J のグラフィックスの概要 (J6 版)

M.Shimura
JCD02773@nifty.ne.jp

2011 年 10 月 21 日

目次

1 J602 の isij グラフィックス 1

2 1000 号キャンバス 3

3 1000 号キャンバスのデモ（Example) 6

4 １号キャンパス 10

5 ピクセル 13

概要

J の Demo → isigraph に優れたデモが入っている。これらの幾つかを描くための、isigraph のガイダンスである

1 J602 の isij グラフィックス

1.1 J のグラッフィクスの種類

J の強力な isi グラフィックスのラインアップ。[*1]

幾何	isigraphics1000 号キャンバス
	isigraph1,2 号キャンバス
	タートルグラフィックス
ピクセル	isigraph1000 号,1･2 号キャンバス
サイエンスグラフ	plot
ペイント	デモにペイントがある

1.2 J6 のグラフィックスの変更点

J6 がら J の isigraphics の使用が一部変更され、それまでのスクリプトが通らなくなった。
主要な変更点は次のとおり。

[*1] plot は別稿

coinsert　スクリプトファイルの最初のほうに **coinsert 'jgl2'** が必要となった。オブジェクト化のためで、使いたくないときは `xxx__jgl2` をつけると動く

glpaint　メモリーバッファがら画面に書き出すコマンドが **glshow** → **glpaint** に変更された

上の 2 点を変更するととりあえず描画はできる

画面構成の変更　(0,0)=左上　(1000,1000)=右下

1.3　J602 のグラフィックスとキャンバスの種類

キャンバスの種類　J の isigraph のキャンバスは次の種類がある。ユーザーは縮尺を変え、自由にアレンジできる。

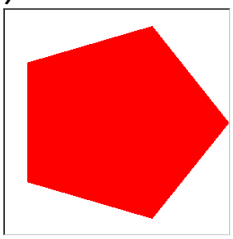
種類	始まり	終わり
gl-1000 号	左上 (xy=0,0)	右下 (xy=1000,1000)

派生したものやスクリプトの作者が自由にアレンジしたものが多くある

種類	始まり	終わり
gd-1 号	(xy=0,0) 左下	(xy=1,1) 右上
gd-2 号	(xy=-1,-1) 左下	(xy=1,1) 右上

- キャンバス 1,2 号は言わば規格化されたデータを扱うのに適している。次のように簡単に図形の座標が求められるのも魅力である

```
polygon_10=: 3 : 0
gdopen ''
RED gdpolygon ,+. r. 2p1*(i.5)%5
gdshow''
)
```



```
   +. r. 2p1*(i.5)%5
        x          y
--------------------
        1          0
 0.309017  0.951057
_0.809017  0.587785
_0.809017 _0.587785
 0.309017 _0.951057
)
```

r. angle 極座標

+. real/imaginary 実部虚部の分離

$\frac{2\pi * (i.5)}{5}$ の極座標を実虚部分離 (x,y)

- キャンバス 1/2 号は左下が始点、右上が終点である

gdlines01 gddraw 0 0 0.3 0.5 0.7 0.9

メモリーバッファ　関数はメモリーバッファに展開されキャンバスに映し出される。メモリーバッファはブラックボックスになっている

キャンバス
↑
メモリーバッファ
↑
関数

1.4 フォームエディタとグラフィックス

J のフォームに isigraph を埋め込むことができる。

section 2 の GL2TEST の画面は form からも作れる。ijs の画面から Edit → FormEditor を起動して empty で isigraph を指定すれば簡略画面が得られる。この画面は 1000 号キャンバスでありフォームで指定した大きさ (例えば 200×200) となる。

2 1000 号キャンバス

J の LAB/Graphics に **Graphics-gl2command** が入っているので観賞してみる。**Ctrl+J** で LAB が一駒づつうごく。このままでは解読しづらいので次の簡単な画面とスクリプトを作った。

