A Device Independent DVI Interpreter Library for Various Output Devices

Hirotsugu Kakugawa
Faculty of Engineering, Hiroshima University
1-4-1 Kagamiyama, Higashi Hiroshima,
Hiroshima, 739-8527 JAPAN
h.kakugawa@computer.org
URL: http://kakugawa.aial.hiroshima-u.ac.jp/

Abstract

In this paper, we describe DVIlib, which is a device independent DVI interpreter library written in C developed by the author. Since DVIlib is completely independent from specific output devices, new printer drivers and previewers (DVIware) can be easily developed. DVIlib is a set of functions to read and render DVI files.

To render a page, DVIlib generates a bitmap for each character in a page and call a callback function to draw a bitmap on a device. Therefore, what a programmer must do to make a new DVIware is to write device dependent routines (initialization and drawing a bitmap on a device). Since DVIlib adopts VFlib as a font module, many font file formats are available, including PK, GF, VF, and Type 1. Thus, any DVIware that adopts DVIlib supports many font file formats.

We developed a program to convert from DVI to bitmap, printer drivers, and previewers for X Window System. These programs are easily developed by adopting DVIlib.

Introduction

Since TeX is a de fact standard of typesetting software, a set of software for handling typeset result (a DVI file) plays an important role to print and view documents in TeX. Such software are called DVIware.

In this paper, we introduce new framework to build a set of DVIware, such as printer drivers and previewers. Many kind of printer drivers and previewers are required by the following reasons:

- Different printers adopt different printer description languages (escape sequences) to represent image to be printed,
- Different previewer programs are required for each window and desktop environment, or
- Novice users and expert users may require different user interfaces.

Although an interpreter for DVI file itself is simple, development of a new DVIware is not a easy task, since it requires complex internal modules for handling font files and the 'special' DVI instruction, for example, figures in Encapsulated PostScript (EPS), changes of text colors, and scaling texts. In addition, we expect that preview image on window system and printed image by printer are the same, except device resolutions.

Most parts of program code for previewers and printer drivers are the same, and therefore, developing previewers and printer drivers individually is not adequate. In [1], Beebe proposed a set of functions written in C that can be used as 'parts' to form a DVIware. To develop a new printer driver or previewer, such functions are lexically included in a DVIware program; only device dependent routine should be developed. Since the same 'core' of DVIware (e.g., a DVI interpreter and a font module) is shared among various DVIware, each DVIware have the same output and functionality, except resolutions of output devices.

In this paper, we introduce a new framework to develop a family of DVIware. We developed a device independent DVI interpreter library written in C named DVIlib. Different from the approach by Beebe[1], DVIlib provides a set of functions to be called by DVIware. Since DVIlib is completely independent from specific output devices, new printer drivers and previewers can be easily developed. DVIlib has following features:

- drawing EPS figures,
- handling change of text colors,
- scaling boxes, and

- support for various font file formats (GF, PK, virtual fonts, Type 1, etc.) by adopting VFlib as a font module [5]

Currently, xdvi, dvips, and ghostscript are widely used. Although combination of them is a strong set of software for TeX DVI files, we may be bothered configuring these software (e.g. font definitions) consistently to obtain the same output. This problem often occurs when we use localized (e.g., Japanese) versions of those software.

DVIware which adopts DVIlib share the same DVI interpreter and a font rasterizer module. In addition to this, the same configuration file can be shared by previewer and printer drivers. Therefore, what we get on paper by printer is exactly the same as what we see on CRT screen (except device resolutions).

This paper is organized as follows. First we explain how we can use DVIlib to develop printer drivers and previewers. Then, internal structure of DVIlib is explained. Next, we explain supported features of the 'special' DVI instruction. Then, we introduce several printer drivers and previewers using DVIlib. Finally, we give concluding remarks.

Structure

In this section, we explain the structure of DVIlib. (See Figure 1.)

DVIlib is a library which is linked against an application software. It offers a set of functions to render a DVI file.

It uses VFlib [5] to obtain glyph of characters in various font formats, (GF, PK, Type 1 and Virtual Font, and more). VFlib uses a file named "vflibcap" as a font database. In this file, we can describe font definition, for example, a Type 1 font is used for cmr10.600pk, or a PK font is used for cmbx10.600pk.

