Grafica 3D - introducere.
Transformări de vizualizare și de proiecție

Mihai-Sorin Stupariu

Sem. al II-lea, 2022 - 2023

Obiecte 3D din biblioteca GLUT/GLU

cub, alte poliedre, sferă, con, tor, ceainic, cilindru
pot fi reprezentate “wire” sau “solid”

Problematizare

- Este necesară trecerea de la scena 3D (obiectele care se doresc a fi reprezentate), teoretic infinită, la imaginea 2D (primitivele care sunt randate), inclusă într-un dreptunghi cu dimensiuni date. Pentru a realiza acest lucru, sunt urmați doi pași. Acești pași corespund unor transformări specifice:

Problematizare

- Este necesară trecerea de la scena 3D (obiectele care se doresc a fi reprezentate), teoretic infinită, la imaginea 2D (primitivele care sunt randate), inclusă într-un dreptunghi cu dimensiuni date. Pentru a realiza acest lucru, sunt urmați doi pași. Acești pași corespund unor transformări specifice:
  - Este stabilit modul în care este vizualizată scena 3D (poziția observatorului), prin introducerea coordonatelor de vizualizare și schimbarea reperului. Funcția specifică din OpenGL este `gluLookAt();` - secțiunea 1.

Problematizare

- Este necesară trecerea de la scena 3D (obiectele care se doresc a fi reprezentate), teoretic infinită, la imaginea 2D (primitivele care sunt randate), inclusă într-un dreptunghi cu dimensiuni date. Pentru a realiza acest lucru, sunt urmați doi pași. Acești pași corespund unor transformări specifice:
  - Este stabilit modul în care este vizualizată scena 3D (poziția observatorului), prin introducerea coordonatelor de vizualizare și schimbarea reperului. Funcția specifică din OpenGL este `gluLookAt();` - secțiunea 1.
  - Este stabilit modul în care se realizează (i) decuparea (infinit $\longrightarrow$ finit); (ii) proiecția (3D $\longrightarrow$ 2D). În OpenGL sunt mai multe funcții specifice, depinzând de obiectivul urmărit: `gluOrtho2D();` `glOrtho();` `glFrustum();` `gluPerspective()` - secțiunea 2.

Coordonate de modelare și coordonate de vizualizare

- **Coordonatele de modelare**

Coordonate de modelare și coordonate de vizualizare

- **Coordonatele de modelare**
  - originea O

Coordonate de modelare și coordonate de vizualizare

- **Coordonatele de modelare**
  - originea O
  - axele de coordonate Ox, Oy, Oz cu versorii e_1, e_2, e_3

Coordonate de modelare și coordonate de vizualizare

- **Coordonatele de modelare**
  - originea O
  - axele de coordonate Ox, Oy, Oz cu versorii e_1, e_2, e_3
  - implicit, obiectele/primitivele (vârfurile) sunt indicate în raport cu acest sistem de coordonate

Coordonate de modelare și coordonate de vizualizare

- **Coordonatele de modelare**
  - originea O
  - axele de coordonate Ox, Oy, Oz cu versorii e_1, e_2, e_3
  - implicit, obiectele/primitivele (vârfurile) sunt indicate în raport cu acest sistem de coordonate
- Apelarea funcției `gluLookAt( );` are ca efect (implicit) generarea unui nou reper / sistem de coordonate, numite **reper de vizualizare / coordonate de vizualizare**

Coordonate de modelare și coordonate de vizualizare

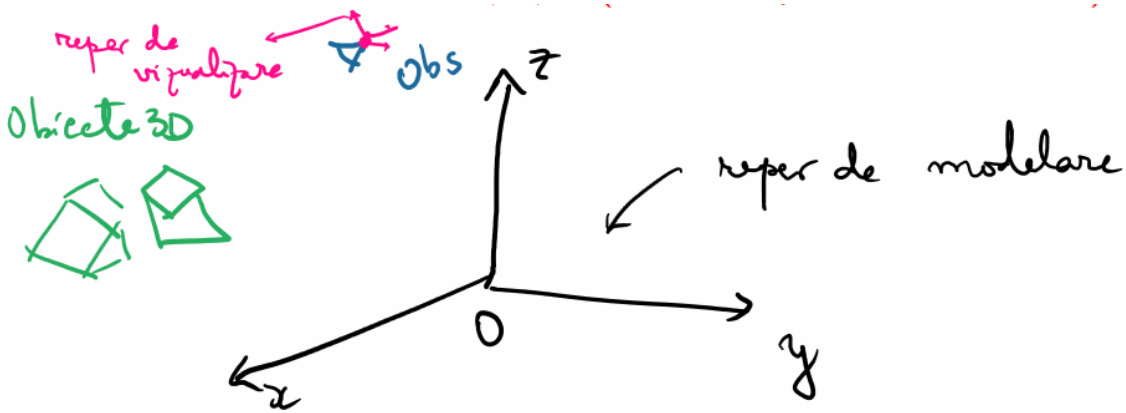
- **Coordonatele de modelare**
  - originea O
  - axele de coordonate Ox, Oy, Oz cu versorii e_1, e_2, e_3
  - implicit, obiectele/primitivele (vârfurile) sunt indicate în raport cu acest sistem de coordonate
- Apelarea funcției `gluLookAt( );` are ca efect (implicit) generarea unui nou reper / sistem de coordonate, numite **reper de vizualizare / coordonate de vizualizare**
  - originea: P_0 (poziția observatorului)

Coordonate de modelare și coordonate de vizualizare

- **Coordonatele de modelare**
  - originea O
  - axele de coordonate Ox, Oy, Oz cu versorii e_1, e_2, e_3
  - implicit, obiectele/primitivele (vârfurile) sunt indicate în raport cu acest sistem de coordonate
- Apelarea funcției `gluLookAt( );` are ca efect (implicit) generarea unui nou reper / sistem de coordonate, numite **reper de vizualizare / coordonate de vizualizare**
  - originea: P_0 (poziția observatorului)
  - axele: date de versorii u, v, n (construiți în continuare)

Coordonate de modelare și coordonate de vizualizare

- **Coordonatele de modelare**
  - originea O
  - axele de coordonate Ox, Oy, Oz cu versorii e_1, e_2, e_3
  - implicit, obiectele/primitivele (vârfurile) sunt indicate în raport cu acest sistem de coordonate
- Apelarea funcției `gluLookAt( );` are ca efect (implicit) generarea unui nou reper / sistem de coordonate, numite **reper de vizualizare / coordonate de vizualizare**
  - originea: P_0 (poziția observatorului)
  - axele: date de versorii u, v, n (construiți în continuare)



Funcția `gluLookAt( )`

- Pentru a înțelege funcția `gluLookAt( );`: care sunt elementele geometrice relevante atunci când vorbim despre observarea unei scene 3D? (de exemplu vederea umană sau folosirea unui aparat fotografic / telefon mobil).

