

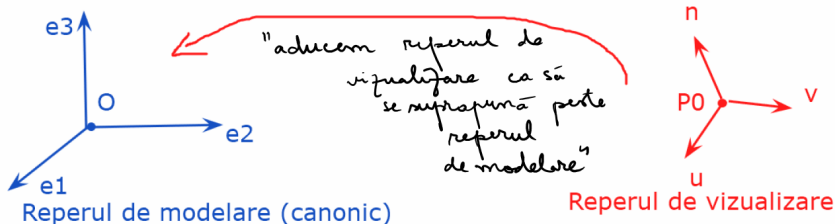
# Transformări (IV). Proiecții

Mihai-Sorin Stupariu

Sem. I, 2022 - 2023

# Schimbarea reperului ca transformare

Schimbarea de reper  $\leftrightarrow$  Efectuarea unei transformări



Descrierea transformărilor:

- translație a.î.  $P_0$  să devină originea, adică aplicarea  $T(-x_0, -y_0, -z_0)$  vectorii sunt  $2 \times 2$ , cu normă 1
- rotație 3D a.î. reperul ortonormat  $(u, v, n)$  să se suprapună cu reperul ON  $(e_1, e_2, e_3)$

- ▶ Rolul transformărilor de proiecție: de a permite reprezentarea unei lumi 3D, teoretic nemărginite, pe un monitor 2D mărginit.

- ▶ Rolul transformărilor de proiecție: de a permite reprezentarea unei lumi 3D, teoretic nemărginite, pe un monitor 2D mărginit.
- ▶ Despre aplicarea proiecțiilor:

- ▶ Rolul transformărilor de proiecție: de a permite reprezentarea unei lumi 3D, teoretic nemărginite, pe un monitor 2D mărginit.
- ▶ Despre aplicarea proiecțiilor:
  - dacă nu a fost efectuată nicio transformare de modelare, proiecția este aplicată în raport cu reperul de modelare, fiind decupat pătratul “standard”  $[-1, 1] \times [-1, 1]$ ,

- ▶ Rolul transformărilor de proiecție: de a permite reprezentarea unei lumi 3D, teoretic nemărginite, pe un monitor 2D mărginit.
- ▶ Despre aplicarea proiecțiilor:
  - dacă nu a fost efectuată nicio transformare de modelare, proiecția este aplicată în raport cu reperul de modelare, fiind decupat pătratul “standard”  $[-1, 1] \times [-1, 1]$ ,
  - dacă a fost efectuată o transformare de vizualizare (observatorul a fost “adus” în origine, axele au fost “aliniat”, etc.): din punctul de vedere al logicii imaginii, decuparea / proiecția sunt realizate în raport cu observatorul și reperul de vizualizare. Altfel spus, proiecția este aplicată după transformarea de vizualizare.

- ▶ Rolul transformărilor de proiecție: de a permite reprezentarea unei lumi 3D, teoretic nemărginite, pe un monitor 2D mărginit.
- ▶ Despre aplicarea proiecțiilor:
  - dacă nu a fost efectuată nicio transformare de modelare, proiecția este aplicată în raport cu reperul de modelare, fiind decupat pătratul “standard”  $[-1, 1] \times [-1, 1]$ ,
  - dacă a fost efectuată o transformare de vizualizare (observatorul a fost “adus” în origine, axele au fost “aliniat”, etc.): din punctul de vedere al logicii imaginii, decuparea / proiecția sunt realizate în raport cu observatorul și reperul de vizualizare. Altfel spus, proiecția este aplicată după transformarea de vizualizare.
- ▶ O proiecție este o transformare care implică (i) decuparea, (ii) proiecția propriu-zisă, fiind necesară o matrice  $4 \times 4$  adecvată. Din punct de vedere al implementării: dacă `matVis` este matricea de vizualizare (dată de `glm::lookAt()`) și `matPr` este matricea de proiecție, atunci în codul sursă trebuie să apară `matPr * matVis`.

