JAPLA 研究会資料 2010/4/24

Ｊの OpenGL グラフィックスーその８
Texture によるパターンの貼り付け
ーサイコロを作って動かすー

西川　利男

０．はじめに

これまで OpenGL グラフィックスもいろいろやってきた[1]~[7]。
OpenGL の Texture 機能を利用したパターンの貼り付け処理はさまざまな用途に使われる。今回はこれを利用してサイコロを作ってみた。なお、ファイル画像の貼り付けなどは次回に行う。

１．ビットマップ画像データの構造

OpenGL では頂点 Vertex を結んで作られた四角形などのモデルの面に画像データをパターンとして貼り付ける Texture という機能がある。

ここで、点をつないで直線を引いて出来た図形と画像データとは全く違うものであることをあらためて認識してもらいたい。

画像データのうち最も基本となるのはビットマップである。ビットマップとは元々は画像を構成する白黒の画素の１点ずつを VRAM の２進法の１ビットで表したことからきていて、ベクターイメージに対する用語である。現在では RGB のカラーの表示のため複数のビットを用いて、いろいろな形式がある。さらに転送、保存のためこれを圧縮した GIF 形式、JPEG 形式などがある。

OpenGL の Texture では画像データのうちビットマップ(Bitmap)形式のデータが貼り付けられる。

RGB のカラー表示の基本原理は次のとおりである。そしてこれらの RGB 値を組み合わせて、いろいろなカラー表示ができる。

	R	G	B			R	G	B
Red:	1	0	0		White:	1	1	1
Green:	0	1	0		Black:	0	0	0
Blue:	0	0	1		Yellow:	1	1	0

---

[1] 「Ｊの OpenGL によるグラフィックスーその１」Ｊ研究会資料 2009/9/26
[2] 「(同題)－その２、正８面体と正 12 面体を動かす」同報 2009/9/26
[3] 「(同題)－その３、J602 版 OpenGL/サイコロの回転」同報 2009/10/24
[4] 「(同題)－その４、正 12 面体と正 20 面体の頂点座標の計算」
同報　2009/12/5
[5] 「(同題)－その５、正 12 面体と正 20 面体を動かす」同報 2009/12/5
[6] 「(同題)－その６、サッカーボールとその仲間たち」同報 2010/1/23
[7] 「(同題)－その７、フラー･ドームと照光表示」同報 2010/2/27

２．ＪのOpenGLのTexture書式とコーディング

典型的なＪのOpenGLのTextureの起動部分のコードは次のようである。

```
texture=:verb define
glClearColor 0 0 1 0                    NB. 背景色青
glEnable GL_DEPTH_TEST                  NB. 陰線処理
glDepthFunc GL_LEQUAL
glPixelStore GL_UNPACK_ALIGNMENT, 1

glTexImage2D(GL_TEXTURE_2D, 0 3 64 64 0, GL_RGBA, GL_UNSIGNED_BYTE);
y.
          NB. ２次元Texture, y. 画像パターン
glTexParameter GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,GL_CLAMP
glTexParameter GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,GL_CLAMP
glTexParameter GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,GL_NEAREST
glTexParameter GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,GL_NEAREST
glTexEnv GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL
glEnable GL_TEXTURE_2D                  NB. ２次元Textureを有効化

glShadeModel GL_FLAT

glClear GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT

drawcube ''     NB.  画像パターンとモデル図形との結合
)
```

ごらんのようにかなりのコーディングである。しかし、必要とするところは数か所のパラメータである。

```
glTexImage2D(GL_TEXTURE_2D, 0 3 64 64 0, GL_RGBA, GL_UNSIGNED_BYTE);
y.
```

これにより２次元のTextureに対して、3ビットのRGBで符号なしバイトを用いて、画素64 x 64の画像パターンでおこなう。この関数の引数y.で画像パターンのデータを指定する。

実際のJ-OpenGLではピクセルの１つの点のカラー値に対して、次のような整数を用いておこなう。詳しい説明は次回にまわす。

```
White: _1
Black:  0
Red:   _16776961
```

まず、ＪのLabの簡単なデモで与えられているチェッカー模様でやってみる。

画像パターンとしてチェッカー模様はつぎのようにして作られる。

```
w=.8 8$_1
b=.8 8$0
```

```
pattern=: ,64 64$,    (8 64$,w,.b),8 64$,b,.w
```

