JAPLA 研究会資料 2017/10/21

Ｊでマンデルブローのプログラムとグラフィックスを楽しむ

西川　利男

「パソコンの上でプログラミングを楽しむ」とは、どういうことだろうか。
そのためには、数学の原理はすっきりと単純なものがよい。それを実現するプログラムも、力づくでしゃにむに作るというより、こなれた技術で素直にやりたい。そして、その結果は自然の不思議にせまるものであって欲しい。

マンデルブロー集合をＪのプログラムで計算してフラクタルな図を得る、というのは、まさにこれにふさわしいテーマではないだろうか。

先月の JAPLA の例会で、志村正人氏の発表[1]に触発されて、しばらくぶりで、Ｊのマンデルブロー・グラフィックをまたやってみた。

マンデルブローの図は、一度見たらば忘れられない奇妙な図だ。どんどん細かくすると、同じようなパターンが次々と現れる。しかし、フラクタル現象の不思議さよりも、その図そのものがアートとして有名になってしまった。

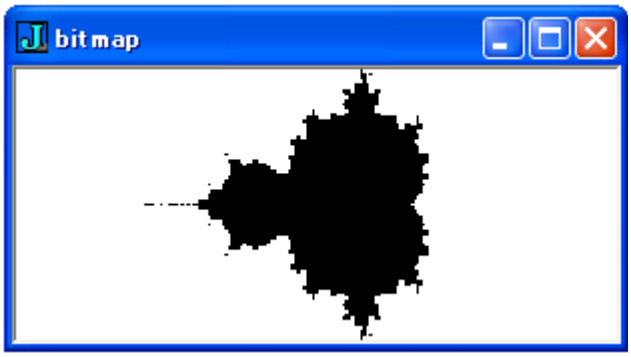
私自身も、フラクタルの翻訳書[2]を出していた頃に集めた BASIC によるフラクタルといった本が手元に何冊もある。[3]

ＪはBASIC とは全く異なるプログラムであり、極めてコンパクトに、あざやかにマンデルブローの図を実現してくれる。志村正人氏の以前の報告に Odaira のマンデルブローのＪプログラムがあるが、あまりにコンパクトすぎて、なかなかそのコーディングを追えない。私なりに、ゆっくり楽しみながら、やってみることにした。

Ｊでは、BASIC など他の言語にない次のような３つの特徴により、マンデルブロー・システムを極めて効率的にプログラミングすることができる。

・常備されている j. など複素数計算の数々の動詞
・繰り返し演算の副詞、パワー ^:
・viewmat などのグラフィックツール

Ｊを使えば、簡単につぎのように描かれる。しかし、内容はなかなか難しい。



[1] 志村　正人「複素数で楽しむグラフィックス」JAPLA, (2017/9/8).
[2] ラウヴェリエール、西川利男訳「初めてのフラクタル」丸善(1996).
[3] R. Devaney, "Chaos, Fractals, and Dynamics", Addison-Wesley(1990).

１．マンデルブロー集合の数学とＪプログラム

マンデルブロー集合を原理に基づいて理解し、プログラムにより計算し、図示することは、知識欲をかきたてる大きな楽しみがある。

ここには、数学の計算とコンピュータ上でプログラムを作るという、言うなれば理学と工学の問題がある。

１．１　複素数を用いた繰り返し演算

ある数(右引数)を２乗して２(左引数)を加えるという、次の関数 mand から始める。

```
   mand =: 4 : 0
x + *: y
)
```

この関数 mand を使って、得られた結果を次の初期値としてさらに計算する、という繰り返し操作を次々とやってみる。

```
   2 mand 0                          NB. 最初は初期値０として mand ０から始める
2
   (2 mand (2 mand 0))               NB. 上の結果２を使って、mand ２を行う
6
   (2 mand (2 mand (2 mand 0)))      NB. さらに上の結果６を使って、mand ６を行う
38
```

パワー(繰り返し)の副詞(^:)を活用して、繰り返し行う関数 mandel を定義する。

```
   mandel =: 4 : 0
n =. x
pos =. y
pos mand^:n 0
)
```

今度は、上の動詞 mandel を使って、２回、あるいは３回と繰り返し計算を行う。

```
   2 mandel 2
6
   3 mandel 2
38
```

ここで、関数 mandel の左引数は繰り返しの回数で、右引数は初期値である。

これを先の関数 mand を用いて、確かめてみよう。

```
   2 mand^:(i.6) 0
0 2 6 38 1446 2090918
```

今度は、初期値として、複素数の値、0.2*i* (J では 0j0.2)を用いてみる。

```
   2 mandel 0j0.2
_0.04j0.2
   3 mandel 0j0.2
_0.0384j0.184
```

