オープンプリンティング
レンダリング・サブルーチン・ライブラリ
マニュアル

1.0

2005 年 8 月 2 日

目 次

第1章　概要

1.1　はじめに

オープンプリンティング・レンダリング・サブルーチン・ライブラリは、Free Standards Group の OpenPrinting Working Group で提案されている Vector Printer Driver API を使い易くするためのライブラリです。ただし、PDF 的な描画をデバイス座標を使用して行うモデルを採用しており、Vector Printer Driver API の仕様をすべて、使えるわけではありません。

以下の特徴を持っています。

1. ベクター型のプリンタとラスター型のプリンタの両方の相違を、殆ど意識せずに利用できます。
2. ドライバによるサポート機能の相違を吸収しています。

   以下の相違を吸収しています。

   (a) クリッピングパスのあるなし
   (b) グラフィックス状態の保存で、クリッピングパスを保存するか否か
   (c) ラインダッシュをサポートするか否か
   (d) イメージの回転、変形をサポートするか否か
3. フォントの扱い、文字の描画を追加しました。
4. イメージ描画に明示マスク機能を追加しました。
5. イメージマスク描画を追加しました。
6. スライス出力機能を付けました。
7. ベクタプリンタドライバのロード機能を付けました。

このマニュアルは、OpenPrinting Vector Printer Driver API の知識を前提にしています。先に、仕様書を御読みください。

ライブラリは、多くのコードを xpdf から流用しています。また、FreeType 2 ライブラリ、t1lib ライブラリを使用しています。必要に応じて、これらのドキュメント等を参照してください。

1.2 座標系

ライブラリでは、デバイス座標系、イメージ座標系、グリフ座標系、フォント座標系の 4 種類の座標系を使用します。描画は、デバイス座標系で行いイメージおよび文字以外では、座標変換は、行いません。

1.2.1 デバイス座標系

デバイス座標系は、描画の対象となる座標系です。単位は、ピクセルです。2 次元の直交座標系で描画対象の左上隅を原点とし、x 座標の値は原点から右へ向って大きくなり、y 座標の値は、下に向って大きくなります。(図 1.1)

stroke や fill などの描画メソッドに渡すパスは、すべてこの座標系で記述されます。

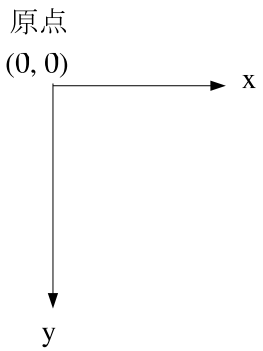


図 1.1: デバイス座標系

1.2.2 イメージ座標系

イメージ座標系は、イメージやイメージマスクを定義する座標系です。2 次元の直交座標系でイメージの左下隅を原点とし、x 座標の値は原点から右へ向って大きくなり、y 座標の値は、上に向って大きくなります。イメージやイメージマスクは、(0, 0) と (1, 1) を境界とする単位矩形に対応します。

イメージ描画メソッド drawImage や fillImageMask に指定する mat 引数は、イメージ座標系からデバイス座標系への変換行列です。この変換行列に従って、イメージやイメージマスクが描画されます。(図 1.2)

PDF のイメージソース座標系とは、異なるので注意してください。

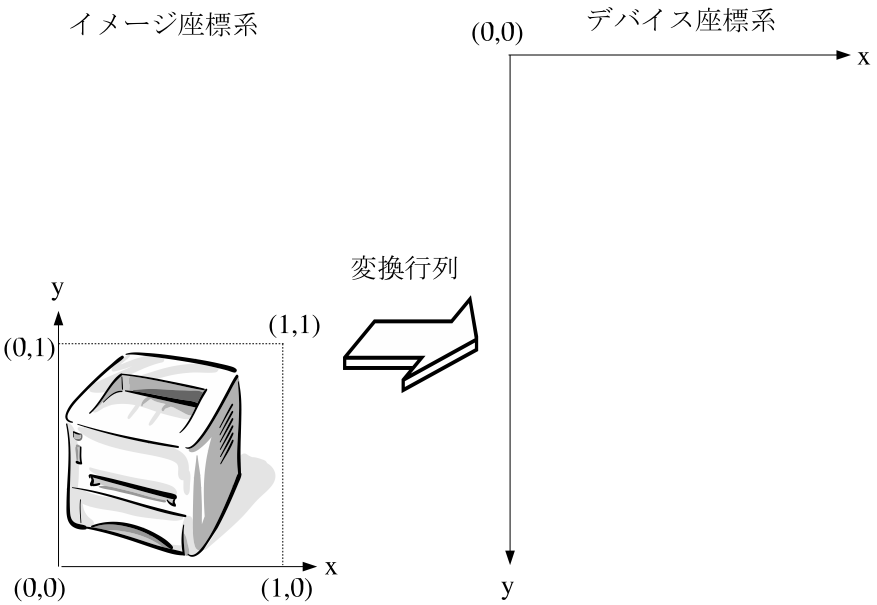


図 1.2: イメージ座標系とデバイス座標系

1.2.3 グリフ座標系

グリフ座標系は、文字のグリフが定義されている座標系です。2 次元の直交座標系で、x 座標の値は原点から右へ向って大きくなり、y 座標の値は、下に向って大きくなります。グリフの行の名目の高さが単位です。つまり、1 ポイントのグリフでは、ポイントが単位となります。(図 1.3)

y 座標の向きが通常とは逆ですが、デバイス座標系の向きと合せているためです。実際に、この座標系でフォントが設計されているわけではなく、API 上このように扱うだけです。

PDF のグリフ座標系とは、異なるので注意してください。

1.2.4 フォント座標系

フォント座標系は、デバイス座標系におけるグリフの形状を定義する座標系です。2 次元の直交座標系で、x 座標の値は原点から右へ向って大きくなり、y 座標の値は、下に向って大きくなります。単位は、デバイス座標系と同じく、デバイスのピクセルです。

フォント行列は、グリフ座標系からフォント座標系への変換を行う行列です。フォント行列の指定によって、文字の拡大、縮小、変形、回転を行うことが

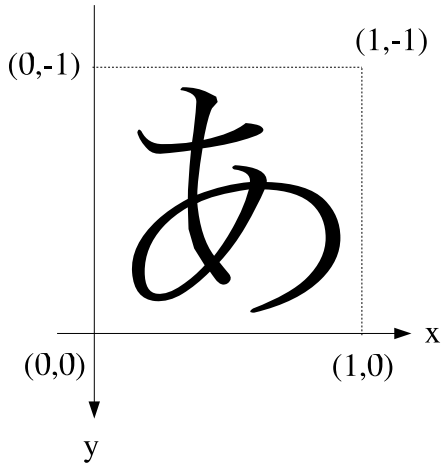


図 1.3: グリフ座標系

できます。

描画する場合は、平行移動のみを行います。そのため、文字描画メソッド fillChar、fillGlyph では、原点のオフセット (x,y) を指定します。fillChar の fontMat 引数は、描画時の座標の変換には、関与しません。(図 1.4)

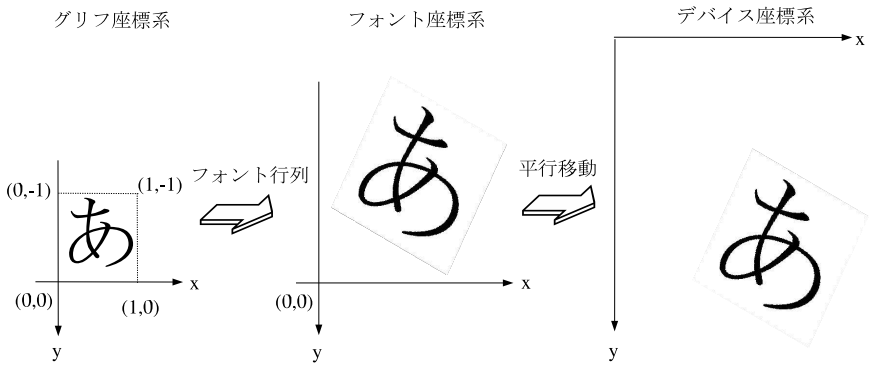


図 1.4: フォント座標系

1.3 描画モード

本ライブラリは、描画コマンドを受け付けるベクタープリンタと、ラスター単位のデータでしか受け付けないラスタープリンタの両方に対応しています。

ベクタープリンタ・ドライバでは、ラスタープリンタの場合は、StartRaster,

SkipRaster, EndRaster などのラスター系の API のみサポートされています。

ベクタープリンタと、ラスタープリンタをプログラマが意識せずに使用できるのが理想です。しかし、現状では、困難であり、本ライブラリでは、初期化やページ出力の際に意識する必要があります。

本ライブラリでは、ベクタープリンタ用のベクターモード、ラスタープリンタ用のラスターモードの二つの描画モードがあります。

描画モードは、基本的には、OPRS::init で指定します。しかし、プリンタ・ドライバが、NewPath をサポートしていない場合は、ベクターモードを指定しても、ラスターモードになります。

ラスターモードでは、描画対象となるピクセル・マップを指定する必要があります。また、ピクセル・マップを指定すると強制的にラスター・モードになります。解像度が高い場合には、ピクセル・マップは、非常に大きくなりメモリ消費が激しくなります。そのため、ラスターモードでは、一度にページ全体を描画させずに、部分に分けて順にプリンタに出力する方式が有効です。

