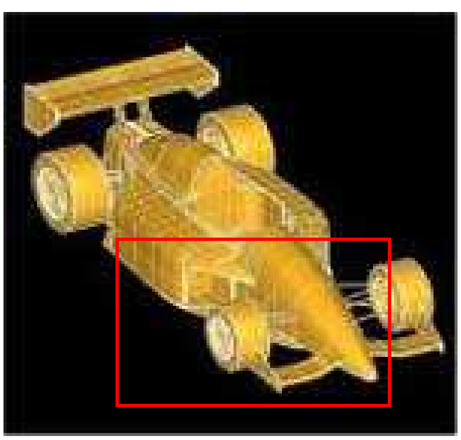
Recorte

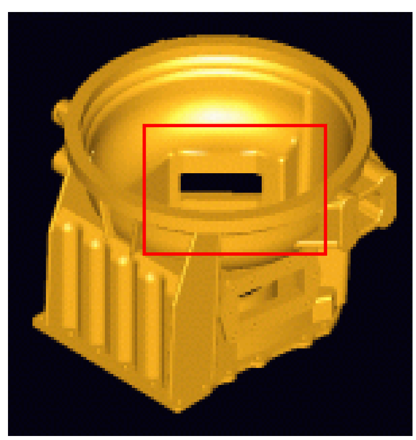
Rogers – Capítulo 3

Apostila – Capítulo 6

Recorte (*Clipping*)

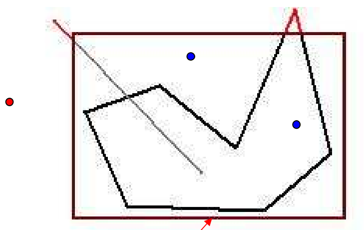
Determinar os pontos contidos no volume de visão/janela de exibição.



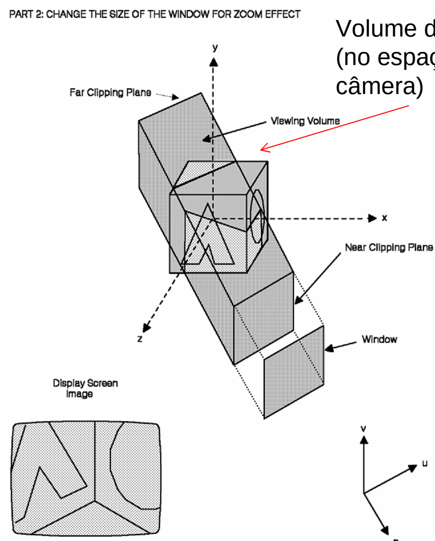


Recorte (*Clipping*)

Partição em dois subespaços: interior e exterior.

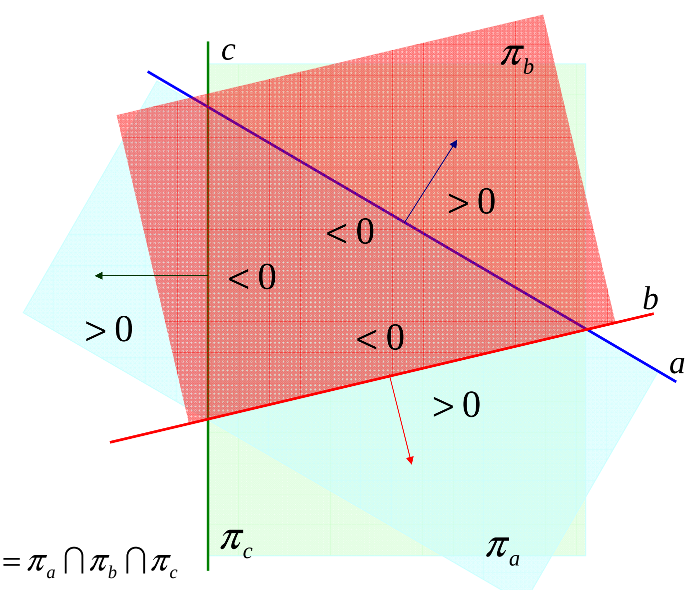


Caixa de recorte (no espaço de imagem)



Volume de recorte (no espaço da câmera)

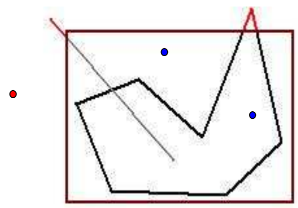
Caixa Recortante



$Triângulo = \pi_a \cap \pi_b \cap \pi_c$

Recorte

Caixa recortante convexa



Pontos: verificar se estão na interseção dos semi-planos definidos pelas retas suporte dos lados da caixa

Sequência de Segmentos (Polilinhas): identificar os pontos dos segmentos que estão na interseção dos semi-planos definidos pelas retas suporte dos lados da caixa

Polígonos: identificar os pontos do polígono que estão na interseção dos semi-planos definidos pelas retas suporte dos lados da caixa

