JAPLA 研究会資料 2013/9/14

実験と3Dグラフィックスによる幾何学
ピラミッド(三角錐)の体積はプリズム(三角柱)の 1/3 になる
……きみにはすぐ分かるだろうか……

西川　利男

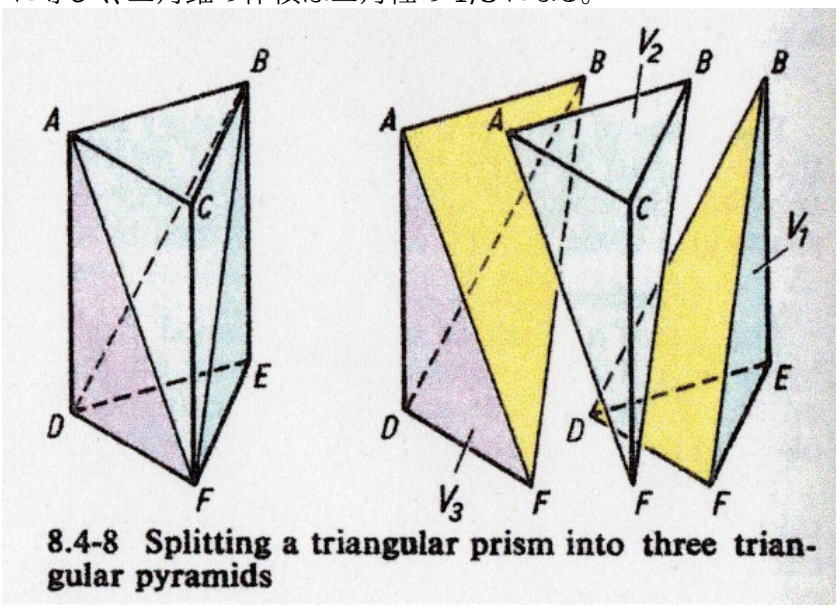
はじめに

錐体(三角錐、四角錐、円錐)の体積は
　柱体(三角柱、四角柱、円柱)の体積の 1/3 である

だれでも知っているこの有名な公式は中学校の教科書にのっているだろうが、その理由をわかるように説明してはいない。

しかし、"The VNR Concise Encyclopedia of Mathematics" の p.194 には、次の図を使って、ていねいな説明がなされている。

まず、三角柱は、3つの三角錐に分解できる。三角錐 V1 と V2 とは、底面(DEF と ABC)は等しく、高さも等しいので体積は等しい。次に、三角錐 V2 と V3 では、底面(ACF と ADF)が等しく、これらの面から点 B までの距離は高さとなり、体積は等しい。よって3つの三角錐の体積は互いに等しく、三角錐の体積は三角柱の 1/3 になる。



8.4-8 Splitting a triangular prism into three triangular pyramids

これは実際に厚紙で立体を作って実験してみれば、だれでもすぐ分かる。また、
教育的にも、すばらしい説明である。

しかしながら、紙の上の立体図を頭の中でイメージするのはそれほど容易ではない。
この課題を OpenGL グラフィックスにより、コンピュータ画面の上で、より分かりやすく再現してみた。

1.射影幾何学と OpenGL

射影幾何学(Projective Geometry)は、ルネッサンス期の画家レオナルド・ダ・ヴィンチやアルブレヒト・ディユーラーたちの透視画法の研究を元としている。その後、フランス革命期、ナポレオンが作ったエコール・ポリテクニックのモンジュ、ポンスレらにより機械設計のための画法幾何学となった。さらに、非ユークリッド幾何学、アフィン幾何学を巻き込み、今や最も抽象性の高い位相幾何学へと発展している。

射影幾何学の基本の考えは、つぎの一言でいいあらわされる。

「3次元空間内の物体を2次元の画像としてどう表したらよいか」

このキャンパス上の絵を現代のコンピュータ・ディスプレーの画像として実現したのがまさに OpenGL グラフィックスといえよう。

OpenGL では、従来のグラフィックスとは異なるやり方を行う。たとえば3角形 ABC を描くのに、直接3点を結ぶ線を引くのではなく、次のように行う。

3次元空間内で3つの点…それぞれの点は座標値 X, Y, Z の3つの値で示される…
で3角形のオブジェクトを定めた上で(レンダリング)

```
glBegin GL_TRIANGLE
  glVertex A
  glVertex B
  glVertex C
glEnd ''
```

