PPKE ITK

A számítógépes grafika alapjai

A geometriai transzformációk megvalósítása OpenGL-ben

Előadó: Benedek Csaba

Tananyag : Szirmay-Kalos László, Benedek Csaba

Koordináta-rendszer transzformációk

- Analógia a műtermi fényképkészítés és az OpenGL képalkotás között
  - Fényképezőgép elhelyezése, ráirányítása a lefényképezendő térrészre - nézőpontba transzformálás
  - A lefényképezendő objektumok elhelyezése a kívánt pozícióba –modell-transzformáció
  - A megfelelő lencse kiválasztása, zoom beállítása – vetítési transzformáció
  - Papírkép a negatívról – képmező transzformálás

OpenGL transzformációk

- Homogén lineáris transzformáció (p itt oszlopvektor)

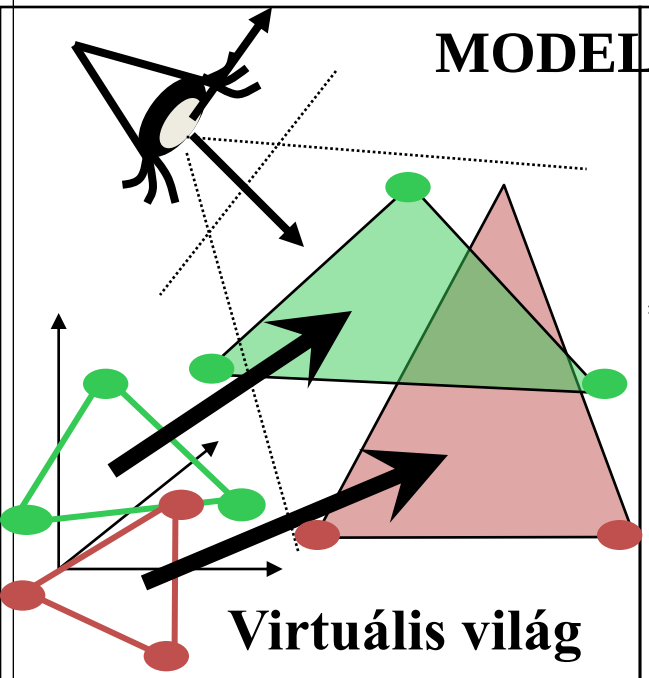
$$\mathbf{p'=Mp}$$

- Transzformációk kompozíciója

$$\mathbf{p'=M_1p \text{ és } p''= M_2p' \rightarrow p''=M_2M_1p}$$

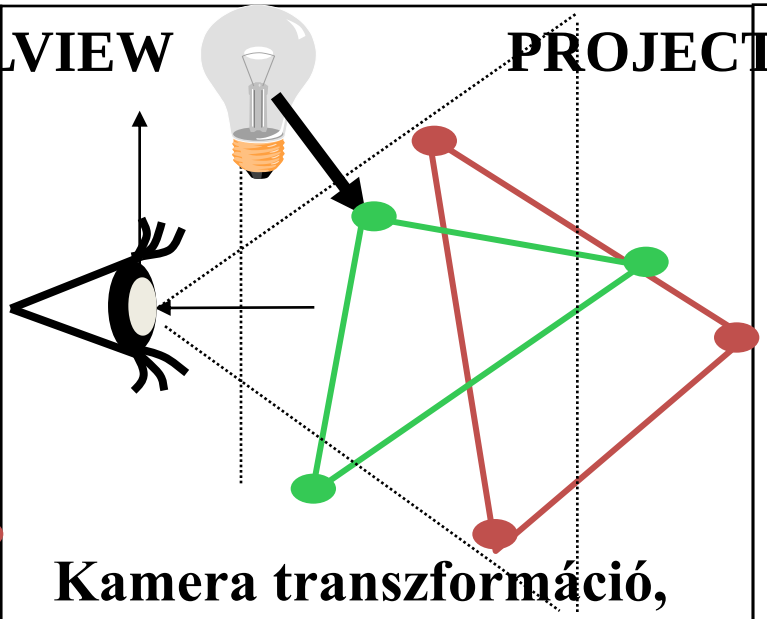
  - a közös transzformáció M_2M_1

OpenGL csővezeték

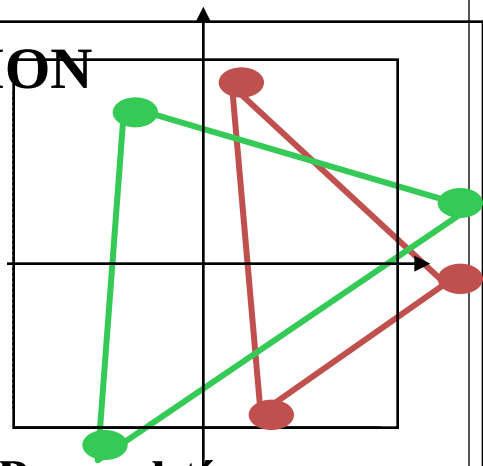


MODELVIEW

Virtuális világ

