Software & Tools

Hacking DVI files: Birth of DVIasm

Jin-Hwan Cho

Abstract

This paper is devoted to the first step of developing a new DVI editing utility, called DVIasm. Editing DVI files consists of three parts: disassembling, editing, and assembling. DVIasm disassembles a DVI file to a human-readable text format (more flexible than DTL), and assembles the output back to a DVI file.

DVIasm is useful for people who have a DVI file without TeX source, but need to modify the document. It enables us to put a preprint number, a watermark, or an emblem on the document without touching the TeX source. DVIasm is attractive to even a TeX expert who wants to modify a few words in his document more than a hundred pages long.

We discuss in the paper how DVIasm supplements TeX. The current version supports only the standard DVI file format as in DVItype and DTL. The next versions will support 16-bit TeX extensions including Omega, pTeX, and XeTeX.

1 Introduction

Have you ever heard of DVI, not the Digital Visual Interface[1] but the DeVice-Independent file format? In past years, every TeX user knew what DVI is and used DVI utilities to view and print TeX results. However, in recent times, TeX users have paid attention to DVI less and less because pdfTeX outputs directly to the PDF[2] file format. It is true without doubt that PDF is more powerful than DVI in almost all aspects. Then, do we have to obsolete DVI as PostScript is gradually replaced by PDF?

The DVI file format was designed by David R. Fuchs in 1979, in contrast to the release of PDF version 1.0 in 1993. It is intended to be both compact and easily interpreted by a machine [4, §14]. The most powerful aspect of DVI compared to PDF is nothing but *simplicity*. Imagine the speed of three previewers of DVI, PostScript, and PDF, and compare also the file size of the three different file formats. Furthermore, simplicity enables us to control DVI files in various ways. One of these is to edit DVI files directly — the main object of this paper.

There are many applications of editing DVI files. The most critical situation is when we have a DVI file without TeX source, but we want to modify or to add something to the document. A technical editor may want to put a preprint number on each paper without touching the TeX source. He may also want to put a watermark or an emblem on every paper.

Editing a DVI file is much faster for a TeX novice than learning TeX, when all he wants to do is to add some decorations to his document, and is not familiar with TeX codes. It may even be attractive to a TeX expert who wants to modify a few words in a long document.

Since a DVI file consists of binary data, it must be converted to a human readable format to inspect and edit its contents. The original DVI utility is DVItype [4], written by Donald E. Knuth in 1982. It has two chief purposes: to validate DVI files, and to give an example for developers of DVI utilities [4, §1]. DVItype is a nice utility to inspect the contents of a DVI file because of its human readable text output. However, it lacks any procedure for converting the output back to a DVI file.

A true DVI editing utility is the Device-independent Text Language (DTL) package [5] developed by Geoffrey Tobin. It includes two utilities `dv2dt` and `dt2dv` for converting from DVI to DTL and vice versa. It is notable that there is a one-to-one correspondence between DTL and DVI, and that DTL does not require TFM (font metric) files, in contrast to DVItype. However, DTL is not flexible for ordinary TeX users. For example, users must choose the correct command from `r1` to `r4` according to the width of the move to the right. Moreover, the latest version of DTL was released in 1995, and so it does not support extended DVI formats generated by Omega[3] or Japanese pTeX.[4]

The development plan for a new DVI editing utility, called DVIasm, consists of three phases. This paper is devoted to the first step, where DVIasm is introduced with several examples. The current version of DVIasm supports only the standard DVI file format, like DVItype and DTL, but is more flexible than DTL.

Editor's note: Reprinted from *The PracTeX Journal* 2007-1 (`http://tug.org/pracjourn`), by permission.

[1] A video interface standard designed to maximize the visual quality of digital display devices such as flat panel LCD computer displays and digital projectors [Wikipedia, `http://en.wikipedia.org/wiki/DVI`].

[2] PDF (Portable Document Format) is an open file format created and controlled by Adobe Systems, for representing two-dimensional documents in a device independent and resolution independent fixed-layout document format [Wikipedia, `http://en.wikipedia.org/wiki/PDF`].

[3] An extension of TeX by John Plaice and Yannis Haralambous, `http://omega.enstb.org`.

