ベクトルフォントライブラリ

VFlib 第2版
利用者マニュアル

角川裕次　広島大学
h.kakugawa@computer.org
http://www.se.hiroshima-u.ac.jp/~kakugawa/

1994 年 1 月 1 日

最終改定 VFlib 2.25.0 / 1999 年 9 月 6 日

配布条件

©1993 – 1999 角川裕次

本文書は、**GNU General Public License** に従って利用して下さい。

無保証

VFlib 及び本マニュアルは無保証です。**VFlib** を利用することで生じた一切の損害等に対して、作者は責任を負いません。各自の責任の範囲内で御利用下さい

注意

VFlib 2.24.2 より、インストールに関して以下の変更を行いましたので、注意してください。

- FreeType ライブラリは、configure スクリプトで指示しないと組み込まれない。
- 共有ライブラリのバージョン番号付け規則を、GNU libtool の推奨する方針に従うようにした。

1 概要

本マニュアルでは、 VFlib 第 2.25.0 版 の概要、 並びに使用法を解説します。VFlib はベクトルフォントからビットマップデータに変換するためのライブラリで、文字の回転、 縮小、 斜体化などを行なう機能も用意されています。

ベクトルフォントとは、 画像データではなく、 文字の輪郭をデータとして保持している書体のデータ表現法です。 輪郭で表現されているために、1 つのアウトラインフォントを元にして、斜体や白抜きにした文字の画像データを生成することが容易という特徴を持ちます。一方、 画像データそのものを保持する方法で、大きな文字の画像データを作ると、 ギザギザが拡大され醜いという欠点があります。今日、 多くの会社が色々なベクトルフォントを発売していますが、データ形式がまちまちです。しかし VFlib では、 複数のデータ形式及び、 複数のベクトルフォントを混合し、データ形式の差を全く気にすることなく、同一の呼び出し形式で使うことができます。

VFlib 第 2.25.0 版では、以下の形式のフォントが利用可能です。

- Sony 社の NWF-605, NWF-629
  SONY 社ワークステーション NEWS 用のベクトルフォント[1]。(ソニー平成ゴシックだと bus error がでる、という報告があります。そのほかの標準明朝や平成明朝などはだいじょうぶです。)
- Zeit 社の「書体倶楽部」フォント。
- Zeit 社の「JG Font」フォントシリーズ (販売は ASCII 社)
  ワードプロセッサ「JG」用のベクトルフォント。
- TrueType フォント
  Microsoft Windows 3.1, Windows95 などで使われている日本語漢字

[1] NEWSOS Release 4.2.1C/R/RN/aRD/aRS での FSlib にバグがあります。オープンとクローズを繰り返すとフォントディスクプリタがどんどん増加し、64 以上になると異常終了するそうです。詳細な情報とパッチは、 http://www1.sony.or.jp/TechSupport/Patch/ にあります。

フォント。(すべての TrueType フォントが利用できる訳ではありません。動作確認が取れているフォントについては、7 を参照して下さい。)

- FontWave フォント
- BDF フォント
  X Window Version 11 で使われる、BDF 形式のビットマップの日本語漢字フォント。
- HBF フォント
  HBF (Hanji Bitmap Font) 形式のビットマップの日本語漢字フォント。

本ライブラリは C 言語で記述され、 ソースコードの形式で提供されます。新たなフォントドライバの組み込みも非常に容易になっています。

現在のところ、 以下のシステムでの動作が確認されています。

- Sun SPARC Station 2, 5, 20, ELC, IPX : SunOS 4.1
- Sun SPARC SPARC Station : SunOS 5.3, 5.4, 5.5
- Sun SPARC Station Classic : SunOS 5.1
- 富士通 DS/90: UXP/DS V20L10 (SVR4.2)
- ソニー NEWS 1400, NEWS-OS 4.1C
- Omron Data General AV-300, : DG/UX 5.4.1
- IBM PC compatibles: Linux-1.1.12
- IBM PC compatibles: FreeBSD 2.2.2 および 3.2
- IBM PC compatibles: OS/2.

本マニュアルは、 以下の構成となっています。第 2 章では、本マニュアルで使用する用語の定義、並びに基礎的な概念の説明を行ないます。第 3 章では、 VFlib のコンパイル方法、並びにインストール方法の解説を行ないます。第 4 章では、 VFlib を用いてのプログラミング方法を、例を用いて解説します。第 5 章では、ビットマップを描く対象のメモリである、フレームバッファの構造について説明します。第 6 章では、使用するフォントを記述するファイル vfontcap の書き方を解説します。第 7 章では、VFlib で取り扱えるフォントについて説明します。第 8 章では、 VFlib が新しいフォントを取り扱えるようにするための方法を解説します。

2　準備

本章では、 このマニュアルで用いる用語の定義とそれらの説明を行ないます。

ベクトルフォント (*vector font*) とは、文字等の形を、 輪郭線を用いて表現したものです。このベクトルフォントを画像データに変換したものは ビットマップ (*bitmap*) と呼ばれます。得られたビットマップは、 フレームバッファ (*frame buffer*) と呼ばれるメモリ領域に書き込まれます。 このフレームバッファは、ビットマップウインドウ用のビットマップ領域とか、プリンタの出力イメージ用のメモリ領域などです。

1 つのベクトルフォントから、 回転した文字や、 斜体文字を作ることが可能です。これは、 輪郭線データの座標変換を等を行なうことで容易に実現ができます。すなわち、 1 つのアウトラインフォントから、複数の形状のビットマップの生成が行えます。ベクトルフォントから、ビットマップを定めるために与えられた変形の指定を属性 (*capability*) と呼びます。

フォントエントリ (*font entry*) とは、ベクトルフォントと属性を与えることで定義されます。このため、 ベクトルフォントは同一でも属性が異なれば、VFlib では別のものとして取り扱います。フォントエントリの定義は、**vfontcap** ファイルと呼ばれるデータベースファイルに記述されます。

フォントエントリは、 いくつかのグループに分かれます。このグループをフォントクラス (*font class*) といい、例えば TruType クラスとか、JG フォントクラスなどがあります。指定できる属性と、属性値に対する解釈は、フォントクラス毎に定まっています。

フォントエントリと似た概念として、 フォント (*font*) があります。VFlib では使用するフォントエントリを指定するのに、フォントエントリ名で指定するのではなく、 フォント名を与えて指定します。VFlib は、 (ある定められた規則により) 与えられたフォント名からそれに一致するフォントエントリを探します。例えば、「フォント名が “min12.400pk’ である時には、フォントエントリ “min” を用いる」、 などです。フォント名からそれに一致するフォントエントリの探し方は、利用者が指定することができます。フォントとフォントエントリは厳密には異なる概念ですが、本稿では特に断らない限り同一視します。

3　VFlib のインストール方法

VFlib のインストールは以下の順で行なって下さい。以下の例では、 VFlib 2.25.0 の場合の配布ファイルでの例を示しています。ここでは、Unix 系のオペレーティングシステムでのインストール作業を想定しています。他のオペレーティングシステムでは異なことがありますので、ご注意下さい。