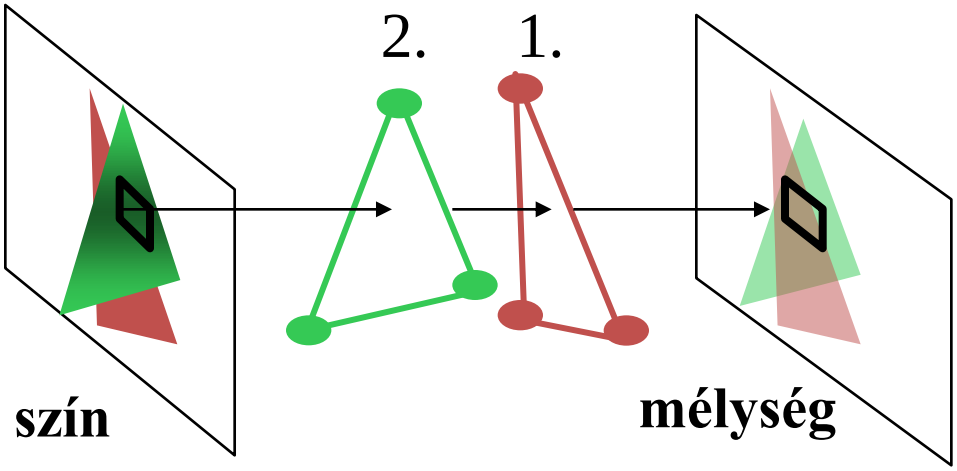


Kamera transzformáció, illumináció

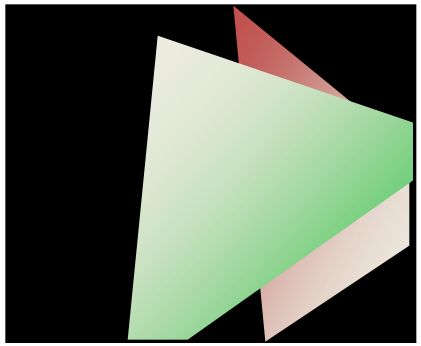


PROJECTION

Perspektív transzformáció + Vágás homogén koordinátákban

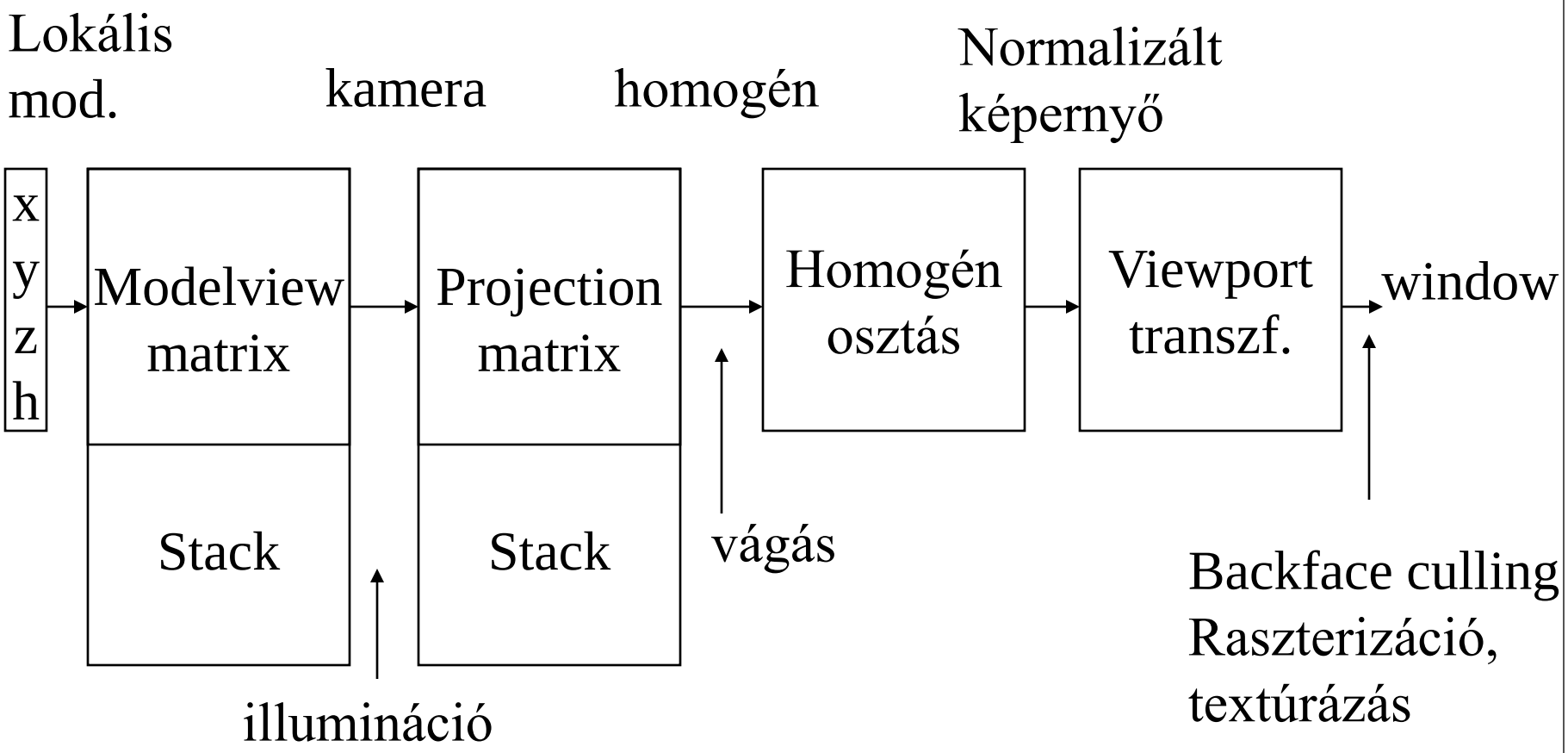


Képernyő transzf+Raszterizáció+interpoláció



megjelenítés

Transzformációk



OpenGL transzformációs mátrixok kezelése

- Műveletek: aktuális mátrix *kijelölése*, *betöltése*, vagy *megszorzása* **jobbról** egy új mátrixszal

OpenGL transzformációs mátrixok kezelése

- `void` **`glMatrixMode(GLenum`** ***`mode`*****`);`**
  - Állapotváltozó: megadjuk, hogy melyik mátrixot akarjuk a következő parancsokkal állítani
  - ***mode*** lehet:
    - GL_MODELVIEW – modell-nézeti transzformáció (objektumok és a kamera elhelyezkedése és iránya)
    - GL_PROJECTION – kamera modell kiválasztás és fókuszálás
    - GL_TEXTURE – textúra
- **`glLoadIdentity();`**
  - aktuális mátrix beállítása az egységmátrixra

Mátrix betöltése, szabad szorzása