前と同様に、関数 mand を用いて、確かめてみる。

```
   0j0.2 mand^:(i.6) 0
0 0j0.2 _0.04j0.2 _0.0384j0.184 _0.0323814j0.185869 _0.0334987j0.187963
```

この計算操作がマンデルブロー演算の基本になる。

１．２　複素数平面上での多数の点の繰り返し収束

　まず、準備として、ある範囲にわたって、多数の複素数を生成する必要がある。
それには、つぎのような、Ｊの関数 posit を定義することで、機能的に行われる。

```
posit =: 3 : 0
j. /&steps/ y
)
```

```
   j. &(/&steps/)
+--+-+---------------+
|j.|&|+-----------+-+|
|  | ||+-+-+-----+|/||
|  | |||/|&|steps|| ||
|  | ||+-+-+-----+| ||
|  | |+-----------+-+|
+--+-+---------------+
```

　ここで、関数 steps は

```
   load 'numeric trig'
```

として、サポートされる便利なツールで、例えば次のようにして、2 から 8 まで 10 間隔の値が得られる。

```
   steps 2 8 10
2 2.6 3.2 3.8 4.4 5 5.6 6.2 6.8 7.4 8
   $ steps 2 8 10
11
```

　以上の準備をした上で、小さな複素数配列データを用いて、マンデルブロー演算の収束から始めていく。
　例として、(_2j_1) から(2j1)まで、実数部 10 等分、虚数部 5 等分の 11 x 6 の配列の値を生成して、マンデルブロー演算を一歩ずつやってみよう。

```
   DM00 =: _2 2 10,: _1 1 5
   DM00
_2 2 10
_1 1  5
```

上のパラメータを引数として、関数 posit によりマンデルブロー演算の初期値とする。

```
   posit DM00
  _2j_1   _2j_0.6   _2j_0.2   _2j0.2   _2j0.6   _2j1
_1.6j_1 _1.6j_0.6 _1.6j_0.2 _1.6j0.2 _1.6j0.6 _1.6j1
_1.2j_1 _1.2j_0.6 _1.2j_0.2 _1.2j0.2 _1.2j0.6 _1.2j1
_0.8j_1 _0.8j_0.6 _0.8j_0.2 _0.8j0.2 _0.8j0.6 _0.8j1
_0.4j_1 _0.4j_0.6 _0.4j_0.2 _0.4j0.2 _0.4j0.6 _0.4j1
   0j_1    0j_0.6    0j_0.2    0j0.2    0j0.6    0j1
 0.4j_1  0.4j_0.6  0.4j_0.2  0.4j0.2  0.4j0.6  0.4j1
 0.8j_1  0.8j_0.6  0.8j_0.2  0.8j0.2  0.8j0.6  0.8j1
 1.2j_1  1.2j_0.6  1.2j_0.2  1.2j0.2  1.2j0.6  1.2j1
 1.6j_1  1.6j_0.6  1.6j_0.2  1.6j0.2  1.6j0.6  1.6j1
   2j_1    2j_0.6    2j_0.2    2j0.2    2j0.6    2j1
```

この複素数配列のそれぞれに繰り返し10回のマンデルブロー演算mandelを行う。

```
   MA0 =: 10 mandel"(0) posit DM00
   $MA0
11 6
```

10回で収束した値のそれぞれに複素数の絶対値をとる。

```
   | MA0
1.31492e134  7.71934e99   3.9474e56   3.9474e56  7.71934e99 1.31492e134
1.56385e103  2.15147e61  8.12733e17  8.12733e17  2.15147e61 1.56385e103
 5.32292e76  2.50918e30    0.363202    0.363202  2.50918e30  5.32292e76
 5.29523e52  2.47891e13    0.588736    0.588736  2.47891e13  5.29523e52
 5.60098e23     0.59857    0.341691    0.341691     0.59857  5.60098e23
    1.41421    0.338419    0.190339    0.190339    0.338419     1.41421
 3.76475e58     1.12221    0.950477    0.950477     1.12221  3.76475e58
8.59674e104  2.80224e75   2.5114e62   2.5114e62  2.80224e75 8.59674e104
9.20893e147 4.13305e128 3.18636e118 3.18636e118 4.13305e128 9.20893e147
3.71105e185 6.35205e171 2.90588e164 2.90588e164 6.35205e171 3.71105e185
3.90171e218  2.2994e208 8.76255e202 8.76255e202  2.2994e208 3.90171e218
```

```
   MA1 =: | MA0
```

