VFlib 3.5.0 — a general font library that supports multiple font formats*

Hirotsugu Kakugawa

Research Institute for Information Science and Education
Hiroshima University
1-7-1 Kagamiyama, Higashi Hiroshima, Hiroshima,
739-8521 JAPAN

Abstract. *VFlib is a font library written in C providing several functions to obtain bitmaps of characters (i.e., a rasterizer). VFlib hides the font format of font files and provides a unified API for all supported font formats. Thus, programmers of application software need not worry about font file formats and any software using VFlib can support various font file formats immediately. In addition to this, when a new font format is supported by VFlib, application software need not be modified to use such new fonts.*

VFlib has been developed for not only Latin fonts but also Asian scripts such as Chinese, Japanese, and Korean. Since it is designed as a general font module, it can be used in DVI drivers for TEX and LATEX. In this paper, we explain the API of VFlib, a font database file called vflibcap, and the internal structure of VFlib.

Keywords: digital fonts, computer typography, multilingual typography, multilingual information processing, TEX

1. Introduction

Commercially and freely available fonts exist in many different font file formats. When we develop software to display or print characters which do not depend on a particular window system and/or an operating system, we must write interface routines for accessing font files for each application program again and again. To do this, programmers must have knowledge of font file formats; it will be a difficult task for programmers if the number of font formats that an application program supports becomes large.

* This paper is based on "VFlib — a general font library that supports multiple font formats," by Hirotsugu Kakugawa, EuroTEX98, Saint-Malo, France, 1998.

VFlib is a font library written in C providing several functions to obtain glyphs (bitmaps). VFlib hides the font format of font files and provides a unified API for all supported font formats so that programmers for application software need not worry about font file formats. Thus, any software using VFlib can support various font file formats immediately without modification even when VFlib is updated to support new font file formats. Furthermore, VFlib is not window- or operating-system dependent.

As far as the author knows, there is no general font library other than VFlib that supports multiple font formats in a platform-independent way and that provides a unified API for font access. For example, X Window servers support multiple font formats. But to use a font service, an X server process is required. Some font libraries have been proposed for general use: FreeType 1.1 by David Turner, Robert Wilhelm, and Werner Lemberg is a library for accessing TrueType fonts [10]. The T1Lib by Rainer Menzner is a library to handle Type 1 PostScript fonts [8]. Although both are very useful libraries not dependent on window or operating systems, each of them supports only one font format and has a different API.

Conversion of font formats so that application software can use multiple font formats is one possible approach. For example, `ttf2pk` [6] (TrueType fonts to PK fonts) and `hbf2gf` [5] (HBF[1] fonts to GF fonts), both written by Werner Lemberg, makes these font formats available to TeX. This method is useful but one drawback is that we must convert many font files in advance.

Currently, VFlib supports the following font file formats: PCF, BDF, HBF, TrueType, Type 1, GF, PK, TFM, VF, Syotai Kurabu and JG [2]. To search TeX font files (e.g., PK, GF, TFM, VF files) and TrueType font files, VFlib uses the kpathsea library 3.2 by Karl Berry [1]. VFlib can be used as a font module for drivers and previewers of DVI files generated by TeX and LaTeX.

This paper describes VFlib version 3.5.0[3]. VFlib versions 1 and 2 were designed and implemented for Japanese fonts only; they are widely used in many localized software packages in Japan, e.g. by Ghostscript, dvi2ps, and xdvi for printing Japanese Kanji characters. VFlib version 3 is designed for multilingual document processing in English, French, Chinese, Japanese, Korean, and other languages.

---

1 The Hanzi Bitmap Font (HBF) format [2] is a binary format for bitmap fonts for Japanese, Chinese, and Korean characters.

2 PCF (Portable Compiled Font) format is a binary format for bitmap fonts used on X-Window. BDF (Bitmap Distribution Format) [3] is an ASCII format for distributing binary fonts. Syotai Kurabu, which means *font club* in English, is a vector font format for Japanese Kanji characters. JG format is another vector font format for Japanese Kanji characters.

3 VFlib version 3.2 is introduced in [7] and version 3.3 is reported in [4].

This paper is organized as follows. In section 2, the basic concepts of VFlib are explained. The API of VFlib is shown in section 3, and the font database called vflibcap is explained in section 4. An interesting feature of VFlib is the ability to provide fonts without font files. Section 5 explains this feature. The author has developed several sample programs using VFlib, and one of these is introduced briefly in section 6. Section 7 gives concluding remarks.

