JAPLA研究会資料 2012/9/22

J-OpenGLグラフィックスによる群論の理解
正４面体を回転させ、群論を実験する

西川　利男

J-OpenGLグラフィックスで、ルービック・キューブの動きをシミュレーションしたいということから、群論に足を踏み込み、いまや群論そのもののとりこになってしまった。ところが、このJ-OpenGLなる環境が群論を理解するのに、またとない優れものであることがわかった。

筆者の独断的な理解では、群論を次のように考えている。

群論とは、平面および立体の図形を動かしたときの
対称性の幾何学を代数演算として扱うツールである。

また、このために群論の参考書を和書、洋書にわたって、かなりあさったが満足行くものは、仲々見つからなかった。唯一といえそうな書はMark Anthony Armstrong（Dept. Math. Sci. Univ. of Durham, England）の本[1]であり、各国で教科書として使われているようである。日本語訳[2]もあるが、あまり良く訳されているとはいえない。

１．正４面体とは

プラトンの立体と呼ぶつぎの５種類の正多面体が、古くから知られている。

正４面体（４つの正３角形）……Tetrahedron
立方体（６つの正方形）……Cube
正８面体（８つの正３角形）……Octahedron
正１２面体（１２の正５角形）……Dodecahedron
正２０面体（２０の正３角形）……Icosahedron

正４面体はその最初のもので、一見簡単そうでありながら、観る角度によって、思いもかけない見え方をする。

くわしく、調べるには４つの頂点の(X, Y, Z)の座標値を用いて行う。その１つの場合の座標値はつぎのようになる。

P(1, 1, 1)
Q(_1, _1, 1)
R(1, _1, _1)
S(_1, 1, _1)

文献

[1] M.A.Armstrong, "Groups and Symmetry", Springer, (1988).
Chap.7 Isomorphisms, Chap.8 Plato's Solids and Cayley's Theorem,
Chap.9 Matrix Groups. p.32 – p.51

[2] M.A.アームストロング、佐藤信哉訳「対称性からの群論」シュプリンガー・ジャパン(2007). 第 7 章 同型写像, 第 8 章 プラトンの立体とケイリーの定理,
第 9 章 行列群. p.38 – p.58

２．３次元立体表示への OpenGL グラフィックスの利点

われわれが立体を理解する難しさとは、現実の空間はたて、よこだけでなく奥行きのある３次元であるにもかかわらず、それを表示するには、紙面にせよ、コンピュータのディスプレイ画面にせよ２次元で行わなくてはならない、ということにある。

(X, Y, Z)の座標値の表現はあくまで数値の並びで図形ではない。最も理想とするのは、紙細工あるいは針金細工で実際に立体を作ればよいことはいうまでもない。

紙面や画面で行う１つの方法として、人間の目の錯覚を利用した立体視(ステレオ表示)がある。

OpenGL グラフィックスでは簡単なコマンド操作で、視点をいろいろ変えて立体を画面表示することができる。つまり正面からだけでなく、立体の側面あるいは裏にまわって見ることができるので、われわれの頭の中に３次元の立体を描くことができる。このように OpenGL は立体の理解には、欠かすことの出来ないツールだといえよう。

３．　座標値の行列計算による図形の運動

２次元の座標において、点(X, Y)の原点の回りの反時計まわり θ の回転はつぎの行列計算で行われる。

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}$$

また、点(X, Y)が、原点を通り正の X 軸から反時計まわりに角度 θ/2 をなす直線に関する鏡映(reflection)の点の座標はつぎの行列計算で行われる。証明は後述。

$$\begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}$$

これを、３次元座標の立体に拡張し、正４面体の運動の行列計算は次のようになる。

・Z 軸のまわりの θ の回転

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

・X 軸のまわりの θ の回転

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

・頂点 P と原点を通る軸のまわりの 120°の回転

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

・頂点 P と原点を通る軸のまわりの 240°の回転

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

４．J-OpenGL グラフィックスのプログラムの概要

これまで、何遍となく述べてきたので、要点のみ記す。

・J-OpenGL ライブラリの使用

```
require 'gl3'
```

・フォームの作成

```
A =: 1 : 0
pc a; 親フォーム a の作成
xywh 7 34 332 287;   画面の大きさをきめて、
cc g isigraph ws_clipchildren ws_clipsiblings; 子フォーム g を属性 isigraph
                                                としての作成
xywh 348 8 34 11;cc zrot30 button;    子フォーム zrot30 の button を作成
---
```

