タートルグラフィックスを J で (1)
図形の描き方

SHIMURA Masato
jcd02773@nifty.ne.jp

2011 年 8 月 1 日

目次

1 タートルグラフィックスの表記法と J 1
2 いろいろな図形 2
3 to とリカーシブ 4
4 図形のポイントの説明 7
5 シエルピンスキー曲線を描く 8
6 小紋への道 12

概要

Fasar Jackson の貢献による本格的な J のタートルグラフィックスは J の伝統的なプログラムスタイルといささか異なっている。差異を理解しながら、極座標を中心にタートルグラフィックスの作図のエクササイズを行う。

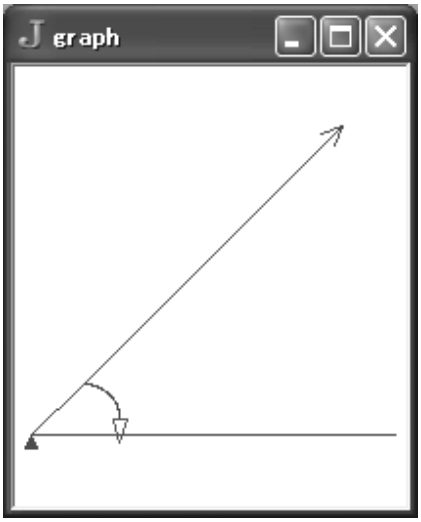
1 タートルグラフィックスの表記法と J

J 上でのタートルグラフィックスの構築は *Frasar Jackson* による。タートルグラフィックスの表記法は J とは相当異なるが *F Jackson* は上手くまとめている。

トップダウン方式 タートルグラフィックスのパスの記法はトップダウンである。

```
show fd 2 rt 90 fd 1 lt 45
```

極座標 タートルグラフィックスの基本は極座標を用いての描画である




```
  show  polar

 NB. ----------------------
polar=: 0 : 0
solida 0.2 point 0
rt 45 fd 1 point 1 fd 5 arrow 0.3
pu goto 0 pdn rt 45 fd 5
pu goto 1  pdn arcr  0.5 90
)
```

相対座標　亀の頭の向かう方向と距離は現在位置を基準とした相対座標である

タートルの方向の初期値は上向き（0) である。 fd 1 rt 90 とすると上に 1 進み、次の fd のために首を右に 90 度曲げて準備する

Explicit **型の方法**　タートルグラフィックスの関数のみを書くときは名詞型 (0 : 0) を用いる。動詞 (3 : 0) は先に座標を計算してしまう

show polar

show polar ” としなくともよい。

文字列?　” で囲った文は展開される

```
  repeat 4;'fd 10 rt 90'
,fd 10 rt 90,fd 10 rt 90,fd 10 rt 90,fd 10 rt 90
```

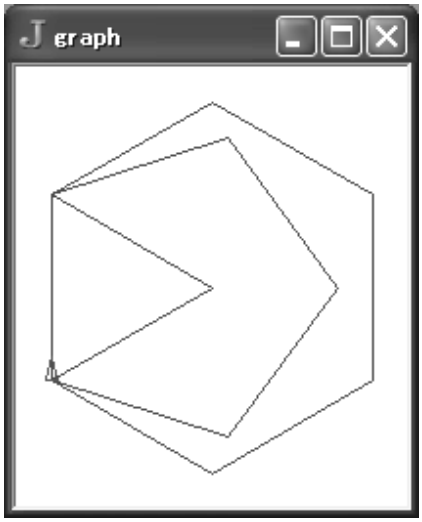
```
  repeat 4 fd 10 rt 90
10 1 2 0 6.12323e_17j_1 1 10 1 2 0 6.12323e_17j_1 1 10 ....
```

do も用意されている（あまり使わない）

```
   do
+--+
|".|
+--+
```

Box　ボックスを開いたり　L : 0 を用いなくとも並列で描画できる

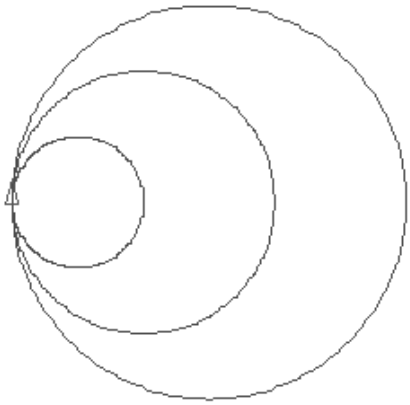
```
tri=: repeat 3 fd 1 rt 120
 pent=: repeat 5 fd 1 rt 72
 hex=: repeat 6 fd 1 rt 60

 show tri;pent;hex
```



