JAPLA 研究会資料 2015/12/5

# J-OpenGL による３Ｄグラフィックスーその１２
# ー朝顔の花とメビウスの帯ー

西川　利男

## ０．はじめに

もう５年以上前になるが、JAPLA の夏の蓼科合宿で、「Ｊによる花のグラフィックス」と題する発表をしたことがある。このときは、本来２次元のグラフィックスツールである J の gl2 を使って、擬似的に３Ｄグラフィックスを行った。

西川利男「Ｊによる花のグラフィックスー２ー朝顔ー」
JAPLA 研究会資料　2009/8/5 夏の蓼科合宿

これをきっかけにして、真の３Ｄグラフィックスである J の gl3、つまり OpenGL について、「J-OpenGL とは」、「正 12 面体、正 20 面体」、「サッカーボール、フラードーム」、「回転するサイコロ」と、数回にわたって、発表した。

しばらく時間が空いてしまったが、一昨年には「メビウスの帯」を題材に、また、昨年は「ユリの花」を OOP の OpenGL でという報告を行った。

西川利男「J-OpenGL による 3D グラフィックスーその 10
メビウスの帯へ向けてーJ-OpenGL をどう理解するかー」
JAPLA 研究会資料　20013/6/15

西川利男「J-OpenGL による花のグラフィックスーその 11
ユリの花ーOOP(オブジェクト指向)方式の J-OpenGL プログラム」
JAPLA 研究会資料　20014/9/20

今回さらに取り上げたのは、いままでのやり残しと、最初の原点に戻って、OpenGL で朝顔の花の３Ｄグラフィックスを完成したいとのことからである。これによりダイナミックに自由自在に回転し、自分の好きな方向からの花の観察が可能になる。

Ｊをどう利用するかは人によって異なるが、私にとっては高度の配列計算はもちろんだが、それ以上にコンピュータ上の視覚による表現が目的である。

とくに最新のグラフィックスとなると、ほとんどがブラックスボックス化したツールで行うのに対して、J-OpenGL ではユーザのきめ細かなプログラミングが可能である。これは私にとってはまたとない楽しみともなっている。

プログラミングする立場から見ると、J の OpenGL プログラミングは、通常の J のコーディングとはかなり違っている。

まず、図形の頂点座標を求める数学の計算処理では、J のプリミティブを使った通常の J の配列計算のコーディング・スタイルは変わらない。

一方、グラフィックス部分では、J での Windows 環境におけるグラフィックスのためのフォームの作成、マウス、キーボードからの入力操作、図形のディスプレイ表示などを OpenGL の仕様に合わせて行うので、これは J のプログラムというより、C など一般の Windows 言語のコーディングスタイルと似たものになる。

これらのプログラミングの実際を、頂点座標の計算、OpenGL グラフィックス
と分けて示していこう。

## １．花びらのりんかくの頂点座標の計算

基本的にはコンピュータグラフィックスでは、直交座標 XY で行う。しかし、極座標表示のほうが、考え方がすっきりすることも多い。そして、極座標から直交座標への変換は簡単にできるので、先月、先々月の渦巻き曲線など、せいぜい極座標でのグラフィックスも行ったらよい。

花のグラフィックスでは、立体の円柱座標（極座標と直交座標の混在）の表示が計算などで便利である。

また、朝顔などの花びらを、単純なラッパ状のものでなく自然な形にするためには、前にも用いた、戸川先生の関数を利用させていただいた。

戸川隼人著「花のＣＧ」、p.93-94、サイエンス社(1988).

円柱座標の動径 $r$ と高さ $h$ を $s$ などをパラメータとした次の式で表す。

$$r = R*(1.03-s)$$

$$h = H - const*s\text{^}5$$

ラッパから自然な花びらにするためのパラメータとして

$$\rho = 1 + R*|\sin 2.5\theta|$$

これらを使って、直交座標 $x$, $y$, $z$ は次のように求められる。

$$x = \rho * r(s)\cos\theta$$

$$y = \rho * r(s)\sin\theta$$

$$z = h$$

これに対するＪの計算プログラムは次のようになる。

```
asagao =: 3 : 0
MM =. 10
NN =. 40
PI =. 3.1416
ZMIN =. 0
ZMAX =. 180
POW =. 5
CONST =. (ZMAX-ZMIN)%MM^POW
Z0 =. ZMAX-CONST*((i.>:MM)^POW)
TT =. 2*PI*(i.>:NN)%NN
RHO =. 1 + 0.07*(|sin (2.5*TT))^0.2
RCT =. RHO*cos TT
RST =. RHO*sin TT
RZERO =. 70
RR =. RZERO*(1.03-(i.>:MM)%MM)

XL =: RR */ RCT
YL =: RR */ RST
ZL =: - ($XL)$(({:$XL)#Z0)
XYZ =: <"(1) (XL,"(0) YL),"(1 0) ZL
NFLOWER =. 0.02*L:0 XYZ
NFLOWER =: (0, 0, 2)+L:0 NFLOWER
)
```

