JAPLA研究会資料　2007/8/4（蓼科夏の合宿）

カオス・フラクタルとＪグラフィックス

西川　利男

０．はじめに

カオス・フラクタルのプログラミングとは、ＩＦＳ（iterative function system）反復関数系なる変換処理を行うものであり、Ｊの配列処理にうって付けである。また、この変換には線形変換より一歩進んだアフィン変換から、さらにはもっと複雑な高次多項式や分数式など非線形の変換まで登場する。一方、Ｊのグラフィックスとしては、手軽なplotパッケージによる方法から、gl2とマウスの入出力インターフェースを用いたウィンドウズ・システムの構築まで、目的に応じて選択して利用できる。

ここではいろいろなカオス・フラクタルへのＪの適用の実際例を紹介する。プログラムのアルゴリズムにはExcel, BASIC による以下の文献[2, 3]を参考にした。

１．ＩＦＳ処理とカオス・フラクタル

平面上の点（x, y）を位置ベクトルとして表したとき、回転、伸縮を行うのに、マトリックスによる線形変換が使われるのはよく知られている。これに移動まで含めた変換がアフィン変換（Affin Transform）と呼ばれるものである。

アフィン変換はしたがって以下の式で計算される。

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} p \\ q \end{pmatrix}$$

なお、C. Reiterはこれをまとめて行うため、Homogenous Coordinateと名付ける座標系を用いているが、上の単純な方法で十分だと筆者は思う。

さらに実際のカオスの計算では、もっと複雑な計算式で行うものも多い。

ＩＦＳ処理とは、このような変換を以下のように繰り返し反復実行する。

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \to \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \to \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \to \cdots$$

このようにして得られる点（x_n, y_n）の組が、ある決まった値に収束する場合もあり、発散する場合もある。あるいは値が不安定で定まらず混沌とした状態を続けるときもある。このような状態をカオス(Chaos)と言っているのである。

ところでこれをグラフで図示すると、ときには思いがけない美的な図形パターンが現れることがある。そして同時にこの図形パターンは自己相似性、つまりフラクタル(Fractal)構造を示すことも多い。

初めはコンピュータ科学者の興味を引いただけだったのが、今ではもっと一般的に、日常の物理現象、あるいは社会現象、さらには宇宙の起源の理解など、量子力学、相対論と並ぶ新しい自然観としてもてはやされてきている[1]。

[1] イアン・スチュアート著、須田不二夫・三村和男訳
「カオス的世界像－神はサイコロ遊びをするか？」白揚社(1992).
[2] 深谷茂樹「フラクタル・グラフィックス」山海堂(1995).
[3] 臼田，東野，井上，伊藤，葭谷「カオスとフラクタル」オーム社(1999).

２．シェルピンスキーのガスケット

フラクタルとして最も有名なシェルピンスキーのガスケットの計算とグラフ表示の過程を、Ｊで少していねいにプログラミングしてみよう。

シェルピンスキー図は３つの点から成る三角形をそれぞれ反復縮小することで作られる。したがってこれを実現するには３つの IFS が必要である。

まず、最初の IFS は簡単な線形変換で次のようになる。

$$ifa:\ \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

これに対するＪの関数は次のようになる。

```
ifa =: 3 : 0
((2 2)$(0.5 0 0 0.5)) +/ . *"(1) y.
)
```

ここで、初期値としての正三角形の３つの点の座標

$$(0,0),\ \ (2,0),\ \ \left(1,\sqrt{3}\right)$$

を決め、これをＪで次のように定義する。

```
ini =: (0, 0);(2, 0);(1, (%:3))
```

これを関数 ifa で実行すると、次のようになる。

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

   ifa L:0 ini
+---+---+------------+
|0 0|1 0|0.5 0.866025|
+---+---+------------+

   ifa L:0 ^:(i.3) ini
+---+-----+-------------+
|0 0|2 0  |1 1.73205    |
+---+-----+-------------+
|0 0|1 0  |0.5 0.866025 |
+---+-----+-------------+
|0 0|0.5 0|0.25 0.433013|
+---+-----+-------------+
```

このように最初の IFS、ifa では原点に近づく反復縮小する三角形の各点を与える。

２番目の IFS は次のようになる。これは移動も行っているのでアフィン変換である。

$$ifb:\ \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

これに対するＪの関数は次のようになる。

```
ifb =: 3 : 0
(2, 0) + ((2 2)$(0.5 0 0 0.5)) +/ . *"(1) y.
)
```

これは点(2, 0)に近づく反復縮小する三角形の各点を与える。

３番目の IFS は次のようになり、これもアフィン変換である。

$$ifc: \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} 1 \\ \sqrt{3} \end{pmatrix}$$