この値が1以下ならば(収束)、1以上ならば(発散)0を値とするフラグの配列で示す。

```
   MA2 =: 1 > MA1
   MA2
0 0 0 0 0 0
0 0 0 0 0 0
0 0 1 1 0 0
0 0 1 1 0 0
0 1 1 1 1 0
0 1 1 1 1 0
0 0 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

先の関数positの結果は、行（タテ）に実数部が増加、列（ヨコ）に虚数部が増加するように出力される。普通の複素数ガウス平面（実数部がヨコ、X軸、虚数部がタテY軸）に合わせるには、転置したほうがよい。

```
  MAN =: |: MA2
  MAN
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
```

これが、小さいデータで行った(0, 1)で表したマンデルブロー集合の図に他ならない。なお、収束値を細かく色データに対応させれば、色つきの図も簡単に出来る。

２．マンデルブロー集合のグラフィックス

Ｊの viewmat ツールを用いて、先の配列 MAN を引数として実行すればマンデルブロー図は描かれるが、ここではもっと詳細に操作したいので、このあと、汎用の gl2 グラフィックスでＪのプログラミングを進めていく。

２つのグラフィックスの方式

一般にグラフィックス表示のプログラムには、２つの方法がある。

・gllines, glpolygon のように頂点座標を結んで行う普通の描画グラフィックス

・画像イメージの表示方式、画像配列の位置での値、
　たとえば値０なら白、値１なら黒のようにデータの位置に対応して、
　色付け表示する。viewmat などがこれである。
　普通、ユーザは引数として配列値を渡すだけで、プログラムはいじれない。

(0, 1)配列からピクセル位置への方式－ここでのやり方

gl2 グラフィックスでは、頂点位置を指定して、gllines, glpolygon と同様にして、glpixel として、点を打つことができる。したがって、ここでは(0, 1)配列からグラフィックス命令のためのピクセル位置を得て、画像表示を行うようにした。

まず、ピクセル位置の座標値 X, Y の集まりを作る。

```
   X =: 500 + 40 * i. 11
   Y =: 500 + 40 * i. 6
   (<"(0) X) (,L:0) /"(1 0) (<"(0) Y)
+-------+-------+-------+-------+-------+-------+-------+-------+-------+
+----|500 500|540 500|580 500|620 500|660 500|700 500|740 500|780 500|820
500|860 +-------+-------+-------+-------+-------+-------+-------+-------+
+-------+----|500 540|540 540|580 540|620 540|660 540|700 540|740 540|780
540|820 540|860
+-------+-------+-------+-------+-------+-------+-------+-------+-------+----
|500 580|540 580|580 580|620 580|660 580|700 580|740 580|780 580|820 580|860
+-------+-------+-------+-------+-------+-------+-------+-------+-------+
+----|500 620|540 620|580 620|620 620|660 620|700 620|740 620|780 620|820
620|860
+-------+-------+-------+-------+-------+-------+-------+-------+-------+
+----|500 660|540 660|580 660|620 660|660 660|700 660|740 660|780 660|820
660|860
+-------+-------+-------+-------+-------+-------+-------+-------+-------+
+----|500 700|540 700|580 700|620 700|660 700|700 700|740 700|780 700|820
700|860
+-------+-------+-------+-------+-------+-------+-------+-------+-------+----
```

ここで、先のマンデルブロー配列 MAN の各値 (0, 1)をフラグと考えて、

```
  MAN
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
```

Ｊのプリミティブ#により、フラグ１のところだけを座標値として取り出ことができる。

```
   (, MA3) # , (<"(0) X) (,L:0) /"(1 0) (<"(0) Y)
+-------+-------+-------+-------+-------+-------+-------+-------+-------+
+----|660 540|700 540|580 580|620 580|660 580|700 580|740 580|580 620|620
620|660
+-------+-------+-------+-------+-------+-------+-------+-------+-------+----
620|700 620|740 620|660 660|700 660|
---+-------+-------+-------+-------+
```

これを元に次のようにして glpixel 命令によりグラフィックス画面に点を打つ。