・プログラムの実行

```
a_run =: 3 : 0
wd A       フォームの実行
glaRC ''   J-OpenGL の実行
---
)
```

・画面 g 上での画像の表示

```
a_g_paint =: 3 : 0   子フォーム g のさらに下の子プログラム paint は
connect ''           メインプログラムの実行とともに実行される
---
)
```

・画面 g 上での文字コマンドの入力

```
a_g_char =: 3 : 0    子フォーム g のさらに下の子プログラム char は
---                  メインプログラムの実行とともに実行される
)
```

・正４面体の頂点座標の設定

```
NB. Tetrahedron Vertex X, Y, Z
Td =:    1, 1, 1, _1, _1, 1, 1, _1, _1, _1, 1, _1
Td =: (4, 3)$Td
```

・頂点の結合

```
connect =: 3 : 0
'i j'=. y.
glBegin GL_LINES
glVertex i{Td
glVertex j{Td
glEnd ''
)
```

全体プログラムは最後に記載した。

５ 行列計算による操作

正４面体の頂点 P, Q, R, S の各座標に対して、いろいろな回転、鏡映などの操作を行い、その結果を更新してグラフィックス表示する。

ボタン zrot30 = z 軸のまわりに 30°の回転、

ボタン zrot60 = z 軸のまわりに 60°の回転

ボタン zrot120 = z 軸のまわりに 120°の回転

ボタン zrot180 = z 軸のまわりに 180°の回転

ボタン xrot30 = x 軸のまわりに 30°の回転

ボタン xrot60 = x 軸のまわりに 60°の回転

ボタン xrot120 = x 軸のまわりに 120°の回転

ボタン prot120 = 頂点 P と原点を結ぶ軸のまわりに 120°の回転

ボタン prot240 = 頂点 P と原点を結ぶ軸のまわりに 240°の回転

ボタン zmirr = 直線 PQ の中点の法線と z 軸で成す平面に対する鏡映

ボタン pqmirr = 直線 PQ と原点を含む平面とに対する鏡映

上の操作を実行するには、キー入力 g または G で行う。

それぞれのボタンでは、前節に示した行列をセットし、キー入力 g または G でその行列計算を行う。

たとえば、ボタン prot120 の操作、頂点 P と原点を通る軸のまわりの 120°の回転は、つぎの行列の計算である。これによって、P, Q, R, S の各座標値が求められる。

```
    Td =:  1, 1, 1, _1, _1, 1, 1, _1, _1, _1, 1, _1
    Td =: (4, 3)$Td
    Td
 1  1  1  NB. P
_1 _1  1  NB. Q
 1 _1 _1  NB. R
_1  1 _1  NB. S
    Mat =: 3 3$0 0 1 1 0 0 0 1 0
    Mat
0 0 1
1 0 0
0 1 0
    |: Mat +/ . * |: Td
 1  1  1  NB. P after rotation
 1 _1 _1  NB. Q after rotation
_1  1 _1  NB. R after rotation
_1 _1  1  NB. S after rotation
```

６．置換群としての操作

以上の操作は、J のプリミティブである置換群操作の動詞 C.を用いても求められる。たとえば、頂点 P と原点を通る軸のまわりの 120°の回転は、つぎの直接置換で行われる。ここで、数学では 1 オリジンであるが、J では 0 オリジンであることに注意。

$$\begin{pmatrix} 0\,1\,2\,3 \\ 0\,2\,3\,1 \end{pmatrix}$$

```
   prot_dir =: 0 2 3 1
   prot_dir
0 2 3 1
   prot_dir C. Td
 1  1  1
 1 _1 _1
_1  1 _1
_1 _1  1
```

一方、これは巡回置換(サイクル)によっても得られる。
(1 2 3)

```
   prot_cyc =: < 1 2 3
   prot_cyc
+-----+
|1 2 3|
+-----+
   prot_cyc C. Td
 1  1  1
 1 _1 _1
_1  1 _1
_1 _1  1
```