pd と pdn　　penup → pu と略記。 pendown → pdn
pd は plotdriver で先に用いられてしまっている

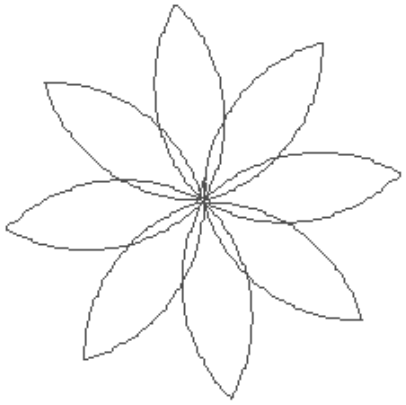
2　いろいろな図形

triangle sq(square) 以外にもいろいろな図形が組み込まれている
チュートリアル（LAB) に入っていない図形もある
circle
show circle each 1 2 3



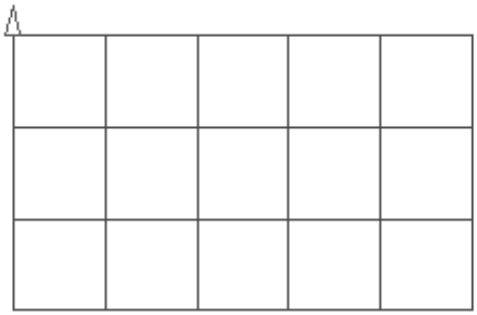
petal
show petal 12 75

flower
show flower 12 75 45

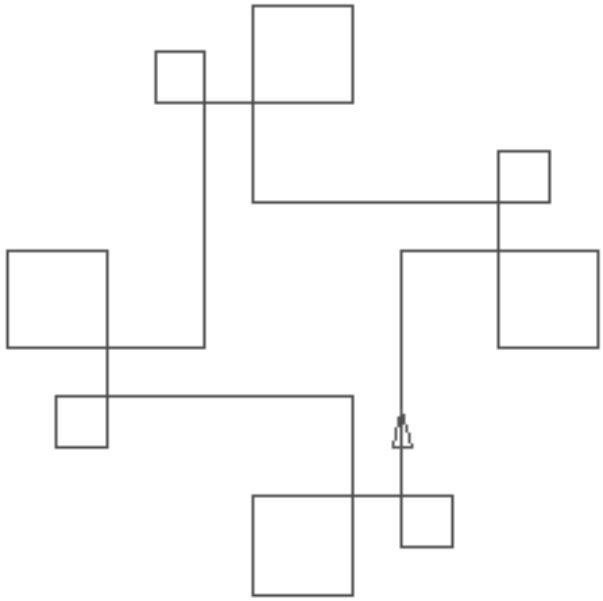


solida
showf solida 100

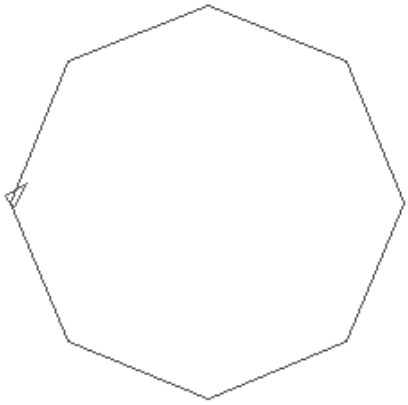
some square
show rowsquares 5 2
show blocksquares 3 5 2



thing
show thing1 30



inscribe
show inscribe 2 45



3　to とリカーシブ

タートルグラフィックスのパスは繰り返しが多いので簡易なリカーシブの定義法が用意されている。
また　to による定義が用意されている

3.1　to による定義

次のように [to] を用いて関数定義が出来る。　引数は n,s(side),h,w,angle,degree などで分かりやすい

to ’sq side’; ’repeat 4 fd side rt 90’
to ’rect h w’;’repeat 2 fd h rt 90 fd w rt 90’

```
to
+-+-+--------------+
|3|:|if. 2=#y do.  |
| | |’nom def’ =. y|
| | |nom todo def  |
| | |else.         |
| | |nom =. >y     |
| | |todo1 nom     |
| | |end.          |
+-+-+--------------+
```

