More TEX-PostScript links

Bogusław Jackowski, Piotr Pianowski, Piotr Strzelczyk
BOP s.c.
ul. Piastowska 70, Gdańsk, Poland
B.Jackowski@gust.org.pl, P.Pianowski@gust.org.pl, P.Strzelczyk@gust.org.pl

Introduction

According to Donald E. Knuth's decision, TEX stays frozen. This does not mean, however, that it cannot be improved. There are several ways to conform to Knuth's idea of keeping TEX frozen while improving it at the same time:

- developing macro packages;
- writing utility programs: drivers, pre- and post-processors of DVI files, programs for generating TEX documents, graphic utilities (such as METAPOST), etc.;
- providing links to other languages and/or systems: RTF, PDF, HTML, SGML, PostScript, databases (bibliography), WWW pages, etc.

We shall focus our attention on one of the many aspects, namely on PostScript applications. Linking TEX and PostScript was a giant step towards making a professional typesetting system out of TEX. PostScript and TEX fit excellently together, as PostScript is a powerful, well-defined, world-wide standard language of graphic and text page description, and so is TEX. Since most phototypesetters and many printers understand PostScript, it is crucial that TEX also understands PostScript.

Of course, the basic link is a good PostScript driver. Fortunately, such a driver exists — it is Tom Rokicki's dvips. But the driver alone is nowhere near enough — it provides access to nearly all of PostScript's features, but many of these need extra tools in order to make them easy to use. The more so as PostScript, unlike TEX, continues to develop rapidly (recently, Adobe released PostScript Level 3) and thus one can presume that more and more new tools will be needed.

We describe here the tools we have developed for our own purposes. We make them available to the public in the hope that others will find them useful too. The tools were released at the GUST meeting in Bachotek, 1998. The release can be considered as a continuation of a series of earlier releases of similar tools, such as the PS_VIEW previewer, the CEP utility for compressing EPS files, EPS2MF and MF2EPS converters, and others.

Tiff2ps

Encapsulated PostScript files (EPS) are commonly used with TEX for including graphics. Unfortunately, not all systems support encapsulated PostScript. It is understandable, since in order to interpret EPS files, a nearly complete PostScript interpreter is necessary. Therefore, some applications prefer simpler formats. Perhaps one of the most popular is TIFF: a tag-based file format for storing and interchanging raster images. Despite being simpler than EPS, TIFF is rich enough to describe a broad range of bitmap images.

In order to convert a TIFF file to an EPS file a special program is needed. There are several programs available, but we do not know of any written in PostScript. We decided to write our tiff2ps converter in PostScript (actually, in Ghostscript) for several reasons: (a) PostScript has fundamental compression algorithms implemented, which simplifies the processing of TIFF data; (b) the portability of PostScript programs is higher than that of those written in C, and comparable with the portability of programs written in TEX; (c) by definition, PostScript programs exist only in source form and thus modifications and enhancements by third parties are possible; and last but not least, (d) employing Ghostscript guarantees surprisingly efficient processing.

The tiff2ps converter accepts most TIFF files conforming to the TIFF 6.0 specification, including gray, palletted, RGB, CMYK colour models, and LZW, RLE, CCITT (fax) compression; JPEG compression is expected to be available soon.

The resulting EPS files can be compressed using LZW, RLE, or Flate PostScript filters; moreover, the resulting bitmap can be written in either a hexadecimal or an ASCII85 encoded form.

The package also can be used for generating colour-separated EPS files (out of CMYK TIFFs) and "EPS thumbnails", i.e., EPS files with a reduced resolution. Moreover, EPS headers, containing only a pointer to a source TIFF file, can be created. Such an approach has many advantages, as headers are usually negligibly small, which increases the

efficiency of the processing of documents and saves disk space. This form is similar to the OPI (*Open Prepress Interface*) standard used in prepress systems. One should be aware, however, that not all PostScript devices (phototypesetters) are equipped to accept such header files.

The present version of the tiff2ps converter is under development — more facilities and more TIFF formats are to be implemented; nevertheless, backward compatibility will be preserved.

Pf2afm

A "canonical" PostScript Type 1 font comes in two files: an AFM (*Adobe Font Metrics*) file and a PFB or PFA file (*PostScript Font Binary* or *PostScript Font ASCII*, respectively). PFB and PFA files contain exactly the same information, namely the description of glyph shapes; the only difference being that PFB — as the name suggests — contains the data in a binary form, while PFA exploits ASCII (hexadecimal) representation of the data. The most important part of an AFM file contains information about the dimensions of glyphs and about kern pairs.