DVIlib has an interface to ghostscript, which is a PostScript interpreter, to render figures in EPS format.

DVIlib defines several callbacks to draw a page. Fundamental callbacks are (1) drawing a given bitmap on a page, and (2) drawing a rectangle on a page. DVIlib internally obtains glyph of characters and converts EPS files to images. Thus, an application program can draw characters and EPS figures by simply implementing a callback to draw a bitmap on a page.

Features and specification of DVIlib are described in detail in the next section.

Using DVIlib

DVIlib provides a set of functions to obtain a page image from a DVI file. DVIware using DVIlib must obey the following framework to handle DVI files.

1. Initialize DVIlib by calling a function named `DVI_INIT()`.
2. Call a function named `DVI_CREATE()`, with a DVI file name as an argument, to create a DVI object. DVI object contains various information of a given DVI file.
3. Call a function named `DVI_DRAW_PAGE()`. Arguments for this function include, a set of callback functions and page number to draw.

In DVIlib, an output device is abstracted by a structure `DVI_DEVICE` which is a data structure for callback functions and device dependent parameters. It contains following members, for example:

- resolution of a device,
- a pointer to a function to draw a bitmap on a page, (This callback is used to draw glyph of characters and EPS figures.)
- a pointer to a function to draw a rectangle on a page,
- pointers to functions to draw a graymap and pixmap on a page. (If one of these callbacks is defined, it is used to draw EPS figures. Otherwise, a callback to draw a bitmap is used to draw EPS figures.)
- a pointer to a function to print error message,
- a pointer to a function to change text colors,
- a pointer to a function to change background colors.

Since we can create multiple DVI objects independently and simultaneously in a single application program, for example, we can create a previewer that opens and displays multiple DVI files at the same time.

Now we explain details of three important functions.

`DVI_INIT(char *`*`vflibcap`*`, char *`*`params`*`)`
— Initialization function for DVIlib. The first argument *`vflibcap`* is a path name of font database file called "vflibcap". It is used by VFlib (a font library) to resolve fonts used in DVI files. In a vflibcap, variables can be used to customize its contents at run time. For example, output device resolution is parameterized by a variable. VFlib has a feature to define values of variables when its initialization function is called. The second argument *`params`* is used to pass variables values of VFlib for runtime customization.