```
ifc =: 3 : 0
(1, (%:3)) + ((2 2)$(0.5 0 0 0.5)) +/ . *"(1) y.
)
```

これは点(1, √3)に近づく反復縮小する三角形の各点を与える。

シェルピンスキー図は初期値の三角形に対して、このような３つの IFS を行うことによって作られる。したがって、全体のプログラムは次のようになる。

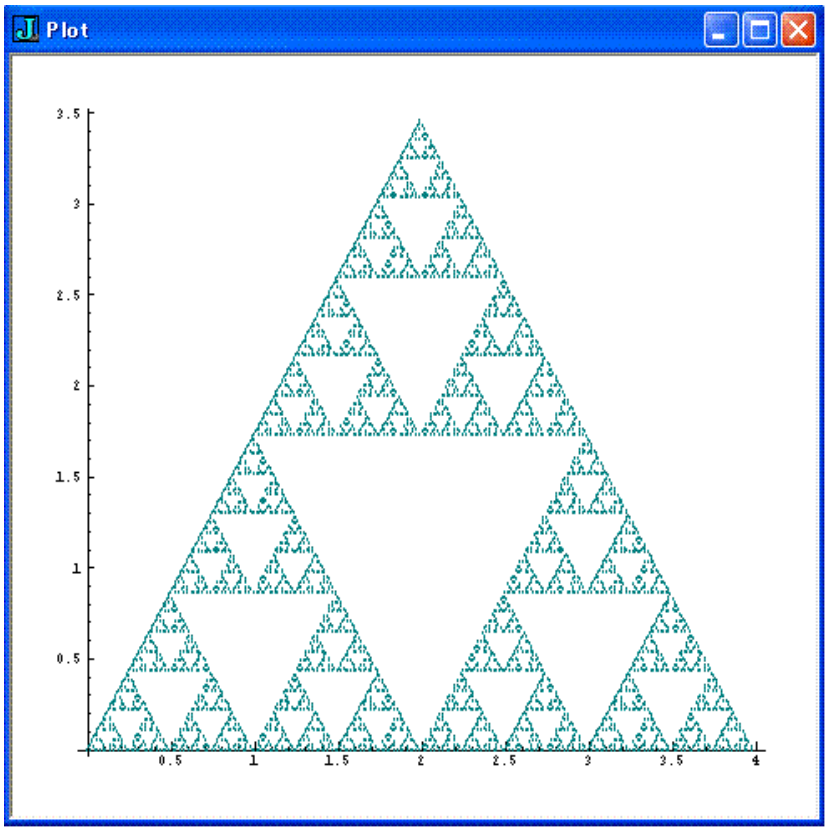
```
sierp =: 3 : 0
DA =. ifa L:0 y.
DB =. ifb L:0 y.
DC =. ifc L:0 y.
DA, DB, DC
)
```

次に点の座標を図示するには、簡単には plot ルーチンで行えばよいが、引数としてボックス化した X, Y の値を必要とするので、そのための次の display 関数を定義した。

```
display =: 3 : 0
'point' plot <"(1) |: > y.
)
```

実際の実行は次のように行われる。

```
 display sierp^:(7) ini
```



３．エノンのカオス・フラクタル

エノン(Henon) のカオス・フラクタルでは、IFS はもっと複雑な反復関数系である。

$$x_1 = 1 - a\,x_0^{\,2} + y_0$$

$$y_1 = b\,x_0$$

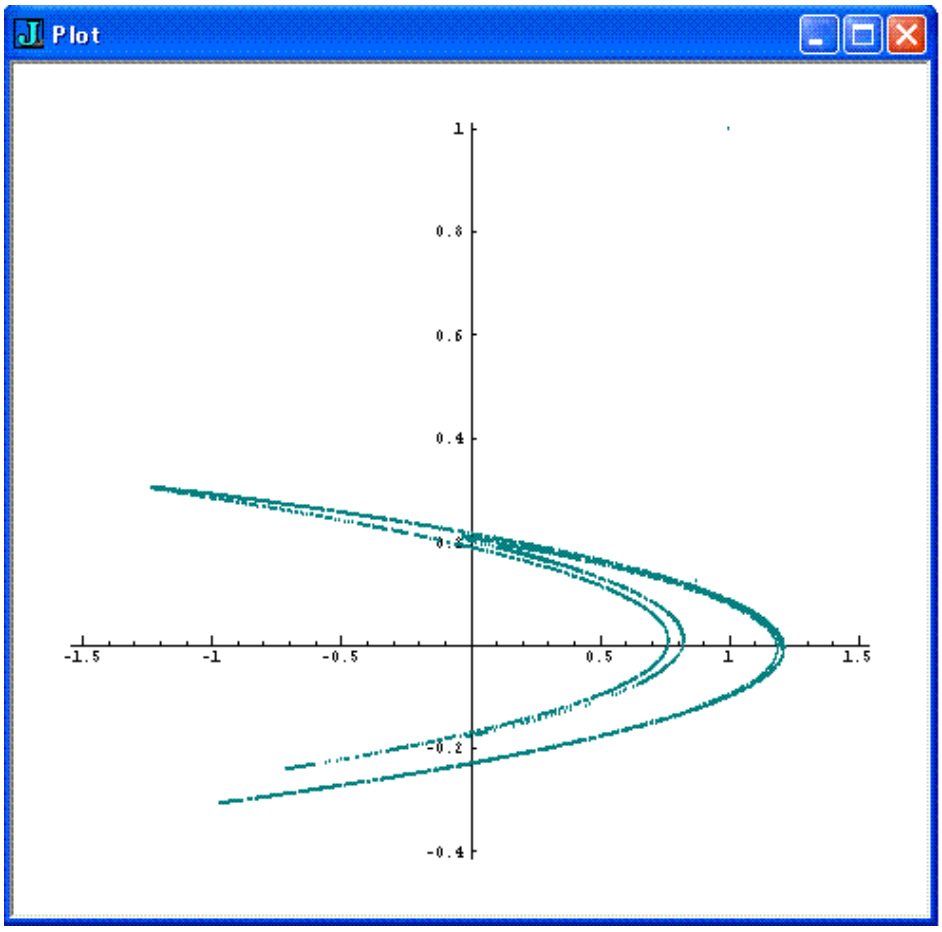
ここでは関数を要素とする一種の拡張されたマトリックスの形で表現してみる。

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} -a\ square & 1 \\ b & 0 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Ｊでは、次のようにしてこれに合わせたコーディングが可能である。

