

# 反復縮小のフラクタル図形 ( 0 ) (J6 版)

M.Shimura

JCD02773@nifty.ne.jp

2014年3月14日

## 目次

1	面の置き換えによる方法	2
1.1	アフィン変換によるガasket	2
1.2	斉時変換によるガasket	3
1.3	IFS	5
2	辺の置き換えによる方法	8
2.1	コッホ曲線 (1904)	8
2.2	ヒルベルト曲線	10
2.3	シェルピンスキーのスノーフレーク	14
付録 A	J 言語について	15
付録 B	J6 のグラフィックス	15

## はじめに

マイケル・ブラッドリー「数学をひらいた人びと (全5巻)」(青土社 2009)は古代ギリシャから現代まで50人の数学者の伝記と業績の概要を紹介している。著者はマサチューセツ州のカレッジの教授で中高生にも幅広く教えている。50人の中にはベルヌイ、リーマンやワイエルシュトラウスは含まれず、クリミア戦争で活躍したナイチンゲールが統計学者として登場したり、エーダ、チューリングなど初期のコンピューター学者が多く登場する。

この50人の中にガasketで知られるシェルピンスキーが含まれていたのが興味深く読んだ。シェルピンスキーの幅広い業績は落ち着いて考察することにして、フラクタルでよく知られるガasketの他にシェルピンスキーの雪片が紹介されていた。

まずはJ言語等で書かれた各種のフラクタル図形のスクリプトを解読してみよう。何れのスクリプトも教科書を一捻りして簡潔かつ巧妙に作成されている。自分で各種のパラメーターを探すのは骨折りのなのでまずは名画の鑑賞から始めよう。

# 1 面の置き換えによる方法

## シエルピンスキー

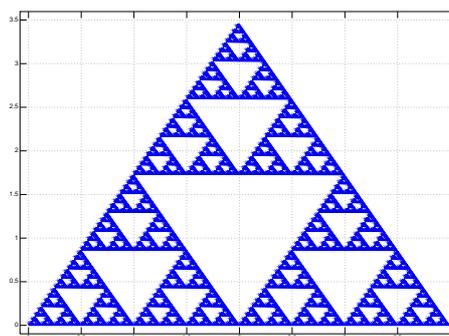
シエルピンスキー (1882-1968) はポーランドの数学者のフロントランナー。第1次大戦ではロシアの捕虜に、第2次大戦ではナチスに自宅を破壊され、投獄されるとなるなどの目にあっている。ポーランドの数学者の半数は第2次大戦で命を亡くしたといわれる中で、戦後はワルシャワ大学でポーランド数学の再興に尽した。

### 1.1 アフィン変換によるガスケット

次はアフィン変換を用いた西川 (2007) のスクリプト。通常アフィン変換は回転、拡大縮小を  $2 \times 2$  のマトリクスで計算し、移動はベクトル値で与える。

```
AMAT=: 0.5 0 ,:0 0.5 NB. matrix
init0=: 0 0;2 0; 1 ,%:3 NB. triangle

sierp=: 3 : 0
DA=. AMAT&mp L:0 y
DB=. 2 0 + L:0 AMAT&mp L:0 y
DC=. (1,%:3) + L:0 AMAT&mp L:0 y
DA,DB,DC
)
```



### 経過と説明

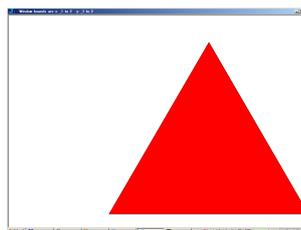
1.  $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$  に縮小するマトリクス (AMAT)

$$\begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$$

2. 最初の三角形の各辺 (init0)。Box で個別に計算する

```
init0
+---+---+-----+
|0 0|2 0|1 1.73205|
+---+---+-----+

_1 _1 2 2 dwin ''
255 0 0 dpoly ,.> init0
```



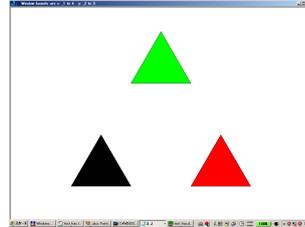
3. ^: 6 外付けのループ (タシット型)