Funcția `gluLookAt( )`

- Pentru a înțelege funcția `gluLookAt( );`: care sunt elementele geometrice relevante atunci când vorbim despre observarea unei scene 3D? (de exemplu vederea umană sau folosirea unui aparat fotografic / telefon mobil).
  - Poziția (coordonatele) observatorului

Funcția `gluLookAt( )`

- Pentru a înțelege funcția `gluLookAt( );`: care sunt elementele geometrice relevante atunci când vorbim despre observarea unei scene 3D? (de exemplu vederea umană sau folosirea unui aparat fotografic / telefon mobil).
  - Poziția (coordonatele) observatorului
  - Direcția / Punctul de referință (spre care este îndreptată privirea sau dispozitivul)

Funcția `gluLookAt( )`

- Pentru a înțelege funcția `gluLookAt( );`: care sunt elementele geometrice relevante atunci când vorbim despre observarea unei scene 3D? (de exemplu vederea umană sau folosirea unui aparat fotografic / telefon mobil).
  - Poziția (coordonatele) observatorului
  - Direcția / Punctul de referință (spre care este îndreptată privirea sau dispozitivul)
  - Orientarea

Funcția `gluLookAt( )`

- Pentru a înțelege funcția `gluLookAt( );`: care sunt elementele geometrice relevante atunci când vorbim despre observarea unei scene 3D? (de exemplu vederea umană sau folosirea unui aparat fotografic / telefon mobil).
  - Poziția (coordonatele) observatorului
  - Direcția / Punctul de referință (spre care este îndreptată privirea sau dispozitivul)
  - Orientarea
- Funcția `gluLookAt`

```
glMatrixMode (GL_MODELVIEW);
gluLookAt (x0, y0, z0, xref, yref, zref, Vx, Vy, Vz);
```

Funcția `gluLookAt( )`

- Pentru a înțelege funcția `gluLookAt( );`: care sunt elementele geometrice relevante atunci când vorbim despre observarea unei scene 3D? (de exemplu vederea umană sau folosirea unui aparat fotografic / telefon mobil).
  - Poziția (coordonatele) observatorului
  - Direcția / Punctul de referință (spre care este îndreptată privirea sau dispozitivul)
  - Orientarea
- Funcția `gluLookAt`

  `glMatrixMode (GL_MODELVIEW);`

  `gluLookAt` $(x_0, y_0, z_0, x_{ref}, y_{ref}, z_{ref}, V_x, V_y, V_z)$;
  - (x_0, y_0, z_0): coordonatele observatorului P_0 în reperul de modelare;

Funcția gluLookAt( )

- Pentru a înțelege funcția `gluLookAt( );`: care sunt elementele geometrice relevante atunci când vorbim despre observarea unei scene 3D? (de exemplu vederea umană sau folosirea unui aparat fotografic / telefon mobil).
  - Poziția (coordonatele) observatorului
  - Direcția / Punctul de referință (spre care este îndreptată privirea sau dispozitivul)
  - Orientarea
- Funcția `gluLookAt`

  `glMatrixMode (GL_MODELVIEW);`

  `gluLookAt` $(x_0, y_0, z_0, x_{ref}, y_{ref}, z_{ref}, V_x, V_y, V_z)$;
  - (x_0, y_0, z_0): coordonatele observatorului P_0 în reperul de modelare;
  - $(x_{ref}, y_{ref}, z_{ref})$: coordonatele unui punct de referință P_{ref} spre care se uită observatorul;

Funcția `gluLookAt( )`

- Pentru a înțelege funcția `gluLookAt( );`: care sunt elementele geometrice relevante atunci când vorbim despre observarea unei scene 3D? (de exemplu vederea umană sau folosirea unui aparat fotografic / telefon mobil).
  - Poziția (coordonatele) observatorului
  - Direcția / Punctul de referință (spre care este îndreptată privirea sau dispozitivul)
  - Orientarea
- Funcția `gluLookAt`

  `glMatrixMode (GL_MODELVIEW);`

  `gluLookAt` $(x_0, y_0, z_0, x_{ref}, y_{ref}, z_{ref}, V_x, V_y, V_z)$`;`
  - (x_0, y_0, z_0): coordonatele observatorului P_0 în reperul de modelare;
  - $(x_{ref}, y_{ref}, z_{ref})$: coordonatele unui punct de referință P_{ref} spre care se uită observatorul;
  - (V_x, V_y, V_z): vector care indică verticala din planul de vizualizare

Funcția `gluLookAt( )`

- Pentru a înțelege funcția `gluLookAt( );`: care sunt elementele geometrice relevante atunci când vorbim despre observarea unei scene 3D? (de exemplu vederea umană sau folosirea unui aparat fotografic / telefon mobil).
  - Poziția (coordonatele) observatorului
  - Direcția / Punctul de referință (spre care este îndreptată privirea sau dispozitivul)
  - Orientarea
- Funcția `gluLookAt`

  `glMatrixMode (GL_MODELVIEW);`

  `gluLookAt` $(x_0, y_0, z_0, x_{ref}, y_{ref}, z_{ref}, V_x, V_y, V_z)$`;`
  - (x_0, y_0, z_0): coordonatele observatorului P_0 în reperul de modelare;
  - $(x_{ref}, y_{ref}, z_{ref})$: coordonatele unui punct de referință P_{ref} spre care se uită observatorul;
  - (V_x, V_y, V_z): vector care indică verticala din planul de vizualizare
- Implicit: $P_0 = (0, 0, 0), P_{ref} = (0, 0 - 1), V = (0, 1, 0)$

Funcția gluLookAt( )

- Pentru a înțelege funcția `gluLookAt( );`: care sunt elementele geometrice relevante atunci când vorbim despre observarea unei scene 3D? (de exemplu vederea umană sau folosirea unui aparat fotografic / telefon mobil).
  - Poziția (coordonatele) observatorului
  - Direcția / Punctul de referință (spre care este îndreptată privirea sau dispozitivul)
  - Orientarea
- Funcția `gluLookAt`

  `glMatrixMode (GL_MODELVIEW);`

  `gluLookAt` $(x_0, y_0, z_0, x_{ref}, y_{ref}, z_{ref}, V_x, V_y, V_z)$;
  - (x_0, y_0, z_0): coordonatele observatorului P_0 în reperul de modelare;
  - $(x_{ref}, y_{ref}, z_{ref})$: coordonatele unui punct de referință P_{ref} spre care se uită observatorul;
  - (V_x, V_y, V_z): vector care indică verticala din planul de vizualizare
- Implicit: $P_0 = (0, 0, 0), P_{ref} = (0, 0 - 1), V = (0, 1, 0)$
- În continuare: construirea reperului de vizualizare pornind de la argumentele funcției `gluLookAt( );`

Funcția gluLookAt( )

- Pentru a înțelege funcția `gluLookAt( );`: care sunt elementele geometrice relevante atunci când vorbim despre observarea unei scene 3D? (de exemplu vederea umană sau folosirea unui aparat fotografic / telefon mobil).
  - Poziția (coordonatele) observatorului
  - Direcția / Punctul de referință (spre care este îndreptată privirea sau dispozitivul)
  - Orientarea
- Funcția `gluLookAt`

  `glMatrixMode (GL_MODELVIEW);`

  `gluLookAt` $(x_0, y_0, z_0, x_{ref}, y_{ref}, z_{ref}, V_x, V_y, V_z)$;
  - (x_0, y_0, z_0): coordonatele observatorului P_0 în reperul de modelare;
  - $(x_{ref}, y_{ref}, z_{ref})$: coordonatele unui punct de referință P_{ref} spre care se uită observatorul;
  - (V_x, V_y, V_z): vector care indică verticala din planul de vizualizare
- Implicit: $P_0 = (0, 0, 0), P_{ref} = (0, 0 - 1), V = (0, 1, 0)$
- În continuare: construirea reperului de vizualizare pornind de la argumentele funcției `gluLookAt( );`
- Originea reperului: $P_0 = (x_0, y_0, z_0)$; axele date de u, v, n

