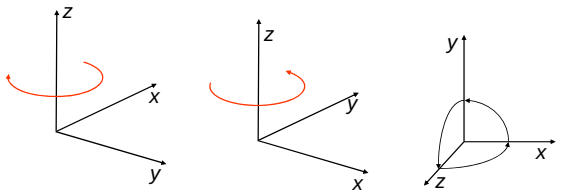
3D koordináta-rendszerek



balkezes
bal-sodrású

jobbkezes
jobb-sodrású

1

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'* = *α*·*x, ***y'* = *α*·*y***, ***z'* = *α*·*z*** és ***w'* = *α*·*w***

Ha ***w*≠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 a homogén koordináta

2

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

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

3

3D skálázás (nagyítás/kicsinyítés)

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

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

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

4

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

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

5

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

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

6

3D kompozíció-mártix

A legáltalánosabb kompozíció alakja:

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

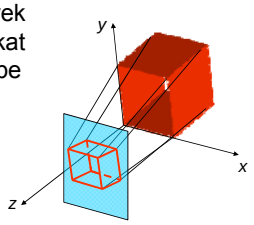
MP kiszámításához a mátrix szorzáshoz képest most is meg lehet takarítani műveleteket.

7

VETÍTÉSEK

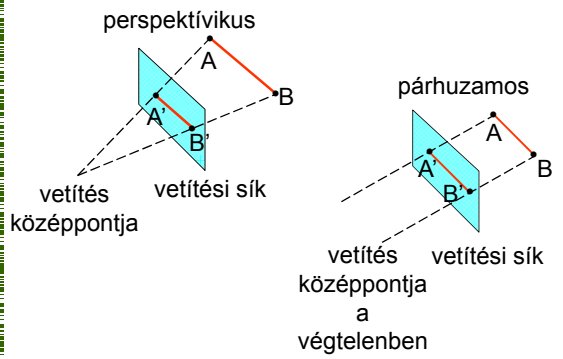
Transzformációk, amelyek *n*-dimenziós objektumokat kisebb dimenziós terekbe visznek át.

Pl. 3D→2D



8

Vetítések fajtái /1



9

Vetítések fajtái /2

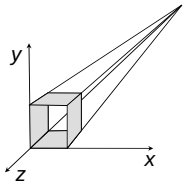
Perspektívikus	**Párhuzamos**
Vetítési középpont	**Vetítési irány**
közel áll látásunkhoz	kevésbé realisztikus
általában: nem mérhetők a távolságok (rövidülés) és a szögek	mérhető távolságok, szögek változnak

10

Perspektív vetítések /1

A vetítési síkkal nem, de egymással párhuzamos egyenesek vetületei egy pontban metszik egymást = <u>távlatpont</u>

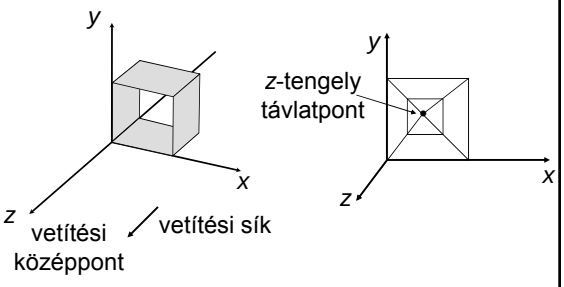
Elsődleges távlatpont: valamelyik fő(tengely) irányhoz tartozó távlatpont