このとき、一度に描画する部分をスライス、それを出力することをスライス出力と呼ぶことにします。本ライブラリでは、スライス出力をサポートしています。

1.4 グラフィックス状態

ライブラリでは、描画に関する属性をグラフィックス状態として持っています。グラフィックス状態は、内部のスタックを使用してに保存、回復ができます。

グラフィックス状態には、以下のものがあります。

1.4.1 クリッピングパス

描画が行われる領域をパスの形で指定します。指定したパスの内部が、描画が行われる領域となります。このパスをクリッピング・パスと呼びます。

OPRS::clipPath で指定し、以前からのクリッピング・パスの示す領域と重なる領域があたらしい描画領域になり、描画領域が広がることは、ありません。描画領域を広げるには、OPRS::restoreState により前の状態にもどすか、OPRS::endPage か、あるいは、OPRS::initGS で初期化するしかありません。

1.4.2 線幅

線の太さを定義します。単位は、ピクセルです。0 の場合は、描画可能なもっとも細い線を表します。

1.4.3 ラインキャップ種別

ラインキャップ種別は、幅のある線を描画するさいの線の端の形状を指定します。

バットキャップ、ラウンドキャップ、スクエアキャップの 3 種類あります。

- バットキャップ (指定値 0)

  端を角型に打ち切ります。端からはみ出す部分は、ありません。

- ラウンドキャップ (指定値 1)

  線幅と同じ直径の半円を端に書きます。

- スクエアキャップ (指定値 2)

  端を越えて、線幅の半分の距離まで書き角型に打ち切ります。

1.4.4 ラインジョイン種別

ラインジョイン種別は、幅があり、接続するふたつの線分の角の部分の形状を指定します。パス上で連続する線分の角のみで意味があります。

マイタージョイン、ラウンドジョイン、ベベルジョインの 3 種類あります。

- マイタージョイン (指定値 0)

  二つの線分の外縁部を交差するまで延長します。延長部分が長すぎる場合には、ベベルジョインが使用されます。長すぎるかどうかの判断には、マイターリミットが参照されます。

- ラウンドジョイン (指定値 1)

  線幅と直径の円弧を線分の交点を中心に描き、角をまるめます。

- ベベルジョイン (指定値 2)

  二つの線分の端を角型に打ち切り、切り欠き部分も描画します。

1.4.5 マイターリミット

マイタージョインの延長部分が長すぎるかどうかを判定するためのパラメータです。線幅に対するマイター長の比率で定義します。この比率よりも大きい場合には、ベベルジョインを用います。

マイター長とは、マイタージョインにおいて、延長した外縁部の二つの交点間の距離です。

1.4.6 ラインダッシュ

ラインダッシュは、破線を指定します。破線のパターンを定義するラインダッシュ配列とラインダッシュ・フェーズで定義されます。

ラインダッシュ配列の各要素は、順に線分部分と隙間部分の長さをしていします。最後の要素まで使うと最初の要素に戻ります。ラインダッシュ・フェーズは、配列のパターン中のどの部分から開始するかを決定します。フェーズは、パターン中の最初から長さを表し、パターンは、その長さ以降から開始します。

ラインダッシュ配列が空の場合は、実線を表します。

1.4.7 塗り潰しパターン

塗り潰しパターンは、塗り潰しの際に使用するパターンを指定します。ただし、現在は、単一色のみがサポートされています。

1.4.8 ストロークパターン

ストロークパターンは、線の描画の際に使用するパターンを指定します。ただし、現在は、単一色のみがサポートされています。

1.5 イメージマスク

イメージマスクは、2 値のビットマップであり、描画をすべき部分としない部分を指定します。描画すべき部分は、塗り潰しパターンを使用して描画されます。

ただし、本ライブラリでは、データの形式に依存しない形で指定し、イメージマスク・ソース関数の返す値で描画すべきかどうかを判断します。

1.6 明示的イメージマスク

明示的イメージマスクは、イメージのピクセル・マップと同時に指定されて、イメージを描画すべきかどうかを指定します。

ただし、本ライブラリでは、データの形式に依存しない形で指定し、イメージ・ソース関数の返すアルファ値で描画すべきかどうかを判断します。

イメージ・ソース関数では、イメージマスクの内容をみてアルファ値を計算してください。

1.7 塗り潰し規則

塗り潰しを行ったり、クリッピングを行う場合には、パスの内部を決定する必要があります。内部を決定する規則には、非ゼロ回転数規則と奇偶規則の二つあります。

- 非ゼロ回転数規則

  ある点から、無限遠まで直線を引き、パスの線との交差場所をみます。パスの線が左から右へ交差すると部分 1、右から左へ交差する部分-1 とします。すべての交差場所の値を加えて 0 となれば、その点は、外部に存在し、0 以外であれば内部に存在すると定義します。

- 奇偶規則

  ある点から、無限遠まで直線を引き、パスの線との交差場所の数をみます。数が、奇数ならば、その点は、内部に存在し、偶数ならば、その点は、外部に存在すると定義します。

1.8 平滑度

平滑度は、曲線がのレンダリングの精度を示します。小さいほど、より滑らかになります。ただし、現在、サポートしていません。OPRS::setFlatness は、なの効果もなく、OPRS::getFlatness は、常に 1 を返します。

1.9 カラーモード

カラーの形式を表わすモードです。現在は、描画の対象としては、splashModeRGB8 のみがサポートされています。

描画対象のカラーモードは、OPRS::setColorMode で設定します。OPRS::initGS で指定するカラーモードは、これと一致している必要があります。OPRS::setBitmap で指定するピクセルマップのカラーモードもこれと一致していなければいけません。また、OPRS::initGS の paperColro、イメージ・ソース関数が返すピクセル値の表現もこのカラーモードにあったものである必要があります。

1.10 プログラムの流れの概略

本ライブラリを利用したプログラムの流れの概略は、以下のようになります。エラー処理は、省略していますので、必要に応じて入れてください。

1. 初期化

```
OPRS *oprs = new OPRS();
GBool rasterMode;
SplashBitmap *bitmap;

oprs->init(...);
oprs->setColorMode(...);
rasterMode = oprs->getRasterMode();

if (rasterMode) {
   /* ラスターモードの場合は、
    ピクセルマップを作成して設定します。*/
   /* ピクセルマップの大きさは、
     スライスの大きさにします。*/
   bitmap = new SplashBitmap(...);
   oprs->setBitmap(bitmap);
}
```

2. ジョブの開始

```
oprs->OPVPStartJob(...);
```

3. ドキュメントの開始

```
oprs->OPVPStartDoc(...);
```

4. ドキュメントの開始

```
oprs->OPVPStartDoc(...);
```

5. ページ出力

以下をページの数だけ繰り返します。

(a) ページの開始

```
oprs->OPVPStartPage(...);
```

(b) スライスの描画

ラスターモードの場合は、以下をスライスの数だけ繰り返します。
ベクターモードの場合は、ページ全体をひとつのスライスとして、一度だけ繰実行します。

i. スライスの開始

```
/* w,h には、スライスの大きさを渡します。*/
oprs->initGS(...);
```

ii. スライスの描画

描画に関するメソッドの呼び出し

iii. スライスの終了と出力

```
oprs->endPage();
oprs->outSlice();
```

(c) ページの終了

```
oprs->OPVPEndPage();
```

6. ドキュメントの終了

```
oprs->OPVPEndDoc();
```

7. ジョブの終了

```
oprs->OPVPEndJob();
```

8. 後始末

```
if (rasterMode) {
  /* ピクセル・マップは、
     自分で削除しなければいけません。*/
  delete bitmap;
}
delete oprs;
```

1.10.1 フォントの操作の概略

フォントの操作は、以下のような手順になります。エラー処理は、省略していますので、必要に応じて入れください。

1. フォント・エンジンの作成

   フォント・エンジンのインスタンスを作成します。

   ```
   SplashFontEngine *fontEngine = new SplashFontEngine(...);
   ```

2. フォントファイルのロード

   フォント・エンジンを利用してフォントのファイルの種類に応じてフォントファイルをロードします。

   まず、キャッシュのためにユニークなフォントファイル ID を割当てます。既にロードしているフォントファイルは、この ID を指定して SplashFontEngine::getFontFile を呼ぶことで、取得できます。

   (a) Type1 フォント

   ```
   ...
   フォント・ファイル名の決定や
   エンコーディングの設定などの処理
   ...
   SplashFontFile *fontFile = fontEngine->loadType1Font(...)
   ```

   (b) Type1C(Compact Font Format) フォント

   ```
   ...
   フォント・ファイル名の決定や
   エンコーディングの設定などの処理
   ...
   SplashFontFile *fontFile = fontEngine->loadType1CFont(...)
   ```

   (c) TrueType(あるいは OpenType) フォント

```
...
フォント・ファイル名の決定や
codeTiGIDマップの設定などの処理
...
SplashFontFile *fontFile = fontEngine->loadTrueTypeFont(...)
```

(d) CID フォント

```
...
フォント・ファイル名の決定などの処理
...
SplashFontFile *fontFile = fontEngine->loadCIDFont(...)
```

