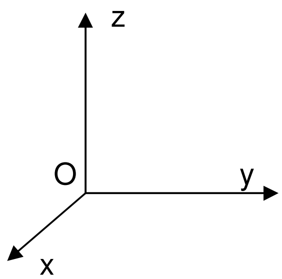
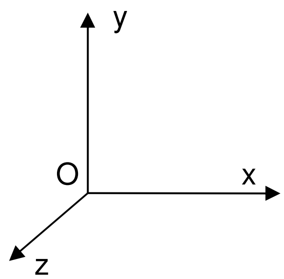
Elemente de geometrie computațională. Sisteme de coordonate. Transformări 3D.

Sistem de axe de coordonate

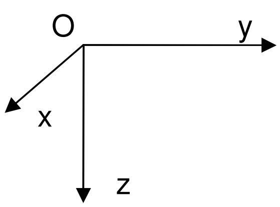
Orientat drept:



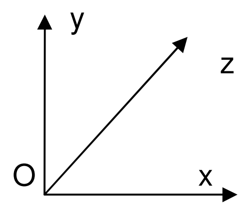
sau



Orientat stâng:



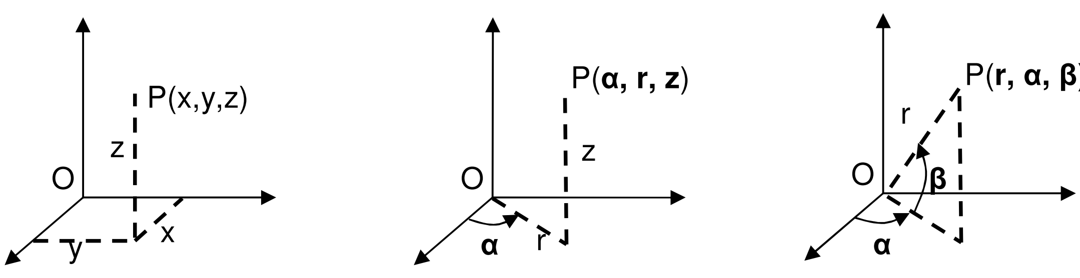
sau



Punct

Un punct P din $\mathbf{R}^3$ se poate preciza prin:

- Coordonate **carteziene**: P(x,y,z)
- Coordonate **omogene**: P(x,y,z,u)
- Coordonate **cilindrice**: P(α, r, z)
- Coordonate **sferice**: P(r, α, β)



carteziene cilindrice sferice

Conversia de la coordonate **carteziene** la cele **omogene**: (x,y,z) => (x,y,z,1)
Conversia de la coordonate **omogene** la cele **carteziene**: (x,y,z,u) => (x/u,y/u,z/u), pentru $u \neq 0$

Avantajele folosirii coordonatelor omogene (după cum se va vedea în continuare):

- descriere unitară a transformărilor
- o succesiune de transformări se poate descrie cu ajutorul unei singure matrice

In OpenGL un vârf dintr-o primitivă de desenare se precizează astfel:

$$\text{glVertex}\left\{\begin{matrix}2\\3\\4\end{matrix}\right\}\left\{\begin{matrix}b\\s\\i\\f\\ub\\us\\ui\end{matrix}\right\}[v](coordonate)$$

unde numărul şi tipul argumentelor se precizează în comandă.
Cu două dimensiuni în comandă se preupune că valoarea z=0, iar cu 4 valori se precizează coordonatele omogene.
Exemple:

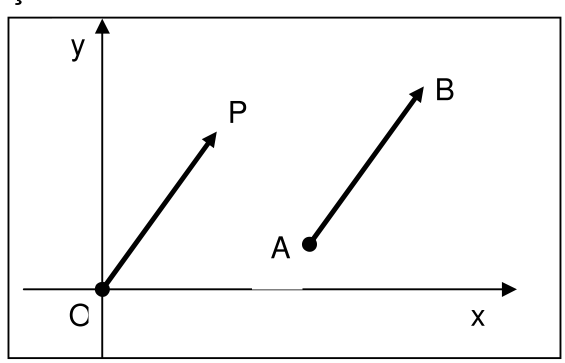
```
glVertex2i(1, 0);
glVertex3f(0.0f, 1.5f, 2.0f);
```

In diferitele calcule ce apar un **punct** (dat prin coordonate omogene) se reprezintă printr-o **matrice** coloană:

$$P=\begin{pmatrix}x\\y\\z\\w\end{pmatrix}$$

Vector

- Un vector este determinat de două vârfuri şi un sens
- Un vector are: origine, direcţie (suport şi sens), lungime.



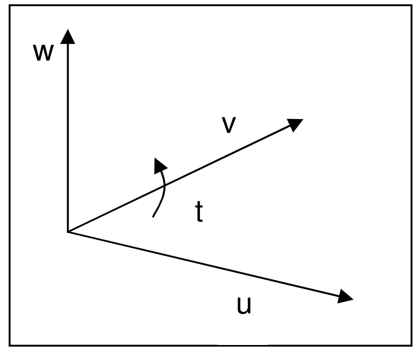
Fie vectorul: $\overrightarrow{AB}$, A(x_1, y_1, z_1), B(x_2, y_2, z_2)

Vectorul $\overrightarrow{AB}$ este **paralel** şi de aceeaşi **lungime** cu vectorul $\overrightarrow{OP}$, unde P($x_2 - x_1$, $y_2 - y_1$, $z_2 - z_1$).