- coinsert でオブジェクト jgl2 の関数を直接用いることができる。

```
require 'gl2'
coinsert 'jgl2'
```

- 画面のサイズは最大 1000×1000 である。ここでは 200×200 とした。1000×1000 は大画面の PC が必要であり、J の画面はマウスでサイズを変更すると図柄は消えて再描画はなされない。

```
GL2TEST=: 0 : 0
pc gl2test closeok;
xywh 0 0 200 200;cc tg isigraph rightmove bottommove;
pas 0 0;
rem form end;
)
```

- run ペイント関数を引けるよう副詞とした。
  (255 255 0)& test_rect0 run 10 30 50 100 NB. yellow // xywh
  動詞の左引数でカラー指定を、右引数に図形描画の数値を指定する。左引数は&が必要

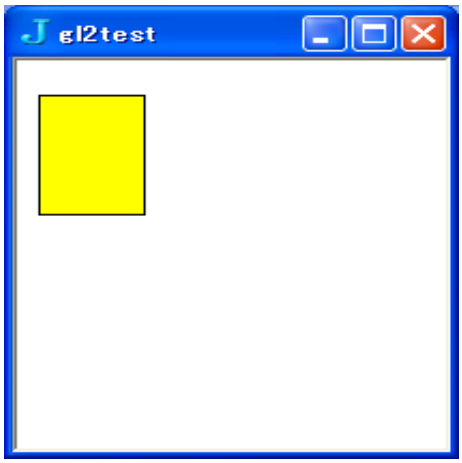
```
run=: gl2test_run=: 1 : 0
NB. Usage:   test_paint0 run
wd GL2TEST
GHANDLE =: wd 'qhwndc tg'
u y
wd 'pshow;'
```

)

- カラーは RGB で与える。1600 万色可能 (eg. 255 255 0 = yellow)

2.1 長方形を描く

glrect　正方形/長方形を描く。




```
test_rect0=: 4 : 0
glclear ''
glrgb  x      NB.  255 0 0  is red(red brush)
glbrush ''    NB. not usuing --> only black outline
glrect  y     NB. 10 30 50 100 is xywh
)
```

xywh　位置と形は **xywh** で与える。左上の始点 (x y) width height

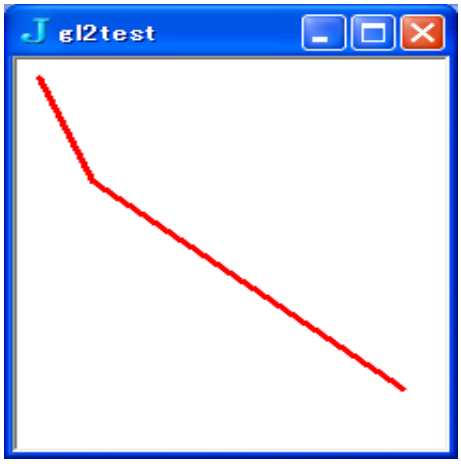
RGB　　RGB は&で連結する

(255 0 0)&test_rect0 run 20 30 50 100

簡易形　次のようにしても長方形を黒の輪郭で描けるが色とブラシは使えない

glrect run 10 20 50 70

2.2 line を描く



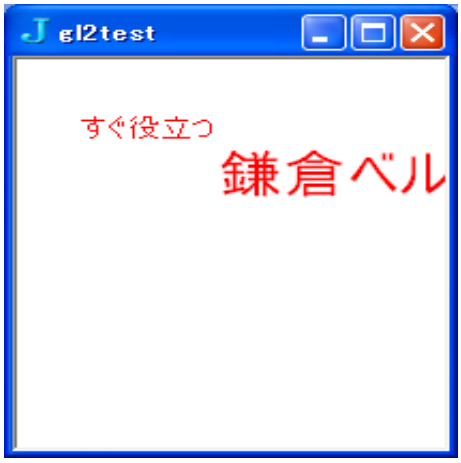

```
test_line1=: 4 : 0
glrgb x   NB. 255 0 0
glpen 3, PS_SOLID
gllines y NB. newdata 20
)
newdata=: 3 : ',(20 * ,.i.y),. 20 + ?y?200' NB. newdata
```