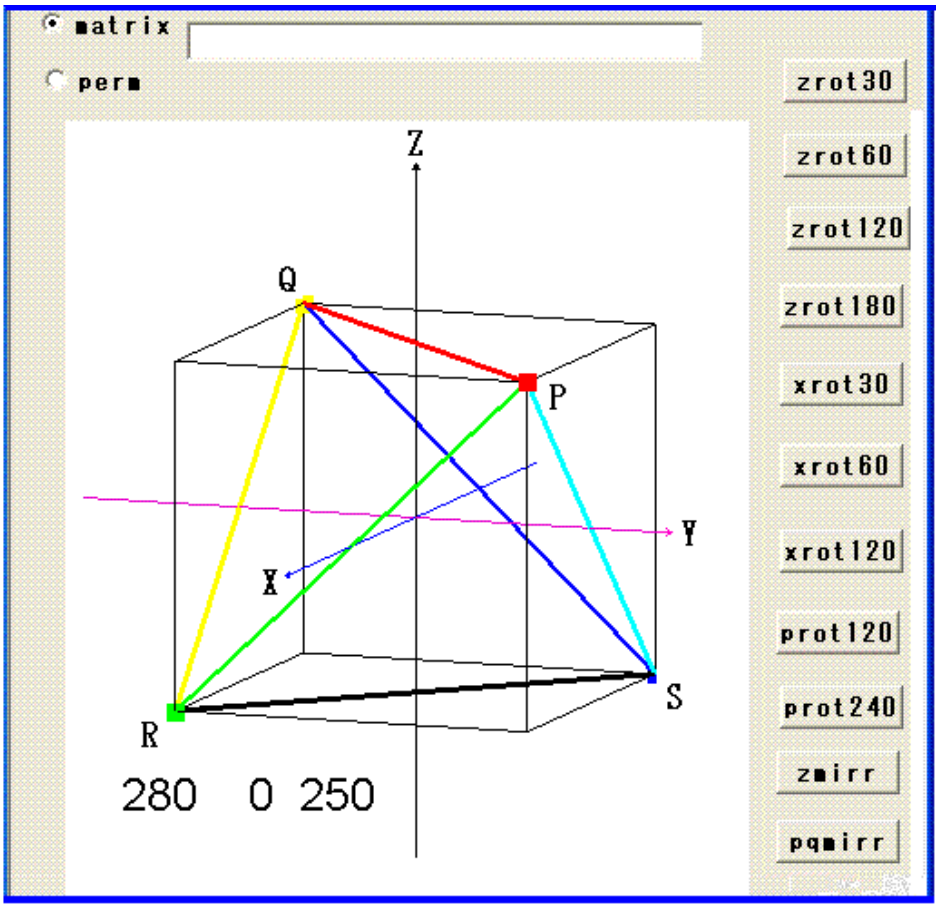
なお、J では直接置換と巡回置換(サイクル)とは、同じプリミティブ C.を単項動詞として用いて、相互に変換できる。

```
   C. 0 2 3 1
+-----+
|1 2 3|
+-----+

   C. (<1 2 3)
0 2 3 1
```

７．正４面体群の J-OpenGL グラフィックスの実際

たとえば
run ''
により、正４面体が表示される。コマンド x, X, y, Y, z, Z の入力により、視点はいろいろに変えられるので、見やすいように設定すればよい。



次のように、回転、鏡映などの操作を行う。

(1) ボタンの選択による実行

zrot30, .., zrot120 などのボタンをクリックして、キー・入力 g では時計方向、G では反時計方向の回転が行われる。zrot120 が群操作になる。

(2) 行列計算による実行

ラジオボタン matrix を選んで、入力窓には例えば
3 3$0 0 1 1 0 0 0 1 0 (Enter)
と入力し、その後キー・入力 g, G で実行する。

(3) 置換群操作による実行

ラジオボタン perm を選んで、入力窓には例えば
0 2 3 1(Enter)

あるいは

<1 2 3 (Enter)

と入力し、その後キー・入力 g, G で実行する。

８．正４面体の群操作と交代群 A_4

正４面体の対称性、つまり正４面体群にはどんなものがあり、また何通りあるであろうか。

これらは大きく分けて、２種類、全部で 12 通りある。

・正４面体の１つの頂点と中心とを結ぶ軸のまわりに、角度 2π/3 (120°)と角度 _2π/3 (240°)との回転操作……4 x 2 = 8 通り

・２つの対する辺の中点同士を結ぶ軸のまわりに、角度 2π/2 (180°)の回転、つまり、この軸を含む平面での鏡映……3 通り

これらに恒等変換（何もしない）を加えて、群の要素（元）は 12 通り（位数）からなる正４面体群を成す。

一方、これを頂点の置換と考えると、４つの対象物から出来る置換群 S_4 の部分群となる偶置換だけから成る交代群 A_4 と等しくなる。

これは J によりプリミティブ A.と C.を用いて簡単に求められる。[3]