"Algebrizar" Problema

Interseção $\longrightarrow$ **Solução de Sistemas de Equações**

Pontos: satisfazer equações dos semi-planos

$$n_{x,i}x + n_{y,i}y + d \le 0$$

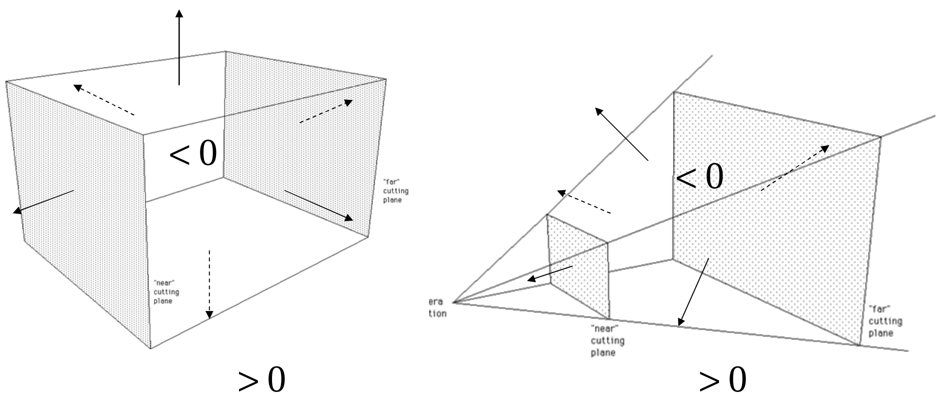
Polilinhas: reduzir o problema à partição de cada segmento P(t) da polilinha em sub-segmentos e identificar os sub-segmentos no interior da caixa

$$P(t) = P_1 + t(P_2 - P_1)$$

$$n_{x,i}x + n_{y,i}y + d \le 0$$

Polígonos: reduzir o problema à partição de cada segmento da polilinha em sub-segmentos e conectar os sub-segmentos para formar regiões de recorte.

Volume Recortante



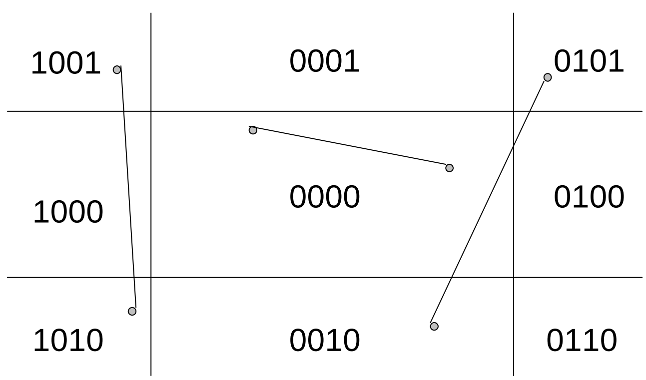
No lugar de retas de suporte, temos planos de suporte

$$n_{x,i}x + n_{y,i}y + n_{z,i}z + d = 0$$

Recorte em 2D

Algoritmo de *Cohen-Sutherland*

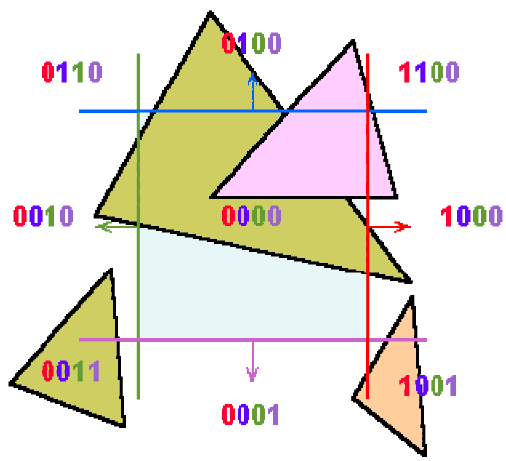
Caixa de recorte retangular alinhada com os eixos



AND *bit* a *bit*
- ≠ 0000, trivialmente fora
- = 0000
  - trivialmente dentro
  - Interseção

