The Cyclops Project
German-Brazilian Cooperation Programme on IT
CNPq GMD DLR

Computação Gráfica:

Aula 4: Clipping (Recorte)

Métodos, Técnicas e Algoritmos para Cálculo de Visualização em 2D

Prof. Dr. rer.nat. Aldo von Wangenheim

Departamento de Informática e Estatística - INE/CTC/UFSC

Capítulo 4: Objetivos

Aprenderemos:

- Windows genéricos em 2D
- View Up Vector
- Sistema de Coordenadas Normalizado ou de Window
- Métodos de Clipping

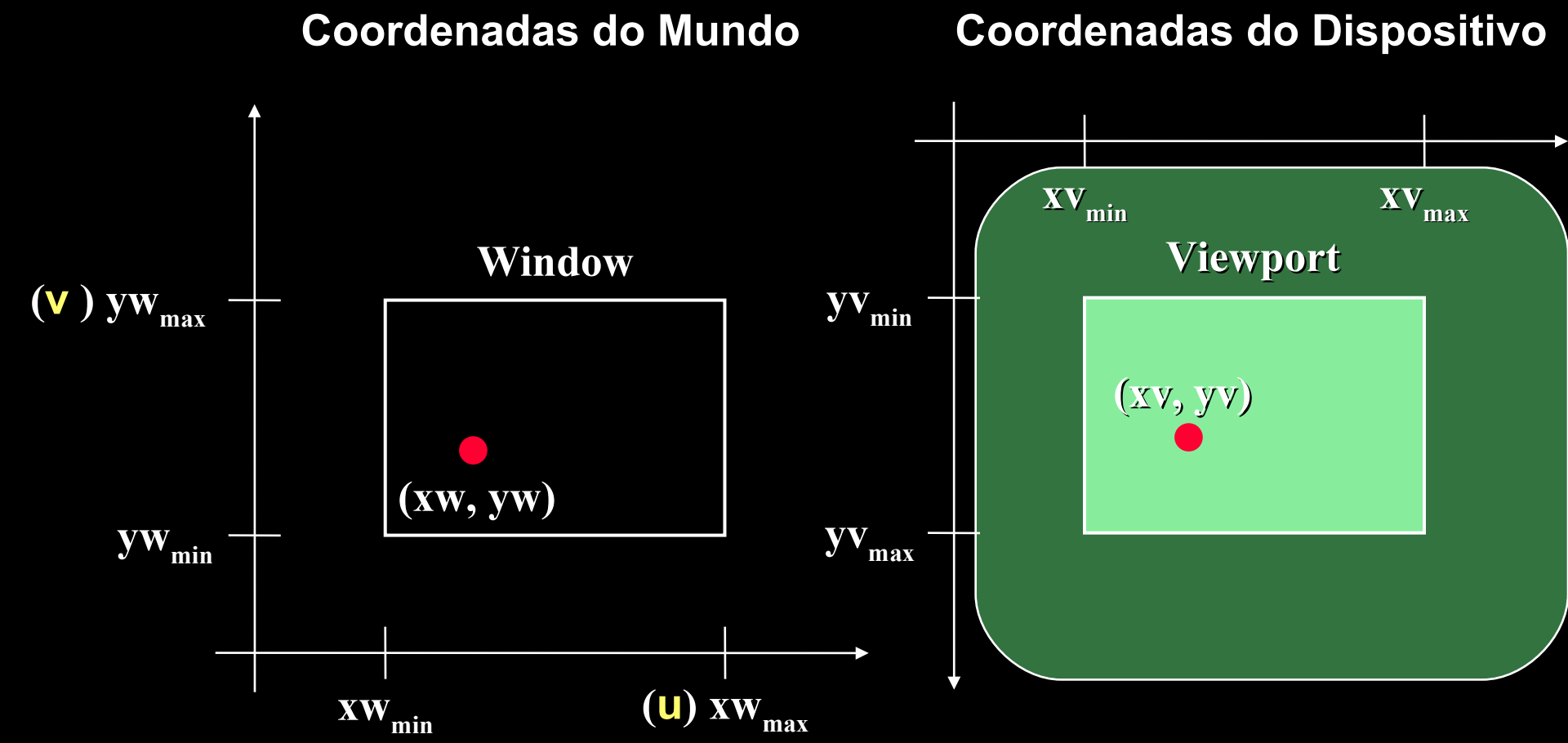
Computação Gráfica:

4.1. Sistema de Coordenadas Normalizado

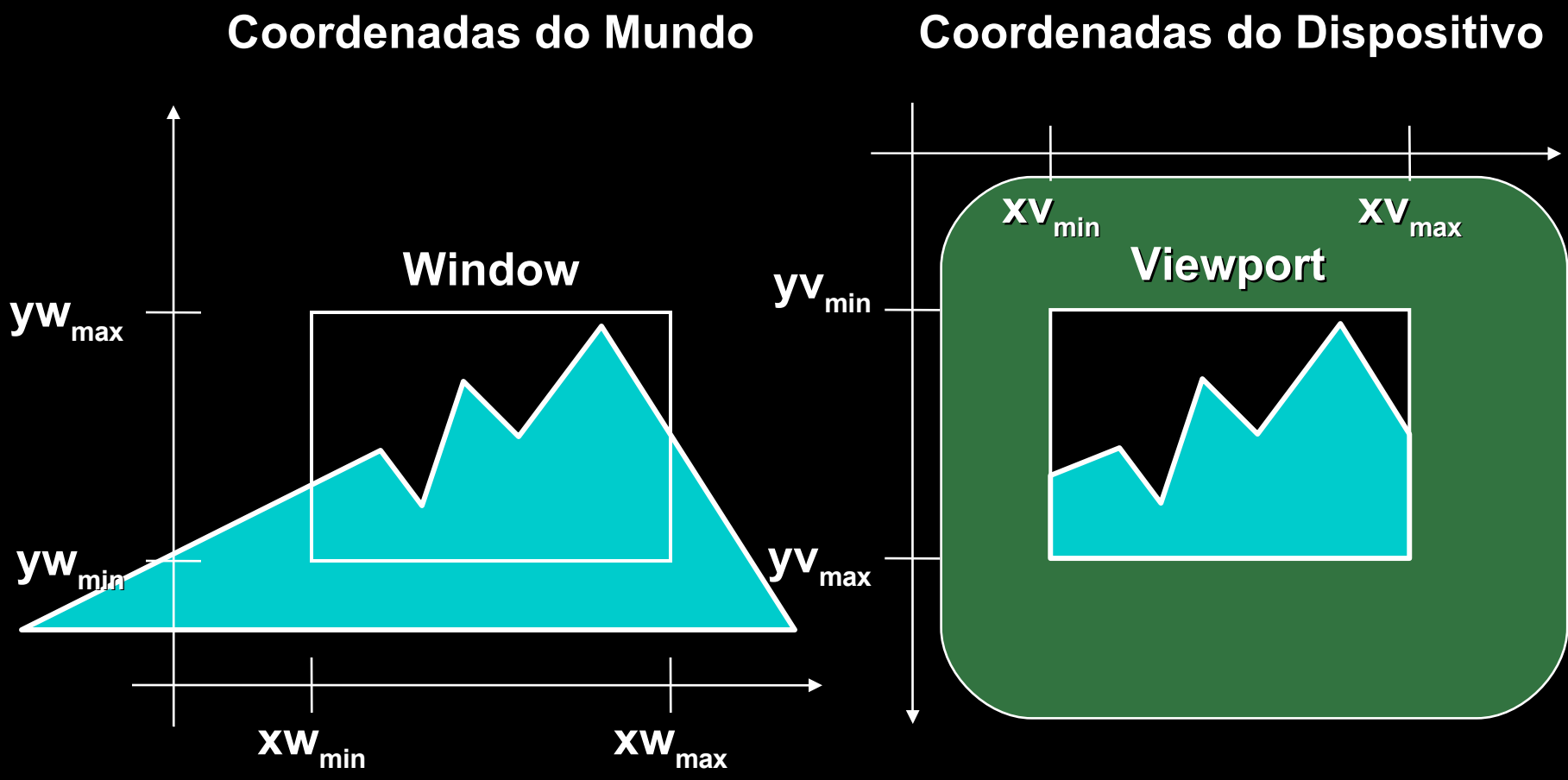
As entidades de Visualização (recapitulando):

- **Window** (estr.dados - janela)
  - Uma área de world-coordinates (coordenadas do mundo) selecionada para ser mostrada.
- **Viewport** (estr.dados - área de desenho da tela)
  - Uma área em um dispositivo de display para a qual o conteúdo de uma window é mapeado.
- **Transformação de visualização** (transformação - viewing transform)
  - O mapeamento de uma parte de uma cena em coordendas do mundo para coordenadas de dispositivo.

Transformada Window-to-Viewport

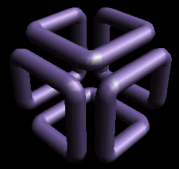


Visualização de Mundos Complexos em duas dimensões



Desvantagens do mapeamento direto window->viewport

- Navegação limitada.
- Eixos de coordenadas de window e de viewport são sempre paralelos
  - Transformada de viewport é apenas um transformação de escala
  - Operações como rotação da window são impossíveis.

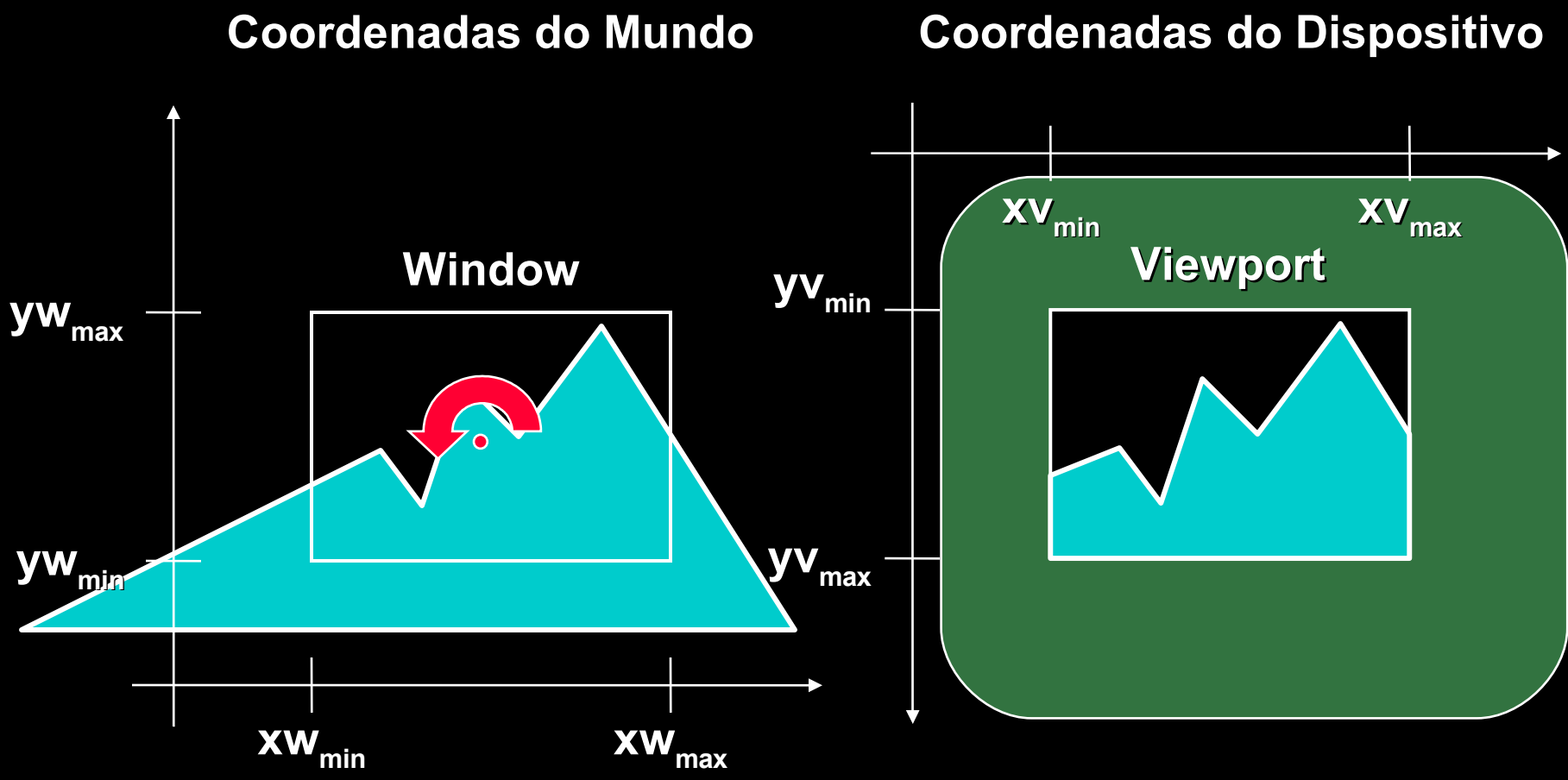


- Efeito de
  - Movime
    através
- Importan
  - Windov
    quadrac

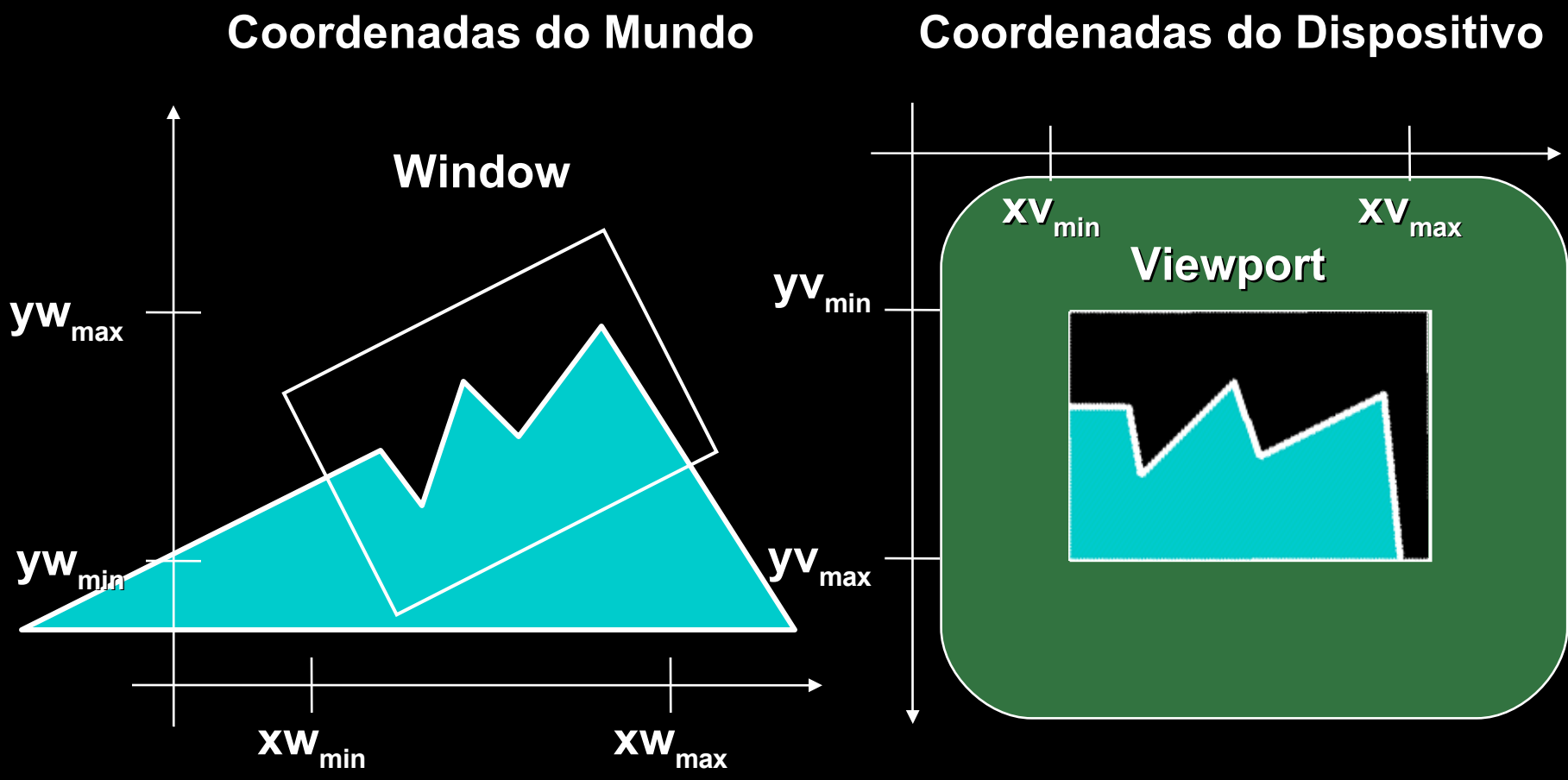
As entidades de Visualização (extendendo o conceito de Window):

- Window: Para permitir todos os graus de liberdade na navegação no mundo, uma window deveria ser um retângulo com qualquer orientação
  - Até agora vimos apenas windows paralelas ao sistema de coordenadas do mundo.
  - Para extendermos uma window de forma a podermos realizar o efeito de panning, mesmo em 2D, temos de definir um terceiro sistema de coordenadas intermediário, entre o sistema de coordenadas do mundo e o sistema de coordenadas do vídeo.
  - Este sistema de coordenadas é chamado de **sistema de coordendas normalizado** ou sistema de coordenadas de plano de projeção.

Visualização de Mundos Complexos em duas dimensões

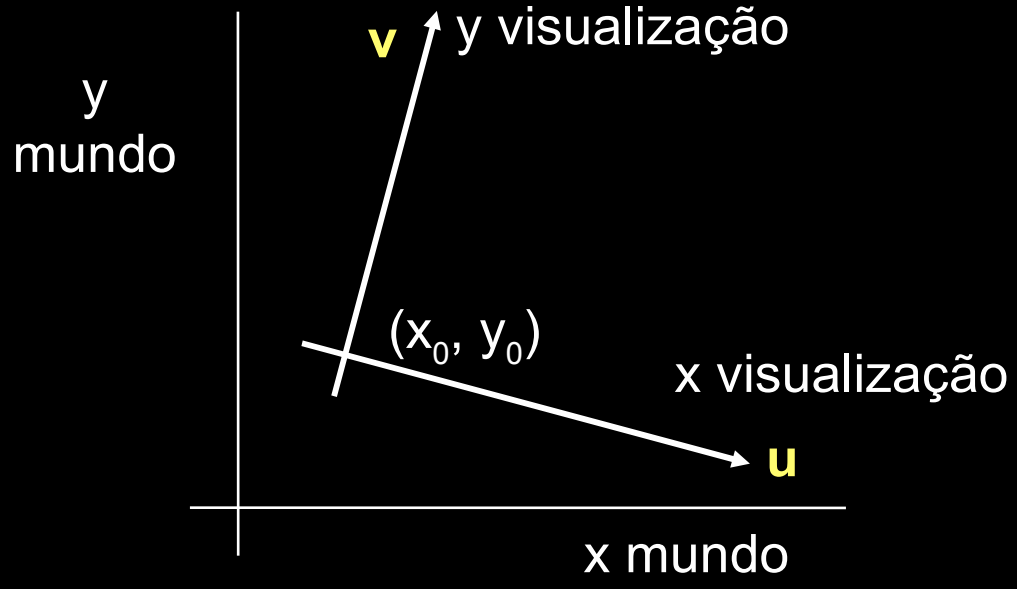


Visualização de Mundos Complexos em duas dimensões



Visualizando o Sistema de Coordenadas Normalizado

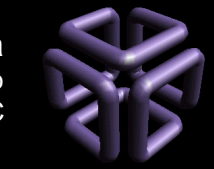
- Esquema de referência para a especificação do window em relação às coordenadas do mundo:
  - Origem das coordenadas de visualização: $P_0 = (x_0, y_0)$
  - *View up vector* ***V***: Define a direção y_v de visualização

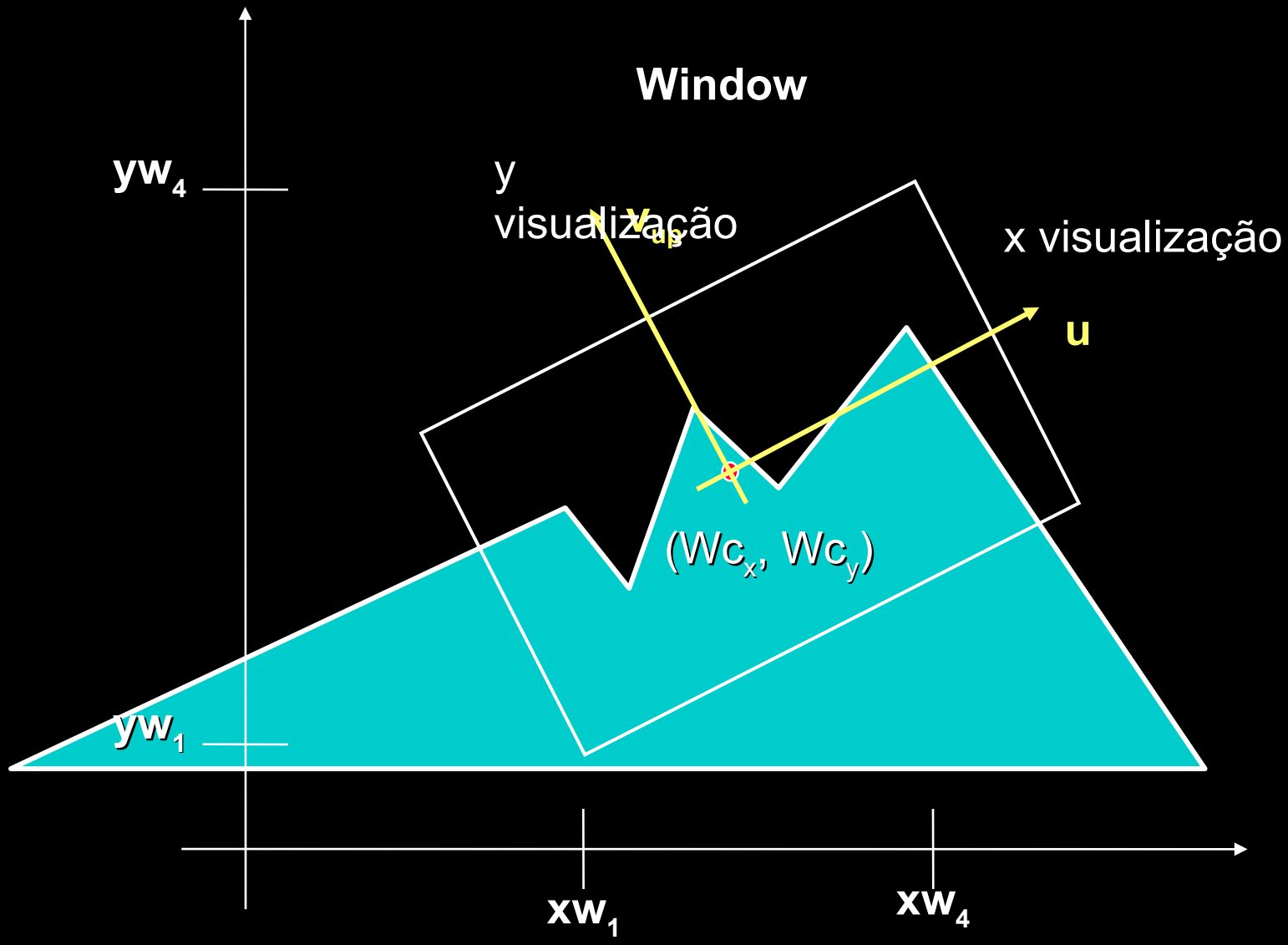


Origem: $p_0 = (x_0, y_0)$

Eixo y_v : $v = (v_x, v_y)$

Eixo x_v : $u = (u_x, u_y)$




Window
yw_4
y visualização
y_{up}
x visualização
u
(Wc_x, Wc_y)
yw_1
xw_1
xw_4

Implicações do Sistema de Coordenadas Normalizado

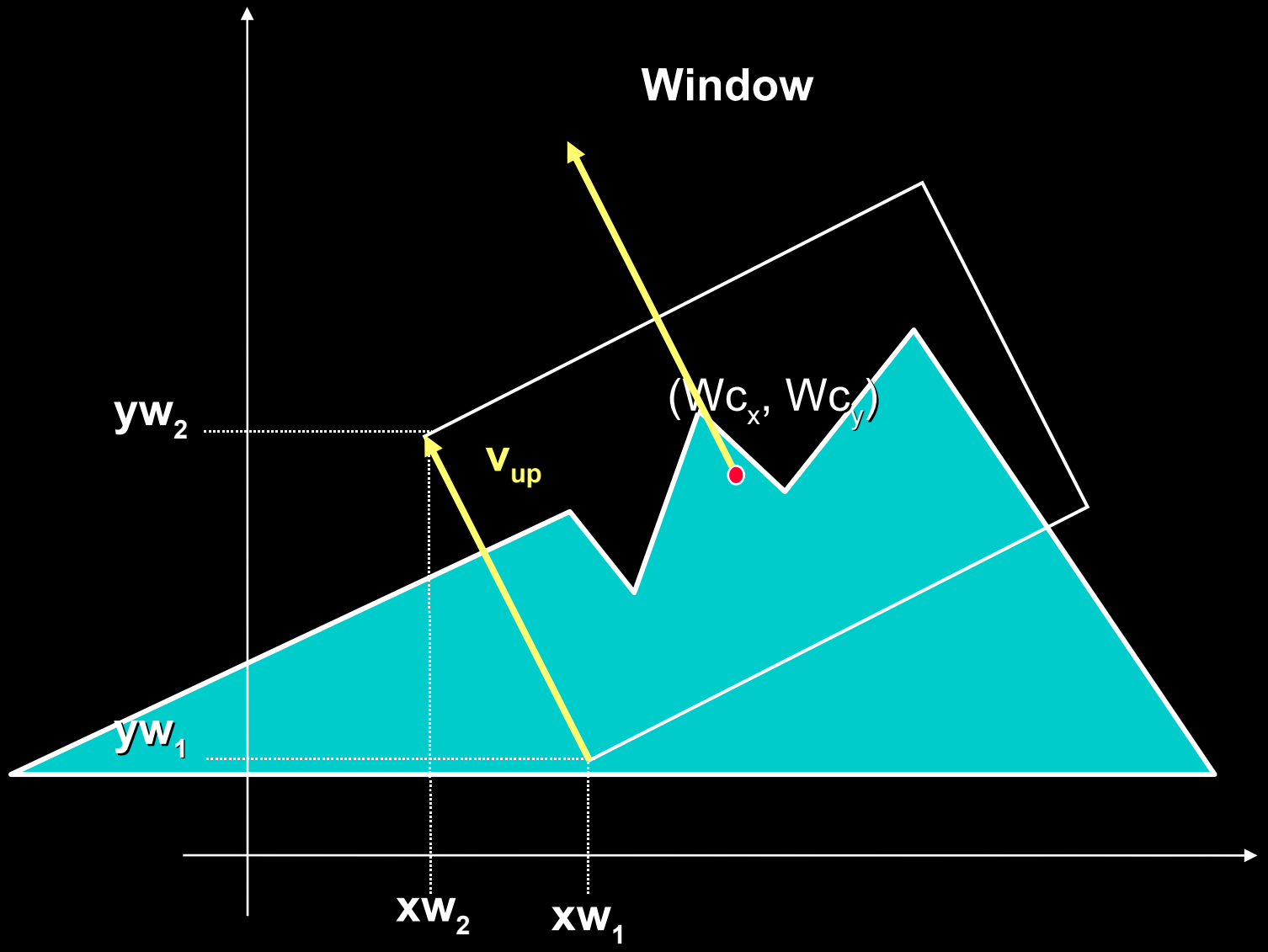
- Cada objeto do mundo é representado em dois sistemas de coordenadas:
- Coordenadas do Mundo: suas coordenadas reais
  - Abreviamos por WC - World Coordinates
- Coordenadas Normalizadas: coordenadas do objeto expressas em termos de **u** e **v** e com origem em (Wc_x, Wc_y)
  - Tradicionalmente normalizadas dentro da window

Por que o nome Sistema de Coordenadas Normalizado ?

- Para tornar um sistema independente do tamanho da viewport e facilitar alguns algoritmos, usava-se normalizar as coordenadas dos objetos
  - Fixava-se os extremos da window em (-1,-1),(1,1)
  - Ao se transformar de um sistema de coordenadas para o outro, normalizava-se os objetos
    - Vantagem: tudo que possuir uma coordenada fora do intervalo [-1,1] está fora da window.
    - Desvantagem: qualquer operação de navegação ou zoom implica em uma nova transformação de normalização de coordenadas.

Alternativas ao Sistema de Coordenadas Normalizado

- Podemos continuar a representar a window em termos da unidade de medida das coordenadas do mundo: sistema de coordenadas de plano de projeção - PPC.
  - Mantemos a unidade de medida do mundo.
    - Muito menos divisões
  - Representamos $\mathbf{v_{up}}$ por $(\mathbf{xw_1}, \mathbf{yw_1})(\mathbf{xw_2}, \mathbf{yw_2})$ ao invés de $(\mathbf{Wc_x}, \mathbf{Wc_y})(\mathbf{v_{upx}}, \mathbf{v_{upy}})$


Window
yw_2
(Wc_x, Wc_y)
v_{up}
yw_1
xw_2
xw_1

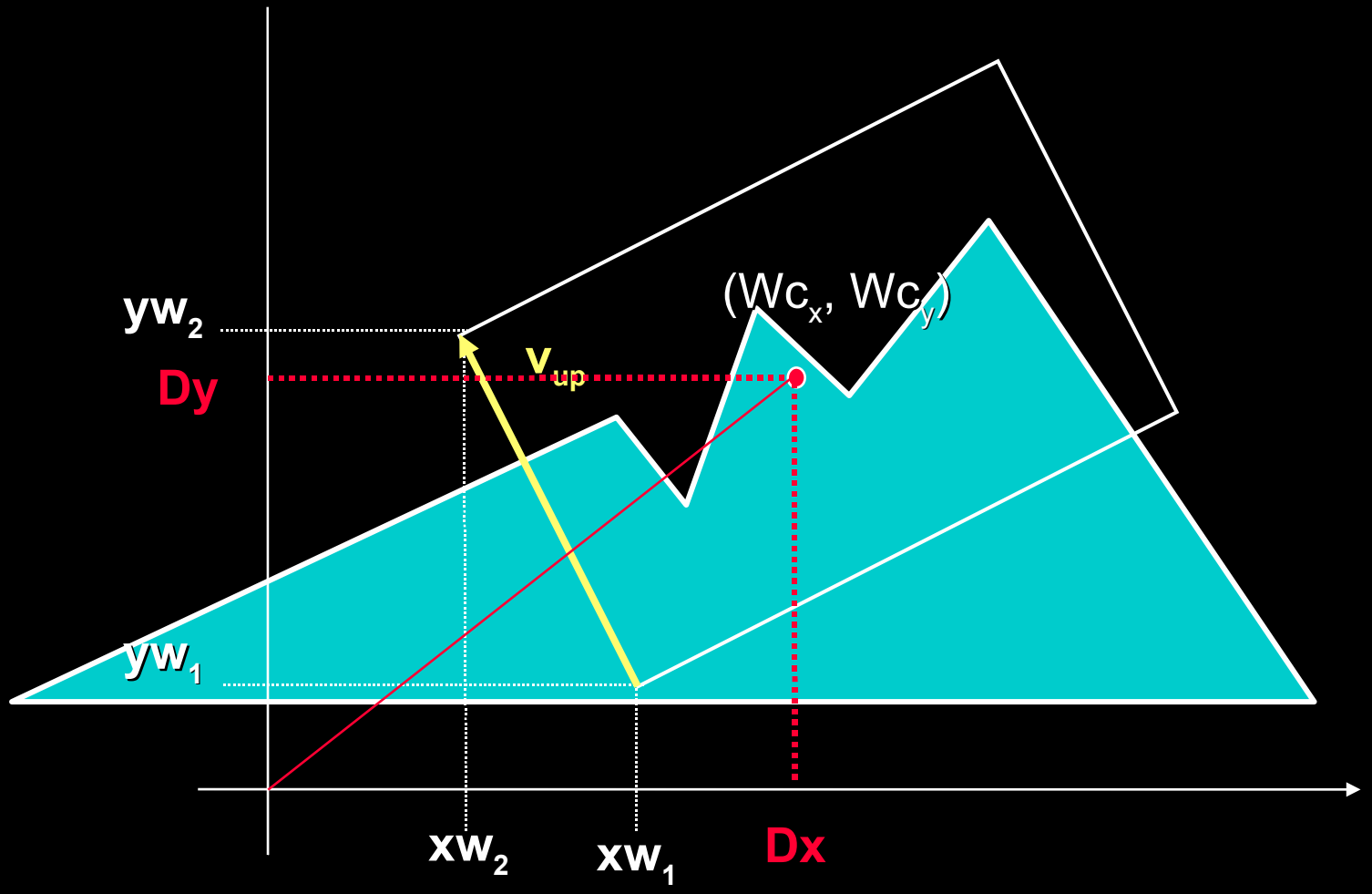
Alternativas ao Sistema de Coordenadas Normalizado

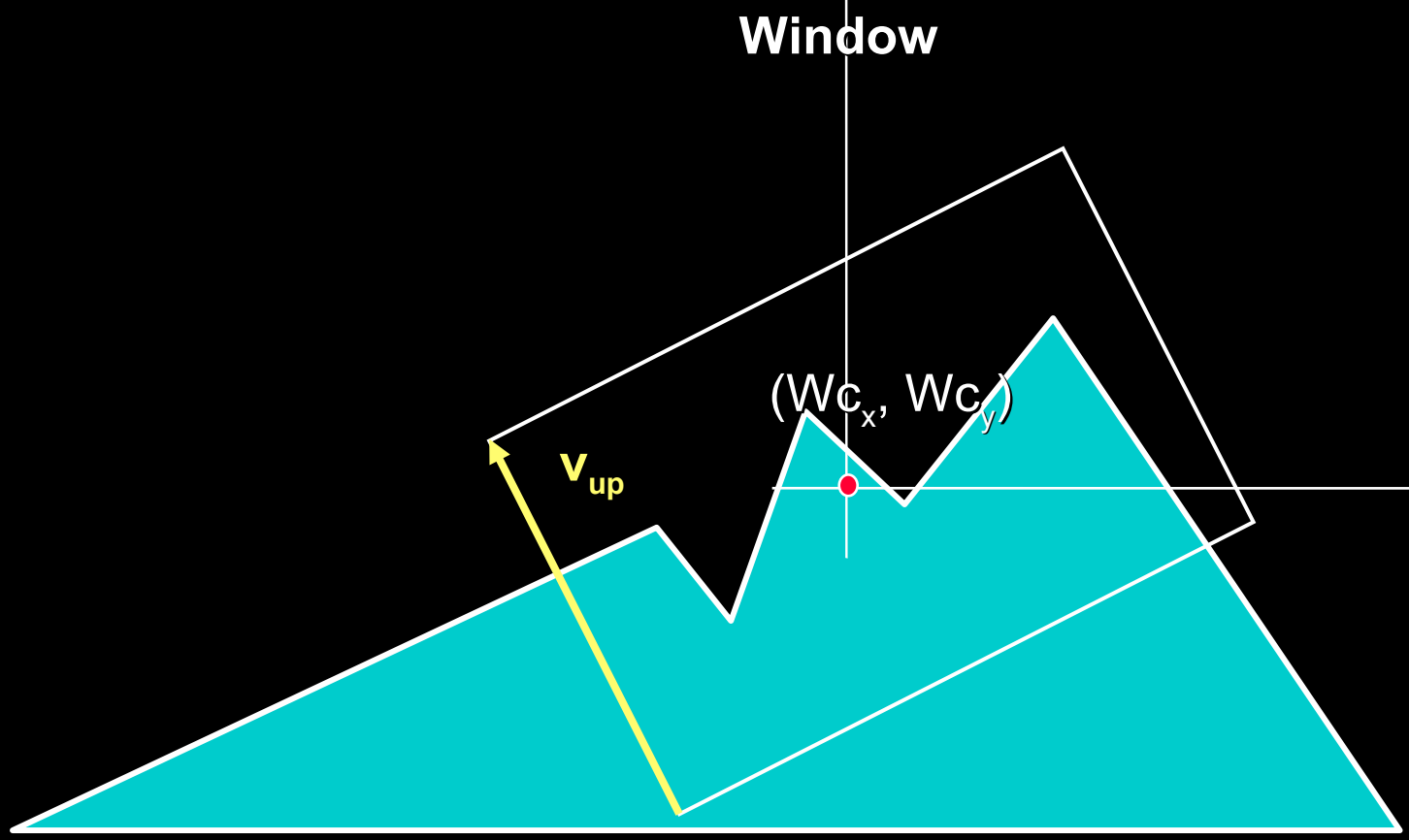
- PPC: Transformamos o sistema de coordenadas para um sistema com origem em $\mathbf{W}_c$ e eixos paralelos aos limites da Window
  - Se $\mathbf{v}_{up}$ for paralelo ao eixo Y, apenas transladamos $\mathbf{W}_c$ para a origem e aplicamos a transformada de viewport.
  - Se $\mathbf{v}_{up}$ não for paralelo ao eixo Y, rotacionamos o mundo e a window em torno de (Wc_x, Wc_y) por -θ(Y, $\mathbf{v}_{up}$).