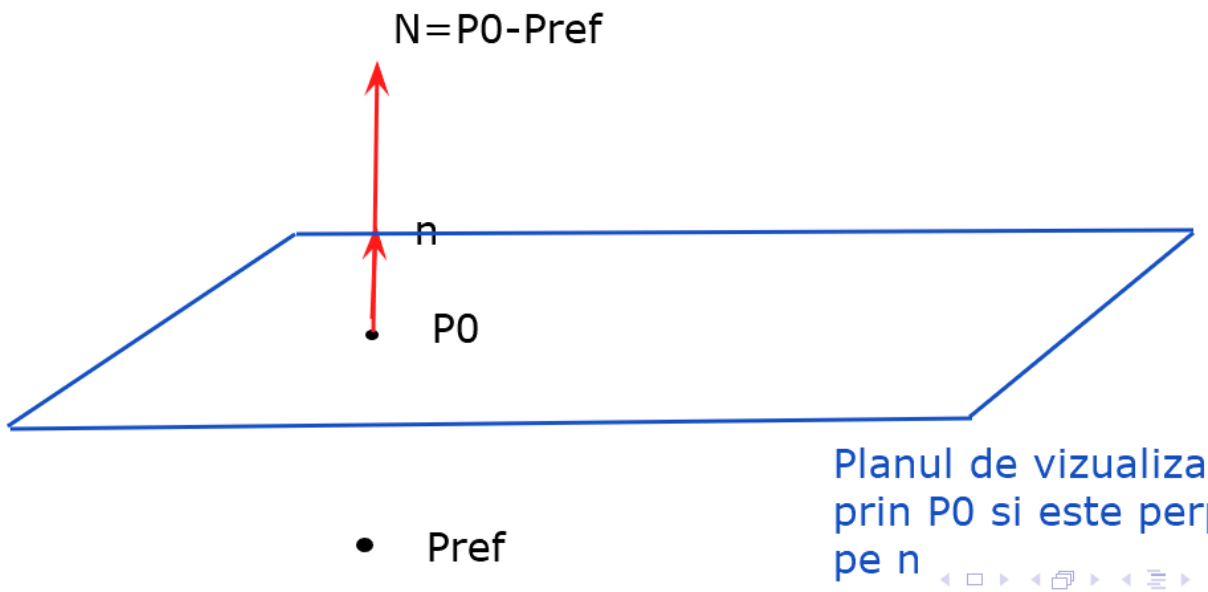
Vectorul n

$$N = \overrightarrow{P_{ref} P_0} = P_0 - P_{ref}; \quad \mathrm{n} = \frac{N}{\|N\|}$$

Vectorul n

$$N = \overrightarrow{P_{ref}P_0} = P_0 - P_{ref}; \quad \mathrm{n} = \frac{N}{\|N\|}$$

al treilea versor al reperului de vizualizare (n) este dat de observator și de punctul de referință (de ce $P_0 - P_{ref}$ și nu invers?); s-a efectuat împărțirea la N pentru a obține un vector de normă 1.



Vectorii v, u

- **primul versor** u - direcționează orizontala din planul de vizualizare: este perpendicular pe vectorul n (ca să fie inclus în planul de vizualizare) și este perpendicular pe vectorul V indicat în `gluLookAt`

$$\mathrm{u} = \frac{V \times \mathrm{n}}{\|V\|}$$

Vectorii v, u

- **primul versor** u - direcționează orizontala din planul de vizualizare: este perpendicular pe vectorul n (ca să fie inclus în planul de vizualizare) și este perpendicular pe vectorul V indicat în `gluLookAt`

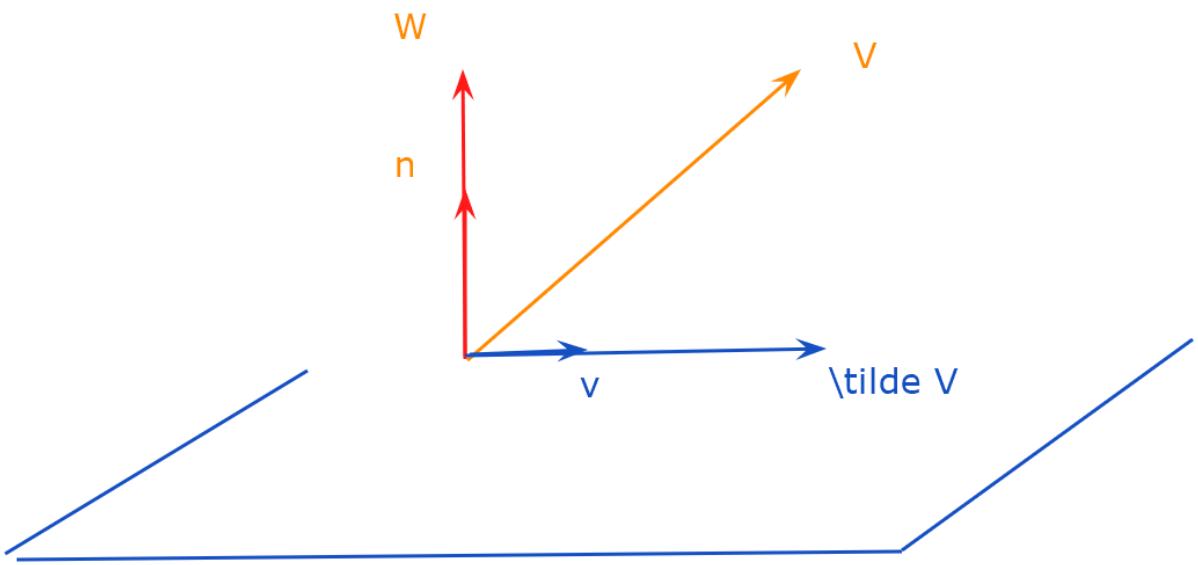
$$\mathrm{u} = \frac{V \times \mathrm{n}}{\|V\|}$$

- **al doilea versor** v - verticala "reală" din planul de vizualizare

$$\mathrm{v} = \mathrm{n} \times \mathrm{u}$$

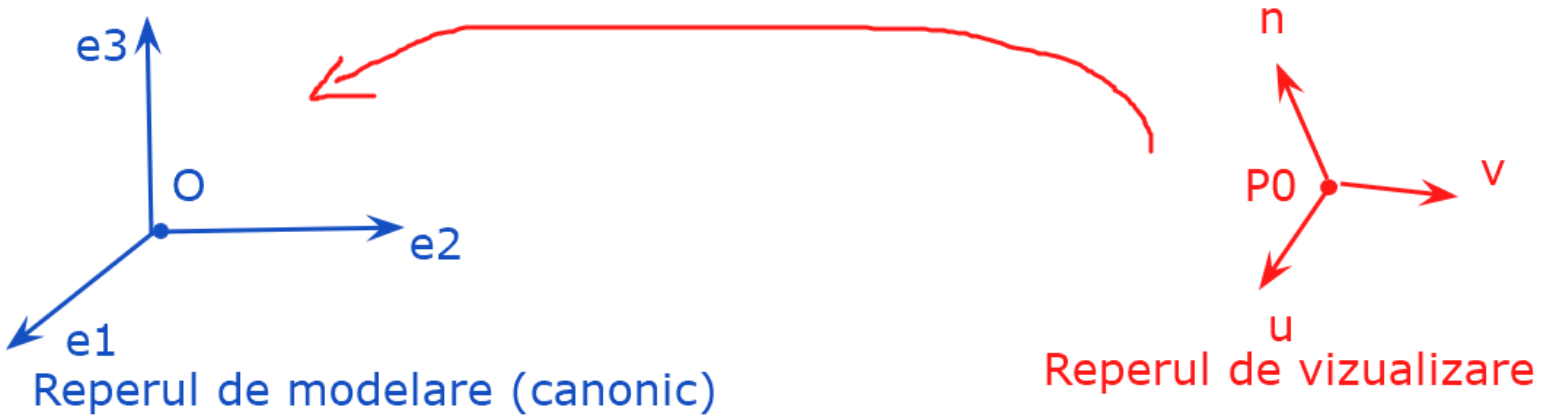
Legătura dintre vectorii V și v

Comentariu: ce legătură există între vectorul V, indicat ca "verticală" în funcția `gluLookAt ( );` și vectorul v, calculat ca fiind al doilea versor al reperului de vizualizare? **R:** Vectorul V se descompune ca suma dintre un vector $\tilde{V}$ (=proiecția lui V pe planul de vizualizare) și un vector W, perpendicular pe planul de vizualizare (coliniar cu n). Are loc relația $v = \frac{\tilde{V}}{\|\tilde{V}\|}$. De exemplu, în codul sursă `07_C_1_obiecte3d.cpp` avem $N = (6, 9, 10) = P_0 - P_{ref}$ și $V = (0, 0, 1)$. Adăugând la V vectori de forma αN, verticala v (implementată efectiv) din planul de vizualizare nu se modifică (deoarece este modificat doar W, nu și $\tilde{V}$).



