

Transformarea de vizualizare din spațiul 3D

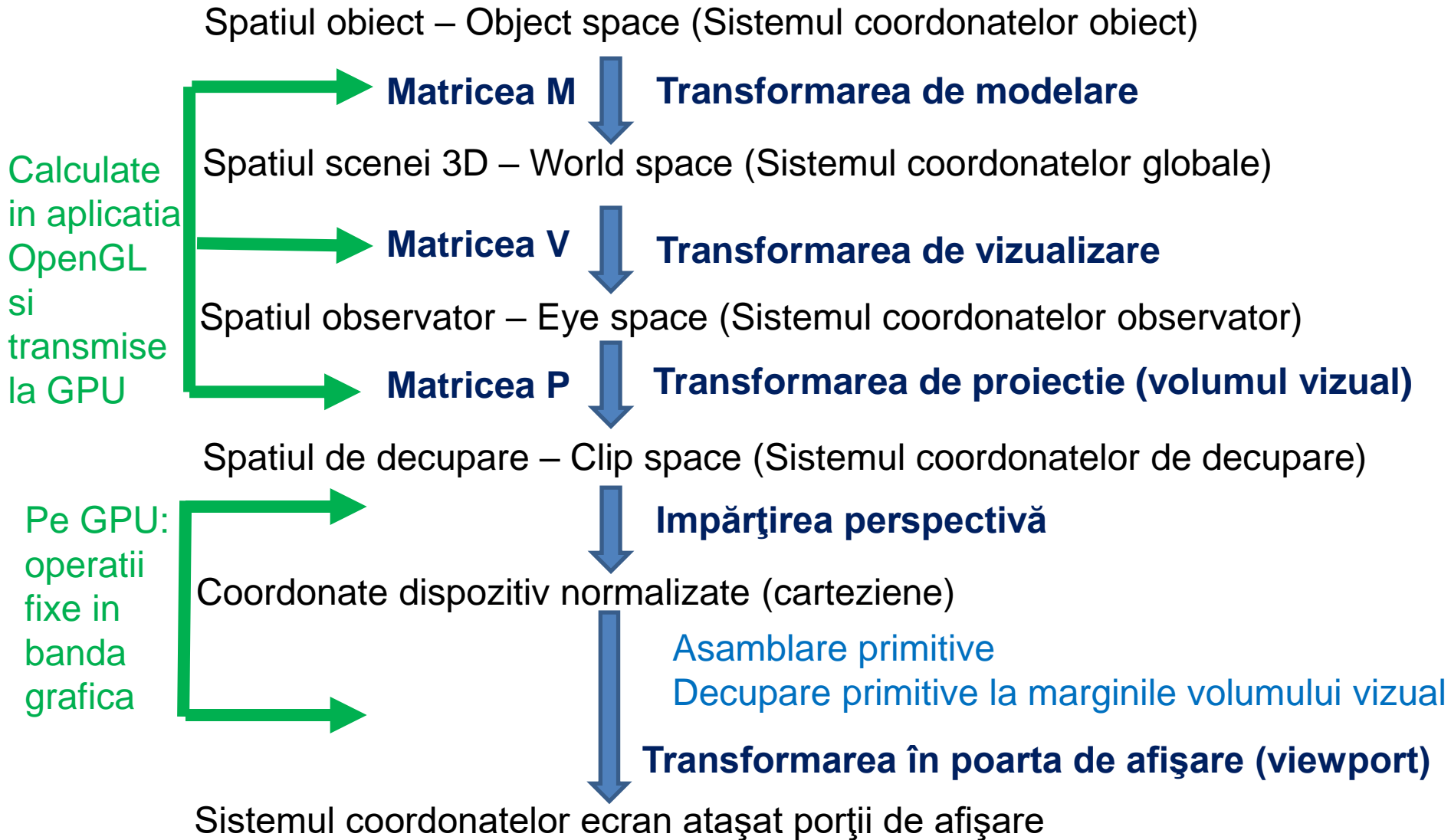
Prof. univ. dr. ing. Florica Moldoveanu

Curs Elemente de Grafică pe Calculator – UPB, Automatică și Calculatoare
2020-2021

Transformarea de vizualizare din spațiul 3D

- Are ca scop **obținerea unei vederi a unei scene 3D** pe o suprafață de afișare.
- Poate fi comparată cu transformarea efectuată de o cameră foto asupra lumii reale pentru a se obține imaginea dintr-o fotografie.
 - Imaginea obținută depinde de:
 - poziția camerei (observatorului)
 - orientarea camerei
 - ceea ce vede ochiul prin vizorul camerei foto (volumul vizual)
- Este o transformare compusă din mai multe transformări, între care și o proiectie $R^3 \rightarrow R^2$
- Transformările se efectuează asupra vârfurilor obiectelor din scena 3D.
- Include și operația de decupare (clipping) a primitivelor grafice la marginea volumului vizual (eliminarea părților din scena 3D care sunt în afara volumului vizual al vederii curente).

Transformarea varfurilor în OpenGL



Matricile transformării varfurilor obiectelor 3D

M – matricea de modelare: difera de la obiect la obiect

V – matricea de vizualizare: aceeași pentru întreaga scenă vizualizată

P – matricea de proiecție: aceeași pentru întreaga scenă vizualizată

$MV = V \cdot M$ (matricea “model-view”)

$MVP = P \cdot V \cdot M$ (matricea “model-view-projection”)

Matricile sunt calculate în programul OpenGL folosind funcții ale bibliotecii **glm** și transferate la GPU

Transformarea vârfurilor din coordonate obiect în coordonate de decupare (clip):

$$[x_c, y_c, z_c]^T = PVM \cdot [x_o, y_o, z_o]^T$$

- Transformarea vârfurilor este efectuată în programul **Vertex Shader executat pe GPU**, folosindu-se matricea MVP transferată din programul OpenGL.

Banda grafică începând cu OpenGL 3.0

Aplicație OpenGL

- Vârfuri în coordonate obiect
- Topologia primitivelor
- Matricile M, V, P

CPU

Buffer vârfuri

Buffere globale

Matrici transformare vârfuri
Topologie primitive
Caracteristici surse lumina

Memoria grafică

Z-buffer
Frame buffer

GPU

Texturi

Generare flux de
înregistrari varf

Inregistrare varf

Vertex shader

- Transformare varf (MVP)

(în paralel)

Impartirea perspectiva

Asamblarea primitivelor

Eliminare fețe nevizibile

Decupare primitive

Transformarea in viewport

(în paralel)

Rasterizare: calcul adrese de pixel
- Generare flux înreg. fragment
Testul de vizibilitate fragment
Operatii raster

Inregistrare fragment

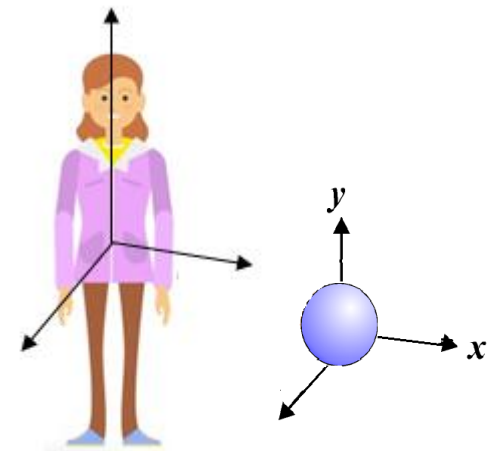
Fragment shader

- Calcul culoare fragment

Spatiul obiect

Spatiul obiect (Sistemul coordonatelor obiect)