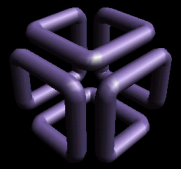
Algoritmo Gerar Descrição em PPC

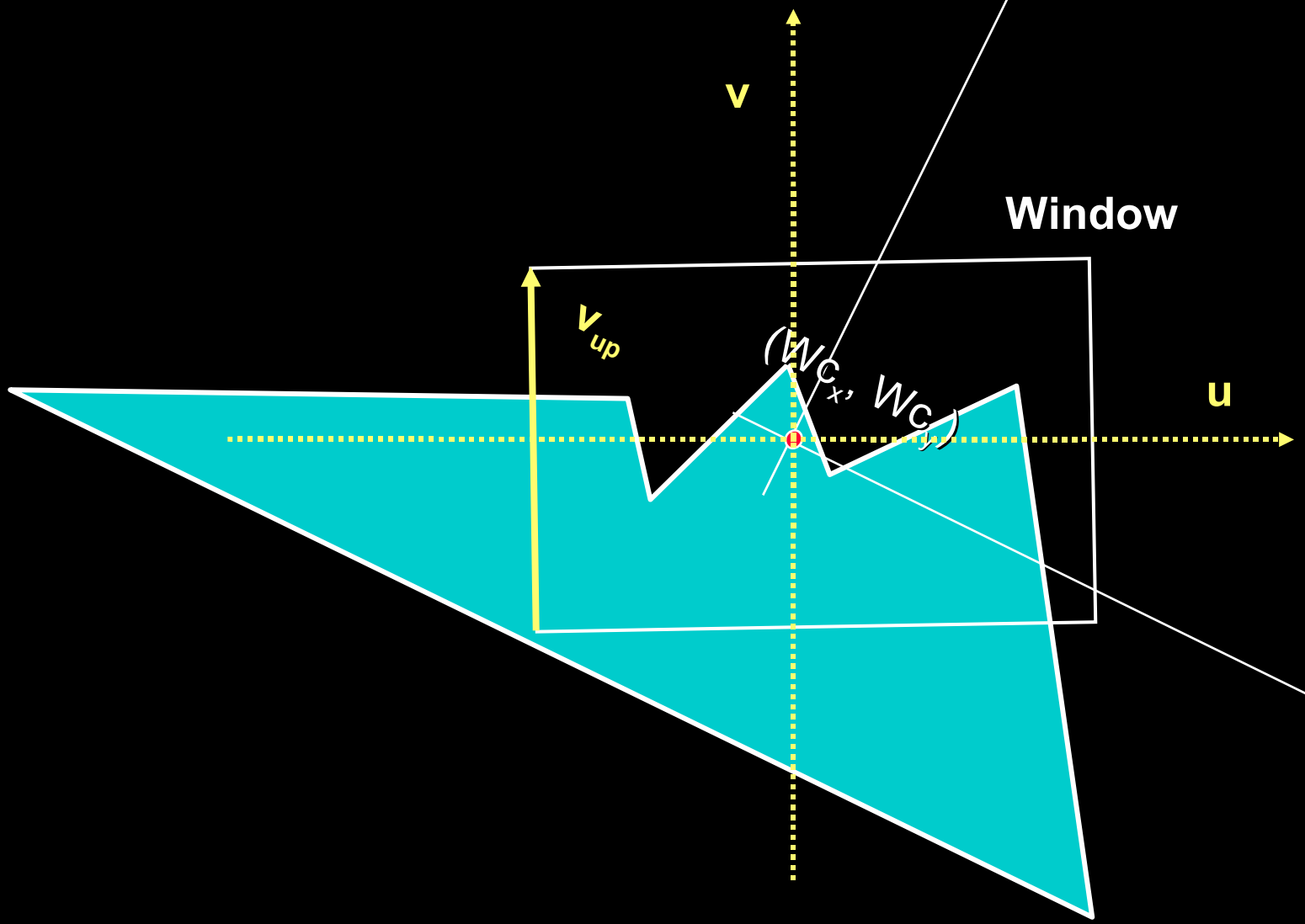
0. Crie ou mova a Window onde desejar.
1. Translade **Wc** para a origem
   - Transladar o mundo de [$-Wc_x$, $-Wc_y$]
2. Determine $\mathbf{V_{up}}$ e o ângulo de $\mathbf{V_{up}}$ com Y
3. Rotacione o mundo de forma a alinhar $\mathbf{V_{up}}$ com o eixo Y
   - Rotacione o mundo por $-\theta(Y, \mathbf{v_{up}})$
4. Armazene as coordenadas CPP de cada objeto.
5. Armazene as coordenadas CPP da Window.
   - Você vai usar na transformada de Viewport.

Algoritmo Gerar Descrição em PPC




Window
(Wc_x, Wc_y)
v_{up}




v
Window
v_{up}
(Wc_x, Wc_y)
u

Como representar e usar o PPC ?

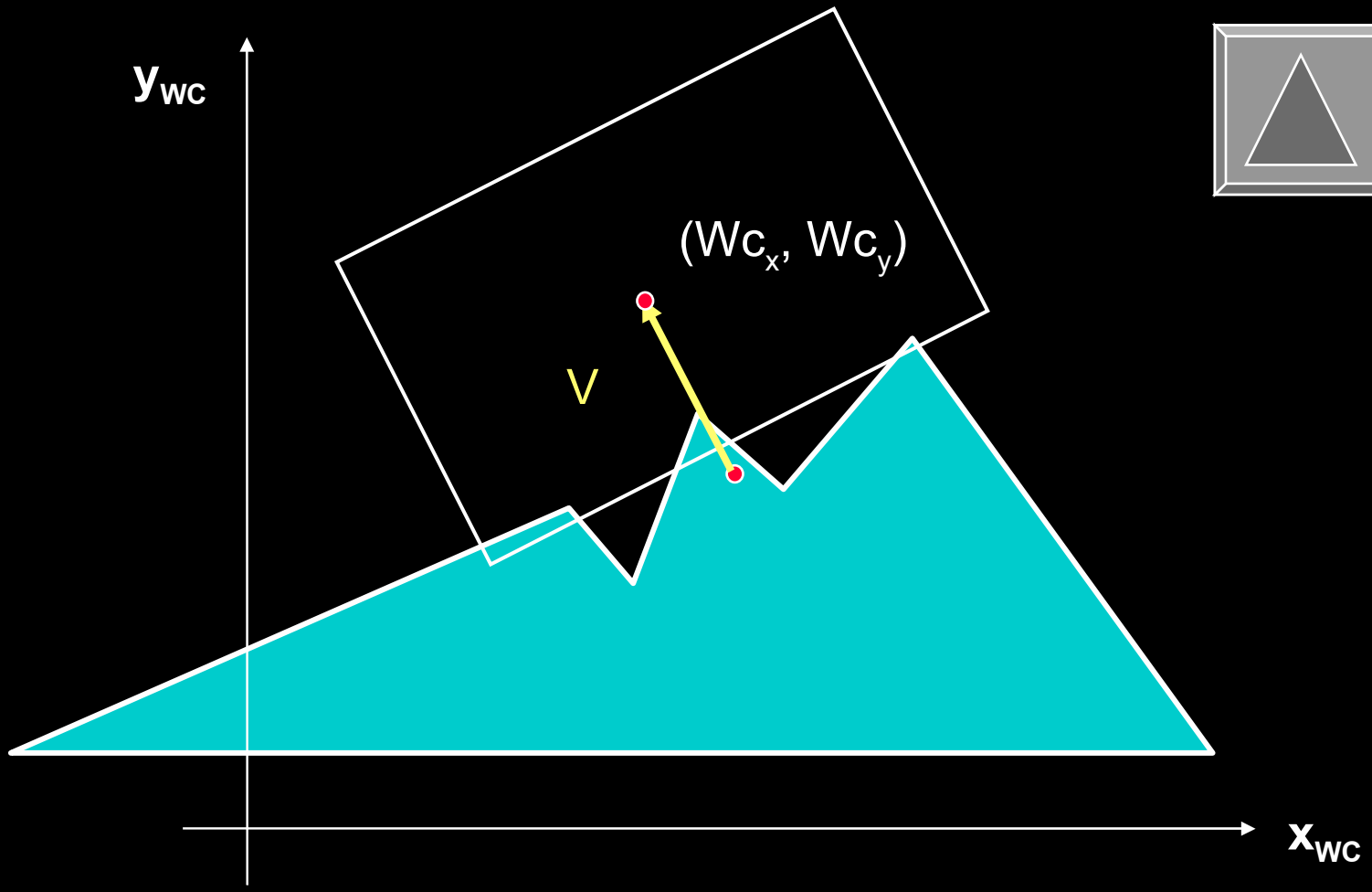
- **Altere o display file**. Cada objeto gráfico de seu display file passará a possuir dois conjuntos de coordenadas:
  - as coordenadas do mundo - WC
  - as PPC
- Altere a definição da Window:
  - Será representada por dois conjuntos de coordenadas também: WC e PPC
- Utilize a representação em PPC dos objetos e da window para a transformada de Viewport.

Como eu navego no Mundo usando o PCC ?

- O referencial do usuário que está sentado ao computador é o PPC
  - Calcule o deslocamento da window no PPC
    - Ex.: "*seta para cima*" faz window mover 10 unidades na direção V_{up}.
- Mova a Window em termos de WC
  - Transforme movimento resultante para WC
  - Mude a posição ou orientação da window no WC
- A plique o algoritmo Gerar Descrição em PPC

Deslocamento Linear

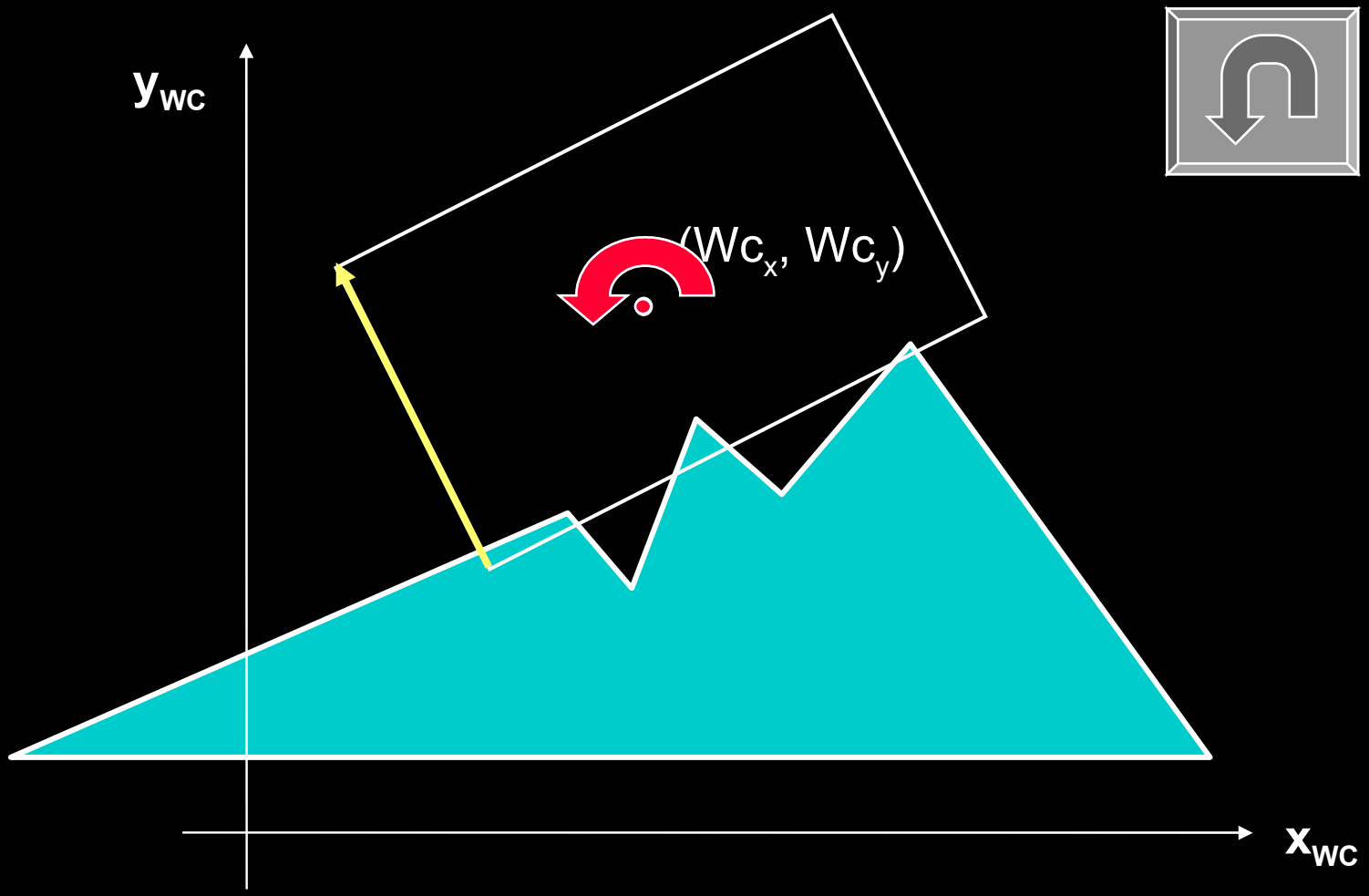
Deslocamento Linear



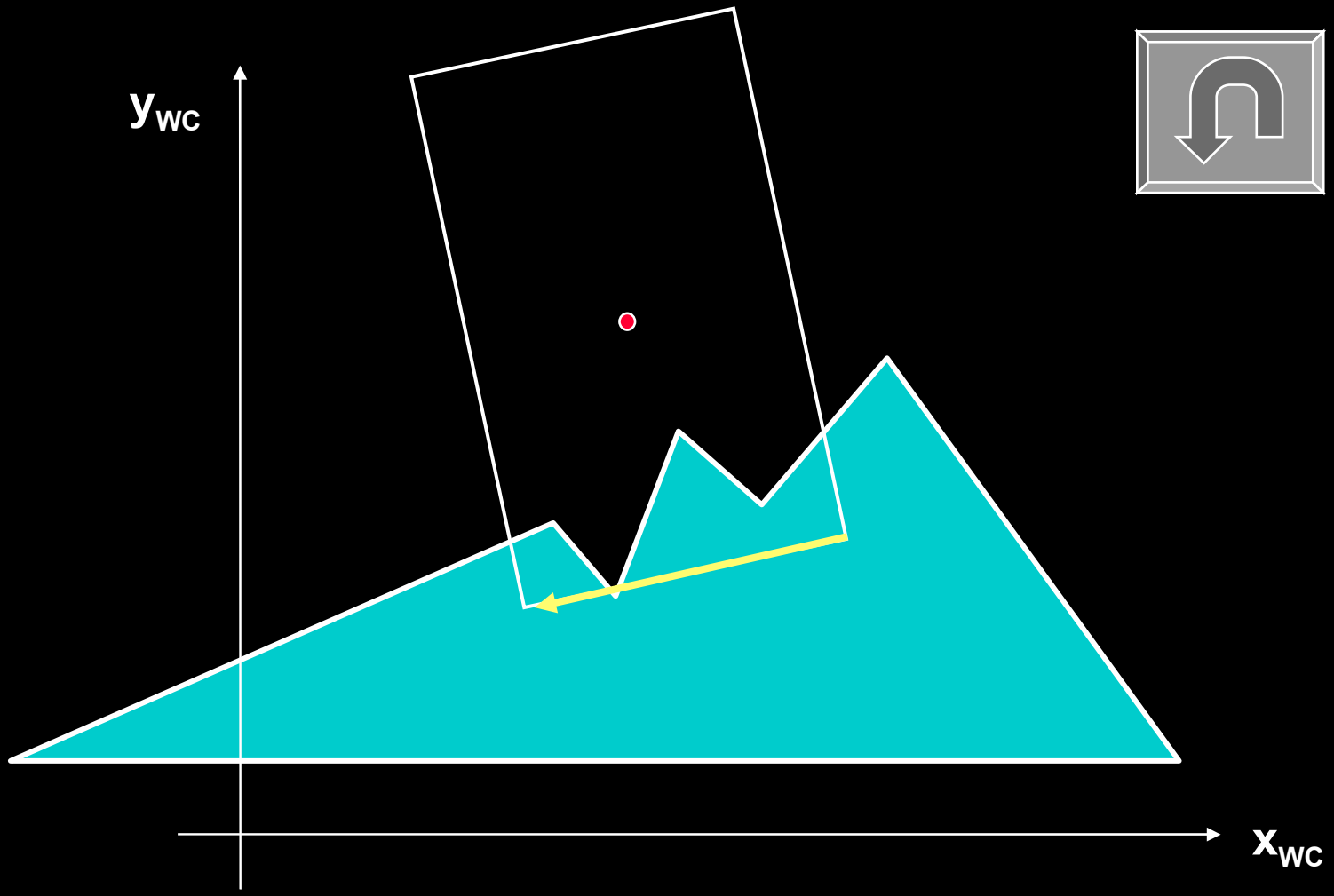
Como implementar a translação da window quando estiver em uma orientação não paralela aos eixos ?

- Ex.: Considere o desenho anterior
  - cada toque em um botão “seta para cima” faz a window se mover 10 unidades no PPC
  - O vetor de deslocamento V é um vetor de mesma direção e sentido que V_{up} e norma 10.
  - V_{up} em WC você tem: são os limites da window.
- Um procedimento possível:
  - Translade V_{up} em WC para a origem.
  - Calcule um vetor de módulo 10 sobre V_{up}
  - Tome as projeções Dx,Dy de V sobre os eixos e adicione a todos os pontos da window e ao W_c.

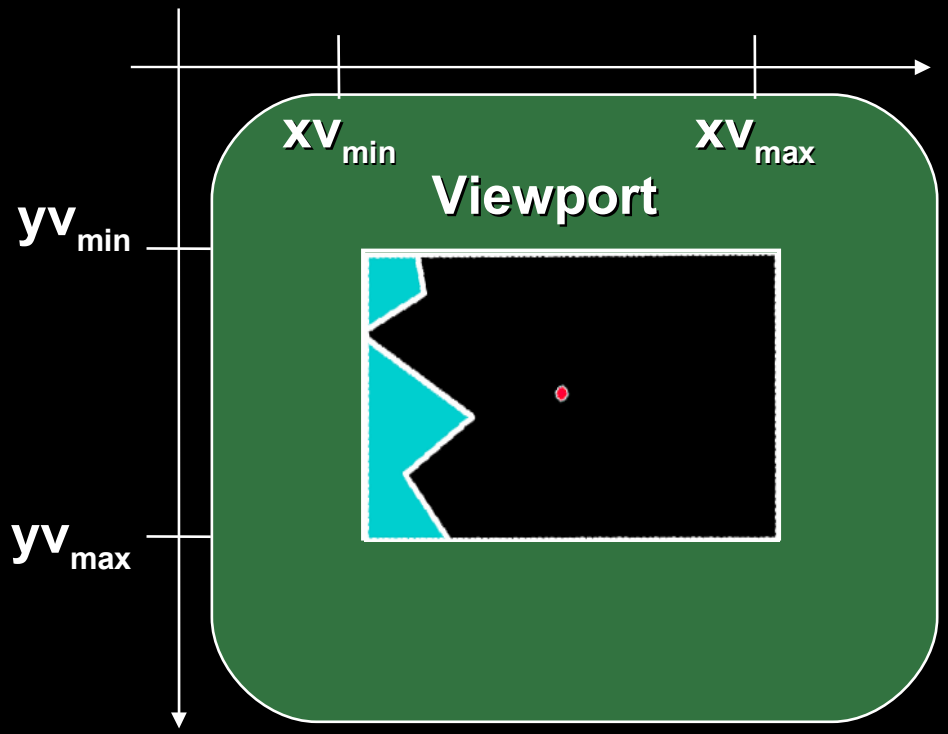
Rotação



Rotação



Rotação

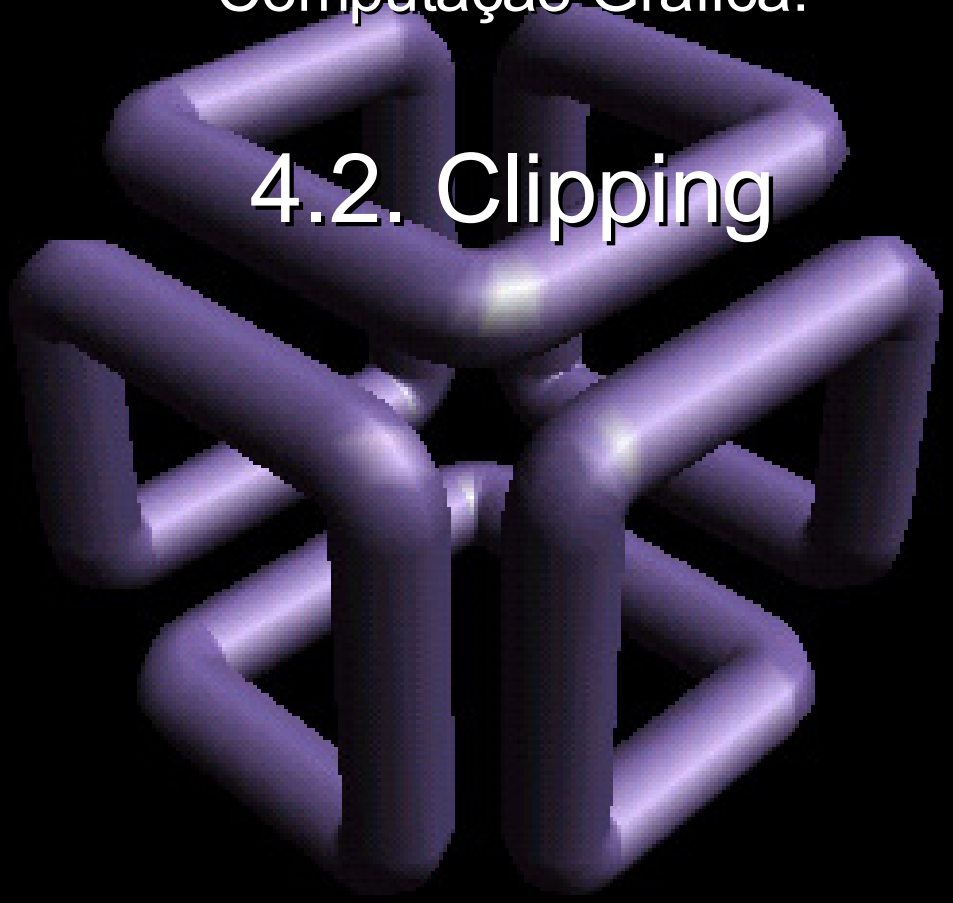


Como implementar a rotação da window durante a navegação ?

- Considere a window como um objeto gráfico qualquer e aplique a rotação de objetos sobre um ponto arbitrário à window em WC.
- Recalcule as coordenadas do mundo em PPC aplicando o algoritmo Gerar Descrição em PPC
  - Observe que o mundo será girado na direção contrária àquela que você girou a window.

Computação Gráfica:

4.2. Clipping



O que é clipping?

- Clipping é um procedimento para o particionamento de primitivas geométricas:
  - Distinguir se primitivas geométricas estão dentro ou fora do **viewing frustum** (regiões do espaço especificadas)
  - Distinguir se primitivas geométricas estão dentro ou fora do **picking frustum**
  - Detectar intersecções entre primitivas
  - Calcular sombras (em 3D)

Por quê recortamos ?

- Clipping é uma otimização importante:
  - É o preprocessamento de visibilidade.
  - Em cenas de mundo real o clipping remove uma porcentagem considerável do ambiente representado.
  - Assegura que somente primitivas potencialmente visíveis serão rasterizadas.

Conceitos de Clipping (Recorte)

- Clip window
  - A região para a qual um objeto deve ser recortado.
  - A forma geométrica de recorte.
- Sempre realizamos recorte em coordenadas da Clip Window
  - No nosso caso utilizaremos a representação em PPC

Conceitos de Clipping (Recorte)

- Tipos de clipping
  - Point clipping
  - Line clipping
  - Area (Polygon) clipping
  - Curve clipping
  - Text clipping

Operações de Clipping (Recorte)

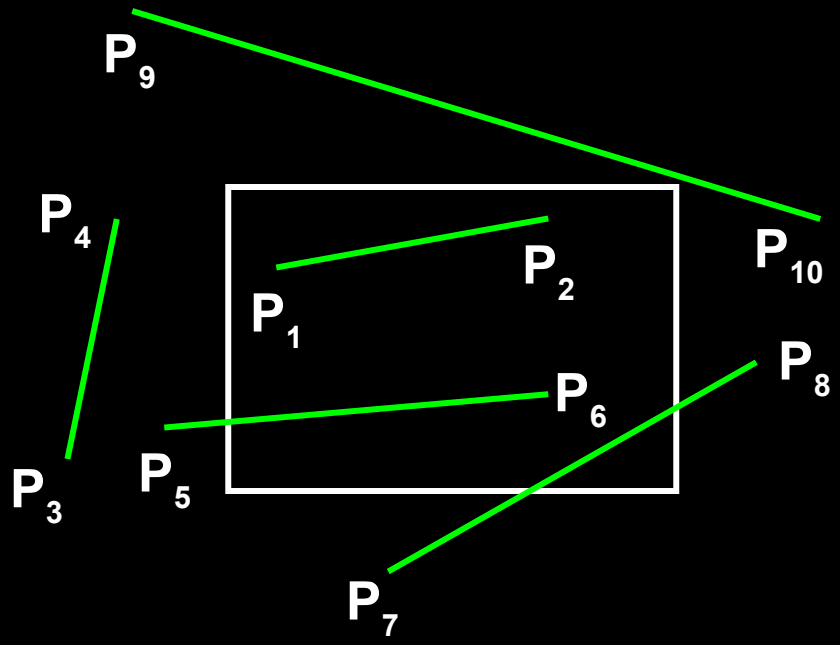
- Point clipping (p/clip window retangular)
  - Processo rápido e muito simples. O ponto que deve ser apresentado na viewport é aquele para o qual as inequações abaixo são satisfeitas:

$$x_{min} \leq x \leq x_{max} \quad e \quad y_{min} \leq y \leq y_{max}$$

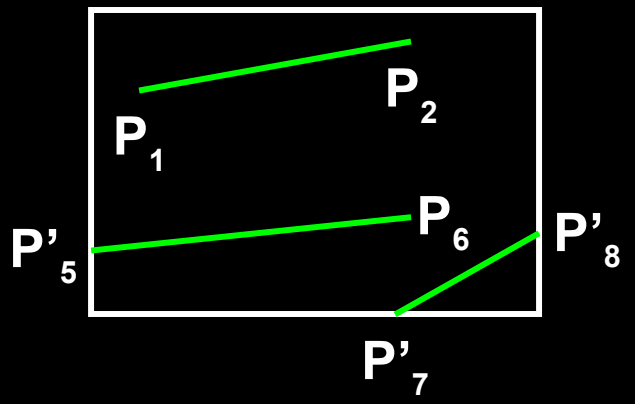
  - Implementado por toda componente de superfície de desenho de linguagens de programação, como os subcanvas ou canvas

Clipping de Linhas

- Relacionamentos possíveis com uma clipping region retangular



Antes do Clipping



Depois do Clipping

Clipping de Linhas

- Relacionamentos possíveis
  - Completamente dentro do window de clipagem (desenhamos)
  - Completamente fora do window (não desenhamos)
  - Parcialmente dentro do window (o que desenhamos ?)

Clipping de Linhas usando representação paramétrica da reta

- Checagem por meio da equação paramétrica envolvendo os limites da janela e a própria linha.
  - Grande quantidade de cálculos e teste e não é muito eficiente. Num display típico, centenas ou milhares de linhas de linha podem ser encontradas.
  - Algoritmo eficiente:
    - Deve promover alguns testes iniciais de forma a determinar se cálculos de interseção são realmente necessários.
    - Par de pontos finais pode ser checado para observar se ambos pertencem à janela.

Clipping de Linhas usando representação paramétrica da reta

- Representação paramétrica da reta

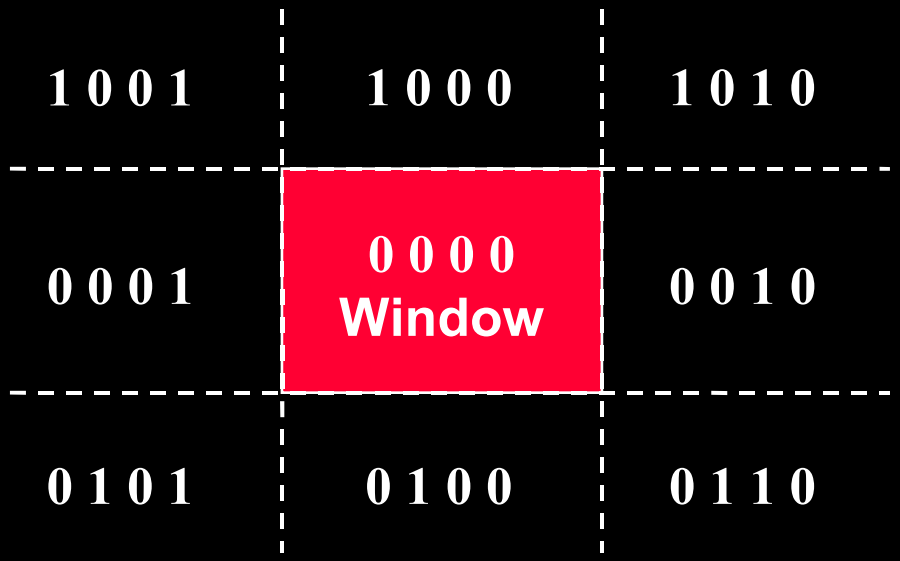
  $x = x_1 + u(x_2 - x_1)$

  $y = y_1 + u(y_2 - y_1)$

  - Usamos o valor de *u* para calcular a intersecção com uma das bordas do retângulo definido pelo window
    - Fora: < 0 ou > 1
    - Dentro: de 0 a 1
  - Se o valor da intersecção for maior que o tamanho do window, a intersecção ocorrerá em algum lugar fora deste.

Recorte de Linhas de Cohen-Sutherland

- Código de regiões
  - Código binário de 4 dígitos atribuído a todo ponto final de uma linha na figura: region codes (RC)
  - Numeramos as posições no código de regiões de 1 a 4. Cantos são representados por soma de valores.



