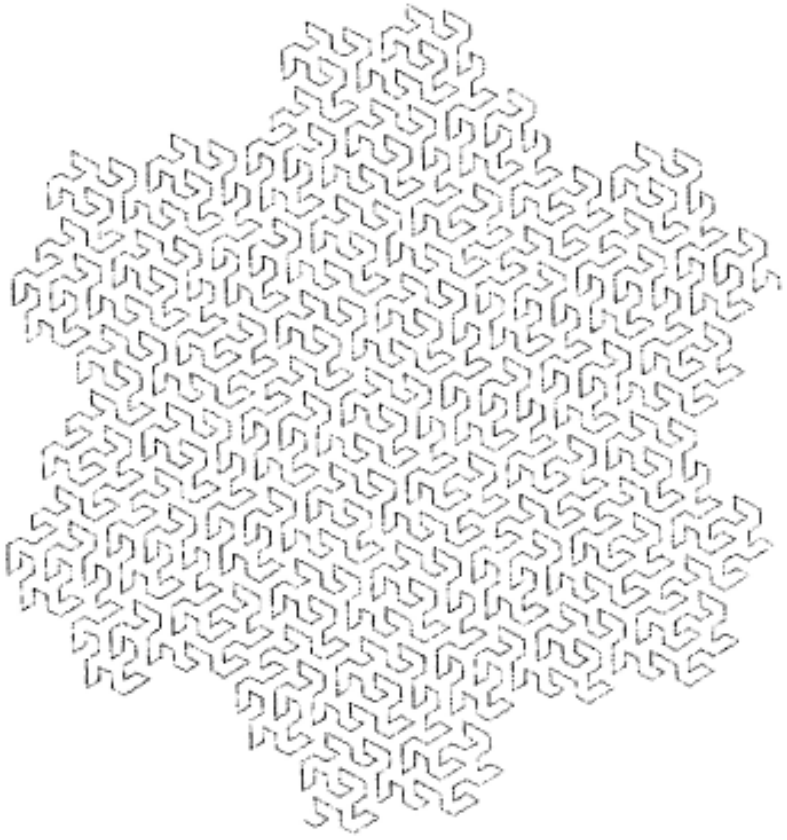
JAPLA 研究会資料 2019/12/7

Ｊによるゴスパー曲線のグラフィックスーその１
ーＪグラフィックス・ツールの整備と蜂の巣模様のタイリングー

西川　利男

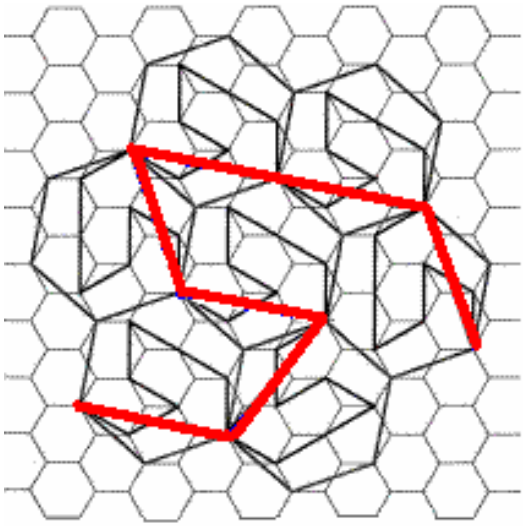
フラクタル図形の一つとして、一次元の線を使って２次元の面を埋めつくすことになるゴスパーの怪物曲線(下図)というのがある。[1],[2]

[1] マーチン・ガードナー、一松信訳「マーチン・ガードナーの数学ゲームⅠ」p.92-95
別冊日経サイエンス１７６(2010).
[2] 福田宏、中村義作「エッシャーの絵から結晶構造へ」p.81-92 海鳴社(2010).