- In mod uzual, fiecare obiect este definit in propriul sau sistem de coordonate (sistemul coordonatelor locale – coordonate obiect)
- Sistem de coordonate carteziane 3D « dreapta »
- Exemple: cubul, sfera, scheletul unui personaj animat, elemente de mobilier, etc.
De regula, obiectul este definit centrat in originea sistemului sau de coordonate.
- Avantaje: usurinta modelarii și reutilizarea modelelor obiectelor 3D

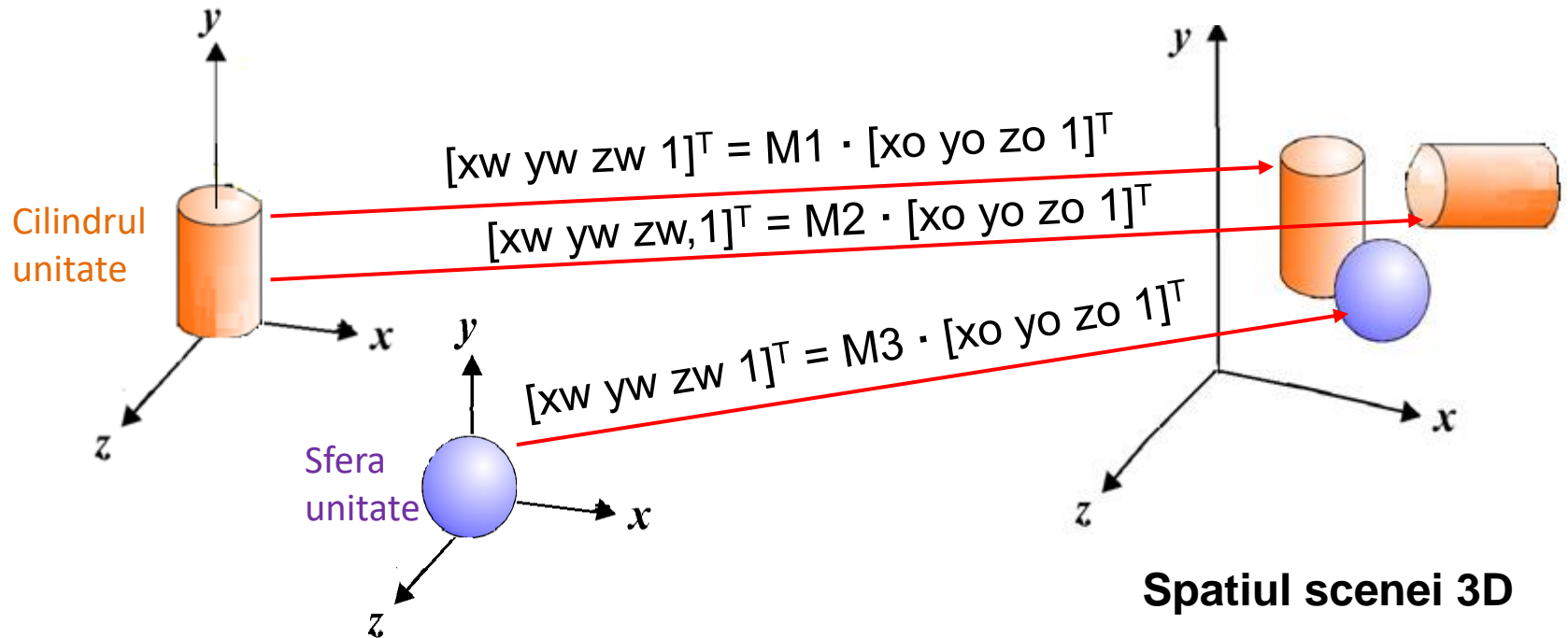


Spatiul scenei 3D -World space (Sistemul coordonatelor globale)

- Spațiul in care este compusă scena 3D
- Scena - raportată la un sistem de referință global – sistem de coordonate carteziane 3D “dreapta”, numit *Sistemul coordonatelor globale*
- Originea sistemului – aleasa de modelatorul scenei

Spațiul scenei 3D

Transformarea de modelare



$[x_o \ y_o \ z_o \ 1]$: coord omogene varf în sist de coord al unui obiect

$[x_w \ y_w \ z_w \ 1]$: coord omogene varf în sist de coord globale

Transformarea de modelare:



Transformare geometrica compusa. Exemple:

$$M1 = T(tx1, ty1, tz1) * S(sx, sy, sz)$$

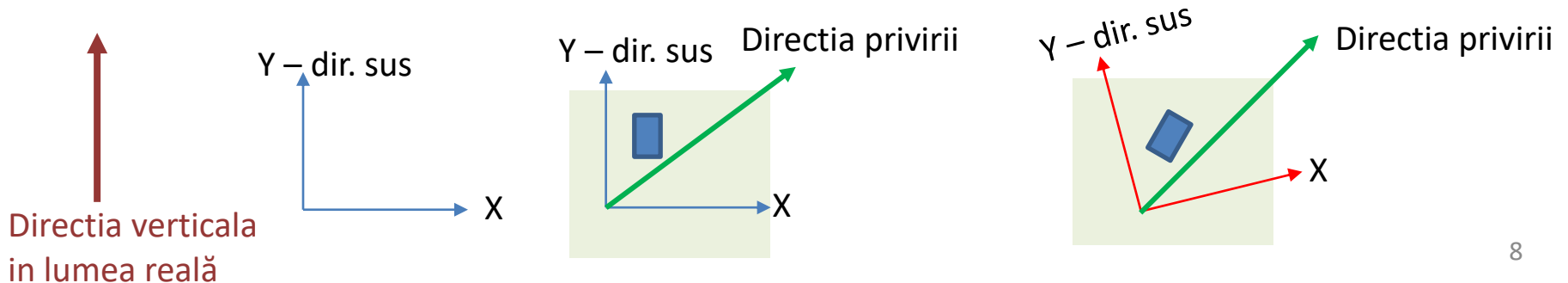
$$M2 = T(tx2, ty2, tz2) * R_{oz}(1.57) * S(sx, sy, sz)$$

Spatiul camerei (observator)

Analogie cu aparatul de fotografiat (camera reala)

Un aparat foto poate produce o fotografie in care apar o parte din obiectele spatiului real (3D).

- Continutul fotografiei depinde de:
 - pozitia aparatului in spatial real
 - orientarea acestuia: directia in care priveste fotograful
 - rotatia aparatului față de pozitia sa normală
- Fotografia este formata intr-un plan 2D, care are atasat un sistem cartezian 2D, XOY:
- **In pozitia normala a aparatului, directia axei OY (axa verticala) este directia “sus” a aparatului.**
- Directia in care priveste fotograful este perpendiculara pe planul imaginii, formand impreuna cu axele OX, OY un sistem de coord carteziane 3D.
- Fotograful poate roti aparatul de fotografiat, astfel incat directia “sus” a aparatului se schimbă.



