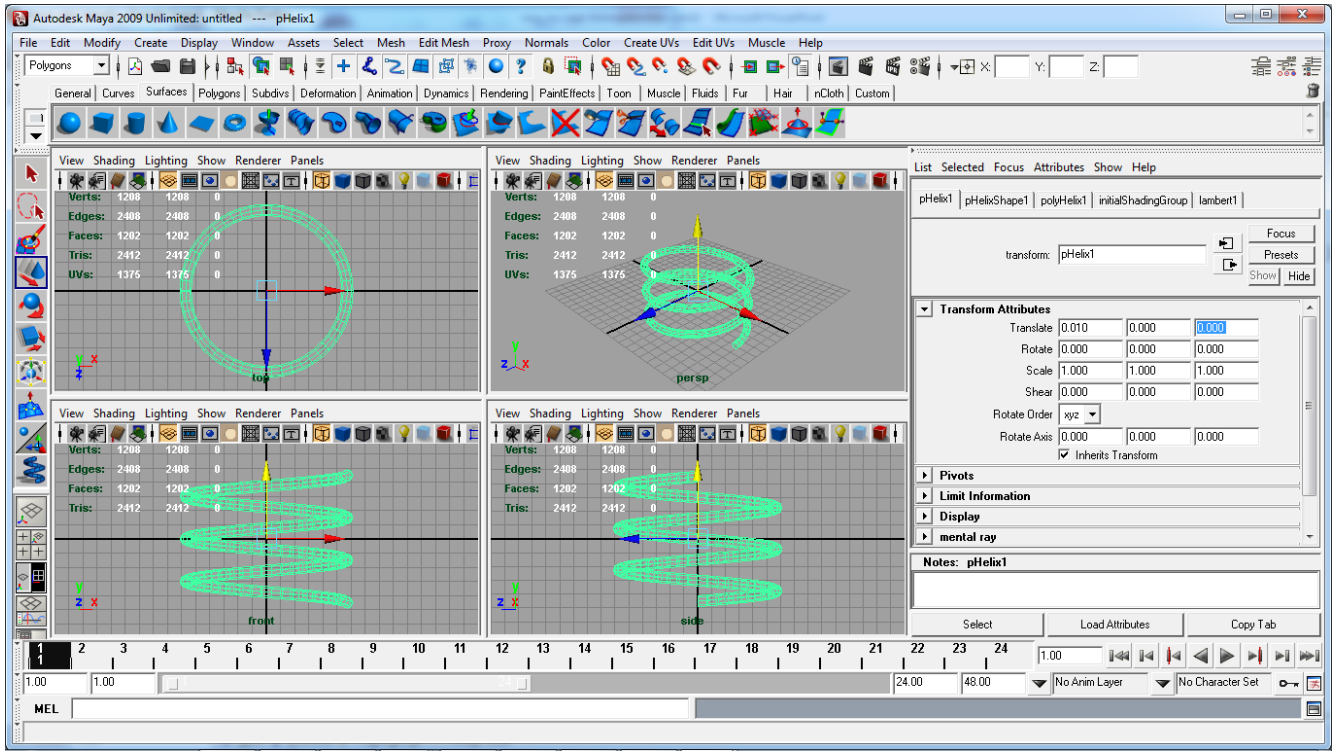
3D kamera

Szécsi László

3D Grafikus Rendszerek

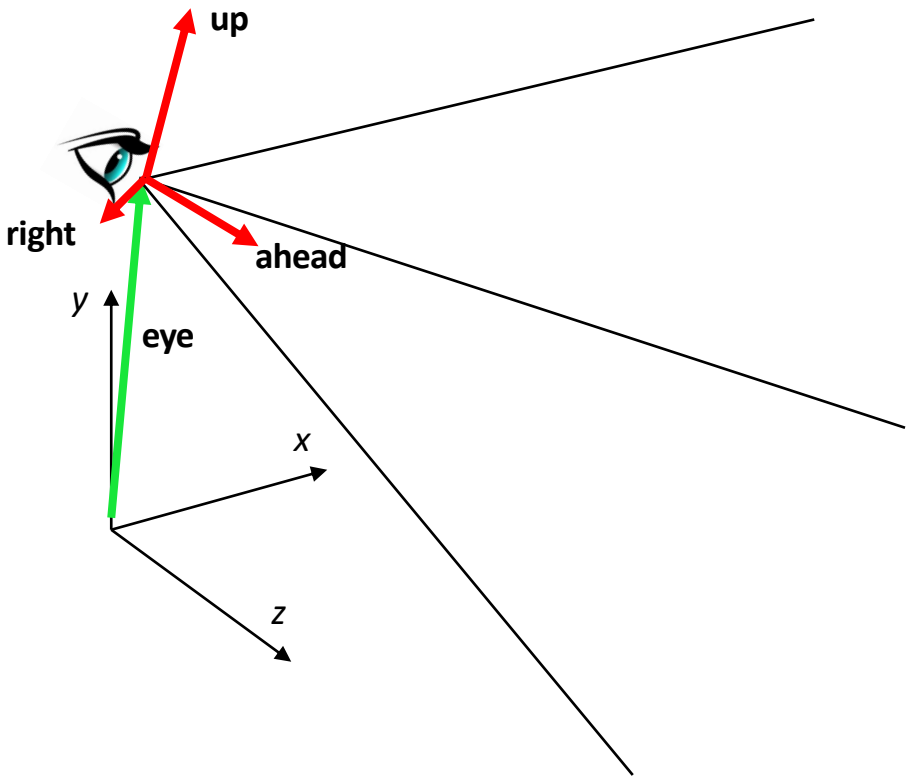
9. előadás

Modelltér

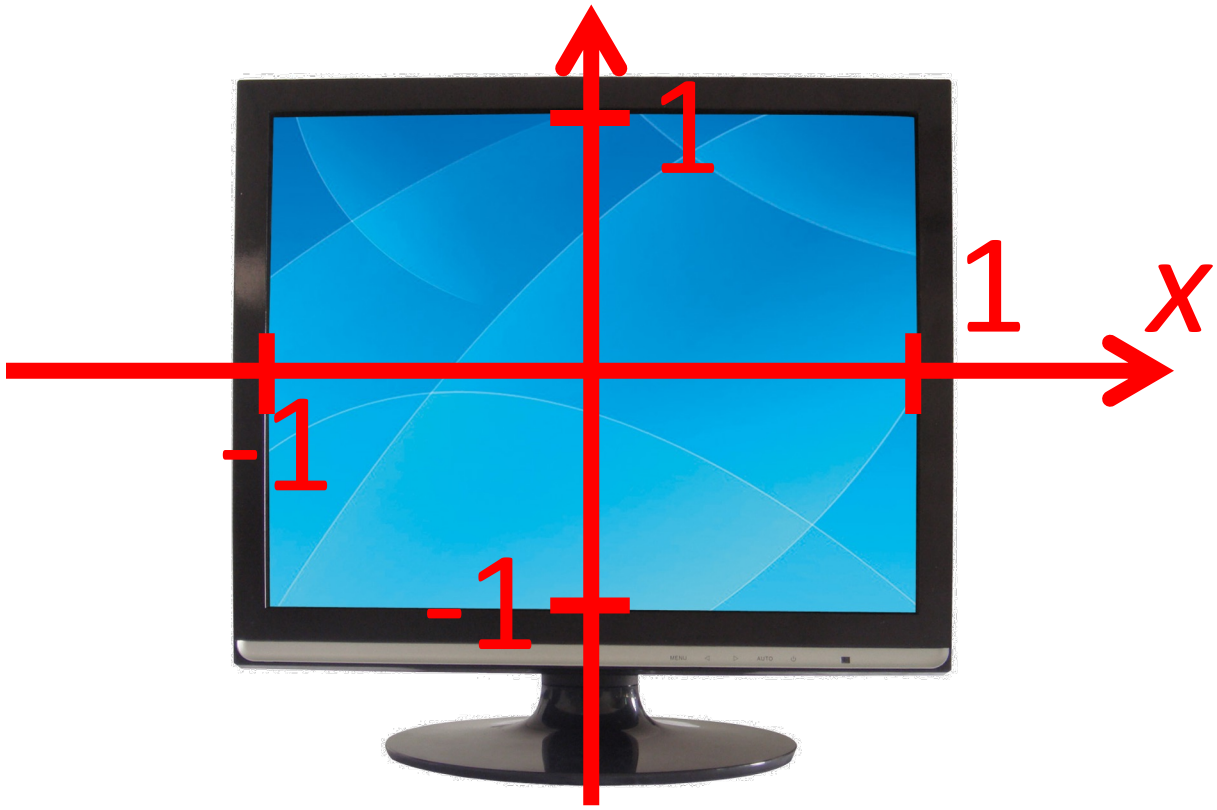

Autodesk Maya 2009 Unlimited: untitled --- pHelix1
File Edit Modify Create Display Window Assets Select Mesh Edit Mesh Proxy Normals Color Create UVs Edit UVs Muscle Help
General Curves Surfaces Polygons Subdivs Deformation Animation Dynamics Rendering PaintEffects Toon Muscle Fluids Fur Hair nCloth Custom
View Shading Lighting Show Renderer Panels
Verts: 1208 1208 0
Edges: 2408 2408 0
Faces: 1202 1202 0
Tris: 2412 2412 0
UVs: 1375 1375 0
top
persp
front
side
List Selected Focus Attributes Show Help
pHelix1 pHelixShape1 polyHelix1 initialShadingGroup lambert1
transform: pHelix1
Focus
Presets
Show Hide
Transform Attributes
Translate 0.010 0.000 0.000
Rotate 0.000 0.000 0.000
Scale 1.000 1.000 1.000
Shear 0.000 0.000 0.000
Rotate Order xyz
Rotate Axis 0.000 0.000 0.000
Inherits Transform
Pivots
Limit Information
Display
mental ray
Notes: pHelix1
Select