つぎに画像パターンとモデル図形との結合について述べる。これは貼り付けパターンをモデル図形のどの頂点座標に対応させるか、という指定であり次の関数 drawcube で定義される。

```
drawcube =: 3 : 0
glBegin GL_QUADS
glTexCoord 0 0 0 1
glVertex _1 _1  1
glTexCoord 0 1 0 1
glVertex _1  1  1
glTexCoord 1 1 0 1
glVertex  1  1  1
glTexCoord 1 0 0 1
glVertex  1 _1  1
  (途中省略)
glEnd''
)
```

上の例では

パターンの始まり位置(0 0 0 1) をモデルの頂点座標(_1 _1 1)に、

ここでパターンの位置は(x, y)に z の 3 次元座標とアルファ値を付加したものを指定されていることに注意。

パターンの角の位置(0 1 0 1)をモデルの頂点座標(_1 1 1)に、

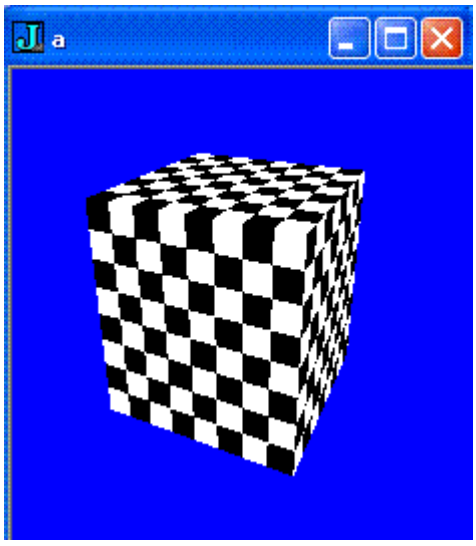
パターンの角の位置(1 1 0 1)をモデルの頂点座標(1 1 1)に、

パターンの角の位置(1 0 0 1)をモデルの頂点座標(1 _1 1)に、

結合させている。

これで 1 面の結合ができたので、後 5 面の結合を指定する。

これまで何回も述べてきた OpenGL の J プログラムに上のコーディングを追加すれば、Texture による貼り付けは行われる。実行すると次のようになる。



３．サイコロのパターンの作成

さて、それでは１から６までのサイコロ面の図形パターンを作ってみよう。まず、小さな、たとえば10 x 10の画素から成る画像でやってみる。この配列の各位置を示す一種の座標値Ｉ行Ｊ列のすべては次のようにして得られる。

```
   ] IJ100 =: 10 #.^:(_1) i.100
0 0
0 1
0 2
- -
9 7
9 8
9 9
```

一方、任意の座標値Ｉ行Ｊ列の要素を中心に半径Ｒの円の内部にあれば1、なければ0を返す関数tcは次のようになる。

```
tc =: 3 : 0
:
'I J R' =. x.
'X Y' =. y.
0 < ((%:2)*R) - (%: +/ *: (X-I), (Y-J))
)
```

これを使ってサイコロの目の図形パターンを作る関数diceを次のように定義する。

```
dice =: 3 : 0
:
'I J R' =. x.
N =. y.
IJN =. N #.^:(_1) i. *: N
NB. ((N, N)$ (I, J, R) tc"(1) IJN) { _1 0
(N, N)$ (I, J, R) tc"(1) IJN
)
```

例えば、10x10の配列で、４行、５列を中心に半径２の円は次のようになる。

```
   4 5 2 dice 10
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 1 1 0 0
0 0 0 1 1 1 1 1 0 0
0 0 0 1 1 1 1 1 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

この配列の(0, 1)値によりカラー値を指定すれば、塗りつぶし円をえがくことが出来る。
実際は64 x 64の配列を用いて、いろいろな場所に塗りつぶし円を作ってサイコロの１から６の目を作った。

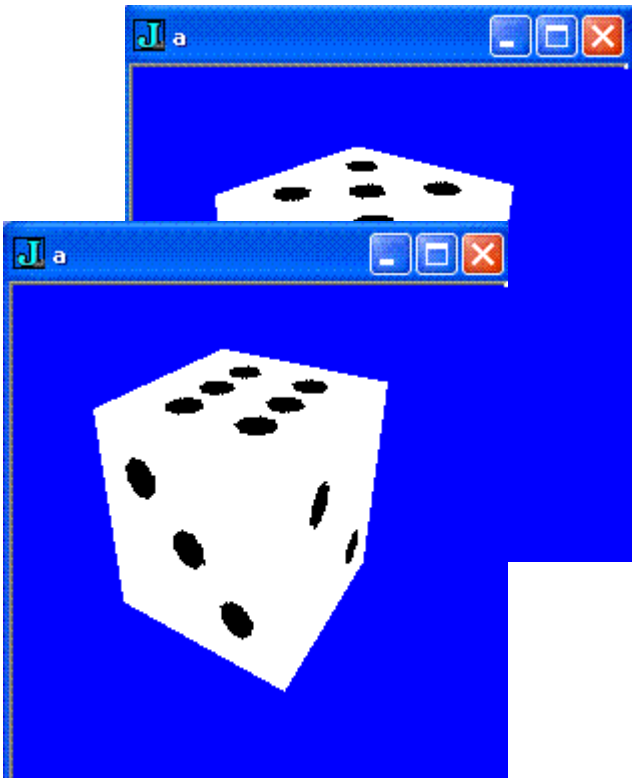
```
pat_d1 =: , (32 32 6 dice 64) { _1 _16776961  NB. Red in White