[3] 西川利男、「J 言語からの群論の理解 - その２」JAPLA 研究会資料 2011/11/26

```
   S4 =: (i. !4) A. i. 4
```

この S_4 の中から、巡回置換が偶数になるものだけをとり出せば A_4 が得られる。

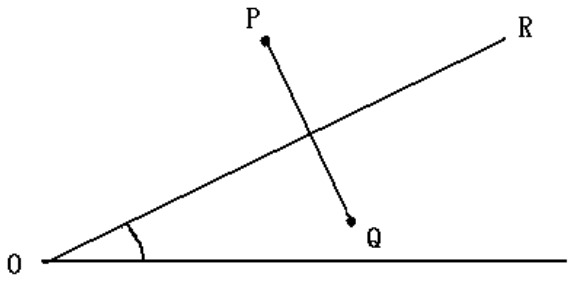
プログラムは後述。

```
   A4 =: alt S4
   A4
0 1 2 3  NB. 恒等変換
0 2 3 1  NB. 点 P と中心を軸に 120°回転
0 3 1 2  NB. 点 P と中心を軸に 240°回転
1 0 3 2  NB. 点 P←→点 Q、点 R←→点 S が入れ替わる鏡映
1 2 0 3  NB. 点 S と中心を軸に 120°回転
1 3 2 0  NB. 点 R と中心を軸に 120°回転
2 0 1 3  NB. 点 R と中心を軸に 240°回転
2 1 3 0  NB. 点 Q と中心を軸に 120°回転
2 3 0 1  NB. 点 P←→点 R、点 Q←→点 S が入れ替わる鏡映
3 0 2 1  NB. 点 R と中心を軸に 240°回転
3 1 0 2  NB. 点 Q と中心を軸に 240°回転
3 2 1 0  NB. 点 P←→点 S、点 Q←→点 R が入れ替わる鏡映
```

このように、立体図形の空間内の運動である正４面体群の群操作は、４つの対象物入れ換えにすぎない交代群と全く同じにふるまう。これが群論の基本概念である同型(Isomorphisms)を示している。

J-OpenGL グラフィックスではこれを目の当たりに動かして、実験することができる。いろいろやってみてほしい。

・変換行列 $\begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix}$ **が鏡映となる理由**



点 P(p, q)の　　直線 OR（原点 O から角度 θ の傾き）の鏡映となる点 Q(r,s)の座標は

$r = p\cos\theta + q\sin\theta$

$s = p\sin\theta - q\cos\theta$

直線 PQ は次のようになる。

$$\frac{x-p}{r-p} = \frac{y-q}{s-q}$$

から

$$y = \frac{s-q}{r-p}x + \left(\frac{s-q}{r-p}p + q\right)$$

これに直交する傾きは

$$-\frac{r-p}{s-q} = -\frac{p\cos\theta + q\sin\theta - p}{p\sin\theta - q\cos\theta - q} = \frac{p(1-\cos\theta) - q\sin\theta}{p\sin\theta - q(1+\cos\theta)}$$

ここで、$\sin\theta = 2\sin\frac{\theta}{2}\cos\frac{\theta}{2}$, $1-\cos\theta = 2\sin^2\frac{\theta}{2}$, $1+\cos\theta = 2\cos^2\frac{\theta}{2}$

の関係を代入する。

$$(\text{原式}) = \frac{2p\sin^2\frac{\theta}{2} - 2q\sin\frac{\theta}{2}\cos\frac{\theta}{2}}{2p\sin\frac{\theta}{2}\cos\frac{\theta}{2} - 2q\cos^2\frac{\theta}{2}} = \frac{2\sin\frac{\theta}{2}(p\sin\frac{\theta}{2} - q\cos\frac{\theta}{2})}{2\cos\frac{\theta}{2}(p\sin\frac{\theta}{2} - q\cos\frac{\theta}{2})} = \tan\frac{\theta}{2}$$