## Cazul 2D

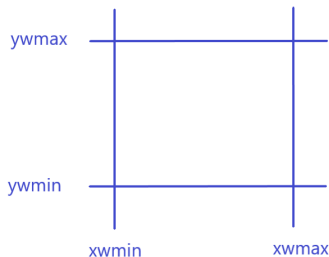
► `glm::ortho (xwmin, xwmax, ywmin, ywmax);`



## Cazul 2D

- ▶ `glm::ortho (xwmin, xwmax, ywmin, ywmax);`
- ▶ Efectul: este decupat un dreptunghi  $\mathcal{D}$  din planul orizontal  $Oxy$  (se presupune că nu au fost aplicate alte transformări). Dreptunghiul  $\mathcal{D}$  are laturile paralele cu axele de coordonate, fiind delimitat de dreptele

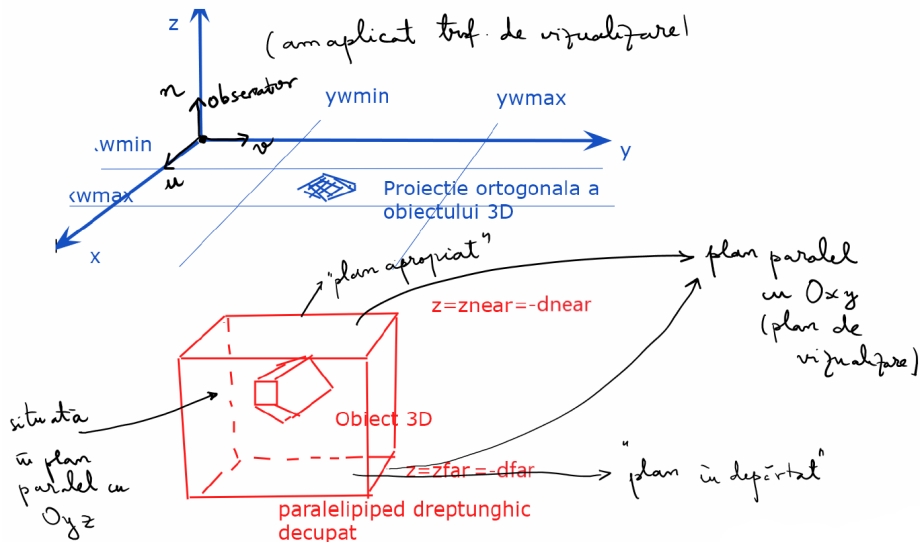
$$x = xwmin, \quad x = xwmax, \quad y = ywmin, \quad y = ywmax.$$



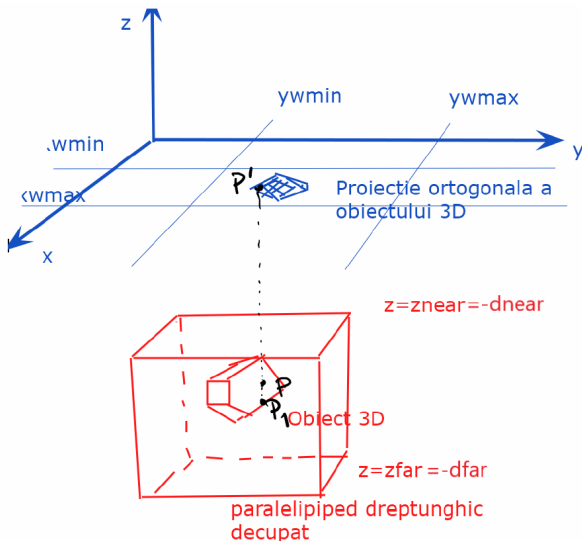
Apoi este realizată o transformare a dreptunghiului  $\mathcal{D}$  în pătratul “standard”  $[-1, 1] \times [-1, 1]$ . Matricea  $4 \times 4$  asociată transformării poate fi determinată explicit.

