

# Visualização 2D

- Pipeline de visualização 2D

- *Janela* (window)

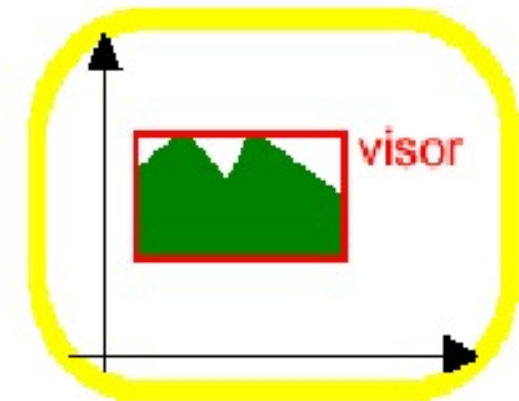
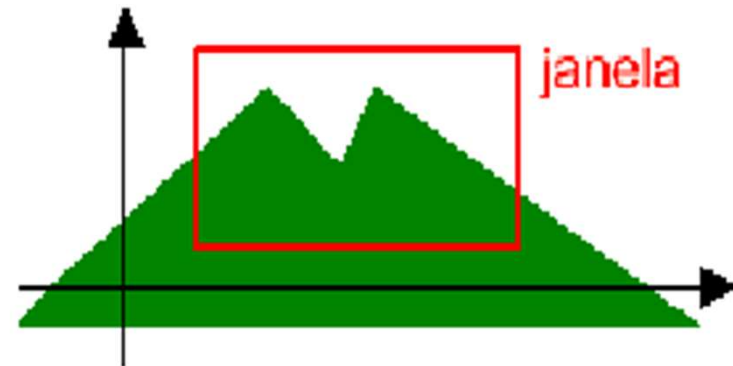
- Área da cena selecionada para visualização

## “O que deve ser visualizado”

- *Visor* (viewport)

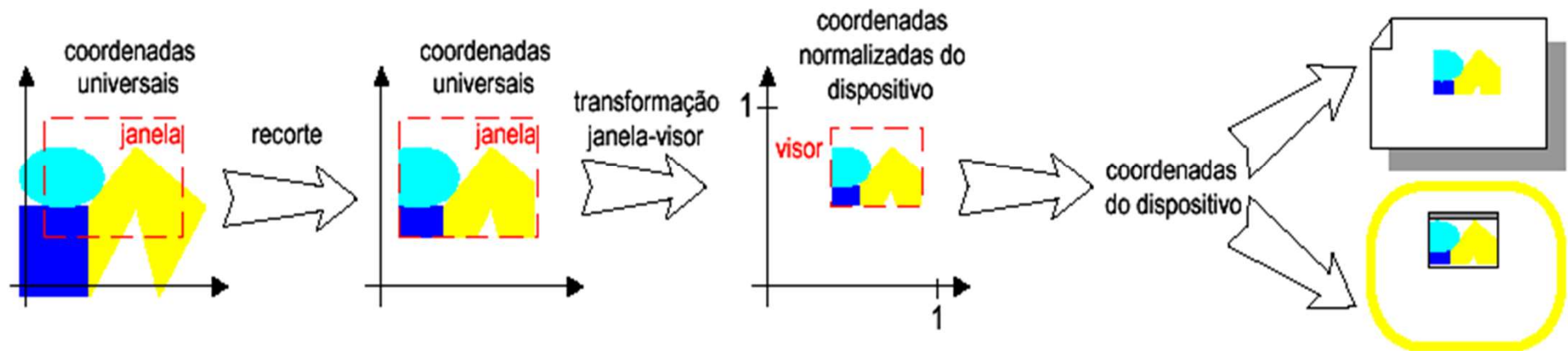
- Área do dispositivo de visualização selecionada para visualização da porção da cena contida na janela