- `void` **`glLoadMatrix{fd}(const TYPE *m);`**
  - m pointer által címzett 16 elemű vektor elemeit betölti az aktuális mátrixba **oszlopfolytonosan**

$$M: \begin{bmatrix} m1 & m5 & m9 & m13 \\ m2 & m6 & m10 & m14 \\ m3 & m7 & m11 & m15 \\ m4 & m8 & m12 & m16 \end{bmatrix}$$

- `void` **`glMultMatrix{fd}(const TYPE *m);`**
  - m pointer által címzett 16 elemű vektor elemeiből képzett mátrixszal szorozza az aktuális mátrixot

OpenGL transzformációs mátrixok kezelése

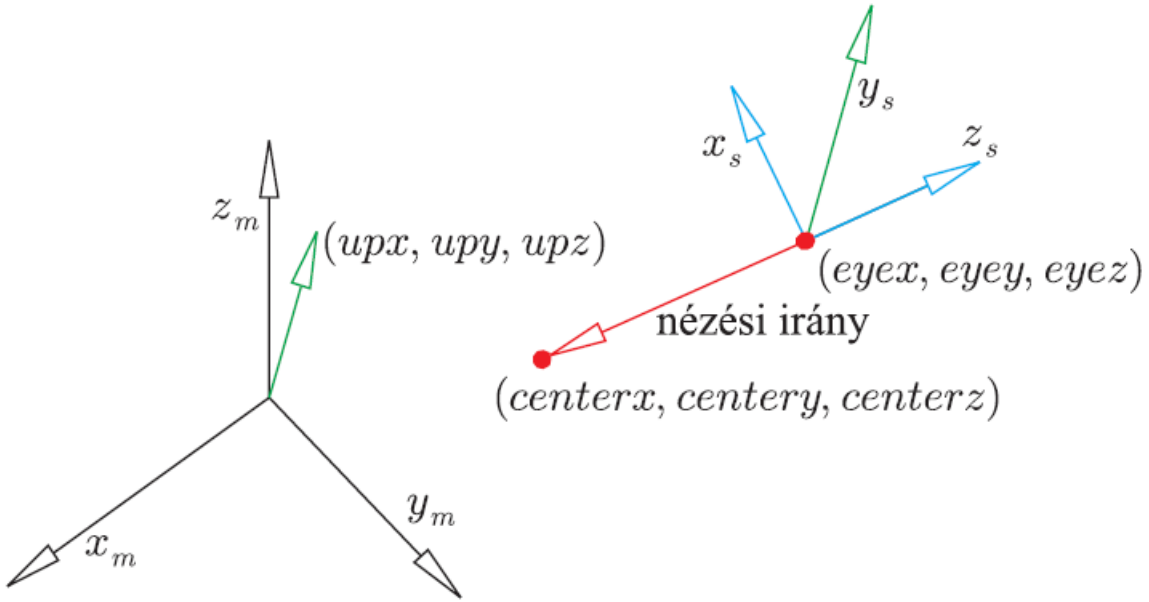
- `void` **`glTranslate{fd}`**`(TYPE x, TYPE y, TYPE z)`
  - előállítja az $[x,y,z,1]^T$ vektorral való eltolás mátrixát és megszorozza vele (jobbról) a kurrens mátrixot
- `void` **`glRotate{fd}`**`(TYPE angle, TYPE x, TYPE y, TYPE z)`
  - előállítja az origón áthaladó, $[x,y,z]^T$ irányvektorú egyenes körül `angle` szöggel elforgató mátrixot és megszorozza vele a kurrens mátrixot. A szög előjeles, fokban kell megadni
- `void` **`glScale{fd}(`*`TYPE x, TYPE y, TYPE z`*`);`**
  - skálázás