このゴスパー曲線を描くには、まず蜂の巣摸様(Honeycomb)のタイリング(下図)を作る。、つぎに、その正六角形の適当な頂点を結んで、基本のゴスパー曲線(赤)をつくる。これを元に縮小、連結を繰り返して、次第に細かくして(黒)、ゴスパー曲線とする。

このように、まず蜂の巣摸様タイリングを作ることがスタートとなる。



このところ、J によるグラフィックスをいろいろやっているが、数学計算とは違うアプローチに思いがけず手こずっている。

志村氏は極座標や複素数表示を使って、数学的にエレガントに処理していられる。

私はというと、今のところプリミティブに直交座標の上で実際的に、つまり職人的に行わざるを得なかった。

１．J グラフィックス・ツール

これまで、オリンピック・エンブレム、反転グラフィックスなどで、すこしずつ整備してきた J のグラフィックス・ツールの定義動詞はつぎのものである。

・図形の回転　　(degree) **rotate** (picture)　原点(0, 0)でおこなう

・図形の拡大、縮小　(value) **ensize** (picture)　原点(0, 0)でおこなう

(enlarge, reduce をまとめて ensize とした)

・図形の移動　　(x, y) **shift**　(picture)

・色つきで図形を描画

(R, G, B) **colorpolylines** picture　複数の点をつないで、指定した色の線を引く

(R, G, B) **colorpolygon** picture　点をつないで多角形の内部を色で塗りつぶす

すべて、右引数の picture データは、点の値(x, y)のボックス集合で示す.。

なお、これらの J プログラム定義は最後に示してある。

２．蜂の巣模様を一歩ずつ作る

基本となる蜂の巣図形(Honeycomb with a tail、しっぽの付いた正六角形、以 TH と呼ぶ) の点座標をつぎのように決める。

```
RT3 =: %: 3      NB. √3
rt3 =: - RT3     NB. -√3
TH =: (2, 0);(1, RT3);(_1, RT3);(_2, 0);(_1, rt3);(1, rt3);(2, 0);(4, 0)
```

２．１　蜂の巣図形を原点(0,0)、中央に描く

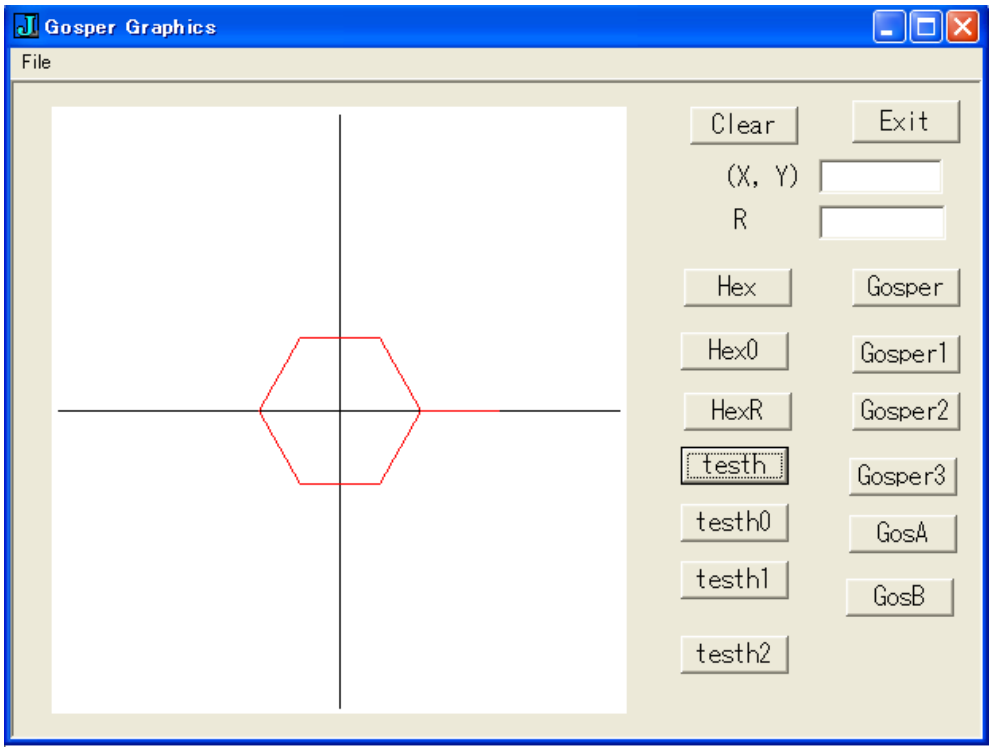
図形を描くには、gl2 グラフィックスでウィンドウ画面を設定して、さきの グラフィックツールの定義動詞を用いて、つぎのようにする。

```
(255 0 0) colorpolylines TH
```