関数　.

run　(255 0 0)&test_line1 run 10 10 35 70 180 190

データ　x, y, x, y, x, y

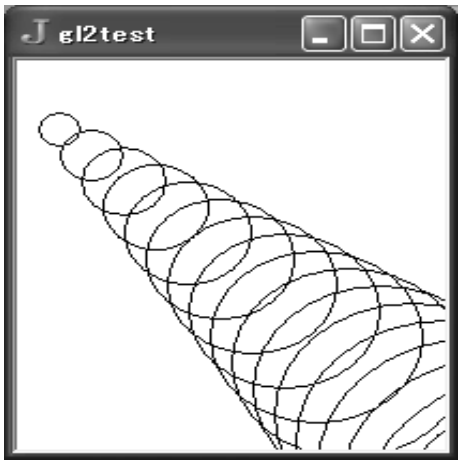
2.3 text を画面に出す



test_text1 run ”

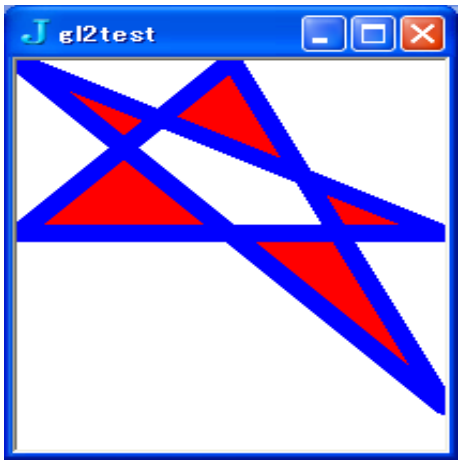
```
  test_text1=: 3 : 0
glfont 'arial 12'
glrgb 255 0 0
gltextcolor ''
pos=. 30 30
gltextxy pos
t=. 'すぐ役立つ'
gltext t
fwh=. glqextent t
gltextxy pos + fwh
glfont 'arial 24'
gltext '鎌倉ベルの会お勧めレシピ'
)
```

2.4 楕円




```
test_ellipse=: 3 : 0
glclear''
xywh=. 10 30 20 20
glellipse xywh (+"1 0) 10*i.50
)
```

2.5 polygon




```
test_poly1=: 3 : 0
glrgb 255 0 0
glbrush ''
glrgb 0 0 255
glpen 10 ,PS_SOLID
glpolygon y NB. 0 0 200 200 100 0 0 100 200 100
)

POLYDATA=: 0 0 200 200 100 0 0 100 200 100
```

データの形　$(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)$
連結　最後の (x_n, y_n) と最初の (x_0, y_0) は自動で連結する

3　1000 号キャンバスのデモ（Example)

この項の多くのグラフィックスはＪ６０２搭載のデモを単独で動くように修正を加えたものである。

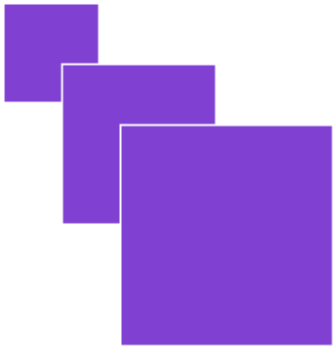
3.1　キャンバスの大きさ

- J の ISIGraph のオリジナルは左上が 0,0 で、右下が 1000,1000 のピクセルで描画する。
- J のデモ (isview.ijs) が使用するキャンバス（画面サイズ）は使いやすい 220 × 200 で、左上が (0,0)、右下が (220,200)
  xywh 0 0 220 200;cc g isigraph rightmove bottommove;
- とりあえずのプリアンプル
  - require 'gl2 numeric trig graph'
    coinsert 'jgl2'
  - J/system/examples のファイル
    require 'system/examples/graphics/isigraph/isview.ijs'
    require 'system/examples/graphics/isigraph/iscolor.ijs'
- 最も簡単な自作の画面用ツール

```
test_gutil=: 3 : 0
gopen''
gclear''
)
```