1. **FreeType** ライブラリのインストール
   標準の VFlib では、TrueType フォントのラスタライズに、FreeType ライブラリ 1.1 版または 1.2 版を使用します。VFlib のインストールに先立ち、 FreeType 1.1 または 1.2 をインストールしておいて下さい。なお、 FreeType 1.0 では正しく動作しないので、かならず FreeType 1.1 か 1.2 を使用して下さい。FreeType は、`http://www.freetype.org/` より入手できます。
2. アーカイブの解凍
   VFlib のソースコードは、 は複数のファイルより構成されていますが、配布の際にはこれらを 1 つのファイルにまとめています。

   - ファイル名が `.tar.gz.uu` で終る時。
     uuencode プログラムによって符合化されていますので、 これを uudecode プログラムを使って復号化します。

     ```
     % uudecode VFlib2-2.25.0.tar.gz.uu
     ```

     すると、 `VFlib2-2.25.0.tar.gz` というファイルができます。次の項に進んで下さい。
   - ファイル名が `.tar.gz` で終る時。
     GNU gip (コマンド名は gunzip) を用いて圧縮されたアーカイブファイルを展開します:

     ```
     % gunzip VFlib2-2.25.0.tar.gz
     ```

     これで圧縮されたアーカイブファイルが展開されました。アーカイブファイルを解凍するには、 次の項に進んで下さい。
   - ファイル名が `.tar` で終る時。
     tar コマンドでアーカイブファイルを解凍します。

     ```
     % tar xvf VFlib2-2.25.0.tar
     ```

     これで VFlib のソースコードが解凍されます。

   以上で VFlib のソースコードが解凍されました。以下より、コンパイルを開始します。
   まず、VFlib のソースコードディレクトリに移動します。

   ```
   % cd VFlib2-2.25.0
   ```

   次に configure スクリプトを実行し、make を実行します。

   ```
   % ./configure
   % make
   ```

もし FreeType ライブラリを組み込んだ VFlib を構築するには、configure の起動を下記のようにします。

```
% ./configure --with-freetype \
              --with-freetype-includedir=/usr/local/include \
              --with-freetype-libdir=/usr/local/lib
```

もし Sony の FSlib を組み込んだ VFlib を構築するには、上に示した configure の起動を、下記のようにします。

```
% ./configure --with-fslib
```

configure には、以下のオプションが定義されています。

- `--enable-shared`
  — VFlib の共有ライブラリ版を作ります。ただし、共有ライブラリをサポートしているシステムでは、デフォルトで共有ライブラリを作り、静的ライブラリは作りません。
- `--disable-static`
  — VFlib の静的ライブラリ版を作らないよう、コンパイルします。ただし、共有ライブラリをサポートしているシステムでは、デフォルトでは静的ライブラリは作りません。
- `--with-fslib`
  — Sony FSlib を VFlib に組み込みます。デフォルトでは組み込みません。
- `--with-freetype=`*LIB*
  — FreeType ライブラリを使うよう、VFlib をコンパイルします。*LIB* は FreeType のライブラリ名で、実際のライブラリのファイル名は、libLIB.a や libLIB.so となります。何も指定しないと ttf が指定されたと見做されます。
- `--with-freetype-includedir=`*DIR*
  — FreeType のヘッダファイルのあるディレクトリを指定します。
- `--with-freetype-libdir=`*DIR*
  — FreeType のライブラリファイルのあるディレクトリを指定します。

3. ディレクトリの説明

- ReleaseNote — 改良した点の履歴。
- doc/ — VFlib のマニュアルのあるディレクトリ
- jTeX/ — ASCII 社による日本語 TEX で各種書体を使うために必要

な JFM ファイルと、その自動生成スクリプト群がおいてあるディレクトリ。

- tools/ — VFlib の動作確認を行なうためのツール群のあるディレクトリ。文字の一覧表示を行なう ktest、フォントデータ逆アセンブラ disol などがある。
- src/ — VFlib のソースコードディレクトリ。
- vfontcap — サンプルの vfontcap ファイル。

4. インストール
   完成した VFlib をインストールするには、

   ```
   % make install
   ```

   を実行して下さい。たぶん、スーパーユーザーでないと目的のディレクトリに書き込めないでしょうから、スーパーユーザーにお願いしてインストールを行なって下さい。
   `libVFlib2.a` が、 ライブラリのディレクトリ (デフォルトでは `/usr/local/lib`) の下に `libVFlib2.a` という名前でインストールされます。共有ライブラリをサポートするシステムでは、共有ライブラリ形式 (`libVFlib2.so`) での VFlib もインストールされます。インクルードファイル `VF.h` は、インクルードファイルのディレクトリ (デフォルトは `/usr/local/include`) にインストールされます。TryeType フォントのインデックスファイル作成プログラム ttindex、BDF フォントのインデックスファイル作成プログラム bdfindex は、実行ファイルのディレクトリ (デフォルトは `/usr/local/bin`) にインストールされます。フォントデータベースファイル `vfontcap` は、デフォルトでは `/usr/local/lib/VFlib/2.25.0/` にインストールされます。

4 VFlib の使用方法

4.1 使用例

プログラムでの VFlib の使用方法を、例を用いて説明します。VFlib は、

- VFlib の初期化,
- フォントのオープン,
- ビットマップの取得 (あるいは, アウトラインデータの取得),
- フォントのクローズ,
- VFlib の終了.

といった流れで使用します。以下に具体的な例を示します。

```
#include "VF.h"
foo(code, x, y, w, h, rast, buff)
  int  code, x, y, w, h, rast;
  char *buf;
{ int Fd;

  VF_Init(NULL);
  Fd = VF_OpenFont("min");
  VF_GetBitmap(code, Fd, w, h, rast, x%8, &buff[x/8+rast*y]);
  VF_CloseFont(Fd);
  VF_Deinit();
}
```

この例では、 buff で示されるメモリ領域 (ラスタ間距離は rast バイト) の座標 (x, y) を左上とし、min フォントの文字コード code のビットマップを書き込みます。(座標 (x, y) の点のあるメモリアドレスと、座標 $(x, y+1)$ の点のあるメモリアドレスとの差が r であるとき、ラスタ間距離が r である、といいます。) フレームバッファ(文字のビットマップを描くメモリ領域) の構造は、第 5 章を読んで下さい。

VFlib を使用するプログラムでは、 各種の定数の宣言や、関数の型宣言を記述しているファイル VF.h を #include でインクルードして下さい。コンパイルする時は、コンパイラの -I オプション (unix cc の場合) で VF.h の置いてあるディレクトリを指定する必要があるでしょう。

VFlib 配布ファイルの中のディレクトリ /tools の下にある vftest.c, ktest.c ならびに disol.c が、 VFlib 2 を使ったプログラム例です。詳細はこれらを参考にして下さい。

