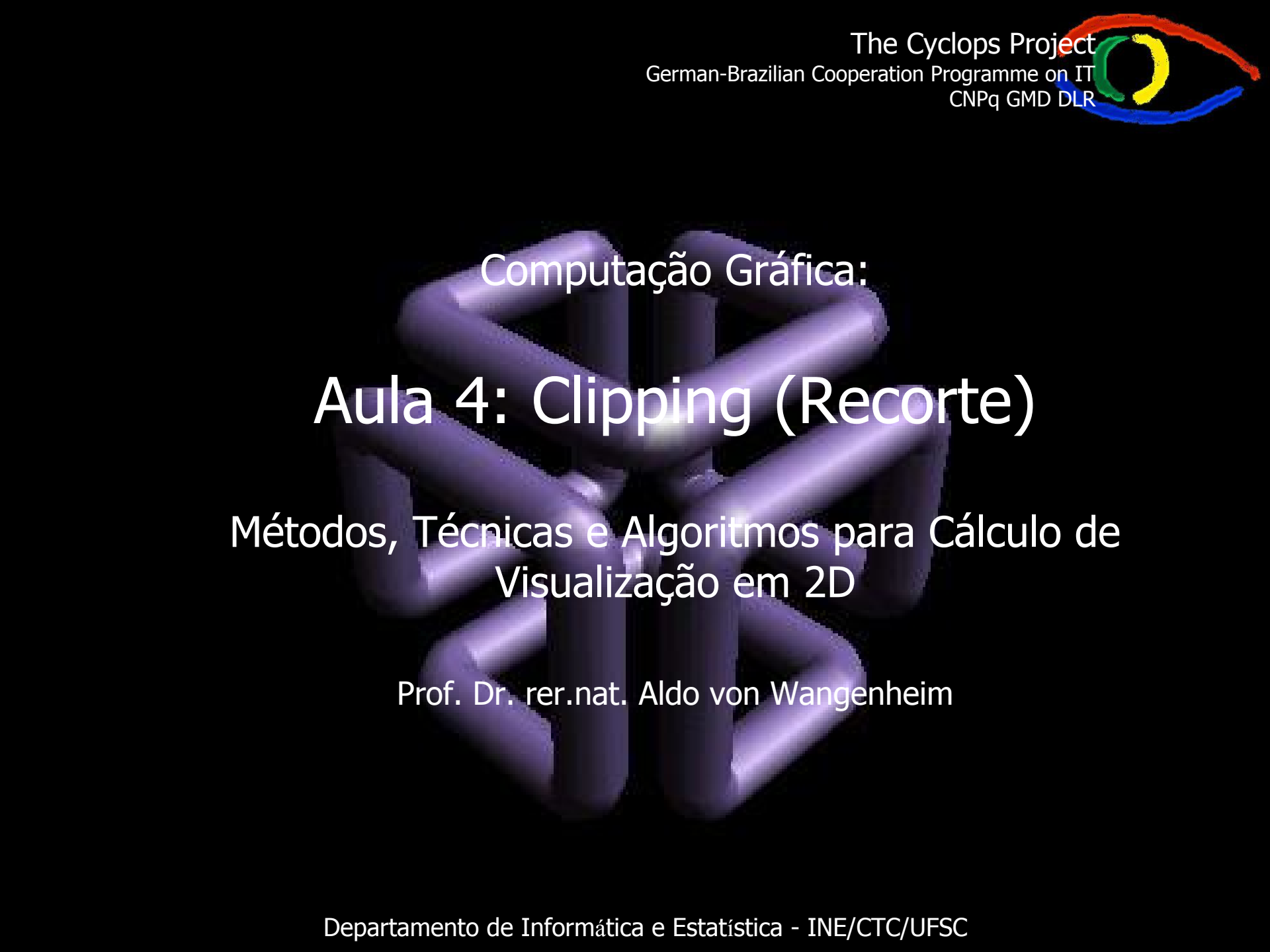
The Cyclops Project
German-Brazilian Cooperation Programme on IT
CNPq GMD DLR



Computação Gráfica:

Aula 4: Clipping (Recorte)

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

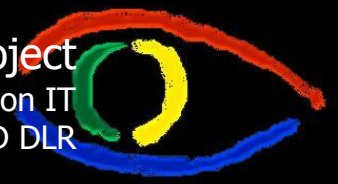
Prof. Dr. rer.nat. Aldo von Wangenheim

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.



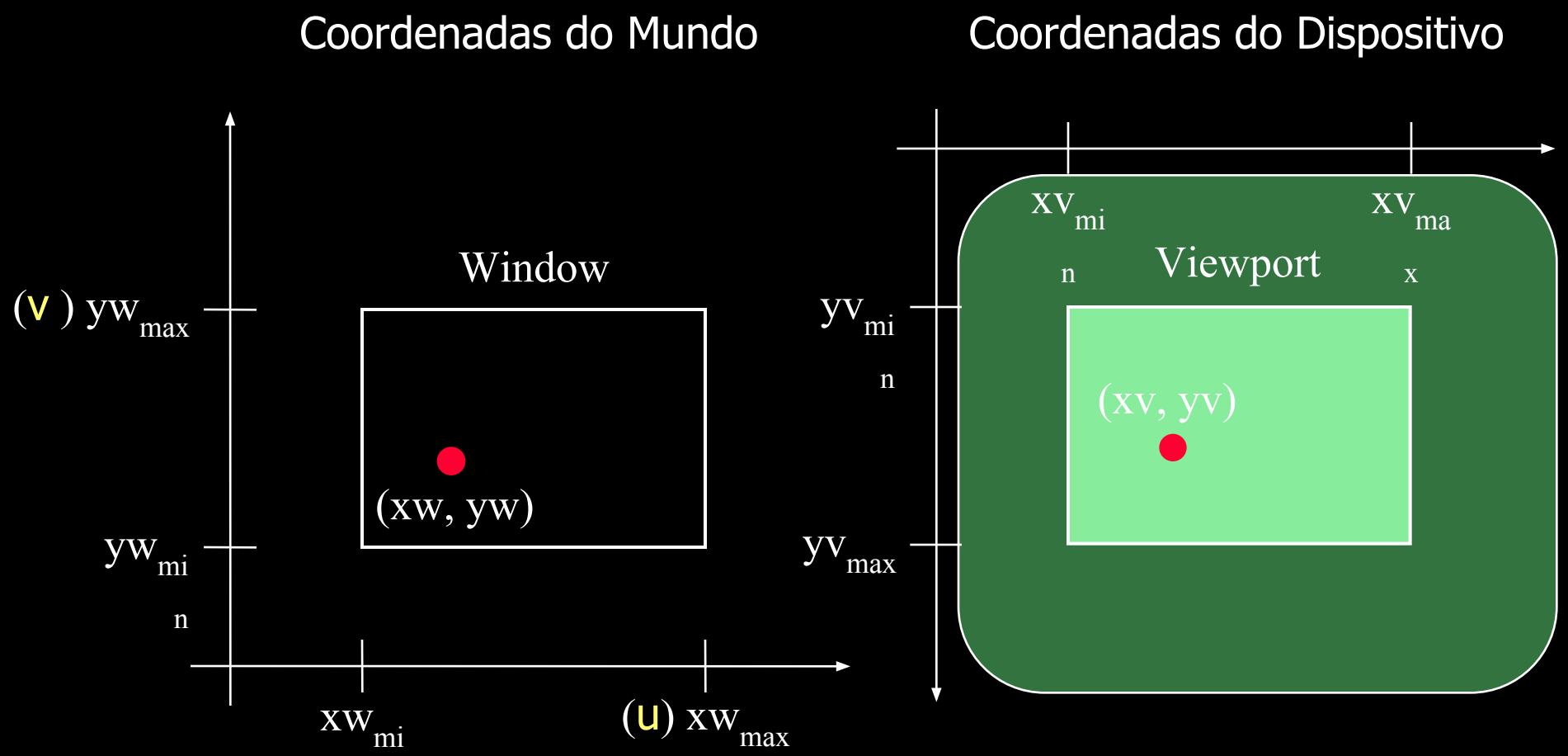
Computação Gráfica:

4.1. Sistema de Coordenadas Normalizado

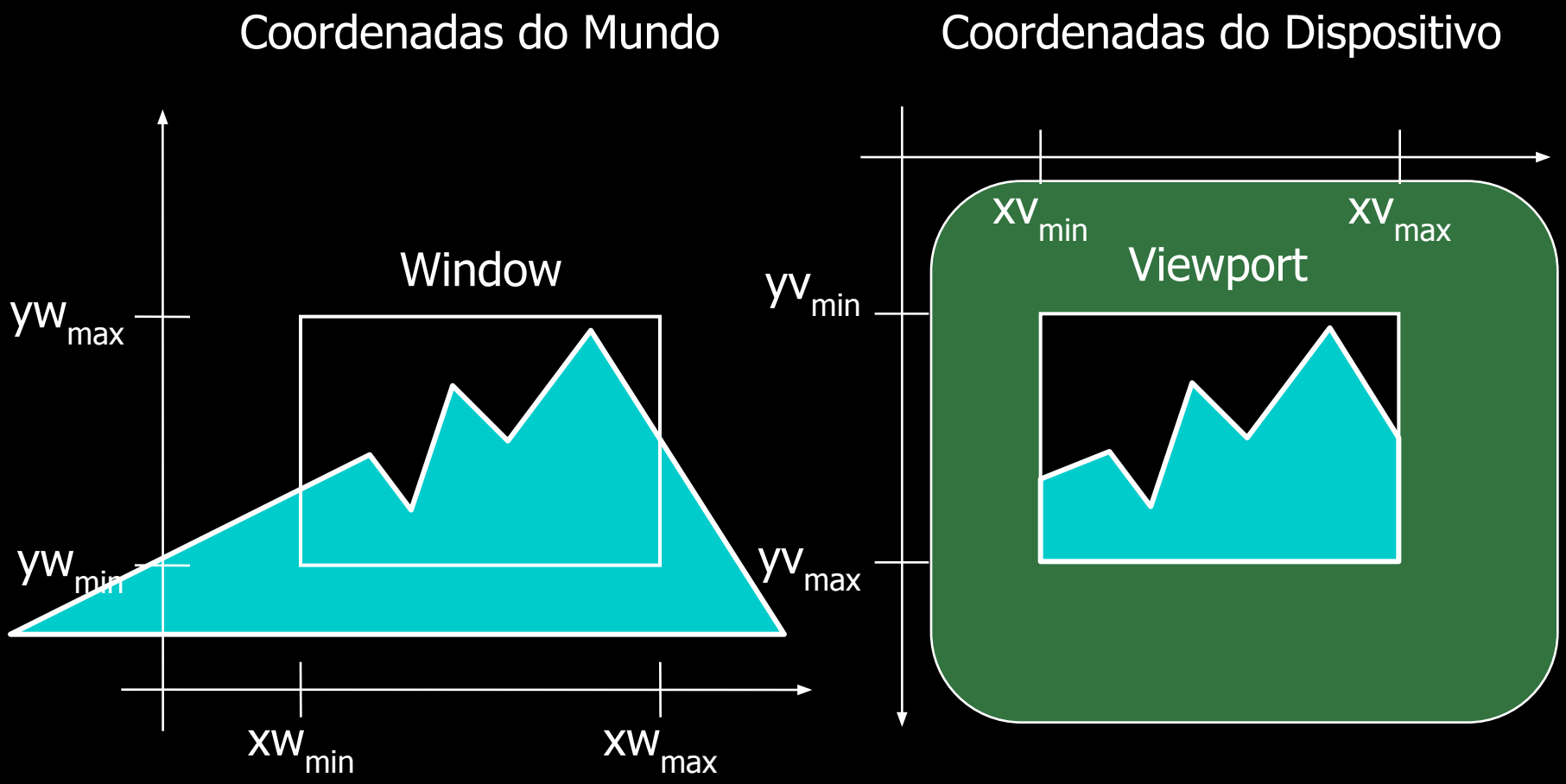
As entidades de Visualização (recapitulando):

- 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 parte de uma cena em coordenadas do mundo para coordenadas do 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.

Efeitos de Visualização com Window

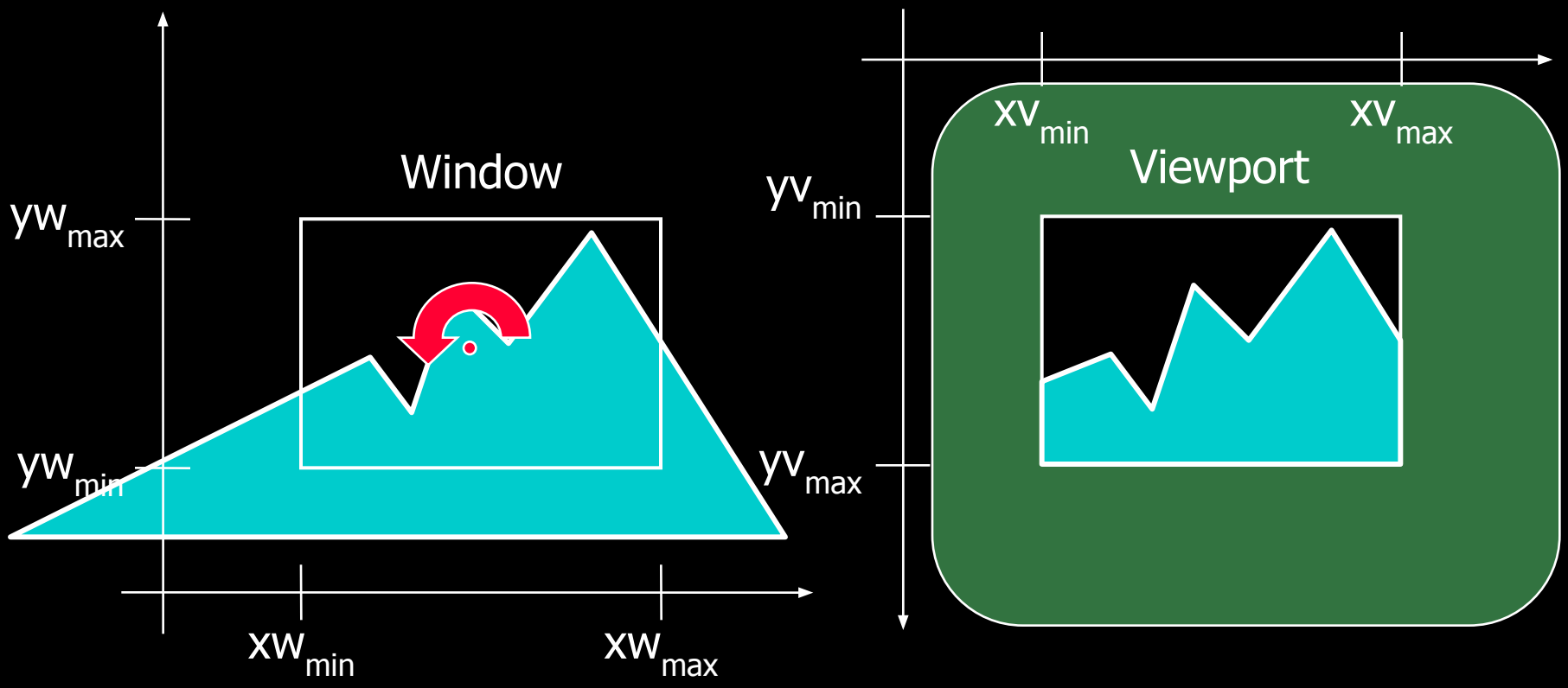
- Efeito de Zooming
  - Mapeamento sucessivo de windows de tamanho diferente em viewports de tamanho fixo.

- Efeito de Panning
  - Movimentação de uma window de tamanho fixo através de vários objetos em uma cena.
- Importante: Independência de dispositivo.
  - Windows são tipicamente definidas dentro de um quadrado unitário (normalized coordinates)

