Лекц: 3D Хувиргалт ба Шаталсан загвар

(3-D Transformations & Hierarchical models)

Ч.Цэнд-Аюуш(Ph.D)
Компьютерийн Ухааны Салбар
304 тоот

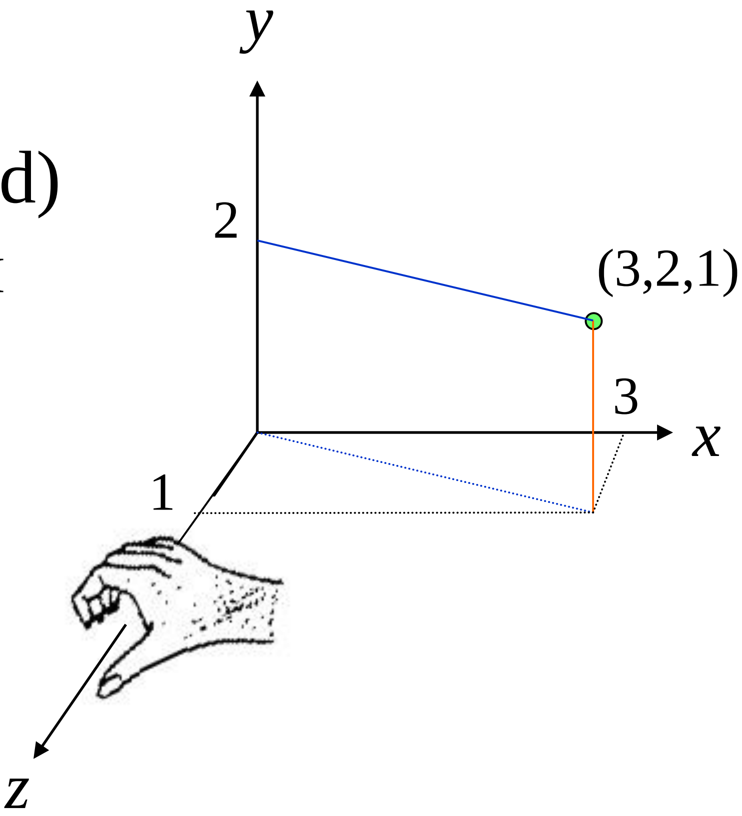
3D Хувиргалт ба шаталсан загвар

- Хувиргалтууд
  - Translation
  - Scaling
  - Rotation
  - Reflection
  - Координатын системийг өөрчлөх
- Шаталсан загвар

3D Хувиргалт

- Баруун гарын (Right-handed) 3D Декартын координатын систем
- Оройн цэгийн байрлал (3-tuple)

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



- homogeneous координатын системд 3 хэмжээст цэг

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \qquad w \neq 0.$$

- Декардын координат $\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$ ба homogeneous $\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$ координат хоорондын харилцан хамаарал нь

$$x' = x/w, \quad y' = x/w, \quad z' = z/w$$

Жнь, Декартын координатын цэг (*3*, *2*,*1*) нь homogeneous координатын (*3*, *2*, *1*, *1*) ба (*6*, *4*, *2*, *2*) цэгтэй харгалзан адил.

homogeneous дахь цэг (1, 2, 3, 0.5) нь Декартын цэг (2, 4, 6)-д харгалзана.

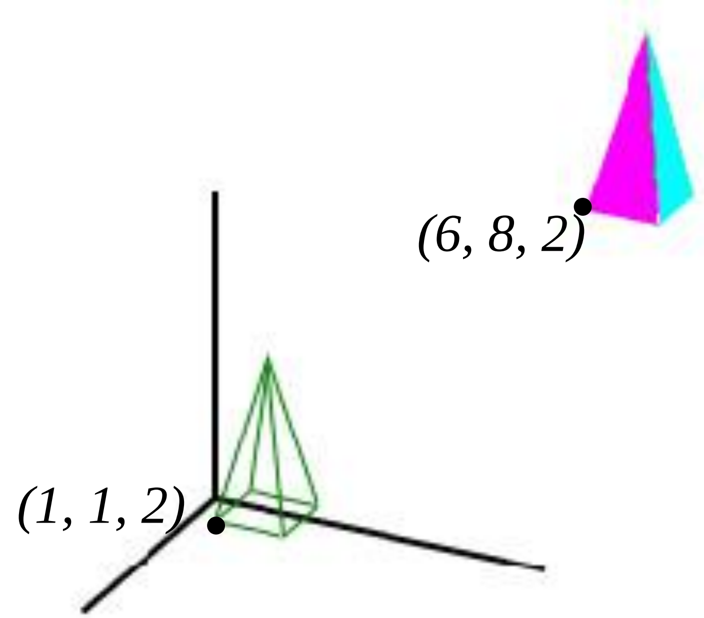
- 3D дэх хувиргалт

  объектийг шинэ байршилд шилжүүлэх

  $Tx = 6 - 1 = 5$

  $Ty = 8 - 1 = 7$

  $Tz = 2 - 2 = 0$



Ерөнхийдөө, $x' = x + Tx, \quad y' = y + Ty,$

$z' = z + Tz$

- Homogeneous координат системд

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z' \\ w \end{bmatrix}. \text{ Eg, } \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 7 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \\ 2 \\ 1 \end{bmatrix}$$

- *Opengl*: glTranslatef(Tx, Ty, Tz)

Translate

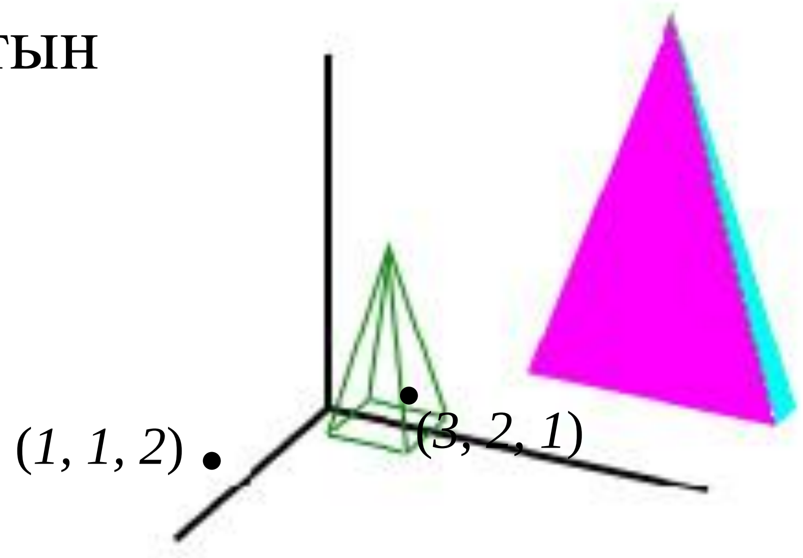
- Масштаб (Scaling) координатын эхээс хамааралтай

Шинэ байршил:

$$x' = Sx \times x,$$

$$y' = Sy \times y,$$

$$z' = Sz \times z$$



- Homogeneous coordinate system

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}. \text{ Eg, } \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 1 \end{bmatrix}$$

- *Opengl*: glScalef(Sx, Sy, Sz)

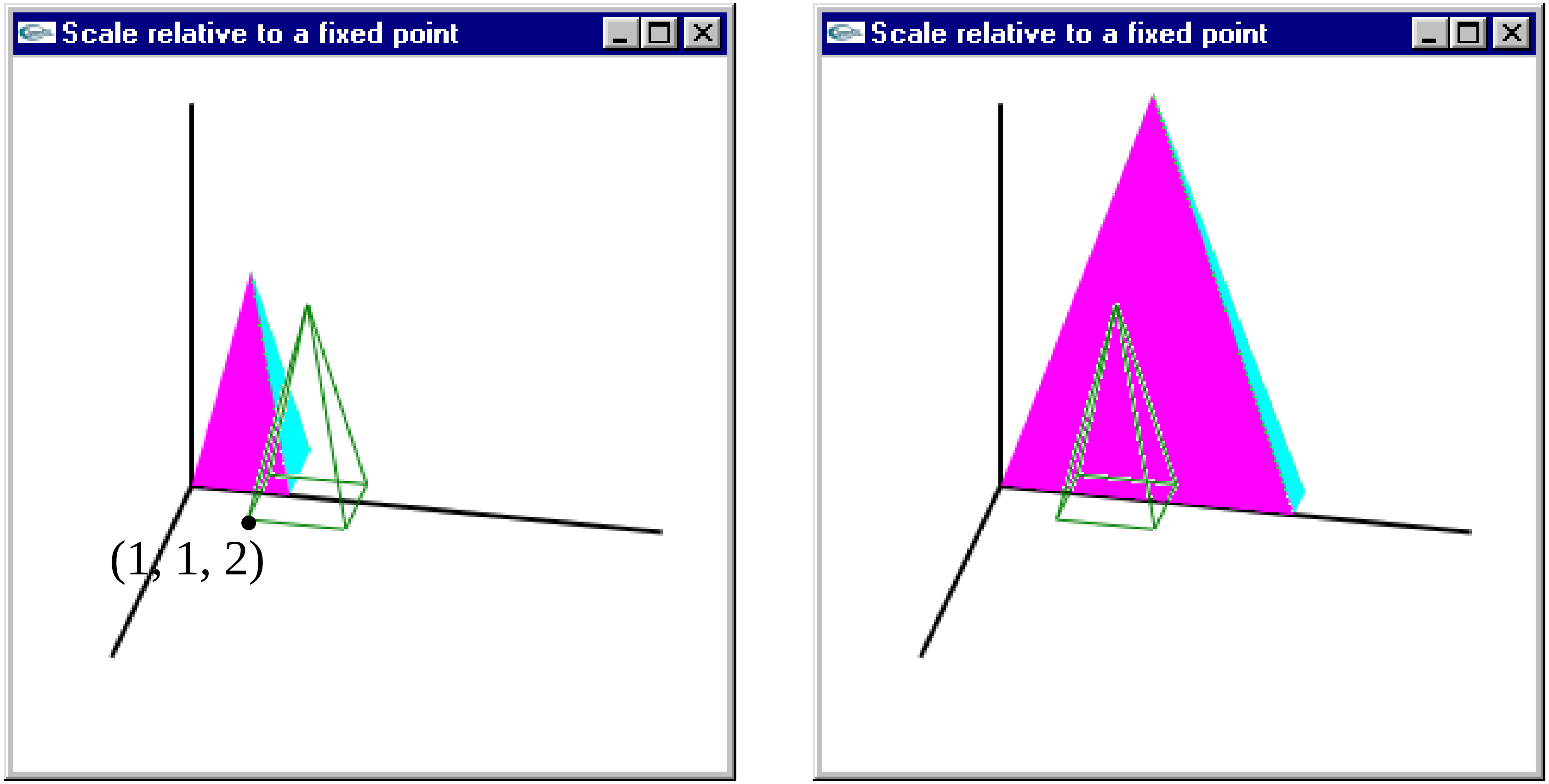
Scale

- Scaling relative to a fixed point, eg, (1, 1, 2)

1. Цэгийг координатын эх рүү шижлүүлнэ

$$Tx = -1,\ Ty = -1,\ Tz = -2$$

2. scaling гүйцэтгэнэ.

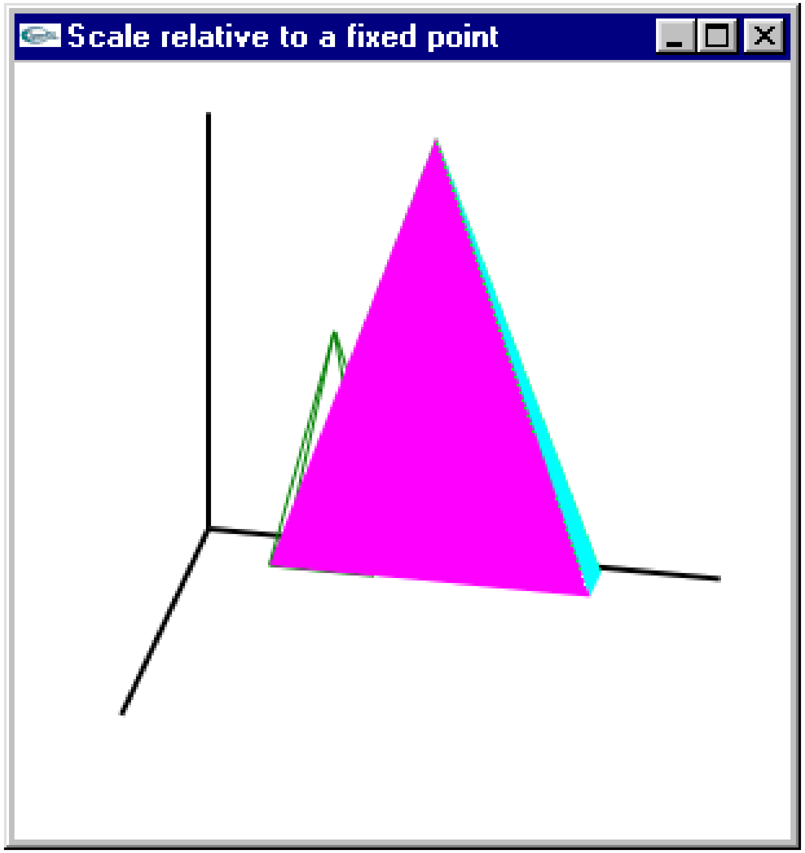


3. Backward translation

Tx = 1, Ty = 1, Tz = 2

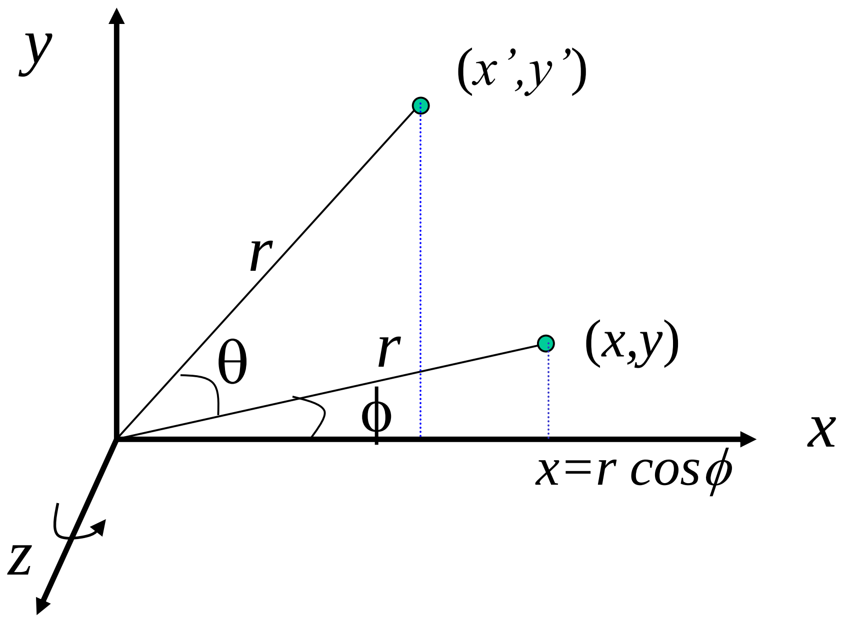
- Хувиргалтуудыг *Opengl-д урвуу дараалаар тогтооно*

```
glTranslatef( 1., 1., 2.);
glScalef( Sx, Sy, Sz);
glTranslatef( -1., -1., -2.);

pyramid();
```



Scale 2

- z-тэнхлэгийн дагуу эргүүлэх (3D)



$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}.$$

- Декарт,

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

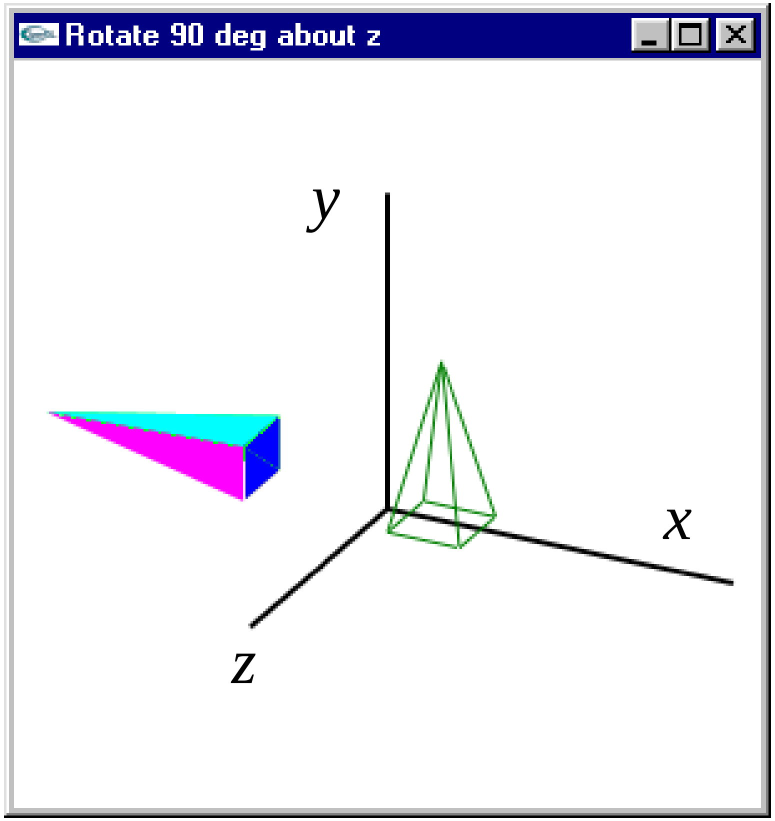
$$z' = z$$

- homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

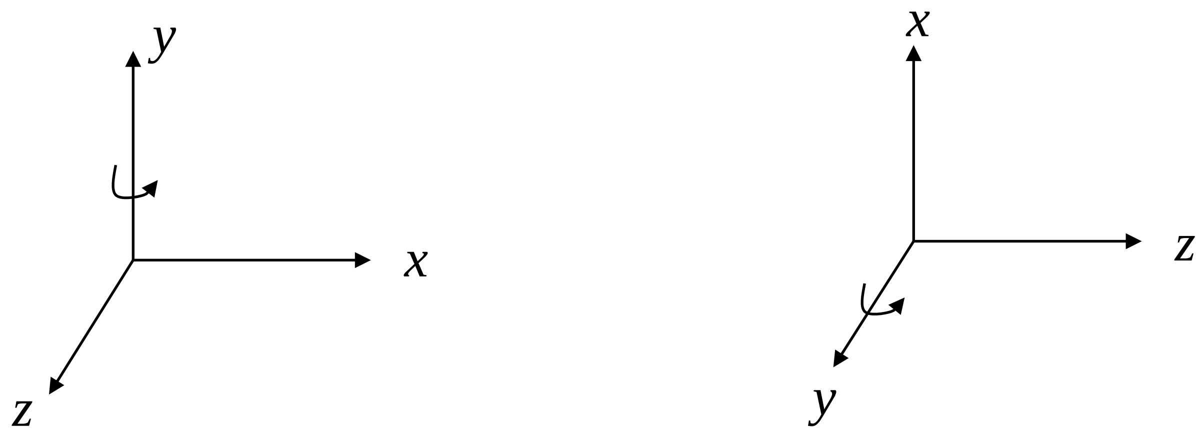
- *Opengl*:

```
glRotatef( degree, 0., 0., 1. );
```



RotateZ

- y–тэнхлэг эргэх (3D)



- Координатын тэнхлэгийг дахин зурах ба энэ нь өмнөхтэйгөө адил харагдана.
  1. уламжлалт z тэнхлэгийн чиглэлд y тэнхлэгийг зурна
  2. x-тэнхлэгийг дээш нь зурнаы.
  3. z-тэнхлэгийг баруун гарын дүрэм ашиглан тодорхойлно.

