比較対照 METAPOST & Asymptote 〜 第2回

最初に業務連絡。

この連載では，拙作の AsyTeX を使って Asymptote に絵を描かせていますが，AsyTeX を使わなくても \sendASY{ } の中味を Asymptote ソースとして Asymptote に渡せば同じ図が描けます。ただし，縦・横の単位長は ASYpic 環境の引数から転送していますので，

\begin{ASYpic}*<1cm> なら real w=1cm; real h=1cm; を，

\begin{ASYpic}*<1cm,2cm> なら real w=1cm; real h=2cm; を

\begin{ASYpic}<1cm,2cm> のように * 無しなら real w=1cm; real h=2cm; unitlength(w,h); を

ソースのはじめに付加してください[1]。

それから，前回 unitsize と size をまとめて「単位長を変える」といいましたが，もう少し詳しく書くと，unitsize がもろに単位長を指定された値に変えるのに対し，size は「図全体」が指定されたサイズに収まるように単位長を変えます。「図全体」のサイズはユーザーが指定することもできますが，指定のない場合は Asymptote に描かせる図の最大サイズ[2]を用います。

以上，業務連絡終わり。

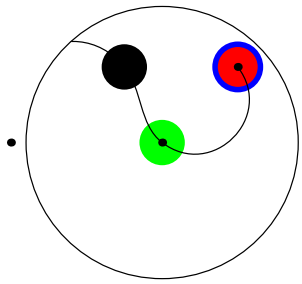
さて，今回は fill や clip といった，塗りつぶしに関する命令を見てみましょう。

1 基本的な使い方

Asymptote における fill や clip の基本的な使い方は，(引数を示す ( ) を省略できないことを除いて) METAPOST と同じです。たとえば，〈巡回パス〉p の内部を塗りつぶしたければ fill(p); とし，p の外部を消去したければ clip(p); とします。

fill の色指定は draw と同様，引数に **pen** 値で与えます[3]。塗りつぶし + 枠線描画の filldraw は，METAPOST と違い[4]，塗りつぶしと枠線にそれぞれ別の色を指定できます。ちょっと例を挙げてみましょう。

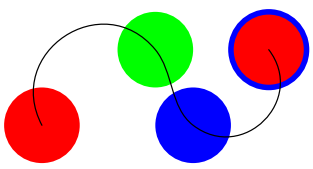
```
\begin{ASYpic}*<1cm>(0,0)(4,-3)
  \sendASY{pair[] z = new pair[4];
    z[0]=(0,0); z[1]=(1.5,1); z[2]=(2,0); z[3]=(3,1);
    path p=z[0]..z[1]..z[2]..z[3]; path q=scale(0.3)*unitcircle;
    path r=shift(z[2])*scale(1.8)*unitcircle;
    fill(shift(z[1])*q); fill(shift(z[2])*q,green);
    filldraw(shift(z[3])*q,red,blue+2pt);
    draw(p);
    clip(r); draw(r);
    for(int i=0; i<4; ++i){dot(z[i],linewidth(3pt));}
  }
\end{ASYpic}
```



filldraw の 2 つの **pen** 値引数は前の方が塗りつぶし，後の方が枠線です。1 つしか指定しなかった場合，塗りつぶしのみ指定されたと見なされ，枠線の方はデフォルト値 (currentpen) が使われます。

clip の作用が及ぶのは，METAPOST と同じく clip が指定されるまでに描かれた図形だけです。上の例ではわかりやすいように clip に指定した枠線と，波形の曲線を構成する点を clip の後に描いていますが，これらは clip で消去されていないことに注意してください。

ちなみに，Asymptote の clip には，「曲線」以外を消去という使い方もできます。言葉だけではちょっとわかりにくいかも知れませんので，次の例を見て下さい。右図が clip 前の図です。



[1] この w と h を使った位置指定法は，METAPOST ではなく METAFONT の仕様に合わせて MEPOTEX マクロを作ったことに由来します。

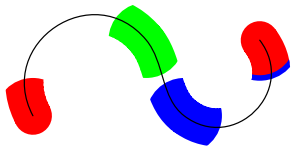
[2] 生成する EPS ファイルに bounding box を書き込む必要上，Asymptote は図の最大サイズを把握しています。