11

Perspektív vetítések /2

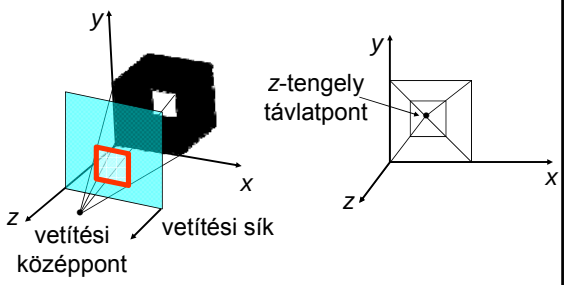
1. Egypontos perspektív vetítés



12

Perspektív vetítések /2

1. Egypontos perspektív vetítés



13

Párhuzamos vetítések

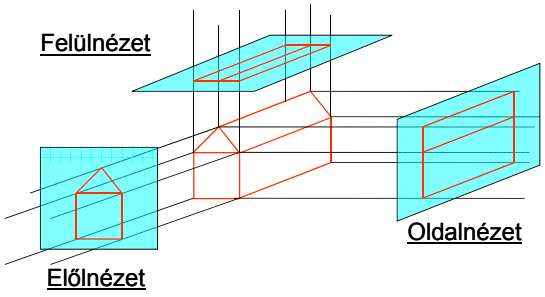
A párhuzamosság megmarad

Osztályozásuk a vetítési irány és a vetítési sík egymáshoz viszonyított helyzete szerint:

1. Merőleges (ortografikus)
2. Tetszőleges irányú

14

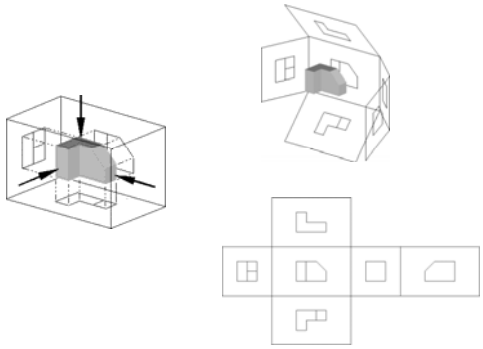
Párhuzamos (ortografikus) /1



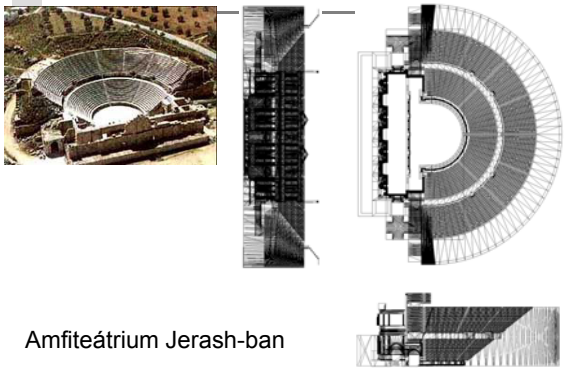
A párhuzamosság megmarad,
a távolságok megmaradnak vagy számíthatók

15

Példa



16



Amfiteátrium Jerash-ban

17

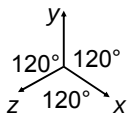
Párhuzamos /2

Axonometrikus (nem merőleges egyik tengelyre sem); szög nem marad meg, távolság számítható

Izometrikus (a vetítési irány (d_x, d_y, d_z) minden tengellyel ugyanakkora szöget zár be), azaz

$|d_x| = |d_y| = |d_z|$,

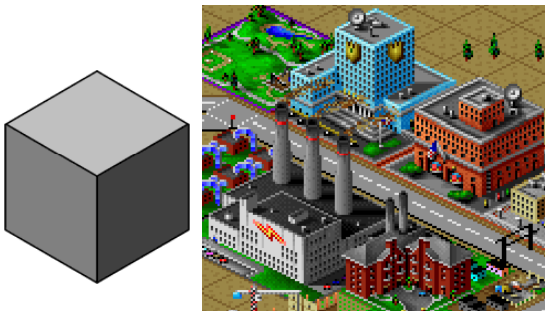
$\pm d_x = \pm d_y = \pm d_z$



8 ilyen irány létezik

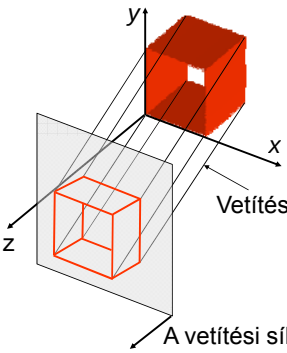
18

Izometrikus vetítés



19

Tetszőleges irányú /1



A vetítési sík normálisa és a vetítési irány nem párhuzamos

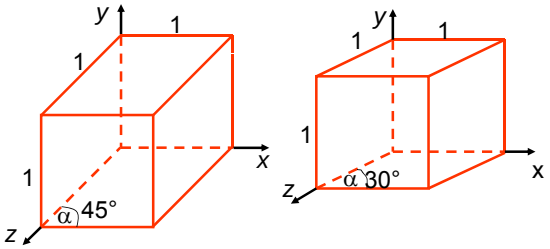
Leggyakoribb fajtái:
- kavalier
- kabinet

