糸掛け曼荼羅と J8 グラフィックス (dwin 版）

SHIMURA Masato

2018 年 6 月 20 日

目次

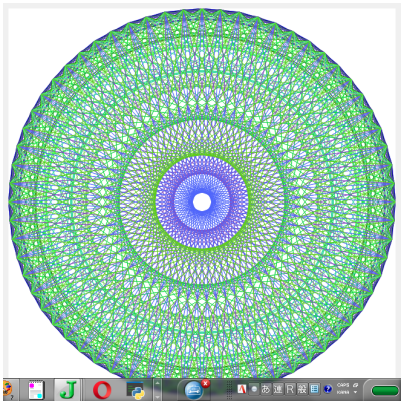
はじめに

ワークショップの帰りに駅前の書店に立ち寄ったら「日経ソフトウエア」が、月遅れ/正価/紐掛けの状態で置かれていたが表紙に Python 特集とあるので求めた。

丁度 Python から J をサーバーにして計算は J に委ねる方法のレポートを作成中であった。Python はいまや JAVA を追い抜いたとの報告もある昇り龍の言語である。

JQT 版は FormEditor をサポートしないと決めたようだ。手作りフォーム用の面白いサンプルグラフィックスを探しているところであった。

日経ソフトウエアの Python の記事中に「糸掛け曼荼羅」のグラフィックスがあった。簡潔なプログラムで綺麗な曼荼羅を描くものであった。糸掛け曼荼羅は初めて見るもので説明だけではアルゴリズムが理解できず、一から調べて J で作成することした。



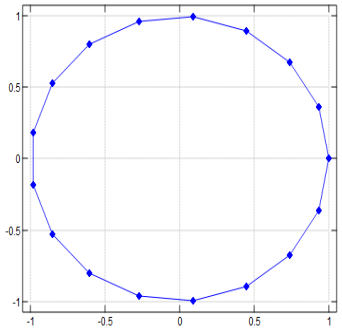
1　ガウスの 17 角形

複素数とオイラーの美しい公式を用いて正多角形を作成するスクリプトはかってガウスの正 17 角形の作図で学習した。

ガウスの 17 角形は次の一行で描けてしまう

$$z = re^{i\theta} = r(cos\theta + isin\theta)$$

```
'line marker' plot {|: (+.r. 2p1* (i.17)%17),1 0
```



1.1　経過と説明

1. 0 から始まる 17 点を打ち出し 17 で割る。
   拡張整数 17x を用いると分数表記になる

   ```
   (i.17)%17x
   ```

```
0 1r17 2r17 3r17 4r17 5r17 6r17 7r17 8r17 9r17
10r17 11r17 12r17 13r17 14r17 15r17 16r17
```

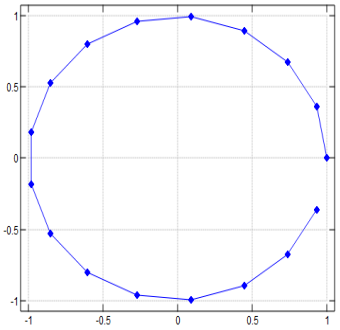
2. 2π を掛け、円周に割り振る

```
      2p1 * (i.17)%17
0 0.369599 0.739198 1.1088 1.4784 1.848 2.21759 2.58719 2.95679
3.32639 3.69599 4.06559 4.43519 4.80479 5.17439 5.54399 5.91359
```

3. $e^{i\theta}$ を演算し、複素数での xy の座標を求める
   r. は Angle

```
   r.  12r17
0.761039j0.648706
   ^j.12r17
0.761039j0.648706
```

4. plot は複素数を扱えるが、polygon のように始点と終点を結んではくれない
   `'marker line' plot r. 2p1 * (i.17)%17`



5. dwin は縦型のデータでグラフィックスに渡す。+. で成形すればよい。
   +.(Real/Imaginary) はグラフィックス用に複素数を実部と虚部に分離するとそのまま X,Y のデータになる。(Y は虚数として取り扱う必要は無くガウス座標でなくデカルト座標で取り扱う)

```
 (+. r. 2p1*(i.17)%17),1 0
        1         0
 0.932472  0.361242
 0.739009  0.673696
```

```
 0.445738  0.895163
0.0922684  0.995734
_0.273663  0.961826
_0.602635  0.798017
_0.850217  0.526432
_0.982973   0.18375
_0.982973  _0.18375
_0.850217 _0.526432
_0.602635 _0.798017
_0.273663 _0.961826
0.0922684 _0.995734
 0.445738 _0.895163
 0.739009 _0.673696
 0.932472 _0.361242
        1         0
```

2　グラフィックスの選択

別稿で述べるように J の正規版 gl2 のグラフィックスは非常にデリケートで扱いづらい。

C.Reiter が Fractal *Visualization and J*〕で用いた *gl2* にかぶせるツールを *QT* に対応させたものを *fvj4* として公開しており、愛用している