PostScript interpreters make no use of AFM files. The information is — as Adobe says — for "communicating font metric information to people and programs."

TEX takes metric information from TFM files, not from AFM ones. Fortunately, the AFM format is so general that it can serve not only for "application programs that generate PostScript language page descriptions"; it can also be used by auxiliary programs that prepare metric data for typesetting systems. One of such programs is the well-known afm2tfm (written by T. Rokicki and D. E. Knuth) which produces TFM files out of AFM ones and thus makes PostScript fonts available for TEX users.

It should be emphasized that for TEX, users having both PFB/PFA *and* AFM files is crucial. Alas, with the advent of Windows systems, a new form of font metrics emerged, namely PFM (*Printer Font Metrics*), containing a subset of the information stored in AFM files. PFM files are used by Adobe Type Manager (ATM) for Windows.

As a result, some vendors started to distribute PostScript Type 1 fonts with PFM files instead of AFM ones. We looked for a reliable program to convert PFM files to TFM ones, but we found none. Although a few programs converting PFM to AFM exist, we were not satisfied with them. We were thinking about writing our own converter, when we encountered in the Ghostscript distribution a PostScript program printafm, written at some point by James Clark. The program is capable of extracting metric information from a PostScript font and writing the data in an AFM form. We decided to enhance Clark's program in such a way that it could make use of the data stored in a related PFM file, whenever available. Moreover, we added an interface facilitating batch processing.

The result of the enhancement is the pf2afm converter, or, more adequately, the pf2afm patch. Figure 1 shows the place of pf2afm in a simplified TEX file processing scheme.

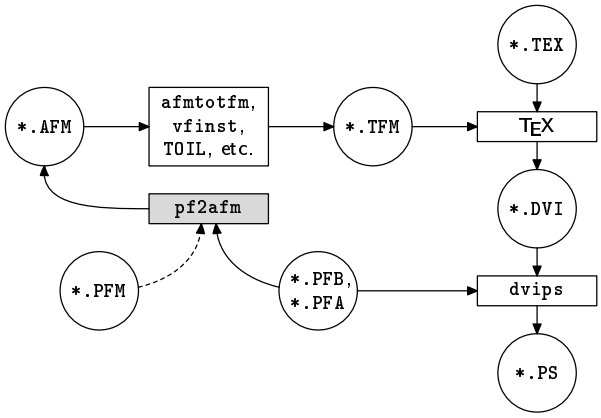

Figure 1: TEX file processing scheme

In general, the retrieval of the complete metric information from PFB/PFA and PFM files is impossible; only if we are lucky, i.e., if the *Encoding* vector of a given font contains all the characters we need, can we conveniently use the resulting AFM; otherwise, hand-tuning may turn out to be necessary.

We believe that such a patch (or converter) may prove useful for people dealing with PostScript fonts, not only for TEX users. Anyway, the pf2afm tool has been included in the standard Ghostscript distribution.

In our practice, on several occasions, we encountered situations when missing AFM files caused trouble. Before we created the pf2afm converter, we had had to use special font programs, such as Fontographer, which we would gladly avoid, to a large extent because of the low reliability of much "professional" software.

Ttf2pf

When a TrueType font format (TTF) was introduced by Apple and Microsoft, Adobe responded by equipping PostScript with a TrueType substitute, Type 42 font format, and by including a TrueType rendering engine in their PostScript interpreter. The relationship between TrueType and Type 42 is schematically shown in Figure 2.

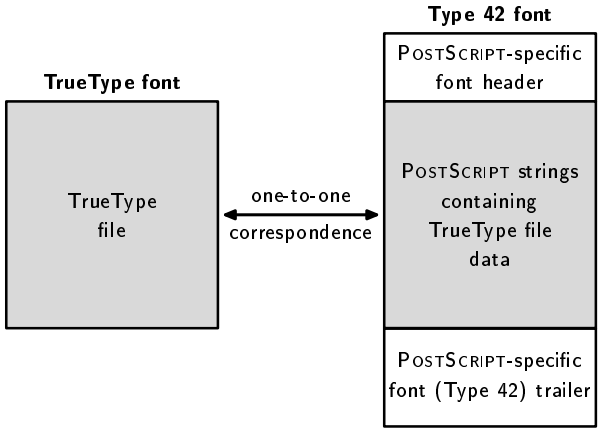

Figure 2: The relationship between TrueType and Type 42 formats

The details are unimportant here. We only observe that Type 42 is, in fact, an equivalent of the TrueType format: TrueType data are simply stored inside a Type 42 file, i.e., no surgery on font intestines takes place and the intact TrueType data can be retrieved from the Type 42 file. Note that the conversion between Type 1 and Type 42/TrueType involves fundamental changes of the representation of glyphs, in particular the hinting information cannot be preserved.