3. フォントの取得

   フォントエンジンを使って、フォント・ファイルからフォントを取得します。

```
...
必要なフォントの大きさ、
形状に合せてフォント行列を作成
...
SplashFont *font = fontEngine->getFont(fontFile, mat);
```

4. 描画します。

```
oprs->fillChar(...);
```

第2章　階層索引

2.1　オープンプリンティング・レンダリング・サブルーチン クラス階層

この継承一覧はおおまかにはソートされていますが、完全にアルファベット順でソートされてはいません。

第3章　構成索引

3.1　オープンプリンティング・レンダリング・サブルーチン 構成

クラス、構造体、共用体、インタフェースの解説です。

第4章　クラスの解説

4.1　クラス OPRS の解説

`#include <OPRS.h>`

このクラスが、オープンプリント・レンダリング・サブルーチンの API の主要部分です。

Public メソッド

- **OPRS** ()
- **∼OPRS** ()
- int **setBitmap** (SplashBitmap ∗bitmapA)
- SplashPattern ∗ **getStrokePattern** ()
- SplashPattern ∗ **getFillPattern** ()
- **SplashCoord getLineWidth** ()
- int **getLineCap** ()
- int **getLineJoin** ()
- **SplashCoord getMiterLimit** ()
- **SplashCoord getFlatness** ()
- **SplashCoord** ∗ **getLineDash** ()
- int **getLineDashLength** ()
- **SplashCoord getLineDashPhase** ()
- void **setStrokePattern** (SplashPattern ∗strokeColor)
- void **setFillPattern** (SplashPattern ∗fillColor)
- void **setLineWidth** (**SplashCoord** lineWidth)
- void **setMiterLimit** (**SplashCoord** miterLimit)
- void **setLineCap** (int lineCap)
- void **setLineJoin** (int lineJoin)
- void **setFlatness** (**SplashCoord** flatness)

- void **setLineDash** (**SplashCoord** ∗lineDash, int lineDashLength, **SplashCoord** lineDashPhase)
- **SplashError clipToPath** (SplashPath ∗path, GBool eo)
- void **saveState** ()
- **SplashError restoreState** ()
- void **clear** (SplashColor color)
- **SplashError stroke** (SplashPath ∗path)
- **SplashError fill** (SplashPath ∗path, GBool eo)
- **SplashError fillChar** (**SplashCoord** x, **SplashCoord** y, int c, SplashFont ∗font, Unicode ∗u, double ∗fontMat)
- **SplashError fillGlyph** (**SplashCoord** x, **SplashCoord** y, SplashGlyphBitmap ∗glyph)
- **SplashError fillImageMask** (**SplashImageMaskSource** src, void ∗srcData, int w, int h, **SplashCoord** ∗mat)
- **SplashError drawImage** (**SplashImageSource** src, void ∗srcData, **SplashColorMode** srcMode, int w, int h, **SplashCoord** ∗mat, GBool mask)
- SplashBitmap ∗ **getBitmap** ()
- void **setDebugMode** (GBool debugModeA)
- int **init** (const char ∗driverName, int outputFD, char ∗printerModel, int rasterMode, int nOptions, char ∗optionKeys[ ], char ∗optionVals[ ])
- void **initGS** (int colorMode, int w, int h, SplashColor paperColor)
- int **setColorMode** (int colorModeA)
- int **unloadVectorDriver** ()
- int **OPVPStartJob** (char ∗jobInfo)
- int **OPVPEndJob** ()
- int **OPVPStartDoc** (char ∗docInfo)
- int **OPVPEndDoc** ()
- int **OPVPStartPage** (char ∗pageInfo, int rasterWidth)
- int **OPVPEndPage** ()
- int **outSlice** ()
- int **getRasterMode** ()
- void **endPage** ()

Static Public メソッド

- void **error** (char ∗msg,...)

4.1.1 コンストラクタとデストラクタの解説

4.1.1.1 OPRS::OPRS ()

コンストラクタでは、プリンタドライバを読み込みません。コンストラクタの後に init メソッドを呼ぶ必要があります。

4.1.1.2 OPRS::~OPRS ()

デストラクタでは、もしプリンタドライバを読み込んでいれば、ClosePrinter を呼出し後、解放します。

4.1.2 メソッドの解説

4.1.2.1 void OPRS::clear (SplashColor *color*)

引数:

color クリアするカラー

color で示されるカラーでページ全面をクリアします。ただし、ベクター・モードでは、何もしません。

4.1.2.2 SplashError OPRS::clipToPath (SplashPath * *path*, GBool *eo*)

引数:

path クリッピング・パス

eo 塗り潰し規則フラグ

戻り値:

エラー値

クリッピング領域を path で規定されるものに設定します。

eo は、クリッピング領域を決める際に、どの塗り潰し規則を使用するかを指定します。gTrue を指定すると、奇遇規則、gFalse ならば非ゼロ回転数規則を用います。塗り潰した領域が、ペイント可能な領域となります。

すでに、クリッピング領域が設定されている場合は、重なっている領域がペイント可能な領域となります。

成功時には、splashOk を返します。その他の場合は、エラーコードを返します。

4.1.2.3 SplashError OPRS::drawImage (SplashImageSource *src*, void * *srcData*, SplashColorMode *srcMode*, int *w*, int *h*, SplashCoord * *mat*, GBool *mask*)

引数:

src イメージ・ソース関数へのポインタ

srcData イメージ・ソース・データ

srcMode ソース・イメージのカラーモード

w ソースイメージの幅 (ピクセル)

h ソースイメージの高さ (ピクセル)

mat イメージ座標からデバイス座標への変換行列

mask 明示的イメージマスクが指定されているか否か

戻り値:

エラー値

イメージを描画します。ソース・イメージは、以下の形式で呼出すことにより、左上のピクセルから行優先の順に得できると仮定しています。

```
Guchar alpha;
SplashColor color;

(*src)(srcData, &color,&alpha);
```

ピクセルのカラーが color にアルファ値が alpha に取得されます。

mask は、描画の手段を選択するのに参照されます。アルファ値が有効なのは、mask が gTrue の場合のみです。その場合でも、0 か 0 でないかのみで判断され、0 で無いときは、ピクセル値でそのまま、描画されます。通常のアルファ値とは、異なる扱いを受けるので注意して下さい。

描画先は、mat で示される行列で示されます。mat が正則でない場合は、エラーとなります。

成功時には、splashOk を返します。その他の場合は、エラーコードを返します。

4.1.2.4 void OPRS::endPage ()

1 ページの処理の終了処理を行います。内部データの解放などを行います。

4.1.2.5 void OPRS::error (char * *msg*, ...) [static]

引数:

msg メッセージ

エラー・メッセージを標準エラー出力に出力します。OPRS クラスのメソッド内でのエラー出力は、このメソッドで行われます。内部用関数なので、呼出さないでください。

4.1.2.6 SplashError OPRS::fill (SplashPath * *path*, GBool *eo*)

引数:

path パス

eo 塗り潰し規則フラグ

戻り値:

エラー値

path で指定される領域を塗り潰します。

eo は、どの塗り潰し規則を使用するかを指定します。gTrue を指定すると、奇遇規則、gFalse ならば非ゼロ回転数規則を用います。

成功時には、splashOk を返します。その他の場合は、エラーコードを返します。

4.1.2.7 SplashError OPRS::fillChar (SplashCoord *x*, SplashCoord *y*, int *c*, SplashFont * *font*, Unicode * *u*, double * *fontMat*)

引数:

x 文字の描画先の X 座標

y 文字の描画先の Y 座標

c 文字の CID

font フォント

u 文字のユニコードへのポインタ

fontMat フォント行列

戻り値:
エラー値

デバイス座標の (x,y) に、c または u で示される文字を描画します。文字は、グリフを塗り潰す形で描画されます。塗り潰す場合には、塗り潰しカラーを使用しますが、パターンで塗り潰すことはできません。デバイス座標 (x,y) は、文字のベース上にあり、グリフの原点を示します。

font は、文字のフォントを示します。font が、TrueType 系の場合は、u を使用してグリフを取得します。それ以外の場合は、c を使用してグリフを取得します。

fontMat は、フォント行列です。ただし、この行列は、グリフのサイズを推定して、描画方法を決定するためにだけ、使用されます。フォント自身は、あらかじめ、この行列を使用して取得しておく必要があります。

成功時には、splashOk を返します。その他の場合は、エラーコードを返します。

4.1.2.8 SplashError OPRS::fillGlyph (SplashCoord *x*, SplashCoord *y*, SplashGlyphBitmap * *glyph*)

引数:

x 文字グリフの描画先の X 座標

y 文字グリフの描画先の Y 座標

glyph 文字グリフのビットマップ

戻り値:
エラー値

glyph で示される文字グリフのビットマップを、デバイス座標 (x,y) に描画します。デバイス座標 (x,y) は、文字のベース上にあり、グリフの原点を示します。

成功時には、splashOk を返します。その他の場合は、エラーコードを返します。

4.1.2.9 SplashError OPRS::fillImageMask (SplashImageMaskSource *src*, void * *srcData*, int *w*, int *h*, SplashCoord * *mat*)