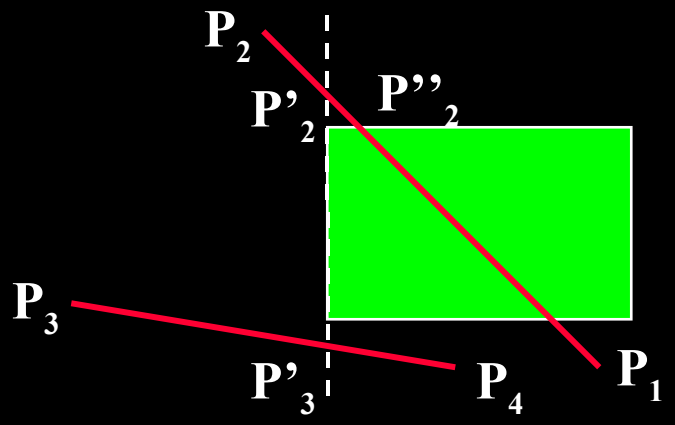
RC[1]: acima
RC[2]: abaixo
RC[3]: direita
RC[4]: esquerda

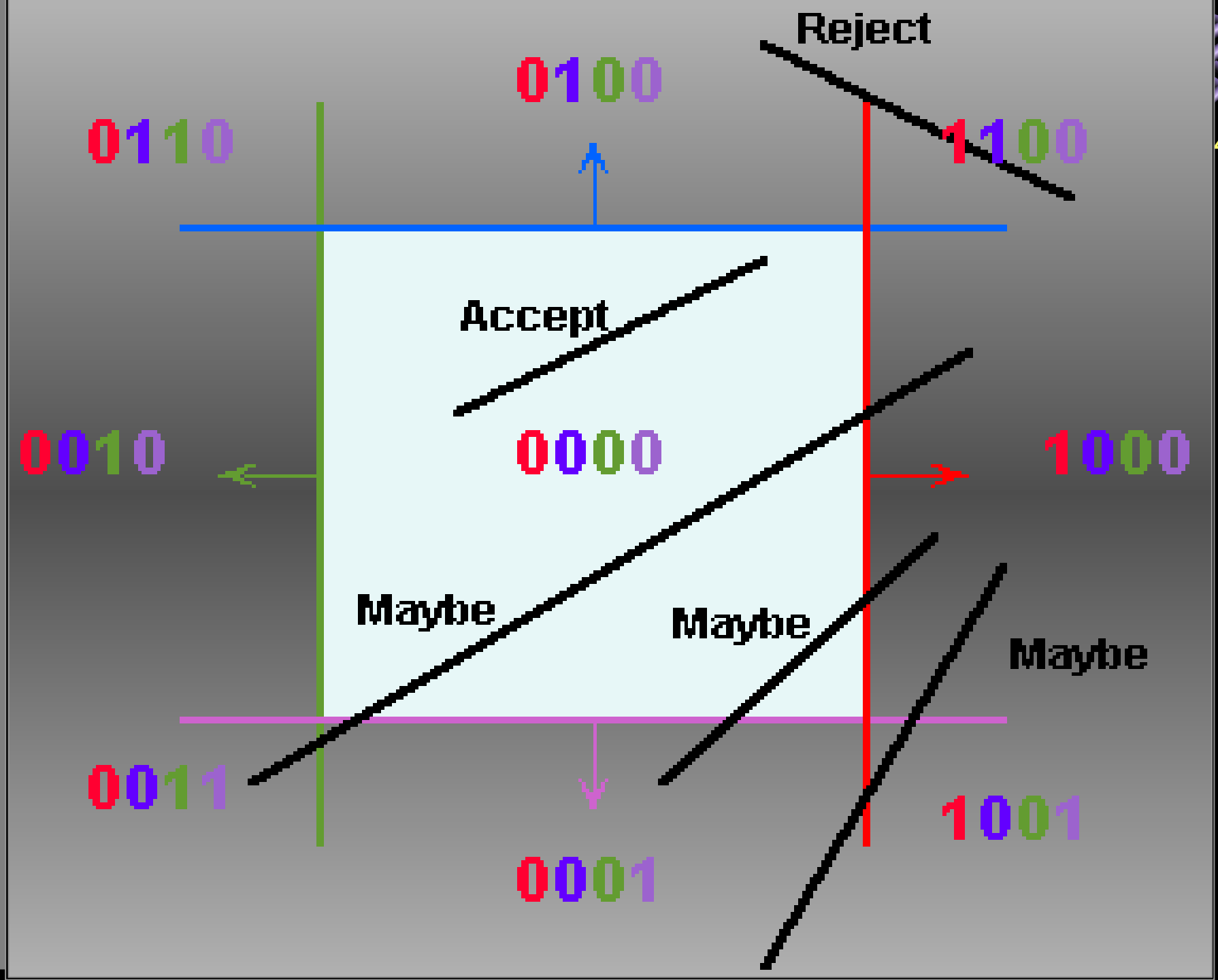
Recorte de Linhas de Cohen-Sutherland

- Valores dos *region codes*
  - Determinados através da comparação das coordenadas das extremidades aos limites da window.
    - Valor 1 in em qquer posição: O ponto está neste quadrante.
  - Determinado:
    - Calcule as diferenças entre coordenadas das extremidades e limites de clipagem.
    - Use o sinal resultante para montar o código, setando o valor do bit correspondente.

Recorte de Linhas de Cohen-Sutherland

- Possíveis relacionamentos:
  - Completamente contida na janela
    - 0000 para ambas as extremidades
  - Completamente fora da janela
    - *And logico* dos *region codes* para ambas as extremidades resulta diferente de 0000
  - Parcialmente
    - Extremidades possuem *region codes* diferentes
    - *And logico* dos *region codes* para ambas as extremidades resulta em 0000




Reject
0100
0110
1100
Accept
0010
0000
1000
Maybe
Maybe
Maybe
0011
1001
0001

Algoritmo Geral para Recorte de Linhas de Cohen-Sutherland

P1: Associar códigos aos pontos extremos c/regra:

Para as coordenadas X de ambos os pontos da linha:

se $x_i < X_{wesq}$ então $RC_i[4] <- 1$ senão $RC_i[4] <- 0$

se $x_i > X_{wdir}$ então $RC_i[3] <- 1$ senão $RC_i[3] <- 0$

Para as coordenadas Y de ambos os pontos da linha:

se $y_i < Y_{wfundo}$ então $RC_i[2] <- 1$ senão $RC_i[2] <- 0$

se $y_i > Y_{wtopo}$ então $RC_i[1] <- 1$ senão $RC_i[1] <- 0$

Algoritmo Geral para Recorte de Linhas de Cohen-Sutherland

- P2: Verificar se a linha é totalmente visível ou invisível ou parcialmente visível

  Completamente contida na janela
  - $RC_{inicio} = RC_{fim} = [0\ 0\ 0\ 0]$

  Completamente fora da janela
  - RC_{inicio} & RC_{fim} <> [0 0 0 0]

  Parcialmente
  - $RC_{inicio} <> RC_{fim}$
  - RC_{inicio} & $RC_{fim} = [0\ 0\ 0\ 0]$

Algoritmo Geral para Recorte de Linhas de Cohen-Sutherland

- P3: Se a linha é parcialmente visível devem ser calculadas as intersecções:

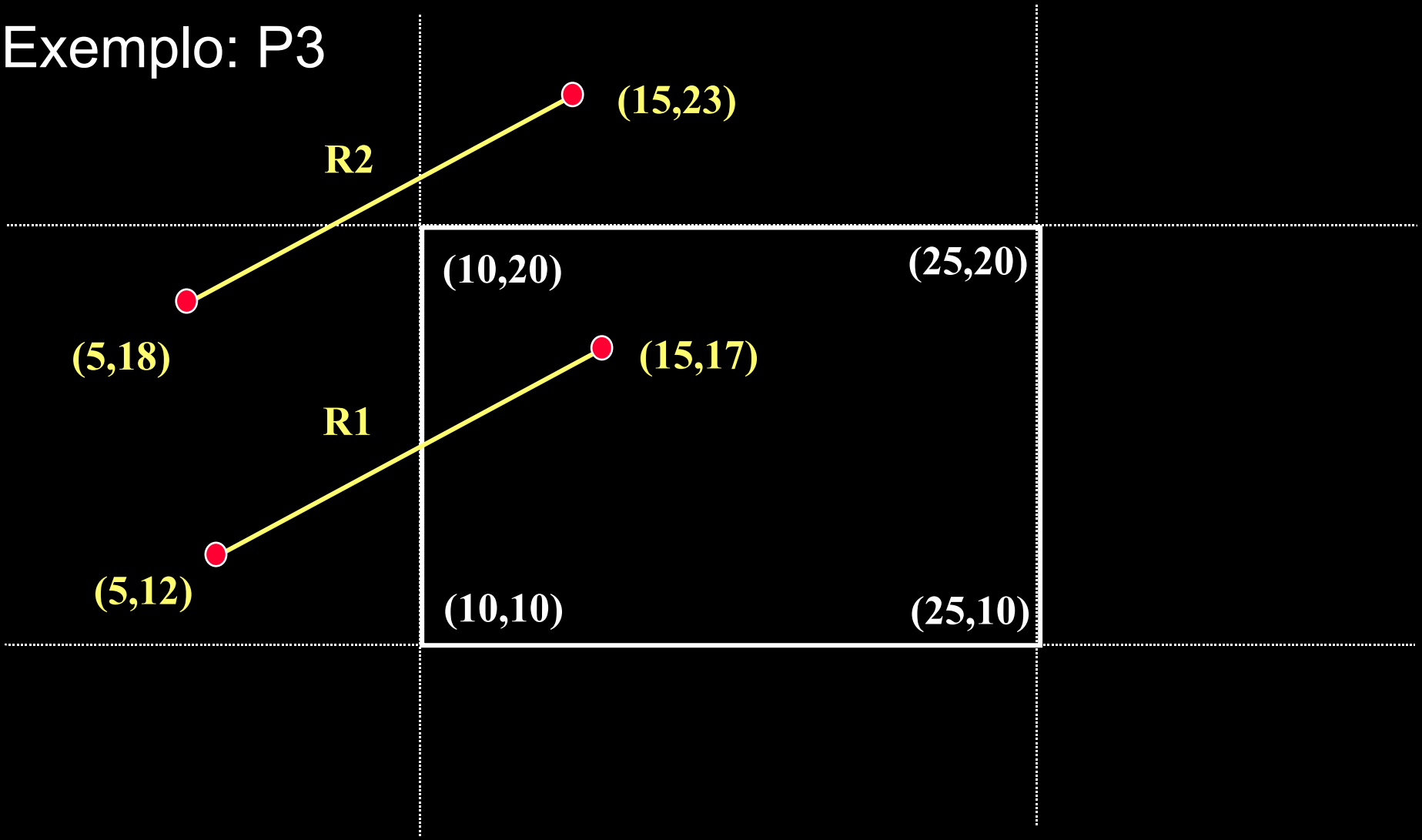
  Esquerda: $x_E, y = m^* (x_E - x_1) + y_1$; **M diferente de 0**

  Direita: $x_D, y = m^* (x_D - x_1) + y_1$

  Topo: $y_T, x = x_1 + 1/m \,.\, (y_T - y_1)$

  Fundo: $y_F, x = x_1 + 1/m \,.\, (y_F - y_1)$

Exemplo: P3



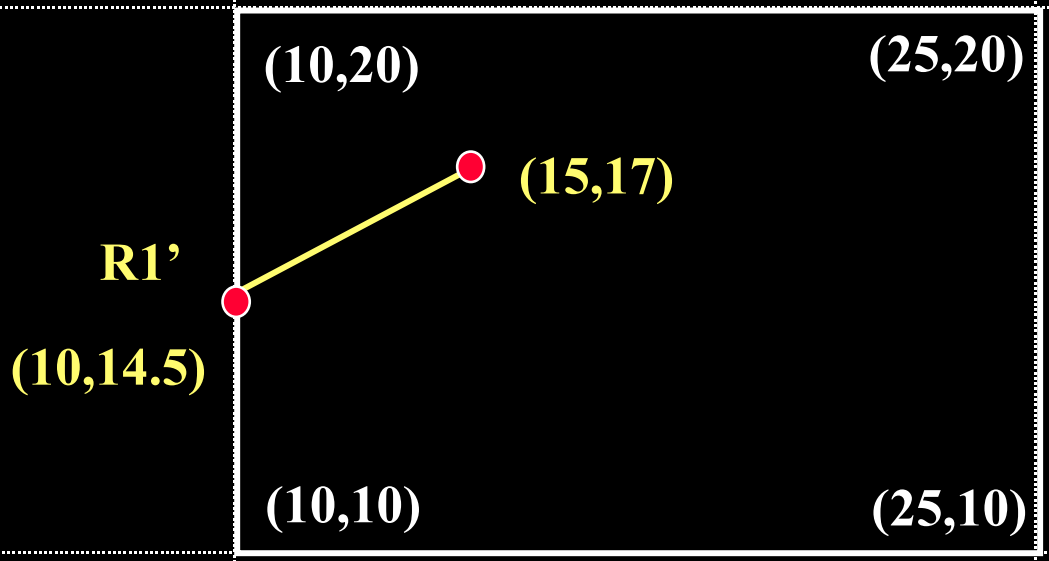
Clipando R1

- RC1 = [0 0 0 1] -> esquerda e RC2 = [0 0 0 0] -> dentro
  - Clipamos apenas à esquerda e calculamos novo y1 para a linha.
- Cálculo do coeficiente angular:
  - m = (y2 - y1)/(x2 - x1) = (17 - 12)/(15 - 5) = 5 / 10 = 0.5
- Aplicamos a fórmula esq.: $y_{intersec} = m^* (x_E - x_1) + y_1$
  - $y_{intersec} = 0.5 * (10 - 5) + 12 = 2.5 + 12 = 14.5$
- Como $14.5 > y_F = 10$ e $14.5 < y_T = 20$ aceitamos.
- Nova linha clipada é:
  - (10,14.5)(15,17)

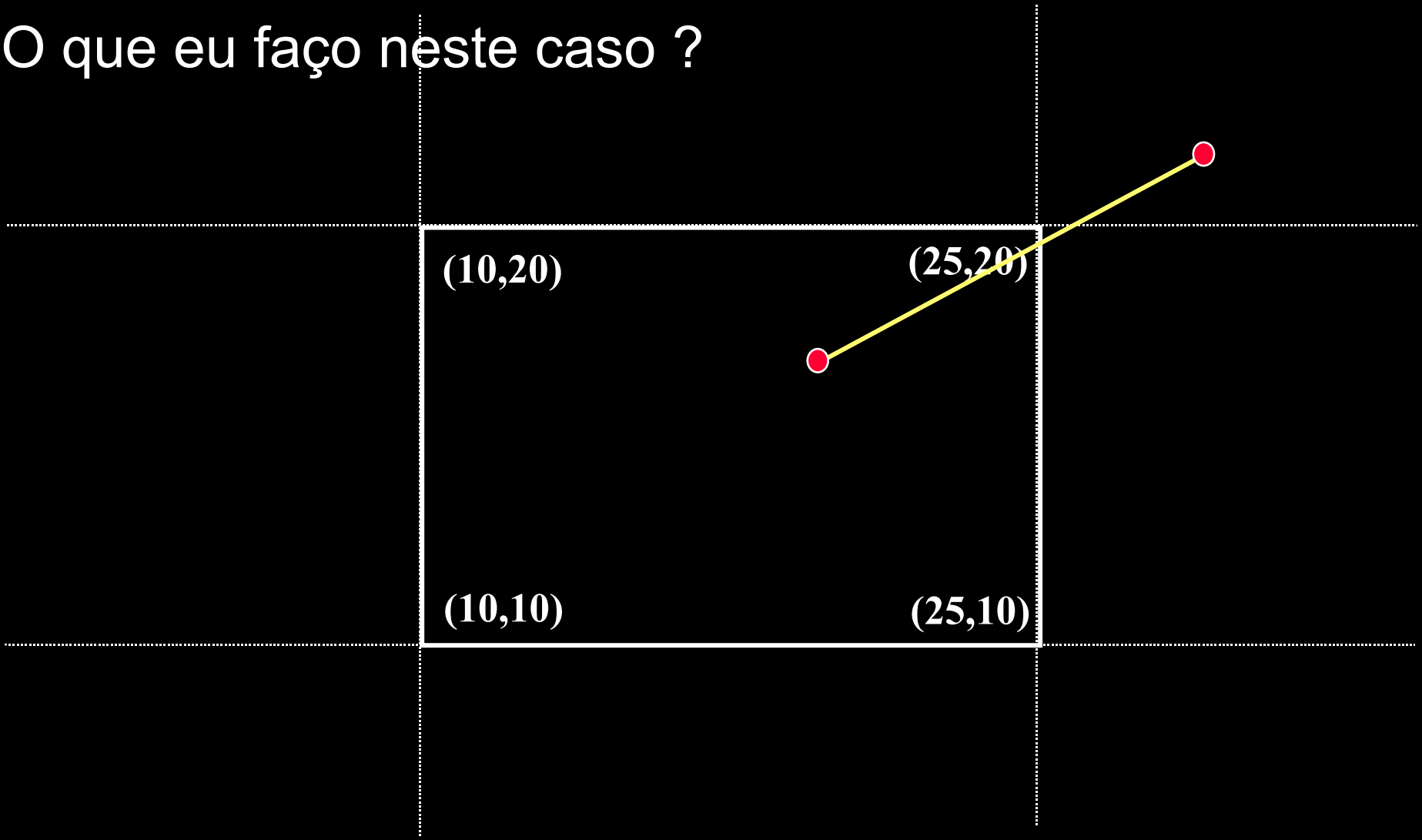
Clipando R2

- RC1 = [0 0 0 1] -> esquerda e RC2 = [1 0 0 0] -> topo
  - Clipamos à esquerda e calculamos novo y1 para a linha.
  - Clipamos ao topo e calculamos novo x2 para a linha
- Cálculo do coeficiente angular:
  - m = (y2 - y1)/(x2 - x1) = (23 - 18)/(15 - 5) = 5 / 10 = 0.5
- Aplicamos a fórmula esq.: $y_{intersec} = m^* (x_E - x_1) + y_1$
  - $y_{intersec} = 0.5 * (10 - 5) + 18 = 2.5 + 18 = 20.5$
- Como $20.5 > y_F = 10$ mas 20.5 não é $< y_T = 20$ rejeitamos.
  - Intersecção é fora da window
  - Não é necessário calcular x2 - se uma intersecção for fora a outra também o será.
- Linha é descartada como não visível.

Resultado



O que eu faço neste caso ?



Quando um ponto cair em um dos cantos...

- Quando o RC associado a um ponto de uma linha possuir dois “1”, calculamos a intersecção com as bordas da window para os dois casos.
  - No exemplo calcularíamos um x2 pelo topo e um y2 pela direita.
- Aceitamos dentre os dois valores calculados, aquele que se encontrar sobre a window
  - O outro estará fora.
  - Exceção: linha cai exatamente sobre o canto.

Clipping de Linhas de Liang-Barsky

- Em 1978 Cyrus e Beck publicaram um novo algoritmo para o clipping de linhas usável para clipar linhas contra um polígono convexo em 2D ou um poliedro convexo em 3D usando a equação paramétrica.
- Em 1984 Liang e Barsky reformularam este algoritmo, tornando-o mais eficiente.
  - Esta versão veremos a seguir.

Liang-Barsky Line Clipping

- Reescreva as equações paramétricas como segue:

$$x = x_1 + u\Delta x$$

$$y = y_1 + u\Delta y, \quad 0 \le u \le 1$$

- Sendo as condições de clipagem já vistas:

$$xw_{min} \le x_1 + u\Delta x \le xw_{max}$$

$$yw_{min} \le y_1 + u\Delta y \le yw_{max}$$

  - Cada uma dessas inequações pode então ser expressa como:

$$up_k \le q_k, \qquad k = 1, 2, 3, 4$$

Liang-Barsky Line Clipping

- Onde os parâmetros *p* e *q* são definidos como:

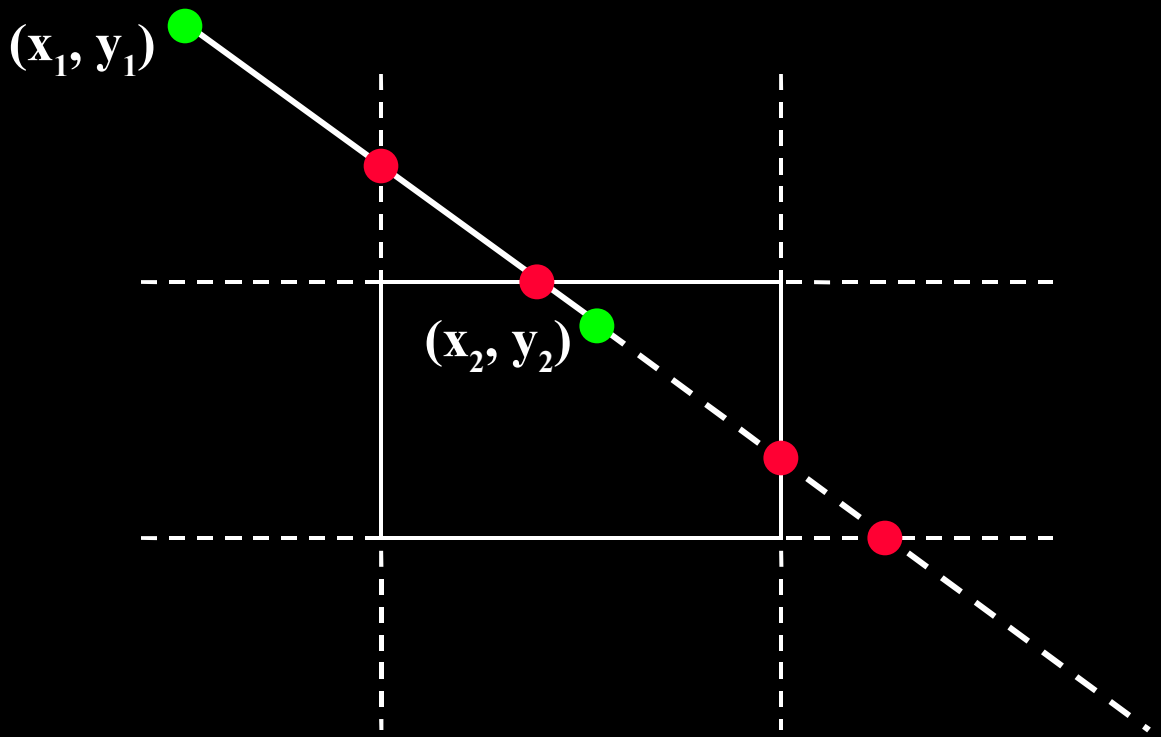
$$p_1 = -\Delta x, \quad q_1 = x_1 - xw_{min}$$
$$p_2 = \Delta x, \quad q_2 = xw_{max} - x_1$$
$$p_3 = -\Delta y, \quad q_3 = y_1 - yw_{min}$$
$$p_4 = \Delta y, \quad q_4 = yw_{max} - y_1$$

Liang-Barsky Line Clipping

Condições

- $p_k = 0$, paralela a um dos limites:
  - $q_k < 0$, fora dos limites
  - $q_k >= 0$, dentro dos limites
- $p_k < 0$, a linha vem de fora para dentro
- $p_k > 0$, a linha vem de dentro para fora

Liang-Barsky Line Clipping



Qual a parte que está dentro ?

Liang-Barsky Line Clipping

- Os parâmetros u_1 and u_2 definem qual parte está dentro do retângulo:
  - O valor de u_1: de fora para dentro ($p_k < 0$)

$$u_1 = max(0, r_k' s) \qquad r_k = \frac{q_k}{p_k}$$

  - O valor de u_2: De dentro para fora ($p_k > 0$)

$$u_2 = min(1, r_k' s) \qquad r_k = \frac{q_k}{p_k}$$

  - Se $u_1 > u_2$, a linha está completamente fora.

Liang-Barsky Line Clipping

$p_1 < 0, \quad p_3 < 0$

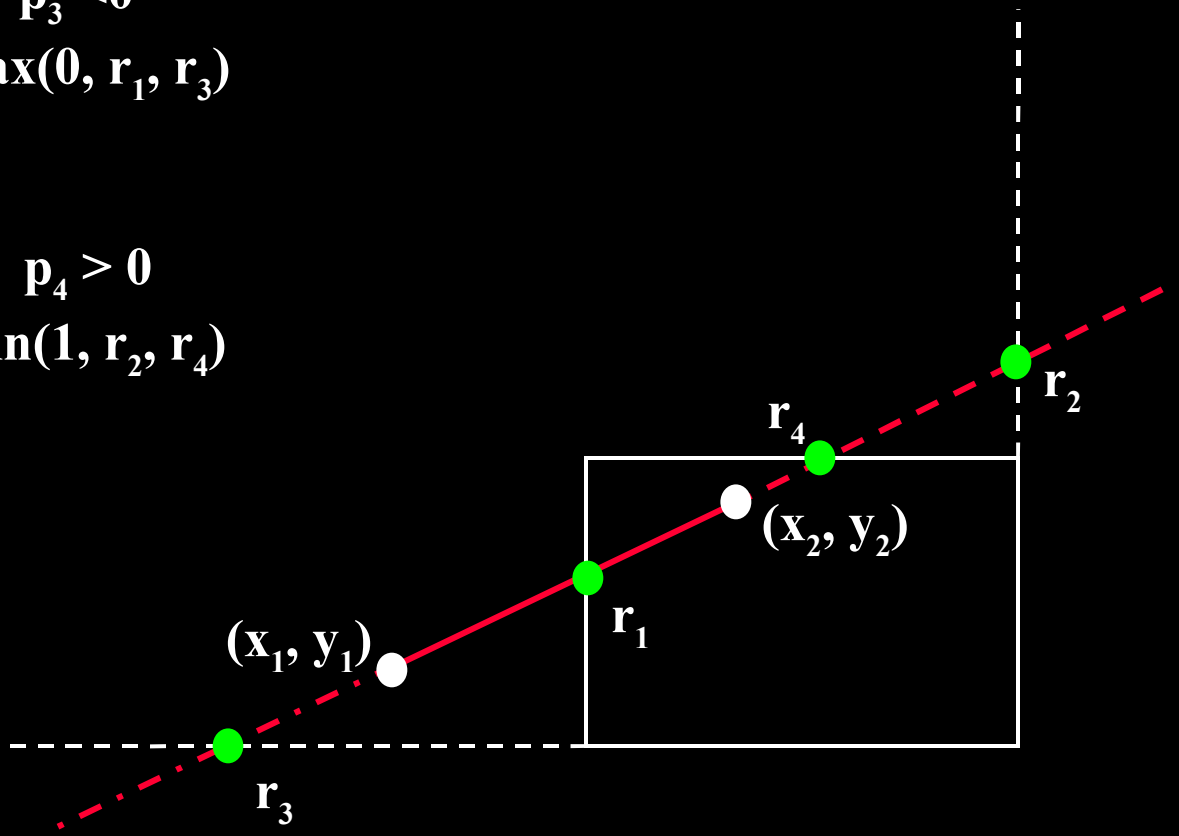
$u_1 = \max(0, r_1, r_3)$

$\quad = r_1$

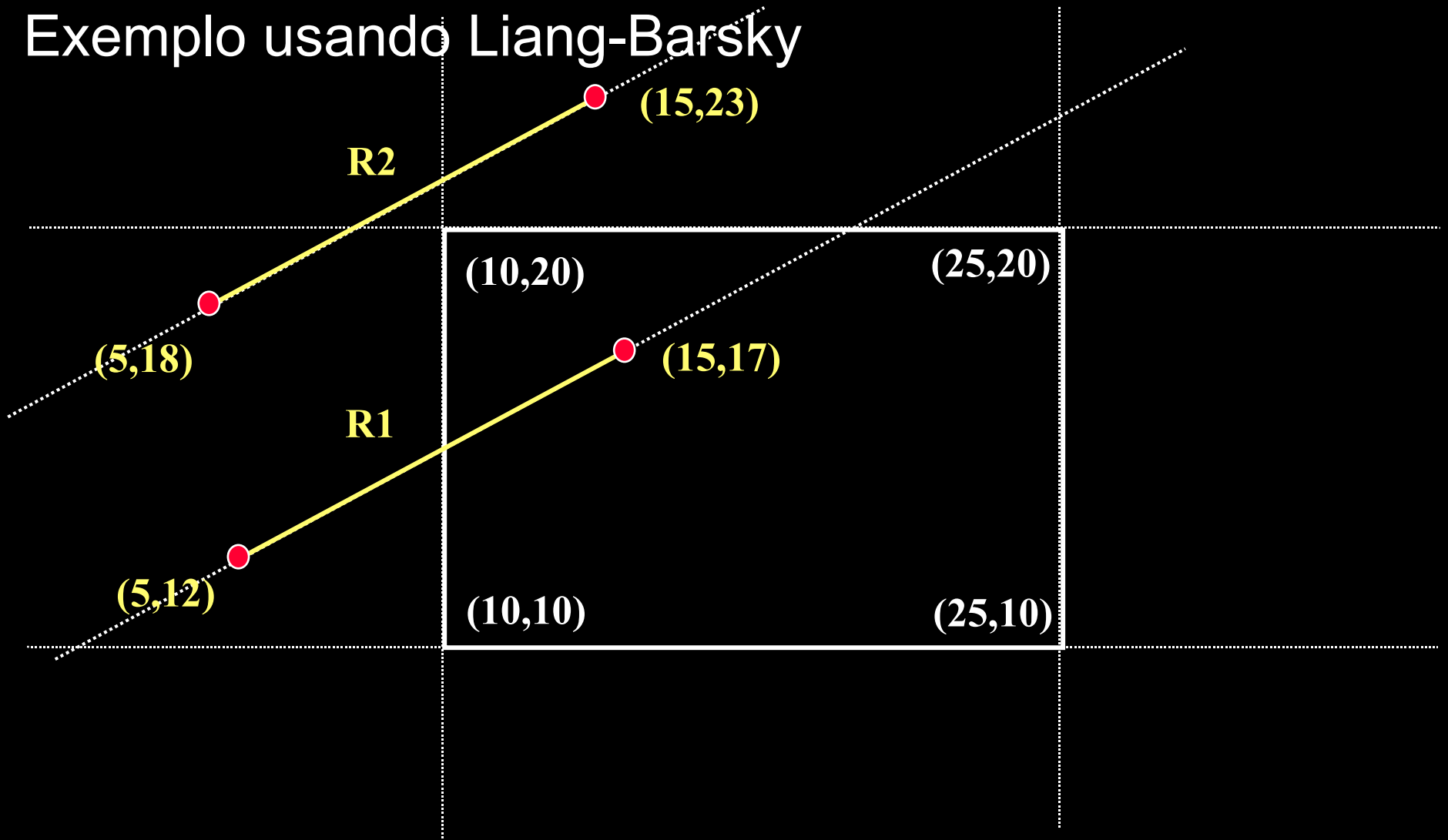
$p_2 > 0, \quad p_4 > 0$

$u_2 = \min(1, r_2, r_4)$

$\quad = 1$



Exemplo usando Liang-Barsky



Calcular p e q para R1

- $p1 = -\Delta x = -(15 - 5) = -10$
- $p2 = \Delta x = 15 - 5 = 10$
- $p3 = -\Delta y = -(17 - 12) = -5$
- $p4 = \Delta y = 17 - 12 = 5$
- $q1 = x1 - x_{wmin} = 5 - 10 = -5$
- $q2 = x_{wmax} - x1 = 25 - 5 = 20$
- $q3 = y1 - y_{wmin} = 12 - 10 = 2$
- $q4 = y_{wmax} - y1 = 20 - 12 = 8$

Calcular u1 para R1

$$x = x_1 + u\Delta x$$
$$y = y_1 + u\Delta y, \quad 0 \leq u \leq 1$$

- p1 = -10 < 0
- p3 = -5 < 0

- r1 = q1/p1 = -5/-10 = 0.5
- r3 = q3/p3 = 2/-5 = - 0.4
- u1 = max (0, r1, r3) = max (0, 0.5, -0.4) = 0.5

Substituindo na eq.paramétrica:

- x = 5 + 0.5 * 10 = 10 (o que nós já sabíamos)
- y = 12 + 0.5 * 5 = 14.5 (o que nós não sabíamos)

Calcular u2 para R1

- p2 = 10 > 0
- p4 = 5 > 0

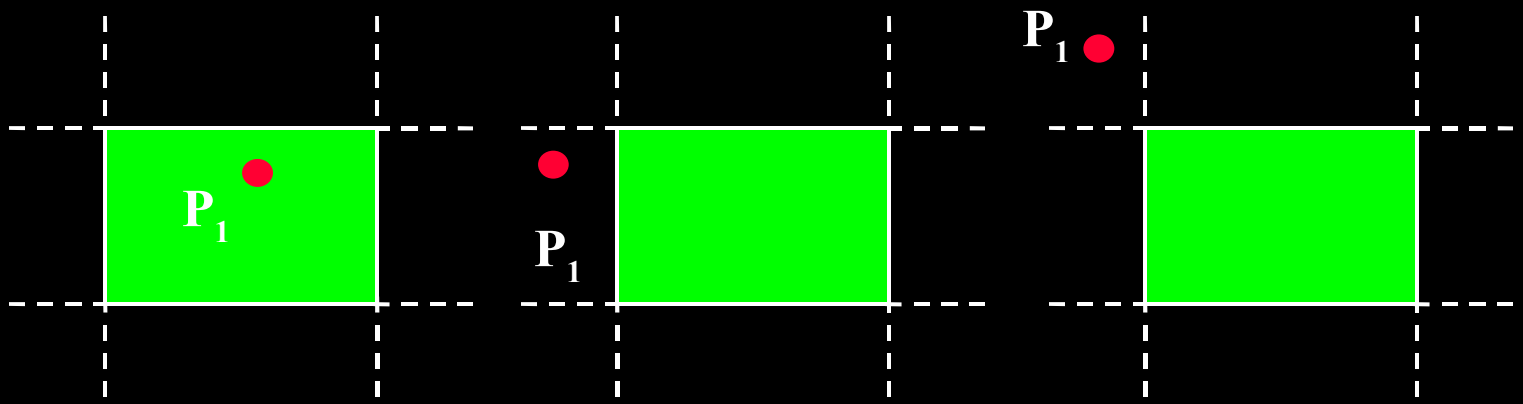
- r2 = q2/p2 = 20/10 = 2
- r4 = q4/p4 = 8/5 = 1.6
- u2 = min (1, r1, r3) = min (1, 2, 1.6) = 1

- Como u2 resulta 1, rejeitamos o cálculo de novos valores de dentro para fora.

Nicholl-Lee-Nicholl Line Clipping

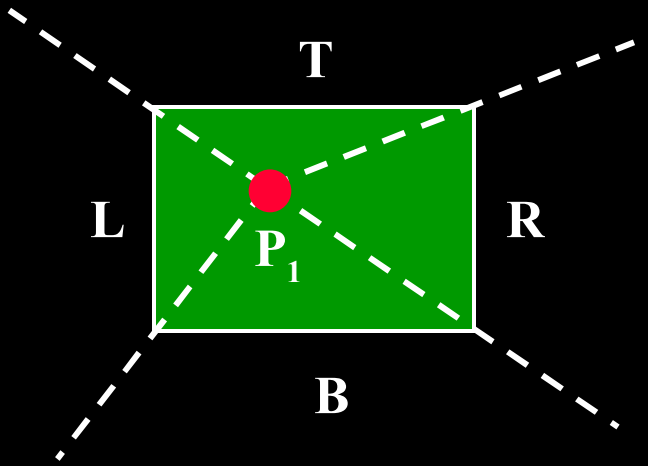
- Comparado aos algoritmos C-S e L-B:
  - NLN realiza menos comparações e divisões.
  - NLN pode ser aplicado somente a 2D clipping.
- O algoritmo NLN
  - Clipa uma linha com extremas P_1 e P_2
  - Passo 1: Determina P_1 para os nove quadrantes.
    - Somente três regiões são consideradas
    - Para o resto usa-se uma transformada de simetria
  - Passo 2: Depois determina-se a posição de P_2 em relação a P_1.