## “Onde deve ser visualizado”



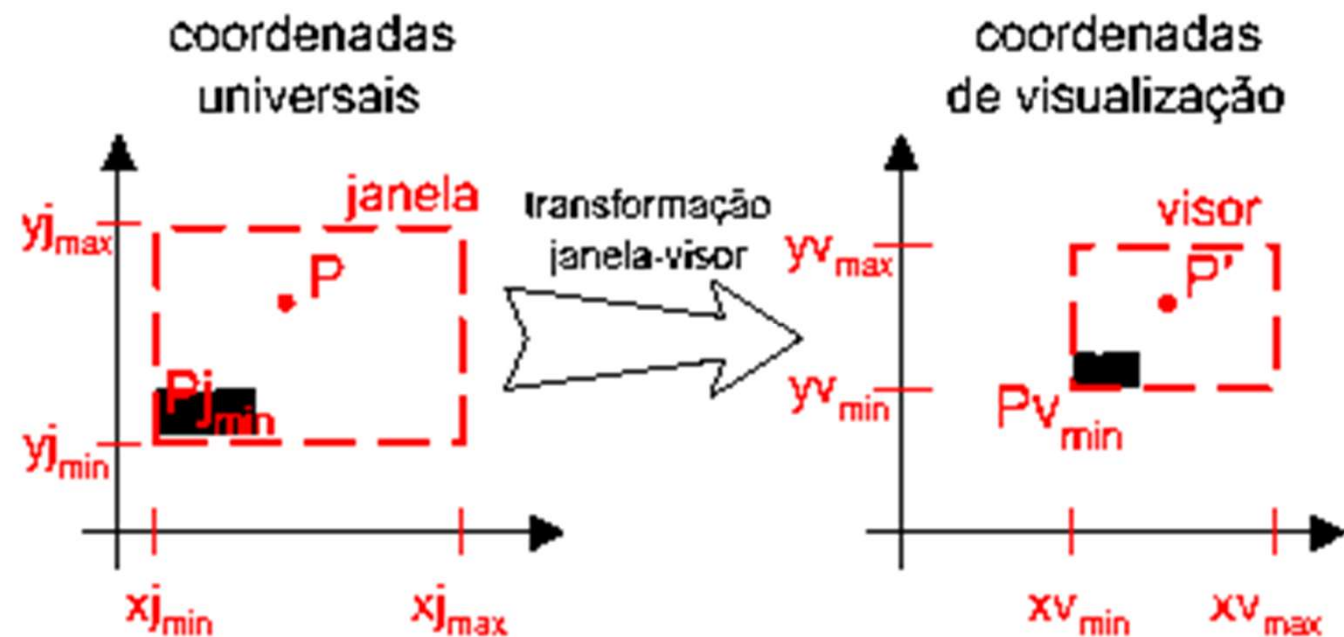
# Visualização 2D

- **Utilizam-se geralmente áreas retangulares**
  - Maior eficiência



# Visualização 2D

- Transformação janela-visor
  - Transformação que mapeia a janela para o visor, convertendo a representação dos objetos em coordenadas universais para as coordenadas de visualização.



# Visualização 2D

- De forma a manter as posições relativas entre os objetos.

$$\left\{ \begin{array}{l} \frac{x' - x_{v_{\min}}}{x_{v_{\max}} - x_{v_{\min}}} = \frac{x - x'_{\min}}{x'_{\max} - x'_{\min}} \\ \frac{y' - y_{v_{\min}}}{y_{v_{\max}} - y_{v_{\min}}} = \frac{y - y'_{\min}}{y'_{\max} - y'_{\min}} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} x' = x_{v_{\min}} + (x - x'_{\min}) \cdot \frac{x_{v_{\max}} - x_{v_{\min}}}{x'_{\max} - x'_{\min}} \\ y' = y_{v_{\min}} + (y - y'_{\min}) \cdot \frac{y_{v_{\max}} - y_{v_{\min}}}{y'_{\max} - y'_{\min}} \end{array} \right.$$

# Visualização 2D

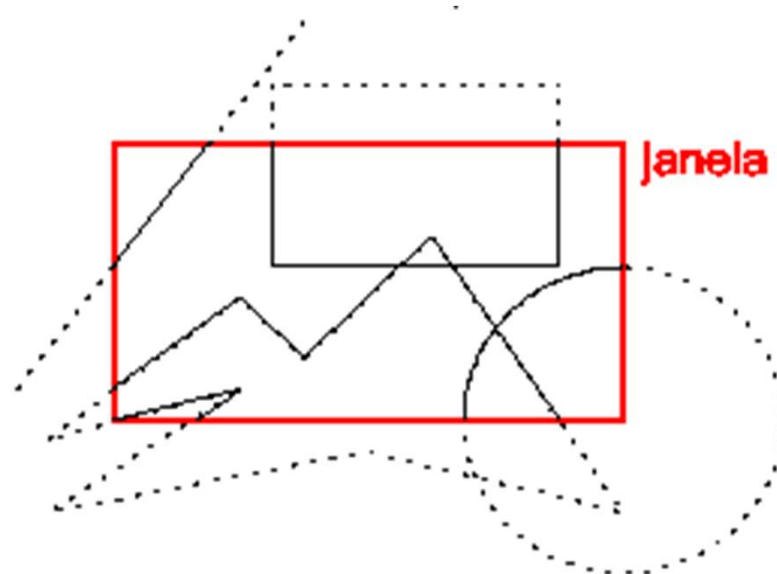
- Utilizando transformações geométricas
  - Translação da origem da janela para a origem do referencial.
  - Alteração de escala para o tamanho do visor.
  - Translação para a origem do visor.