[4] ASCII Nihongo TeX by ASCII Corporation, `http://www.ascii.co.jp/pb/ptex/index.html`.

In the second phase we will focus on 16-bit characters, for instance, Chinese, Japanese, Korean, and Unicode, to support Omega, pTEX, and the subfont scheme[5] which enables us to use 16-bit characters in TEX and pdfTEX. In the final phase, DVIasm will communicate with the Kpathsea library, so that it will read font metric information from TFM, OFM, JFM, TrueType, and OpenType font files. DVIasm will also support XƎTEX[6] which reads font metric information directly from the font file itself.

2 Prerequisite

2.1 Download and installation

The current version of DVIasm is written in the Python programming language.[7] Why Python not C? The main reason is that Python does not require compiling and linking to get an executable file. Thus, DVIasm consists of a single Python program `dviasm.py` in a human-readable text format and it can run on any platform in which Python is installed. If speed-up is required later, some parts of DVIasm will be translated into C.

The development of DVIasm is controlled by Subversion, a popular version control system, and all revisions of DVIasm can be downloaded at [2]. From now on we assume that `dviasm.py` is in the working directory. The basic usage of DVIasm will be output if the option `--help` is specified as follows:

```
python dviasm.py --help
```

2.2 Creating a DVI file without TEX

For our first example, let's suppose we have saved following three lines as `hello.dump`. (The number at the beginning of each line is just the line number for reference and should not be typed.)

```
1  [page 1 0 0 0 0 0 0 0 0 0]
2  fnt: cmr10 at 50pt
3  set: 'Hello, World!'
```

Then run the following command:

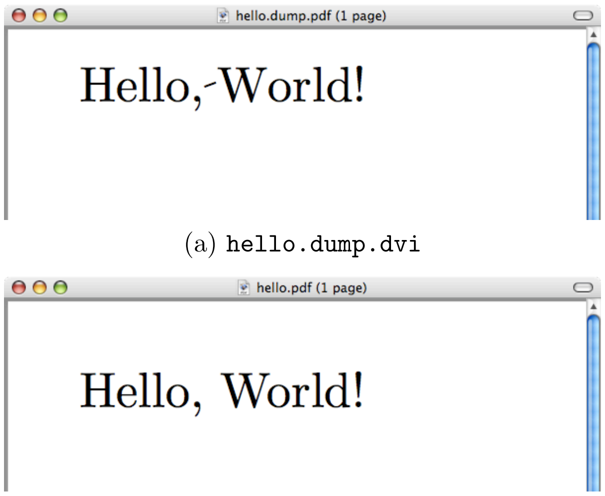
```
python dviasm.py hello.dump -o hello.dump.dvi
```

to get a new DVI file, `hello.dump.dvi`. Its contents are shown in Figure 1(a).

[5] The subfont scheme is a way of splitting a set of 16-bit characters into groups of 256 characters or less, the number of characters that TFM format can accommodate [3].

[6] A typesetting system based on a merger of TEX with Unicode and Mac OS X font technologies, by Jonathan Kew, `http://scripts.sil.org/xetex`.

[7] Python is a dynamic object-oriented programming language that runs on almost all operating systems. Just type '`python`' and hit the return key in the terminal to check whether Python is already installed or not. If not installed, visit the official website `http://www.python.org`.



(a) `hello.dump.dvi`

(b) `hello.dvi`

Figure 1: DVI result generated by (a) DVIasm and (b) TEX.

All DVI files in this paper are converted to PDF with DVIPDFMx[8] version 20061211. The DVI result can also be converted to PostScript with Dvips,[9] or viewed on the screen with the DVI previewers, xdvi,[10] dviout,[11] or yap.[12]

In the input, each page begins with the opening square bracket followed by the string '`page`' (without a colon), ten numbers, and the closing square bracket. Among the numbers the first one usually stands for the page number. In the second line the DVI command '`fnt:`' selects the Computer Modern font, `cmr10` scaled at 50 pt. In the last line the text '`Hello, World!`' is typeset by the command '`set:`'.