VFlib を使用したプログラムのリンクでは、libVFlib2.a をリンクします。(共有ライブラリが利用可能なシステムでは、libVFlib2.so をリンクします。) もし Sony フォントも使うようにした VFlib のときは、libFSlib.a もリンクして下さい。また、FreeType を使うフォントドライバ freetype を使う場合は、FreeType ライブラリ libttf.so または libttf.a もリンクする必要があります。(標準のインストールでは FreeType を使うよう VFlib が構築されますので、libttf.so または libttf.a は忘れずにリンクして下さい。)

FreeType を使用するかどうかとか、VFlib のヘッダーファイルのディレクトリなどの情報は、インストールした後には忘れてしまうことがよくあります。そのため、どのような構成で VFlib がインストールされた、どのようなライブラリをリンクすべきかなどの情報を取出すプログラム VFlib2-config が用意されています。(VFlib2-config は VFlib 2.25.0 より導入されました。)

コンパイル例は、以下の通りです。

```
> cc foo.c `VFlib2-config --cflags --libs` -lm
```

VFlib2-config を --help オプションで起動すると、その使い方が表示されます。

4.2 VFlib 提供関数と制御変数の一覧

以下に、 VFlib で用意されている C 言語関数の一覧を示します。VFlib version 1 では、漢字コードの指定に「連続漢字コード」を使っていましたが、version 2 からはフォントクラス依存という仕様になりました。ですが、現在用意されているフォントクラス SONY NWF, 書体倶楽部, JG Font, TrueType いずれも、JIS コードを使って漢字の指定をします。

- **VFlib の初期化**
  関数 int VF_Init(char* vfcap) を呼ぶことで、VFlib の初期化を行ないます. 引数 vfcap で、 vfontcap ファイルのパスを与えます。なお、 vfontcap で示される文字列は、 必ず / で始まらないといけません。(すなわち、 完全パスを与えないといけない、ということです。) もし vfcap に NULL ポインタを与えると、デフォルトの vfontcap ファイルのパスが使われます。もし VF_Init 返り値が負の数の時はエラーで、それ以外は正常に終了したことを表します。

- **VFlib の終了**
  関数 int VF_Deinit(void) を呼ぶことで、 VFlib の終了を行ないます。もし VF_Init 返り値が負の数の時はエラーで、それ以外は正常に終了したことを表します。

- フォントのオープン
  フォントのオープンは、関数 int VF_OpenFont(char* font) を呼び出します。ここで font はフォント名の文字列を表します。もしオープンに成功すれば、 0 以上の数が返されます。ここで返された値をフォント記述子といいます。以降はフォント記述子によって、用いるフォントを指定します。オープンに失敗すると、 負の整数が返されます。

- フォントのクローズ
  関数 int VF_CloseFont(int fid) を呼び出すことで、フォント記述子 fid のフォントのクローズを行ないます。返り値が負の数の時はエラーで、 それ以外は正常に終了したことを表します。

- 全てのフォントのクローズ
  関数 void VF_CloseAllFonts(void) により、オープンされている全てのフォントをクローズします。 返り値はありません。

- アウトラインデータの取得
  関数 long* VF_GetOutline(int code, int fid) によって、フォント fid の 文字コード code に対するアウトラインデータの取得を行ないます。返り値は、取得したアウトライン情報が格納されているフレームバッファへのポインタです。もし NULL ポインタが返されれば、取得に失敗したことを表します。取得したデータは VF_FreeOutline で解放して下さい。アウトラインデータの形式は、 次節の “アウトラインのデータ形式” を参照して下さい.

  注: フォントによっては、 輪郭線の回転方向 (時計周り/反時計周り) の区別がないものがあります。回転方向に依存したラスタライズルーチンを使う場合はご注意下さい。

- アウトラインデータの解放
  関数 int VF_Free""Outline(""long *vfdata,"" int fid により、VF_GetOutline で取得したアウトラインデータの解放を行います。

- ビットマップの取得 (文字コードより)
  関数 int VF_GetBitmap(int code,int fid,int w,int h,int bw, int bo,char*bm_buf) により、 フォント記述子 fid での 文字コード code に対するビットマップを得ます。得られるビットマップの大きさは、横 w ドット、 縦 h ドットです。ビットマップは bm_buf から始まるフレームバッファに書き込まれます。bw はフレームバッファのラスタ間距離 (1 行当たりのバイト数) で、bo はビットオフセットで、これで指定されたドット数だけ右にずらして、ビットマップがフレームバッファにおかれます. (bo の値は、 0 以上 7 以下でないといけません。) なお、 漢字コードの指定 code は JIS コードで行ないます。(厳密にいえば、 文字コード体系はフォントクラスに依存しますが、SONY、 書体倶楽部、 JG Font、 TrueType の各フォントクラスでは、漢字コードに JIS コードを用いています。)

- ビットマップの取得 (アウトラインデータより)
  関数 int VF_DrawOutline(long *vfdata, int fid, int w, int h, int bw, int bo, unsigned char *bm_buf) により、 アウトラインデータ vfdata のビットマップをフレームバッファに書き込みます。意味およびその他の引数は、 VF_GetBitmap と同じです。

- フォント名からフォントエントリ名への変換関数の設定
  関数 void VF_SetFn2""EntFunc(char"" *(*func)()) は、フォント名からフォントエントリ名への変換を行なう関数として、引数 func で与えられた関数を用いるように設定します。なお、 変換関数として用いられる関数 func は、char* func(char*) の型を持っていないとい

けません。すなわち、 func は引数として (文字列で) フォント名を受けとり、返り値としてフォントエントリ名に変換した文字列を返します。

なお、 func で指される関数は、引数で与えられた文字列を書き換えてはいけません。一方 func を呼び出した側は、 func から戻った直後に変換された結果を別の領域に複写しますので、func で指された関数は、呼ばれる都度新たにメモリを確保する必要はありません。

VFlib では変換関数として、 以下の 2 つをあらかじめ準備しています.