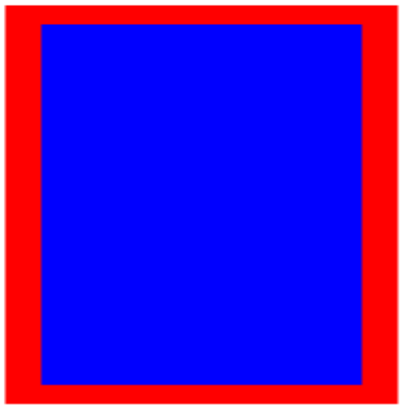
3.2　rect

```
rect0=: 3 : 0
NB. u ''
test_gutil''
gbrush 128 64 210
xywh=. 10 30 50 50
glrect xywh
glrect xywh+30
glrect xywh+60
)
```



rect0 ”

```
rect1=: 3 : 0
NB. u ''
test_gutil''
glrgb 0 0 255
glbrush ''
glrgb 255 0 0
glpen 10,PS_SOLID
NB. red pen, 10 pixels wide
glrect 10 30 100 200
)
```



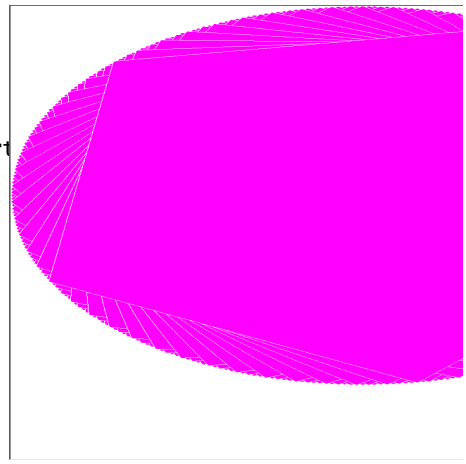
rect1 ”

パラメーターを指定して長方形を描くツールの例

```
draw_rect=: 4 : 0
NB. 255 0 255  draw_rect 10 30 100 200
test_gutil ''
gbrush x       NB. green
glrect y NB. 10 30 50 100 NB. rectangle - default pen, green brush
)
```

3.3 polygon

```
spinner=: 3 : 0
NB. y (e.g.1000)
rot=. 1r36p1 & rotate
start=. polygon 5
turns=. y    NB. e.g. 1000
dat=. ,"2 roundint gscale rot^: (i. y) start
NB. ------------------------------
test_gutil ''
gbrush 255 0 255 NB. RGB
while. #dat do.
  glpaint glpolygon {.dat
  dat=. }.dat
end.
)
```



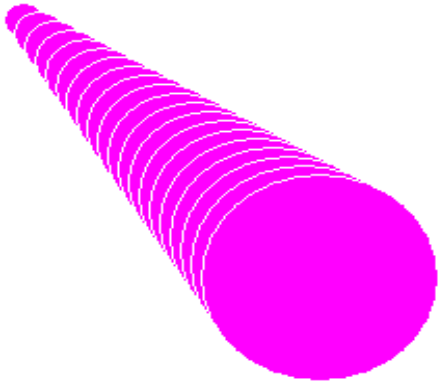
3.4 text

```
title1=: 3 : 0
NB. u ''
test_gutil ''
glfont'arial 12 bold italic'
glrgb 255 0 0
gltextcolor''
gltextxy 50 50
gltext 'how now brown cow'
glrgb 0 255 0
gltextcolor ''
gltextxy 100 100
gltext 'JALPA'
glrgb 0 0 255
gltextcolor ''
gltextxy 150 150
gltext 'あさき夢みし'
)
```

title1 ”