- x, y, z тэнхлэгийг z, x, y сольж, тус бүрд нь өмнөх томъёог авч үзвэл

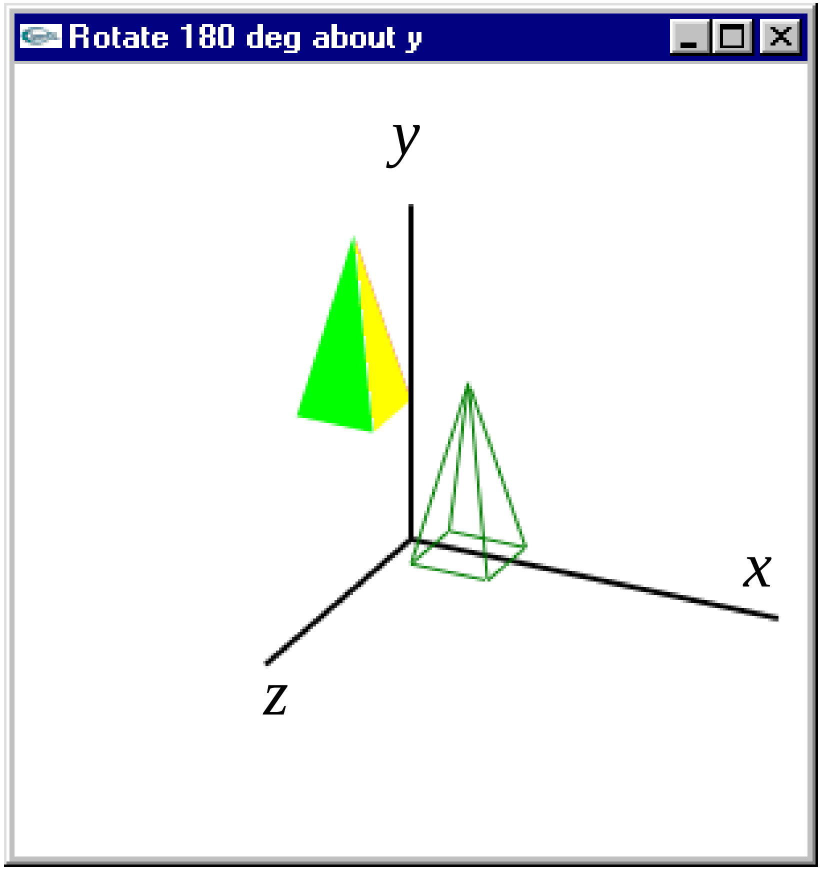
$$z' = z\cos\theta - x\sin\theta$$
$$x' = z\sin\theta + x\cos\theta$$
$$y' = y$$

- homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

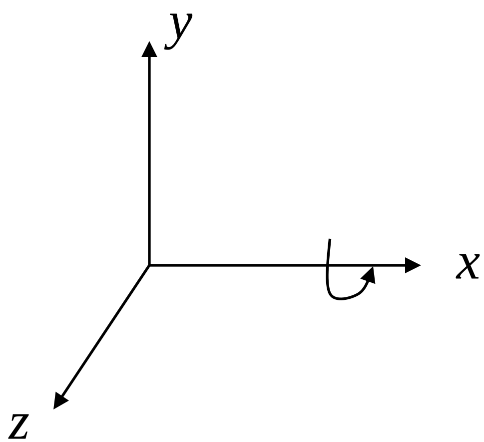
- *Opengl*:

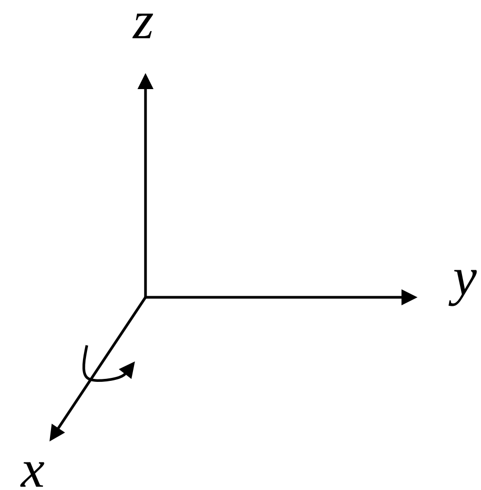
glRotatef(degree, 0., 1., 0.);



RotateY

- *x*–тэнхлэг дагуу эргүүлэх (3D)





- Координатын тэнхлэгийг дахин зурах ба өмнөх нөхцөлтэй ижил харагдана.
  1. x тэнхлэгийг z тэнхлэгийн чиглэлд зурна
  2. y тэнхлэгийг x-тэнхлэг чиглэлд зурна.
  3. *z*-тэнхлэгийг баруун гарын дүрэм ашиглан тодорхойлно.
- *x*, *y*, *z* тэнхлэгийг *y*, *z*, *x болгон сольж*, өмнөх томъёоны дагуу

$$y' = y\cos\theta - z\sin\theta$$

$$z' = y\sin\theta + z\cos\theta$$

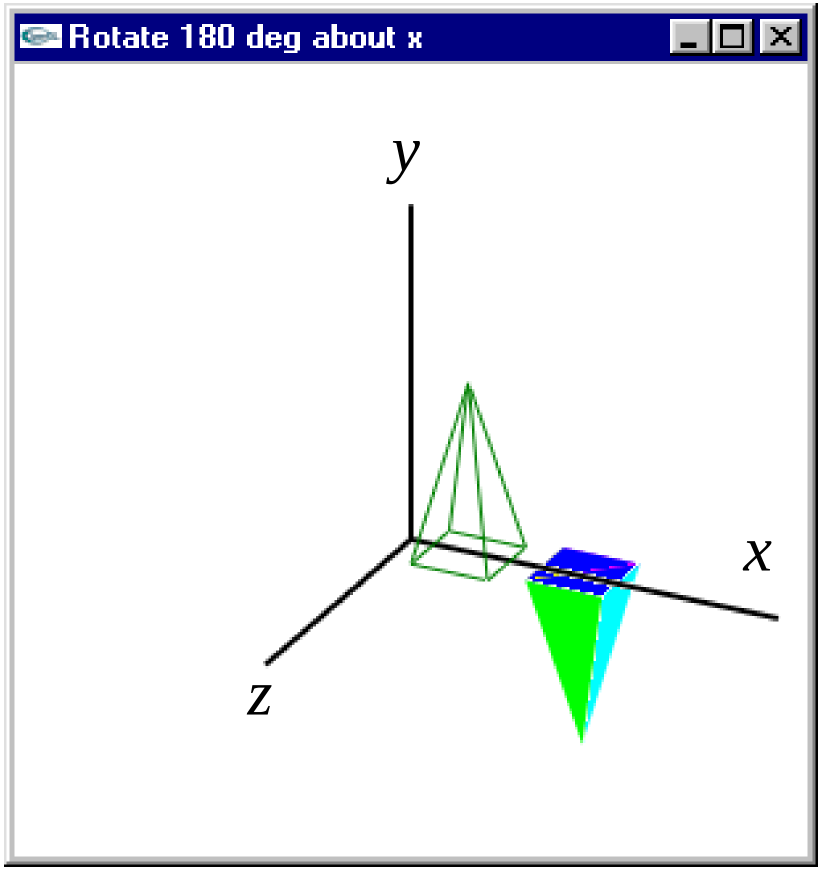
$x' = x$ *болно.*

- homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- *Opengl*:

glRotatef(degree, 1., 0., 0.);

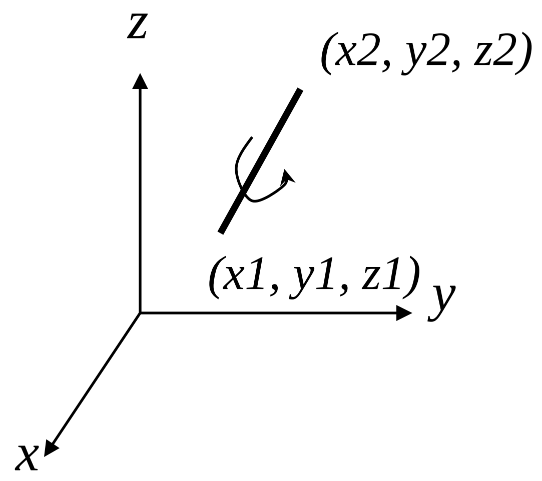


RotateX

Дурын тэнхлэг тойруулан эргүүлэх

тэнхлэг ($x1, y1, z1$),

($x2, y2, z2$) дамжсан гэж үзье



- 1 алхам

  ($x1, y1, z1$) координатын эх рүү шилжүүлэх

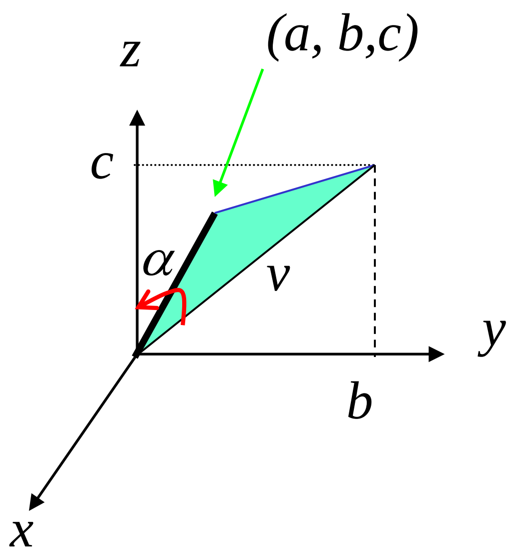
$$a = x_2 - x_1$$

$$b = y_2 - y_1$$

$$c = z_2 - z_1$$

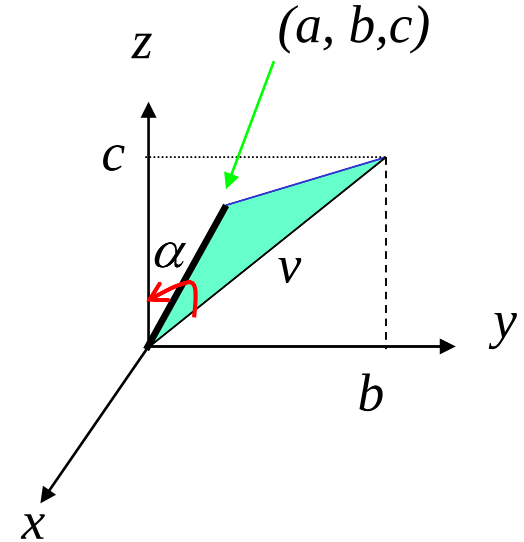
$$v = \sqrt{b^2 + c^2}$$

$$\alpha = \tan^{-1}(b / c)$$



- 2 алхам

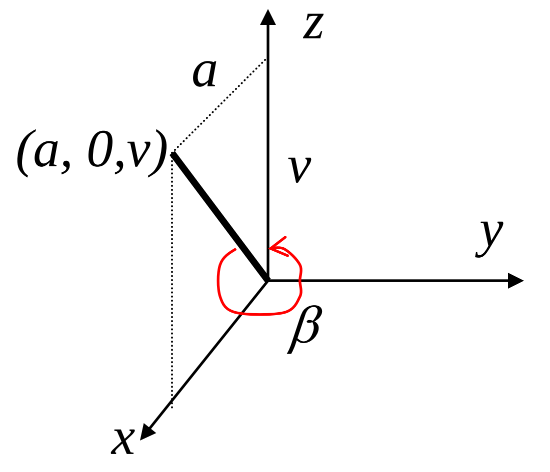
  X тэнхлэгийг x-z хавтгайд байхаар x тэнхлэгийг α-р эргүүлнэ.



- 3 алхам

  Y тэнхлэгийг z тэнхлэгтэй давхцуулахаар β -ээр эргүүлнэ

$$\beta = -\tan^{-1}(a/v)$$



- 4 алхам

  Z тэнхлэгийн дагуу эргүүлнэ

- 5 алхам (Reverse of Step 3)

  y-axis дагуу β *өнцөгөөр эргүүлнэ*.

- 6 алхам (Reverse of Step 2)

  x-axis дагуу α өнцөгөөр эргүүлнэ.

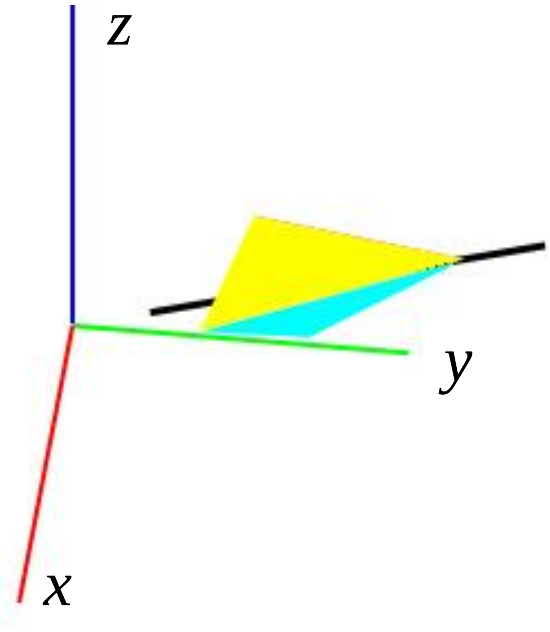
- 7 алхам (Reverse of Step 1)

  координатын эхээс (*x1*, *y1*, *z1*) рүү шилжүүлнэ.

```
//Rotate deg degree along the axis that passes through
//  (x1, y1, z1) and (x2, y2, z2)
    double  a = x2 - x1,  b = y2 - y1,  c = z2 - z1;
    double  v = sqrt(b * b + c * c);
    double  alpha = atan( b/c)*180./3.1416;
    double   beta = -atan(a/v)*180./3.1416;

    glTranslatef(x1, y1, z1);              //Step 1

    glRotatef( -alpha, 1., 0., 0.);        //Step 2
    glRotatef( -beta, 0., 1., 0.);         //Step 3

    glRotatef( deg, 0., 0., 1. );          //Step 4

    glRotatef( beta, 0., 1., 0.);          //Step 5
    glRotatef( alpha, 1., 0., 0.);         //Step 6

    glTranslatef(-x1, -y1, -z1);//Step 7

    pyramid();
```

OpenGL дээр дурын тэнхлэгийг тойрон эргэх тохиромжтой арга

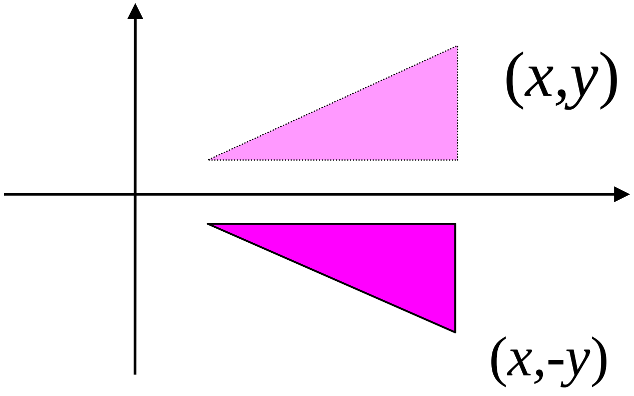
```
//Rotate deg degree along the axis that passes
//   through (x1, y1, z1) and (x2, y2, z2)

glTranslatef(x1, y1, z1)
glRotatef( deg, x2-x1, y2-y1, z2-z1);
glTranslatef( -x1, -y1, -z1)
pyramid();
```

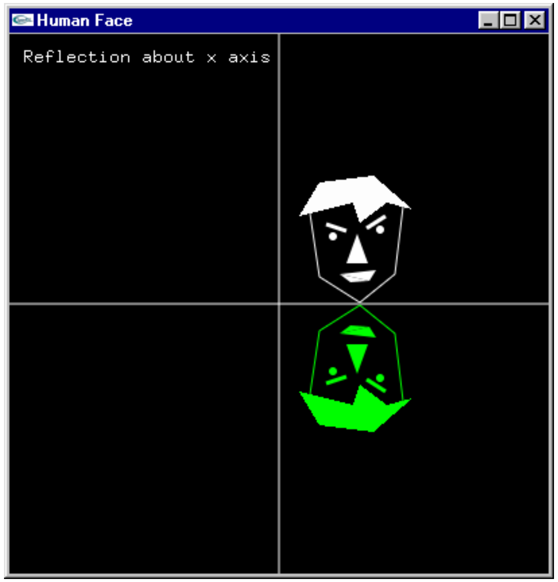


RotateA

> *x* тэнхлэгийн дагуу толин тусгал (2D)

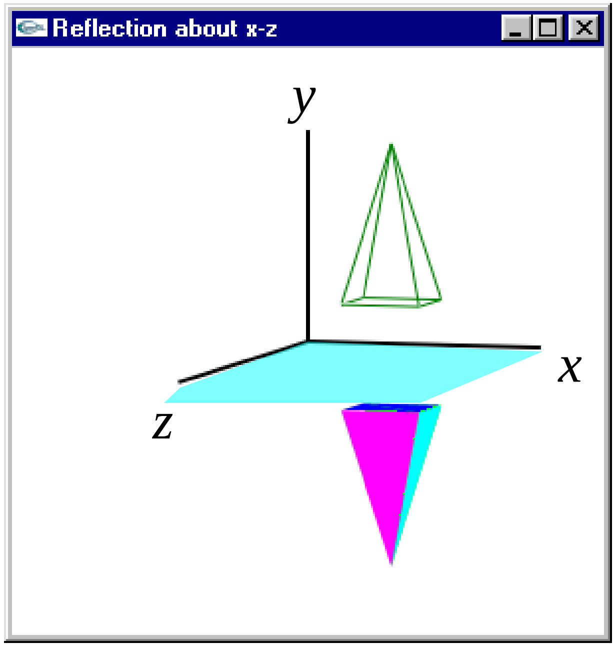
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}.$$





> 3-D counterpart is the reflection about the *x*-*z* plane

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$



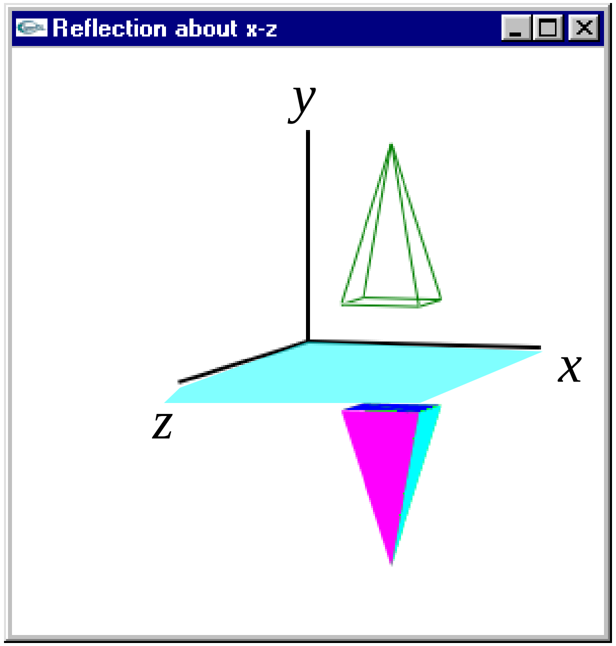
- Хувиргалтын матриц нь *Sx* = 1, *Sy* = -1 and *Sz* = 1 scaling матрицтай ижил. Иймд толин тусгалыг дараах функцээр хэрэгжүүлэх боломжтой.

glScalef(1.0, -1.0, 1.0)

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

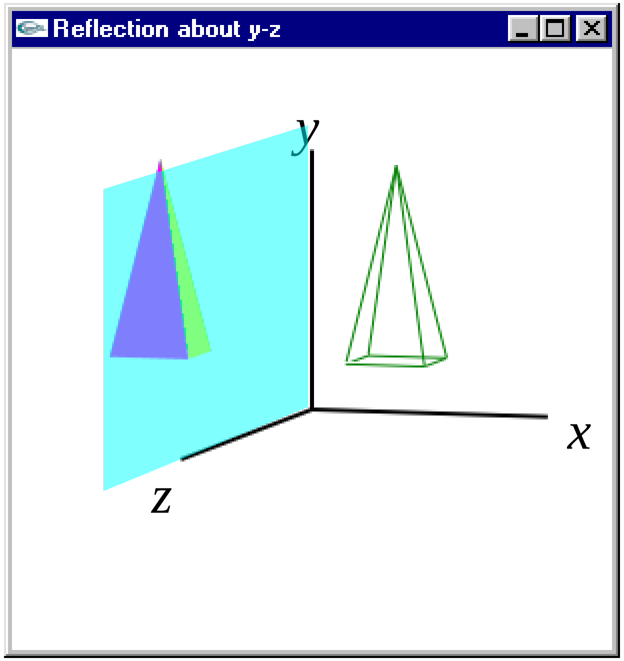
- Жишээлбэл

```
glPushMatrix();
      glScalef( 1.0, -1.0, 1.0);
       pyramid();
glPopMatrix();
```