引数:

imageMaskSrc イメージマスク・ソース関数へのポインタ

imgMaskData イメージマスク・ソース・データ

width イメージマスクの幅

height イメージマスクの高さ

mat イメージ座標からデバイス座標への変換行列

戻り値:

エラー値

イメージマスクを描画します。イメージマスクは、描画の場所を示すマスクであり、ビットマップです。ビットの値が 1 の部分に塗り潰しカラーで描画します。

ソース・イメージマスクは、以下の形式で呼出すことにより、左上のピクセルから行優先の順に得できると仮定しています。

```
Guchar alpha;
SplashMono1 color;

(*src)(srcData, &color);
```

ピクセルのカラーが color に取得されます。color は、1 または 0 の値を持ちます。

描画先は、mat で示される行列で示されます。mat が正則でない場合は、エラーとなります。

成功時には、splashOk を返します。その他の場合は、エラーコードを返します。

4.1.2.10 SplashBitmap* OPRS::getBitmap ()

戻り値:

ピクセルマップ

描画対象となっているピクセルマップを返します。

ベクターモードの場合は、ピクセルマップを持ちませんので、NULL を返します。

4.1.2.11 SplashPattern* OPRS::getFillPattern ()

戻り値:
塗り潰しパターン

現在の塗り潰しパターンを返します。名前は、パターンですが現在単一色のみをサポートしています。

4.1.2.12 SplashCoord OPRS::getFlatness ()

戻り値:
平滑度

現在の平滑度を返します。ただし、現在は、常に 1 を返します。

4.1.2.13 int OPRS::getLineCap ()

戻り値:
ラインキャップ種別

現在のラインキャップ種別を返します。

4.1.2.14 SplashCoord* OPRS::getLineDash ()

戻り値:
ラインダッシュ配列

現在のラインダッシュのパターンの配列を返します。設定されていない場合には、NULL を返します。

4.1.2.15 int OPRS::getLineDashLength ()

戻り値:
ラインダッシュ配列の要素数

現在のラインダッシュのパターンの配列の要素数を返します。設定されていない場合には、0 を返します。

4.1.2.16 SplashCoord OPRS::getLineDashPhase ()

戻り値:
ラインダッシュのフェーズ

現在のラインダッシュのフェーズを返します。設定されていない場合には、0 を返します。

4.1.2.17 int OPRS::getLineJoin ()

戻り値:
ラインジョインの種別

現在のラインジョインの種別を返します。

4.1.2.18 SplashCoord OPRS::getLineWidth ()

戻り値:
線幅

現在の線幅を返します。

4.1.2.19 SplashCoord OPRS::getMiterLimit ()

戻り値:
マイターリミット

現在のマイターリミットを返します。

4.1.2.20 int OPRS::getRasterMode () [inline]

戻り値:
描画モード

現在の描画モードを返します。ラスターモードの場合は、gTrue、ベクターモードの場合は、gFalse を返します。

4.1.2.21 SplashPattern* OPRS::getStrokePattern ()

戻り値:
ストロークパターン

現在のストロークパターンを返します。名前は、パターンですが現在単一色のみをサポートしています。

4.1.2.22 int OPRS::init (const char * *driverName*, int *outputFD*, char * *printerModel*, int *rasterMode*, int *nOptions*, char * *optionKeys*[], char * *optionVals*[])

引数:

driverName プリンタドライバのライブラリ名

outputFD プリンタへの出力データの出力先を示すファイル・ディスクリプタ

printerModel プリンタのモデル名

rasterMode 描画モード

nOptions オプション引数の数

optionKeys オプション名の配列

optionVals オプション値の配列

戻り値:
エラー値

オープンプリント・レンダリング・サブルーチンの初期化を行います。driverName で示されるベクタプリンタ・ドライバを読み込み、OpenPrinter を呼出します。

driverName は、ドライバのファイル名を示しており、以下のパターンを試します。

```
<driverName>.so
<driverName>.dll
lib<drivername>.so
```

ライブラリは、dlopen を使用して読み込みます。従って、LD_LIBRARY_PATH が参照されます。

outputFD および printerModel は、ベクタプリンタ・ドライバの OpenPrinter に引数として渡されます。

rasterModeA が、gTrue の場合は、ラスターモード、gFalse の場合は、ベクターモードになります。ただし、gFalse を指定してもベクタプリンタ・ドライバが NewPath をサポートしていない場合は、ラスターモードになります。描画モードは、後で変更できません。

オプションは引数は、OPRS の動きを制御する引数です。オプション名と、そのオプションに対する値の二つの文字列の対で、ひとつのオプションを指定します。optionKeys には、オプション名の配列を指定し、optionVals には、オプションの値の配列を指定します。同じ添字の文字列が対となります。nOptions には、これらの配列の要素数を指定します。現在、有効なオプションは、以下の通りです。

オプション名	オプション値	説明
OPVP_OLDLIPSDRIVER	任意の文字列	プリンタドライバが古いタイプの CanonPageColor であることを示します。
OPVP_CLIPPATHNOTSAVED	任意の文字列	プリンタドライバが、グラフィックス状態保存の際に、クリッピングパスを保存しないことを示します。
OPVP_NOSHEARIMAGE	任意の文字列	プリンタドライバが、イメージの回転や変形をサポートしないことを示します。
OPVP_NOLINESTYLE	任意の文字列	プリンタドライバが、ラインダッシュをサポートしないことを示します。SetLineStyle あるいは SetLineDash がドライバに存在しなければ、自動的に設定されます。
OPVP_NOCLIPPATH	任意の文字列	プリンタドライバが、クリッピングパスをサポートしないことを示します。SetClipPath がドライバに存在しなければ、自動的に設定されます。
OPVP_NOBITMAPCHAR	任意の文字列	ベクターモードで文字をビットマップとしてプリンタドライバに渡さない様にします。

成功時には、splashOk を返します。その他の場合は、エラーコードを返します。

4.1.2.23 void OPRS::initGS (int *colorMode*, int *w*, int *h*, SplashColor *paperColor*)

引数:

colorMode カラーモード

w 描画対象の幅

h 描画対象の高さ

paperColor ペーパカラー

グラフィック状態の初期化を行います。ラスターモードの場合は、ペーパカラーでクリアします。(w, h) の大きさのクリッピング領域を設定します。

4.1.2.24 int OPRS::OPVPEndDoc ()

戻り値:

エラー値

ドキュメント終了処理を行います。ベクタープリンタ・ドライバの EndDoc を呼出し、EndDoc の返値を返します。

成功時には、0 を返します。

4.1.2.25 int OPRS::OPVPEndJob ()

戻り値:

エラー値

ジョブ終了処理を行います。ベクタープリンタ・ドライバの EndJob を呼出し、EndJob の返値を返します。

成功時には、0 を返します。

4.1.2.26 int OPRS::OPVPEndPage ()

戻り値:

エラー値

ページ終了処理を行います。ベクタープリンタ・ドライバの EndPage を呼出し、EndPage の返値を返します。

endPage メソッドとの違いは、endPage メソッドは、PDF ファイル内に定義されているページ終了時の処理を行うのに対し、このメソッドは、プリンタドライバに対するページ終了処理を行います。

本メソッドは、、endPage の後に呼出すことになります。スライス出力を行う場合には、各スライス毎に、endPage を呼出しますが、本メソッドは最後に一度だけ呼出します。

成功時には、0 を返します。

4.1.2.27 int OPRS::OPVPStartDoc (char * *docInfo*)

引数:

docInfo ドキュメント情報

戻り値:

エラー値

ドキュメント開始処理を行います。docInfo を引数にして、ベクタープリンタ・ドライバの StartDoc を呼出し、StartDoc の返値を返します。

成功時には、0 を返します。

4.1.2.28 int OPRS::OPVPStartJob (char * *jobInfo*)

引数:

jobInfo ジョブ情報

戻り値:

エラー値

ジョブ開始処理を行います。jobInfo を引数にして、ベクタープリンタ・ドライバの StartJob を呼出し、StartJob の返値を返します。

成功時には、0 を返します。

4.1.2.29 int OPRS::OPVPStartPage (char * *pageInfo*, int *rasterWidth*)

引数:

pageInfo ページ情報

rasterWidth 描画対象の幅 (ピクセル)

戻り値:

エラー値

ページ開始処理を行います。pageInfo を引数にして、ベクタープリンタ・ドライバの StartPage を呼出し、StartPage の返値を返します。

成功時には、0 を返します。

4.1.2.30 int OPRS::outSlice ()

戻り値:

エラー値

ラスターモードの場合に、スライス・データをプリンタドライバに出力します。ベクターモードの場合は、なにもしません。

成功時には、0 を返します。

4.1.2.31 SplashError OPRS::restoreState ()

戻り値:

エラー値

以前に保存したグラフィックス状態を回復します。

成功時には、splashOk を返します。

4.1.2.32 void OPRS::saveState ()

グラフィックス状態を保存します。

4.1.2.33 int OPRS::setBitmap (SplashBitmap * *bitmapA*)

引数:

bitmapA ピクセルマップ

戻り値:

エラー値

描画対象のピクセルマップを設定します。ラスターモードの場合は、描画に先立ってピクセルマップを作成して、このメソッドを呼ぶ必要があります。ピクセルマップの削除は、呼び出し側の責任になります。