2.1　Form を使わないなら C.Reiter の dwin が便利

C.Reiter のフラクタルを作った時のグラフィックスが *addons/fvj4* に入っている。

グラフィックスの名手が作った *gl2* の *addon* で *gl2* ネイティブより *10* 倍は使い勝手がよい。フォームには組み込めないのでボタンや *edit* のみの小さなフォームを別に作ってもよい。

特色　　特色は次の通り

1. *JQT* よりも遙かにタフ
2. *gl2* の画像一枚に額縁一枚の制限が無い
3. 左下が（*0*，*0*）
4. 相対座標が使える
5. 描画関数は異なるが難しくは無い

ロード　　`addons/graphics/fvj4/dwin.ijs`

キャンバス　キャンバスの指定

1. 号数 初期設定は *500* 号

   ```
   setWIN_WH 500 500
   ```

2. 相対画像　（左下（*xy)* 右上 *(xy))*

   ```
   _1 _1 1 1 dwin 'itomaki'
   ```

   これで*-1* から *1* までを *55* 号で描くことが出来る

描画関数は *3* つのみでグラフィックスの面倒くさい手続きは関数に組み込まれている。追加したければ *dline2* などとしてコピーした上で所用の改変を施せばよい。

- *dline*
- *dpoly*
- *dpixel*

カラー　左引数に *RGB* で指定。指定しなければ黒

255 0 255 dline i.10

データの形式　縦型（*x,.y*）なので扱いやすい

2.2 dwin で描く

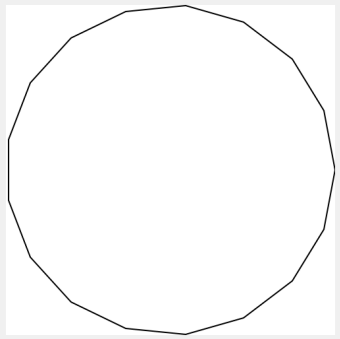
1. 正多角形用の汎用関数を作成
2. 額縁は不要
3. 正多角形の頂点座標を求める汎用関数を作成

```
poly0=: 3 : ' r. 2p1*(i.y)%y ' NB. complex number
poly=: 3 : '(+. poly0 y) , 1 0'    NB. for dline add 1 0
```

4. *dwin* で描く。*2* 行で足り、手打ちで足りる。

```
_1 _1 1 1 dwin ''
dline (poly 17),1 0
```

5. 画像



3　糸掛け曼荼羅に挑戦-dwin 版

既に正多角形の複素座標生成関数 *poly0* と使い勝手がよい *dwin* を用意してある
糸掛け曼荼羅は正 *32* 角形や正 *48* 角形、正 *64* 角形があるようだ。
素数に限定しないで好きな数でやる方法もあるようだがここでは素数曼荼羅とする。

1. *64* までの素数を求める

```
      find_prime 64
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61
```

2. ここで数学頭から工作頭に切り替えなければならない。*32* ピンで次の関係をみると、*31* の糸掛けとは左回りに *1* の糸掛けをしていることである。大きい素数を幾つか省かなければならない。

```
2 3 5 7 11 13 17 19 23 29 31   NB.  右回り
        21 19 15 13  9  3  1  NB. 左回り

find_prime=: 3 : 0
ind=: y> tmp=: p: i. y  NB. pickup index
ind # tmp               NB. pickup prime
)
```

3. 適当な素数でやめるが、適当を定義しなければならない。上の *32* ピンの例では *17* までとしたい
ピン数*0.6* から *1* 減じた数がよさそうだ。

```
 take_prime=: 3 : 0
NB. take half +1
NB. usage: u 64
num=:<:+/ (2r3*y)>:tmp=:find_prime y
 num {. tmp
NB. |:  num {. tmp
)
```

```
,.>take_prime (L:0) {@>32 48 64
2 3 5 7 11 13 17  0  0  0  0  0
2 3 5 7 11 13 17 19 23 29  0  0
2 3 5 7 11 13 17 19 23 29 31 37
```

ワークショップでは大きい素数から糸掛けを行っているようだが、グラフィックスでは小さい数から先に描いた方が大きい数が上書きされるので見た目がよいようだ。

4. *Stop* の目印=起点に戻る
   最少公倍数で終結する　*(*. LCM)*　ことが理解できたので最大回数を見積もる

```
   |:  (p:i.15),.   (p: i.15) *. / 16 32 48 64
 2   3   5   7  11  13   17   19   23   29   31   37   41   43   47 NB. prime
16  48  80 112 176 208  272  304  368  464  496  592  656  688  752 NB. 16
32  96 160 224 352 416  544  608  736  928  992 1184 1312 1376 1504 NB. 32
48  48 240 336 528 624  816  912 1104 1392 1488 1776 1968 2064 2256 NB. 48
64 192 320 448 704 832 1088 1216 1472 1856 1984 2368 2624 2752 3008 NB. 64
```