## Cazul 3D - proiecții ortogonale

```
glm::ortho (xwmin, xwmax, ywmin, ywmax, dnear, dfar);
```



# Ce este o proiecție ortogonală?



$P'$  este în pr.  
ortog. a lui  $P$ .

$P'$  = proiecția  
ortogonală  
a lui P  
(prin P ducem  
o dreaptă  
perp. pe Oxy,  
apoi intersecția  
cu Oxy)

# Cazul 3D - proiecții ortogonale

Matricea  $4 \times 4$  asociată este

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

Această matrice are rolul de a transforma un **paralelipiped dreptunghic** decupat în paralelipipedul “standard”  $[-1, 1] \times [-1, 1] \times [-1, 1]$ , apoi, în mod implicit, sunt reținute primele două coordonate.

# Cazul 3D - proiecții perspective

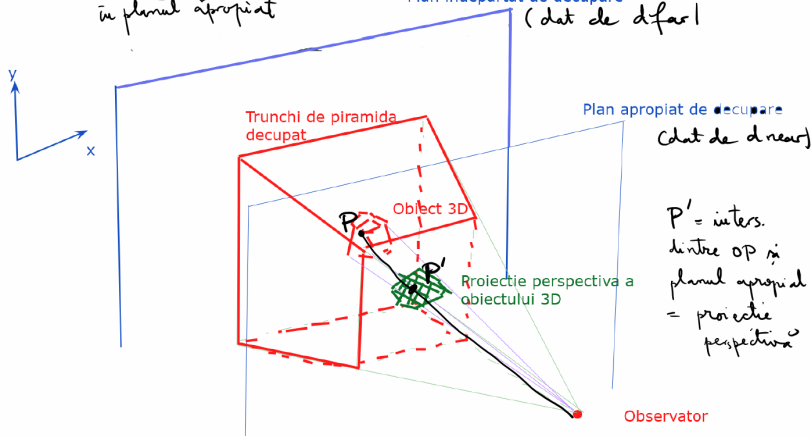
`glm::frustum()` - este decupat un trunchi de piramidă.

`glm::frustum(xwmin, xwmax, ywmin, ywmax, dnear, dfar);`

în planul apropiat

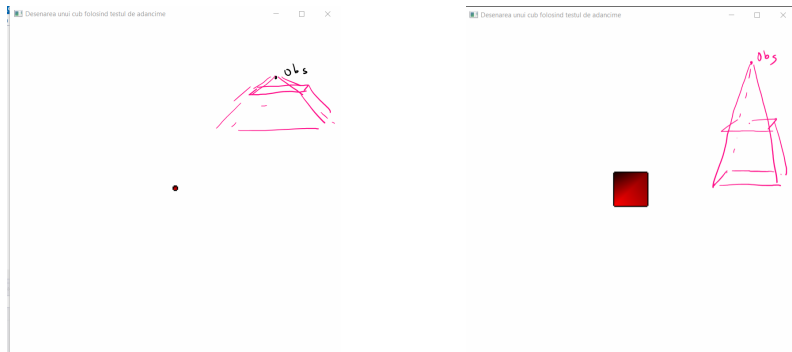
Plan îndepărtat de decupare

(dat de `dfar`)

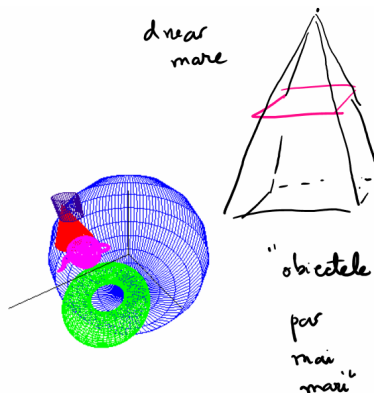
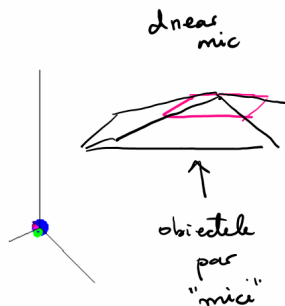