In other words, the conversion between TrueType and Type 42 is purely formal, something like the conversion between PFB and PFA, although the latter is significantly simpler.

Since the font market has been recently flooded with TrueType fonts, we decided that they should be available for TEX users. The best starting point — as usual — turned out to be Ghostscript. Actually, the ttf2pf converter is based on PostScript programs to be found in a standard Ghostscript distribution; ttf2pf itself will be probably included in the standard Ghostscript distribution.

The ttf2pf converter generates two files: Type 42 and AFM; the Type 42 file appears in an ASCII form, thus it can be included by dvips as an ordinary header file; the AFM file allows the generation of a TFM file using standard tools.

It should be noted that not all phototypesetters cope with Type 42 fonts. We also had trouble in converting PostScript files containing Type 42 fonts to PDF format using Adobe's Acrobat Distiller. A careful inspection showed that they were properly embedded in PDF files as genuine TrueType fonts, but the Acrobat Reader displayed uniform rectangles instead of glyphs (although it was able to recognize that a document contained TrueType fonts). That's it for the bad news.

For the good news: Ghostscript renders Type 42 fonts smoothly. Incidentally, there is a possibility that during the TUG meeting in Toruń the Ghostscript release compatible with PostScript Level 3 will be available.[1]

Colormap

Occasionally, a modification of a bitmap graphic is necessary. For example, one may wish to have a pale version of a scanned photo in order to use it as a background. Such an intervention can be easily accomplished using special graphics programs, but this means that both the original and the modified images need to be stored, which is inconvenient. Moreover, GUI programs usually do not allow users to define such changes numerically.

Fortunately, modifications of this kind can be performed by a PostScript engine, and thus it is possible to also perform them at the TEX level; colormap is a tiny package of TEX macros that makes possible nearly arbitrary colouring of gray bitmap images.

The basic method of specifying colour change is to define the colours to which black and white should be mapped, assuming that for the intermediate colours a linear interpolation is applied. You can specify the mapping using either gray or CMYK

[1] Ghostscript 5.50 (released a few weeks after the conference) is still not fully compatible with PostScript Level 3 but has a lot of its features.

Figure 3: The picture to the left is an original image, placed using the command `\epsffile{tiger.eps}` (assuming the usage of the epsf package from the standard dvips distribution); the picture to the right is obtained using the command `\lingraymap[.85:.95]{\epsffile{tiger.eps}}`, where `\lingraymap` is defined in colormap.

models. For example, in order to make an image pale, one should map black to ca 15% of black and white to ca 5% of black, i.e., following the PostScript custom, to 85% and 95% of gray, respectively. Figure 3 shows the result of such a modification.

The linear interpolation of colour is not obligatory. It is possible to apply an arbitrary mapping, given by an array (having 256 entries), represented as a PostScript hexadecimal string. The string can be either created manually or computed by an auxiliary program.

Actually, the colormap package defines four macros, allowing users to map a grey-scale image to:

- another grey-scale image using a linear interpolation (`\lingraymap`)
- a CMYK-model image using a linear interpolation (`\lincmykmap`)
- another grey-scale image using an arbitrary mapping (`\gengraymap`)
- a CMYK-model image using using an arbitrary mapping (`\gencmykmap`)

For details, see the file `colormap.tex`. Although the colormap macro package is written in plain TEX, it is also supposed to work with LATEX.

Acknowledgements

It should be emphasized that all the tools described in this paper could be developed only thanks to L. Peter Deutsch's marvelous interpreter of the PostScript language: it is Ghostscript that provided a convenient platform for creating such tools. We are grateful to L. Peter Deutsch for making Ghostscript available as freeware, for maintaining and developing it, and for helping us promptly whenever we met difficulties.