プログラムは以下のようになる。

```
invergraph_testh_button=: 3 : 0
(255 0 0) colorpolylines ; TH
glshow ''
)
```

ボタン testh を押すと、次のように赤い線で描画される。

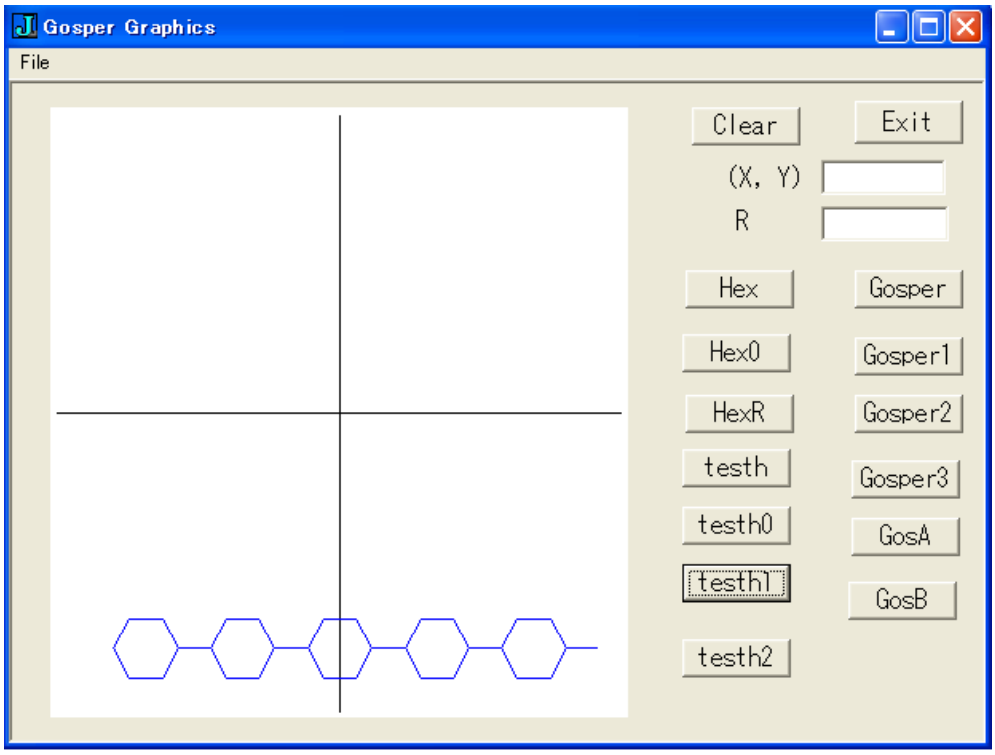


２．２　途中の経過

上の図形を小さくして、横に 5 個をつないで、位置を移動する。

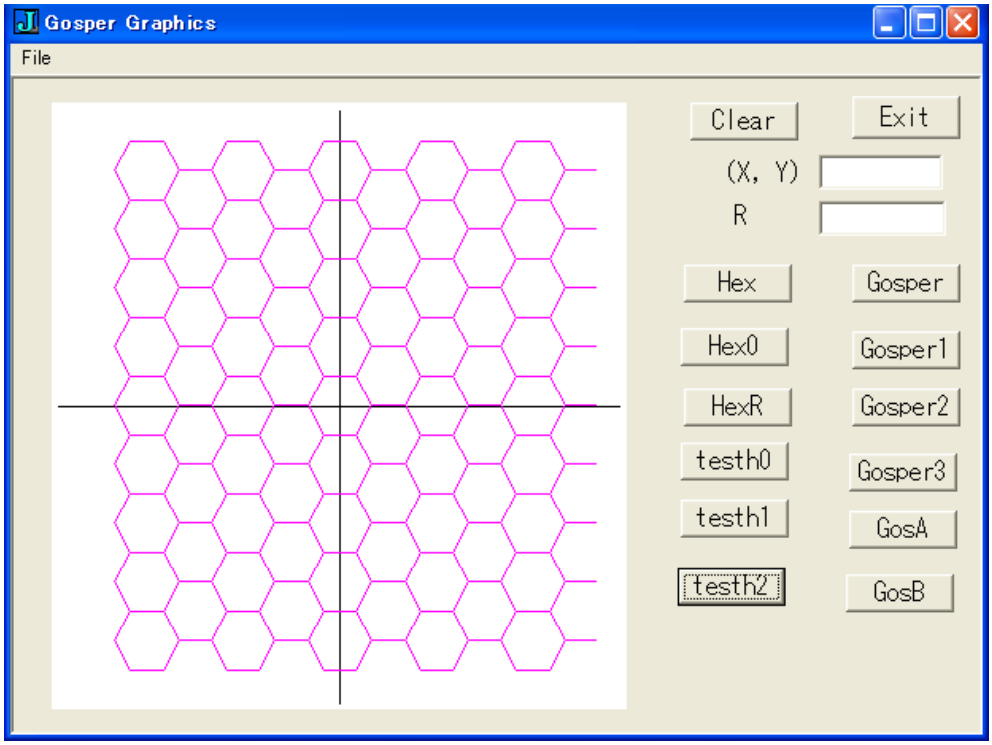
プログラムはツール動詞、ensize, shift, colorpolines を使って次のようになる。

```
invergraph_testh1_button=: 3 : 0
  j =. 0
  while. j < 5
    do.
        thn =. ((j*2.4), 1) shift ; 0.4 ensize TH
       (0 0 255) colorpolylines ; thn
       j =. j + 1
    end.
glshow ''
)
```



３．蜂の巣摸様タイリングの完成図

ボタン test2 を押すと以下のように、蜂の巣摸様のタイリングが描画される。



プログラムは以下のようになる。

```
invergraph_testh2_button=: 3 : 0
i =. 0
while. i < 9
  do.
    j =. 0
    while. j < 5
      do.
        thn =. 0.4 ensize ((_12 + j*6), (_13.8 + i*2*RT3)) shift ; TH
        (255 0 255) colorpolylines ; thn