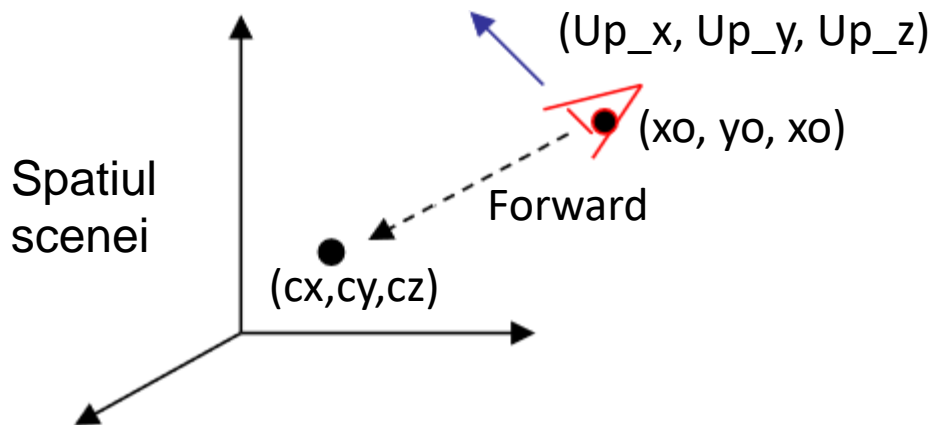
Spațiul camerei (observator)

Sistemul de coordonate observator

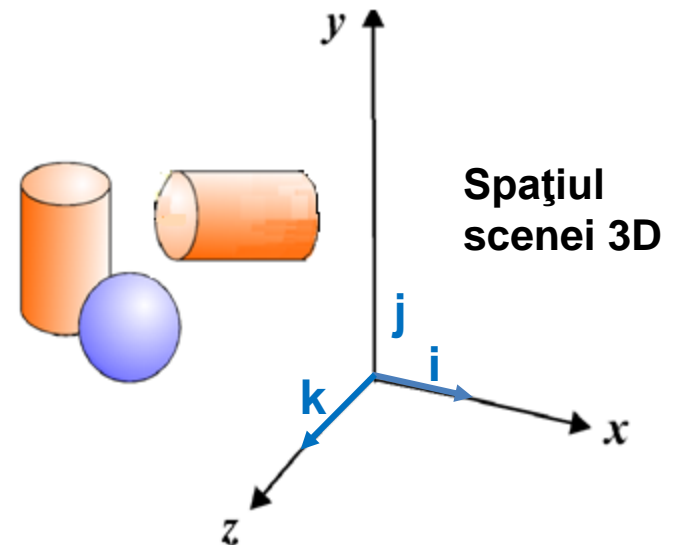
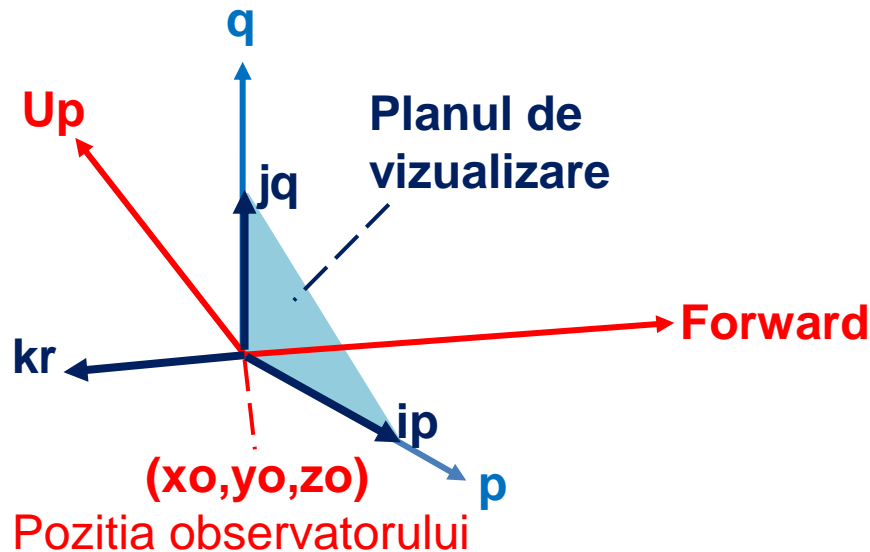
- Spațiul camerei este spațiul scenei 3D văzut din poziția camerei virtuale (observatorului) pe direcția în care privește observatorul.
- Spațiul camerei este raportat la un sistem de coordonate atasat camerei – sistemul de coordonate observator.

➤ **Sistemul de coordonate observator este definit prin 3 parametri:**

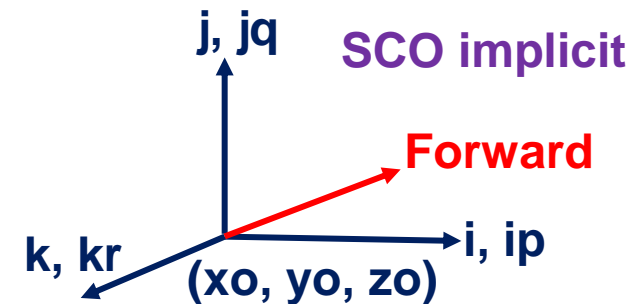
1. Originea sa: **poziția camerei virtuale** în spațiul scenei (x_o, y_o, z_o).
2. **Direcția în care privește observatorul** (spre un punct din scenă) – **vectorul Forward**
3. **Direcția “sus” a camerei (vectorul Up)**: reprezintă **rotatia camerei**



Sistemul de coordonate observator (SCO)

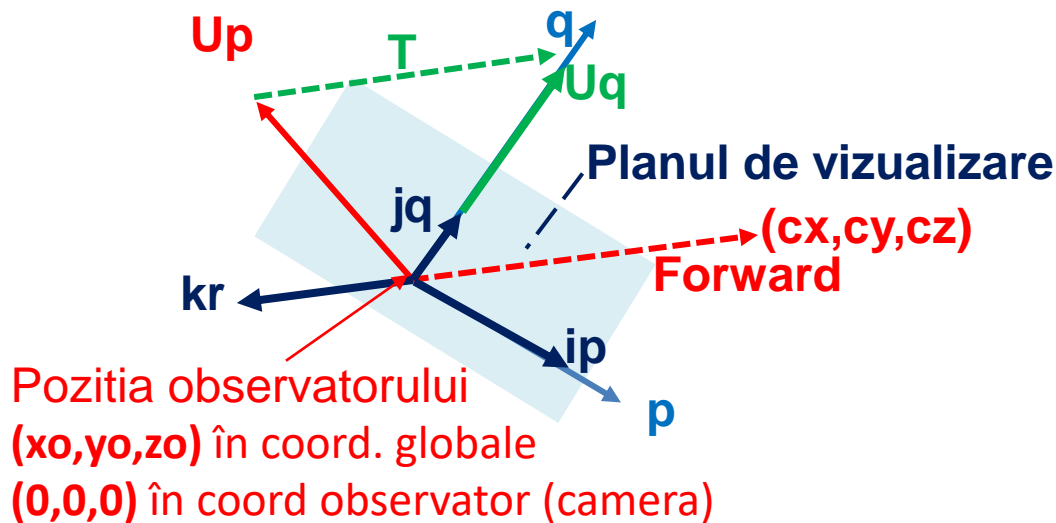


- (x_o, y_o, z_o) : originea sistemului de coordonate observator (poziția camerei)
- (x_o, y_o, z_o) împreună cu vectorul **Forward** definesc un plan, numit **planul de vizualizare**.
- **Forward** – normala la planul de vizualizare
- axa q : axa verticală a planului de vizualizare
- axa p : axa orizontală a planului de vizualizare
- (ip, jq, kr) formează un sistem de coordonate 3D dreaptă



ip, jq, kr : versorii axelor sistemului de coordonate observator coincid cu cei ai sist. coordonatelor globale

