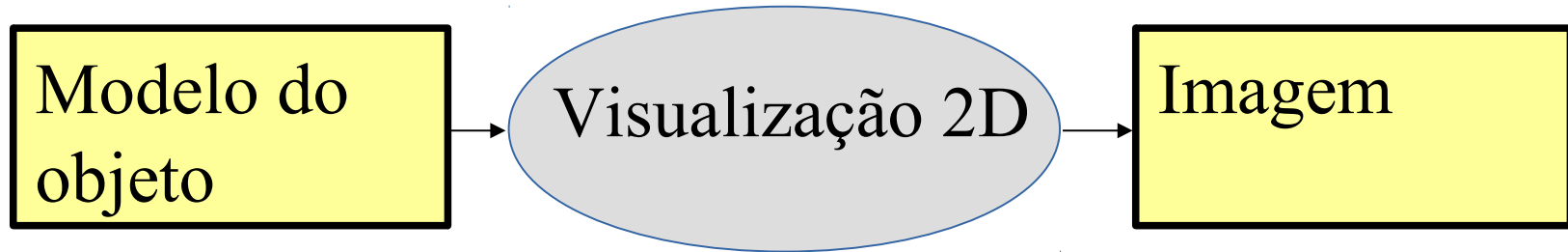
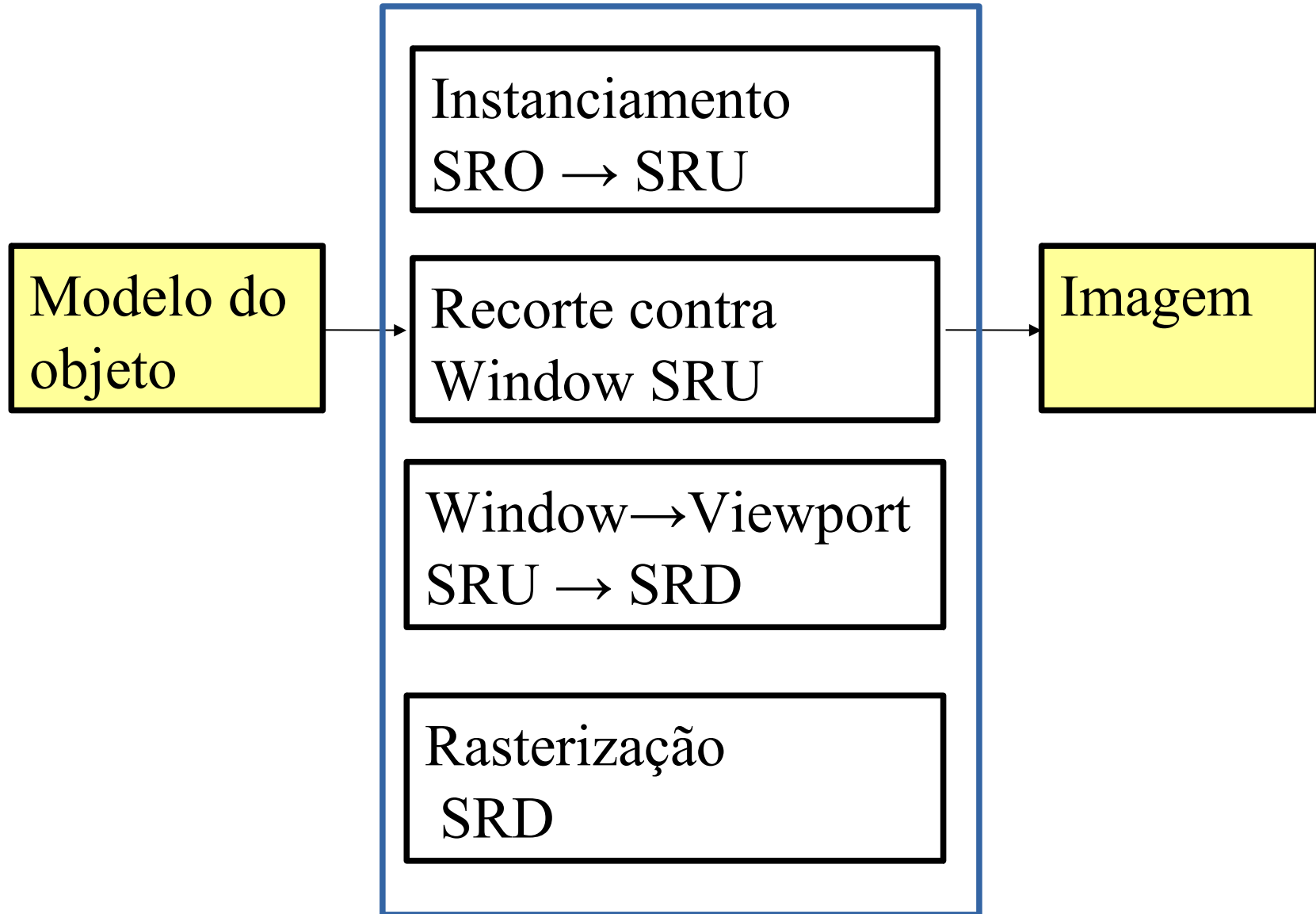


# PROCESSO DE VISUALIZAÇÃO 2D



# PROCESSO DE VISUALIZAÇÃO 2D



# PROCESSO DE VISUALIZAÇÃO 2D

1. Transformações de Instanciamento (escala, rotação, translação)
2. Recorte dos objetos contra a Window
3. Mapeamento Window-Viewport
4. Conversão de Varredura (rasterização)

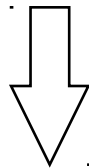
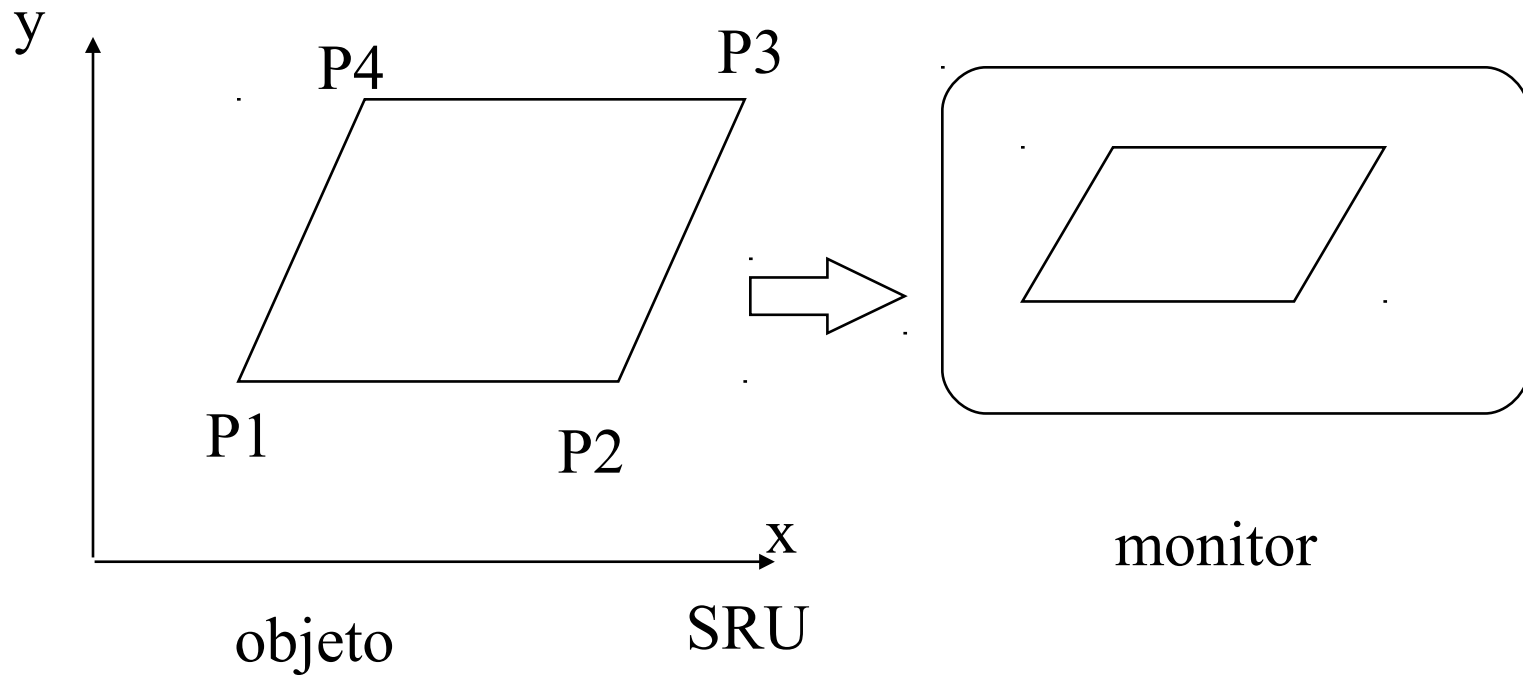


Imagem na tela



# VISUALIZAÇÃO 2D

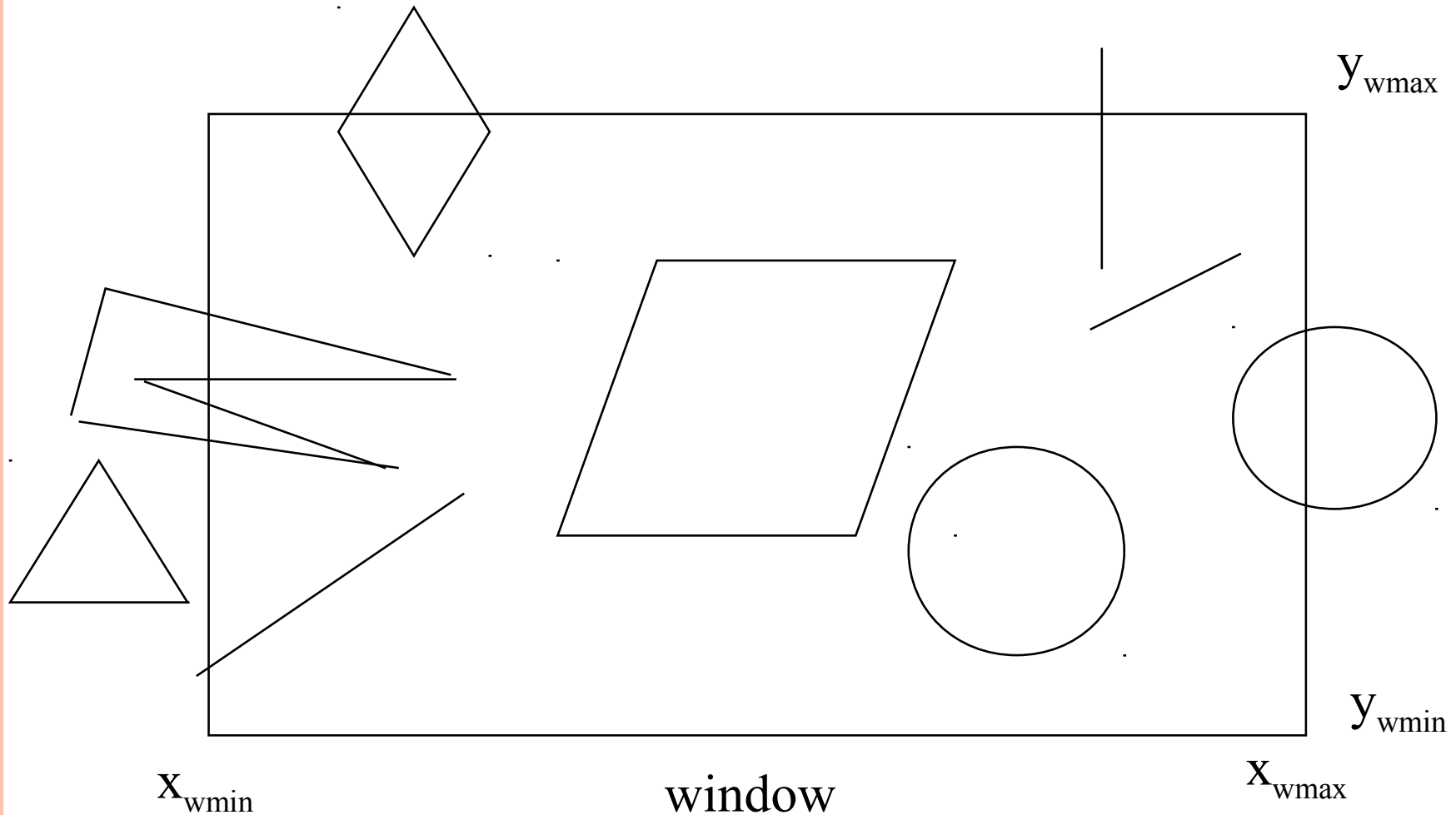


# RECORTE

- Por que? Não perder tempo rasterizando objetos fora da janela de visualização!
- Eliminação dos **objetos geométricos** que estão fora da window
- Eliminação das **porções dos objetos geométricos** que estão fora da window
- Algoritmos específicos: Pontos, Linhas e Polígonos
- Recorte em hardware
- Para otimizar o desempenho: envelope para objetos complexos



# RECORTE



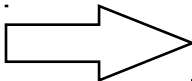
O recorte de um polígono convexo produz no máximo um polígono convexo.  
O recorte de um polígono côncavo pode produzir mais de um polígono.



# RECORTE

- Recorte de pontos

- $x_{wmin} \leq x \leq x_{wmax}$
- $y_{wmin} \leq y \leq y_{wmax}$

  $(x,y)$  é visível

- Recorte de linhas

- Linhas que jazem nas bordas da window são consideradas dentro da window e são exibidas.
- Para recortar uma linha contra a window:
  - Se os dois vértices estão dentro então a linha está dentro.
  - Se apenas um está dentro da window é necessário calcular uma intersecção da linha com a window.
  - Se os dois vértices estão fora, então a linha pode estar fora ou parcialmente dentro.