- VF_Fn2Ent_AsItIs — 変換を行なわない関数です。フォント名とフォントエントリ名が同一である時に用います。
- VF_Fn2Ent_TeX — 最初に数字が現れたところで文字列を切る変換を行なう関数です。これは、 例えば、 フォント名 min12 に対し、min と変換します。VFlib を TEX のデバイスドライバで用いる時に使用すると便利です。

なお、 何も指定をしなければ、 VF_Fn2Ent_AsItIs がデフォルトの変換関数として用いられます。

- オープンするファイルの最大数の指定
  変数 int VF_MaxOpenFontFiles に、オープンするファイルの最大数の指定をします。この変数の初期値は、コンパイル時に指定した数です。(ですのでなにも変更しない時は、この変数の値を変更する必要はありません。) この数の変更は、 VF_Init() を呼ぶ前に行なって下さい。

  オープンするファイルの最大数とは、ライブラリ関数 fopen() や、システムコール open() にて同時にオープンされるファイルの数の最大数であって、VF_Open() にて同時にオープンされるフォントの最大数ではありません。もしこの数を越えてファイルをオープンする時は、使用の度合の少ないファイルを VFlib 内部でクローズします。そしてその後そのファイルがアクセスされると、VFlib 内部で他のファイルをクローズし、アクセスしようとしているフォントファイルを再びオープンします。これの機能は、多くのフォントを同時に使用する時に、ファイルディスクプリタが不足しないようにするためのものです。

4.3 アウトラインのデータ形式

ここでは、 アウトラインデータの形式について解説します [1]。アウトラインデータは long* 型のオブジェクトで、long 型オブジェクトの列で構成されています。オブジェクトには、 次の 3 種類があります。

- ヘッダ — 2 つのオブジェクトより構成されます。最初のオブジェクトは文字コードです。次のオブジェクトはアウトラインデータの座標形式

が整数の形でおかれています。VF_GetOutline でアウトラインを取得する限りは、常に 1 となっています。(各フォントの座標形式に従ったアウトラインデータ取得機能の追加を、将来の拡張で予定しています。)

- トークン — これは、 閉曲線の開始とか、 直線の開始といったものです。トークン以降には点データの列が来ますが、トークン以降の点データが、 例えば、 閉曲線を構成する点となります。
- 点データ — 閉曲線や、 ベジェ曲線を構成する点です。1 つのオブジェクトに、 x, y 2 つの値が符合化されています。

アウトラインデータは、 一番最初に文字コードが置かれます。次に、 点の座標形式がおかれます。そしてそれ以降に、 図形オブジェクトが並びます。図形オブジェクトはトークンで始まる、 一連の点データより構成されます。そして一番最後に 0 が置かれ、 データ終了となります。

トークンには、 以下のものがあります。

トークンの印	0x80000000
文字データの開始	0x80000001
時計周りの閉曲線の開始	0x80000002
反時計周りの閉曲線の開始	0x80000004
直線の開始	0x80000008
円弧の開始	0x80000010
ベジェ曲線の開始	0x80000020

点データは、 座標を (x, y) とすれば、((x)<<16) | (y)) と符合化されています。但し、 x, y ともに、 $(32768 - 8192)/2$ 以上 $(32768 + 8192)/2$ 未満の範囲にないといけません[2]。

サンプルプログラム disol は、指定したフォントの指定した文字コードのアウトラインデータを取得し、それを解読する “逆アセンブラ” です。これを参考にすると良いでしょう。

5 フレームバッファの構造

VFlib で想定しているフレームバッファの構造を、図 1 に示します。座標は左上を原点とし、 右へ行けば x は増加、 下に行けば y が増加します。フレームバッファは unsigned char* の配列で、配列の先頭が buff、 横幅が

[2]SONY のマニュアル [1] には、この値の範囲はフォントファイル毎に異なり得る、と記述されています。しかし、FSlib のソースでは、点の範囲は固定として実装されています。はて??? (明朝体、ゴチック体のフォントファイルはともにこの値をとっていますが....)

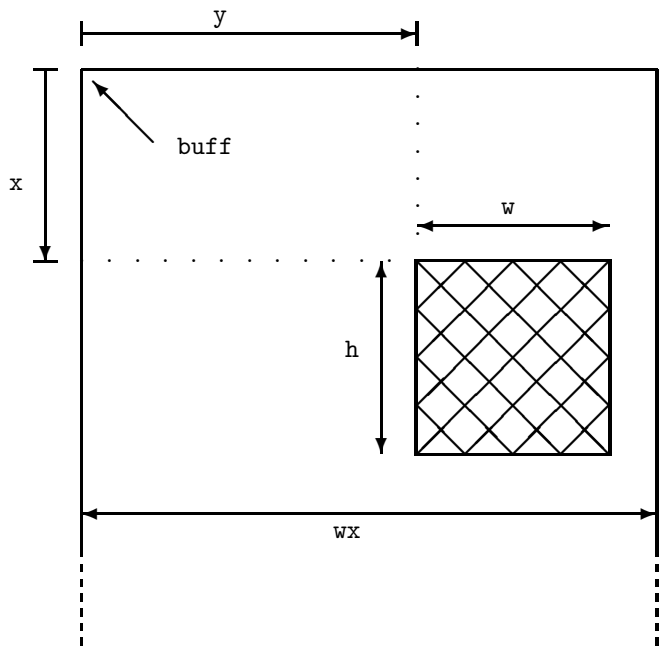

Figure 1: フレームバッファの構造

wx ピクセルあると仮定します。すると、 横 1 行 (1 ラスタ) の占めるメモリは $\lfloor(\mathrm{wx}+7)/8\rfloor$ バイトとなります[3]。以降、 $\mathrm{rast} = \lfloor(\mathrm{wx}+7)/8\rfloor$ とします。

横 w ピクセル、 縦 h ピクセルの大きさのビットマップを、フレームバッファの位置 (x, y) を左上として書き込む場合を考えます。このときの書き込まれるアドレスは、 buff + y*rast + x/8 で求まります。(C 言語では、`&buff[y*rast+x/8]` となります。) このアドレスを以下では ptr とおきます。

次に 1 byte 内でのピクセルの構造を説明します。図 2 を参照して下さい。1 byte は 8 ピクセルより構成されます。一番左の bit が bit 7 (MSB) で、1 byte 内での一番左側のピクセルです。逆に、 一番右の bit が bit 0 (LSB) で、 1 byte 内での一番右側のピクセルです。1 byte 内でのビットの位置は、$\mathrm{x} \bmod 8$ で計算できます[4]。図 2 の例では、$\mathrm{bo} = 2$ (MSB のとき、 $\mathrm{bo} = 0$) となります。

以上のことより、関数 `int VF_GetBitmap(code, fid, w, h, bw, bo, bm_buf)` を呼び出すには、 `bw` に rast を、 `bo` に bo を、`bm_buf` に ptr を与えれば良いことになります。

[3] $\lfloor\alpha\rfloor$ は、 α を越えない最大の整数を表します。例えば、 $\lfloor(14+7)/8\rfloor = \lfloor 21/8\rfloor = 4$ です。
[4] $x \bmod n$ は、 x を n で割った余りです。