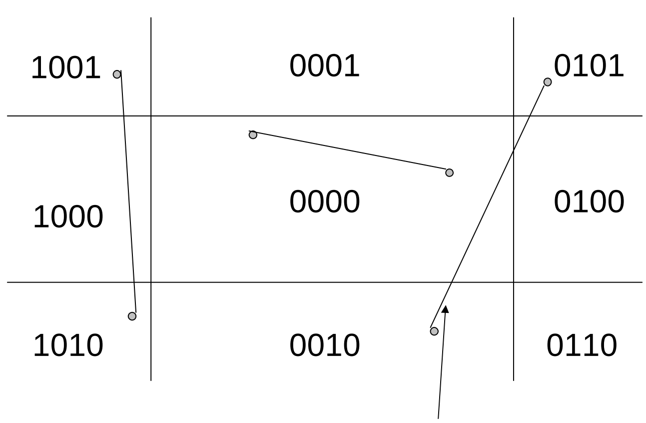
Recorte em 2D

Algoritmo de *Cohen-Sutherland*



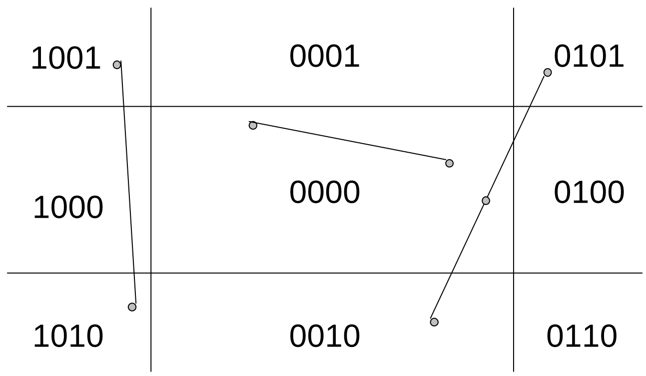
Recorte em 2D

Algoritmo de Ponto Médio

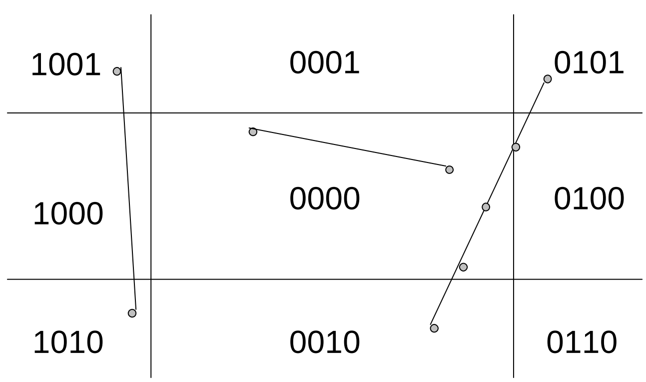


Caso não trivial → Caso trivial

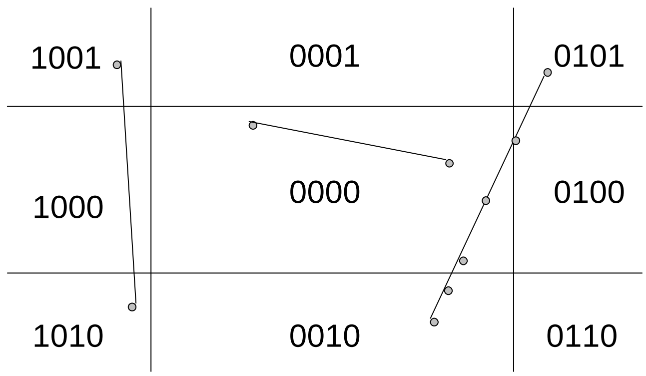
Recorte em 2D
Algoritmo de Ponto Médio



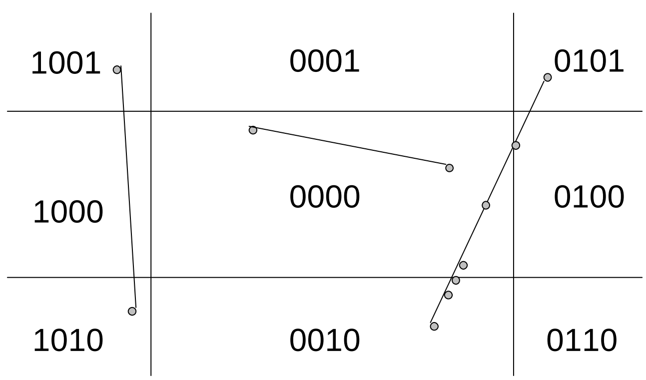
Recorte em 2D
Algoritmo de Ponto Médio



Recorte em 2D
Algoritmo de Ponto Médio

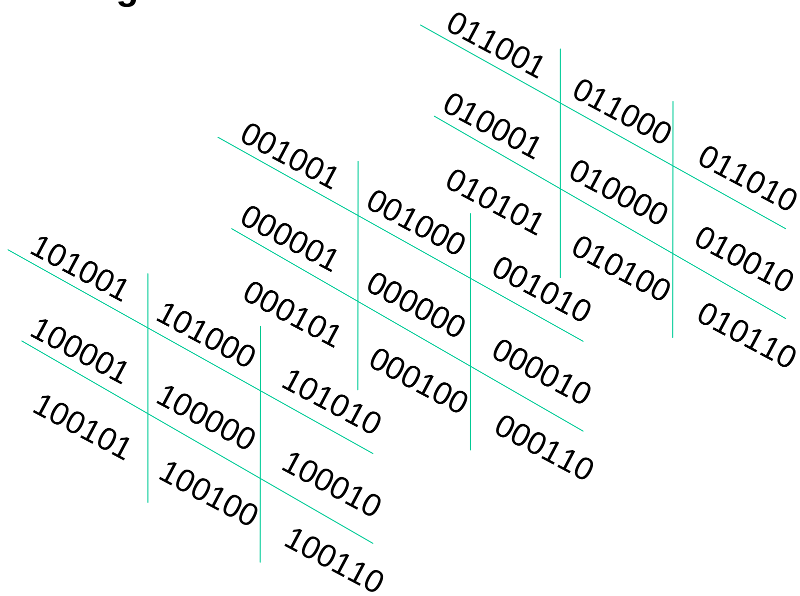


Recorte em 2D
Algoritmo de Ponto Médio



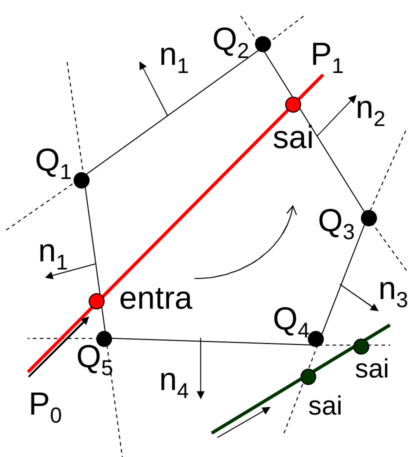
Recorte em 3D

Algoritmo de *Cohen-Sutherland*



Recorte (*Clipping*)

Algoritmo de *Cyrus-Becker*
Caixa recortante convexa



Determinar e ordenar os pontos de interseção:

$$t = \frac{N_i\,(P_0 - Q_i)}{-N_i\,(P_1 - P_0)}$$

Somente intercepta em 2 pontos:

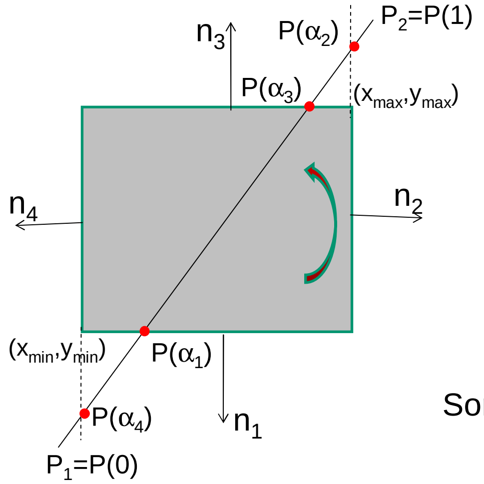
- entra $((P_1 - P_0).n_i < 0)$
- sai $((P_1 - P_0).n_i > 0)$

Somente na ordem (entra,sai)

Recorte em 2D

Algoritmo de *Liang-Barsky*

Caixa recortante retangular



$$P(\alpha) = (1-\alpha)\,P_1 + \alpha\,P_2$$

$$\begin{pmatrix} x(\alpha) \\ y(\alpha) \end{pmatrix} = (1-\alpha)\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \alpha \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$$

$\alpha_1 = (y_{min}-y_1)/(y_2-y_1)$
$\alpha_2 = (x_{max}-x_1)/(x_2-x_1)$
$\alpha_3 = (y_{max}-y_1)/(y_2-y_1)$
$\alpha_4 = (x_{min}-x_1)/(x_2-x_1)$

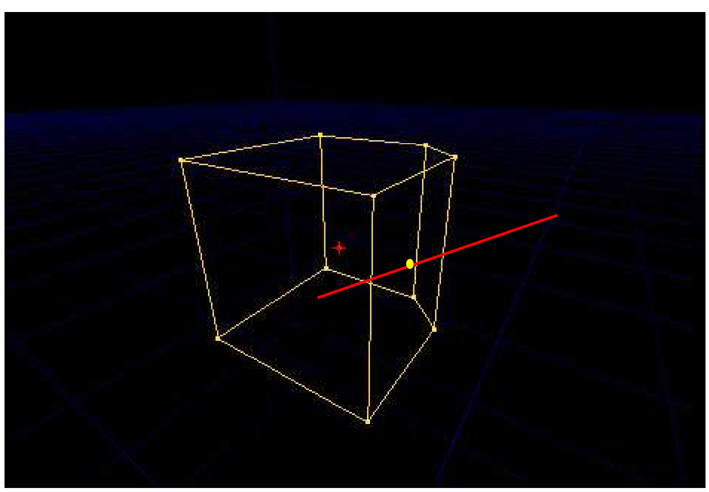
$\alpha_1,\alpha_2,\alpha_3,\alpha_4 \in [0,1]$

Somente na ordem (entra,sai)

