

1 グラフィックス

1.1 グラフィックス

PDF で図形を描く方法は大きく分けて二つあります。いわゆるベクタ・グラフィックスとビットマップ画像です。

TeX ソース中に PDF オペレータを使って直接図形を描くにはスペシャルコマンドを使います。



1.2 ベクタ・グラフィックス

PDF では直線や Cubic Bézier 曲線などでパスを構築して線を描いたり、パスの内部領域を塗りつぶしたりすることができます。

簡単な例を示しましょう。次の例は矢印です。

```
0.4 0.4 0.6 RG 0.6 0.6 0.8 rg
2 w
0 10 m
20 10 1 20 0 1 40 15 1
20 30 1 20 20 1 0 20 1
b
```

最初の行は色の指定です。RG, rg はそれぞれストロークと塗りつぶしに使う色を RGB で指定します。^{*1}オペレータ w は線の太さを指定します。m は現在地点を移動するオペレータで引数には x,y 座標の値を与えます。PDF では基本単位としてポイント (72 ポイントで 1 インチ) を使います。オペレータ l は構築中のパスに直線を追加します。引数は二つ x,y で、現在地点から (x,y) までの直線をパスに追加して現在地点を (x,y) に移します。l などのパス構築オペレータはページ上に何も描きません。s, S, b, B や f などのペイント・オペレータを使うとはじめてページ上に何かが描画されます。例にある b オペレータはパスを閉じたうえでパスの内部を塗り

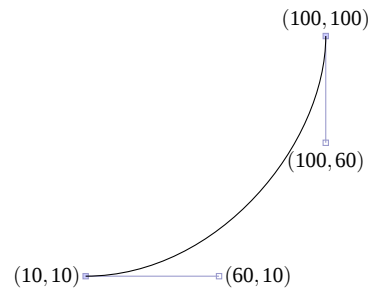
^{*1} PostScript と同様にオペレータはその引数の後にきます。

つぶし、さらにパスに沿って線を引きます。

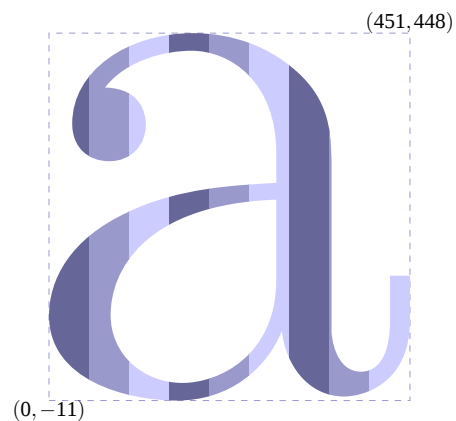
曲線には Bézier 曲線を使います。

```
10 10 m 60 10 100 60 100 100 c S
```

これを制御点とともに示すと以下のようになります。



また、破線にしたりクリッピングを行ったりすることもできます。



この例では 'a' は文字としてではなく、Bézier 曲線と直線を使って描かれています。^{*2}

TeX で使う場合は picture 環境を使うとラベルを張ったりするときに便利です。



^{*2} Computer Modern フォントの a です。アウトラインは Bluesky Research 版の PostScript Type 1 フォントから抽出しました。

1.3 ビットマップ画像

PDF でビットマップ画像をページに表示させる方法は二つあります。

1. インライン画像
2. *Image XObject* として画像を取り込んでから **Do** オペレータによって参照する方法

XObject とは文書中のどの場所からも何度でも参照することができる外部オブジェクト (*external Object*) です。*Image XObject* を使う方法は例えばアイコンなど複数のページに渡って何度も使用されるような画像に適しています。ここでは *XObject* の詳しい解説は行ないません。インライン画像はページの内容を記述したデータ (*Content Streams*) に画像データが直接埋め込まれたものを指します。



さて、インライン画像を描いてみましょう。

```
32 0 0 32 0 0 cm
BI
/Width 8
/Height 8
/ColorSpace /DeviceCMYK
/BitsPerComponent 1
/Filter [/ASCIIHexDecode]
ID
8040201008040201
1080402001080402
2010804002010804
4020108004020108>
EI
```