Fackson の各種の定義ファイルを見てみよう。先の図形の定義が多く含まれている。

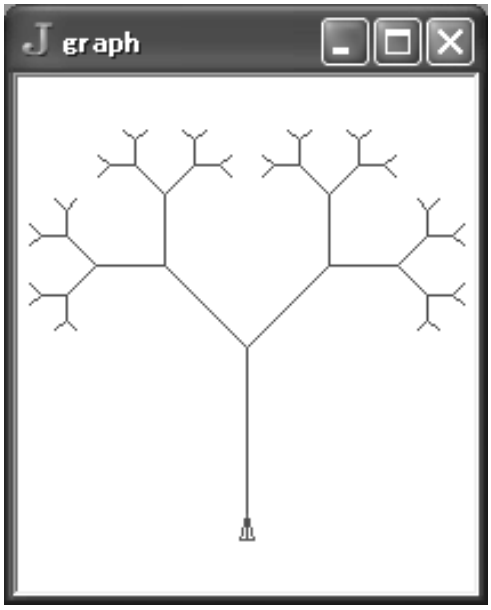
```
NB. -----------poly---------
to ’poly side angle’ ; ’repeat ((360*.angle)%angle) fd side rt angle’
to ’polyn side n’ ; ’repeat n fd side rt 360%n’
to ’polysn s n’; ’repeat n fd (s*1p1%n) rt 360%n’
NB. -----------square------------
to ’sq side’; ’repeat 4 fd side rt 90’
to ’sqc s’;’penup bk s2 lt 90 fd s2 rt 90 pendown sq s pu rt 90 fd s2 lt 90 fd (s2 =.s%2)pendown rt 0’
```

```
to 'rect h w';'repeat 2 fd h rt 90 fd w rt 90'
NB. ------------------------------------
to 'rowsquares s n'; '(repeat n sq s rt 90 fd s lt 90) fd s lt 90 fd n rt 90'
to 'colsquares s n';'repeat n sq  s penup fd 1 pendown '''''
to 'blocksquares s r n';'repeat r rowsquares s ,n'
to 'hundreds r'
(repeat r pendown blocksquares 1 10 10 penup fd 1)penup bk (r*11)
NB. -----circle  arrow flower ----------------------
to 'petal rad deg';'arcr(rad,deg)rt (180-deg)arcr(rad,deg) rt 180-deg'
to 'flower rad deg angle';'repeat ((360*.angle)%angle) petal(rad,deg) rt angle'
to 'circle r'; 'repeat 2 fd d ,(repeat 179 rt 1 fd (2*d)),rt 1  fd (d =.r*sin 1p1%360)'
to 'circlec r'; 'penup seth 0 plusx (-r)pendown circle r pu plusx ( r)pendown rt 0'
to 'arrow s';'rt 155 fd s bk s lt 310 fd s bk s rt 155'
to 'arrow1 s';'rt 155 fd s rt 115 fd (2*s*cos 1p1*65%180)rt 115 fd s lt 25'
to 'arcr r deg';'rt 0.5 fd d,(repeat (deg-1) fd (2*d) rt 1) fd (d =.r*sin 1p1%360) rt 0.5'
to 'arcl r deg';'lt 0.5 fd d,(repeat (deg-1) fd (2*d) lt 1) fd (d =.r*sin 1p1%360) lt 0.5'
NB. ------etc---------------------------------
to 'thing s';'fd s rt 90 fd s rt 90 fd s2 rt 90 fd s2 rt 90 fd s rt 90 fd (s%4) rt 90 fd (s%4) rt 90 fd s2=.s
to 'thing1 s';'repeat 4 thing s'
to 'inscribe r d';'rt (d%2),(repeat (360%d) fd (2*r*cos 0.5p1* 1-d%180 ) rt d),rt (d%2)'
to 'solida s';'fill arrow1 s line'''' '
)
```