このピクセルマップの範囲を越えて描画すると、不正なメモリアクセスを起します。クリッピング領域を設定するなどして、ピクセルマップの範囲外に描画しないように注意してください。

成功時には、0 を返します。

4.1.2.34 int OPRS::setColorMode (int *colorModeA*)

引数:

colorModeA カラーモード

戻り値:

エラー値

カラーモードを設定します。現状では、splashModeRGB8 のみサポートします。

成功時には、0 を返します。

4.1.2.35 void OPRS::setDebugMode (GBool *debugModeA*)

引数:

debugMode デバッグモード

デバッグモードをを設定します。

4.1.2.36 void OPRS::setFillPattern (SplashPattern * *fillColor*)

引数:

fillPattern 塗り潰しパターン

塗り潰しパターンを設定します。名前は、パターンですが現在単一色のみをサポートしています。

4.1.2.37 void OPRS::setFlatness (SplashCoord *flatness*)

引数:

flatness 平滑度

平滑度を設定します。ただし、現状では何もしません。

4.1.2.38 void OPRS::setLineCap (int *lineCap*)

引数:

lineCap ラインキャップ種別

ラインキャップ種別を設定します。

4.1.2.39 void OPRS::setLineDash (SplashCoord * *lineDash*, int *lineDashLength*, SplashCoord *lineDashPhase*)

引数:

lineDash ラインダッシュ配列

lineDashLength ラインダッシュ配列の要素数

lineDashPhase ラインダッシュのフェーズ

ラインダッシュを設定します。

lineDash が NULL か、lineDashLength が 0 の場合は、実線を表します。

4.1.2.40 void OPRS::setLineJoin (int *lineJoin*)

引数:

lineJoin ラインジョイン種別

ラインジョイン種別を設定します。

4.1.2.41 void OPRS::setLineWidth (SplashCoord *lineWidth*)

引数:

lineWidth 線幅

線幅を設定します。

4.1.2.42 void OPRS::setMiterLimit (SplashCoord *miterLimit*)

引数:

miterLimit マイターリミット

マイターリミットを設定します。

4.1.2.43 void OPRS::setStrokePattern (SplashPattern * *strokeColor*)

引数:

strokePattern ストロークパターン

ストロークパターンを設定します。名前は、パターンですが現在単一色のみ
をサポートしています。

4.1.2.44 SplashError OPRS::stroke (SplashPath * *path*)

引数:

path パス

戻り値:

エラー値

パスに沿って線を描画します。

成功時には、splashOk を返します。

4.1.2.45 int OPRS::unloadVectorDriver ()

戻り値:

エラー値

ベクタープリンタ・ドライバを解放します。

成功時には、0 を返します。

このクラスの解説は次のファイルから生成されました:

- **OPRS.h**

4.2 クラス SplashBitmap の解説

`#include <SplashBitmap.h>`

ピクセルマップを表すクラスです。名前は、Bitmap ですが、ピクセルのビット数は、1 とは限りません。

Public メソッド

- **SplashBitmap** (int widthA, int heightA, **SplashColorMode** modeA)
- **~SplashBitmap** ()
- int **getWidth** ()
- int **getHeight** ()
- int **getRowSize** ()
- **SplashColorMode getMode** ()
- SplashColorPtr **getDataPtr** ()
- **SplashError writePNMFile** (char *fileName)

4.2.1 コンストラクタとデストラクタの解説

4.2.1.1 SplashBitmap::SplashBitmap (int *widthA*, int *heightA*, SplashColorMode *modeA*)

引数:

widthA ピクセルマップの幅 (ピクセル)

widthA ピクセルマップの高さ (ピクセル)

modeA カラーモード

幅 widthA, 高さ heightA, カラーモード modeA のピクセルマップを作成します。

4.2.1.2 SplashBitmap::~SplashBitmap ()

ピクセルマップを破壊します。

4.2.2 メソッドの解説

4.2.2.1 SplashColorPtr SplashBitmap::getDataPtr () [inline]

戻り値:
ピクセルマップデータの先頭へのポインタ

ピクセルマップのデータの先頭へのポインタを返します。

4.2.2.2 int SplashBitmap::getHeight () [inline]

戻り値:
ピクセルマップの高さ (ピクセル数)

ピクセルマップの高さを返します。

4.2.2.3 SplashColorMode SplashBitmap::getMode () [inline]

戻り値:
ピクセルマップのカラーモード

ピクセルマップのカラーモードを返します。

4.2.2.4 int SplashBitmap::getRowSize () [inline]

戻り値:
ピクセルマップの 1 行のバイト数

ピクセルマップの 1 行のバイト数を返します。

4.2.2.5 int SplashBitmap::getWidth () [inline]

戻り値:
ピクセルマップの幅 (ピクセル数)

ピクセルマップの幅を返します。

4.2.2.6 SplashError SplashBitmap::writePNMFile (char * *fileName*)

4.2.3 フレンドと関連する関数の解説

4.2.3.1 friend class OPVPSplash [friend]

4.2.3.2 friend class Splash [friend]

引数:

fileName ファイル名

ファイル名 fileName のファイルにピクセルマップを PNM ファイルとして出力します。

成功時には、splashOk を返します。

このクラスの解説は次のファイルから生成されました:

- **SplashBitmap.h**

4.3 共用体 SplashColor の解説

`#include <SsplashTypes.h>`

カラーを表す共用体です。ピクセル値を表すこともあります。

Public 変数

- **SplashMono1 mono1**
- **SplashMono8 mono8**
- **SplashRGB8 rgb8**
- **SplashBGR8 bgr8**

4.3.1 変数の解説

4.3.1.1 SplashBGR8 SplashColor::bgr8

BGR8 形式のカラーを示します。

上位のビットから 8 ビット単位で Blue、Green、Red で構成されます。値が大きほど輝度が大きくなります。

4.3.1.2 SplashMono1 SplashColor::mono1

1 ビットの白黒値を示します。

4.3.1.3 SplashMono8 SplashColor::mono8

8 ビットのグレイスケールを示します。値が大きいほど輝度は大きくなります。

4.3.1.4 SplashRGB8 SplashColor::rgb8

RGB8 形式のカラーを示します。

上位のビットから 8 ビット単位で Red、Green、Blue で構成されます。値が大きほど輝度が大きくなります。

この共用体の解説は次のファイルから生成されました:

- **SplashTypes.h**

4.4 共用体 SplashColorPtr の解説

`#include <SplashTypes.h>`

カラーを表す SplashColor へのポインタを表します。

Public 変数

- **SplashMono1P * mono1**
- **SplashMono8 * mono8**
- **SplashRGB8 * rgb8**
- **SplashBGR8P * bgr8**

4.4.1 変数の解説

4.4.1.1 SplashBGR8P* SplashColorPtr::bgr8

BGR8 形式のカラーへのポインタです。

4.4.1.2 SplashMono1P* SplashColorPtr::mono1

1 ビット白黒のカラーへのポインタです。

4.4.1.3 SplashMono8* SplashColorPtr::mono8

8 ビットのフレイスケールカラーへのポインタです。

4.4.1.4 SplashRGB8* SplashColorPtr::rgb8

RGB8 形式のカラーへのポインタです。

この共用体の解説は次のファイルから生成されました:

- **SplashTypes.h**

4.5 クラス SplashFont の解説

`#include <SplashFont.h>`

フォントを表すクラスです。グリフのビットマップのキャッシュを備えています。

Public メソッド

- **SplashFont** (SplashFontFile *fontFileA, **SplashCoord** *matA, GBool aaA)
- void **initCache** ()
- virtual ~**SplashFont** ()
- SplashFontFile * **getFontFile** ()
- GBool **matches** (SplashFontFile *fontFileA, **SplashCoord** *matA)
- virtual GBool **getGlyph** (int c, int xFrac, int yFrac, SplashGlyphBitmap *bitmap, Unicode *u)
- virtual GBool **makeGlyph** (int c, int xFrac, int yFrac, SplashGlyphBitmap *bitmap, Unicode *u)=0
- virtual SplashPath * **getGlyphPath** (int c, Unicode *u)=0

4.5.1 コンストラクタとデストラクタの解説

4.5.1.1 SplashFont::SplashFont (SplashFontFile * *fontFileA*, SplashCoord * *matA*, GBool *aaA*)

引数:

fontFileA フォントファイル

matA フォント行列

aaA アンチエイリアシング・フラグ

fontFileA で表され、フォント行列 matA のフォントを作成します。aaA が gTrue の場合は、グリフのビットマップを作成するときにアンチエイリアシングを行います。

4.5.1.2 virtual SplashFont::~SplashFont () [virtual]

フォントを削除します。

4.5.2 メソッドの解説

4.5.2.1 SplashFontFile* SplashFont::getFontFile () [inline]

戻り値:
フォントファイル

フォントファイルへのポインタを返します。

4.5.2.2 virtual GBool SplashFont::getGlyph (int *c*, int *xFrac*, int *yFrac*, SplashGlyphBitmap * *bitmap*, Unicode * *u*) [virtual]