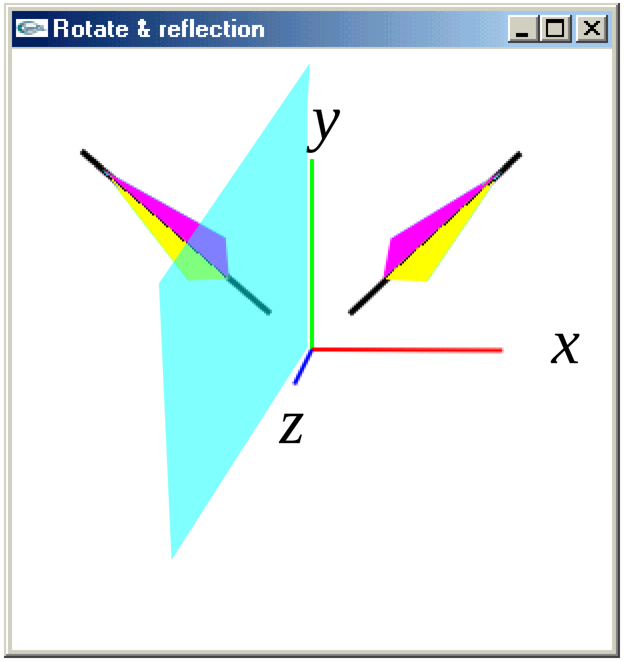


- *y*-z хавтгайд хамааралтай толин тусгал

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$




```
glPushMatrix();
      glScalef( -1.0, 1.0, 1.0);
      pyramid();
glPopMatrix();
```



rotate&reflect y-z

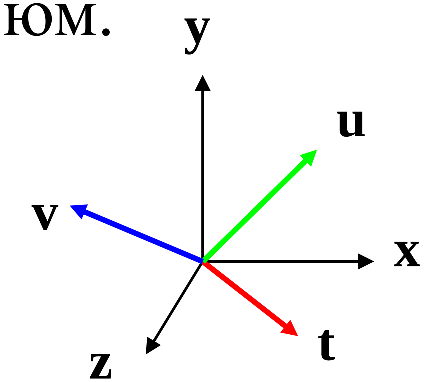
- *x*-*y* хавтгайд хамааралтай толин тусгал

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

```
glPushMatrix();
    glScalef( 1.0, 1.0, -1.0);
    pyramid();
glPopMatrix();
```

Координатын системийг өөрчлөх

- x-y-z координатын хувьд вектор (*x*, *y*, *z*) илэрхийлнэ. Тухайлбал **x** = (1, 0, 0), **y** = (0, 1, 0) ба **z** = (0, 0, 1). **x**, **y**, **z** нь тэнхлэгүүдийг тодорхойлдог харилцан перпендикуляр гурван нэгж вектор юм.



- Өгөгдсөн **t**, **u**, ба **v** нь харилцан перпендикуляр гурван нэгж вектор юм. **t**, **u**, болон **v** тодорхойлогдсон тэнхлэгийг ашиглан шинэ координатын систем үүсгэж болно. Орон зайн дурын цэгийг (t, u, v) гэж дүрсэлж болно.
- Цэгийн хувьд (x, y, z) -ээс (t, u, v) хэрхэн тооцоолох вэ?

- Координатын системийг өөрчлөх томъёо нь

$$\begin{bmatrix} t \\ u \\ v \\ w \end{bmatrix} = \begin{bmatrix} x_t & y_t & z_t & 0 \\ x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

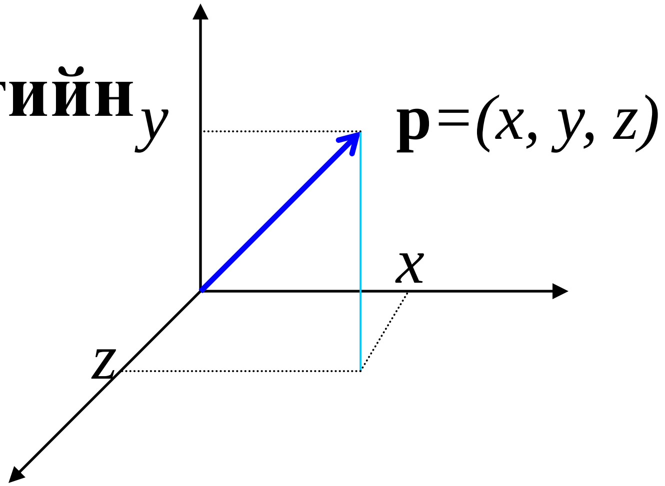
- x-y-z координат дахь **p** = (*x*, *y*, *z*) векторыг авч үзье. **p** –н эхний компонент x нь x-тэнхлэг дээрх **p**-ийн проекцийн урт байна.

  Тус компонент нь **p** ба **x** –н **цэгийн үржвэр**тэй тэнцүү

  $$\mathbf{x} = (1,0,0)$$

  $$\mathbf{p} = (x, y, z)$$

  $$\mathbf{p} \bullet \mathbf{x} = x \times 1 + y \times 0 + z \times 0 = x$$



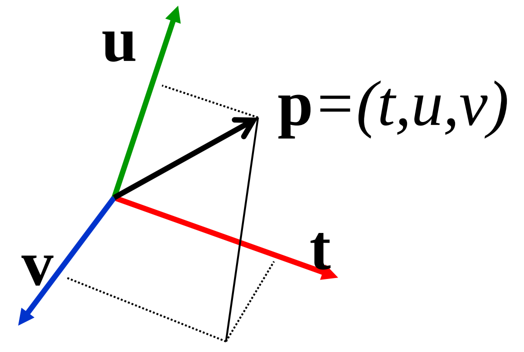
- **p** ба нэгж вектор **q** үржвэр нь **q** дээрх **p** –н проекцийн урт.

  $$\mathbf{p} = (x, y, z)$$

  $$\mathbf{q} = (x_q, y_q, z_q)$$

  $$\mathbf{p} \bullet \mathbf{q} = x \times x_q + y \times y_q + z \times z_q$$

- x-y-z координат систем дээр гэж үзье. **t**, **u** ба **v** дараах байдлаар илэрхийлнэ



$$\mathbf{t} = (x_t, y_t, z_t)$$

$$\mathbf{u} = (x_u, y_u, z_u)$$

$$\mathbf{v} = (x_v, y_v, z_v)$$

- Эдгээр векторууд шинэ координат систем үүсгэдэг. t-u-v систем дэх **p** вектор (*t*, *u*, *v*) координаттай болох юм.
- *t* эхний компонент гэж тэмдэглэн, **t** дээрх **p** –ийн проекц дараах томъёогоор тодорхойлогдоно.

$$t = \mathbf{p} \bullet \mathbf{t} = x \times x_t + y \times y_t + z \times z_t$$

Similarly,

$$u = \mathbf{p} \bullet \mathbf{u} = x \times x_u + y \times y_u + z \times z_u$$

$$v = \mathbf{p} \bullet \mathbf{v} = x \times x_v + y \times y_v + z \times z_v$$

$$t = \mathbf{p} \bullet \mathbf{t} = x \times x_t + y \times y_t + z \times z_t$$

$$u = \mathbf{p} \bullet \mathbf{u} = x \times x_u + y \times y_u + z \times z_u$$

$$v = \mathbf{p} \bullet \mathbf{v} = x \times x_v + y \times y_v + z \times z_v$$

- Эдгээр томъёог матриц ба вектораар илэрхийлбэл дараах байдалтай болно.

  (homogeneous координатын системээр өргөтгөсөн).

$$\begin{bmatrix} t \\ u \\ v \\ w \end{bmatrix} = \begin{bmatrix} x_t & y_t & z_t & 0 \\ x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- Жишээ авч үзье.

t=(.707, 0, .707), **u** = (0, 1, 0), **v** = (-.707, 0, .707)

- Эдгээр нь нэгж векторууд юм: **t** • **t** = **u** • **u** = **v** • **v** = 1

- Эдгээр нь харилцан перпендикуляр юм: **t** • **u** = **u** • **v** = **v** • **t** = 0

- x-y-z систем дэх **p** = (1,2,1) гэж үзье. t-u-v систем дэх **p** –ийн координат нь

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \begin{bmatrix} .707 & 0 & .707 \\ 0 & 1 & 0 \\ -.707 & 0 & .707 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.414 \\ 2 \\ 0 \end{bmatrix}$$

- Солих коодинатын системийн матрицийг **M** болгоё. Сүүлийн жишээнд,

$$M = \begin{bmatrix} .707 & 0 & .707 \\ 0 & 1 & 0 \\ -.707 & 0 & .707 \end{bmatrix}.$$

- **M** шилжилт нь үүний урвуу байна. Жнь $\mathbf{M}^{-1} = \mathbf{M}^{T}$.

$$\begin{bmatrix} .707 & 0 & .707 \\ 0 & 1 & 0 \\ -.707 & 0 & .707 \end{bmatrix}\begin{bmatrix} .707 & 0 & -.707 \\ 0 & 1 & 0 \\ .707 & 0 & .707 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Гаргалгаа
  - **t, u, v** нь нэгж векторууд тул диагональ элементүүд нь 1 байна.
  - Диагональ бус элементүүд нь харилцан перпендикуляр тул 0 байна.

Өмнөх жишээний t–u хавтгайд харгалзах толин тусгал

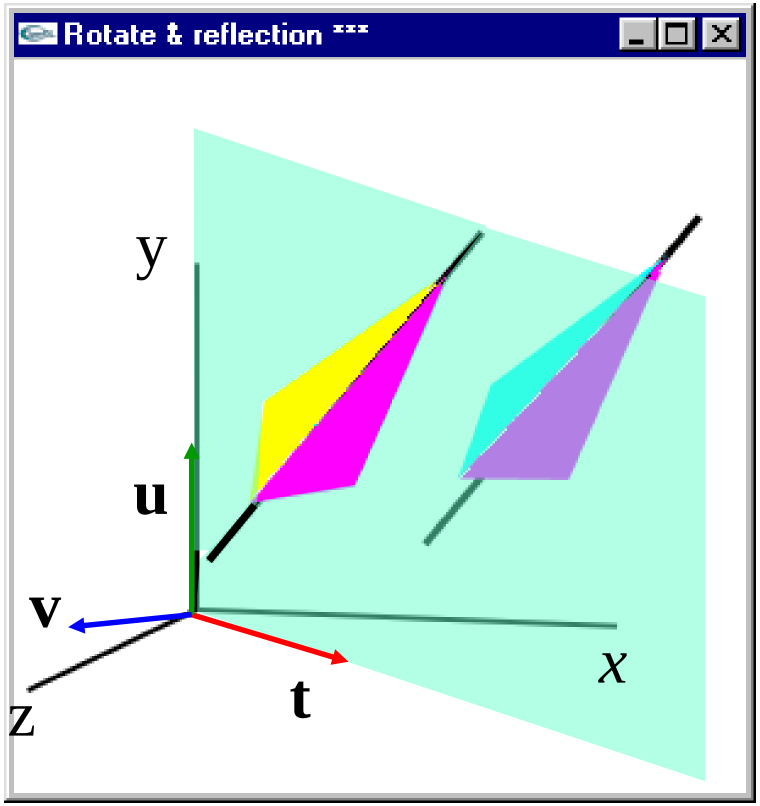
- x–y–z координатыг t–u–v координат руу хөрвүүлэх (**M**-ээр үржүүлнэ)
- t–u хавтгайд харгалзах толин

тусгал

glScalef(1.0, 1.0, -1.0);

- Хөрвүүлэлтийн урвуу
  (**M**-ийн урвуугаар үржүүлнэ)

  ($\mathbf{M}^{-1} = \mathbf{M}^{T}$)



Rotate reflect2

- **t**=(.707, 0, .707), **u** = (0, 1, 0), **v** = (-.707, 0, .707)

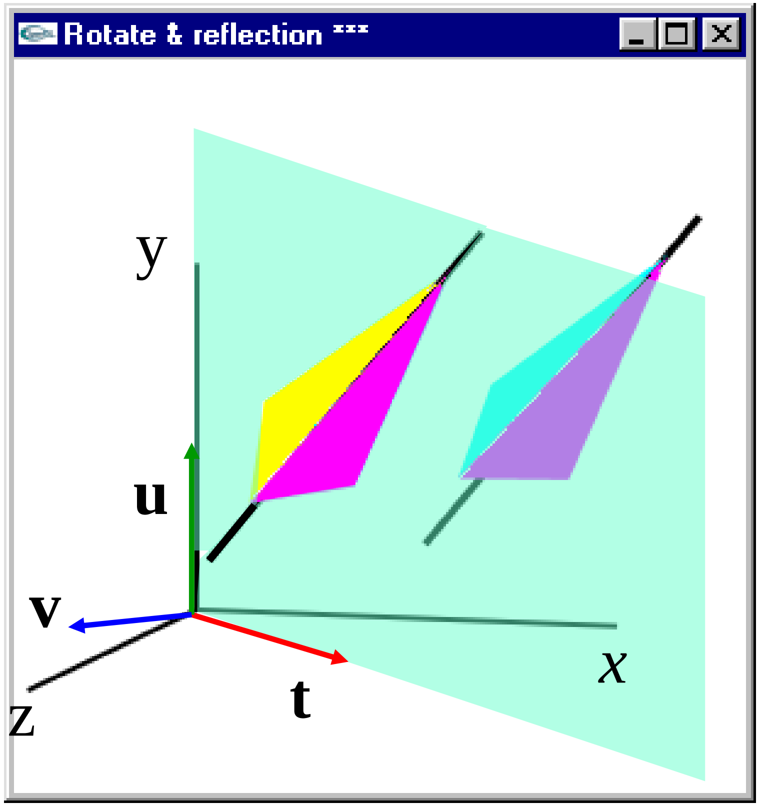
GLfloat M[16] = { .707, 0, -.707, 0,   0, 1, 0., 0, .707, 0, .707, 0, 0, 0, 0, 1};

$$M = \begin{bmatrix} .707 & 0 & .707 & 0 \\ 0 & 1 & 0 & 0 \\ -.707 & 0 & .707 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

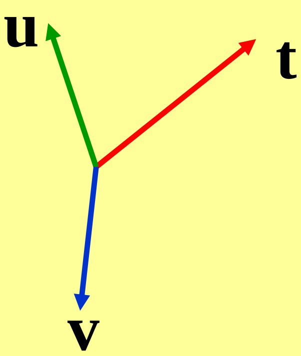
//Column by Column

rotate&reflect2

```
GLfloat  MT[16] = { .707, 0, .707, 0,   0, 1,
                    0., 0,   -.707, 0, .707, 0,
                    0, 0, 0, 1};

glPushMatrix();
        glMultMatrixf( MT);
        glScalef( 1.0, 1.0, -1.0);
        glMultMatrixf( M);
        pyramid();
glPopMatrix();
```



Cross product of two vectors



Let $\mathbf{t} = \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix}$, $\mathbf{u} = \begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix}$ and $\mathbf{t}$ is not parallel to $\mathbf{u}$.

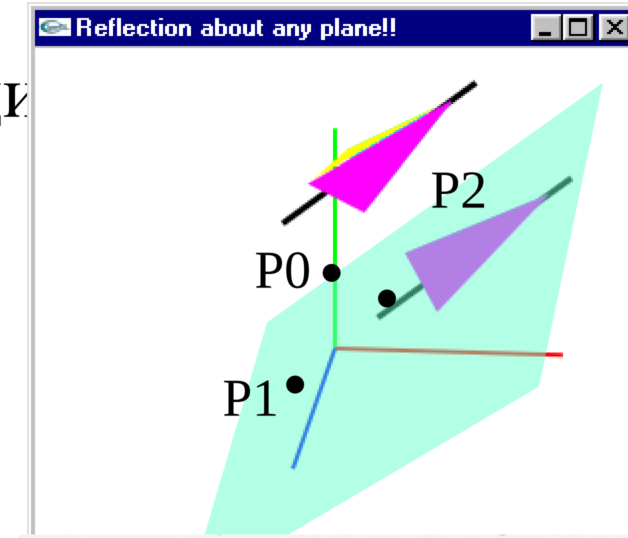
The cross product of $\mathbf{t}$ and $\mathbf{u}$ is written as $\mathbf{v} = \mathbf{t} \times \mathbf{u}$.

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} y_t z_u - y_u z_t \\ -(x_t z_u - x_u z_t) \\ x_t y_u - x_u y_t \end{bmatrix}.$$ $\mathbf{v}$ is perpendicular to both $\mathbf{t}$ and $\mathbf{u}$.

In right-handed coordinate system. When we curl the fingers of our right hand from $\mathbf{t}$ to $\mathbf{u}$, the thumb points to the direction of $\mathbf{v}$.

Example. Let $\mathbf{t} = (1, 0, 0)$ and $\mathbf{u} = (0, 1, 0)$. $\mathbf{v} = \mathbf{t} \times \mathbf{u} = (0, 0, 1)$.