Load Attributes
Copy Tab
No Anim Layer
No Character Set
MEL

Világtér

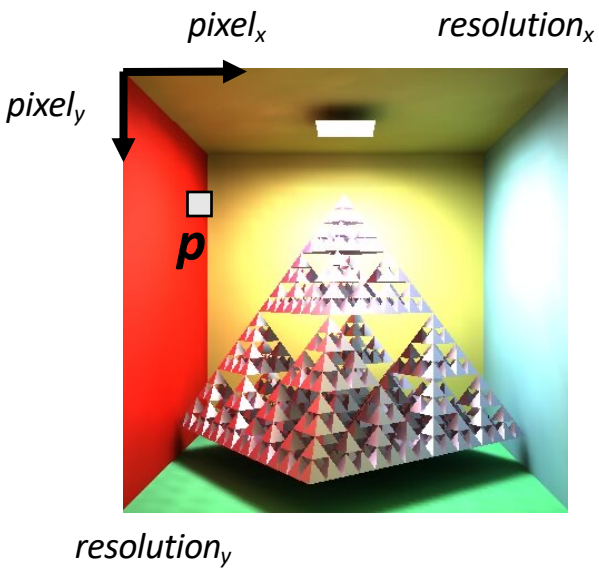
Kameratér



Eszköztér

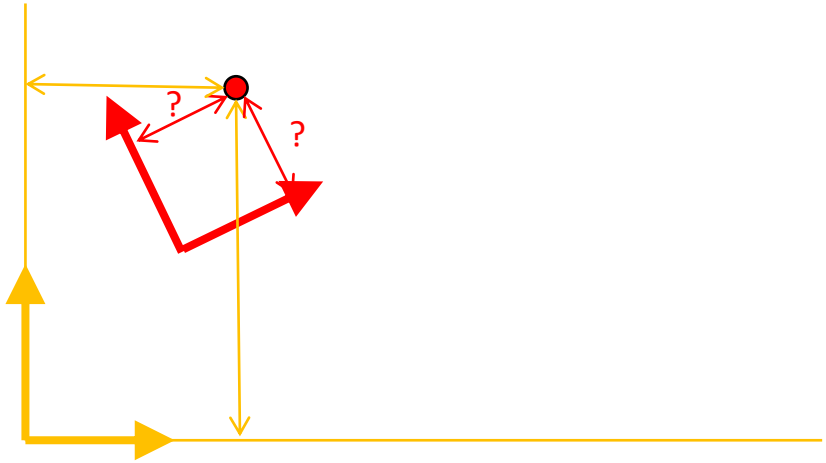


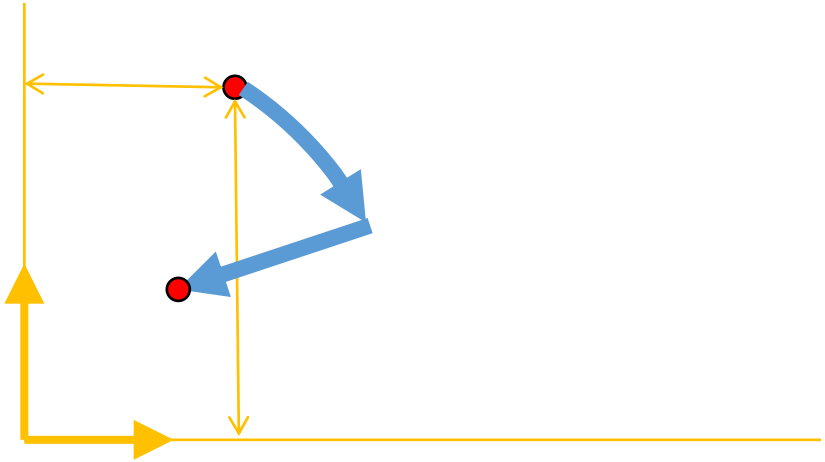
Nézetablak-tér



Transzformációk

- Ha egy térben adottak egy pont koordinátái, mik a koordinátái a másik térben?
  - statikus interpretáció
- Mit tegyek az eredeti ponttal, hogy a másik térbeli helyére kerüljön?
  - dinamikus interpretáció





Az inkrementális képalkotási probléma

- adott a tesszellált 3D modell
  - háromszögek csúcspontjai [modellezési koordináták]
- adott egy 2D háromszögrajzoló algoritmus
  - képpontokat színez [viewport koordináták]
- a feladat:
  - a csúcspont modellezési koordinátáiból számoljuk ki, mely pixelben jelenik meg

Mi befolyásolja ezt?

- Hova, hogyan helyezzük el a modellt a virtuális világban
  - modellezési trafó, model, world
- Hol van, merre néz a kamera
  - kamera trafó, view
- Mekkora a látószög (és képméretarány)
  - perspektív projekció
- Hányszor hány pixel az ablak és hol van
  - viewport trafó