2.3 Disassembling a DVI file

We now try to disassemble a DVI file. First, make a TEX file `hello.tex` consisting of the following:

```
\nopagenumbers \font\fnt=cmr10 at 50pt
\noindent\fnt Hello, World! \bye
```

and run TEX (not LATEX) to get `hello.dvi`. The result is shown in Figure 1(b).

[8] A DVI to PDF converting utility by Shunsaku Hirata and Jin-Hwan Cho, `http://project.ktug.or.kr/dvipdfmx/`. It is an extension of DVIPDFM written by Mark A. Wicks, `http://gaspra.kettering.edu/dvipdfm/`.

[9] A DVI to PostScript converter by Tom Rokicki, `http://www.radicaleye.com/dvips.html`.

[10] A DVI previewer in X Window system by Paul Vojta, `http://math.berkeley.edu/~vojta/xdvi.html`.

[11] The most popular DVI previewer in Japan that supports pTEX, `http://akagi.ms.u-tokyo.ac.jp/dviout-ftp.html`.

[12] The DVI previewer in the MiKTEX system by Christian Schenk, `http://www.miktex.org`.

```
[preamble]
id: 2
numerator: 25400000
denominator: 473628672
magnification: 1000
comment: ' TeX output 2007.01.24:1740'

[postamble]
maxv: 667.202545pt
maxh: 469.754990pt
maxs: 2
pages: 1

[font definitions]
fntdef: cmr10 (10.0pt) at 50.0pt

[page 1 0 0 0 0 0 0 0 0 0]
push:
  down: -14.0pt
pop:
down: 643.202545pt
push:
  down: -608.480316pt
  push:
    fnt: cmr10 (10.0pt) at 50.0pt
    set: 'Hello,'
    right: 16.666687pt
    set: 'W'
    right: -4.166702pt
    set: 'orld!'
  pop:
pop:
down: 24.0pt
```

Code 1: Output of disassembling `hello.dvi` with DVIasm.

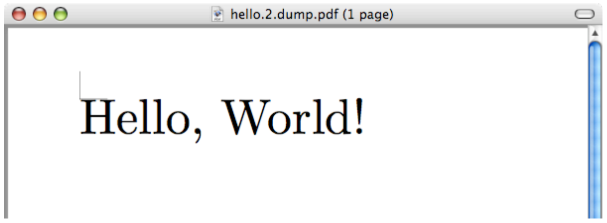
One may easily find two points of difference between (a) and (b) in Figure 1. The first is the location of the text,[13] and the other one is the bar for the Polish letters ł and Ł[14] in (a) instead of the blank space in (b).

Looking at the figures closely, one more difference can be found: there is no kerning between the two characters 'W' and 'o' in (a). The kerning information is stored in TFM files; the implication is that DVIasm would need to communicate with the Kpathsea library to fetch the information. Thus, DVIasm no longer works if the whole TEX system is not installed. This is the reason why DTL and the current version of DVIasm do not require TFM files.

[13] The upper left corner of the paper has the coordinate (−1 in,−1 in), since the default x- and y-offsets are both one inch as usual. So the reference point of 'H' is the origin (0,0) in Figure 1(a). However, it is common to place the upper left corner of 'H' at the origin as in Figure 1(b).

[14] The ASCII code of the blank space is 32, and glyph at position 32 in `cmr10` is the bar for Polish ł and Ł.

```
[page 1 0 0 0 0 0 0 0 0 0]
putrule: 1cm 0.5pt
putrule: 0.5pt 1cm
push:
  down: -14.0pt
pop:
... (skip) ...
```



Code 2: Put a mark at the origin (0,0).

To see the exact differences, let us disassemble `hello.dvi` with DVIasm by running

```
python dviasm.py hello.dvi
```

to get the output[15] shown in Code 1. One can see four new commands, '`push:`', '`pop:`', '`right:`', and '`down:`'. An amount to move follows '`right:`' and '`down:`' as an argument. The meaning of these two commands seems clear.

However, there are two things to keep in mind. First, the coordinate system of DVI is different from the standard Cartesian coordinate system[16] used in PostScript and PDF. In DVI the x-coordinate increases from left to right, like Cartesian coordinates, but the y-coordinate increases from top to bottom, the opposite of Cartesian coordinates. Second, all positions in DVI are specified relatively, not absolutely. It is not possible in DVI to give a command like "go to the coordinate (100 pt, 100 pt)." Only '`right:`' and '`down:`' are allowed in DVI.