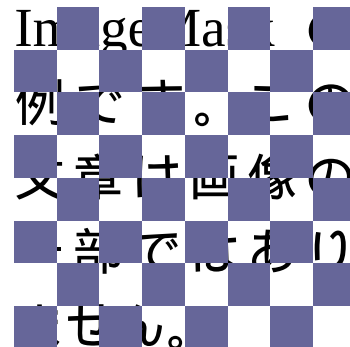
インライン画像は **BI** と **EI** で挟みます。**BI** と **ID** の間には画像データに関する情報が記述してあります。画像データは **ID** と **EI** に挟まれた部分にあります。

この例では *Color Space* に **DeviceCMYK** が指定されています。^{*3}これは CMYK カラーをもちいるという意味です。**BitsPerComponent** が 1 になっているのでそれぞれの色の要素 (ここではシアン、マゼンタ、イエロー、ブラック) の強さ (CMYK では濃度といった方が適切でしょう) を表すのに 1 ビットずつ必要とします。つまり、その色を最大限に使うか全く使わないかのどちらかで、ひとつのピクセルの色を表すのに 4 ビットを必要とします。

Filter には画像データをデコードするのに必要なフィルタの種類を (フィルタをかけるべき順番に) 列挙します。この例では画像データは 16 進表記にしてある^{*4}ので **ASCIIHexDecode** が指定してあります。



マスクイメージもつくれます。



本格的な透過色のサポートは PDF 1.4 からです。また、PDF 1.4 では多数のブレンド・モードがサポートされています。^{*5}

^{*3} PDF 利用可能な *Color Space* (色空間) についてはこの後で述べます。

^{*4} 最後の ‘>’ は **ASCIIHexDecode** フィルタにデータの終りを示すマークです。

^{*5} “*Portable Document Format: Changes from Version 1.3 to 1.4*” (Technical Note #5409) および “*Transparency in PDF*” (Technical Note #5407) を参照。

1.4 カラー

PDF はグレースケール、RGB、CMYK など通常の入出力デバイス (装置) で利用されているもののほかにもいくつかの色空間をサポートしています。PDF では文書の体裁だけではなく、『色』もさまざまな環境下でできる限り正確に再現できるように配慮されています。

PDF 1.3 では以下の色空間が利用可能です。

DEVICE

グレースケールと RGB、CMYK で、**DeviceGray**、**DeviceRGB**、**DeviceCMYK** があります。これらはデバイス依存だといわれています。ことなる環境下でも同じように再現されるとは限りません。

CIE-BASED

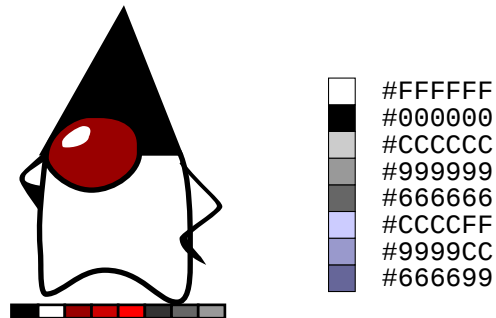
CalGray、**CalRGB**、**Lab**、**ICCBased** があります。CIE 1976 $L^*a^*b^*$ 色空間や ICC プロファイルに基づく色空間などでデバイス非依存となるように配慮されたものです。色彩工学の研究を通して国際照明委員会 CIE が定めた規格に基づいた色空間ということだそうです。

SPECIAL

色空間といってよいのか分かりませんが、**Indexed**、**Pattern**、**Separation**、**DeviceN** があります。インデックスカラーやパターンなどが含まれます。



例えばインデックスカラーを使ってみましょう。これはカラーパレットを想像されると良いでしょう。この例では RGB 色空間上で選んだいくつかの色に番号がつけられ、この番号を通して色が指定されます。



色空間の切替えは CS および cs オペレータで行ないます。sc, SC, scn, SCN で色の切替えを行ないます。ただし、*DEVICE* に分類されるものでは G, g, RG, rg, k, K という簡略化されたオペレータが用意されていて通常はこれを使います。