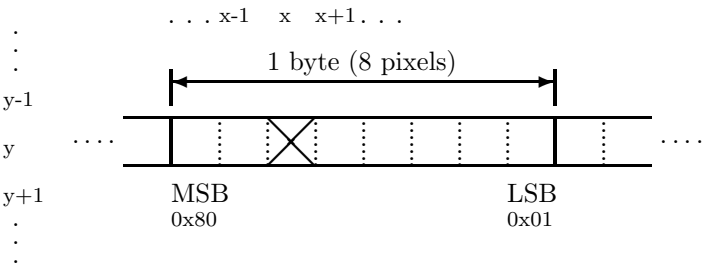

Figure 2: 語とピクセルの関係

6 vfontcap ファイル

6.1 基礎と例

フォントエントリが具体的にどのフォントファイルより生成され、そしてどのような属性を持つかは、 vfontcap ファイルというフォントエントリに関するデータベースファイルに記述されます。vfontcap ファイルは、 termcap ファイルに似た形式で記述されます。(事実 vfontcap ファイルアクセス部は、GNU termcap を改造して実現されています。)

vfontcap の書き方を、 例を用いて示します。

```
##  SONY FONT
r_sony_mincho|Sony Vector Font (Mincho):\
        :ft=sony:\
        :ff=/usr/local/font/devnwp533/Mincho.vfont:
min:\
        :fc=r_sony_mincho:
##  NANAME MOJI
minsl|minslR:\
        :sl#10:fc=min:
```

最初の文字が # で始まる行はコメント行です。ですので、## SONY FONT はコメント行です。エントリ名は行の最初の位置から書き始めます。1 つの定義に対して、複数のエントリ名を書くことも出来ます。例えば minsl|minslR:\ では、エントリに 2 つの名前 (minsl と minslR) を与えています。縦棒 | は、エントリ名とエントリ名の区切りの文字です。最後にコロン (:) を書くことで、エントリ名の並びの終りを表します。それ以降には、属性の記述が置かれます。

属性の記述は属性の並びで、属性と属性の間はコロン (:) で区切られます。タブや改行があっても構いません。もし記述が 1 行に収まらない時は、 行の最後に \ を書くことで、次の行に続きを書くことが出来ます。\ のすぐ後で

改行しないといけない点に注意して下さい。

読みやすさのために、さらに行を継続する時は行の終りを :\ とし、改行に続いてタブ、: とすることを奨めます。(r_sony_mincho の定義を参照して下さい。) このように属性部分をタブ文字で字下げすることで、フォントエントリと区別しやすくなり、 読みやすい記述となります。

属性には、 真理型、 整数型、 文字列型の 3 種類があり、属性名はいずれもアルファベット 2 文字です。真理型は真または偽の値をとり、 エントリに現れれば真、そうでなければ偽となります。整数値は、 例えば :sl#10: のように、 属性名の後に# を書き、その後に数を書きます。文字列は :ft=sony:\ のように、属性名の後に = を書き、 コロン (:) までがその値となります。もし文字列にコロンを入れたいときは、: を書くのではなく、\072 と書いて下さい。また、 バックスラッシュ \ そのものを書きたい時は、\\ としてください。

6.2 属性の一覧

先に用語を説明しておきます。

必須属性— 必ず指定しないといけない属性値です。(tc にて継承されていても構いません。)

アウトライン属性— VF_GetOutline() にて文字のアウトラインデータを得る時に有効となる属性です。例えば ro が指定されているエントリにて、 VF_GetOuitline() で取得したアウトラインデータは、元の文字を回転したアウトラインデータとなっています。

ビットマップ属性— ビットマップを生成する時に有効となる属性です。この属性は、 VF_GetOuitline() で取得するアウトラインデータには全く影響を与えず、VF_DrawOutline() または VF_GetBitmap() が呼ばれた時にこの属性が意味を持ちます。

属性の一覧を示します。フォントクラス固有の属性は、後に出てくるフォントクラスの説明のときに説明します。ここで説明する属性は多くのフォントクラスで使われますが、なかには使えないフォントクラスもあります。

注: 現在の実装上の制約より、 整数型の属性値に -1 を指定できません。(-1 を指定すると、 無指定とみなされてしまいます。)

注 (**MS-DOS** での利用時): MS-DOS でのフォントファイルパスが例えば c:/font""/micho の場合、単純に ff=c:/font/micho としてはいけません。何故なら、 ドライブとファイルとの区切りである : が属性記述の区切り文字とみなされてしまうからです。このときは面倒ですが、 ff=c\0x72/font

/micho として下さい。(コロン : を その文字コードで記述し、問題を回避します。)

- ft — フォントタイプ。 (文字列、 必須属性)
  フォントの型を指定します。この値として、sony, zeit, jg, truetype, bdf, hbf, compound, bitop, null が指定可能で、これらのうちのいずれかを指定してください。 (例: ft=sony)

- ff — フォントファイルのパス名。 (文字列)
  フォントファイルのパス名を指定します。(例: ff=/usr/font/Mincho.vfont) フォントクラスによっては、拡張子をパス名に含めないものもあります。

- tc — 継承。 (文字列)
  フォントエントリ毎に、 全ての属性を記述するのは大変です。良く似た定義があればそのフォントエントリの定義に、あらたな別の属性を追加することが出来ます。これが属性の継承です。この属性は、継承するエントリを指定します。継承の記述は、エントリの属性記述の最後の位置に置かないといけません。(例: tc=minsl)

- ro — 回転 (整数、 デフォルト値は 0、 アウトライン属性)
  指定した回数だけ 90 度時計回りに回転されます。 (例: ro#1)

- rx — 左右反転 (真理値、 アウトライン属性)
  これが定義されれば左右反転されます。 (例: rx)

- ry — 上下反転 (真理値、 アウトライン属性)
  これが現れれば上下反転されます。 (例: ry)

- sl — 傾き値指定。 (整数値、 デフォルト値は 0、アウトライン属性)
  文字の傾きを指定します。 0 だと傾きません。正の値だと右へ、 負だと左へ傾きます。-100 から 100 までの範囲でないといけません。 (例: sl#10)

- fr — 枠のみ。 (真理値、 ビットマップ属性)
  これが現れれば白抜き文字になります。 (例: fr)

- ox — X 方向のオフセット (整数)
  解釈はフォントクラス依存です。 (例: ox#20)

- oy — Y 方向のオフセット (整数)
  解釈はフォントクラス依存です。 (例: oy#-10)

- fx — X 方向の拡大係数 (整数)
  解釈はフォントクラス依存です。 (例: fx#105)

- fy — Y 方向の拡大係数 (整数)
  解釈はフォントクラス依存です。(例: fy#97)

ファイル vfontcap がサンプルのフォントデータベースファイルですので、参考にして下さい。

7 フォントクラス

VFlib でサポートされているそれぞれのフォントクラスの説明と、実装されている属性、ならびに現在での実装上の理由による制約点を述べます。

- **Sony NWF-605 フォント (フォントクラス名 sony)**
  解釈できる属性は以下の通りです: ft, ff, tc, ro, rx, ry, sl, fr

  フォントファイルのパス名は、 例えば

  ```
  :ff=/usr/local/font/devnwp533/Mincho.vfont:
  ```

  のように、 完全なパス名を与えて下さい。

- **Zeit 書体倶楽部フォント (フォントクラス名 zeit)**
  解釈できる属性は以下の通りです: ft, ff, tc, ro, rx, ry, sl, fr, ox, oy, fx, fy