Schimbarea reperului ca transformare

Apelarea funcției `gluLookAt` are ca efect:
- generarea unui nou reper (de vizualizare): $(P_0; u, v, n)$;
- generarea matricei 4×4 asociate transformării care "transportă" reperul de vizualizare nou construit în reperul canonic de modelare $(0; e_1, e_2, e_3)$ și utilizarea acestei matrice în stiva de matrice de modelare/transformare `GL_MODELVIEW` – scop: realizarea proiecției (v. proiecții!)



Determinarea matricei 4×4 prin care reperul de vizualizare este "transformat" în reperul de modelare

- **1. Translație** $P_0 \mapsto O$; matricea 4×4

$$T = \begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Determinarea matricei 4×4 prin care reperul de vizualizare este "transformat" în reperul de modelare

- **1. Translație** $P_0 \mapsto O$; matricea 4×4

$$T = \begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- **2. Rotație**, astfel ca reperul ortonormat (u, v, n) să fie transformat în reperul canonic (e_1, e_2, e_3). Matricea asociată este

$$R = \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Determinarea matricei 4×4 prin care reperul de vizualizare este "transformat" în reperul de modelare

- **1. Translație** $P_0 \mapsto O$; matricea 4×4

$$T = \begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- **2. Rotație**, astfel ca reperul ortonormat $(\mathrm{u}, \mathrm{v}, \mathrm{n})$ să fie transformat în reperul canonic (e_1, e_2, e_3). Matricea asociată este

$$R = \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Matricea căutată este

$$M = R \cdot T = \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} u_x & u_y & u_z & - < \mathrm{u}, P_0 > \\ v_x & v_y & v_z & - < \mathrm{v}, P_0 > \\ n_x & n_y & n_z & - < \mathrm{n}, P_0 > \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Despre matricea de rotație

- Este cunoscută matricea 3×3 prin care reperul (e_1, e_2, e_3) este transformat în reperul (u, v, n):

$$A = \begin{pmatrix} u_x & v_x & n_x \\ u_y & v_y & n_y \\ u_z & v_z & n_z \end{pmatrix}$$

(coloanele matricei sunt componentele vectorilor u, v, n în reperul canonic).

Despre matricea de rotație

- Este cunoscută matricea 3×3 prin care reperul (e_1, e_2, e_3) este transformat în reperul (u, v, n):

$$A = \begin{pmatrix} u_x & v_x & n_x \\ u_y & v_y & n_y \\ u_z & v_z & n_z \end{pmatrix}$$

  (coloanele matricei sunt componentele vectorilor u, v, n în reperul canonic).

- Matricea care transformă reperul (u, v, n) în reperul (e_1, e_2, e_3) este A^{-1} (inversa).

Despre matricea de rotație

- Este cunoscută matricea 3×3 prin care reperul (e_1, e_2, e_3) este transformat în reperul (u, v, n):

$$A = \begin{pmatrix} u_x & v_x & n_x \\ u_y & v_y & n_y \\ u_z & v_z & n_z \end{pmatrix}$$

(coloanele matricei sunt componentele vectorilor u, v, n în reperul canonic).

- Matricea care transformă reperul (u, v, n) în reperul (e_1, e_2, e_3) este A^{-1} (inversa).
- **Obs. fundmentală.** Deoarece reperul (u, v, n) este ortonormat, are loc relația

$$A^T \cdot A = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{pmatrix} \cdot \begin{pmatrix} u_x & v_x & n_x \\ u_y & v_y & n_y \\ u_z & v_z & n_z \end{pmatrix} = \mathbb{I}_3,$$

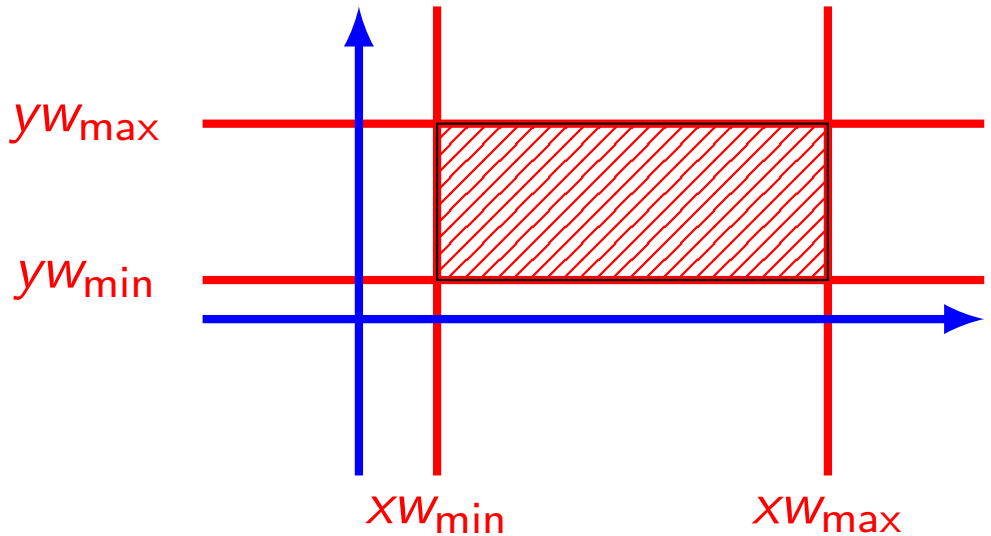
deci $A^{-1} = A^T$ (A s.n. matrice ortogonală), de unde se deduce forma matricei R.

Cazul 2D

```
glMatrixMode (GL_PROJECTION);
gluOrtho2D (xwmin, xwmax, ywmin, ywmax);
```

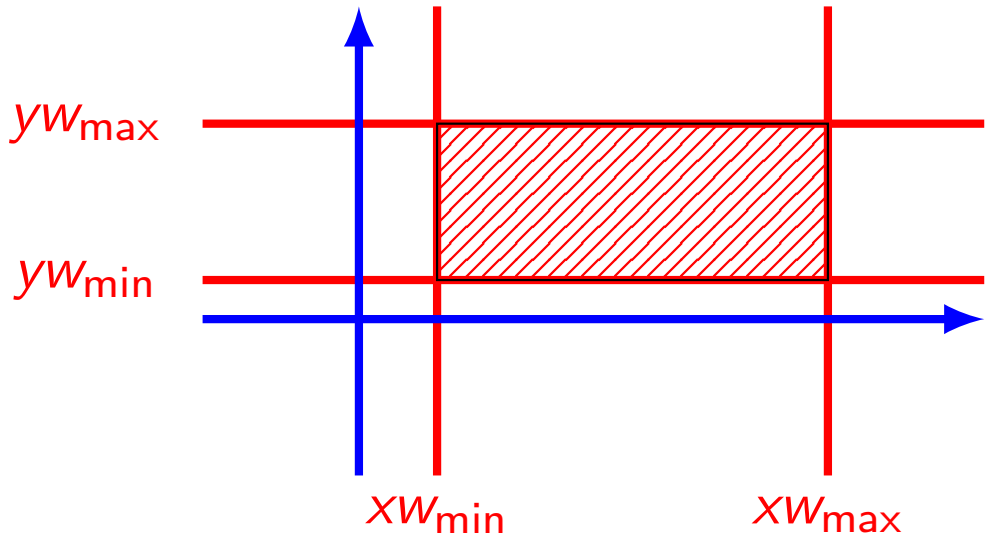
Cazul 2D

```
glMatrixMode (GL_PROJECTION);
gluOrtho2D (xwmin, xwmax, ywmin, ywmax);
```