べつに定めた投影処理により、2次元のディスプレー上に画像として描かれる。

ここで、射影幾何学の次の基本性質を OpenGL の目で見てみよう。
それまでのユークッリド幾何学の定理では、図形の長さと角度とはそれぞれ重要な意味を持っていた。しかし、射影幾何学では図形の長さと角度とは意味を持たない。

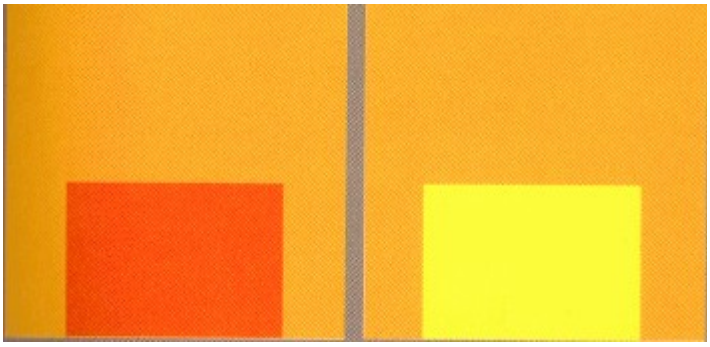
これは OpenGL によりつぎのように実現される。
つまり、2次元のディスプレー画像は投影の条件に従って

頂点を結ぶ辺の長さはさまざまに変わる
2つの辺の間の角もさまざまに変わる

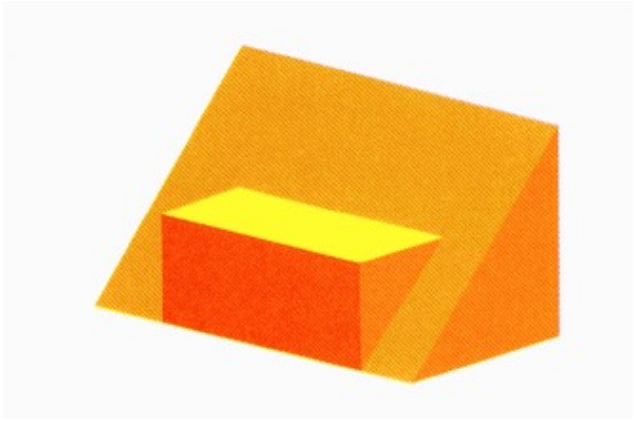
このように、ディスプレー上では、元の3次元立体からは、思いもかけない形が現れることになる。

これは自分の目で実験してみれば、すぐ分かることである。3次元立体をイメージするのが難しいというのはこのことである。これも OpenGL で示してみよう。

ところで、パズルの本「四角の迷宮」p.35-6 から、奇妙な見取り図として、前からと上から見た図から、どんな3次元立体かわかるか?、というのがあった。考えてほしい。



答えはつぎのようになる。このように3次元立体をイメージするのは難しい。



ここで射影幾何学について、私の個人的理解のしかたについてちょっと述べたい。
数学の書を開くと、Pappus, Desargues, Pascal, Brianchon とさまざまな定理が延々と説明されている。数学では射影という見方の環境を変えたときに、不変に保たれる原理の探求が大切とされ、これらの定理はそれを示すのであろう。

また、別のパラメータとして複比(double ratio、これは cross-ratio, anharmonic ratio とも呼ばれる)がある。これは、例えば3次元の多面体でいろいろな辺の長さの比同士を比べたとき、その割合(＝複比)は変わらない、というものである。

しかし、OpenGL を利用する立場の私の関心は、3次元の立体(実体)をどう2次元の画面に表現するかという射影変換だけである。これは座標値を元とした行列演算で行われる。ところが、この計算部分は OpenGL のコマンド操作によってすべてやってくれて、これを直接コーディングする必要はない。

結論をいえば、OpenGL を使うのに、射影幾何学を意識することはない。とくに言えば、先に述べた、射影幾何学の素朴な考え方

頂点を結ぶ辺の長さはさまざまに変わる
2つの辺の間の角もさまざまに変わる

という日常の体験だけである。

2.J-OpenGL のプログラム－3つの3角ピラミッドで3角プリズムをつくる

J の OpenGL のプログラムの構成などは、いままで何回か説明したとおりである。

・フォームの作成……A_g
・プログラムの実行…run_a
・レンダーリング……a_g_paint
・投影条件の設定……a_g_size
・キー入力コマンド…a_g_char