2. Basic Concepts

2.1. System components

The VFlib system consists of two parts:

1. A library (`libVFlib.a`)
   It provides several C functions. Any application software using VFlib must link with this library.
2. A font database file (`vflibcap`)
   This file defines fonts and their properties (called *capabilities*), such as point size and the font file format. Its syntax is lisp-like notation.

When we initialize VFlib, we can specify a vflibcap file to be used and thus different font sets can be used by different software.

2.2. Font classes and font drivers

VFlib can handle multiple font file formats. Reading a font file according to the font file format is done by an internal module in VFlib corresponding to its font file format. This internal module is called a *font driver*. Service units provided by font drivers are called *font classes*. From an end-user's point of view, various font formats are distinguished by various names of font classes. Font drivers are internal to VFlib and invisible to end-users.

Some font drivers may not read font files on disk: they may generate glyphs and outlines by internal computation only. In addition, some font drivers may return glyphs which are obtained as glyphs by another font class.

2.3. A view of VFlib font from end-users perspective

Each (virtual) font by VFlib has its inherent information of point size, pixel size, and resolution of the target device. In addition, font metrics are defined for each glyph. Some font file formats may not have such concepts. For instance, TrueType font files are vector font files and do not have information on point size. Syotai Kurabu format fonts do not have font metric information at all. In such case, either (1) the lacking information is given in vflibcap or (2) the specific font driver gives such information as default values.

2.4. Font names and font searching mechanism

In VFlib, a font is specified by a *font name* on opening. First, VFlib checks whether the font name is given in vflibcap or not. If the font name is found, VFlib reads the description for the font in vflibcap. The description contains a font class name; VFlib then invokes a font driver corresponding to the font class name. Finally the font driver opens the font file (if necessary).

If the font name is not given in a vflibcap file, a font searching mechanism is invoked. Since there are many font files for X Window and TEX, this feature is introduced to avoid writing an entry for each font file. Various font drivers will be called to see whether the font can be opened; a list of font drivers for font searching is given in the vflibcap file. If a font driver succeeds in opening the font, font searching finishes and the VFlib font opening function returns successfully. Otherwise, font open fails.

Fonts described in a vflibcap file are called *explicit fonts* and fonts that are searched for by the font search feature are called *implicit fonts*.

2.5. Two modes of opened fonts

The following two modes are provided to obtain glyphs of fonts.

— High resolution device-oriented mode (mode 1)
  The size of glyphs is specified by the physical size of glyphs and device resolution.
— Low resolution device-oriented mode (mode 2)
  Glyph sizes are specified by pixel size rather than by device resolution.

When the size of a glyph in the source font is different from the target size, VFlib scales the source glyph internally. Thus, users need not know know original size of glyphs in font files.

Two modes are provided by the following reason. When we write a application programs that prints documents on printer, it is convenient to specify glyph size by point and device resolution such as glyph of 12 point for a 300 dpi printer. On the other hand, when we write a application programs that displays documents on CRT screen, it is convenient to specify glyph size by pixel.

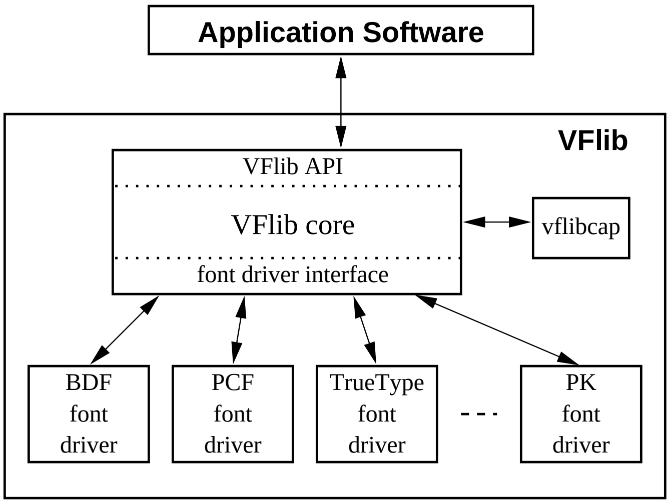

Figure 1: Internal structure of VFlib

2.6. Internal Structure

The VFlib library consists of a *core* and several *font drivers*. The VFlib core provides entry functions of the API, as well as a font driver table, opened font table, vflibcap access module, and other utility modules. The internal structure of VFlib is depicted in Figure 1.