Fie $\vec{v}$ (a, b, c), atunci **(a, b, c)** sunt **parametrii directori ai vectorului**,
iar $\|v\| = \sqrt{a^2 + b^2 + c^2}$ este **lungimea** vectorului

Fie $\vec{u}$ (a, b, c), $\vec{v}$ (a', b', c'), doi vectori.

- Produsul scalar: **scalar**(u,v) = aa'+bb'+cc'=$\|u\| . \|v\| . \cos(t)$, unde *t* este unghiul dintre cei doi vectori. Cu produsul scalar se poate calcula **unghiul dintre doi vectori.** Dacă produsul scalar este nul, atunci cei doi vectori sunt **perpendiculari**.
- Produsul vectorial: **vect(u,v)** este un vector ***w*** perpendicular pe planul determinat de vectorii *u* şi *v* şi pentru care triedrul (u,v,w) este **drept**



$$\vec{w} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a & b & c \\ a' & b' & c' \end{vmatrix} = \vec{i}\,(bc'-b'c) - \vec{j}\,(ac'-a'c) + \vec{k}\,(ab'-a'b),$$

deci $\vec{w}(bc'-b'c, ac'-a'c, ab'-a'b)$

- Alte operatii utile cu vectori:
  - adunarea a doi vectori
  - produsul unui vector cu un scalar
  - scăderea a doi vectori
  - combinația liniară a doi sau mai mulți vectori

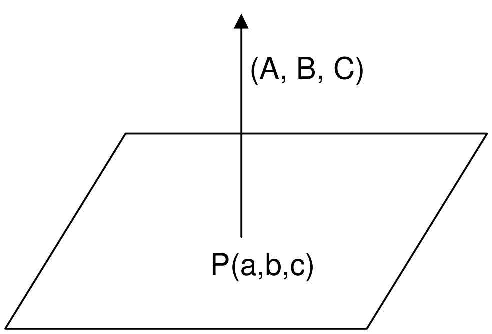
Plan

Un plan se poate preciza în următoarele moduri:

1. Planul care trece prin punctul P(a,b,c) şi este perpendicular pe vectorul $\vec{v}$ (A, B, C):

Ax + By + Cz + D = 0, unde D= -Aa-Bb-Cc (punctul P aparține planului)

deci coeficienții variabilelor din ecuația planului sunt parametrii directori ai normalei la plan:



2. Planul determinat de trei puncte P_i (x_i, y_i, z_i), i=1,2,3 (**ecuația planului sub formă de determinant)** este:

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0$$

3. Planul determinat de trei puncte P_i (x_i, y_i, z_i), i=1,2,3 (**ecuația planului sub formă parametrică)** este:

$$\begin{cases} x = x_1 + u(x_2 - x_1) + v(x_3 - x_1), \\ y = y_1 + u(y_2 - y_1) + v(y_3 - y_1), \text{ unde } u, v \in R \\ z = z_1 + u(z_2 - z_1) + v(z_3 - z_1), \end{cases}$$

Probleme utile:

- Determinarea distanței de la un punct **P(a, b, c)** la planul **(α) Ax+By+Cz+D=0**
- Determinarea unghiului dintre două plane
- Condiția ca două plane să fie perpendiculare

Dreapta

1. O dreaptă precizată prin intersecția a două plane:

$$\begin{cases} A_1x + B_1y + C_1z + D_1 = 0 \\ A_2x + B_2y + C_2z + D_2 = 0 \end{cases} \text{ sau echivalent: } \begin{cases} x = mz + n, \\ y = pz + q \end{cases}$$

2. O dreaptă care trece prin două puncte P_i (x_i, y_i, z_i), i=1,2, are ecuația:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1},$$

sau parametric (mai uşor de folosit): $\begin{cases} x = x_1 + t(x_2 - x_1), \\ y = y_1 + t(y_2 - y_1), \\ z = z_1 + t(z_2 - z_1), \quad t \in R \end{cases}$

3. O dreaptă care trece prin punctul P(x_0, y_0, z_0) şi are direcția dată de vectorul $\vec{v}$ (a, b, c):

$$\frac{x - x_0}{a} = \frac{y - y_0}{b} = \frac{z - z_0}{c},$$

sau sub formă parametrică (mai utilă): $\begin{cases} x = x_1 + ta, \\ y = y_1 + tb, \\ z = z_1 + tc, \quad t \in R \end{cases}$

Transformari 3D de baza

Un punct **P(x,y,z)** se poate transforma într-un punct **P'(x', y', z')** printr-o matrice **4x4**, dacă **P** şi **P'** se precizează în **coordonate omogene**:

P(x,y,z,1) ---> P'(x', y', z',1)

$$\begin{pmatrix} x' & y' & z' & 1 \end{pmatrix} = \begin{pmatrix} x & y & z & 1 \end{pmatrix} * A, \text{ unde: } A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix},$$

sau $\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = T * \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$, unde $T = A^t$ (transpusa lui A).

In OpenGL se foloseşte a doua variantă de precizare a transformării, cu un **vector coloană** pentru coordonatele omogene ale unui punct.

In continuare se va determina matricea T pentru diverse **transformări de bază**.

Translația

Translația este transformarea prin care toate punctele unui obiect se deplasează în aceeaşi direcție şi cu aceeaşi distanță.