20

Tetszőleges irányú /2

Kavalier:

a vetítési irány és a vetítési sík szöge = 45°

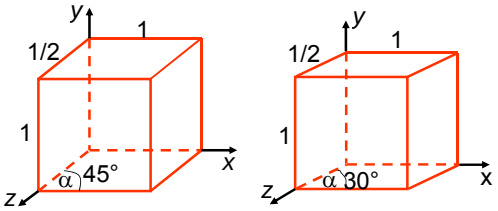


21

Tetszőleges irányú /3

Kabinet:

a vetítési irány és a vetítési sík szöge = arctg(2) = 63,4°



22

3-D megjelenítés specifikálása /1

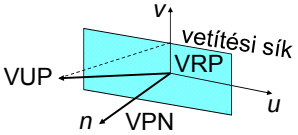
Szükséges: - látótér } megadása
- vetítés }

Vetítési sík megadása:

egy pontjával - referencia pont (VRP)

és a normálisával (VPN)

v a fölfelé mutató vektor (VUP) vetülete irányába mutat



23

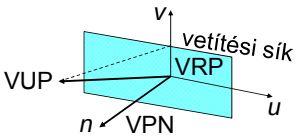
3-D megjelenítés specifikálása /2

3D referencia koordináta rendszer (VRC)

megadása:

Origó = VRP

A tengelyek:



n = VPN,

v = VUP-nek a vetítési síkra eső vetülete

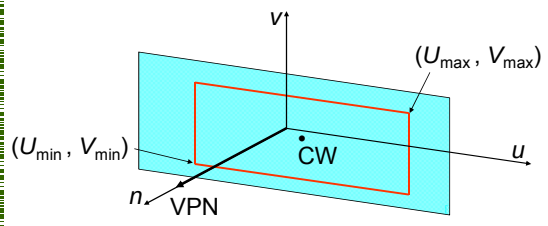
u = olyan, hogy *u, v, n* jobb-kezes derékszögű koordináta rendszert határozzon meg

24

3-D megjelenítés specifikálása /3

Ablak: Téglalap a vetítési síkon. Ami azon belül van, az megjelenik, a többi nem

CW a közepe



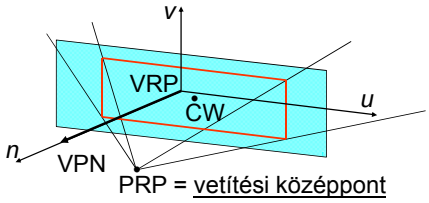
25

3-D megjelenítés specifikálása /4

PRP: vetítési referencia pont:

(párhuzamos és perspektív vetítésre is)

Perspektívikus vetítésnél

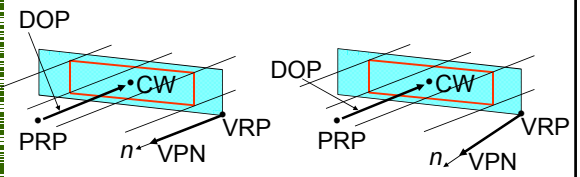


PRP = vetítési középpont

26

3-D megjelenítés specifikálása /5

DOP: vetítési irány



DOP || VPN DOP ⫲ VPN

27

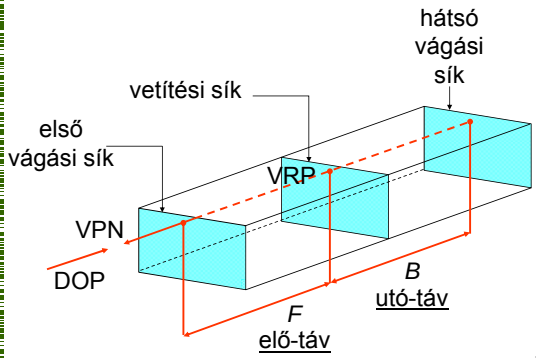
Látótér maghatározása első és hátsó vágási síkokkal /1