```
glpixel 660 540, glpixel 700 540, glpixel 580 580, , ,
```

ここでの制約として、マンデルブロー配列 MAN とピクセル位置の座標値配列とは、行数、列数が完全に等しく、同じ大きさでなければならない。

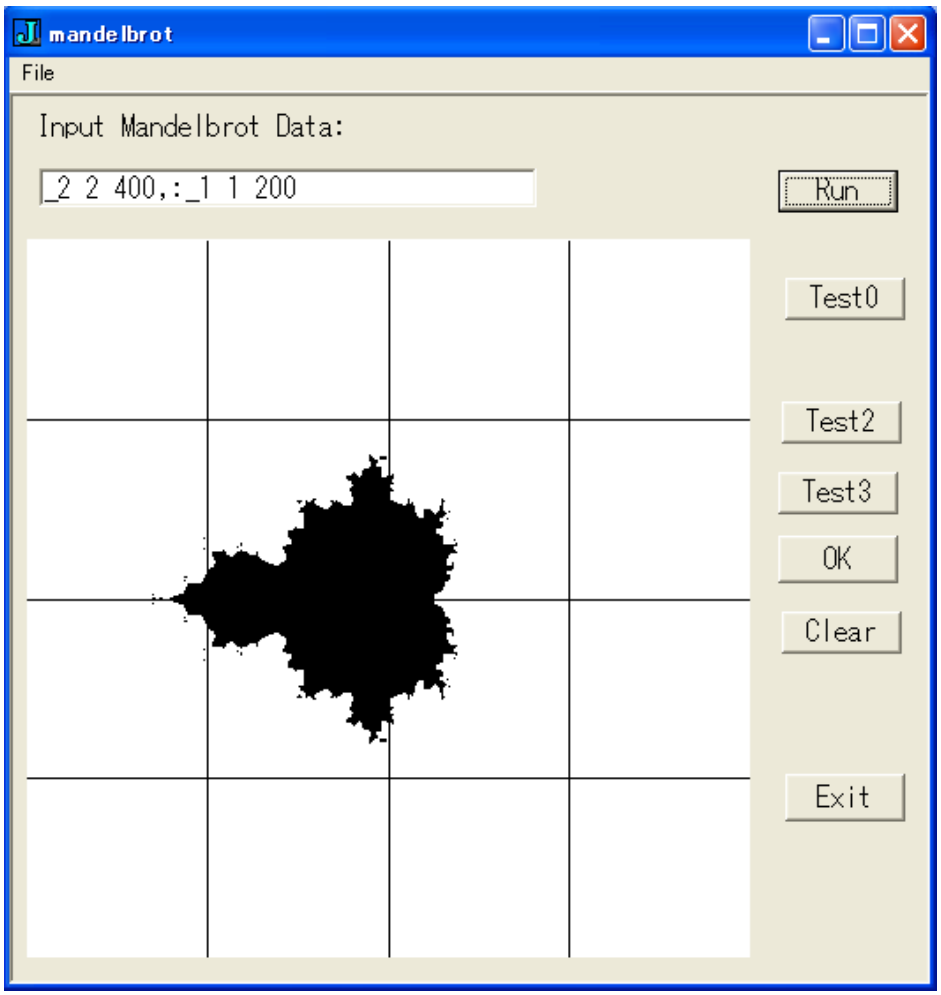
これでは、使いにく不便である。つぎの対策をおこなった。

マンデルブロー配列 MA を画面全体のピクセル座標値に拡大整合させる。
　つまり、一般に小さい配列である(0,1) マンデルブロー配列を全画面ピクセル配列 (0, 0) - (1000, 1000) (画素ハードウェアに依存) に拡大整合させる。
　そのための J プログラムを matrix_expand(詳細は付録に)として作ったが、これをクラスファイルとして、そのインスタンスを取り込み使用した。

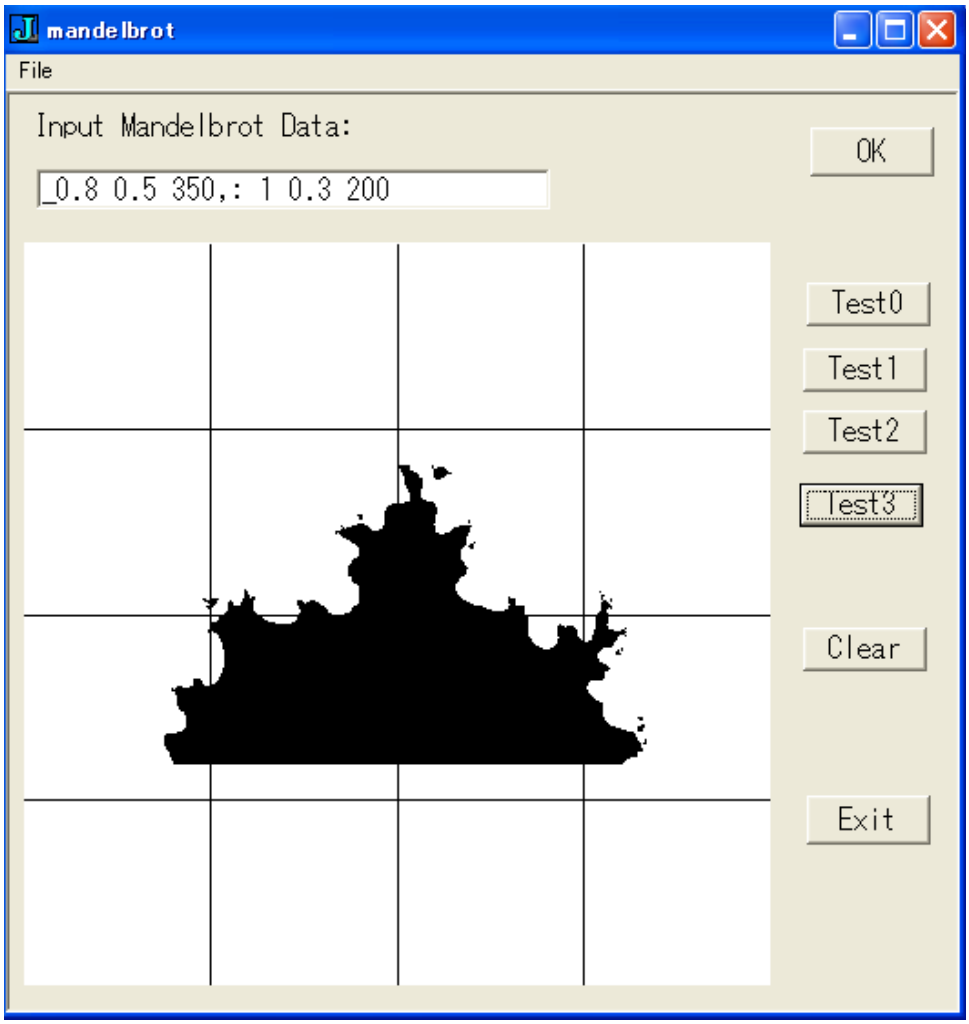
```
load 'user¥classes¥pmatexpand.ijs'
mp =: conew 'pmatexpand'
matexpand =: matexpand__mp
```

３．マンデルブローの J グラフィックス の実際

　J のウィンドウズ・グラフィックスとして、マンデルブローの位置、拡大度などをユーザが決めて実行できるようなシステムとして構築した。パラメータはエディットボックスから入力指定する。