Precizarea acestei transformări se poate face în trei moduri:

1. Prin indicarea **deplasărilor**, pe cele trei axe de coordonate, **(dx, dy, dz)**, deplasări pozitive sau negative, necesare pentru a transforma un punct $\mathbf{P_1}$ (x_1, y_1, z_1) în punctul $\mathbf{P_2}$ (x_2, y_2, z_2):
   $x_2 = x_1 + dx$,
   $y_2 = y_1 + dy$,
   $z_2 = z_1 + dz$
   Lungimea cu care se face translația este: $d = \sqrt{dx^2 + dy^2 + dz^2}$
2. Prin precizarea **direcției**, un vector: $\vec{v}$ (a, b, c), şi a **lungimii** deplasării: **d**.
   Ecuația dreptei care trece prin $\mathbf{P_1}$ (x_1, y_1, z_1) (punctul care se transformă) şi are direcția $\vec{v}$ este:

$$x= x_1+at,\ y=y_1+bt,\ z=z_1+ct.$$

   Deoarece punctul P_2 (x_2, y_2, z_2)=P_2(x_1+dx, y_1+dy, z_1+dz) va fi pe această dreaptă, iar distanța dintre P_1 şi P_2 va fi **d**, se determină: $t = d / \sqrt{a^2 + b^2 + c^2}$, deci:

$$dx = t * a = d * a / \sqrt{a^2 + b^2 + c^2},$$
$$dy = d * b / \sqrt{a^2 + b^2 + c^2}, \qquad .$$
$$dz = d * c / \sqrt{a^2 + b^2 + c^2}$$

3. Prin coordonatele (x_2, y_2, z_2) în care se translatează (x_1, y_1, z_1), deci:

$$dx = x_2 - x_1,\ dy = y_2 - y_1,\ dz = z_2 - z_1.$$

Folosind valorile **(dx, dy, dz)**, care se pot determina pentru fiecare caz, **matricea de translație** este:

$$T(dx, dy, dz) = \begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

In OpenGL această transformare se precizează prin comanda:

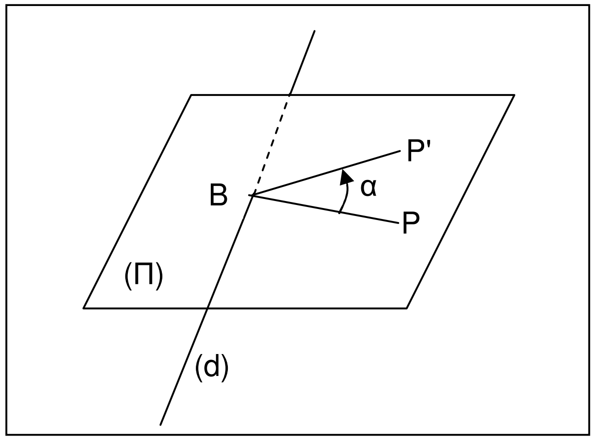
```
glTranslate{f|d}(dx,dy,dz)
```

Rotația in jurul unei axe

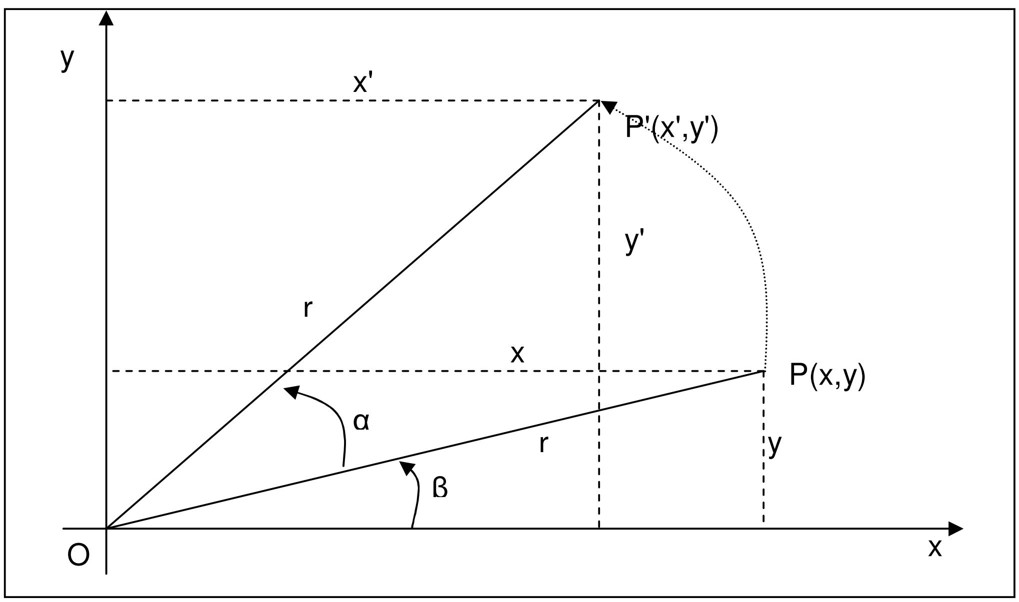
Rotația se face cu un unghi în jurul:

- unui **punct** - dacă transformarea are loc în plan (reprezentată în figura de mai jos)
- unei **drepte** (d) - dacă transformarea se face în spațiu

Pentru rotația în spațiu a unui punct P, fie α unghiul de rotație şi (Π) planul care trece prin punctul P şi este perpendicular pe dreapta de rotație (d), iar B punctul de intersecție dintre (d) şi (Π). Punctul P, care nu este pe dreapta de rotație (d), se transformă într-un punct P' astfel încât punctele P şi P' sunt situate în planul (Π), distanțele de la dreapta de rotație la P şi P' sunt egale (BP=BP'), şi unghiul PBP' are măsura α.