$$P' = T(xv_{min}, yv_{min}) \cdot S \left( \frac{xv_{max} - xv_{min}}{xj_{max} - xj_{min}}, \frac{yv_{max} - yv_{min}}{yj_{max} - yj_{min}} \right) \cdot T(-xj_{min}, -yj_{min}) \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{xv_{max} - xv_{min}}{xj_{max} - xj_{min}} & 0 & xv_{min} - xj_{min} \cdot \frac{xv_{max} - xv_{min}}{xj_{max} - xj_{min}} \\ 0 & \frac{yv_{max} - yv_{min}}{yj_{max} - yj_{min}} & yv_{min} - yj_{min} \cdot \frac{yv_{max} - yv_{min}}{yj_{max} - yj_{min}} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

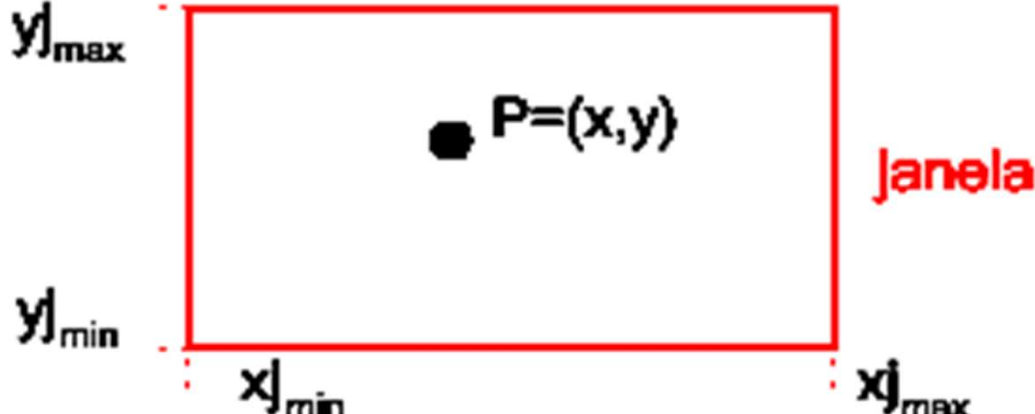
# Recorte 2D

- Recorte (clipping)
  - Operação que identifica as partes das primitivas de uma determinada cena que estão dentro ou fora de uma região do espaço.



# Recorte 2D

- Recorte de pontos
  - Um ponto situa-se dentro da janela de visualização se:

$$\begin{cases} x_{j_{\min}} \leq x \leq x_{j_{\max}} \\ y_{j_{\min}} \leq y \leq y_{j_{\max}} \end{cases}$$


The diagram illustrates a 2D rectangular window, labeled "janela" in red text. The window is defined by a red rectangular border. The horizontal axis is labeled with  $x_{j_{\min}}$  at the left edge and  $x_{j_{\max}}$  at the right edge. The vertical axis is labeled with  $y_{j_{\min}}$  at the bottom edge and  $y_{j_{\max}}$  at the top edge. A point  $P=(x,y)$  is shown as a black dot inside the rectangle, representing a point being tested for visibility within the window.