Determinarea axelor sistemului de coordonate observator (1)



$kr = - \text{Forward} / |\text{Forward}|$
 Versorul direcției axei z a SCO

- Uq : proiecția ortogonală a vectorului Up în planul de vizualizare: $Uq = Up + T$
- Uq dă direcția axei verticale a planului de vizualizare

$$T \parallel (-kr) \rightarrow T = s^*(-kr) = -s^*kr \quad (s \text{ un scalar oarecare}) \rightarrow Uq = Up - s^*kr$$

$$Uq \text{ perpendicular pe } kr \rightarrow Uq \cdot kr = 0 \rightarrow (Up - s^*kr) \cdot kr = 0 \rightarrow Up \cdot kr - s^*kr \cdot kr = 0 \rightarrow s = Up \cdot kr$$

$$Uq = Up - (Up \cdot kr) \cdot kr$$

$jq = Uq / |Uq|$ este versorul axei verticale a planului de vizualizare

$ip = jq \times kr$ este versorul axei orizontale a planului de vizualizare

(ip, jq, kr) formează un sistem de coordonate carteziane 3D dreapta

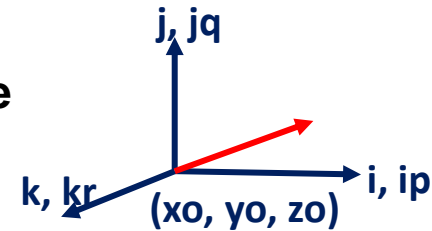
Matricea transformării de vizualizare (V) -1

Transformarea de vizualizare: din spatiul scenei → în spatiul camerei

$$[x_e \ y_e \ z_e \ 1]^T = V \cdot [x_w \ y_w \ z_w \ 1]^T \quad \text{eye coordinates} \leftarrow \text{world coordinates}$$

Sistemul de coordonate observator implicit in OpenGL

- poziția observatorului: **originea sistemului de coordonate globale**
- direcția în care privește observatorul: **direcția negativă a axei OZ**
- direcția sus a planului de vizualizare: **direcția pozitivă a axei OY**



Cu aceste valori implicite, transformarea de vizualizare este transformarea identica.

Pentru calculul matricei transformarii de vizualizare se folosește funcția **lookAt**:

glm::mat4 V =

glm::lookAt (glm::vec3 (xo,yo,zo), glm::vec3 (cx,cy,cz), glm::vec3 (Upx, Upy, Upz))

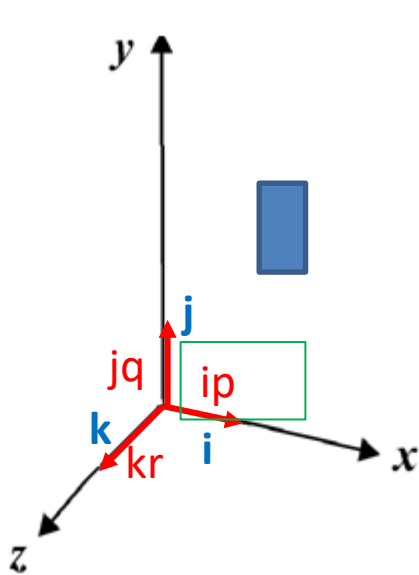
(xo, yo, zo): **poziția observatorului**

(cx,cy,cz): centrul de interes - punctul din scena 3D catre care priveste observatorul;

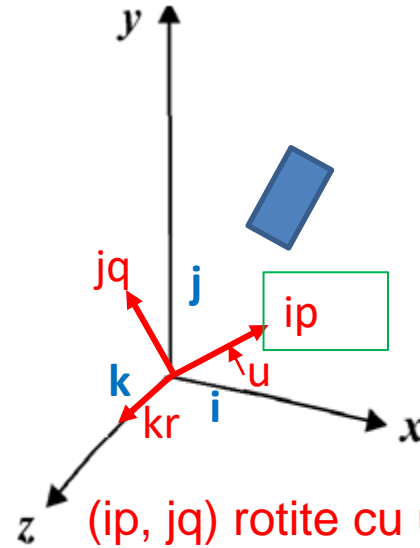
vectorul (xo,yo,zo) → (cx,cy,cz) reprezinta **vectorul Forward**

(Upx, Upy, Upz): **vectorul « sus » al camerei (Up)**

Matricea transformării de vizualizare -2



$V = I$ (matricea identica)



$V?$

V : matricea care transforma varfurile obiectelor din spatiul scenei in spatiul camerei

(observator - eye) $\rightarrow [x_e \ y_e \ z_e \ 1]^T = R_{Oz}(-u) \cdot [x_w \ y_w \ z_w \ 1]^T$

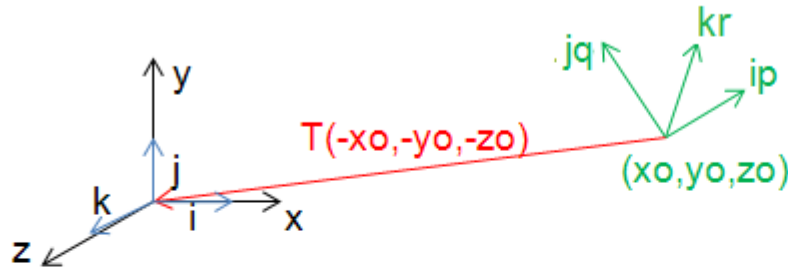
Matricea transformarii de vizualizare: $V = R_{Oz}(-u)$

➤ Aplicand V axelor sistemului de coordonate observator, se vor suprapune pe cele ale sistemului de coordonate globale.