- entra $((P_2-P_1).n_i < 0)$
- sai $((P_2-P_1).n_i > 0)$

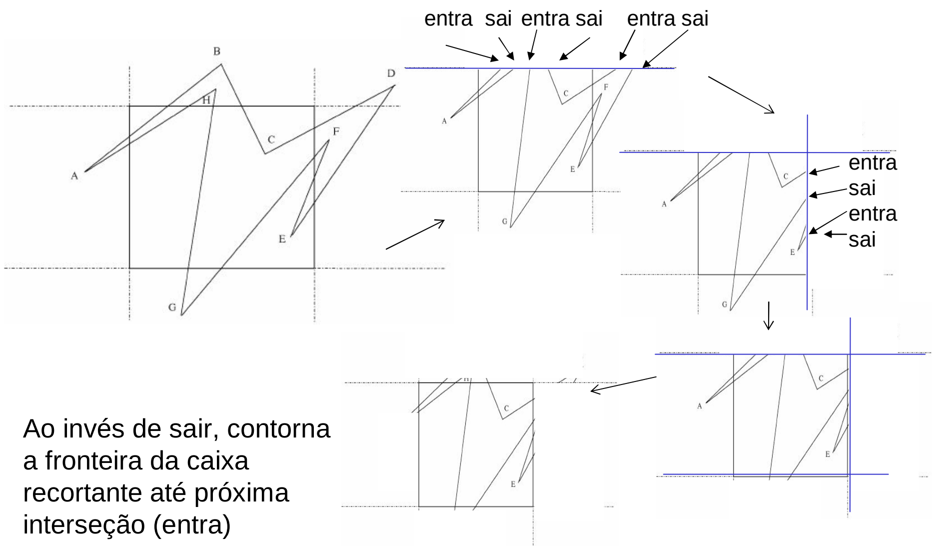
Recorte em 3D

Algoritmo de *Liang-Barsky*



Recorte de Áreas em 2D

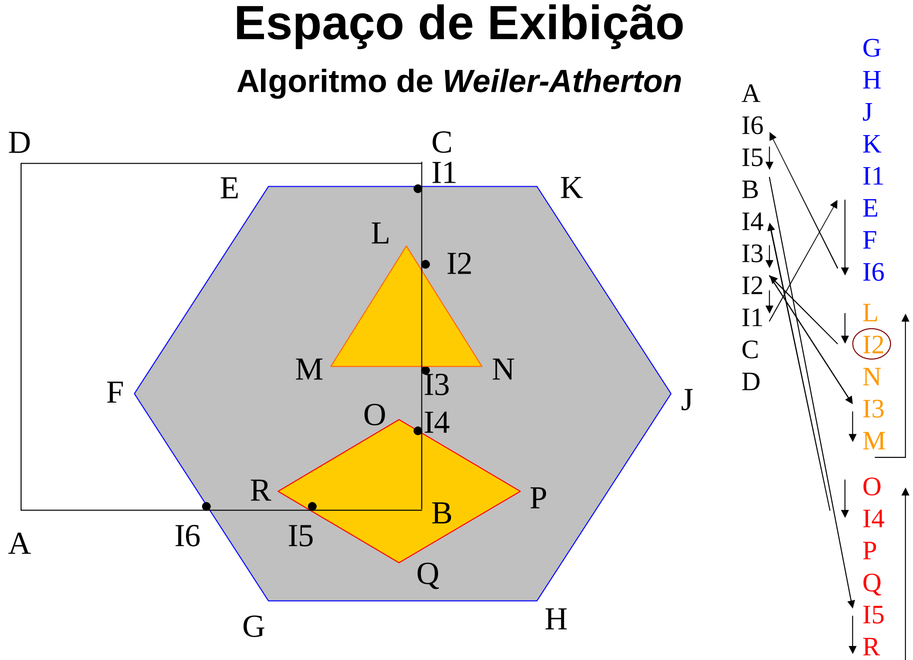
Algoritmo de *Sutherland-Hodgman*



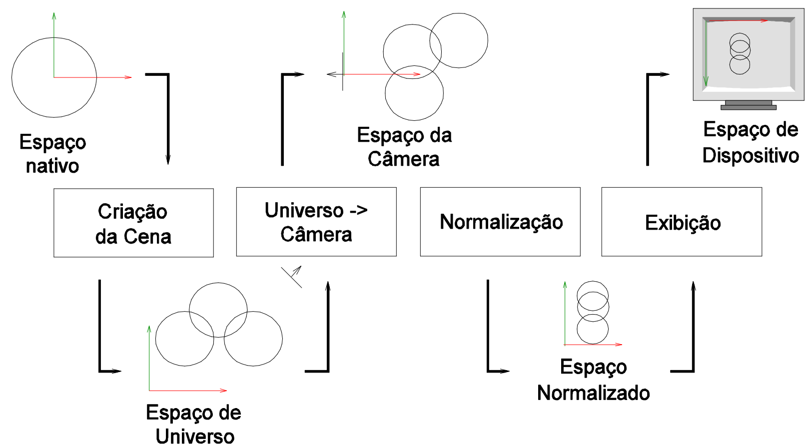
Ao invés de sair, contorna a fronteira da caixa recortante até próxima interseção (entra)

Espaço de Exibição

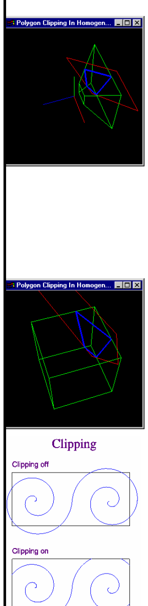
Algoritmo de *Weiler-Atherton*

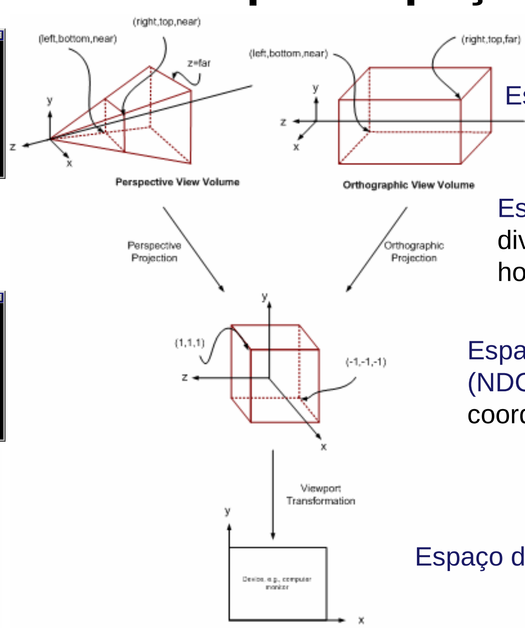


Espaço de Recorte



Em qual espaço?





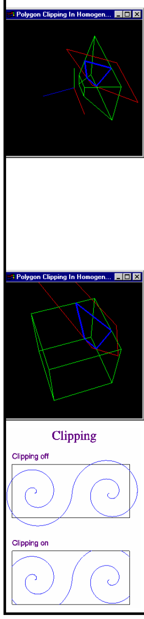
Espaço da Câmera (VRC)

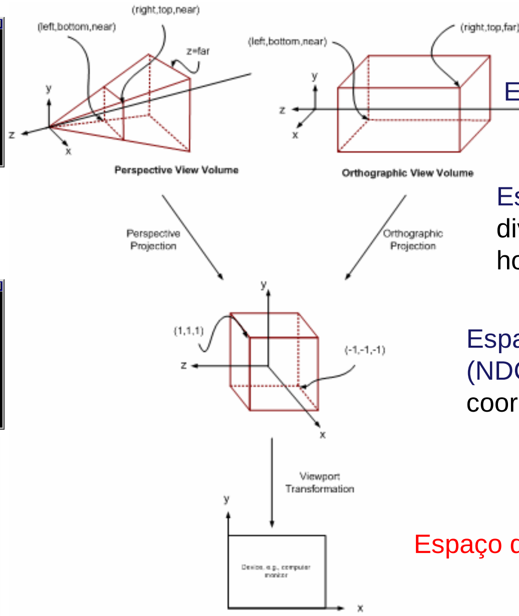
Espaço 4D: antes da divisão pela coordenada homogênea w

Espaço 3D Normalizado (NDC): após a divisão pela coordenada homogênea w

Espaço de Dispositivo 2D (DC)

Em qual espaço?





