6. TRANSFORMĂRI

6.1. Translaţie

Funcţia

```
void glTranslate<t>(<type> tx, <type> ty, <type> tz);
```

unde

```
<t> ::= d | f
<type> ::= GLdouble | GLfloat
```

Produce translaţia [`tx, ty, tz`] prin înmulţirea matricei curente *M* cu matricea de translaţie *T*.

$$T = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matricea curentă *M* va fi înlocuită cu $M \cdot T$.

Dacă modul curent este `GL_MODELVIEW` sau `GL_PROJECTION`, toate obiectele desenate după apelul funcţiei vor fi desenate mutate. Trebuie de folosit `glPushMatrix` şi `glPopMatrix` pentru a memoriza şi a restabili sistemul de coordonate netranslat.

Exemplu:

```
glTranslatef(0.0f, 0.0f, -10.0f);
```

Acelaşi lucru poate fi făcut şi prin funcţia `glMultmatrixd()` astfel:

```
static GLdouble t[16] =
{
   1.,   0.,   0.,   0.,
   0.,   1.,   0.,   0.,
   0.,   0.,   1.,   0.,
   0.,   0., -10.,   1.
};
glMultMatrixd(t);
```

Funcţia

```
void glMultMatrix<t>(const <type> *m);
```

unde

```
<t> ::= d | f
<type> ::= GLdouble | GLfloat
```

înmulţeşte matricea curentă cu una specificată in vectorul `m`. Dacă matricea curentă este *M*, şi cea pasată funcţiei `glMultMatrixd(t)` este *T*, atunci *M* va fi înlocuită cu $M \cdot T$. Matricea curentă poate fi matricea de proiectare, sau matricea de modelare, sau matricea de texturare, ceia ce se stabileşte cu funcţia `glMatrixMode()`.

Parametrul `m` este un pointer la matricea 4x4 de precizie ordinară sau dublă, cu elementele memorizate pe coloane astfel:

$$\begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix}$$

6.2. Rotire

Funcţia

```
void glRotate<t>(<type> angle, <type> x, <type> y, <type> z);
```

unde

```
<t> ::= d | f
<type> ::= GLdouble | GLfloat
```

Produce rotirea spaţiului în jurul vectorului `[x, y, z]` la unghiul `angle` în grade, prin înmulţirea matricei curente *M* cu matricea de rotaţie *R*. Matricea curentă *M* va fi înlocuită cu $M \cdot R$.

Dacă modul curent este `GL_MODELVIEW` sau `GL_PROJECTION`, toate obiectele desenate după apelul funcţiei vor fi desenate rotite. Trebuie de folosit `glPushMatrix` şi `glPopMatrix` pentru a memoriza şi a restabili sistemul de coordonate nerotit.

Câteva exemple de matrice *R*.

Rotirea în jurul axei *OX*.

```
glRotatef(fi, 1.0f, 0.0f, 0.0f);
```

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(fi) & -\sin(fi) & 0 \\ 0 & \sin(fi) & \cos(fi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De exemplu pentru a roti scena în jurul axei *OX* cu 30º apelăm

```
glRotatef(30.f, 1.0f, 0.0f, 0.0f);
```

exact acelaşi lucru poate fi făcut prin înmulţirea matricelor astfel:

```
static GLdouble rx30[16] =
{
   1.,   0.,    0.,   0.,
   0.,   0.866, 0.5,  0.,
   0.,   -0.5,  0.866,0.,
   0.,   0.,    0.,   1.
};
glMultMatrixd(rx30);
```

Valorile pentru *sin* şi *cos* au fost luate din următorul tabel auxiliar:

Unghiul *fi* în grade	*sin(fi)*		*cos(fi)*	
	valoarea exactă	valoarea aproximativă	valoarea exactă	valoarea aproximativă
0º	0	0.0	1	1.0
30º	½	0.5	√3/2	0.866
45º	√2/2	0.707	√2/2	0.707
60º	√3/2	0.866	½	0.5
90º	1	1.0	0	0.0

Rotirea în jurul axei *OY*.

```
glRotatef(fi, 0.0f, 1.0f, 0.0f);
```

$$R_y = \begin{bmatrix} \cos(fi) & 0 & \sin(fi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(fi) & 0 & \cos(fi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De exemplu pentru a roti scena în jurul axei *OY* cu 60º apelăm

```
glRotatef(60.f, 0.0f, 1.0f, 0.0f);
```

exact acelaşi lucru poate fi făcut prin înmulţirea matricelor astfel:

```
static GLdouble ry60[16] =
{
   0.5,   0.,  0.866, 0.,
   0.,    1.,  0.,    0.,
```

```
  -0.866, 0.,  0.5,   0.,
   0.,    0.,  0.,    1.
};
glMultMatrixd(ry60);
```