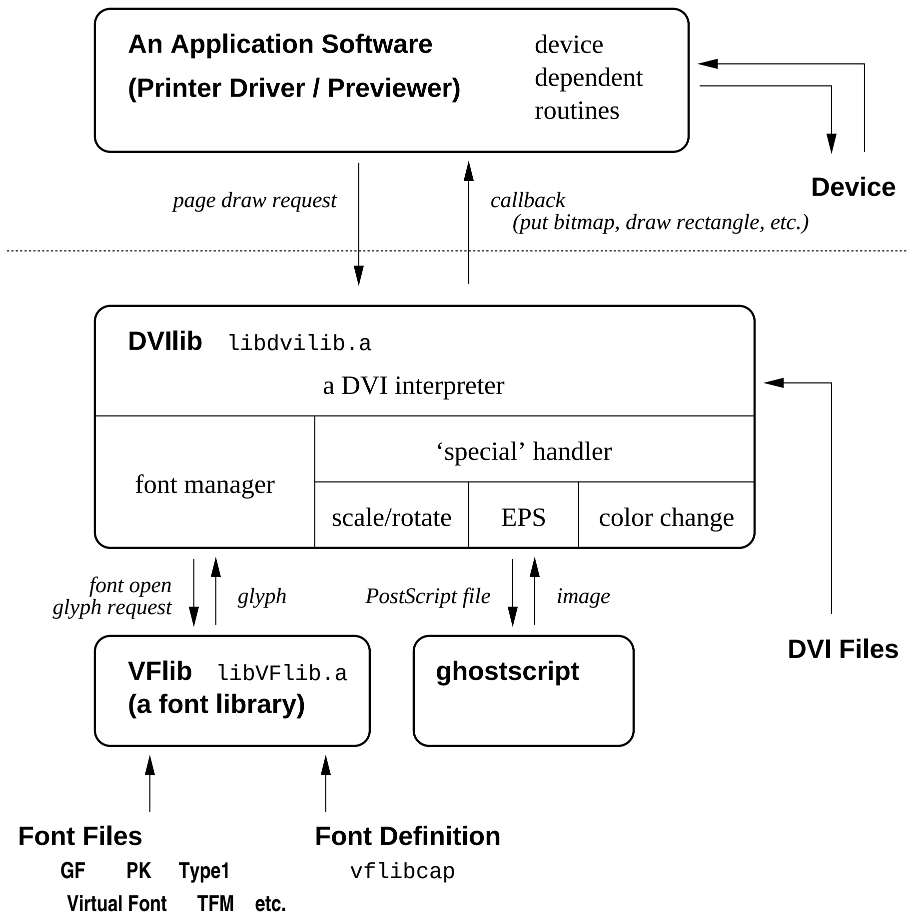

Figure 1: The structure of DVIware based on DVIlib

If null pointers are given for these parameters, default values are used, i.e., default vflibcap file is used and no variable value customization.

`DVI_CREATE(DVI_DEVICE` *`dev`*`, char *`*`file`*`, DVI_PROPERTY` *`prop`*`)`

— This function creates a DVI object for a given DVI file *file*. The first argument *dev* is a set of callback functions.

The third argument *prop* is used to control behavior of a DVI interpreter. Features of a DVI interpreter is characterized as *properties*, and they can be enabled or disabled. Data type `DVI_PROPERTY` is used to describe a set of properties to be enabled. DVIlib has a set of functions to operate a data of this data type.

Following features of a DVI interpreter are controled by the third argument of `DVI_CREATE`. (Although there are other features to be controled, they are omitted here.)

- Delay opening fonts until they are necessary. This property is effective for previewers to display the first page quickly, since not all fonts must be opened to draw the first page.
- Invoke ghostscript program immediately when a DVI object is created. Standard behavior is to invoke ghostscript when an DVI interpreter encounters the first EPS figure. This property is effective for previewers to display the first EPS figure quickly, since ghostscript may finish its initialization before the first EPS figures in a DVI file is encountered.
- Skipping execution of the 'special' DVI instructions. For example, drawing EPS figures can be ignored.
- Printing a list of fonts in a DVI file.

DVIlib has a function to change DVI properties after a DVI object is created. It can used in a previewers, for example, to control behavior of a DVI interpreter interactively.

`DVI_DRAW_PAGE(DVI` *`dvi`*`, DVI_DEVICE` *`dev`*`, int` *`page`*`, double` *`shrink`*`)`

— A function `DVI_DRAW_PAGE()` plays an important role in DVIlib. When it is called, it invokes a DVI instruction interpreter for a given page of a DVI file. The fourth parameter *shrink* is shrink factor; obtained bitmaps and their positions are automatically shrinked by DVIlib.

Whenever a DVI interpreter encounters a `SET_CHAR`, `SET_`*x*, or `PUT_`*x* DVI instruction[1], it obtains a glyph by calling a function to obtain glyph of character in VFlib[5]. Then, callback function to draw a bitmap is invoked with obtained glyph and its position on a page as arguments.

When it encounters a `SET_RULE` or `PUT_RULE` DVI instruction[2], it invokes a callback function to draw a rectangle with position, width and height of a rectangle.

Supported for the 'special' DVI instructions are descried later.

Callback functions

In this section, we describe callback functions. Fundermental callback functions are (1) drawing a bitmap, and (2) drawing a rectangle. Although many other callbacks can be defined, we explain only important callbacks here.

`dev_put_bitmap(DVI_DEVICE` *dev*`, DVI` *dvi*`, DVI_BITMAP` *bm*`, int` *font_id*`, long` *k*`, long` *code_point*`, long` *x*`, long` *y*`)`

— This callback function draws a bitmap *bm*. Its position is *x* and *y*. Other arguments are supplementary information, such as font number and character code, for example.

`dev_put_rectangle(DVI_DEVICE` *dev*`, DVI` *dvi*`, long` *x*`, long` *y*`, long` *w*`, long` *h*`)`

— This callback function draws a rectangle of width w and height h on page. Its position is *x* and *y*.