Nézőpont és nézeti irány beállítása

- Alapértelmezés: nézőpont a modellkoordináta-rendszer origója, nézési irány a negatív z-tengely
- Új nézőpont megadása:
  - OpenGL: az objektumot toljuk/forgatjuk ellentétes irányban (glTranslate*, glRotate*)
  - GLU: a nézőpont és a kamera nézet iránya közvetlenül is megadható

Nézőpont és nézeti irány beállítása

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



OpenGL transzform. kompozíció

- Forgatás (R) **majd** eltolás (T) elvégzése

  p'=Rp és p''= Tp' → p''=TRp

  - a közös transzformáció TR:
  - OpenGL mátrixszorzás jobbról történik, ezért a mátrixokat a végrehajtás sorrendjével **ellentétes** sorrendben kell megadni:

    (1) lépés **`glLoadIdentity();`**

    M:=E egységmátrix

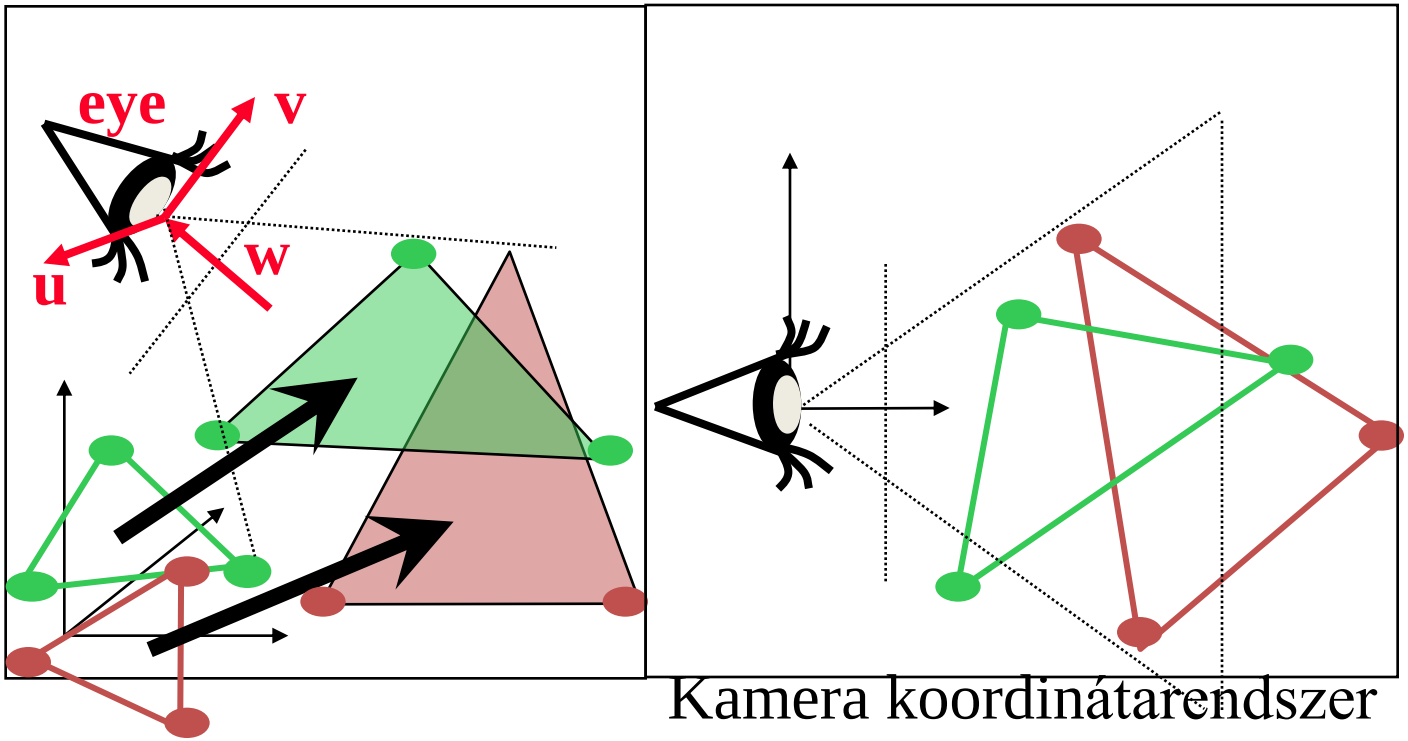
    (2) lépés **`glTranslated`**`(x,y,z);`