　実行の実際を示す。J プログラムの詳細は付録にまわした。



マンデルブロー集合の図は、さらに細かく見ると構造が見られる。これがフラクタル現象いわれるものである。したがってその部分を拡大して見ることにする。
パラメータを変えて、実行してみる。このようにさらに細かく構造があらわれる。いくつかの小さな島が見られるが、それぞれが最初の大きいマンデルブロー集合のコピーのようになっている。まさにフラクタル現象のようすを示す。



４．Ｊによるマンデルブロー・フラクタル図形の特徴とテスト作成しシムの難しさ

Ｊ言語の計算処理方法として、簡単に出来ると思ったものの、思いがけない数々の難しさに出合った。
・配列の対応に注意をするタテｘヨコ、行ｊ列
i の配列要素 ＝＞ 画像座標(Y, X)の点
なる位置における繰り返し収束を示す属性の図がマンデルブロー図である。
これは、もし高さで表すとすると３次元グラフィックスで行う問題でもある。
・マンデルブローの(0,1)配列と画像ピクセル座標との大きさ整合、調整が厄介。
・座標数値など、計算の途中をトレースしようとすると巨大数のため扱いにくい。
しかし、マンデルブロー図のようなピクセルグラフィックスは、通常の線画のグラフィックスとは違うととがわかり、これも新たな発見として、楽しませてもらった。

Ｊのプログラム・リスト

