JAPLA 研究会資料 2010/2/27

Ｊの OpenGL グラフィックスーその７
ーフラー･ドームと照光表示ー

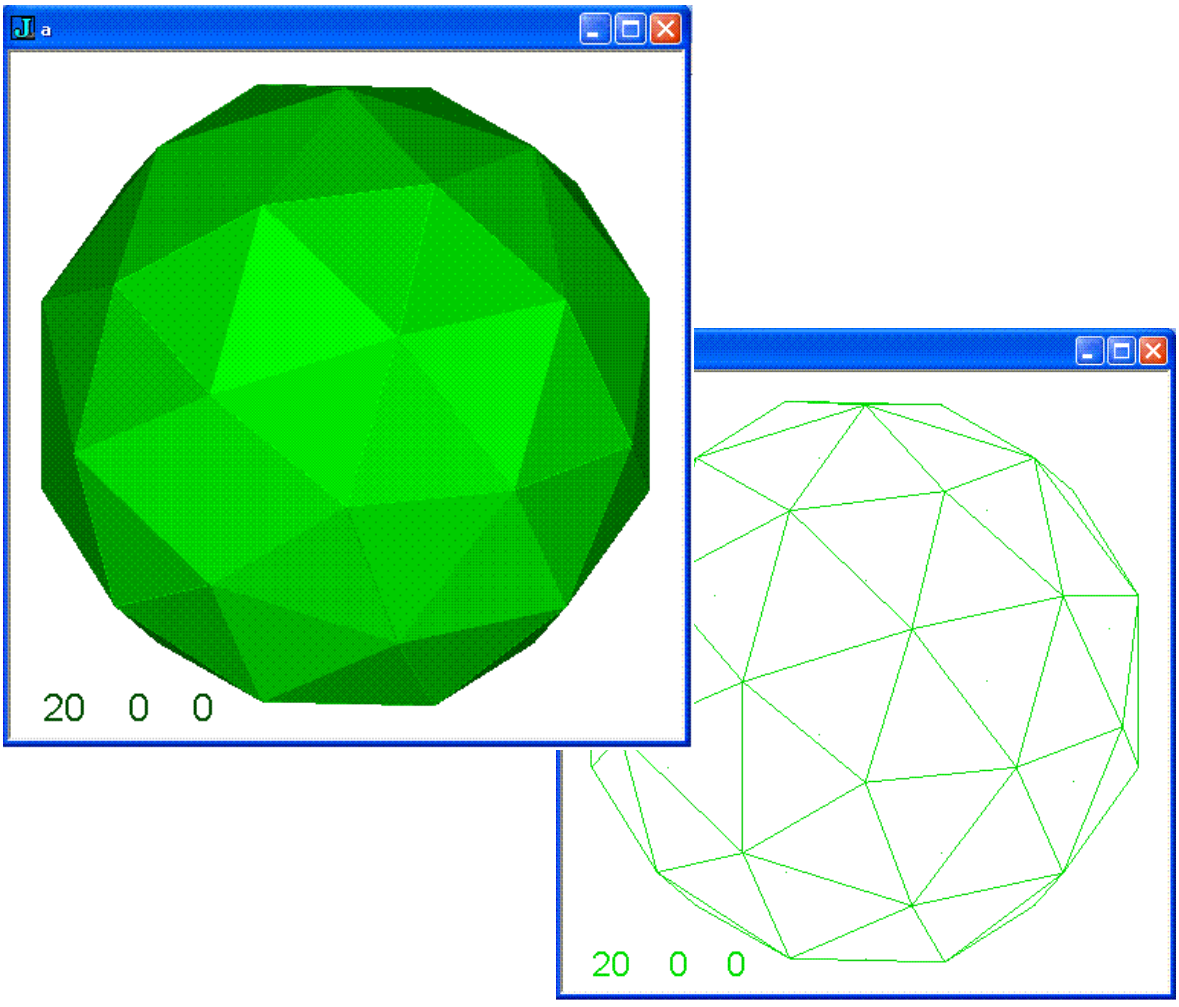
西川　利男

０．はじめに

OpenGL 正多面体グラフィックスとして、今回はフラー・ドームに挑戦してみた。バックミンスター・フラー(Richard Buckminster Fuller, 1895-1983)は多才な建築家、科学者、思想家として知られ、その名前を冠した(eponymous)フラー・ドームは世界各地の記念碑的建築物として話題を呼んでいる。