    M:=M·T=E·T = T

    (3) lépés **`glRotated`**`(angle,x,y,z);`

    M:=M·R=E·T·R = TR

MODELVIEW Transzformáció



Kamera koordinátarendszer

$$\begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ \text{eye} & & & 1 \end{pmatrix}^{-1}$$

```
glMatrixMode(GL_MODELVIEW);

glLoadIdentity( );
gluLookAt(eyex, eyey, eyez, vrpx, vrpy, vrpz,upx, upy, upz);  //VIEW

glTranslatef(px, py,pz);                                      //MODEL
glRotatef(ang, axisx,axisy,axisz);  glScalef(sx, sy, sz);

glMultMatrixf( mat[4][4] );
```

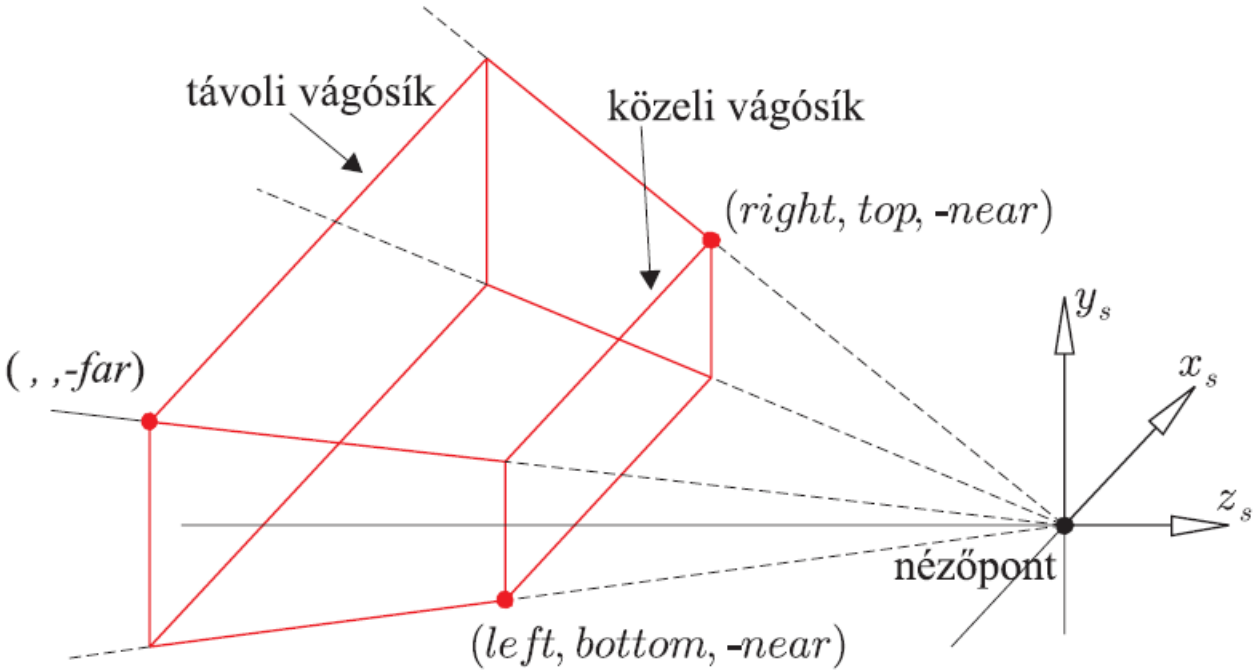
sorrend

OpenGL vetítési transzfromációk

- Centrális vetítés (valószerű képek):
  - **`glFrustum()`**
  - **`glPerspective()`**
- Merőleges vetítés (méret/méretarány helyes képek):
  - **`glOrtho()`**
  - **`glOrtho2D()`**