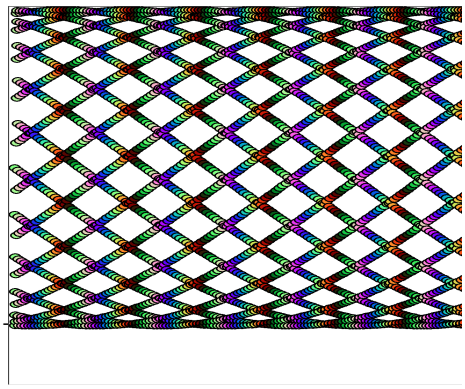
3.5 ellipse

```
ellipse0=: 3 : 0
NB. u ''
test_gutil''
glrgb  255 0 255  NB. 64 210
glbrush ''
NB. ----------
xywh=. 10 30 20 20
glellipse xywh (+"1 0) 5*i.20
)
```



ellipse0 ”

```
sine=: 3 : 0
test_gutil ''
glpen 3,PS_SOLID
SIZE=. 21 21
X=: steps 0 1000 3334
fn=: glellipse@[ gbrush
pos=: (X ,. gscale sin X) ,"1 SIZE
clr=: 255 <. 128 * 1 + (sin ,. cos,. sin @
pos fn"1 clr
)
```



sine ”

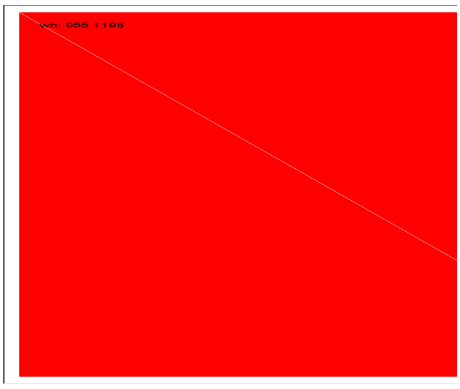
3.6 line

```
line0=: 3 : 0
NB. u ''
test_gutil''
glrgb  255 0 255
glpen 2,PS_SOLID
gllines 20 20 80 130 150 50
)
```

line0 ”

3.7 mixed

```
mix_test=: 3 : 0
test_gutil ''
wh=.glqwh''
glrgb 255 0 0
glbrush''
glrect 20 20,wh-40
gllines 20 20,wh-20
glfont 'arial 12'
gltextxy 50 50
gltext 'wh: ',":wh
)
```



3.8 簡易動画

副詞として描画関数をループで動かせば簡易動画となる。

line1 animate 100

```
animate=:1 : 0
NB. line1 animate 100
D=: newdata 20
test_gutil''
for. i. y do.
 u y
 glpaint'' NB. update screen
end.
)
line1=: 3 : 0
NB. u 10
test_gutil''
glrgb 0 255 0
glpen 3,PS_SOLID
gllines newdata y
)
newdata=: 3 : ',(20*,.i.y),.20+?y$200'
D=: newdata 10
```

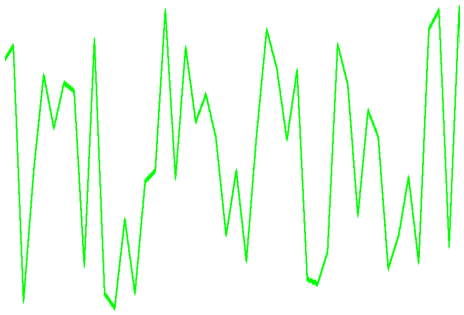


図 1　rect1

4　１号キャンパス

4.1　アウトライン

最初に　require 'graph'

classes/graph/graph.ijs で定義されている

tacit/**正調**　ijs 画面に gdopen → gdshow を一行づつ書く

- A

```
gdopen ''                  NB. open graph window
gdcolor GREEN              NB. set color
gdrect _0.4 _0.4 0.8 0.8   NB. define rectangle
gdshow''                   NB. show it
```

- B

```
 gdopen ''                   NB. open graph window
YELLOW gdrect _0.4 _0.4 0.8 0.8  NB. define rectangle
gdshow''                   NB. show it
```

2 行で書く　. gdopen で画面の下準備（現れない）をして gdshow で表示する

```
gdopen 'def'
  gdshow TEAL gdpolygon ,0.7 * (cos,.sin) 2p1 * int01 5
```