例えばこのページのリソース辞書で先ほどの例の右側にあるインデックスカラーに Cs1 という名前が関連づけられているとしましょう。このとき、

```
/Cs1 CS /Cs1 cs 0 SC 1 sc
0 0 10 10 re B
0.4 0.4 0.6 RG 0.6 0.6 0.8 rg
10 0 10 10 re B
```

とすると  が描画されます。左側の箱にはインデックスカラーが使用されており、右側の箱では **DeviceRGB** で全く同じ色が直接指定してあります。

次はパターンです。パターンには *tiling patterns* (PDF 1.2 から) と *shading patterns* (PDF 1.3 から) があります。



これ^{*6}は *shading pattern* の例です。色空間には Lab が使用されています。

^{*6} PDF 1.3 に対応していないビューアでは表示されません。



2 ナビゲーション

2.1 アーティクル

新聞などでひとつの記事が紙面のいくつかの場所に分散していてどのように読みすすむべきかわかりづらいことがないでしょうか。PDF にはこのような場合でも読者が正しい筋道に沿って文章を読みすすめるように誘導するための機能が備わっています。この機能は文書の製作者がアーティクル (article) を定義することで利用できます。

アーティクルとは互いに関連した複数の要素を筋道にそって並べたひとつのかたまりです。アーティクルを構成するそれぞれの要素をビーズ (bead) と呼び、ページ上の矩形領域を指定することで定義されます。ひとつのアーティクルにおける各要素を筋道に沿って並べたときの論理的な流れをアーティクル・スレッド (article thread) と呼びます。通常、ビーズはひとつのパラグラフを囲む箱になるでしょう。パラグラフが話の流れに沿って集まるとアーティクルになります。アーティクルはひとつのセクションと一致するかもしれません。

PDF のナビゲーション機能に対応したビューア

を利用しているのであれば非常に効率良く文書を読みすすむことができます。Acrobat Reader ではビーズを定義する領域にポインタをあわせ、そこでクリックするとその部分が拡大されます。あとはつづけてクリックするだけでアーティクル・スレッドに沿って読みすすむことができます。^{*7}

アーティクルの少しかわった例をあげましょう。このページに右下隅に数字が書かれた 4 つの箱が見えるでしょうか。それぞれのコマがビーズでこれらを 1、2、3、4 の順番に並べてひとつのアーティクルを構成します。



^{*7} また、メニューにある Show Articles という項目を有効にすると PDF 文書中にあるアーティクルの一覧がアーティクルのタイトルとともに表示されます。



2.2 プレゼンテーション

PDF にはプレゼンテーション用の機能もあります。ハイパーリンクなどの機能だけでも簡単なプレゼンテーションを行なうには充分かもしれませんが、PDF 1.1 からそれ以外にもいくつかの機能が追加されました。

各ページごとにそのページが表示される時間を指定することができます。一定の時間が過ぎると自動的に次のページに移動させることができますようになっています。また、ページの移り変わり (Transition) に効果を付け加えることもできます。PDF 1.3 で定義してある効果には次のものがあります。

Split

幕が開くような効果

Blinds

ブラインドを開いたような効果

Box

四角い箱がページを置き換えていく

Wipe

前のページを拭き取るようにして次のページが現れる

Dissolve

フェード・インとフェード・アウトのようなもの

Glitter

Dissolve と Wipe をあわせたようなもの

R

何もせずただ置き換える (デフォルト)

これらの効果は PDF の仕様書でも細かいことまで

定めてあるわけではないので、ビューアによって違って見えるかもしれません。まったくサポートしていないビューアもあります。

PDF には動画や音声を埋め込むこともできるので^{*8}、プレゼンテーションにもある程度は使えるのではないのでしょうか。また、Acrobat Form と JavaScript を使うことでもっといろいろな要素をくわえることができます。^{*9}



^{*8} Acrobat Reader ではプラグインがないと再生できません。

^{*9} “Acrobat Forms JavaScript Object Specification” (Technical Note #5186) を参照。