# Cazul 3D - proiecții perspective. Valoarea *dnear* și efectul asupra desenului în cazul *glFrustum*

Presupunem că nu modificăm valoarea  $xwmin$ ,  $xwmax$ ,  $ywmin$ ,  $ywmax$ . Dacă *dnear* este mic, atunci trunchiul de piramidă decupat are “deschidere mai mare” și obiectele vor părea mai mici (stânga). Dacă *dnear* este mare, atunci trunchiul de piramidă decupat este “mai îngust” și obiectele vor părea mai mari (dreapta).



# Cazul 3D - proiecții perspective. Valoarea *dnear* și efectul asupra desenului în cazul `glFrustum` - alt exemplu



# Cazul 3D - proiecții perspective

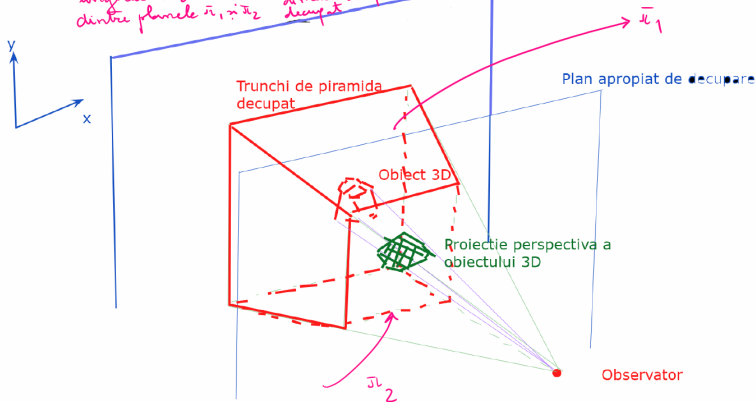
`glm::perspective()` - este decupat un trunchi de piramidă decupat dintr-o piramidă în care înălțimea dusă din vârful piramidei inițiale (observator) cade în centrul dreptunghiului. "Deschiderea" piramidei este dată de fov.

```
glm::perspective(fov, aspect, ywmax, dnear, dfar);
```

Plan indepartat de decupare

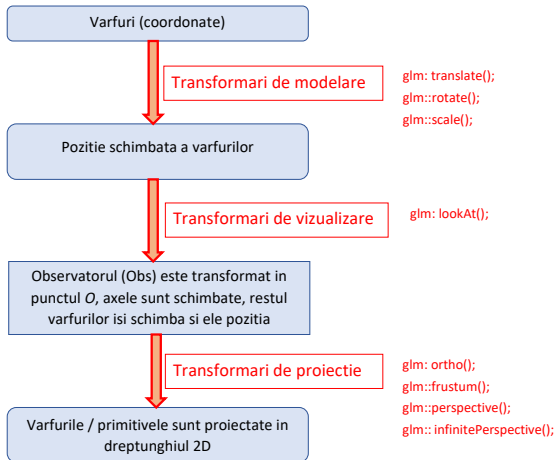
unghiul diedru dintre planele  $\pi_1, \pi_2$

raport dintre dimensiunilor decupate





# Concluzie - fluxul transformărilor

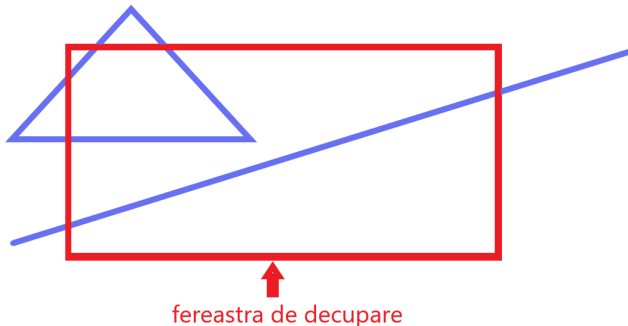


Atenție la ordinea înmulțirii matricelor corespunzătoare din shader:

`matrProiectie * matrVizualizare * matrModelare.`

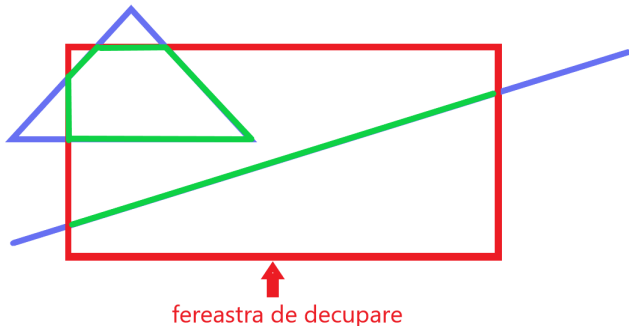
## Algoritmi de decupare - motivație

În cazul primitivelor din desen, doar o parte a acestora intersectează fereastra de decupare și urmează să fie randate.



## Algoritmi de decupare - motivație

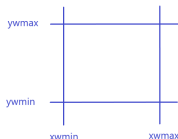
Pentru segment va fi randată doar o porțiune. În cazul triunghiului, va fi decupată o porțiune a sa, fiind randat un pentagon.



## Algoritmi de decupare - observații generale

- Trebuie stabilit dacă o primitivă geometrică intersectează sau nu fereastra de decupare. În continuare presupunem că suntem în cazul 2D, fereastra de decupare fiind delimitată de dreptele

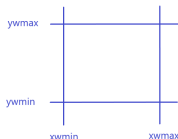
$$x = xwmin, x = xwmax, y = ywmin, y = wymax.$$



## Algoritmi de decupare - observații generale

- ▶ Trebuie stabilit dacă o primitivă geometrică intersectează sau nu fereastra de decupare. În continuare presupunem că suntem în cazul 2D, fereastra de decupare fiind delimitată de dreptele

$$x = xwmin, x = xwmax, y = ywmin, y = wymax.$$

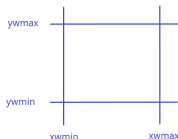


- ▶ Algoritmii de decupare depind de tipul primitivei. În cazul punctelor: se bazează pe inegalități.

## Algoritmi de decupare - observații generale

- ▶ Trebuie stabilit dacă o primitivă geometrică intersectează sau nu fereastra de decupare. În continuare presupunem că suntem în cazul 2D, fereastra de decupare fiind delimitată de dreptele

$$x = xwmin, x = xwmax, y = ywmin, y = wymax.$$

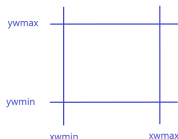


- ▶ Algoritmii de decupare depind de tipul primitivei. În cazul punctelor: se bazează pe inegalități.
- ▶ În continuare: algoritmi pentru segmente de dreaptă (pornind de la aceștia se poate ajunge la algoritmi pentru poligoane). Sunt descriși doi algoritmi: (i) Cohen-Sutherland (calitativ), (ii) [Liang-Barski](#) (cantitativ).

## Algoritmi de decupare - observații generale

- ▶ Trebuie stabilit dacă o primitivă geometrică intersectează sau nu fereastra de decupare. În continuare presupunem că suntem în cazul 2D, fereastra de decupare fiind delimitată de dreptele

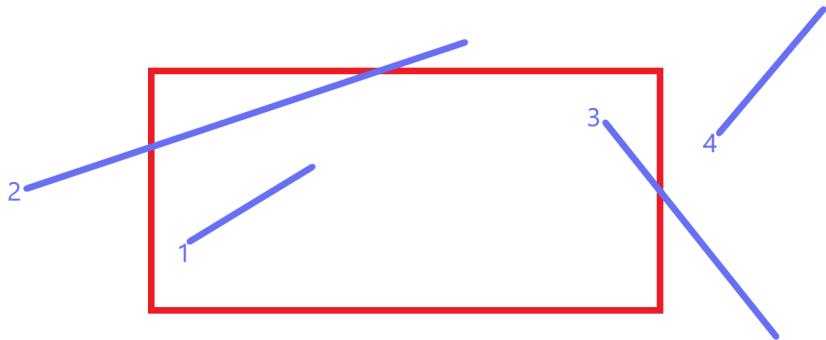
$$x = xwmin, x = xwmax, y = ywmin, y = ywmax.$$