```
HA =: 1.5  ] HB =: 0.25
dia =: i.@#}        NB. extract diagnal terms
henon =: 3 : 0
'x0 y0' =. y.
x1 =. +/ dia (1: - (HA"_ * *:))`]`:(0) x0, y0
y1 =. +/ dia (HB"_ * ])`0:`:(0) x0, y0
x1, y1
)

display henon^:(i.10000) 1 1
```



４．グモウスキーとミラのカオス・フラクタル

これは、分数式をも含めた非線形のさらに複雑な反復関数系である。パラメータを変えるといろいろなパターンが現れるということから、アートを目的としたカオス・フラクタルと言えるものかもしれない。パラメータを変えてもやってみよう。

$$x(t+1) = y(t) + a\,[1 - b\,y^2(t)]\,y(t) + G\,[x(t)]$$

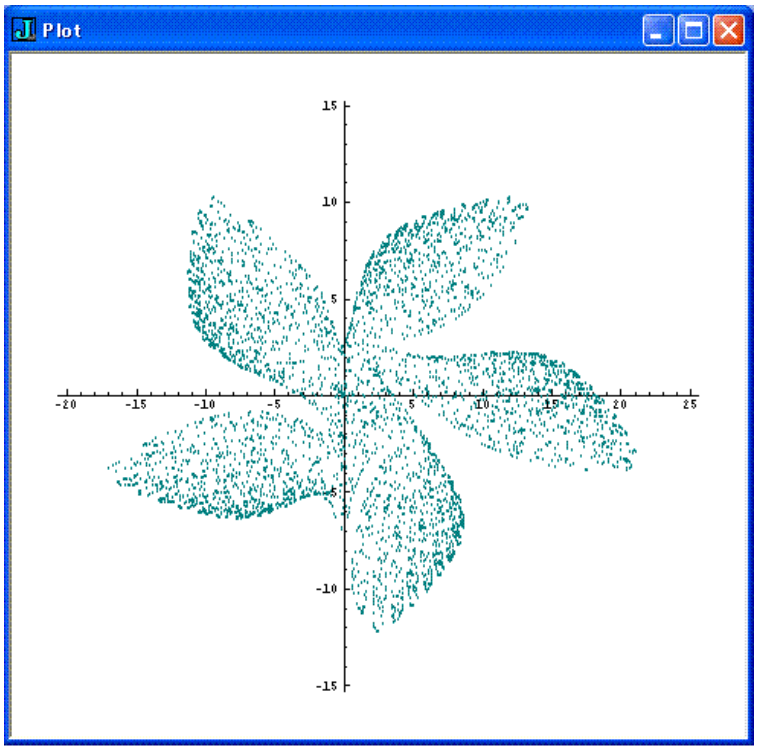
$$y(t+1) = -\,x(t) + G\,[x(t+1)]$$

$$G\,[x(t)] = \mu\,x(t) + \frac{2(1-\mu)\,x^2(t)}{1+x^2(t)}$$

```
MU =: _0.8
GA =: 0.008
GB =: 0.05
GX =: 3 : 0
:
(x. * y.) + (2*(1-x.)*(*: y.))%(1 + *: y.)
)
gumow =: 3 : 0
:
'x0 y0' =. y.
x1 =. y0 + (y0*GA*(1 - GB * *:y0)) + (x. GX x0)
y1 =. (-x0) + (x. GX x1)
x1, y1
)
```

実行は次のようになる。

