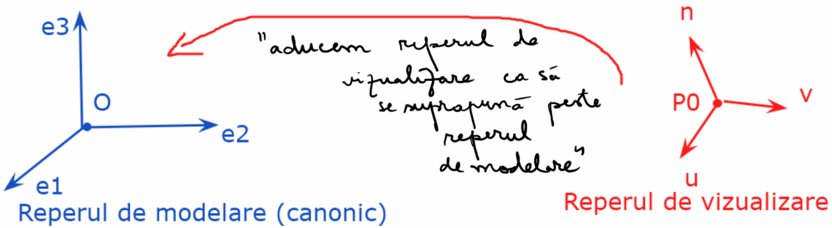
Transformări (IV). Proiecții

Mihai-Sorin Stupariu

Sem. I, 2023 - 2024

Schimbarea reperului ca transformare

Schimbarea de reper $\leftrightarrow$ Efectuarea unei transformări



Descrierea transformărilor:

- translatăm a.î. P_0 să devină originea, adică aplicăm $T_{(-x_0,-y_0,-z_0)}$

vectorii sunt $\perp$ 2×2, cu norma 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 "aliniate", 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 “aliniate”, 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 4×4 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`.

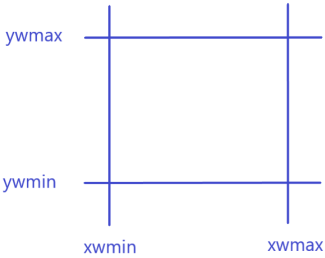
Cazul 2D

- `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

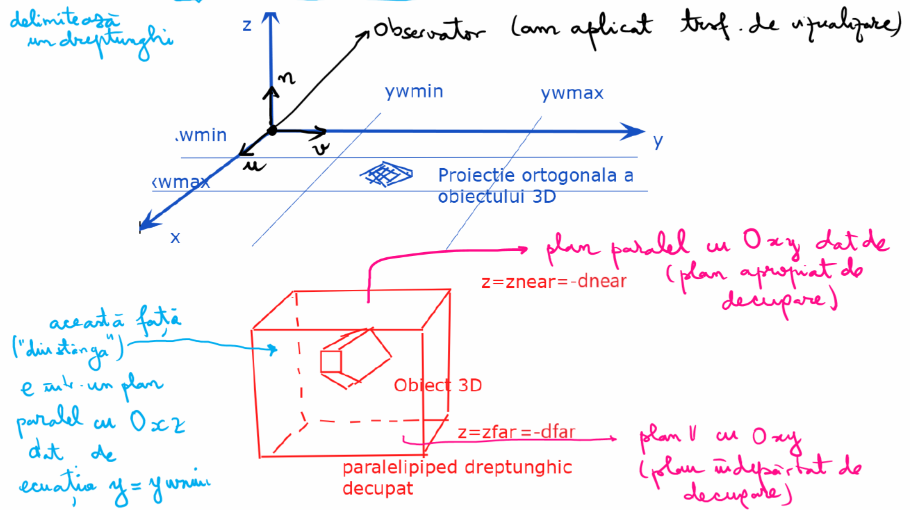
$$x = xwmin, \; x = xwmax, \; y = ywmin, \; y = wymax.$$



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

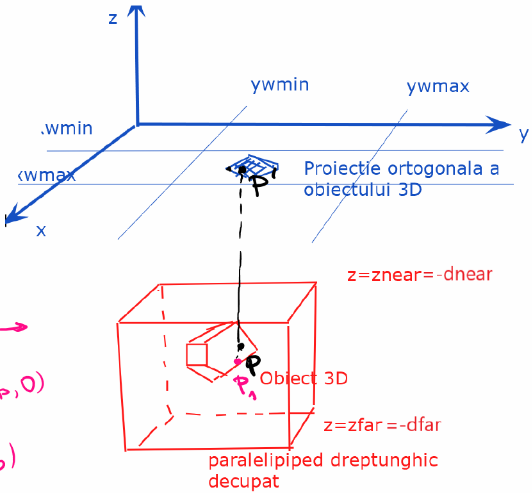
Cazul 3D - proiecții ortogonale

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



Ce este o proiecție ortogonală?