これは正 20 面体の面をさらに分割して出来た、一種の球面近似の立体であり、富士山頂のレーダー・ドームにその形を見た人も多いだろう。

このようなグラフィックスでは、各面を色で塗り分けるのはあまり実用的ではない。ふつうはそれぞれの面を明暗で示す照光表示で行う。これと境界線だけのワーヤーフレームの２種類のフラー・ドームのグラフィックスを比較して示す。



１．フラー・ドームの頂点座標

フラー・ドームを図示するには次のように行う。単位となる正20面体の正３角形の各頂点の中点をとり、これを正20面体の外接球まで膨らました点を新たに頂点として作る。これらの点を３つずつつないで正３角形を作る。このようにして、正20面体の１つの面から４つの正３角形ができる。つまり全部で８０の面から成る多面体ができる。これがフラー・ドームと呼ばれるものである。なお、もうこのときは多面体ではあるが、正多面体ではなくなる。

最初に出発点として正20面体の頂点座標が必要になるが、それにはJのlocale機能を用いて、前回のプログラム[1]をそのまま利用することができる。

[1] 西川利男「Jの0penGLグラフィックスーその４
－正12面体と正20面体の頂点座標の計算－」JAPLAシンポジウム資料2009/12/5

まず、次のように前回のプログラムファイルをロケール'polyh'に登録する。

```
'polyh' load 'user¥polyhedron.ijs'
```

すると、この中の動詞、名詞が自由に使えるので、頂点座標VICと連結インデックスINDICは次のように得られる。

```
VIC =: icosa_polyh_ 2
INDIC =: IND_ICO_polyh_
```

一般に、２つ点AとBとが与えられたとき、中点Dの座標は次のようにして求められる。

$$VD = \frac{1}{2}(VA + VB)$$ 、

ここで　VA、VB, VDは点A, B, Dのそれぞれ座標値(X, Y, Z)である。

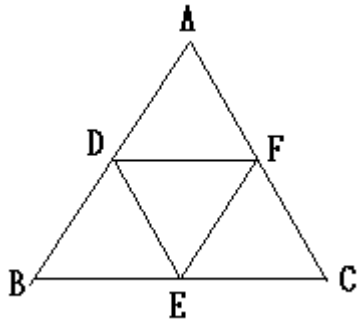
次に、この点を外接球まで延長するには

$$Ratio = \frac{D\text{から中心までの距離}}{\text{外接球の半径}} = \frac{D\text{から中心までの距離}}{A\text{から中心までの距離}}$$

なる比を掛けて正規化すればよい。これを正３角形ABCの各頂点に対して行う。

このようなフラー・ドームの頂点座標を得て、４つの正３角形を描くJの関数fullerは次のように定義される。

```
fuller =: 3 : 0
'VA VB VC' =. y.
R =. %: +/ *: VA
VD =. -: VA + VB
Ratio =: (%: +/ *: VD) % R
VD =. Ratio * VD
VE =. -: VB + VC
VE =. Ratio * VE
VF =. -: VC + VA
VF =. Ratio * VF
polygon >VA;VD;VF
polygon >VD;VB;VE
polygon >VF;VE;VC
polygon >VD;VE;VF
)
```



なお、ここで使用している関数polygonは前回も使用したが、複数の頂点から多角形を描く処理をOpenGLの書式に合わせて、まとめてコーディングするものである。

```
polygon=: 3 : 0
glBegin GL_POLYGON
y =. y.
n =. #y
i =. 0
while. i < n do.
 glColor 1 1 1 0
 glVertex > i { y
 i =. i + 1
end.
glEnd''
)
```

