

J の OpenGL グラフィックス—その 3 —J602 版 OpenGL / サイコロの回転(cube)を例として—

西川 利男

J 4 と J 6 の OpenGL

J の OpenGL グラフィックスは、本来の OpenGL とほとんど同じスタイルで、C 言語を用いることなく、J のコンパクトで強力な機能のもとでプログラミングが行なえるすばらしいグラフィックス環境である。その基本、正多面体の回転などについて前回報告した[1, 2]。

このときは、筆者が多年なじんでいる J 4 レベルで行なったが、ぜひ J 6 レベルでもとの要望から、サイコロの回転の処理を例とした J 6 版 OpenGL プログラムを作った。

J 6 版への変更の大きな違いはつぎのようである。

- J 4 では gl3 の命令 (gl..., gla..., glu...) を直接用いる。
- J 6 では、これに対してオブジェクト指向により行なう。

つまり、

```
coinsert 'jgl3'
```

として、クラスファイルを取り込み、

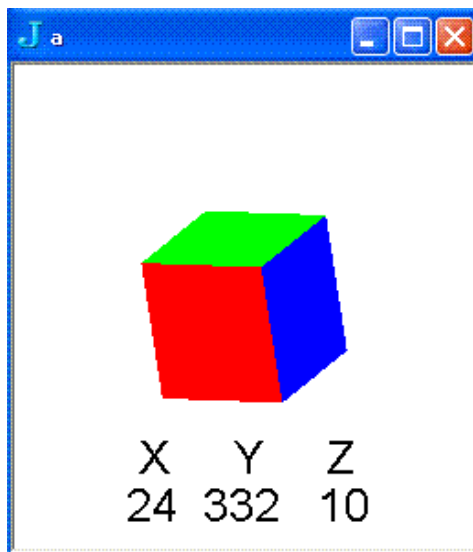
```
ogl=: '' conew 'jzopengl'
```

として、インスタンス ogl を設定し、その中の高度化した動詞、名詞により処理を行う。

これは、ユーザへの複雑な J のコーディングを軽減する効用はあるが、一方では処理の内容をブラックボックス化することにもなる。

また、クラス、インスタンスなどオブジェクト指向へのなじみも要請される。J のオブジェクト指向については、すでに発表したやさしい解説を参照されたい[3]。

筆者の個人的意見としては、これらを勘案した上で選択したらよいと思う。



文献

- [1] 西川利男「J の gl3-OpenGL によるグラフィックス—その 1」JAPLA 研究会資料 2009/9/26
- [2] 西川利男「J の gl3-OpenGL によるグラフィックス—その 2、正 8 面体と正 12 面体を動かす」JAPLA 研究会資料 2009/9/26
- [3] 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

以下、主要な箇所にコメントした実際のプログラミング・コードをあげる。
 NB. OpenGL_cube.ijs
 NB. サイコロの回転
 NB. referred from system¥examples¥graphics¥opengl¥lab¥ogl_cube.ijs
 NB. rotate => key-in: x/X, y/Y, z/Z
 NB. rotate angle showed

```
require'opengl gl3'
coinset'jgl3'
```

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

```
run=: a_run
a_run=: 3 : 0
wd A
ogl=: ''conew'jzopengl'      NB. インスタンス ogl の作成
R=: 0 0 0                    NB. 回転のパラメーターの初期値
wd'pshow;'
)
```

```
a_close=: 3 : 0
destroy__ogl''
wd'pclose'
)
```

```
a_g_paint=: 3 : 0
RC =: rc__ogl ''           NB. インスタンスにレンダラーの作成
if. RC do. g_draw_init wh__ogl end.
g_draw''
show__ogl''               NB. インスタンスの動詞 show を実行
)
```

```
a_g_char =: 3 : 0
wd'psel a'
R=: 360 | R + 2 * 'xyz' = {.sysdata NB. キーインにより R を変更
R=: 360 | R - 2 * 'XYZ' = {.sysdata
a_g_paint''
)
```

```
g_draw_init=: 3 : 0
```

```

glViewport 0 0, y
('arial';30) glUseFontBitmaps__ogl 32 95 32 NB. 文字フォント指定
glMatrixMode GL_PROJECTION
glLoadIdentity''
glOrtho _1 1 _1 1 _1 1 NB. 正規投影
NB. gluPerspective 30, (%/y), 1 10 NB. 射影投影
)

g_draw=: 3 : 0
glClearColor 1 1 1 0
glClear GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT NB. 陰線バッファークリア
glEnable GL_DEPTH_TEST NB. 陰線処理
glMatrixMode GL_MODELVIEW
glLoadIdentity''
glColor 0 0 0 0
glRasterPos _0.4 _0.7 0
glCallLists ' X Y Z'
glRasterPos _0.4 _0.9 0
glCallLists 5 ": R
glTranslated 0 0 0
glRotated R ,. 3 3 $ 1 0 0 0
NB. glPolygonMode GL_FRONT_AND_BACK, GL_LINE NB. Wired
drawbox ''
)

BLUE=: 0 0 1 0
GREEN=: 0 1 0 0
RED=: 1 0 0 0

drawbox=:3 : 0
p=: _1 ^ #: i.8 NB. 立方体の頂点座標の作成
p=. 0.3*p
BLUE polygon 0 1 3 2{p
GREEN polygon 0 1 5 4{p
RED polygon 0 2 6 4{p
(RED+BLUE) polygon 4 5 7 6{p NB. Violet
(RED+GREEN) polygon 1 3 7 5{p NB. Yellow
(BLUE+GREEN) polygon 2 3 7 6{p NB. Aqua
)

polygon=: 4 : 0 NB. 立方体表面の正方形の作成
glColor4d x
glBegin GL_POLYGON
glVertex y
glEnd ''

```

)