Then how do we move to a specific position in DVI? We can use the two commands '`push:`' and '`pop:`'. The command '`push:`' stores the current position in the stack, and '`pop:`' restores the position saved in the stack to the current position.

3 DVI commands

Let's now assume that the lines in Code 1 from the 17th line to the end are saved as `hello.dump`. The first example is to put some mark at the origin (0,0).

[15] DVIasm always outputs to standard output if the `-o` option is not specified.

[16] The Cartesian coordinate system is used to determine each point uniquely in a plane through a pair of numbers (x, y), usually called the x-coordinate and the y-coordinate of the point [Wikipedia, `http://en.wikipedia.org/wiki/Cartesian_coordinate_system`].

command	argument		description
`set:`	string		draw [string] and move to the right by the total width of the string
`put:`	string		draw [string] without moving to the right
`setrule:`	length1	length2	draw a box with width [length2] and height [length1] and then move to the right by [length2]
`putrule:`	length1	length2	draw a box with width [length2] and height [length1] without moving to the right
`push:`			save the current position to the stack
`pop:`			restore the position in the stack to the current position
`right:`	length		move to the right by [length] move to the left if [length] is negative
`down:`	length		move down by [length] move up if [length] is negative
`fnt:`	name `at` length		select the font [name] scaled at [length] [name] does not allow spaces
`xxx:`	string		DVI special command to be processed by DVI utilities; see the next section

Figure 2: DVIasm commands.

command	argument	description
`w:`	length	the same as `right:`, but [length] is stored in the 'w' variable
`x:`	length	the same as `right:`, but [length] is stored in the 'x' variable
`y:`	length	the same as `down:`, but [length] is stored in the 'y' variable
`z:`	length	the same as `down:`, but [length] is stored in the 'z' variable
`w0:`		move to the right by the length in the 'w' variable
`x0:`		move to the right by the length in the 'x' variable
`y0:`		move down by the length in the 'y' variable
`z0:`		move down by the length in the 'z' variable

Figure 3: DVIasm move commands.

This is achieved by inserting two lines after the first line, as in Code 2.

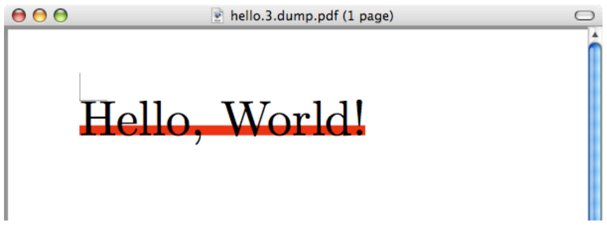
DVI has only two commands for drawing graphics, '`putrule:`' and '`setrule:`'. Both commands draw a box filled with black. The first and the second arguments indicate the size of the height and the width of the box, respectively. (Do not confuse the order of height and width!) The command '`setrule:`' is the same as '`putrule:`' except for moving to the right by the amount of the width after drawing the box.

The next example is to put a box filled with red *under* the text. Since DVI has no color command, Code 3 uses the special command '`xxx:`' that will be explained in the next section.

Exercise. Put the red box *over* the string to hide the overlapped part of the text.

Figures 2 and 3 give the input commands supported by DVIasm. There are two types of arguments, string and length. The string type consists of a text string surrounded by either apostrophes (`'`) or double quotation marks (`"`). It has the same format as the Python string type.[17] The length type is either an integer or a floating point number followed

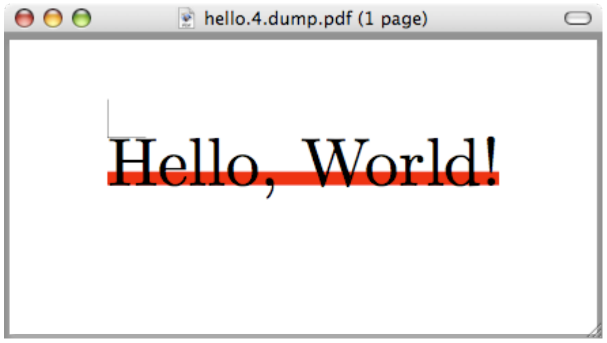
```
... (skip) ...
down: -608.480316pt
xxx: 'color push rgb 1 0 0'
putrule: 10pt 4in
xxx: 'color pop'
push:
... (skip) ...
```



Code 3: Put a box filled with red under the text.