一行で書く　gddraw を用いて一行で済ます

- rect

```
 RED gdrect gddraw _0.4 _0.4 0.8 0.8

   (RED,:GREEN) gdrect gddraw _0.4 _0.4 0.8 0.8,:0.4 0.4 0.2 0.5 NB. draw 2 rect
```

- rect01 .
  BLUE gdrect01 gddraw 0.3 0.3 0.4 0.4
- polygon
  TEAL gdpolygon gddraw , 0.7 * (cos,.sin) 2p1 * int01 5
  255 0 255 gdpolygon gddraw , +. r. 2p1 * (i.5)%5

gl2 の gd 関数　gl2 のキャンバス 1 号、2 号の関数は gdxx で定義されている。後ろに 01 をつけると 1 号用になり (e.g. gdlines01)、付けないと 2 号用である
classes/graph/graph.ijs で定義された関数

```
gdarc       draw arc
gdchord     draw chord
gdellipse   draw ellipse
gdlines     draw lines
gdpie       draw pie-shaped wedge
gdpixel     draw pixel
gdpolygon   draw polygon
gdrect      draw rectangle
gdroundr    draw rounded rectangle
gdtext      draw text
```

```
-------------------------------------
gdadd       wrap gd add drawing command
gdbmp       save to bitmap file
gdclip      save to clipboard (windows only)
gddraw      wrap gd drawing command
gdemf       save to emf file (windows only)
gdopen      open/clear graphics window
gdshow      show graph
```

4.2　カンバス 1,2 号のデモ (Example)

Ｊ 6 の isigraph のオリジナルは左上が 0,0 で、右下が 1000,1000 のピクセルで描画する。
Ｊ 4,J5 の isigraph のオリジナルは左下が 0,0 で、右上が 1000,1000 であった。

号数	1000 号	２号	１号
xy	0,0/1000,1000	0,0/2,2	-1,-1/1,1

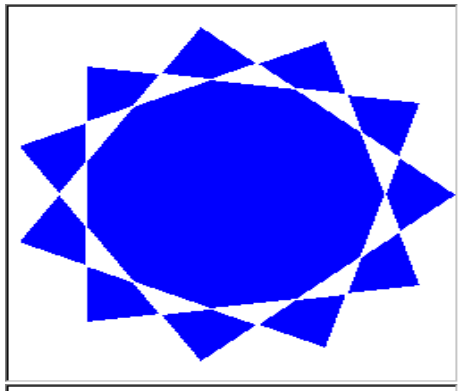
今回のキャンバスのサイズは１，２号であり、簡易グラフィックスを用いる。
１，２号キャンバスは次の 150 × 150 のキャンバスを用い図形データの方を調整している
(gopen) のキャンバス
wd 'xywh 0 0 150 150;cc g0 isigraph rightmove bottommove'

(isigraph.ijs)

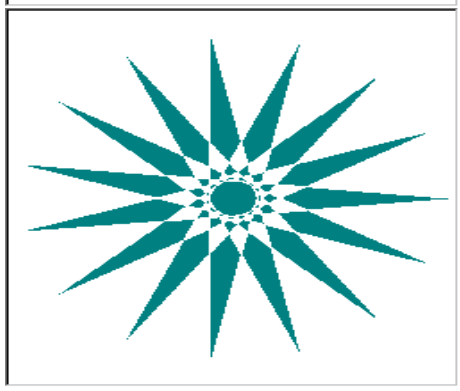
```
NB.*gfit v fit data to graphics window
NB. scales each column of data independently to range (0,1000)
gfit=: 3 : 0
min=. <./y
max=. >./y
(y-"1 min)*"1 [ 1000%max-min
)

NB. gscale      scale from range (_1,1) to (0,1000)
```

