OpenGL 入門（その 0）

SHIMURA Masato
jcd02773@nifty.ne.jp

2010 年 3 月 25 日

目次

1 はじめに 1

2 Knot 系 2

3 Surface 系（1）数理関数で作成 5

4 Surface 系（2）工芸 9

概要

J のデモに入っている魅力的な OpenGL のグラフィックスを鑑賞しながら、Source-Code を解析して、OpenGL の技法を学習する

1 はじめに

J の Studio/demo/OpenGl に魅力的な OpenGL のデモが入っている。J の新しいバージョンでは OpenGL のうち *simple* と *lab* に入っている *Example* は *Stand-alone* になっているので、*J* に *load* すれば直ちに動く。

Demo の方は細分化されているが額縁は着いていない。次により額縁に入れて *1* ファイルのみで *OpenGl* のデモを見ることができるようにする。

- *packages/grapghis/opengl/demo* の各ファイルを *copy*
- 同じ *directory* にある *glviews.ijs* の *XX* の箇所に張り付け
- *opengl_run* に　*paint”* など起動するための *1* 行を記入する

幾つかのデモファイルと *Sorce-Code* を鑑賞しよう。*Source-Code* は概ね次の *4* の部分に分けられる。このうち *2* は *jzopenglutil* に入っている。また　*OpenGL* 用の多くの関数もこの *jzopenglutil*

で定義されている。

- *J* の *Form*
- *OpenGL* の *Graphic* 関数（殆どは *util* に）
- *OpenGL* の幾何（追加、変更のみ）
- 数学と *J* の幾何

1，*2* は額縁、*3*，*4* が実体部分である。
OpenGL の *ulility* 関数は *Object* になっているのでロケール *gldemo_*を付けて動かす。

2 Knot 系

2.1 リサージュ曲線

リサージュ曲線 *(1855)* は直交する 2 つの単振動を合成したもので次のように表され、周波数や信号の解析に用いられる。
（*J.A.Lissajous 1822-1880)*

$$x(t) = Acos(\omega_x t - \sigma_x)$$
$$y(t) = Bsin(\omega_y t - \sigma_y)$$

ω	角振動数	$0\langle\omega \leq$
	振幅比	$\frac{a}{b}$
σ	位相差	$0 \leq \sigma \leq 360$

また、次のようにも書かれる

$$x(t) = asin(\omega_x t + \sigma)$$
$$y(t) = bsin(t)$$

x,y のみとして素直にプログラムしてみる。

```
lissajou=: 4 : 0
NB. 1 2 _1r4p1 u steps_gldemo_ _5 5 100
'A B SIGMA'=: x
X0=:   2 o. SIGMA +  A * y
Y0=: 1 o. SIGMA +  B*y
X0;Y0
)

 plot 2 5 _1r4p1 lissajou_gldemo_  steps_gldemo_ _50 50 10000
```

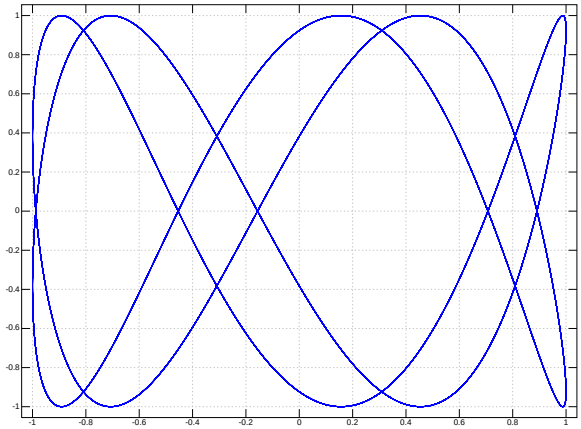
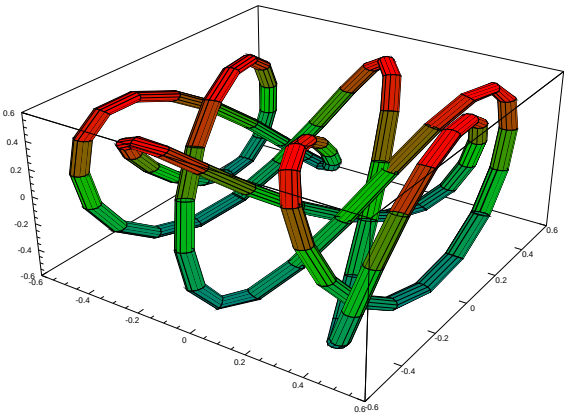