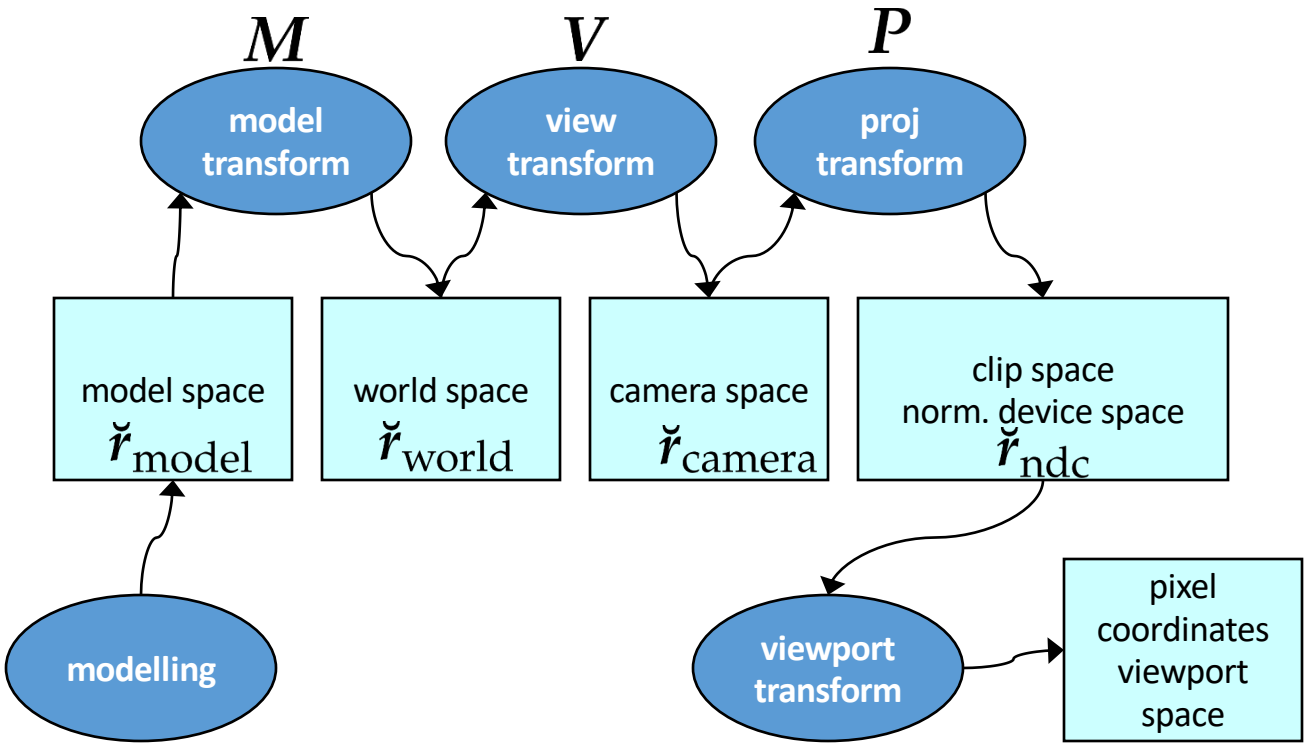
Bónusz feladat: árnyalás

- Felületi pont, fények koordinátái, szempozíció, felületi normálisok...
  - ugyanabban a 3D koordináta rendszerben legyenek
  - logikusan a modellezési transzformáció után
  - de a kamera trafó szög és távolságtartó

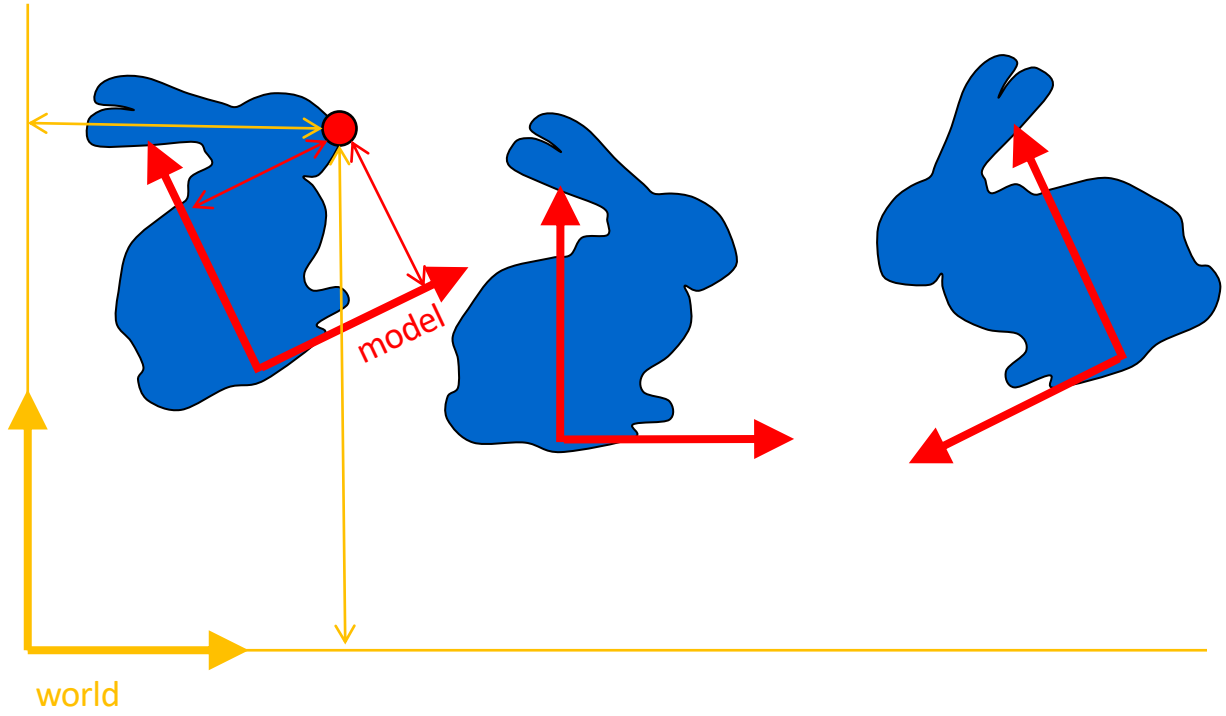
Bónusz feladat: takarás

- takarási probléma
  - a 2D pixel koordináták mellett kell egy mélységértéket is számolni
- fix pontos z-buffer
  - [0, 1]-beli értékeket tud összehasonlítani
  - a távolság, amihez a 0-t rendeljük: első vágósík
  - amihez az 1-et: hátsó vágósík

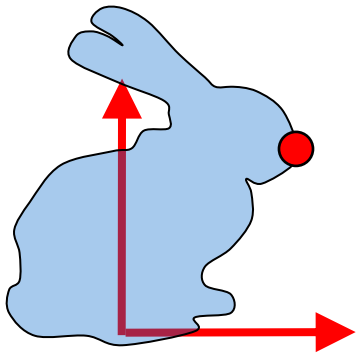
Transzformációs csővezeték



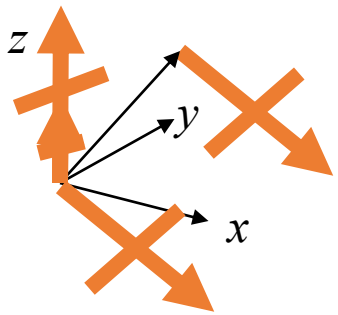
Modellezési és világkoordináták (statikus interpretáció)



Modellezési és világkoordináták (dinamikus interpretáció)



Modelezési transzformáció



1. scaling: s_x, s_y, s_z
2. orientation: φ, θ, ψ
3. position: q_x, q_y, q_z

$$\boldsymbol{M} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} & & & 0 \\ & \mathrm{R} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ q_x & q_y & q_z & 1 \end{bmatrix}$$

model trafo matrix

Transzformációs csővezeték

- modellezési transzformáció: ✓

$$\check{\boldsymbol{r}}_{\text{world}} = \check{\boldsymbol{r}}_{\text{model}}\boldsymbol{M}$$
$$\check{\boldsymbol{n}}_{\text{world}} = \check{\boldsymbol{n}}_{\text{model}}\boldsymbol{M}^{-1T}$$

- nézeti transzformáció:

$$\check{\boldsymbol{r}}_{\text{camera}} = \check{\boldsymbol{r}}_{\text{world}}\boldsymbol{V}$$

- vetítési transzformáció:

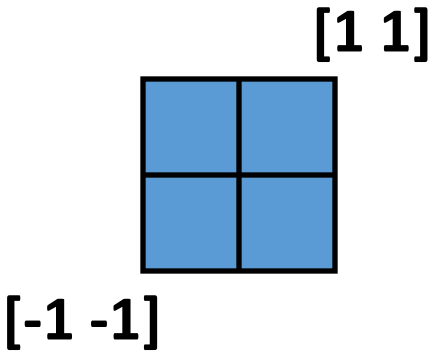
$$\check{\boldsymbol{r}}_{\text{ndc}} = \check{\boldsymbol{r}}_{\text{camera}}\boldsymbol{P}$$

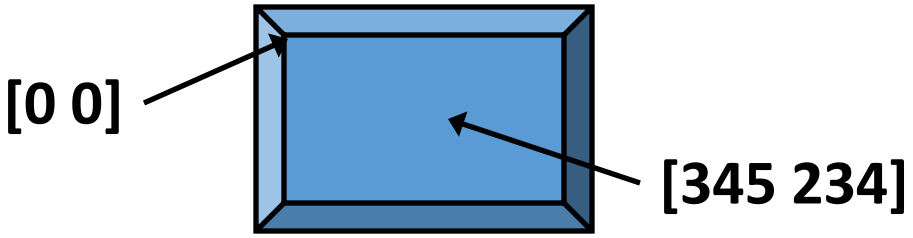
- együtt:

$$\boldsymbol{MVP}$$

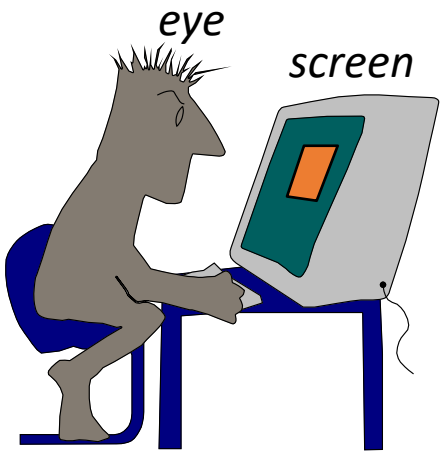
Hova kell ezt rajzolni a képernyőn?

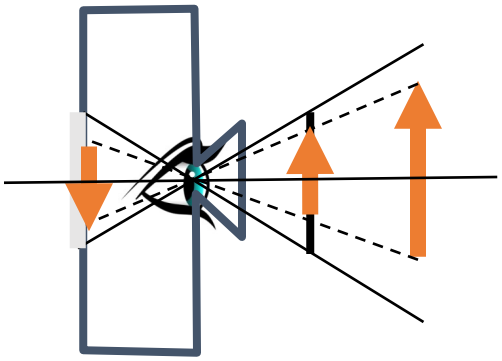
- Az összes többi transzformáció ennek a kiszámolására megy
  - kamera transzformáció
    - hol van a pont a kamerához képest
  - vetítési transzformáció
    - hova vetül ez az ablak téglalapjára
  - viewport transzformáció
    - melyik pixel



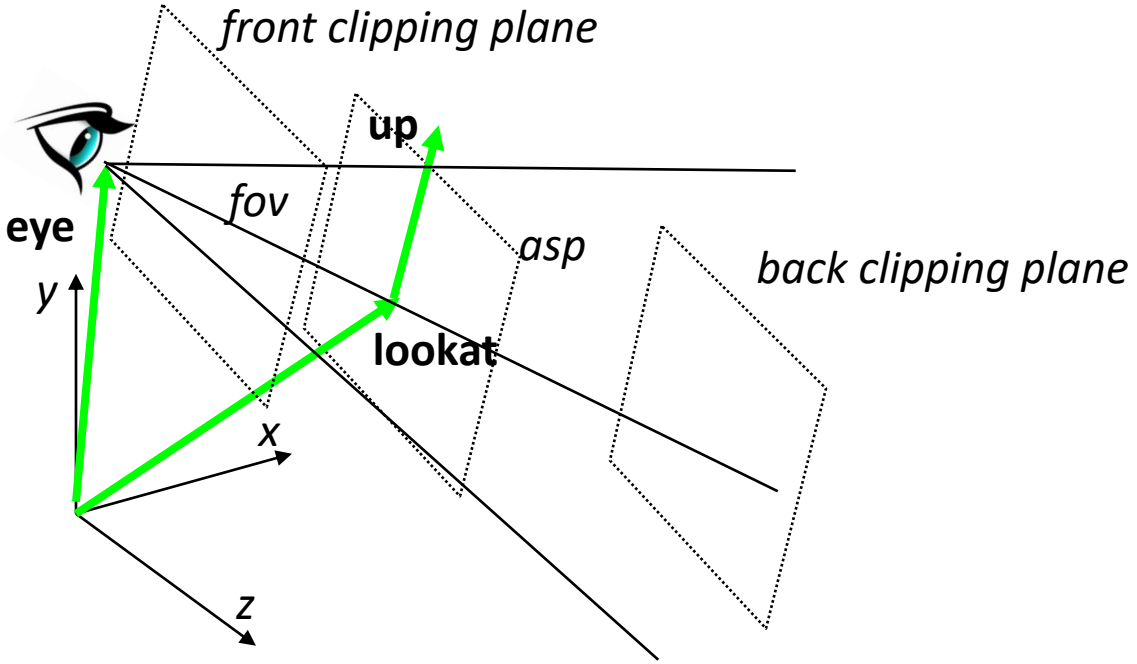


Kameramodell

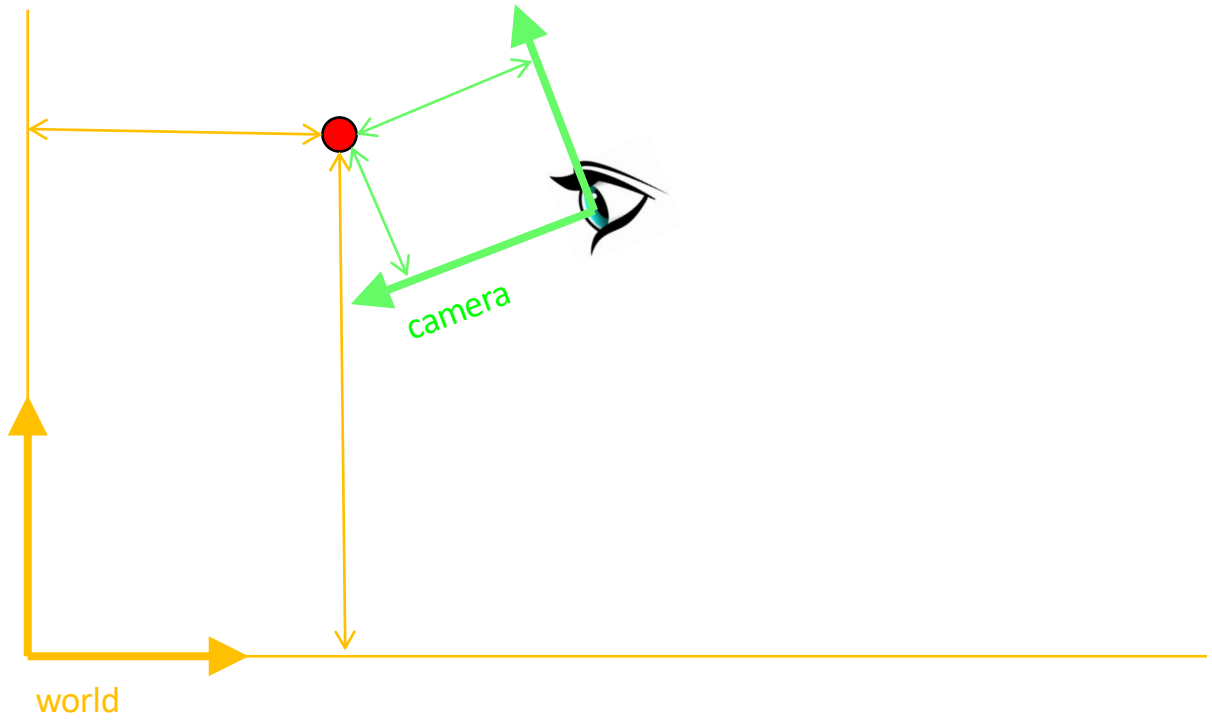




Camera obscura



Nézeti transzformáció (statikus interpretáció)