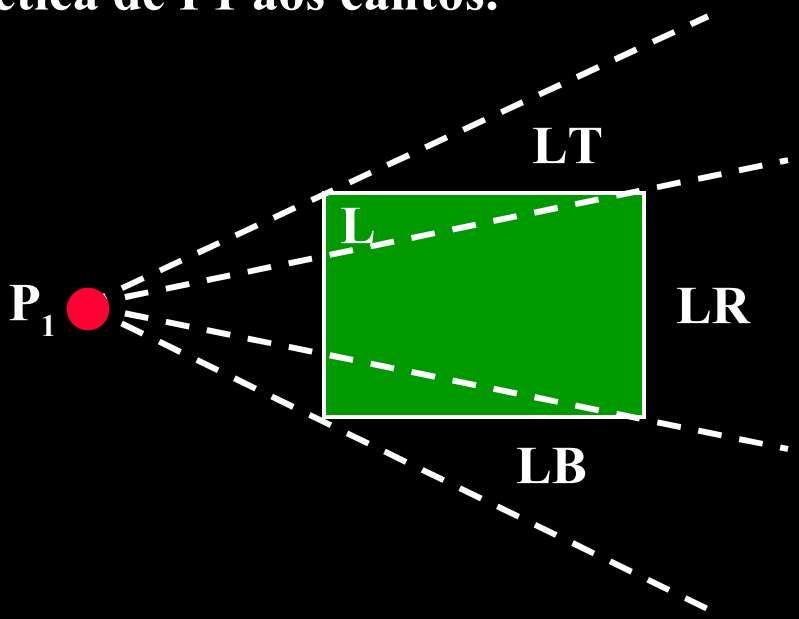
Nicholl-Lee-Nicholl Line Clipping: Passo 1a



Nicholl-Lee-Nicholl Line Clipping: Passo 1b

Cálculo dos ângulos de uma linha hipotética de P1 aos cantos.
Determinar regiões L,T,R,L,TR,etc



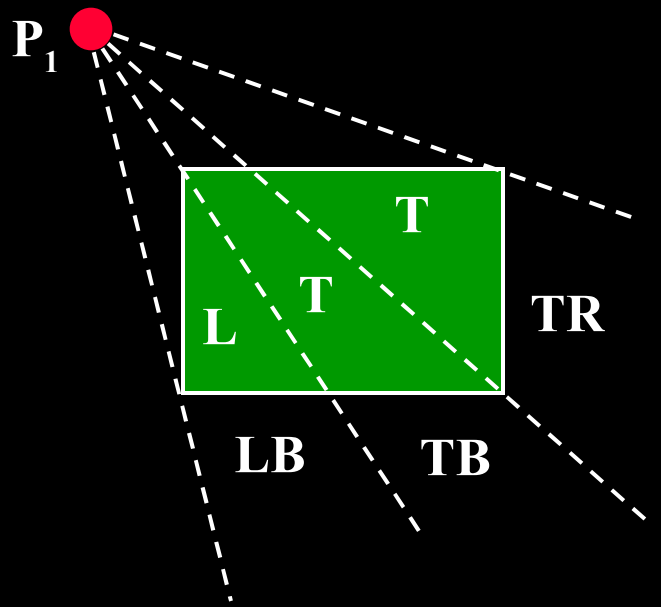


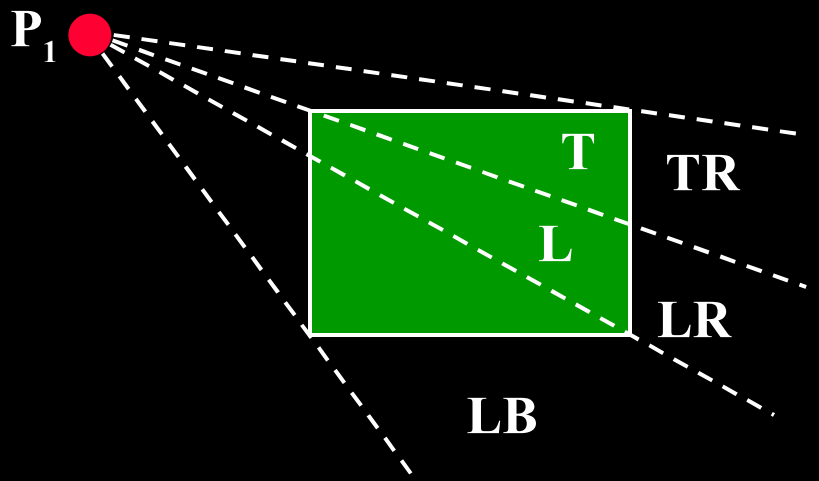
P_1 is inside the clip window and P_2 is outside

P_1 is directly to the left of the clip window

Nicholl-Lee-Nicholl Line Clipping: Passo 1b

Cálculo dos ângulos de uma linha hipotética de P1 aos cantos.
Determinar regiões L,T,R,L,TR,etc





Nicholl-Lee-Nicholl Line Clipping

- Para determinar em qual região P_2 está:
  - Compare-se a inclinação da reta com as inclinações das retas hipotéticas calculadas.
    - Exemplo: P_1 está à esquerda, P_2 está em LT.

$$\text{inclinação } \overline{P_1P_{TR}} < \text{inclinação } \overline{P_1P_2} < \text{inclinação } \overline{P_1P_{TL}}$$

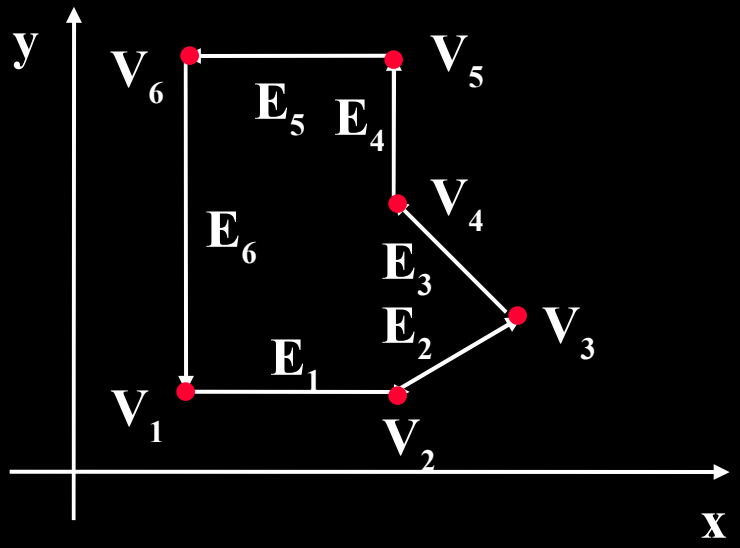
$$\frac{y_T - y_1}{x_R - x_1} < \frac{y_2 - y_1}{x_2 - x_1} < \frac{y_T - y_1}{x_L - x_1}$$

Clip Windows Não Retangulares

- Algoritmos baseados em equações parametricas podem ser extendidos a Polígonos Convexos
  - Método de Liang-Barsky
  - Modifique o algoritmo para incluir as equações paramétricas de todas as bordas definidas pelo polígono de clipping.
- Métodos de Clipping de Regiões Côncavas
  - Divida em um conjunto de Polígonos Convexos
  - Aplique métodos paramétricos
- Círculos e outras regiões de clipagem curvas
  - Linhas podem ser clipadas através do retângulo de-limítrofe.

Divisão de Polígonos Côncavos

- Identifique se um polígono é Côncavo
  - Calcule o produto cruzado dos vetores de borda.

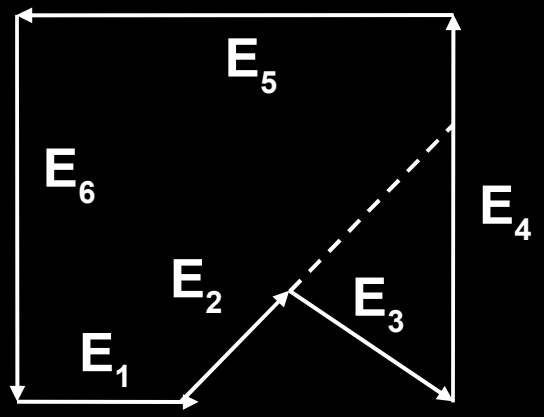


$$(E_1 \times E_2)_z > 0$$
$$(E_2 \times E_3)_z > 0$$
$$(E_3 \times E_4)_z < 0$$
$$(E_4 \times E_5)_z > 0$$
$$(E_5 \times E_6)_z > 0$$
$$(E_6 \times E_1)_z > 0$$

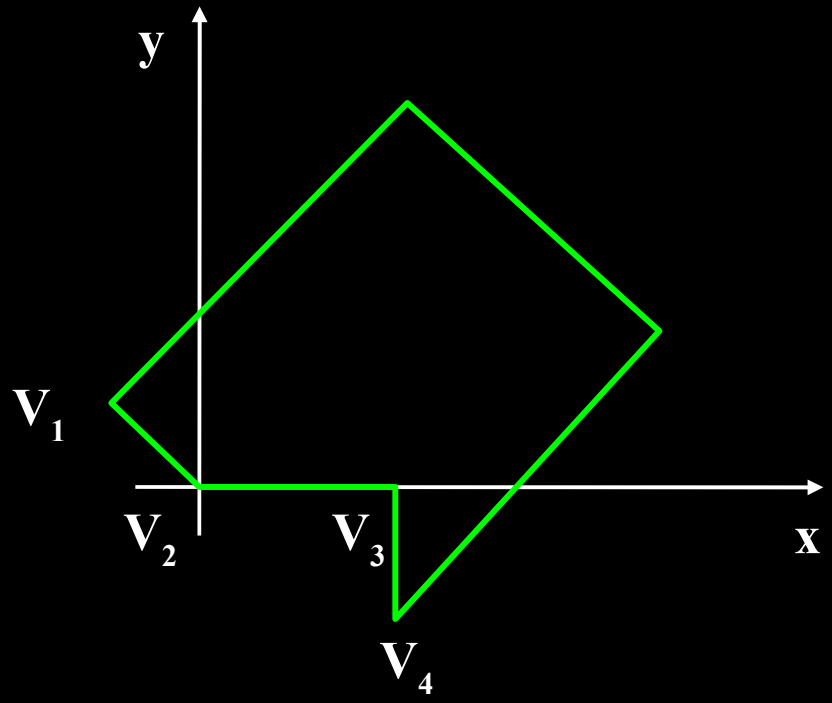
  - Uma componente z negativa resultante da multiplicação posicionada entre componentes positivas indica uma concavidade local.

Divisão de Polígonos Côncavos: Método Vetorial

- Calcule o produto cruzado dos vetores de borda em sentido anti-horário.
- Se alguma componente Z for negativa:
  - É côncavo.
  - Divida-o ao longo da primeira das duas linhas do produto cruzado:



Divisão de Polígonos Côncavos: Método da Rotação

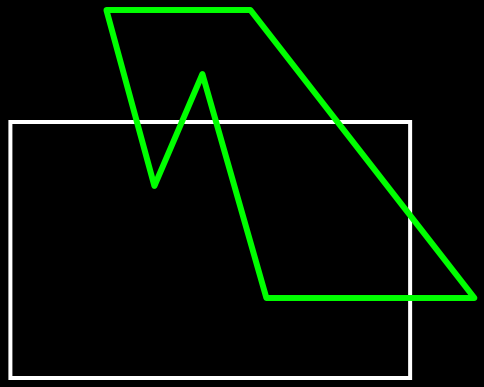


After rotating V_3 onto the x axis, V_4 is below the x axis.

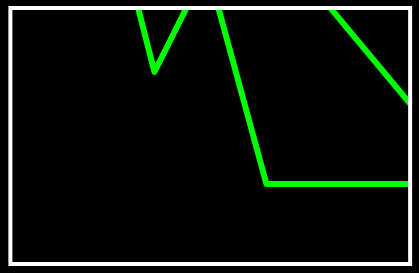
Split the polygon along the line of $\overline{V_2V_3}$.

Clipping de Polígonos

Clipping de Linhas



Antes do clipping

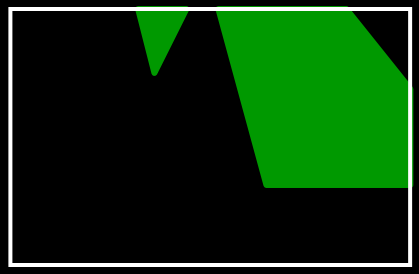


Depois do clipping

Resultam 2 polígonos distintos:

Clipping de Polígonos

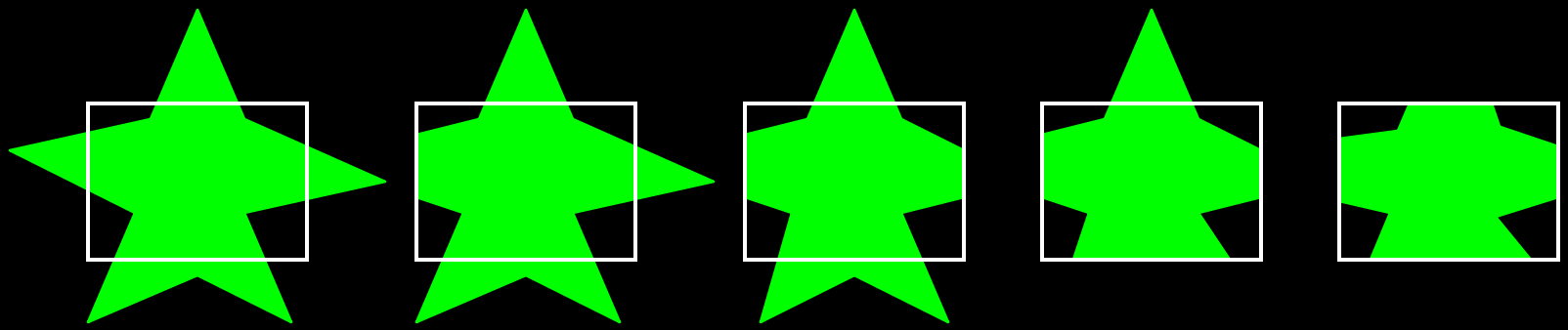
Antes do clipping



Depois do clipping

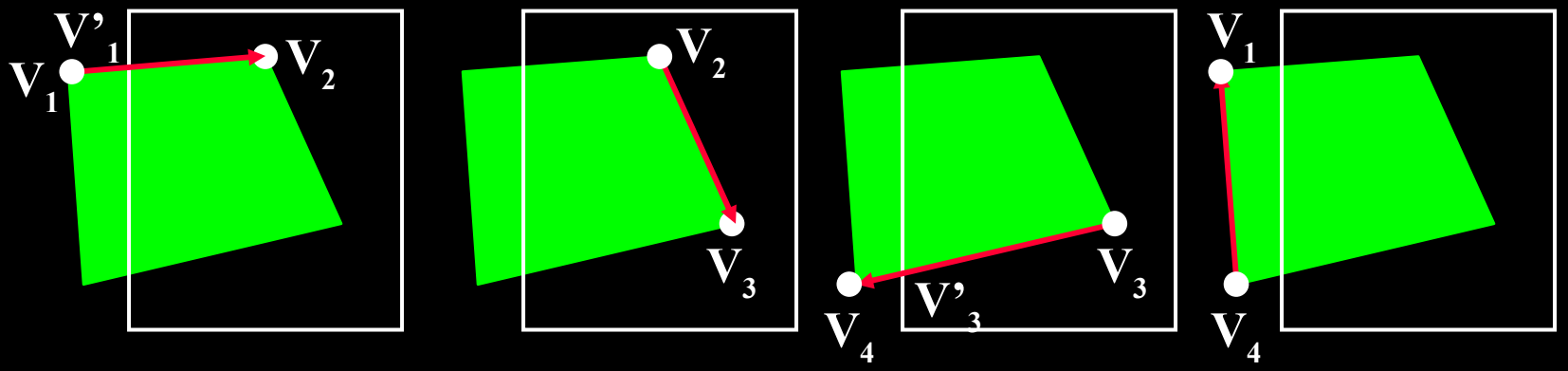
Clipping de Polígonos de Sutherland-Hodgeman

- Processa as bordas do polígono como um todo contra cada aresta do window
  - Todos os vértices do polígono são processados contra cada uma das arestas do window



Clipping de Polígonos de Sutherland-Hodgeman

- Passe cada par de vértices adjacentes do polígono a um clipador de linhas qualquer, criando novos pontos em um novo polígono
  - 4 casos:



out ⟶ in	in ⟶ in	in ⟶ out	out ⟶ out
salve V'_1, V_2	salve V_3	salve V'_3	salve nada

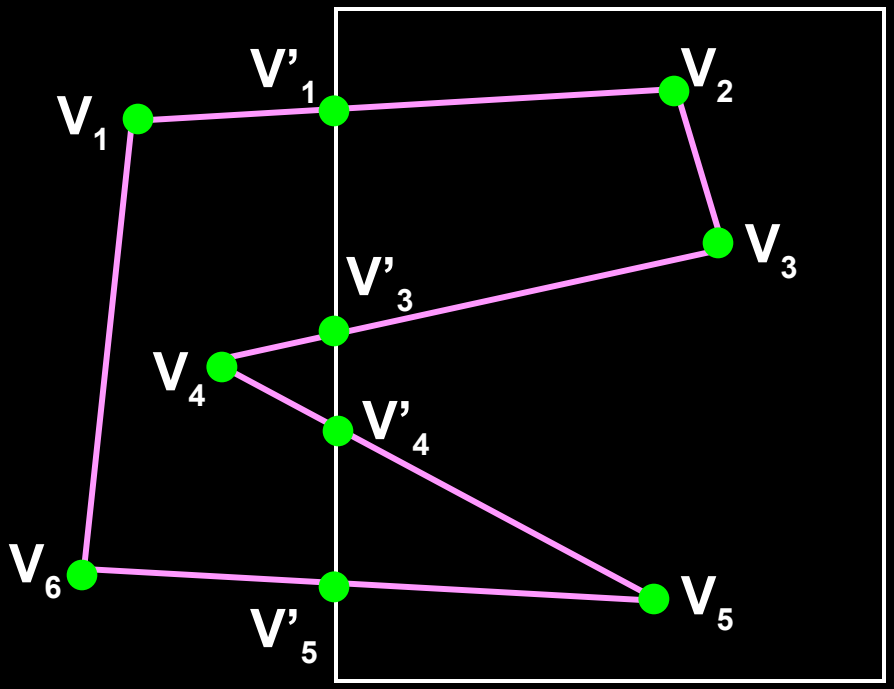
Clipping de Polígonos de Sutherland-Hodgeman

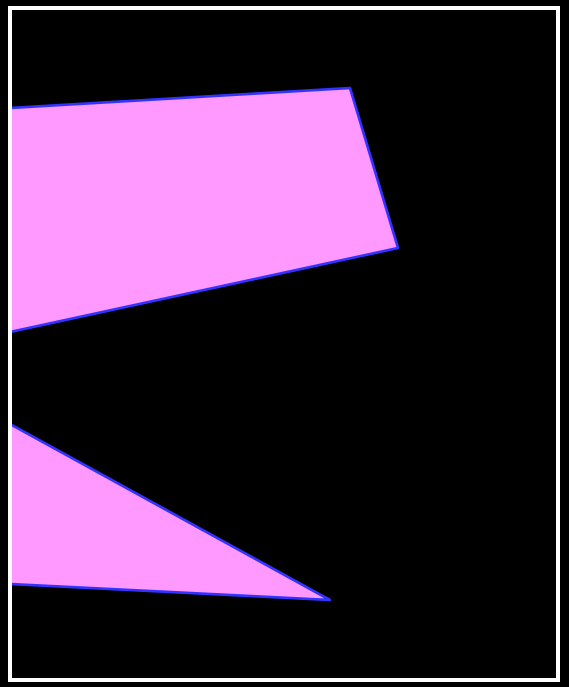
- Lista de vértices intermediários
  - Cada vez que realizamos a clipagem para todos os vértices contra uma das 4 bordas do window, regeneramos o polígono.
  - Este novo polígono é clipado contra outra das 4 bordas do window.
- Polígonos convexos são tratados corretamente.
- Caso seja côncavo:
  - Divida em subpolígonos côncavos

Clipping de Polígonos de Weiler-Atherton

- Desenvolvido para identificação de superfícies visíveis
  - Pode ser aplicado a uma região de clipagem arbitrária.
  - Pode ser usado para seguir as bordas de qualquer coisa com qualquer formato.
- Se processamos em sentido horário procedemos assim:
  - For an outside-to-inside pair of vertices, follow the polygon boundary.
  - For an inside-to-outside pair of vertices, follow the window boundary in clockwise direction.

Clipagem de Polígonos de Weiler-Atherton





Depois do Clipping

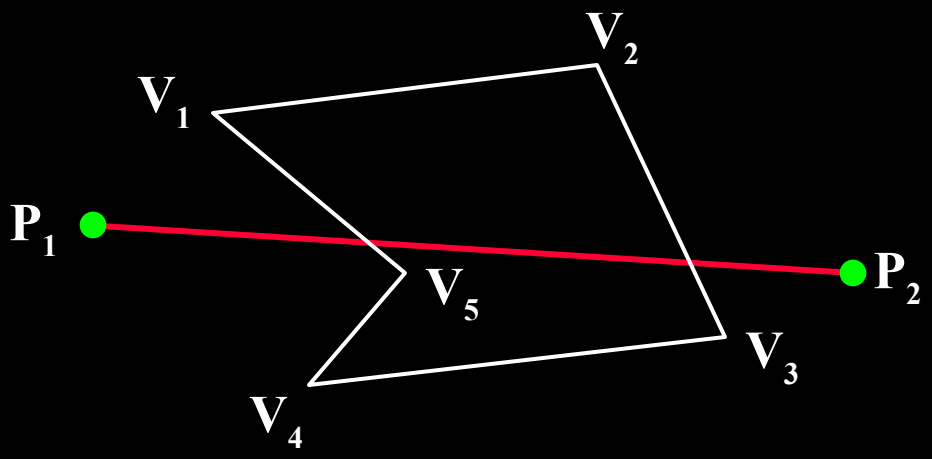
Outros Clippings

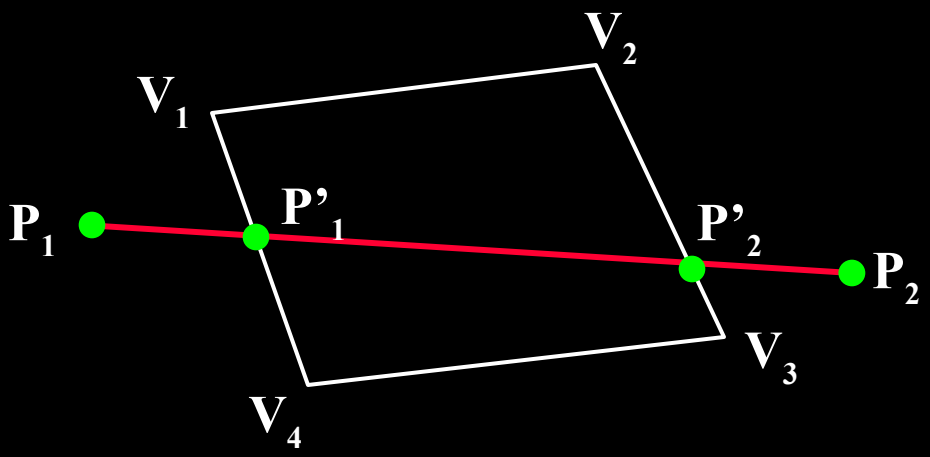
- Clipping de Curvas
  - Use um casco convexo (retângulo) para testar.
  - Para o círculo
    - Use as coordenadas de quadrantes individuais (gere)
    - então use octantes (gere valores mais precisos)
- Clipping de Texto
  - All-or-none string-clipping: Pegar limites do casco convexo de um string.
  - All-or-none character-clipping: Clipar carateres individuais pela regra de pertinência do ponto.

Clipping Exterior

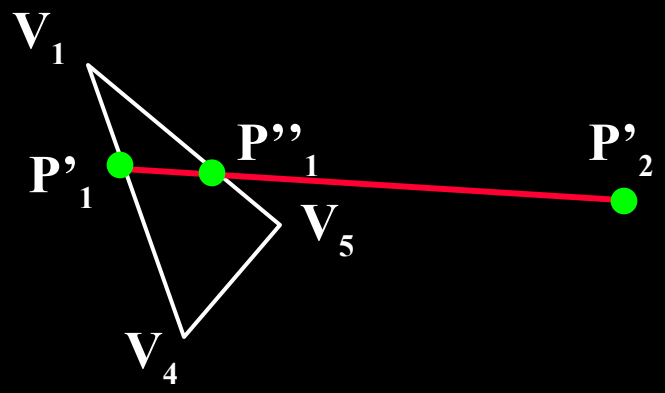
- Salvar uma região externa
- Applicações
  - Sistemas de windows múltiplos
  - Layouts de páginas para design (Corel, Photoshop,...)
- Utilizamos os procedimanetos de clipagem usados para o interior de polígonos côncavos

Clipping Exterior





Interior Clip



Exterior Clip

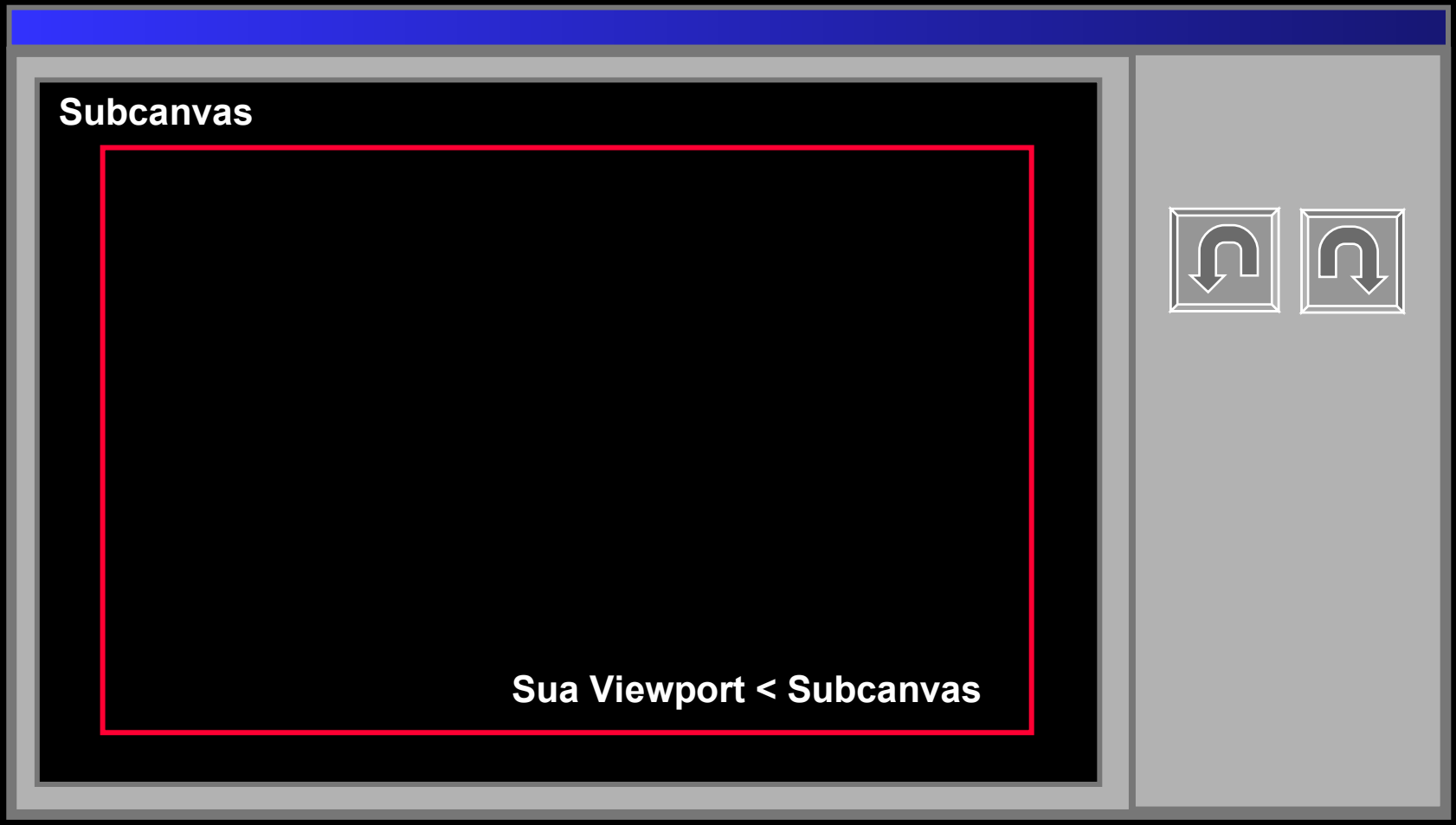
Trabalho: Clipping - parte #1

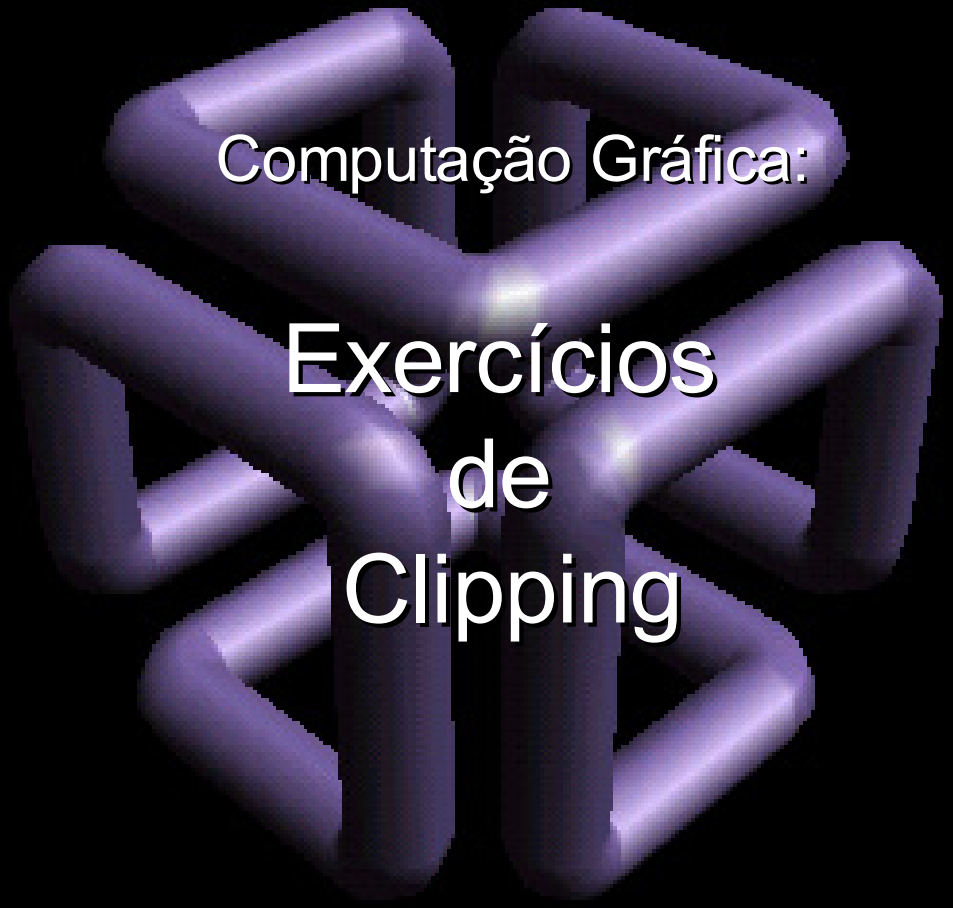
- Extenda seu sitema gráfico interativo para realizar navegação livre com a window
  - Extenda o seu display file para comportar as coordenadas dos objetos em PPC paralelamente às coordenadas WC
  - Faça o mesmo com a window
  - Implemente os algoritmos descritos em 4.1
  - Inclua botões de navegação adicionais que permitam a rotação da window
    - Um clique no botão faz a window rodar para a direita ou esquerda de um ângulo fixo.

Trabalho: Clipping - parte #2

- Extenda seu sitema gráfico interativo para realizar a clipagem de seus elementos gráficos
  - Defina uma subárea dentro de seu Subcanvas ou View que você utiliza para desenhar como sendo a área de clipagem: uns 10 píxeis para dentro nos quatro cantos.
  - Defina seu viewport como sendo essa área interna, que não vai mais começar em (0,0), mas sim em (10,10).
  - Dessa forma você vai enxergar se o algo for desenhado fora da window/viewport porque o algoritmo de clipagem nativo não vai cortá-lo nas bordas que você definiu.