pat_d30  =: 16 16 4 dice 64
pat_d31  =: 32 32 4 dice 64
pat_d32  =: 48 48 4 dice 64

pat_d2   =: , (pat_d30 +. pat_d32) { _1 0     NB. Black in White
pat_d3   =: , (pat_d30 +. pat_d31 + pat_d32) { _1 0

pat_d40  =: 16 16 4 dice 64
pat_d41  =: 16 48 4 dice 64
pat_d42  =: 48 16 4 dice 64
pat_d43  =: 48 48 4 dice 64

pat_d4   =: , (pat_d40 +. pat_d41 +. pat_d42 +. pat_d43) { _1 0
pat_d5   =: , (pat_d40 +. pat_d41 +. pat_d42 +. pat_d43 +. pat_d31)
  { _1 0

pat_d60  =: 16 20 4 dice 64
pat_d61  =: 16 44 4 dice 64
pat_d62  =: 32 20 4 dice 64
pat_d63  =: 32 44 4 dice 64
pat_d64  =: 48 20 4 dice 64
pat_d65  =: 48 44 4 dice 64

pat_d6    =: , (pat_d60 +. pat_d61 +. pat_d62 +. pat_d63 +. pat_d64 +.
  pat_d65) { _1 0
```

上の各値pat_d1からpat_d6がサイコロの１から６の目のパターンとなる。
これをモデルの立方体の各面に、一面ずつ先に述べたOpenGLのTextureの起動処理texdrawと結合処理drawface1からdrawface6のプログラムに従い貼り付ければサイコロは出来上がる。なお、texdrawは副詞として定義した。

```
  1 drawface1 texdraw pat_d1
  drawface2 texdraw pat_d2
  drawface3 texdraw pat_d3
  drawface4 texdraw pat_d4
  drawface5 texdraw pat_d5
  drawface6 texdraw pat_d6
```

４．サイコロの実行例

いままでと同様、キーボードの x，X，y，Y，z，Z によりサイコロの位置を自由に回転できる。また、キーボードの g，G によりランダムな回転を行い、サイを振ることを実感できる。



５．あとがき

学術論文であれば、トライ・アンド・エラーでやっとたどり着いた過程、いわゆる足場をはずして完成した過程だけをエレガントに示すのがよいとされる。

J-OpenGL の理解には、J の Lab のデモプログラムを大いに参考にしたが、実はこれもエレガントに過ぎる。特にオブジェクト指向とやらで、その部分がブラックボックスになってしまう。私が J4 のような古いバージョンにこだわるのは、自分で理解して、自分のものとして思うように自由に使いたいからである。これにより時間はかかるが、知的好奇心を十分に満足させてくれた。

先日、亡くなった井上ひさし氏の言として、新聞に次のようなのが載っていた。

むずかしいことをやさしく　やさしいことをふかく
ふかいことをゆかいに　　ゆかいなことをまじめに
よみうり寸評　2010.4.12　から

まことに、もって銘すべきだと思う。

```
NB. Cube with Texture Selected Pattern
NB.  Checker Pattern 2010/2/17
NB.  Dice Pattern    2010/4/5
NB.  Dice Casting    2010/4/13

NB. run '' => display dice pattern
NB. run 0  => display checker pattern

require 'gl3'

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

run =: a_run
a_run=: verb define
T =: y.
wd A
glaRC''
R=: 0 0 0
wd 'pshow;ptop'
)

NB. Dice will be cast 2010/4/13, 4/17
a_g_char =: verb define
NB. if. press 'g' or 'G', then dice is cast
if. 'g' = 0 { sysdata do. R =: 90*(3?5) end.     NB. cast
if. 'G' = 0 { sysdata do. R =: 360 | 3?360 end.  NB. random

R=: 360 | R + 5 * 'xyz' = 0 { sysdata
R=: 360 | R - 5 * 'XYZ' = 0 { sysdata
glpaintx''
)
a_g_size=:verb define
glViewport 0 0,glqwh''
glMatrixMode GL_PROJECTION
glLoadIdentity''
NB. glOrtho _2 2 _2 2 _2 2
```