Cazul 2D

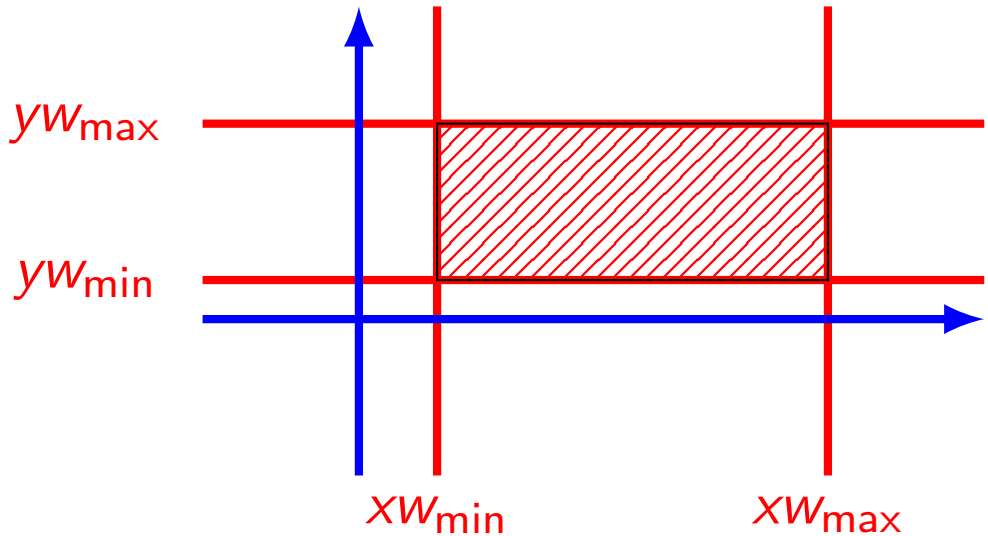
```
glMatrixMode (GL_PROJECTION);
gluOrtho2D (xwmin, xwmax, ywmin, ywmax);
```



Valorile implicite sunt $xwmin = ywmin = -1, xwmax = ywmax = 1$

Cazul 2D

```
glMatrixMode (GL_PROJECTION);
gluOrtho2D (xwmin, xwmax, ywmin, ywmax);
```



Valorile implicite sunt $xwmin = ywmin = -1, xwmax = ywmax = 1$

La apelarea funcției de mai sus, dreptunghiul indicat în funcție este transformat în pătratul "normalizat" $[-1, 1] \times [-1, 1]$, **apoi** se efectuează scalarea la fereastra indicată

Cazul 2D - exemple

- Care este aria dreptunghiului decupat dacă se apelează funcția `gluOrtho2D (a, b, c, d)`? ($a < b, c < d$).

Cazul 2D - exemple

- Care este aria dreptunghiului decupat dacă se apelează funcția `gluOrtho2D (a, b, c, d)`? ($a < b, c < d$).
- Ce diferențe sunt (din punctul de vedere al (i) dimensiunii scenei decupate, (ii) obiectelor - dimensiune, etc.) între apelarea funcției `gluOrtho2D (a, b, c, d)` și apelarea funcției `gluOrtho2D (2a, 2b, 2c, 2d)`? ($a < b, c < d$)

Transformări de proiecție în cazul 3D

(i) Proiecții ortogonale

Transformări de proiecție în cazul 3D

(i) Proiecții ortogonale

(ii) Proiecții perspective

Transformări de proiecție în cazul 3D

(i) Proiecții ortogonale

(ii) Proiecții perspective

(iii) Proiecții paralele generale

Transformări de proiecție în cazul 3D

(i) Proiecții ortogonale
(ii) Proiecții perspective
(iii) Proiecții paralele generale
(iv) Proiecții oblice

Transformări de proiecție în cazul 3D

(i) Proiecții ortogonale

(ii) Proiecții perspective

(iii) Proiecții paralele generale

(iv) Proiecții oblice

- Pentru (i) și (ii) există funcții specifice în OpenGL

Proiecții ortogonale

- Funcții asociate:

```
glMatrixMode (GL_PROJECTION);
glOrtho (xwmin, xwmax, ywmin, ywmax, dnear, dfar);
```

Proiecții ortogonale

- Funcții asociate:
```
glMatrixMode (GL_PROJECTION);
glOrtho (xwmin, xwmax, ywmin, ywmax, dnear, dfar);
```
- Coordonatele sunt cele de modelare; apelarea matricei asociate are loc după ce reperul de vizualizare a fost transformat în reperul de modelare, deci P_0 a devenit origine, iar axele sistemului sunt date de vectorii u, v, n.

Proiecții ortogonale

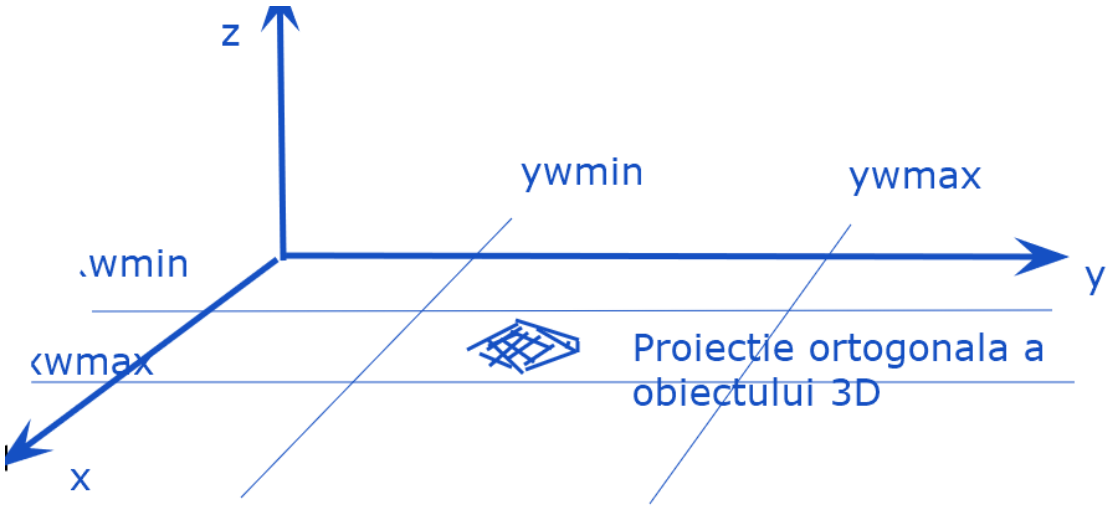
- Funcții asociate:
  ```
  glMatrixMode (GL_PROJECTION);
  glOrtho (xwmin, xwmax, ywmin, ywmax, dnear, dfar);
  ```
- Coordonatele sunt cele de modelare; apelarea matricei asociate are loc după ce reperul de vizualizare a fost transformat în reperul de modelare, deci P_0 a devenit origine, iar axele sistemului sunt date de vectorii u, v, n.
- Este decupat un paralelipiped delimitat de planele $x = xw_{\min}, x = xw_{\max}$; $y = yw_{\min}, y = yw_{\max}$, respectiv $z = z_{\text{near}}$, unde $z_{\text{near}} = -d_{\text{near}}$, $z = z_{\text{far}}$, unde $z_{\text{far}} = -d_{\text{far}}$.