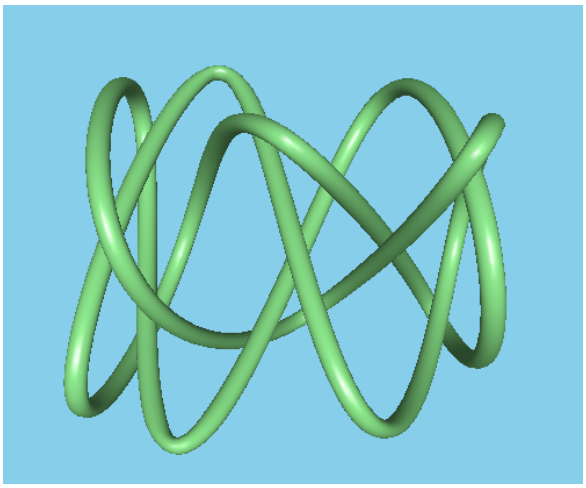
図 1　lissajou(2 5 _1r4p1)

J の *demo* の計算スクリプトで *3D* である。*gsmakefknot* は *OpenGL* のデータ生成関数、*gsdrawknot* が描画関数である。

```
fx=: 0: + 2&*
fy=: 1: + 5&*
fz=: 2.1"_ + 7&*
fn=: cos @ (fx,fy,fz) f.
sp=: o. (<:i.92) % 45
KNOT=: fn gsmakefknot sp;11;0.2

  plot <"2 |: KNOT_gldemo_
```





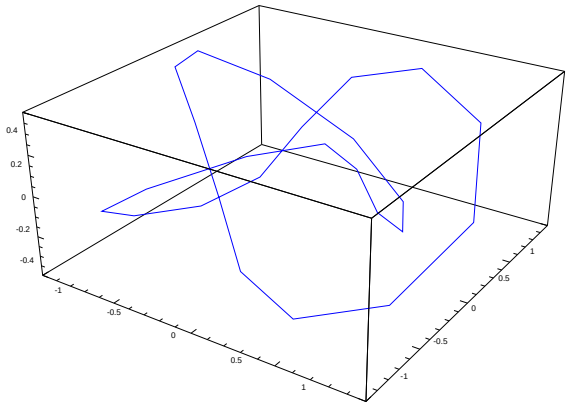
2.2 三つ葉模様

両端が繋がった *3* 次元の *plot* 用の関数が出来れば、*OpenGL* に落とし込めばよい

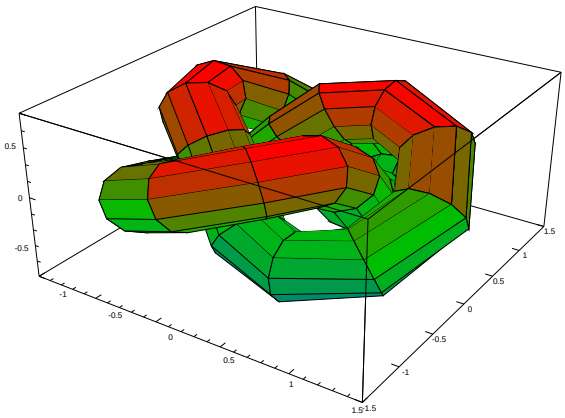
```
plot  <"2 |: TREFOIL_gldemo_
plot (fx_gldemo_ ;fy_gldemo_;fz_gldemo_) L:0 steps_gldemo_ 0 16r4p1 25
```