  ひとつの書体倶楽部フォントは mincho.vf1, mincho.vf2 のように、2 つのファイルで構成されています。フォントファイルの指定は

  ```
  :ff=/usr/local/font/syotai_club/mincho:
  ```

  のように、拡張子をはずした名を書いて下さい。(ff で与えたパス名に .vf1, .vf2 をつけたファイルを、フォントファイルとしてアクセスします。)

  のように、 パス名には拡張子 .vf1, .vf2 を付けないで下さい. (.vf1, .vf2 ともに、 同一のディレクトリにないといけません。)

  属性 ox, oy, fx, fy について説明します。ox は、 生のフォントデータの点の座標を ox で指定されただけ左に (即ち、 X 軸の負の方向へ) 移動させます。(書体倶楽部フォントでは、左上が原点とし、 点の範囲は X,Y 軸ともに 0 以上 1024 未満となっています。) oy は、 ox 同様に、生のフォントデータの点の座標を oy で指定されただけ上に (即ち、 Y 軸の負の方向へ) 移動させます。fx は、 fx で指定された値を 100 で割ったものと生のフォントデータでの点の X 座標を乗算した値を、 点の X 座標とします。fy は、 fy と同様に、 Y 軸に関して拡大します。

- **Zeit JG Font** (フォントクラス名 `jg`)
  解釈できる属性は以下の通りです: `ft`, `ff`, `tc`, `ro`, `rx`, `ry`, `sl`, `fr`, `ox`, `oy`, `fx`, `fy`

  ひとつのフォントは `zkyo0by.fn0`, `zkyo0by.fn1`, `zkyo0by.fn2` のように、3 つのファイルで構成されています。フォントファイルの指定では、

  ```
  :ff=/usr/local/font/jg_font/zkyo0by:
  ```

  のように、拡張子をはずしたパス名を書いて下さい。(`ff` で与えたパス名に `.fn0`, `.fn1`, `.fn2` をつけたファイルを、フォントファイルとしてアクセスします。)

  属性 `ox`, `oy`, `fx`, `fy` は、書体倶楽部フォントと同様です。但し JG Font では、 点の範囲は X,Y 軸ともに 0 以上 4096 未満となっています。

- **TrueType フォント** (フォントクラス名 `truetype`)
  解釈できる属性は以下の通りです: `ft`, `ff`, `tc`, `ro`, `rx`, `ry`, `sl`, `fr`, `ox`, `oy`, `fx`, `fy`

  属性 `ox`, `oy`, `fx`, `fy` は、書体倶楽部フォントと同様ですが、点の範囲は 0 以上 8192 未満となっています。

  本 TrueType フォント部は、松田一朗氏 (東京理科大学理工学部電気工学科) によって作られたものを、角川が一部変更を加えたものです。 松田さんからのコメントです:

  > ...
  > VFlib で漢字 TrueType フォントを利用する場合は，フォントファイル foo.ttf と同一のディレクトリに foo.tti という名前の補助インデックスファイルを用意しておく必要があります . foo.tti はフォントの JIS コード ⇒ 内部コードの対応表 (MS 標準の漢字 TrueType フォントは unicode ⇒ 内部コードの対応表しか用意されていない) が納められたファイルで，VFlib に付属する ttindex というツールを使って作成します .
  >
  > ```
  > % ttindex foo.ttf
  > ```
  >
  > また，vfontcap の `ff` 属性には，フォントファイルの絶対パス名から末尾の .ttf を取り除いた文字列を指定しておきます .
  >
  > なお，このプログラムは MS-Windows 3.1J に標準添付されている msmincho.ttf, msgothic.ttf での動作を確認しています (おそらく RICOH 製の漢字 TrueType font なら大丈夫でしょう) が，DynaFont TT シリーズ (Ami Pro のおまけフォントなど) は使用できません . `m(__)m`
  > ...

  Windows 3.1J に加えて、 VFlib-2.22 およびそれ以降では、Windows95 のフォントや一部の DynaFont TT シリーズも利用可能となっていま

す。(Ami Pro のおまけフォントの DynaFont TT シリーズは駄目なようです。) Windows95 のフォントファイルは、.ttc で終るファイル名を持っています。VFlib は、まず最初に .ttf で終るフォントファイルを読もうとします。もし見つからなければ、次に .ttc で終るフォントファイルを読もうとします。

Windorw95 フォントのために、ttindex も改良されました。Windorw95 フォントでの ttindex の実行は次のようにします。

```
% ttindex foo.ttc
```

ここで foo.ttc は、使用しようとするフォントの名前です。

現在のところ、 以下の TrueType フォントでの動作が確認されています。(これ以外のフォントで動作を確認された方は、角川までぜひ御報告下さい。)

- Windows 3.1J 附属の TrueType Font — MS 明朝体, MS ゴシック体.
- Windows 95 附属の TrueType Font — MS 明朝体, MS ゴシック体.
- Ricoh TrueType World HG シリーズ — HG 平成明朝体 (W3, W9), HG 平成角ゴシック体 (W3, W5, W7, W9), HG 創英角ポップ体, HG 創英丸ポップ体, HG 正楷書体, HG 丸ゴシック体 L, HG 丸ゴシック体 B, HG 教科書体, 行書体, HG 拡張明朝 L, HG 拡張ゴシック B.
- Fujitsu Fontcity TrueType コレクション — FC 明朝体-L, FC 明朝体-B, FC ゴシック体-L, FC ゴシック体-B.
- DynaFont リョービプレミアム 17 書体パック — DynaFont リョービプレミアム 17 書体パック (開発: リョービイマジニクス、販売: ダイナラブ · ジャパン)
- DynaFont プレミアム 30 書体パック
- 「具満タン」附属の DynaFont —『具満タン』(発売元: エーアンドピーコディネータージャパン株式会社) 付属の DynaFont (DF 極太ゴシック体, DF 平成明朝体 W3, DF 平成明朝体 W5, DF 平成明朝体 W7, DF 平成明朝体 W9, DF POP1 体).
- 一太郎 Ver 7 の TrueType フォント (ジャストシステム社) — JS 明朝 (jsmincho.ttc), JS ゴシック.
- 「白舟書体」— 篆書、楷書、行書、古印体の 4 書体が、フリーソフトウエアとして公開されています。http://www.net135.or.jp/~maruoka/

を御覧下さい。ひらがな、カタカナ、および漢字 1000 文字強を含んでいます。字数が少ないのが残念なのですが、字母は印章業を営んでおられる方による製作で、品質は大変良いのだそうです。

- OS/2 Warp 附属の TrueType フォント — 平成明朝 (W3), 平成角ゴシック (W5), リョービ本明朝-L, リョービ本明朝-B II, リョービゴシック-M II, リョービゴシック-B, リョービシリウス-L II, リョービシリウス-M, リョービ中楷書体.
- Canon FontGallery TrueType DeluxeCD '97 — 33 書体ある、キヤノンの Windows3.1 / Windows95 / Macintosh 用のフォント。Windows 95 用 .ttc ファイルでの動作を確認。
- NEC PC-9821 用 Windows 付属 Font Avenue フォントの fangot5.ttc (FA ゴシック) と fanmin3.ttc (FA 明朝)