4.3 Polygon

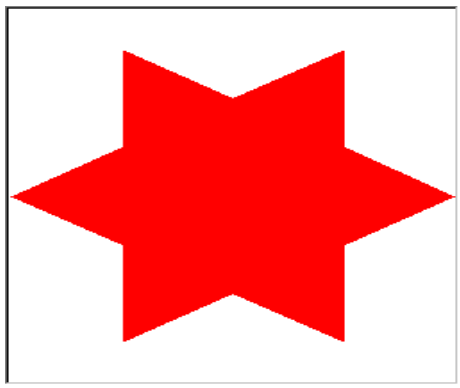
```
polygon_11=: 3 : 0
gdopen ''
BLUE gdpolygon ,+. r.6p1 *int01 11
gdshow''
)
```



```
polygon_12=: 3 : 0
TEAL gdpolygon gddraw ,7 polygon 15
)
```



```
  polygon_13=: 3 : 0
  poly13_calc ''
  RED gdpolygon gddraw ,refine ^:1 p
  )
```



```
poly13_calc=: 3 : 0
   tri1=: 2r3&*@[ + 1r3&*@]           NB. one third point
   tri2=: 1r3&*@[ + 2r3&*@]           NB. two thirds point
   mid=:  -:@+                        NB. midpoint
   nor=: _1 1&*@|.                    NB. left normal vector
   bulge=: mid + (%:%12)&*@nor@:-     NB. bulge point
   segdiv=: [ , tri1 , bulge ,: tri2
   refine=: ,/@(] segdiv"1 (1&|.))
   p=: |:2 1 o./ 2r3p1*i.3   NB. make a triangle
   )
```

```
 polygon_14=: 3 : 0
RED gdpolygon gddraw,reine ^:4 p
)
```

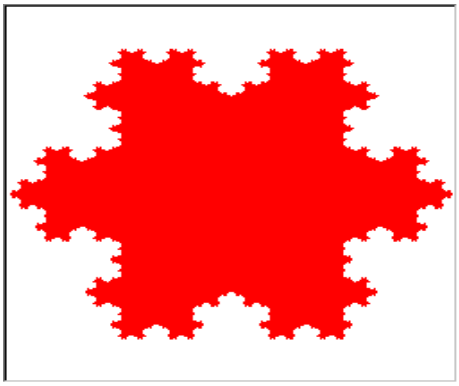


図 2　polygon

5　ピクセル

ピクセル, ビットマップ, ラスター (走査線）系のグラフィックスを取り扱う

J のデモは Studio　→ LAB → Grapgics → Viewmat に入っている。 Ctrl + J で進めながら観賞しよう

```
polygon_15=: 3 : 0
a=.(,refine ^:4 p);,refine ^:1 p
(BLUE;RED) gdpolygon gddraw a
)
```

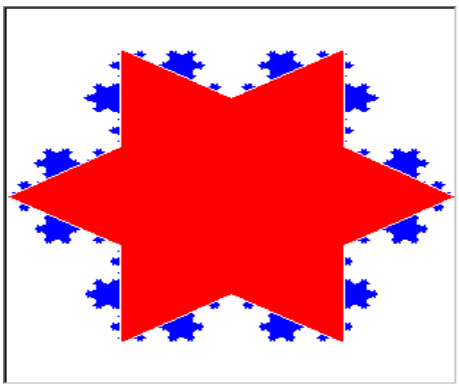
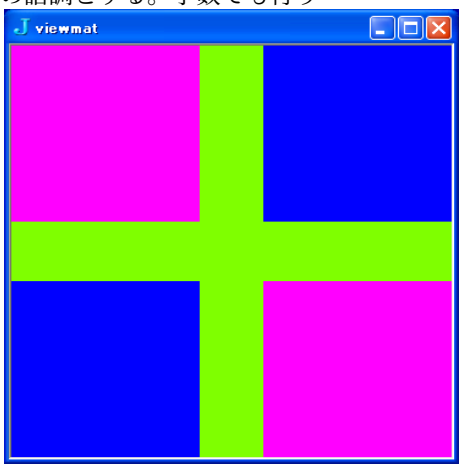