引数:
c 文字のコード (あるいは CID)

xFrac y 方向オフセット

yFrac y 方向オフセット

bitmap 取得したビットマップグリフが返される

u 文字のユニコードへのポインタ

戻り値:
取得出来たか否か

c または、u で示される文字のグリフをビットマップ形式で取得します。取得時には、まず、内部のキャッシュを検索して、存在すれば、それを返します。

TrueType あるいは OpenType フォントの場合で、ロード時に codeToGID が指定されなかった場合は、u が文字の選択に使用されます。

取得出来た場合は、gTrue、出来なかった場合は gFalse を返します。

x, y は、描画開始点の小数部分です。splashFontFraction を掛けたものを渡します。これは、グリフをスケールする際に誤差を小くするために使用されます。

4.5.2.3 virtual SplashPath* SplashFont::getGlyphPath (int *c*, Unicode * *u*) [pure virtual]

引数:
c 文字のコード (CID)

u 文字のユニコードへのポインタ

戻り値:
パス

c または、u で示される文字のグリフをパスにして返します。TrueType あるいは OpenType フォントの場合で、ロード時に codeToGID が指定されなかった場合は、u が文字の選択に使用されます。

アウトラインが取得できない場合は、NULL を返します。

4.5.2.4 void SplashFont::initCache ()

ビットマップグリフのキャッシュを初期化します。

4.5.2.5 virtual GBool SplashFont::makeGlyph (int *c*, int *xFrac*, int *yFrac*, SplashGlyphBitmap * *bitmap*, Unicode * *u*) [pure virtual]

引数:
c 文字の CID

xFrac y 方向オフセット

yFrac y 方向オフセット

bitmap 取得したビットマップグリフが返される

u 文字のユニコードへのポインタ

戻り値:
取得出来たか否か

c または、u で示される文字のグリフをビットマップ形式で取得します。getGlyph と異り、取得時には、キャッシュを参照しません。

TrueType あるいは OpenType フォントの場合で、ロード時に codeToGID が指定されなかった場合は、u が文字の選択に使用されます。

取得出来た場合は、gTrue、出来なかった場合は gFalse を返します。

x, y は、描画開始点の小数部分です。splashFontFraction を掛けたものを渡します。これは、グリフをスケールする際に誤差を小くするために使用されます。

4.5.2.6 GBool SplashFont::matches (SplashFontFile * *fontFileA*, SplashCoord * *matA*) [inline]

引数:

fontFileA フォントファイル

matA フォント行列

戻り値:

一致したか否か

このフォントのフォントファイルとフォント行列が fontFileA と matA に一致するかどうかを返します。

一致した場合は、gTrue、そうで無い場合は、gFalse を返します。

このクラスの解説は次のファイルから生成されました:

- **SplashFont.h**

4.6 クラス SplashFontEngine の解説

`#include <SplashFontEngine.h>`

フォントのファイルを扱うクラスです。フォントのキャッシュも扱います。

Public メソッド

- **SplashFontEngine** (GBool aa)
- ~**SplashFontEngine** ()
- SplashFontFile * **getFontFile** (SplashFontFileID *id)
- SplashFontFile * **loadType1Font** (SplashFontFileID *idA, char *fileName, GBool deleteFile, char **enc)
- SplashFontFile * **loadType1CFont** (SplashFontFileID *idA, char *fileName, GBool deleteFile, char **enc)
- SplashFontFile * **loadCIDFont** (SplashFontFileID *idA, char *fileName, GBool deleteFile)
- SplashFontFile * **loadTrueTypeFont** (SplashFontFileID *idA, char *fileName, GBool deleteFile, Gushort *codeToGID, int codeToGIDLen)
- SplashFont * **getFont** (SplashFontFile *fontFile, **SplashCoord** *mat)

4.6.1 コンストラクタとデストラクタの解説

4.6.1.1 SplashFontEngine::SplashFontEngine (GBool *aa*)

引数:

enableT1lib T1 ライブラリを使うかどうか

enableFreeType FreeType2 ライブラリを使うかどうか

aa アンチエイリアシングを行うかどうか

フォントのキャッシュの初期化など初期化を行います。

enableT1lib は、T1 ライブラリを使うかどうかを指定します。gTrue を指定すると使用します。gFalse を指定すると Type1 フォントが使用できなくなります。

enableFreeType は、FreeType2 ライブラリを使うかどうかを指定します。gTrue を指定すると使用します。gFalse を使用すると、TrueType や OpenType、CID フォントが使用できなくなります。通常は、enableT1lib, enbaleFreeType 共に gTrue を指定してください。

aa は、アンチエイリアシングを行うかどうかを指定します。gTrue を指定すると行います。ベクターモードの場合は、gFalse を指定してください。

4.6.1.2 SplashFontEngine::~SplashFontEngine ()

キャッシュなどの解放を行います。

4.6.2 メソッドの解説

4.6.2.1 SplashFont* SplashFontEngine::getFont (SplashFontFile * *fontFile*, SplashCoord * *mat*)

引数:

fontFile フォントファイル

mat フォント行列

戻り値:

フォント

フォントファイル fontFile、フォント行列 mat で示されるフォントを取得します。まず、キャッシュ内を検索して、存在すれば、それを返します。

フォントへのポインタを返します。エラー時には、NULL を返します。

4.6.2.2 SplashFontFile* SplashFontEngine::getFontFile (SplashFontFileID * *id*)

引数:

id フォントファイル ID

戻り値:

フォントファイル

キャッシュを検索し、id に一致する ID を持つフォントファイルを返します。

ID は、使用する側で、ロード時にユニークになるように付与したものです。

見付からなければ、NULL を返します。

4.6.2.3 SplashFontFile* SplashFontEngine::loadCIDFont (SplashFontFileID * *idA*, char * *fileName*, GBool *deleteFile*)

引数:

id フォントファイル ID

fileName ファイル名

deleteFile ファイル削除フラグ

戻り値:

フォントファイル

fileName で指定されるファイルを CID フォントとしてロードし、フォントファイル・オブジェクトを作成します。

id は、使用する側で、ロード時にユニークになるように作成し、指定します。後に、キャッシュを検索する場合に使用します。

deleteFile に gTrue を指定すると、ロード後にファイル fileName を削除します。

フォントファイル・オブジェクトへのポインタを返します。エラー時には、NULL を返します。

4.6.2.4 SplashFontFile* SplashFontEngine::loadTrueTypeFont (SplashFontFileID * *idA*, char * *fileName*, GBool *deleteFile*, Gushort * *codeToGID*, int *codeToGIDLen*)

引数:

id フォントファイル ID

fileName ファイル名

deleteFile ファイル削除フラグ

codeToGID CID から GID への変換テーブル

codeToGIDLen codeToGID の大きさ

戻り値:

フォントファイル

fileName で指定されるファイルを TrueType あるいは OpenType フォントとしてロードし、フォントファイル・オブジェクトを作成します。

id は、使用する側で、ロード時にユニークになるように作成し、指定します。後に、キャッシュを検索する場合に使用します。

deleteFile に gTrue を指定すると、ロード後にファイル fileName を削除します。

codeToGID は、CID(Character ID) から GID(Glyph ID) への変換テーブルです。codeToGID[cid] が GID を表す配列です。NULL を指定すると SplashFont::getGlyph や SplashFont::getGlyphPath のグリフの取得の際には、CID を指定する引数 c ではなく、ユニコードを指定する u を使用します。

codeToGIDLen は、codeToGID 配列の要素数を示します。codeToGID が NULL の場合は、0 を指定します。

フォントファイル・オブジェクトへのポインタを返します。エラー時には、NULL を返します。

4.6.2.5 SplashFontFile* SplashFontEngine::loadType1CFont (SplashFontFileID * *idA*, char * *fileName*, GBool *deleteFile*, char ** *enc*)

引数:

id フォントファイル ID

fileName ファイル名

deleteFile ファイル削除フラグ

enc エンコーディング・テーブル

戻り値:

フォントファイル

fileName で指定されるファイルを Type1C フォント (Compact File Format で表現された Type1 と同等のフォント) としてロードし、フォントファイル・オブジェクトを作成します。

id は、使用する側で、ロード時にユニークになるように作成し、指定します。後に、キャッシュを検索する場合に使用します。

deleteFile に gTrue を指定すると、ロード後にファイル fileName を削除します。

enc は、エンコーディング・テーブルで要素数 256 の文字列へのポインタの配列です。文字列は、文字グリフの名前を表します。ポインタが NULL の場合は、".notde" と同等です。文字グリフの名前は、アドビ社の PDF や Type1 フォントのドキュメントを参照してください。enc[code] の表す文字列が、文字 code のグリフの名前となります。

フォントファイル・オブジェクトへのポインタを返します。エラー時には、NULL を返します。

4.6.2.6 SplashFontFile* SplashFontEngine::loadType1Font (SplashFontFileID * *idA*, char * *fileName*, GBool *deleteFile*, char ** *enc*)

引数:

id フォントファイル ID

fileName ファイル名

deleteFile ファイル削除フラグ

enc エンコーディング・テーブル

戻り値:

フォントファイル

fileName で指定されるファイルを Type1 フォントとしてロードし、フォントファイル・オブジェクトを作成します。

id は、使用する側で、ロード時にユニークになるように作成し、指定します。後に、キャッシュを検索する場合に使用します。

deleteFile に gTrue を指定すると、ロード後にファイル fileName を削除します。

enc は、エンコーディング・テーブルで要素数 256 の文字列へのポインタの配列です。文字列は、文字グリフの名前を表します。ポインタが NULL の場合は、".notde" と同等です。文字グリフの名前は、アドビ社の PDF や Type1 フォントのドキュメントを参照してください。enc[code] の表す文字列が、文字 code のグリフの名前となります。