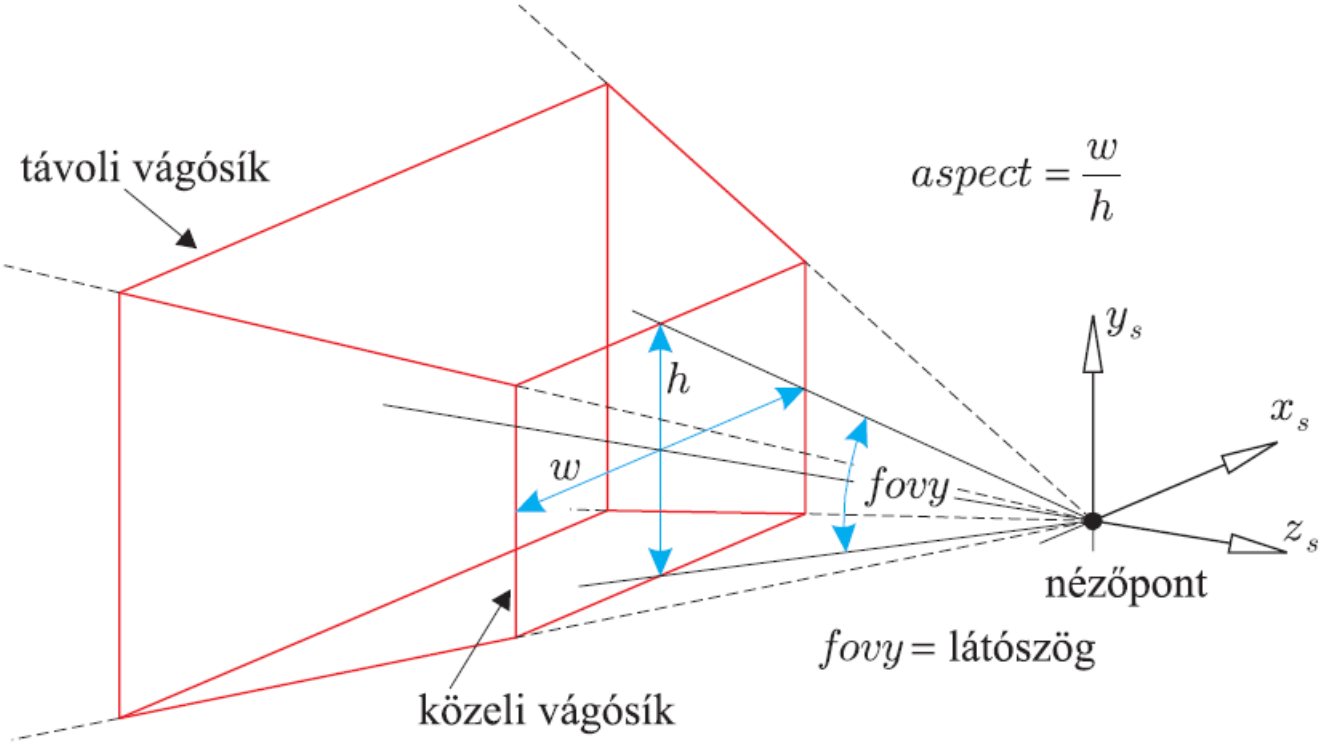
Projektív transzformáció

- `void` **`glFrustum`**`(GLdouble` *`left`*`, GLdouble` *`right`*`, GLdouble` *`bottom`*`, GLdouble` *`top`*`, GLdouble` *`near`*`, GLdouble` *`far`*`);`