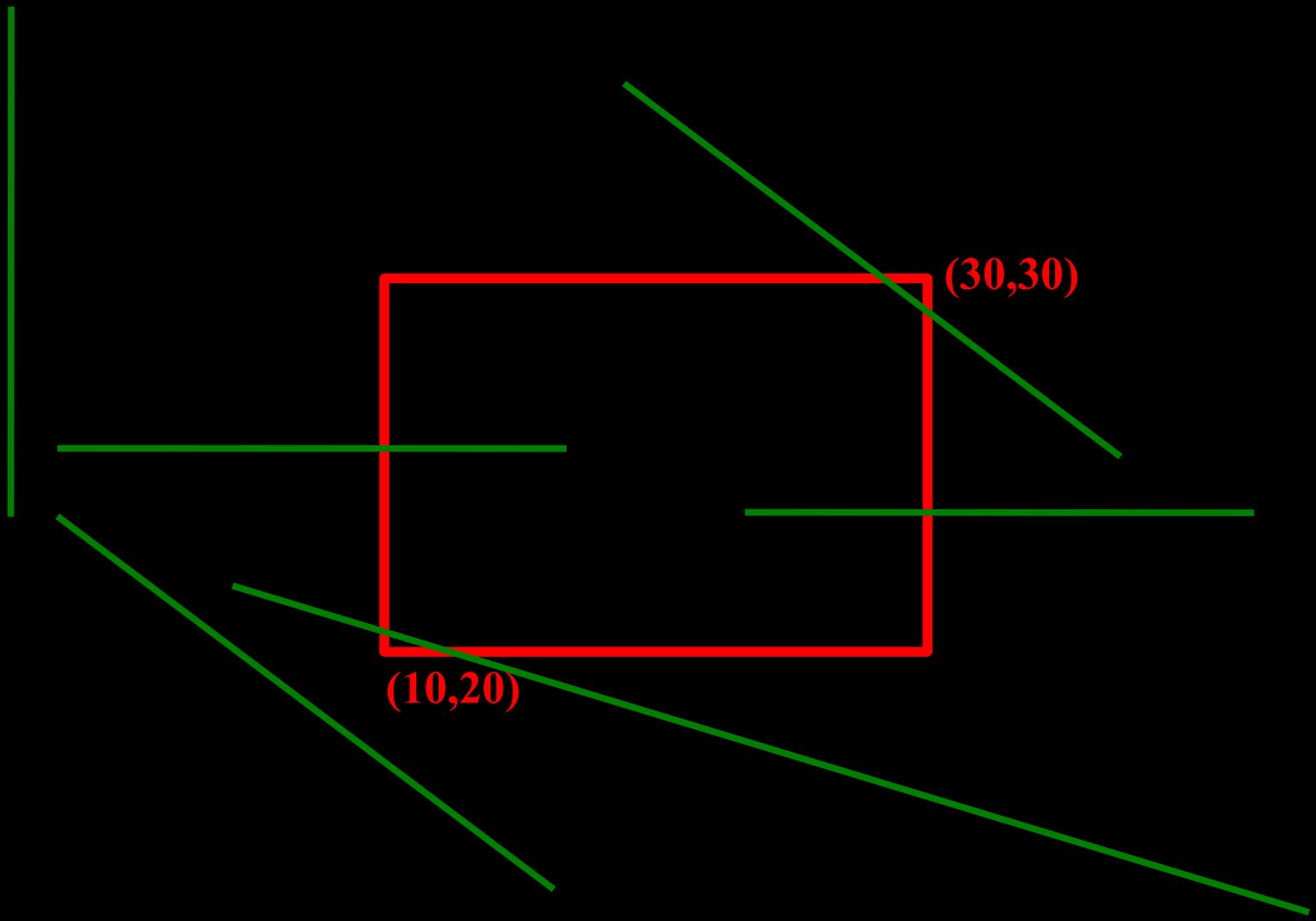
Trabalho: Clipping



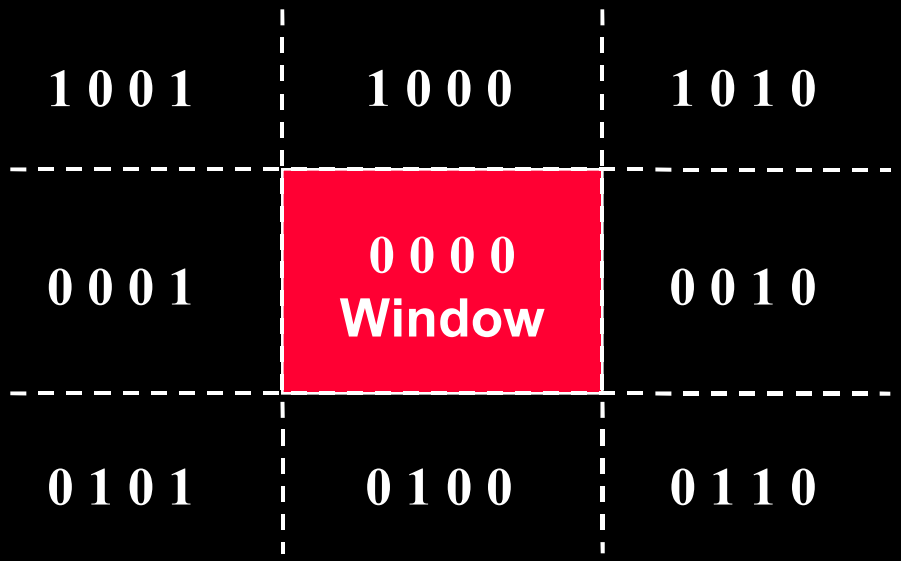


Computação Gráfica:

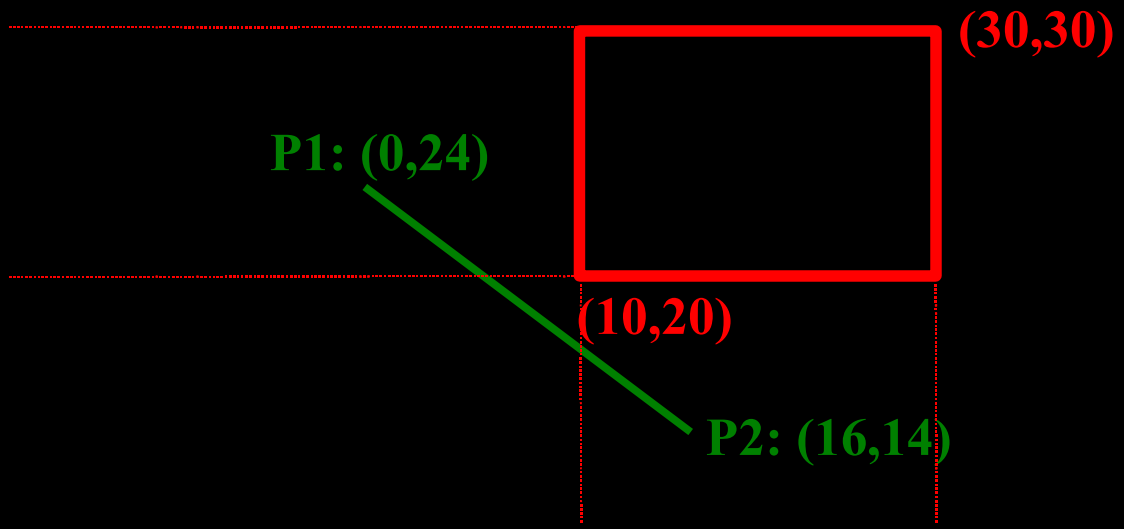
Exercícios de Clipping


(30,30)
(10,20)

Parte 1: Cohen - Sutherland



RC[1]: acima
RC[2]: abaixo
RC[3]: direita
RC[4]: esquerda



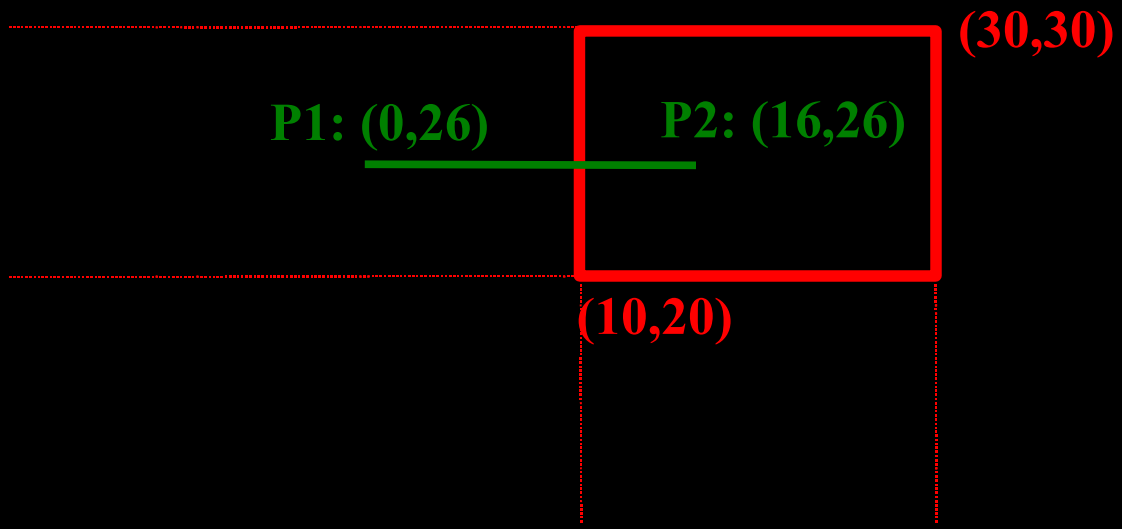
- RC: [0001, 0100] -> Esquerda, Abaixo
- Verificar se ocorrem intersecções!

- Coeficiente Angular:

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \frac{(14 - 24)}{(16 - 0)} = -0.625$$

- Calculando para a Esquerda:

$$y = m.(x_e - x_1) + y_1$$
$$y = -0.625.(10 - 0) + 24$$
$$y = 17.75$$
$$17.75 < y_{min}$$ ← fora da window!



- RC: [0001, 0000] -> Esquerda, Centro
- Ocorre intersecção
  - Aonde?

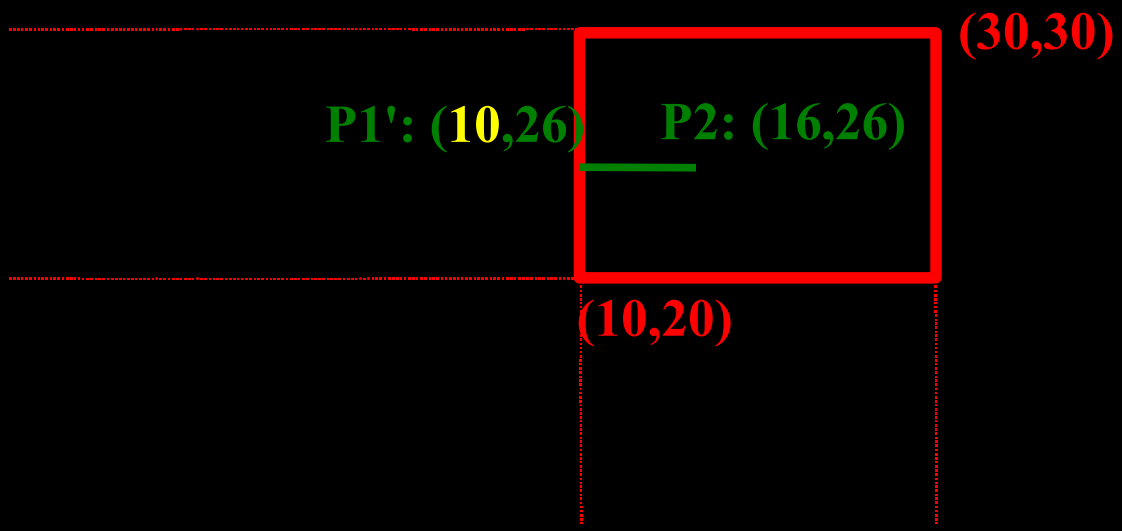
- Coeficiente Angular:

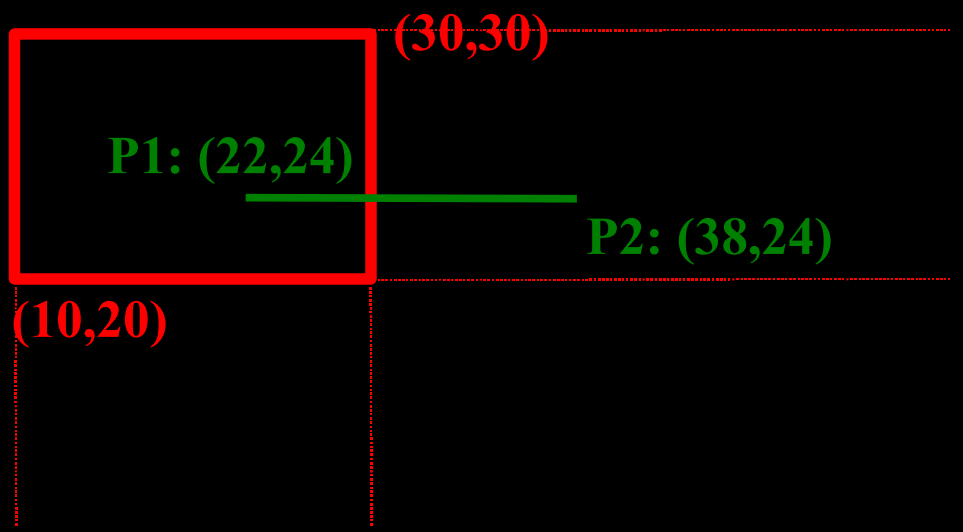
$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \frac{(26 - 26)}{(16 - 0)} = 0$$

- Calculando para a Esquerda:

$$y = m.(x_e - x_1) + y_1$$
$$y = 0.(10 - 0) + 26$$
$$y = 26$$
$$y_{min} < 26 < y_{max} \leftarrow \text{OK!}$$

- Nova reta:





- RC: [0000, 0010] -> Centro, Direita
- Ocorre intersecção
  - Aonde?

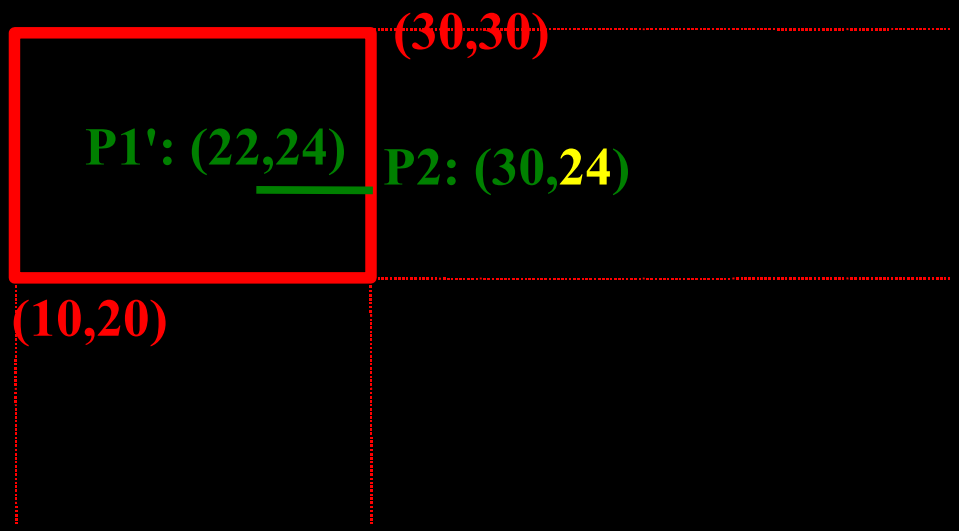
- Coeficiente Angular:

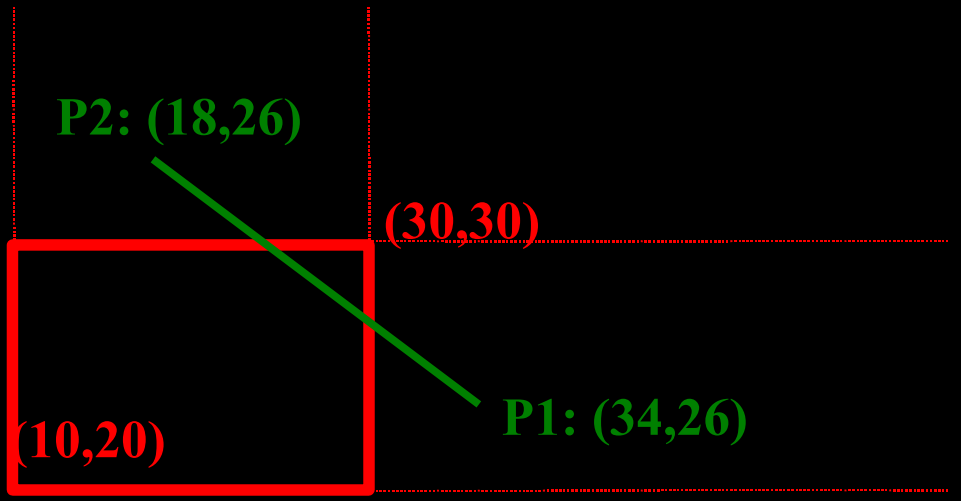
$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \frac{(24 - 24)}{(38 - 22)} = 0$$

- Calculando para a Esquerda:

$$y = m.(x_d - x_1) + y_1$$
$$y = 0.(30 - 8) + 24$$
$$y = 24$$
$$y_{min} < 24 < y_{max} \leftarrow \text{OK!}$$

- Nova reta:





- RC: [0010, 1000] -> Direita, Topo
- Verificar se ocorrem intersecções!

- Coeficiente Angular:

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \frac{(36 - 26)}{(18 - 34)} = -0.625$$

- Calculando para a Direita:

$$y = m.(x_d - x_1) + y_1$$
$$y = -0.625.(30 - 34) + 26$$
$$y = 28.5$$
$$y_{min} < 28.5 < y_{max}$$ ← OK!

- Calculando para o Topo:

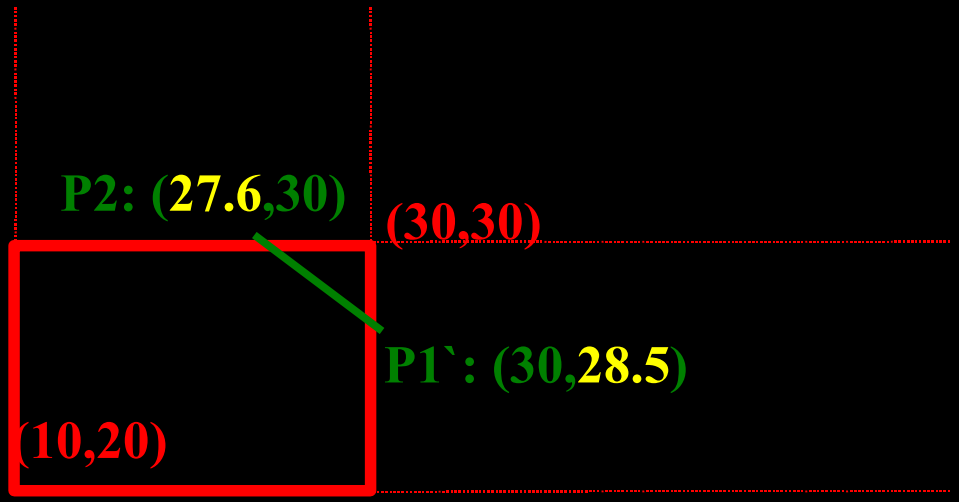
$$x = x_1 + \frac{1}{m} . (y_t - y_1)$$

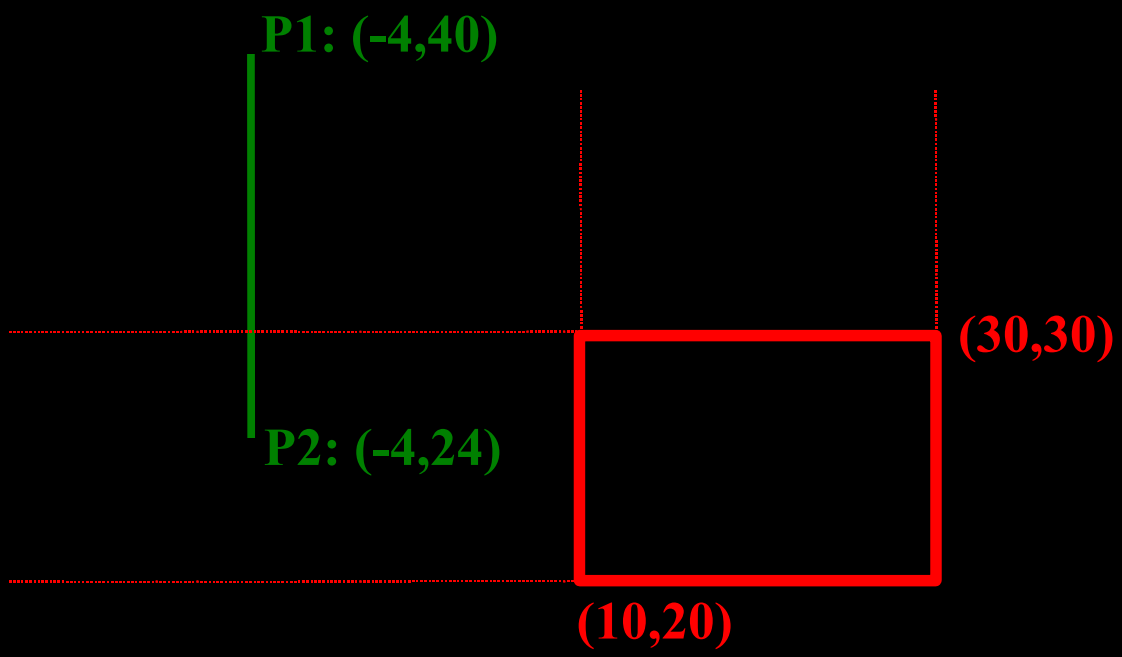
$$x = 34 - \frac{1}{0.625} . (30.26)$$

$$x = 27.6$$

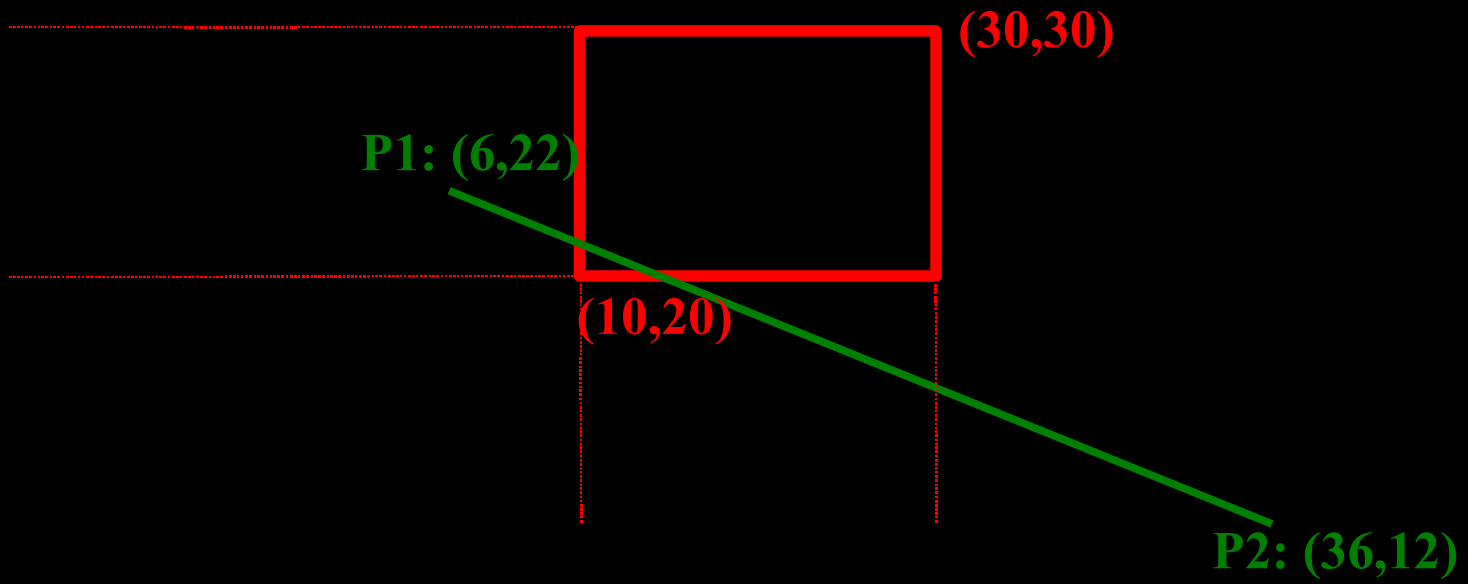
$$x_{min} < 27.6 < x_{max} \quad \leftarrow \quad \text{OK!}$$

- Nova reta:





- RC: [1001, 0001] -> “E” Lógico dá 0001
- Totalmente fora.



- RC: [0001, 0110] -> Esquerda, Abaixo-Direita
- Verificar se ocorrem intersecções!

- Coeficiente Angular:

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \frac{(12 - 22)}{(36 - 6)} = -0.3334$$

- Calculando para a Esquerda:

$$y = m.(x_e - x_1) + y_1$$
$$y = -0.3334.(10 - 6) + 22$$
$$y = 20.6$$
$$y_{min} < 20.6 < y_{max}$$ ← OK!

- Calculando para a Direita:

$$y = m.(x_d - x_1) + y_1$$
$$y = -0.3334.(30 - 6) + 22$$
$$y = 14$$
$$14 < y_{min}$$ ← fora da Window!

- Finalmente, calculando para o Fundo:

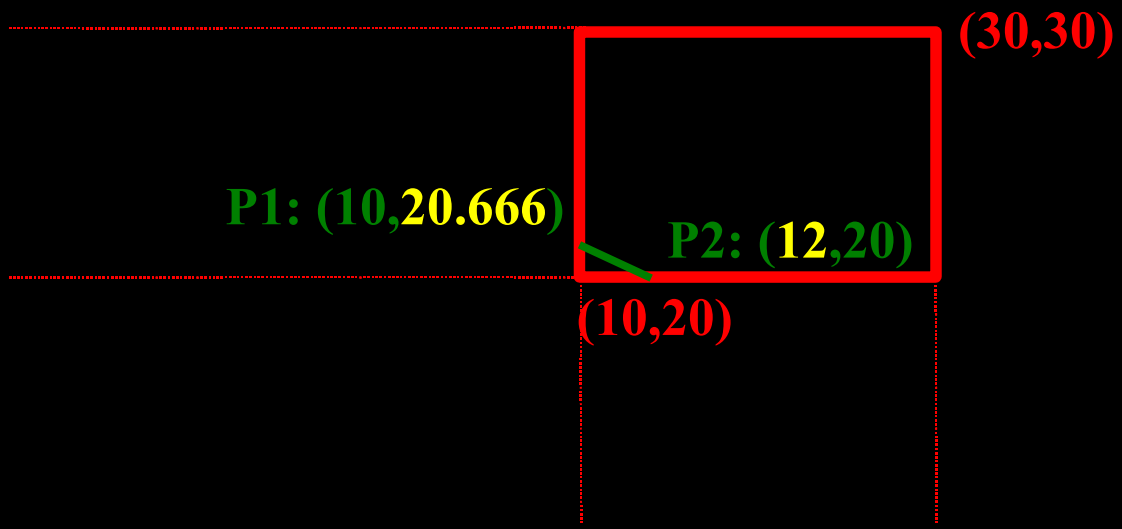
$$x = x_1 + \frac{1}{m} . (y_f - y_1)$$

$$x = 6 - \frac{1}{0.3334} . (20 - 22)$$

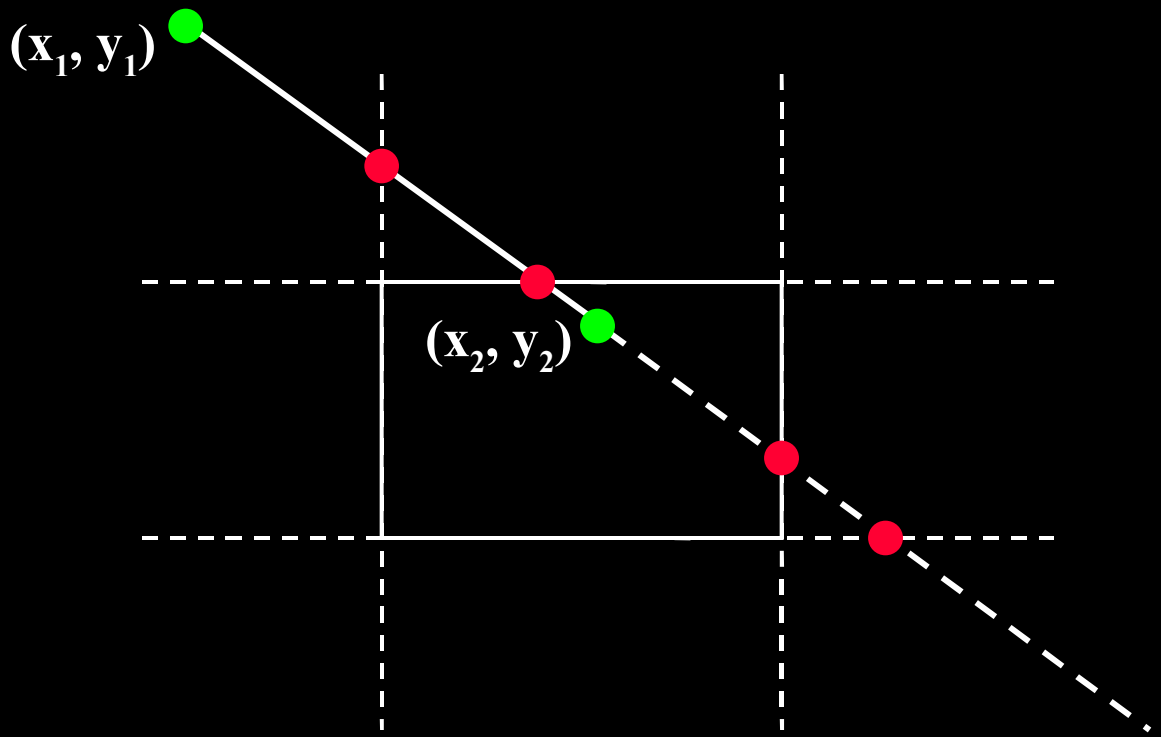
$$x = 12$$

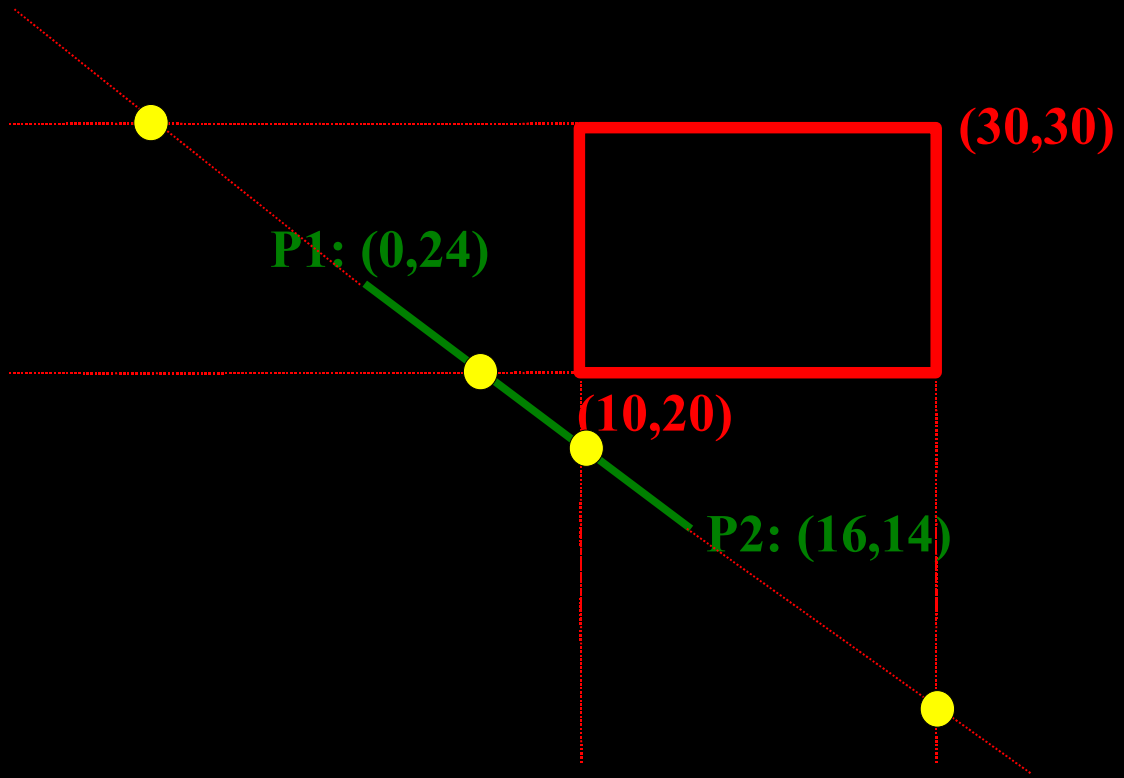
$$x_{min} < 12 < x_{max} \quad \leftarrow \quad \text{OK!}$$

- Nova reta:



Parte 2: Liang-Barsky




(30,30)
P1: (0,24)
(10,20)
P2: (16,14)

- p1, p2... q1, q2...

$$p_1 = -\Delta x = -16$$
$$p_2 = \Delta x = 16$$
$$p_3 = -\Delta y = -(14 - 24) = 10$$
$$p_4 = \Delta y = -10$$
$$q_1 = x_1 - x_{min} = 0 - 10 = -10$$
$$q_2 = x_{max} - x_1 = 30 - 0 = 30$$
$$q_3 = y_1 - y_{min} = 24 - 20 = 4$$
$$q_4 = y_{max} - y_1 = 30 - 24 = 6$$

- u1 – valores menores do que zero – p1 e p4

$$r_1 = \frac{q_1}{p_1} = \frac{-10}{-16} = 0.625$$

$$r_4 = \frac{q_4}{p_4} = \frac{6}{-10} = -0.6$$

$$u_1 = max(0, r_{1,} r_4) = 0.625$$

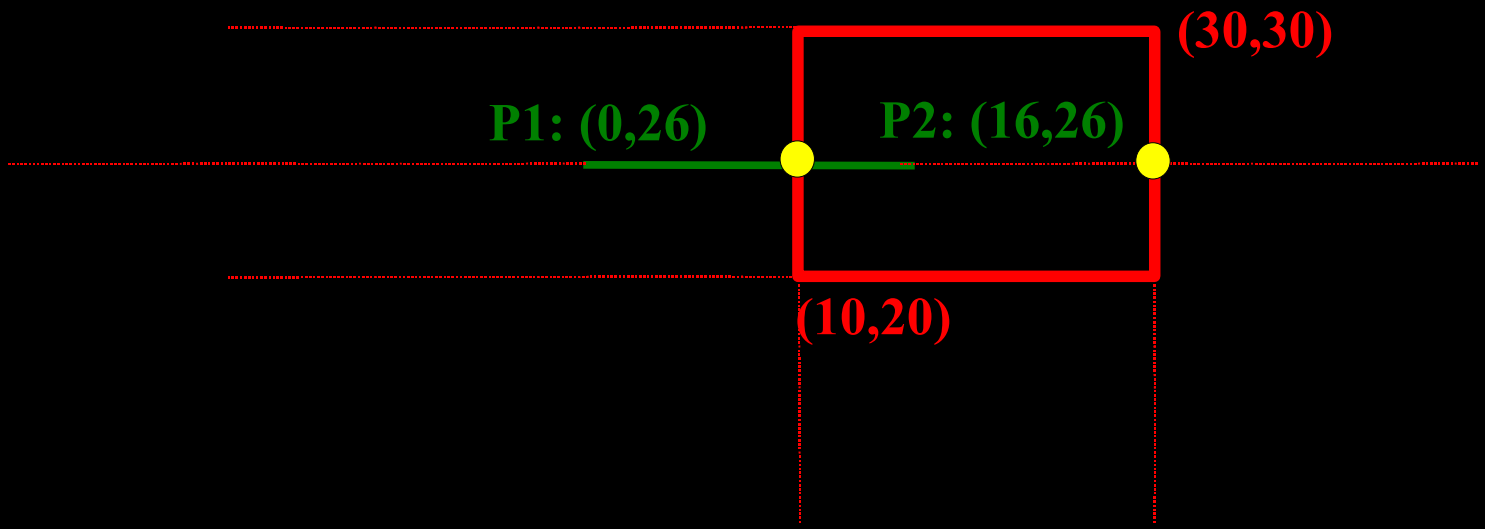
- u2 – valores menores do que zero – p2 e p3

$$r_2 = \frac{q_2}{p_2} = \frac{30}{16} = 1{,}875$$

$$r_4 = \frac{q_4}{p_4} = \frac{4}{10} = 0{,}4$$

$$u_2 = min(1, r_{2,} r_3) = 0{,}4$$

- Como u1 > u2, a reta está fora!