フォントファイル・オブジェクトへのポインタを返します。エラー時には、NULL を返します。

このクラスの解説は次のファイルから生成されました:

- **SplashFontEngine.h**

4.7 クラス SplashFontFile の解説

`#include <SplashFontFile.h>`

フォントのファイルを表すクラスです。

Public メソッド

- virtual ~**SplashFontFile** ()
- virtual SplashFont * **makeFont** (**SplashCoord** *mat)=0
- SplashFontFileID * **getID** ()
- void **incRefCnt** ()
- void **decRefCnt** ()

4.7.1 コンストラクタとデストラクタの解説

4.7.1.1 virtual SplashFontFile::~SplashFontFile () [virtual]

引数:

idA フォントファイル ID

fileNameA ファイル名

deleteFileA ファイル削除フラグ

ID を idA、ファイル名 fileNameA、ファイル削除フラグ deleteFileA のフォントファイルを作成します。

通常、直接作成してはいけません。SplashFontEngine を利用して作成します。

4.7.1.2 SplashFontFile::SplashFontFile (SplashFontFileID * *idA*, char * *fileNameA*, GBool *deleteFileA*) [protected]

デストラクタです。

4.7.2 メソッドの解説

4.7.2.1 void SplashFontFile::decRefCnt ()

リファレンス・カウンタを 1 減らします。0 になると、自身を削除します。

4.7.2.2 SplashFontFileID* SplashFontFile::getID () [inline]

戻り値:
ID

ID を返します。

4.7.2.3 void SplashFontFile::incRefCnt ()

リファレンス・カウンタを 1 増やします。

4.7.2.4 virtual SplashFont* SplashFontFile::makeFont (SplashCoord * *mat*) [pure virtual]

引数:
mat フォント行列

戻り値:
フォント

このフォントファイルで示され、mat で指示されるフォント行列を持つフォントを作成します。

通常は、直接呼出しません、SplashFontEngine の getFont を使用します。

作成したフォントへのポインタを返します。エラー時は、NULL を返します。

このクラスの解説は次のファイルから生成されました:

- **SplashFontFile.h**

4.8 構造体 SplashGlyphBitmap の解説

`#include <SplashGlyphBitmap.h>`

文字のグリフのビットマップを表す構造体です。

Public 変数

- int **x**
- int **y**
- int **w**
- int **h**
- GBool **aa**
- Guchar * **data**
- GBool **freeData**

4.8.1 変数の解説

4.8.1.1 GBool SplashGlyphBitmap::aa

アンチエイリアシングを行ったどうかのフラグです。gTrue の場合は、アンチエイリアシングを行っており、データは 8bit のアルファ値のビットマップです。アンチエイリアシングを行うとベクターモードでは、扱えません。

gFalse の場合は、ビットマップです。

4.8.1.2 Guchar* SplashGlyphBitmap::data

実際のビットマップ・データの先頭を指すポインタです。

4.8.1.3 GBool SplashGlyphBitmap::freeData

data を解放しないといけないかどうかを示しています。ただし、これは、内部で使用されるフラグなので、ユーザが解放してはいけません。

4.8.1.4 int SplashGlyphBitmap::h

グリフの高さ (bit) を示します。

4.8.1.5 int SplashGlyphBitmap::w

グリフの幅 (bit) を示します。

4.8.1.6 int SplashGlyphBitmap::x

グリフの x オフセットを示します。もとのグリフよりも x だけ、左にオフセットして描画されていることを示します。

4.8.1.7 int SplashGlyphBitmap::y

グリフの y オフセットを示します。もとのグリフよりも y だけ、下にオフセットして描画されていることを示します。

この構造体の解説は次のファイルから生成されました:

- **SplashGlyphBitmap.h**

4.9 クラス SplashPath の解説

`#include <SplashPath.h>`

パスを表すクラスです。

Public メソッド

- **SplashPath** ()
- SplashPath * **copy** ()
- ~**SplashPath** ()
- void **append** (SplashPath *path)
- **SplashError moveTo** (**SplashCoord** x, **SplashCoord** y)
- **SplashError lineTo** (**SplashCoord** x, **SplashCoord** y)
- **SplashError curveTo** (**SplashCoord** x1, **SplashCoord** y1, **SplashCoord** x2, **SplashCoord** y2, **SplashCoord** x3, **SplashCoord** y3)
- **SplashError arcCWTo** (**SplashCoord** x1, **SplashCoord** y1, **SplashCoord** xc, **SplashCoord** yc)
- **SplashError close** ()
- void **offset** (**SplashCoord** dx, **SplashCoord** dy)
- GBool **getCurPt** (**SplashCoord** *x, **SplashCoord** *y)
- void **getBBox** (int *xMinA, int *yMinA, int *xMaxA, int *yMaxA)

4.9.1 コンストラクタとデストラクタの解説

4.9.1.1 SplashPath::SplashPath ()

空のパスを作成します。

4.9.1.2 SplashPath::~SplashPath ()

パスを削除します。

4.9.2 メソッドの解説

4.9.2.1 void SplashPath::append (SplashPath * *path*)

引数:

path パス

path で示されるパスを現在のパスに追加します。

4.9.2.2 SplashError SplashPath::arcCWTo (SplashCoord *x1*, SplashCoord *y1*, SplashCoord *xc*, SplashCoord *yc*)

引数:

x1 終了点 x 座標

y1 終了点 y 座標

xc 中心点 x 座標

yc 中心点 y 座標

戻り値:

エラー値

現在の点から (x1, y1) に (xc, yc) を中心とする円弧をパスに加えます。

ただし、現在、オープンプリント・レンダリング・サブルーチンでは、扱えないの使用しないでください。

成功時には、splashOk を返します。

4.9.2.3 SplashError SplashPath::close ()

戻り値:

エラー値

パスを閉じます。

成功時には、splashOk を返します。

4.9.2.4 SplashPath* SplashPath::copy () [inline]

戻り値:

新しいパス

パスをコピーします。

コピーして作成した新しいパスへのポインタを返します。

4.9.2.5 SplashError SplashPath::curveTo (SplashCoord *x1*, SplashCoord *y1*, SplashCoord *x2*, SplashCoord *y2*, SplashCoord *x3*, SplashCoord *y3*)

引数:

x1 第 1 制御点 x 座標

y1 第 1 制御点 y 座標

x2 第 2 制御点 x 座標

y2 第 2 制御点 y 座標

x3 終了点 x 座標

y3 終了点 y 座標

戻り値:

エラー値

現在の点から (x3, y3) まで、(x1, y1)、(x2, y2) を制御点として 3 次ベジェ曲線を追加します。

成功時には、splashOk を返します。

4.9.2.6 void SplashPath::getBBox (int * *xMinA*, int * *yMinA*, int * *xMaxA*, int * *yMaxA*)

引数:

xMinA x 座標最小値へのポインタ

yMinA y 座標最小値へのポインタ

xMaxA x 座標最大値へのポインタ

yMaxA y 座標最大値へのポインタ

バウンディングボックスを xMinA, yMinA, xMaxA, yMaxA に返します。

4.9.2.7 GBool SplashPath::getCurPt (SplashCoord * x, SplashCoord * y)

引数:

x 現在の点の x 座標

y 現在の点の y 座標

戻り値:

現在の点が存在しているか否か

現在の点が存在していれば、x, y に返す。

現在の点が存在していれば、gTrue を、存在していなければ gFalse を返す。

4.9.2.8 SplashError SplashPath::lineTo (SplashCoord x, SplashCoord y)

引数:

x 終了点 x 座標

y 終了点 y 座標

戻り値:

エラー値

現在の点から (x, y) まで直線を追加します。

成功時には、splashOk を返します。

4.9.2.9 SplashError SplashPath::moveTo (SplashCoord x, SplashCoord y)

引数:

x 終了点 x 座標

y 終了点 y 座標

戻り値:

エラー値

現在の点を (x, y) に移動します。

成功時には、splashOk を返します。

4.9.2.10 void SplashPath::offset (SplashCoord *dx*, SplashCoord *dy*)

引数:

dx x オフセット

dy y オフセット

パス全体を (x,y) だけ平行移動します。

このクラスの解説は次のファイルから生成されました:

- **SplashPath.h**

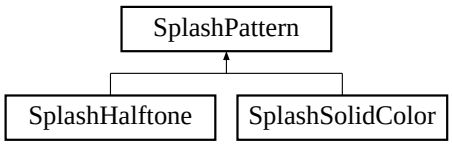
4.10 クラス SplashPattern の解説

`#include <SplashPattern.h>`

パターンを表す抽象クラスです。現在、単一色のみをサポートしています。

SplashHalfTone と SplashSolidColor の実装がありますが、SplashHalfTone は、使用しませんので、ここでは解説しません。

SplashPattern に対する継承グラフ:



Public メソッド

- **SplashPattern** ()
- virtual SplashPattern * **copy** ()=0
- virtual ~**SplashPattern** ()
- virtual SplashColor **getColor** (int x, int y)=0

4.10.1 コンストラクタとデストラクタの解説

4.10.1.1 SplashPattern::SplashPattern ()

パターンを作成します。

4.10.1.2 virtual SplashPattern::~SplashPattern () [virtual]

パターンを削除します。

4.10.2 メソッドの解説

4.10.2.1 virtual SplashPattern* SplashPattern::copy () [pure virtual]

戻り値:

新しいパターンオブジェクトへのポインタ

コピーをします。

コピーをした新しいオブジェクトへのポインタを返します。エラー時には、NULL を返します。

SplashSolidColor (p. 63) が実装しています.