- **TrueType フォント (フォントクラス名 `freetype`)**

  FreeType ライブラリを使った、もうひとつの TrueType ドライバです。truetype ドライバよりも、こちらのドライバを利用することをおすすめします。FreeType は、ヒント処理を行うので、小さなビットマップを作るときに綺麗な表示を得ることができます。さらに、使用可能な TrueType フォントはこちらの方が多いです。

  VFlib 2.24 では、 FreeType 1.1 対応になっています。(FreeType 1.0 では動作しませんので、注意して下さい。)

  仕様は truetype ドライバとほぼ同じですが、フォントファイルの指定 (`ff` 属性) は、拡張子 .ttf, .ttc も含めて完全パス名で与えてください。freetype ドライバでは、.tti ファイルは使いませんので、.tti ファイルを作る必要はありません。

  本ドライバ部は、松田一朗氏 (東京理科大学理工学部電気工学科) によって作られたものを、角川が一部変更を加えたものです。

  詳細は、`http://itohws03.ee.noda.sut.ac.jp/~matsuda/VFlib-FT/index.html` を参照してください。

  FreeType に関しては、`http://www.freetype.org/` を参照してください。

- **FontWave フォント (フォントクラス名 `fontwave`)**
  解釈できる属性は以下の通りです: `ft`, `ff`, `tc`, `ro`, `rx`, `ry`, `sl`, `fr`, `ox`, `oy`, `fx`, `fy`

  これらの属性の意味は、TrueType での意味と同じです。

- **BDF フォント (フォントクラス名 bdf)**
  BDF フォントは、X Window version 11 で使われるビットマップフォントの一形式です。BDF フォントはビットマップフォントですが、VFlib は内部でアウトラインに変換します。現在では、X Window version 11 Release 5 での、k14.bdf, jiskan16.bdf, jiskan24.bdf での動作を確認しています。

  BDF フォントを使うときは、X Window は必要なく、フォントファイルさえあれば十分です。BDF フォントクラスでの文字コード体系は、元の BDF フォントでの文字コード体系と同じです。(k14, jiskan16, jiskan24 ともに JIS コードを使っていますので、他のフォントクラスと入れ換えても問題なく使えます。)

  このフォントクラスが解釈する属性は以下の通りです: `ft`, `ff`, `tc`, `ro`, `rx`, `ry`, `sl`, `ox`, `oy`, `fx`, `fy`, `ds`, `dz`

  属性 `ox`, `oy`, `fx`, `fy` は、TrueType のときと同じです。

  `ff` は、BDF フォントのパス名を与えますが、拡張子 `bdf` はつけないで下さい。
  (例: `:ff=/usr/local/font/bdf/k14:`)
  ファイルアクセスを高速化するために、BDF フォントドライバはインデックスファイルを使います。インデックスファイルは、附属の bdfindex を使って生成します。使い方は、BDF フォントのファイル名から拡張子を除いたものを、引数に与えます。例えば、`jiskan16.bdf` のインデックスファイルを作るときは、次のようにします:

  ```
  % bdfindex jiskan16
  ```

  このコマンドにより、`jiskan16.bdx` という名前のファイルが作られます。これがインデックスファイルです。なお、BDF ファイルとそのインデックスファイルは、同一のディレクトリにないといけません。例えば `:ff=/usr/local/font/bdf/k14:` と指定するときは、ディレクトリ `/usr/local/font/bdf/` に `k14.bdf` と `k14.bdx` がないといけません。

  元データはビットマップで、ドットの集まりです。VFlibd はドットの集まりをそそのまま拡大します。`ds` では、描くドットの形を指定します。この値が `square` なら、それぞれのドットを四角で描きます。もし `ds` の値が `diamond` なら、それぞれのドットをダイアモンドの形で描きます。`ds` のデフォルト値は `square` となっています。(例: `:ds=diamond:`)

  `dz` は、ドットの大きさの係数を整数で指定します。この値が 100 なら、ドット間はびっちりと詰まったものになります。この値がちいさくするほどドットの大きさが小さくなり、ドットの間隔が大きくなります。`dz` のデフォルト値は 100 となっています。(例: `:dz#80:`)

- **HBF フォント (フォントクラス名 hbf)**
  HBF (Hanji Bitmap Font) 形式のフォントも利用可能です。HBF フォントは BDF フォントと同様にビットマップフォントですが、VFlib は内部でアウトラインに変換します。(Ross Paterson 氏による HBF API ライブラリを利用しています。) 指定可能な属性は BDF フォントクラスと同一です。BDF フォントと異なり、HBF フォントの場合はインデックスファイルは不要です。

- 複合フォント (フォントクラス名 compound)
  漢字、 仮名、 記号各々に用いるフォントを指定できるフォントクラスです。例えば、 漢字はゴチック体を使い、 平仮名に明朝体を用いるといった、漫画の吹き出しのようなフォントが 2 つの基本的なフォントを元に合成できます。

  属性は以下の通りです。

  - kj — 漢字用フォントエントリ名。 (文字列) (例: kj=min)
  - kn — 仮名用フォントエントリ名。 (文字列) (例: kj=goth)
  - sy — 記号用フォントエントリ名。 (文字列) (例: kj=goth)

- 複合フォント (フォントクラス名 compound2)
  文字コードの区間を指定し、区間ごとに使用するフォントを指定できるフォントクラスです。16 個までの区間が使用でき、A から P までの名前が付けられています。それぞれの区間に対し、文字コード範囲のリストと、使用するフォントを指定します。属性は以下の通りです。

  - rA, rB, ..., rP — それぞれの区間の文字コードの範囲のリストを与える。文字コードは 16 進数で与える。(例: rA=2421-2473, 2521-2576)
  - fA, fB, ..., fP — それぞれの区間の文字に対するフォント名を与える。(例: fA=tmin-normal-rt)

  サンプルの vfoncatp ファイルが vfontcap.comp2 なので、こちらも参照して下さい。

- ビット演算フォント (フォントクラス名 bitop)
  他のフォントエントリで得られるビットマップに対して、ビット演算を施して新たなビットマップを得るためのクラスです。解釈できる属性は、 op, fo, sf の 3 つです。それぞれ、 ビット演算命令列、 フォントエントリ名、副フォントエントリ名を表します。ビット演算命令列属性 op は必須属性です。

  演算機能の説明をします。3 つのビットマップレジスタ、 SRC, DEST, REG があり、これらを使って各種演算を行ないます。fo 属性で指定

されたフォントエントリの VF_GetBitmap() 等での呼出しで指定された文字コードに対するビットマップが、SRC の初期値として入れられます。書き込み対象となっているフレームバッファの位置にあるビットマップが DEST の初期値として入れられます。REG は零クリアされます。