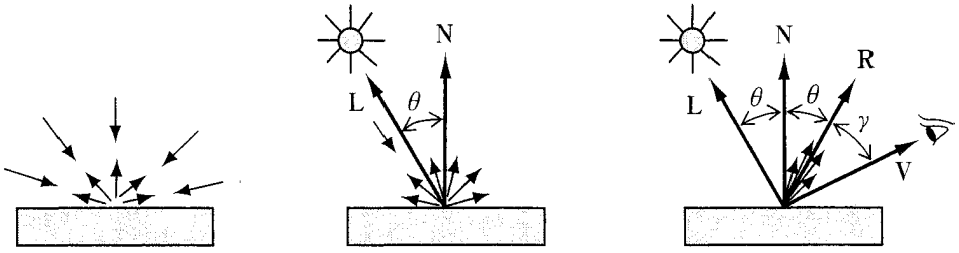
２．照光処理と法線ベクトル

OpenGLの照光(Lighting)表示とは次のような原理に基づいている。３次元表示された立体のある面に対して適当な光を当てると、光と面との傾斜角に応じて明暗に差が生ずる。これにより面の凹凸が明らかになり立体感があらわれる。

OpenGLの操作としては、レンダリングとモデリングの両方が必要である。
まず、レンダリングについて行う。

立体表面の明るさは次に図示したように３つの要素に分けられる。

(a) 環境光(ambient light)
(b) 拡散光(diffuse light)
(c) 鏡面光(specular light)



(a) 環境光　　**(b) 拡散光**　　**(c) 鏡面光**

そして、これはOpenGLのレンダリング・プログラムとして次の書式で示す。

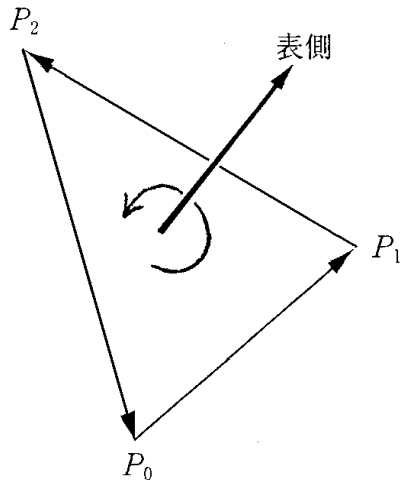
```
light=: verb define
glLight GL_LIGHT0, GL_AMBIENT, 0.1 0.1 0.1 1
glLight GL_LIGHT0, GL_DIFFUSE, 0.6 0.6 0.6 1
glLight GL_LIGHT0, GL_SPECULAR, 0.0 0.0 0.0 1
glLight GL_LIGHT0, GL_POSITION, 0.0 0.0 1.0 0   NB. Parallel Light
NB. glLight GL_LIGHT0, GL_POSITION, 4.0 6.0 10.0 1  NB. Positioned Light
glEnable GL_LIGHT0
glEnable GL_LIGHTING
glMaterial GL_FRONT,GL_AMBIENT_AND_DIFFUSE, 1 0 0 1
)
```

すなわち、オブジェクトGL_LIGHT0に対して、その方式を指定しコマンドglEnableで有効化する。またglEnable GL_LIGHTINGは照光処理の実行である。これらを立体の材質特性(Material)として、前方表面(GL_FRONT)に対して指定する。なお、GL_POSITIONの指定により平行光源、点光源の選択も可能である。

一方、モデリングとしては、モデルの立体面の向きを示してやらなくてはならない。そのためには面の法線ベクトルを利用することが必要になる。

法線ベクトルとは３次元上の３つの点P_0, P_1, P_2で定まる１つの平面に対して垂直なベクトルNである。これはベクトルを用いて、
ベクトル積として得られる。

$$N = (P_1 - P_0) \times (P_2 - P_0)$$



これを行うJの関数は次のようになる。

```
norm_vec =: 3 : 0
M =. > y.
NX =. -/ . * (<0 1;1 2) { M
NY =. - -/ . * (<0 1;0 2) { M
NZ =. -/ . * (<0 1;0 1) { M
NX, NY, NZ
)
```