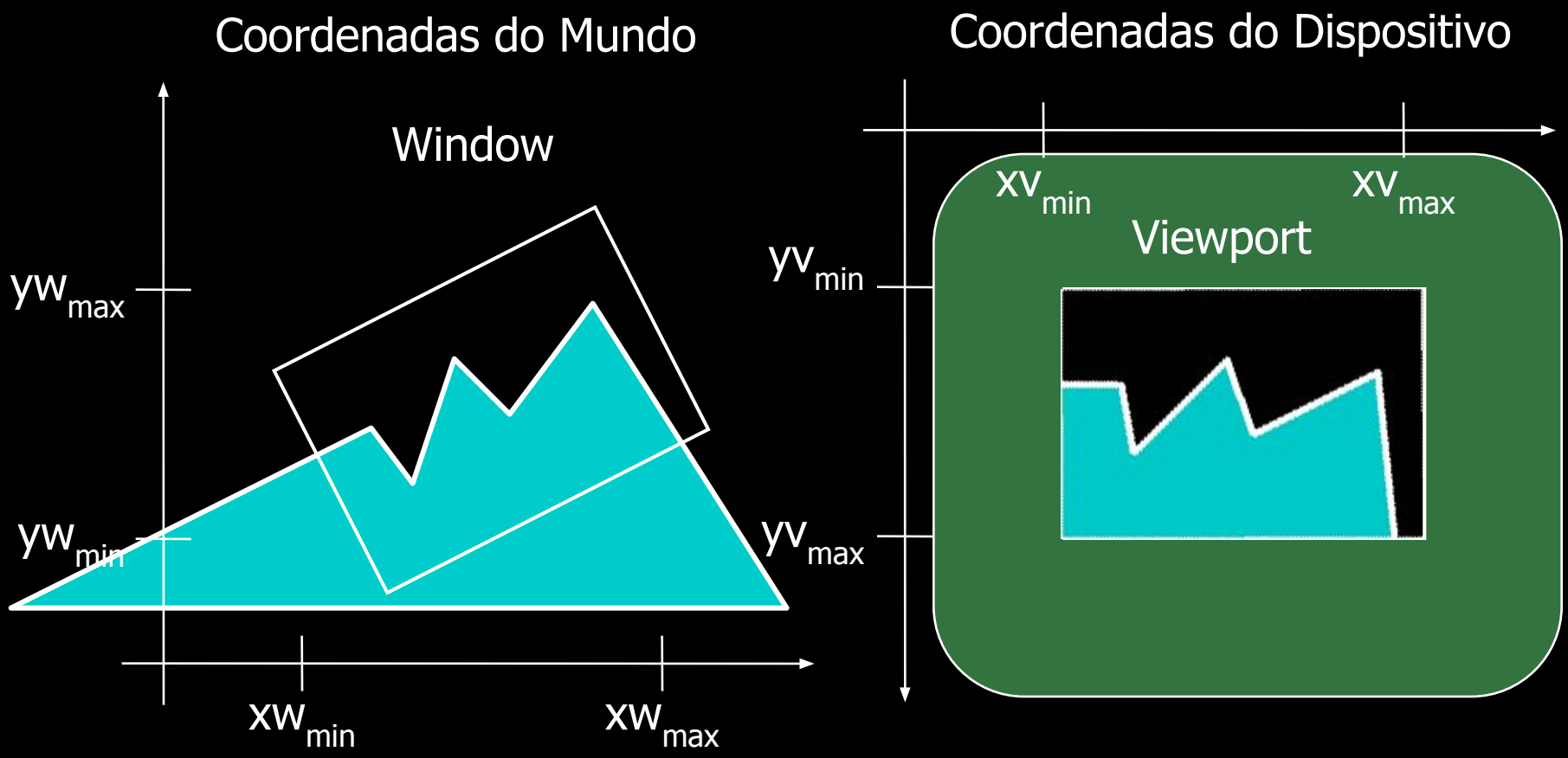
As entidades de Visualização (extendendo o conceito de Window):

- 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

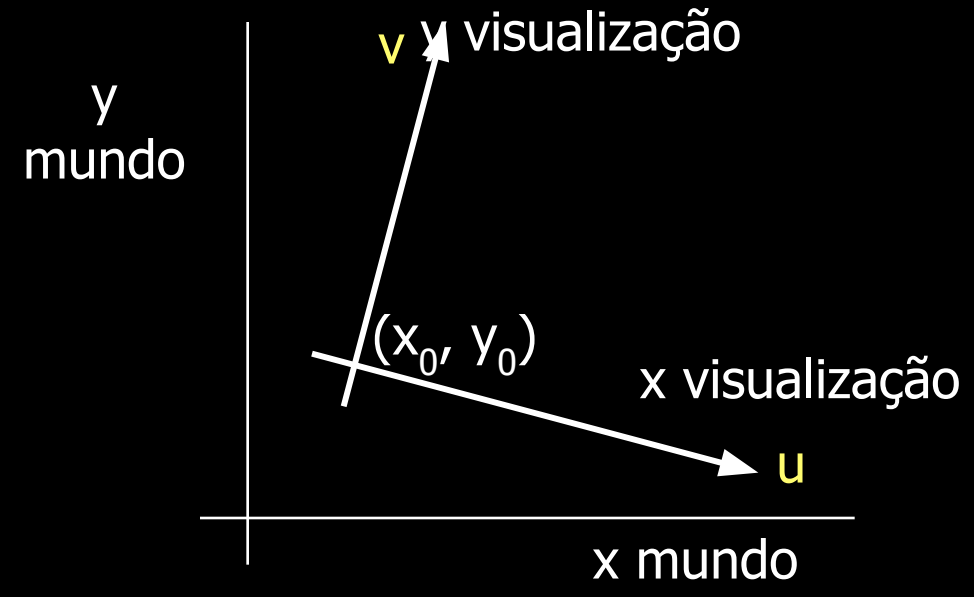


Visualização de Mundos Complexos em duas dimensões

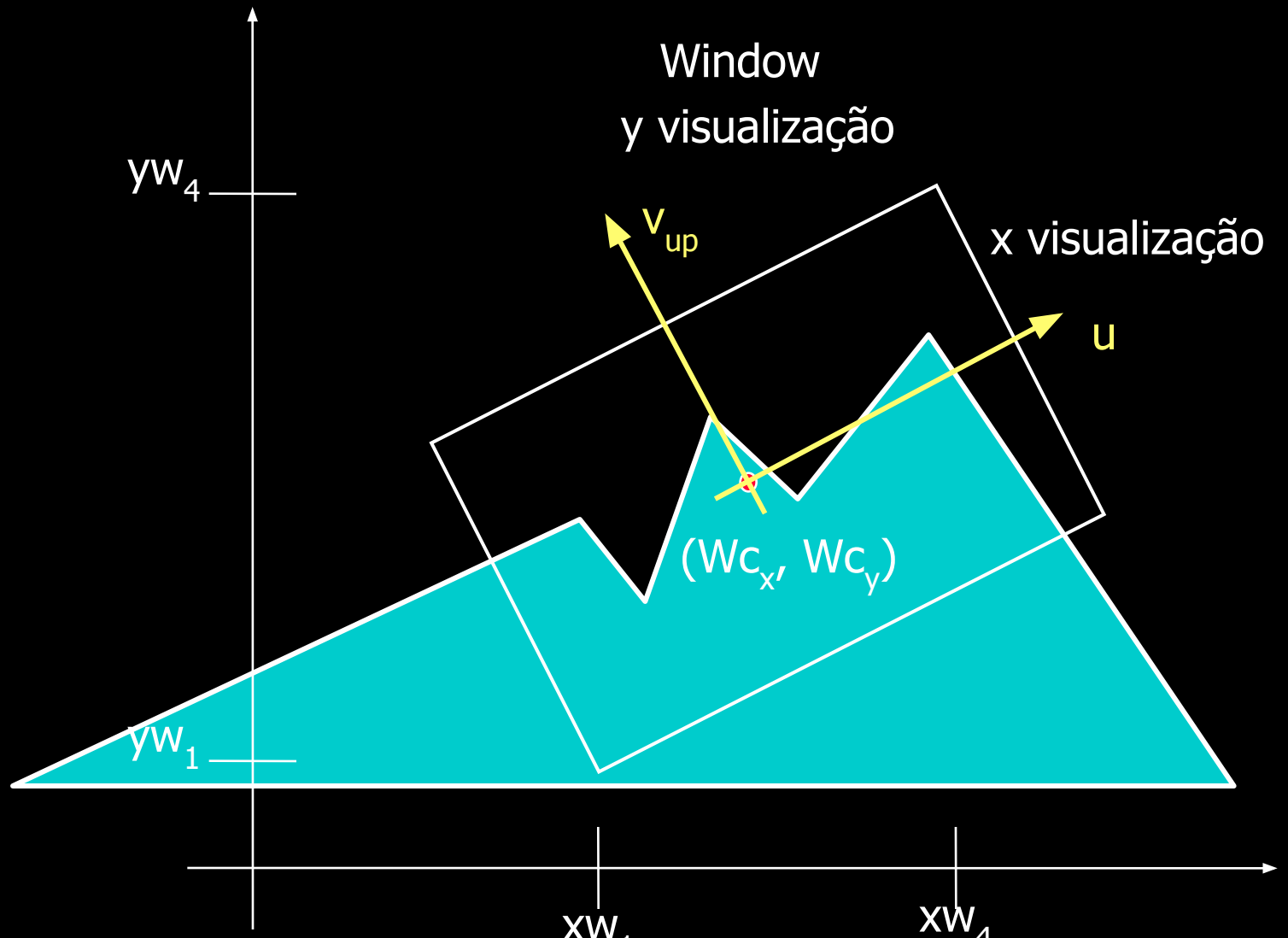


Visualizando o Sistema de Coordenadas Normalizado

- 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
y visualização
x visualização
yw_4
v_{up}
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.