```
display MU gumow^:(i.10000) 0.1 0
```



５．ジャパニーズ・アトラクタ

難しい式を用いるが、結果は図を見てのとおりで、日本人の発見によるものである。名前は「日本の魅力」とも読みとれるかもしれない。

$$\frac{dx}{dy} = y, \qquad \frac{dy}{dt} = -ky - x^3 + B\cos t$$

計算には、微分方程式の代わりに、差分方程式を使って順次求める。

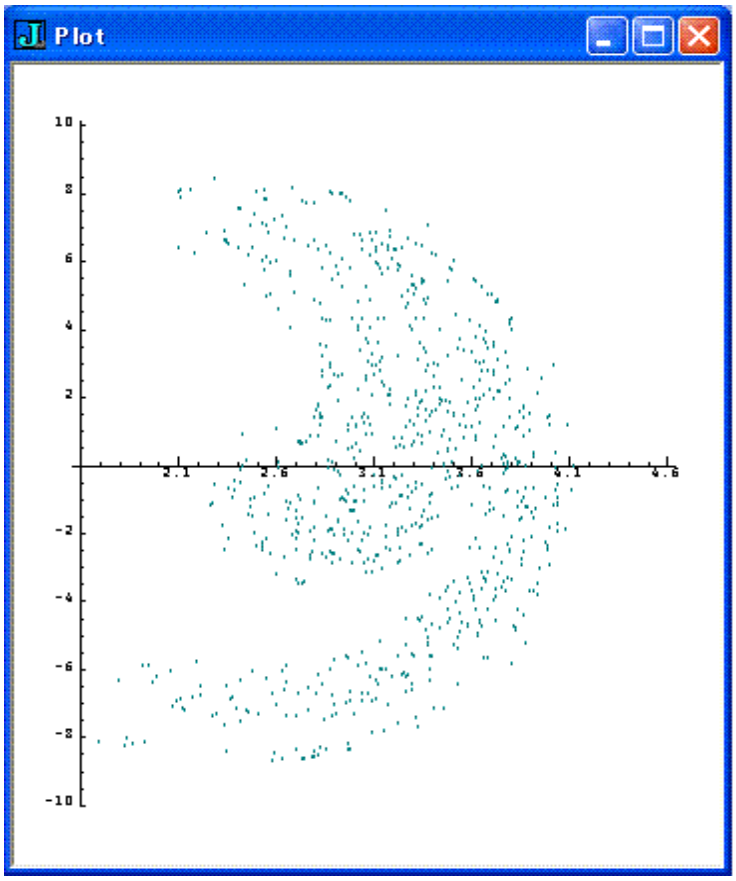
$$\Delta x = y\Delta t, \qquad \Delta y = (-kx - x^3 + B\cos t)\Delta t$$

$$x_1 = x_1 + \Delta x, \qquad y_1 = y_0 + \Delta y$$

８００回の反復で、cos tが１になる時間でのx，yの値を取り出し、これを１０００回、つまり８００×１０００回の計算を行って値JAが得られるが、相当の時間がかかる。そのうち最初の２００項は除いた値JBをグラフ化している。

```
JA =: jaxy 1000
JB =. 200 }. JA
display 2 }.″(1) JB
```

実行結果は次のようになる。



Ｊのプログラムは以下の通りである。

```
JK =: 0.1
JB =: 12
dt =: 2r800p1
japat =: 3 : 0
't0 c0 x0 y0' =. y. NB. t, cos_t, x, y
t1 =. t0 + dt
dx =. y0*dt
dy =. ((-JK*y0)+(-x0^3)+(JB*cos t1))*dt
x1 =. x0 + dx
y1 =. y0 + dy
t1, (cos t1), x1, y1
)

jaxy =: 3 : 0
n =. y.
JJ =. japat^:(i.800*n) 0 1 0.2 0.1
(<: 800* >: i.n) { JJ
)
```

６．羊歯のフラクタル図

フラクタル図として有名な羊歯のパターンは、４種類のパラメータを持つアフィン変換ＩＦＳを、別に発生させた乱数の値ｐにより選択して行い、その重ね合わせで図形をつくる。ここでは、深谷氏の値[2]を用いた。

$$ferna:\ \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1.6 \end{pmatrix}$$

$$fernb:\ \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 0.2 & -0.26 \\ 0.23 & 0.22 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1.6 \end{pmatrix}$$

$$fernc:\ \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0.44 \end{pmatrix}$$

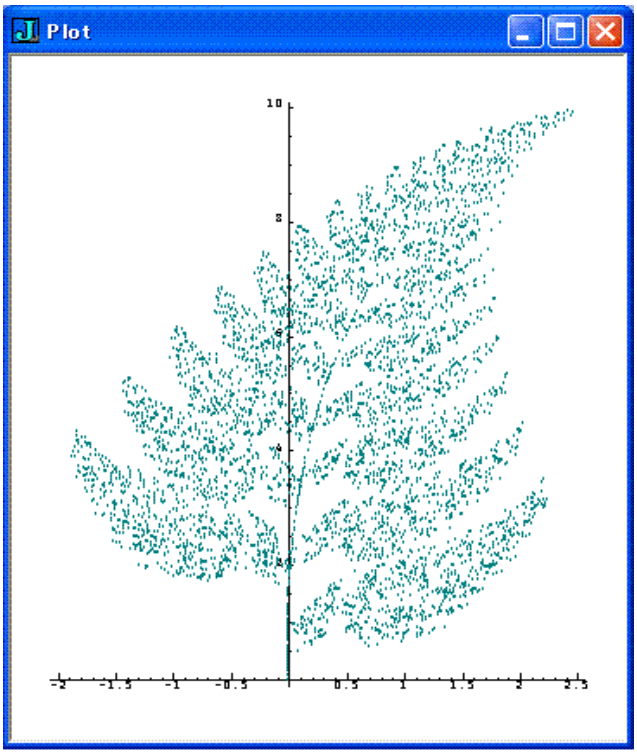
$$fernd:\ \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0.16 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

乱数の値ｐによるＩＦＳの選択は次のように行う。

0 ＜ p ＜ 0. 73 のとき　*ferna*
0.73 ＜ p ＜ 0.86 のとき *fernb*
0.86 ＜ p ＜ 0.99 のとき *fernc*
0.99 ＜ p ＜ 1　　のとき *fernd*

Ｊのプログラムと実行例は次のようになる。