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



formula:

$(x_P, y_P, z_P) \mapsto$

$\mapsto (x_P, y_P, 0)$

$\parallel$

(x_P, y_P)

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

P' e și proiecția ortogonală a lui P_1

Cazul 3D - proiecții ortogonale

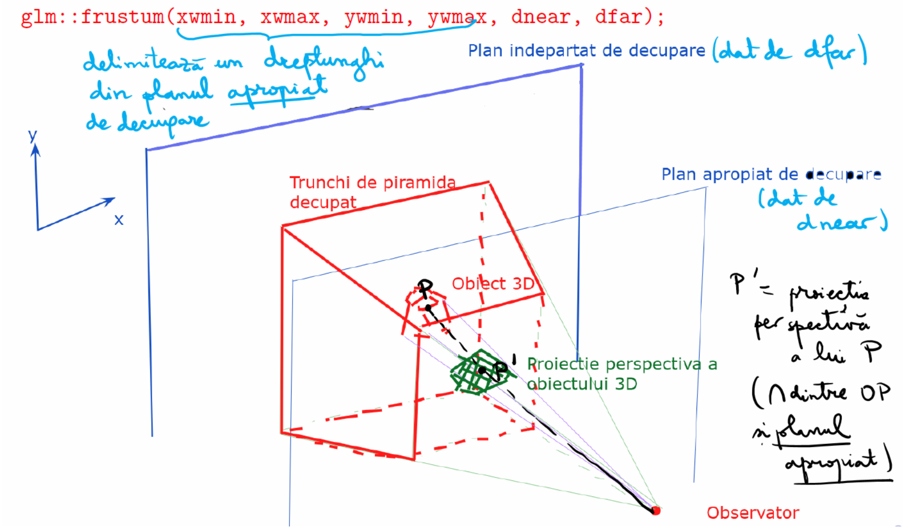
Matricea 4×4 asociată este

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

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.

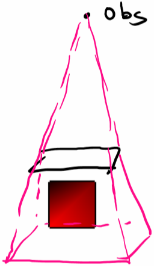
Cazul 3D - proiecții perspective

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



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

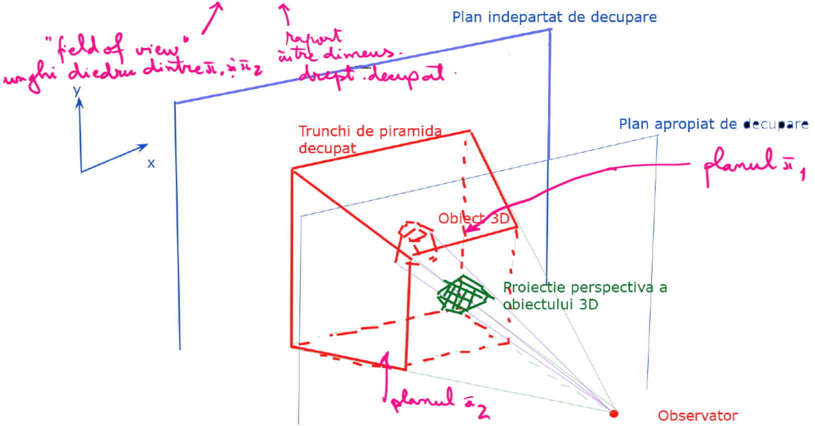
Presupunem că nu modificăm valoarea *xwmin*, *xwmax*, *ywmin*, *ywmax*. Dacă *dnear* este mic, atunci trunchiul de piramidă decupat are “deschidere mai mare” și obiectele vor părea mai mici (stânga). Dacă *dnear* este mare, atunci trunchiul de piramidă decupat este “mai îngust” și obiectele vor părea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