- ▶ Algoritmii de decupare depind de tipul primitivei. În cazul punctelor: se bazează pe inegalități.
- ▶ În continuare: algoritmi pentru segmente de dreaptă (pornind de la aceștia se poate ajunge la algoritmi pentru poligoane). Sunt descriși doi algoritmi: (i) Cohen-Sutherland (calitativ), (ii) [Liang-Barski](#) (cantitativ).
- ▶ Alte metode / referințe se găsesc în [Line Clipping in 2D: Overview, Techniques and Algorithms, 2022](#)

# Abordare calitativă. Algoritmul Cohen-Sutherland

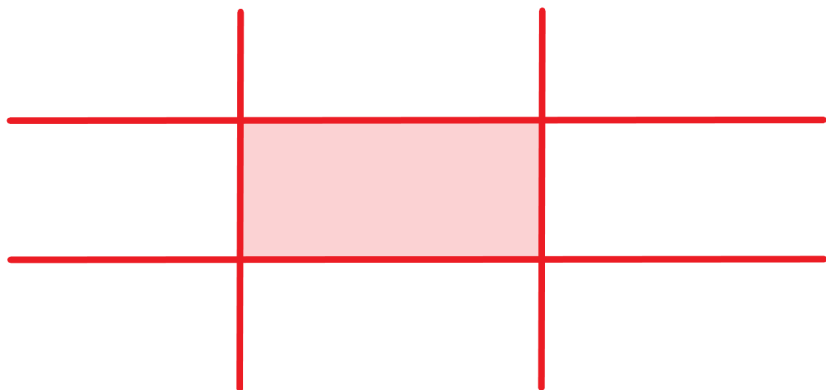
Diverse configurații ale segmentelor față de fereastra de decupare.





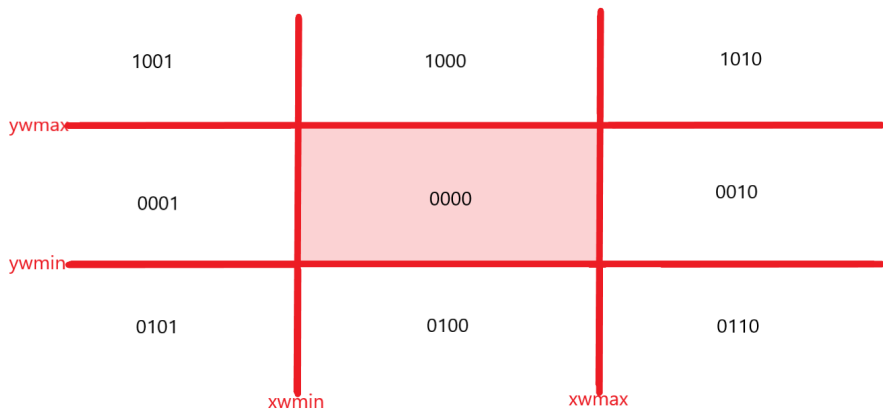
## Algoritmul Cohen-Sutherland - împărțirea planului

Planul este împărțit în 9 regiuni, fiecareia dintre ele îi este asociat un cod pe 4 biți (TBRL – Top Bottom Right Left).