# RECORTE DE LINHAS

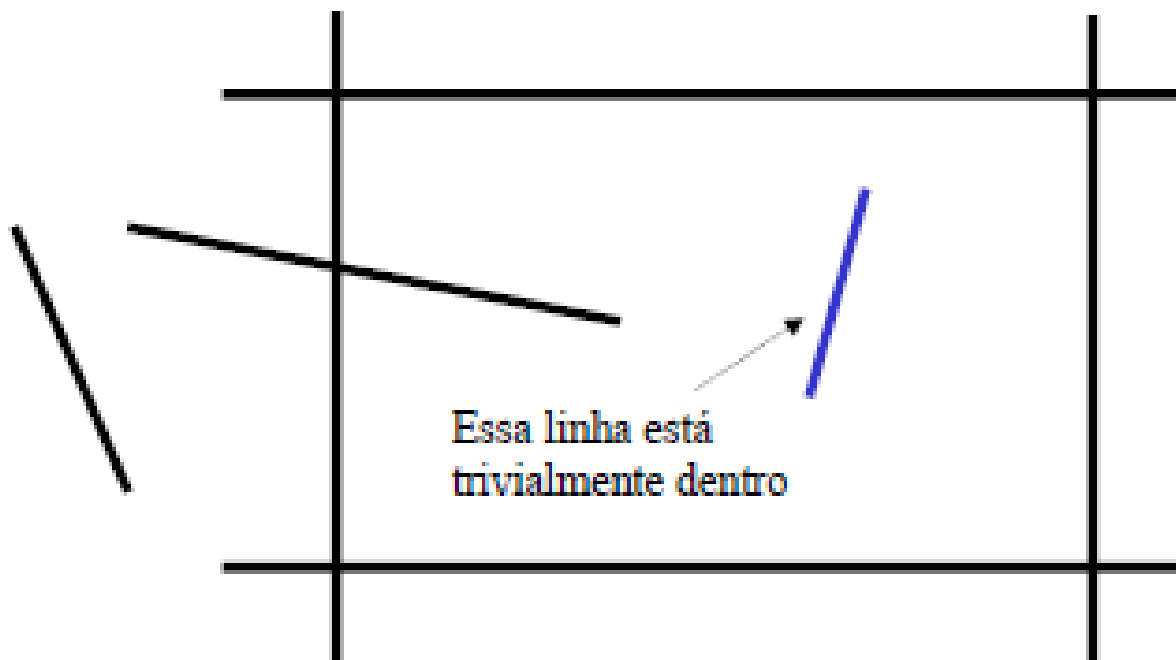
- Algoritmo Cohen-Sutherland
- Algoritmo de Solução de Equações Simultâneas
- Algoritmo Cyrus-Beck
  - Para todos os algoritmos: Primeiro testar aceitação e rejeição trivial para otimização





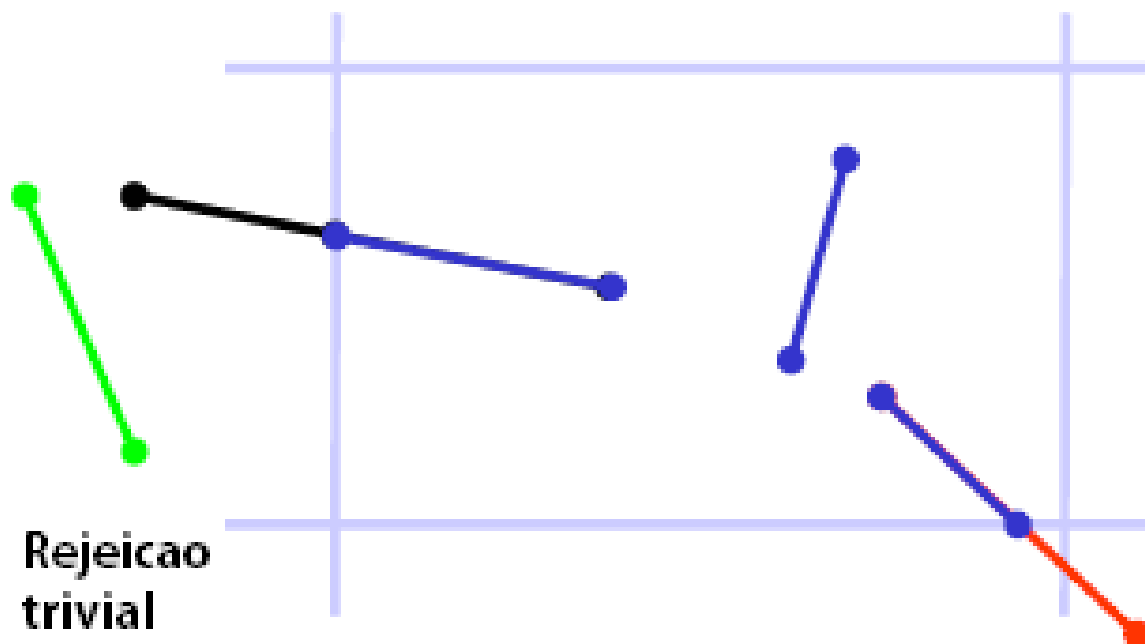
# RECORTE DE LINHAS

- Aceitação trivial



# RECORTE DE LINHAS

- Rejeição trivial
  - Os vértices estão do lado de fora da mesma aresta



## RECORTE - ALGORITMO DE EQUAÇÕES SIMULTÂNEAS – 1/2

- Window:  $(x_{\min}, y_{\min})$  e  $(x_{\max}, y_{\max})$  no SRU
- Reta:  $P_0 = (x_0, y_0)$  a  $P_1 = (x_1, y_1)$  no SRU
- Para cada borda da window (esquerda, direita, inferior, superior):
  - Calcular a intersecção com a borda e trocar vértice original pela intersecção, se necessário
    - Borda esquerda:

$$x_{\text{intersec}} = x_{\min};$$

$$m = (y_1 - y_0) / (x_1 - x_0);$$

$$y_{\text{intersec}} = y_0 + m * (x_{\text{intersec}} - x_0);$$

Se a intersecção ocorrer entre os limites superior e inferior da Window, então: Troca-se  $P_0$  pelo ponto recém calculado  $P_{\text{intersec}}$ ;

- Casos especiais: linhas horizontais e verticais.



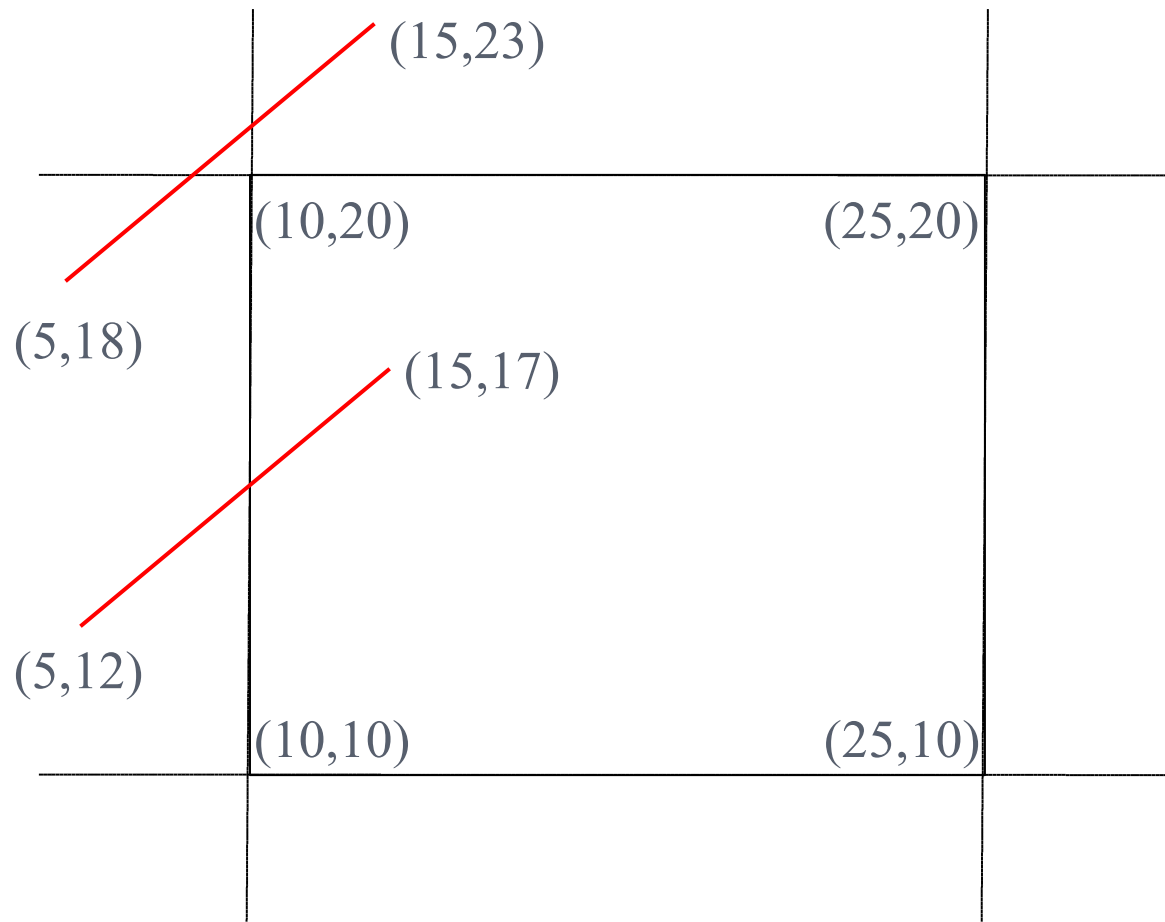
## RECORTE - ALGORITMO EQUAÇÕES SIMULTÂNEAS – 2/2

- Utiliza equação da reta envolvendo as bordas da janela e a própria linha.
- Grande quantidade de cálculos e teste e **não é eficiente**.
- Num display temos centenas ou milhares de linhas.
- Um algoritmo eficiente deve realizar alguns testes iniciais para determinar se cálculos de interseção são necessários.
- Inicialmente, aceitar ou rejeitar trivialmente a linha.



## EXERCÍCIO

- Utilizar o Algoritmo de Equações Simultâneas para recortar as linhas:



## RECORTE - ALGORITMO DE COHEN-SUTHERLAND

- Se a linha não cai nos casos triviais, subdividi-la de forma que um ou os dois segmentos possam ser descartados
  - Selecione uma aresta da janela de visualização que a linha cruza, arestas testadas sempre na mesma ordem
  - Ache a interseção da linha com a aresta
  - Descarte a parte do segmento do lado de fora da aresta e atribua novo código ao novo vértice
  - Aplique os testes triviais; repetidamente se necessário

1001	0001	0101
1000	0000	0100
1010	0010	0110



## RECORTE - ALGORITMO DE COHEN-SUTHERLAND – 1/5

- Os vértices do segmento são classificados em relação a cada semi-plano que delimita a janela, gerando um código de 4 bits:
  - $\text{Bit1} = (x < x_{\min})$
  - $\text{Bit2} = (x > x_{\max})$
  - $\text{Bit3} = (y < y_{\min})$
  - $\text{Bit4} = (y > y_{\max})$
- Se ambos os vértices forem classificados como fora em um mesmo semi-plano, descartar o segmento (rejeitado trivialmente)
- Se ambos forem classificados como dentro, testar o próximo semi-plano
- Se um vértice estiver dentro e outro fora, computar o ponto de interseção  $Q$  e continuar o algoritmo com o segmento recortado ( $P_0$ - $Q$  ou  $P_1$ - $Q$ ) que está parcialmente dentro.



## RECORTE ALGORITMO DE COHEN-SUTHERLAND – 2/5

- Código de cada vértice composto por 4 bits, um bit para cada semi-plano
  - $\text{in} = 0$  e  $\text{out} = 1$
- Rejeição trivial:
  - $\text{Código}(P_0) \& \text{Código}(P_1) = 1$
- Aceitação trivial:
  - $\text{Código}(P_0) | \text{Código}(P_1) = 0$
- Senão
  - Intersecção com semi-planos

1001	0001	0101
1000	0000	0100
1010	0010	0110





## RECORTE - ALGORITMO DE COHEN-SUTHERLAND – 3/5

### Intersecção com semi-planos

1) Se  $P_0$  estiver fora da Window então vá para o passo 2.

Senão troque  $P_0$  com  $P_1$ .

2) Se  $P_0$  está à esquerda da window ( $1^\circ \text{ bit} = 1$ ):

Calcular  $P_i$ , intersecção de  $P_0$ - $P_1$  com o lado  
esquerdo da window.

$$P_0 = P_i.$$

Vá para o passo 6.

Senão vá para passo 3.



## RECORTE ALGORITMO DE COHEN-SUTHERLAND – 4/5

3) Se  $P_0$  está à direita da window (2º bit = 1):

Calcular  $P_i$ , intersecção de  $P_0$ - $P_1$  com o lado direito da window.

$$P_0 = P_i.$$

Vá para o passo 6.

Senão vá para passo 4.

4) Se  $P_0$  está abaixo da window (3º bit = 1):

Calcular  $P_i$ , intersecção de  $P_0$ - $P_1$  com o lado inferior da window.

$$P_0 = P_i.$$

Vá para o passo 6.

Senão vá para passo 5.



## RECORTE ALGORITMO DE COHEN-SUTHERLAND - 5/5

5) Se  $P_0$  está acima da window (4º bit = 1):

Calcular  $P_i$ , intersecção de  $P_0$ - $P_1$  com o lado superior da window.

$$P_0 = P_i.$$

Vá para o passo 6.

Senão vá para passo 6.

6) Recalcule o código de  $P_0$ .