ここで、上図のように３つの点を左まわり、右ネジの向きに取ったとき、正の法線ベクトルが得られるようにした。頂点の取り方の順序には注意が必要である。

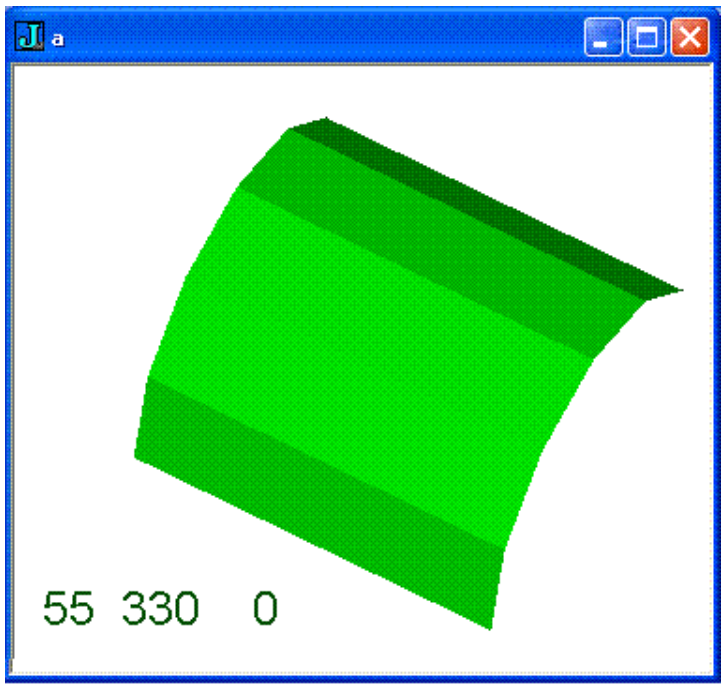
先に出した多角形表示の関数polygonは次のような関数polygonnormalに修正される。すなわち右引数は頂点の値を、左引数には上のnorm_vecで得られた法線ベクトルの値を入れる。するとOpenGL処理としてはこの法線ベクトルに対してglNormalを作用させることにより、照光処理を行ったモデリングプログラムとなる。

```
polygonnormal=: 4 : 0
glBegin GL_POLYGON
glNormal x.
NB. revised by TN.
y =. y.
n =. #y
i =. 0
while. i < n do.
 glVertex > i { y
 i =. i + 1
end.
glEnd''
)
```

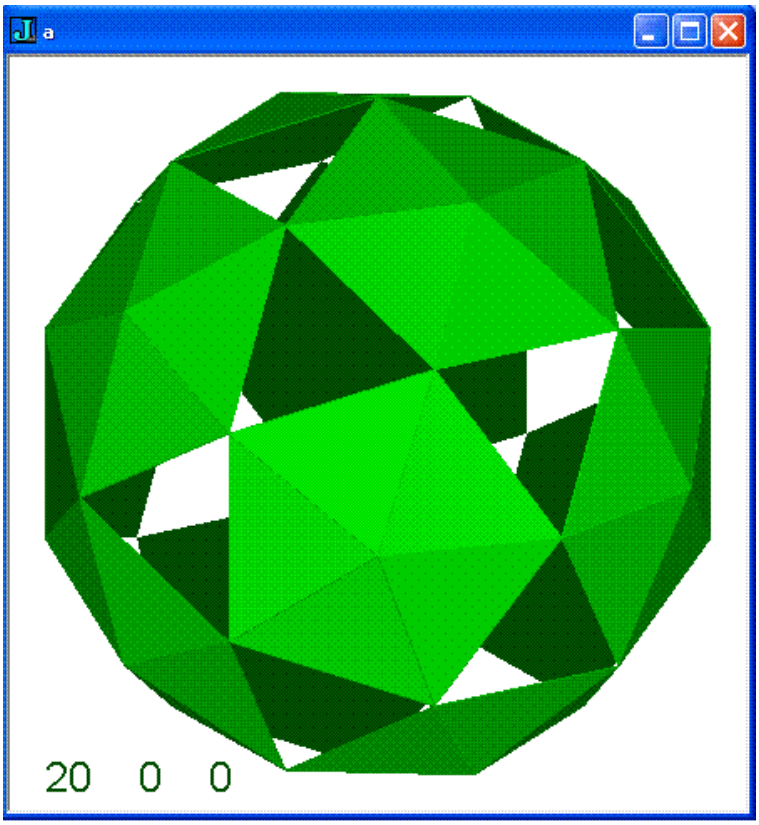
以上、レンダリングのlight、モデリングのpolygonnormalの２つの追加により
OpenGL照光処理のグラフィックスが実行される。