Яливаа хавтгайд харгалзах толин тусгал (advanced)



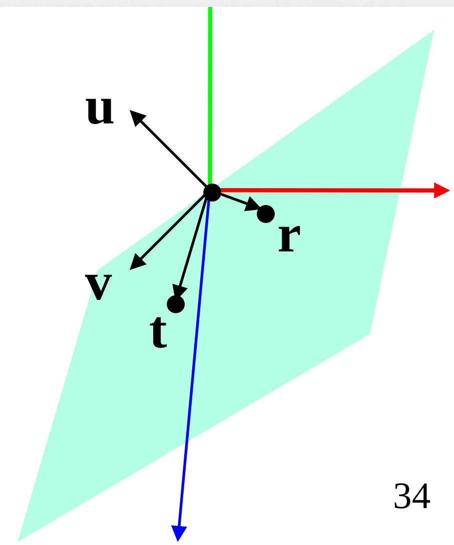
- Хавтгай дээрх non-colinear цэгүүди сонгоно (P0, P1, P2).
- Хавтгайг шилжүүлж, P0 эхлэл

(origin) болно

glTranslatef(-x0, -y0, -z0);

- **t** = P1 - P0 ба **r** = P2 - P0 нь өөрчлө хавтгай дээрх хоёр вектор юм
- **u = t × *r*, v = t × u**
- **u** ⊥ to the plane
- **t** ба **v** хавтгайд оршино
- **t, u** ба **v нь** харилцан перпендикуляр

Reflect Rotate



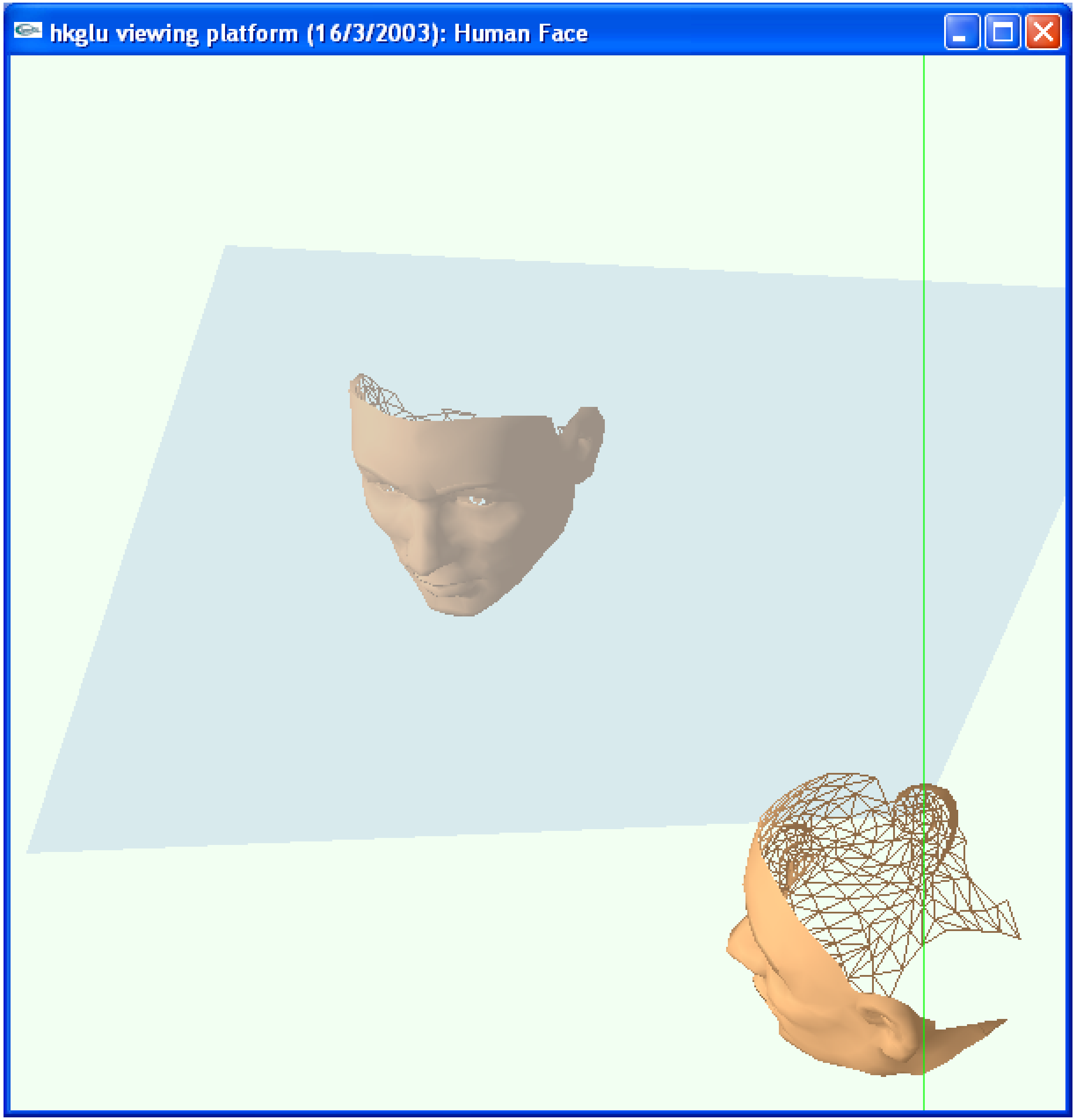
- Normalize **t, u** and **v**
- Change the coordinates from x-y-z to t-u-v
- Reflect relative to t-v plane
- Change the coordinates back to x-y-z
- Backward translation

```
//Reflect about the plane  that passes through P0:(x0,y0,z0),
//   P1:(x1, y1, z1),  and P2:(x2, y2, z2)
void hkgluReflect( double x0,  double y0,  double z0,  double x1,  double y1,
                   double z1,  double x2,  double y2,  double z2 )  {

    //t: (xt, yt, zt),  r: (xr, yr, zr),  u: (xu, yu, zu), v: (xv, yv, zv)
    double  xt, xr, xu, xv,   yt, yr, yu, yv,   zt, zr, zu, zv,  a;

    xt = x1 - x0;  yt = y1 - y0;  zt = z1 - z0;  //t=P1-P0
    xr = x2 - x0;  yr = y2 - y0;  zr = z2 - z0;  //r=P2-P0
```

```
// u = t × r.  × is the cross-product. u is perpendicular to the plane
xu = yt*zr - yr*zt;   yu = -(xt*zr - xr*zt);   zu = xt*yr - xr*yt;

// v = t × u . v is on the translated plane
xv = yt*zu - yu*zt;  yv = -(xt*zu - xu*zt);  zv = xt*yu - xu*yt;

//Normalize  t,  u,  v
a = sqrt( xt*xt + yt*yt + zt*zt);
if (a != 0.) { xt = xt/a;  yt = yt/a;  zt = zt/a; }

a = sqrt( xu*xu + yu*yu + zu*zu);
if (a != 0.) {xu = xu/a;  yu = yu/a;  zu = zu/a; }

a = sqrt( xv*xv + yv*yv + zv*zv);
if (a != 0.) {xv = xv/a;  yv = yv/a;  zv = zv/a; }

//M is the matrix that change the coordinate system from x-y-z to t-u-v
GLfloat M[16] = { xt, xu, xv, 0,  yt, yu, yv, 0,  zt, zu, zv, 0,  0, 0, 0, 1};

//MT is the inverse of M (transpose of M in this case)
GLfloat  MT[16] = { xt, yt, zt, 0,  xu, yu, zu, 0,  xv, yv, zv, 0,  0, 0, 0, 1};
```

```
    //Note that the transformations are specified in reverse order
    glTranslatef( x0, y0, z0);         //5. Backward translation
    glMultMatrixf( MT);                //4. Change coord. system back to x-y-z
    glScalef( 1.0, -1.0, 1.0);         //3. Reflect relative to t-v plane
    glMultMatrixf( M);                 //2. Change coord. system to t-u-v
    glTranslatef( -x0, -y0, -z0);      //1. Translate P0 to origin

} // hkgluReflect()
```

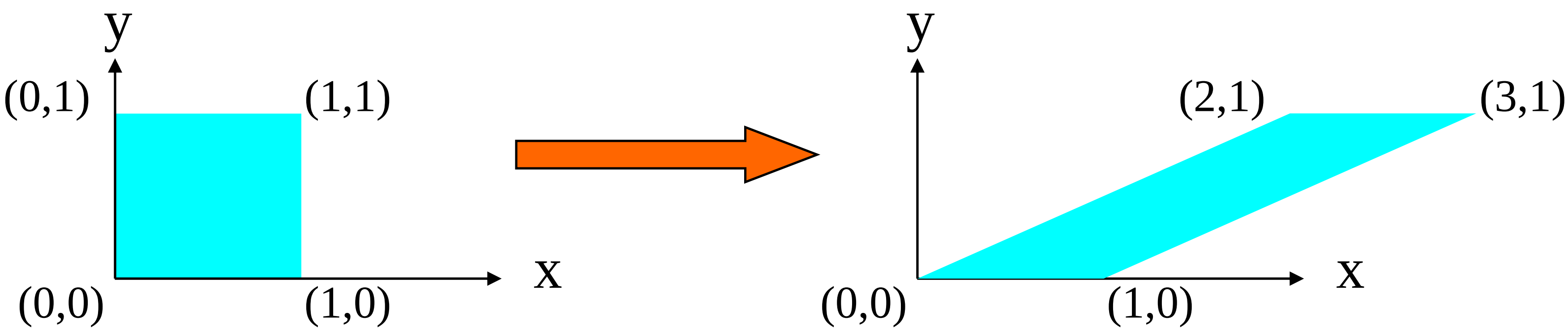
```
//Calling example: The drawing of the reflection of a rotating pyramid
glPushMatrix();
        hkgluReflect(x1, y1, z1,  x2, y2, z2,  x3, x3, z3);
        glRotatef( deg, xb-xa, yb-ya, zb-za);
        pyramid();
glPopMatrix();
```



Mirror

Shearing

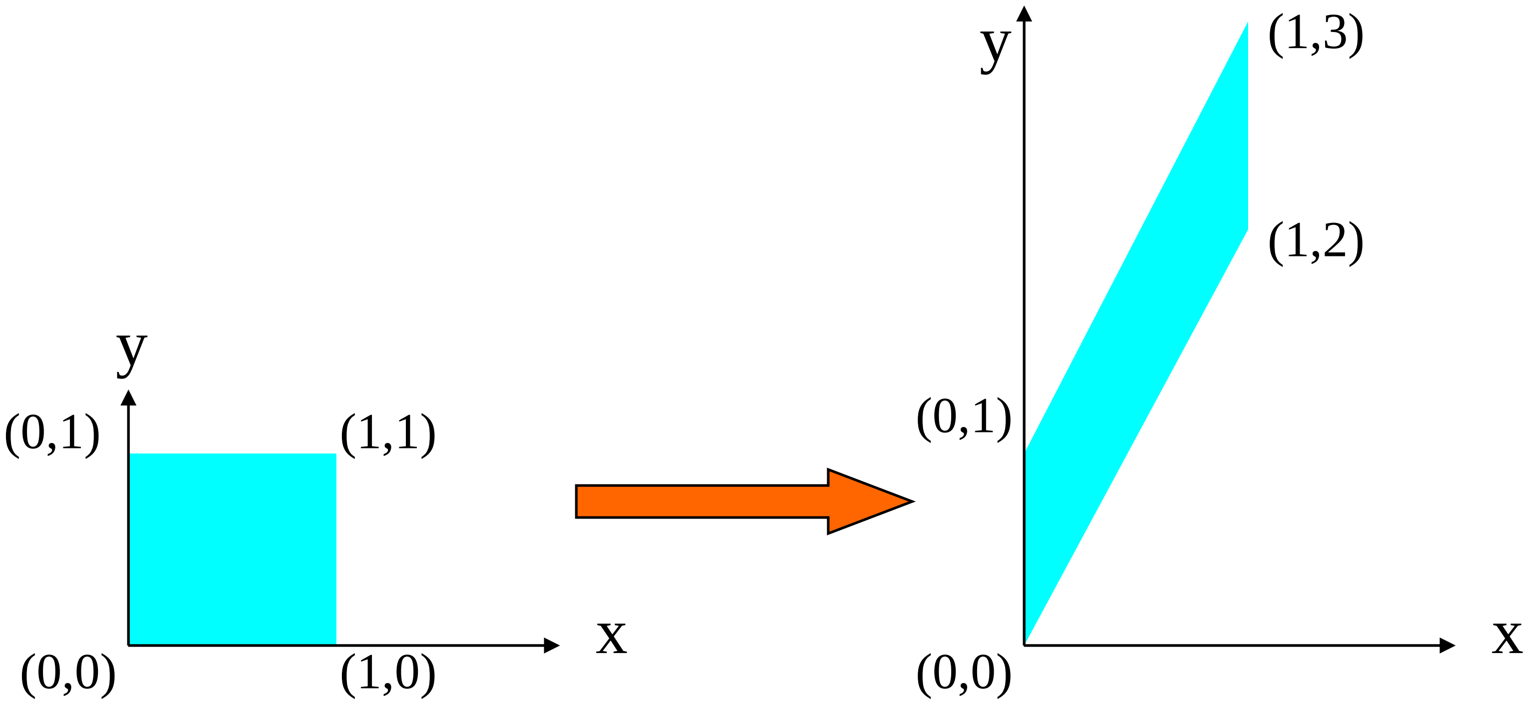
- Translation, rotation, reflection болон scaling объектийн сүүдрийг өөрчилдөггүй.
- scaling ба shearing объектийн хэлбэрийг өөрчилдөг.
- *x* чиглэлд 2D shearing (x-координатыг y утгатай пропорциональ хэмжээгээр өөрчлөх)



$$x' = x + 2y$$
$$y' = y$$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

➢ *y* чиглэлд 2D shearing



$$x' = x$$

$$y' = 2x + y$$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Shearing in 3D

- x чиглэлд $Sr \times y$ утгаар Shear.

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & Sr & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

x чиглэлд $Sr \times z$ утгаар Shear.

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & Sr & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

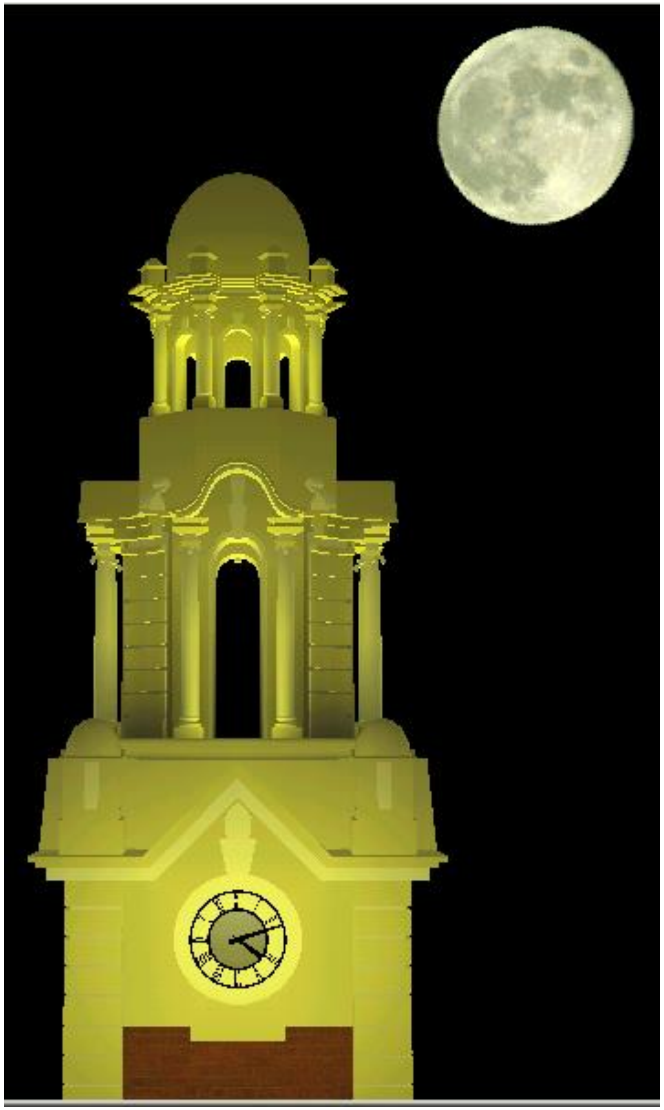
- x ба y чиглэлд $S_1 \times z$, ба $S_2 \times z$.утгаар Shear хийхы

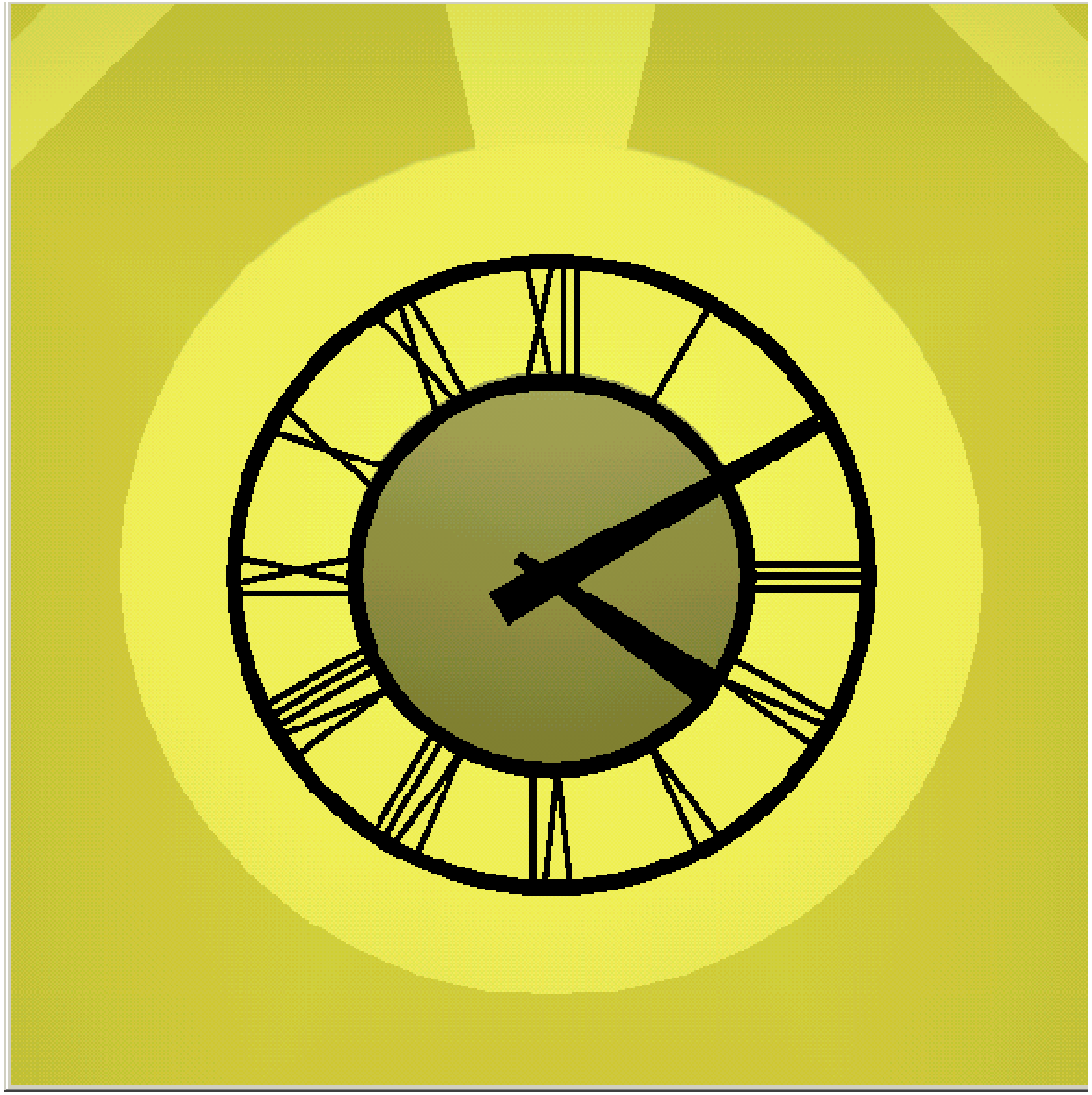
$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & S_1 & 0 \\ 0 & 1 & S_2 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

y ба z хоёуланд нь пропорциональ байхаар *x* хэрхэн shearing хийх вэ?