5. 素数で糸を掛けるピンを抜き出す。複素数のままの方が扱いやすい。

```
   4 4 $ clean 2 calc_pos 16
 1   0.707107j0.707107  0j1 _0.707107j0.707107
_1 _0.707107j_0.707107 0j_1 0.707107j_0.707107
 1   0.707107j0.707107  0j1 _0.707107j0.707107
_1 _0.707107j_0.707107 0j_1                  1
```

```
calc_pos=: 4 : 0
NB.Usage: 2  u 16
Max=. x find_LCM y                  NB. LCM
Times=. Max % x                     NB. Maximum times
Cand=. Max {. ; x #<poly0 y         NB. all
PicIndex=. (i. Max) e. x * i. Times NB. pop index
add_lastone PicIndex # Cand         NB. pickup and draw to start
)
```

6. 小物ツール

最後の点から起点へ戻る線を加える

```
add_lastone=: 3 : ' y , }: y '
```

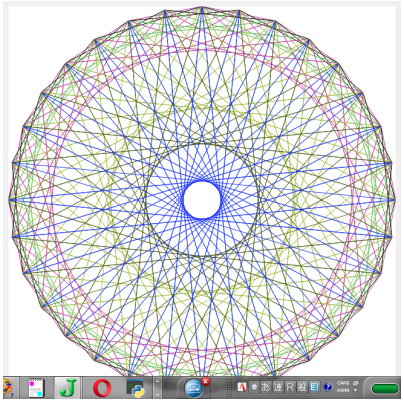
色（*RGB)* を乱数で打ち出す

```
color=: 3 : ' 3 ? 256'  NB. random
```

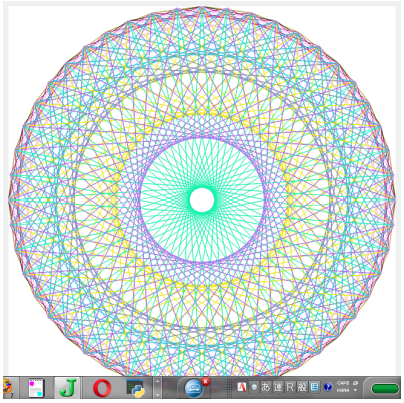
7. メインプログラム

```
itokake_main=: 3 : 0
NB.  Usage u 32
Prime=.  take_prime y  NB. regular order
_1 _1 1 1 dwin 'Itokake Mandala'  NB. 相対座標 (左下：右上)
for_ctr. i. # Prime  do.
Dat=: (ctr{ Prime)  calc_pos y     NB. 素数毎にピン座標を求める
(color '') dline +. Dat           NB.  乱数で RGB を求め素数毎に描く
```

32pin



48pin



4　一本ずつの糸掛け

ワークショップ用に一本ずつの糸掛けグラフィックスを作成する。

1. 一本ずつ糸掛けするスクリプトを作成する

```
itokake_each=: 4 : 0
NB.  Usage:  17 u 32
_1 _1 1 1 dwin 'Itokake Mandara'
x itokake_each0 y
)
```

```
itokake_each0=: 4 : 0
NB. Usage: 5 u 32
Dat=:  x calc_pos y
(color '') dline1 +. Dat
)
```

2. *dwin* ではコマンド打ち込みになる。最初に糸掛けする素数を打ち出しておく

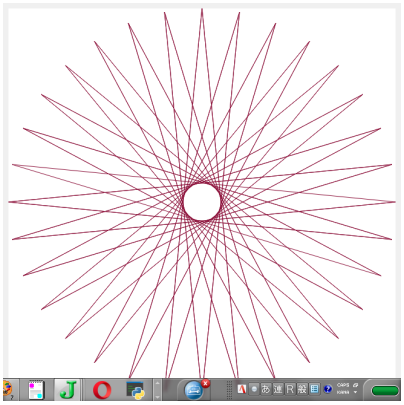
```
   take_prime 32
2 3 5 7 11 13 17
```

3. 最初に `itokake_each` でフォームと一本掛ける

```
   17 itokake_each 32
```

4. `itokake_each0` で一本ずつ掛けていく

```
   13 itokake_each0 32
```



付録 A　グラフィックスに渡すデータのフォーム

三者三様である。

plot　*plot* の *XY* のフォームは次の通り
（横に *X,Y* を *2* 行に並べ `{` *(Catalogue)* で整型すると簡単）

X :	1	2	3	4	5
Y :	11	12	13	14	15

→

1	2	3	4	5	11	12	13	14	15

dwin　*dwin* は縦型 *+.* で複素分離すればよい

gl2　*gl2* の様式は珍しい。複素分離したものを*;* で一行にする

```
data=: x,y,x,y,x,y,x,y
```