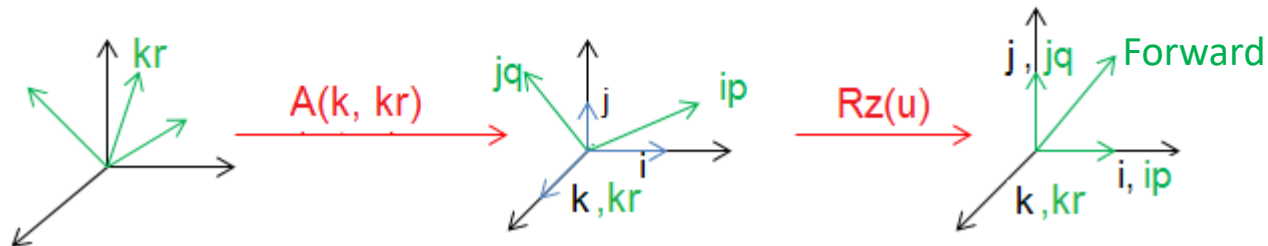
Matricea transformării de vizualizare -3

Sistemul de coordonate al scenei

Sistemul de coordonate observator



Sistemul de coordonate implicit in OpenGL



- $A(k, kr)$ este o transformare prin care se aliniaza axele z ale celor 2 sisteme de coordonate: consta din 2 rotatii – o rotatie in jurul axei X si una in jurul axei Y .
- Prin rotatia in jurul axei z (Rz) se aliniaza celelalte 2 axe.

Matricea transformarii de vizualizare: $V = Rz(u) * A(k, kr) * T(-x_o, -y_o, -z_o)$

Transformarea de modelare și vizualizare

din sistemul de coordonate obiect → în sistemul de coordonate observator

❖ Transformare compusa (model-view) : $MV = V \cdot M$

M – matricea de modelare folosita pentru transformarea unui obiect in spatiul scenei

V – matricea de vizualizare a scenei

Transformarea varfurilor din coordonate obiect in coordonate observator (eye):

$$[x_e \ y_e \ z_e \ w_e]^T = MV \cdot [x_o \ y_o \ z_o \ w_o]^T = V \cdot M \cdot [x_o \ y_o \ z_o \ w_o]^T$$

➤ Pentru calculele de iluminare a obiectelor scenei 3D se folosesc vectori normali atasati varfurilor obiectelor.

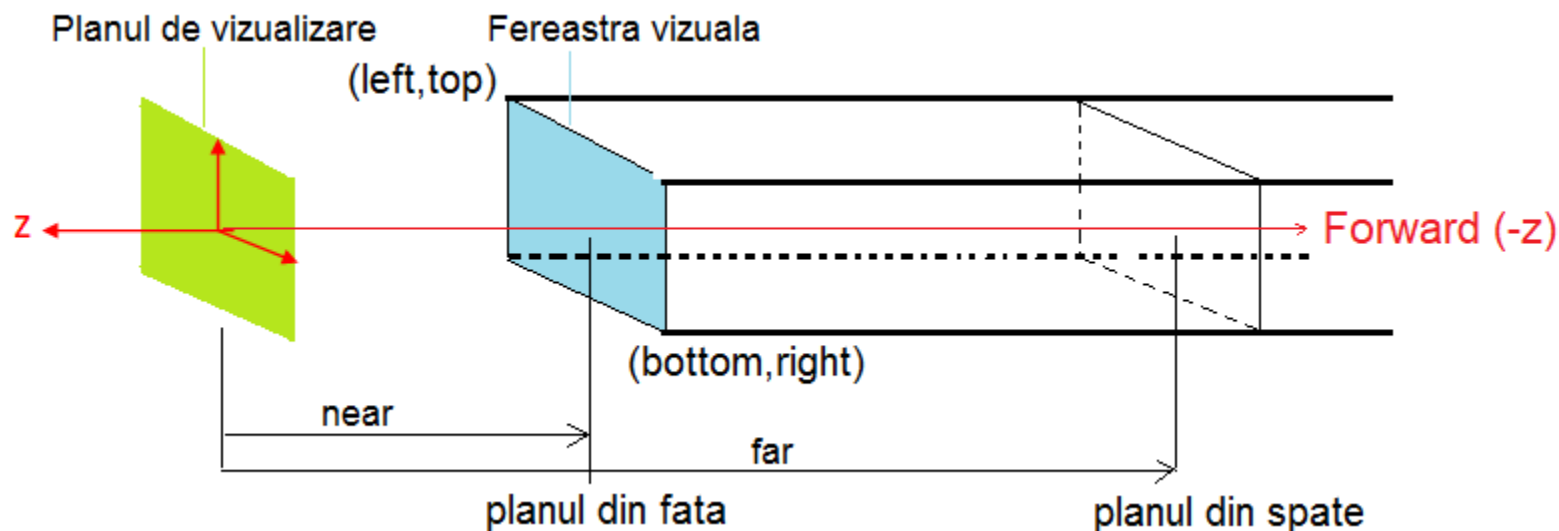
Transformarea normalelor din coordonate obiect in coordonate observator:

$$[n_xe \ n_ye \ n_ze \ w_e]^T = (MV^{-1})^T \cdot [n_xo \ n_yo \ n_zo \ n_w o]^T$$

Transformarea de proiectie (1)

Determinata de **volumul vizual** definit de programator

Volumul vizual al proiectiei ortografice: paralelipiped dreptunghic cu laturile paralele cu axa z a sistemului de coordonate observator (SCO), delimitat in adancime de planul din fata (near) si planul din spate (far) al vederii; near si far sunt distante (pozitive) față de planul de vizualizare masurate pe axa z negativa a SCO.

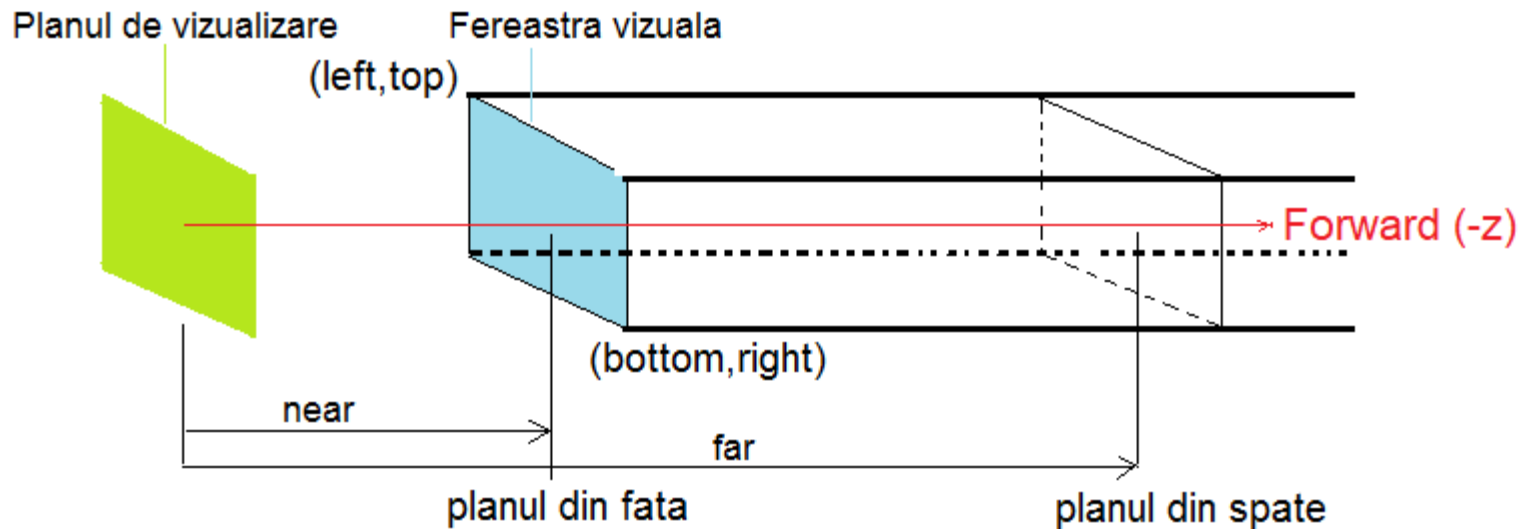


- Coordonatele colturilor ferestrei vizuale sunt raportate la sistemul de coordonate al planului din față, care are axele paralele cu cele ale planului de vizualizare.