- Assumindo janela retangular

# Recorte 2D

- **Recorte de segmentos de reta**

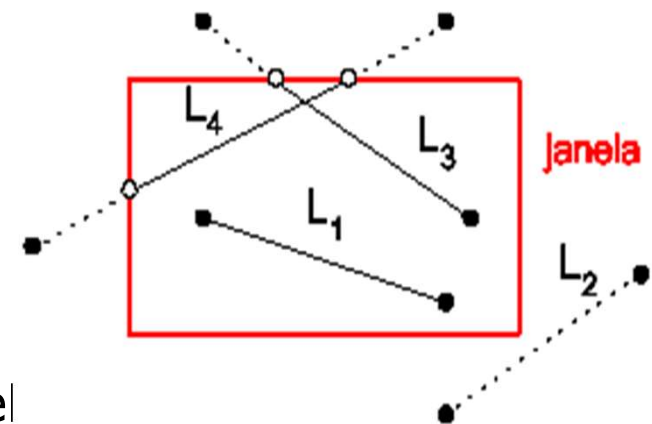
- Dois pontos definem a parte visível de um segmento de reta.

- Várias situações possíveis:

- $L_1$  : completamente dentro da janela de visualização.

- $L_2$  : fora da janela de visualização.

- $L_3$  e  $L_4$ : parcialmente dentro da janela de visualização.





# Recorte de segmentos de reta

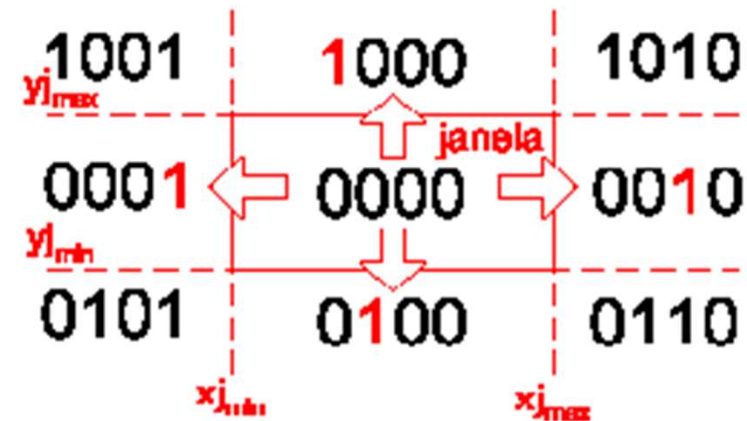
- Algoritmo de Cohen-Sutherland
  - Codificação da localização relativa dos extremos do segmento de reta em relação à janela.
  - Region code

Para cada ponto extremo  $P=(x,y)$

gera-se o código ativando os

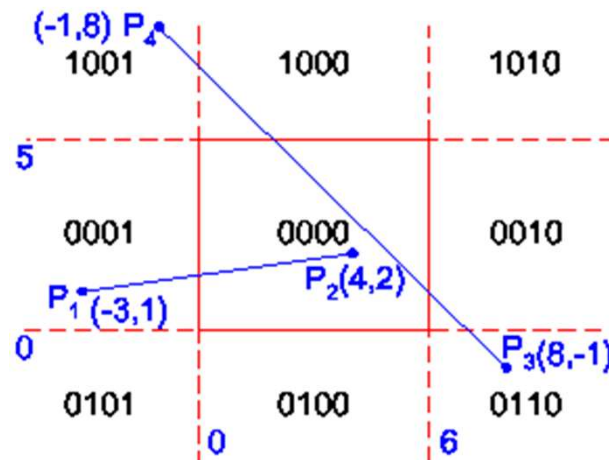
respectivos bits:

$$\begin{cases} x < x_{j_{\min}} \Rightarrow \_1 \\ x > x_{j_{\max}} \Rightarrow \_1\_ \\ y < y_{j_{\min}} \Rightarrow \_1\_ \\ y > y_{j_{\max}} \Rightarrow 1\_ \end{cases}$$