[17] We can input any 8-bit character with hexadecimal value *hh* by `'\x`*hh*`'`. Thus, '`\\`' must be used to type the escape character '\(backslash)'.

```
[page 1 0 0 0 0 0 0 0 0 0]
xxx: 'papersize=6in,3in'
putrule: 1cm 0.5pt
putrule: 0.5pt 1cm
push:
  down: -14.0pt
pop:
... (skip) ...
```



Code 4: Resize the page of Code 3.

by unit (e.g., sp, pt, bp, mm, cm, in).[18] If no unit is specified, the number is in units of sp by default. The argument of '`fnt:`' is exceptional. The name of the font is given without apostrophes.

4 DVI specials

We saw all the DVI commands in the previous section, and we may note that there are no commands for color, graphics, or transformations in DVI. But we already know that they are possible in TeX. How do they work?

The answer is the DVI special command '`xxx:`'. It is the only way for TeX to communicate with DVI utilities. However, each DVI utility supports its own DVI specials. For example, neither DVIPDFM nor DVIPDFMx support a PostScript literal special containing PostScript code. On the other hand, almost none of the PDF specials work with Dvips.

In this section we introduce common DVI specials and show some examples using DVIasm. The material in this section is based on the author's talk at the TUG 2005 conference [1].

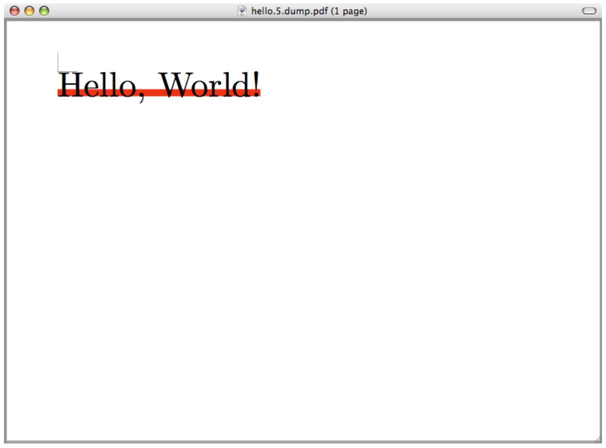
4.1 Page specials

There are two kinds of page specials. Code 4 is an example of the first, specifying a page size; it resizes the previous example (Code 3).

`papersize=[width],[height]` changes the size of whole pages. But it has no effect on the paper size that can be changed by the command line option

[18] 1 in = 2.54 cm = 25.4 mm = 72 bp = 72.27 pt, and 1 pt = 2^{16} sp = 65,536 sp

```
[page 1 0 0 0 0 0 0 0 0 0]
xxx: 'landscape'
putrule: 1cm 0.5pt
putrule: 0.5pt 1cm
push:
  down: -14.0pt
pop:
... (skip) ...
```



Code 5: Landscape orientation.

or by the configuration file (supported by Dvips*,[19] DVIPDFM, and DVIPDFMx).

`pdf:pagesize width [length] height [length]` changes the size of the page containing this special (supported by DVIPDFM*(?) and DVIPDFMx).

Code 5 shows an example of the second kind of page special: landscape paper orientation, rather than portrait.

`landscape` swaps the width and the height of the paper size (supported by Dvips*, DVIPDFM, and DVIPDFMx).