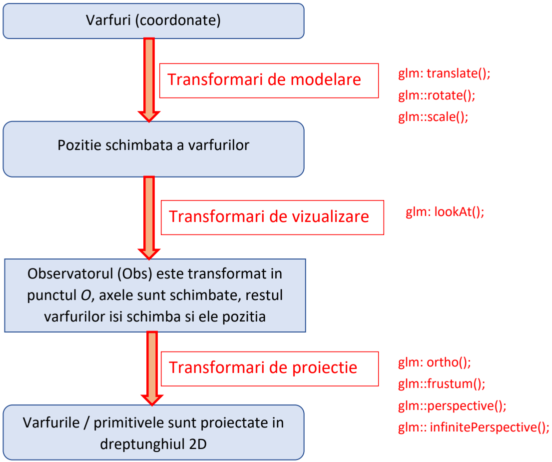
Cazul 3D - proiecții perspective

`glm::perspective()` - este decupat un trunchi de piramidă decupat dintr-o piramdiă 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);
```



Concluzie - fluxul transformărilor

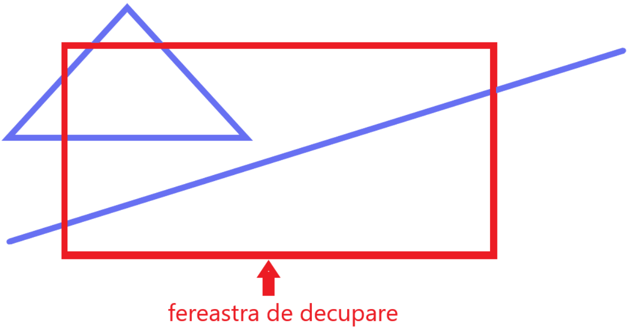


Atenție la ordinea înmulțirii matricelor corespunzătoare din shader:

`matrProiectie * matrVizualizare * matrModelare.`

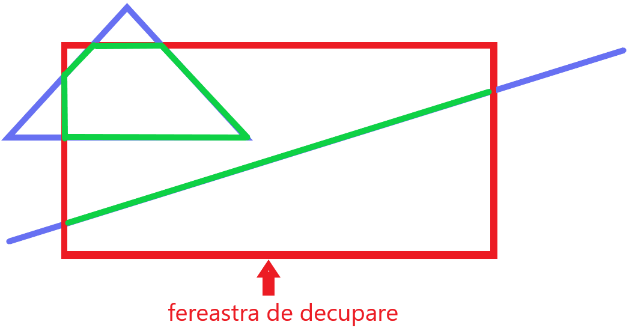
Algoritmi de decupare - motivație

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



Algoritmi de decupare - motivație

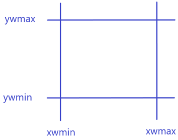
Pentru segment va fi randată doar o porțiune. În cazul triunghiului, va fi decupată o porțiune a sa, fiind randat un pentagon.



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

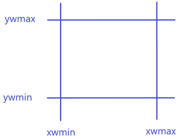
$$x = xwmin, \; x = xwmax, \; y = ywmin, \; y = wymax.$$



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

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

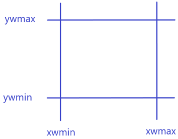


- Algoritmii de decupare depind de tipul primitivei. În cazul punctelor: se bazează pe inegalități.