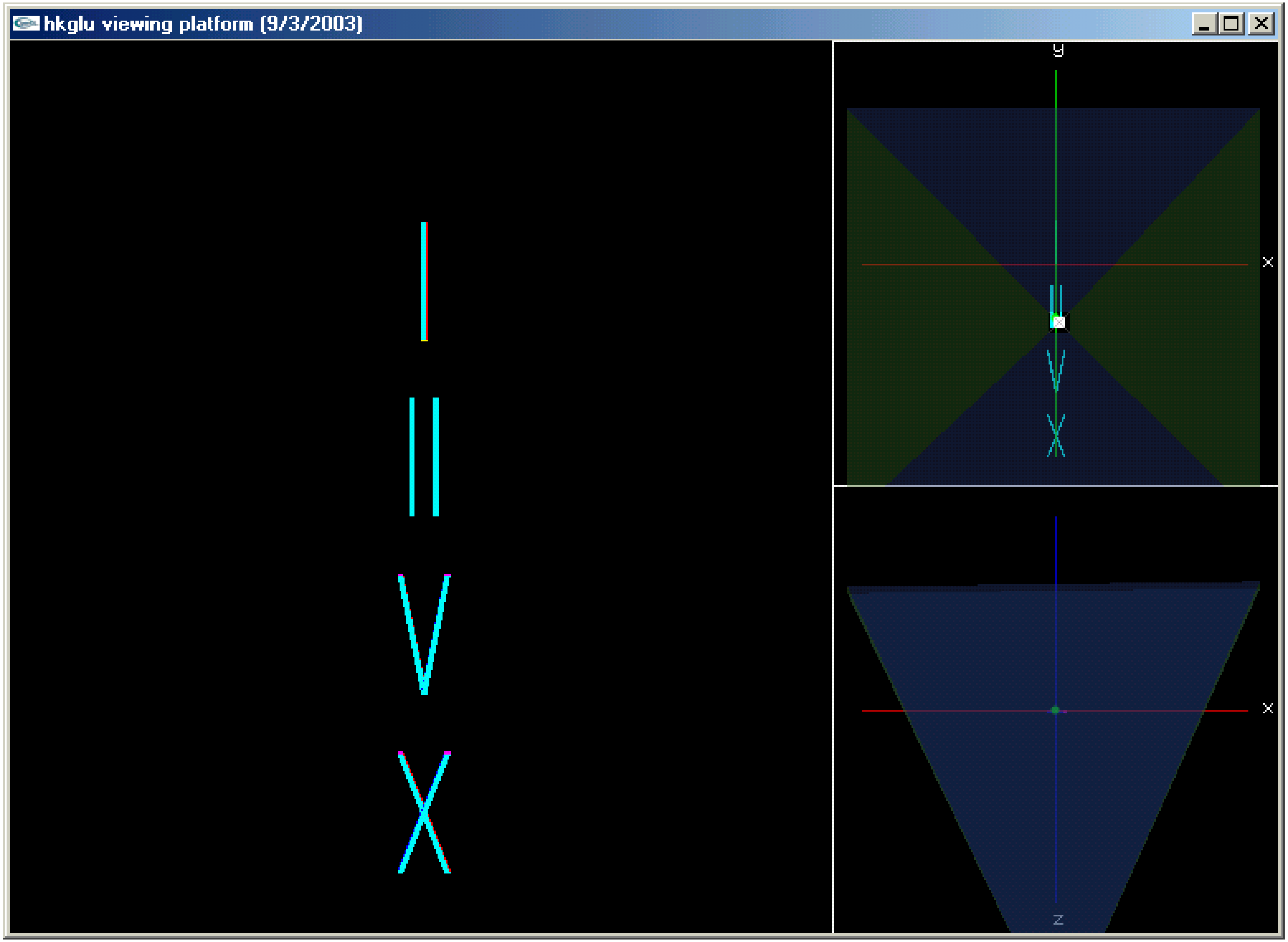
- Shearing жишээ

  LYH Tower-ийн цагийн ромб тоог зурах



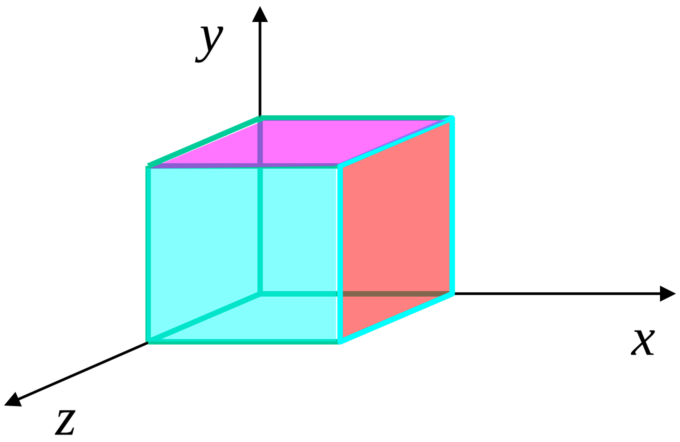


Бүх зураасуудыг нэг кубыг хувиргана.

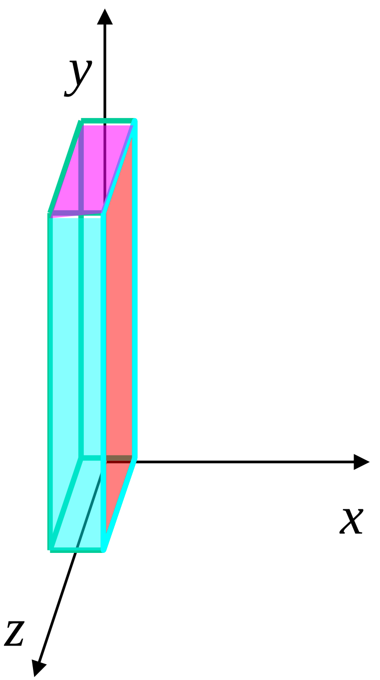


Shear

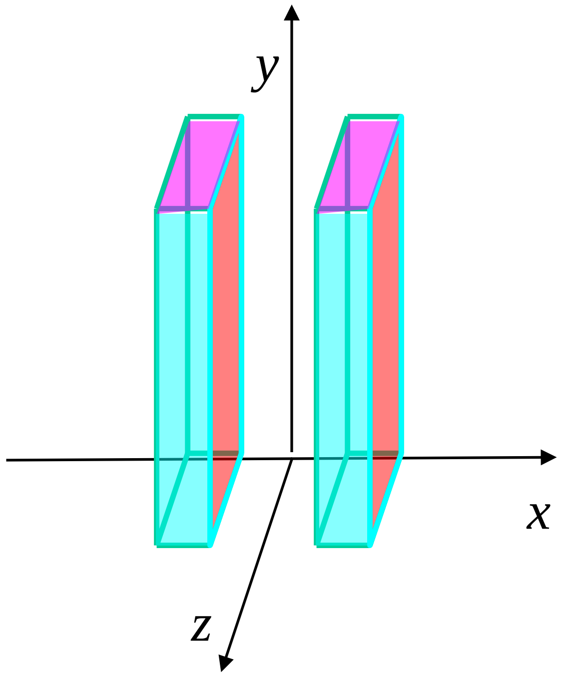
//cube() draws a unit cube at the origin




```
void one( double w, double h, double d) {
    glPushMatrix();
        glTranslatef(-.5*w, 0, 0);
        glScalef( w, h, d);
        cube();
    glPopMatrix();
}
```

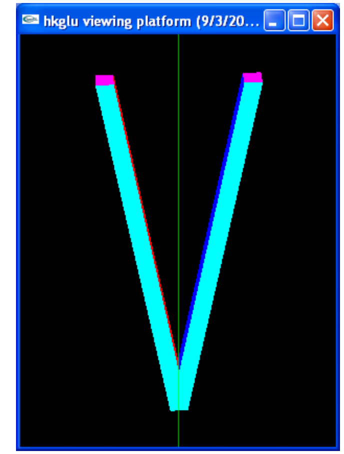


```
void two(double w, double h, double d) {
    glPushMatrix();
        glTranslatef( -2*w, 0., 0.);
        one( w, h, d);
        glTranslatef( 4*w, 0., 0.);
        one( w, h, d);
    glPopMatrix();
}   //two();
```



```
void five(double w, double h, double d) {
    //M нь зүүн тийш shear хийх матриц (in x  by -4w/h * y)
     GLfloat M[16] = {1, 0, 0, 0,  -4*w/h, 1, 0, 0,  0, 0, 1, 0,  0, 0, 0, 1};

     glPushMatrix();
         glMultMatrixf( M);
         one( w, h, d);
     glPopMatrix();
```

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & -4w/h & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

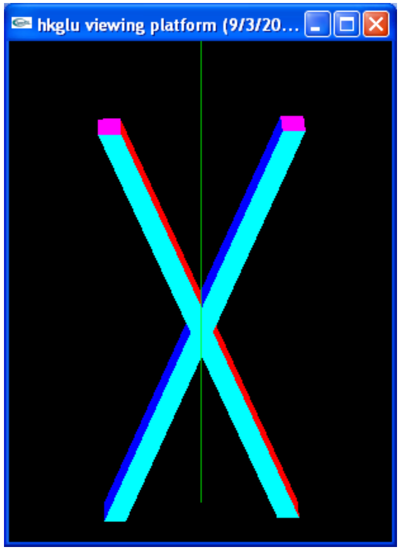



```
    // M нь зүүн тийш shear хийх матриц (in x  by 4w/h * y)
     GLfloat M2[16] = {1, 0, 0, 0,  4*w/h, 1, 0, 0,  0, 0, 1, 0,  0, 0, 0, 1};

     glPushMatrix();
         glMultMatrixf( M2);
         one( w, h, d);
     glPopMatrix();
```

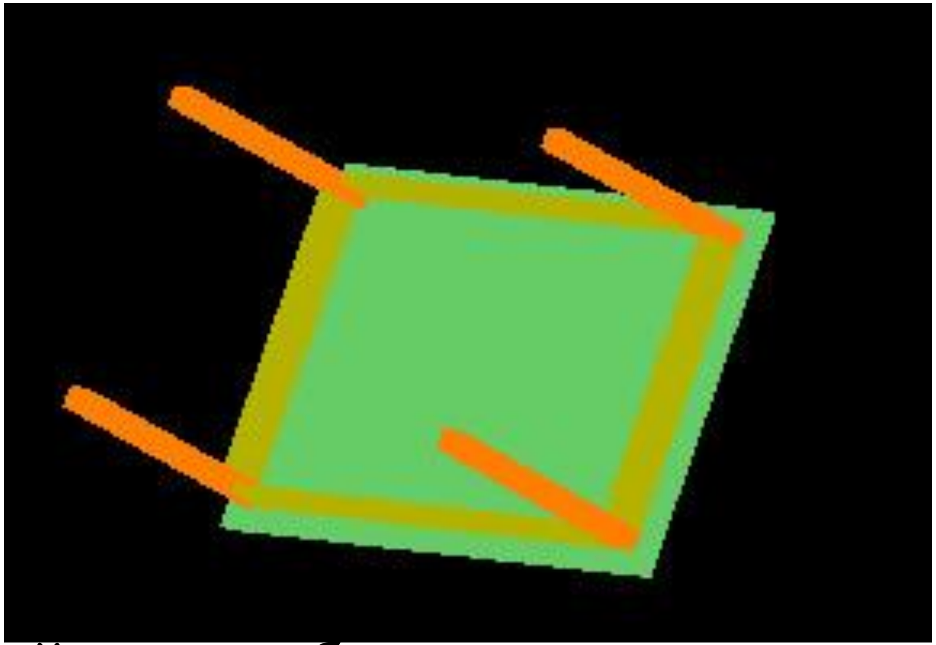
$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 4w/h & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

```
}    //five();
```

```
void ten(double w, double h, double d) {
    //M нь -8w/h аар зүүн тийш shear хийх матриц
    GLfloat M[16] = {1, 0, 0, 0,  -8*w/h, 1, 0, 0,  0, 0, 1, 0,  0, 0, 0, 1};

    glPushMatrix();
        glTranslatef( 4*w, 0., 0.);
        glMultMatrixf( M);
        one( w, h, d);
    glPopMatrix();

    //M нь 8w/h аар баруун тийш shear хийх матриц
    GLfloat M2[16] = {1, 0, 0, 0,  8*w/h, 1, 0, 0,  0, 0, 1, 0,  0, 0, 0, 1};

    glPushMatrix();
        glTranslatef( -4*w, 0., 0.);
        glMultMatrixf( M2);
        one( w, h, d);
    glPopMatrix();

}    //ten();
```



Шаталсан загвар (Hierarchical Modeling)



- Төвөгтэй объектийг

Ихэвчлэн примитив

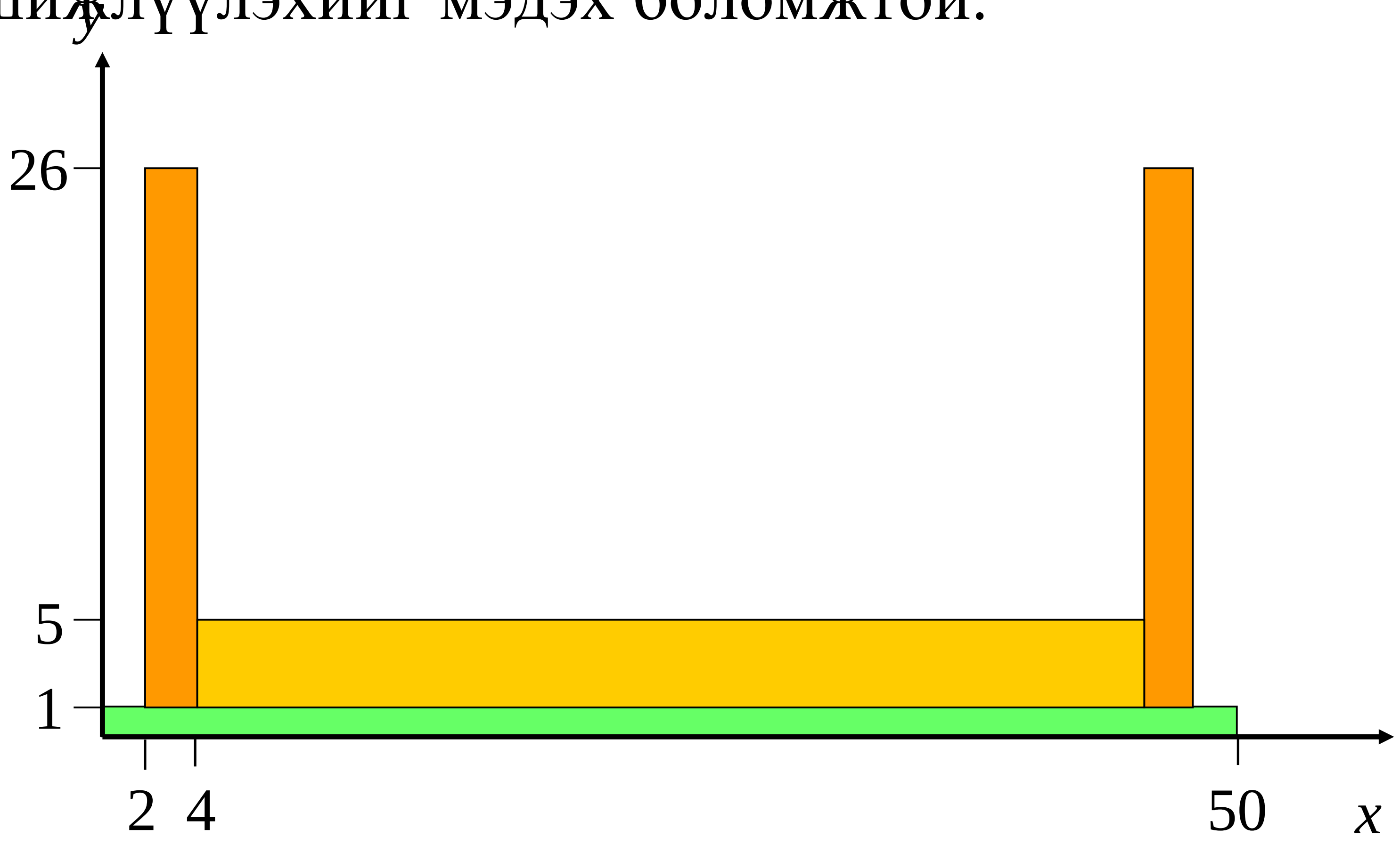
Объектоос байгуулдаг.

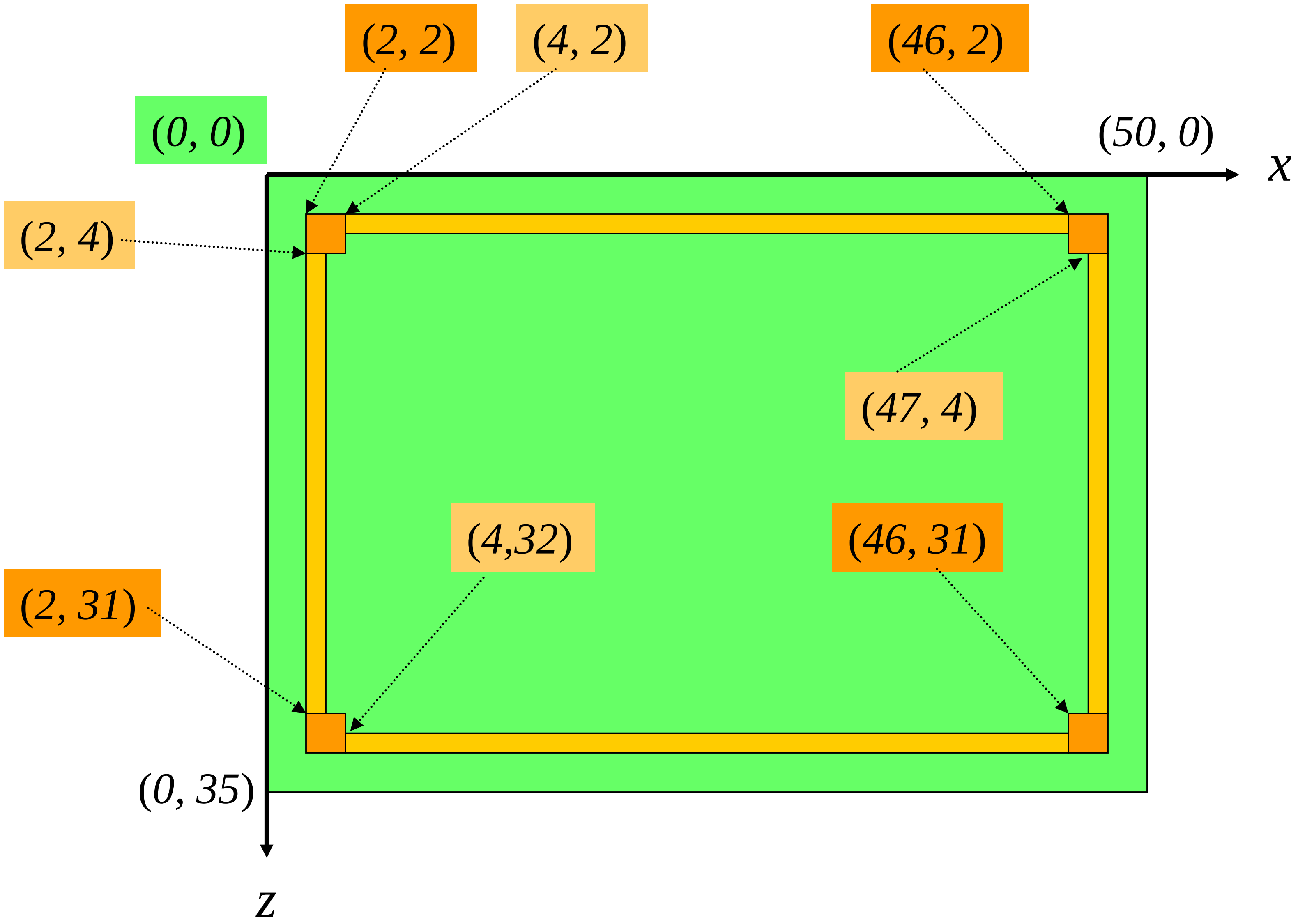
- Ширээ дараах 4 төрлийн байгуулагч блокуудыг агуулна
  - 1 тавцан (top)
  - 2 урт хөндлөвч (long flanks)
  - 2 богино хөндлөвч (short flanks)
  - 4 хөл (legs)

table

- Байгуулагч блокуудыг бүгд нэгж кубээс масштаблана.

- Нэгдүгээрт блок бүрийн хэмжээсийг олж мэдэх хэрэгтэй
- Дараа нь блок тус бүрийн байршилыг нарийн тогтооно. Ингэснээр бид блокыг зөв газарт хэрхэн шижлүүлэхийг мэдэх боломжтой.