Each font driver has corresponding functions for each font operation of VFlib. Such functions are implemented to provide VFlib API-compatible behaviour. The set of capabilities that can be used for each font class in vflibcap file may differ; each font class defines the capabilities it needs.

3. The API

In this section we describe the API of VFlib. The API that VFlib defines is simple. For example, as a contrast, FreeType defines a rich set of functions including access to kerning information. The simplicity of VFlib API is a result of the limitation that it must be common to every font format that VFlib supports. VFlib does not have features for typesetting such as obtaining kerning information of fonts. But it is enough strong to print and display typesetted documents such as DVI files.

3.1. Data types

VFlib defines the following three data types for font access: bitmaps and metrics for modes 1 and 2.

— Bitmap object
A bitmap object is a set of font metrics and bitmap data. The following is the definition of bitmap structure.

```
struct vf_s_bitmap {
  int            bbx_width, bbx_height; /* in pixels */
  int            off_x, off_y;          /* in pixels */
  int            mv_x,  mv_y;           /* in pixels */
  unsigned char*  bitmap;
  int            raster;
};
```

The members `bbx_width` and `bbx_height` are the width and height of the bitmap, respectively. The members `bitmap` and `raster` are pointers to the glyph data and the number of bytes of a raster. The members `off_x` and `off_y` form a vector from the reference point to the upper-left corner of a bitmap. The members `mv_x` and `mv_y` form a vector to the next reference point. Metric information is given in pixel form.

— Metric object (modes 1 and 2)
Metric objects for modes 1 (high resolution device-oriented mode) and 2 (low resolution device-oriented mode) are defined similarly as bitmap objects except that they do not have `bitmap` and `raster` members. Member types of mode 1 metric objects are `double`, not `int`; their units are points rather than pixels.

3.2. Functions

— `int VF_Init(char*` *vflibcap*`, char*` *variable_list*`)`
Initialize VFlib. The first argument *vflibcap* is the file name of a vflibcap file. The second argument *variable_list* is a list of parameters passed to VFlib for parameterization of vflibcap. (See subsection 4.3 for details of parameterization.)

— `int VF_OpenFont1(char*` *font_name*`, double` *dpi_x*`, double` *dpi_y*`, double` *point_size*`, double` *mag_x*`, double` *mag_y*`)`
Open a font in mode 1. The font name is given by the first argument. Two arguments *mag_x* and *mag_y* are horizontal and vertical magnification factors. The actual font size is determined by these arguments. This function returns a font identifier (font id) for the opened font. All font operations take this font id to specify a target font.

— `int VF_OpenFont2(char *`*font_name*`, int `*pixel_size*`, double `*mag_x*`, double `*mag_y*`)`
Open a font in low resolution mode. This function is similar to `VF_OpenFont1()` except font size is given in pixels.

— `VF_BITMAP VF_GetBitmap1(int `*font_id*`, long `*code_point*`, double `*mag_x*`, double `*mag_y*`)`
Obtain a glyph bitmap of a given font id (in mode 1) and code point. The font id *font_id* must be an id returned by `VF_OpenFont1()`. The size of the bitmap to be obtained can be specified by the *mag_x*, and *mag_y* arguments.

— `VF_BITMAP VF_GetBitmap2(int `*font_id*`, long `*code_point*`, double `*mag_x*`, double `*mag_y*`)`
Obtain a glyph bitmap of a given font (in mode 2) and code point.

VFlib defines other functions such as `VF_METRIC1 VF_GetMetric1()` and `VF_METRIC2 VF_GetMetric2()` to obtain font metrics of a character of mode 1 and 2 fonts, respectively; `VF_OUTLINE VF_GetOutline()` to obtain VFlib format vector data of a character of mode 1 fonts; and `VF_BITMAP VF_OutlineToBitmap()` to convert VFlib format vector data to a bitmap. By calling `VF_InstallFontDriver()`, a font driver is installed.