まず、1ページの見取り図を元に、3つの3角ピラミッドの頂点の値を定める。

```
A =: 0, 2, 0
B =: 0, 2, _1
C =: 1, 2, 0
D =: 0, 0, 0
E =: 0, 0, _1
F =: 1, 0, 0
```

これらはまとめて、次のようにした。

```
 cylpyd =: (0, 0, 0);(1, 0, 0);(0, 0, _1);(0, 2, 0);(1, 2, 0);(0, 2, _1)
```

オブジェクト立体の作成(レンダリング)は、立体の面に対して3つの頂点座標を指定することで行う。

見取り図の3つの3角ピラミッド V1, V2, V3 はそれぞれ次のように名付けた。

P……V3,　Q……V2,　R……V1

レンダリング・ルーチン a_g_paint から呼ぶ3角ピラミッド作成の OpenGL プログラムは、それぞれ次のようになる。

P の各面は

```
drawp =: 3 : 0
glBegin GL_TRIANGLES
 glColor 1 0 0 0
 glVertex >0{cylpyd
 glVertex >1{cylpyd
 glVertex >3{cylpyd
glEnd ''
glBegin GL_TRIANGLES
 glColor 0 0 1 0
 glVertex >0{cylpyd
 glVertex >3{cylpyd
 glVertex >5{cylpyd
glEnd ''
glBegin GL_TRIANGLES
 glColor 1 0 1 0
 glVertex >1{cylpyd
 glVertex >5{cylpyd
 glVertex >3{cylpyd
glEnd ''
)
```

Q の各面は

```
drawq =: 3 : 0
glBegin GL_TRIANGLES
 glColor 1 0 0 0
 glVertex y. +"(1) >1{cylpyd
 glVertex y. +"(1) >4{cylpyd
 glVertex y. +"(1) >3{cylpyd
glEnd ''
glBegin GL_TRIANGLES
 glColor 1 1 0 0
 glVertex y. +"(1) >4{cylpyd
 glVertex y. +"(1) >1{cylpyd
 glVertex y. +"(1) >5{cylpyd
glEnd ''
glBegin GL_TRIANGLES
 glColor 1 0 1 0
 glVertex y. +"(1) >3{cylpyd
 glVertex y. +"(1) >5{cylpyd
 glVertex y. +"(1) >1{cylpyd
glEnd ''
```

```
glBegin GL_TRIANGLES
  glColor 0 1 0 0
  glVertex y. +"(1) >3{cylpyd
  glVertex y. +"(1) >4{cylpyd
  glVertex y. +"(1) >5{cylpyd
glEnd ''
)
```

なお、ここでは右引数として移動の座標値(x, y, z)を指定できるようにした。

Rの各面は

```
drawr =: 3 : 0
glBegin GL_TRIANGLES
  glColor 0 0 1 0
  glVertex y. +"(1) >0{cylpyd
  glVertex y. +"(1) >5{cylpyd
  glVertex y. +"(1) >2{cylpyd
glEnd ''
glBegin GL_TRIANGLES
  glColor 0 1 1 0
  glVertex y. +"(1) >0{cylpyd
  glVertex y. +"(1) >1{cylpyd
  glVertex y. +"(1) >5{cylpyd
glEnd ''
glBegin GL_TRIANGLES
  glColor 1 1 0 0
  glVertex y. +"(1) >2{cylpyd
  glVertex y. +"(1) >5{cylpyd
  glVertex y. +"(1) >1{cylpyd
glEnd ''
glBegin GL_TRIANGLES
  glColor 0 1 0 0
  glVertex y. +"(1) >0{cylpyd
  glVertex y. +"(1) >2{cylpyd
  glVertex y. +"(1) >1{cylpyd
glEnd ''
)
```