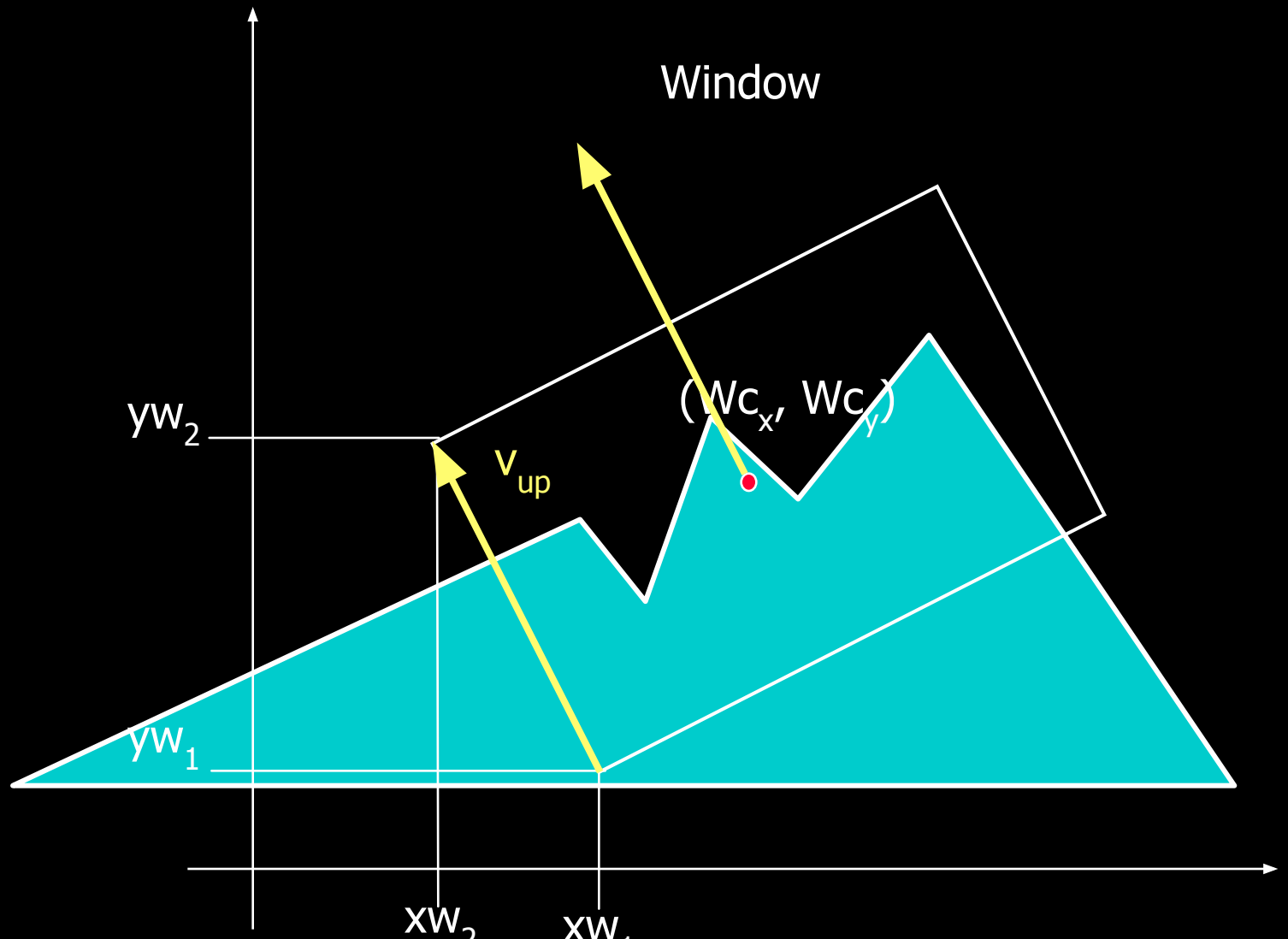
Algoritmo Gerar Descrição em SCN

0. Crie ou mova a window onde desejar;
1. Translade Wc para a origem;
   - Translade o mundo de [-Wcx, -Wcy].
2. Determine vup e o ângulo de vup com Y
3. Rotacione o mundo de forma a alinhar vup com o eixo Y;
   - Rotacione o mundo por $-\theta(Y, v_{up})$.
4. Normalize o conteúdo da window, realizando um escalonamento do mundo;
5. Armazene as coordenadas SCN de cada objeto.
   - Você vai usar na transformada de viewport.

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 v_{up} por $(xw_1, yw_1)(xw_2, yw_2)$ ao invés de $(Wc_x, Wc_y)(v_{upx}, v_{upy})$

Parte I: Computação Gráfica Básica - Implementação de um Sistema Gráfico Interativo



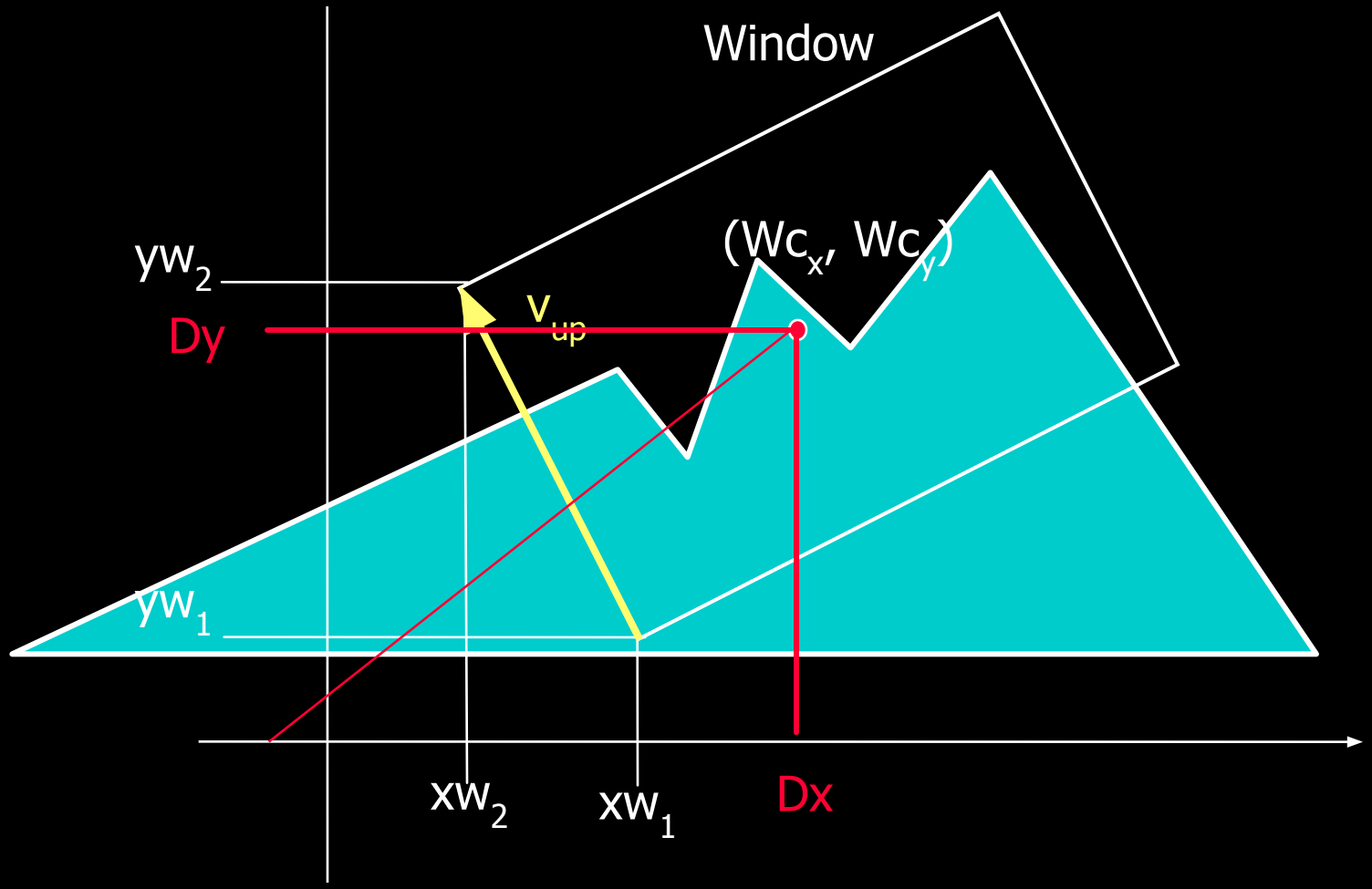
Alternativas ao Sistema de Coordenadas Normalizado