#### 4. 最初の1回

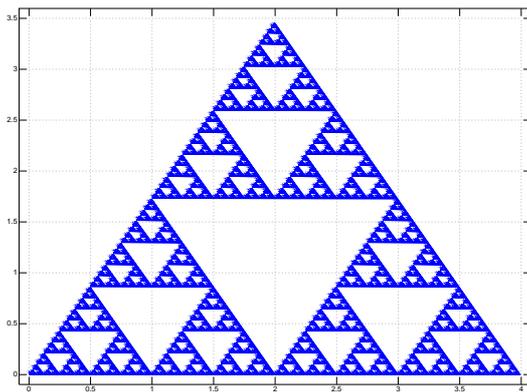
```
,.> sierp ^:1 init0
0      0
1      0
0.5 0.866025

2      0
3      0
2.5 0.866025

1  1.73205
2  1.73205
1.5 2.59808
```



#### 5. 次の反復からは理解が難しいが、8回でフラグメントが繋がりガスケットが完成する



```
plot {@|: > sierp ^:8 init0
pd 'eps d:/sierp3.eps'
```

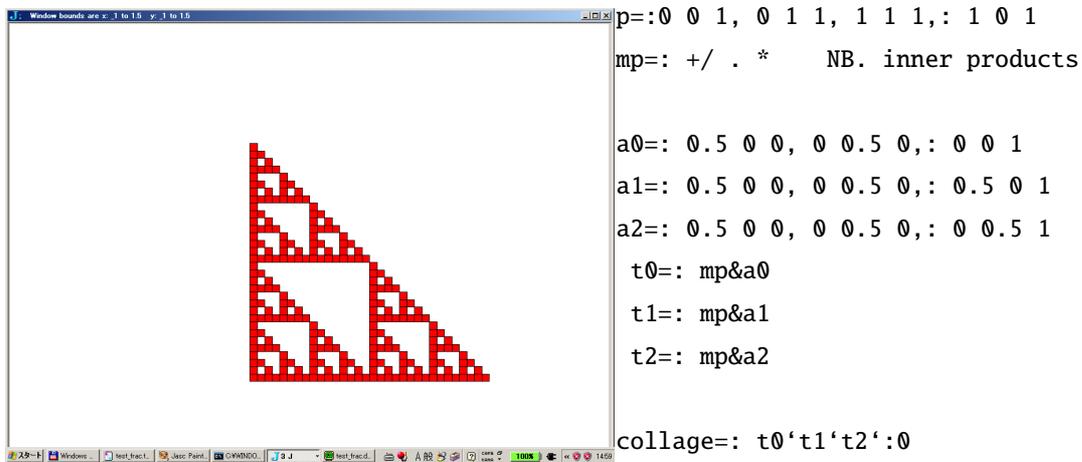
### 1.2 斉時変換によるガスケット

\*1

マトリクスのサイズを1つ上げてアフィン変換の移動部分を取り込む方法。2次元では $3 \times 3$ となる。

```
255 0 0 dpoly L:0 <"2}:"1 collage ^:5 p
```

\*1 Homogenous Coordinate 同時変換、斉時変換の訳がある



```

_2 _2 2 2 dwin ''
255 0 0 dpoly L:0 <"2}:"1 collage ^:5 p/

```

## 経過と解説

1. 最初の種 (p)。右の列を落とすと正方形になる

```

p
0 0 1
0 1 1
1 1 1
1 0 1

```

2. 3の斉時行列 最下行にアフィン変換の場合の別動作の移動部分を取り込む

```

a0;a1;a2
+-----+-----+-----+
|0.5  0 0|0.5  0 0|0.5  0 0|
| 0 0.5 0| 0 0.5 0| 0 0.5 0|
| 0  0 1|0.5  0 1| 0 0.5 1|
+-----+-----+-----+

```

3. collage の作用。(斉時行列の計算のために付加した右端の列は落し、3のピースはボックスに入れた)

```

<"2}:"1 collage p
+-----+-----+-----+
|  0  0|0.5  0|  0 0.5|
|  0 0.5|0.5 0.5|  0  1|
|0.5 0.5| 1 0.5|0.5  1|
|0.5  0| 1  0|0.5 0.5|
+-----+-----+-----+