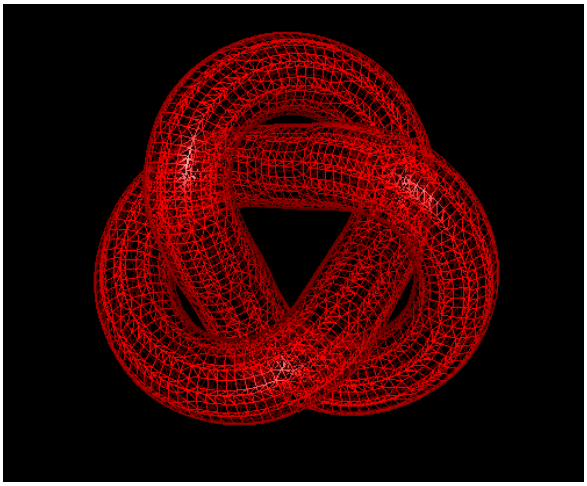
```
V=. "_
r=: 1: + 0.4 V * cos@(1.5&*)
fx=: r * cos
fy=: r * sin
fz=: 0.5 V * sin@(1.5&*)
fn=: (1.7 V * fx,fy,fz) f.
TREFOIL=: fn gsmakefknot (steps 0 16r4p1 25);11;2
```

plot



OpenGL 用に整形したもの





3　Surface 系（1）数理関数で作成

3.1　Drum

Drum はベッセル関数の *OpenGL* での描画である。関数を眺めてみよう。

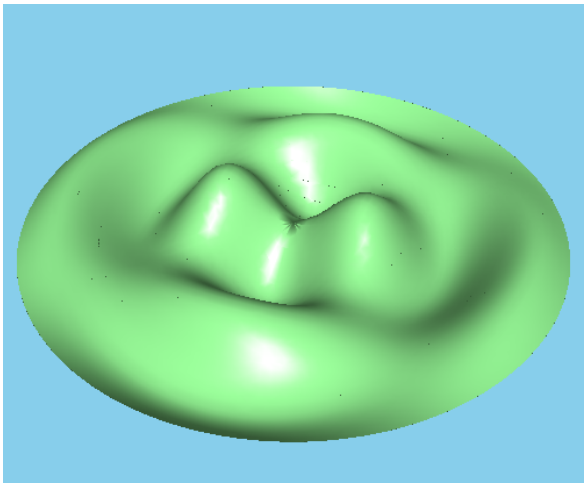


図 2　cube

```
   drum_gldemo_
3 : 0
R=. steps 0 1 21
T=. steps 0 36r16p1 21
X=. R */ cos T
Y=. R */ sin T
Z=. 0.1+0.4 * (BesselJ2 R*11.6) */ cos +:T
1.8 * gsmakexyz X;Y;Z
```

gsmakexyz は *izopenglutil* の関数である。

BesselJ2 は *Bessel* 関数。

X;Y;Z のサイズとその *plot*

```
   $ L:0  X;Y;Z
+-----+-----+-----+
|22 22|22 22|22 22|
+-----+-----+-----+
```

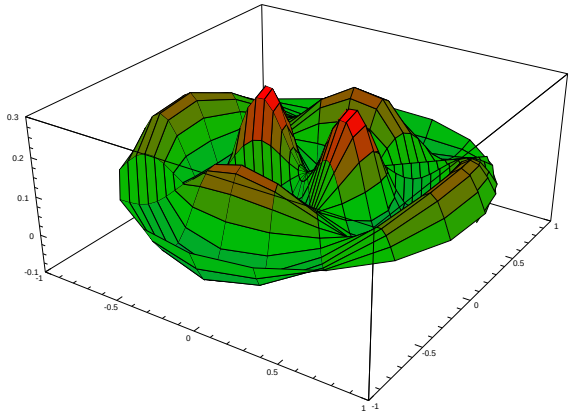


図 3　cube

gsmakexyz は *OpenGL* 用のデータ作成関数であり、*X;Y;Z* の *3* 面から *22* 行 *3* 列 *(x,.y,.z)* を *22* 面作成する。