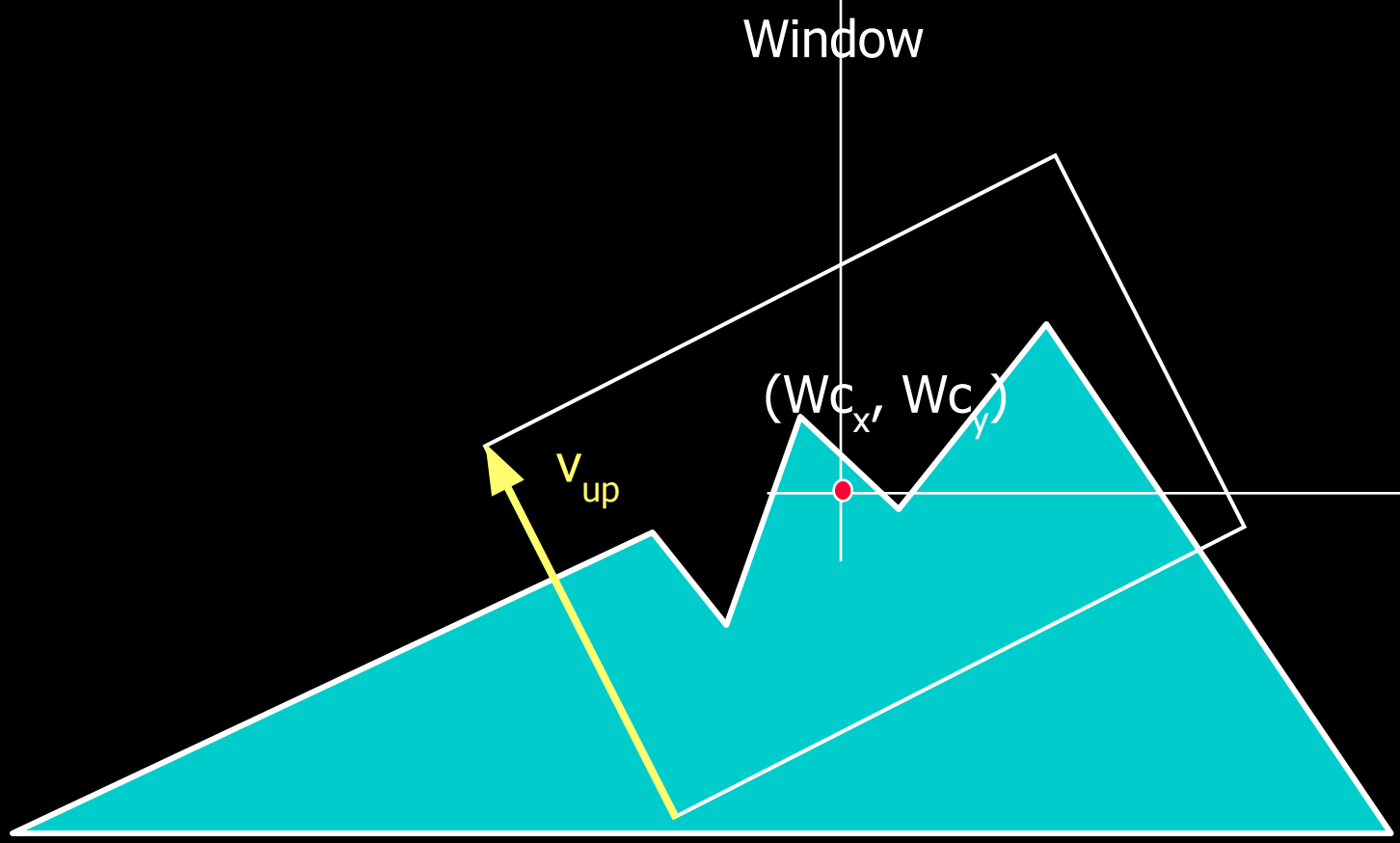
- PPC: Transformamos o sistema de coordenadas para um sistema com origem em W_c e eixos paralelos aos limites da Window
  - Se v_{up} for paralelo ao eixo Y, apenas transladamos W_c para a origem e aplicamos a transformada de viewport.
  - Se v_{up} não for paralelo ao eixo Y, rotacionamos o mundo e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

Algoritmo Gerar Descrição em PPC

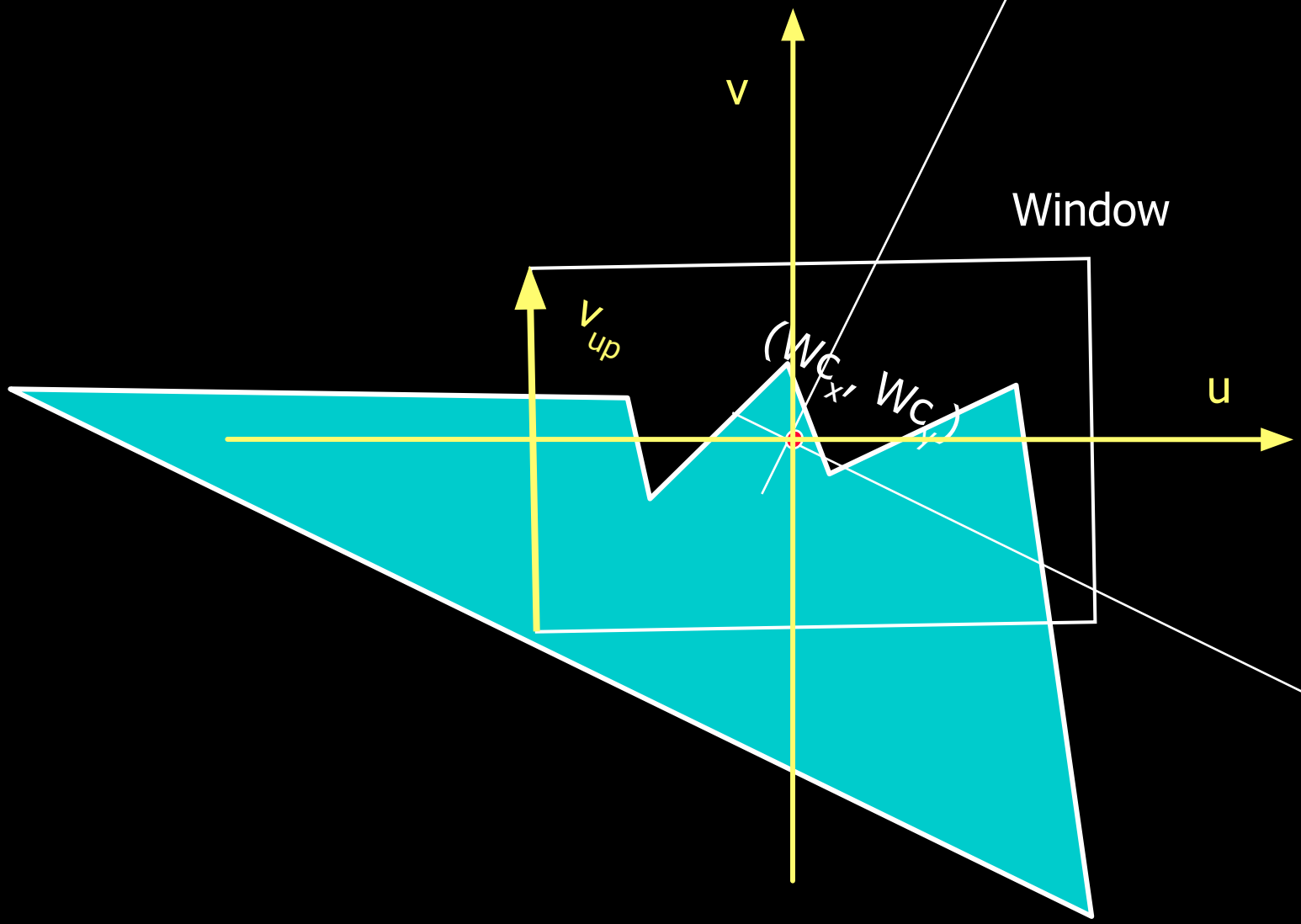
0. Crie ou mova a Window onde desejar.
1. Translade Wc para a origem
   - Transladar o mundo de [$-Wc_x$, $-Wc_y$]
2. Determine v_{up} e o ângulo de v_{up} com Y
3. Rotacione o mundo de forma a alinhar v_{up} com o eixo Y
   - Rotacione o mundo por $-\theta(Y, v_{up})$
4. Armazene as coordenadas CPP 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}

Parte I: Computação Gráfica Básica - Implementação de um Sistema Gráfico Interativo