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

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

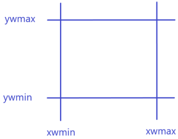


- Algoritmii de decupare depind de tipul primitivei. În cazul punctelor: se bazează pe inegalități.
- În continuare: algoritmi pentru segmente de dreaptă (pornind de la aceștia se poate ajunge la algoritmi pentru poligoane). Sunt descriși doi algoritmi: (i) Cohen-Sutherland (calitativ), (ii) Liang-Barski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

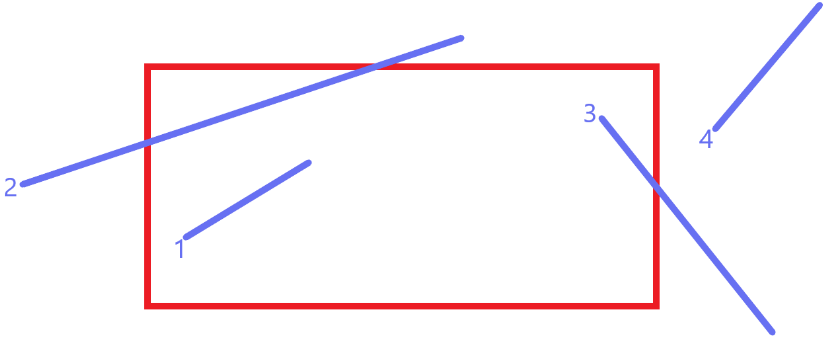
$$x = xwmin,\ x = xwmax,\ y = ywmin,\ y = wymax.$$



- 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).
- Alte metode / referințe se gšesc în *Line Clipping in 2D: Overview, Techniques and Algorithms*, 2022

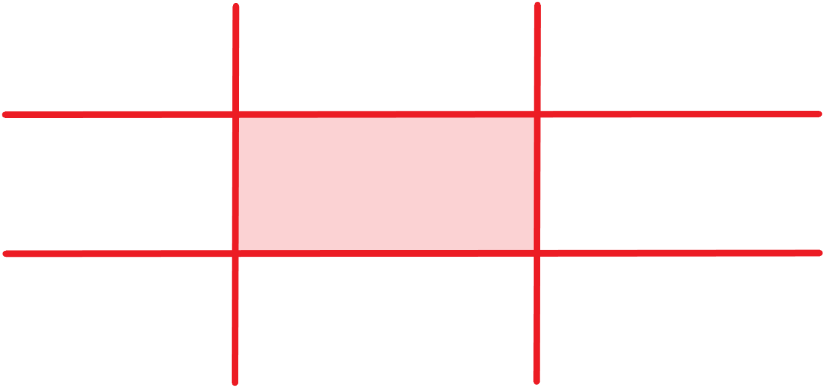
Abordare calitativă. Algoritmul Cohen-Sutherland

Diverse configurații ale segmentelor față de fereastra de decupare.



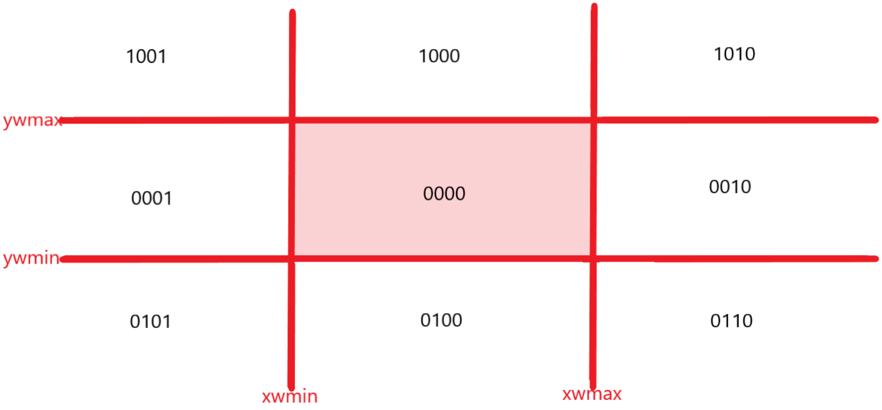
Algoritmul Cohen-Sutherland - împărțirea planului

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



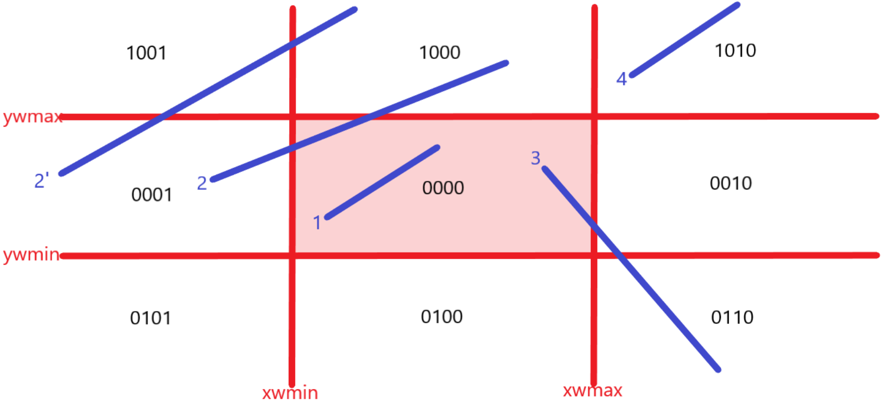