4. A Font Database File "vflibcap"

A vflibcap file is a database of font definitions for VFlib. It is read when VFlib is initialized. A simple example of a vflibcap file is shown below:

```
(define-default  VFlib
  (uncompression-programs  (".Z" "zcat") (".gz" "gzip -cd"))
  (implicit-font-classes  pcf bdf)
  (extension-hints  (".bdf" bdf) (".pcf" pcf)))

(define-default  bdf
  (filename-extensions ".bdf")
  (font-directories  "/usr/local/share/fonts/X11//")
  (compression-extensions ".gz" ".Z"))

(define-default  pcf
  (filename-extensions ".pcf")
  (font-directories  "/usr/X11R6/lib/X11/fonts//"
                     "/usr/openwin/lib/X11/fonts//")
  (compression-extensions ".gz" ".Z"))

(define-font  timR24    ; Times Roman 24pt, BDF format
  (font-class bdf)
  (dpi 300) (point-size 24)
```

```
  (font-file "timR24.bdf"))

(define-font  timR18    ; Times Roman 18pt, PCF format
  (font-class pcf)
  (dpi 300) (point-size 24)
  (font-file "timR18.pcf"))
```

By a `define-default` construct, a default values of a font class is defined. By a `define-font` construct, a font is defined. Semicolon (`;`) starts a comment. In this example, there are three definitions for font class default (`VFlib`, `BDF`, and `PCF`) and two font definitions (`timR24` and `timR18`). The first three entries are used to give default parameters for VFlib and each font class. We explain these entries later and explain the other two entries first. Although many capabilities are defined, we explain only the fundamental ones.

4.1. Font entries

The entry `timR24` has several capabilities. A capability `font-class` specifies the font class name. In this example, `timR24` belongs to the `bdf` font class. The capabilities `dpi` and `point-size` give device resolution and point size of a font. These values are used when this font is opened in mode 1. A capability `pixel-size` gives the pixel size. This value is used when this font is opened in mode 2. The capabilities `pixel-size`, `dpi`, and `point-size` can be omitted since a BDF font file contains their values. A capability `font-file` gives the file name of the font. Similarly, `timR18` is defined except in cases of PCF fonts.

The two font files `timR24.bdf` and `timR18.pcf` are both bitmap fonts used in X Window. Although pixel size, point size, and target device resolution are together with the bitmap given in the font file, VFlib internally enlarges or shrinks bitmaps to yield the requested size. From a user's point of view, only the font names (`timR24` and `timR18`) are visible; users need not be aware of font formats.

4.2. Default descriptions

In the example above, there are three default descriptions. The first entry `VFlib` is used to give global parameters of VFlib. In our example, the relation between the filename extension and an uncompression program is given by the capability `uncompression-programs`; it specifies that files whose names end in `.Z` and `.gz` are uncompressed by running commands `zcat` and `gzip -cd`, respectively. The capability `implicit-font-classes` specifies a list of font classes that search implicit fonts. When a font is opened and a corresponding entry is missing in vflibcap, font drivers given by this capability are called to search the font in given order. Suppose a font named `timR10.bdf` is requested to open. Since such an entry does not exist in the vflibcap file, the font is searched as an implicit font by calling the PCF font driver first, and then the BDF font driver. A capability `extension-hints` gives a relation of font name extension and font class. In the example, if an extension of a font name is `.pcf` (`.bdf`), the PCF (BDF) font driver is called for implicit font search. For example, if a font named `timR08.bdf` is requested to open, the BDF font driver is called. This is useful for searching implicit fonts fast.

The next definition `BDF` is a default description for the BDF font class. A capability `font-directories` is a list of font directories in which font files are stored. If the directory name is terminated by `//`, files are searched recursively under the directory. The entry `compression-extensions` gives a list of file compressions that BDF font class supports. Similarly, default definition for `PCF` font class is given.

4.3. Parameterized vflibcap

Capability values in vflibcap can be overridden at execution time. By this feature, called *parameterization*, several applications can share the same vflibcap file. The next example is a vflibcap for printer driver for TeX DVI file for 300 dpi Canon Laser Shot. Note that all fonts are implicit fonts.

```
(define-default  VFlib
  (implicit-font-classes  pk)
  (extension-hints  ("pk" pk))
  (variable-values  (TeX_USE_KPATHSEA      "Yes")
                    (TeX_DPI               "300")
                    (TeX_KPATHSEA_MODE     "cx")
                    (TeX_KPATHSEA_PROGRAM  "/usr/local/bin/xldvi"))
  (use-kpathsea           $TeX_USE_KPATHSEA)
  (kpathsea-mode          $TeX_KPATHSEA_MODE)
  (kpathsea-program-name  $TeX_KPATHSEA_PROGRAM) )

(define-default  TeX
  (dpi $TeX_DPI)
  (tfm-directories  "TEXMF" "/usr/local/fonts/tfm")
  (tfm-filename-extensions  ".tfm"))

(define-default pk
  (font-directories  "TEXMF" "/usr/local/fonts/pk"))
```