レンダリング・ルーチン a_g_paint は次のようになる。

```
a_g_paint =: verb define
glClearColor 1 1 1 0
glClear GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT
draw0 ''            NB. 初期設定
drawp ''            NB. 3角ピラミッドPの作成
drawq XYZQ          NB. XYZQだけ移動して、3角ピラミッドQの作成
drawr XYZR          NB. XYZRだけ移動して、3角ピラミッドRの作成
glaSwapBuffers ''
)
```

3. 3つの3角ピラミッドでつくる3角プリズムープログラム実行の実際

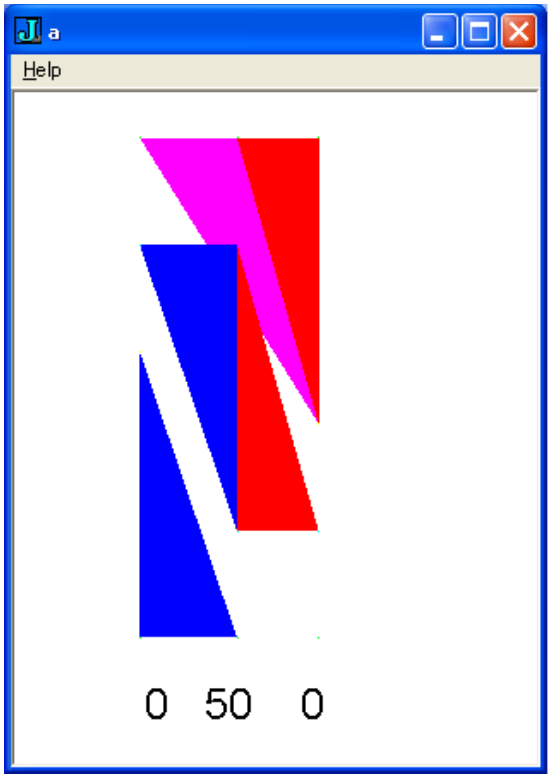
3角ピラミッドP、Q、Rをy方向に適当にずらして配置する。これを3次元グラフィックスとして、いろいろな方向から見たようすを観察することができる。

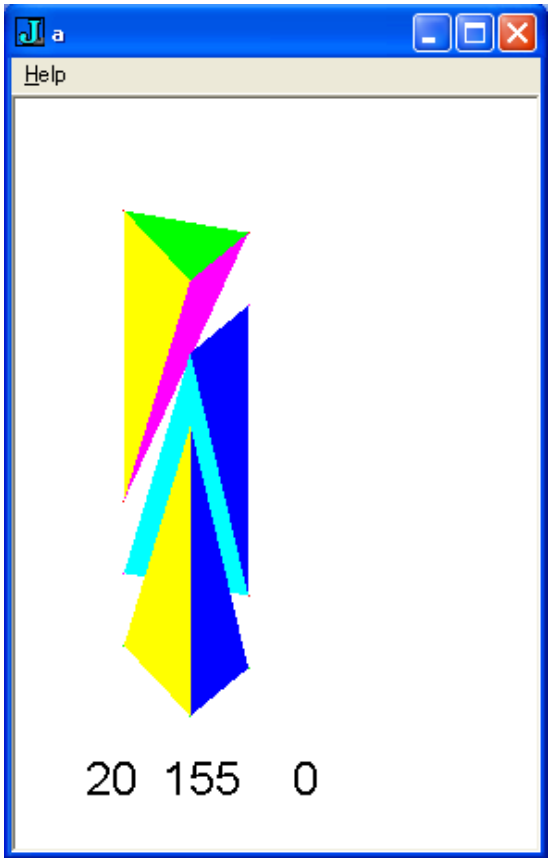
ここで、3角ピラミッド P、Q、R の配置はそれぞれキー入力コマンドにより移動できる。
3角ピラミッド P はコマンド u(上方向、+y 方向)と U(下方向、-y 方向)
3角ピラミッド R はコマンド v(上方向、+y 方向)と V(下方向、-y 方向)
にと、別々に移動する。もちろん、3つのピラミッドを接触合体して、3角プリズムとすることもできる。その過程を見てみよう。