これらのレジスタを使っての演算命令の列を、文字列の形で属性 op に与えます。使用可能なビットマップ演算は、下の表の通りです。演算は、命令列の左から右に向かって順々に命令が実行され、並びの終りが来ればそこで演算は終了します。そしてその時の DEST レジスタの値が、実行結果としてフレームバッファに書き込まれます。

命令についてより詳しく説明します。各命令は 1 文字で表されます。命令には、演算対象や演算結果を格納するレジスタなどの引数を持つものもあります。以下の表で、 r_i または r はビットマップレジスタを表し、S, D, あるいは R のいずれか 1 文字でないといけません (小文字であっても構いません)。S, D, R はそれぞれ、 SRC, DEST, REG レジスタを表します。

Nr	r を白黒反転させます。
Cr	r を零クリアします。
Er_1r_2	r_1 と r_2 の内容を交換します。
Lr_1r_2	r_1 に r_2 の内容を代入します。
A$r_1r_2r_3$	r_2 と r_3 の内容のビットごとの論理積をとり、 その結果を r_1 に代入します。
O$r_1r_2r_3$	r_2 と r_3 の内容のビットごとの論理和をとり、 その結果を r_1 に代入します。
X$r_1r_2r_3$	r_2 と r_3 の内容のビットごとの排他的論理和をとり、 その結果を r_1 に代入します。
B$rn_1n_2n_3n_4$	属性 sf で指定されるフォントエントリの文字コード $n_1n_2n_3n_4$ (4 桁の 16 進数) のビットマップを r に入れます。

なお、スペース文字、改行、タブの各文字は、何も行なわない命令です。よって、命令列を読みやすくするために、これらの文字を使うこともできます。

例えば、

```
minstar:\
        :ft=bitop:\
        :op=LDS BS217a XDDS:fo=min:sf=goth:
```

では、 次のようなビットマップ演算を行ないます: まず、 SRC には min フォントでの指定された文字コードのビットマップが、DEST にはこれから書き込もうとしているフレームバッファに既に書き込まれている内容が、各々初期値として入ります。一方、REG は零クリアされます。最初の命令は LDS ですので、 SRC の内容を DEST に書き込みます。その次に、命令 BS217a によって、 副フォント (この場合だと goth) の文字コード 0x217a (記号 '★' です) を SRC に書き込みます。そして最後に XDDS により、 DEST と SRC の排他的論理和を求めて、その結果を DEST に書き込んでいます。これで命令列の実行が終了し、DEST の内容がフレームバッファに書き込まれます。

なお、 このクラスに対する VF_GetOutline は、 fo で指定されたフォントエントリへの VF_GetOutline と同じです。

- ヌルフォント (フォントクラス名 null)
  このフォントクラスは、文字を描かないクラスです。属性として ty を持ち、この値によって以下の動作をします。
  - white — 何も描きません。
  - black — 真っ黒な箱を描きます。
  - frame — 四角い縁を描きます。

  このフォントクラスは何もしないという性質から、VFlib の拡張をしている時のデバッグや、VFlib を使ったアプリケーションプログラムのデバッグに使えます。

8 新たなフォントクラスの追加方法

ソースコード VF_Comp.c や VF_Zeit.c などを読んで下さい。

9 バグ報告ならびに最新版の入手法

バグ報告/改良/提案/要望がありましたら、 御報告願います。角川裕次まで e-mail で御連絡下さい。 寄付も大歓迎です。(^o^)

最新版の VFlib は、

ftp://ftp.se.hiroshima-u.ac.jp/pub/VFlib/

に置いていますので、 匿名 ftp で入手して下さい。IP 接続になっていない人は、ftp できる環境にある人にお願いするとか、パソ通などに誰かがアップロードするのを期待して待って下さい。

最近はフリーソフトウエアを CD-ROM に収録して、本の形で販売しているものがいくつもありますので、それらを利用するのもいいでしょう。私の手元には、

> "CD-ROM & BOOK フリーソフト&シェアウエア PACK 10000," 1996 年前期版, 編集 · 発行 ベクターデザイン, 発売 (株) アスキー, ISBN 4-7561-0592-0.

というのがあり、これに VFlib が収録されています。(そのほかのものにも収録されている可能性はあります。) ftp サーバーでのリリースから本の出版までに、数カ月の間隔があります。そのために、 最新版の VFlib である保証はありませんが、ftp できない人はこのような形での入手方法もあります。

10 行なうべきこと

- フォントアセンブラ — disol で作られるアウトラインデータの逆アセンブル結果のフォーマットを解釈し、各種フォントフォーマットのアウトラインフォントファイルを作成するアセンブラが欲しい。
- フォントエディタ — ウインドウ環境 (X Window, MS Windows, etc.) で、アウトラインフォントを作成できる CAD ツールが欲しい。
- 影つき文字のサポート。
- 濃淡の指定できるクラスの作成。 (BitOP クラスの改造で十分。)
- 「ロゴ的フォント」クラス — 例えば、 IBM 社のロゴみたいに、フォントからあるビットマップパターンを引いて作るフォントクラス。

References

[1] "NEWS-OS Release 4.0 ユーザーズガイド Vol.1" 第 7 章 フォント、第 17 章 アウトラインフォント操作ライブラリ, ソニー株式会社, 1990.

謝辞

VFlib 第 2 版を開発するにあたり、 多くの方々にお世話になりました。書体倶楽部フォントモジュールは、 dviprt 並びに Ghostscript のソースコードを参考にさせて頂きました。JG Font モジュールは、 森下英男氏による VFlib 第 1 版用の JG Font モジュールを基にして作成しました。 また、 VFlib 第 1 版でのバグ報告を行なって下さった方々(鈴木治氏、 峯松信明氏、 杉浦茂樹

氏) にも感謝します。松田一朗氏には TrueType, FreeType フォントモジュールを提供してくださいました。寺川愛印氏は、FontWave モジュールを作成し提供してくださいました。Werner Lemberg 氏には HBF フォントドライバを提供してくださいました。青笹茂氏は、Compund2 フォントクラスのドライバを提供してくださいました。御意見をおよせいただいた内山たかのり氏、大上茂雄氏、風間一洋氏に感謝します。淺山和典氏からは色々アドバイスを頂きました。以下の方々からは、バグ報告やいろいろな情報をお寄せ頂きました：清水正氏、 岡川宏氏、 越野ようすけ氏、 SOLITON 氏、林康久氏、 伊東直史氏、 桜井貴文氏、佐々木毅史氏、村瀬高史氏、上野洋一郎氏、福田くにゆき氏、梅野たかし氏、知念氏、酒居敬一氏、田原鉄也氏、内川喜章氏、清水淳氏、横山淳氏、柳谷雅之氏、植原氏、石井義規氏、小林和男氏、野村俊彰氏、平賀十志男氏、荒井良重氏、沢田石順氏、田岡智志氏、梶山たみと氏, Matt Gushee 氏

多くの方々の御協力に感謝いたします。