In the `VFlib` entry, the capability `variables-values` gives a list of variable names and their default values. In this example, there are three variables. For instance, the default value of `TeX_DPI` is `300`. A capability value can be a value of a variable if a dollar sign (`$`) followed by a variable name is given. In the `VFlib` entry, initialization parameters for kpathsea are given.

私は日本人です。

Figure 2: Mixture of *gothic* and *mincho* fonts in Japanese comics

In the `TeX` entry, which is used to specify a common default description for TeX related font classes, the value of the capability `dpi` is given as the value of a variable `TeX_DPI`, for example. Capabilities `tfm-directories` is a list of TFM files directories, and `tfm-filename-extensions` is a extension of a file name for TFM files. The name `TEXMF"` in `tfm-directories` is a special name to seach a file using kpathsea.

The entry `pk` entry is used to specify default description for font files in PK format. A PK file is search by kpathsea first since the first item of the value of the capability `font-directories` is `TEXMF"`. If it is not found, then it is searched in `/usr/local/fonts/pk` directory next.

Default variable values can be overridden by giving a list of pairs of variable names and their values when VFlib is initialized by `VF_Init()`. If a Unix environment variable `VFLIBCAP_PARAM_`*var* (e.g., `VFLIBCAP_PARAM_TeX_DPI` is defined, its value becomes the value of the vflibcap variable `TeX_DPI`.) The example vflibcap file can be used for 600 dpi HP Laser Jet 4 printers if we override variable values so that `TeX_DPI` is `600` and `TeX_KPATHSEA_MODE` is `ljfour` on execution time without any file modification.

5. Font Classes without Font Files

Fonts provided by font classes need not be associated with font files. As examples of such font classes, the *Japanese comic* font class and the *try* font class are implemented.

5.1. The Japanese comic font class

In Japanese comics, a gothic font is used for Kanji characters and a *mincho* font is used for Kana characters as shown in Figure 2. Without creating a new font, we can implement such fonts by the Japanese comic font driver: a font F of the Japanese comic class needs two sub-fonts that are specified in a vflibcap file; fonts for Kanji characters, say F_1, and Kana characters, say F_2. A glyph for code point c of font F is obtained from F_1 if c is a Kanji character, and from F_2 if c is a Kana character. The font classes of F_1 and F_2 need not be the same.

5.2. The try font class

The try font class provides a feature to open a font among list of fonts. Each font f of the try font class has a list of fonts, say $g_1, g_2, \ldots$. When f is requested to open, the font driver tries to open fonts $g_1, g_2, \ldots$ in this order until one of them, say g_s, is opened successfully. Then, any font operation on f is applied on g_s and therefore, g_s is used as a font f.

This font class is useful when we want to use non-standard (i.e., site dependent) fonts. Suppose that a non-standard font f_n is listed followed by a standard font f_s in a font list of f of try font class. Then, to open f is the same as opening f_n if it is avaiable; otherwise, f_s is used. Since f_s is a standard font, opening f always succeeds. Candidate fonts $g_1, g_2, \ldots$ can be a fonts of different font classes; for instance, g_1 can be a TrueType font and g_2 can be a PCF font.

6. Sample Programs

VFlib is distributed with sample programs. One of them is a previewer for DVI files. In a DVI interpreter, a TEX font is opened by the following sequence.

```
sprintf(f_name, "%s.pk", name);
fid = VF_OpenFont1(f_name, h_dpi, v_dpi, -1, mag, mag);
```

The variable `name` is a font name as appearing in a DVI file (e.g., `cmr10`); `h_dpi` and `v_dpi` are the horizontal and vertical device resolution of the target device in dpi, respectively. The variable `mag` is a magnification factor. The PK font driver finds an appropriate font file from parameters given in `VF_OpenFont1()` and the default values given in `TeX` entry in vflibcap. For instance, if the font name is `cmr10.pk` and `h_dpi` = `v_dpi` = 300, and `mag` = 1.2, the PK font driver looks for the font file `cmr10.360pk`. Glyphs are simply obtained by calling the `VF_GetBitmap1()` function.