Pentru început vom analiza o rotație în plan. Considerăm un sistem de axe xOy în plan şi un punct P(x,y) care se roteşte în jurul punctului O. După o rotație de unghi **α** se obține un punct P'(x',y').



Obținem succesiv relațiile:

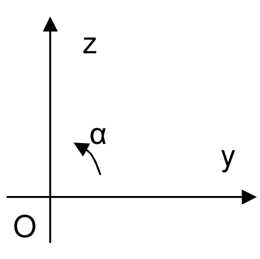
$$x = r * \cos(\beta);\ y = r * \sin(\beta)$$

$$x' = r * \cos(\alpha + \beta) = r * \cos(\alpha) * \cos(\beta) - r * \sin(\alpha) * \sin(\beta) = x * \cos(\alpha) - y * \sin(\alpha),$$

$$y' = r * \sin(\alpha + \beta) = r * \sin(\alpha) * \cos(\beta) + r * \sin(\beta) * \cos(\alpha) = x * \sin(\alpha) + y * \cos(\alpha); \qquad (*)$$

Folosim relațiile de mai sus pentru a obține formulele de translație în $\mathbf{R^3}$. Pentru o rotație în spațiu, în jurul axei **Oz**, un punct P(x,y,z) devine punctul P'(x', y', z'), unde x' şi y' se calculează după formulele (*) de mai sus, iar z'=z.

Pentru o rotație în jurul axei **Ox** obținem:

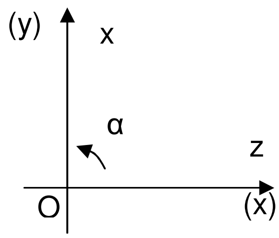


$$x' = x;$$

$$y' = y * \cos(\alpha) - z * \sin(\alpha),$$

$$z' = y * \sin(\alpha) + z * \cos(\alpha);$$

Pentru o rotație în jurul axei **Oy** obținem:



$$
\begin{aligned}
z' &= z * \cos(\alpha) - x * \sin(\alpha), \\
y' &= y; \\
x' &= z * \sin(\alpha) + x * \cos(\alpha);
\end{aligned}
$$

Folosind coordonatele omogene, pentru rotațiile amintite mai sus se obțin următoarele matrici de transformare:

Rotație în jurul axei Ox: $T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = RX(\alpha)$

Rotație în jurul axei Oy: $T = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = RY(\alpha)$

Rotație în jurul axei Oz: $T = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = RZ(\alpha)$

In OpenGL rotația se precizează prin comanda:

```
glRotate{f|d}(a,p,q,r)
```

care precizează o rotatie de unghi "a", dat în grade, în jurul dreptei specificate de parametrii directori (p,q,r). Pentru diferite valori particulare ale parametrilor directori (p,q,r) se obțin rotațiile particulare descrise mai sus.

Scalarea

Scalarea presupune înmulțirea coordonatelor cu anumite constante, numiți **factori de scară**. Dacă aceste constante sunt f_x, f_y, f_z (corespunzător celor trei axe de coordonate), atunci matricea de transformare este:

$$T = \begin{pmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & f_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = SC(f_x, f_y, f_z)$$

Simetria

Vom preciza **simetria (reflexia)** pentru: origine, axele de coordonate, planele de coordonate. Aceste transformări se pot descrie unitar prin matricea de transformare:

$$T = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \text{SIM (a, b, c)}.$$

Cazuri particulare de transformări obținem astfel:

- a=b=c=-1, se obține simetria față de origine;
- a=b=1, c=-1, se obține simetria față de planul xOy;
- a=1, b=-1, c=1, se obține simetria față de planul xOz;
- a=-1, b=c=1, se obține simetria față de planul yOz;
- a=1, b=c=-1, se obține simetria față de axa Ox;
- a=-1, b=1, c=-1, se obține simetria față de axa Oy;
- a=b=-1, c=1, se obține simetria față de axa Oz;

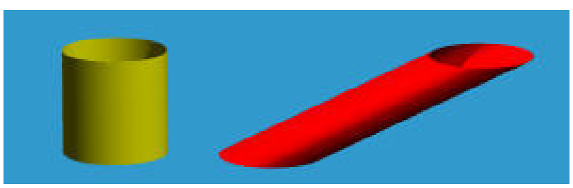
Observație: Din matricea de transformare pentru scalare şi simetrie se observă că acestea se pot descrie unitar, printr-o singură matrice, deci scalarea şi simetria se pot preciza în paralel printr-o singură matrice de transformare.

