J-OpenGL グラフィックス(J6 版)－その９
－OOP 利用の OpenGL と Dixon の Mathographics－

西川　利男

１．J-OpenGL と OOP

これまで J-OpenGL グラフィックスをいろいろやってきた[1-8]。ところで、筆者は J6 版での要望があることは知りながら、J4 でプログラミングを行ってきた。その大きな理由は J6 版の OpenGL では数多くのインクルードされるプログラムが OOP(オブジェクト指向)を利用するものであり、そのため J-OpenGL のしくみがあらわにされず、ブラックボックス化するのをおそれるからである。

例えば、J6 の[Studio]-[Demo]には多数の OpenGL のデモプログラムが載っていて、簡単に実行してみることができる、また J のコーディングを見ることもできる。しかし、これを理解して、自分のものとして利用するのは容易ではない。そのバリアが OOP を利用の各種プログラムやいろいろな設定である。一方、J4 版の OpenGL では OOP を利用せず、一般の OpenGL と同じ仕様の命令を直接使用している。

なお、J における OOP についての基本の考え方についてはすでに発表した[9,10]。

今回の発表は J6 版の OpenGL のプログラミングを J4 版のものと比較して、特に OOP から解説する。また、最近 Dixon の Mathographics[11]なる楽しい本を見つけたが、その中からいくつかの課題をやってみた。

文献 ----------

[1] 西川利男「J の gl3-OpenGL によるグラフィックスーその１」
JAPLA 研究会資料　2009/9/26
[2] 西川利男「J の gl3-OpenGL グラフィックスーその２、正 8 面体と正 12 面体を動かす」JAPLA 研究会資料 2009/9/29
[3] 西川利男「J の OpenGL グラフィックスーその３、J602 版 OpenGL / サイコロの回転(cube)を例として」　JAPLA 研究会資料　2009/10/24
[4] 西川利男「J の OpenGL グラフィックスーその４、正 12 面体と正 20 面体との頂点座標の計算」JAPLA シンポジウム資料 2009/12/5
[5] 西川利男「J の OpenGL グラフィックスーその５, 正 12 面体と正 20 面体を動かす」JAPLA シンポジウム資料 2009/12/5
[6] 西川利男「J の OpenGL グラフィックスーその６、サッカーボールとその仲間たち」JAPLA 研究会資料 2010/1/23
[7] 西川利男「J の OpenGL グラフィックスーその７、フラー･ドームと照光表示－」JAPLA 研究会資料 2010/2/27
[8] 西川利男「J と OpenGL 思考からの線形代数の理解－その８、Gram-Schmidt の QR 過程の図表示と固有値の計算」JAPLA 研究会資料 2010/3/27
[9] 西川利男「J のオブジェクト指向プログラミング (OOP) － その１
J の OOP とは－簡単な例でやってみる」JAPLA シンポジウム資料　2005/12/10
[10] 西川利男「J のオブジェクト指向プログラミング (OOP) － その２
J のスプレッドシート(Grid)と数独パズルへの適用」JAPLA シンポジウム資料 2005/12/10
[11] Robert Dixon, “Mathographics”, Dover Pub. (1991).

２．J602 [Studio]-[Demo]-[OpenGL] / [Basic]-[Simple] のリスティング
オリジナルのコードと処理動作の説明

```
NB. Standalone version of opengl demo.
NB.
NB. This text can be cut and pasted into a single script.

require 'opengl'          NB. ライブラリ opengl(gl-命令) を取り込む
cocurrent 'gldemo'        NB. 現在のロケールを base から gldemo に変える
coinsert 'jzopenglutil'   NB. クラス jzopenglutil(gs-命令、GS-設定)を取り込む

gsetdefaults''            NB. デフォルト値(GS-値)をセットする。

OPENGL=: 0 : 0            NB. フォームの作成
pc opengl;
xywh 0 0 200 150;cc g isigraph opengl rightmove bottommove;
pas 0 0;
rem form end;
)

opengl_run=: 3 : 0        NB. フォームの実行
wd OPENGL
ogl=: '' conew 'jzopengl' NB. インスタンス ogl の作成
wd'pshow;'
)

opengl_close=: 3 : 0      NB. フォームの終了
destroy__ogl''
ogl=: ''
wd 'pclose'
)

opengl_cancel=: opengl_close  NB. 終了ボタン

paint=: 3 : 0                 NB. paint 処理
gsinit''                      NB. gsinit の実行、初期設定
gsdrawviewbox RED             NB. gsdrawviewbox 実行、赤色で輪郭を描く
gsdrawdodecahedron ''         NB. gsdrawdodecahedron 実行、正 12 面体を描く
gsfini''                      NB. gsfini 実行、終了処理
)

opengl_g_paint=: paint        NB. paint(上のコード)をチャイルド g で実行
opengl_g_char=: gschar        NB. 文字入力処理を gschar で行う
opengl_default=: gsdefault    NB. デフォルト処理を gsdefault で行う
```

opengl_run''
　上のプログラムをロードすると、即実行するのでデモを見ることだけはできる。
しかし、プログラムを理解するには、自分なりに、変更、改造して、コードを一行づつ、確かめることが大切である。