結局、直線 PQ は原点を通る傾き θ/2 のの直線 OR と直交する。ゆえに点 Q は直線 OR に関する点 P の鏡映(reflection)になる。

```
NB. Permutation Group and Alternating Group
=====================================
S3 =: (i. !3) A. i.3
S4 =: (i. !4) A. i.4

alt =: 3 : 0
i =. 0
AA =. (1, ({:$y.))$0
while. i < #y.
  do.
   IS =. i{y.
   IN =. # C. IS
   if. 0 = 2|IN
       do.
            AA =. AA ,"(2) IS
       end.
   i =. i + 1
end.
}. AA
)

NB. OpGLN_MatGr.ijs Matrx Group in OpenGL
=================================
NB.   2012/7/18, 8/1
NB. imported from OpGLN1.ijs, which is from opgl_wire.ijs
NB.   Sphere and Cylinder Wired Rotation
NB.   glCallList, glNewList
NB.   2009/8/23, 2009/9/12

require 'gl3'

A=: noun define
pc a closeok;
menupop "&Help";
menu help "&Help" "" "" "";
menupopz;
xywh 15 33 309 261;cc g isigraph ws_clipchildren ws_clipsiblings
rightmove bottommove;
xywh 337 35 34 11;cc zrot30 button;
xywh 337 53 34 11;cc zrot60 button;
xywh 338 71 34 11;cc zrot120 button;
xywh 336 109 34 11;cc xrot30 button;
xywh 336 129 34 11;cc xrot60 button;
xywh 336 150 34 11;cc xrot120 button;
xywh 335 170 34 11;cc prot120 button;
xywh 336 188 34 11;cc prot240 button;
xywh 335 204 34 11;cc zmirr button;
xywh 335 221 34 11;cc pqmirr button;
xywh 76 8 213 11;cc incom edit ws_border es_autohscroll;
xywh 8 4 50 11;cc matrix radiobutton;
xywh 9 18 50 11;cc perm radiobutton group;
```

```
xywh 336 90 34 11;cc zrot180 button;
pas 0 0;
rem form end;
)

NB. run '' ==> Tetrahedron R: 280 0 250 in Cube
NB. run 0  ==> Tetrahedron R: 0 0 0
NB. run 1  ==> Tetrahedron R: 280 0 250
NB. run 2  ==> Tetrahedron R: 280 0 250 in Cube
NB. run 3  ==> Tetrahedron R: 280 0   0 in Cube
NB. run 4  ==> Tetrahedron R: 195 220 180 in Cube - 2012/8/8
NB. run 6  ==> Cube

NB. rotate figure => zrot button, then enter 'g' or 'G' charcter


run =: a_run
a_run=: verb define
C_T =: y.
if. 0 = #C_T do. C_T =: 2 end.
wd A
select. C_T
 case. 0 do. R =:   0   0   0
 case. 1 do. R =: 280   0 250
 case. 2 do. R =: 280   0 250
 case. 3 do. R =: 280   0   0
 case. 4 do. R =: 195 220 180
end.
Mat =: 3 3$1 0 0 0 1 0 0 0 1
MAT_OR_PER =: 0
glaRC''
SPOINTS =: 2
spoints ''
glaFont 'arial 30'
glaUseFontBitmaps 0 32 26 32
wd 'pshow;ptop'
)

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 _10
glRotate R ,. 3 3 $ 1 0 0 0
drawaxis ''
glLineWidth 4
select. C_T
 case. 0 do. td_connect '' [ td_points ''
 case. 1 do. td_connect '' [ td_points ''
```

```
  case. 2 do. td_connect '' [ td_points '' [ cb_connect ''
  case. 3 do. td_connect '' [ td_points '' [ cb_connect ''
  case. 4 do. td_connect '' [ td_points '' [ cb_connect ''
  case. 6 do. cb_connect '' [ cb_points ''
  case. 9 do. glCallList SPOINTS
end.
drawtext ''
glaSwapBuffers ''
)

NB. input character command
a_g_char =: verb define
R =: 360 | R + 5 * 'xyz' = 0 { sysdata
