PDFMakerの紹介

PDFMakerはPDFファイルを生成するユニットです。PDFファイルはプラットホームに依存しないファイルフォーマットで、印刷することを前提にしたフォーマットであるため、例えば、HTML等と比較するとはるかに正確な位置に表示・印刷することが可能です。また、

1.非常に強力で多彩な描画機能を持っている。

2.仕様が完全に公開されている。

3.AcrobatReaderという高性能なビューワが無償で公開されている。

等の理由によりここ数年でかなりメジャーなフォーマットになりつつあります。

また、近年CD-Rのような大容量記録媒体の急速な発達にともない、帳票をデータとして保存したいというニーズも年々高まりつつあります。帳票を紙に出力するのではなくファイルに保存したい場合、レポートツールのファイル出力機能を使用したり、プリンタドライバのファイル保存機能を使用する方法がありますが、それらは専用のビューワが必要だったり、プリンタの機種に依存していたりするため、あまり実用的ではありません。正確にレイアウト可能でさまざまなプラットフォームに対応しているPDFファイルは帳票の保存形式として最も適したものの一つであるといえます。

通常PDF形式で保存する場合、レポートツールの出力をAcrobat等のアプケーションでPDF形式に変換しますが、逆転の発想でアプリケーションから直接PDFファイルを出力し、印刷はAcrobatReaderなどのビューワで行えばプリンタの制御からも解放されて、よりシンプルになる。というのがPDFMakerの開発理由です。PDFMakerは非常に軽いユニット（テキストファイルを出力しているだけなので当たり前ですが）であるため、夜間のバッチ処理で大量の帳票を作成する場合や、本社にあるサーバーで帳票をまとめて作成して、営業所のクライアントで後で印刷したい場合などに効果的に使用できると思います。

実装されている機能

PDFファイルはOSや機種に依存しない印刷環境と高速な表示を実現するため、比較的簡単でテキストエディタでも作成可能なHTML等のファイルフォーマットと比較すると、かなり複雑な構造を持っています。PDFMakerは、フォントやCross Refarenceテーブルなど PDFファイルの複雑な部分をラッピングして、比較的簡単にPDFファイルの作成を可能にするためのユニットです。

動作環境

Delphi3および4にて開発をおこなっています。OSはWindows95/98/NT4.0にて動作確認をおこなっていますが、標準的なファイル入出力の機能以外は使用していないため、Delphi2.0以上であればほぼ間違えなく動作可能だと思います。

作成されたPDFファイルに関しては、Windows95/98/NT上のAcrobatReader3.0およびLinux(RedHat6.0英語版)上のAcrobatReader4.0+日本語フォントにて表示を確認しています。日本語を使用していないファイルについてはLinux(英語版RedHat6.0)上のGhostViewでも表示可能です。その他の環境(MacやSolaris等)での確認ができる方がいらっしゃいましたら、ぜひ教えてください。

基本的な使用方法

PDFMakerは現時点ではベータ版ですので、本格的なマニュアルはまだ用意していません。使用方法については、ここから下に書かれている簡単なプログラム例と、サンプルプログラムを参考にしてください。

簡単なプログラム

1)uses節に PDFMaker.pasおよびPMFonts.pasを加えます。

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, PDFMaker, PMFonts;
```

または

```
implementation
uses PDFMaker, PMFonts;
{$R *.DFM}
```

2) TPDFMakerのインスタンスを作成します。

```
var
  FPDFMaker: TPDFMaker;
```

と定義しておくか、Form内のprivateセクションに

```
private
  FPDFMaker: TPDFMaker;
public
```

と入れておき、適当な場所で、

```
  FPDFMaker := TPDFMaker.Create;
```

のようにしてPDFMakerのインスタンスを作成します。

3) BeginDocを呼び出し印刷を開始します。パラメタには出力先のストリームを入れます。
通常は出力先はファイルなので、

```
  FPDFMaker.BeginDoc(TFileStream.Create('ファイル名', fmCreate));
```

のようになります。

4) TPDFMaker.Canvasの描画メソッドを呼び出すことで、描画を行います。

```
  FPDFMaker.Canvas.TextOut(100, 600, 'Hello World');
```

5) 必要であれば、PDFMaker.Canvasのプロパティを設定します。

```
  FPDFMaker.Canvas.Font := pfMincyo;
  FPDFMaker.Canvas.FontSize := 15;
  FPDFMaker.Canvas.CharSpace := 1;
  FPDFMaker.Canvas.TextOut(100, 400, 'こんにちは 世界');
```

6) NewPageメソッドで、改ページします。