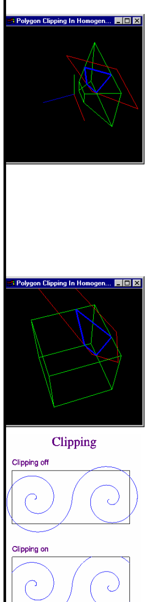
Espaço da Câmera (VRC)

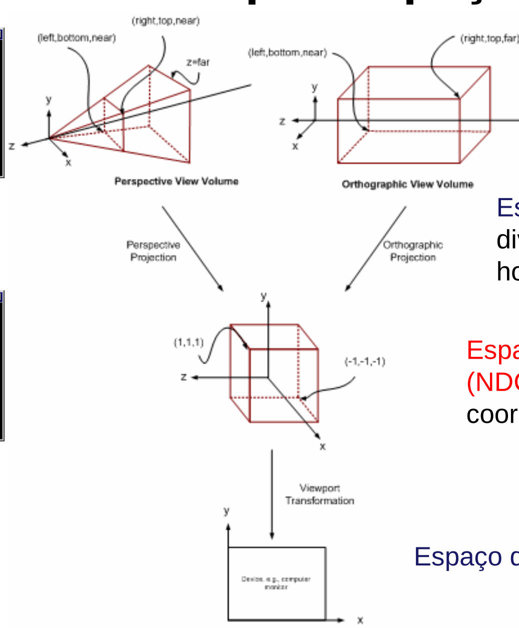
Espaço 4D: antes da divisão pela coordenada homogênea w

Espaço 3D Normalizado (NDC): após a divisão pela coordenada homogênea w

Espaço de Dispositivo 2D (DC)

Em qual espaço?





Espaço da Câmera (VRC)

Espaço 4D: antes da divisão pela coordenada homogênea w

Espaço 3D Normalizado (NDC): após a divisão pela coordenada homogênea w

Espaço de Dispositivo 2D (DC)

Recorte em 3D (w=1)