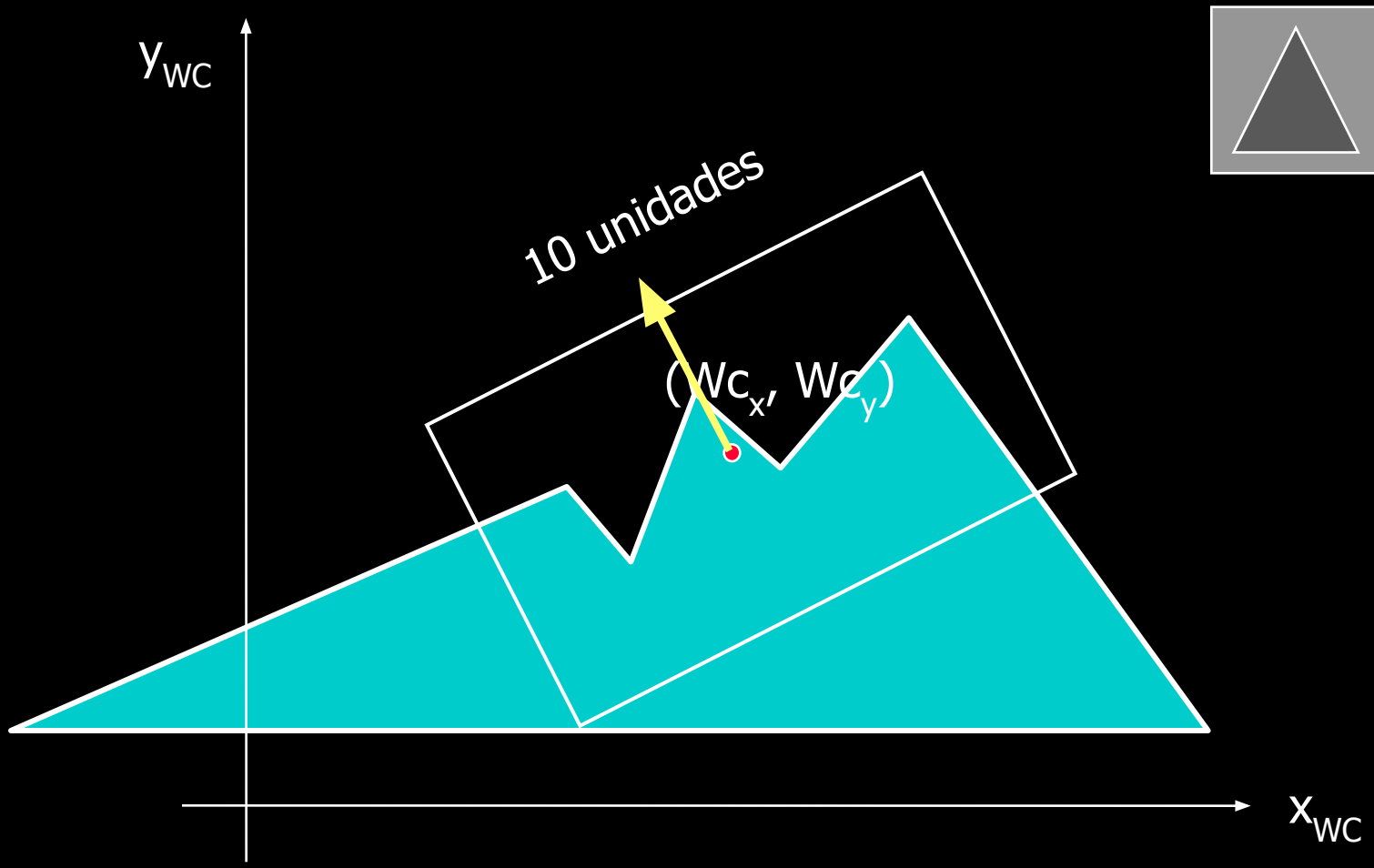
Como representar e usar o PPC ?

- 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.

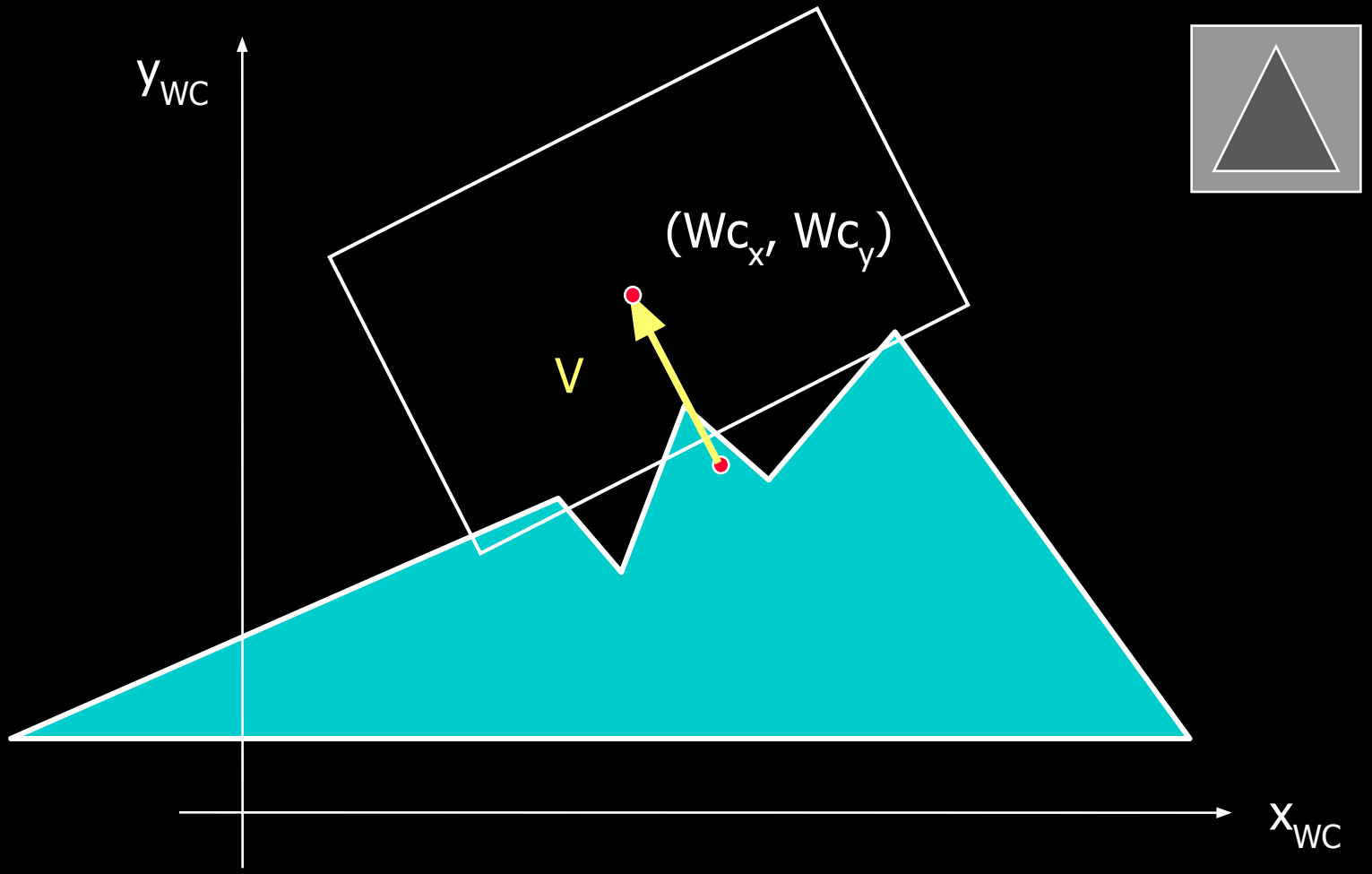
Como eu navego no Mundo usando o PCC ?

- 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.
- Aplique o algoritmo Gerar Descrição em PPC.

Deslocamento Linear



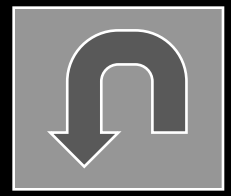
Deslocamento Linear



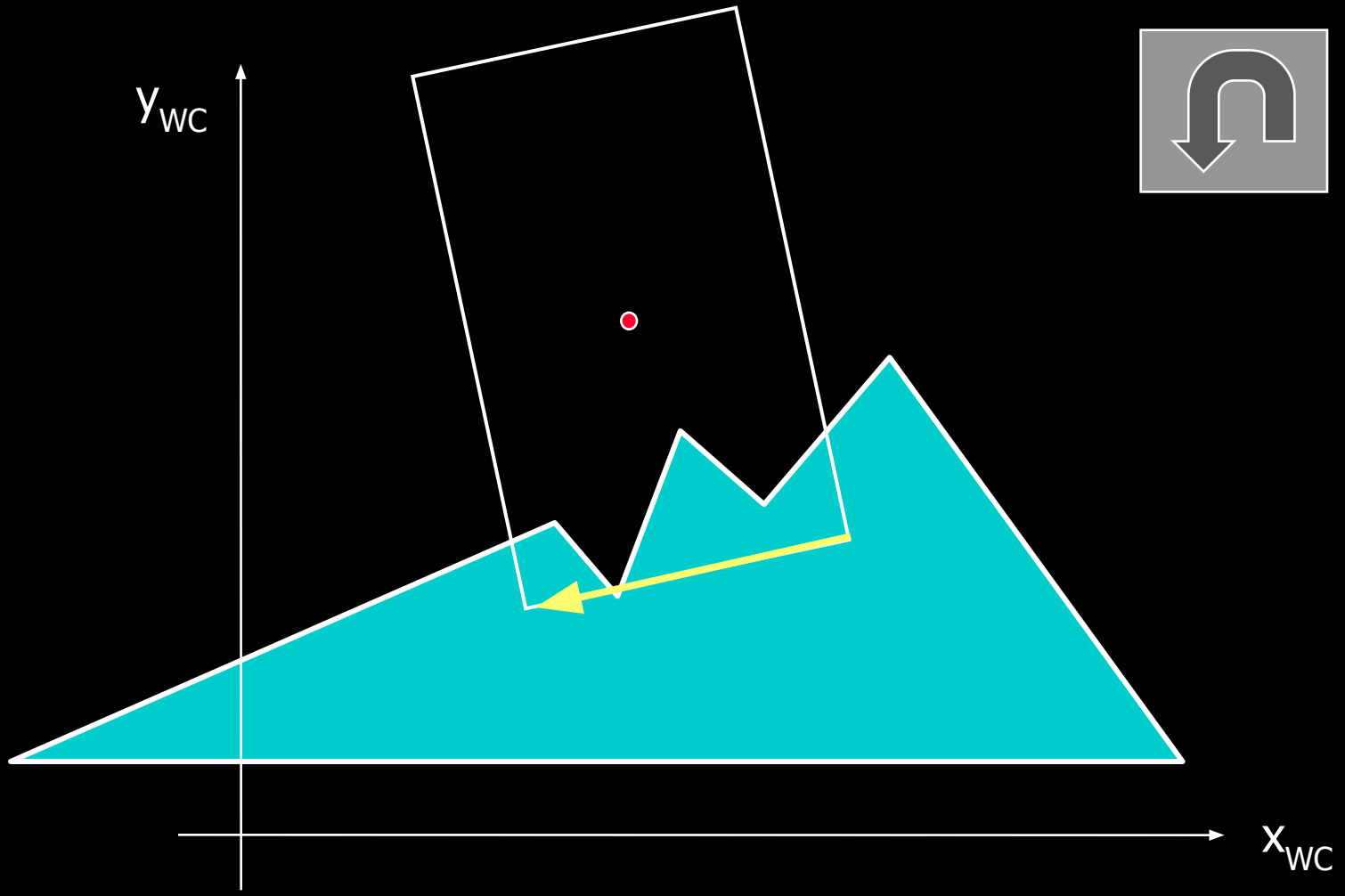
Como implementar a translação da window quando estiver em uma orientação não paralela aos eixos ?