```
  FPDFMaker.NewPage;
```

7) 線を引く場合は、LineToメソッドを使用します。

```
  FPDFMaker.Canvas.LineTo(50, 250, 100, 300);
```

8) 終了する時はEndDocメソッドを呼び出します。
通常はこの時に出力ストリームを閉じるので、パラメタにはTrueを渡します。

```
  FPDFMaker.EndDoc(true);
```

9) 全ての出力が終了し、必要がなくなったらPDFMakerのインスタンスを解放します

```
FPDFMaker.Free;
```

詳しくはサンプルプログラムの"HelloWorld.dpr"を見てください。

メソッド・プロパティの説明

PDFMakerはさまざまな種類のクラスを使用していますが、これらのほとんどはPDFファイルを出力するために内部的に使用しているものであり、実際にプログラミングで使用するのはTPDFMakerとTPDFMakerのCanvasの役割を持つTPDFContentsの２つだけです。ここではこの2つのクラスのメソッドとプロパティについて説明します。

TPDFMakerのプロパティとメソッド

プロパティ

property Canvas: TPDFContents;

仮想キャンバスを示します。このCanvasの描画ルーチンを呼び出すことで、PDFファイルを作成していきます。

property Author: string;

書類情報の"作成者"の項目に表示する文字列を指定します。

property Creator: string;

書類情報の"作成"の項目に表示する文字列を指定します。

property Title: string;

書類情報の"タイトル"の項目に表示する文字列を指定します。

property Subject: string;

書類情報の"サブタイトル"の項目に表示する文字列を指定します。

property PageHeight: integer;

Canvasの縦サイズを指定します。デフォルトは842でこのサイズはA4の縦に相当します。

他のサイズの帳票を作成する場合はこのサイズを変更します。

property PageWidth: integer;

Canvasの横サイズを指定します。デフォルトは596でこのサイズはA4の横に相当します。

メソッド

procedure BeginDoc(AStream: TStream);

BeginDocメソッドはPDFファイルの出力を開始する時に呼び出します。AStreamパラメタには出力先のストリームを指定します。

procedure EndDoc(ACloseStream: boolean);

EndDocメソッドはPDFファイルの出力を終了します。出力先のストリームを閉じる場合はACloseStreamパラメタにtrueを指定します。

procedure NewPage;

NewPageメソッドは、改ページ処理を行います。

TPDFContentsのプロパティとメソッド

プロパティ

property Font: TPDFFontID;

Fontプロパティは、文字列のフォントの種別を指定します。

property FontSize: Single;

FontSizeプロパティは、フォントのサイズを指定します。

property LineWidth: Single;

LineWidthプロパティは、線の太さを指定します。

property LineJoinStyle: TLineJoinStyle;

TLineJoinStyle = (ljMiterJoin, ljRoundJoin, ljBevelJoin);

LineJoinStyleプロパティは線と線との接点の形を指定します。

property LineCapStyle: TLineCapStyle;

TLineCapStyle = (lcButtEnd, lcRoundEnd, lcProjectingSquareEnd);

LineCapStyleプロパティは、線の末端の形を指定します。

property FillColor: TColor;

FillColorプロパティは、領域や文字を塗りつぶす色を指定します。

property StrokeColor: TColor;

StrokeColorプロパティは、線の色を指定します。

property Leading: Single;

Leadingプロパティは、文字列を出力するときのインデントを指定します。

property CharSpace: Single;

CharSpaceプロパティは、文字間の間隔を指定します。

property WordSpace: Single;

WordSpaceプロパティは、単語間の間隔を指定します。

メソッド

procadure TextOut(x, y: Single; Text: string);

TextOutメソッドは、x, yで指定された位置にsで指定された文字列を出力します。

function TextWidth(S: string): Single;

TextWidthメソッドは、現在の設定で文字列Sを出力した時の幅を返します。

function MeasureText(S: string; AWidth: Single): integer;

MeasureTextメソッドは、現在の設定で文字を出力した時にAWidthの幅に収まる文字数を返します。

function ArrangeText(Src: string; var Dst: string; AWidth: Single): integer;