Fajtái:

- párhuzamos (ortografikus)
- párhuzamos (tetszőleges irányú)
- perspektivikus

28

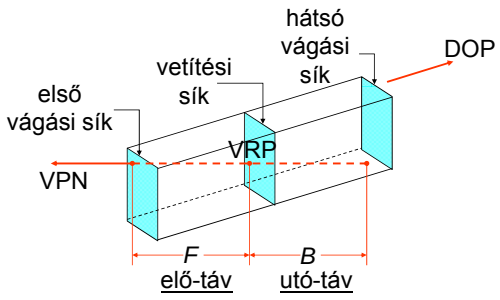
Látótér maghatározása első és hátsó vágási síkokkal /2



29

Látótér maghatározása első és hátsó vágási síkokkal /3

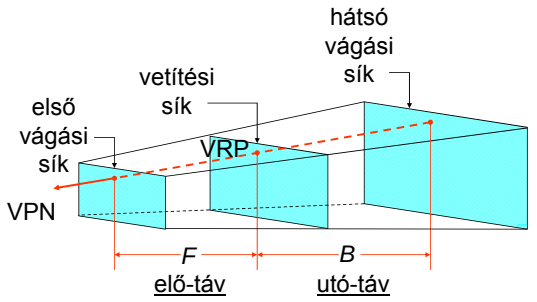
Párhuzamos (tetszőleges irányú):



30

Látótér maghatározása első és hátsó vágási síkokkal /4

Perspektivikus:



31

Vetítések matematikai leírása

32

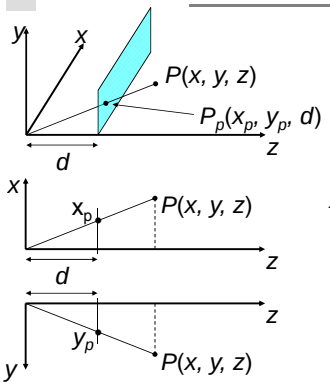
Perspektivikus vetítések /1

Az egyszerűség kedvéért tegyük fel hogy:

a) A vetítési sík merőleges a *z*-tengelyre
z = *d*-nél,
PRP = 0

33

Perspektivikus vetítések /2



Hasonló háromszögekből:

$$\frac{x_p}{d}=\frac{x}{z},\quad \frac{y_p}{d}=\frac{y}{z},\quad x_p=\frac{x}{z/d},\quad y_p=\frac{y}{z/d},\quad z\neq 0$$

vetületi síkok

34

Perspektivikus vetítések /3

Hasonló háromszögekből: $\frac{x_p}{d}=\frac{x}{z},\quad \frac{y_p}{d}=\frac{y}{z},\quad x_p=\frac{x}{z/d},\quad y_p=\frac{y}{z/d},\quad z\neq 0,$

$$\begin{pmatrix} \frac{x}{z/d} \\ \frac{y}{z/d} \\ d \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix} = P_p,$$ mivel ez homogén koordináta,

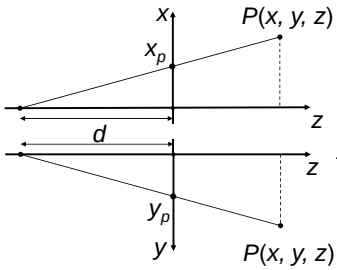
tehát $M_{per}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix}$, mert $M_{per}\cdot\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}=\begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix}$

35

Perspektivikus vetítések /4

Más lehetőség

b) A vetítési sík merőleges a *z*-tengelyre *z* = *0*-ban



$$\frac{x_p}{d}=\frac{x}{z+d},\quad \frac{y_p}{d}=\frac{y}{z+d},\quad x_p=\frac{x}{(z/d)+1},\quad y_p=\frac{y}{(z/d)+1},$$