- 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.

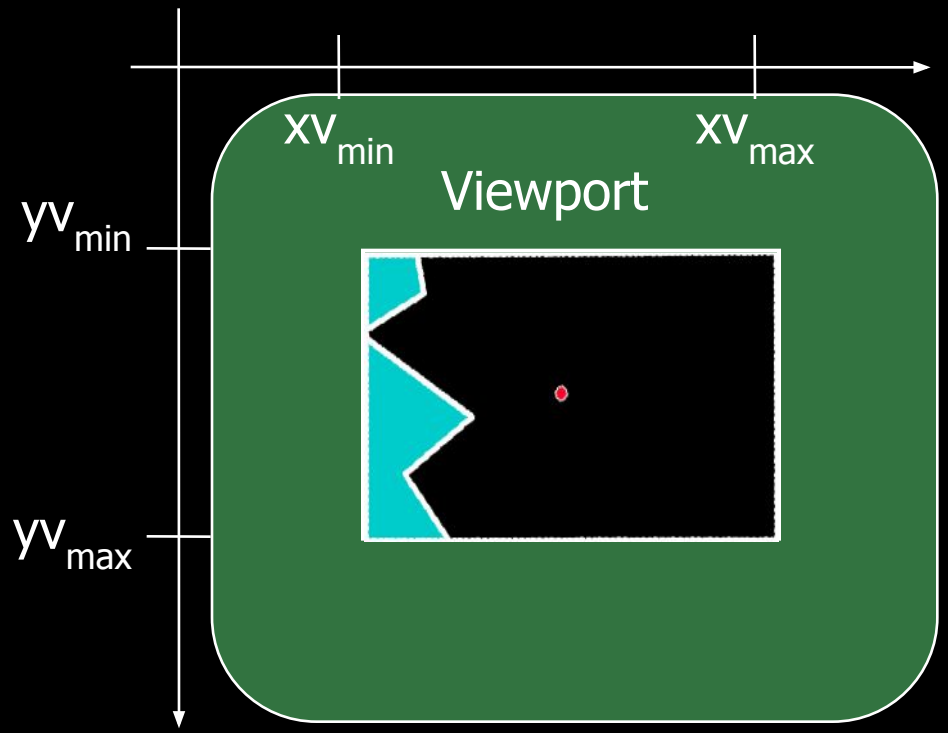
Rotação

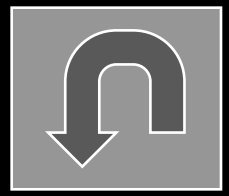


Rotação



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)

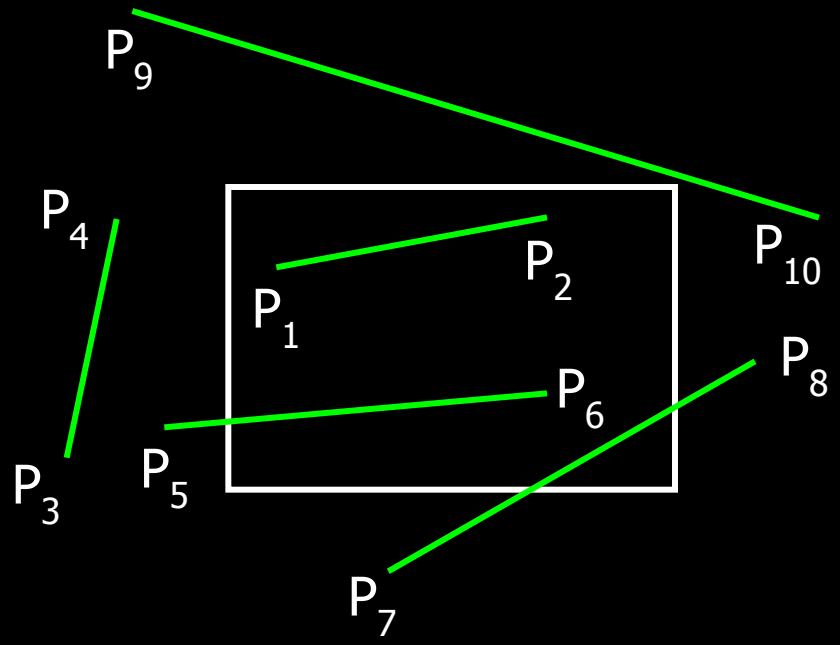
- Point clipping (p/clip window retangular)
  - Processo rápido e muito simples. O ponto que deve ser apresentado na viewport é aquele para o qual as inequações abaixo são satisfeitas:

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

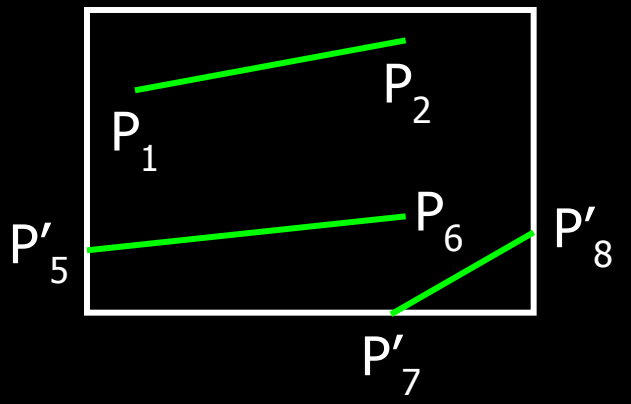
  - Implementado por toda componente de superfície de desenho de linguagens de programação, como os subcanvas ou canvas

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

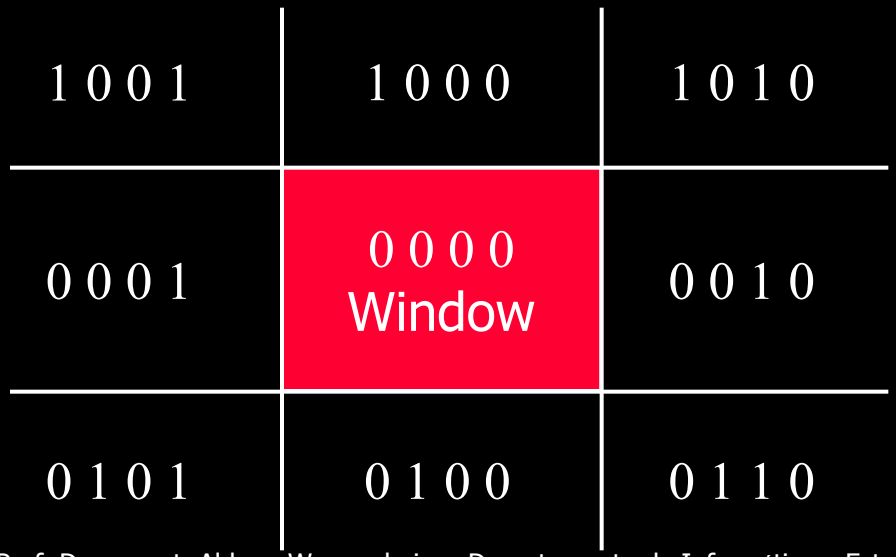
  $x = x_1 + u(x_2 - x_1)$

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

  - 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.



