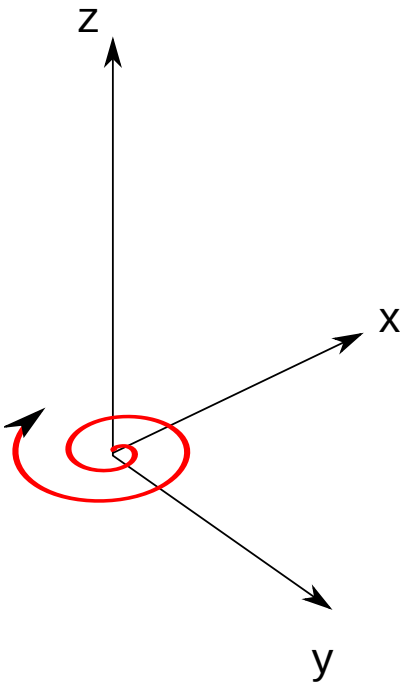
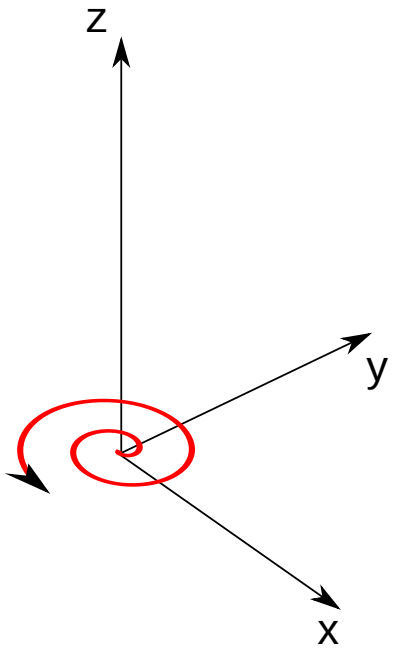
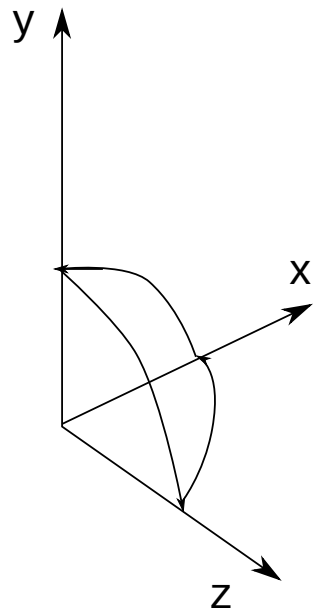
3D koordináta-rendszerek



balkezes
bal-sodrású





jobbkezes
jobb-sodrású

3D transzformációk - homogén koordináták

(x, y, z) megadása homogén koordinátákkal: $(x, y, z, 1)$

$(x, y, z, w) = (x', y', z', w')$, ha van olyan α, hogy
$x' = \alpha \cdot x$, $y' = \alpha \cdot y$, $z' = \alpha \cdot z$, és $w' = \alpha \cdot w$

Ha $w \neq 0$: $(x/w, y/w, z/w, 1)$ a szokásos jelölés.

Ha $w = 0$: $(x, y, z, 0)$ végtelen távoli pont.

Kapcsolat: (x, y, z) - egyenes a 4-dimenziós térben, aminek a $w = 1$ 3D térrel való metszete homogén koordináta.

3D eltolás

$$T(d_x, d_y, d_z) = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

mert

$$T(d_x, d_y, d_z) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + d_x \\ y + d_y \\ z + d_z \\ 1 \end{pmatrix}$$

$$T^{-1}(d_x, d_y, d_z) = T(-d_x, -d_y, -d_z)$$

3D skálázás (kicsinyítés / nagyítás)

$$S(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

mert

$$S(s_x, s_y, s_z) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} s_x \cdot x \\ s_y \cdot y \\ s_z \cdot z \\ 1 \end{pmatrix}$$

$$S^{-1}(s_x, s_y, s_z) = \begin{pmatrix} \frac{1}{s_x} & 0 & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 & 0 \\ 0 & 0 & \frac{1}{s_z} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3D forgatások

A z-tengely körül:

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

Az x-tengely körül:

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

Az y-tengely körül:

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

3D nyírás

$$SH_{xy}(sh_x, sh_y) = \begin{pmatrix} 1 & 0 & sh_x & 0 \\ 0 & 1 & sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

mert

$$SH_{xy}(sh_x, sh_y) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + sh_x \cdot z \\ y + sh_y \cdot z \\ z \\ 1 \end{pmatrix}$$

3D kompozíció-mátrix

A legáltalánosabb kompozíció alakja:

$$M = \begin{pmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A mátrix szorzáshoz képes most is meg lehet takarítani műveleteket.