# Algoritmo de Cohen-Sutherland

- Verificação da localização do segmento de reta
  - Se ambos os extremos possuem código zero, o segmento está completamente dentro da janela.
  - Se ambos os extremos possuem um bit 1 na mesma posição, o segmento está completamente fora da janela.
  - Nos casos restantes, o segmento está parcialmente dentro da janela.



```
var xmin, xmax,ymin, ymax; {limites da janela}  
procedimento Cohen-Sutherland (x1,y1,x2,y2);  
início  
  aceite=Falso; feito=Falso;  
  enquanto não feito faça  
    c1=region_code(x1,y1);  
    c2=region_code(x2,y2);  
    se (c1=0) e (c2=0) então {segmento completamente dentro}  
      aceite=Verdade; feito=Verdade;  
    senão se (c1 e c2) <>0 {segmento completamente fora}  
      feito=Verdade;  
    senão  
      se (c1<>0) então {determina um ponto exterior}  
        cfora=c1;  
      senão  
        cfora=c2;  
    fim {se}
```

```

se (bit(cfora, 0)=1) então {limite esquerdo}
    xint=xmin; yint=y1+(y2-y1)*(xmin-x1)/(x2-x1);
senão se (bit(cfora, 1)=1) então {limite direito}
    xint=xmax; yint=y1+(y2-y1)*(xmax-x1)/(x2-x1);
senão se (bit(cfora, 2)=1) então {limite abaixo}
    yint=ymin; xint=x1+(x2-x1)*(ymin-y1)/(y2-y1);
senão se (bit(cfora, 3)=1) então {limite acima}
    yint=ymax; xint=x1+(x2-x1)*(ymax-y1)/(y2-y1);
fim {se}
se (c1=cfora) então
    x1=xint; y1=yint;
senão
    x2=xint; y2=yint;
fim {se} fim {se} fim {enquanto}
se (aceite) então
    desenha_linha(round(x1), round(y1), round(x2),round(y2));
    fim {se};
fim {procedimento Cohen-Sutherland}

```

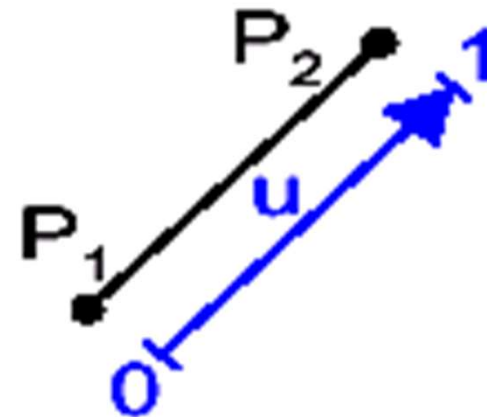
```
função region_code(x,y);  
var  
  código  
início  
  código=0;  
  
  {esquerda - bit 0}  
  se (x < xmin) então  
    código = código +1;  
  fim {se}  
  {direita- bit 1}  
  se (x > xmax) então  
    código = código +2;  
  fim {se}  
  {baixo - bit 2}  
  se (y < ymin) então  
    código = código +4;  
  fim {se}  
  {cima - bit 3}  
  se (y > ymax) então  
    código = código +8;  
  fim {se}  
  
  retorna código;  
fim {função region_code}
```

# Recorte de segmentos de reta

- Algoritmo de Liang-Barsky
  - Utilização da equação paramétrica da reta

$$\begin{cases} x = x_1 + (x_2 - x_1).u = x_1 + \Delta x.u \\ y = y_1 + (y_2 - y_1).u = y_1 + \Delta y.u \end{cases}$$

- Variando  $u$  de 0 a 1 obtêm-se todos os pontos do segmento de reta de  $\mathbf{P}_1$  a  $\mathbf{P}_2$ .



# Algoritmo de Liang-Barsky

- Para um determinado ponto  $(x,y)$  de um segmento de reta, que se encontre dentro de uma janela:

$$\begin{cases} xj_{\min} \leq x1 + \Delta x \cdot u \leq xj_{\max} \\ yj_{\min} \leq y1 + \Delta y \cdot u \leq yj_{\max} \end{cases}$$

# Algoritmo de Liang-Barsky

- Podemos escrever equações como:

$$p_k \cdot u \leq q_k \quad k = 1, 2, 3, 4$$

$$\left\{ \begin{array}{ll} p_1 = -\Delta x, & q_1 = x_1 - xj_{\min} \\ p_2 = \Delta x, & q_2 = xj_{\max} - x_1 \\ p_3 = -\Delta y, & q_3 = y_1 - yj_{\min} \\ p_4 = \Delta y, & q_4 = yj_{\max} - y_1 \end{array} \right\} \begin{array}{l} \textit{esquerda} \\ \textit{direita} \\ \textit{abaixo} \\ \textit{acima} \end{array}$$