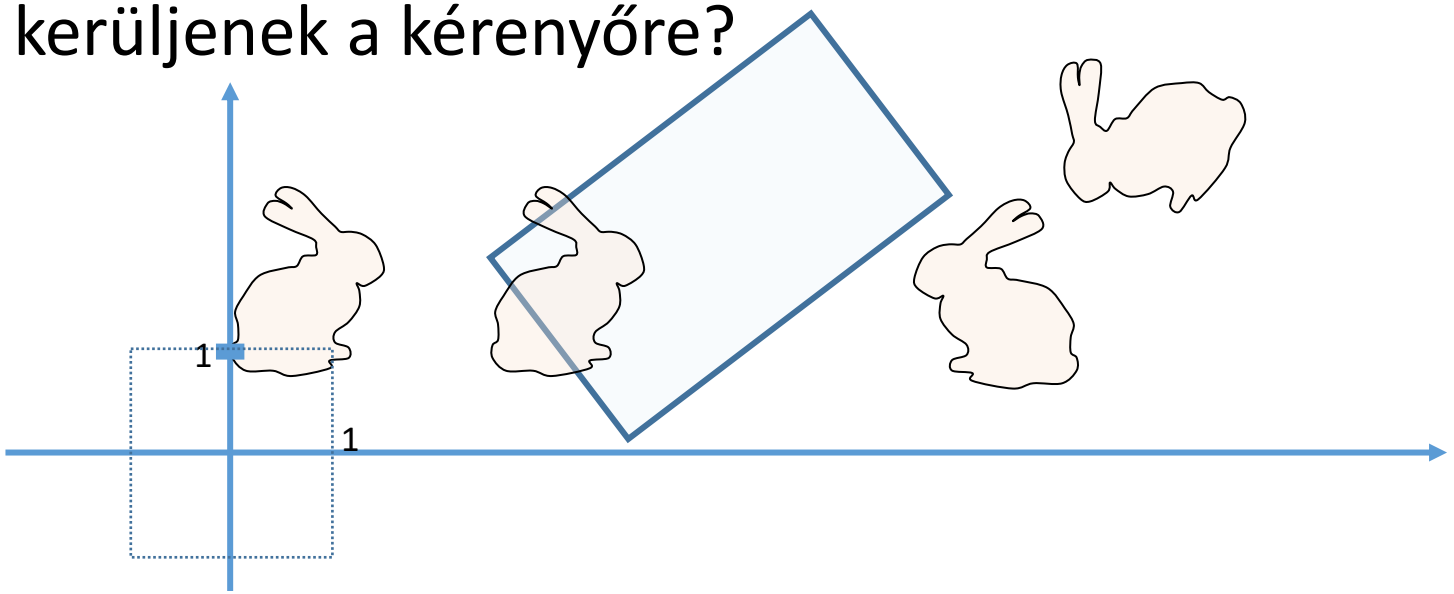


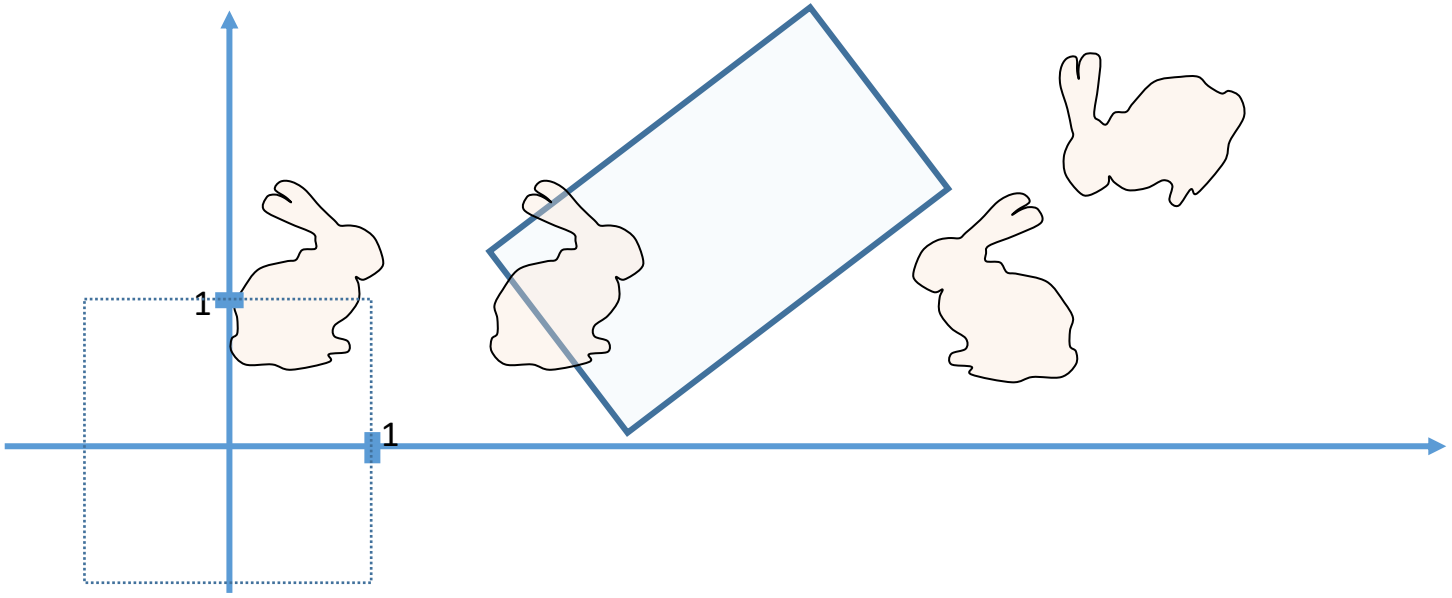
Emlékeztető – 2D trafó

- Hova kerül a pont a képernyőn?