ArrangeTextは、Srcで指定した文字列をAWidthの幅に収まるように自動的に改行コードを挿入してDstで指定した文字列に入れ、戻り値として行数を返します。

procedure LineTo(x1, y1, x2, y2: Single);

LineToメソッドは、(x1, y1)で指定されたポイントから(x2, y2)で指定されたポイントに直線を描きます。

procedure DrawRect(x1, y1, x2, y2: Single; Clip: boolean);

DrawRectメソッドは、(x1, y1, x2, y2)で指定された長方形をStrokeColorプロパティで指定された色とLineWidthプロパティで指定された線の幅で出力します。Clipパラメタにtrueを指定するとその領域をクリッピングします。

procedure FillRect(x1, y1, x2, y2: Single; Clip: boolean);

FillRectメソッドは、(x1, y1, x2, y2)で指定された長方形をFillColorプロパティで指定された色で塗りつぶします。Clipパラメタにtrueを指定するとその領域をクリッピングします。

procedure DrawAndFillRect(x1, y1, x2, y2: Single; Clip: boolean);

DrawAndFillRectメソッドは、(x1, y1, x2, y2)で指定された長方形をtrokeColorプロパティで指定された色とLineWidthプロパティで指定された線の幅で出力するとともにFillColorプロパティで指定された色で塗りつぶします。Clipパラメタにtrueを指定するとその領域をクリッピングします。

procedure CancelClip;

CancelClipメソッドは、DrawRectやFillRectメソッドでClipパラメタにtrueを指定したときのクリッピングを取り消します。

補足事項

座標系

PDFファイルは、ハードやOSに依存しない出力を実現するため、ユーザースペースという独自の座標系に出力を行います。これは左下が(0,0)、右上が（PageWidth,PageHeight）となる座標系のため、DelphiにおけるTCanvasの実装と一部異なります。

標準描画メソッドと低レベルメソッド

PDFMakerには、標準描画メソッドと、頭が"p"で始まる低レベルメソッドがあります。標準描写メソッドは、いくつかの低レベルメソッドを組み合わせて簡単に扱えるようにしたものです。標準描写メソッドのみを使用している場合、フォントや行ピッチ、文字ピッチ等のプロパティを設定するだけで安全に、PDFファイルを作成することが可能ですが、現在のところ数種類の描画メソッドしか用意されていないため、あまり複雑な内容のPDFファイルは作成することができません。これに対し低レベルメソッドは、PDFのオペレータを直接出力するメソッドなので、標準描画メソッドよりはるかに複雑な描画が可能ですが、呼び出し順序やパラメタの設定を誤ると、意図した表示が行われなかったり、エラーが出て表示できなかったりする場合もあります。低レベルメソッドを使用する場合は、Adobe社のサイトから"PDF仕様書"を入手し、オペレータについての知識を勉強することをお勧めします。

未実装の機能と今後実装予定の機能

PDFは非常に多機能なフォーマットであるため、PDFMakerで実装されている機能はそのほんの一部にすぎません。今後実装予定の機能としては、以下のものを考えています。

イメージの描画

PDFファイルには多彩なイメージの表示機能がありますが、このうちJPEG形式の表示に関してはなるべく早い時期に実装を予定しています。

テキストのエンコード

現在、出力されるPDFファイルは全く圧縮が行われていません。テキスト圧縮の機能を使用することで、ファイルサイズを半分近い大きさにすることが可能です。この機能もなるべく早い時期に実装する予定です。

カタログツリー

ページ数の多いマニュアル等では、左側に目次がツリー表示され特定のページにすばやくアクセスできるPDFファイルがありますが、この機能もまだ実装されていません。この機能も将来的には実装する予定です。

今後実装が変更される可能性がある部分

フォント関連

PDFファイルはさまざまな言語・OS上で使用されることを想定したフォーマットであるため、フォントについては非常に複雑な実装になっています。現在のところ、PDFMakerでは頻繁に使用されると思われる８種類のフォントのみ使用可能です。フォント部分の実装については現在試行錯誤中ですので今後実装が変更される可能性もあります。このユニットに独自の改造を加える場合は、その点を頭に入れておいてください。

サンプルプログラムの説明

HelloWorld.dpr

最もシンプルなプログラムです。

FontDemo.dpr

各種フォントと文字列処理のサンプルです。

LineDemo.dpr

各種の線画のサンプルです。

DBList.dpr

データベースからの出力のサンプルです。実行にはDBDEMOSデータベースが必要です。

MakeGraphPaper.dpr

10ドット単位の目盛りがついた方眼紙を出力するサンプルです。レポートのデザインに使用してください。

MakeManual.dpr

このファイルを出力するプログラムです。

最後に

ライセンスについて

PDFMakerは現在まだベータ版の段階ですので、各自の責任のもとで使用してください。また、PDFMakerはGNUライブラリ一般公有使用許諾(LGPL)に基づいて公開しています。つまり、プログラムに組み込む場合に限り有償・無償を問わず自由に使用可能ですが、PDFMaker自体に変更を加える場合はフリーかつオープンソースである場合のみ公開することが可能です。