        j =. j + 1
      end.
  i =. i + 1
  end.
glshow ''
)
```

４．おわりに－－蜂の巣模様は以外にややこしい

今回の発表はここまでとする。ゴスパー曲線の準備だけであったが、はじめは簡単にと思えたこのような処理に以外にてこずってしまった。

グラフィックス処理という観点からは、いろいろな数学の関数をプロットしての図示す

るのとは全く違うことが分かった。まさに図形の作図、アートである。

つまり、点をつないだり、回転、移動、拡大縮小という操作のためのＪの動詞は中々やっかいであった。とくに、回転、拡大縮小の操作のための幾何学計算は原点において行わなくてはならない。そして、必要な位置へ移動する。

操作される図形を表す複数の点をＪの名詞としてどう表すか。私は各点の座標値(x, y)をボックス化し、その集合として図形を扱った。

先にあげた、蜂の巣図形(Honeycomb with a tail、しっぽの付いた正六角形)がそれである。

```
   RT3 =: %: 3     NB. √3
   rt3 =: - RT3    NB. -√3
   TH =: (2, 0);(1, RT3);(_1, RT3);(_2, 0);(_1, rt3);(1, rt3);(2, 0);(4, 0)
   TH
+---+---------+----------+----+----------+----------+---+---+
|2 0|1 1.73205|_1 1.73205|_2 0|_1 _1.73205|1 _1.73205|2 0|4 0|
+---+---------+----------+----+----------+----------+---+---+
```