(2, 2)
(4, 2)
(46, 2)
(0, 0)
(50, 0)
x
(2, 4)
(47, 4)
(4,32)
(46, 31)
(2, 31)
(0, 35)
z

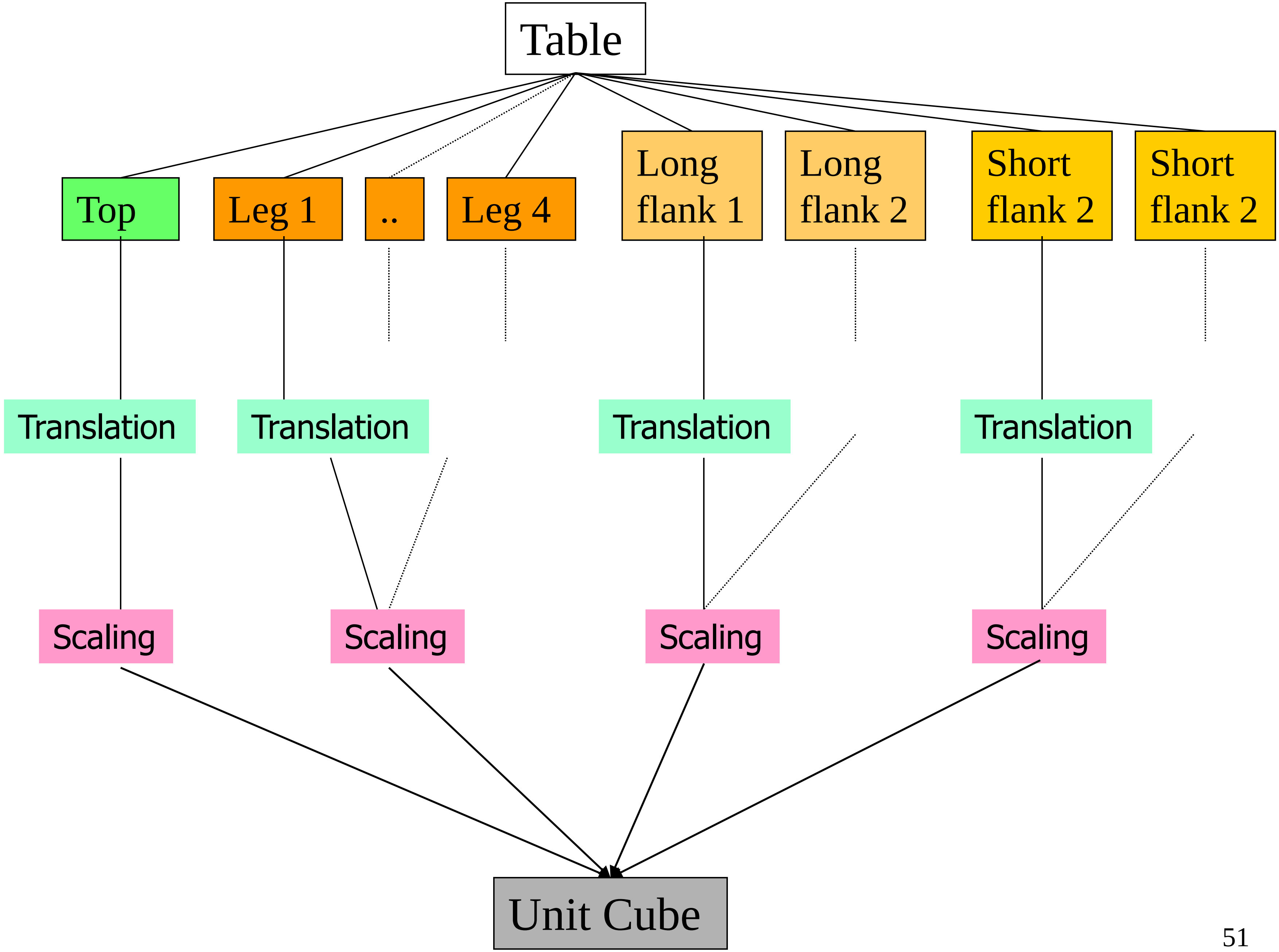


Table
Top
Leg 1
..
Leg 4
Long flank 1
Long flank 2
Short flank 2
Short flank 2
Translation
Translation
Translation
Translation
Scaling
Scaling
Scaling
Scaling
Unit Cube

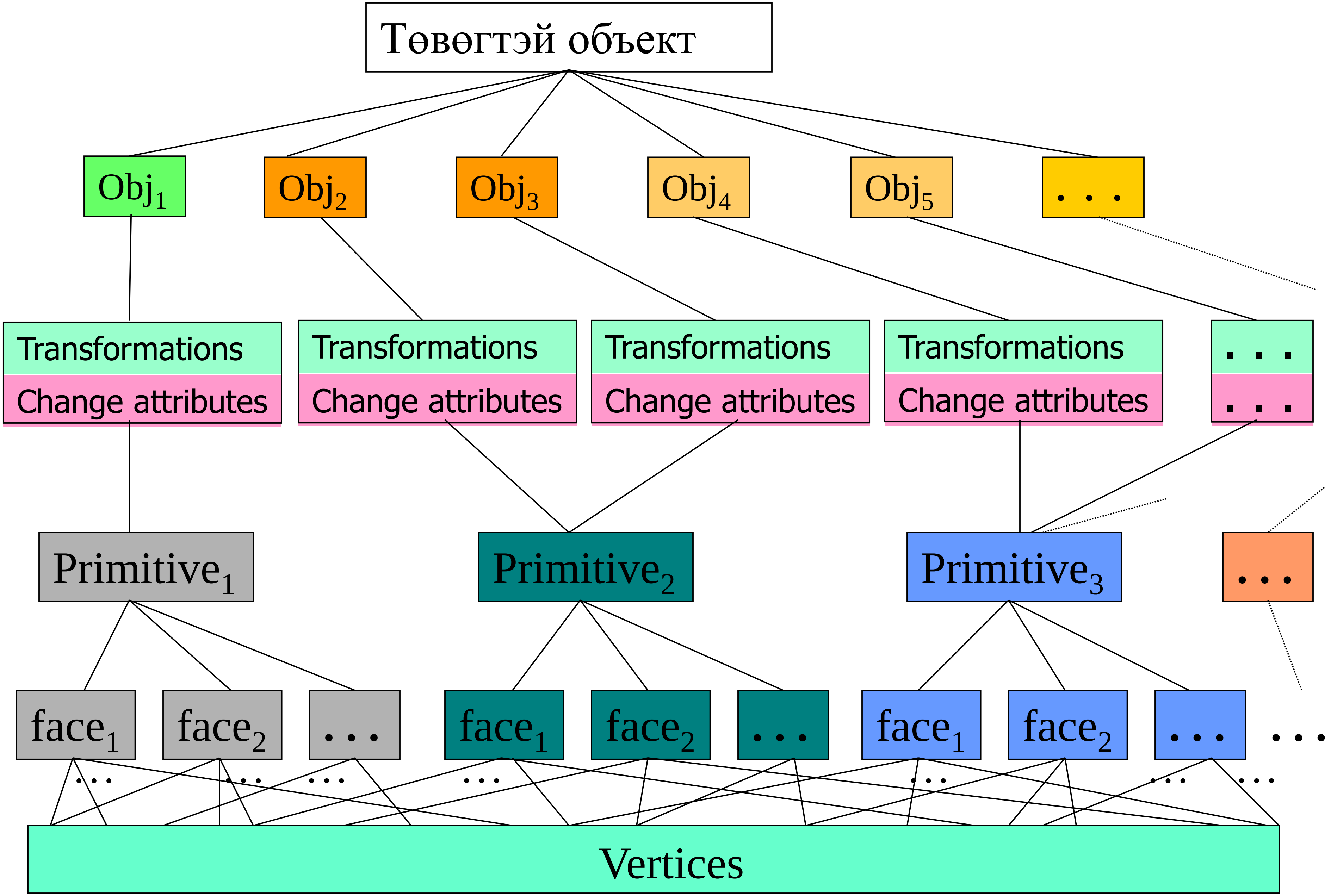
```
void  table()  {
    //Ширээний тавцан
    glColor3f( 0.4, .8, 0.4);
    glPushMatrix();
        glScalef(50., 1., 35.);
        cube();
    glPopMatrix();

    //Эхний хөл
    glColor3f( 1.0, .5, 0.);
    glPushMatrix();
        glTranslatef( 2., 1., 2.);
        glScalef(2., 25., 2.);
        cube();
    glPopMatrix();

    //Хоёр дахь хөл
    glPushMatrix();
        glTranslatef( 2., 1., 31.);
        glScalef(2., 25., 2.);
        cube();
    glPopMatrix();

    //Гурав дахь хөл
    glPushMatrix();
        glTranslatef( 46., 1., 2.);
        glScalef(2., 25., 2.);
        cube();
    glPopMatrix();
```

```
//Дөрөв дэх хөл
glPushMatrix();
    glTranslatef( 46., 1., 31.);
    glScalef(2., 25., 2.);
    cube();
glPopMatrix();

//Эхний урт хэсгийн хөндлөвч
glColor3f( .7, .7, 0.);
glPushMatrix();
    glTranslatef( 4., 1., 2.);
    glScalef(42., 4., 1.);
    cube();
glPopMatrix();
```

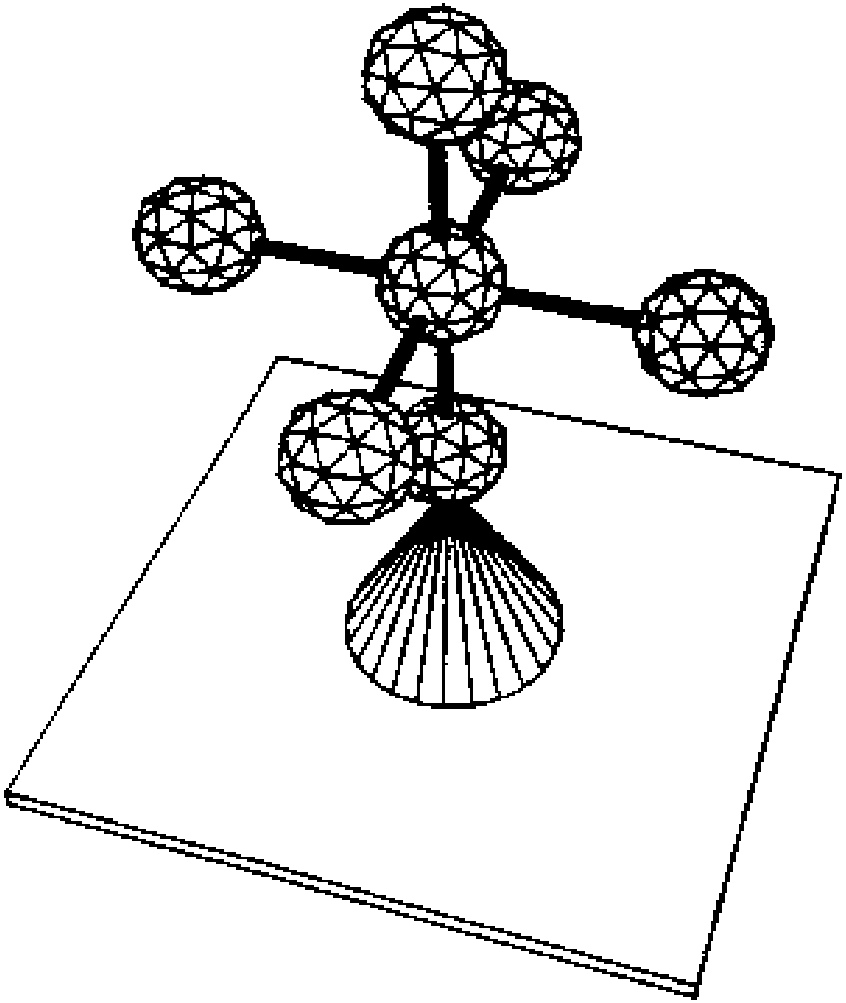
```
//хоёр дахь урт хэсгийн хөндлөвч
glPushMatrix();
    glTranslatef( 4., 1., 32.);
    glScalef(42., 4., 1.);
    cube();
glPopMatrix();

//эхний богино хэсгийн хөндлөвч
glPushMatrix();
    glTranslatef( 2., 1., 4.);
    glScalef(1., 4., 27.);
    cube();
glPopMatrix();
```

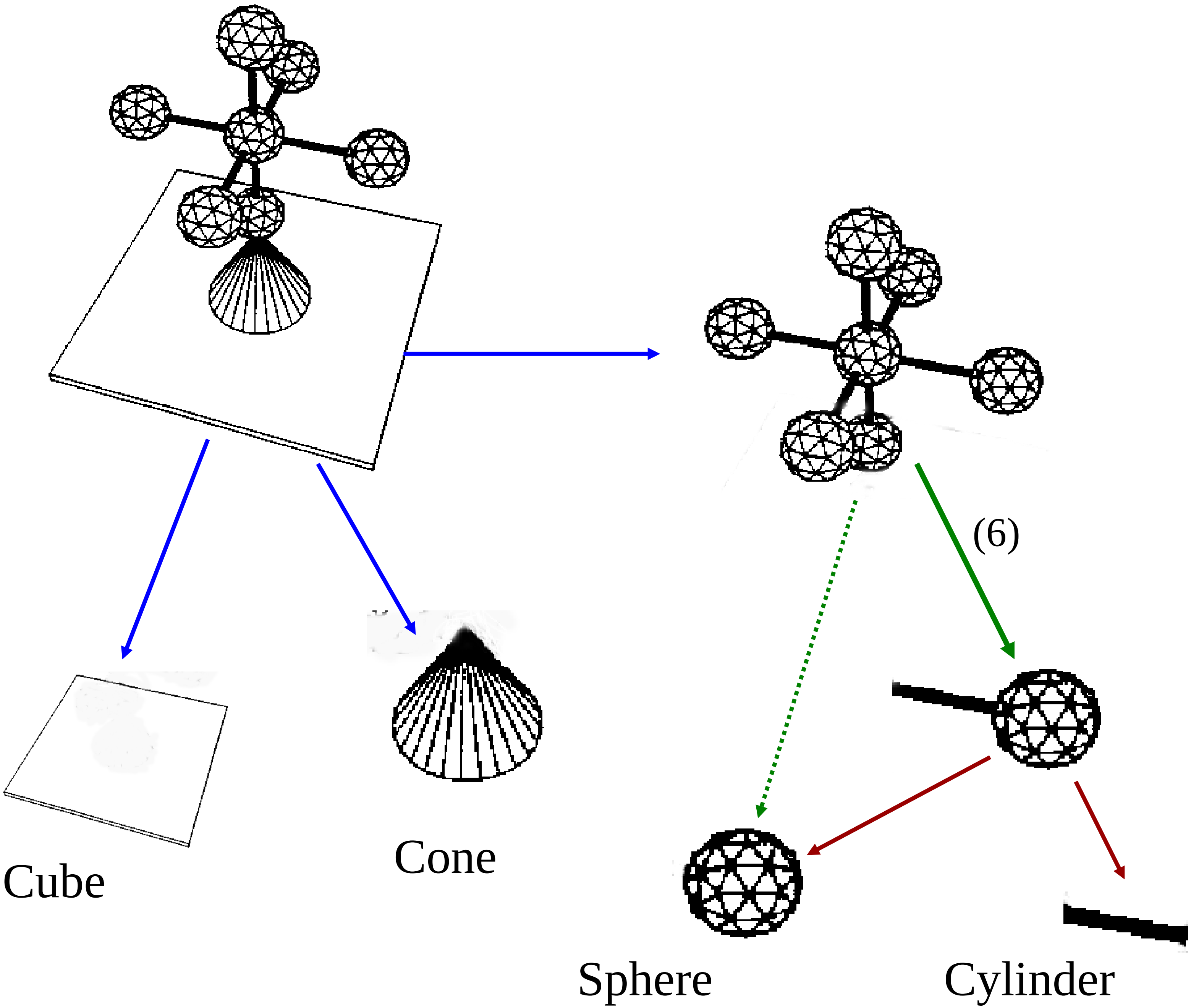
```
//хоёр дахь богино хэсгийн
    хөндлөвч
glPushMatrix();
        glTranslatef( 47., 1., 4.);
        glScalef(1., 4., 27.);
        cube();
    glPopMatrix();

}  //table()
```

Ердийн шаталсан загвар

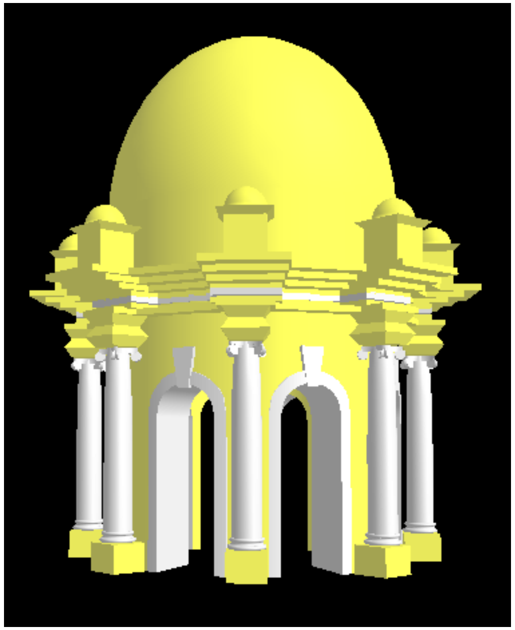


- Энэ зургийг байгуулахад шаардлагатай примитив объект ба шилжилтийг төсөөлнө үү.