R =: 360 | R - 5 * 'XYZ' = 0 { sysdata
if. MAT_OR_PER = 0
   do.
      if. 1 = 'g' = 0 { sysdata do.
         Td =: |: Mat +/ . * |: Td
         Cb =: |: Mat +/ . * |: Cb
      end.
      if. 1 = 'G' = 0 { sysdata do.
         Td =: |: (%. Mat) +/ . * |: Td
         Cb =: |: (%. Mat) +/ . * |: Cb
      end.
   else.
      if. 1 = 'g' = 0 { sysdata do.
         Td =: Perm C. Td
      end.
      if. 1 = 'G' = 0 { sysdata do.
         Td =: (Perm&C.)^:_1 Td
      end.
   end.
glpaintx''
)

NB. input command ==================================
NB. select matrix calc. or permutation group
======================
a_matrix_button=: 3 : 0
MAT_OR_PER =: 0
)

a_perm_button=: 3 : 0
MAT_OR_PER =: 1
)

NB. e.g.
NB.   radiobutton: matrix, Enter '3 3$0 0 1 1 0 0 0 1 0'
NB.   radiobutton: perm  , Enter '<1 2 3'
NB.         "       "    , Enter '0 3;1 2'
a_incom_button=: 3 : 0
```

```
if. MAT_OR_PER = 0
   do.   Mat =: ". incom
   else. Perm =: ". incom
end.
wd 'setfocus g'
)

NB. Tetrahedron Vertex X, Y, Z
Td =:    1, 1, 1, _1, _1, 1, 1, _1, _1, _1, 1, _1
Td =: (4, 3)$Td
Td =: 8 * Td

NB. Cube Vertex X, Y, Z
Cb =:    1,  1, 1, 1,  1, _1, _1,  1, _1, _1, 1, 1
Cb =: Cb, 1, _1, 1, 1, _1, _1, _1, _1, _1, _1,_1, 1
Cb =: (8, 3)$Cb
Cb =: 8 * Cb

NB. Color
CC =: 1 0 0 0;1 1 0 0;0 1 0 0;0 0 1 0

drawaxis =: verb define
glLineWidth 1
glColor 0 0 1 0
glBegin GL_LINES
   glVertex 16 0 0
   glVertex _16 0 0
glEnd ''
glLineWidth 3
glBegin GL_TRIANGLES
   glVertex 16 0.1 0.1
   glVertex 16 _0.1 0.1
   glVertex 16 0.1 _0.1
glEnd ''
glLineWidth 1
glColor 0.9 0 0.8 0
glBegin GL_LINES
   glVertex 0  16 0
   glVertex 0 _16 0
glEnd ''
glLineWidth 3
glBegin GL_TRIANGLES
   glVertex  0.1 15.8 0.1
   glVertex _0.1 15.8 0.1
   glVertex  0.1 15.8 _0.1
glEnd ''
glLineWidth 1
glColor 0 0 0 0
glBegin GL_LINES
   glVertex 0 0  16
   glVertex 0 0 _16
```

```
glEnd ''
glLineWidth 3
glBegin GL_TRIANGLES
   glVertex 0.1 0.1 16
   glVertex _0.1 0.1 16
   glVertex 0.1 _0.1 16
glEnd ''
)

NB. Vertex Connections ============================
connect =: verb define
:
'i j' =. y.
if. C_T = 6   do. PQ =. Cb else. PQ =. Td end.
PQ =. x.
glBegin GL_LINES
  glVertex i{PQ
  glVertex j{PQ
glEnd ''
)

NB. draw Tetrahedron Vertex Connected
td_connect =: verb define
   glLineWidth 3
   glColor 1 0 0 0
   Td connect L:0 (0 1)
   glColor 1 1 0 0
   Td connect L:0 (1 2)
   glColor 0 1 0 0
   Td connect L:0 (2 0)
   glColor 0 1 1 0
   Td connect L:0 (0 3)
   glColor 0 0 1 0
   Td connect L:0 (3 1)
   glColor 0 0 0 0
   Td connect L:0 (2 3)
)

NB. draw Cube Vertex Connected
cb_connect =: verb define
   glLineWidth 1
   glColor 0 0 0 0
   Cb connect L:0 (4 5;5 6;6 7;7 4)
   Cb connect L:0 (3 7)
   Cb connect L:0 (2 6)
   Cb connect L:0 (1 5)
   Cb connect L:0 (0 4)
   Cb connect L:0 (0 1;1 2;2 3;3 0)
)
```

