3. Compunerea transformărilor geometrice în OpenGL

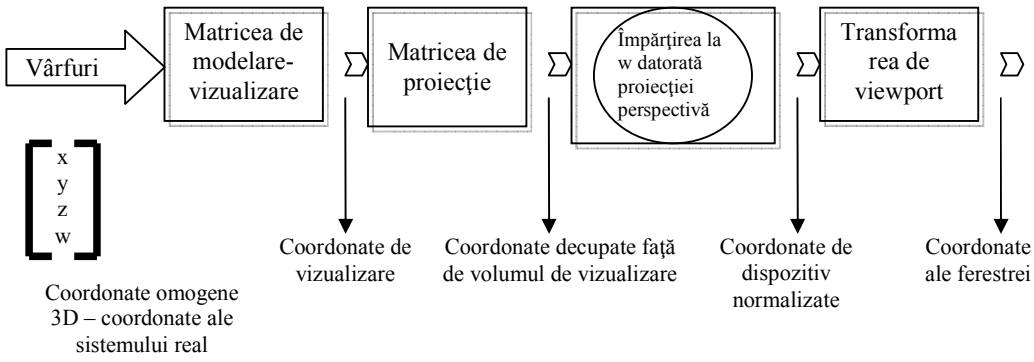
3.1 Suport teoretic

Procesul de transformare a unei scene este similar procesului de fotografiere cu o cameră. Într-o aplicaţie grafică un obiect poate trece prin 3 tipuri de transformări şi anume transformări de vizualizare, de modelare şi de proiecţie. Transformările geometrice sunt folosite pentru realizarea transformărilor de vizualizare şi modelare.

Funcţiile pentru transformări geometrice sunt:

- `glRotate#(alfa, x, y, z)` - rotaţie cu unghiul alfa faţă de axele sistemului;
- `glTranslate#(Tx, Ty, Tz)` - translaţie cu vectorul (Tx, Ty, Tz);
- `glScale#(Sx, Sy, Sz)` - scalare faţă de originea sistemului.

Transformarea de vizualizare trebuie să preceadă transformările de modelare în codul programului, dar se poate specifica proiecţia şi transformările de vizualizare în orice punct înainte ca desenarea să se producă. Schema următoare arată ordinea în care aceste operaţii au loc:



Pentru specificarea unei transformări se foloseşte o matrice `4x4` care multiplică coordonatele fiecărui vârf din scenă.

$$v' = v * M$$

În OpenGL relaţia se transpune:

`[ x' y' z' w'] = [ x y z w ] * M / T` // T-transpus

$$[x' \quad y' \quad z' \quad w']^T = M^T * [x \quad y \quad z \quad w]^T$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = M^T * \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

!OpenGL lucrează transpus, iar transformările se dau în ordine inversă.

! ! Deoarece relaţia este transpusă în OpenGL ordinea în care se vor specifica transformările (în cazul în care sunt mai multe transformări) va fi inversă succesiunii logice.

Dacă M este compusă din 3 transformări ($M = M_1 * M_2 * M_3$) în urma transpunerii se obţine:

$$M^T = M_3^T * M_2^T * M_1^T .$$

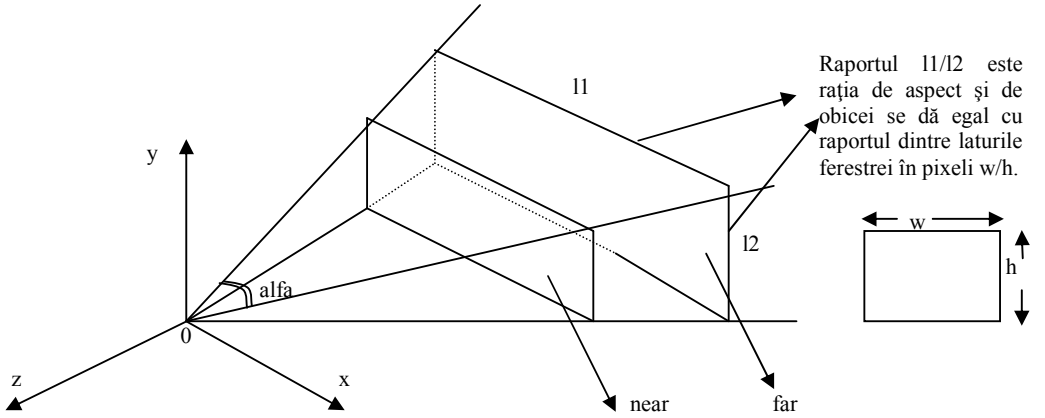
Deci, transformările trebuie apelate în ordine inversă.