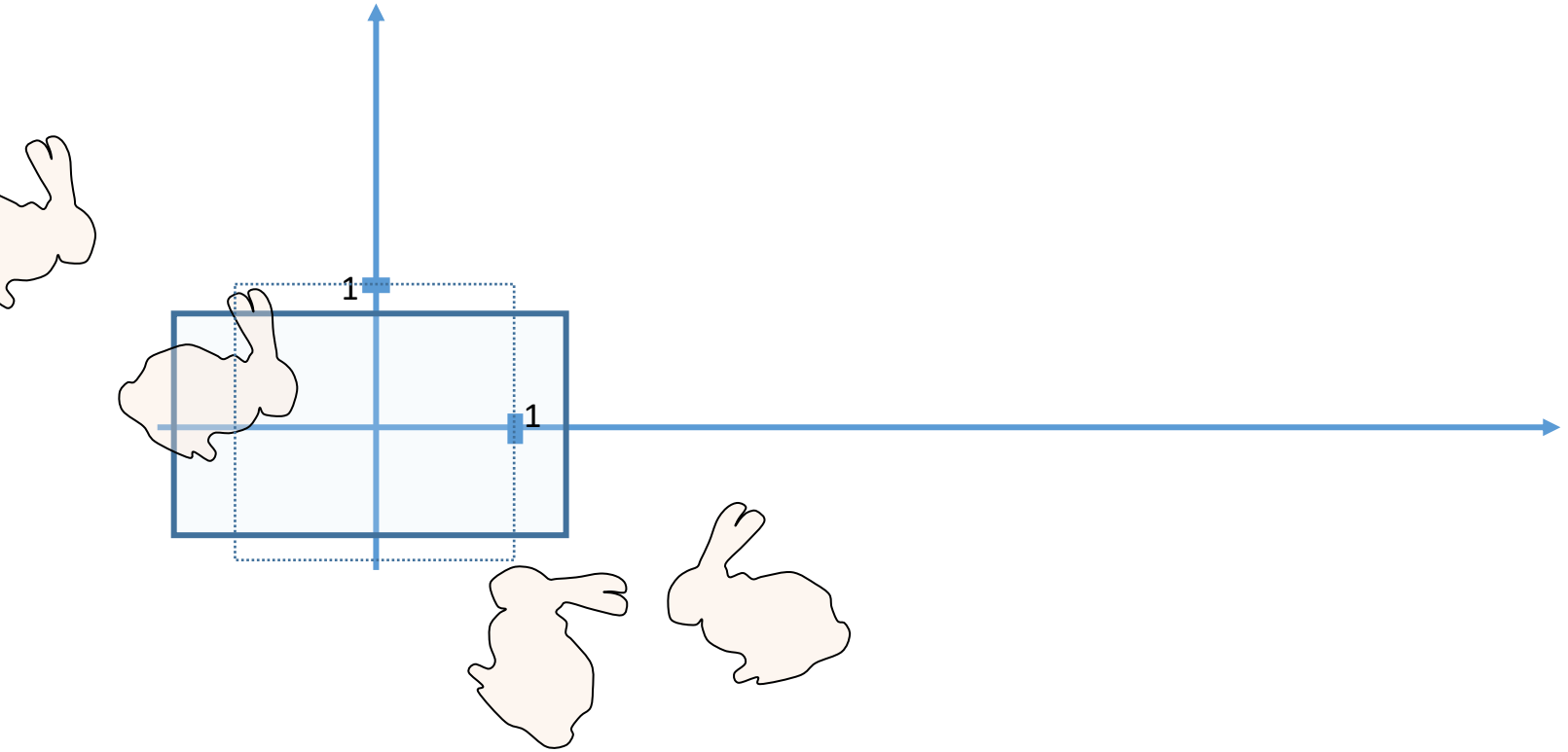
  illetve:

- Mit csináljunk a pontokkal, hogy a kívánt részek kerüljenek a kérenyőre?

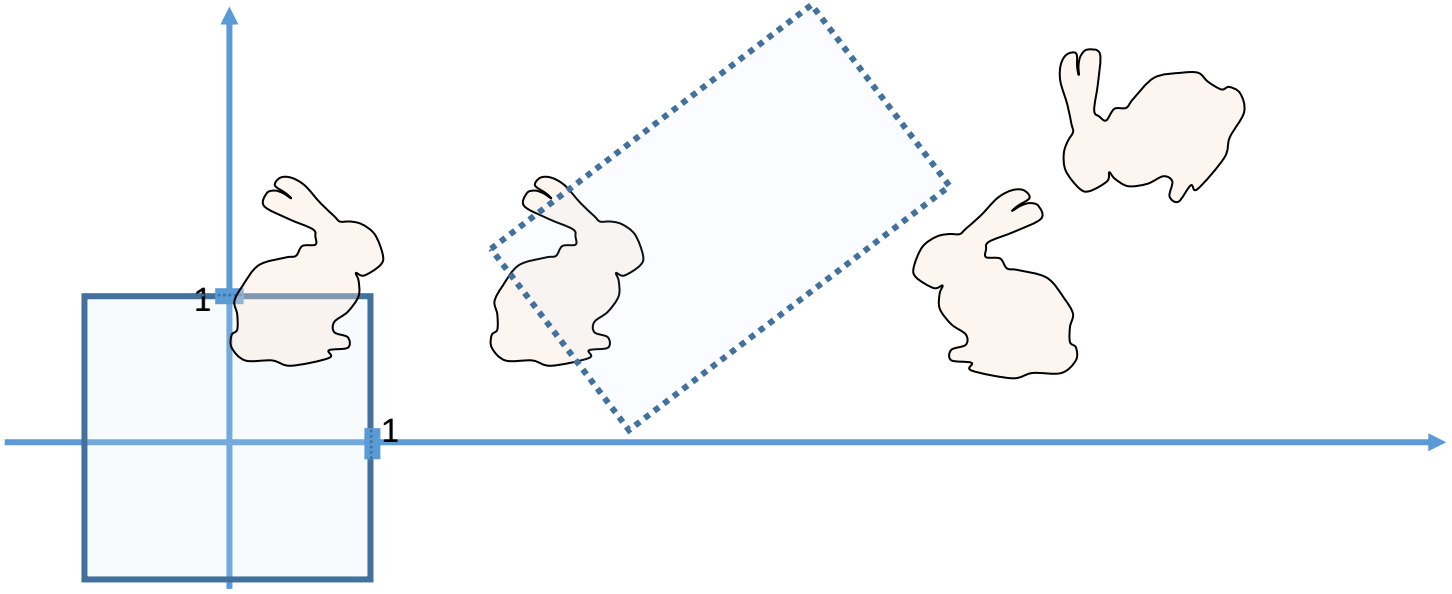




$$\begin{pmatrix} c_x & c_y & 1 \end{pmatrix} = \begin{pmatrix} w_x & w_y & 1 \end{pmatrix} \boldsymbol{T}_{\text{view}} \boldsymbol{R}_{\text{view}}$$



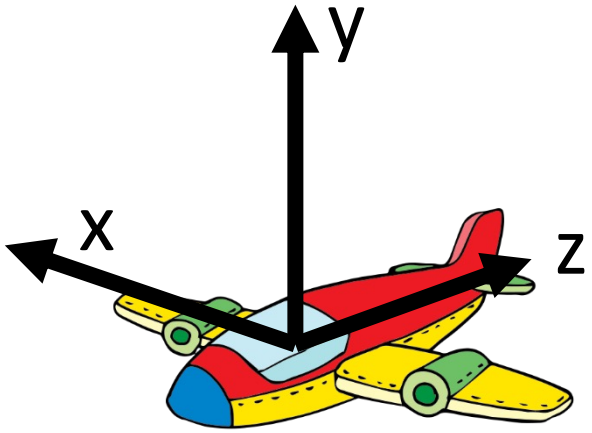
$$\begin{pmatrix} c_x & c_y & 1 \end{pmatrix} = \begin{pmatrix} w_x & w_y & 1 \end{pmatrix} \boldsymbol{T}_{\text{view}} \boldsymbol{R}_{\text{view}} \boldsymbol{S}_{\text{view}}$$



$$\begin{pmatrix} c_x & c_y & 1 \end{pmatrix} = \begin{pmatrix} w_x & w_y & 1 \end{pmatrix} \boldsymbol{T}_{\text{view}} \boldsymbol{R}_{\text{view}} \boldsymbol{S}_{\text{view}} \qquad \left(\frac{1}{2} \boldsymbol{S}_{\text{windowsize}} \boldsymbol{R}_{\text{windowangle}} \boldsymbol{T}_{\text{windowpos}} \right)^{-1}$$

Kameraorientáció geometriája

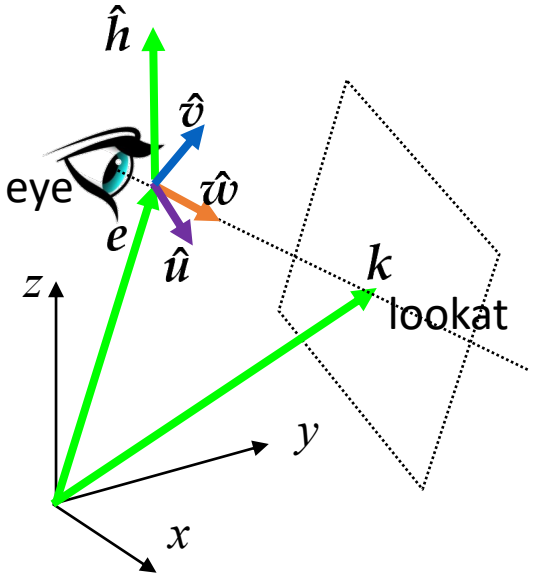
- jobbkezes
- főirányok
  - x: jobbra
  - y: fel
  - z: hátra (előre a -z)