DVIlib defines other useful callbacks.

`dev_put_graymap(DVI_DEVICE` *dev*`, DVI` *dvi*`, DVI_GRAYMAP` *gm*`, int` *font_id*`, long` *k*`, long` *code_point*`, long` *x*`, long` *y*`)`

— This callback is used to draw EPS figures in grayscale, if defined. It this callback is not defined, `dev_put_bitmap` is used to display EPS figures.

`dev_put_pixmap_rgb(DVI_DEVICE` *dev*`, DVI` *dvi*`, DVI_PIXMAP_RGB` *pm*`, int` *font_id*`, long` *k*`, long` *code_point*`, long` *x*`, long` *y*`)`

— This callback is used to draw EPS figures in color, if defined. It this callback is not defined, `dev_put_graymap` or `dev_put_bitmap` is used to display EPS figures.

`dev_color_rgb(DVI_DEVICE` *dev*`, DVI` *dvi*`, int` *f*`, double` *r*`, double` *g*`, double` *b*`)`

— Change text colors. Parameters *r*, *g*, *b* represents intensities of red, green, and blue.

`dev_message_error(DVI_DEVICE,DVI,char*,...)`

— Print an error message. This callback can be used to display an error message on a message dialog window, for example.

Skelton of a DVIware

Following is the outline of a simple printer driver program with DVIlib:

```
#include <dvi-2.5.h>

int  main()
{
  int   p;
  DVI_DEVICE  dev;

  /* Initialize DVIlib */
  DVI_INIT(NULL, NULL);

  /* Make a set of callback functions */
  dev = DVI_DEVICE_ALLOC();
  dev->h_dpi = 300;  /* 300 dpi */
  dev->v_dpi = 300;  /* 300 dpi */
  dev->put_rectangle = dev_put_rectangle;
  dev->put_bitmap    = dev_put_bitmap;

  /* Create a DVI object */
  dvi = DVI_CREATE(dev, file, NULL);

  for (p = 1; p < dvi->npage; p++){
    /* Clear page buffer */
    page_clear();
    /* Invoke a DVI interpreter */
    DVI_DRAW_PAGE(dvi, dev, p, 1.0);
    /* Send page buffer to a printer */
    page_send_printer();
  }

  return 0;
}

void
dev_put_bitmap(DVI_DEVICE dev, DVI dvi,
   DVI_BITMAP bm, int font_id, long k,
   long code_point, long x, long y)
{
    /* Put a bitmap 'bm' at <x, y>. */
}
```

[1] These are DVI instructions to draw a character on a page.

[2] These are DVI instructions to draw a rectangle on a page.

```
void
dev_put_rectangle(DVI_DEVICE dev, DVI dvi,
   long x, long y, long w, long h)
{
    /* Draw a rectange of width ‘w’ and */
    /* height ‘h’ at positon <x, y>.    */
}
```

Similary, previewers can be built easily by adding user interface to change pages to be displayed by pushing buttons by mouse, for example.

Support for the ‘Special’ DVI Instruction

When an interpreter encounters the ‘special’ DVI instruction, it invokes a handling routine according to parameter string of the instruction.

- A figure in EPS.
  Pass the EPS file to ghostscript to render. Output of ghostscript is a bitmap and call a callback function to draw a bitmap. If a callback to draw a graymap (pixmap) is defined, PGM (PPM) format is selected as an output format of ghostscript. (Otherwise, PGM format is selected.) Then, a callback is invoked to put obtained image.
- Change of text or background colors.
  If callbacks to change text and background colors are defined, they are invoked with RGB values for new color.
- Change of scaling factors.
  This feature is offered by `graphics.sty` and `graphicx.sty` pacakges (`\scalebox` and `\resizebox` commands). These macro packages generates embedded PostScript code as a parameter of ‘special’ DVI instruction. DVIlib parses embedded PostScript code (by simple pattern matching) and generates scaled glyph of characters and rectangles.

  Currently, change of rotation angles is not supported.

By this software architecture of DVIlib, every DVIware can support EPS figures and scaled text and rectangles.