Reţine, vârfurile au întotdeauna 4 coordonate, deşi în multe cazuri `w` este 1 şi pentru obiectele `2D` coordonata `z` este `0`. OpenGL lucrează cu 3 matrice curente: modelare-vizualizare, proiecţie, texturare, dar numai una dintre ele este activă la un moment dat. Pentru comutarea între ele se foloseşte rutina `glMatrixMode()` cu unul dintre parametrii: `GL_PROJECTION`, `GL_MODELVIEW`, `GL_TEXTURE`. Pentru păstrarea unei matrice curente se utilizează operaţiile cu stiva: `glPushMatrix()`- copiază matricea curentă şi o adaugă în vârful stivei, iar `glPopMatrix()`- iniţializează matricea curentă cu matricea din vârful stivei.

Transformarea de proiecţie, aşa cum îi spune şi numele, determină cum sunt proiectate obiectele pe ecran. Există 2 tipuri de proiecţie: proiecţie perspectivă, care corespunde tipului de proiecţie realizată pe retina umană (ex: obiectele care se află în depărtare se văd mici) şi proiecţie ortografică ce afişează obiectele pe ecran fără a modifica dimensiunile. Acest ultim tip se utilizează în proiectarea asistată unde imaginea finală trebuie să reflecte dimensiunile exacte ale obiectelor, paralelismul laturilor şi nu aspectul lor. Arhitecţii creează desene în perspectivă doar pentru a arăta imaginea anumitor clădiri sau spaţii interioare văzute din diferite puncte.

Proiecţia perspectivă se specifică prin:

- `gluPerspective(alfa,` $\frac{w}{h}$`, near, far)` unde:
  - `alfa` este unghiul din vârful piramidei;
  - $\frac{w}{h}$ reprezintă rata de aspect;
  - `near` indică planul apropiat;
  - `far` planul îndepărtat.



Valorile `near` şi `far` se dau întotdeauna pozitive. OpenGL le va negativa deoarece în OpenGL centrul proiecţiei este totdeauna în originea sistemului de coordonate şi volumul de vizualizare se formează în semispaţiul cu `z` negativ.

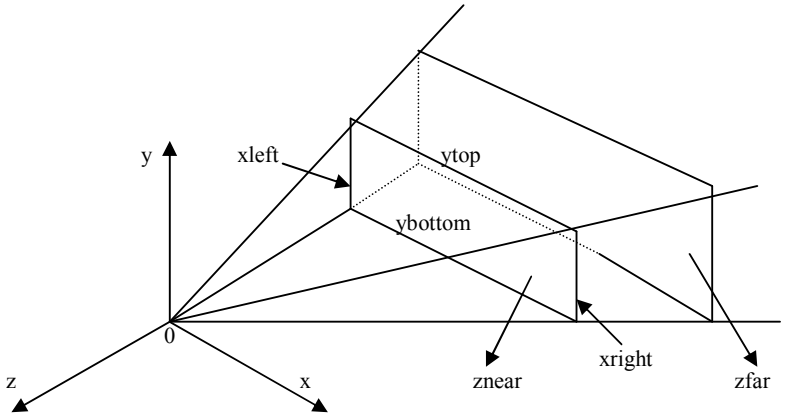
Exemplu:

```
gluPerspective(65.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);
```

În urma acestui apel OpenGL construieşte matricea de proiecţie care va fi folosită în fluxul OpenGL pentru transformarea coordonatelor 3D în coordonate 2D.