```
gluPerspective 80 1 1 100
)

a_g_paint=:verb define
glMatrixMode GL_MODELVIEW
glLoadIdentity ''
glTranslate 0 0 _5
glRotate R ,. 3 3$ 1 0 0 0
if. 0 = #T do.
  1 drawface1 texdraw pat_d1
  drawface2 texdraw pat_d2
  drawface3 texdraw pat_d3
  drawface4 texdraw pat_d4
  drawface5 texdraw pat_d5
  drawface6 texdraw pat_d6
else.
  select. T
   case. 0  do. texture pattern
  end.
end.
glaSwapBuffers''
)

NB. DrawTexture one face, one pattern
=======================================
NB. Adverb definition / left arg(u.): verb, right arg(y.): noun
NB.  revised 2010/4/7 adverb definition to verb(u.) with 2 noun
arguments(x. y.)

NB. first, clear buffer_bit => 1 drawface1 texdraw pattern1
NB. other, not clear buffer => drawfaceN texdraw patternN

texdraw =: 1 : 0
0 u. texdraw y.
:
glClearColor 0 0 1 0
glEnable GL_DEPTH_TEST
glDepthFunc GL_LEQUAL
glPixelStore GL_UNPACK_ALIGNMENT, 1

glTexImage2D (GL_TEXTURE_2D, 0 3 64 64 0, GL_RGBA, GL_UNSIGNED_BYTE);
y. NB. Select Pattern
```

```
glTexParameter GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,GL_CLAMP
glTexParameter GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,GL_CLAMP
glTexParameter GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,GL_NEAREST
glTexParameter GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,GL_NEAREST
glTexEnv GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL
glEnable GL_TEXTURE_2D
glShadeModel GL_FLAT
if. x. do. NB. when x.=1, clear buffer bit
  glClear GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT
end.
u. '' NB. execute verb as left argument u.
)


NB. Bind Texture Coordinates to Vertex
================================
drawface1 =: 3 : 0
glBegin GL_QUADS
glTexCoord 0 0 0 1  NB. face1
glVertex _1 1  1
glTexCoord 0 1 0 1
glVertex _1  _1  1
glTexCoord 1 1 0 1
glVertex  1  _1  1
glTexCoord 1 0 0 1
glVertex  1  1  1
glEnd''
)
drawface2 =: 3 : 0
glBegin GL_QUADS
glTexCoord 0 0 0 1  NB. face2
glVertex  1  1  1
glTexCoord 0 1 0 1
glVertex  1 _1  1
glTexCoord 1 1 0 1
glVertex  1 _1  _1
glTexCoord 1 0 0 1
glVertex  1  1  _1
glEnd''
)
```

```
drawface3 =: 3 : 0
glBegin GL_QUADS
glTexCoord 0 0 0 1  NB. face3
glVertex  1  1  _1
glTexCoord 0 1 0 1
glVertex  1 _1  _1
glTexCoord 1 1 0 1
glVertex  _1 _1 _1
glTexCoord 1 0 0 1
glVertex  _1  1 _1
glEnd''
)

drawface4 =: 3 : 0
glBegin GL_QUADS
glTexCoord 0 0 0 1  NB. face4
glVertex  _1  1 _1
glTexCoord 0 1 0 1
glVertex  _1 _1 _1
glTexCoord 1 1 0 1
glVertex  _1 _1  1
glTexCoord 1 0 0 1
glVertex  _1  1  1
glEnd''
)

drawface5 =: 3 : 0
glBegin GL_QUADS
glTexCoord 0 0 0 1  NB. face5
glVertex  _1  1 _1
glTexCoord 0 1 0 1
glVertex  _1  1  1
glTexCoord 1 1 0 1
glVertex   1  1  1
glTexCoord 1 0 0 1
glVertex   1  1 _1
glEnd''
)

drawface6 =: 3 : 0
glBegin GL_QUADS
glTexCoord 0 0 0 1  NB. face6
```

```
glVertex  _1 _1  1
glTexCoord 0 1 0 1
glVertex  _1 _1 _1
glTexCoord 1 1 0 1
glVertex   1 _1 _1
glTexCoord 1 0 0 1
glVertex   1 _1  1
glEnd''
)