　まず、cocurrent 'gldemo'の行は、削除あるいはNB.として、通常のロケールbaseとして現在のJ OpenGL環境チェックする。
　require 'opengl'により、OpenGLと互換のgl命令、GL値が有効になる。
　coinsert 'jzopenglutil'ではJ特有のgs命令、GS値が可能となる。
これはnames ''により見ることができる。
　プログラムの全体の流れはまず、グラフィックのフォームを作成、表示する。そのチャイルドgとして、属性openglを持ったisigraphによりOpenGLのグラフィックスを実現する。つまりopengl_runを実行すると、OOP機能のconewによりインスタンスoglが作られ、ただちに次のプログラムが自動的に起動される。
　opengl_g_paint、opengl_g_size、opengl_g_char、opengl_close
そのイベントとして、いろいろな処理が行われる。
　opengl_g_paint … OpenGL図形の描画
　opengl_g_size　… OpenGLの大きさ
　opengl_g_char　… OpenGLへのキーコマンド入力
　opengl_close …… OpenGLの終了
　この中では、gsdrawviewbox、gsdrawdodecahedronなど多数のgs命令が使われている。便利ではあるがブラックボックス化してしまう。

　わかり易くするには、まずはOpenGL仕様のgl命令を直接使った、次のようなもっと簡単なもののので一歩づつ理解を進めた方が良い。

```
paint=: 3 : 0
gsinit''
glClearColor 1 1 1 0                                    NB. 背景を白色
glClear GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT       NB. バッファをクリア
glColor 1 0 0 1                                         NB. 図形表示を赤色
dat =: _1 0 0, 1 0 0, 0 _1 0, 0 1 0, 0 0 _1,. 0 0 1 NB. 簡単なX,Y,Z座標値
drawlines dat                                           NB. 下の関数により図形表示
gsfini''
)
drawlines =: 3 : 0                                      NB. 線表示の関数定義
glBegin GL_LINES                                        NB. OpenGLの仕様
glVertex y                                              NB. 座標値
glEnd ''
)
```

　また、プログラムスクリプトの最後にある
　opengl_run''
のような、即実行は削除してしまって、
実行画面上で、あらためてキー入力により実行する方がよい。

３．Dixonのデイジープログラム
(Mathographics, p.131のBASIC をＪに直したもの)

```
NB. OpGLN_Dixon.ijs - J602 version
NB.  2010/6/9 by T.Nishikawa
NB.  referred from [Demo]-[Basic]-[Simple] / [Demo]-[Basic]-[View]

require 'opengl'
coinsert 'jzopenglutil'

gsetdefaults''

OPENGL=: 0 : 0
pc opengl;
xywh 0 0 200 150;cc g isigraph opengl rightmove bottommove;
pas 0 0;
rem form end;
)

run =: opengl_run
opengl_run=: 3 : 0
wd OPENGL
ogl=: '' conew 'jzopengl'
S =: 0.5
T =: 0 0 _1
wd'pshow;'
)

opengl_close=: 3 : 0
destroy__ogl''
ogl=: ''
wd 'pclose'
)

opengl_cancel=: opengl_close

opengl_default=: gsdefault

NB. paint graphic picture ==================================
opengl_g_paint=: 3 : 0
gsinit''
glClearColor 1 1 1 0
glClear GL_COLOR_BUFFER_BIT
glTranslate T
glScale S, S, S
glPolygonMode GL_FRONT_AND_BACK, GL_FILL  NB. draw paint
```

```
NB. display Fermat's spiral -------------------------------
K =. 0.12
C =. 1.618034  NB. Fibonacci tau-value
D =. 2p1 % C
A =. 0
s =. 0
while. s < 200
  do.
      glColor (1, *: (s, s)%200 ),"(1) 0  NB. red gradually faded
      R =. K * %: A         NB. Fermat's spiral
      _6 drawpolygon R, A
      A =. A + D
      s =. s + 1
  end.

gsfontbmpw  'IJS';18
drawtext''
gsfini''
)

NB. make polygon as a model ---------------------------------
NB. sub program called from opengl_g_paint
NB. 6 drawpolygon R A  => start from 0
NB. _6 drawpolygon R A => start from pi/2

drawpolygon =: 4 : 0
glBegin GL_POLYGON
n =. x
'R A' =: y
X =. R * cos A
Y =. R * sin A
RR =. 0.1
i =. 0
while. i < | n
  do.
  AA =. (2p1 % n) * i
  if. n<0 do. AA =. AA + 1r2p1 end.
  XX =. RR * cos AA + A
  YY =. RR * sin AA + A
  glVertex (XX+X), (YY+Y), 0
  i =. i + 1
end.
glEnd ''
)
```

```
drawlines =: 3 : 0
glBegin GL_LINES
glVertex y
glEnd ''
)

NB. write parametes ---------------------------------------
drawtext =: verb define
NB. glMatrixMode GL_MODELVIEW
gscolor 0 0 0 0
glLoadIdentity ''
glTranslate 0 0 _0.01
NB. glColor 0 0 0 0
glRasterPos _0.12 _0.25 _1
glaCallLists  8j1 ": (}:T), S   NB. for J602
)

NB. project the picture on the screen ===================
opengl_g_size =: verb define
glViewport 0 0  , glqwh ''
glMatrixMode GL_PROJECTION
glLoadIdentity ''
glOrtho _2.5 2.5 _2.5 2.5 _2.5 2.5
)

NB. key-in commands =====================================
opengl_g_char=: verb define
NB. translate
T =: T + 0.1 * 'xyz' = 0 { sysdata
T =: T - 0.1 * 'XYZ' = 0 { sysdata
NB. zoom
S =: S + 0.1 * 'z' = 0 { sysdata
S =: S - 0.1 * 'Z' = 0 { sysdata
gspaint''
)
```

４．Dixon のデイジーー実行例