36

Perspektivikus vetítések /5

$$\frac{x_p}{d}=\frac{x}{z+d},\quad \frac{y_p}{d}=\frac{y}{z+d},\quad x_p=\frac{x}{(z/d)+1},\quad y_p=\frac{y}{(z/d)+1},$$

tehát $P'_{per}=\begin{pmatrix}\frac{x}{z/d+1}\\ \frac{y}{z/d+1}\\ 0\\ 1\end{pmatrix}=\begin{pmatrix}x\\ y\\ 0\\ \frac{1}{z/d+1}\end{pmatrix}$

ezért $M'_{per}=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&1/d&1\end{pmatrix}$

37

Párhuzamos orthografikus vetítés

$$P=\begin{pmatrix}x\\y\\z\\1\end{pmatrix}\xrightarrow{M_{ort}}\begin{pmatrix}x_p\\y_p\\z_p\\1\end{pmatrix}=\begin{pmatrix}x\\y\\0\\1\end{pmatrix}=P_p,$$

ahol

$M_{ort}=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}$ (határértéke M'_{per}-nek, $d\to\infty$).

38

3D megjelenítés implementálása /1

3D objektumok VK-ban → (normalizálás) → 3D objektumok kanonikus látótérben, 3D vágás → (vetítés) → 2D vetített objektumok → (transzformáció) → 2D megjelenített objektum

39

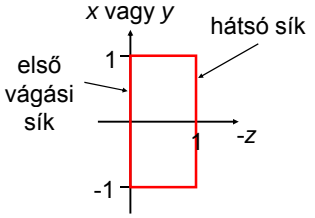
3D megjelenítés implementálása /2

A 3D vágás drága művelet, ezért érdemes előtte a 3D objektumokat u.n. kanonikus látótérbe transzformálni (normalizálás), ahol a vágás egyszerűbb és gyorsabb.

40

3D megjelenítés implementálása /3

párhuzamos vetítésnél



Kanonikus látótér vágási síkjai:

$x = -1,$	$x = 1$
$y = -1,$	$y = 1$
$z = 0,$	$z = -1$

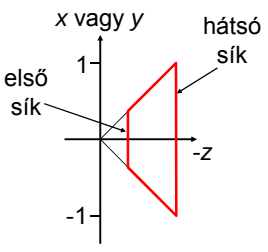
41

3D megjelenítés implementálása /4

perspektív vetítésnél

Kanonikus látótér vágási síkjai:

$x = z,$	$x = -z$
$y = z,$	$y = -z$
$z = -z_{min},$	$z = -1$



42

Mátrix műveletek (OpenGL)

Az OpenGL oszlopfolytonosan tárolja a mátrixokat

Az egység mátrix:

```
GLfloat M[] = {
   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}
```

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

Új aktuális mátrix betöltése:

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

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

43

Mátrix műveletek (OpenGL)

Az aktuális mátrix legyen az egységmátrix:

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

Az aktuális mátrix szorzása:

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

Pl.:

```
GLfloat M[] = {
          1.0,  0.0, 0.0, 0.0,
          0.0, 10.0, 0.0, 0.0,
          0.0,  0.0, 1.0, 0.0,
          0.0,  0.0, 0.0, 1.0}
glMatrixMode(GL_MODELVIEW);
glMultMatrix(M);
```

A szorzat lesz az új aktuális mátrix

44

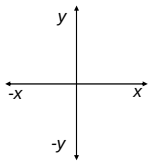
Koordináta transzformációk (OpenGL)

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

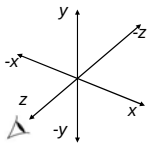
45

Nézeti koordináták (OpenGL)

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



Ahogy a megfigyelő látja a modellt



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

46

Nézeti (Viewing) transzformáció (OpenGL)