- elforgatási szögek:
  - yaw – legyezés
  - pitch – menetemelkedés
  - roll - orsózás

Nézeti transzformáció: főirányok nézett pontból

generic upwards

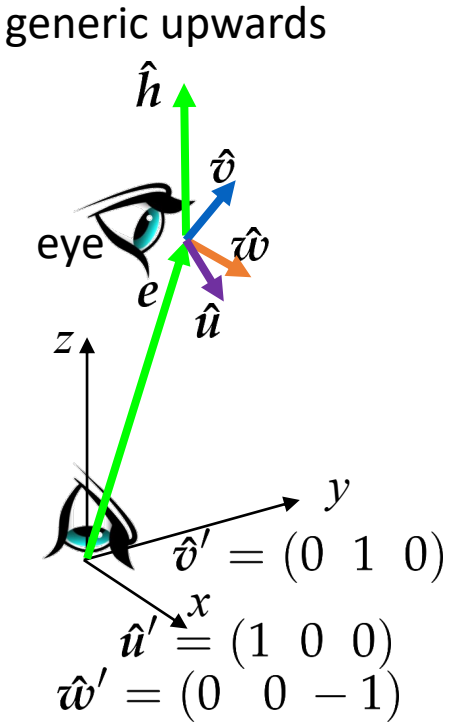


$$\hat{w} = \frac{k - e}{|k - e|}$$ ahead

$$\hat{u} = \frac{\hat{w} \times \hat{h}}{|\hat{w} \times \hat{h}|}$$ right

$$\hat{v} = \hat{u} \times \hat{w}$$ up

Nézeti transzformáció: mátrix a főirányokból és a pozícióból



$$\hat{w} = \frac{k - e}{|k - e|} \qquad \hat{u} = \frac{\hat{w} \times \hat{h}}{|\hat{w} \times \hat{h}|} \qquad \hat{v} = \hat{u} \times \hat{w}$$

ahead right camera up

$$V = \left(\begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ -w_x & -w_y & -w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ e_x & e_y & e_z & 1 \end{bmatrix} \right)^{-1}$$

$$\check{r}_{\text{camera}} = \check{r}_{\text{world}} V$$

Yaw-pitch-roll: másik mód a kameraorientáció megadására

$$\boldsymbol{V} = \left(\begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ e_x & e_y & e_z & 1 \end{bmatrix} \right)^{-1}$$

roll pitch yaw

- egérrel vezérléshez kényelmesebb
- egérmozgás változtatja a szöget

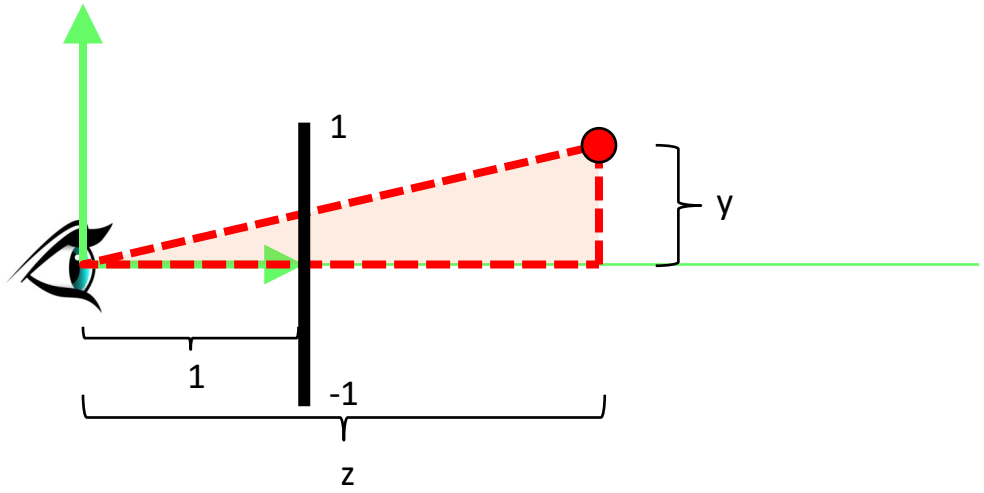
Előreirány a yaw-pitch-ből (nulla roll)

$$\hat{\boldsymbol{w}} = \begin{bmatrix} 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ e_x & e_y & e_z & 1 \end{bmatrix}$$

$$\hat{\boldsymbol{w}} = \begin{bmatrix} 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

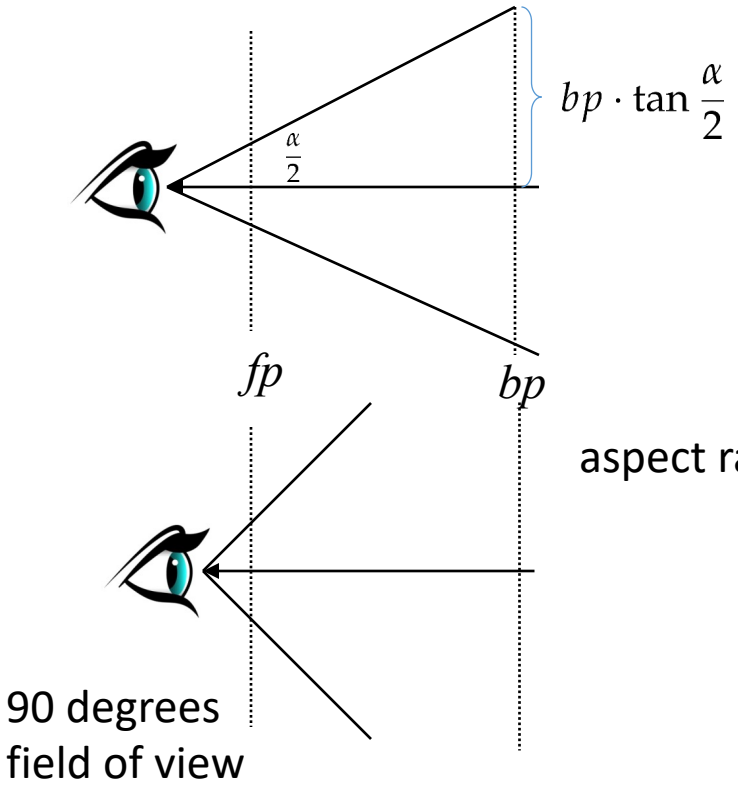
$$\hat{\boldsymbol{w}} = \begin{bmatrix} 0 & \sin\theta & -\cos\theta & 0 \end{bmatrix} \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\cos\theta\sin\varphi & \sin\theta & -\cos\theta\cos\varphi & 0 \end{bmatrix}$$

Vetítési transzformáció (statikus interpretáció)



képernyőkoordináták: (y/z, 1)
homogén koordinátákban (y/z, 1, 1)
vagyis (y, z, z)

Normalizálás



field of view angle (FOV)

aspect ratio

$$\begin{bmatrix} \frac{1}{a \cdot \tan \frac{\alpha}{2}} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan \frac{\alpha}{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Osztás mátrixszorzással? Csak homogén koordinátákban!