NB. make dice pattern
================================================
IJ100 =: 10 #.^:(_1) i.100
IJ64 =: 8 #.^:(_1) i.64
IJ36 =: 6 #.^:(_1) i.36

tc =: 3 : 0
:
'I J R' =. x.
'X Y' =. y.
0 < ((%:2)*R) - (%: +/ *: (X-I), (Y-J))
)

dice =: 3 : 0
:
'I J R' =. x.
N =. y.
IJN =. N #.^:(_1) i. *: N
NB. ((N, N)$ (I, J, R) tc"(1) IJN) { _1 0
(N, N)$ (I, J, R) tc"(1) IJN
)

pat_d1 =: , (32 32 6 dice 64) { _1 _16776961  NB. Red in White

pat_d30  =: 16 16 4 dice 64
pat_d31  =: 32 32 4 dice 64
pat_d32  =: 48 48 4 dice 64

pat_d2   =: , (pat_d30 +. pat_d32) { _1 0     NB. Black in White
pat_d3   =: , (pat_d30 +. pat_d31 + pat_d32) { _1 0
```

```
pat_d40  =: 16 16 4 dice 64
pat_d41  =: 16 48 4 dice 64
pat_d42  =: 48 16 4 dice 64
pat_d43  =: 48 48 4 dice 64

pat_d4   =: , (pat_d40 +. pat_d41 +. pat_d42 +. pat_d43) { _1 0
pat_d5   =: , (pat_d40 +. pat_d41 +. pat_d42 +. pat_d43 +. pat_d31)
{ _1 0

pat_d60  =: 16 20 4 dice 64
pat_d61  =: 16 44 4 dice 64
pat_d62  =: 32 20 4 dice 64
pat_d63  =: 32 44 4 dice 64
pat_d64  =: 48 20 4 dice 64
pat_d65  =: 48 44 4 dice 64

pat_d6    =: , (pat_d60 +. pat_d61 +. pat_d62 +. pat_d63 +. pat_d64 +.
pat_d65) { _1 0

NB.
========================================================================

NB. Checker Texture for Reference
NB. note GL_RGBA so that pattern is 1 integer per texel
texture=:verb define
glClearColor 0 0 1 0
glEnable GL_DEPTH_TEST
glDepthFunc GL_LEQUAL
glPixelStore GL_UNPACK_ALIGNMENT, 1
glTexImage2D (GL_TEXTURE_2D, 0 3 64 64 0, GL_RGBA, GL_UNSIGNED_BYTE);
y. NB. Checker

glTexParameter GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,GL_CLAMP
glTexParameter GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,GL_CLAMP
glTexParameter GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,GL_NEAREST
glTexParameter GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,GL_NEAREST

glTexEnv GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL
glEnable GL_TEXTURE_2D
glShadeModel GL_FLAT

glClear GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT
```

```
drawcube ''
)

Cube =: (_1 _1 1);(_1 1 1);(1 1 1);(1 _1 1)

drawcube =: 3 : 0
glBegin GL_QUADS

glTexCoord 0 0 0 1
glVertex _1 _1  1
glTexCoord 0 1 0 1
glVertex _1  1  1
glTexCoord 1 1 0 1
glVertex  1  1  1
glTexCoord 1 0 0 1
glVertex  1 _1  1

glTexCoord 0 0 0 1
glVertex  1 _1  1
glTexCoord 0 1 0 1
glVertex  1  1  1
glTexCoord 1 1 0 1
glVertex  1  1  _1
glTexCoord 1 0 0 1
glVertex  1 _1  _1

glTexCoord 0 0 0 1
glVertex  1 _1 _1
glTexCoord 0 1 0 1
glVertex  1  1  _1
glTexCoord 1 1 0 1
glVertex  _1  1 _1
glTexCoord 1 0 0 1
glVertex  _1 _1 _1

glTexCoord 0 0 0 1
glVertex  _1 _1 _1
glTexCoord 0 1 0 1
glVertex  _1  1  _1
glTexCoord 1 1 0 1
glVertex  _1  1  1
```

```
glTexCoord 1 0 0 1
glVertex  _1 _1  1

glTexCoord 0 0 0 1
glVertex  1  1  1
glTexCoord 0 1 0 1
glVertex  1  1  _1
glTexCoord 1 1 0 1
glVertex  _1  1  _1
glTexCoord 1 0 0 1
glVertex  _1 1  1

glTexCoord 0 0 0 1
glVertex   1 _1  1
glTexCoord 0 1 0 1
glVertex   1 _1 _1
glTexCoord 1 1 0 1
glVertex  _1 _1 _1
glTexCoord 1 0 0 1
glVertex  _1 _1  1

glEnd''
)

NB. checker pattern ================================================
w=.8 8$_1
b=.8 8$0
pattern=: ,64 64$,    (8 64$,w,.b),8 64$,b,.w
```