Rotirea în jurul axei *OZ*.

```
glRotatef(fi, 0.0f, 0.0f, 1.0f);
```

$$R_z = \begin{bmatrix} \cos(fi) & -\sin(fi) & 0 & 0 \\ \sin(fi) & \cos(fi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De exemplu pentru a roti scena în jurul axei *OZ* cu 45º apelăm

```
glRotatef(45.f, 0.0f, 0.0f, 1.0f);
```

exact acelaşi lucru poate fi făcut prin înmulţirea matricelor astfel:

```
static GLdouble rz45[16] =
{
   0.707, 0.707, 0., 0.,
  -0.707, 0.707, 0., 0.,
   0.,    0.,    1., 0.,
   0.,    0.,    0., 1.
};
glMultMatrixd(rz45);
```

6.3. Scalare

Funcţia

```
void glScale<t>(<type> sx, <type> sy, <type> sz);
```

unde

```
<t> ::= d | f
<type> ::= GLdouble | GLfloat
```

Produce scalare (mărire sau micşorare) spaţiului cu factorii `sx,sy,sz` pentru axele respective prin înmulţirea matricei curente *M* cu matricea de scalare *S*. Matricea curentă *M* va fi înlocuită cu $M \cdot S$.

$$S = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dacă modul curent este `GL_MODELVIEW` sau `GL_PROJECTION`, toate obiectele desenate după apelul funcţiei `glScale` vor fi desenate scalate. Trebuie de folosit `glPushMatrix` şi `glPopMatrix` pentru a memoriza şi a restabili sistemul de coordonate nescalat.

Câteva exemple de scalare.

Scalare uniformă de două ori (cu 200%)

```
glScaled(2.0, 2.0, 2.0);
```

exact acelaşi lucru poate fi făcut prin înmulţirea matricelor astfel:

```
static GLdouble su200[16] =
{
   2.,   0.,    0.,   0.,
   0.,   2.,    0.,   0.,
   0.,   0.,    2.,   0.,
   0.,   0.,    0.,   1.
};
glMultMatrixd(su200);
```

Scalare neuniformă de 1.5 ori după X şi Z (cu 150%)

```
glScaled(1.5, 1.0, 1.5);
```

exact acelaşi lucru poate fi făcut prin înmulţirea matricelor astfel:

```
static GLdouble s
nu150[16] =
{
   1.5,  0.,    0.,   0.,
   0.,   1.,    0.,   0.,
   0.,   0.,    1.5,   0.,
   0.,   0.,    0.,   1.
};
glMultMatrixd(snu150);
```

6.4. Deplasare

În OpenGL nu este funcţie specială pentru deplasare cum ar fi pentru translaţie, rotire şi scalare. Totuşi această transformaţie poate fi realizată prin înmulţirea matricelor. Matricea generală *D* pentru deplasare este

$$D = \begin{bmatrix} 1 & k_{xy} & k_{xz} & 0 \\ k_{yx} & 1 & k_{yz} & 0 \\ k_{zx} & k_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Aplicând matricea *D* la punctul [*x*, *y*, *z*, 1] obţinem punctul [*x*′, *y*′, *z*′, 1], unde

$x' = x + k_{xy} \cdot y + k_{xz} \cdot z$

$y' = k_{yx} \cdot x + y + k_{yz} \cdot z$

$z' = k_{zx} \cdot x + k_{zy} \cdot y + z$

Matricea *D* în program trebuia să fie creată şi utilizată astfel:

```
static GLdouble D[16] =
{    1., kyx, kzx,  0.,
   kxy,  1., kzy,  0.,
   kxz, kyz,  1.,  0.,
     0.,  0.,  0.,  1.,
};
glPushMatrix();
glMultMatrixd(D);
// Mai departe cream primitivele deplasate
…
glPopMatrix(); // restabilim matricea de transformări
```

Evident că coeficienţii $k_{yx}, k_{zx}, k_{xy}, k_{zy}, k_{xz}, k_{yz}$ la iniţializarea matricei `D` trebuie să fiu expresii numerice constante de tipul GLdouble sau GLfloat.

6.5. Alte transformări

Descriem fugitiv şi câteva alte transformări.

Oglindirea în planul *XY* se exprimă prin matricea

$$O_{XY} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Oglindirea în planul *YZ* se exprimă prin matricea

$$O_{YZ} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Oglindirea în planul *ZX* se exprimă prin matricea

$$O_{ZX} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Interschimbul axelor de coordonate ($X \to Z$, $Y \to X$, $Z \to Y$) se exprimă prin matricea

$$IS = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$