(30,30)
P1: (0,26)
P2: (16,26)
(10,20)

- p1, p2... q1, q2...

$$p_1 = -\Delta x = -16$$
$$p_2 = \Delta x = 16$$
$$p_3 = -\Delta y = 0$$
$$p_4 = \Delta y = 0$$
$$q_1 = x_1 - x_{min} = 0 - 10 = -10$$
$$q_2 = x_{max} - x_1 = 30 - 0 = 30$$
$$q_3 = y_1 - y_{min} = 26 - 20 = 6$$
$$q_4 = y_{max} - y_1 = 30 - 26 = 4$$

- p3 e p4 valem 0!
  - Mas q3 e q4 >= 0
  - Dentro dos limites

- u1 – valores menores do que zero – p1

$$r_1 = \frac{q_1}{p_1} = \frac{-10}{-16} = 0{,}625$$

$$u_1 = max(0, r_1) = 0{,}625$$

- u2 – valores maiores do que zero – p2

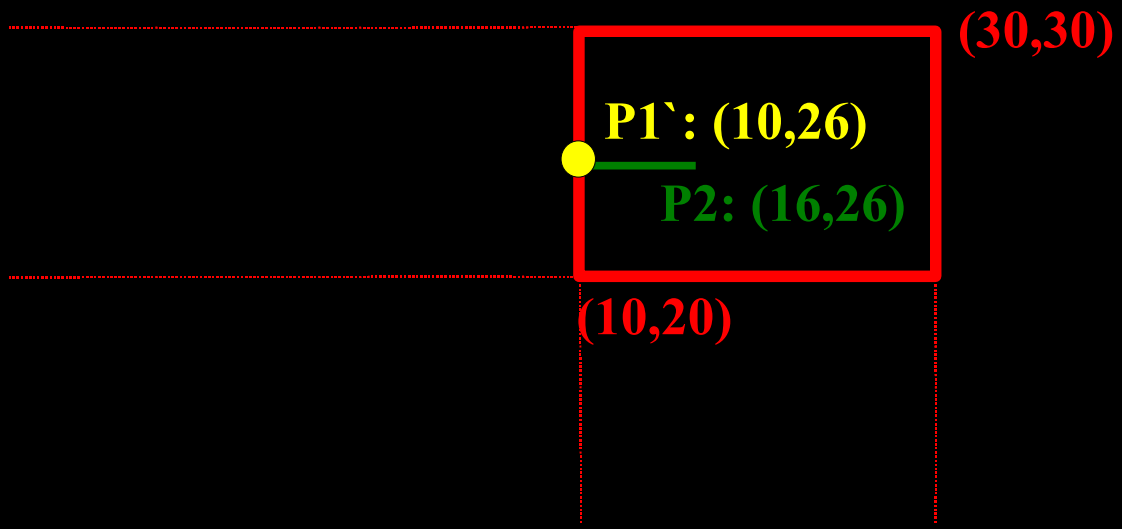
$$r_2 = \frac{q_2}{p_2} = \frac{30}{16} = 1{,}875$$

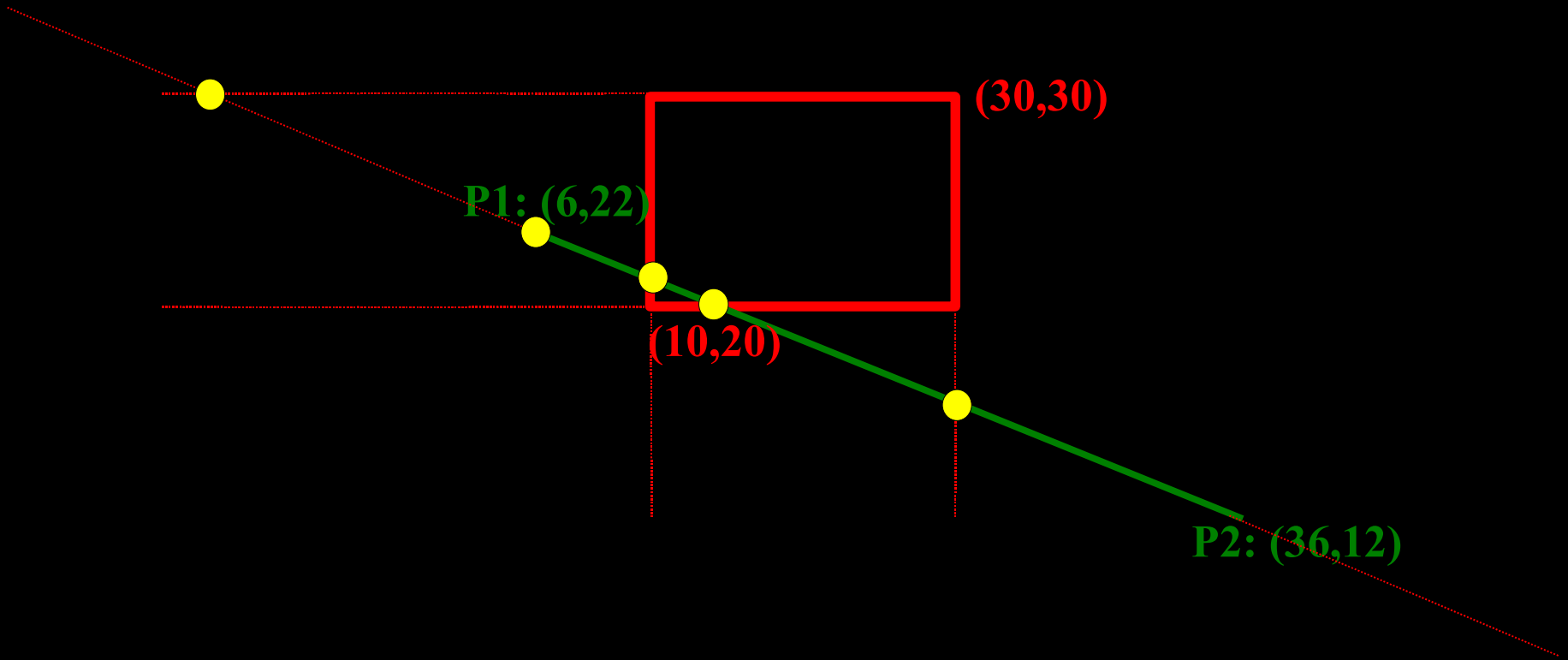
$$u_2 = min(1, r_2) = 1$$

- u2 é maior do que 1
  - Logo, de dentro para fora o segmento não toca a window.
- Colocando u1 na equação paramétrica:

$$x = x_1 + u\,\Delta x$$
$$x = 0 + 0{,}625.16$$
$$x = 10$$
$$y = y_1 + u\,\Delta y$$
$$y = 26 + 0{,}625.0$$
$$y = 26$$

- Novo ponto, de fora para dentro: (10, 26)


(30,30)
P1`: (10,26)
P2: (16,26)
(10,20)


(30,30)
P1: (6,22)
(10,20)
P2: (36,12)

- p1, p2... q1, q2...

$$p_1 = -\Delta x = -(36-6) = -30$$
$$p_2 = \Delta x = 30$$
$$p_3 = -\Delta y = -(12-22) = 10$$
$$p_4 = \Delta y = -10$$
$$q_1 = x_1 - x_{min} = 6-10 = -4$$
$$q_2 = x_{max} - x_1 = 30-6 = 24$$
$$q_3 = y_1 - y_{min} = 22-20 = 2$$
$$q_4 = y_{max} - y_1 = 30-22 = 6$$

- u1 – valores menores do que zero – p1 e p4

$$r_1 = \frac{q_1}{p_1} = \frac{-4}{-30} = 0{,}1334$$

$$r_4 = \frac{q_4}{p_4} = \frac{6}{-10} = -0{,}6$$

$$u_1 = max(0, r_1, r_4) = 0{,}1334$$

- u2 – valores maiores do que zero – p2 e p3

$$r_2 = \frac{q_2}{p_2} = \frac{30}{16} = 1{,}875$$

$$r_3 = \frac{q_3}{p_3} = \frac{2}{10} = 0{,}2$$

$$u_2 = min(1, r_2, r_3) = 0{,}2$$

- Colocando u1 na equação paramétrica:

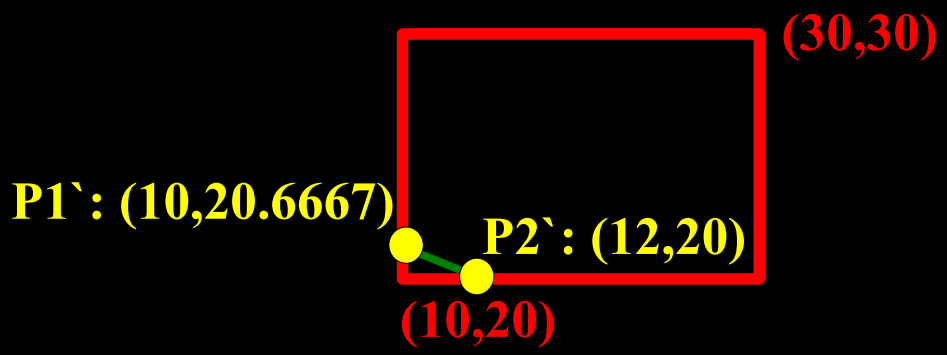
$$x = x_1 + u\,\Delta x$$
$$x = 6 + 0{,}1334.30$$
$$x = 10$$
$$y = y_1 + u\,\Delta y$$
$$y = 22 + 0{,}1334.(-10)$$
$$y = 20.6667$$

- Novo ponto, de fora para dentro: (10, 20.6667)

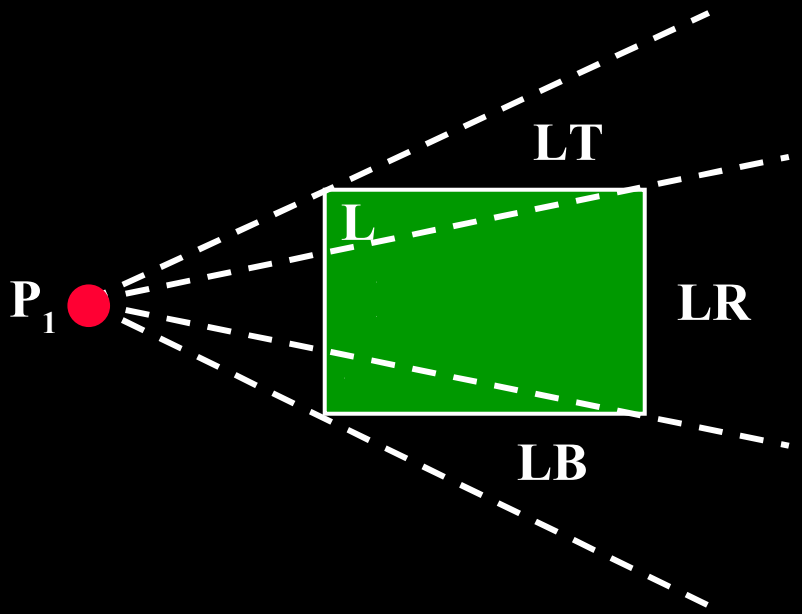
- Colocando u2 na equação paramétrica:

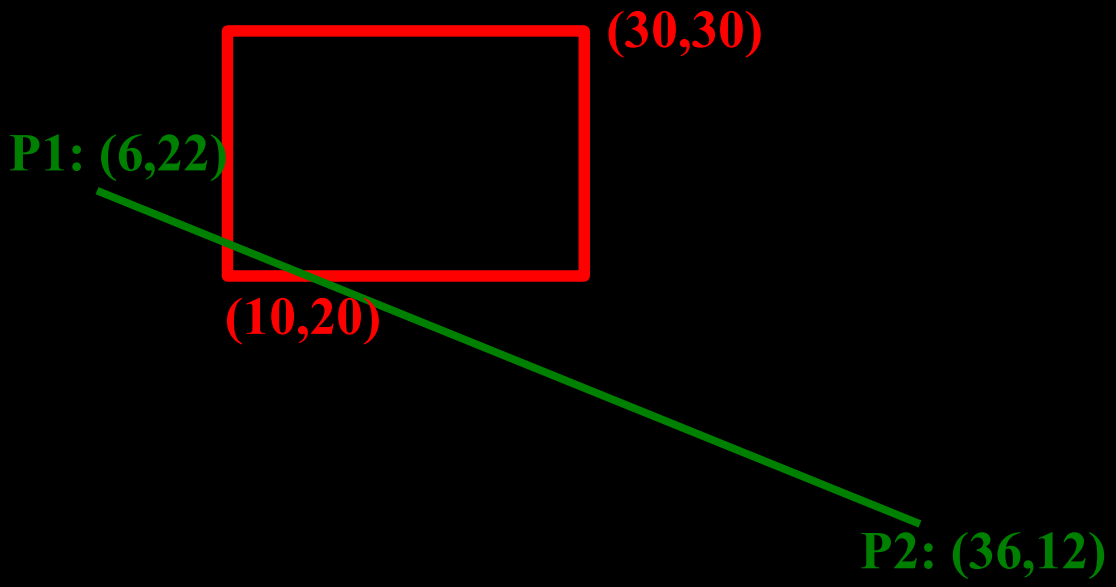
$$x = x_1 + u\,\Delta x$$
$$x = 6 + 0{,}2.30$$
$$x = 12$$
$$y = y_1 + u\,\Delta y$$
$$y = 22 + 0{,}2.(-10)$$
$$y = 20$$

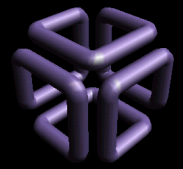
- Novo ponto, de dentro para fora: (12, 20)

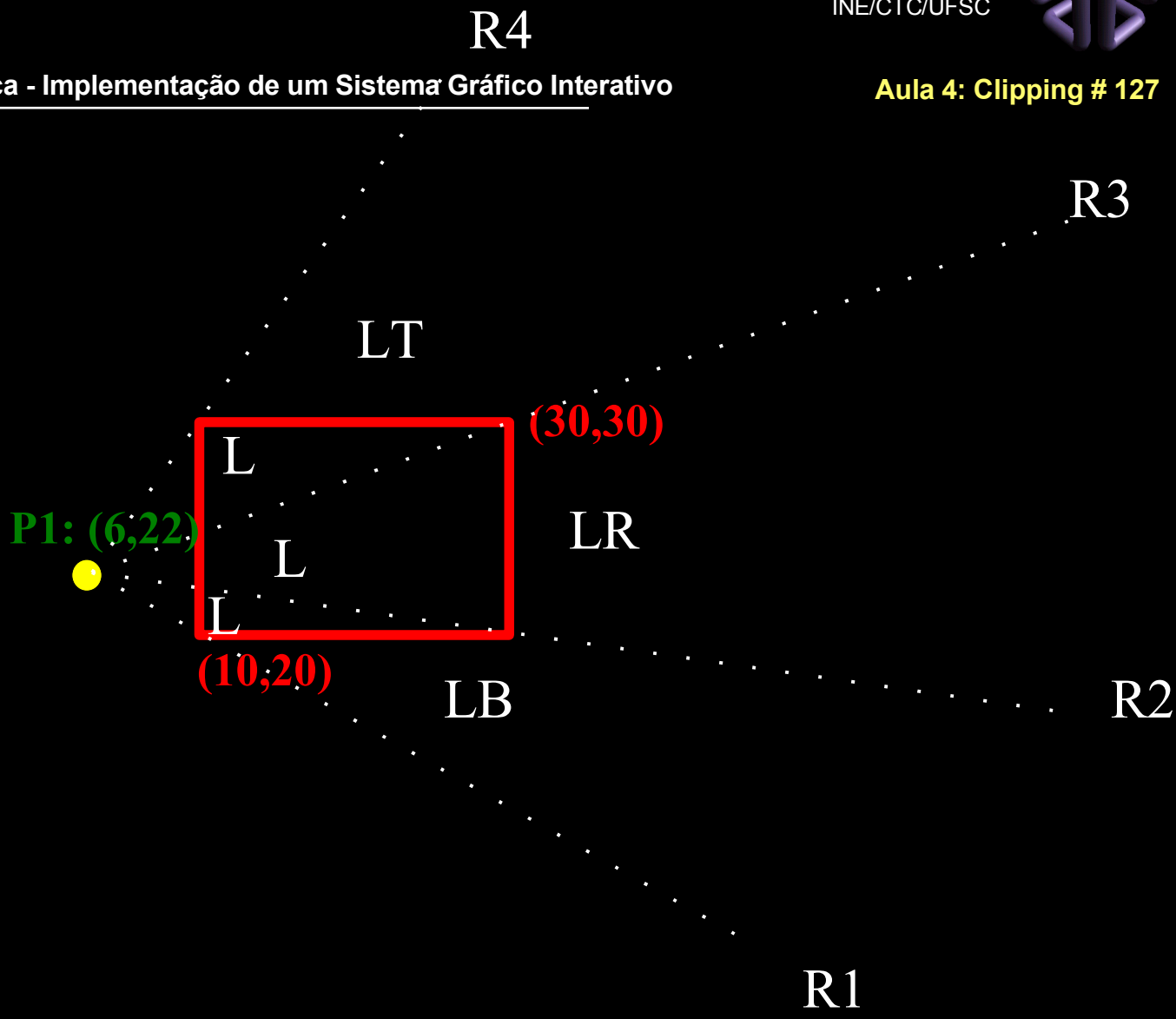

(30,30)
P1`: (10,20.6667)
P2`: (12,20)
(10,20)

Parte 3: Nicholl-Lee-Nicholl




(30,30)
P1: (6,22)
(10,20)
P2: (36,12)




R4
R3
LT
(30,30)
L
P1: (6,22)
LR
L
L
(10,20)
LB
R2
R1

- Coeficiente angular das novas retas:

$$m_1 = \frac{(20-22)}{(10-6)} = \frac{-2}{4} = -0{,}5$$

$$m_2 = \frac{(20-22)}{(30-6)} = \frac{-2}{24} = -0.0833$$

$$m_3 = \frac{(30-22)}{(30-6)} = \frac{8}{24} = 0.3334$$

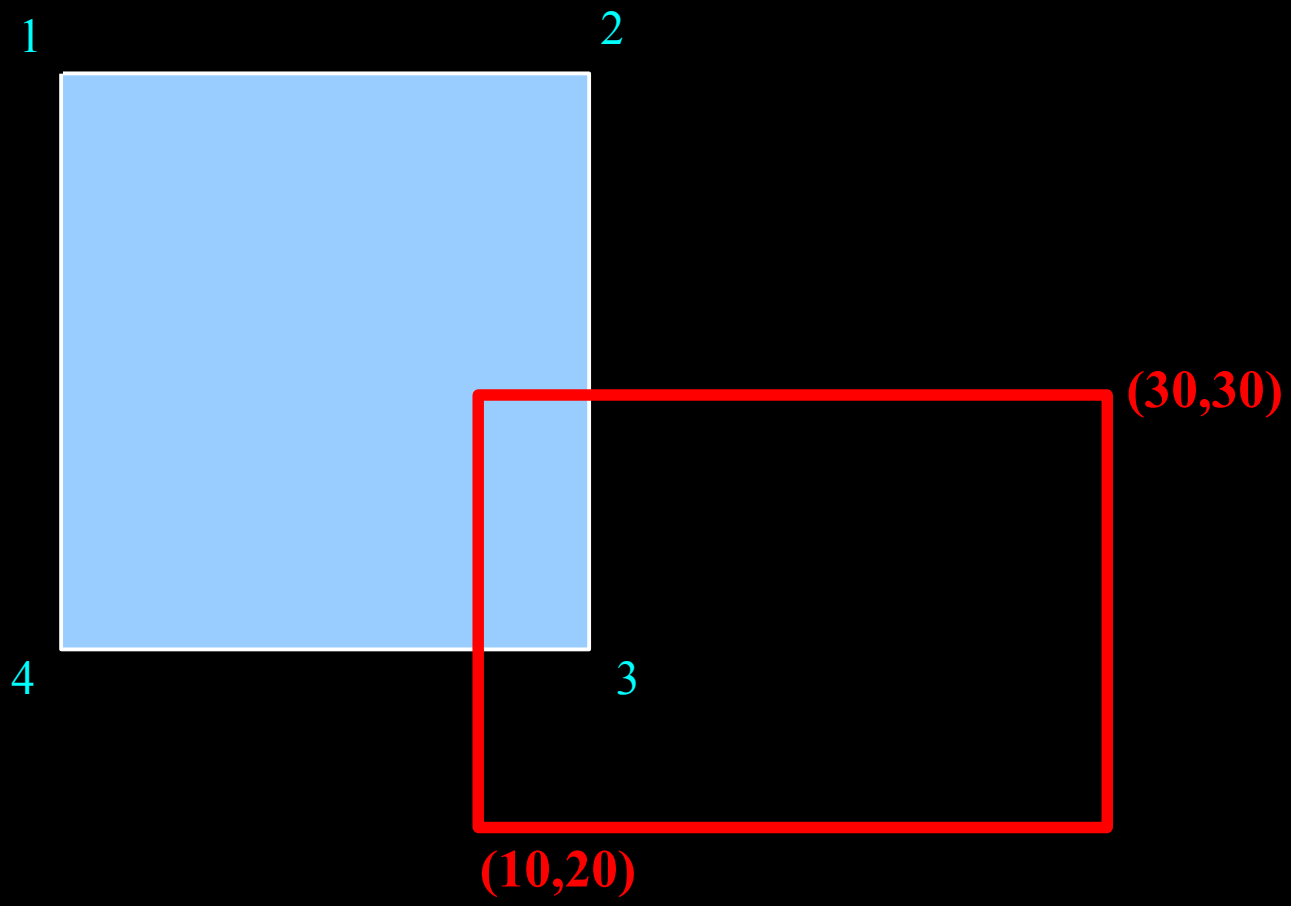
$$m_4 = \frac{(30-22)}{(10-6)} = \frac{8}{4} = 2$$

- Coeficiente angular da reta de teste:

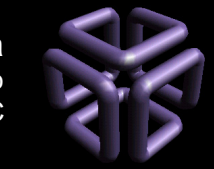
$$m_r = \frac{(12-22)}{(36-6)} = \frac{-10}{30} = -0.3334$$

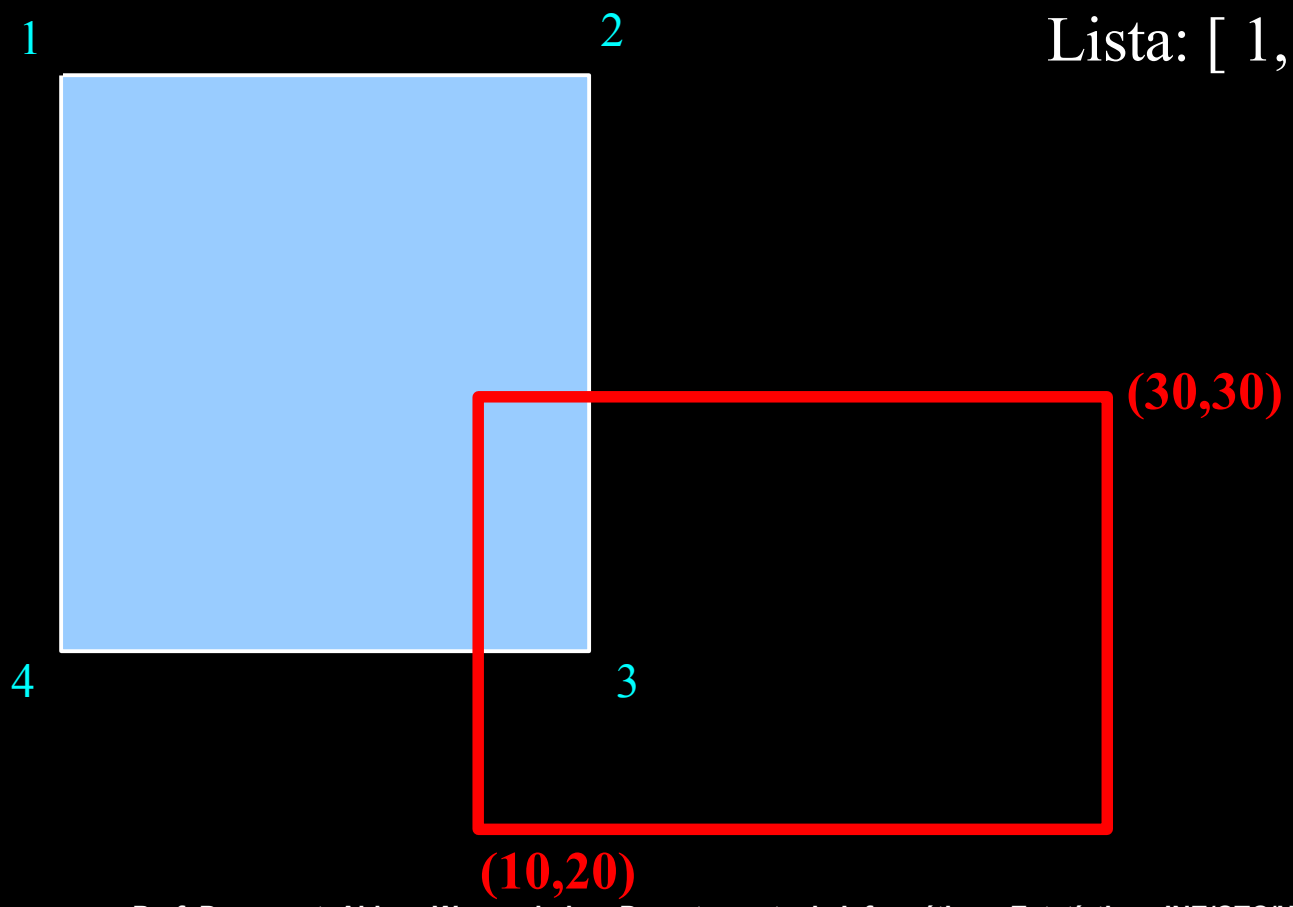
- Se encontra entre m1 e m2
  - Logo, basta testar a intersecção com os limites esquerdo e inferior da Window.
  - Se mr fosse < que m1 ou > m4, a reta estaria descartada automaticamente.

Clipping de Polígonos

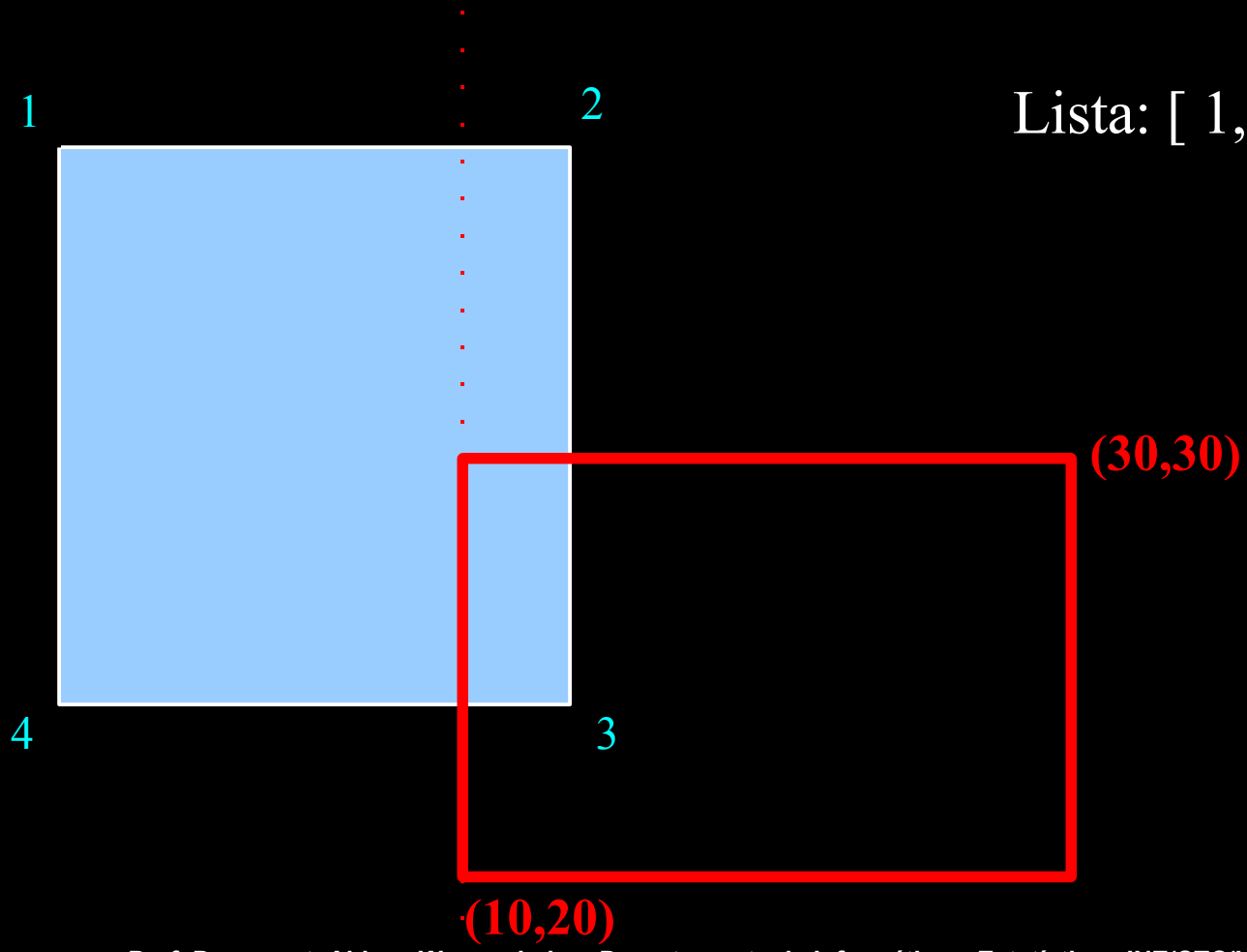


- Por Sutherland-Hodgeman:
- Clipe contra cada borda da Window
  - Percorra a lista de vértices adicionando ou removendo pontos conforme o caso.