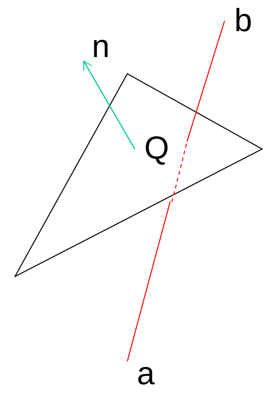
Interseção segmento-plano

$P(t) = a + t\,(b-a)$

$f(P(t)) = n\,(P(t) - Q) = 0$

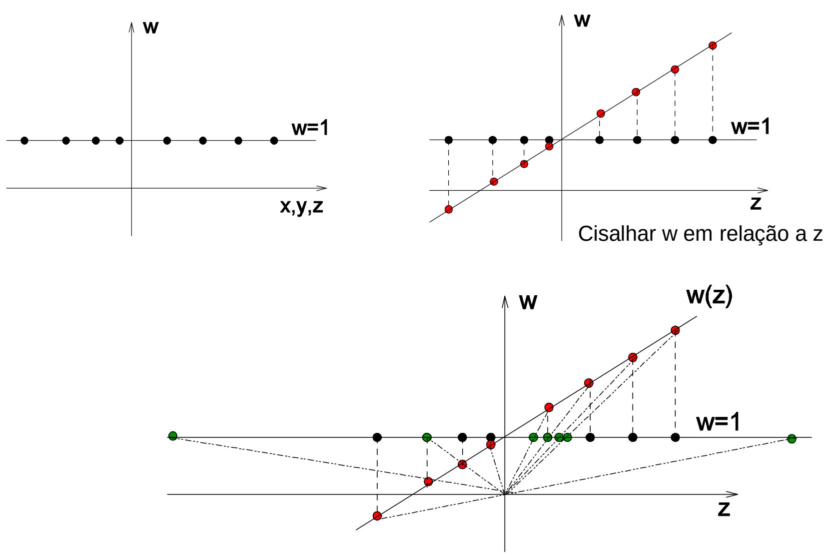
$n\,((a + t\,(b-a) - Q) = 0$

$$t = \frac{n\,(a - Q)}{n\,(a-b)}$$



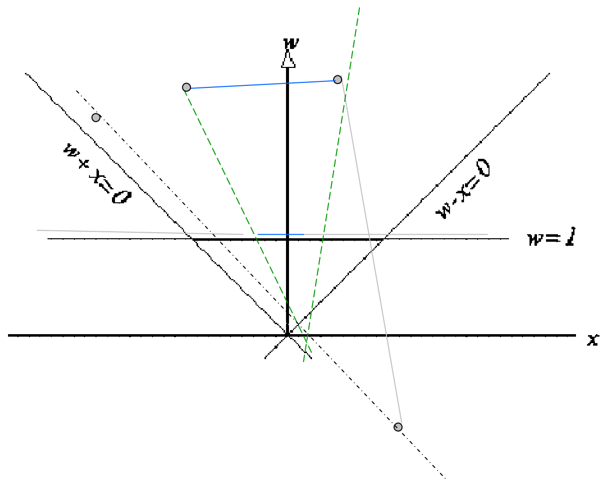
Projeções Perspectivas

Interpretação geométrica



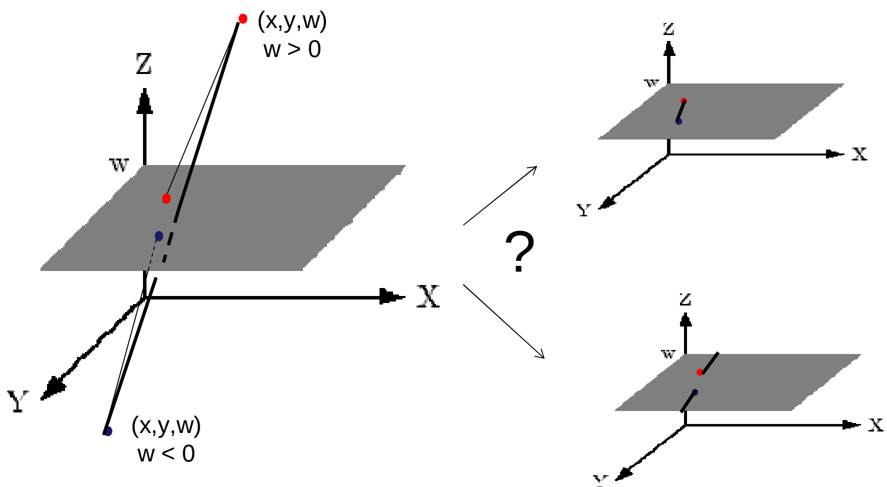
Recorte em 4D (w ≠ 1)

Recorte de um segmento pode resultar em 2 sub-segmentos!



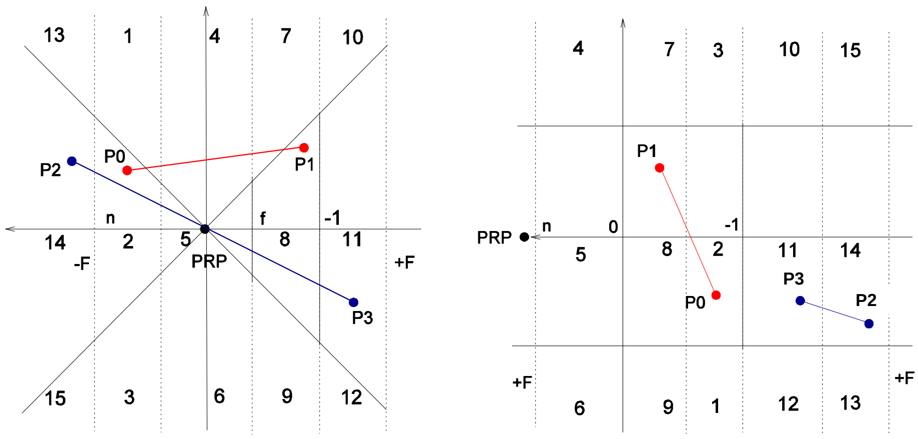
Recorte em 4D (w ≠ 1)

Após a divisão por w?



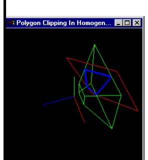
Recorte em 4D

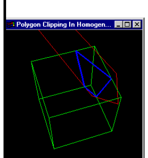
Após a divisão por w

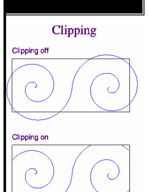


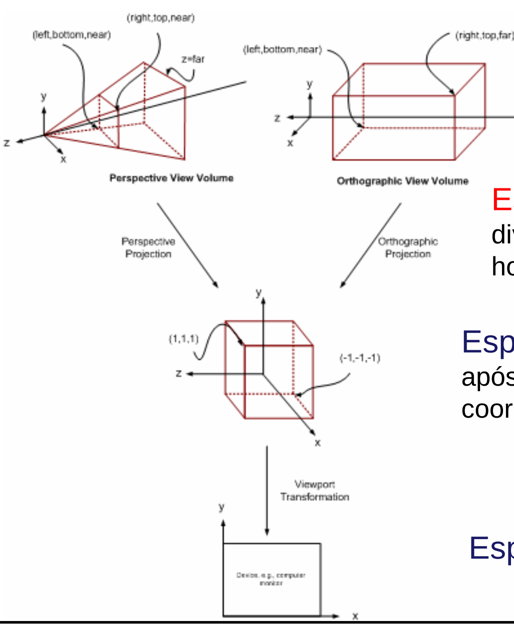
Em qual espaço?

w ≤ 0









Espaço da Câmera

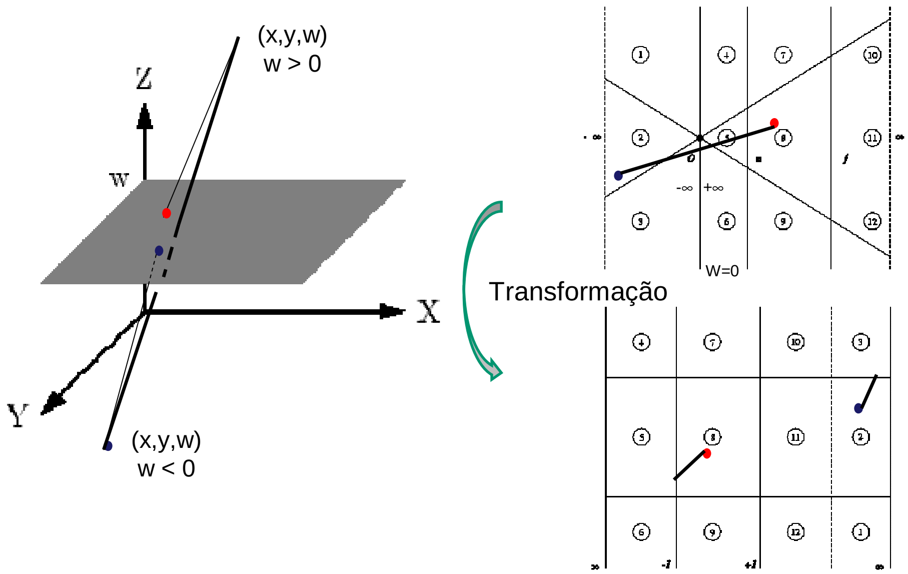
Espaço 4D: antes da divisão pela coordenada homogênea w