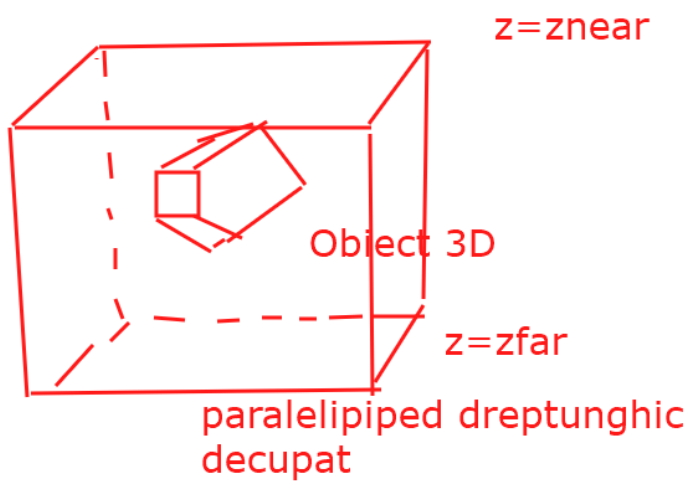
Proiecții ortogonale

- Funcții asociate:
  ```
  glMatrixMode (GL_PROJECTION);
  glOrtho (xwmin, xwmax, ywmin, ywmax, dnear, dfar);
  ```
- Coordonatele sunt cele de modelare; apelarea matricei asociate are loc după ce reperul de vizualizare a fost transformat în reperul de modelare, deci P_0 a devenit origine, iar axele sistemului sunt date de vectorii u, v, n.
- Este decupat un paralelipiped delimitat de planele $x = xw_{\min}, x = xw_{\max}$; $y = yw_{\min}, y = yw_{\max}$, respectiv $z = z_{\text{near}}$, unde $z_{\text{near}} = -d_{\text{near}}$, $z = z_{\text{far}}$, unde $z_{\text{far}} = -d_{\text{far}}$.
- Valorile implicite sunt $-1.0, 1.0, -1.0, 1.0, 0.0, 1.0$ (valori normalizate), iar pentru valori arbitrare se efectuează normalizarea (aducerea parametrilor indicați la valorile implicite), care este o scalare, iar matricea 4×4 asociată este

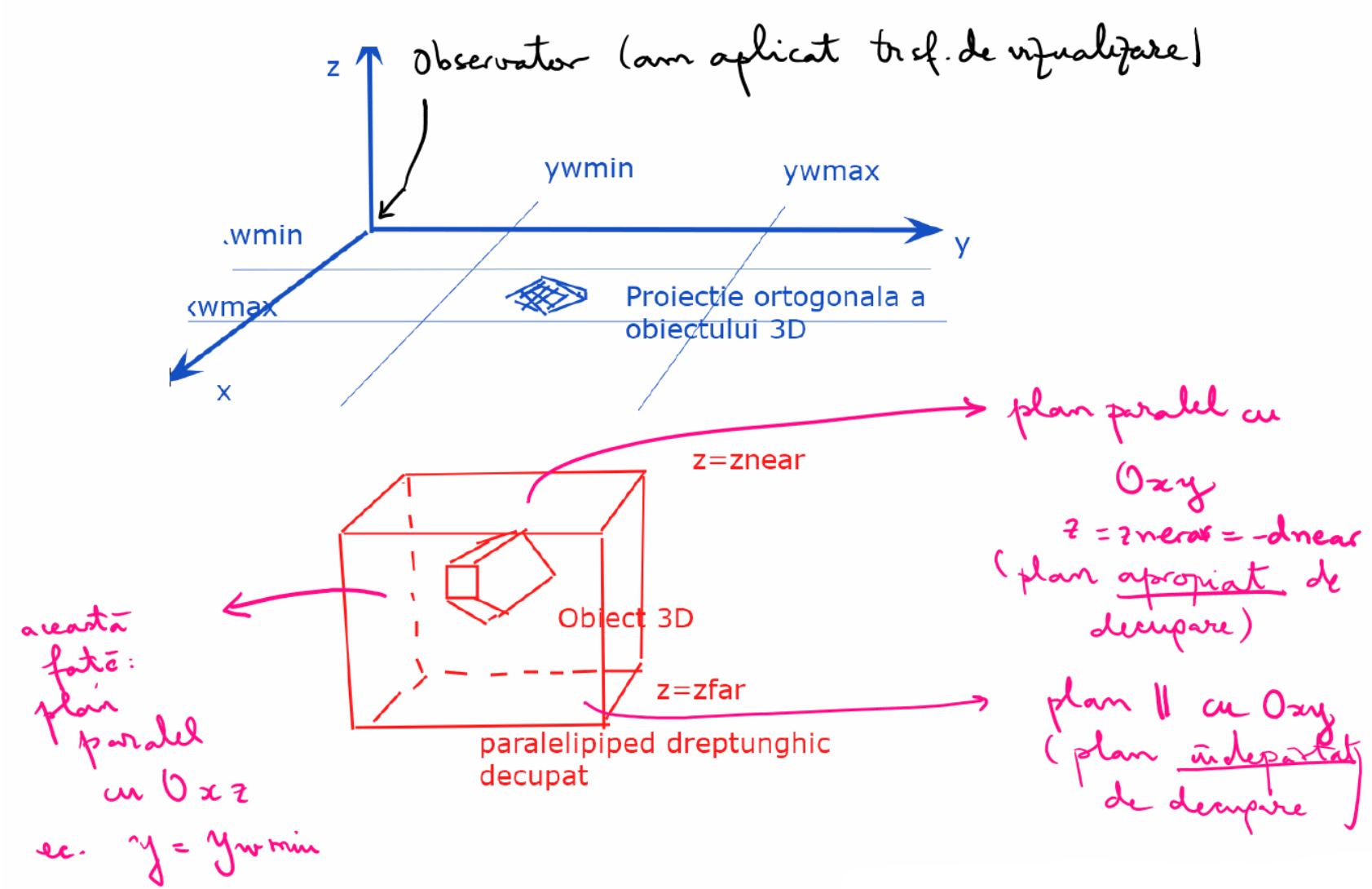
$$\mathcal{M}_{\text{orto,norm}} = \begin{pmatrix} \frac{2}{xw_{\max} - xw_{\min}} & 0 & 0 & -\frac{xw_{\max} + xw_{\min}}{xw_{\max} - xw_{\min}} \\ 0 & \frac{2}{yw_{\max} - yw_{\min}} & 0 & -\frac{yw_{\max} + yw_{\min}}{yw_{\max} - yw_{\min}} \\ 0 & 0 & -\frac{2}{z_{\text{near}} - z_{\text{far}}} & \frac{z_{\text{near}} + z_{\text{far}}}{z_{\text{near}} - z_{\text{far}}} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (1)$$