Ez a transzformáció hajtódik végre először, ezt kell legelőször definiálni

Nézőpont meghatározása

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

47

Nézeti (Viewing) transzformáció (OpenGL)

```
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 (up-vektor, VUP)

Pl.:

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

48

Nézeti (Viewing) transzformáció (OpenGL)

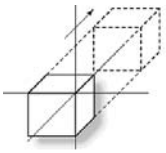
A **`gluLookAt`** eljárás kiszámítja a megadott nézeti transzformáció inverzét, majd az aktuális mátrixot megszorozza a kapott inverz transzformációs mátrixszal.

Szokásos használata:

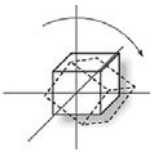
```
glMatrixMode(GL_MODELVIEW);
glLoad Identity();
gluLookAt(x,y,z,tx,ty,tz,0.0f,1.0f,0.0f);
```

49

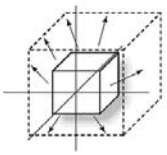
Modell (Modeling) transzformáció (OpenGL)



eltolás (transzláció)



forgatás (rotáció)



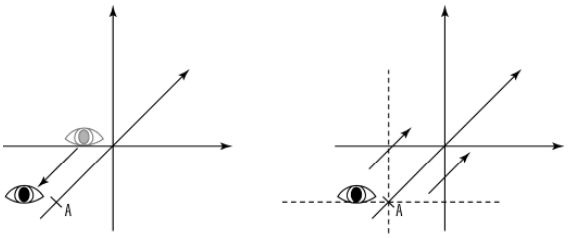
skálázás

A modell vagy egy részének a transzformálására használjuk

A csúcspont (vertex) koordinátákat kell csak transzformálni

50

Modell-nézeti dualitás (OpenGL)



nézeti koordináta rendszer mozgatás

modell koordináta rendszer mozgatás

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

51

Modell transzformációk (OpenGL)

```
glMatrixMode(GL_MODELVIEW);
```

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

a: forgatás fokban; **(*x*, *y*, *z*):** forgási tengely

pl. 45 fokos forgatás az *x*-tengely körül:

```
glRotated(45, 1.0, 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);`**

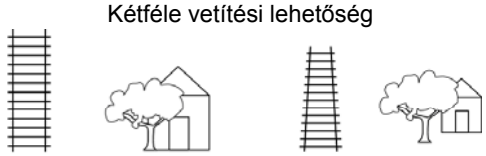
0.5-szörös uniform nagyítás

52

Vetítési (projection) transzformáció (OpenGL)

```
glMatrixMode(GL_PROJECTION);
```

Kétféle vetítési lehetőség



orthografikus és perspektivikus

Megadjuk a látóteret is

Végrehajtás:

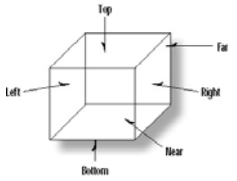
új projekciómátrix =
projekciómátrix · specifikált mátrix

53

Orthografikus vetítés (OpenGL)

```
void glOrtho(double left, double right,
             double bottom, double top,
             double near, double far);
```

Orthogonális (ortografikus) vetítés vágási terének megadása



2D eset:

```
void gluOrtho2D(
        double left, double right,
        double bottom, double top);
```

54

Perspektív vetítés (OpenGL)

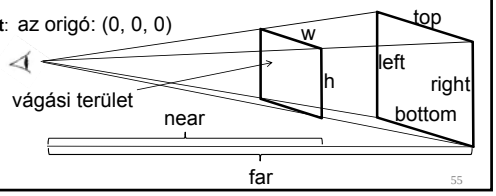
```
void glFrustum (double left, double right,
                double bottom, double top,
                double znear, double zfar);
```

left, right: a bal és jobb oldali vágósík *x* koordinátája

bottom, top: az alsó és felső vágósík *y* koordinátája

znear, zfar: a közeli és távoli vágósík *z* koordinátája.