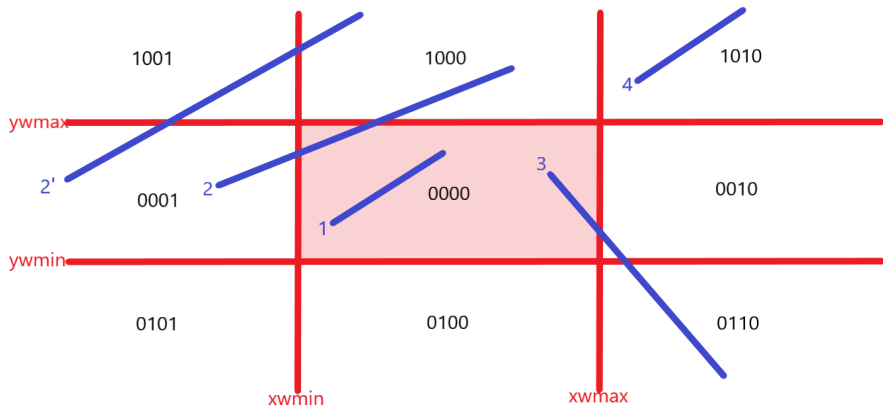
# Algoritmul Cohen-Sutherland - codurile regiunilor

Planul este împărțit în 9 regiuni, fiecareia dintre ele îi este asociat un cod pe 4 biți (TBRL – Top Bottom Right Left).



# Algoritmul Cohen-Sutherland - coduri și segmente

Dat un segment  $s$ , codurile asociate extremităților sale dau o primă informație despre poziția segmentului  $s$  față de fereastra de decupare.



# Algoritmul Cohen-Sutherland - sinteză

- Pentru un punct  $M = (x_M, y_M)$  se stabilește codul regiunii:  
 $x < x_{wmin} \Rightarrow L = 1, R = 0$ , deci \*\*01  
 $x > x_{wmax} \Rightarrow L = 0, R = 1$ , deci \*\*10, etc

# Algoritmul Cohen-Sutherland - sinteză

- ▶ Pentru un punct  $M = (x_M, y_M)$  se stabilește codul regiunii:  
 $x < x_{wmin} \Rightarrow L = 1, R = 0$ , deci \*\*01  
 $x > x_{wmax} \Rightarrow L = 0, R = 1$ , deci \*\*10, etc
- ▶ Date două puncte,  $P$  și  $Q$ , cele două coduri asociate dau o primă informație referitoare la poziția segmentului  $[PQ]$  față de fereastra de decupare.

## Algoritmul Cohen-Sutherland - sinteză

- ▶ Pentru un punct  $M = (x_M, y_M)$  se stabilește codul regiunii:  
 $x < x_{wmin} \Rightarrow L = 1, R = 0$ , deci \*\* 01  
 $x > x_{wmax} \Rightarrow L = 0, R = 1$ , deci \*\* 10, etc
- ▶ Date două puncte,  $P$  și  $Q$ , cele două coduri asociate dau o primă informație referitoare la poziția segmentului  $[PQ]$  față de fereastra de decupare.
- ▶ Există cazuri în care folosind codurile se poate lua o decizie referitoare la poziția relativă a segmentului față de fereastră (de exemplu două coduri de tipul \*\* 10, ambele coduri 0000, etc.).
- ▶ Există cazuri în care folosind codurile nu se poate lua o decizie (de exemplu 0001 și 1000), fiind necesari alți algoritmi, cantitativi (se determină explicit coordonatele intersecțiilor dintre segment și dreptele suport ale ferestrei de decupare, se stabilește dacă aceste puncte sunt pe laturile dreptunghiului, etc.).

## Abordare cantitativă. Algoritmul Liang-Barsky

- **Principiu:** Utilizează **reprezentarea parametrică** a unei drepte. Ideea centrală: fiecărui punct de pe dreaptă îi corespunde exact un număr real (parametru) și reciproc.