Proiecții ortogonale - figura

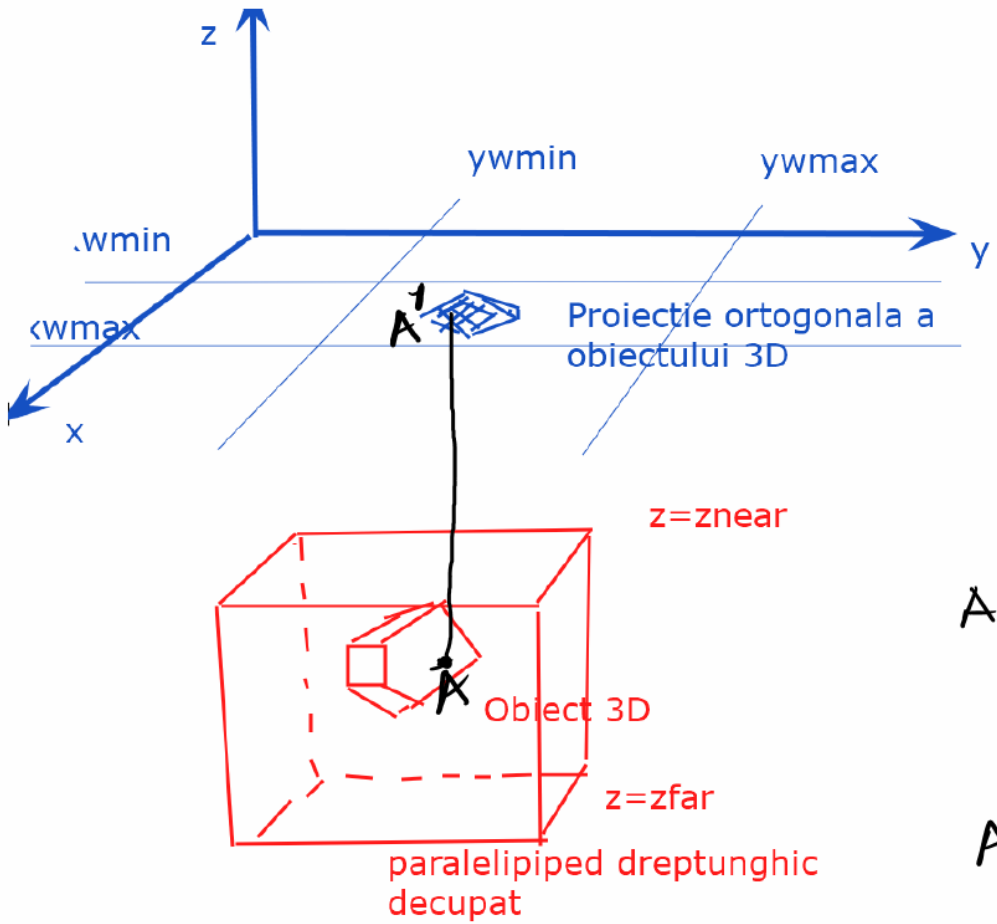




Proiecții ortogonale - figura



Proiecții ortogonale - figura



Orice vârf (punct) A este proiectat ortogonal în A' (se duce prin A ⊥ pe Oxy, A' este intersecția dintre această ⊥ și Oxy)

În coordonate

$A = (x_A, y_A, z_A)$

$\downarrow$

$A' = (x_A, y_A, 0)$

$\equiv$

(x_A, y_A)

Proiecții ortogonale - comentarii

- De discutat rolul elementelor care definesc paralelipipedul dreptunghic decupat.

Proiecții ortogonale - comentarii

- De discutat rolul elementelor care definesc paralelipipedul dreptunghic decupat.
- De testat pe codul sursă `07_C_1_obiecte3d.cpp` cum este realizată proiecția dacă modificăm diverși parametri ai funcției `glOrtho( );`. De urmărit: (i) cum este realizată decuparea; (ii) cum arată obiectele randate. De exemplu: ce se întâmplă dacă modificăm `dnear`?

Proiecții perspective

- Funcție asociată (piramidă oarecare):

```
glMatrixMode (GL_PROJECTION);
glFrustum (xwmin, xwmax, ywmin, ywmax, dnear, dfar);
```

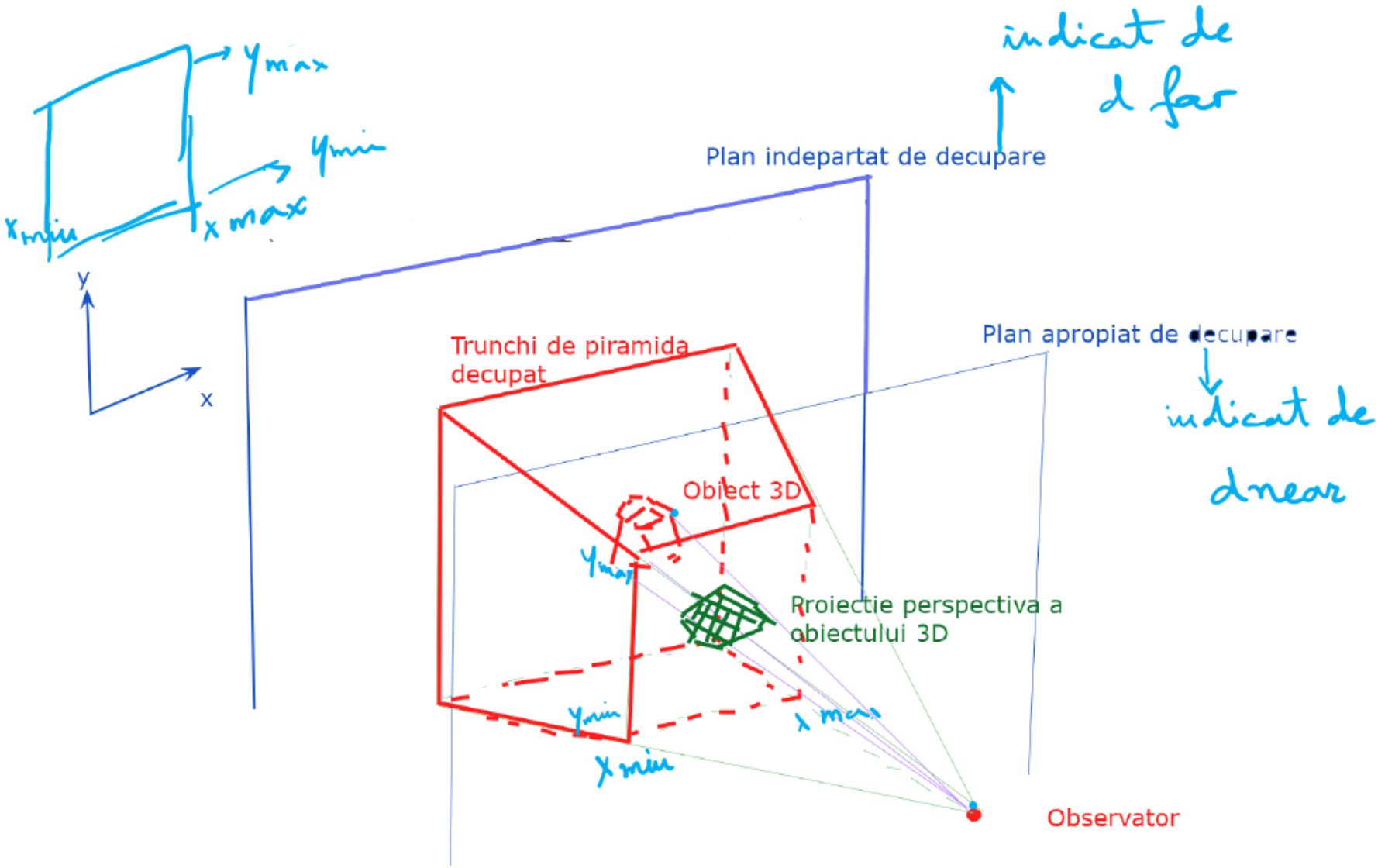
Proiecții perspective

- Funcție asociată (piramidă oarecare):
```
glMatrixMode (GL_PROJECTION);
glFrustum (xwmin, xwmax, ywmin, ywmax, dnear, dfar);
```
- **Important:** se delimitează dreptunghiul din planul apropiat, având ecuațiile dreptelor de forma $z = z_{\text{near}}, x = xw_{\text{min}}$, etc., **apoi** se determină dreptunghiul din planul îndepărtat și trunchiul de piramidă