Transformarea de proiectie (2)

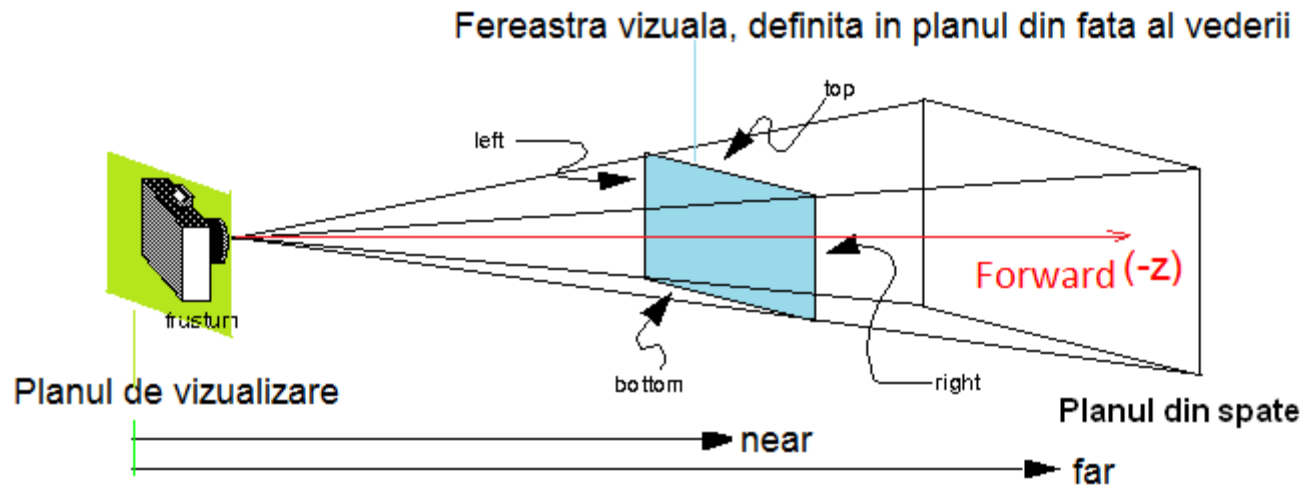
Proiectia ortografica

- Se efectueaza proiectia ortografica în fereastra vizuala a obiectelor cuprinse in volumul vizual. Primitivele grafice situate complet in afara volumului vizual sunt eliminate iar cele care intersecteaza volumul sunt decupate la frontiera volumului creându-se noi primitive.



Transformarea de proiecție (3)

Volumul vizual al proiecției perspective (View frustum): trunchi de piramidă delimitat în adâncime de planul din față (near) și planul din spate (far) al vederii, cu vârful în poziția observatorului.



Se efectuează proiecția perspectiva în planul din față (în fereastra vizuala), cu centrul de proiecție în poziția observatorului, a obiectelor cuprinse în volumul vizual. Obiectele situate în afara volumului vizual sunt eliminate sau decupate la frontiera volumului vizual.

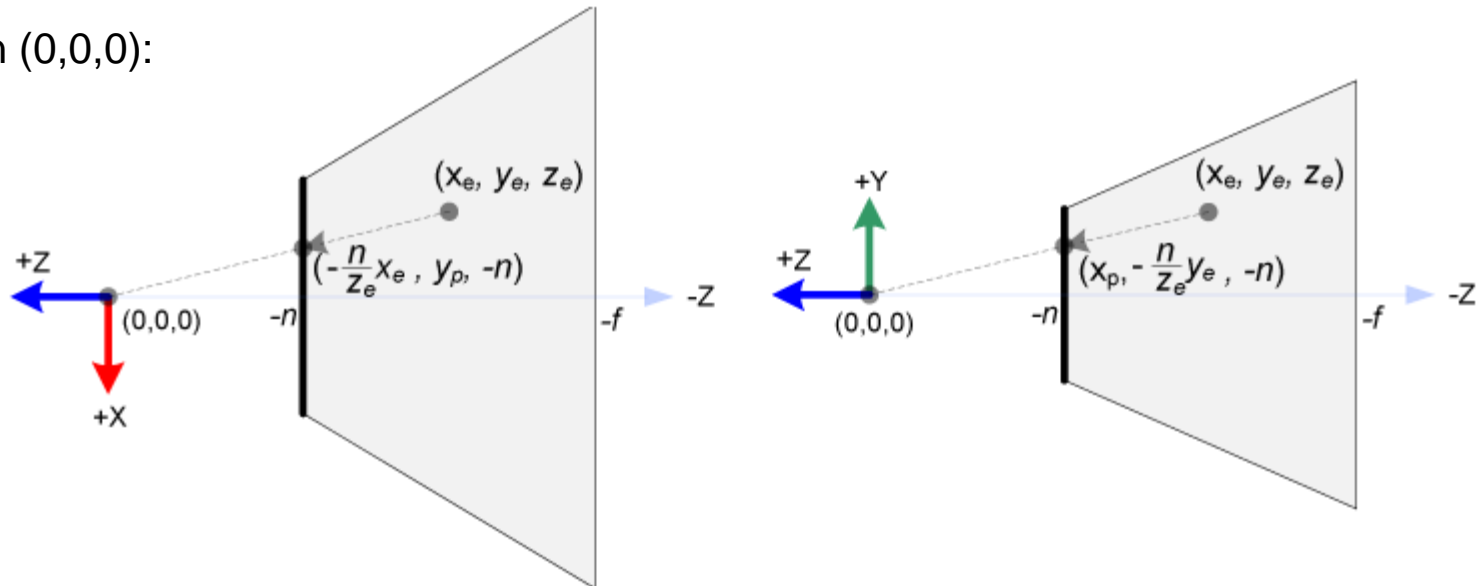
Transformarea de proiecție (4) - proiecția

- ❖ Proiecția se efectuează în fereastra vizuală din planul din față: planul $z = -\text{near}$ (în SCO).
- ❖ Varfurile proiectate sunt raportate la sist. coordonatelor observator.