In OpenGL ultimele două transformări se precizează prin comanda:

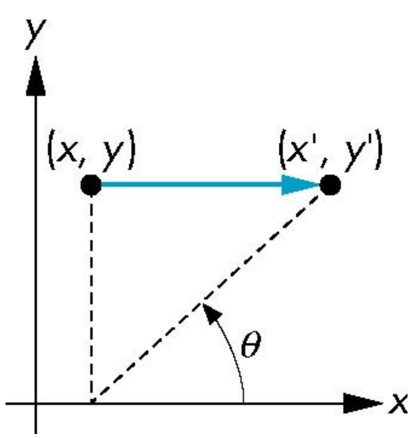
```
glScale{f|d}(x,y,z)
```

Forfecare (shear)

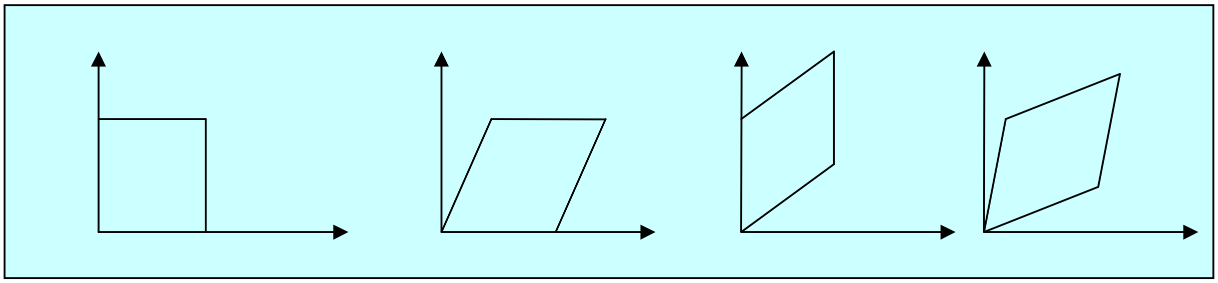
Această transformare modifică dimensiune şi forma obiectelor. In imaginea următoare se dă un exemplu de astfel de transformare, unde primul obiect (cilindru) este transformat în al doilea obiect.



Transformarea se poate face în direcția axei Ox, a axei Oy, a axei Oz, a două sau trei axe. In următoarea figură se precizează faptul că un punct (x,y) din plan se transformă în punctul (x',y'), unde valoarea coordonatei x este proporționată cu valoarea lui y.



Următoarea figură precizează un patrat care se transformă în direcția axei Ox, apoi Oy, iar în final în direcția ambelor axe.



In cazul în care această transformare se face în spațiu, în direcția axelor x şi y, cu coordonata z neschimbată, formula de calcul este:

$$x' = x + az$$
$$y' = y + bz$$
$$z' = z$$

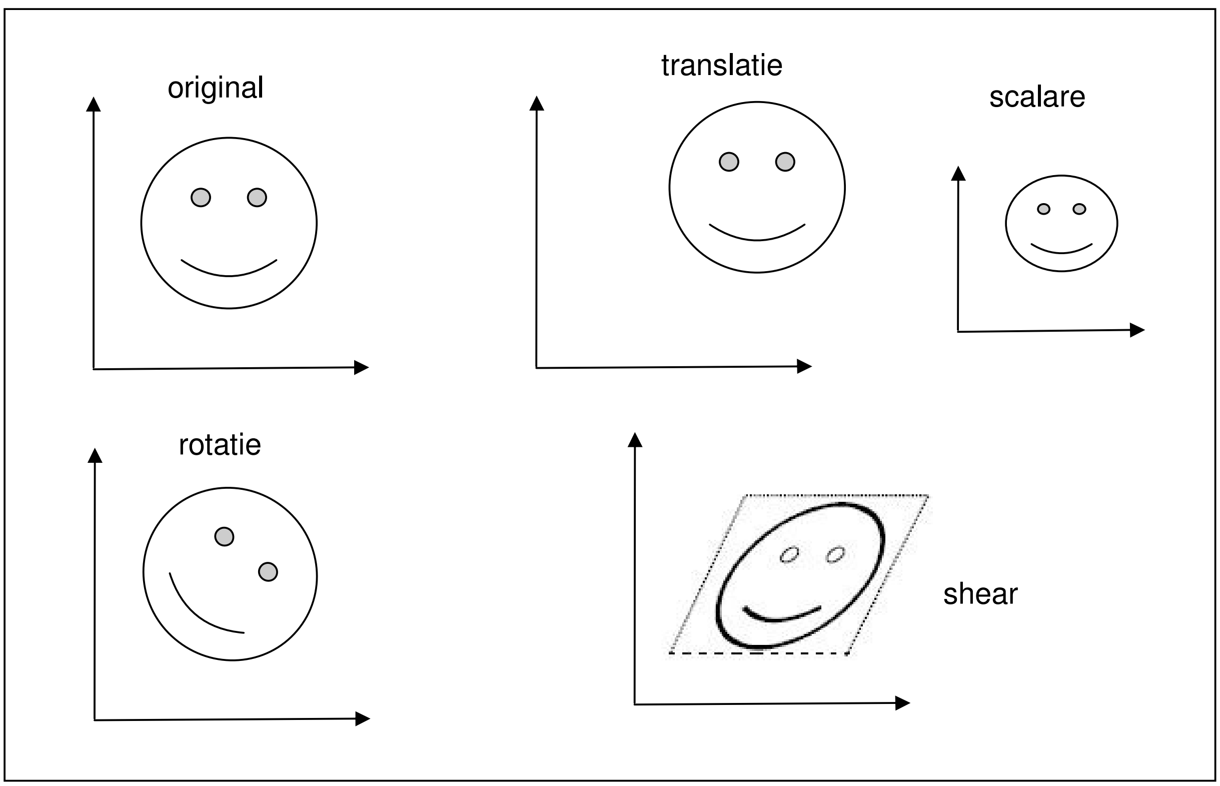
In OpenGL nu există comenzi pentru precizarea acestor tipuri de transformări. Ele se pot specifica prin folosirea comenzilor:

- **glLoadMatrix{f|d}(matrice, adr)** - se încarcă o matrice precizată de argument (se iau datele începând cu o anumită adresă)
- **glMultMatrix{f|d}(matrice, adr)** - matricea curentă se înmulțeşte din argument