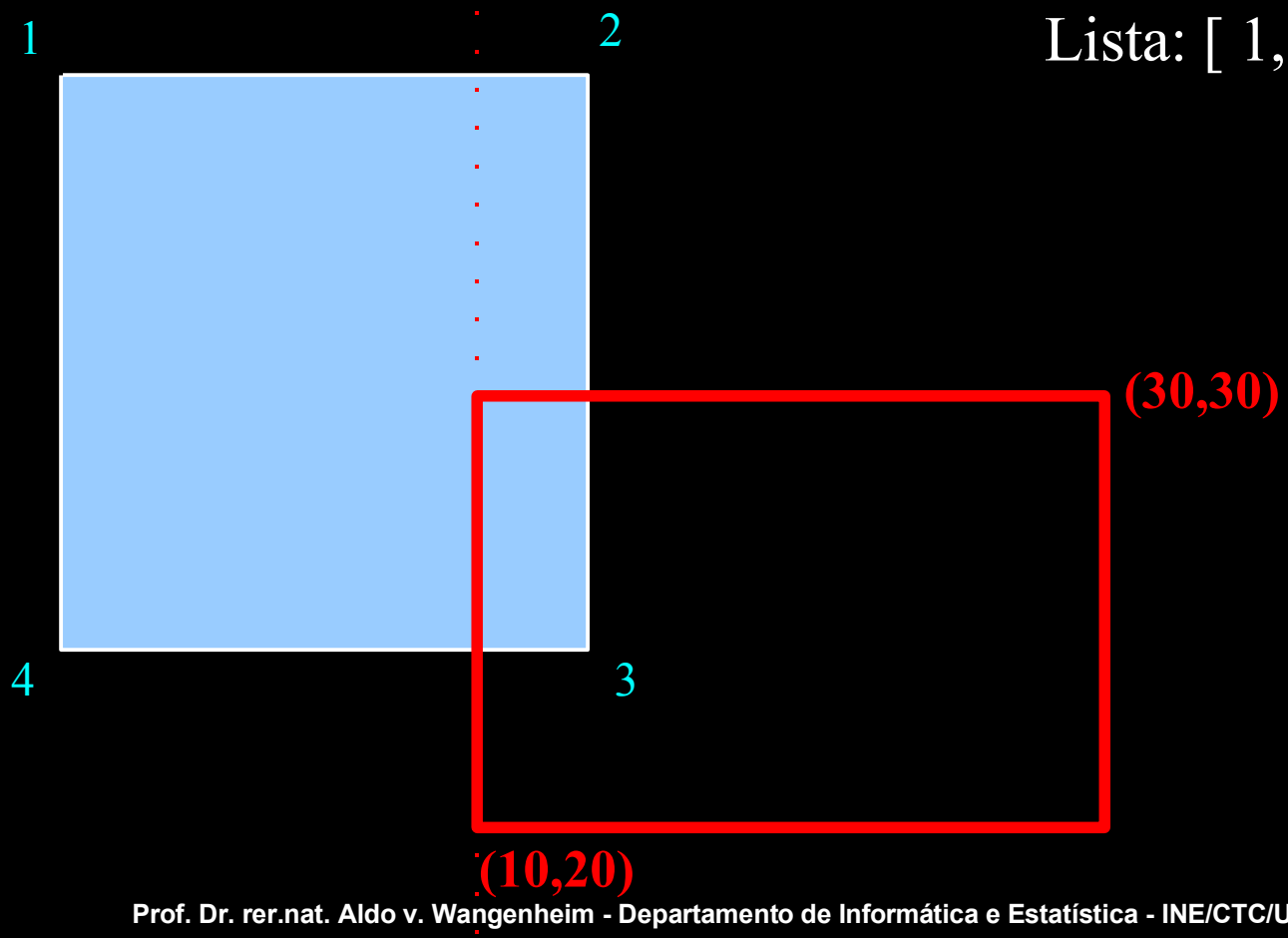



1
2
Lista: [1, 2, 3, 4]
(30,30)
4
3
(10,20)

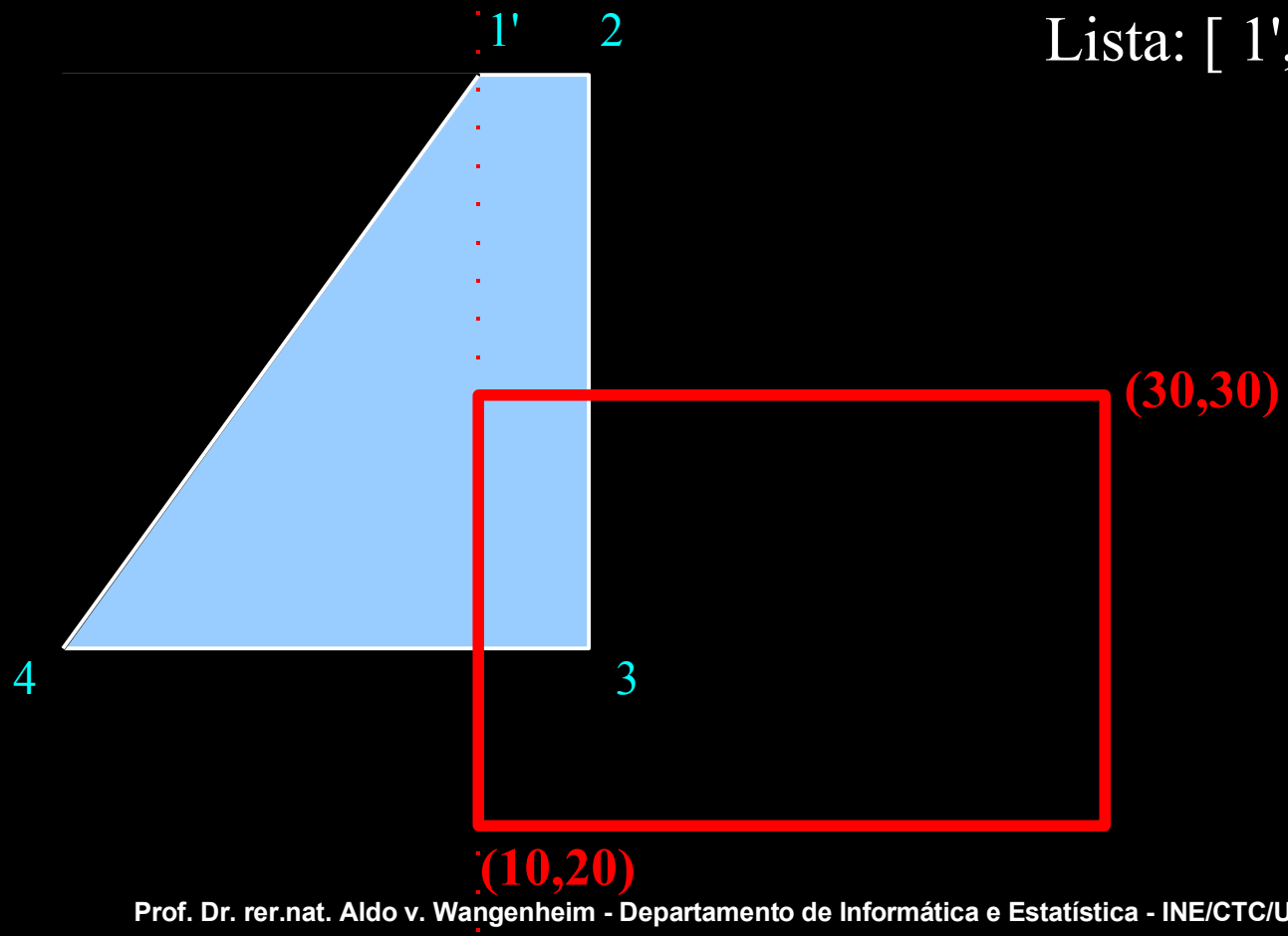
- Borda Esquerda:



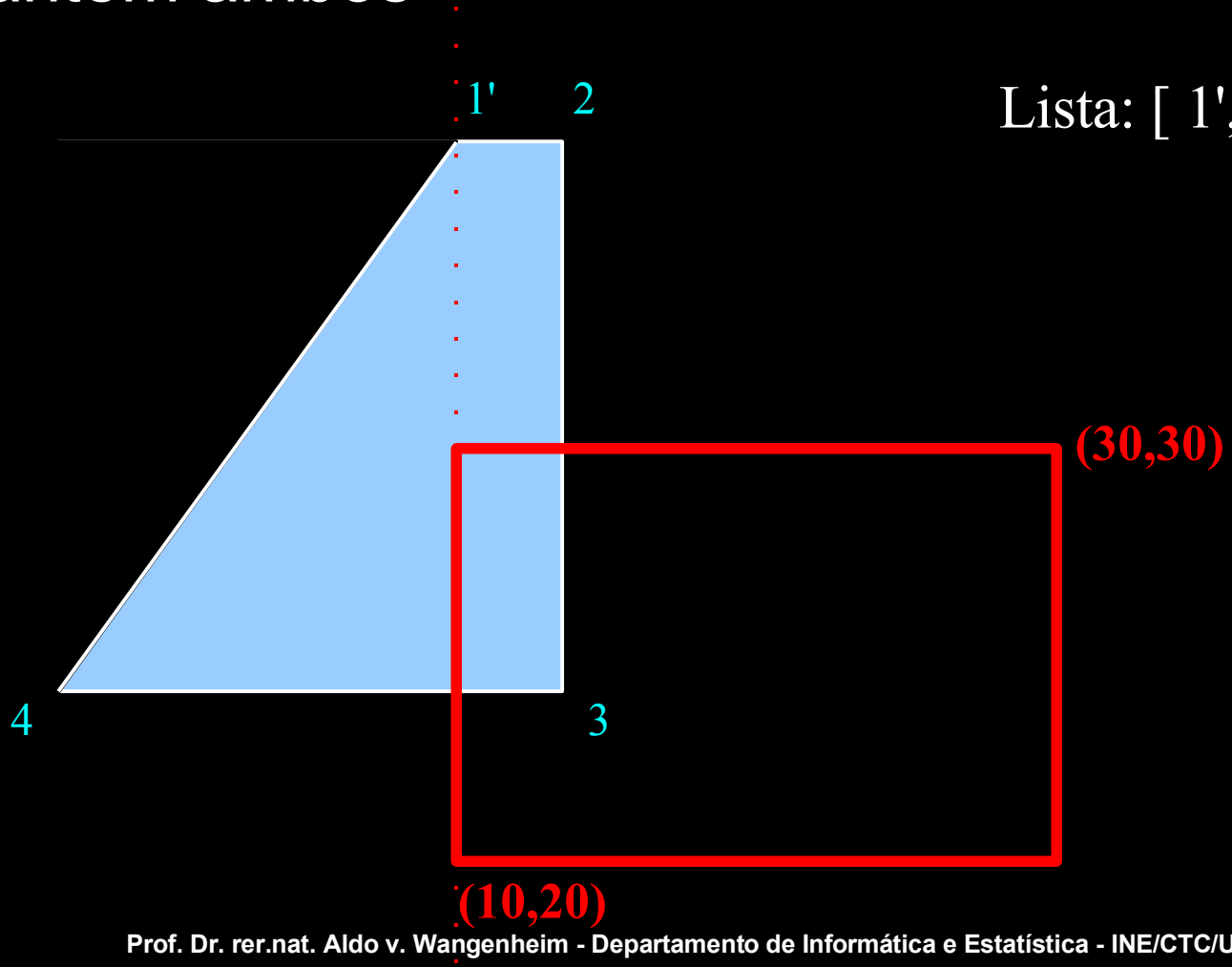
- Vértices 1 e 2: Fora, Dentro
- Remove 1, Mantém 2, Adiciona 1' entre 1 e 2



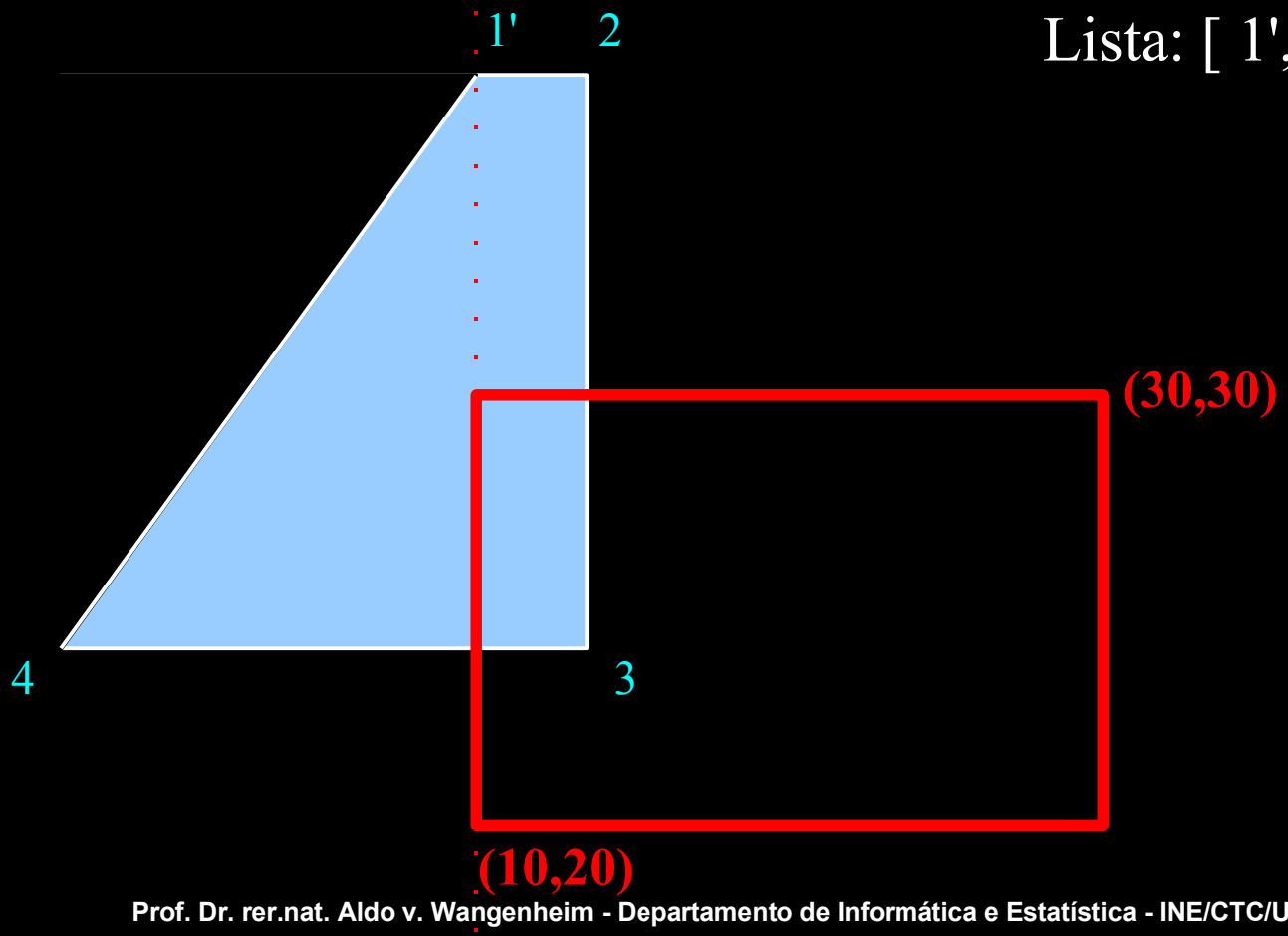
- Vértices 1 e 2: Fora, Dentro
- Remove 1, Mantém 2, Adiciona 1' entre 1 e 2



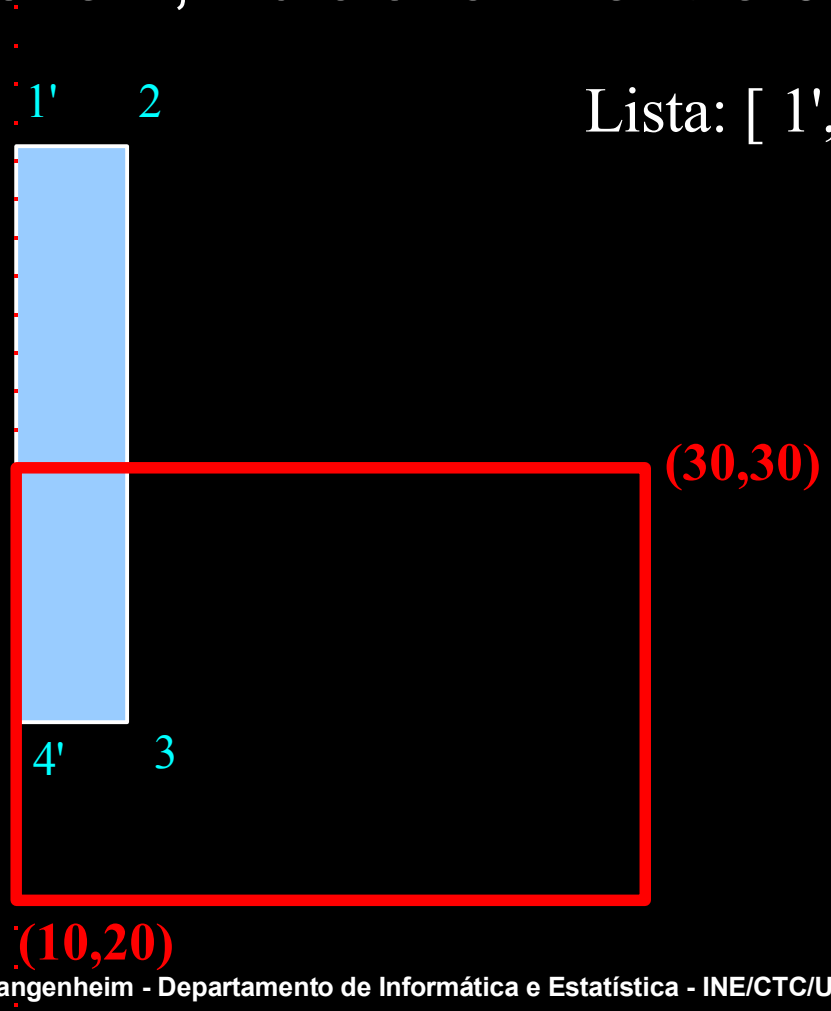
- Vértices 2 e 3: Dentro, Dentro
- Mantém ambos



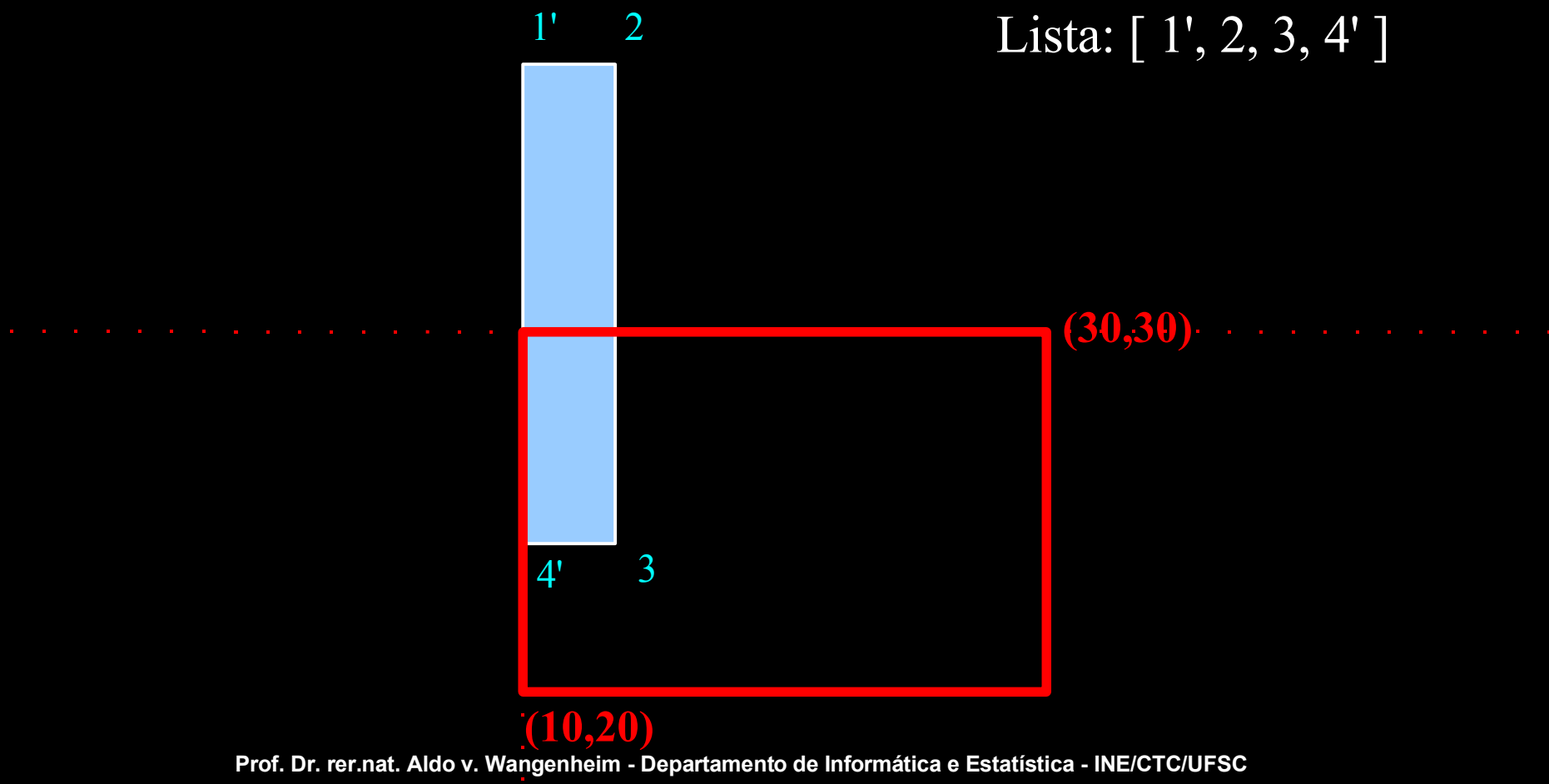
- Vértices 3 e 4: Dentro, Fora
- Mantém 3, Remove 4, Adiciona 4' entre 3 e 4



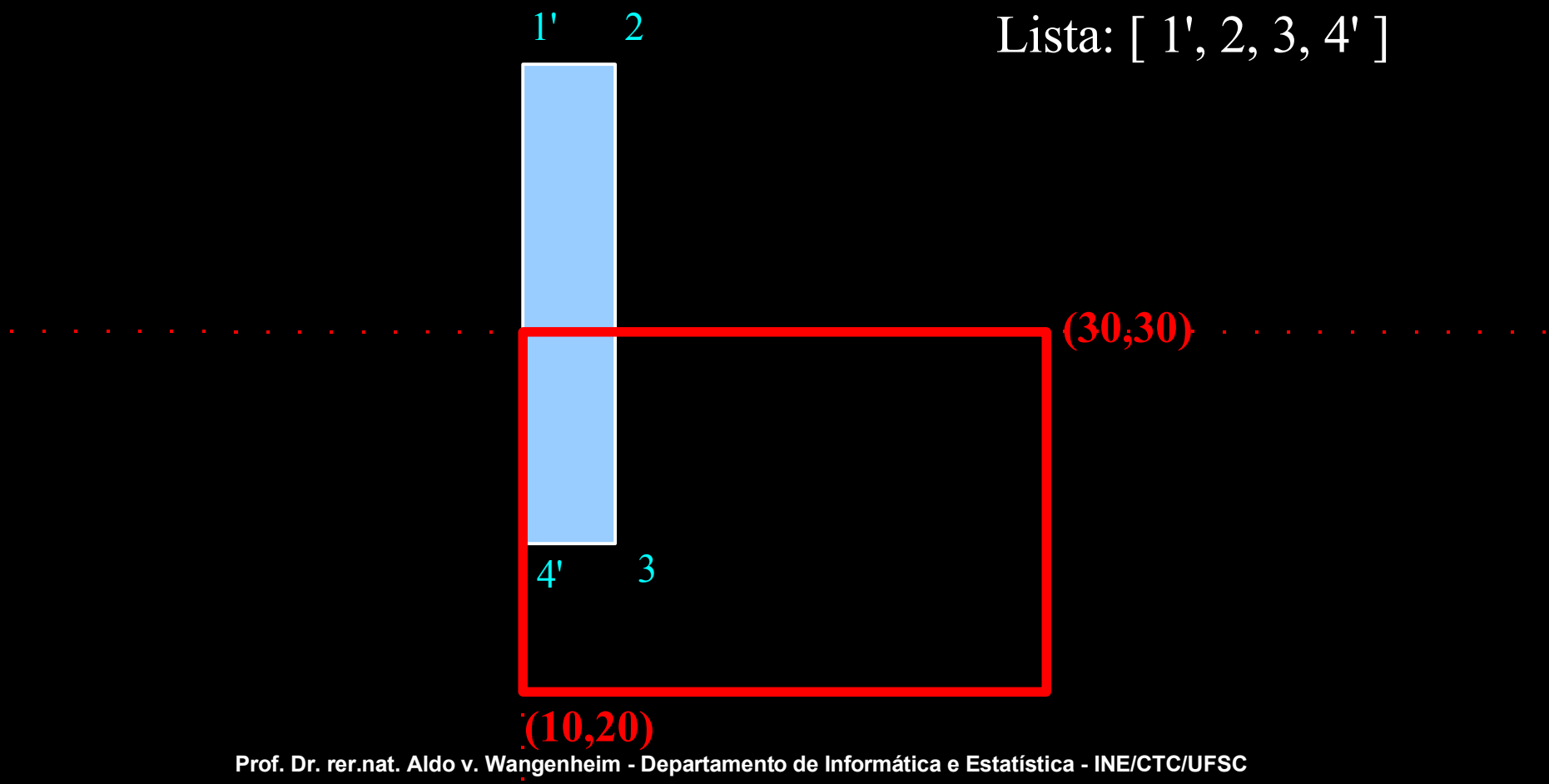
- Vértices 3 e 4: Dentro, Fora
- Mantém 3, Remove 4, Adiciona 4' entre 3 e 4



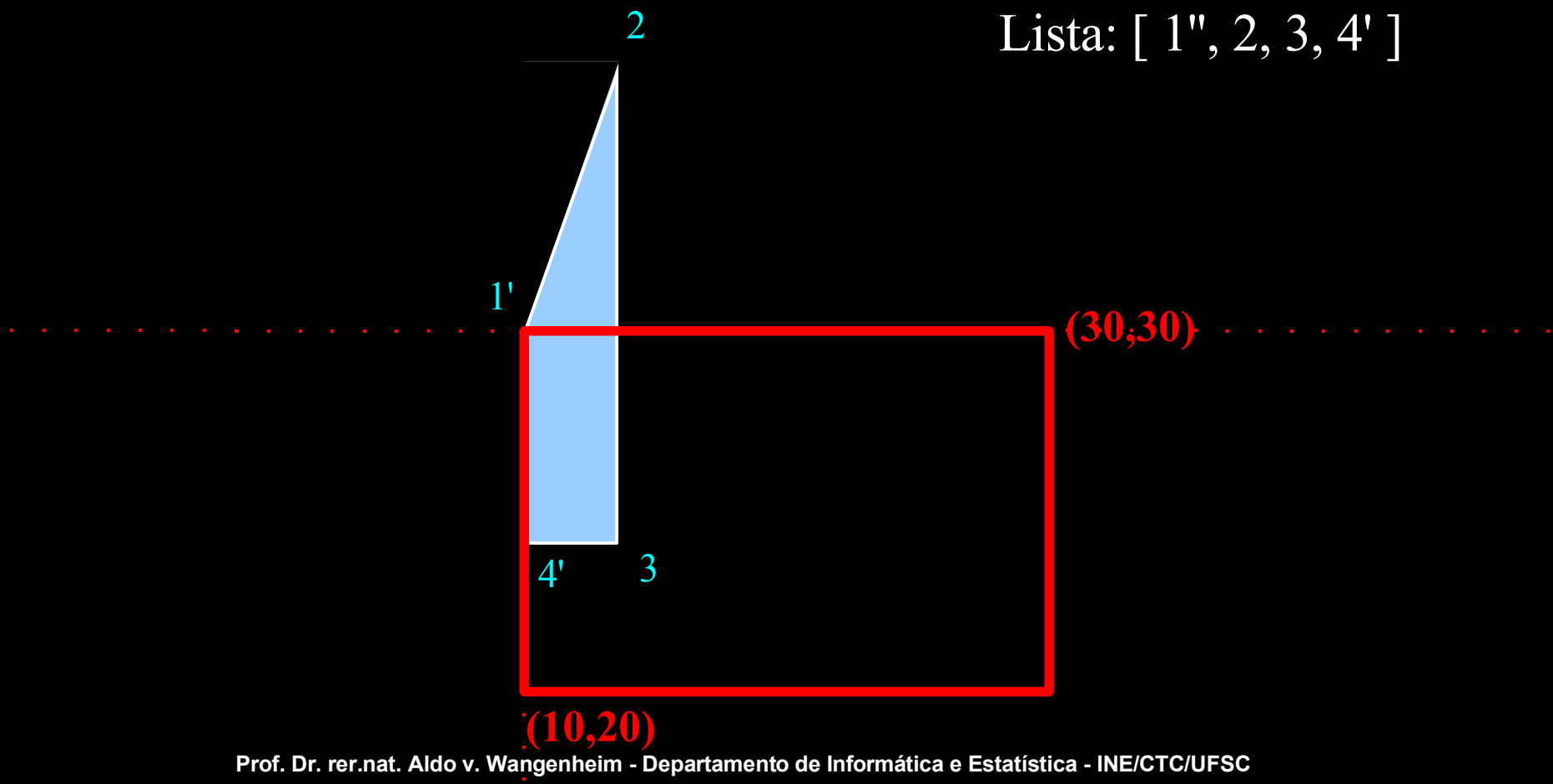
- Borda Superior:



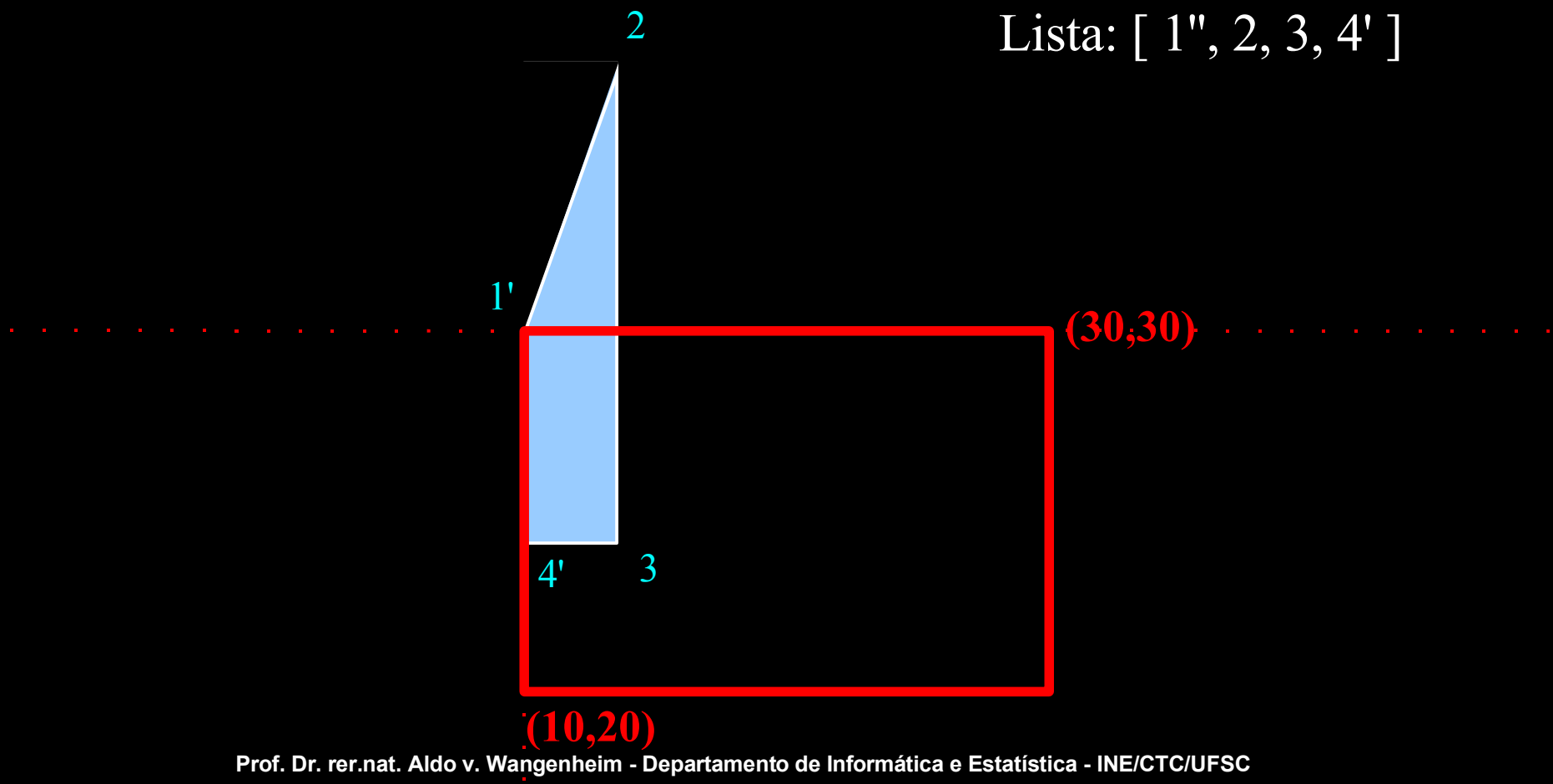
- Vértices 1' e 2: Fora, Fora
- Remove 1', Adiciona 1" entre 1' e 4



- Vértices 1' e 2: Fora, Fora
- Remove 1', Adiciona 1" entre 1' e 4

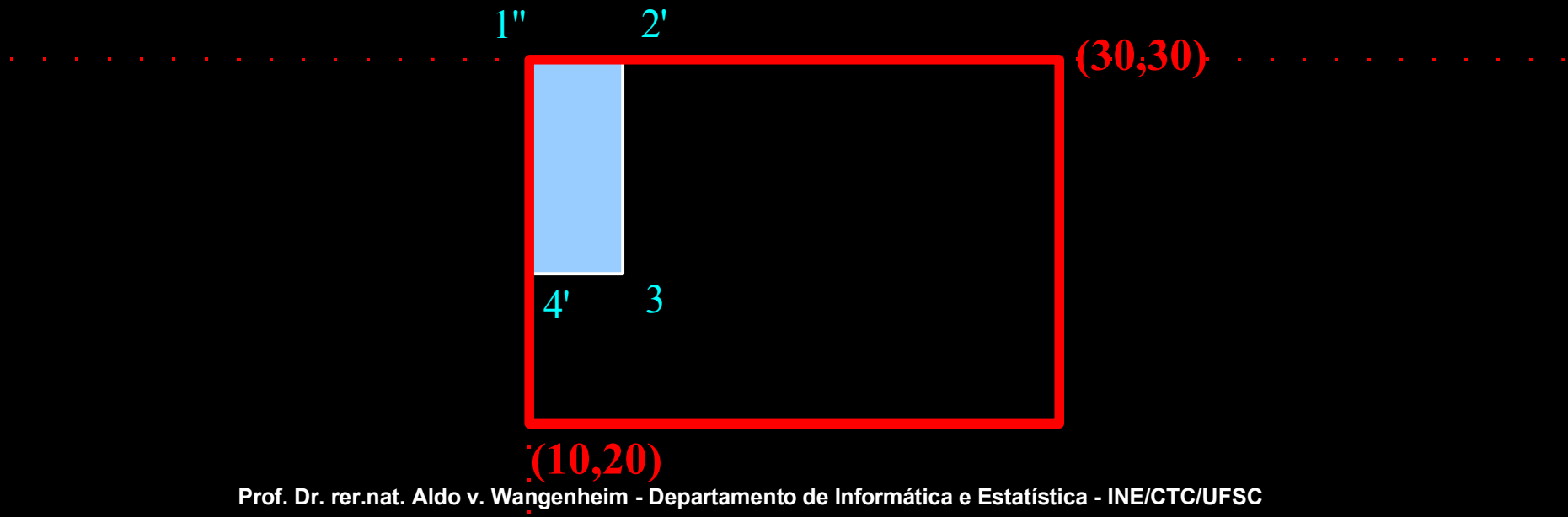


- Vértices 1' e 2: Fora, for a (cont)
- Remove 2, Adiciona 2' entre 2 e 3



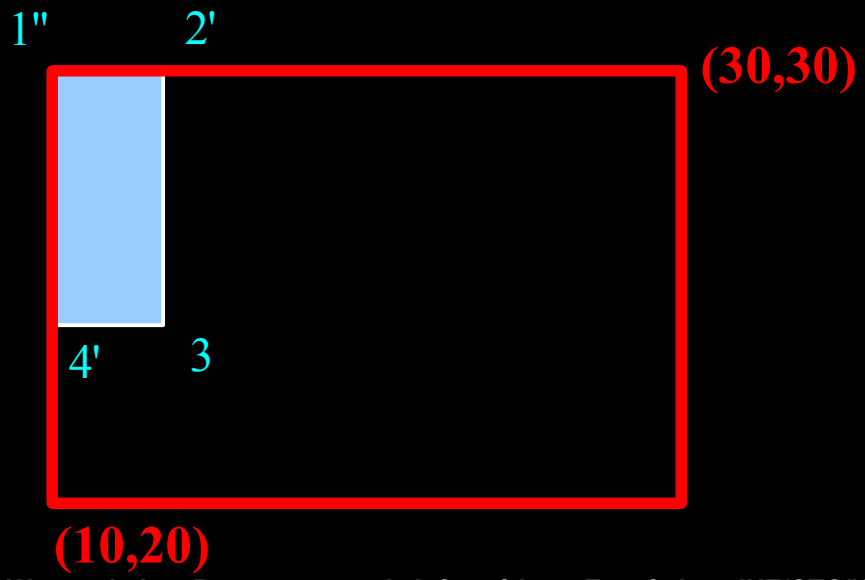
- Vértices 1' e 2: Fora, for a (cont)
- Remove 2, Adiciona 2' entre 2 e 3

Lista: [1'', 2', 3, 4']

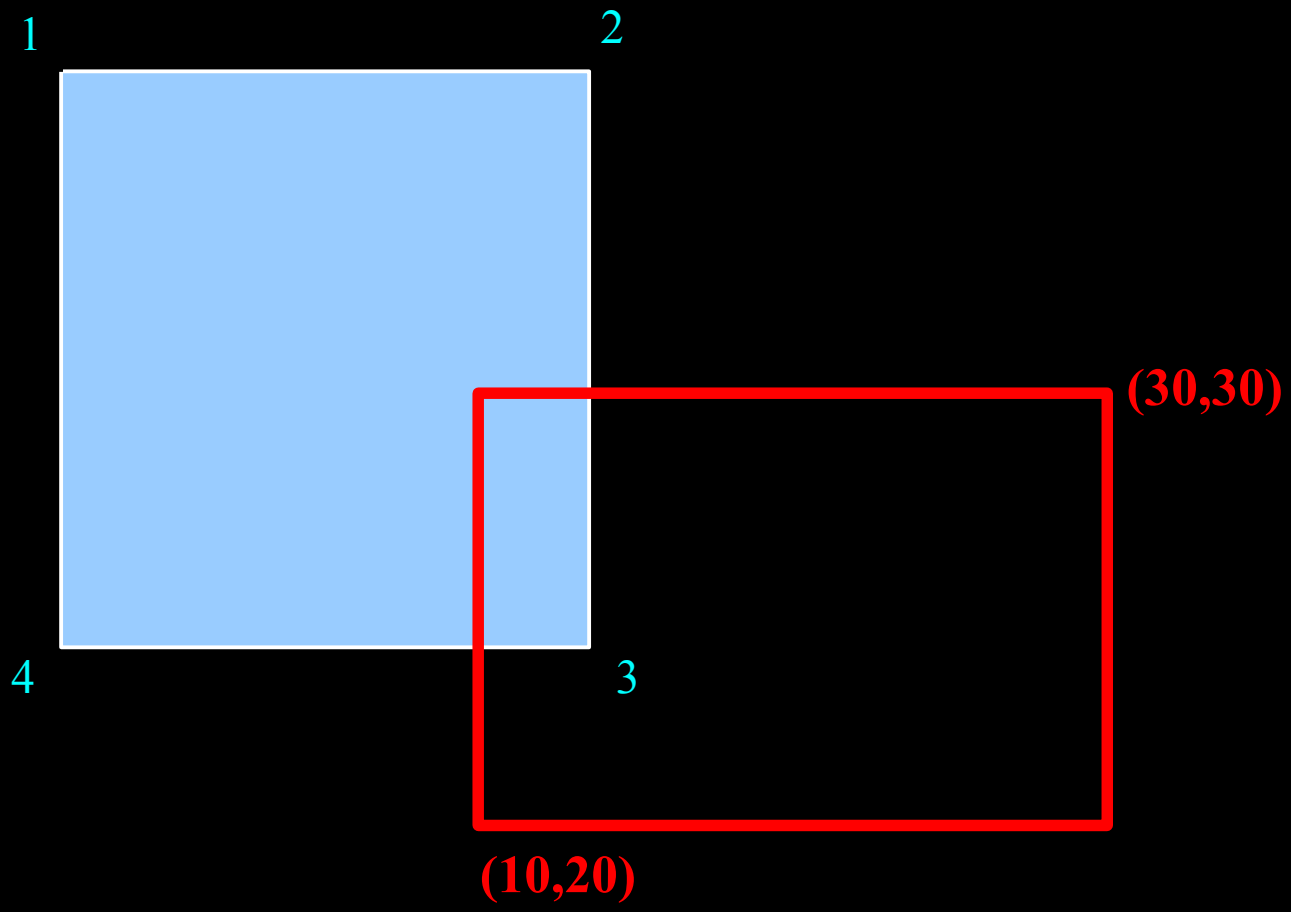


- Polígono final clippado:
  - O algoritmo continua, mas todos os vértices já estão dentro, neste caso...