関数 asagao を実行するとつぎのようにりんかく値が求められる。

分割点をMM =. 10、NN =. 40として、座標値はXL, YL, ZLとして計算され、

```
   $XL
11 41
   XL
72.1 75.3259 73.0497 68.6678 62.4132  54.495 ...
   YL
0 11.9305  23.7353 34.9881  45.346 54.4952 ...
   ZL
    _180      _180      _180      _180      _180      _180 ...
```

これらをXYZとボックス化してまとめる。

```
   XYZL
+----------+------------------+------------------+------------------
+-|72.1 0 _180|75.3259 11.9305 _180|73.0497 23.7353 _180|68.6678 34.9881
_180| .
+----------+------------------+------------------+------------------+-
```

値の大きさなどを調整して次のOpenGLに引き渡す値として、NFLOWERとする。

また、テストのためラッパ形のモデル立体を次のように作り、利用した。

```
cleanz =: * | >: 1e_10"_
NS =: 6
NT =: 3
SS =: stepn 0, 2p1, NS
TT =: stepn _1, 1, NT
makeflower0 =: 3 : 0  NB. calc. flower x, y, z values
's t' =. y.
x =. 5 * (% 5 - t) * sin s
y =. 5 * (% 5 - t) * cos s
z =. t + 1.5
NB. z =. t + 1
cleanz |: > x;y;z
)

NFLOWER0 =: |: makeflower0 L:0 { SS ; TT
   4j1 ": L:0 NFLOWER0
+----------+----------+----------+----------+----
| 0.0 0.8 0.5| 0.7 0.4 0.5| 0.7_0.4 0.5| 0.0_0.8 0.5| ...
+----------+----------+----------+----------+----
| 0.0 0.9 1.2| 0.8 0.5 1.2| 0.8_0.5 1.2| 0.0_0.9 1.2| ...
+----------+----------+----------+----------+----
| 0.0 1.1 1.8| 0.9 0.5 1.8| 0.9_0.5 1.8| 0.0_1.1 1.8| ...
|----------+----------+----------+----------+----
| 0.0 1.3 2.5| 1.1 0.6 2.5| 1.1_0.6 2.5| 0.0_1.3 2.5| ...
+----------+----------+----------+----------+----
```

## ２．J-OpenGLのプログラム

J-OpenGLはWindowsの環境でのグラフィックスであるので、Windowsの画面設定（ふつうフォームと呼ばれる）などを行った上で、OpenGLの書式でグラフィックスのプログラムを作成する。その概要を流れに沿って示す。

・require 'gl3' … J-OpenGLの基本定義関数を読み込む
load 'jzopengl'… J-OpenGLのライブラリを読み込む

・フォームの作成

```
A=: noun define
pc a closeok;
xywh 0 0 340 300;cc g isigraph ws_clipchildren ws_clipsiblings rightmove
)
```

・フォームの実行

```
a_run=: verb define
JOB =: y.
wd A
glaRC''
(いろいろな初期値の設定)
setfocus g
wd 'pshow;ptop'
)
```

・フォームの表示、大きさなど

```
a_g_size=:verb define
wh=.glqwh''
glViewport 0 0,wh
glMatrixMode GL_PROJECTION
glLoadIdentity''
gluPerspective 45, (%/wh),1 8
)
```

・フォームへのキーボード入力（キーイン・コマンド）

```
a_g_char =: verb define
R =: 360 | R + 2 * 'xyz' = 0 { sysdata
R =: 360 | R - 2 * 'XYZ' = 0 { sysdata
(途中省略)
glpaintx''
)
```

・フォームへの描画

```
a_g_paint =: verb define
   (途中省略)
  select. JOB
    case. 0 do. nflower NFLOWER0  NB. Flower Test Line
    case. 1 do.
           asagao ''             NB. Set Asagao Data
           nflower3 NFLOWER      NB. front=> fill, back=>line
    end.
    glaSwapBuffers ''
    )
```

・頂点座標の設定（a_g_paint の中から呼ぶサブプログラム）点をつなぐ

```
nflower =: 3 : 0
NF =. y.
glLineWidth 4
i =. 0
 while. i < #NF
   do.
    glBegin GL_LINE_STRIP        頂点をつなぐ
    glColor  >(NT|i){NCOLOR      色の指定
    glVertex >i{NF               頂点座標を取り出す
    glEnd ''
   i =. i + 1
end.
)
```