Exemplu:

```
float M1[] = {1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1};  //matricea unitate
M1[4]=a;                   //x'=x+a*y
M1[1]=b;                   //y'=b*x+y
gl.glMultMatrixf(M1,0);  //coloana dupa coloana
```

Rezumat transformări 3D



Transformari 3D compuse

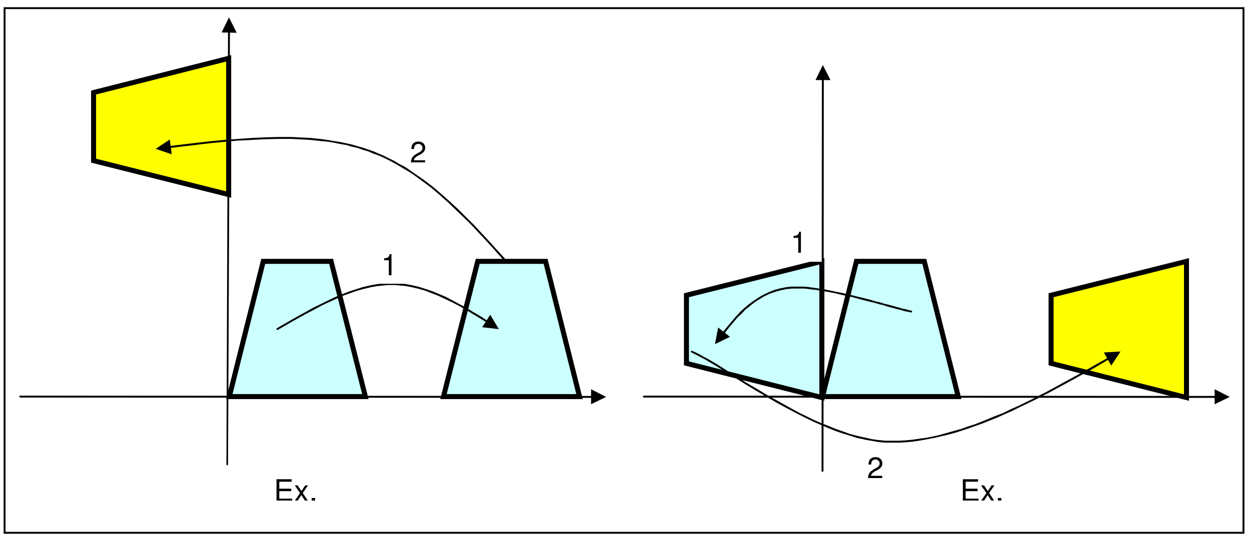
Fie T_1 şi T_2 două transformări, care transformă succesiv **P(x,y,z)** în **$P_1(x_1,y_1,z_1)$**, apoi **P_1** în **$P_2(x_2,y_2,z_2)$.**

Fie A_1 şi A_2 matricile corespunzătoare acestor două transformări. In acest caz obținem formula de calcul pentru punctele **P_1** şi **P_2**:

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} = A_1 * \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}; \quad \begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} = A_2 * \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} = A_2 * A_1 * \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

Din ultima relație se observă că produsul matricelor de transformare (deci şi precizarea lor în OpenGL) se face în **ordinea inversă efectuării transformărilor** ($A_2 * A_1$). Deoarece produsul matricelor nu este comutativ, rezultatul succesiunii de transformări T_1 urmată de T_2 este (în general) diferit de transformarea T_2 urmată de transformarea T_1.

In continuare este un exemplu, în plan, din care se vede o justificare a acestei afirmatii.



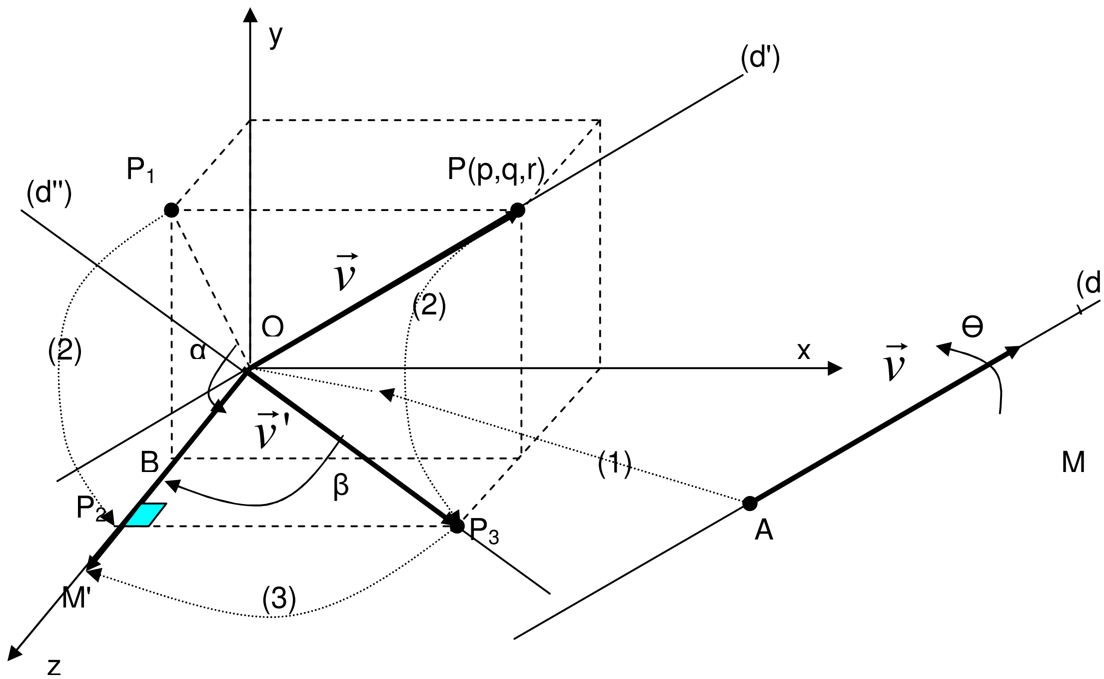
Transformările din acest exemplu sunt:
Ex.1: ***1*** - translație pe Ox; ***2*** - rotație de 90^0;
Ex.2: ***1*** - rotație de 90^0; ***2*** - translație pe Ox;