各面はスライスに相当し、各面が *x　y　z* の *3* 列のデータとなる。*OpenGL* は *x y* で座標を組み合わせ、*z* で高さの値を取得する。

```
  $ L:0 <"2 drum ''
+----+----+----+----+----+----+----+----+----+----+
|22 3|22 3|22 3|22 3|22 3|22 3|22 3|22 3|22 3|22 3|
+----+----+----+----+----+----+----+----+----+----+.....
```

3.1.1　Bessel 関数

Bessel の微分方程式

$$x^2\frac{d^2y}{dx^2}+x\frac{dy}{dx}+(x^2-\alpha^2)y=0$$

その解は *Bessel* 関数である

$$J_\alpha(x)=\sum_{m=0}^{\infty}\frac{(-1)^m}{m!\Gamma(m+\alpha+1)}\left(\frac{x}{2}\right)^{2m+\alpha}$$

1 次の球 *Bessel* 関数

$$j_1(x)=\frac{sinx-xcosx}{x^2}$$

$j_1(x)$ の分子と分母を $f(x),g(x)$ とすると

$$f'(x) = cosx - (cosx - xsinx) = xsinx$$
$$g'(x) = 2x$$

$j(x)$ は原点を通る。

$$lim_{x\to 0} j_1(x) = lim_{x\to 0}\frac{f(x)}{g(x)} = 0$$

j_1 の *Bessel* 関数の定理から

$$j_1(x) = \frac{x}{2}\left(\frac{1}{\Gamma(2)} - \frac{1}{\Gamma(3)}\frac{x^2}{4} + \frac{1}{2\Gamma(3)}\frac{x^4}{16} - \cdots\right) = \frac{x}{2} - \frac{x^3}{16} + \frac{x^5}{384} - \cdots$$

微分すると

$$j_1'(x) = \frac{1}{2} - \frac{3}{16}x^2 + \frac{5}{384}x^4 - \cdots$$

(Bessel 関数の数式は数多くあるがこれが近いか*?)*

多項式の解を求めると

```
  p. 1r2 0 _3r16 0 5r384
+-----+--------------------------------+
|5r384|3.29637 _3.29637 1.87988 _1.87988|
+-----+--------------------------------+

 plot x; 1r2 0 _3r16 0 5r384&p.x=.  steps_gldemo_ _4 4 210
   pd 'eps /temp/bessel_1.eps'

   BesselJ2_gldemo_
3 : 0
t=. 1r128p1 * i.129
r=. (cos +:t) * cos (sin t) */ y
(+/ r * 1,(127$4 2),1) % 384
)
```