```
fera =: 3 : 0
(0 1.6) + ((2 2)$(0.85 0.04 _0.04 0.85)) +/ . * y.  NB. Hukaya's data
)
ferb =: 3 : 0
(0 1.6) + ((2 2)$(0.2 _0.23 0.26 0.22)) +/ . * y.   NB. Hukaya's data
)
ferc =: 3 : 0
(0 0.44) + ((2 2)$(_0.15 0.26 0.28 0.24)) +/ . * y. NB. Hukaya's data
)
ferd =: 3 : 0
(0 0) + ((2 2)$(0 0 0 0.16)) +/ . * y.              NB. Hukaya's data
)
fern =: 3 : 0
P =. (? 100)%100
if. P < 0.73
   do. fera y.
   elseif. P < (0.73+0.13)
     do. ferb y.
     elseif. P < (0.73+0.13+0.13)
       do. ferc y.
       elseif. 1 do. ferd y.
end.
)
display fern^:(i. 10000) 0 0
```

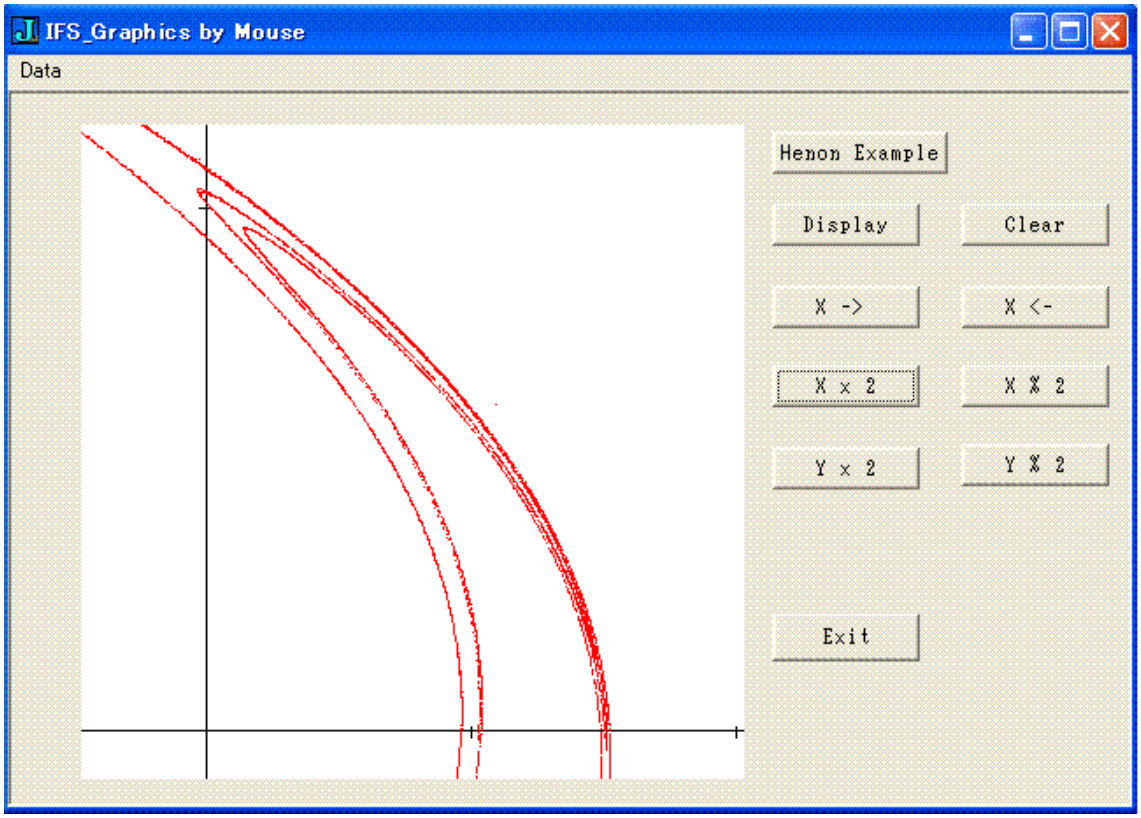


７．マウス入力インターフェースによるＪカオス/フラクタル・システム

いろいろなカオス/フラクタルの図を見てきたが、パターンの細かい部分を拡大したり、適当な位置に移動したり、自分の望みのようにして見てみたい。

まずは、今回構築したＪカオス/フラクタル・システムを全体構成を示してみる。そして、先のエノンのカオス/フラクタル図の詳細部分を見てみよう。いろいろなボタンを押すことでグラフ図形の拡大・縮小・移動が行われる。さらに、マウスの左ボタンのドラッグで自分の望みの位置に移動できる。そしてフラクタル構造を観察することができるのである。

グラフィックスにおいて、このような双方向の対話的ウィンドウズシステムはもっと活用してほしい。マウスはボタンを押すためだけにあるのではない。



Ｊでは gl2(11 !:*)命令を使えば、マウス操作のプログラミングが可能である。これをカオス/フラクタル・システムの構築により示してみよう。

Ｊのグラフィックス・ウィンドウズ・プログラミングについてはすでに何回か解説したが、当日、プログラムコードを通して、実際に行ってみたいと思う。

Ｊではウィンドウズ・プログラムの最初となるフォームの設計については、Form Editor により簡単に行える。　ここでは実行のためのいろいろなボタンとともに、isigraph のグラフィック画面を設定する。各ボタンの処理実行の内容はＪのプログラム・コードとして記述する。

最も関心のもとであるマウス操作のプログラムは次のようになる。
例えばマウスを押したことによるイベントが発生するが、