3.2 リカーシブ

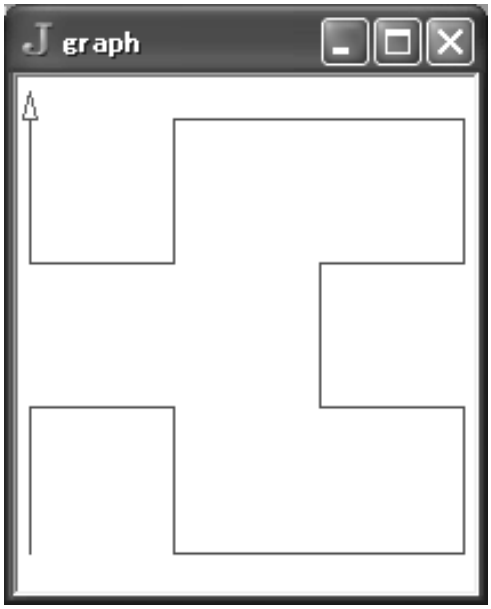
リカーシブ (再帰）は決まると簡潔で美しいが失敗すると虫の巣である。

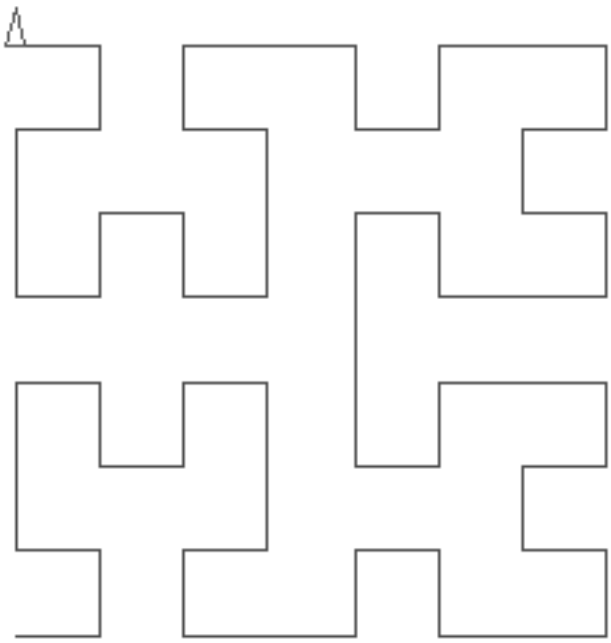
```
to 'branch length level'
if. level = 0 do. i.0 return. end.
len2 =. 0.6* length
fd length,lt 45,(branch len2,level-1), rt 90,( branch len2,level-1),lt 45 bk length
)
```



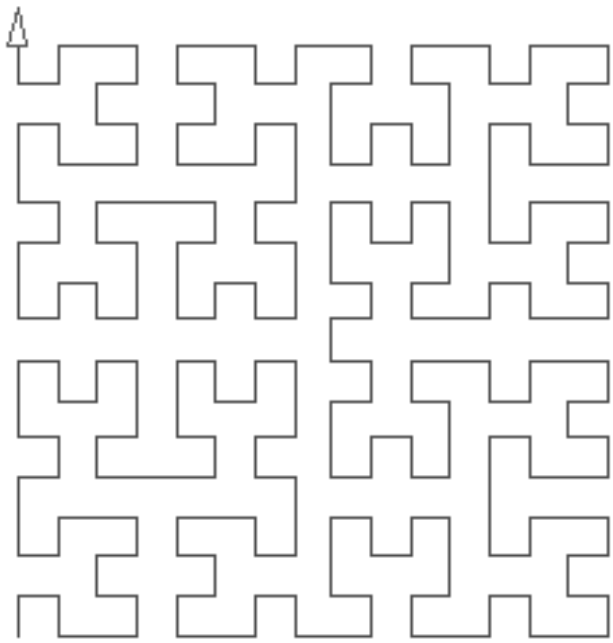
3.3 ヒルベルト曲線

こうしてみるとヒルベルト曲線の構造が分かる





show rhilbert 1 3



show rhilbert 1 3

F.Jackson のヒルベルト曲線のスクリプト。リカーシブに加えて左右同形ではないので対称に作り相互に参照している。[*1]

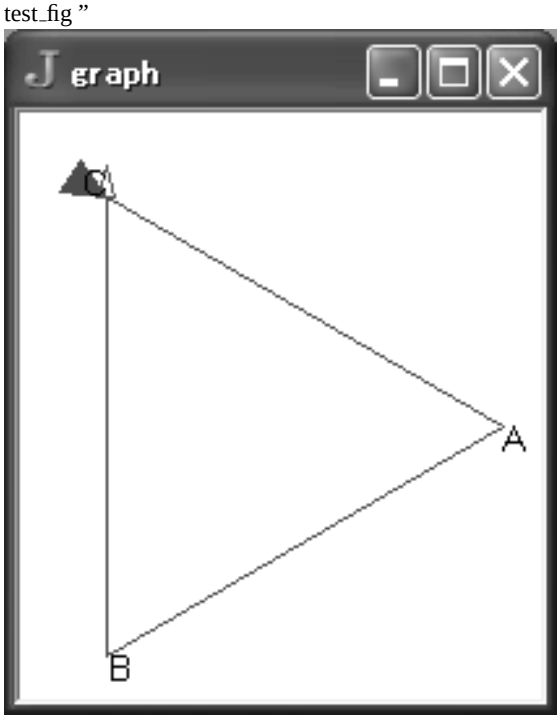
```
to 'lhilbert size level'
if. level = 0 do. i.0 return. end.
lt 90 ,(rhilbert size,level-1), fd size rt 90 ,(lhilbert size,level-1), fd size, (lhilbert size,level-1), rt
)

to 'rhilbert size level'
if. level = 0 do. i.0  return. end.
rt 90 ,(lhilbert size,level-1), fd size lt 90 ,(rhilbert size,level-1), fd size, (rhilbert size,level-1), lt
)
```