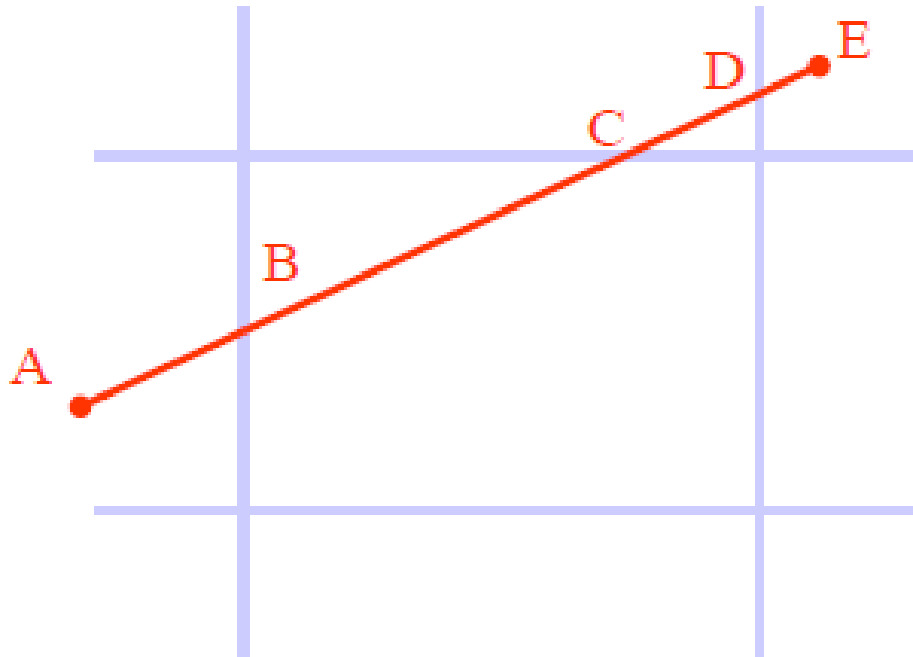
Se  $(P_0 \text{ OR } P_1) = 0$  então linha é  $P_0$ - $P_1$  e encerra o algoritmo.

senão Se  $(P_0 \text{ AND } P_1) \neq 0$  então linha está fora da window e encerra o algoritmo senão volta ao passo 1.



# EXEMPLO

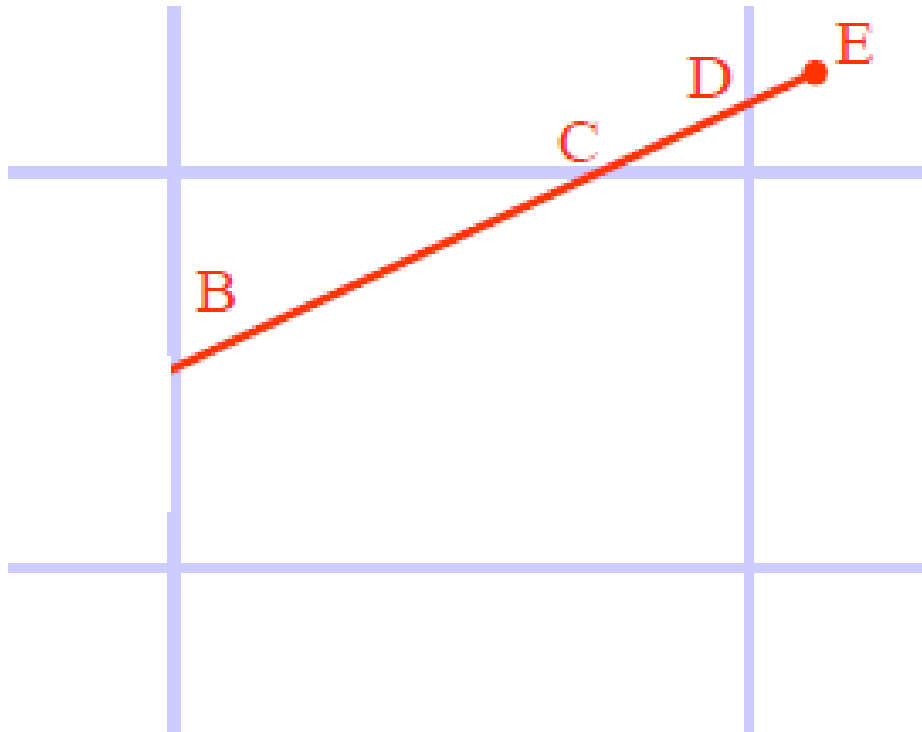
## . Recorte de Cohen-Sutherland



# EXEMPLO

## . Recorte de Cohen-Sutherland

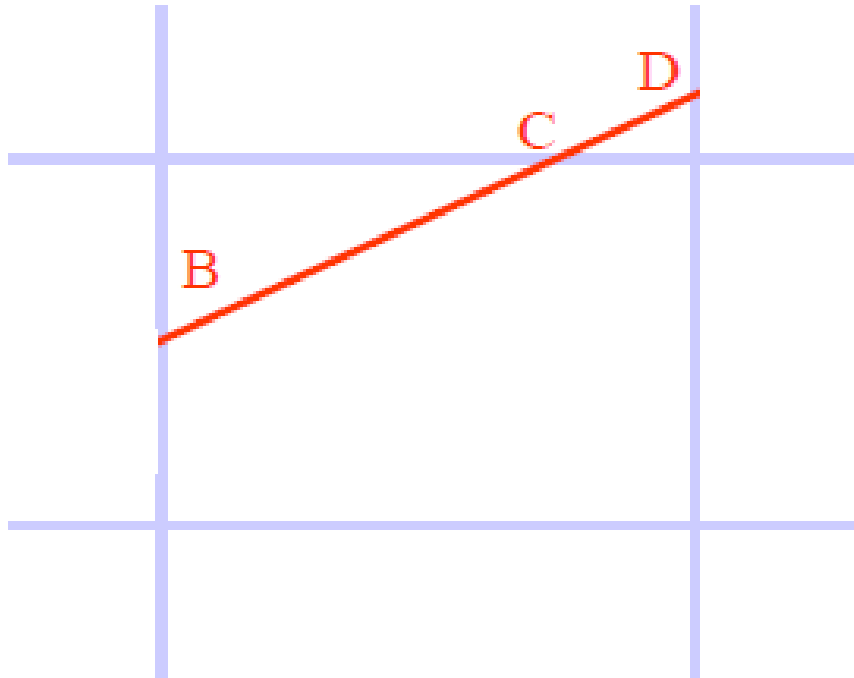
Recorte borda esquerda



# EXEMPLO

## . Recorte de Cohen-Sutherland

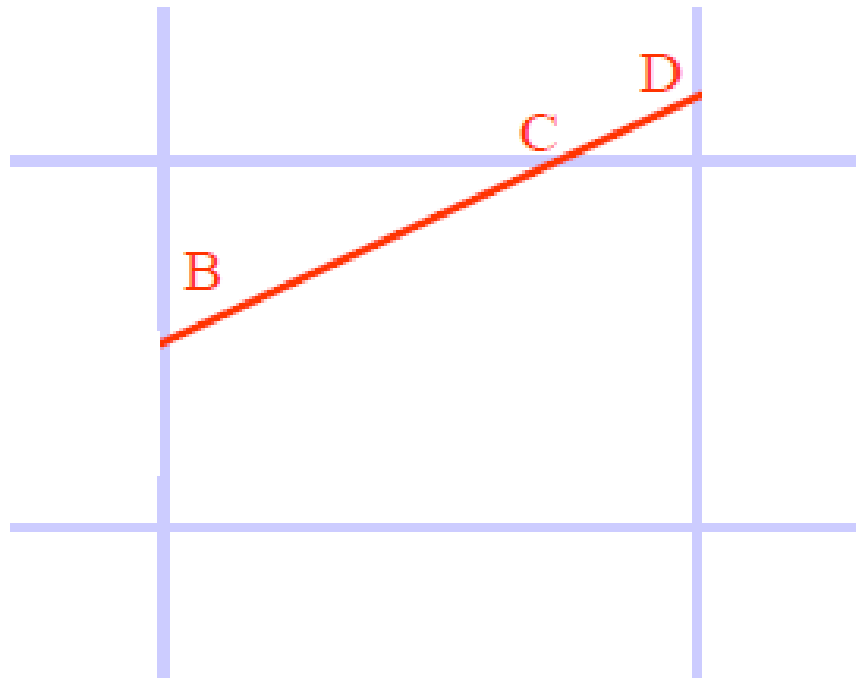
Recorte borda direita



# EXEMPLO

## . Recorte de Cohen-Sutherland

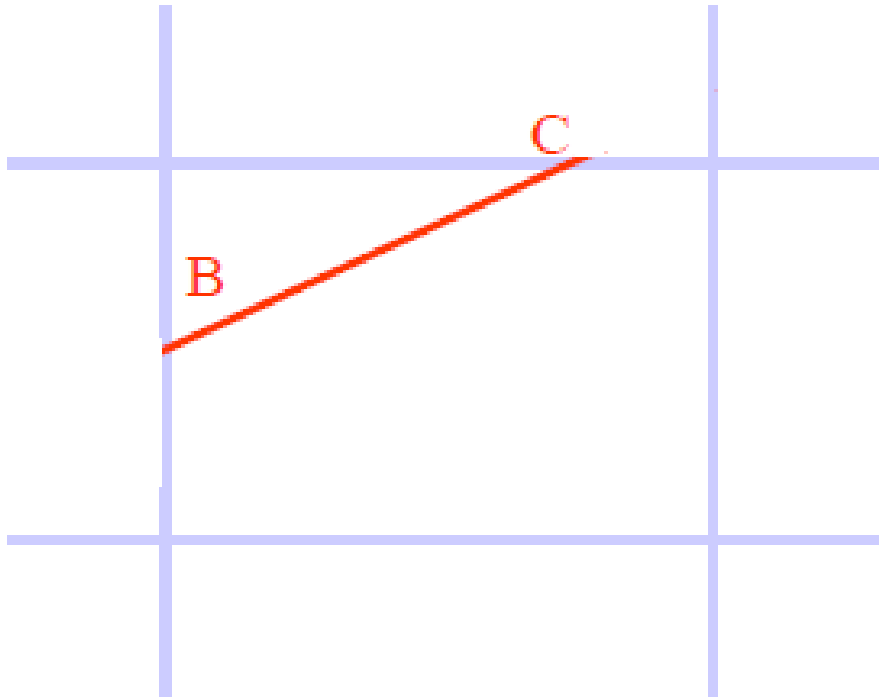
Recorte borda inferior



# EXEMPLO

## . Recorte de Cohen-Sutherland

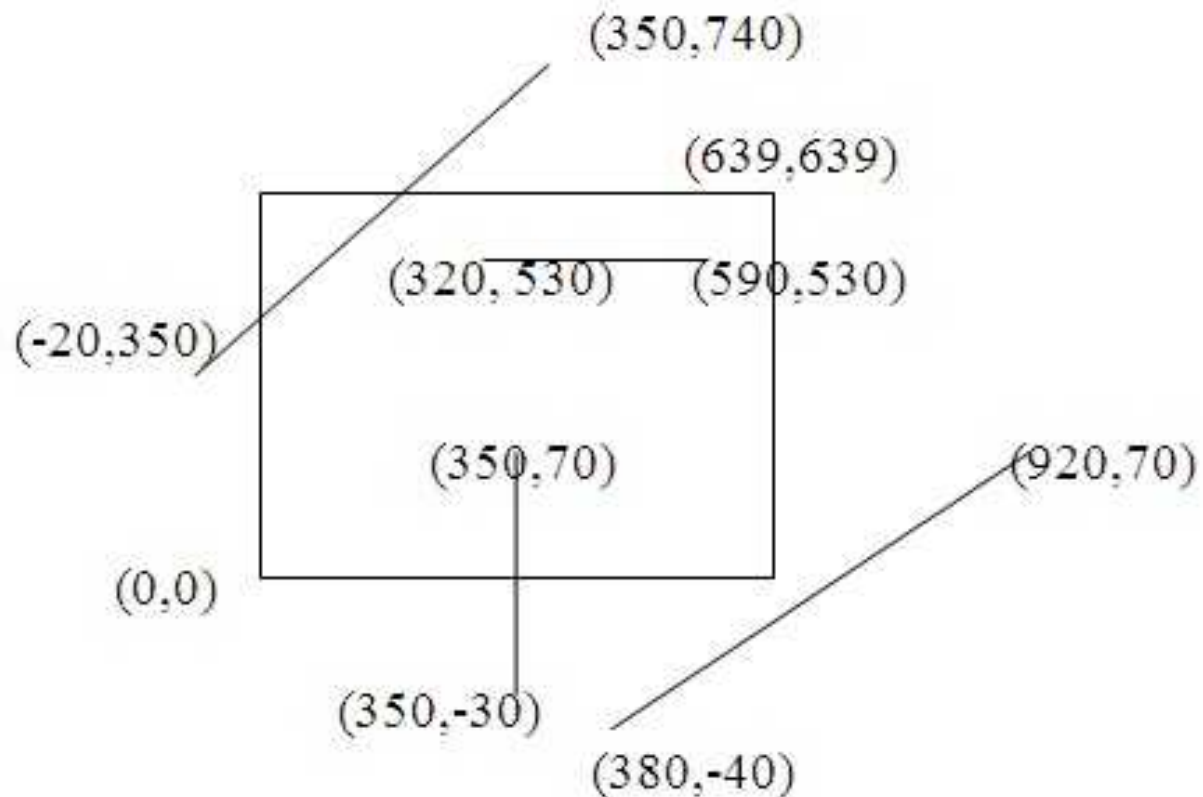
Recorte borda superior





## EXERCÍCIO

- . Suponha a janela abaixo e as linhas. Encontre, usando o recorde de Cohen-Sutherland, as porções visíveis de cada uma das retas:



# CÁLCULO DA INTERSECÇÃO

## Equações de reta

- Explícita:  $y = mx + b$
- Implícita:  $Ax + By + C = 0$
- Paramétrica: linha definida por 2 pontos,  $P0$  e  $P1$ 
  - $P(t) = P0 + (P1 - P0)t$
  - $x(t) = x0 + (x1 - x0)t$
  - $y(t) = y0 + (y1 - y0)t$



# CÁLCULO DA INTERSECÇÃO

Equação paramétrica de reta

- Descreve segmento (linha finita)

- $0 \leq t \leq 1$ , define linha entre P0 e P1
- $t < 0$ , define linha antes de P0
- $t > 1$ , define linha depois de P1