Multilingual Issues

John Plaice and Yannis Haralambous propose Ω as a multilingual extension of TEX[8]. They propose ΩFM (Ω font metric), which is extended version of TFM. pTEX also defines extended font metric JFM (Japanese font metric). Since VFlib supports ΩFM, JFM, ΩVF(Ω virtual font), DVIware based on DVIlib can display and print DVI files by Ω and pTEX. (Currently, level-0 ΩFM is supported by VFlib, but level-1 is not.)

In Japanese community, a variant of TEX, named pTEXis widely used. pTEX supports Japanese characters and vertical writing directionality[2]. For supporting vertical writing, pTEX defines new byte code to change writing directionality in DVI instruction set, and new register in DVI virtual machine to hold current writing directionality. DVIlib supports extended specification of pTEX. This extension is capsulated indside of DVIlib, DVIware based on DVIlib are not ware of such extension. Thus, all DVIware based on DVIlib can make use of such extension.

DVIware with DVIlib

We developed several DVIware that adopts DVIlib.

- xgdvi
  A previewer on X Window System with GTK+ 1.2 toolkit. This software has fancy GUI for novice users.
- spawx11
  A simple previewer on X Window System.
- dvi2rpdl
  A printer driver for Ricoh RPDL printers.
- dvi2escpage
  A printer driver for EPSON ESC/Page printers.
- dvi2img
  A converter program to generate a PGM image file from a DVI file.
- dvifontlist
  A utility program to print a list of fonts used in a DVI file.
- dvispecials
  A utility program to print a list of ‘special’ DVI instructions in a DVI file.

xgdvi is implemented about 6000 lines of C code. It supports displaying color EPS figures. Since complex routines for DVI interpretation, handling EPS figures and font files are managed in DVIlib, most of the code of xgdvi is for fancy GUI. Screen shot of xgdvi is shown in Figure 2. It supports multiple buffers: multiple DVI files can be opened simultaneously and they are switched without opening DVI files again.

dvifontlist and dvispecials are implemented by making use of DVI proporties; a DVI file is opened by disabling character rendering and enabling printing DVI file information. These programs are implemented about 350 lines of C code.

We developed another previewer spwx11[3] on X Window System with minimum factionality, as a

[3] spwx11 stands for “the Simplest Previewer in the World for X11.