4　図形のポイントの説明

- point
  幾何では図形のポイントの説明が必須である。タートルグラフィックスの図にもう一枚透明の図を重ね説明のポイントを記述する。test_fig は名詞型 (0 : 0) では上手く動かなかったので動詞とした。

[*1] 頭が痛くなるのでまねはしない

test_fig ”



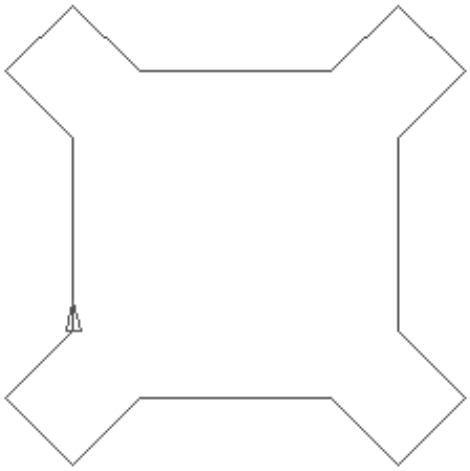

```
 test_point=: 0 : 0
point 0
rt 120 solida 0.1 fd 1 point 1
rt 120 fd 1 point 2
rt 120 fd 1 point 3
)

test_fig=: 3 : 0
show test_point
label 0
label 1 0 _0.05 ;2; 'A'
label 2 0.05 _0.05 ;1; 'B'
label 3 _0.0 0.05 ;2; 'C'
addlabels ''
)
```

- label 0 はこのように書くらしい
  0 _0.05 ; 0 ; A　この 3 個のボックスは a;b;c とすると c はテキスト　b は (0/1/2) で右, 中央, 左 a は NO と微細な位置調整?
- solida 0.1 で始点を明示した
- save は
  save ’filename.emf’ NB. emf 形式
  clip ” NB. クリップボード経由でお好みのフォーマットへ

5　シエルピンスキー曲線を描く

コッホ曲線やヒルベルト曲線、シェルピンスキー曲線をマーチン · ガードナーは解析泣かせの怪物曲線と呼ぶ。
シエルピンスキー曲線は閉じられている
次のような愚直なタートルのプログラムでも基本形が描ける

show repeat 4 ,(fd 2 lt 45 fd 1 rt 90 fd 1 rt 90 fd 1 lt 45)
show lt 45 repeat 4 ,(fd 1 lt 45 fd 2 lt 45 fd 1 rt 90 fd 1 rt 90)



シエルピンスキー曲線は１６辺の閉じた曲線である。
次のように分解する。フラクタルの描画法は用いないで実直にやる

- 4 の部分に分ける
- 次の左のように入力する
  s0=: 1 2 1 1 ,. 1 1 0 0

1	1	*fd* 1 *lt* 45
2	1	*fd* 2 *lt* 45
1	0	*fd* 1 *rt* 90
1	0	*fd* 1 *rt* 90

- ４回繰り返す

  show form_sier s0,s0,s0,s0
- script

```
  form_sier=: 3 : 0
NB. Usage. u s2
ans=. lt 45
for_ctr.  i. # y do.
  select. {: ctr{ y
    case. 0 do. tmp=. fd ,({. ctr{y), rt 90
    case. 1 do. tmp=. fd,({. ctr{y), lt 45
  end.
 ans=. ans,tmp
end.
)
```

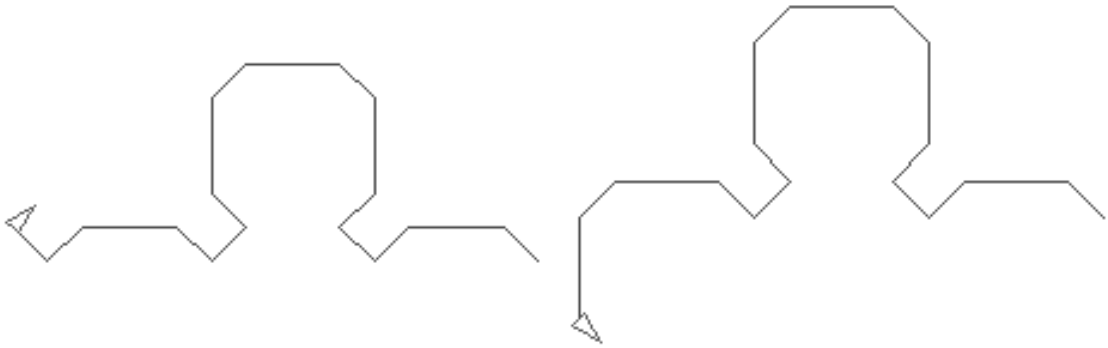
- 基本パーツ
  ２ステップ目が次のようなピースを作る (s1)
  これで右下を始点とできる
  s1=: 1 2 1 2 ,. 1 1 1 1