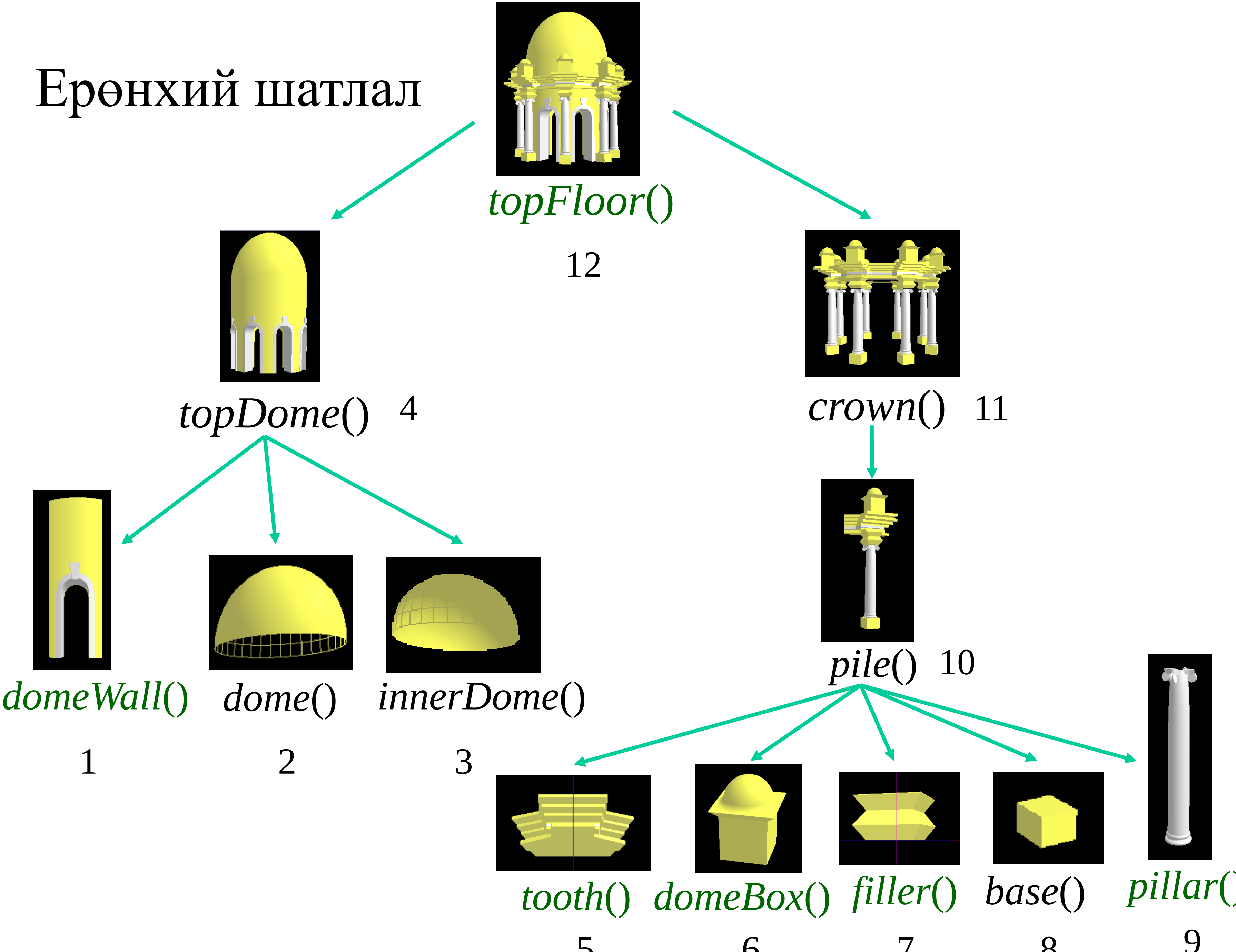

(6)
Cube
Cone
Sphere
Cylinder

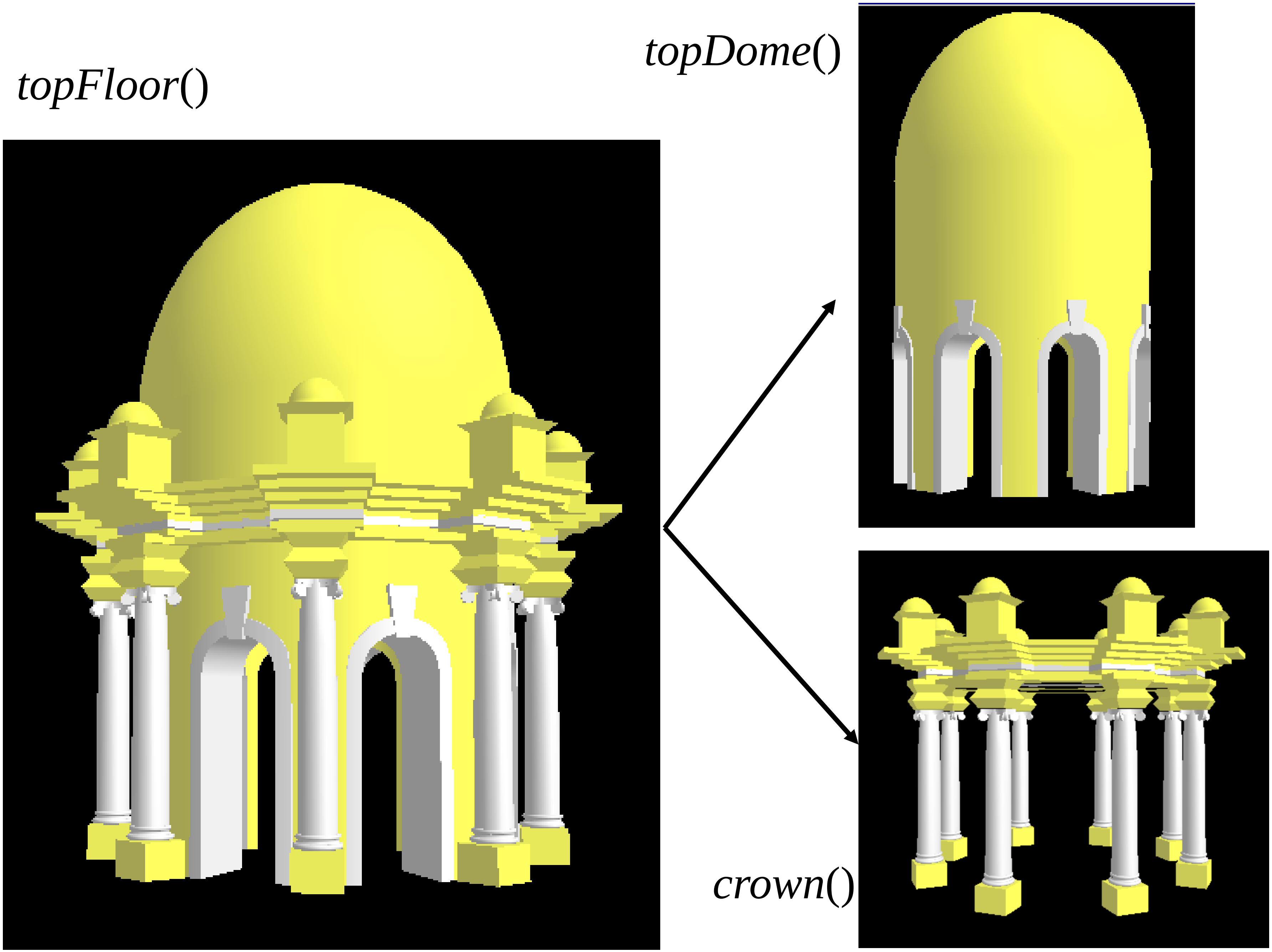
LYH clock tower



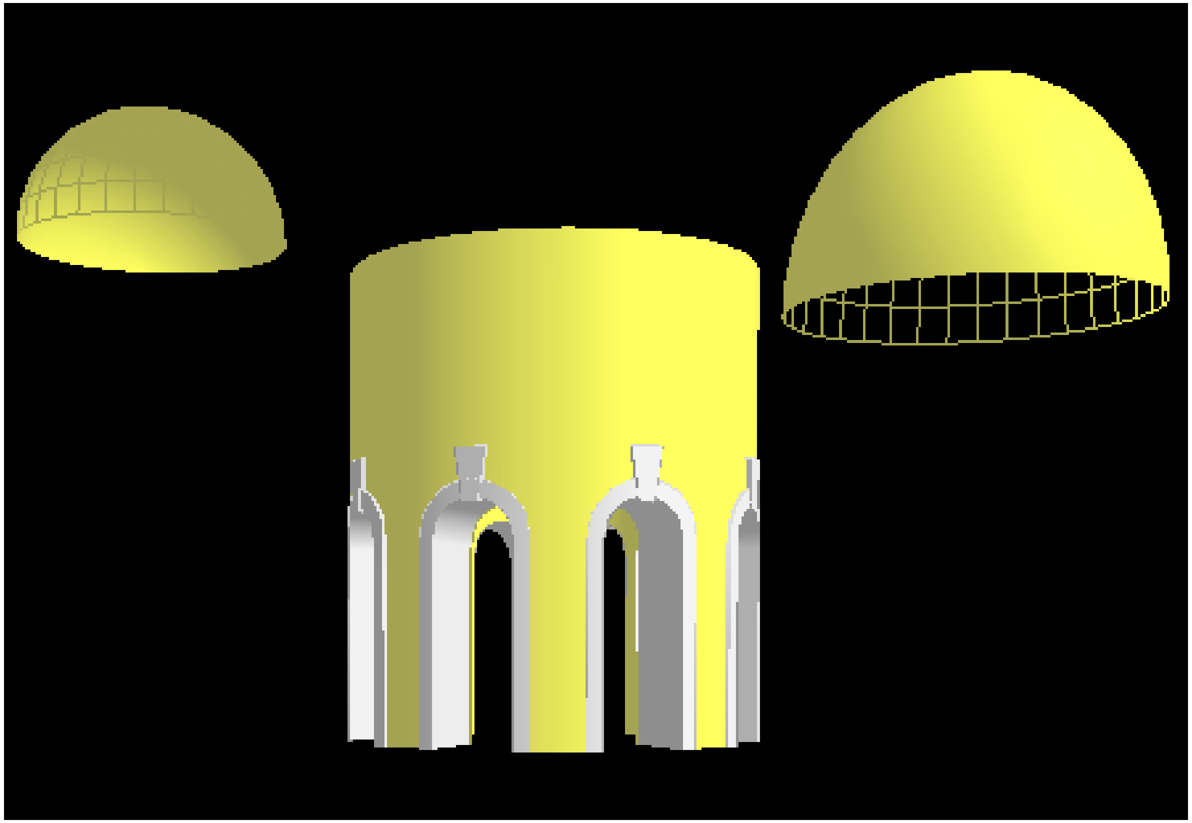
Top Floor

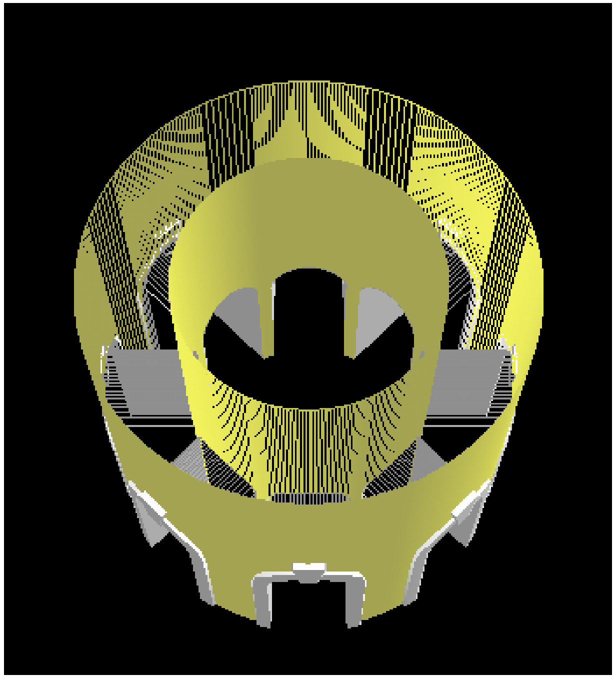
Ерөнхий шатлал



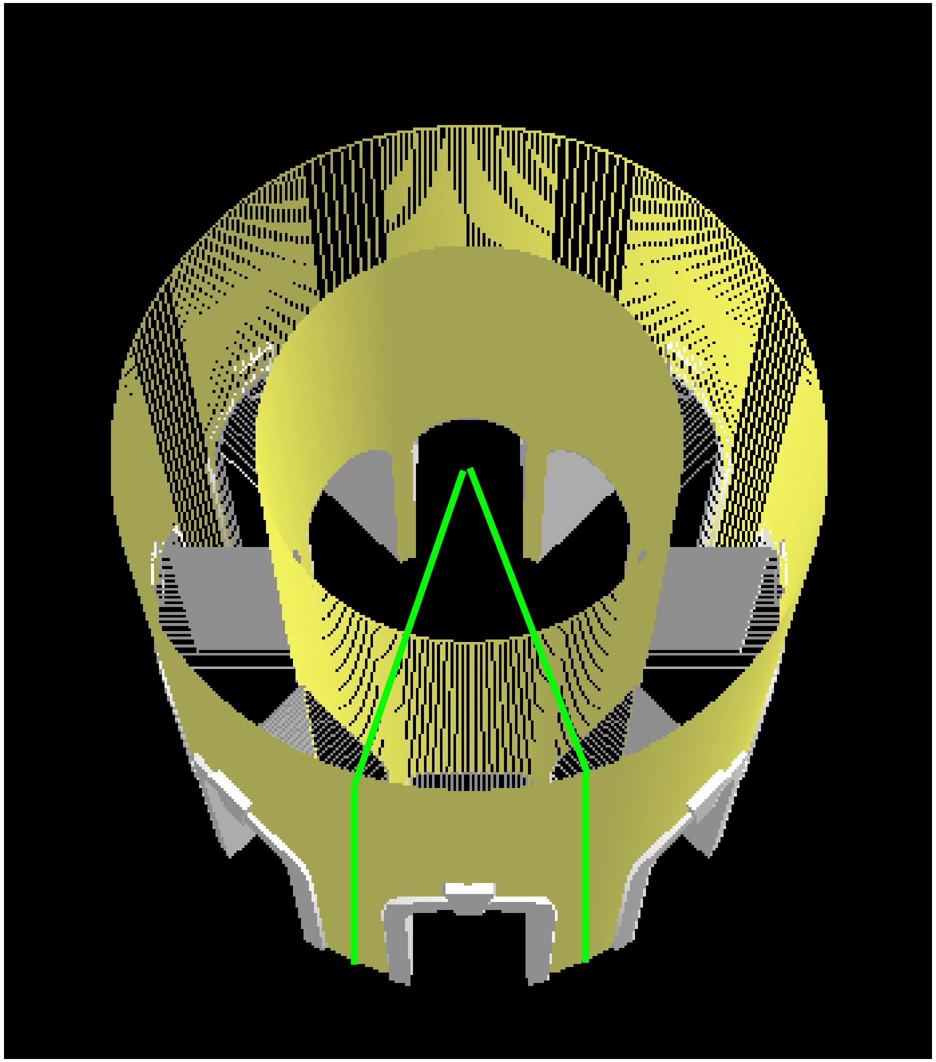

topFloor()
topDome()
crown()

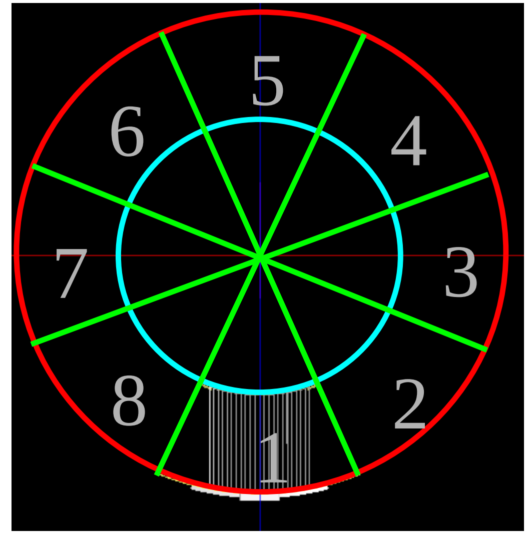
topDome () нь гаднах бөмбөгөр, дотор талын бөмбөгөр, нуман хаалга бүхий дугуй хэлбэртэй хананаас бүрдэнэ.

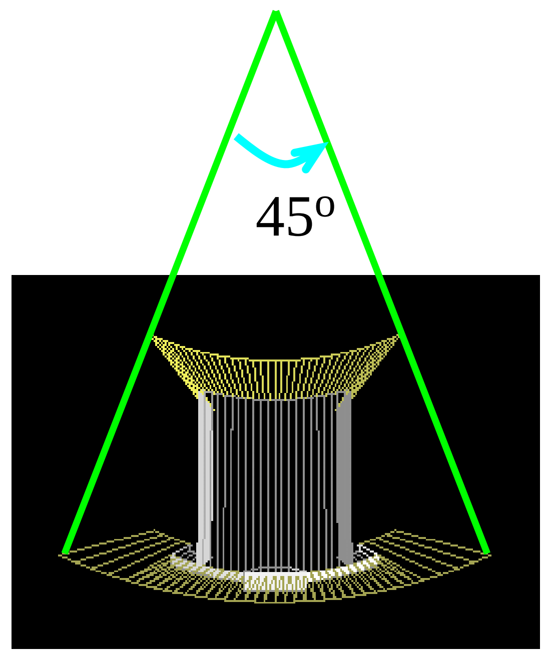




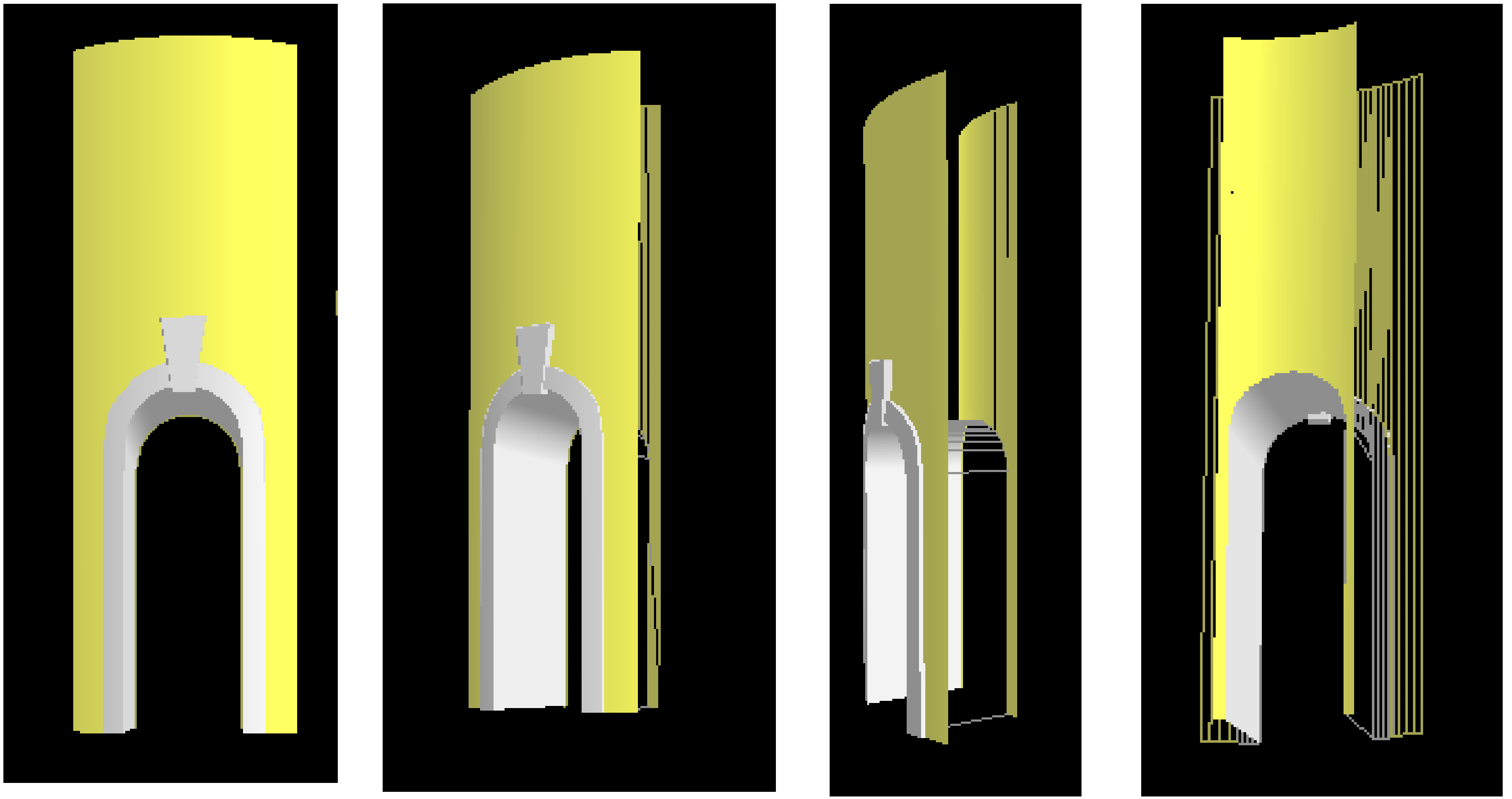
Дугуй хана нь domeWall () -ийн найман ижил хуулбараас бүрдэнэ.



