disabled. Documents (and `.dvi`) files created in the compatibility mode are fully compatible with any other TEX.
- VTEX is TEX since VTEX enhancements cannot be detected by TRIP. (But one can surely design a special single-purpose TRIP that VTEX will fail).
- VTEX is not really TEX, because the way one uses it is different. Many new macros can be written that would be VTEX-specific.
- VTEX is not really TEX, because switching to VTEX is like following a one-way street. VTEX can read generic TEX files, but the reverse is not always true.

And, finally:

- VTEX is not really TEX, because ultimately it will not be. Historically, a successful program has a life span of only 3 – 5 years. TEX just celebrated its eleventh birthday, which is in itself an unparalleled achievement. However, TEX is losing ground to more modern packages (notably, WordPerfect). Long-time survival of TEX requires substantial improvements to the program, and some compromises on compatibility. We would like to consider current VTEX as just a first tentative step in this direction, and (let us hope) not the last.

Acknowledgements

The author would like to thank Lin Tay, Donald Tsai, Denny Chen and others for their help in creating VTEX, early VTEX users for struggling with embarassing bugs but not giving up, and last, but not least, Lincoln Durst, for greatly improving the readability of this paper.

Appendix

Examples of VTEX Fonts

The fonts below are all at 30 points. The examples were printed on an HP LaserJet printer at 300 dpi.

THIS IS LIKE STENCIL

This is like Hobo

This is like Broadway

This is like Futura Black

This is like Park Avenue Script

This is like Blippo

This is like Palatino Regular

This is like Cloister (Old English)

This is like Korinna Regular

This is like Euler Fraktur Medium

This is like Times Roman

This is like Windsor Bold

This is like Brush Script

This is like Clarendon Bold

This is like Amer. Typewriter

This is like Handel Gothic

This is like Avant Garde Bold

Examples of VTEX Effects

The examples below were printed by VTEX on an HP LaserJet II printer at 300 dpi.

```
\font\test=mvavnb scaled 3000 outline...
```

This is an Outline font.

```
\font\test=mvhlvb scaled 3000 shadow outline...
```

This is Shadow with Outline.

```
\font\test=mvtmsb scaled 3000 smallcaps fillpattern 3...
```

THIS IS SHADED SMALLCAPS.

```
\font\test=mvkorb scaled 3000 outline fillpattern 7...
```

This is an Outline with Shade.

```
\font\test=mvfrac scaled 3000 shadow fillpattern 12...
```

This is a Shadow with Shade.

```
\font\test=mvavnm scaled 3000 aspect 800...
```

This is an expanded font.

```
\font\test=mvavnm scaled 3000 aspect 1200...
```

This is a compressed font.

```
\font\test=mvpalr scaled 3000 slant 250...
```

This is slanted right.

```
\font\test=mvpalr scaled 3000 slant -250...
```

This is slanted left.

```
\font\test=mvssbx10 scaled 3000 slant 200 outline smallcaps fillpattern 6...
```

SMALLCAPS, SHADED, SLANTED ...