```
ifsgraph_ifsgrap_mbldown=: 3 : 0
d=. ". sysdata
x0=:(0{d) * 1000 % (2{d)
y0=:(1{d) * 1000 % (3{d)
…
……
)
```

以上のプログラムで、マウスで指定した位置はピクセル値 x0, y0 として返されるので、これをもとにいろいろな処理を行えばよい。

gl2 によるグラフィックス・プログラミングは、基本的には次のような命令の組み合わせで行われる。

require ‘gl2’	gl2 命令の組み込み
glrgb	色の指定
glpen	ペンの種類、太さなどの指定
gllines X0, Y0, X1, Y1, X2, Y2, ……	点を結ぶ
glpixel X0, Y0, X1, Y1, X2, Y2, ……	点を打つ
glshow	指定した値、仕様でグラフを描く

なお、J の座標系では画面の左下を(0, 0)、右上を(1000, 1000)とする(BASIC とは異なる)ピクセル値となっている。
計算値からピクセル値への変換、座標軸、目盛の刻みなど、gl2 ではおっくうだが、ユーザ自身でやらなくてはならない。また、(X, Y)の値の指定順序も先の plot の場合と異なっているので注意を要する。
先の plot パッケージを使うより、gl2 ではプログラムを組むことになり、やや面倒であるが非常に広範囲の処理が可能となる。グラフ化した結果を見るだけではなく、このような融通性のあるグラフィックスシステムを作ることができる。

８．おわりに

カオス・フラクタルについては現代を代表するテーマであり、いろいろな本も出ているし、インターネット上ではそのグラフィック図形を見るソフトも数多くある。
しかしながら、自分なりに理論を理解して、自分でプログラムを組んで動かすことの意義は大きく、まさにコンピュータ黎明期に戻った感があり、筆者にとってその工程はまことに楽しいものであった。

Ｊのプログラム・リスト

```
NB. Fractals on Iterative Function System
NB. Mouse Graphics using gl2
============================================================
IFSGRAPH=: 0 : 0
pc ifsgraph;pn "IFS_Graphics by Mouse";
menupop "Data";
menu heno "&Henon" "" "" "";
menu gumo "&Gumowsky" "" "" "";
menu japa "&Japanese" "" "" "";
menu ferr "&Fern" "" "" "";
menusep ;
menu exit "&Exit" "" "" "";
menupopz;
xywh 20 8 189 166;cc ifsgrap isigraph;
xywh 217 10 50 12;cc ok button;cn "Henon Example";
xywh 217 28 48 11;cc Display button;
xywh 271 28 44 11;cc Clear button;
xywh 217 49 48 11;cc SR button;cn "X -> ";
xywh 271 49 46 11;cc SL button;cn "X <- ";
xywh 217 69 46 11;cc XL button;cn "X x 2";
xywh 271 69 42 11;cc XS button;cn "X % 2";
xywh 217 90 47 11;cc YL button;cn "Y x 2";
xywh 271 89 42 11;cc YS button;cn "Y % 2";
xywh 217 132 41 12;cc cancel button;cn "Exit";
pas 6 6;pcenter;
rem form end;
)

run =: ifsgraph_run
ifsgraph_run=: 3 : 0
wd IFSGRAPH
require 'gl2'
XP =: 250          NB. adjust factor of value to pixel
YP =: 1000
X0 =: 500          NB. pixel of graph origin
Y0 =: 500
XE =: 1                  NB. enlarge factor of pictures
YE =: 1
NG =: 20           NB. number of grid interval
LG =: 2000         NB. length of axis
axis_grid X0, Y0, NG
glshow ''
NB. initialize form here
wd 'pshow;'
```

```
)
ifsgraph_close=: 3 : 0
wd'pclose'
)

ifsgraph_cancel_button=: 3 : 0
ifsgraph_close''
)

ifsgraph_Clear_button=: 3 : 0
glclear ''
XP =: 250          NB. adjust factor of value to pixel
YP =: 1000
XO =: 500          NB. pixel of graph origin
YO =: 500
XE =: 1                  NB. enlarge factor of pictures
YE =: 1
NG =: 10           NB. number of grid interval
LG =: 1000         NB. length of axis
axis_grid XO, YO, NG
glshow ''
)