1	1	*fd* 1 *lt* 45
2	1	*fd* 2 *lt* 45
1	1	*fd* 1 *lt* 45
2	1	*fd* 2 *lt* 45

- s2=s0, s1, s0, s0

  0 を右折れ、1 を左折れとする。シエルピンスキー曲線は基本は次の 2 つの組み合わせである

  NB. basic parts s2=: s0,s1,s0,s0 NB. 0 1 0 0 s3=: s0,s1,s0,s1 NB. 0 1 0 1
  s2=: s0,s1,s0,s0 NB. 0 1 0 0 s3=: s0,s1,s0,s1 NB. 0 1 0 1



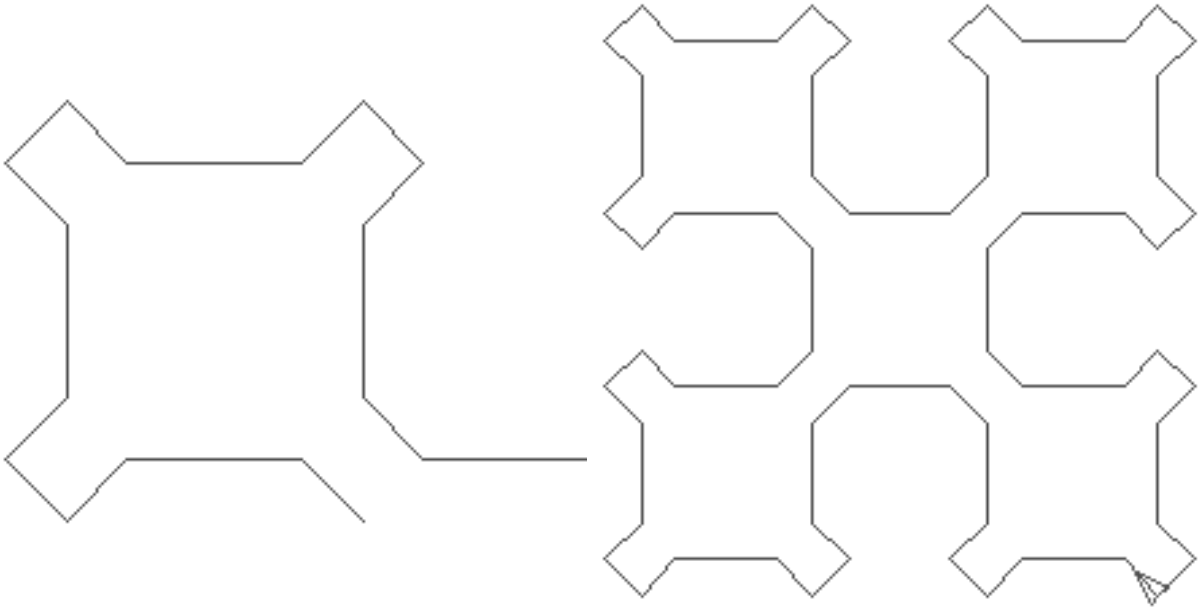
5.0.1 **4個に拡大**

s2 を 4 回繰り返す

s40=: s2,s2,s2,s2 NB. sierpinski 4

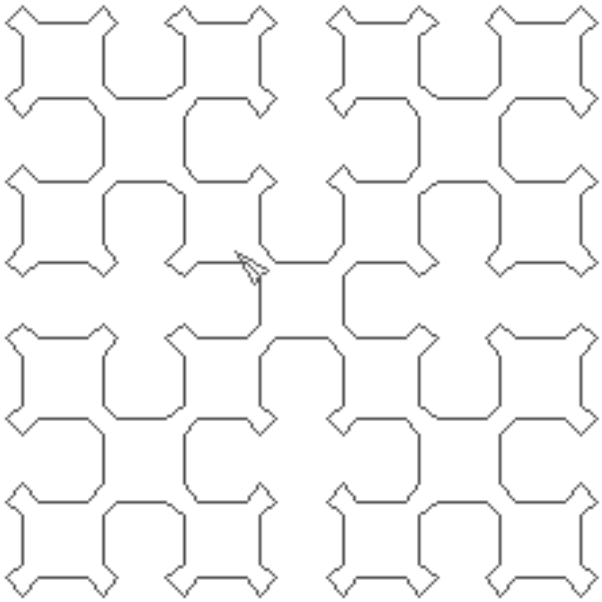
form_sier s40

接合部

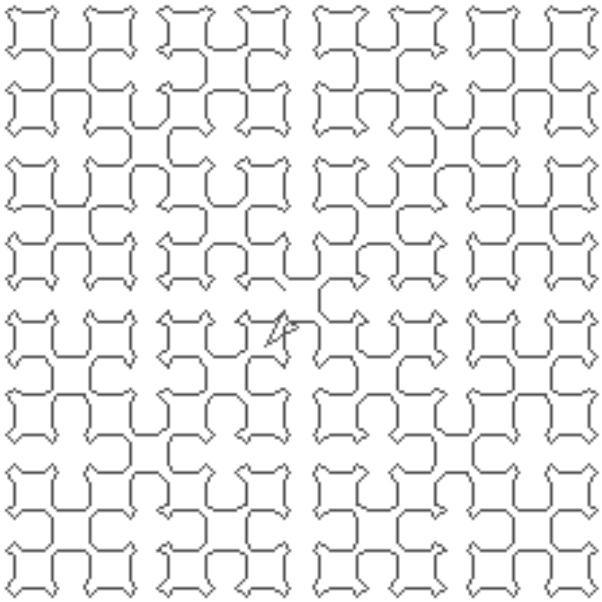


5.0.2 16 個に拡大

- 接合部
  - 2,4 ステップ目を次のようにしたピースを作る (接合用)
    s3=: s0,s1,s0,s1 NB. 0 1 0 1