Lista: [1'', 2', 3, 4']



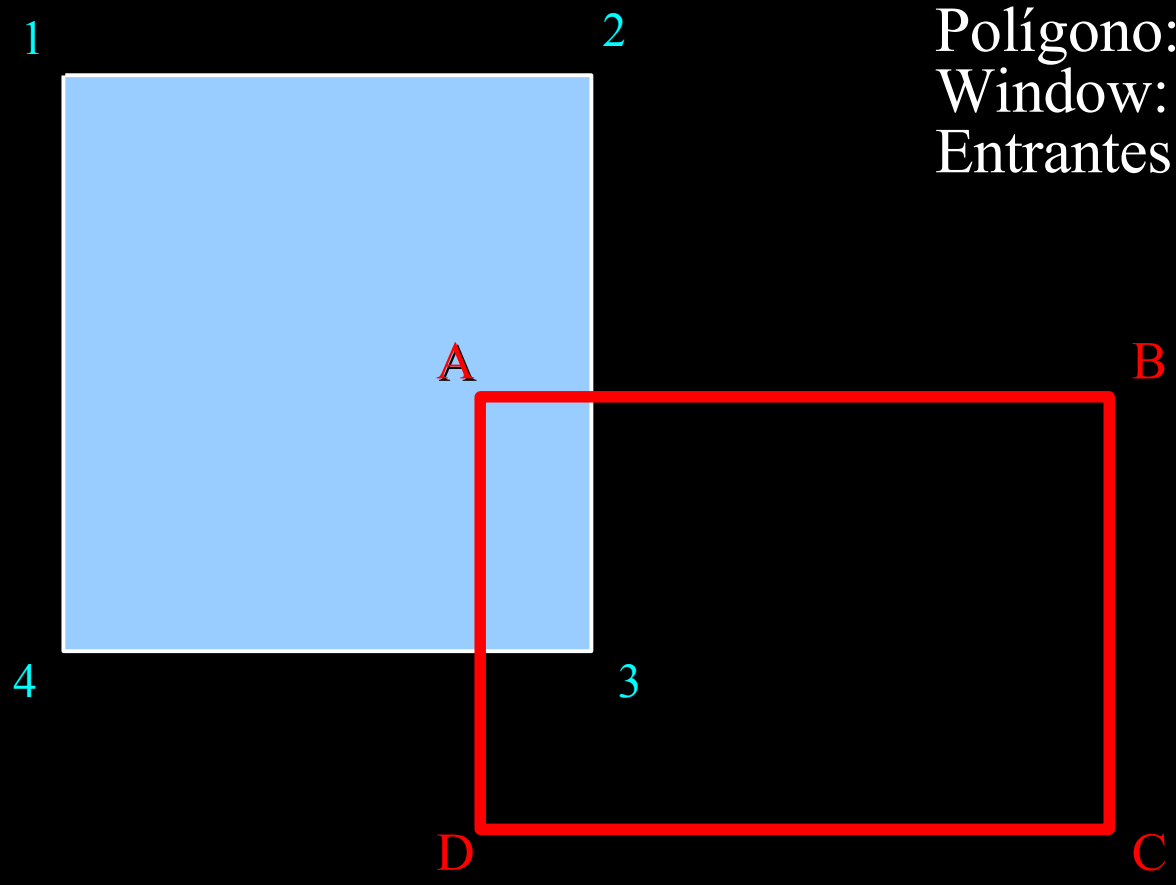
Clipping de Polígonos



- Por Weiler-Atherton:
- Percorra a lista de vértices
  - Percorra a lista de vértices adicionando ou removendo pontos conforme o caso.
- Se processamos em sentido horário procedemos assim:
  - For an outside-to-inside pair of vertices, follow the polygon boundary.
  - For an inside-to-outside pair of vertices, follow the window boundary in clockwise direction.

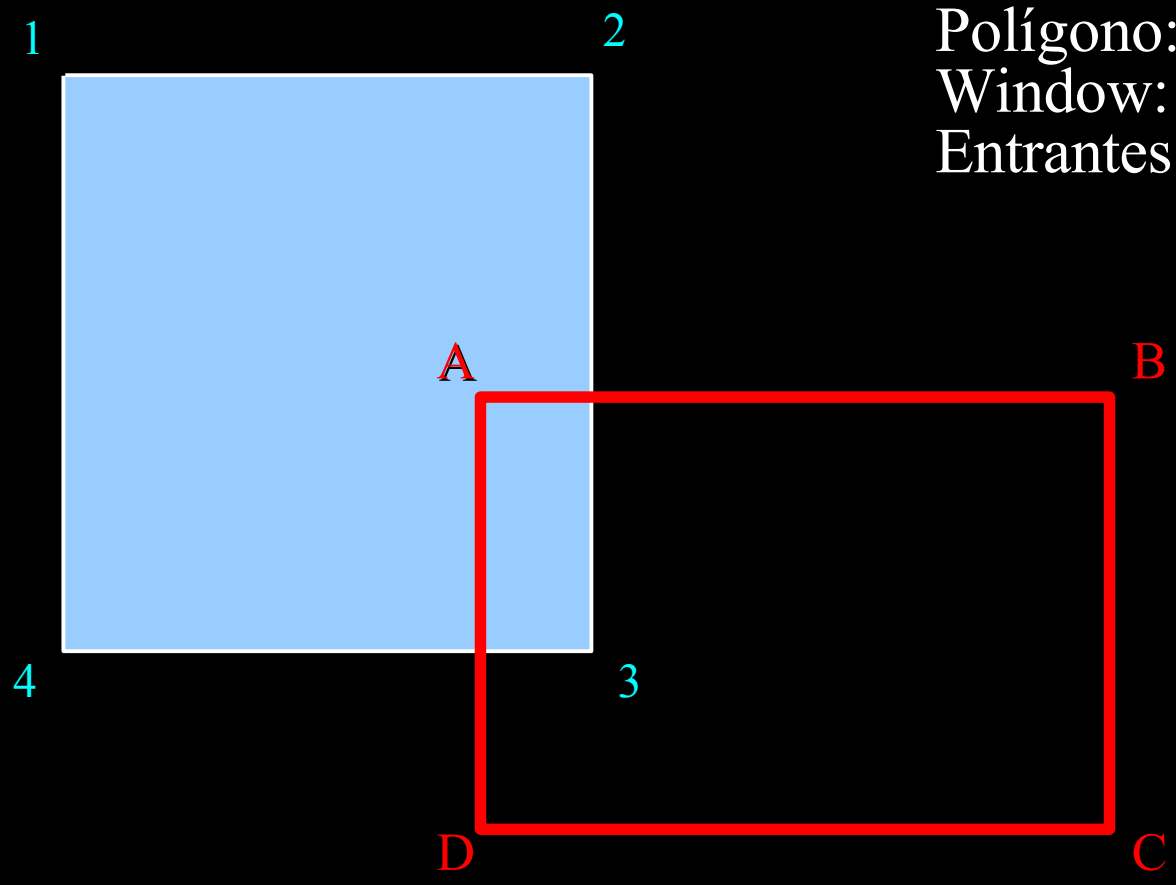
- Listas Ligadas:
  - Uma com os vértices do polígono
  - Outra com os vértices da Window
  - Uma terceira temporária para vértices entrantes

- Vértices 1 e 2: Não corta Window



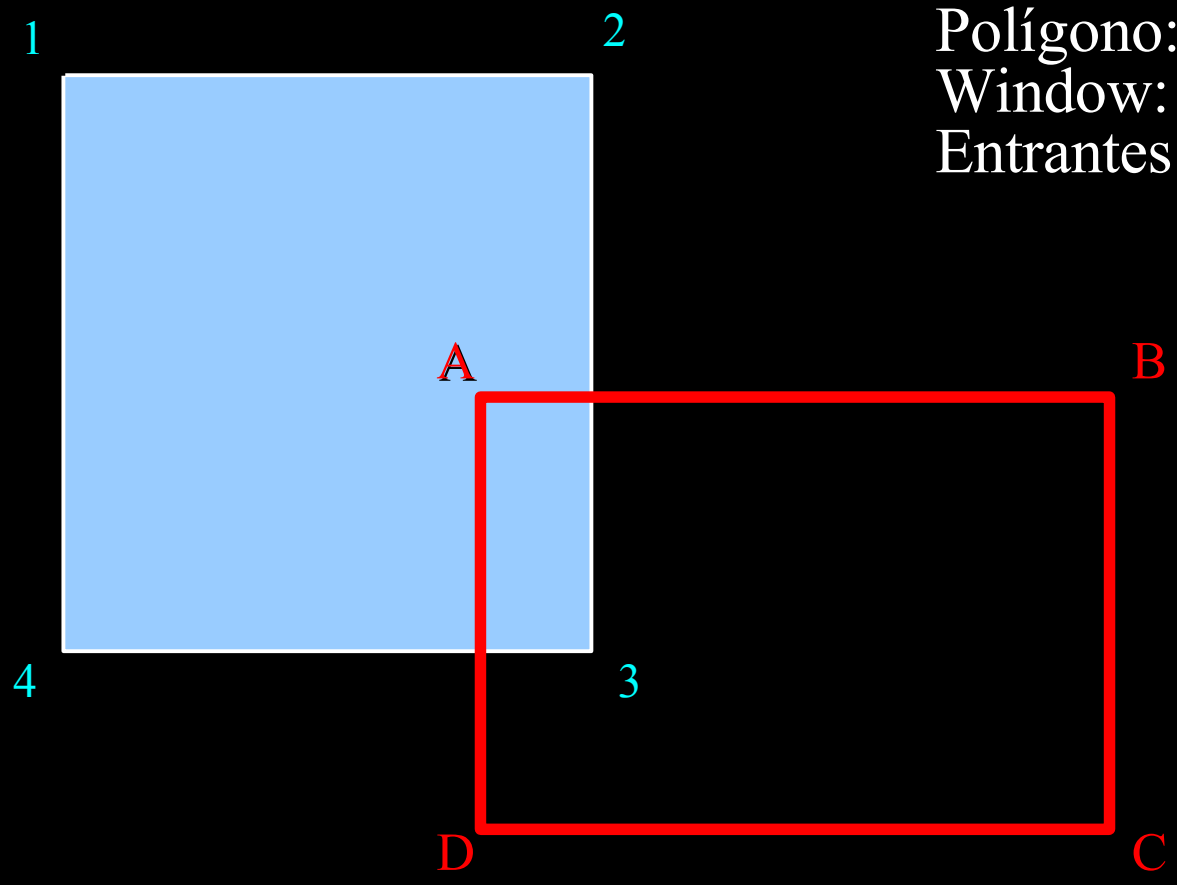
Polígono: [1, 2, 3, 4]
Window: [A, B, C , D]
Entrantes: [ ]

- Vértices 2 e 3: Corta a Window de fora pra dentro:



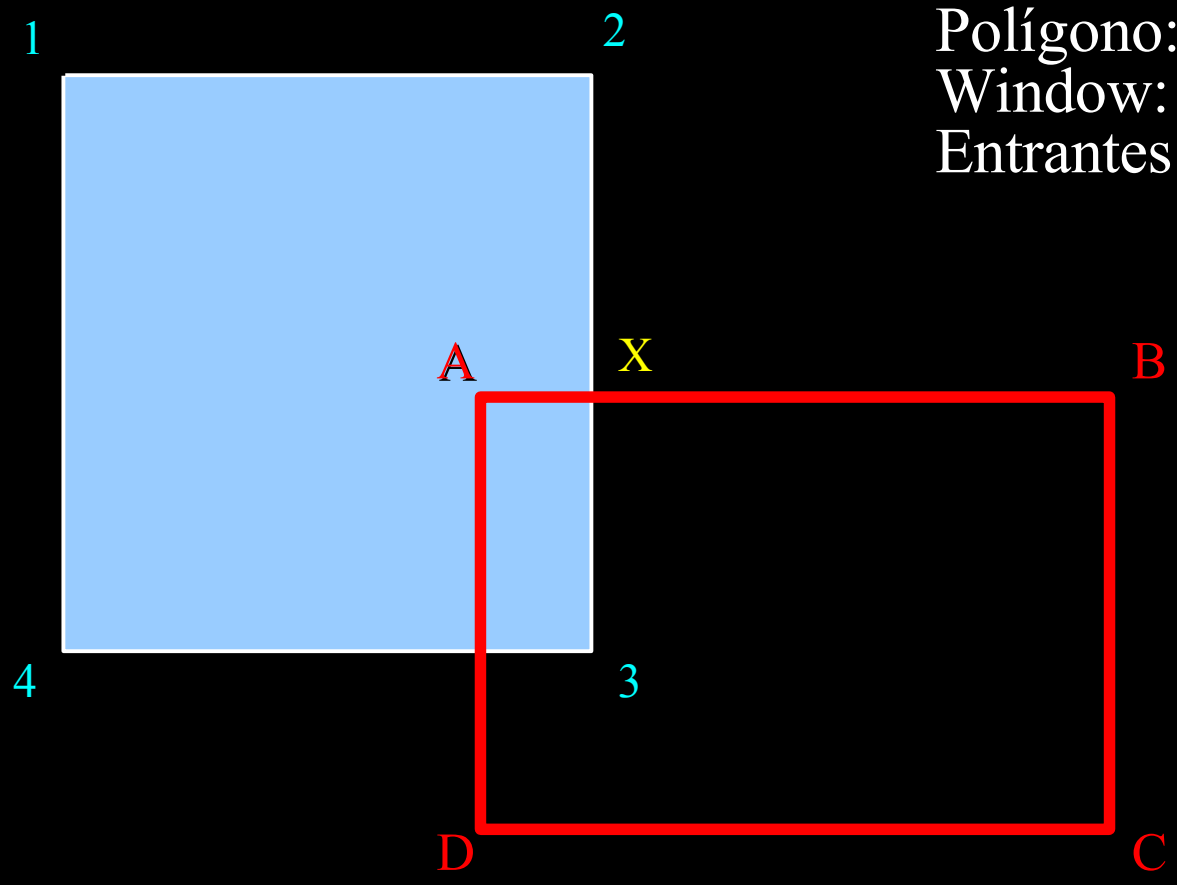
Polígono: [1, 2, 3, 4]
Window: [A, B, C , D]
Entrantes: [ ]

- Vértices 2 e 3: Corta a Window de fora pra dentro:
- Criamos vértice X, adicionado às 3 listas



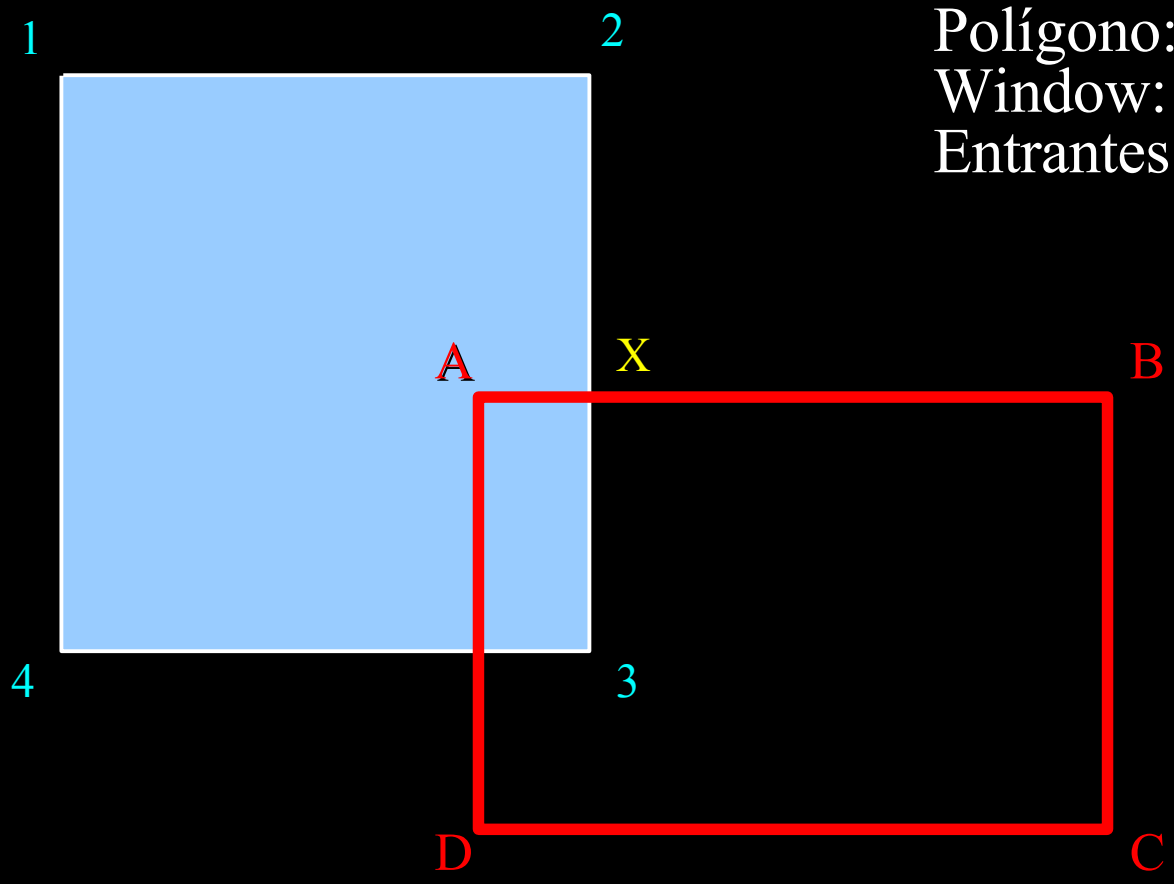
Polígono: [1, 2, 3, 4]
Window: [A, B, C , D]
Entrantes: [ ]

- Vértices 2 e 3: Corta a Window de fora pra dentro:
- Criamos vértice X, adicionado às 3 listas



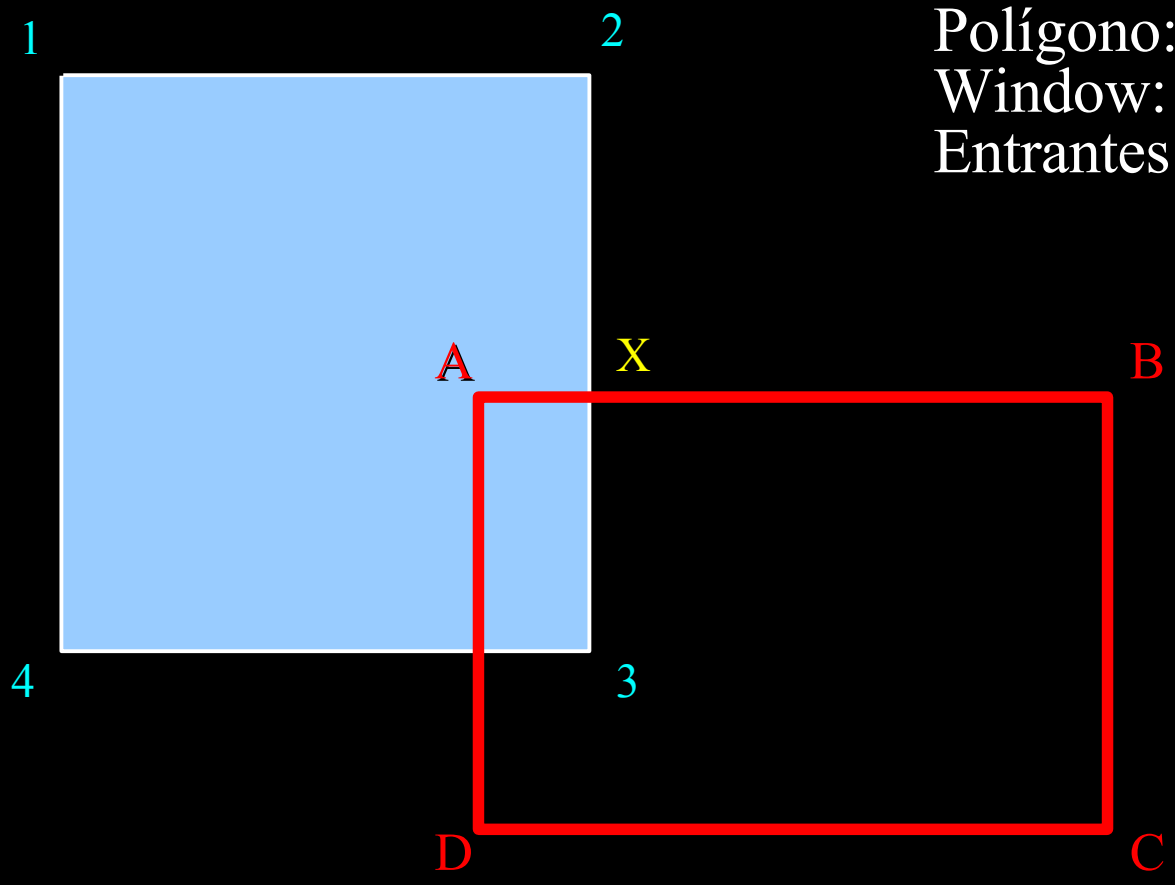
Polígono: [1, 2, X, 3, 4]
Window: [A, X, B, C , D]
Entrantes: [X]

- Vértices 3 e 4: Corta a Window de dentro pra fora:



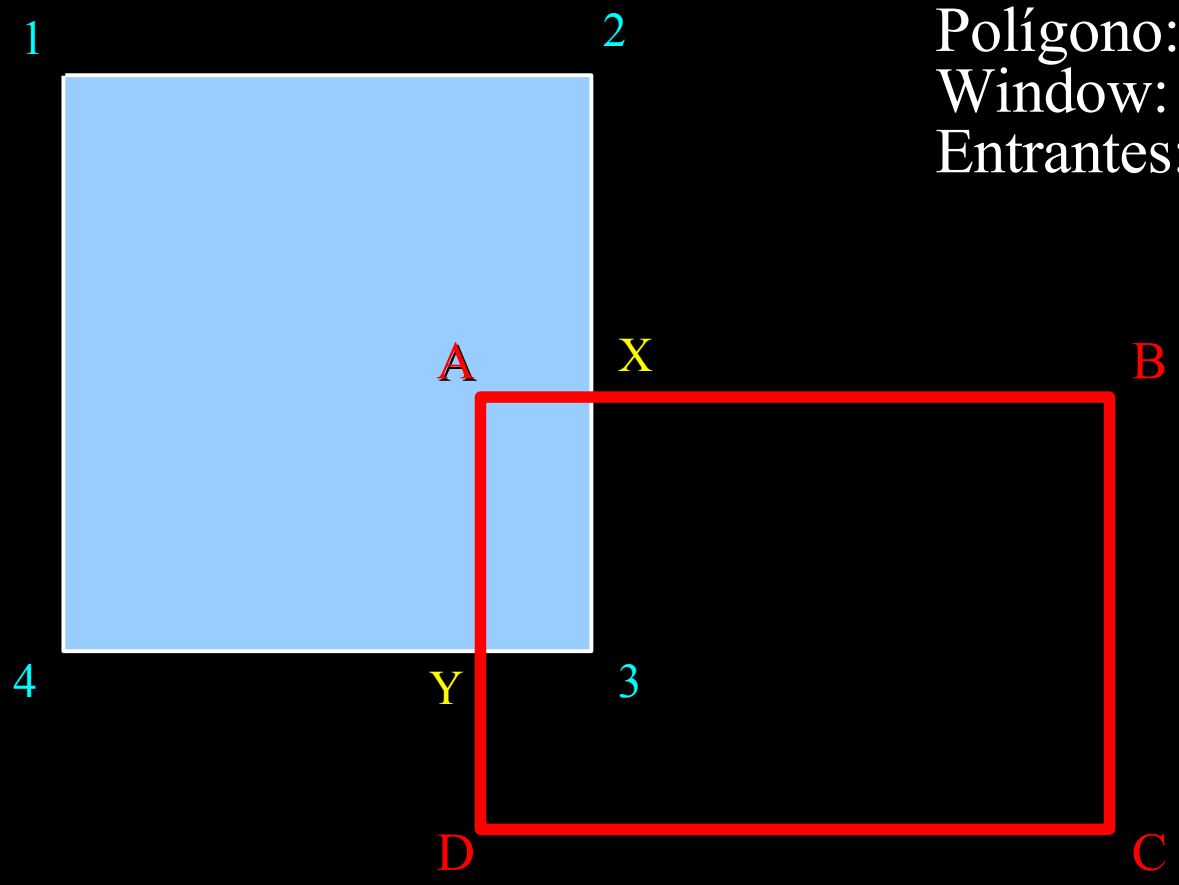
Polígono: [1, 2, X, 3, 4]
Window: [A, X, B, C , D]
Entrantes: [X]

- Vértices 3 e 4: Corta a Window de dentro pra fora:
- Criamos vértice Y, adicionado às 2 primeiras listas



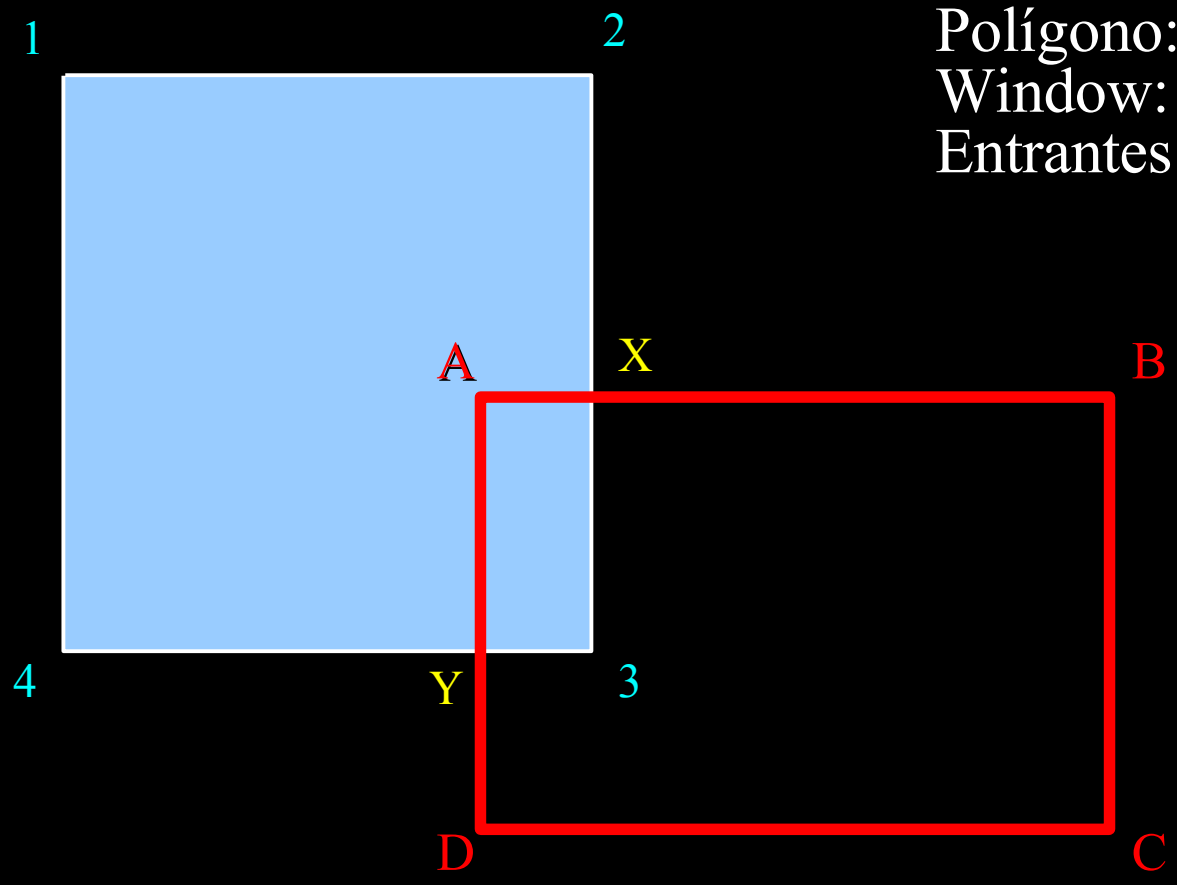
Polígono: [1, 2, X, 3, 4]
Window: [A, X, B, C , D]
Entrantes: [X]

- Vértices 3 e 4: Corta a Window de dentro pra fora:
- Criamos vértice Y, adicionado às 2 primeiras listas



Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]

- Os próximos vértices não cortam a Window



Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]

- Próximo passo:
- Para cada Ponto em “Entrantes”...:
  - Percorra a lista “Polígono” partindo deste ponto em diante – vá guardando os vértices percorridos;
  - Ao encontrar outro vértice adicionado artificialmente, troque para a lista “Window”;

- Para cada Ponto em “Entrantes”... (cont):
  - Percorra a lista “Window” até encontrar outro vértice artificial – continue guardando os vértices;
  - Ao encontrar outro vértice artificial, pare;
  - Os vértices percorridos formam o novo polígono;
  - Remova o Ponto de “Entrantes”.

Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]
Temporária: [ ]

- Ponto em Entrantes: “X”;

Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]
Temporária: [ ]

- Ponto em Entrantes: “X”;
- Percorrendo “Polígono” de X até Y ...

Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]
Temporária: [X]

- Ponto em Entrantes: “X”;
- Percorrendo “Polígono” de X até Y ...

Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]
Temporária: [X, 3]

- Ponto em Entrantes: “X”;
- Percorrendo “Polígono” de X até Y ...

Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]
Temporária: [X, 3, Y]

- Ponto em Entrantes: “X”;
- Percorrendo “Polígono” de X até Y ...

Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]
Temporária: [X, 3, Y]

- Ponto em Entrantes: “X”;
- Percorrendo “Polígono” de X até Y ...
- Percorrendo “Window” de Y até X ...

Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]
Temporária: [X, 3, Y, A]

- Ponto em Entrantes: “X”;
- Percorrendo “Polígono” de X até Y ...
- Percorrendo “Window” de Y até X ...

Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]
Temporária: [X, 3, Y, A, X]

- Ponto em Entrantes: “X”;
- Percorrendo “Polígono” de X até Y ...
- Percorrendo “Window” de Y até X ...

Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]
Temporária: [X, 3, Y, A, X]

- Ponto em Entrantes: “X”;
- Percorrendo “Polígono” de X até Y ...
- Percorrendo “Window” de Y até X ...
- Alcançamos outro “artificial”
  - Não precisava ser X, necessariamente!

Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]
Temporária: [X, 3, Y, A, X]

- Polígono final: [X, 3, Y, A, X]

Polígono: [1, 2, X, 3, Y, 4]
Window: [A, X, B, C , D, Y]
Entrantes: [X]
Temporária: [X, 3, Y, A]

- Polígono final: [X, 3, Y, A, X]