```

```

% 255 0 0 dpoly L:0 <"2}:"1 collage p

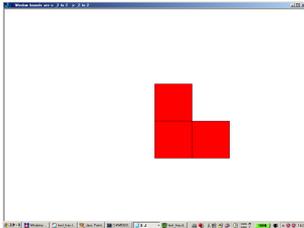
```

#### 4. 反復 1 回の場合

```

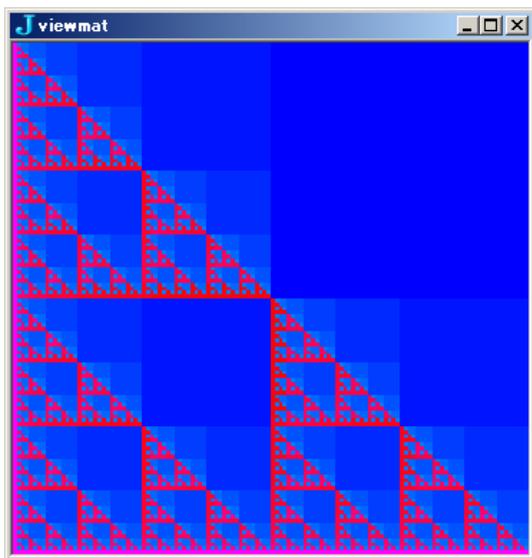
255 0 0 dpoly L:0 <"2}:"1 collage ^:1 p

```



### 1.3 IFS

バースレーの羊歯でお馴染みの反復法 (Iteration Function System) を用いる。Script は C.Reiter による



```
viewmat isier escapet (3 64) xy 128
```

NB. C.Reiter FVJ p.157

```
xy=: |., "0 ~/~(i. % <:)6
```

escapet=: 2 : 0 NB. OK

```
#@((,u@{:})^(<&({ n)@#*.(<&({. n)@:(+/@:|@:{:})^:_))@, :f."1  
)
```

NB. 4 inverse function

```
i0=: +: NB. 2 倍
```

```
i1=: +:@(-&0 0.5) NB. y から 0.5 を引いて xy を 2 倍
```

```
i2=: +:@(-&0.5 0) NB. x から 0.5 を引いて xy を 2 倍
```

```
i3=: _"0 NB. 無限大
```

```
quad=: #.@(>&0.5)
```

```
isier=: i0'i1'i2'i3@.quad
```

1. 種 (xy) と各始点 始点は 64、256,512,1024 などを指定する。

```
<"1 xy 5  
+-----+-----+-----+-----+  
|0 1 |0.25 1 |0.5 1 |0.75 1 |1 1 |  
+-----+-----+-----+-----+  
|0 0.75|0.25 0.75|0.5 0.75|0.75 0.75|1 0.75|  
+-----+-----+-----+-----+  
|0 0.5 |0.25 0.5 |0.5 0.5 |0.75 0.5 |1 0.5 |  
+-----+-----+-----+-----+  
|0 0.25|0.25 0.25|0.5 0.25|0.75 0.25|1 0.25|  
+-----+-----+-----+-----+  
|0 0 |0.25 0 |0.5 0 |0.75 0 |1 0 |  
+-----+-----+-----+-----+
```

2. i0,i1,i2,i3 の動作

```
#.@(>&0.5) L:0 (0 0.5;0.5 1;0.8 0.4;1 1)  
+-----+
```

```
|0|1|2|3|
+--+--+--+
```

3. quad による振り分け quad=: #.@(>&0.5)

2 ポイントを 0.5 より大きい小さいか判別し、その結果を 2 桁の 2 進法と見て、#. で 10 進法の (0,1,2,3) に変換し、適用する関数を決定する。 case 文に相当する。

```
#. @(>&0.5) L:0 (0 0.5;0.5 1)
+--+--+
|0|1|
+--+--+
```