NB. Select and Calculate Item on the Menu Bar, Return DA(global)
ifsgraph_heno_button=: 3 : 0
DA =: henon^:(i.20000) 1 1
)

ifsgraph_gumo_button=: 3 : 0
DA =: MU gumow^:(i.10000) 0.1 0
)

ifsgraph_japa_button=: 3 : 0
JA =. jaxy 1000
NB. JB =. 200 }. JA
DA =: 2 }."(1) JA
)

ifsgraph_ferr_button=: 3 : 0
DA =: fern^:(i. 10000) 0 0
)

NB. Run Henon Picture
ifsgraph_ok_button=: 3 : 0
DA =: henon^:(i.20000) 1 1
```

```
ifsgraph_Display_button ''
)
NB. Display Any Picture in DA
ifsgraph_Display_button=: 3 : 0
axis_grid X0, Y0, NG
HX =: X0 + XE * XP * {."(1) DA
HY =: Y0 + YE * YP * {:"(1) DA
HXY =: HX ,. HY
glrgb 255 0 0
glpen 1 0
glpixel L:0 <"1 HXY
glshow ''
)

NB. X Shift to Right
ifsgraph_SR_button=: 3 : 0
glclear ''
X0 =: X0 + 100
axis_grid X0, Y0, NG
ifsgraph_Display_button ''
)

NB. X Shift to Left
ifsgraph_SL_button=: 3 : 0
glclear ''
X0 =: X0 - 100
axis_grid X0, Y0, NG
ifsgraph_Display_button ''
)

NB. X enlarge picture
ifsgraph_XL_button=: 3 : 0
glclear ''
XE =: XE * 2
ifsgraph_Display_button ''
)

NB. X shrink picture
ifsgraph_XS_button=: 3 : 0
glclear ''
XE =: XE % 2
ifsgraph_Display_button ''
)

NB. Y enlarge picture
```

```
ifsgraph_YL_button=: 3 : 0
glclear ''
YE =: YE * 2
ifsgraph_Display_button ''
)

NB. Y shrink picture
ifsgraph_YS_button=: 3 : 0
glclear ''
YE =: YE % 2
ifsgraph_Display_button ''
)

NB. draw X, Y axes with grid graduations
axis_grid =: 3 : 0
'X0 Y0 N' =. y.
NB. calculate x_grid and y_grid positions
GRX0 =. ((-@|.),}.)@i.<.N%XE*2
GRX1 =: 2*(LG%N)*GRX0
GRX =: X0 + XE*GRX1
GRY0 =. ((-@|.),}.)@i.<.N%YE*2
GRY1 =: 2*(LG%N)*GRY0
GRY =: Y0 + YE*GRY1
glrgb 0 0 0
glpen 1 0
NB. draw x-axis
gllines (X0-LG), Y0, (X0+LG), Y0
NB. draw y-axis
gllines X0, (Y0-LG), X0, (Y0+LG)
NB. draw x-grid
X_gr =. <"1 (GRX,.(Y0-10)), "(1) GRX,.(Y0+10)
gllines L:0 X_gr
NB. draw y-grid
Y_gr =. <"1 ((X0-10),.GRY), "(1) (X0+10),. GRY
gllines L:0 Y_gr
glshow ''
)

NB. Mouse_Left / Move Picture from MB_Down to MB_Up Position
ifsgraph_ifsgrap_mbldown=: 3 : 0
d=. ". sysdata
x0=:(0{d) * 1000 % (2{d)
y0=:(1{d) * 1000 % (3{d)
glclear ''
axis_grid X0, Y0, NG
```

```
ifsgraph_Display_button ''
glrgb 0 0 255
glpen 1 0
glrect x0, y0, 10, 10     NB. indicate mouse point
glshow ''
)

ifsgraph_ifsgrap_mmove=: 3 : 0
d=.". sysdata
if. -.4{d do. return. end.
x1=.(0{d) * 1000 % (2{d)
y1=.(1{d) * 1000 % (3{d)
)

ifsgraph_ifsgrap_mblup=: 3 : 0
d=.". sysdata
NB. if. -.4{d do. return. end.
x2=.(0{d) * 1000 % (2{d)
y2=.(1{d) * 1000 % (3{d)
glclear ''
X0 =: (x2-x0) + X0
Y0 =: (y2-y0) + Y0
axis_grid X0, Y0, NG
ifsgraph_Display_button ''
glshow ''
)

NB. Mouse_Right / Move Origin of Graph at MB_Right Position
ifsgraph_ifsgrap_mbrdown=: 3 : 0
d=. ". sysdata
X0=:(0{d) * 1000 % (2{d)
Y0=:(1{d) * 1000 % (3{d)
glclear ''
axis_grid X0, Y0, NG
ifsgraph_Display_button ''
)
```