Transformarea de proiecţie se realizează parcurgând următorii paşi: se foloseşte `glMatrixMode()` cu argumentul `GL_PROJECTION`. Aceasta indică faptul că matricea curentă specifică transformarea de proiecţie. După câteva linii de cod care specifică matricea de proiecţie este apelată `glMatrixMode()` cu parametrul `GL_MODELVIEW`. Aceasta indică faptul că transformările următoare afectează matricea de modelare-vizualizare. Tot pentru specificarea proiecţiei perspective se poate folosi funcţia `glFrustum()`.

- `glFrustum(xleft, xright, ybottom, ytop, znear, zfar);`

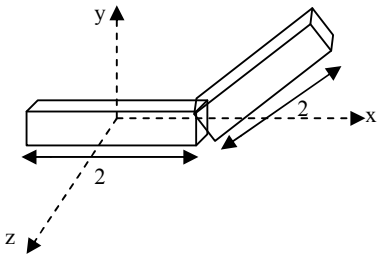


3.2 Desfăşurarea lucrării

Utilizând transformările geometrice corespunzătoare realizaţi următoarele aplicaţii:

Aplicatia 1. Se dă sursa `robot.c`.

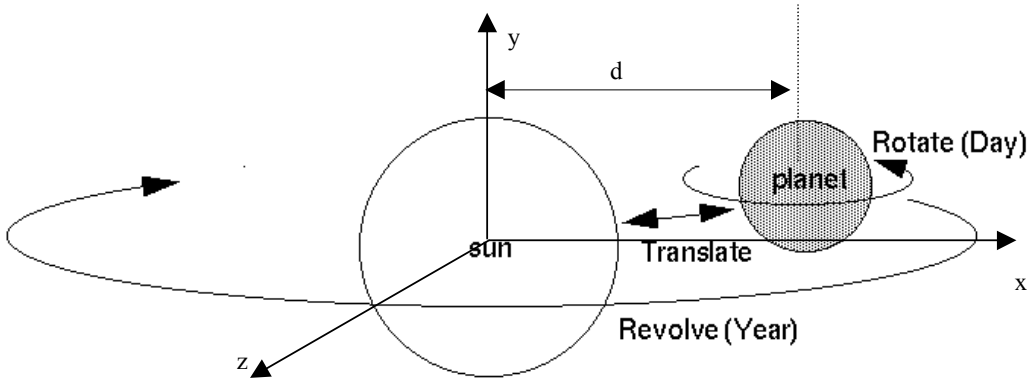
Ordinea logică a transformărilor pentru rotaţia sus-jos în încheieturile umărului şi braţului este următoarea:



/* rotaţia braţului în umăr */
3. Tx (1); // translaţie cu o unitate pe axa x a axei de rotaţie astfel încât să se suprapună peste z;
2. Rz(umăr); // rotaţie în umăr faţă de axa z;
1. Tx (-1); // se duce axa de rotaţie în poziţia originală;
4. desenare braţ;
/* rotaţia în încheietura umărului */
7. // translaţie cu 2 unităţi pe axa x;
7.1 Tx(2); // se duce antebraţul în continuarea braţului;
7.2 Tx(-1); // se translatează axa de rotaţie peste axa z;
6. Rz(cot); // rotaţie în încheietura cotului faţă de axa z;
5. Tx(1); // se translatează axa de rotaţie în poziţia originală;
8. desenare antebraţ.

Numerotarea conţine ordinea specificării operaţiilor în OpenGL.
Să se urmărească pe cod implementarea aplicaţiei. În încheieturile umărului şi cotului să se adauge rotaţia faţă-spate a braţului respectiv a antebraţului.

Aplicaţia 2. Se dă sursa `planet.c`.



Ordinea transformărilor în sursa originală:
1. Desenează soare

Se anulează deoarece înmulţite dau I.
Tx(d); // se deplasează planeta la distanţa d faţă de Soare;
Tx(-d); // se duce axa rotaţiei astfel încât să se suprapună peste axa x;

4. Ry(day) // rotaţie diurnă faţă de axa y (rotaţia planetei în jurul axei proprii);
3. Tx (d) // se duce axa de rotaţie în poziţia originală;
2. Ry(year) // rotaţia planetei în jurul Soarelui;
5. Desenează planetă.

Să se adauge un satelit planetei, care se va roti în jurul planetei, în jurul axei proprii dar şi odată cu planeta în jurul Soarelui.