OpenGL の描画関数は *gsdrawsurface2* である

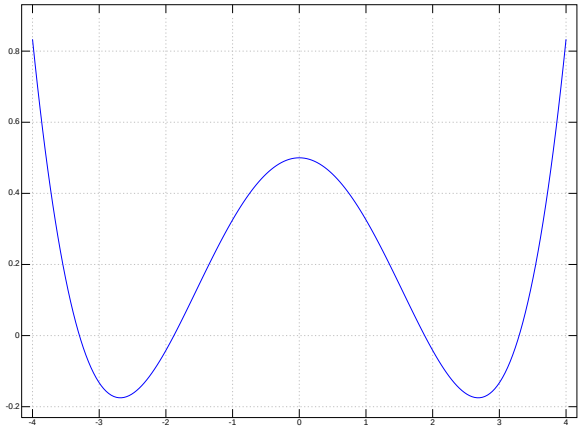
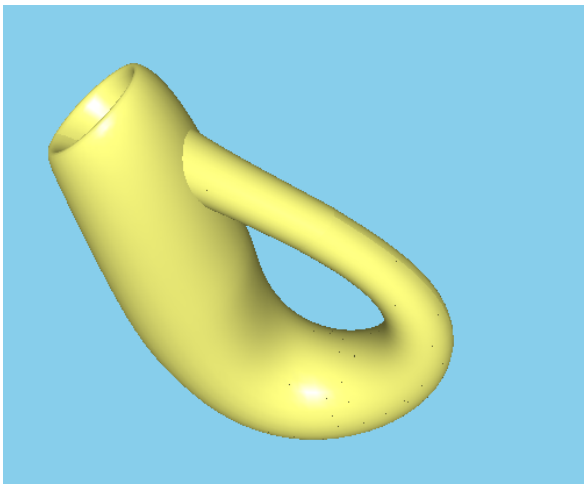


図 4　cube

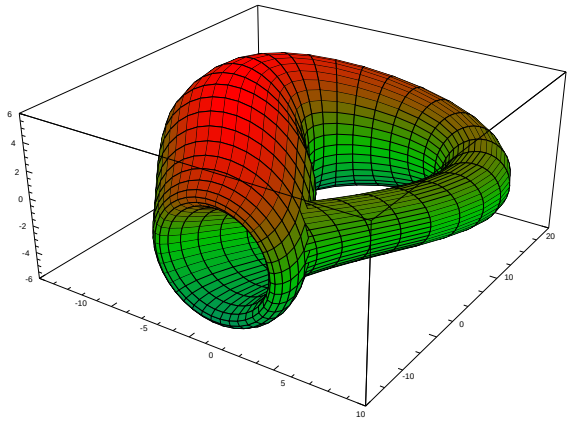
3.2　Klein

古代ペルシャのリュトン。



```
plot  klein2_gldemo_ ''
```

plot klein2 ”

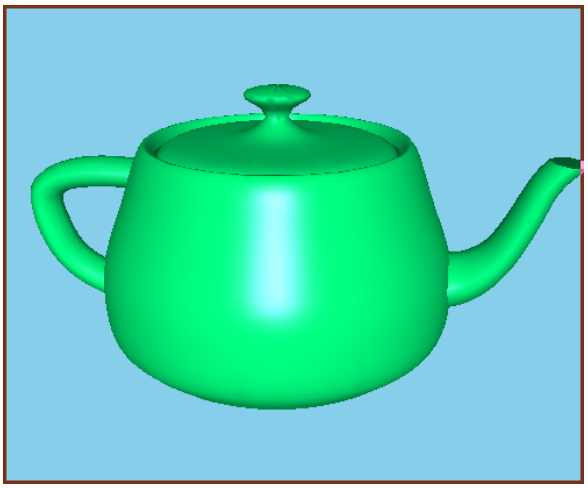


```
klein2=: 3 : 0
stp=. 19
u1=. steps 0 1p1, stp
u2=. }. steps 1p1 2p1, stp
u=. v=. u1,u2  NB. linear 0 to 2p1 (with steps 37)
r=. 4 * -. 0.5 * cos u NB. upward parabolic
j=. ((cos u1) */ cos v) , (0 * u2) +/ cos v + 1p1
x=. (6 * (cos u) * 1 + sin u) + r * j
y=. (16 * sin u) + r * ((sin u1), 0 * u2) */ cos v
z=. r * (0 * u) +/ sin v
DAT=. x;y;z
NB. 1.6 * gsfit11 gsmakexyz x;y;z
)
```