これらの値に対して、回転、移動、拡大縮小など操作をするための動詞の定義がポイントである。これらをしっかり整備することにした。タイリングのＪプログラムは、Ｊの手法としては異例であるが、繰り返し処理でやらざるをえなかった。

ゴスパー曲線は、正月の楽しみとして、ゆっくりとやることにした。

NB. Gosper Curve ==========

```
NB. Honeycomb Pattern
RT3 =: %: 3
rt3 =: - RT3
NB. HX =: (RT3, 1);(0, RT3);(rt3, 1);(rt3, _1);(0, rt3);(RT3, _1);(RT3, 1)

NB. TH0 =: (2, 0);(1, RT3);(_1, RT3);(_2, 0);(_1, rt3);(1, rt3);(2, 0)   NB. a honeycomb = a hexagon
TH =: (2, 0);(1, RT3);(_1, RT3);(_2, 0);(_1, rt3);(1, rt3);(2, 0);(4, 0) NB. a honeycomb with a tail

invergraph_testh_button=: 3 : 0
(255 0 0) colorpolylines ; TH
glshow ''
)



invergraph_testh0_button=: 3 : 0
  j =. 0
  while. j < 5
    do.
        thn =. ((j*2.4), 1) shift ; 0.4 ensize TH
       (0 0 255) colorpolylines ; thn
       j =. j + 1
    end.
glshow ''
)

invergraph_testh1_button=: 3 : 0    NB. paint 5 honeycombs from left bottom
==================================
  j =. 0
  while. j < 5
    do.
       thn =. 0.4 ensize ((_12 + j*6), _13.8) shift ; TH
       (0 0 255) colorpolylines ; thn
       j =. j + 1
    end.
```

```
glshow ''
)

invergraph_testh2_button=:   3   :   0          NB.   tate_ni   paint   using
invergraph_testh1_button program ====
i =. 0
while. i < 9
  do.
    j =. 0
    while. j < 5
      do.
         thn =. 0.4 ensize ((_12 + j*6), (_13.8 + i*2*RT3)) shift ; TH
         (255 0 255) colorpolylines ; thn
         j =. j + 1
      end.
  i =. i + 1
  end.
glshow ''
)
```

```
NB. Graphics Utilities by TN ==============================
adj0 =:  3 : 0    NB. centered
1 adj0 y.
:
NB. 500 + 100 * x. * y.
(x0, y0) + "(1) > cut2 100 * x. * y.  NB. y-axis down
NB. (500, 400) + "(1) > cut2 100 * x. * y.  NB. y-axis down
)

adj =:  3 : 0     NB. start left down
1 adj y.
:
NB. (500&+)@(100&*)
200 + 40 * x. * y.
)



NB.  cut2 i.10
NB. +---+---+---+---+---+
NB. |0 1|2 3|4 5|6 7|8 9|
NB. +---+---+---+---+---+
cut2 =: 3 : 0
y =. ((-:#y.),2)$y.
<"(1) y
)

rot =: 3 : 0
:
'x y' =. y.
t =. x.
((x*cosd t) - (y*sind t)), ((x*sind t) + (y*cosd t))
)

rotate =: 3 : 0
:
ANG =. x.
```

```
RXY =. cut2 , > y.
RX =. {."(1) > ANG rot L:0 RXY
RY =. {:"(1) > ANG rot L:0 RXY
, |: RX ,: RY
)

NB. eg. (1, 2) shift x, y
shift =: 3 : 0
:
'Xs Ys' =. x.
, (Xs, Ys) +"(1) > cut2 y.
)

colorpolygon =: 3 : 0
:
glrgb x.
glbrush ''
glpen 1 0
NB. glpolygon , sz adj0 y.
glpolygon , sz adj0 y.
)

colorpolylines =: 3 : 0
:
glrgb x.
glbrush ''
glpen 1 0
gllines , sz adj0 y.
)

draw_xy =: 3 : 0
glrgb 0 0 0
glpen 1 0
gllines 10 500 990 500
gllines 500 10 500 990
)
```