Espaço 3D Canônico: após a divisão pela coordenada homogênea w

Espaço 2D de Exibição

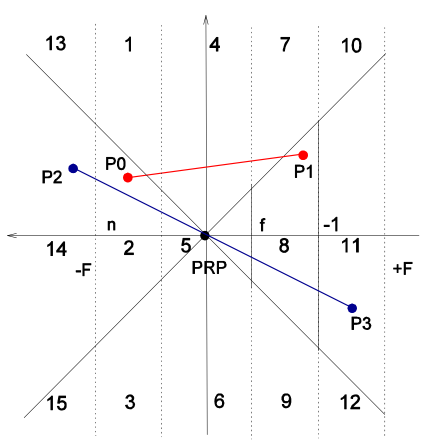
Recorte em 4D

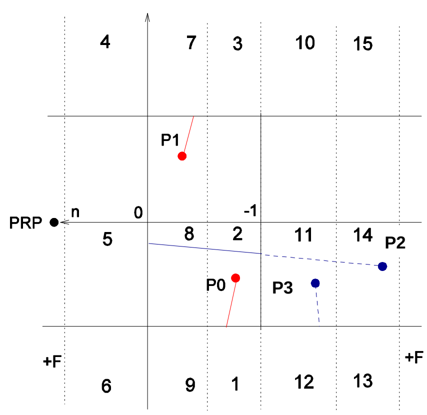
Antes da divisão por w



Recorte em 4D

Antes da divisão por w





Recorte em 4D

$$-1 \le \frac{x, y, z}{w} \le +1$$

$$-w \le x,y,z \le w$$

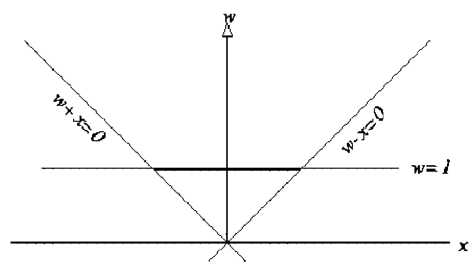
Dado: $P(t) = a + t\,(b-a)$

Interseção com x=-w:
$w_a + t\,(w_b - w_a) = -(x_a + t\,(x_b - x_a))$

$$t = \frac{w_a + x_a}{(w_a + x_a) - (w_b + x_b)}$$

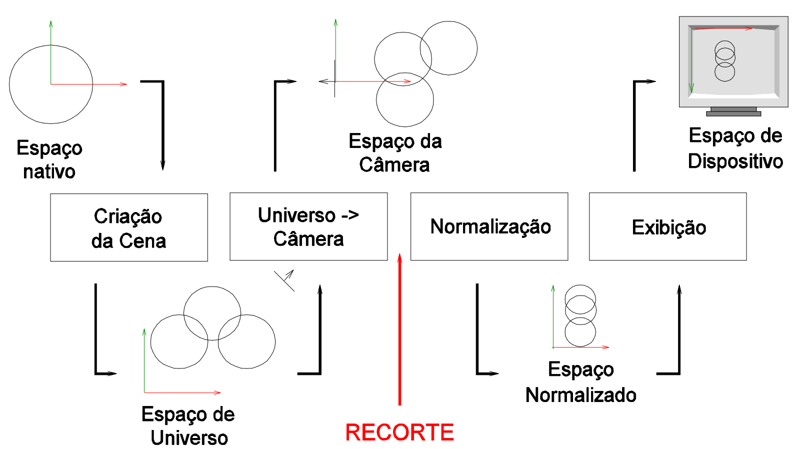
Analogamente, com
x=w, y=-w, y=w, z=-w, z=w

Possibilidades:
(entra,sai)
(sai,entra)



Fluxo de Projeção

Espaço de Recorte para w≠1



OpenGL

É possível selecionar um semi-espaço pela equação de plano eq:

$$ax + by + cz + d \geq 0$$

Um volume de recorte pode ser especificado como interseção de 6 semi-espaços:

```
glClippPlane(GL_CLIP_PLANEi,eq);
glEnable(GL_CLIP_PLANEi);
```

OpenGL

```
void display(void) {
        GLdouble eqn[4] = {0.0, 1.0, 0.0, 0.0};
        GLdouble eqn2[4] = {1.0, 0.0, 0.0, 0.0};
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f (1.0, 1.0, 1.0);
        glPushMatrix();
        glTranslatef (0.0, 0.0, -5.0); /* clip lower half -- y < 0 */
        glClipPlane (GL_CLIP_PLANE0, eqn);
        glEnable (GL_CLIP_PLANE0); /* clip left half -- x < 0 */
        glClipPlane (GL_CLIP_PLANE1, eqn2);
        glEnable (GL_CLIP_PLANE1);
        glRotatef (90.0, 1.0, 0.0, 0.0);
        glutWireSphere(1.0, 20, 16);
        glPopMatrix();
        glFlush ();
}
```

http://www.opengl.org/resources/code/samples/redbook/clip.c