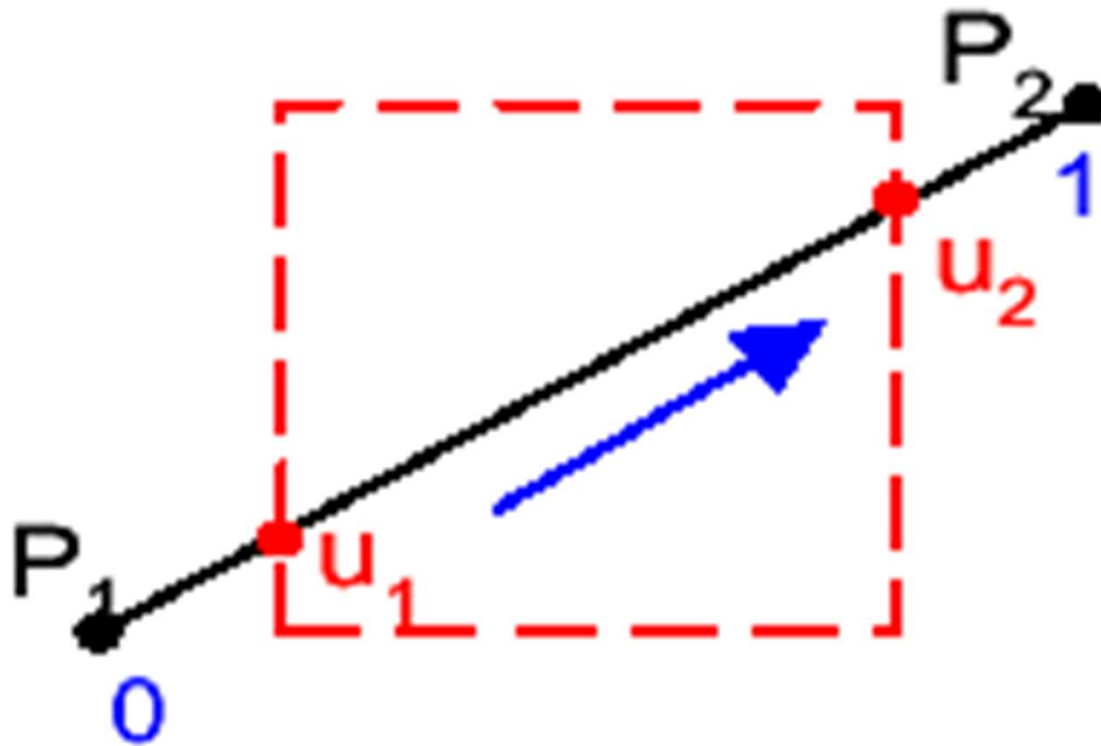
# Algoritmo de Liang-Barsky

- Se  $p_k = 0$ , a linha é paralela à fronteira  $k$ 
  - Se  $q_k < 0$  a linha está completamente fora da janela
- Se  $p_k < 0$  a linha segue de fora para dentro
- Se  $p_k > 0$  a linha segue de dentro para fora

# Algoritmo de Liang-Barsky

- Para  $p_k \neq 0$ , o ponto de interseção com a fronteira é  $u_k = q_k / p_k$
- Para cada segmento de reta, calculam-se os valores de  $u_1$  e  $u_2$ , que definem a parte da reta interior à janela
  - O valor  $u_1$  é determinado em relação às fronteiras, onde a linha segue de fora para dentro e  $u_2$ , de dentro para fora.

# Algoritmo de Liang-Barsky



# Recorte de segmentos de reta

## Algoritmo de Liang-Barsky

```
var xmin, xmax, ymin, ymax; {limites da janela}  
procedimento Liang-Barski (x1,y1,x2,y2);
```