このようにして、最初の命題

錐体の体積は柱体の体積の 1/3 である

が、コンピュータの3次元グラフィックスにより、実感できるものになるだろう。





```
NB. OpGLN_PriPyd.ijs  2013/7/24  T.Nishikawa
NB. "Volume of pyramid equals to 1/3 of prism"
NB.
NB. Demo from "VNR Encyclopedia of Mathematics", p.194
NB.
NB. Usage:
NB. run ''
NB. Enter 'u' split up Center Pyramid(V2), 'U' down it (=return)
NB.       'v' split down Right Pyramid(V1), 'V' up it (=return)
NB.       'r' split right Center Pyramid(V2), 'R' left it (=return)
NB.       'l' split left Right Pyramid(V1), 'L' right it (=return)
NB. Then, you will examine the pyramids from various eye-views:
NB. Enter 'x X y Y z Z' rotate the object along the axis

cylpyd =: (0, 0, 0);(1, 0, 0);(0, 0, _1);(0, 2, 0);(1, 2, 0);(0, 2, _1)

wr =: 1!:2&2

require 'gl3'

A=: 0 : 0
pc a closeok;
menupop "&Help";
menu help "&Help" "" "" "";
menupopz;
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=: 3 : 0
V =: y.
wd :: ] 'psel a;pclose'
wd A
glaRC ''
R =: 0 0 0

XYZQ =: 0 0 0
XYZR =: 0 0 0

PQR =: 1 1 1

glaFont 'arial 30'
glaUseFontBitmaps 0 32 26 32
wd 'pshow;ptop'
)

NB. display the model picture
=========================================
```

```
a_g_paint =: verb define
glClearColor 1 1 1 0
glClear GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT

glTranslate _0.5, _0.5 , 0

draw0 ''

if. 0 = #V
 do.
   drawp ''    NB. display the left pyramid(V3)

   drawq XYZQ  NB. display the center pyramid(V2) at the position XYZQ

   drawr XYZR  NB. display the right pyramid(V1) at the position XYZR
 else.
  select. V     NB. display one selected pyramid only
    case. 'p';'P'  do. drawp ''
    case. 'q';'Q'  do. drawq XYZQ
    case. 'r';'R'  do. drawr XYZR
    case. 'pq';'PQ'  do. drawp '' [ drawq XYZQ
    case. 'pr';'PR'  do. drawp '' [ drawr XYZR
    case. 'qr';'QR'  do. drawq XYZQ [ drawr XYZR
  end.
 end.

drawtext''
glaSwapBuffers ''
)

draw0 =: 3 : 0
glPolygonMode GL_FRONT, GL_FILL
glPolygonMode GL_BACK, GL_POINT
glEnable GL_DEPTH_TEST
glMatrixMode GL_MODELVIEW
glLoadIdentity ''
glRotate R ,. 3 3 $ 1 0 0 0
)

drawp =: 3 : 0
glBegin GL_TRIANGLES
 glColor 1 0 0 0
 glVertex >0{cylpyd
 glVertex >1{cylpyd
 glVertex >3{cylpyd
glEnd ''
glBegin GL_TRIANGLES
 glColor 0 0 1 0
 glVertex >0{cylpyd
 glVertex >3{cylpyd
 glVertex >5{cylpyd
```

```
glEnd ''
glBegin GL_TRIANGLES
  glColor 1 0 1 0
  glVertex >1{cylpyd
  glVertex >5{cylpyd
  glVertex >3{cylpyd
glEnd ''
glBegin GL_TRIANGLES
  glColor 0 1 1 0
  glVertex >0{cylpyd
  glVertex >5{cylpyd
  glVertex >1{cylpyd
glEnd ''
)

drawq =: 3 : 0
glBegin GL_TRIANGLES
  glColor 1 0 0 0
  glVertex y. +"(1) >1{cylpyd
  glVertex y. +"(1) >4{cylpyd
  glVertex y. +"(1) >3{cylpyd
glEnd ''
glBegin GL_TRIANGLES
  glColor 1 1 0 0
  glVertex y. +"(1) >4{cylpyd
  glVertex y. +"(1) >1{cylpyd
  glVertex y. +"(1) >5{cylpyd