３．プログラムの実行例

OpenGL 照光処理のテストのために、ドラム状の立体を表示してみる。



フラー・ドームは最初のページに示したが、中央部分を除いたスケレトン表示は裏側のようすがわかり、これもなかなかおもしろい。



プログラム・リスト

```
NB. OpGLN_Fuller.ijs
NB.   2010/2/10 OK
NB. Fuller Dome with Lighting

NB. run '' => Fuller Solid
NB. run 0  => Fuller Skeleton
NB. run 1  => Fuller Wired
NB. run 2  => Icosahedron
NB. run 3  => Dodecahedron
NB. run 4  => Octahedron
NB. run 5  => Drum

require 'gl3'

A=: 0 : 0
pc a closeok;
xywh 0 0 220 200;cc g isigraph ws_clipchildren ws_clipsiblings rightmove
bottommove;
pas 0 0;
rem form end;
)

run=: a_run
a_run=: 3 : 0
T =: y. NB. Select '' = Fuller
wd :: ] 'psel a;pclose'
wd A
glaRC ''
LS =: 0      NB. LS=1: Fill LS<>1: Line
Hid =: 1
R =: 0 0 0
glaFont 'arial 30'
glaUseFontBitmaps 0 32 26 32
wd 'pshow;ptop'
)

NB. paint a model picture ===========================================
a_g_paint =: verb define
glClearColor 1 1 1 0
glClear GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT
glEnable GL_DEPTH_TEST
glMatrixMode GL_MODELVIEW
glLoadIdentity''
glTranslate 0 0 0   NB. for Ortho
```

```
NB. glTranslate 0 0 _10 NB. for Perspective
light ''
if. 0 = #T do. drawfuller '' end.
select. T
  case. 0 do. drawfuller ''
  case. 1 do. drawfuller ''
  case. 2 do. drawicosa ''  NB. Omitted Listing
  case. 3 do. drawdodec ''  NB. Omitted Listing
  case. 4 do. drawocta ''   NB. Omitted Listing
  case. 5 do. drawdrum ''
end.
drawtext''
glaSwapBuffers ''
)


NB. project the picture on the screen ====================
a_g_size =: verb define
glViewport 0 0  , glqwh ''
glMatrixMode GL_PROJECTION
glLoadIdentity ''
glOrtho _2 2 _2 2 _2 2  NB. Ortho
wh =: glqwh ''
NB. gluPerspective 90, (%/wh),1 30
)

NB. key-in x, y, z, X, Y, Z for rotation
a_g_char =: verb define
R =: 360 | R + 5 * 'xyz' = 0 { sysdata
R =: 360 | R - 5 * 'XYZ' = 0 { sysdata
NB. LS =: ('w' = 0 { sysdata) { LS, -. LS
NB. Hid =: ('h' = 0 { sysdata) { Hid, -. Hid
glpaintx''
)

NB. indicate rotated angle x, y, z in degree ============
drawtext =: verb define
glMatrixMode GL_MODELVIEW
glLoadIdentity ''
glColor 0 0 0 0
glRasterPos _2 _1.9 0
glCallLists 5 ": R
)
```

```
NB. Lighting & Normal Vector ============================================
norm_vec =: 3 : 0
M =. > y.
NX =. -/ . * (<0 1;1 2) { M
NY =. -  -/ . * (<0 1;0 2) { M
NZ =. -/ . * (<0 1;0 1) { M
NX, NY, NZ
)

light=: verb define
glLight GL_LIGHT0, GL_AMBIENT, 0.1 0.1 0.1 1
glLight GL_LIGHT0, GL_DIFFUSE, 0.6 0.6 0.6 1