3D síkok transzformációi

A sí egyenlete: $Ax + By + Cz + D = 0$
Legyen P a sík egy tetszőleges pontja!

Ha $P = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$, akkor $N = \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix}$ a sík normálisa, hiszen $N^T P = 0$.

Ha a sík pontjait M-mel transzformáljuk, akkor hogyan transzformálódik a sík normálisa?

3D síkok transzformációi

Legyen P egy tetszőleges pont a síkban! Ekkor $N^T P = 0$.

Melyik az a Q mátrix, amelyre $(QN)^T(MP) = 0$?

Ha M^{-1} létezik, akkor

$$((M^{-1})^T N)^T (MP) = N^T((M^{-1})^T)^T MP = N^T P = 0$$

$$\Downarrow$$

$$Q = (M^{-1})^T$$

$$N' = (M^{-1})^T N$$

Megyjegyzés: Nem biztos, hogy M^{-1} létezik! Pl.: vetítés esetén)

3D koordináta-rendszerek váltása - 1

$P^{(j)}$: a P pont a j koordináta-rendszerben.

$M_{i \leftarrow j}$: transzformáció, amely a j-koordináta-rendszerbeli pontokat az i koordináta-rendszerbe viszi át.

Ekkor

$$P^{(i)} = M_{i \leftarrow j} P^{(j)}$$

Ha $P^{(j)} = M_{j \leftarrow k} P^{(k)}$ akkor

$$P^{(i)} = M_{i \leftarrow j} P^{(j)} = M_{i \leftarrow j}(M_{j \leftarrow k} P^{(k)}) = M_{i \leftarrow k} P^{(k)}$$

ahol

$$M_{i \leftarrow k} = M_{i \leftarrow j} M_{j \leftarrow k}$$

3D koordináta-rendszerek váltása - 2

Továbbá

$$M_{i\leftarrow j} = M_{j\leftarrow i}^{-1}$$

Pl.:

a) Ha $M_{i\leftarrow j} = T(t_x, t_y)$, akkor $M_{j\leftarrow i} = T(-t_x, -t_y)$.

b) Ha R jobb-, L pedig bal-kezes koordináta-rendszer azonos origóval és párhuzamos tengelyekkel, akkor

$$M_{R\leftarrow L} = M_{L\leftarrow R}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3D transzformációk alakja (Különböző koordináta-rendszerekben)

$P^{(}j)$: pont a j-koordináta-rendszerben
$Q^{(j)}$: transzformáció a j-koordináta-rendszerben

Melyik az a $Q^{(i)}$, amelyre

$$Q^{(i)}P^{(i)} = M_{i\leftarrow j}Q^{(j)}P^{(j)}?$$

Mivel $P^{(i)} = M_{i\leftarrow j}P^{(j)}$, ezért

$$Q^{(i)}M_{i\leftarrow j}P^{(j)} = M_{i\leftarrow j}Q^{(j)}P^{(j)}$$

$$\Downarrow$$

$$Q^{(i)}M_{i\leftarrow j} = M_{i\leftarrow j}Q^{(j)}$$

$$Q^{(i)} = M_{i\leftarrow j}Q^{(j)}M_{i\leftarrow j}^{-1}$$

OpenGL

Mátrix műveletek

OpenGL-ben oszlopfolytonosan tároljuk a mátrixokat:
Az egység mátrix:

$$GLfloatM[] = \{ \begin{matrix} 1.0, & 0.0, & 0.0, & 0.0, \\ 0.0, & 1.0, & 0.0, & 0.0, \\ 0.0, & 0.0, & 1.0, & 0.0, \\ 0.0, & 0.0, & 0.0, & 1.0\} \end{matrix}$$

$$\begin{pmatrix} a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \\ a_4 & a_8 & a_{12} & a_{16} \end{pmatrix}$$

Új aktuális mátrix betöltése:

`void glLoadMatrix{fd}(T` $M[16]$`);`

```
glMatrixMode(GL_MODELVIEW); // típus
glLoadMatrix(M); // betöltés
```

Mátrix műveletek

Az aktuális mátrix legyen az egységmátrix:

```
void glLoadIdentity(void);
```

Az aktuális mátrix szorzása:

```
void glMultMatrix{fd}(T M[16]);
```

Pl.:

$$GLfloatM[] = \{ \begin{matrix} 1.0, & 0.0, & 0.0, & 1.0, \\ 0.0, & 1.0, & 0.0, & 0.0, \\ 0.0, & 0.0, & 1.0, & 0.0, \\ 0.0, & 0.0, & 0.0, & 1.0\} \end{matrix}$$

```
glMatrixMode(GL_MODELVIEW);
glMultMatrix(M);
```

A szorzat lesz az új aktuális mátrix.

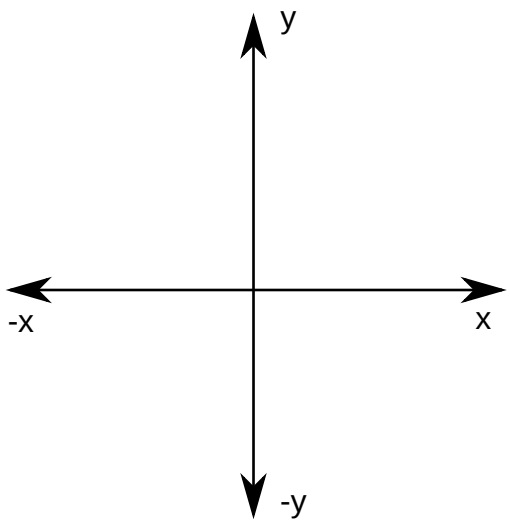
Koordináta transzformációk

- **Nézeti (Viewing)**
  a néző (kamera) helyének a megadása
- **Modell (Modeling)**
  az objektumok (modell) mozgatása
- **Modell-nézet (ModelView)**
  a nézeti és a modell transzformációk együtt
- **Vetítés (Projection)**
  a nézet vágása és látótérbe méretezése
- **Ablak**
  az eredmény ablakra való leképezése

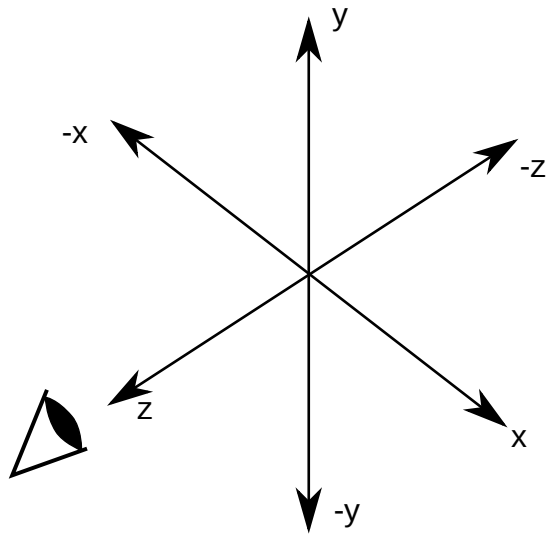
Nézeti koordináták

- A megfigyelő nézőpontja kezdetben $(0, 0, 0)$.
- A megfigyelő a z tengely negatív irányába néz.