4.10.2.2 virtual SplashColor SplashPattern::getColor (int *x*, int *y*) [pure virtual]

引数:

x x 座標

y y 座標

戻り値:

カラー

(x, y) 座標のカラーを返します。

SplashSolidColor (p. 63) が実装しています.

このクラスの解説は次のファイルから生成されました:

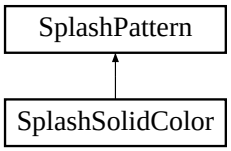
- **SplashPattern.h**

4.11 クラス SplashSolidColor の解説

`#include <SplashPattern.h>`

単一色を表す SplashPattern クラスの実装です。

SplashSolidColor に対する継承グラフ:



Public メソッド

- **SplashSolidColor** (SplashColor colorA)
- virtual SplashPattern * **copy** ()
- virtual ~**SplashSolidColor** ()
- virtual SplashColor **getColor** (int x, int y)

4.11.1 コンストラクタとデストラクタの解説

4.11.1.1 SplashSolidColor::SplashSolidColor (SplashColor *colorA*)

引数:

colorA カラー

colorA で示されるカラーの単一色パターンを作成します。

4.11.1.2 virtual SplashSolidColor::~SplashSolidColor () [virtual]

パターンを削除します。

4.11.2 メソッドの解説

4.11.2.1 virtual SplashPattern* SplashSolidColor::copy () [inline, virtual]

戻り値:

新しいパターンオブジェクトへのポインタ

コピーをします。

コピーをした新しいオブジェクトへのポインタを返します。エラー時には、NULL を返します。

SplashPattern (p. 61) に実装されています.

4.11.2.2 virtual SplashColor SplashSolidColor::getColor (int *x*, int *y*) [virtual]

引数:

x x 座標

y y 座標

戻り値:

カラー

(x, y) 座標のカラーを返します。単一色のため、x,y にかかわらず同一のカラーを返します。

SplashPattern (p. 61) を実装しています。

このクラスの解説は次のファイルから生成されました:

- **SplashPattern.h**

第5章　オープンプリンティング・レンダリング・サブルーチン ファイルの解説

5.1　OPRS.h の解説

```
#include <aconf.h>
#include "SplashTypes.h"
#include "opvp_common.h"
#include "Splash.h"
#include "OPVPSplash.h"
```

構成

- class **OPRS**

Typedef

- typedef GBool(∗ **SplashImageMaskSource**)(void ∗data, **Splash-Mono1** ∗pixel)
- typedef GBool(∗ **SplashImageSource**)(void ∗data, SplashColor ∗pixel, Guchar ∗alpha)

5.1.1 Typedef の解説

5.1.1.1 typedef GBool(∗ SplashImageMaskSource)(void ∗data, SplashMono1 ∗pixel)

fillImageMask に指定するイメージマスク・ソース関数のタイプです。

この関数は、以下の形式で呼出されますので、左上のピクセルから行優先の順にピクセルのカラーを color に返さなければいけません。

```
Guchar alpha;
SplashMono1 color;

(*src)(srcData, &color);
```

5.1.1.2 typedef GBool(∗ SplashImageSource)(void ∗data, SplashColor ∗pixel, Guchar ∗alpha)

drawImage に指定するイメージ・ソース関数のタイプです。

この関数は、以下の形式で呼出されますので、左上のピクセルから行優先の順にピクセルのカラーを color に、アルファ値を alpha に返さなければいけません。

```
Guchar alpha;
SplashColor color;

(*src)(srcData, &color,&alpha);
```

ただし、アルファ値は、0 か 0 でないかのみで判断され、0 で無いときは、ピクセル値でそのまま、描画されます。通常のアルファ値とは、異なる扱いを受けるので注意して下さい。

5.2 SplashTypes.h の解説

```
#include <aconf.h>

#include "gtypes.h"

#include "../xpdf/CharTypes.h"
```

構成

- union **SplashColor**
- union **SplashColorPtr**

マクロ定義

- #define **splashMaxColorComps** 3
- #define **splashRGB8R**(rgb8) (((rgb8) >> 16) & 0xff)
- #define **splashRGB8G**(rgb8) (((rgb8) >> 8) & 0xff)
- #define **splashRGB8B**(rgb8) ((rgb8) & 0xff)
- #define **splashMakeRGB8**(r, g, b) ((((r) & 0xff) << 16) | (((g) & 0xff) << 8) | ((b) & 0xff))
- #define **splashBGR8R**(bgr8) ((bgr8) & 0xff)
- #define **splashBGR8G**(bgr8) (((bgr8) >> 8) & 0xff)
- #define **splashBGR8B**(bgr8) (((bgr8) >> 16) & 0xff)
- #define **splashMakeBGR8**(r, g, b) ((((b) & 0xff) << 16) | (((g) & 0xff) << 8) | ((r) & 0xff))

Typedef

- typedef double **SplashCoord**
- typedef Guchar **SplashMono1**
- typedef Guchar **SplashMono1P**
- typedef Guchar **SplashMono8**
- typedef Guint **SplashRGB8**
- typedef Guint **SplashBGR8**
- typedef Guchar **SplashBGR8P**
- typedef int **SplashError**

Enum

- enum **SplashColorMode** { **splashModeMono1**, **splashModeMono8**, **splashModeRGB8**, **splashModeBGR8Packed** }

5.2.1 マクロ定義の解説

5.2.1.1 #define splashBGR8B(bgr8) (((bgr8) >> 16) & 0xff)

SplashBGR8 タイプ bgr8 の Blue の値を取り出して返します。

5.2.1.2 #define splashBGR8G(bgr8) (((bgr8) >> 8) & 0xff)

SplashBGR8 タイプ bgr8 の Green の値を取り出して返します。

5.2.1.3 #define splashBGR8R(bgr8) ((bgr8) & 0xff)

SplashBGR8 タイプ bgr8 の Red の値を取り出して返します。

5.2.1.4 #define splashMakeBGR8(r, g, b) ((((b) & 0xff) << 16) | (((g) & 0xff) << 8) | ((r) & 0xff))

Red、Green、Blue の値が各々r、g、b である SplashBGR8 の値を返します。

5.2.1.5 #define splashMakeRGB8(r, g, b) ((((r) & 0xff) << 16) | (((g) & 0xff) << 8) | ((b) & 0xff))

Red、Green、Blue の値が各々r、g、b である SplashRGB8 の値を返します。

5.2.1.6 #define splashMaxColorComps 3

色の構成コンポーネントの最大数を示します。

5.2.1.7 #define splashRGB8B(rgb8) ((rgb8) & 0xff)

SplashRGB8 タイプ rgb8 の Blue の値を取り出して返します。

5.2.1.8 #define splashRGB8G(rgb8) (((rgb8) >> 8) & 0xff)

SplashRGB8 タイプ rgb8 の Green の値を取り出して返します。

5.2.1.9 #define splashRGB8R(rgb8) (((rgb8) >> 16) & 0xff)

SplashRGB8 タイプ rgb8 の Red の値を取り出して返します。

5.2.2 Typedef の解説

5.2.2.1 typedef Guint SplashBGR8

BGR8 の色の型です。

5.2.2.2 typedef Guchar SplashBGR8P

パックした BGR8 の色の型です。

5.2.2.3 typedef double SplashCoord

座標を示す型です。

5.2.2.4 typedef int SplashError

エラーを表す型です。

現在、以下の値があります。

splashOk	成功
splashErrNoCurPt	現在の点が存在しない
splashErrEmptyPath	パスが空
splashErrBogusPath	パスのデータがおかしい
splashErrNoSave	グラフィックス状態のスタックが空
splashErrOpenFile	ファイルをオープンできない
splashErrNoGlyph	文字グリフを取得できない
splashErrModeMismatch	カラーモードの組合せが不正
splashErrSingularMatrix	変換行列が正則でない
splashErrOPVP	ベクタプリンタドライバがエラーを返した

5.2.2.5 typedef Guchar SplashMono1

白黒 1 ビットのカラーを表す型

5.2.2.6 typedef Guchar SplashMono1P

パックされた白黒 1 ビットのカラーを表す型

5.2.2.7 typedef Guchar SplashMono8

グレイスケール 8 ビットのカラーを表す型

5.2.2.8 typedef Guint SplashRGB8

RGB8 のカラーを表す型 RGB8 は、Red, Green, Blue 各々8bit で、この順に配置されます。

5.2.3 Enum の解説

5.2.3.1 enum SplashColorMode

カラーモード

Enum 値:

splashModeMono1 白黒モード

splashModeMono8 8 ビット・グレイスケール・モード

splashModeRGB8 RGB8 モード

splashModeBGR8Packed

パック ド BGR8 モード

索 引