Figure 3 shows a screen shot of a sample previewer on X Window using Motif. This previewer supports drawing EPS figures and color changes.

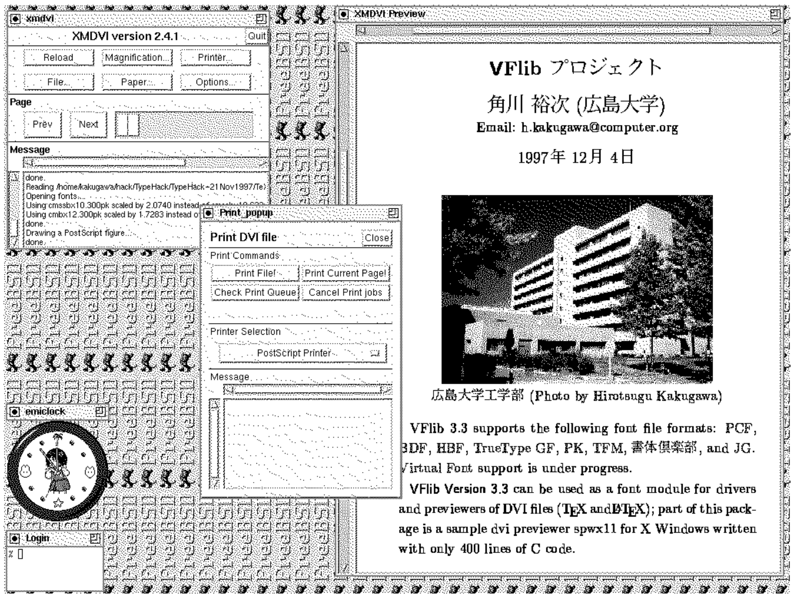

Figure 3: A DVI file previewer xmdvi

7. Conclusion

In this paper, we have introduced a font library VFlib which supports multiple font formats with a unified API. It is especially useful for DVI drivers.

VFlib has been tested on FreeBSD 2.2.2 and Linux for Pentimun, Solaris 2.5.1 and SunOS 4.1.4 for SPARC, and SunOS 4.1 for SPARC; there is no difficulty to port it to other Unix-like operating systems. The source code and the latest information on VFlib is available at `http://www.se.hiroshima-u.ac.jp/~kakugawa/VFlib/`.

Acknowledgements

The author would like to thank Werner Lemberg for helpful comments on specification and implementation of VFlib. He also thank Ken'ich Handa and Satoru Tomura for valuable discussions.

Bibliography

[1] Karl Berry. Kpathsea version 3.2. Included in web2c 7.2, which is available electronically from CTAN, `systems/web2c/web2c-7.2.tar.gz/`, March 1998.

[2] Nelson Chin et al. Hanzi Bitmap Font (HBF) file format version 1.1. Available electronically from `ftp://ftp.ifcss.org/pub/software/info/HBF-1.1.tar.gz`, September 1994.

[3] Adobe Systems Incorporated. Glyph Bitmap Distribution Format (BDF) specification version 2.2. Available electronically from `http://www.adobe.com/supportservice/devrelations/PDFS/TN/5005.BDF_Spec.pdf`, 1993.

[4] Hirotsugu Kakugawa. Vflib — a general font library that supports multiple font formats. In *Proceedings of EuroTEX 98*, March 1998.

[5] Werner Lemberg. hbf2gf. Available electronically from CTAN, `language/chinese`.

[6] Werner Lemberg. ttf2pk. Available electronically from CTAN, `language/chinese`.

[7] Werner Lemberg. New font tools for TEX. In *Proceedings of TUG 97*, July 1997.

[8] Rainer Menzner. A library for generating character bitmaps from adobe type 1 fonts. Available electronically from `ftp://ftp.neuroinformatik.ruhr-uni-bochum.de/pub/software/t1lib/t1lib-0.7.1-beta.tar.gz`, 1998.

[9] Microsoft corporation. TrueType 1.0 font files. Available electronically from `ftp://ftp.microsoft.com/developr/drg/TrueType/ttspec.zip`, November 1995.

[10] David Turner, Robert Wilhelm, and Werner Lemberg. The FreeType package 1.1. Available electronically from `http://www.freetype.org/`, 1998.