Virtuálisan rögzített koordináta rendszer



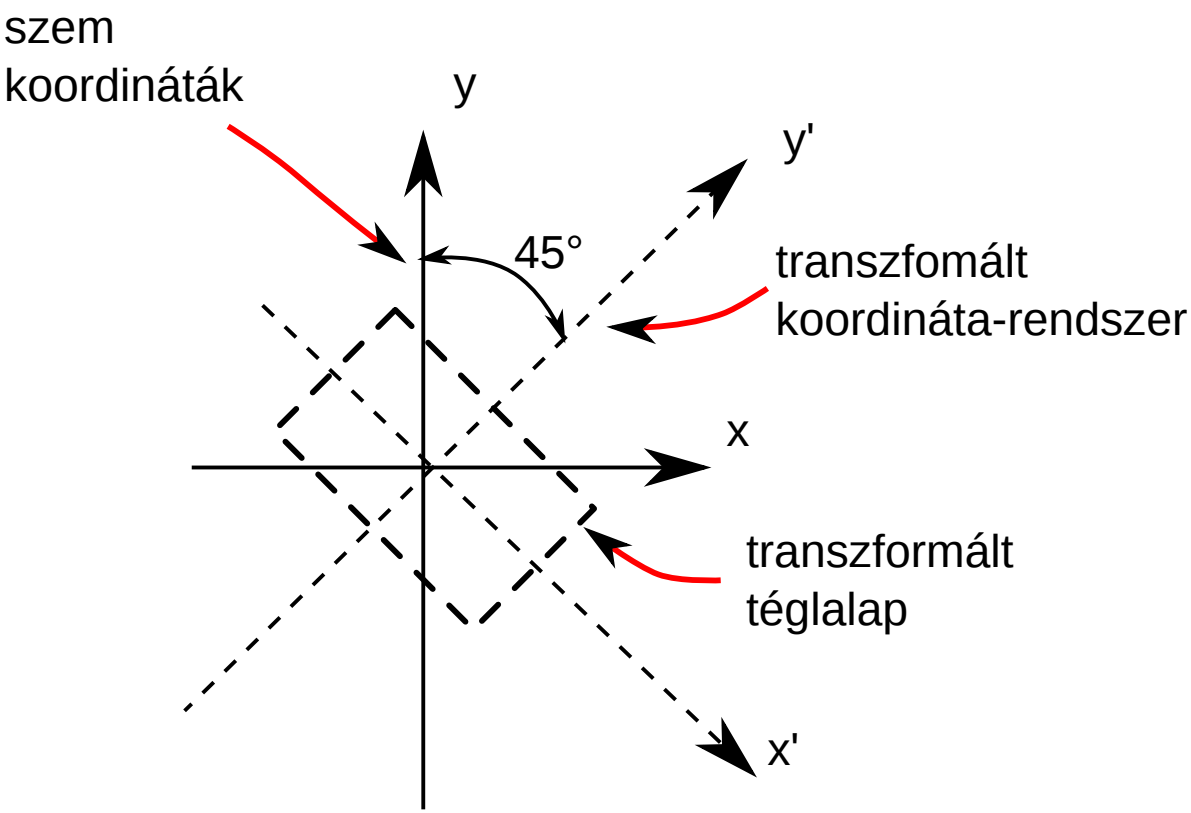
Ahogy a megfigyelő látja a modellt



Így látnánk oldalról a megfigyelőt a pozíciójának a z tengely irányába történő elmozdítása után

Nézeti koordináták

A nézeti koordináta rendszer elforgatása 45°-kal az óramutató járásával megegyező irányban



Nézeti (Viewing) transzformáció

Ez hajtódik végre először, ezt kell legelőször definiálni

Nézőpont meghatározása:

- Kezdeti nézőpont $(0, 0, 0)$
- `gluLookAt` paranccsal módosítható

Nézeti (Viewing) transzformáció

```
void gluLookAt(
    GLdouble eyex,
    GLdouble eyey,
    GLdouble eyez,
    GLdouble centerx,
    GLdouble centery,
    GLdouble centerz,
    GLdouble upx,
    GLdouble upy,
    GLdouble upz)
```

(eyex, eyey, eyez) a szem pozíciója
(centerx, centery, centerz) referenciapont, ahová a szem néz
(upx, upy, upz) felfelé mutató vektor (VUP)

Pl.:

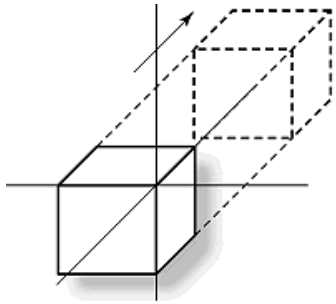
```
gluLookAt(0.0,0.0,2.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
```

Nézeti (Viewing) transzformáció

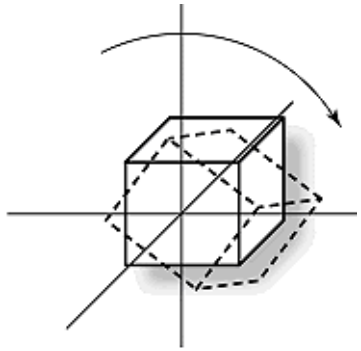
A `gluLookAt` eljárás kiszámítja a megadott nézeti transzformáció inverzét, majd az aktuális mátrixot megszorozza a kapott inverz transzformációs mátrixszal.