```
NB. Chaos Fractal Programs ========================================
NB. 臼田, 東野, 井上, 伊藤, 葭谷「カオスとフラクタル」オーム社
NB. Henon's Chaos p.96-99
HA =: 1.5           NB. Henon's Parameter A
HB =: 0.25          NB. Henon's Parameter B
dia =: i.@#}
NB. e.g. henon^:(i.10) 1 1
NB. e.g. display henon^:(i.10000) 1 1
henon =: 3 : 0
'x0 y0' =. y.
x1 =. +/ dia (1: - (HA"_ * *:))`]`:(0) x0, y0
y1 =. +/ dia (HB"_ * ])`0:`:(0) x0, y0
x1, y1
)

NB. Gumowski-Mira's Chaos p.106-108
MU =: _0.8
GA =: 0.008
GB =: 0.05
NB. e.g. MU gumow^:(i.10000) 0.1 0
NB. e.g. display MU gumow^:(i.10000) 0.1 0
GX =: 3 : 0
:
(x. * y.) + (2*(1-x.)*(*: y.))%(1 + *: y.)
)
gumow =: 3 : 0
:
'x0 y0' =. y.
x1 =. y0 + (y0*GA*(1 - GB * *:y0)) + (x. GX x0)
y1 =. (-x0) + (x. GX x1)
x1, y1
)


NB. Japanese Attractor p.109-114
NB. test calculated values
NB. e.g. japat 0 1 0.2 0.1
NB. display attractor picture
NB. e.g.   JA =: jaxy 1000 => takes some minutes
NB. e.g.   display 2 }."(1) JA
NB. e.g.   JB =. 200 }. JA => drop first 200 values
NB. e.g.   display 2 }."(1) JB

JK =: 0.1
JB =: 12
```

```
dt =: 2r800p1
japat =: 3 : 0
't0 c0 x0 y0' =. y. NB. t, cos_t, x, y
t1 =. t0 + dt
dx =. y0*dt
dy =. ((-JK*y0)+(-x0^3)+(JB*cos t1))*dt
x1 =. x0 + dx
y1 =. y0 + dy
t1, (cos t1), x1, y1
)

jaxy =: 3 : 0
n =. y.
JJ =. japat^:(i.800*n) 0 1 0.2 0.1
(<: 800* >: i.n) { JJ
)


NB. Sierpinsky in Hukaya's Parameters
NB. 深谷茂樹「フラクタル・グラフィックス」p.139-140
ifa =: 3 : 0
((2 2)$(0.5 0 0 0.5)) +/ . *"(1) y.
)
ifb =: 3 : 0
(2, 0) + ((2 2)$(0.5 0 0 0.5)) +/ . *"(1) y.
)
ifc =: 3 : 0
(1, (%:3)) + ((2 2)$(0.5 0 0 0.5)) +/ . *"(1) y.
)

ini =: (0, 0);(2, 0);(1, (%:3))

sierp =: 3 : 0
DA =. ifa L:0 y.
DB =. ifb L:0 y.
DC =. ifc L:0 y.
DA, DB, DC
)

display =: 3 : 0
'point' plot <"(1) |: > y.
)
```

```
NB. Fern Fractal 2007/6/26 ==============================
NB. Usage: display fern^:(i. 1000) fini

fera =: 3 : 0
NB. (0.0375 0.17) + ((2 2)$(0.85 _0.04 0.04 0.85)) +/ . * y.
(0 1.6) + ((2 2)$(0.85 0.04 _0.04 0.85)) +/ . * y.  NB. Hukaya's data
NB. (0.07 0.147) + ((2 2)$(0.856 0.0414 _0.0205 0.355)) +/ . * y.
)

ferb =: 3 : 0
NB. (0.2 0.1025) + ((2 2)$(0.2 0.23 _0.26 0.22)) +/ . * y.
(0 1.6) + ((2 2)$(0.2 _0.23 0.26 0.22)) +/ . * y.   NB. Hukaya's data
NB. (0.393 _0.102) + ((2 2)$(0.244 _0.385 0.176 0.224)) +/ . * y.
)

ferc =: 3 : 0
NB. (0.2875 _0.021) + ((2 2)$(_0.15 0.26 0.28 0.24)) +/ . * y.
(0 0.44) + ((2 2)$(_0.15 0.26 0.28 0.24)) +/ . * y. NB. Hukaya's data
NB. (0.527 _0.014) + ((2 2)$(_0.144 0.39 0.181 0.259)) +/ . * y.
)

ferd =: 3 : 0
NB. (0.25 0) + ((2 2)$(0 0 0 0.16)) +/ . * y.
(0 0) + ((2 2)$(0 0 0 0.16)) +/ . * y.                    NB. Hukaya's data
NB. (0.486 0.05) + ((2 2)$(0 0 0.355 0.216)) +/ . * y.
)

fini =: 0 0

fern =: 3 : 0
P =. (? 100)%100
if. P < 0.73
   do. fera y.
   elseif. P < (0.73+0.13)
     do. ferb y.
     elseif. P < (0.73+0.13+0.13)
       do. ferc y.
       elseif. 1 do. ferd y.
end.
)
```