図 3　polygon

5.1　Viewmat

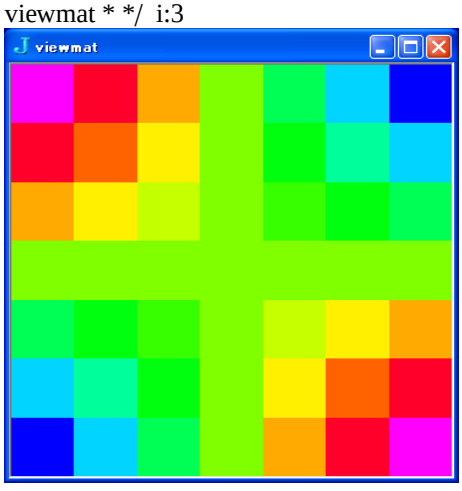
メモリーバッファの画像化　メモリーバッファが 0/1 の場合は各ピクセルを白黒（又は指定した 2 色）で表す。
有理数の場合　適当に同じものを同じ色とし近い数は類似の諧調とする。小数でも行う

```
     * */~ i:3
 1  1  1 0 _1 _1 _1
 1  1  1 0 _1 _1 _1
 1  1  1 0 _1 _1 _1
 0  0  0 0  0  0  0
_1 _1 _1 0  1  1  1
_1 _1 _1 0  1  1  1
_1 _1 _1 0  1  1  1
```



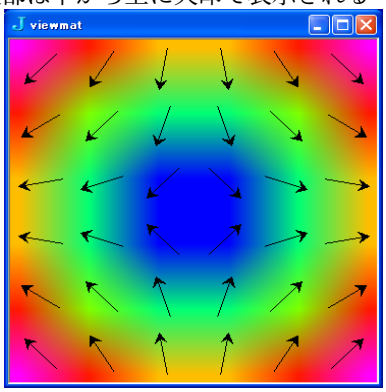
viewmat * */ i:3

```
   */~ i:3
 9  6  3 0 _3 _6 _9
 6  4  2 0 _2 _4 _6
 3  2  1 0 _1 _2 _3
 0  0  0 0  0  0  0
_3 _2 _1 0  1  2  3
_6 _4 _2 0  2  4  6
_9 _6 _3 0  3  6  9
```



viewmat */ i:3

複素数　複素数はベクトル場を矢印で表す。実数部は左から右に、複素数部は下から上に矢印で表示される




```
   |.  j.~/~ i:2j5
     _2j2    _1.2j2    _0.4j2    0.4j2    1.2j2    2j2
   _2j1.2  _1.2j1.2  _0.4j1.2  0.4j1.2  1.2j1.2  2j1.2
   _2j0.4  _1.2j0.4  _0.4j0.4  0.4j0.4  1.2j0.4  2j0.4
  _2j_0.4 _1.2j_0.4 _0.4j_0.4 0.4j_0.4 1.2j_0.4 2j_0.4
  _2j_1.2 _1.2j_1.2 _0.4j_1.2 0.4j_1.2 1.2j_1.2 2j_1.2
    _2j_2   _1.2j_2   _0.4j_2   0.4j_2   1.2j_2   2j_2
```

リサイズ　Viewmat の画像はマウスでサイズを変更できる

色指定　色の指定は 1 色ごとに RGB で行う

(”. COLOR16) viewmat i. 16

```
   ".  COLOR16
  0 255 255
  0   0   0
  0   0 255
255   0 255
128 128 128
```

RGB **の合成と分解**　基底を用い、256 進法で演算する。

```
    256 256 256 #. 0  255  0  NB. 一つの数値に変換
65280

   256 256 256 #: 65280       NB. 戻す
0 255 0
```

5.2　viewrgb と viewbmp

readbmp　ビットマップファイルを読む。a は RGB の数値が得られ、256 進の基底逆変換で RGB 情報に戻せる

ここで扱えるのは bmp のみで、jpg は C.Reiter のアドオンを用いることとなる。

```
  a=: readbmp jpath '~system\examples\data\toucan.bmp'
```

viewrgb　ビットマップファイルを表示

viewbmp　ビットマップファイルを読んで表示する

References

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 OCT 2011