・頂点座標の設定（a_g_paint の中から呼ぶサブプログラム）

頂点座標を単純につなぐだけでなく、３角形の集合とみて、色わけをする関数として、上の関数を一部手直しして nflower3 を作った。

```
nflower3 =: 3 : 0
NF =. y.
'PN QN' =. $NF
glPolygonMode GL_FRONT, GL_FILL
glPolygonMode GL_BACK, GL_LINE
glLineWidth 2
glBegin GL_TRIANGLES
p =. 0
 while. p < (<:PN)
   do.
   q =. 0
   while. q < (<:QN)
     do.
     p1 =. p + 1
     q1 =. q + 1
     NF0 =. (<p, q)   { NF
     NF1 =. (<p1, q)  { NF
     NF2 =. (<p, q1)  { NF
     glColor 1 0 0 1
     glVertex > NF0, NF1, NF2
     NG0 =. (<p1, q)  { NF
     NG1 =. (<p1, q1) { NF
     NG2 =. (<p, q1)  { NF
     glColor 1 0 1 1
     glVertex > NG0, NG1, NG2
   q =. q + 1
   end.
  p =. p + 1
  end.
glEnd ''
```

```
   NB. X, Y, Z 座標軸を引く ====================
     glLineWidth 1
     glBegin GL_LINES
     glColor 0 0 0 1  NB. X-軸、黒
     glVertex 0 0 0
     glVertex 4 0 0
     glColor 0 0 1 1  NB. Y-軸、青
     glVertex 0 0 0
     glVertex 0 4 0
     glColor 1 0 0 1  NB. Z-軸、赤
     glVertex 0 0 0
     glVertex 0 0 4
     glEnd ''
)
```

プログラムの実行は引数としていろいろなJOBを指定して、例えば

```
  a_run 0  …  テストのモデル立体
  a_run 1  …  アサガオの立体
```

のようにして、実行する。

フォームaが起動すると、自動的にa_g_size、 a_g_char、a_g_paint が実行されて、グラフィックが描かれる。

その詳細は、つぎのようになる。JAB = 1のときには

```
    プログラム関数asagao
```

が実行され

```
    NFLOWER (アサガオのX, Y, Z値)が、計算される。
```

この値を引数として、

```
    OpenGLの関数nflower3
```

が実行され、

```
    a_paint
```

により、グラフィックスとして描画される。

この状態で、キーボードより文字x, SHTFT-x, y, SHIFT-y, z, SHIFT-z などがキーインされたときには、それをイベントとして図形はそれに応じて、回転する。

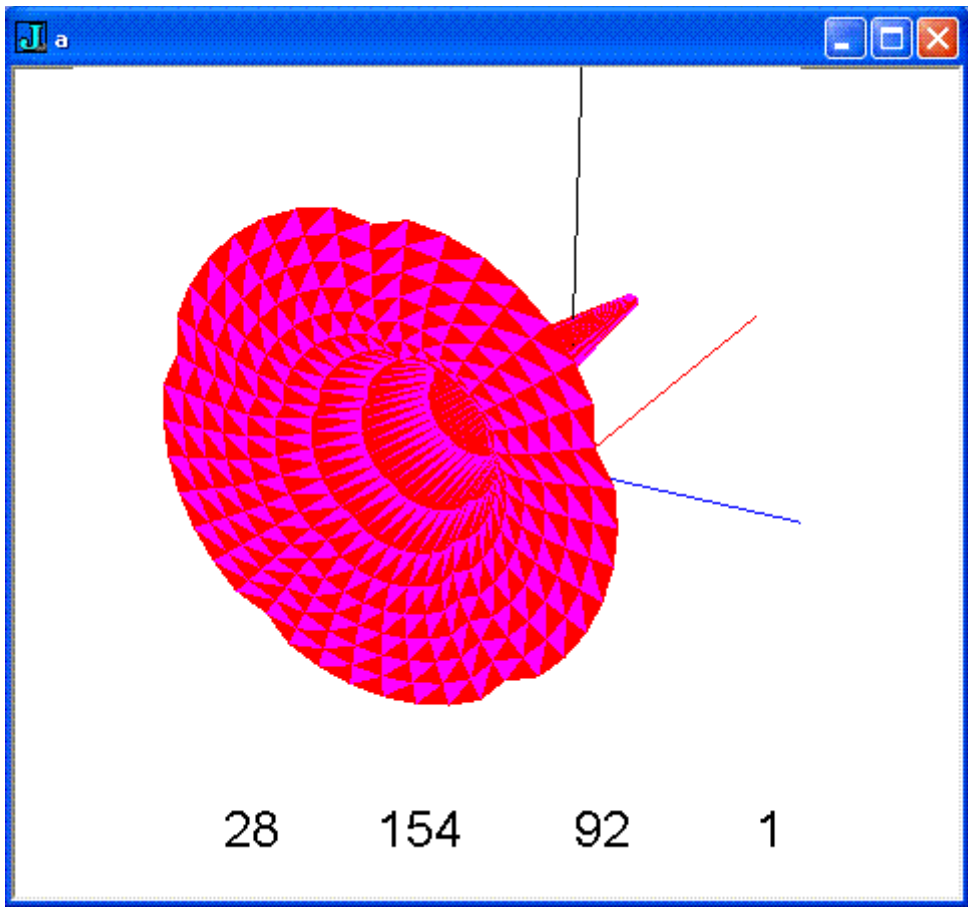
ここで、OpenGLの特徴的なことは

```
        glBegin GL_LINE_STRIP       頂点をつなぐ
        glColor  >(NT|i){NCOLOR     色の指定
        glVertex >i{NF              頂点座標を取り出す
        glEnd ''
```

頂点座標などの値の組は、起動の最初でコンパイルされて、それがキーインされたイベントのフォームの応じて、さまざまに画面が変わるのである。

## ３．アサガオ・グラフィックスの実行の実際

run 1




テストとして、
モデル立体の実行は

run 0