GDを使ったグラフ描画

2008年01月31日 03時14分45秒

これは何？

perlで書いたグラフの描画コード。

perlでグラフを書く

GD.pm

有名どころに□□なるものがある。このモジュールを利用すると、昔のBASICに実装されていたグラフィックステートメントの様な雰囲気のメソッド群が利用できるようになる。 ただし、直接画面を描画するわけではなく、仮想画面を準備しそこへ描画。最終的に仮想画面の画像データを必要とする画像形式で取出しする。このあたりはさすがに昨今の描画システムを踏襲しているようだ。

グラフを書くには絵を描く道具はあるものの、これを有用なものにくみ上げるのは大変手間がかかる。グラフ描画コンポーネントが数多く作られ、利用されているのは、こういった手間を省いて、本来の要件に注力出来るようにするためである。

これでどうする？

たとえば、円グラフであれば、表示するデータのスケールによって描画精度を変更し、美しく見えるようにする必要があるし、棒グラフであれば、各データの比率を出して美しく見える大きさ(比率)を算出しなくてはいけない。 こういった事を考えつつ作業を行うのは非常に困難が伴うはずだ。

しかし、自分の目的に沿ったコンポーネントが無い場合はやはり、原始的な部分から組み上げていかなければならない。 とりあえず生成品質はおいて。

ソース

以下は、とりあえずの円グラフを書き出すperlスクリプトである□CGIとして利用する。 値としてP1,P2を入力。その比率を円グラフであらわす。円グラフはPNG形式の画像として出力される。

circle.pl

```
#!/usr/bin/perl

use GD;
use CGI;

my $PI   = 3.141592653;
my $R    = 50;
my $X    = 150;
```

```
 my $Y    = 150;

 my $form = new CGI;
 my $image = new GD::Image(300,300);

   $p1 = $form->url_param("P1");
   $p2 = $form->url_param("P2");

   $white = $image->colorAllocate(255,255,255);
   $black = $image->colorAllocate(  0,  0,  0);
   $blue  = $image->colorAllocate(128,128,255);
   $red   = $image->colorAllocate(255,  0,  0);

   $image->transparent($white);

   $image->filledRectangle( 20,50, 50,55, $blue);
   $image->string(gdMediumBoldFont, 55,45, "P1 " . ($p1)."/".($p1+$p2)
, $black);

   $image->filledRectangle( 20,60, 50,65, $red);
   $image->string(gdMediumBoldFont, 55,55, "P2 " . ($p2)."/".($p1+$p2)
, $black);

   if ($p1+$p2 > 0) {
     $p1_percent    = ( $p1 / ( $p1 + $p2 ) );
     $p2_percent    = ( $p2 / ( $p1 + $p2 ) );
     $p1_percentDo  = ( $p1 / ( $p1 + $p2 ) ) * 360;
     $p2_percentDo  = ( $p2 / ( $p1 + $p2 ) ) * 360;
     $p1_percentR   = ( $p1 / ( $p1 + $p2 ) ) * 2 * $PI;
     $p2_percentR   = ( $p2 / ( $p1 + $p2 ) ) * 2 * $PI;

     ## p1
     $image->arc( $X,$Y, 2*$R,2*$R, 0
,$p1_percentDo, $black);
     $image->line( $X,$Y, $X+int(cos(0) * $R)
,$Y+int(sin(0) * $R),  $black);
     $image->line( $X,$Y, $X+int(cos($p1_percentR) * $R)
,$Y+int(sin($p1_percentR) * $R), $black);
     $image->fill( $X+int(cos($p1_percentR/2) * ($R/2))
,$Y+int(sin($p1_percentR/2) * ($R/2)), $blue);

     ## p2
     $image->arc( $X,$Y, 2*$R,2*$R, $p1_percentDo
,$p1_percentDo+$p2_percentDo, $black);
     $image->line( $X,$Y, $X+int(cos($p1_percentR) * $R)
,$Y+int(sin($p1_percentR) * $R), $black);
     $image->line( $X,$Y, $X+int(cos($p1_percentR+$p2_percentR) * $R)
,$Y+int(sin($p1_percentR+$p2_percentR) * $R), $black);
     $image->fill( $X+int(cos($p1_percentR+($p2_percentR/2)) * ($R/2))
,$Y+int(sin($p1_percentR+($p2_percentR/2)) * ($R/2)), $red);
   }
```

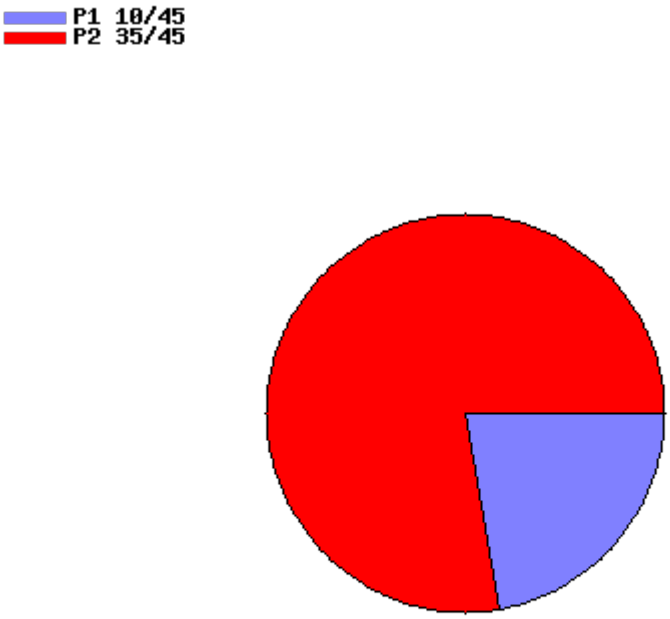
```
  else {
    $image->arc( $X,$Y, 2*$R,2*$R, 0,360, $black);
  }

print("Content-type: image/png\n\n");
print $image->png;
```

これを “circle.pl” とでも名前をつけて保存し、

```
http://hogehoge.co.jp/circle.pl?P1=10&P2=35
```

のように呼び出す。



実行してみればわかるが、ジャギーは目立つし、極端な比率の場合の表示もプア。 描画開始位置もおかしい。 こういった細かい部分にどれだけ手が入っているかで、グラフ描画コンポーネントの価値が決まる。

Perl, GD, 技術資料