## Abordare cantitativă. Algoritmul Liang-Barsky

- ▶ **Principiu:** Utilizează **reprezentarea parametrică** a unei drepte. Ideea centrală: fiecărui punct de pe dreaptă îi corespunde exact un număr real (parametru) și reciproc.
- ▶ **Explicit:** Fie  $P_0 = (x_0, y_0), P_1 = (x_1, y_1) \in \mathbb{R}^2$ , presupunem (fără a restrânge generalitatea) că  $x_0 < x_1, y_0 < y_1$ .  
Notăm  $\Delta x = x_1 - x_0, \Delta y = y_1 - y_0$ .  
Dreapta  $P_0P_1$  are reprezentarea parametrică

$$\begin{cases} x = x_0 + u \cdot \Delta x \\ y = y_0 + u \cdot \Delta y \end{cases}, u \in \mathbb{R}.$$

A da un punct de pe dreapta  $P_0P_1$  revine la a indica o valoare a lui  $u$  și reciproc.

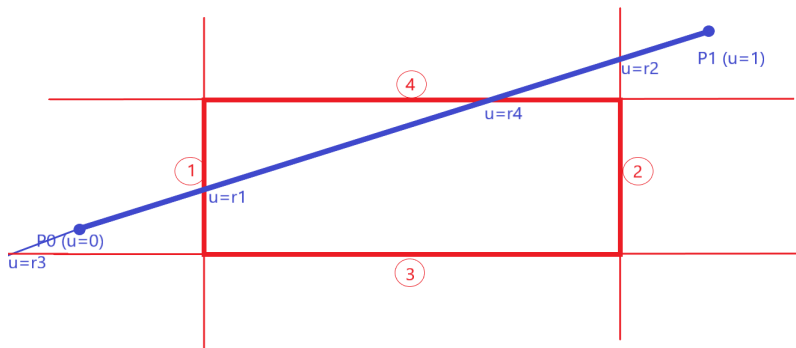


## Algoritmul Liang-Barsky - intersecții cu fereastra

**Intersecții cu fereastra de decupare:** Dreapta  $P_0P_1$  intersectează dreptele suport ale laturilor dreptunghiului (notate 1, 2, 3, 4) în puncte care corespund unor valori  $r_1, r_2, r_3, r_4$  ale parametrului (calculabile explicit!).

De exemplu,  $r_1$  se determină din condiția

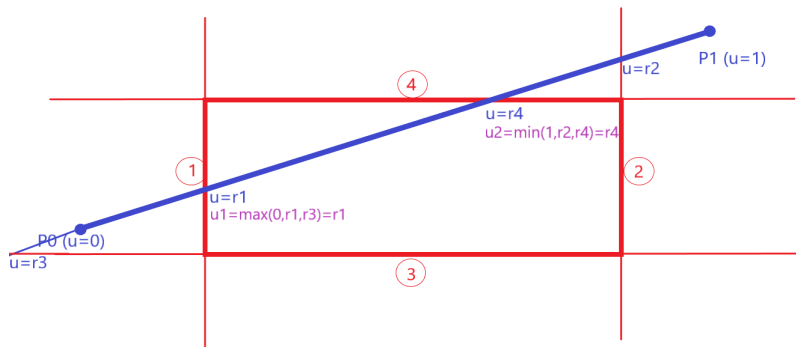
$$x_{wmin} = x_0 + r_1 \cdot \Delta x, \text{ deci } r_1 = \frac{x_{wmin} - x_0}{\Delta x}, \text{ etc.}$$



# Algoritmul Liang-Barsky - valori importante ale parametrului

“Intrare” în dreptunghi:  $u_1 = \max(0, r_1, r_3)$   
 (start segment, fâșia verticală, fâșia orizontală)

“Ieșire” din dreptunghi:  $u_2 = \min(1, r_2, r_4)$   
 (stop segment, fâșia verticală, fâșia orizontală)



## Algoritmul Liang-Barsky - condiția de intersecție

Condiția ca **segmentul**  $[P_0P_1]$  să intersecteze dreptunghiul de decupare:

$$u_1 < u_2$$

(intuitiv: “înrăm în dreptunghiul de decupare înainte de a ieși”).  
Mai mult, segmentul care urmează să fie decupat are extremitățile corespunzătoare parametrilor  $u_1$  și  $u_2$ .