# CÁLCULO DA INTERSECÇÃO

## Equação paramétrica de reta

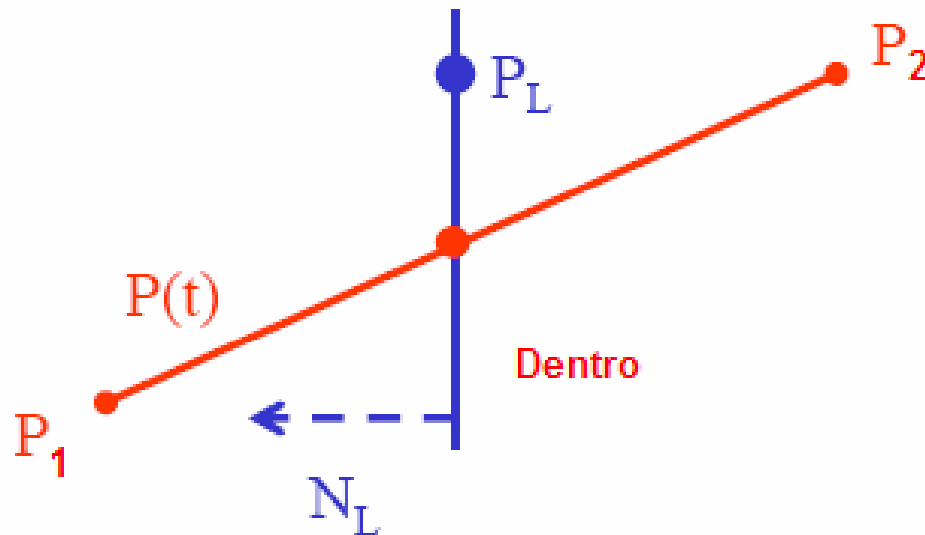
- Definir cada linha do objeto na forma paramétrica
- Definir cada aresta da window na forma paramétrica:  
 $P_{\text{Left}}(t)$ ,  $P_{\text{Right}}(t)$ ,  $P_{\text{Top}}(t)$ ,  $P_{\text{Bottom}}(t)$
- Realiza testes de interseção de Cohen-Sutherland usando linhas e arestas paramétricas
- Equações paramétricas permitem recortes mais eficientes



## RECORTE

### ALGORITMO DE CYRUS-BECK ou Liang-Barsky

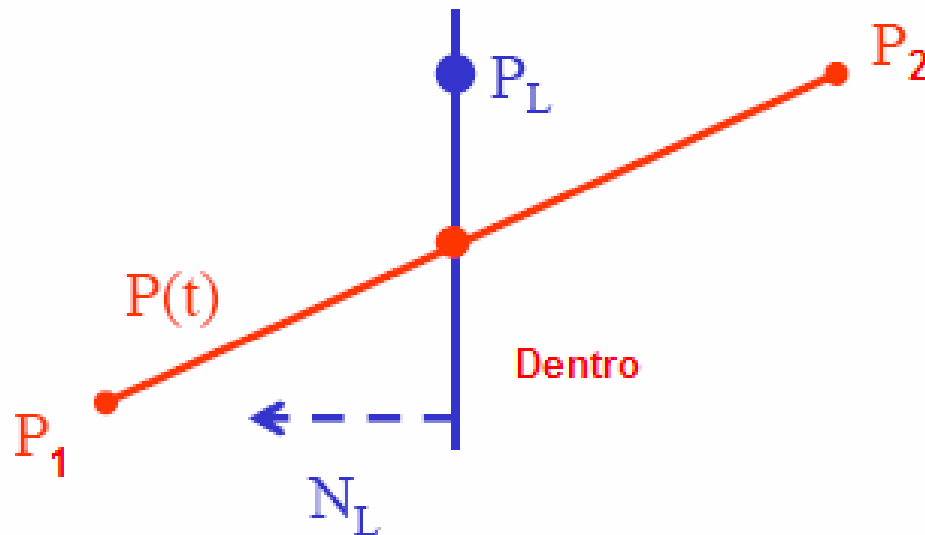
Dado segmento  $P_1P_2$ , seja  $L$  a fronteira esquerda da janela e seja  $P_L$  um ponto sobre  $L$ . Além disso, seja  $N_L$  o vetor normal a  $P_L$  apontando para fora da janela.



## RECORTE

### ALGORITMO DE CYRUS-BECK ou Liang-Barsky

Podemos determinar se um ponto  $P(t)$  está dentro ou fora do subespaço delimitado por  $L$  com base no ângulo entre os vetores  $[P(t)-P_L]$  e  $N_L$



## RECORTE

### ALGORITMO DE CYRUS-BECK ou Liang-Barsky

Seja  $\theta$  o ângulo entre  $[P(t)-P_L]$  e  $N_L$ , então:

—  $\theta < 90^\circ$ ,  $P(t)$  está fora

—  $\theta > 90^\circ$ ,  $P(t)$  está dentro

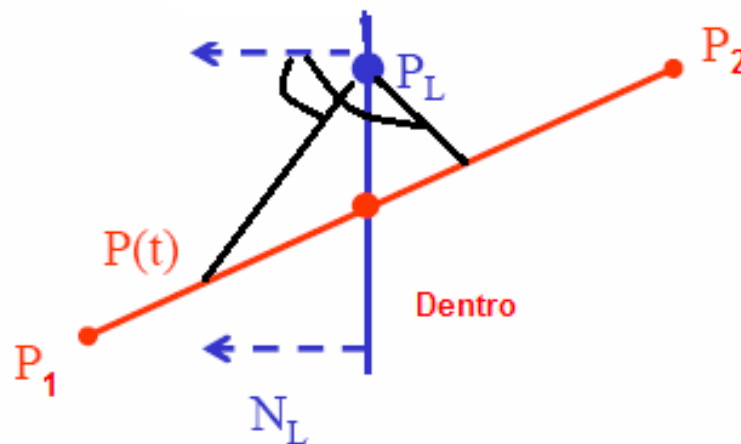
—  $\theta = 90^\circ$ ,  $P(t)$  está sobre  $L$

• Daí, se

—  $\cos \theta > 0$ ,  $P(t)$  está fora

—  $\cos \theta < 0$ ,  $P(t)$  está dentro

—  $\cos \theta = 0$ ,  $P(t)$  está sobre  $L$

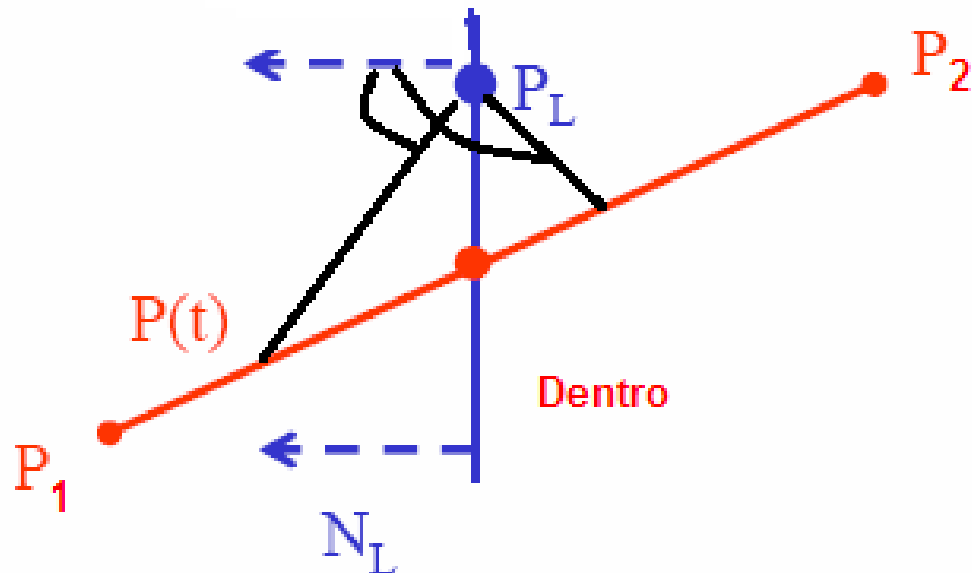


## RECORTE

### ALGORITMO DE CYRUS-BECK ou Liang-Barsky

O sinal de  $\cos \theta$  pode ser obtido pelo sinal do produto interno entre  $[P(t)-P_L]$  e  $N_L$

$$\overrightarrow{[P(t)-P_L]} \cdot \overrightarrow{N_L} = |\overrightarrow{[P(t)-P_L]}| |\overrightarrow{N_L}| \cos(\theta)$$





# RECORTE

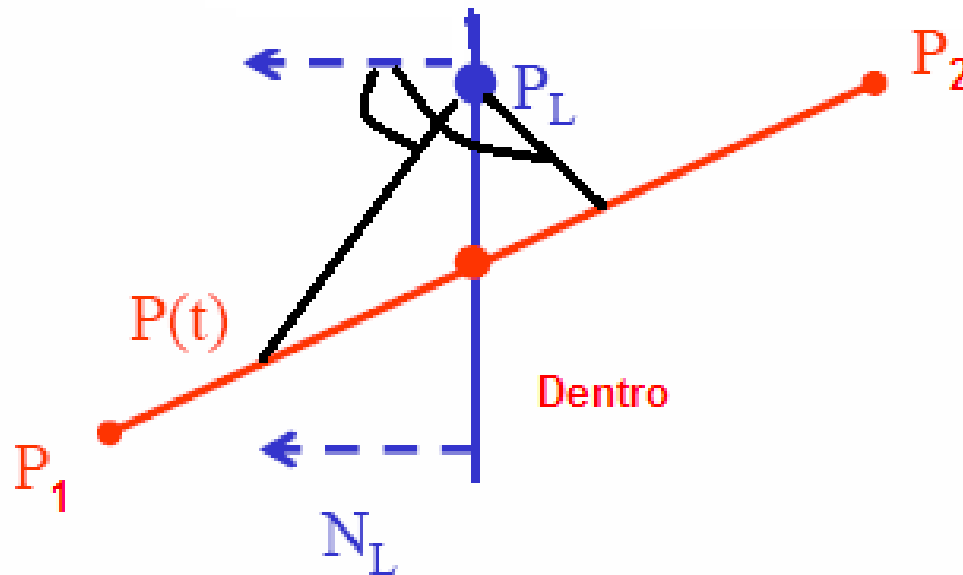
## ALGORITMO DE CYRUS-BECK ou Liang-Barsky

Além disso, o ponto de interseção entre o segmento P1-P2 e a fronteira L pode ser obtido substituindo

$P(t) = P1 + (P2 - P1) t$  na equação do produto interno:

$$\vec{NL} \cdot [P1 + (P2 - P1) t - PL] = 0$$

$$t = \frac{N_L \bullet [P_1 - P_L]}{-N_L \bullet [P_2 - P_1]}$$



# RECORTE

## ALGORITMO DE CYRUS-BECK ou Liang-Barsky

A questão agora é identificar os valores de  $t$  tais que os pontos estejam na janela de recorte.

$$t = \frac{N_L \bullet [P_1 - P_L]}{-N_L \bullet [P_2 - P_1]}$$

A idéia é identificar as situações onde o segmento está potencialmente entrando ou potencialmente saindo da janela de recorte

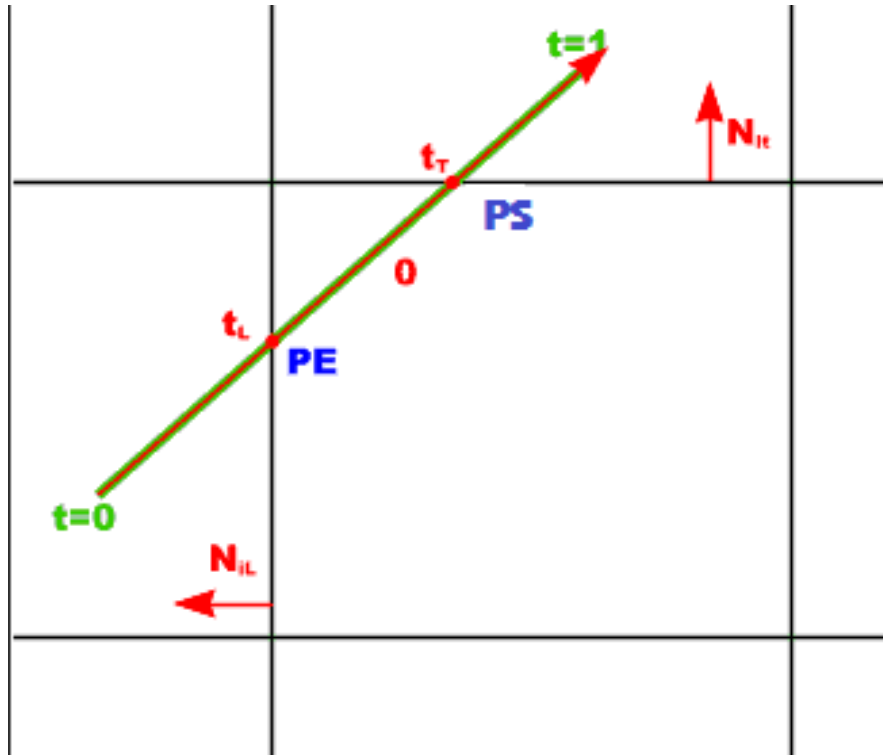
- Isto é feito usando o ângulo  $\theta$  entre os vetores  $[P_2 - P_1]$  e  $N_L$ .
- Se  $\cos \theta > 0$ , o segmento está potencialmente saindo
- Se  $\cos \theta < 0$ , o segmento está potencialmente entrando



# RECORTE

## ALGORITMO DE CYRUS-BECK ou Liang-Barsky

O trecho do segmento contido dentro da janela será aquele compreendido entre o maior valor de  $t$  potencialmente entrando e o menor valor de  $t$  potencialmente saindo.