  - 4 ステップ目は s3 を用いる s41=: s2,s2,s2,s3 NB. 4 for connect
- show 始点を右下にして 4 回繰り返す
  s160=:s41,s41,s41,s41 NB. sierpinski 16 /home is center
  show form_sier s160



5.0.3 64 個に拡大



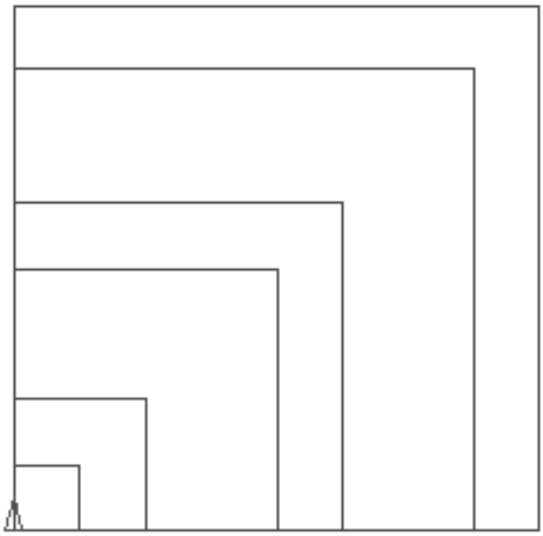
- 始点の変更
  64 個に拡大するには 16 個の始点を右下の端に持ってこなければならない
  s42=: s2,s3,s2,s2
  NB. closed another s161 for 64 test
  s161=: s42,s42,s42,s42 NB. start is bottom right
- 接合用のピース
  16 個の最後は開いた接合用のピースが必要
  s43=: s2,s3,s2,s3
  s162=: s42,s42,s42,s43 NB. for connect
- S32 s32=: s162,s161
- s64
  s64=: s162,s162,s162,s161
- color show color YELLOW fill form_sier s64