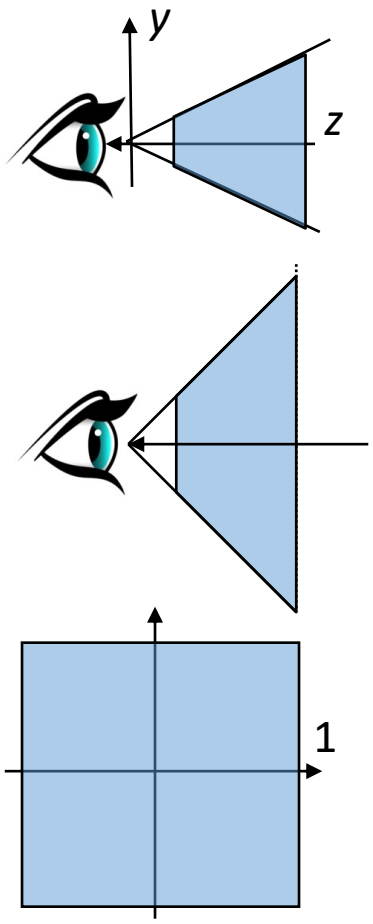
$$\breve{x}_{\text{ndc}} / \breve{w}_{\text{ndc}} = \breve{x}_{\text{ncam}} / \breve{z}_{\text{ncam}}$$
$$\breve{y}_{\text{ndc}} / \breve{w}_{\text{ndc}} = \breve{y}_{\text{ncam}} / \breve{z}_{\text{ncam}}$$

$$\begin{bmatrix} \breve{x}_{\text{ndc}} & \breve{y}_{\text{ndc}} & \breve{z}_{\text{ndc}} & \breve{w}_{\text{ndc}} \end{bmatrix} =$$

$$\begin{bmatrix} \breve{x}_{\text{ncam}} & \breve{y}_{\text{ncam}} & \breve{z}_{\text{ncam}} & \breve{w}_{\text{ncam}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & ? & 1 \\ 0 & 0 & ? & 0 \end{bmatrix}$$

milyen mátrix jön ide?

Perspektív vetítés mélységgel



$$\begin{bmatrix} \frac{1}{a\tan(\frac{\alpha}{2})} & 0 & 0 & 0 \\ 0 & \frac{1}{a\tan(\frac{\alpha}{2})} & 0 & 0 \\ 0 & 0 & a\frac{f+b}{b-f} & -1 \\ 0 & 0 & -\frac{2fb}{b-f} & 0 \end{bmatrix}$$

$$\check{\boldsymbol{r}}_{\text{ndc}} = \check{\boldsymbol{r}}_{\text{camera}}\boldsymbol{P} \qquad w_{\text{ndc}} = -z_{\text{camera}}$$

$$\boldsymbol{r}_{\text{ndc}} = \begin{pmatrix} x_{\text{ndc}} & y_{\text{ndc}} & z_{\text{ndc}} \end{pmatrix} / w_{\text{ndc}}$$

Kameraparaméterek

```
class PerspectiveCamera(vararg programs : Program) :
UniformProvider("camera") {

  val position by Vec3(0.0f, 0.0f, 0.0f)
  var roll = 0.0f
  var pitch = 0.0f
  var yaw = 0.0f

  var ahead = Vec3(0.0f, 0.0f,-1.0f)
  var right = Vec3(1.0f, 0.0f, 0.0f)
  var up    = Vec3(0.0f, 1.0f, 0.0f)
```

jobbkezes
koordinátarendszer:
x jobbra, y fel, z hátrafelé

Kameraparaméterek

```
var fov = 1.0f
var aspect = 1.0f
var nearPlane = 0.1f
var farPlane = 1000.0f
```

Tagváltozók mozgatáshoz

```
var speed = 0.005f
var isDragging = false
val mouseDelta = Vec2(0.0f, 0.0f)
```

Mátrixok

```
val rotationMatrix = Mat4()
val viewProjMatrix by Mat4()
val rayDirMatrix by Mat4()
```

Világ felfelé iránya

```
companion object {
    val worldUp = Vec3(0.0f, 1.0f, 0.0f)
 }
```

View mátrix számítása

```
rotationMatrix.set().
  rotate(roll).
  rotate(pitch, 1.0f, 0.0f, 0.0f).
  rotate(yaw, 0.0f, 1.0f, 0.0f)
viewProjMatrix.set(rotationMatrix).
  translate(position).
  invert()
```

Proj mátrix számítása

```
val yScale = 1.0f / tan(fov * 0.5f)
val xScale = yScale / aspect
val f = farPlane
val n = nearPlane
viewProjMatrix *= Mat4(
    xScale ,    0.0f ,          0.0f ,   0.0f,
      0.0f ,  yScale ,          0.0f ,   0.0f,
      0.0f ,    0.0f ,  (n+f)/(n-f) ,  -1.0f,
      0.0f ,    0.0f ,  2*n*f/(n-f) ,   0.0f)
```

Mozgatás: yaw, pitch drag

yaw y körül forgat balra
ha jobbra húzom, jobbra
forduljon

```
fun move(dt : Float, keysPressed : Set<String>) {
  if(isDragging) {
    yaw -= mouseDelta.x * 0.002f
    pitch -= mouseDelta.y * 0.002f
    if(pitch > 3.14f/2.0f) { pitch = 3.14f/2.0f }
    if(pitch < -3.14f/2.0f) {
      pitch = -3.14f/2.0f
    }
    mouseDelta.set()
  }
```

pitch x körül forgat felfelé
ha lefelé húzom, lefelé forduljon

Mozgatás: főirányok a szögekből

```
ahead = (Vec3(0.0f, 0.0f, -1.0f).xyz0 *
rotationMatrix).xyz
    right = (Vec3(1.0f, 0.0f,  0.0f).xyz0 *
rotationMatrix).xyz
    up    = (Vec3(0.0f, 1.0f,  0.0f).xyz0 *
rotationMatrix).xyz
```

Mozgatás gombokkal

```
if("W" in keysPressed) { position += ahead * (speed * dt) }
    if("S" in keysPressed) { position -= ahead * (speed * dt)  }
    if("D" in keysPressed) { position += right * (speed * dt) }
    if("A" in keysPressed) { position -= right * (speed * dt) }
    if("E" in keysPressed) { position += up * (speed * dt) }
    if("Q" in keysPressed) { position -= up * (speed * dt) }
```

Eseménykezelők – meg is kell hívni őket!

```
 fun mouseDown() {
    isDragging = true
    mouseDelta.set()
  }

  fun mouseMove(event : MouseEvent) {
    mouseDelta.x += event.asDynamic().movementX as Float
    mouseDelta.y += event.asDynamic().movementY as Float
    event.preventDefault()
  }
  fun mouseUp() {
    isDragging = false
  }
}
```

Képméretarány állítása – ezt is meg kell hívni!

```
fun setAspectRatio(ar : Float) {
    aspect = ar
    update()
 }
```

Sugárirány kiszámítása NDC-ből

$$\boldsymbol{x}_{\mathrm{w}}\boldsymbol{VP} = \boldsymbol{x}_{\mathrm{ndc}}$$

$$\boldsymbol{d} = \boldsymbol{x}_{\mathrm{w}} - \boldsymbol{e} \qquad \hat{\boldsymbol{d}} = \frac{\boldsymbol{d}}{|\boldsymbol{d}|}$$

$$\boldsymbol{d} = \boldsymbol{x}_{\mathrm{ndc}}(\boldsymbol{VP})^{-1} - \boldsymbol{e}$$

$$\boldsymbol{d} = \boldsymbol{x}_{\mathrm{ndc}}(\boldsymbol{VP})^{-1}\boldsymbol{E}^{-1}$$

$$\boldsymbol{d} = \boldsymbol{x}_{\mathrm{ndc}}(\boldsymbol{EVP})^{-1}$$

(**EVP**)$^{-1}$ -et nevezzük rayDirMatrix-nak

Konstruktorban, ill. mozgatás után

```
rayDirMatrix.set().translate(position)
rayDirMatrix *= viewProjMatrix
rayDirMatrix.invert()
```