Potencialmente entrando (PE)

$$N_i \cdot P_1 P_2 < 0 \ (\theta > 90) \Rightarrow t_{\text{Left}}$$

Potencialmente saindo (PS)

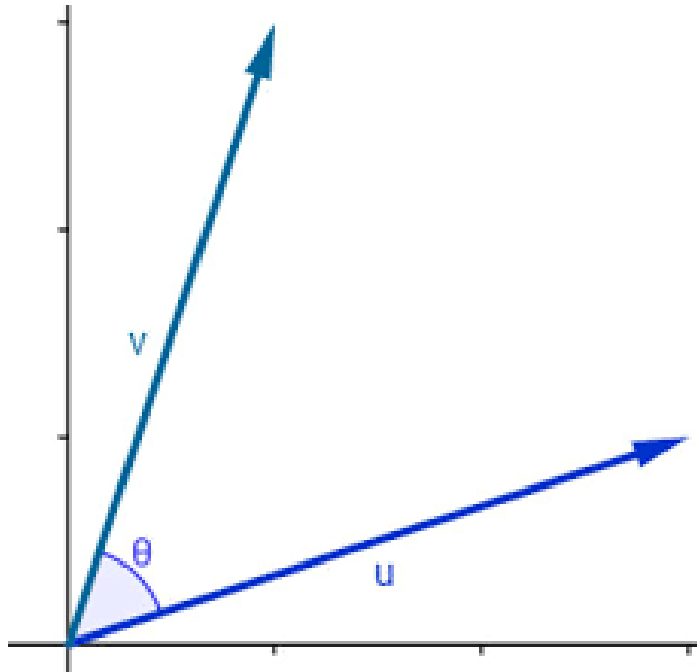
$$N_i \cdot P_1 P_2 > 0 \ (\theta < 90) \Rightarrow t_{\text{Top}}$$

# RECORTE

## ALGORITMO DE CYRUS-BECK ou Liang-Barsky

Considerando os dois vetores  $u = (a,b)$  e  $v = (c,d)$ , o produto interno entre  $u$  e  $v$  é dado por:

$$\langle u, v \rangle = \langle (a,b), (c,d) \rangle = a \cdot c + b \cdot d$$



# RECORTE

## ALGORITMO DE CYRUS-BECK ou Liang-Barsky

Computar  $t$  para a interseção do segmento com todas as fronteiras da janela

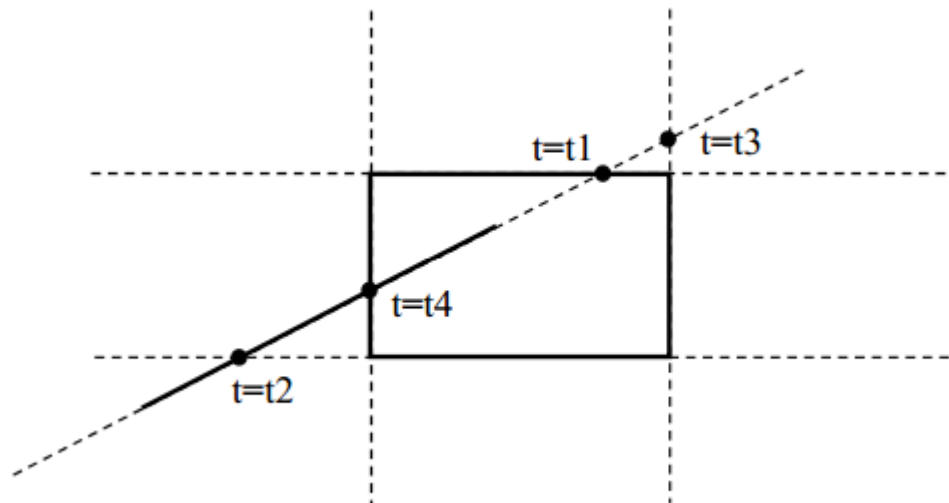
- Descarte todos ( $t < 0$ ) e ( $t > 1$ )
- Classifique as interseções restantes em potencialmente entrando (PE) ou potencialmente saindo (PS) da seguinte forma:
  - $-NL \cdot [P2 - P1] > 0$  implica PS
  - $-NL \cdot [P2 - P1] < 0$  implica PE
- Ache o ponto PE com maior  $t$  e o PS com menor  $t$
- Recorte nesses 2 pontos



# RECORTE

## ALGORITMO DE CYRUS-BACK

- Dado um segmento  $P_0P_1$ , sua reta suporte intersecta as quatro retas laterais do retângulo em quatro pontos.
- Usando a equação paramétrica calculamos os **quatro valores do parâmetro  $t$**  no qual a reta intersecta as quatro laterais da window.
- Após calcular o valor de  $t$  para as quatro retas descartamos os valores de  $t$  fora do intervalo  **$[0,1]$** .



# RECORTE

## ALGORITMO DE CYRUS-BACK

Calcule Normal ( $N_i$ ) e escolha um  $P_i$  para cada borda

```
tEntra = 0;
```

```
tSai = 1;
```

```
for cada borda (esquerda, direita, cima, baixo) {
```

```
    if ( $N_i \cdot (P_2 - P_1) \neq 0$ ) { // borda não é paralela a reta  
        calcular t;
```

```
        sinal de  $N_i \cdot (P_2 - P_1)$  classifica como PE ou PS;
```

```
        if( PE ) tEntra = max(tEntra, t);
```

```
        if( PS ) tSai = min(tSai, t);
```

```
    } else { /* borda paralela à reta */
```

```
        if ( $N_i \cdot (P_2 - P_i) > 0$ ) /* está fora */
```

```
            return nulo;
```

```
    }
```

```
if (tEntra > tSai)
```

```
    return nulo;
```

```
else
```

```
    return P(tEntra) and P(tSai)
```

```
}    }    }
```

# RECORTE

## ALGORITMO DE CYRUS-BACK -4/4

Fronteira	$N_L$
Esquerda ( $x = x_{min}$ )	$(-1, 0)$
Direita ( $x = x_{max}$ )	$(1, 0)$
Inferior ( $y = y_{min}$ )	$(0, -1)$
Superior ( $y = y_{max}$ )	$(0, 1)$





# Comparação

- Cohen-Sutherland

Recorte repetitivo é caro

Melhor utilizado quando a maioria das linhas se encaixam nos casos de aceitação e rejeição triviais

- Cyrus- Beck ou Liang-Barsky

Cálculo de  $t$  para as interseções é barato

Cálculo dos pontos  $(x,y)$  de corte é feita apenas uma vez

Algoritmo não considera os casos triviais

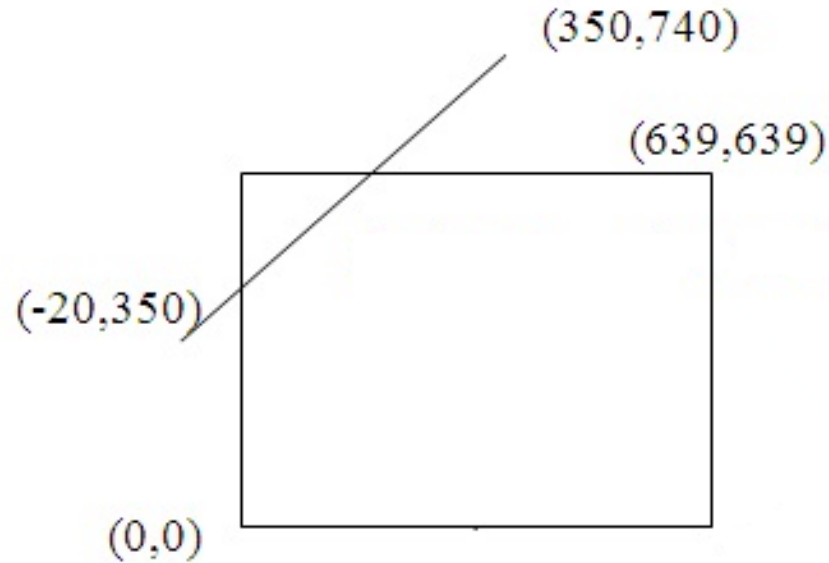
Melhor quando a maioria das linhas precisa ser recortada

Permite que a janela não seja retangular e/ou alinhada com os eixos



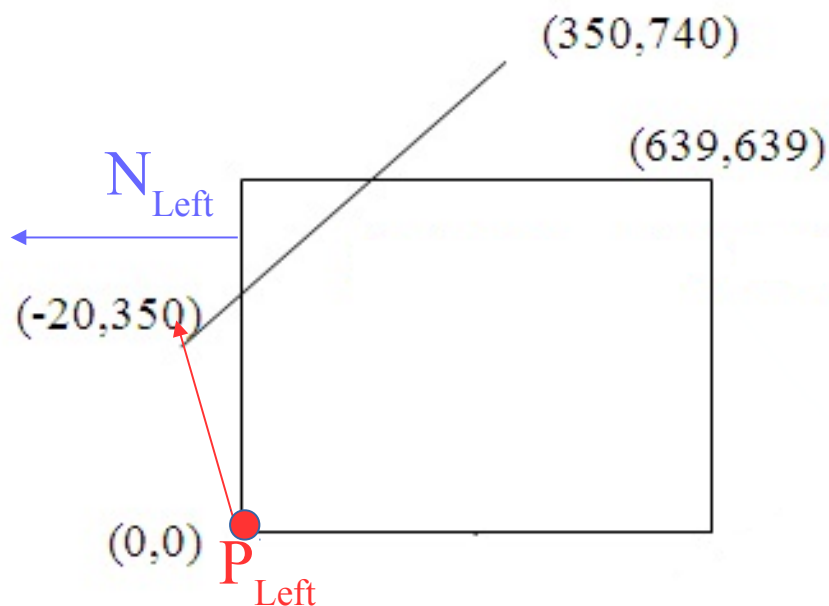
# EXEMPLO

- Aplique o algoritmo Cyrus Beck para recortar a linha abaixo



## EXEMPLO

- Borda esquerda



$$N_{left} = (-1, 0)$$

$$P_{Left} = (0,0)$$

$$D = P2 - P1 = (350+20, 740 - 350) \\ = (370,390)$$

Calculando denominador de t:

$$N_{left} \cdot D = (-1, 0) \cdot (370, 390) \\ = -1 \times 370 + 0 \times 390 \\ = -370, \text{ negativo, PE}$$

Calculando numerador de t:

$$N_{left} \cdot [P1 - P_{left}] = ?$$

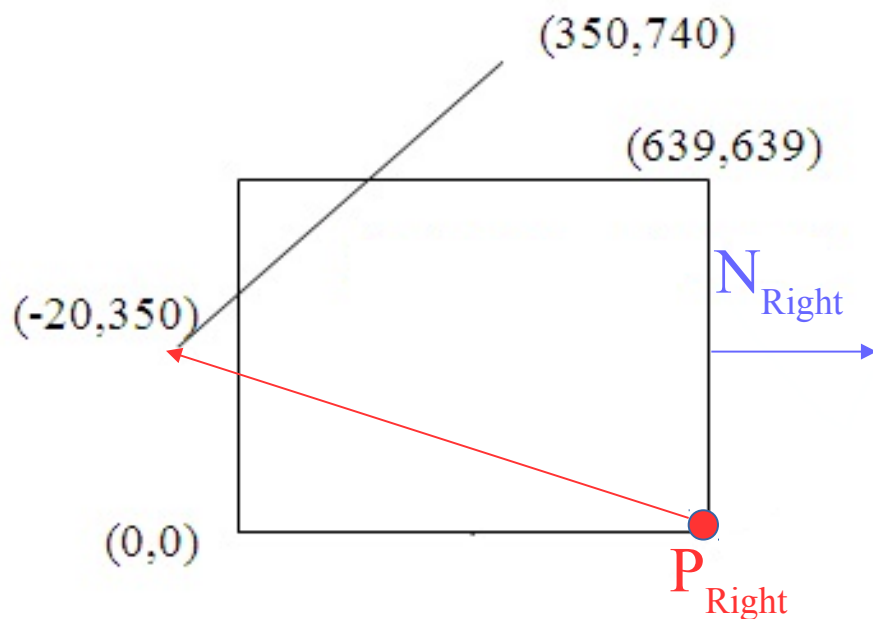
$$[P1 - P_{left}] = (-20-0, 350-0) \\ = (-20, 350)$$

$$N_{left} \cdot [P1 - P_{left}] = (-1, 0) \cdot (-20, 350) \\ = -1 \times (-20) + 0 \times 350 = 20$$

$$t = +20 / -(-370) = 20/370 = 0,054$$

## EXEMPLO

- Borda direita



$$N_{\text{Right}} = (+1, 0)$$

$$P_{\text{Right}} = (639, 0)$$

$$D = P_2 - P_1 = (350+20, 740 - 350) \\ = (370, 390)$$

Calculando denominador de t:

$$N_{\text{Right}} \cdot D = (+1, 0) \cdot (370, 390) \\ = +1 \times 370 + 0 \times 390 \\ = +370, \text{ positivo, PS}$$

Calculando numerador de t:

$$N_{\text{Right}} \cdot [P_1 - P_{\text{Right}}] = ?$$

$$[P_1 - P_{\text{Right}}] = (-20-639, 350-0) \\ = (-659, 350)$$

$$N_{\text{Right}} \cdot [P_1 - P_{\text{Right}}] = (+1, 0) \cdot (-659, 350) \\ = 1 \times (-659) + 0 \times 350 = -659 \\ t = -659 / -(370) = 1,78, \text{ descartar}$$

# RECORTE

## RECORTE DE POLÍGONOS

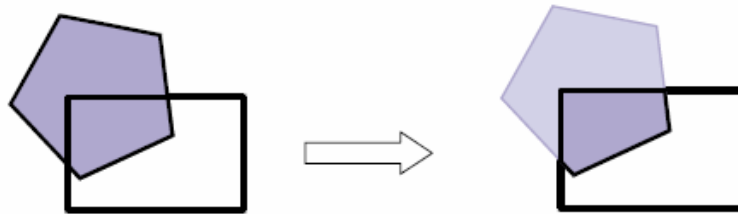
- Dois algoritmos clássicos
- Sutherland-Hodgman
  - Figura de recorte pode ser qualquer polígono convexo
- Weiler-Atherton
  - Figura de recorte pode ser qualquer polígono