```
NB. Small Example Data
  DMO =: _2 2 20,: _1 1 10
  DMO
_2 2 20   NB. Real Part = Start End Range
_1 1 10   NB. Imaginary = Start End Range

   steps _2 2 20
_2 _1.8 _1.6 _1.4 _1.2 _1 _0.8 _0.6 _0.4 _0.2 0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
```

```
   steps _1 1 10
_1 _0.8 _0.6 _0.4 _0.2 0 0.2 0.4 0.6 0.8 1

   (steps _2 2 20) (j./) (steps _1 1 10)
  _2j_1   _2j_0.8   _2j_0.6   _2j_0.4   _2j_0.2   _2   _2j0.2   _2j0.4   _2j0.6   _2j0.8   _2j1
_1.8j_1 _1.8j_0.8 _1.8j_0.6 _1.8j_0.4 _1.8j_0.2 _1.8 _1.8j0.2 _1.8j0.4 _1.8j0.6 _1.8j0.8 _1.8j1
_1.6j_1 _1.6j_0.8 _1.6j_0.6 _1.6j_0.4 _1.6j_0.2 _1.6 _1.6j0.2 _1.6j0.4 _1.6j0.6 _1.6j0.8 _1.6j1
_1.4j_1 _1.4j_0.8 _1.4j_0.6 _1.4j_0.4 _1.4j_0.2 _1.4 _1.4j0.2 _1.4j0.4 _1.4j0.6 _1.4j0.8 _1.4j1
_1.2j_1 _1.2j_0.8 _1.2j_0.6 _1.2j_0.4 _1.2j_0.2 _1.2 _1.2j0.2 _1.2j0.4 _1.2j0.6 _1.2j0.8 _1.2j1
  _1j_1   _1j_0.8   _1j_0.6   _1j_0.4   _1j_0.2   _1   _1j0.2   _1j0.4   _1j0.6   _1j0.8   _1j1
_0.8j_1 _0.8j_0.8 _0.8j_0.6 _0.8j_0.4 _0.8j_0.2 _0.8 _0.8j0.2 _0.8j0.4 _0.8j0.6 _0.8j0.8 _0.8j1
_0.6j_1 _0.6j_0.8 _0.6j_0.6 _0.6j_0.4 _0.6j_0.2 _0.6 _0.6j0.2 _0.6j0.4 _0.6j0.6 _0.6j0.8 _0.6j1
_0.4j_1 _0.4j_0.8 _0.4j_0.6 _0.4j_0.4 _0.4j_0.2 _0.4 _0.4j0.2 _0.4j0.4 _0.4j0.6 _0.4j0.8 _0.4j1
_0.2j_1 _0.2j_0.8 _0.2j_0.6 _0.2j_0.4 _0.2j_0.2 _0.2 _0.2j0.2 _0.2j0.4 _0.2j0.6 _0.2j0.8 _0.2j1
   0j_1    0j_0.8    0j_0.6    0j_0.4    0j_0.2    0    0j0.2    0j0.4    0j0.6    0j0.8    0j1
 0.2j_1  0.2j_0.8  0.2j_0.6  0.2j_0.4  0.2j_0.2  0.2  0.2j0.2  0.2j0.4  0.2j0.6  0.2j0.8  0.2j1
 0.4j_1  0.4j_0.8  0.4j_0.6  0.4j_0.4  0.4j_0.2  0.4  0.4j0.2  0.4j0.4  0.4j0.6  0.4j0.8  0.4j1
 0.6j_1  0.6j_0.8  0.6j_0.6  0.6j_0.4  0.6j_0.2  0.6  0.6j0.2  0.6j0.4  0.6j0.6  0.6j0.8  0.6j1
 0.8j_1  0.8j_0.8  0.8j_0.6  0.8j_0.4  0.8j_0.2  0.8  0.8j0.2  0.8j0.4  0.8j0.6  0.8j0.8  0.8j1
   1j_1    1j_0.8    1j_0.6    1j_0.4    1j_0.2    1    1j0.2    1j0.4    1j0.6    1j0.8    1j1
 1.2j_1  1.2j_0.8  1.2j_0.6  1.2j_0.4  1.2j_0.2  1.2  1.2j0.2  1.2j0.4  1.2j0.6  1.2j0.8  1.2j1
 1.4j_1  1.4j_0.8  1.4j_0.6  1.4j_0.4  1.4j_0.2  1.4  1.4j0.2  1.4j0.4  1.4j0.6  1.4j0.8  1.4j1
 1.6j_1  1.6j_0.8  1.6j_0.6  1.6j_0.4  1.6j_0.2  1.6  1.6j0.2  1.6j0.4  1.6j0.6  1.6j0.8  1.6j1
 1.8j_1  1.8j_0.8  1.8j_0.6  1.8j_0.4  1.8j_0.2  1.8  1.8j0.2  1.8j0.4  1.8j0.6  1.8j0.8  1.8j1
   2j_1    2j_0.8    2j_0.6    2j_0.4    2j_0.2    2    2j0.2    2j0.4    2j0.6    2j0.8    2j1