4.2 Color specials

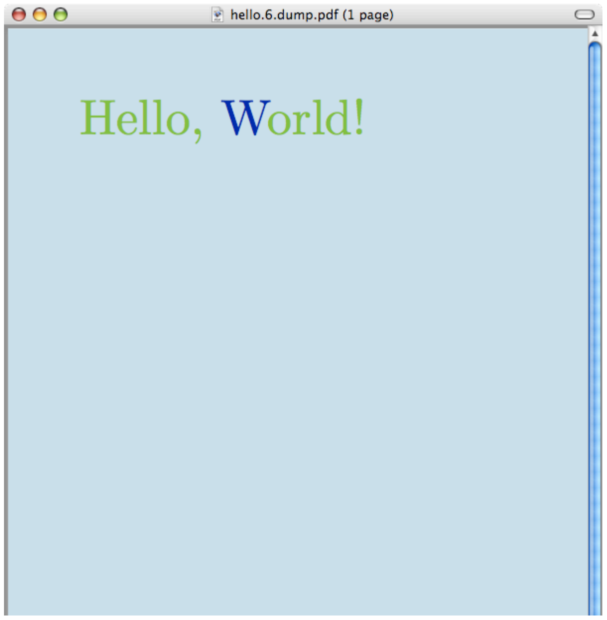
All of the common color specials originated with Dvips. In the specials below, color values can be specified in various ways (Code 6):

- `cmyk [c] [m] [y] [k]`
- `rgb [r] [g] [b]`
- `hsb [h] [s] [b]`
- `gray [g]`
- or a predefined color name.

The value of each color component is a number between 0.0 and 1.0. We refer to [6, pp. 12–13] and [1, p. 11] for PDF color specials, which are easier to understand than PostScript color specials.

[19] * denotes the original source of the feature, and (?) means that the behavior looks mysterious or buggy.

```
[page 1 0 0 0 0 0 0 0 0 0]
xxx: 'background cmyk .183 .054 0 0'
down: 643.202545pt
push:
  down: -608.480316pt
  xxx: 'color push LimeGreen'
  push:
    fnt: cmr10 (10.0pt) at 50.0pt
    set: 'Hello,'
    right: 16.666687pt
    xxx: 'color push rgb 0 0 .625'
    set: 'W'
    xxx: 'color pop'
    right: -4.166702pt
    set: 'orld!'
  pop:
  xxx: 'color pop'
pop:
```



Hello, World!

Code 6: Example of coloring background and text.

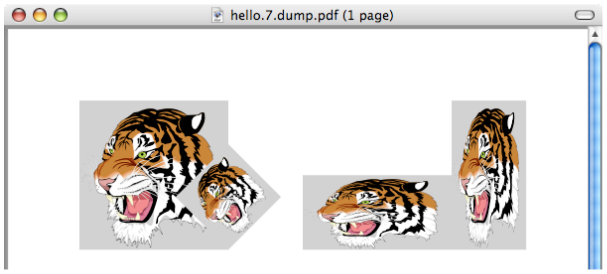
`background [PScolor]` sets a fill color for the background (supported by Dvips*, DVIPDFM, and DVIPDFMx).

`color push [PScolor]` saves the current color on the color stack and sets the current color to the given one (supported by Dvips*, DVIPDFM, and DVIPDFMx).

`color pop` pops a color from the color stack and sets the current color to be that color (supported by Dvips*, DVIPDFM, and DVIPDFMx).

`color [PScolor]` clears the color stack, and saves and sets the given color (supported by Dvips*, DVIPDFM(?), DVIPDFMx).

```
[page 1 0 0 0 0 0 0 0 0 0]
down: 150bp
xxx: 'psfile=tiger.eps rhi=1500
      llx=17 lly=171 urx=617 ury=771 clip'
right: 150bp
xxx: 'psfile=tiger.eps rhi=750
      llx=17 lly=171 urx=617 ury=771
      angle=45 clip'
right: 75bp
xxx: 'psfile=tiger.eps rwi=1500 rhi=750
      llx=17 lly=171 urx=617 ury=771 clip'
right: 150bp
xxx: 'psfile=tiger.eps rwi=750 rhi=1500
      llx=17 lly=171 urx=617 ury=771 clip'
```



Code 7: Manipulating an image. (Line breaks are editorial.)