NB. Group Moves with Matrix Calc.
==========
NB.   2012/7/22
a_zrot30_button=: 3 : 0
Mat =: 3 3$(Cos 30), (-Sin 30), 0, (Sin 30), (Cos 30), 0, 0, 0, 1
wd 'setfocus g'
)

a_zrot60_button=: 3 : 0
Mat =: 3 3$(Cos 60), (-Sin 60), 0, (Sin 60), (Cos 60), 0, 0, 0, 1
wd 'setfocus g'
)

a_zrot120_button=: 3 : 0
Mat =: 3 3$(Cos 120), (-Sin 120), 0, (Sin 120), (Cos 120), 0, 0, 0, 1
wd 'setfocus g'
)

a_zrot180_button=: 3 : 0
Mat =: 3 3$(Cos 180), (-Sin 180), 0, (Sin 180), (Cos 180), 0, 0, 0, 1
wd 'setfocus g'
)

a_xrot30_button=: 3 : 0
Mat =: 3 3$1, 0, 0, 0, (Cos 30), (-Sin 30), 0, (Sin 30), (Cos 30)
wd 'setfocus g'
)

a_xrot60_button=: 3 : 0
Mat =: 3 3$1, 0, 0, 0, (Cos 60), (-Sin 60), 0, (Sin 60), (Cos 60)
wd 'setfocus g'
)

a_xrot120_button=: 3 : 0
Mat =: 3 3$1, 0, 0, 0, (Cos 120), (-Sin 120), 0, (Sin 120), (Cos 120)
wd 'setfocus g'
)

a_prot120_button=: 3 : 0
Mat =: 3 3$0 0 1 1 0 0 0 1 0
wd 'setfocus g'
)

a_prot240_button=: 3 : 0
Mat =: 3 3$0 1 0 0 0 1 1 0 0
wd 'setfocus g'
)

a_zmirr_button=: verb define

```
Mat =: 3 3$_1 0 0 0 _1 0 0 0 1
wd 'setfocus g'
)

a_pqmirr_button=: verb define
Mat =: 3 3$0 1 0 1 0 0 0 0 1
wd 'setfocus g'
)

NB. draw points
NB. Vertex Points ============================
points =: verb define
:
i =. y.
PQ =. x.
glPointSize 10
glBegin GL_POINTS
  glVertex i{PQ
glEnd ''
)

NB. point Tetrahedron Vertex
td_points =: verb define
   glColor 0 0 1 0
   Td points L:0 (3)
   glColor 0 1 0 0
   Td points L:0 (2)
   glColor 1 1 0 0
   Td points L:0 (1)
   glColor 1 0 0 0
   Td points L:0 (0)
)

NB. point Cube Vertex
cb_points =: verb define
   glColor 0 0 1 0
   Cb points L:0 (7)
   glColor 0 1 0 0
   Cb points L:0 (6)
   glColor 1 1 0 0
   Cb points L:0 (5)
   glColor 1 0 0 0
   Cb points L:0 (4)
   glColor 0 0 1 0
   Cb points L:0 (3)
   glColor 0 1 0 0
   Cb points L:0 (2)
   glColor 1 1 0 0
   Cb points L:0 (1)
   glColor 1 0 0 0
   Cb points L:0 (0)
```

```
)

NB. indicate rotated angle x, y, z in degree
drawtext =: verb define
glMatrixMode GL_MODELVIEW
glLoadIdentity ''
glColor 0 0 0 0
glRasterPos _16 _16 0
glCallLists 5 '': R
)


a_g_size=:verb define
wh=.glqwh''
glViewport 0 0,wh
glMatrixMode GL_PROJECTION
glLoadIdentity''
glOrtho _24 24 _24 24 _24 24
NB. gluPerspective 30, (%/wh),1 10
glPolygonMode GL_FRONT, GL_LINE
glPolygonMode GL_BACK, GL_POINT
)


a_help_button=: verb define
wd'mb OpenGL *Press x y z to rotate.'
wd 'setfocus g'
NB. Mat =: 3 3$1 0 0 0 0.866025 _0.5 0 0.5 0.866025
NB. Td =: Td +/ . * Mat
NB. glCallList POINTS
)

Sin =: sin@rfd
Cos =: cos@rfd


NB. imported from trig
================================================
===========
sin=: 1&o.
cos=: 2&o.
tan=: 3&o.

sinh=: 5&o.
cosh=: 6&o.
tanh=: 7&o.

arcsin=: _1&o.
arccos=: _2&o.
arctan=: _3&o.
```

```
arcsinh=: _5&o.
arccosh=: _6&o.
arctanh=: _7&o.

pi=: 1p1

dfr=: *&(180%pi)
rfd=: *&(pi%180)
```