4　Surface 系（2）工芸

4.1　急須

陶芸や硝子/金属工芸教室。方眼紙と鉛筆でデッサンが必要である。

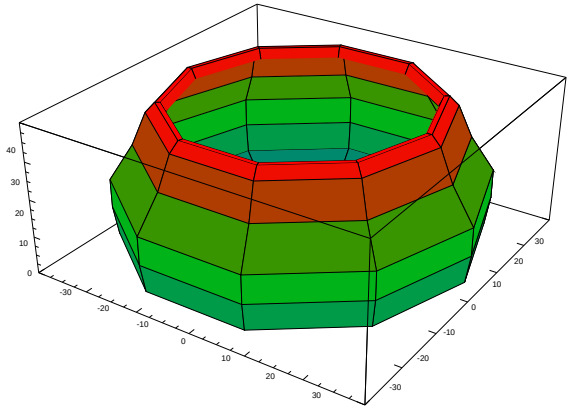


```
   gstonum_gldemo_
0&".;._2

 gssurfacerev_gldemo_
10&$: :(4 : 0)
```

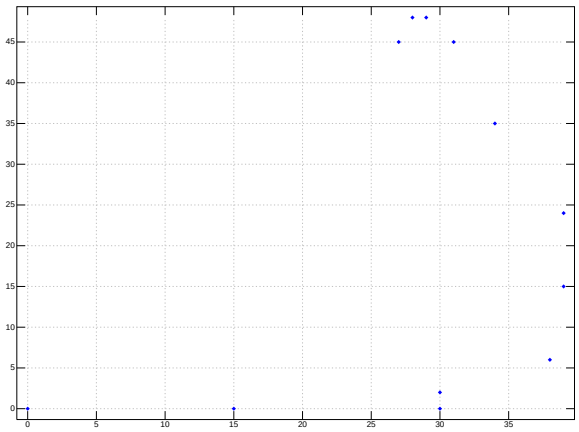
```
'r z'=. |: y
ix=. i. x+2
'x y'=. 0 1 |: r */ +. r. ix * 2p1%x
x,"0 1 y,"0 z
)

plot <"2 |:  BODY_gldemo_
```



図を見ると便宜上 10 角形だが OpenGL で最後に滑らかになる。*rev* は *revolution* であれば轆轤が回転するイメージ。

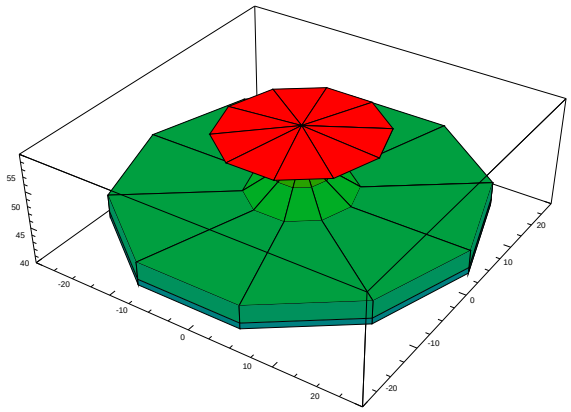
```
   BODY=: gssurfacerev gstonum 0 : 0
27 45
28 48
29 48
31 45
34 35
39 24
39 15
38  6
30  2
30  0
15  0
0   0
)
```



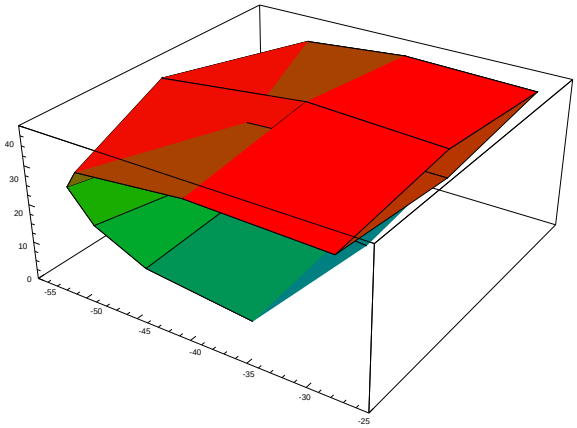
gssurfacerev の用法
BODY の数値は *BODY* の半分の *X,Y* 座標を示している（両側にすると間違える?）