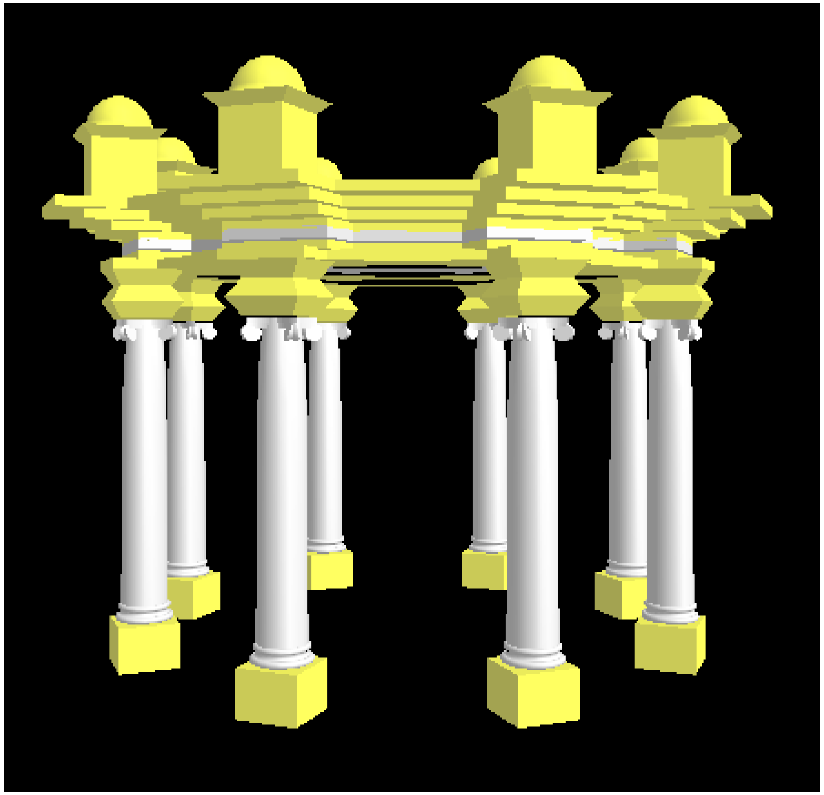


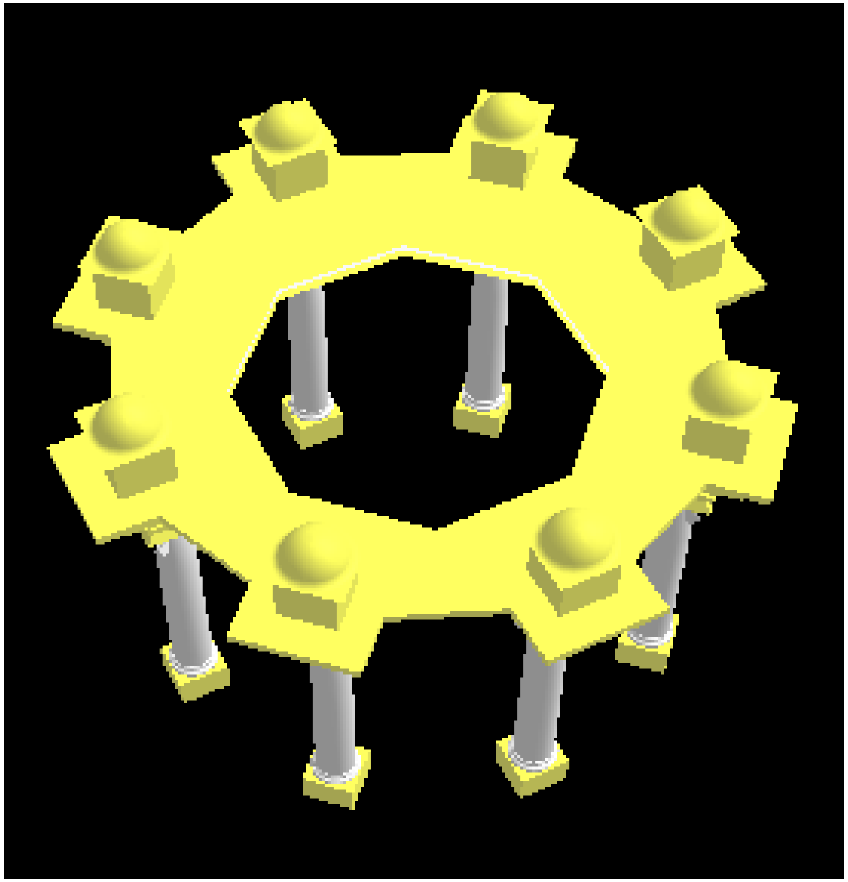


domeWall(r1, r2, h1, 45.);

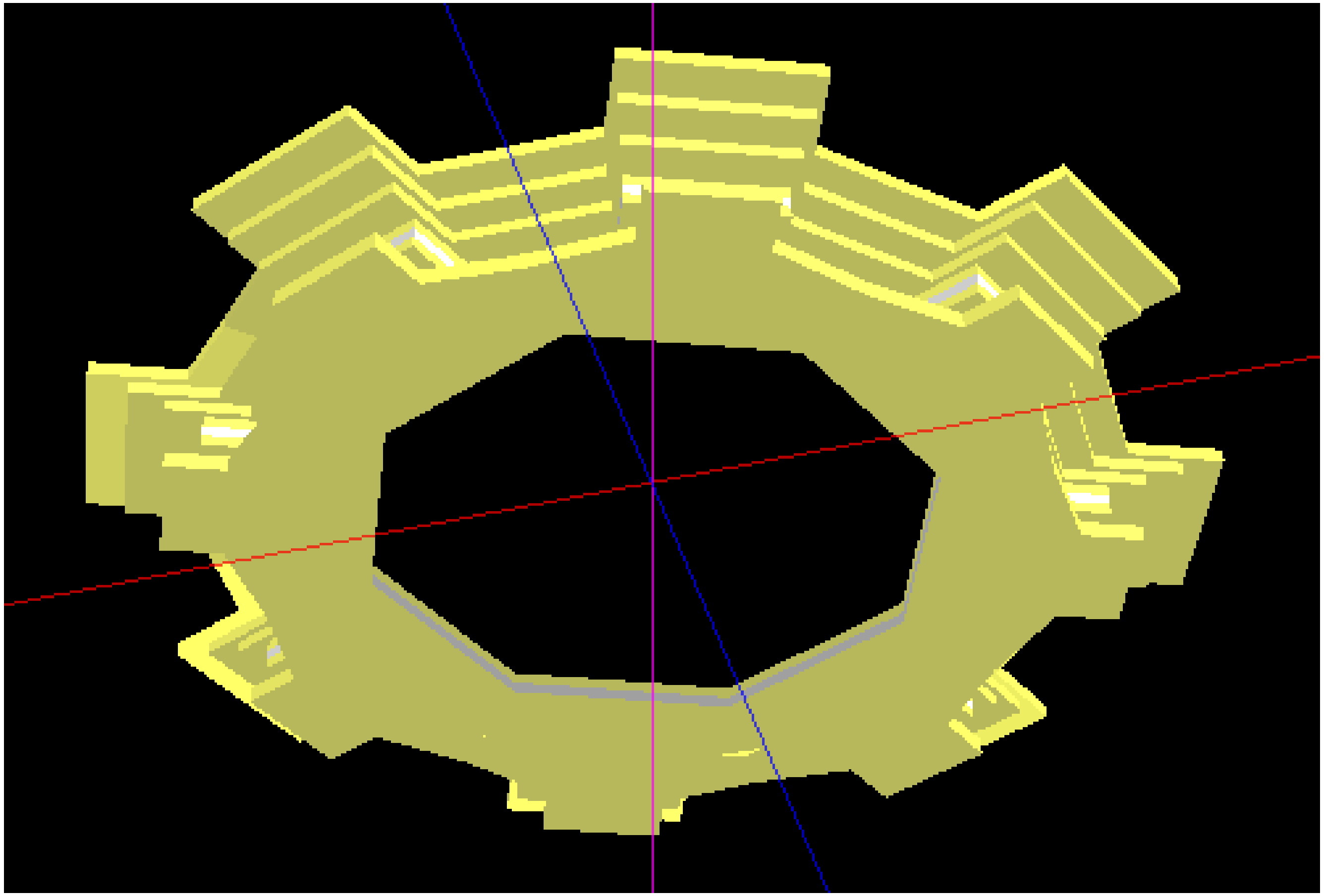


TopFloor () -ын өөр нэг бүрэлдэхүүн хэсэг нь crown() юм.

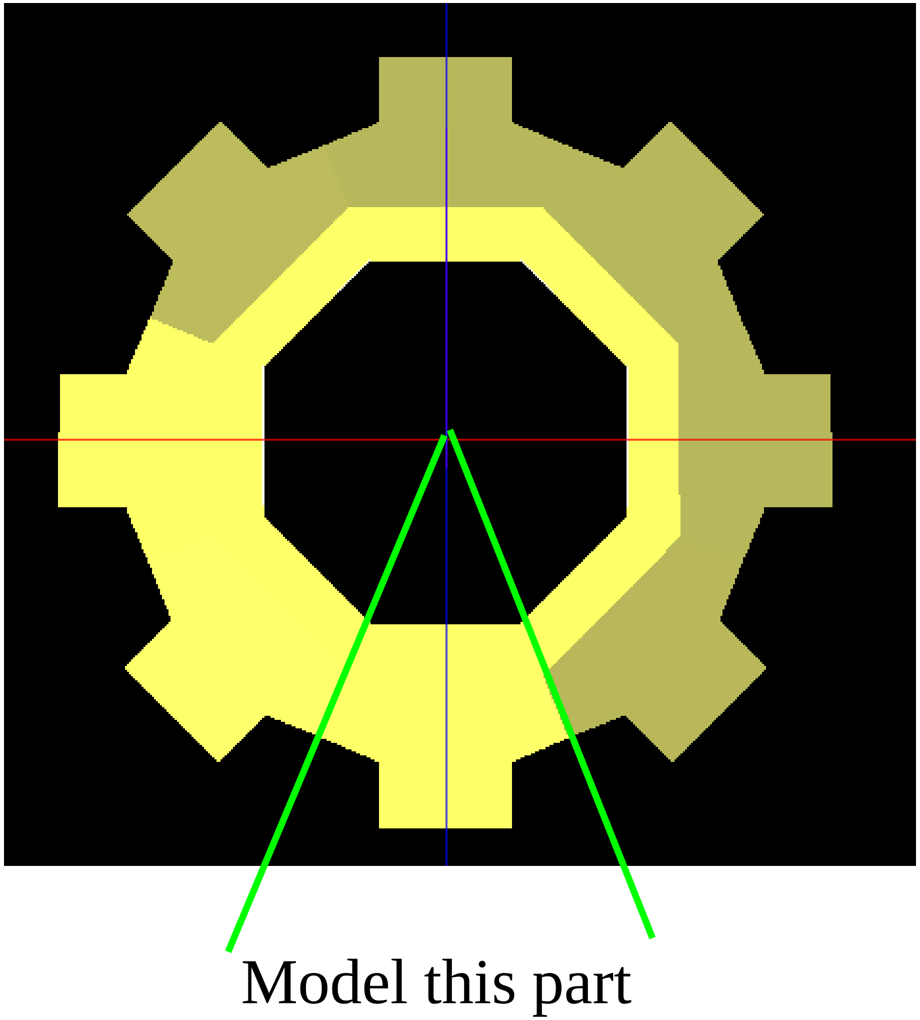


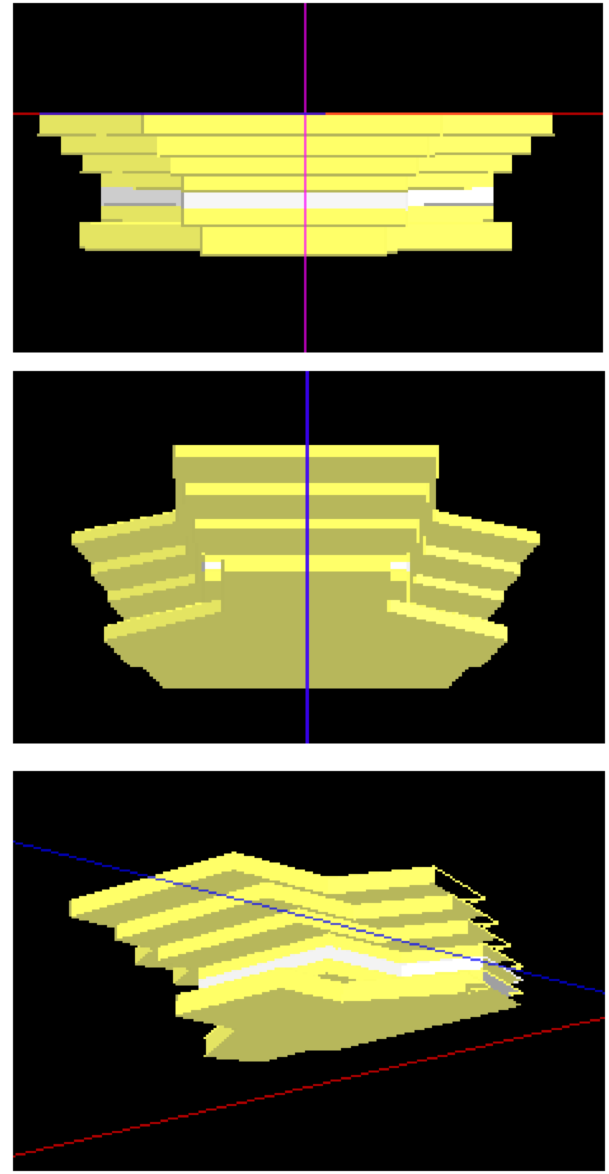


crown() –ийн тогтоох хэсэг нь gear юм

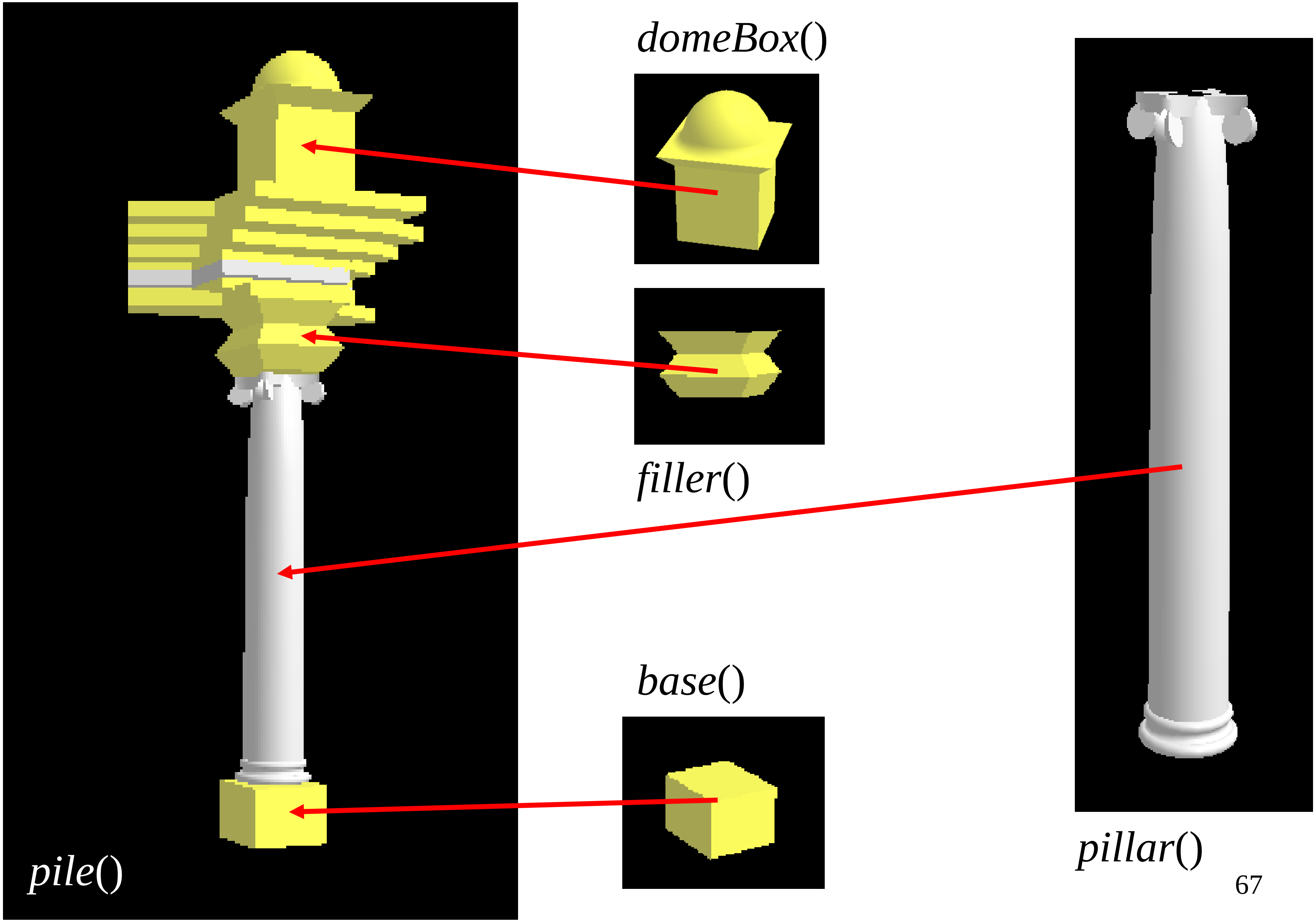


- *tooth*() нь gear-ийн давтагдах хэсэг. Энэ нь жигд бус хэлбэртэй ба stack of boards бүрдэнэ.

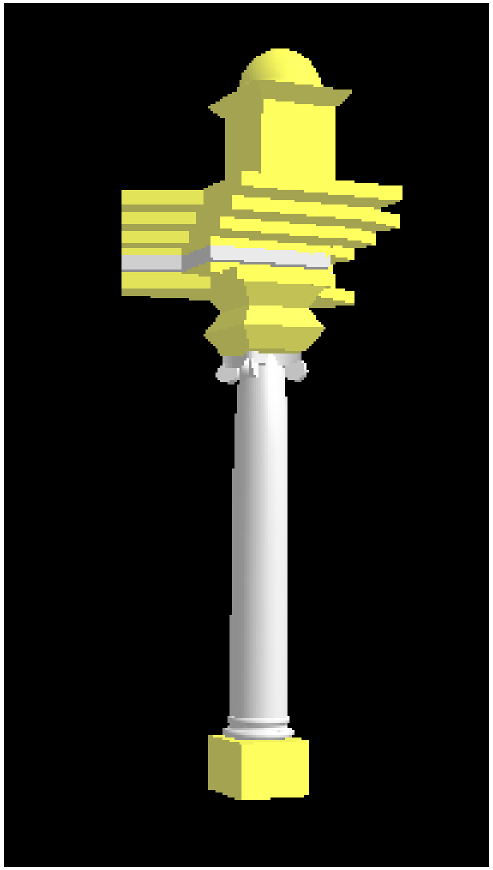




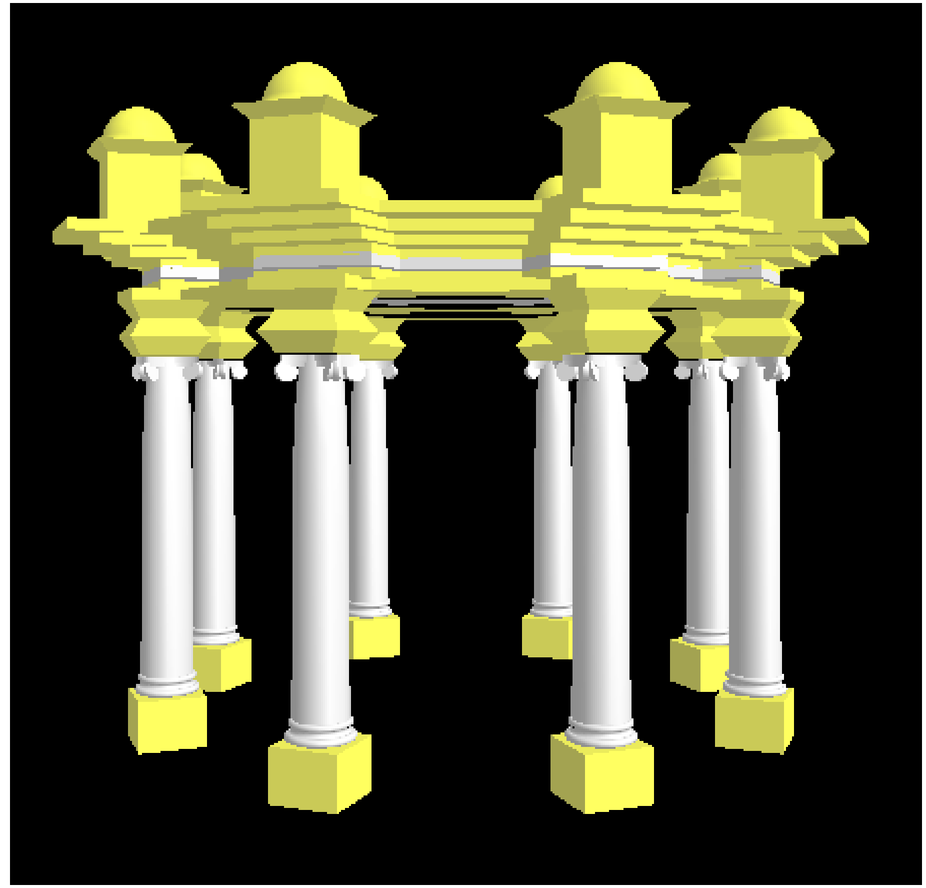
The other objects of a pile reside above or below the tooth



crown() consists of eight copies of the pile



pile()



Overall hierarchy