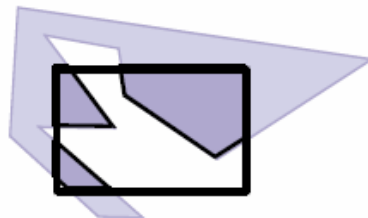
# RECORTE

## RECORTE DE POLÍGONOS

- Inclui o problema de recorte de segmentos de reta, mas é bem mais complicado



- Ainda mais quando o polígono é côncavo



## RECORTE

### RECORTE DE POLÍGONOS

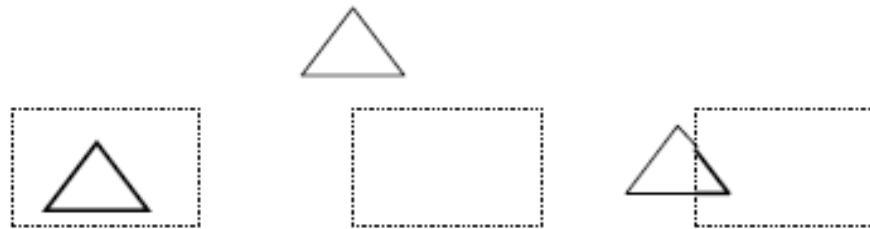
- Polígono resultante tem vértices que são:
  - Vértices da janela,
  - Vértices do polígono original, ou
  - Pontos de interseção aresta do polígono contra aresta da janela



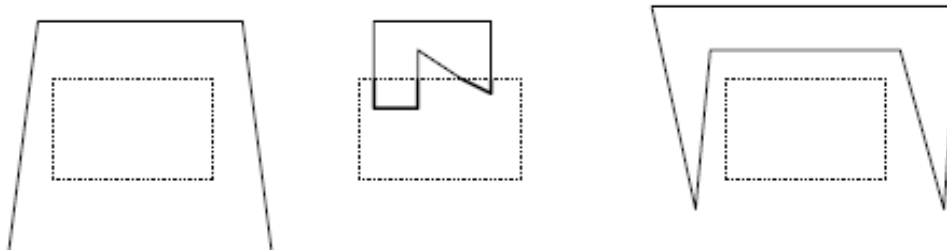
# RECORTE

## RECORTE DE POLÍGONOS

- Casos Simples



- Casos Complexos

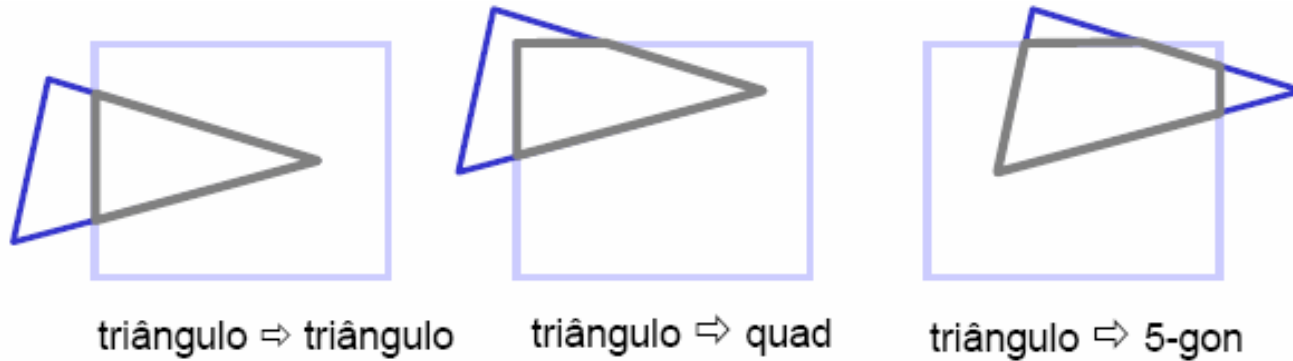




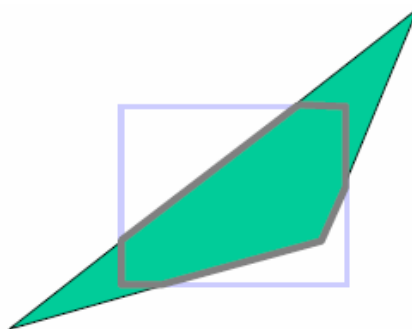
# RECORTE

## RECORTE DE POLÍGONOS

- O que pode acontecer com um triângulo?



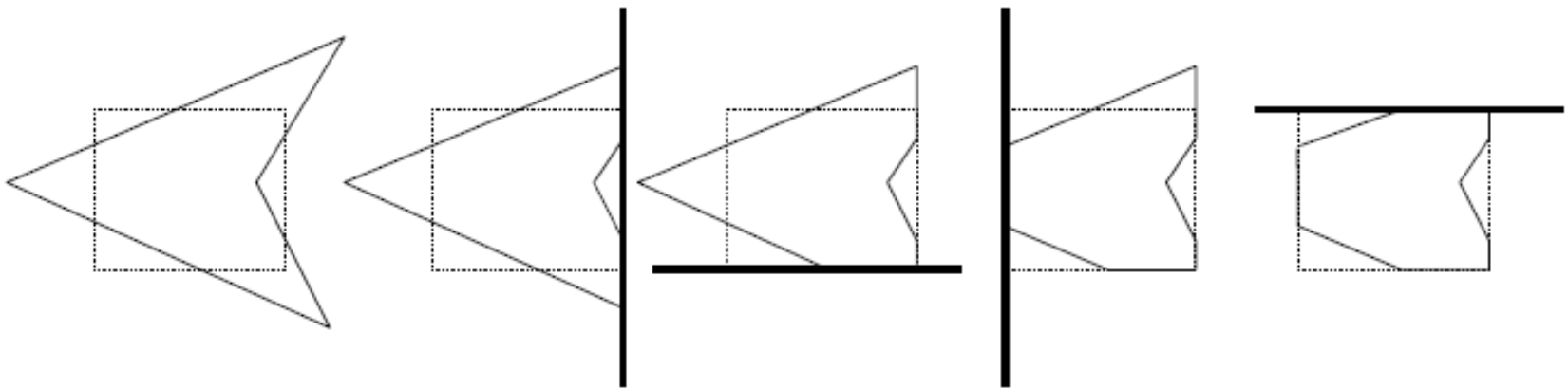
- O recorte de um triângulo pode gerar um polígono com quantos lados?



# RECORTE DE POLÍGONOS

## Algoritmo de Sutherland-Hodgman

- Recortar o polígono sucessivamente contra todos os semi-espacos planos da figura de recorte
- Vértices e arestas são processados em sequência e classificados contra o semi-espaço corrente



# RECORTE DE POLÍGONOS

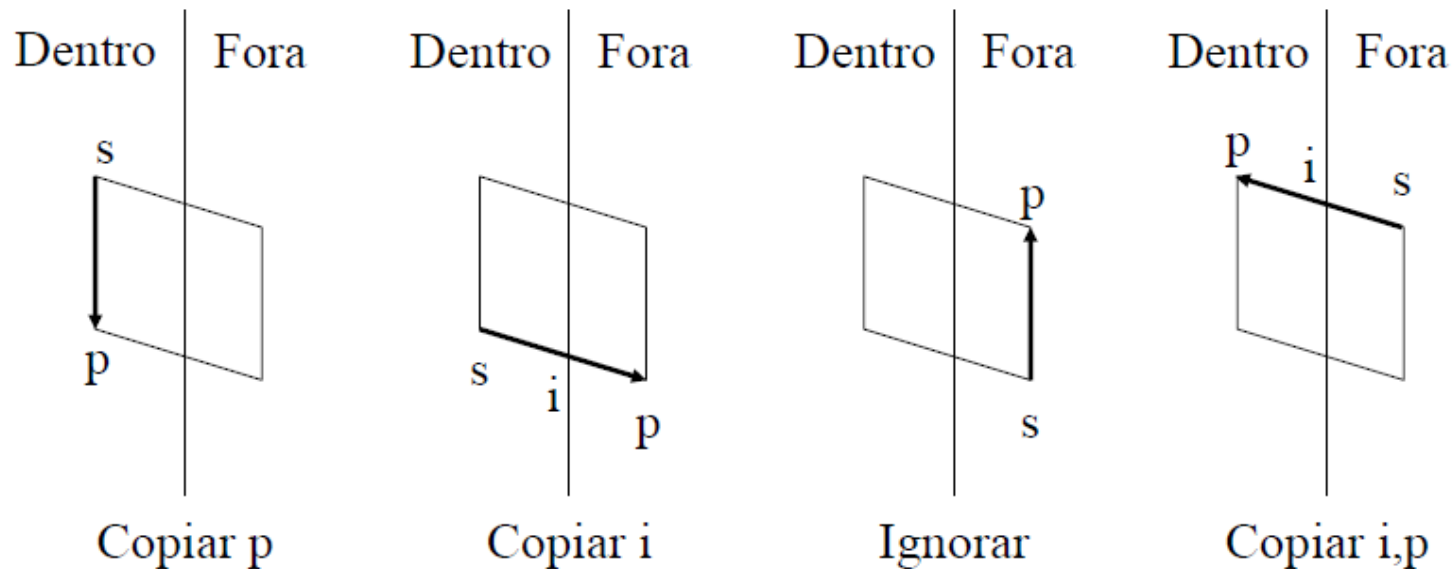
## Algoritmo de Sutherland-Hodgman

- Vértice:
  - Dentro: copiar para a saída
  - Fora: ignorar
- Aresta
  - Intercepta semi-espaco plano (vértices têm classificações diferentes) : Copiar ponto de interseção para a saída
  - Não intercepta: ignorar



# RECORTE DE POLÍGONOS

## Algoritmo de Sutherland-Hodgman



s: Vértice antecessor

p: Vértice atual



# RECORTE DE POLÍGONOS

## Algoritmo de Sutherland-Hodgman

4 casos:

—s e p dentro do plano

Coloque p na lista de saída (guarde p)

Nota: s já estaria na lista pela aresta anterior

—s dentro do plano e p fora

Ache ponto de interseção i e guarde i na lista de saída

—s e p fora do plano

Não coloque nada na lista de saída

—s fora do plano e p dentro

Ache ponto de interseção i e guarde i e p na lista de saída

s: Vértice antecessor

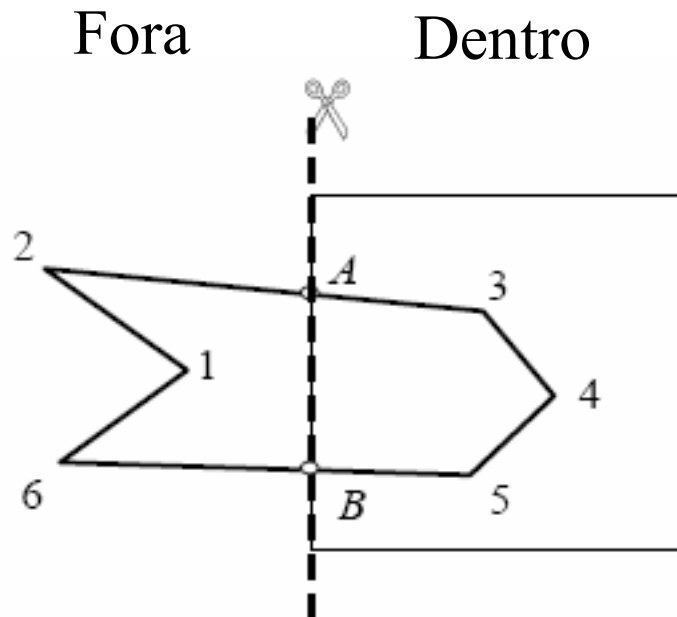
p: Vértice atual



# RECORTE DE POLÍGONOS

## Algoritmo de Sutherland-Hodgman

Exemplo 1:

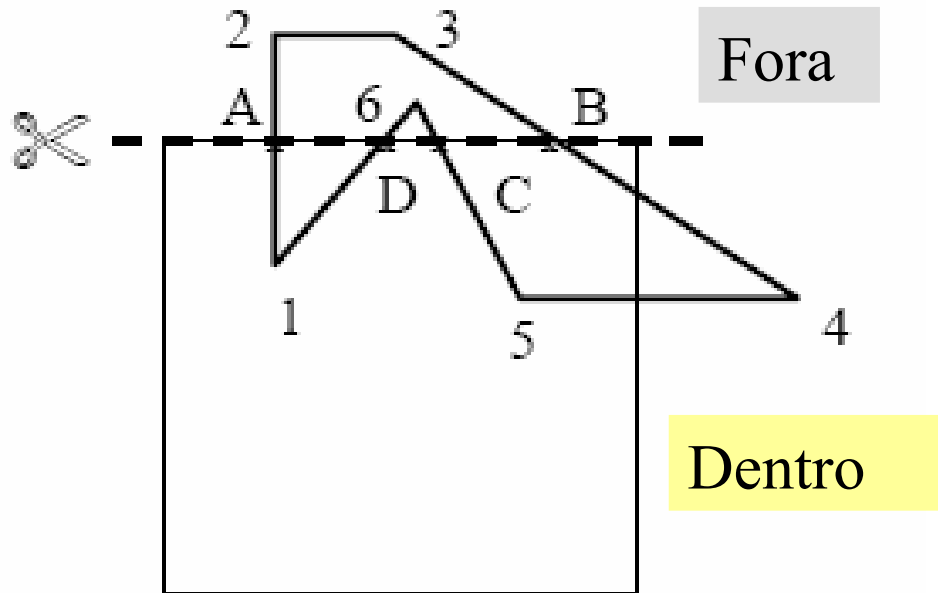


S	P	Ação
1	2	x
2	3	store A,3
3	4	store 4
4	5	store 5
5	6	store B
6	1	x

# RECORTE DE POLÍGONOS

## Algoritmo de Sutherland-Hodgman

Exercício:



# RECORTE DE POLÍGONOS

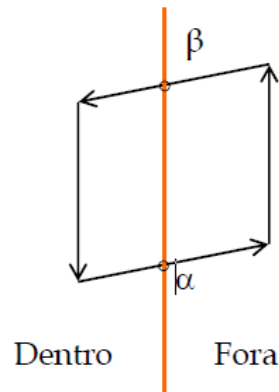
## Algoritmo de Sutherland-Hodgman

Problema: Arestas Fantasma

Solução:

Distinguir os pontos de interseção gerados

- De dentro para fora: rotular como do tipo  $\alpha$
- De fora para dentro: rotular como do tipo  $\beta$
- Iniciar o percurso de algum vértice “fora”
- Ao encontrar um ponto de interseção  $\alpha$ , ligar com o último  $\beta$  visitado

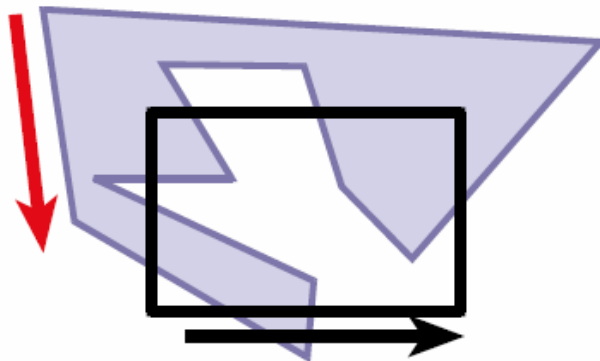




# RECORTE DE POLÍGONOS

## Algoritmo de Weiler-Atherton

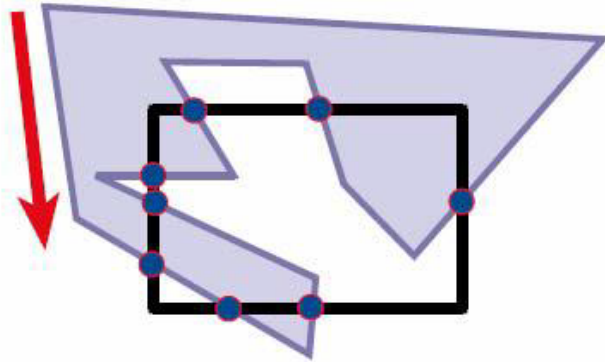
- Recorta qualquer polígono contra qualquer outro polígono
- Mais complexo que o algoritmo de Sutherland-Hodgman
- “Caminhar” pelas bordas do polígono e da janela
- Polígonos são orientados no sentido anti-horário



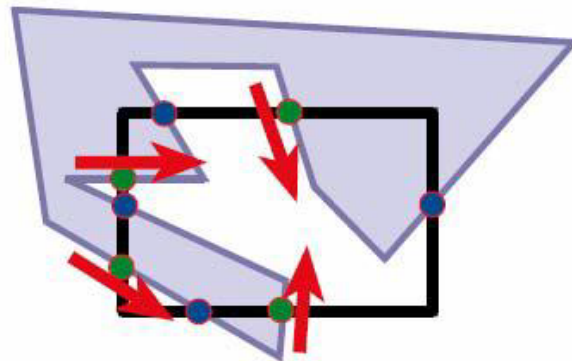
# RECORTE DE POLÍGONOS

## Algoritmo de Weiler-Atherton

- Encontre os pontos de intersecção



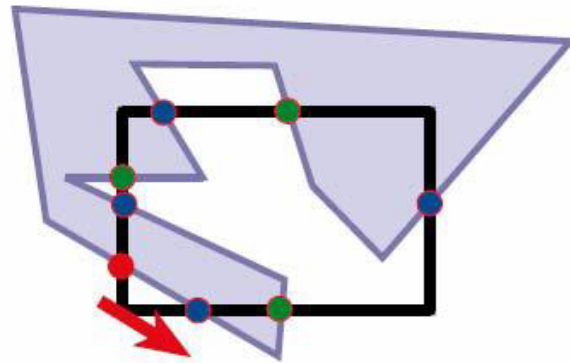
- Marque os pontos onde o polígono entra na window



# RECORTE DE POLÍGONOS

## Algoritmo de Weiler-Atherton

- Enquanto houver interseção de entrada não processada:
  - “caminhar” pelas bordas do polígono ou da janela



# RECORTE DE POLÍGONOS

## Algoritmo de Weiler-Atherton

- Regras da “caminhada”

–Par In-to-out:

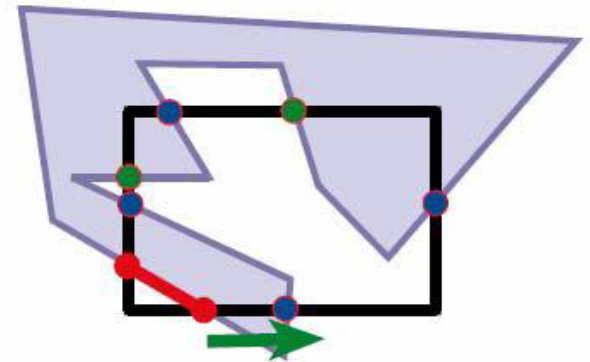
Grave o ponto de Interseção

Caminhe pela aresta do polígono

–Par Out-to-in:

Grave o ponto de interseção

Caminhe pela aresta da janela

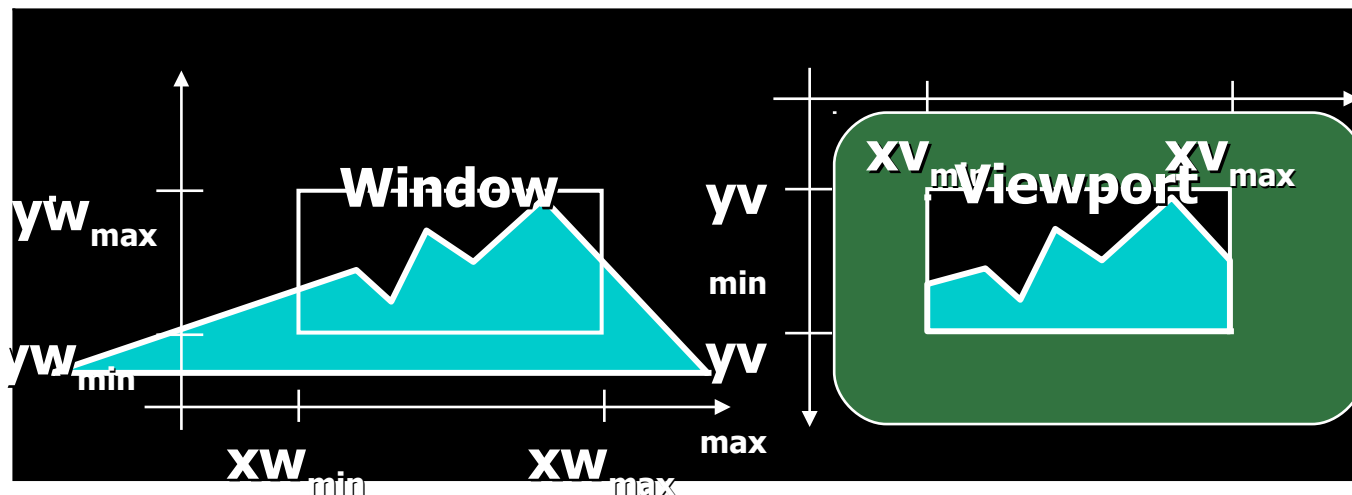


# MAPEAMENTO WINDOW-VIEWPORT

Window : O QUE nós queremos ver

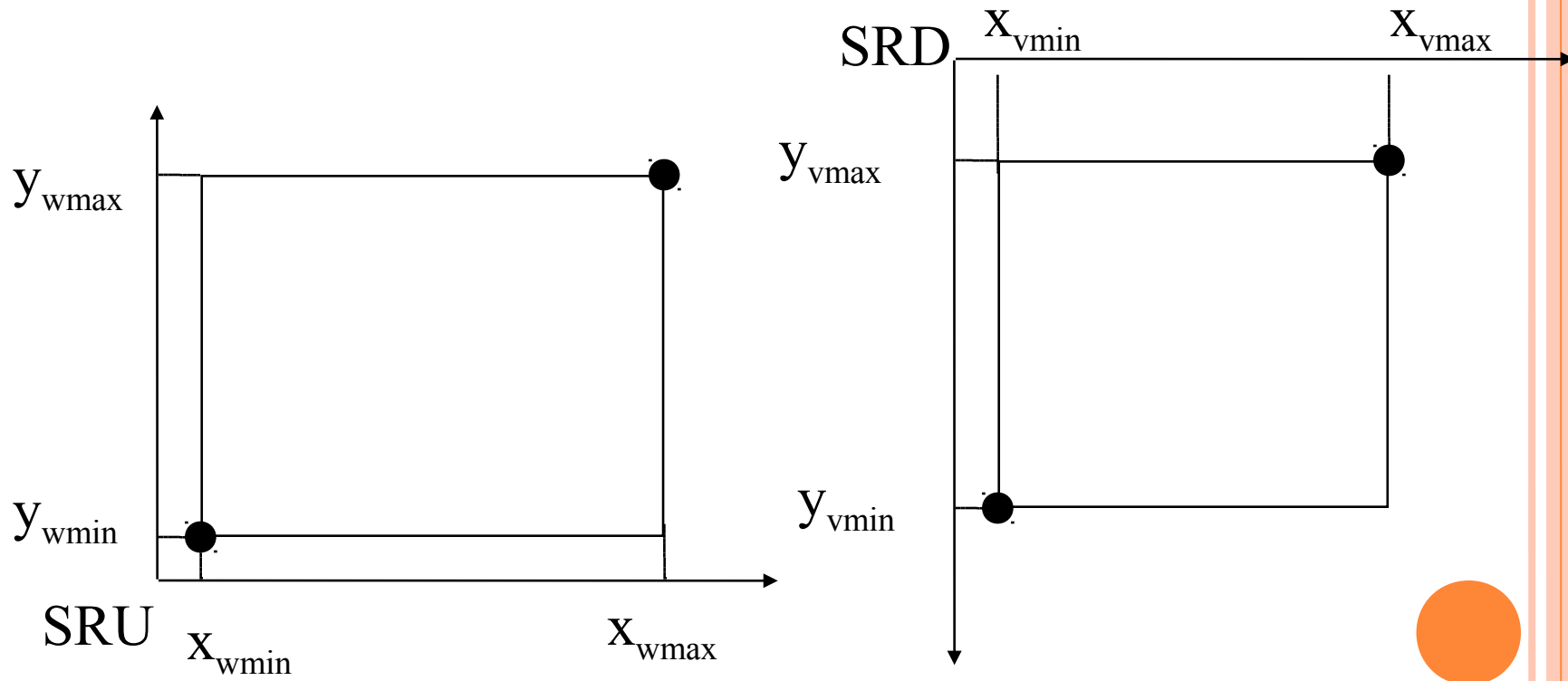
Viewport: ONDE será visualizado no dispositivo de saída

Normalmente Windows e Viewport são retângulos com as arestas dos retângulos paralelas aos eixos de coordenadas, porque requerem menos processamento do que recorte contra janelas com fronteiras não lineares.



# MAPEAMENTO WINDOW-VIEWPORT

Mudança de coordenadas dos pontos do SRU para o SRD



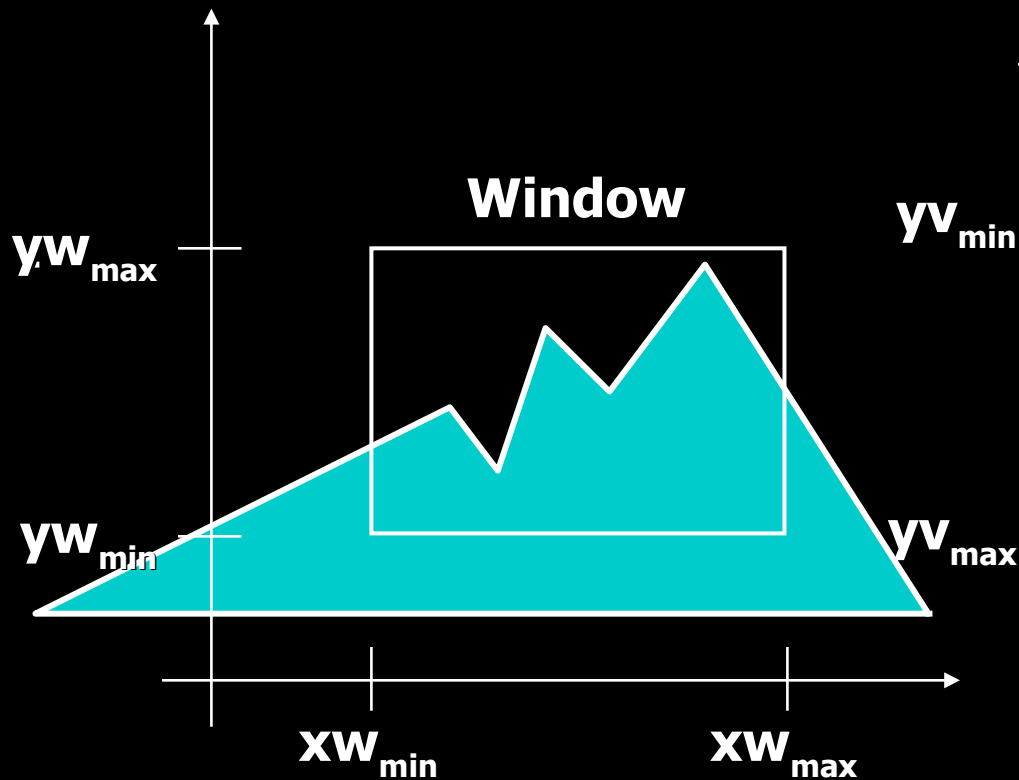
# MAPEAMENTO WINDOW-VIEWPORT

- **Window**
  - Uma área de coordenadas do mundo selecionada para ser mostrada
  - Coordenadas normalizadas (unitário)
- **Viewport**
  - Uma área em um dispositivo de display para a qual o conteúdo da window é mapeado