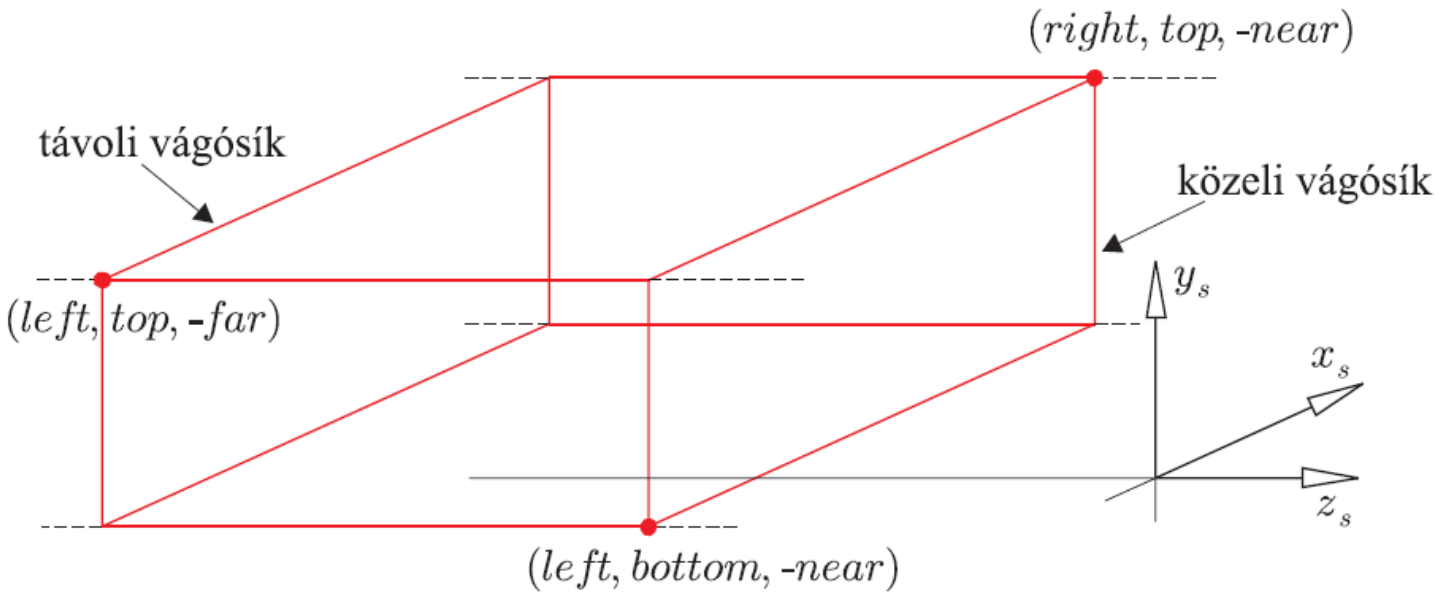
Projektív transzformáció

- `void` **`gluPerspective`**`(GLdouble` *`fovy`*`, GLdouble` *`aspect`*`, GLdouble` *`near`*`, GLdouble` *`far`*`);`



Merőleges vetítés

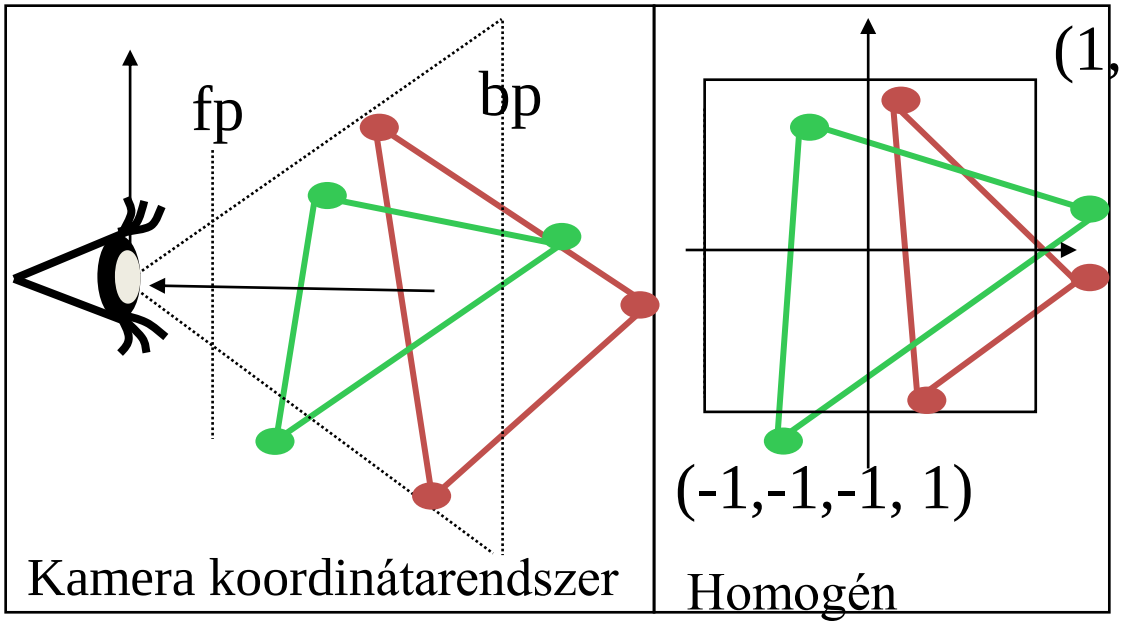
- `void` **`glOrtho`**`(GLdouble` *`left`*`, GLdouble` *`right`*`, GLdouble` *`bottom`*`, GLdouble` *`top`*`, GLdouble` *`near`*`, GLdouble` *`far`*`);`
  - a nézőpont helye itt közömbös, csak a nézési irány számít



2D alakzatok ábrázolása

- `void` **`gluOrtho2D`**`(GLdouble` *`left`*`, GLdouble` *`right`*`, GLdouble` *`bottom`*`, GLdouble` *`top`*`);`
  - 2D: nem kell közeli és távoli vágósíkot megadni (automatikusan [-1,1])

PROJECTION Transzformáció




```
glMatrixMode(GL_PROJECTION);
glLoadIdentity( );
```

gluPerspective(fov, asp, fp, bp);