4. isier

```
<"1 isier "0 1 ^:1 xy 5
+-----+-----+-----+-----+-----+
|0 1 |0.5 1 |1 1 |_ _ |_ _ |
+-----+-----+-----+-----+-----+
|0 0.5|0.5 0.5|1 0.5|_ _ |_ _ |
+-----+-----+-----+-----+-----+
|0 1 |0.5 1 |1 1 |0.5 1 |1 1 |
+-----+-----+-----+-----+-----+
|0 0.5|0.5 0.5|1 0.5|0.5 0.5|1 0.5|
+-----+-----+-----+-----+-----+
|0 0 |0.5 0 |1 0 |0.5 0 |1 0 |
+-----+-----+-----+-----+-----+
```

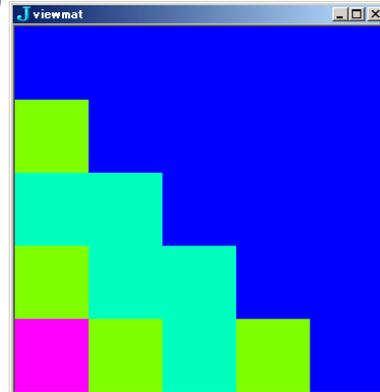
5. 反復収束回数の行列と viewmat 表示

*escapet* の呪文のような関数は発散までの回数をカウントし、その数を *viewmat* で色分け表示する

```

isier escapet 1 5 xy 5
1 1 1 1 1
3 1 1 1 1
2 2 1 1 1
3 2 2 1 1
5 3 2 3 1

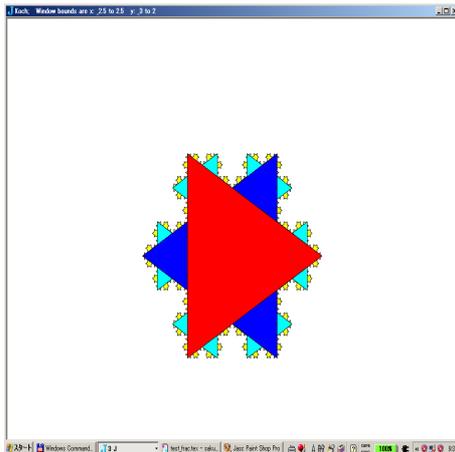
```



## 2 辺の置き換えによる方法

### 2.1 コッホ曲線 (1904)

Koch はスエーデンの数学者。



```

run_frac_koch ''
0 255 255 dpoly refine ^:2 T
0 0 255 dpoly refine T
255 0 0 dpoly T

```

### Script

NB. Koch(1904)

```

T=: |:2 1 o./ 2r3p1*i.3
tri1=: 2r3&*[ + 1r3&*@]
tri2=: 1r3&*[ + 2r3&*@]
mid=: -:@+
nor=: _1 1&*@|.
bulge=: mid + (%:%12)&*@nor@:-
segdiv=: [ , tri1 , bulge ,: tri2
refine=: ,/@[] segdiv"1 (1&|.))

```

```

NB. make
NB. one third point
NB. two thirds point
NB. midpoint
NB. left normal vector
NB. bulge point

```

```

NB. =====2.4=====
run_frac_koch=: 3 : 0
_2.5 _3 2.5 2 dwin 'Koch'
  255 255 0 dpoly refine ^:4 T
)

```

## 経過と説明

1. 最初の三角形(種)は一片を垂直にとる赤の三角形。

```

T
  1      0
_0.5 0.866025
_0.5 _0.866025

```

2.  $\left(\frac{2}{3}\right)\pi \times 0, 1, 2$

```

2r3p1 * i.3
0 2.0944 4.18879

```

3. cos sin の値を求め、転置する。o. は円関数

```

|: 2 1 o./ 2r3p1 * i.3 NB. cos and sin

  1      0
_0.5 0.866025
_0.5 _0.866025

```

```

clean refine T
  1      0
0.5 0.288675
0.5 0.866025
  0  0.57735

_0.5 0.866025
4. _0.5 0.288675
  _1      0
_0.5 _0.288675

_0.5 _0.866025
  0  _0.57735
  0.5 _0.866025
  0.5 _0.288675

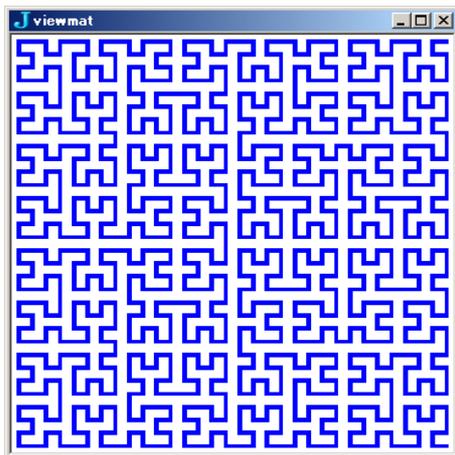
```