Nézőpont: az origó: (0, 0, 0)



55

Perspektív vetítés (OpenGL)

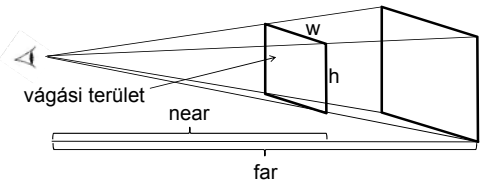
Szimmetrikus látótér megadása:

```
void gluPerspective (double fovy,
 double aspect, double near, double far);
```

fovy: a látótér szöge y irányban

aspect: w/h

near, far: a vágósíkok távolsága a megfigyelőtől



56

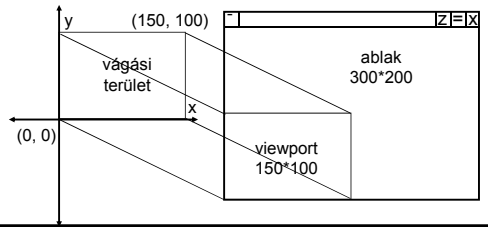
Ablak (OpenGL)

2D-s leképzés az ablak egy téglalap alakú (viewport) részébe:

```
void glViewport(GLint x, GLint y,
     GLsizei width, GLsizei height);
```

x, y: a viewport bal alsó sarka az ablakban,

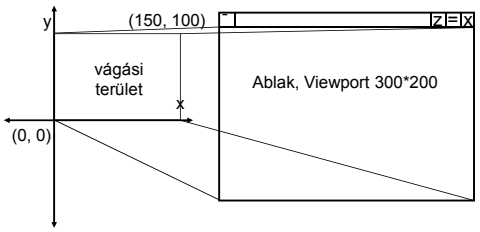
width, height: a viewport mérete pixelben



57

Ablak (OpenGL)

Alapértelmezés: `(0, 0, winWidth, winHeight)`, ahol **`winWidth`** és **`winHeight`** az ablak méretei



58

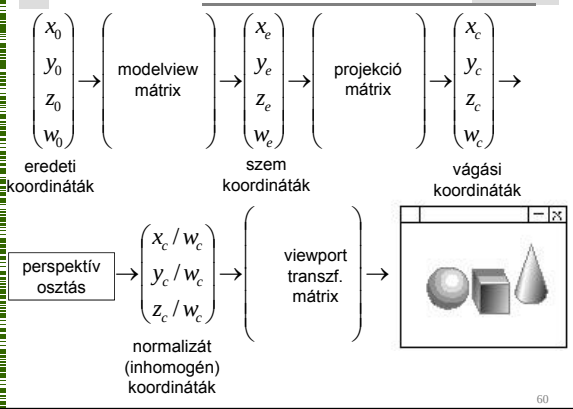
Perspektív vetítés (OpenGL)

Pl.: Módosítsuk viewport-ot és a vágási teret perspektív vetítésnél

```
void ChangeSize(GLsizei w, GLsizei h){
     GLfloat fAspect;
     if(h == 0) h = 1; // ne ossszunk 0-val
  // az ablakon beállítjuk a viewport-ot
     glViewport(0, 0, w, h);
     fAspect = (GLfloat)w/(GLfloat)h;
  // vetítési mátrix
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
  // vágási tér megadás, perspektív vetítés
     gluPerspective(60.0f, fAspect,
          1.0, 400.0);
     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity();
}
```

59

Transzformációs mátrixok (OpenGL)



60

Mátrix vermek (OpenGL)

Mátrix módok: **`GL_PROJECTION, GL_MODELVIEW, GL_TEXTURE`**

Minden mátrix mód számára van egy mátrix verem

Az aktuális mátrix a verem tetején lévő mátrix.
A műveletek:

```
void glPushMatrix(void);
void glPopMatrix(void);
```

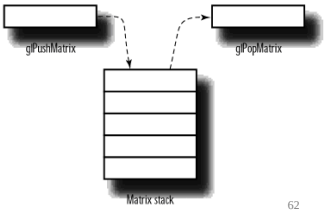
61

Mátrix vermek (OpenGL)

`glGet(GL_MAX_MODELVIEW_STACK_DEPTH)`
(Microsoft: maximális mélység 32)

`glGet(GL_MAX_PROJECTION_STACK_DEPTH)`
(Microsoft: maximális mélység 2)

`GL_STACK_OVERFLOW, GL_STACK_UNDERFLOW`



Alapállapot:
egységmátrix,
`GL_MODELVIEW`

62

Feladat (OpenGL)

Rajzoljuk meg egy kocka perspektivikus képét!

63