glEnd ''
glBegin GL_TRIANGLES
  glColor 1 0 1 0
  glVertex y. +"(1) >3{cylpyd
  glVertex y. +"(1) >5{cylpyd
  glVertex y. +"(1) >1{cylpyd
glEnd ''
glBegin GL_TRIANGLES
  glColor 0 1 0 0
  glVertex y. +"(1) >3{cylpyd
  glVertex y. +"(1) >4{cylpyd
  glVertex y. +"(1) >5{cylpyd
glEnd ''
)

drawr =: 3 : 0
glBegin GL_TRIANGLES
  glColor 0 0 1 0
  glVertex y. +"(1) >0{cylpyd
  glVertex y. +"(1) >5{cylpyd
  glVertex y. +"(1) >2{cylpyd
glEnd ''
glBegin GL_TRIANGLES
  glColor 0 1 1 0
```

```
  glVertex y. +"(1) >0{cylpyd
  glVertex y. +"(1) >1{cylpyd
  glVertex y. +"(1) >5{cylpyd
glEnd ''
glBegin GL_TRIANGLES
  glColor 1 1 0 0
  glVertex y. +"(1) >2{cylpyd
  glVertex y. +"(1) >5{cylpyd
  glVertex y. +"(1) >1{cylpyd
glEnd ''
glBegin GL_TRIANGLES
  glColor 0 1 0 0
  glVertex y. +"(1) >0{cylpyd
  glVertex y. +"(1) >2{cylpyd
  glVertex y. +"(1) >1{cylpyd
glEnd ''
)

NB. project the picture on the screen ====================
a_g_size =: verb define
wh =. glqwh ''
glViewport 0 0, wh
glMatrixMode GL_PROJECTION
glLoadIdentity ''
glOrtho _2.7 2.7 _2.7 2.7 _2.7 2.7
NB. gluPerspective 60, (%/wh), 1 30
)


NB. key-in x, y, z, X, Y, Z for rotation =================
a_g_char =: verb define
k =. 0 { sysdata
R =: 360 | R + 5 * 'xyz' = 0 { sysdata
R =: 360 | R - 5 * 'XYZ' = 0 { sysdata
XYZQ =:  XYZQ + (0, 0.25, 0) * 'u' = 0 { sysdata
XYZQ =:  XYZQ - (0, 0.25, 0) * 'U' = 0 { sysdata
XYZR =:  XYZR + (0, 0.25, 0) * 'V' = 0 { sysdata
XYZR =:  XYZR - (0, 0.25, 0) * 'v' = 0 { sysdata

XYZQ =:  XYZQ + (0.25, 0, 0) * 'r' = 0 { sysdata
XYZQ =:  XYZQ - (0.25, 0, 0) * 'R' = 0 { sysdata
XYZR =:  XYZR + (0.25, 0, 0) * 'L' = 0 { sysdata
XYZR =:  XYZR - (0.25, 0, 0) * 'l' = 0 { sysdata

glpaintx''
)

NB. indicate rotated angle values x, y, z in degree ============
drawtext =: verb define
glMatrixMode GL_MODELVIEW
glLoadIdentity ''
```

```
glColor 0 0 0 0
NB. glRasterPos _1 _2.5 0
glRasterPos _0.5 _1.5 0
glCallLists 5 '': R
)

a_help_button =: verb define
wd 'mb OpenGL *Press keys, x/X, y/Y, z/Z rotate, s: line or solid, h: line
hidden toggle.'
wd 'setfocus g'
)
```

```
NB. key-in x, y, z, X, Y, Z for rotation ================
a_g_char =: verb define
k =. 0 { sysdata
R =: 360 | R + 5 * 'xyz' = 0 { sysdata
R =: 360 | R - 5 * 'XYZ' = 0 { sysdata
XYZQ =:  XYZQ + XYZ0 =. (0, 0.25, 0) * 'u' = 0 { sysdata
XYZQ =:  XYZQ - XYZ0 =. (0, 0.25, 0) * 'U' = 0 { sysdata
XYZR =:  XYZR + XYZ0 =. (0, 0.25, 0) * 'V' = 0 { sysdata
XYZR =:  XYZR - XYZ0 =. (0, 0.25, 0) * 'v' = 0 { sysdata

LS =: ('s' = k) { LS, -. LS
Hid =: ('h' = k) { Hid, -. Hid
glpaintx''
)

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

```
glCallLists 5 ": R
)

a_help_button =: verb define
wd 'mb OpenGL *Press keys, x/X, y/Y, z/Z rotate, s: line or solid, h: line
hidden toggle.'
wd 'setfocus g'
)
```