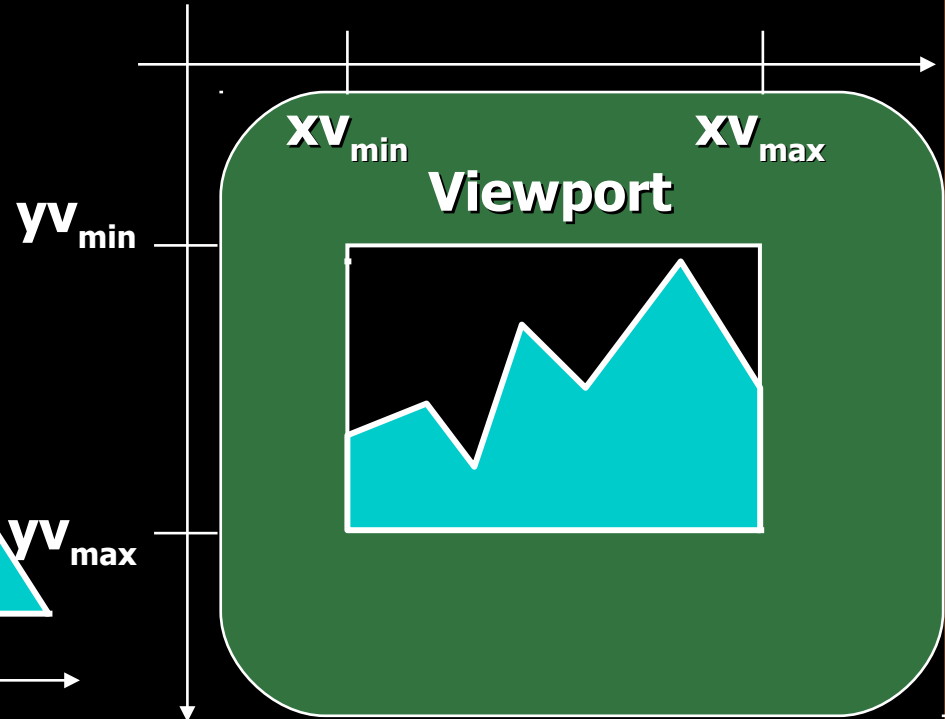


# MAPEAMENTO WINDOW-VIEWPORT

**Coordenadas do Mundo**



**Coordenadas do Dispositivo**





# MAPEAMENTO WINDOW-VIEWPORT

- Efeito de Zoom
  - Mapeamento sucessivo de windows de tamanho diferente em viewports de tamanho fixo
- Efeito de Panning
  - Movimentação de uma window de tamanho fixo através de vários objetos em uma cena.



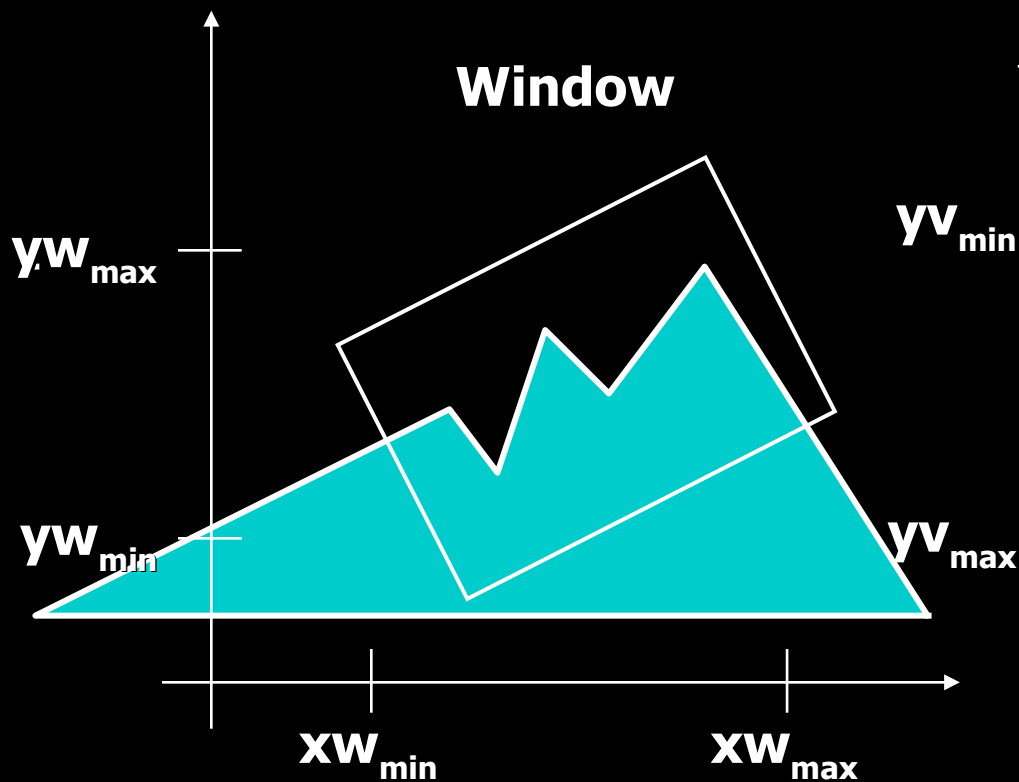
# WINDOW ARBITRÁRIA

- Window pode ser um retângulo com qualquer orientação
  - Sistema de coordenadas normalizado ou sistema de coordenadas de plano de projeção
  - Sistema intermediário entre o sistema de coordenadas do mundo e o sistema de coordenadas do vídeo.

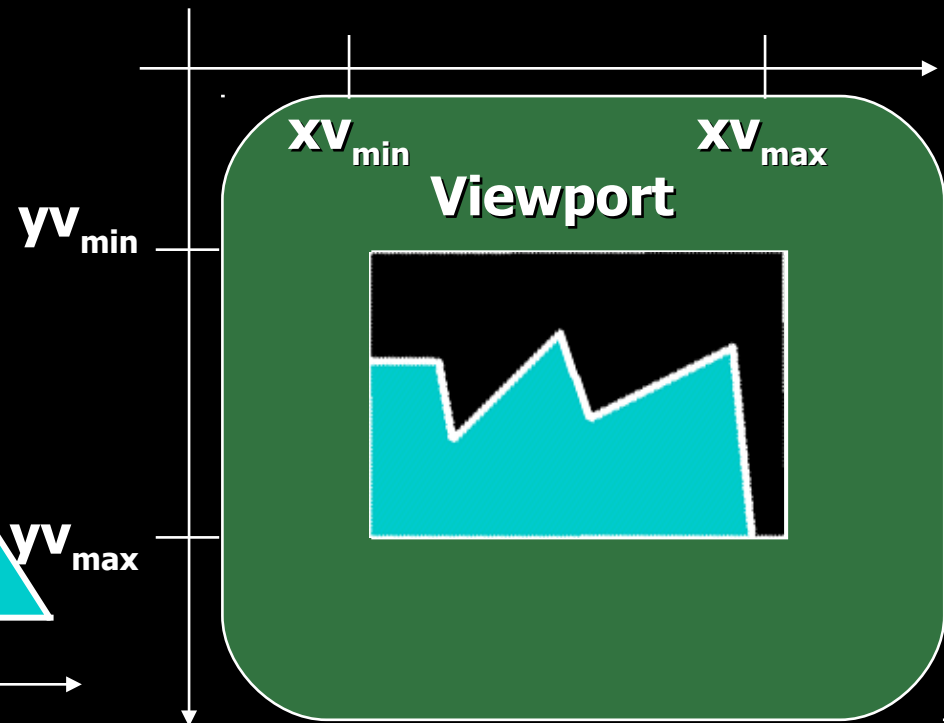


# WINDOW ARBITRÁRIA

**Coordenadas do Mundo**

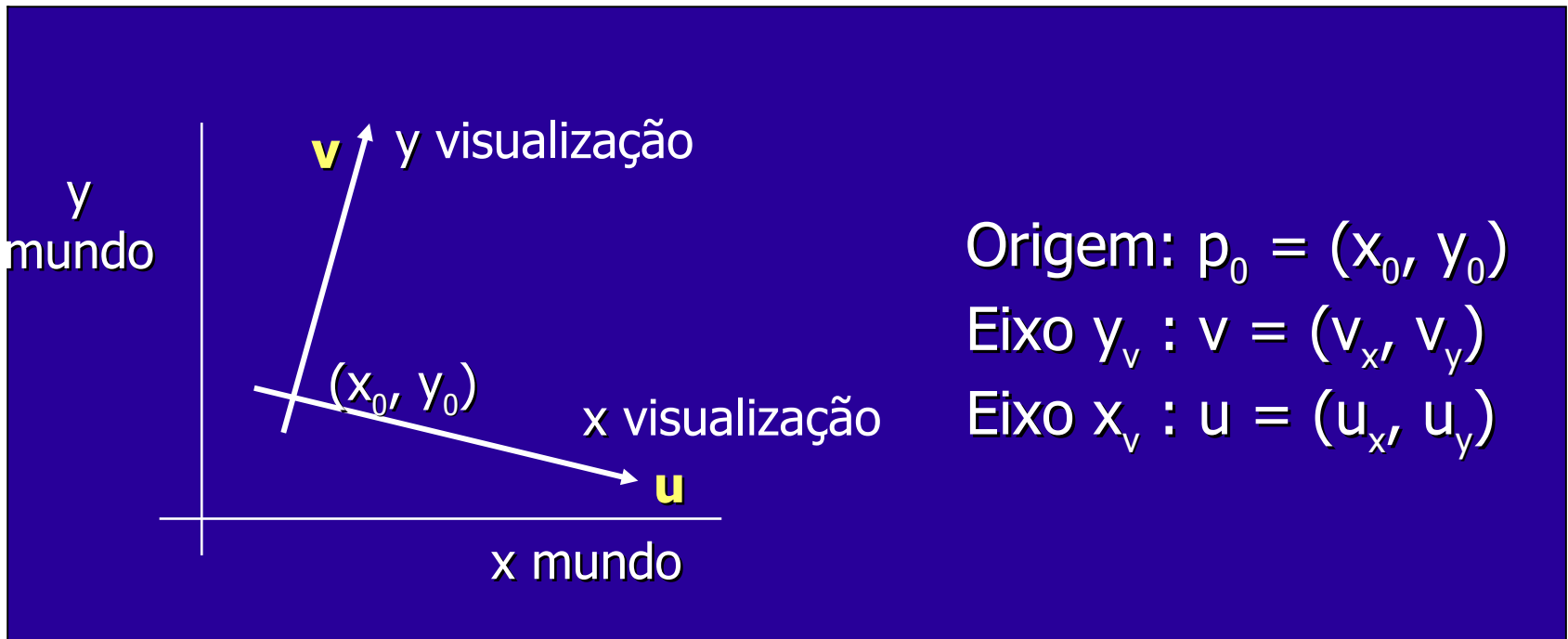


**Coordenadas do Dispositivo**

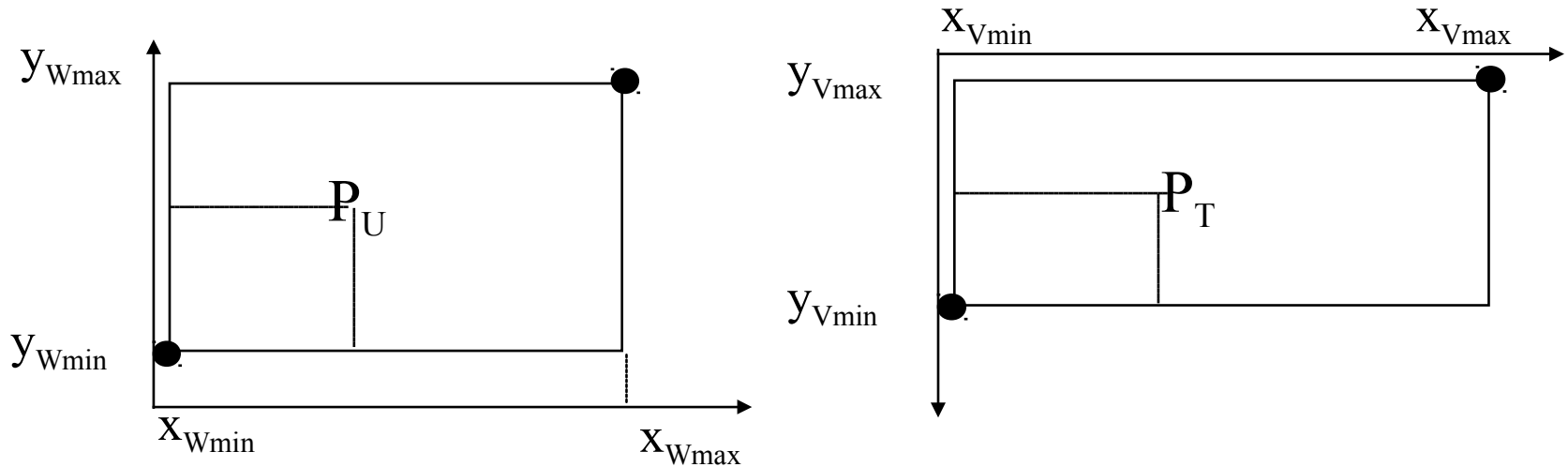


# WINDOW ARBITRÁRIA

- Especificação da window em relação ao SRU:
- Origem das coordenadas de visualização:  $P_0 = (x_0, y_0)$
- Vetor V (View up): Define a direção y de visualização



# MAPEAMENTO WINDOW-VIEWPORT



$$X_T = X_{Vmin} + (X_U - X_{Wmin}) \frac{(X_{Vmax} - X_{Vmin})}{(X_{Wmax} - X_{Wmin})}$$

$$Y_T = Y_{Vmin} + (Y_U - Y_{Wmin}) \frac{(Y_{Vmax} - Y_{Vmin})}{(Y_{Wmax} - Y_{Wmin})}$$



# EXERCÍCIO

- Dado o objeto:
  - $V1 = (5,5)$ ,  $V2=(15,5)$ ,  $V3=(10,15)$
  - Apresente o mapeamento do objeto acima para uma tela (SRD) com resolução 640 X 480, dada window de  $(0,0) - (20, 20)$