Transformările complexe pot să fie descompuse în transformări de bază (descrise mai sus). Ca exemplificare vom lua câteva astfel de transformări.

1. Simetria față de un punct P(a,b,c) oarecare:
T_1: Translatăm P în origine: T(-a,-b,-c);
T_2: Determinăm simetria față de origine: SIM(-1,-1,-1);
T_3: Translatăm O în P: T(a,b,c).

De aici se deduce o matrice pentru transformare:
$\mathrm{T(a,b,c) * SIM(-1,-1,-1) * T(-a,-b,-c)}$.

2. Rotația în jurul unei drepte oarecare:



Presupunem că dreapta (axa) de rotație (d) trece prin punctul A(a,b,c) şi are parametrii directori $\vec{v}$ (p,q,r). Se cere efectuarea unei rotații de unghi **Θ** în jurul dreptei (d). Fie M un punct oarecare pentru care se face această rotație.

O modalitate de reducere a acestei transformări la transformări de bază amintite mai sus este descrisă în continuare (nu este singura variantă). Se va aduce dreapta (d) peste una din axele de coordonate, prin transformări de bază, iar în jurul acestei axe de coordonate se va efectua rotația de unghi **Θ.** După acestă rotație se va readuce dreapta (d) la locul ei inițial.

T_1: Translația lui A în O, descrisă de matricea T1=T(-a,-b,-c). După această transformare dreapta (d) se va translata într-o dreaptă paralelă (d'), iar vectorul $\vec{v}$ se va translata în vectorul **OP** paralel cu $\vec{v}$.

T_2: Rotație în jurul axei Ox astfel încât vectorul $\vec{v}$ să ajungă în planul xOz, deci vectorul $\vec{v}$ se transformă în vectorul $\vec{v}\,' = \overrightarrow{OP_3}$. După această rotație punctul P ajunge în punctul P_3, punctul P_1 ajunge în P_2, iar dreapta (d') se transformă în (d''), inclusă în planul xOz. Unghiul de rotație α se poate determina din triunghiul P_1OB. Transformarea astfel precizată este T2=RX(α).

T_3: Rotație în jurul axei Oy astfel încât dreapta (d'') se suprapune peste axa Oz (vectorul $\vec{v}\,'$ se suprapune peste Oz). Unghiul de rotație β se poate determina din triunghiul OP_2P_3. Rotația în jurul axei Oy se face de la axa Ox spre Oz, deci transformarea de bază aşa cum este descrisă mai sus (care este descrisă de la z spre x) se face cu unghiul (-β). Transformarea astfel realizată este T3=RY(-β).

T_4: In jurul axei Oz se face rotația de unghi Θ, deci transformarea este T4=RZ(Θ).

T_5: Se efectueze transformarea inversă de la T_3, deci o retație de unghi β în jurul axe Oy. Transformarea astfel descrisă este T5=RY(β).

T_6: Se face o rotație de unghi (-α) în jurul axei Ox, deci avem transformarea T6= RX(-α) (este transformarea inversă de la T_2.

T_7: Se face o translație T7=T(a,b,c), inversă transformării T_1.

Din cele descrise mai sus rezultă că matricea de transformare pentru rotația în jurul unei axe oarecare este:

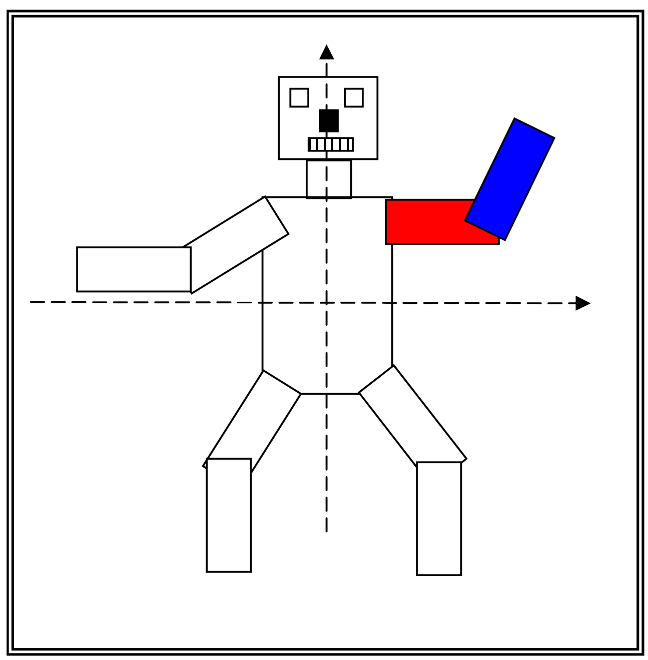
$$\mathbf{T=T(a,b,c)} * \mathbf{RX(-\alpha)} * \mathbf{RY(\beta)} * \mathbf{RZ(\Theta)} * \mathbf{RY(-\beta)} * \mathbf{RX(\alpha)} * \mathbf{T(-a,-b,-c).}$$