Proiecția ortografică a unui punct din sist. coordonatelor observator, (x_e, y_e, z_e) , în fereastra vizuală:

$$x_p = x_e, y_p = y_e, z_p = -\text{near}.$$

Proiecția perspectivă a unui punct din SCO, (x_e, y_e, z_e) , în planul $z = -\text{near}$, cu centrul de proiecție în $(0,0,0)$:



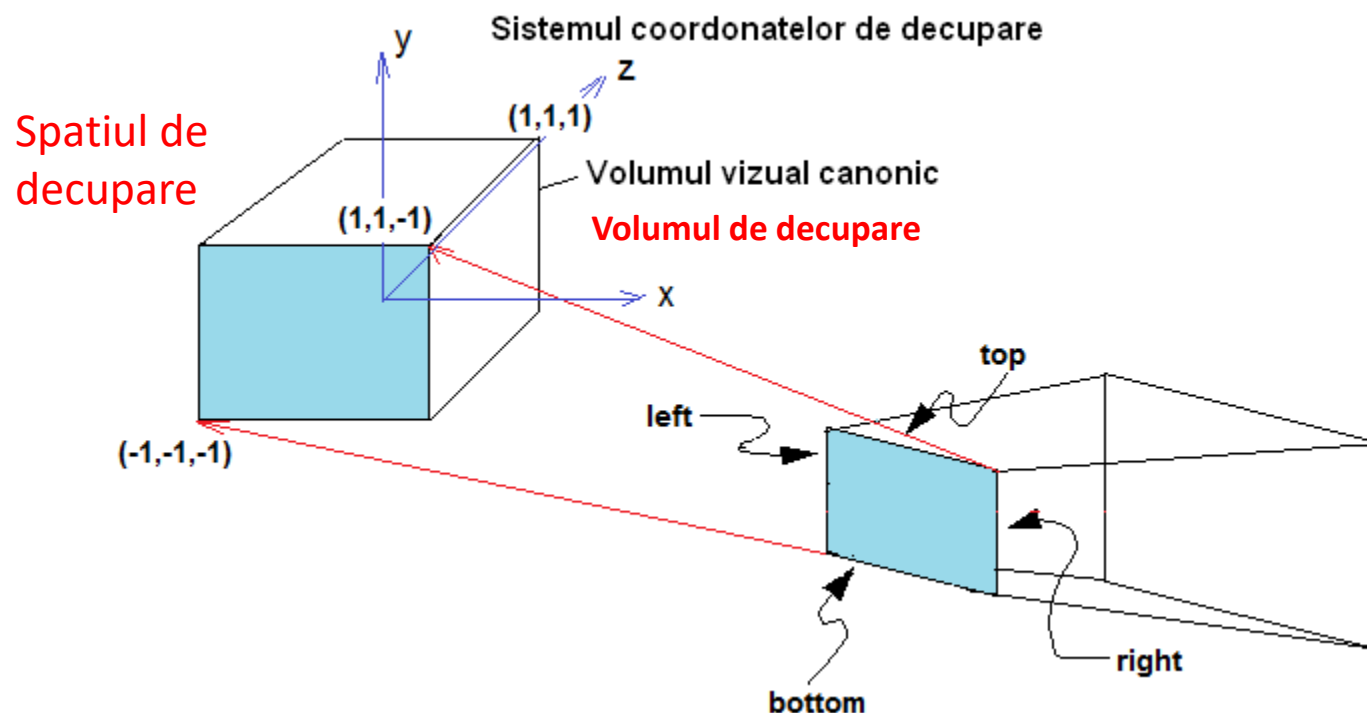
$x_p = (-\text{near}/z_e)*x_e$, $y_p = (-\text{near}/z_e)*y_e$ – se pot obtine aplicand formulele proiectiei perspectiva cu centrul de proiectie in $(0,0,0)$ si planul de proiectie $z = -\text{near}$.

Transformarea de proiecție (5) – Sistemul coordonatelor de decupare

- ❖ Transformarea de proiecție este însoțită de operația de decupare a primitivelor grafice la marginile volumului vizual.
- ❖ Pentru simplificarea calculelor de decupare, volumul vizual definit de programator este transformat în volumul vizual canonic: cub cu latura de 2 unități, centrat în originea sistemului coordonatelor de decupare, cu laturile paralele cu axele.
- ❖ Transformarea prin care volumul vizual este transformat în volumul vizual canonic se numește transformarea de normalizare.
- ❖ În OpenGL transformarea de proiecție include: proiecția definită prin volumul de vizualizare compusă cu transformarea de normalizare.
= transformarea coordonatelor observator în coordonate de decupare.
- ❖ După transformarea varfurilor prin transformarea de proiecție, varfurile sunt unite pe baza topologiei definite de programator, formând primitive grafice 3D. Acestea sunt decupate la marginile volumului vizual canonic.

Transformarea de proiecție (6) - normalizarea

- *Sistemul coordonatelor de decupare*: sistem de coordonate carteziane 3D stanga.
- **Prin transformarea de normalizare**:
 - colțurile ferestrei 2D din planul de proiecție, **(left, bottom, -near)** și **(right, top, -near)** sunt mapate pe colțurile stânga-jos și dreapta-sus ale ferestrei 2D din sistemul coordonatelor de decupare, adică **$(-1, -1, -1)$** și **$(1, 1, -1)$** ;
 - planul **$z = -near$** se mapeaza pe **$z = -1$** , iar **$z = -far$** se mapeaza pe **$z = 1$**



Transformarea de proiecție (7) calculul coordonatelor de decupare

- Impunand conditia $(\text{left}, \text{bottom}) \rightarrow (-1, -1)$ si $(\text{right}, \text{top}) \rightarrow (1, 1)$, rezulta transformarea punctelor proiectate pe planul din față, (x_p, y_p) , in sistemul coordonatelor de decupare – este o transformare fereastra-poarta:

$$x_d = 2 * x_p / (\text{right} - \text{left}) - (\text{right} + \text{left}) / (\text{right} - \text{left})$$

$$y_d = 2 * y_p / (\text{top} - \text{bottom}) - (\text{top} + \text{bottom}) / (\text{top} - \text{bottom})$$

Pentru proiectia perspectiva:

Inlocuind x_p si y_p prin expresiile lor in functie de (x_e, y_e, near) rezulta:

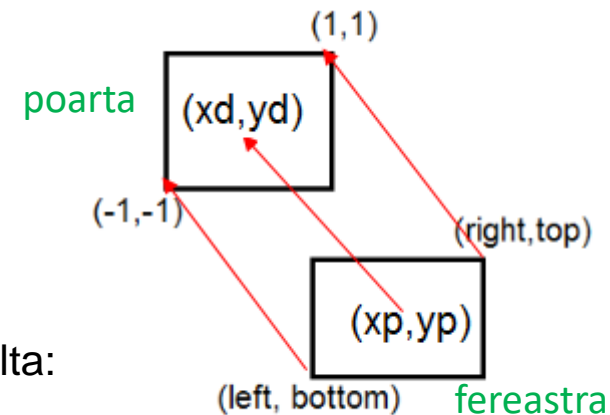
$$x_d = (x_e * 2 * \text{near} / (\text{right} - \text{left}) + z_e * (\text{right} + \text{left}) / (\text{right} - \text{left})) / (-z_e)$$

$$y_d = (y_e * 2 * \text{near} / (\text{top} - \text{bottom}) + z_e * (\text{top} + \text{bottom}) / (\text{top} - \text{bottom})) / (-z_e)$$