Proiecții perspective

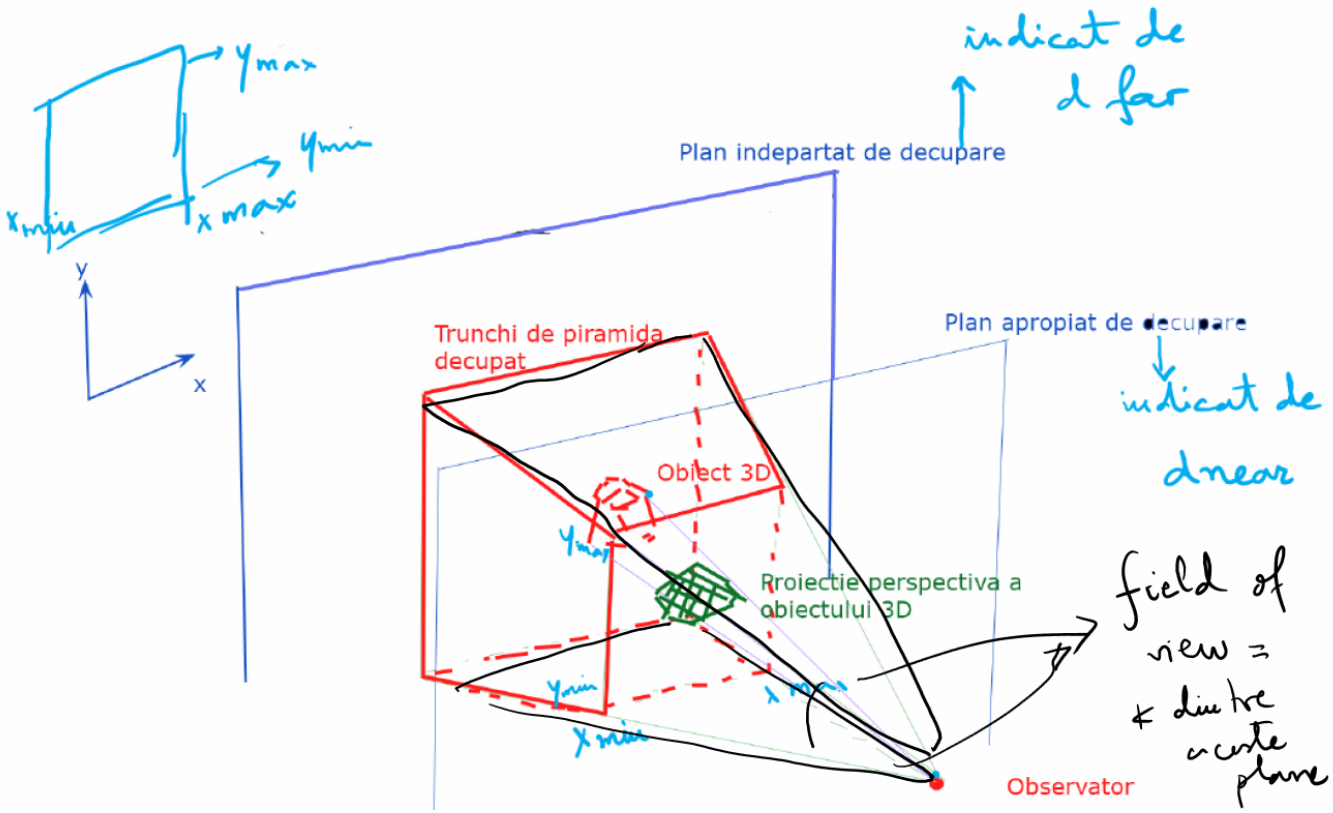
- Funcție asociată (piramidă oarecare):
```
glMatrixMode (GL_PROJECTION);
glFrustum (xwmin, xwmax, ywmin, ywmax, dnear, dfar);
```
- **Important:** se delimitează dreptunghiul din planul apropiat, având ecuațiile dreptelor de forma $z = z_{\text{near}}, x = xw_{\text{min}}$, etc., **apoi** se determină dreptunghiul din planul îndepărtat și trunchiul de piramidă
- Funcții asociate (piramidă cu piciorul înălțimii în centrul bazei, care este un dreptunghi):
```
glMatrixMode (GL_PROJECTION);
gluPerspective (fov, aspect, dnear, dfar);
```

Proiecții perspective - figura



Proiecții perspective - figura

În cazul funcției `gluPerspective( );` sunt considerați ca parametri `fov`=*field of view*, un unghi între plane și `aspect`=raportul dintre lungimile laturilor dreptunghiului decupat.



Proiecții perspective - comentarii

- De discutat rolul elementelor care definesc trunchiul de piramidă decupat. De comentat diferența dintre `glFrustum( );` - se poate decupa un trunchi de piramidă arbitrar și `gluPerspective( );` - se poate decupa un trunchi de piramidă care provine dintr-o piramidă având baza un dreptunghi, iar piciorul înălțimii piramidei (dusă din vârf) coincide cu centrul bazei. În cazul funcției `gluPerspective( );` sunt considerați ca parametri `fov`=*field of view*, un unghi între plane și `aspect`=raportul dintre lungimile laturilor dreptunghiului decupat.

Proiecții perspective - comentarii

- De discutat rolul elementelor care definesc trunchiul de piramidă decupat. De comentat diferența dintre `glFrustum( );` - se poate decupa un trunchi de piramidă arbitrar și `gluPerspective( );` - se poate decupa un trunchi de piramidă care provine dintr-o piramidă având baza un dreptunghi, iar piciorul înălțimii piramidei (dusă din vârf) coincide cu centrul bazei. În cazul funcției `gluPerspective( );` sunt considerați ca parametri `fov`=*field of view*, un unghi între plane și `aspect`=raportul dintre lungimile laturilor dreptunghiului decupat.
- De testat pe codul sursă `07_C_1_obiecte3d.cpp` cum este realizată proiecția dacă modificăm diverși parametri ai funcției `glFrustum( );`. De urmărit: (i) cum este realizată decuparea; (ii) cum arată obiectele randate. De exemplu: ce se întâmplă dacă modificăm `dnear`?
  Aceleași întrebări pentru `gluPerspective ( );`

Aplicație - codul sursă `08_C_1_aplicatie.cpp`

Se aplică funcția `gluLookAt(3,5,7,1,5,7,0,0,1)`. Este desenat triunghiul determinat de vârfurile $A(0,3,7), B(0,7,7), C(0,4,9)$. Se presupune că se aplică o proiecție ortogonală cu parametri adecvați (adică, după aplicarea acesteia, triunghiul este desenat complet). Să se arate că în randare triunghiul are o latură orizontală și să se stabilească dacă cel de-al treilea vârf este reprezentat deasupra sau dedesubtul acestei laturi.

Aplicație - codul sursă 08_C_1_aplicatie.cpp

- "interpretăm" funcția gluLookAt( ):
  - observator: $P_0 = (3, 5, 7)$
  - pct de referință: $P_{ref} = (1, 5, 7)$
  - verticala "propusă": $V = (0, 0, 1)$

- reperul de vizualizare:
  - $N = P_0 - P_{ref} = (2, 0, 0) \rightarrow n = (1, 0, 0)$
  - orizontala din planul de vizualizare:

$$u = \frac{V \times n}{\|V\|} = (0, 1, 0)$$

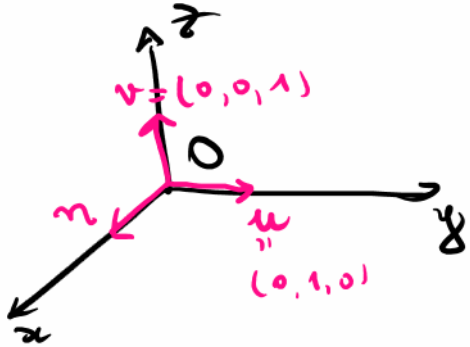
  - verticala din planul de vizualizare $v = n \times u = (0, 0, 1)$

(în exemplu: v coincide cu V)

Aplicație - codul sursă 08_C_1_aplicatie.cpp

- Înțelegerea contextului geometric

Planul de vizualizare (pe care sunt proiectate obiectele) este generat de vectorii $u = (0, 1, 0)$ și $v = (0, 0, 1)$, deci este paralel cu Oyz. Proiecția este ortogonală și se realizează de-a lungul axei Ox.

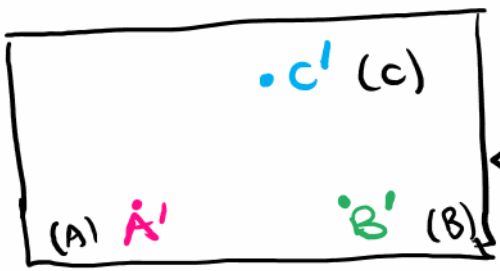


Cele trei vârfuri vor fi proiectate ortogonal astfel:

$A = (0, 3, 7) \longmapsto A' \equiv (3, 7)$

$B = (0, 7, 7) \longmapsto B' \equiv (7, 7)$

$C = (0, 4, 9) \longmapsto C' \equiv (4, 9)$



← randare: [AB] orizontală, C deasupra lui [AB]