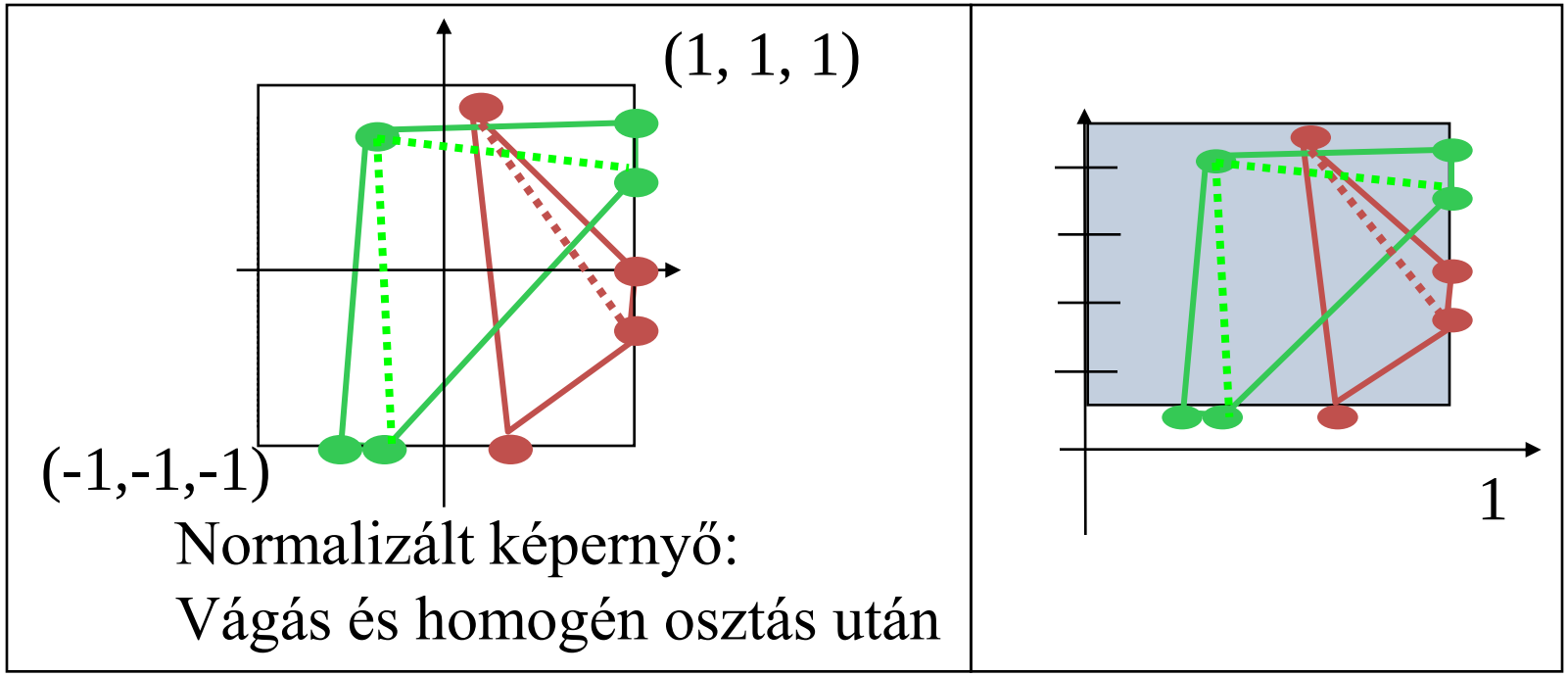
Projekció után homogén osztás: normalizált koordináták

$$\begin{pmatrix} 1/(tg(fov/2)\cdot asp) & 0 & 0 & 0 \\ 0 & 1/tg(fov/2) & 0 & 0 \\ 0 & 0 & -(fp+bp)/(bp-fp) & -1 \\ 0 & 0 & -2fp*bp/(bp-fp) & 0 \end{pmatrix}$$

Képmező transzformáció

- `void` **`glViewport`**`(GLint` *`x`*`, GLint` *`y`*`, GLsizei` *`width`*`, GLsizei` *`height`*`);`
  - képmező:
    - téglalap alakú rajzterület a képernyőn
    - oldalai párhuzamosak az ablak oldalaival
    - (x,y): a képmező bal alsó sarka
    - width, height: a képmező szélessége, hosszúsága

Képernyő Transzformáció



glViewport(0, 0, width, height);

Aktuális transzf. mentése és újra töltése

- `void` **`glPushMatrix(void);`**
  - A glMatrixMode() paranccsal beállított kurrens verem minden elemét egy szinttel lejjebb tolja. A legfelső (kurrens) mátrix a második mátrix másolata lesz
- `void` **`glPopMatrix(void);`**
  - A glMatrixMode() paranccsal beállított kurrens verem legfelsőbb elemét eldobja, és minden további elemet egy szinttel feljebb tol.