4.3 Image specials

The special '`psfile`' is used for including an EPS (PostScript) graphics file. EPS files contain bounding box information. For example, the bounding box of the EPS file in the following example[20] is

```
%%BoundingBox: 17 171 567 739
```

Four options `llx`, `lly`, `urx`, and `ury` specify the clipping area of the EPS file, and two options `rwi` and `rhi` (0.1 bp unit) are used to resize the clipped area.

```
psfile=[name] hsize=[num] vsize=[num]
hoffset=[num] voffset=[num]
hscale=[num] vscale=[num] angle=[num]
llx=[num] lly=[num] urx=[num] ury=[num]
rwi=[num] rhi=[num] [clip]
```

Although Dvips*, DVIPDFM, and DVIPDFMx all recognize `psfile`, neither DVIPDFM nor DVIPDFMx have internal PostScript interpretation support, so they cannot process EPS files without Ghostscript or another PostScript "distiller" available.

However, both DVI utilities support JPEG and PDF image files, which are not processed by Dvips. The PDF image special for JPEG and PDF images

[20] Namely `tiger.eps`, which can be found in the `examples` directory of Ghostscript, the most popular free software interpreter (available under the GPL) for PostScript and PDF. See `http://www.ghostscript.com` for more information.

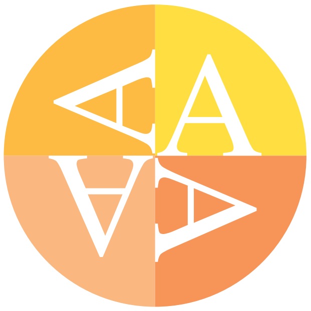
has reader-friendly syntax. We refer to [6, p. 13] and [1, pp. 12–14] for examples.

```
pdf:image width [length] height [length]
depth [length] rotate [num]
scale [num] xscale [num] yscale [num]
bbox [ulx] [uly] [lrx] [lry]
matrix [a] [b] [c] [d] [x] [y] ([name])
```

(Supported by DVIPDFM*(?) and DVIPDFMx).

4.4 Transformation specials

It is possible in LaTeX to rotate and scale text and figure. But Dvips has no transformation special for this purpose. Instead, it enables us to insert literal PostScript code.



`" [PScode]` inserts literal PostScript code surrounded by a `gsave` and `grestore` pair, so that it will have no effect on the rest of the document (supported by Dvips* only).

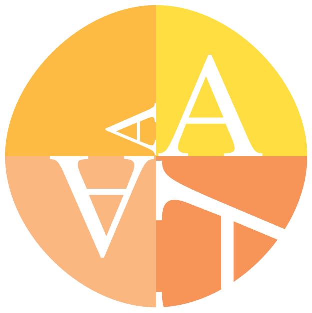
`ps:[PScode]` inserts literal PostScript code without `gsave` and `grestore` (supported by Dvips* only).

The code for the example above follows (line breaks in the long specials are editorial):

```
[page 1 0 0 0 0 0 0 0 0 0]
xxx: 'papersize 2in,2in'
xxx: '" Goldenrod newpath 0 0 moveto 50 0 lineto
       0 0 50 0 90 arc closepath fill'
xxx: '" Dandelion newpath 0 0 moveto 0 50 lineto
       0 0 50 90 180 arc closepath fill'
xxx: '" Apricot newpath 0 0 moveto -50 0 lineto
       0 0 50 180 270 arc closepath fill'
xxx: '" Peach newpath 0 0 moveto 0 -50 lineto
       0 0 50 270 0 arc closepath fill'
xxx: 'color gray 1'
fnt: ptmr8r at 50pt
xxx: 'ps:gsave'
put: 'A'
xxx: 'ps:currentpoint currentpoint translate
        90 rotate neg exch neg exch translate'
put: 'A'
xxx: 'ps:currentpoint currentpoint translate
      90 rotate neg exch neg exch translate'
put: 'A'
xxx: 'ps:currentpoint currentpoint translate
      90 rotate neg exch neg exch translate'
put: 'A'
xxx: 'ps:grestore'
```

On the other hand, DVIPDFM and DVIPDFMx have a PDF transformation special for rotation and scaling, etc. Note that literal PDF code is used in the following example.

`pdf:btrans [`*same options as* `pdf:image]` applies the specified transformation to all subsequent text (supported by DVIPDFM* and DVIPDFMx).



`pdf:etrans` concludes the action of the immediately preceding `pdf:btrans` special (supported by DVIPDFM* and DVIPDFMx).