Az aktuális mátrix mód a `GL_MODELVIEW` legyen!

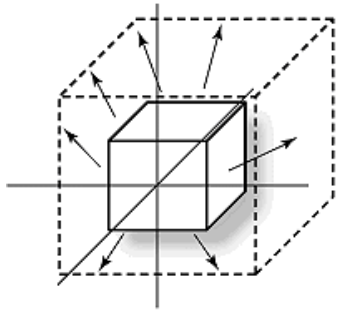
Modell (Modelling) transzformáció



eltolás



forgatás



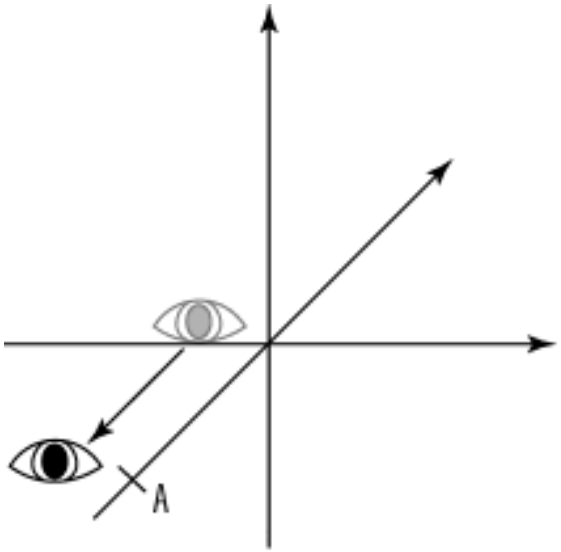
skálázás

A modell vagy egy részének a transzformálásra használjuk.

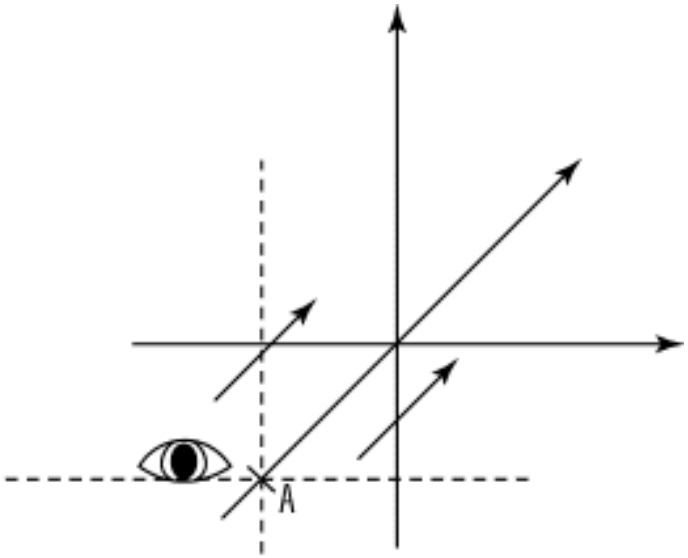
A csúcspont (vertex) koordinátákat transzformálja.

Modell-nézeti dualitás

A nézeti és a modell transzformációk duálisak, ezért elegendő csak a modell koordináta rendszert transzformálni.



nézeti koordináta rendszer mozgatás



modell koordináta rendszer mozgatás

Modell transzformációk

```
glMatrixMode(GLMODELVIEW);
void glRotate{fd}(T a, T x, T y, T z);
```

- `a`: forgatás fokban
- `(x,y,z)`: forgatási tengely
- pl. 45 fokos forgatás az x-tengely körül:
  `glRotated(45, 1.0, 0.0, 0.0, 0.0)`

```
void glTranslate{fd}(T x, T y, T z);
```

- `(x,y,z)`: az eltolás vektora
- pl. x-tengely mentén 50 egységgel való eltolás
  `glTranslated(50, 0,0);`

```
void glScale{fd}(T x, T y, T z);
```

- `(x,y,z)` skálázás mértéke a tengelyek mentén
- pl. `glScaled(0.5, 0.5, 0.5)`
  0.5-szörös uniform nagyítás