NB. Mandelbrot Function ==========================================
   mand =: 4 : 0
x. + *: y.
)
   mandel =: 4 : 0
n =. x.
pos =. y.
pos mand^:n 0
)
   mandelb =: 3 : 0
Z =: 10 mandel"(0)  |: posit y.
W =: 1 > | Z
NB. viewmat  |: |. W
)
```

```
NB. Data Generate Function ==========================
posit =: 3 : 0
j. /&steps/ y.
)
```

```
   j. &(/&steps/)
+--+-+---------------+
|j.|&|+-----------+-+|
|  | ||+-+-+-----+|/||
|  | |||/|&|steps|| ||
|  | ||+-+-+-----+| ||
|  | |+-----------+-+|
+--+-+---------------+
```

```
   posit DM0
  _2j_1   _2j_0.8   _2j_0.6   _2j_0.4   _2j_0.2   _2   _2j0.2   _2j0.4   _2j0.6   _2j0.8   _2j1
_1.8j_1 _1.8j_0.8 _1.8j_0.6 _1.8j_0.4 _1.8j_0.2 _1.8 _1.8j0.2 _1.8j0.4 _1.8j0.6 _1.8j0.8 _1.8j1
_1.6j_1 _1.6j_0.8 _1.6j_0.6 _1.6j_0.4 _1.6j_0.2 _1.6 _1.6j0.2 _1.6j0.4 _1.6j0.6 _1.6j0.8 _1.6j1
_1.4j_1 _1.4j_0.8 _1.4j_0.6 _1.4j_0.4 _1.4j_0.2 _1.4 _1.4j0.2 _1.4j0.4 _1.4j0.6 _1.4j0.8 _1.4j1
_1.2j_1 _1.2j_0.8 _1.2j_0.6 _1.2j_0.4 _1.2j_0.2 _1.2 _1.2j0.2 _1.2j0.4 _1.2j0.6 _1.2j0.8 _1.2j1
  _1j_1   _1j_0.8   _1j_0.6   _1j_0.4   _1j_0.2   _1   _1j0.2   _1j0.4   _1j0.6   _1j0.8   _1j1
_0.8j_1 _0.8j_0.8 _0.8j_0.6 _0.8j_0.4 _0.8j_0.2 _0.8 _0.8j0.2 _0.8j0.4 _0.8j0.6 _0.8j0.8 _0.8j1
_0.6j_1 _0.6j_0.8 _0.6j_0.6 _0.6j_0.4 _0.6j_0.2 _0.6 _0.6j0.2 _0.6j0.4 _0.6j0.6 _0.6j0.8 _0.6j1
_0.4j_1 _0.4j_0.8 _0.4j_0.6 _0.4j_0.4 _0.4j_0.2 _0.4 _0.4j0.2 _0.4j0.4 _0.4j0.6 _0.4j0.8 _0.4j1
_0.2j_1 _0.2j_0.8 _0.2j_0.6 _0.2j_0.4 _0.2j_0.2 _0.2 _0.2j0.2 _0.2j0.4 _0.2j0.6 _0.2j0.8 _0.2j1
   0j_1    0j_0.8    0j_0.6    0j_0.4    0j_0.2    0    0j0.2    0j0.4    0j0.6    0j0.8    0j1
 0.2j_1  0.2j_0.8  0.2j_0.6  0.2j_0.4  0.2j_0.2  0.2  0.2j0.2  0.2j0.4  0.2j0.6  0.2j0.8  0.2j1
 0.4j_1  0.4j_0.8  0.4j_0.6  0.4j_0.4  0.4j_0.2  0.4  0.4j0.2  0.4j0.4  0.4j0.6  0.4j0.8  0.4j1
 0.6j_1  0.6j_0.8  0.6j_0.6  0.6j_0.4  0.6j_0.2  0.6  0.6j0.2  0.6j0.4  0.6j0.6  0.6j0.8  0.6j1
 0.8j_1  0.8j_0.8  0.8j_0.6  0.8j_0.4  0.8j_0.2  0.8  0.8j0.2  0.8j0.4  0.8j0.6  0.8j0.8  0.8j1
   1j_1    1j_0.8    1j_0.6    1j_0.4    1j_0.2    1    1j0.2    1j0.4    1j0.6    1j0.8    1j1
 1.2j_1  1.2j_0.8  1.2j_0.6  1.2j_0.4  1.2j_0.2  1.2  1.2j0.2  1.2j0.4  1.2j0.6  1.2j0.8  1.2j1
 1.4j_1  1.4j_0.8  1.4j_0.6  1.4j_0.4  1.4j_0.2  1.4  1.4j0.2  1.4j0.4  1.4j0.6  1.4j0.8  1.4j1
 1.6j_1  1.6j_0.8  1.6j_0.6  1.6j_0.4  1.6j_0.2  1.6  1.6j0.2  1.6j0.4  1.6j0.6  1.6j0.8  1.6j1
 1.8j_1  1.8j_0.8  1.8j_0.6  1.8j_0.4  1.8j_0.2  1.8  1.8j0.2  1.8j0.4  1.8j0.6  1.8j0.8  1.8j1
   2j_1    2j_0.8    2j_0.6    2j_0.4    2j_0.2    2    2j0.2    2j0.4    2j0.6    2j0.8    2j1
```