`pdf:content [PDFcode]` inserts literal PDF code surrounded by a `q` and `Q` pair, so it will have no effect on the rest of the document (supported by DVIPDFM* and DVIPDFMx).

`pdf:literal [PDFcode]` inserts literal PDF code without the `q` and `Q` pair (supported by DVIPDFMx* only).

Here is the PDF implementation of the figure above (again, line breaks are editorial):

```
[page 1 0 0 0 0 0 0 0 0 0]
xxx: 'papersize 2in,2in'
xxx: 'color Goldenrod'
xxx: 'pdf:content 0 0 m 50 0 l
      50 25 25 50 0 50 c f'
xxx: 'color Dandelion'
xxx: 'pdf:content 0 0 m 0 50 l
      -25 50 -50 25 -50 0 c f'
xxx: 'color Apricot'
xxx: 'pdf:content 0 0 m -50 0 l
      -50 -25 -25 -50 0 -50 c f'
xxx: 'color Peach'
xxx: 'pdf:content 0 0 m 0 -50 l
      25 -50 50 -25 50 0 c f'
xxx: 'color gray 1'
fnt: ptmr8r at 50pt
put: 'A'
xxx: 'pdf:btrans rotate 90 scale .5'
put: 'A'
xxx: 'pdf:btrans rotate 90 scale 2'
put: 'A'
xxx: 'pdf:btrans rotate 90 scale 2'
put: 'A'
xxx: 'pdf:etrans'
xxx: 'pdf:etrans'
xxx: 'pdf:etrans'
```

(This figure is not quite circular, compared to the previous PostScript one, since that would require much longer code.)

To this point, we have discussed common DVI specials, mostly originated by Dvips. There are also many PDF specials not yet mentioned. DVIPDFM originates almost all PDF specials, and its manual [6] is a good source. Moreover, the present author discussed at TUG 2005 [1] how differently the three DVI utilities, Dvips, DVIPDFM, and DVIPDFMx behave on the same special command.

5 Conclusion

Imagine that one has a DVI file without TEX source, but wishes to modify or to add something to the document. For example, a technical editor may want to put a preprint number on each paper, which was not fixed at the time of writing. He may also want to put a watermark or an emblem on every paper.

We also imagine a TEX novice who wants to include some decorations in his document, but has trouble writing TEX code. Is it the best advice for him to learn TEX? It might be — if he has enough time. If not, DVIasm is an alternative. In fact, he may learn DVI commands more quickly than TEX commands. DVIasm may even be attractive to a TEX expert who wants to modify a few words in a long document.

DVIasm is written for these purposes, as a supplement to TEX and its extended versions. Of course, DVIasm is not an alternative to TEX! Neither line breaking nor page breaking is (or ever will be) supported.

As mentioned at the beginning of the paper, DVIasm development is in its first phase. The next paper will discuss how to support 16-bit characters in DVIasm. Any comments are welcome, and will be helpful to improve the program.

References

[1] Jin-Hwan Cho, *Practical Use of Special Commands in DVIPDFMx*, TUG 2005, International Typesetting Conference. Wuhan, China. `http://project.ktug.or.kr/dvipdfmx/doc/tug2005.pdf`

[2] Jin-Hwan Cho, *The DVIasm Python script*. `http://svn.ktug.or.kr/viewvc/dviasm/?root=ChoF`

[3] Jin-Hwan Cho and Haruhiko Okumura, *Typesetting CJK languages with Omega*. TEX XML, and Digital Typography, Lecture Notes in Computer Science **3130** (2004), 139–148.

[4] Donald E. Knuth, *The DVItype processor* (Version 3.6, December 1995). `http://ctan.org/tex-archive/systems/knuth/texware/dvitype.web`.

[5] Geoffrey Tobin, *The DTL Package* (Version 0.6.1, March 1995). `http://ctan.org/tex-archive/dviware/dtl/`.

[6] Mark A. Wicks, *DVIPDFM User's Manual* (Version 0.12.4, September 1999). `http://gaspra.kettering.edu/dvipdfm/dvipdfm-0.12.4.pdf`.

◇ Jin-Hwan Cho
Department of Mathematics
The University of Suwon
Republic of Korea
`chofchof (at) ktug dot or dot kr`