Pentru transformarea coordonatelor z se impun conditiile: $-\text{near} \rightarrow -1$, $-\text{far} \rightarrow 1$; rezulta:

$$z_d = (-z_e * (\text{far} + \text{near}) / (\text{far} - \text{near}) - 2 * \text{far} * \text{near} / (\text{far} - \text{near})) / (-z_e)$$

(x_d, y_d, z_d) sunt coordonate carteziane in sistemul coordonatelor de decupare (clip coordinates);



Matricile transformării de proiecție -1

Pentru proiecția perspectivă:

$$\text{Per} = \begin{bmatrix} \frac{2 * \text{near}}{\text{right} - \text{left}} & 0 & A & 0 \\ 0 & \frac{2 * \text{near}}{\text{top} - \text{bottom}} & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad \begin{aligned} A &= \frac{\text{right} + \text{left}}{\text{right} - \text{left}} & B &= \frac{\text{top} + \text{bottom}}{\text{top} - \text{bottom}} \\ C &= -\frac{\text{far} + \text{near}}{\text{far} - \text{near}} & D &= -\frac{2 * \text{far} * \text{near}}{\text{far} - \text{near}} \end{aligned}$$

$$[x_c \ y_c \ z_c \ w_c]^T = \text{Per} \cdot [x_e \ y_e \ z_e \ 1]^T \quad \begin{aligned} x_c &= x_d * w_c, \ y_c = y_d * w_c, \ z_c = z_d * w_c, \ w_c = -z_e \end{aligned}$$

Pentru proiecția ortografică:

$$x_p = x_e \text{ si } y_p = y_e \rightarrow x_d = 2 * x_e / (\text{right} - \text{left}) - (\text{right} + \text{left}) / (\text{right} - \text{left})$$

$$y_d = 2 * y_e / (\text{top} - \text{bottom}) - (\text{top} + \text{bottom}) / (\text{top} - \text{bottom})$$

$$\text{Par} = \begin{bmatrix} \frac{2}{\text{right} - \text{left}} & 0 & 0 & -A \\ 0 & \frac{2}{\text{top} - \text{bottom}} & 0 & -B \\ 0 & 0 & \frac{-2}{\text{far} - \text{near}} & C \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [x_c \ y_c \ z_c \ 1]^T = \text{Par} \cdot [x_e \ y_e \ z_e \ 1]^T$$

Matricile transformării de proiecție -2

Matricile de transformări de proiecție pot fi obținute apelând următoarele funcții din biblioteca glm:

`glm::ortho(float left, float right, float bottom, float top, float zNear, float zFar);`

`glm::frustum (float left, float right, float bottom, float top, float zNear, float zFar);`

Exemplu:

`glm::mat4 PerspProjectionMatrix = glm::frustum(2, 100, -2, 100, 0.1f, 1000.f);`

`glm::perspective (float fov, float aspect, float znear, float zfar);`

Creaza o matrice de proiecție pentru un volum perspectiva simetric:

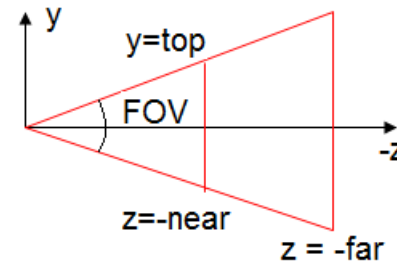
- fov: Field of View – unghi in grade

$$\tan(\text{FOV}/2) = \text{top}/\text{near}$$

- aspect: aspect ratio

$$\text{aspect ratio} = (\text{right-left})/(\text{top-bottom})$$

= raportul dintre latimea si inaltimea ferestrei vizuale



Împartirea perspectivă

Transformarea de proiecție: coordonate observator → coordonate de decupare

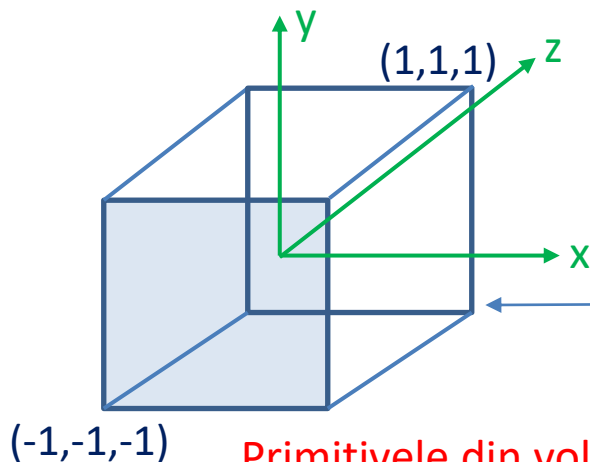
$$[x_c \ y_c \ z_c \ w_c]^T = \text{Par/Per} \cdot [x_e \ y_e \ z_e \ 1]^T \text{ (clip coordinates)} \leftarrow \text{eye coordinates}$$

Împartirea perspectivă: coordonate de decupare → coordonate dispozitiv normalizate

$$x_d = x_c/w_c, \ y_d = y_c/w_c, \ z_d = z_c/w_c$$

- Varfurile rezultate din împartirea perspectivă sunt asamblate în primitive grafice care sunt decupate la frontiera volumului vizual canonic.

Sistemul coordonatelor de decupare



După transformarea de proiecție, observatorul este la infinit pe axa OZ negativă a sistemului coord. de decupare.



Distanța față de observator se măsura pe axa OZ pozitivă.

Volumul de decupare

După decupare, coord. primitivelor: $-1 \leq x_d, y_d, z_d \leq 1$

Primitivele din volumul de decupare sunt proiectate ortografic pe fața “din față” a volumului: $(-1, -1, 1) - (1, 1, 1)$.

Transformarea în poarta de afișare

coordonate dispozitiv normalizate \rightarrow coordonate ecran

- Este o transformare fereasta-poarta, în care:
- **Fereastră** este fereastră 2D din sistemul coordonatelor de decupare având colțurile în $(-1, -1, -1)$ - $(1, 1, -1)$.

- **Poarta** este definită de programator folosind funcția:

`void glViewport(GLint px, GLint Py, GLsizei width, GLsizei height);`

(px, py) reprezintă coordonate relative la colțul stanga-jos al ferestrei aplicației (în pixeli). Valorile implicite : $(0,0)$.

width, height reprezintă lățimea/ înălțimea porții de afișare.

Valorile implicite sunt: lățimea și înălțimea ferestrei aplicației.

- **Transformarea în poarta de afișare este definită astfel:**

$$xv = (\text{width}/2)xd + ox, \quad ox = px + \text{width}/2$$

$$yv = (\text{height}/2)yd + oy, \quad oy = py + \text{height}/2$$

$$zv = ((f-n)/2)zd + (n+f)/2 \quad // \text{ conserva coordonata } z \text{ pentru eliminarea partilor nevizibile}$$

n (near) și f (far) au valorile implicite 0.0 respectiv 1.0, pentru care $0 \leq zv \leq 1$

n și f pot fi modificate la valori cuprinse în intervalul $[0,1]$ folosind funcția **DepthRange()**