```
NB. Mandelbrot - Small (0, 1) Display (exchanged x, y axis above)
   mandelb"(0) |: posit DM0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 0 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 0
0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
0 1 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 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 0
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
NB. Mandelbrot Graphics ====================================
require 'gl2'
MANDELBROT=: 0 : 0
pc mandelbrot;
menupop "File";
menu new "&New" "" "" "";
menu open "&Open" "" "" "";
menusep ;
menu exit "&Exit" "" "" "";
menupopz;
xywh 219 171 34 12;cc cancel button;cn "Exit";
xywh 4 36 205 181;cc mandel isigraph;
xywh 218 130 34 11;cc Clear button;
xywh 217 19 34 11;cc Run button;
xywh 7 4 110 10;cc label static;cn "Input Mandelbrot Data:";
xywh 7 18 142 11;cc EntDM edit ws_border es_autohscroll;
pas 6 6;pcenter;
rem form end;
)
run =: mandelbrot_run
mandelbrot_run=: 3 : 0
wd MANDELBROT
NB. initialize form here
wd 'pshow;'
)

DM0 =: _2 2 20,: _1 1 10                    NB. thin
DM1 =: _2 2 200,: _1 1 100                  NB. moderate
DM2 =: _2 2 400,: _1 1 200                  NB. most thick

mandelbrot_Run_button=: 3 : 0
DMM =. DM2                                  NB. === Input Parameter
MA0 =. mandelb"(0) |: posit DMM             NB. Mandelbrot Process
MA1 =. ($PIJ2) matexpand MA0                NB. expand & adjust
MA =. (, MA1) # (, PIJ2)                    NB. Mandelbrot Result

glrgb 0 0 0
glpen 1, 0
glbrush ''
gllines 250, 0, 250, 1000                   NB. draw axis lines
 (途中省略)
EdDM =. ({. ": DMM) ,',:', ({: ": DMM)      NB. for Edit Box Data
wd 'set EntDM *', EdDM
glpixel L:0 , MA                            NB. dot Mandelbrot pixel
glshow ''
)
NB. matrix expand as class definition ================================
coclass'pmatexpand'

create=:3 : 0
0
)
destroy=:codestroy

NB. Matrix Expand ============
NB.   2017/9/24 T.N
Mmat =: 9
Nmat =: 13
```

```
AAmat =: 7 9$10 + i.64

NB. (Mmat, Nmat) matexpand AAmat
matexpand =: 3 : 0
:
'M N' =. x.
AA =. y.
K =. {. $ AA
L =. {: $ AA

B0 =. ((<.@-:) K, L) }. AA
BB =. ((>.@-:) M, N) {. B0

C0 =. ((>.@-:)K) {. AA
C1 =. ((<.@-:)L) }."(1) C0
C2 =. |. C1
C3 =. (((>.@-:)M),(>.@-:)N)  {. C2
CC =. |. C3

DD =. CC, }. BB

NB. ==================================
E0 =. ( ((>.@-:) K), (>.@-:) L) {. AA
E1 =. |. E0
E2 =. ((>. @ -:) L) {. E1
E3 =. |."(1) E2
E4 =. (((>.@-:) M), (>.@-:)N) {. E3
E5 =. |. "(1) E4
EE =. |. E5

F0 =. ((<.@-:) K) }. AA
F1 =. ((>.@-:) L) {."(1) F0
F2 =. |. "(1) F1
F3 =. ((>.@-:) N) {."(1) F2
F4 =. |."(1) F3
F5 =. ((>.@-:) M) {. F4

GG =. EE, }.F5

HH =. GG,"(1) }."(1) DD
)
```