[3] 前回も書きましたが METAPOST は fill の「後置文」で指定しますので，指定方法が違ってます。

[4] METAPOST は，フォント作成用の METAFONT がベースになっていますので，filldraw で fill と draw の色は同じ — つまり枠線の太さの分，塗りつぶし領域が拡がるだけ — でした。

これを，曲線の部分 (太さ 5mm) だけ残して消去してみます。

```
\begin{ASYpic}<1cm>(0,0)(4,-3)
  \sendASY{pair[] z = new pair[4];
    z[0]=(0,0); z[1]=(1.5,1); z[2]=(2,0); z[3]=(3,1);
    path p=z[0]..z[1]..z[2]..z[3]; path q=scale(0.5)*unitcircle;
    fill(shift(z[0])*q,red);  fill(shift(z[1])*q,green);
    fill(shift(z[2])*q,blue); filldraw(shift(z[3])*q,red,blue+2pt);
    draw(p); clip(p,true,linewidth(5mm));
  }
\end{ASYpic}
```



clip の **path** 値引数と **pen** 値引数の間に **bool** 値の true[5] を入れることに注意して下さい。この場合，曲線は〈巡回パス〉である必要はありません。

2　固定サイズの図形

前回，Asymptote では size や unitsize などを使うと，1mm のように具体的な単位を指定したところまで拡大・縮小されてしまうということを説明しましたが，これは場合によっては不都合を生じる可能性があります。では，図を構成する特定の部品だけサイズを変更したくない場合，どうすればいいのでしょうか。

Asymptote では，一旦 (currentpicture でない) 別の **picture** 値変数を用意し，そこに図を描き込んでから currentpicture に貼り付けると，貼り付けた図形は，currentpicture のサイズ変更の影響を受けないという仕様になっています[6]。

貼り付けには add という命令を使うのですが，わざわざ新しい **picture** 値変数を用意して貼り付けるのは面倒ですので，「**picture** 値変数の用意，描き込み，貼り付け」の一連の操作を内部でやってくれる命令があります。命令名は draw , fill など，通常の描画命令と同じです。ただ，これまでと違うのは，1 つめの引数が **pair** 値だということです。

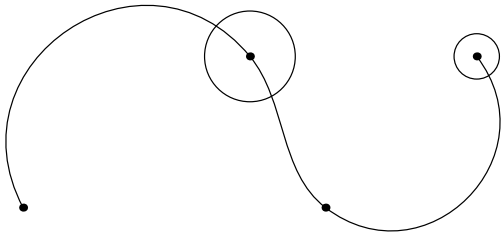
たとえば，

```
draw((2,3),unitcircle);
```

とすれば，まず内部で **picture** 値変数を用意し，unitcircle (単位円：原点が中心で半径 1 の円) を描き，それを，内部で用意した **picture** 値変数の原点が currentpicture の (サイズ変更後の) 座標 (2,3) の点に重なるように貼り付けてくれます。

1 つ例を挙げましょう。

```
\begin{ASYpic}*<1cm>(0,0)(7,-2)
  \sendASY{unitsize(2);
    pair[] z = new pair[4];
    z[0]=(0,0); z[1]=(1.5w,h); z[2]=(2w,0); z[3]=(3w,h);
    path p=z[0]..z[1]..z[2]..z[3];
    path q=scale(0.3w)*unitcircle;
    draw(shift(z[1])*q); draw(z[3],q);
    draw(p);
    for(int i=0; i<4; ++i){dot(z[i],linewidth(3pt));}
  }
\end{ASYpic}
```



最初の unitsize(2); によって currentpicture は単位長が 2bp — デフォルトの単位長の 2 倍，つまり図全体のサイズが 2 倍される — になっています。それに伴い，draw(shift(z[1])*q); で描いた円や draw(p); で描いた波形の曲線は 2 倍に拡大されているのですが，draw(z[3],q); で描かれた円は拡大されていないことに注意して下さい。

と，ここで紙幅が尽きましたので，続きは次回。次回予定は「曲線の内部って？」「グラデーション塗り」。余裕があったら，上でちらっと出た add のこととかも。

[5] stroke=true の意味だそうです。ちなみに省略時は false です。
[6] 「サイズ」という要素が，currentpicture も含めて各 **picture** 値変数毎に設定されているからです。