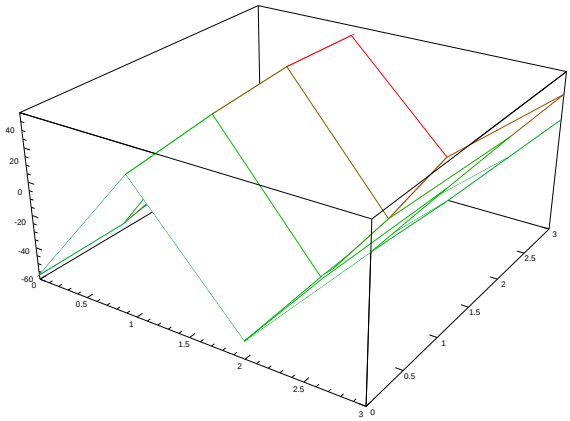
```
DA=:27 28 29 31 34 39 39 38 30 30 15 0
DB=:45 48 48 45 35 24 15  6  2  0  0 0
DAT=: DA,.DB

 'marker' plot  { |:  DAT
```



```
LID=: gssurfacerev gstonum 0 : 0
 0 59
12 58
 0 54
 4 51
 8 48
26 48
26 45
26 44
)
```



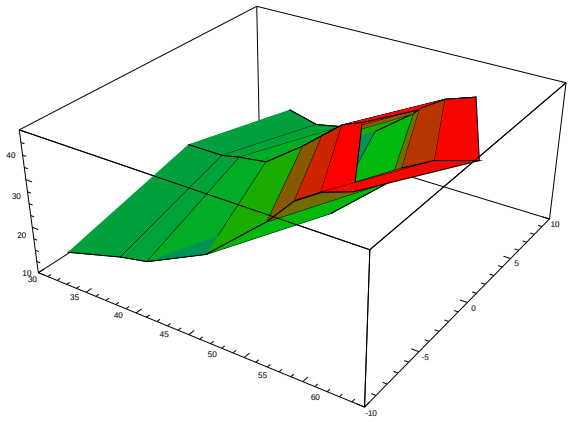


```
HANDLE=: gsmaketube gstonum 0 : 0
_35  8 _35 18 6
_46 11 _44 22 6
_55 21 _45 26 6
_57 32 _48 32 6
_56 40 _47 34 6
_41 45 _41 35 6
_28 43 _28 35 6
)
```

tube の作成。*tube* の座標は輪切りではなく縦方向に座標を持つ。

```
 3 2 1 0 1 2 3;0 1 2 3 4 ; |: DAT2
+-------------+---------+-------------------------+
|3 2 1 0 1 2 3|0 1 2 3 4|_35 _46 _55 _57 _56 _41 _28|
|             |         |  8  11  21  32  40  45  43|
|             |         |_35 _44 _45 _48 _47 _41 _28|
```

```
|              |          | 18   22   26   32   34   35   35|
|              |          |  6    6    6    6    6    6    6|NB.tube の太さ?
+--------------+----------+---------------------------+
```




```
SPOUT=: gsmaketube gstonum 0 : 0
34 27 34 11 10
38 27 39 12 9
40 28 43 14 9
43 29 50 21 8
47 36 56 34 7
49 40 57 38 6
51 44 60 42 6
53 46 64 45 6
)
```

```
paint=: 3 : 0
if. gsinit GS_LIGHT do.
  gsnewlist 1
  scale=. 0.03
  4 3 gsdrawsurface (0{VIEW) # BODY*scale
  4 3 gsdrawsurface (1{VIEW) # LID*scale
  4 3 gsdrawsurface (2{VIEW) # HANDLE*scale
  4 3 gsdrawsurface (3{VIEW) # SPOUT*scale
  gsendlist ''
end.
glCallList 1
gsfini''
)
```