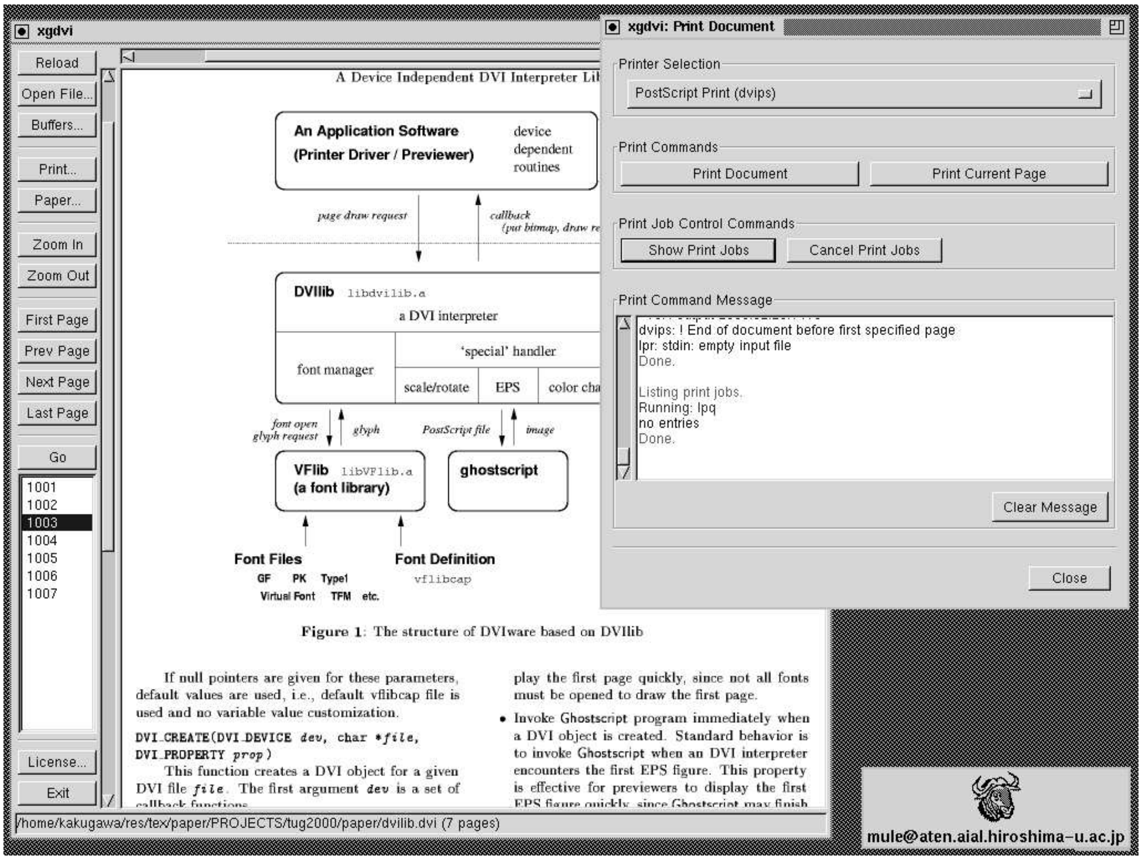

Figure 2: Screen shot of xgdvi

challange. It is implemented by only 140 lines of C code; the program consists of interface to DVIlib and X Window System to draw bitmaps and rectangles. Although this program does not support anti-aliased display, it does support displaying EPS figures and various font file formats. spawx11 is a previewer on X Window System with anti-aliasing display feature.

Typically, a printer driver can be implemented about 500 lines of C code (if it does not support various printer description languages).

Conclusion

In this paper, we introduced DVIlib which is device independent DVI interpreter. Since it adopts VFlib for font module, fonts in various font file formats can be available. We also developed several printer drivers and previewers that adopts DVIlib.

By adopting DVIlib, we can develop a simple printer driver within a day. When we develop a previewer, we can concentrate on our efforts for fancy GUI. Since all DVIware share the same DVI engine, printed result is exactly the same as what we see on display.

Currently, following featreus are not implemented, for example.

- support for HyperTEX
- full support for embedded PostScript literals (e.g., support for PSTricks packages)
- rotation of figures and texts (e.g., the \rotatebox command of graphics and graphicx packages),

DVIlib is written in C and about 6400 lines of code, half of which is for handling the 'special' DVI instruction. (VFlib, a font module of DVIlib, is written in C and about 33000 lines of code.)

DVIlib a part of the TeX-Guy package which is a set of DVIlib and DVIware based on DVIlib (including xgdvi and spwx11). DVIlib and VFlib are free software and distributed under the terms of Library GNU Public License. DVIware such as xgdvi are also free software and distributed under the terms of GNU Public License.

They are tested on FreeBSD 3.2, Solaris 2.5.1; it is not difficult to port them to other Unix-like systems such as Linux. Visit our web pages `http://TypeHack.aial.hiroshima-u.ac.jp/TeX-Guy/` and `http://TypeHack.aial.hiroshima-u.ac.jp/VFlib/` for further information and download.

References

[1] Nelson H. F. Beebe. DVIxxx. Available on CTANas `/dviware/beebe/`.

[2] ASCII Coop. pTeX web page. `http://www.ascii.co.jp/pb/ptex/`. Publishing TeX for Japanese, with vertical writing.

[3] Roger D. Hersch, editor. *Visual and Technical Aspect of Type*. Cambridge University Press, 1993.

[4] Hirotsugu Kakugawa. The web page of VFlib. `http://TypeHack.aial.hiroshima-u.ac.jp/VFlib/`.

[5] Hirotsugu Kakugawa. VFlib — a general font library that supports multiple font formats. In *Proceedings of the 10th European TEX conference (EuroTEX 98)*, pages 221–222, March 1998.

[6] Donald E. Knuth. *The TEX book*. Addison-Wesley, 1986.

[7] Donald E. Knuth. *TEX: the Program*. Addison-Wesley, 1986.

[8] John Plaice and Yannis Haralambous. The Omega project home page. `http://www.gutenberg.eu.org/omega/`.