## 2.2 ヒルベルト曲線

D. ヒルベルト (1862-1934) は東プロセインの生まれ。ケニッヒスベルクで育ち、同大学で数学を学ぶ。同じ大学にミンコフスキーがいて若い新任教師のフルビッツと3人は生涯の友人で研究を共にした。1895年にゲッチンゲン大学に移る。第一次大戦の当局や台頭したヒットラーに抗った気骨の人でもある。

此処で紹介するアルゴリズムは拡大反復したものをグラフィックス上で小さくなるように見せている。

0/1 のマトリクスを用い、viewmat でカラー表示する。グラフィックス座標は用いない。



### Script

```

viewmat hp ^:5 HP

HP=: 0, 0 1 0 ,: 0      NB. initial curve

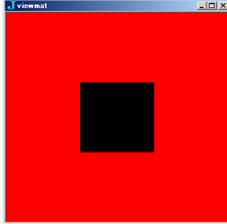
```

```
hp=: 3 : '(|.,]) 1 (0 _2 _2 ,&.> _2 _1 0 + #y) } (.,|:) y'
```

## 経過と説明

1. viewmat は数字のマトリクスをカラーの点 ( 柀 ) で表示する。細密描画も可能
2. 種 HP

```
255 0 0 viewmat HP
```



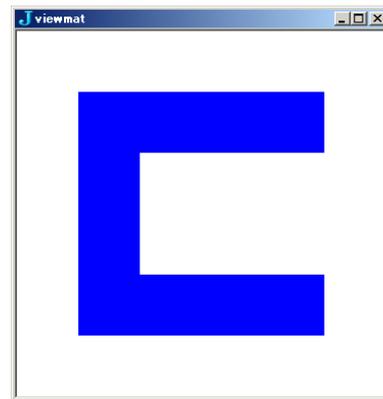
```
HP
0 0 0
0 1 0
0 0 0
```

3. 種 HP 横に 2 個並べる。(., |:) HP      これを書き換えの素

```
0 0 0 0 0 0
0 1 0 0 1 0
0 0 0 0 0 0
```

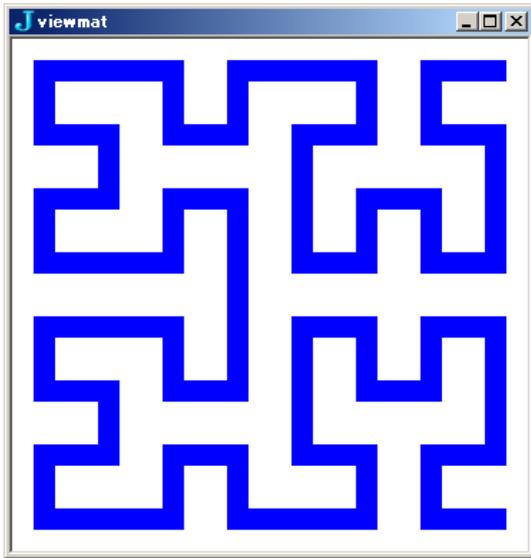
4. hp HP      グラフィックスのピース

```
hp ^:1 HP
0 0 0 0 0 0
0 1 1 1 1 0
0 1 0 0 0 0
0 1 0 0 0 0
0 1 1 1 1 0
0 0 0 0 0 0
```



5. hp ^:2 HP





7. 変換アドレス

```

0 _2 _2 ,&.>_2 _1 0+#HP
+---+---+---+
|0 1|_2 2|_2 3|
+---+---+---+