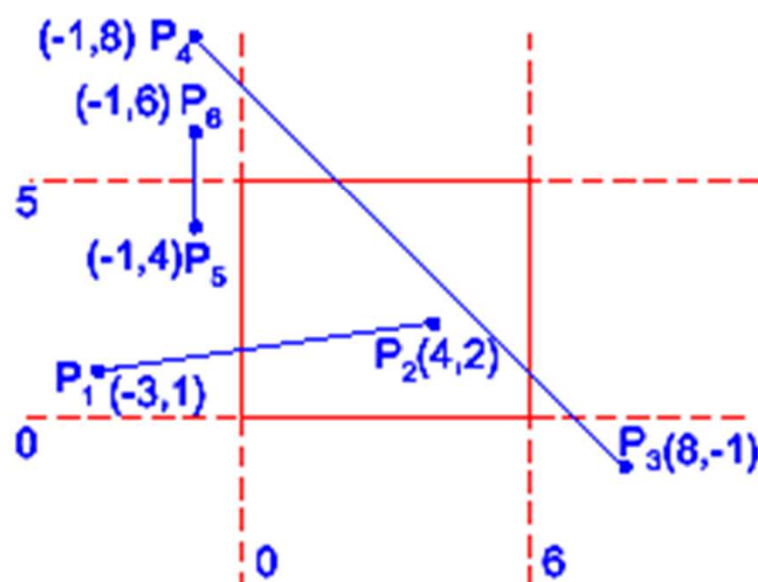
```
    função result=cliptest(p, q, u1, u2);  
    início  
    ...  
    fim {função cliptest}
```

```
função cliptest(p, q, u1, u2);  
início  
    result=Verdade;  
    se (p<0.0) então {fora para dentro}  
        r=q/p;  
        se (r>u2) então result=Falso;  
        senão se (r>u1) então u1=r;  
    fim {se}  
    senão se (p>0.0) então {dentro para fora}  
        r=q/p;  
        se (r<u1) então result=Falso;  
        senão se (r<u2) então u2=r;  
    fim {se}  
    senão se (q<0.0) então result=Falso;  
    fim {se}  
    retorna result;  
fim {função cliptest}
```

```

Início
u1=0.0; u2=1.0;
dx = x2-x1;
dy = y2-y1;
se (cliptest(-dx, x1-xjmin, u1, u2)) então
  se (cliptest(dx, xmax - x1, u1, u2)) então
    se (cliptest(-dy, y1 - yjmin, u1, u2)) então
      se (cliptest(dy, ymax - y1, u1, u2)) então
        se (u2 < 1.0) então
          x2 = x1 +u2*dx;
          y2 = y1 +u2*dy;
        fim {se}
        se (u1 > 0.0) então
          x1 = x1 +u1*dx;
          y1 = y1 +u1*dy;
        fim {se}
        desenha_linha(round(x1),round(y1),round(x2),round(y2));
      fim {se}
    fim {se}
  fim {se}
fim {se}
fim {procedimento Liang-Barski}

```



# Recorte de Polígonos

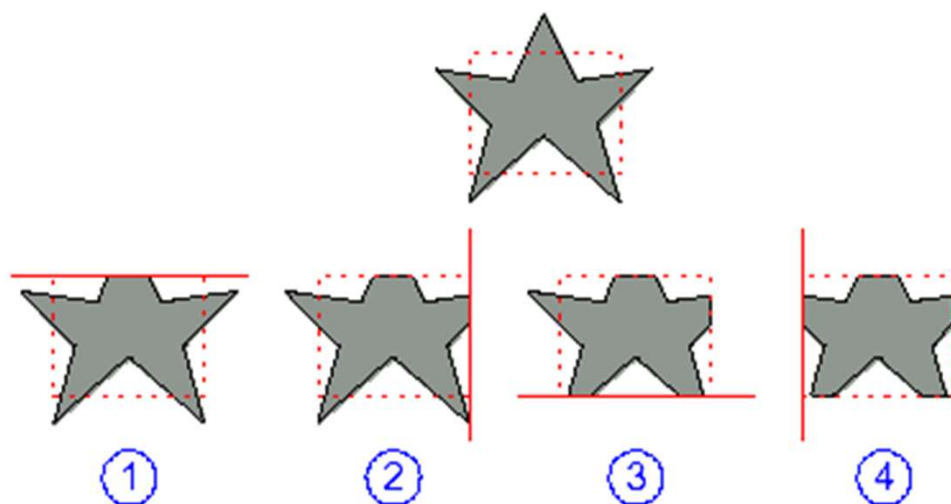
- Um polígono é constituído por vários segmentos de reta.
  - recorte dos segmentos de reta individuais.
    - não se obtém uma área fechada

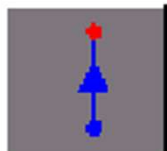
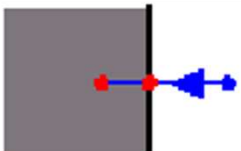
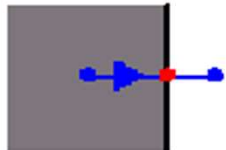