glLight GL_LIGHT0, GL_SPECULAR, 0.0 0.0 0.0 1
glEnable GL_LIGHTING
glEnable GL_LIGHT0
glMaterial GL_FRONT,GL_AMBIENT_AND_DIFFUSE, 1 0 0 1
)

polygon=: 3 : 0
glBegin GL_POLYGON
y =. y.
n =. #y
i =. 0
while. i < n do.
 glColor 1 1 1 0
 glVertex > i { y
 i =. i + 1
end.
glEnd''
)

NB. Lighting version of polygon
polygonnormal=: 4 : 0
glBegin GL_POLYGON
glEdgeFlag GL_TRUE
glNormal x.
NB. revised by TN.
y =. y.
n =. #y
i =. 0
while. i < n do.
 glVertex > i { y
 i =. i + 1
```

```
end.
glEnd''
)
normpolygon =: 3 : 0
V =. y.
'VA VB VC' =. 3 {. V
(norm_vec (VB-VA);(VC-VA)) polygonnormal V
)


NB. Drum Graphic for Test Lighting ====================================
DS =: 0.5
DC =: 0.866

D0 =:  1, 0, 1
D1 =: _1, 0, 1
D2 =:  1, DS, DC
D3 =: _1, DS, DC
D4 =:  1, DC, DS
D5 =: _1, DC, DS
D6 =:  1, 1, 0
D7 =: _1, 1, 0
D8 =:  1, DC, -DS
D9 =: _1, DC, -DS
D10 =: 1, DS, -DC
D11 =: _1, DS, -DC
D12 =:  1, 0, _1
D13 =: _1, 0, _1

drawdrum =:verb define
glMatrixMode GL_MODELVIEW
glLoadIdentity ''
glClearColor 1 1 1 0
glClear GL_COLOR_BUFFER_BIT
glTranslate 0 0 0
glRotate R ,. 3 3 $ 1 0 0 0
glPolygonMode GL_BACK, Hid{GL_LINE, GL_POINT
glMaterial GL_FRONT,GL_AMBIENT_AND_DIFFUSE, 0 1 0 1  NB. Green

normpolygon D0;D2;D3;D1
normpolygon D2;D4;D5;D3
normpolygon D4;D6;D7;D5

normpolygon D6;D8;D9;D7
normpolygon D8;D10;D11;D9
```

```
normpolygon D10;D12;D13;D11
)


NB. import polyhedron.ijs ========================================
'polyh' load 'user¥polyhedron.ijs'
VIC =: icosa_polyh_ 2
INDIC =: IND_ICO_polyh_

NB. Fuller Dome ==================================================
drawfuller =:verb define
glMatrixMode GL_MODELVIEW
glLoadIdentity ''
glClearColor 1 1 1 0
glClear GL_COLOR_BUFFER_BIT
glTranslate 0 0 0
glRotate R ,. 3 3 $ 1 0 0 0
glPolygonMode GL_BACK, Hid{GL_LINE, GL_POINT
glMaterial GL_FRONT,GL_AMBIENT_AND_DIFFUSE, 0 1 0 1  NB. Green
I =. 0
while. I<20 do.
  fuller (>I{INDIC) { VIC
  I =. I + 1
end.
)

fuller =: 3 : 0
VV =. y.
R =. {. +/"(1) *: VV
'VA VB VC' =. VV
VD =. -: VA + VB
Ratio =: (%: R) % %: +/ *: VD
VD =. Ratio * VD
VE =. -: VB + VC
VE =. Ratio * VE
VF =. -: VC + VA
VF =. Ratio * VF
if. (0 = #T) +. (0 = T) do.
glPolygonMode GL_FRONT_AND_BACK, GL_FILL
normpolygon >VA;VD;VF
normpolygon >VD;VB;VE
normpolygon >VF;VE;VC
if. 0 = #T do. normpolygon >VD;VE;VF end.
end.
glPolygonMode GL_FRONT, GL_LINE
```

```
glPolygonMode GL_BACK, GL_POINT
polygon >VA;VD;VF
polygon >VD;VB;VE
polygon >VF;VE;VC
polygon >VD;VE;VF
)
```