color
show color YELLOW fill form_sier s64

6 小紋への道

江戸小紋にたどり着くまえのエクササイズをまとめてみた

- for

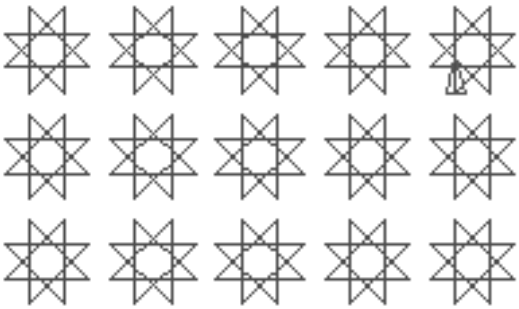


show sq for 1 2 4 5 7 8
each(または&.>) でも同じパスを作る

- start

```
show ({1 2 3 4 5;1 2 3 ) start poly 1 90
```

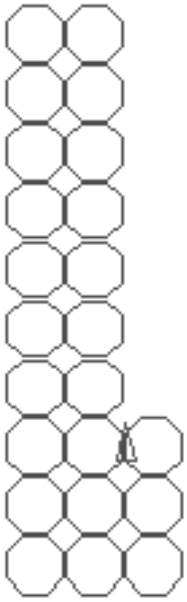
```
   {1 2 3 4 5;1 2 3
+---+---+---+
|1 1|1 2|1 3|
+---+---+---+
|2 1|2 2|2 3|
+---+---+---+
|3 1|3 2|3 3|
+---+---+---+
|4 1|4 2|4 3|
+---+---+---+
|5 1|5 2|5 3|
+---+---+---+
```



- 複素数による座標指定

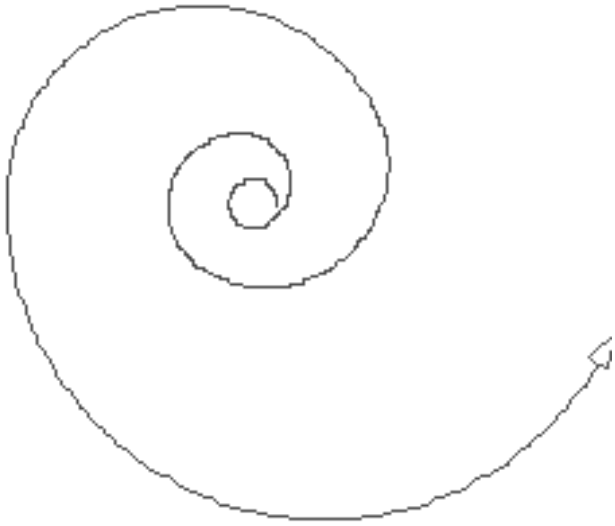
```
show (23{. , a j./ a=. i.10) start poly 0.4
```

show 1j3 2j4 3j5 4j4 5j3 start poly 0.4 45



- フィボナッチ　スパイラル

show arcl ”1 [1 1 2 3 5 8 13 ,.150

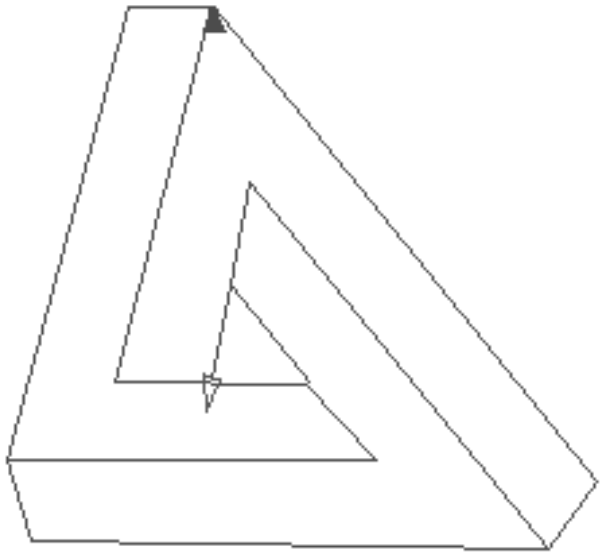


- color
  基本 16 色は RED BLUE GREEN などで OK. 他は RGB で指定する

```
cOL0=: 255 165 132
  show  color cOL0 fill poly 1 60
```

- ペンの移動、始点の移動

```
esjr=: 0 : 0
solida 0.3 lt 90 fd 1
lt 75 fd 5.4 point 0
lt 105  fd 4.3
lt 133  fd 1.2
lt 46  fd 2.2
rt 103 fd 4.5
NB. --------------
pu goto 0 pdn rt 150 fd 1
lt 73 fd 6 point 1
lt 130  fd 5.5
lt 130  fd 1.2 point 2
lt 50  fd 1.5
NB. -------------------
pu goto 1 pdn lt 105 fd 1
lt 75 fd 7.1
NB.---------------
pu goto 2 pdn lt 130 fd 1.1
)
```



エッシャー

References

Frasar Jackson J LAB のタートルグラフィックスのチュートリアル
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(新装版)」日経サイエンス社 2010
J602 J701 はトロントから DL 出来ます
http://www.jsoftware.com
スクリプトは次から DL できます
http://japla.sakura/ne.jp の Workshop AUG 2011