Algoritmul Cohen-Sutherland - codurile regiunilor

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:
  $x < xwmin \Rightarrow L = 1, R = 0$, deci $** 01$
  $x > xwmax \Rightarrow L = 0, R = 1$, deci $** 10$, etc

Algoritmul Cohen-Sutherland - sinteză

- Pentru un punct $M = (x_M, y_M)$ se stabilește codul regiunii:
  $x < xwmin \Rightarrow L = 1, R = 0$, deci $**01$
  $x > xwmax \Rightarrow L = 0, R = 1$, 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:
  $x < xwmin \Rightarrow L = 1, R = 0$, deci $**01$
  $x > xwmax \Rightarrow L = 0, R = 1$, 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

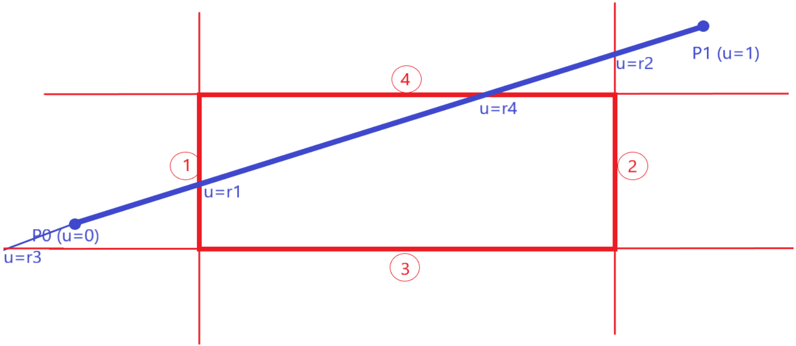
$$\begin{cases} x = x_0 + u \cdot \Delta x \\ y = y_0 + u \cdot \Delta y \end{cases}, u \in \mathbb{R}.$$

  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

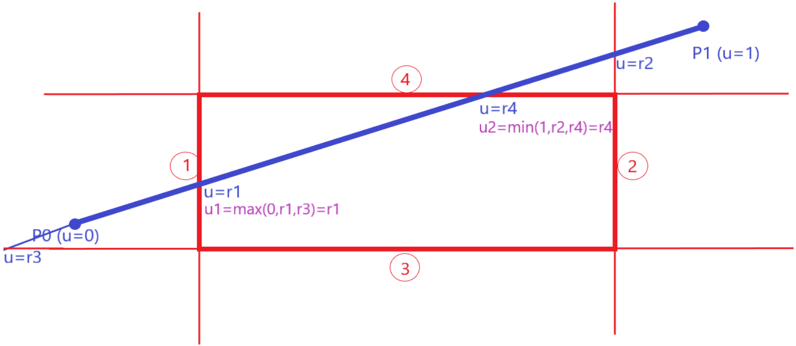
$$xwmin = x_0 + r_1 \cdot \Delta x, \text{ deci } r_1 = \frac{xwmin - x_0}{\Delta x}, \text{ etc.}$$



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** $[P_0 P_1]$ să intersecteze dreptunghiul de decupare:

$$u_1 < u_2$$

(intuitiv: "intrăm 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.