Precizarea transformărilor succesive în OpenGL:

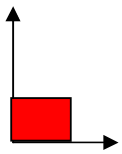
- la precizarea unui punct dintr-o primitivă se foloseşte o matrice curentă de transformare prin care se determină poziția punctului curent în spațiu
- pentru stabilirea matricei curente de transformare se folosesc următoarele comenzi:
    - se precizează o matrice inițială prin **glLoadIdentity()** sau **glLoadMatrix**{f|d}(matrice, adr)
    - matricea curentă se poate înmulți (la dreapta) cu o matrice dată prin comanda **glMultMatrix**{f|d}(matrice, adr) sau printr-o matrice determinată de una din transformările descrise mai sus: **glRotate, glScale, glTranslate**.
    - comenzile **glLoadMatrix** şi **glMultMatrix** se folosesc pentru transformări particulare (de ex. forfecarea)
    - ordinea de precizare a comenzilor este inversă ordinii de efectuare a transformărilor (ultima comandă/matrice precizată corespunde la prima transformare care se aplică).
- In OpenGL se poate gestiona o **stivă** de matrici de transformare, cu ajutorul a două comenzi:
    - **glPushMatrix**() - care salvează matricea curentă (o memorează în vârful stivei),
    - **glPopMatrix**() - care înlocuieşte matricea curentă cu matricea aflată în vârful stivei (iar această poziție se elimină din stivă).

Ierarhii de transformări

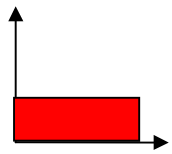
Sa presupunem că dorim să vizualizăm o figură asemănătoare cu cea din imaginea alăturată (un robot). Pentru construirea acestui obiect se poate proceda astfel (se parcurg mai mulți paşi).



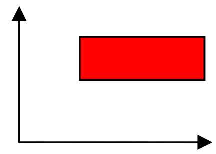
Pasul 1: Se pleacă de la un patrat.



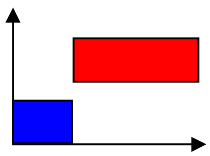
Pasul 2: Se face o scalare.



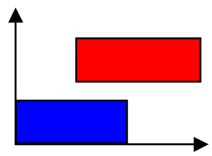
Pasul 3: Se face o translație.



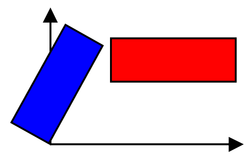
Pasul 4: Se adaugă un nou patrat.



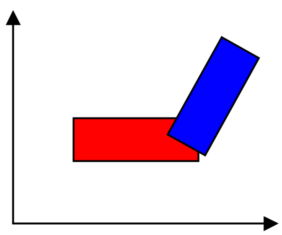
Pasul 5: Se face o scalare.



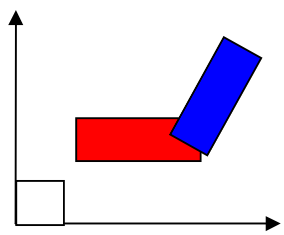
Pasul 6: Se face o rotație.



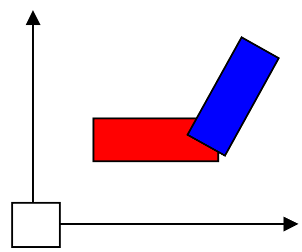
Pasul 7: Se face o translație.



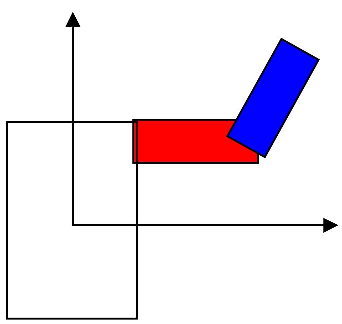
Pasul 8: Se adaugă un nou patrat.



Pasul 9: Se face o translație.



Pasul 10: Se face o scalare.



Transformările se pot face asupra:

- Intregului obiect (după construirea lui)
- Fiecărei componente (în momentul în care se adaugă aceasta, pentru fiecare componentă se foloseşte o mulțime de transformări, independente de transformările celorlalte componente)
- Unor componente din obiect (dacă ele se memorează eficient şi se pot face astfel de transformări pentru un grup de componente).

O astfel de memorare eficientă este cea ierarhică (arborescentă), sugerată în continuare:

Intregul obiect

1. Capul robotului
    1.1. Gura robotului
    1.2. Nasul robotului
    1.3. Ochiul stâng al robotului

        1.4. Ochiul drept al robotului
    2. Corpul robotului
    3. Brațul stâng al robotului
        3.1. Brat inferior
        3.2. Brat superior
    4. Brațul drept al robotului
        4.1. Brat inferior
        4.2. Brat superior
    5. Piciorul stâng al robotului
        5.1. Partea inferioară
        5.2. Partea superioară
    6. Piciorul drept al robotului
        6.1. Partea inferioară
        6.2. Partea superioară

La fiecare dintre aceste componente (noduri din arbore) se poate asocia o mulțime de transformări