# Recorte de Polígonos

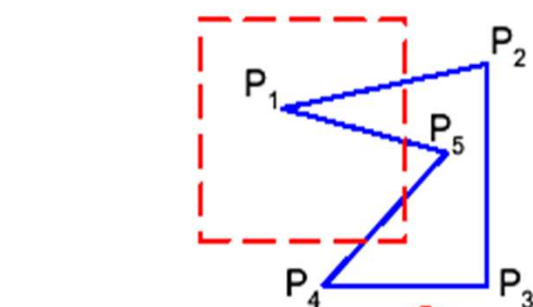
- Algoritmo Sutherland-Hodgeman
  1. Fazer o recorte do polígono com cada uma das retas que delimitam a janela de visualização.
  2. A área do polígono é especificada por uma lista ordenada de vértices.
  3. A lista de vértices é percorrida segundo uma determinada ordem, e é atualizada conforme a posição de cada vértice e do seu antecessor.

# Recorte de Polígonos



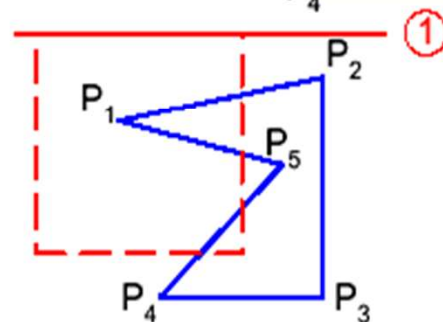
|               |                 | vértice anterior   |   |
|---------------|-----------------|--|---|
|               |                 | <i>interior</i>  | <i>exterior</i>   |
| vértice atual | <i>interior</i> |  <p>junta vértice atual à lista</p> |  <p>junta interseção e vértice atual à lista</p> |
|               | <i>exterior</i> |  <p>junta interseção à lista</p>   |   |



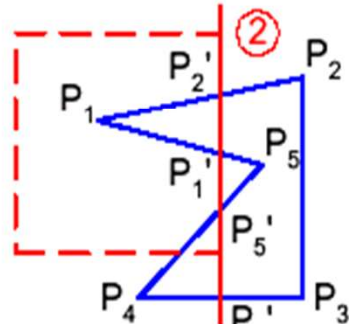


Lista de Vértices

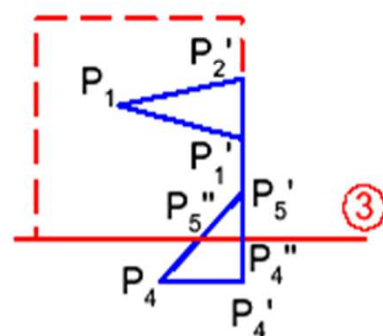
$P_1, P_2, P_3, P_4, P_5$



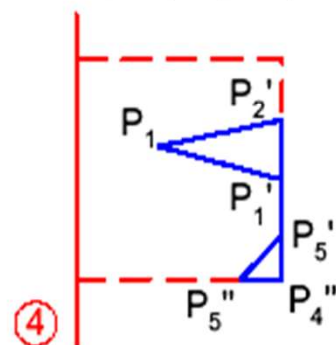
$P_1, P_2, P_3, P_4, P_5$



$P_1', P_1, P_2', P_4', P_4, P_5'$



$P_1', P_1, P_2', P_4'', P_5'', P_5'$



$P_1', P_1, P_2', P_4'', P_5'', P_5'$

**função** recorta\_poligono (vértices);

**início**

**para** fronteira de 1 até 4 **faça**

**início**

nova\_lista={};

Vant=vértices[comp(vértices)];

**para** i de 1 até comp(vértices) **faça**

**início**

**se** (interior(vértices[i], fronteira)) **então**

{vértice interior}

**se** (não interior(Vant, fronteira)) **então**

Vint=intersecção(Vant, vértices[i], fronteira);

adiciona(nova\_lista, Vint);

**fim** {se}

adiciona(nova\_lista, vértices[i]);

**senão**

{vértice exterior}

**se** (interior(Vant, fronteira)) **então**

Vint=intersecção(Vant, vértices[i], fronteira);

adiciona(nova\_lista, Vint);

**fim** {se}

**fim** {se}

Vant=vértices[i];

**fim** {para}

vértices=nova\_lista;

**fim** {para}

**retorna** nova\_lista;

**fim** {procedimento recorta\_poligono}