```

8. 0/1 テーブルの変換。 1 を指定アドレスに入れる。書き換えはアmend } を用いる

```

(|.,]) 1 (0 _2 _2 ,&.> _2 _1 0 + #HP) } (.,|:) HP
0 0 0 0 0 0
0 1 1 1 1 0
0 1 0 0 0 0
0 1 0 0 0 0
0 1 1 1 1 0
0 0 0 0 0 0

```

9. 種

```

a
0 0 0 0 0 0
0 1 0 0 1 0
0 0 0 0 0 0

```

## 10. 変換アドレス

```
0 _2 _2 ,&.> _2 _1 0
+----+----+----+
|0 _2|_2 _1|_2 0|
+----+----+----+
```

```
0 _2 _2 ,&.> _2 _1 0 + #HP NB. first # HP is 3
+----+----+----+
|0 1|_2 2|_2 3|
+----+----+----+
```

## 11. 種はこのように変換される ( 1 をアドレスの箇所に入れ、 0 を変換する )

```
1 (0 _2 _2 ,&.> _2 _1 0 + #HP) } a
0 1 0 0 0 0
0 1 1 1 1 0
0 0 0 0 0 0
```

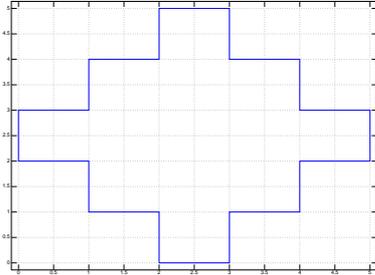
## 12. 変換後の反転と合成

```
(|.,]) 1 (0 _2 _2 ,&.> _2 _1 0 + #HP) } a
0 0 0 0 0 0
0 1 1 1 1 0
0 1 0 0 0 0
0 1 0 0 0 0
0 1 1 1 1 0
0 0 0 0 0 0
```

## 2.3 シェルピンスキーのスノーフレーク

```
plot {@|: SNW
```

```
pd 'eps d:/snow0.eps'
```



```
viewmat SNB
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0 0 0
0 0 0 1 1 1 0 1 1 1 0 0 0 0
0 0 0 1 0 0 0 0 0 1 0 0 0 0
0 1 1 1 0 0 0 0 0 1 1 1 0 0
0 1 0 0 0 0 0 0 0 0 0 0 1 0
0 1 1 1 0 0 0 0 0 1 1 1 0 0
0 0 0 1 0 0 0 0 0 1 0 0 0 0
0 0 0 1 1 1 0 1 1 1 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0 0 0
0 0 0 0 0 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

## 付録 A J 言語について

J 言語はトロントから DL できる。支援のために寄付は募っているが、基本は無償

<http://www.jsoftware.co>.

WIN32/64 MAC64 (Intel) Linux32/64 ipad や Android でも利用できる

最新バージョンは 8.01 WIN8 のような Net 中心の版で発展途上

開発は安定版の 6.02 を推奨します

## 付録 B J6 のグラフィックス

最初に 最初にスクリプトに次の 2 行を書いておく。coinsert はオブジェクトになる。

```
require 'plot gl2 viewmat'  
coinsert 'jgl2'
```

葛飾北斎 次は J のグラフィックスを少し改良した C.Reiter のツールを筆者がピックアップしたもの。J の C.Reiter の addon は必要としない。

1. 次の 2 本の Script をロードする

```
hokusai_tool_improve.ijs  
hokusai_tool_besier.ijs
```

2. キャンバスは描く前にサイズを指定して開いておく。(複数使用可能)

```
_2 _2 2 2 dwin ''  
255 0 0 dpoly L:0 <"2}:"1 collage ^:5 p/
```

3. 描画の始点は左下、終点は右上 サイズは 1000 以上指定可能。(オリジナルは J6 から始点が左上に変更されている。)
4. dwin のデータ形式は縦長の行列。(J のオリジナルは  $x, y, x, y, \dots$ ) のべたのリスト
5. 描画関数は dpoly dline dpixel など (glpolygon など J のオリジナルも使用可能)

## References

C.Reiter[Fractal Visualization and J 3rd.edition] Lulu 2007

Michael J Bradlly 松浦俊輔訳「数学を切りひらいた人びと 4」青土社 2009

郡山彬・原正雄・峰崎俊哉「CG のための線形代数」森北出版 2000