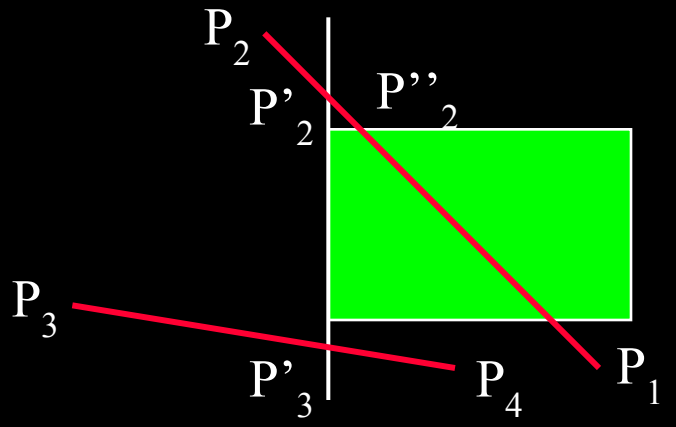
RC[1]: acima
RC[2]: abaixo
RC[3]: direita
RC[4]: esquerda

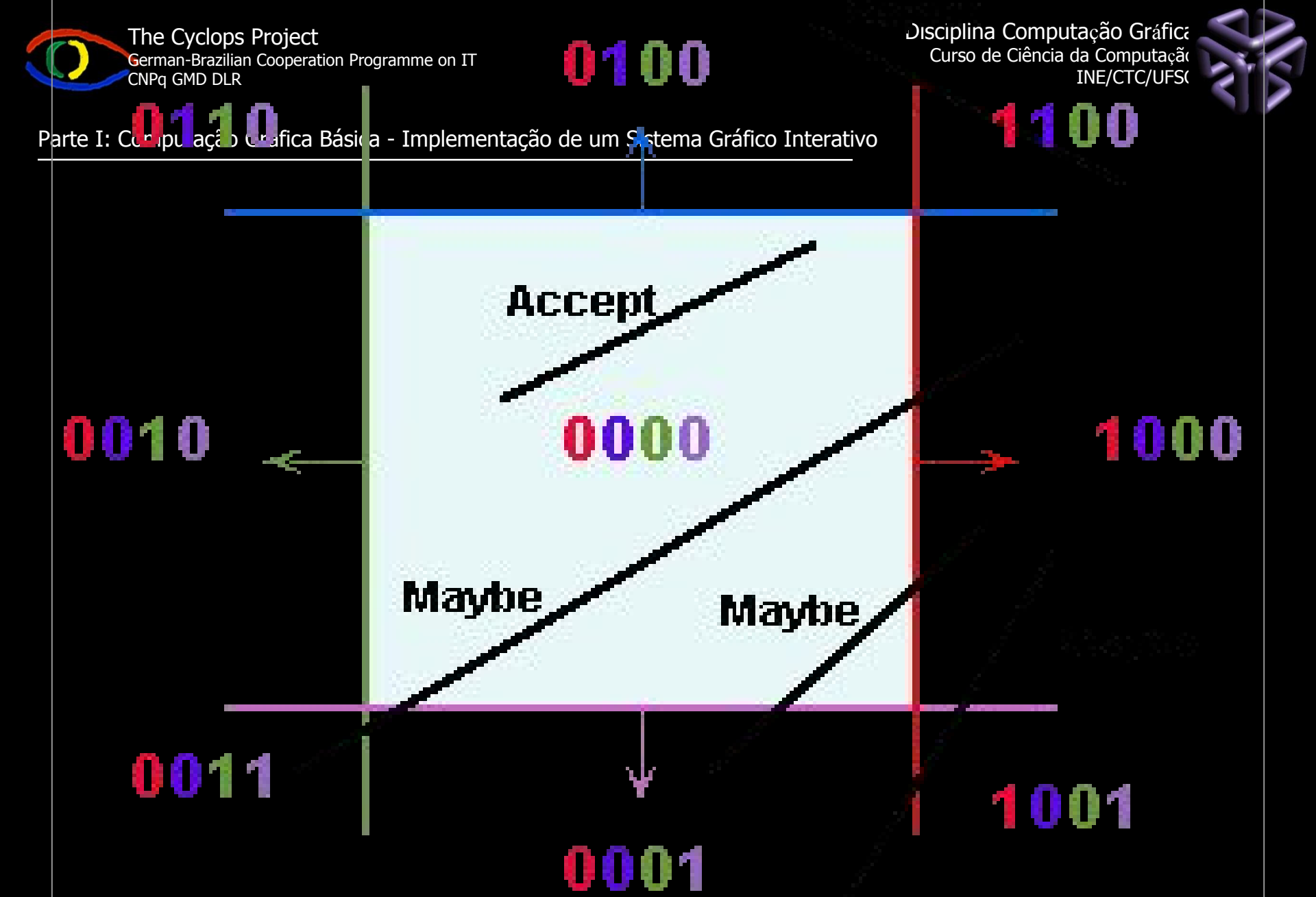
Recorte de Linhas de Cohen-Sutherland

- 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




0100
0110
1100
Accept
0010
0000
1000
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:

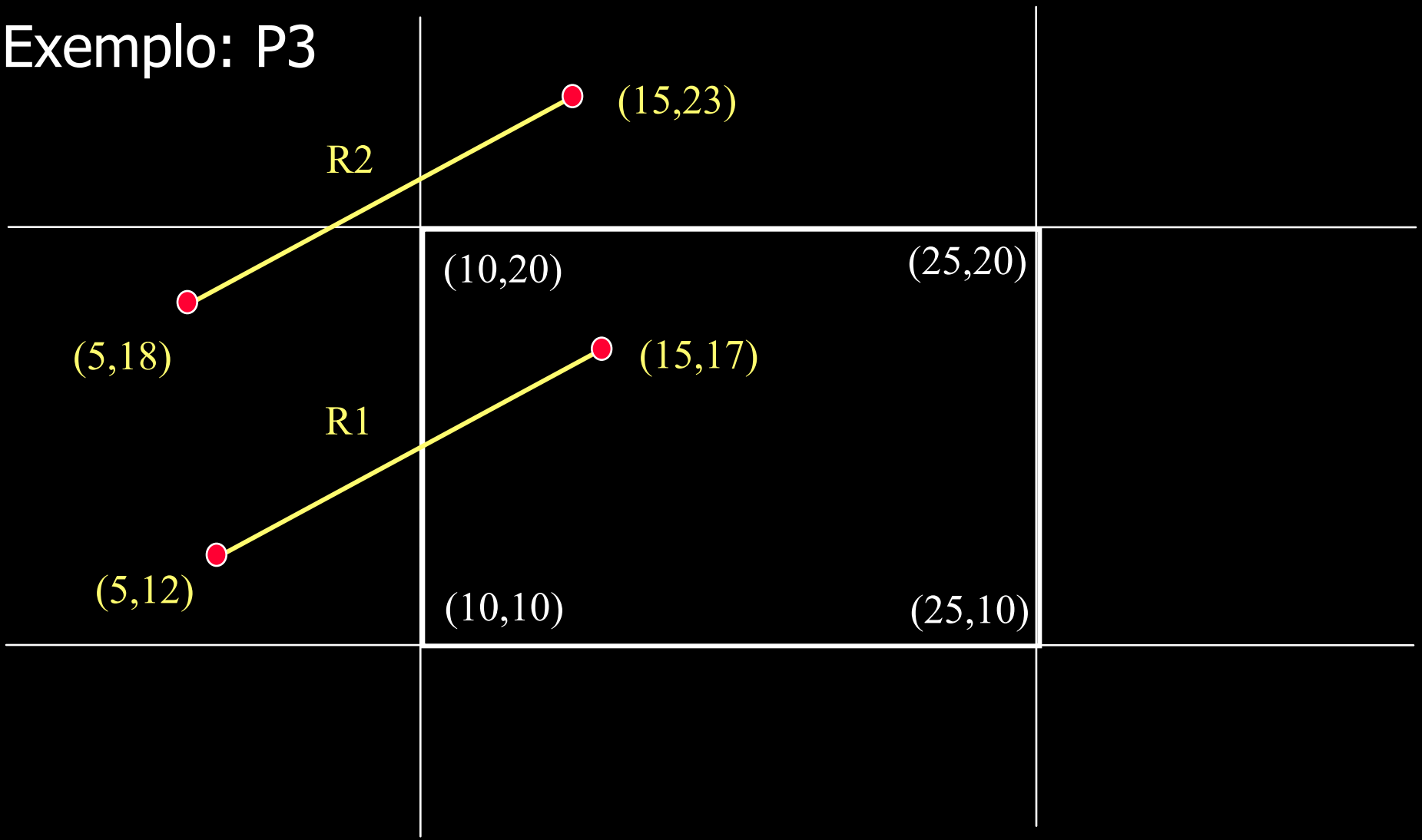
  Esquerda: $x_E, y = m^* (x_E - x_1) + y_1$; M diferente de 0

  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



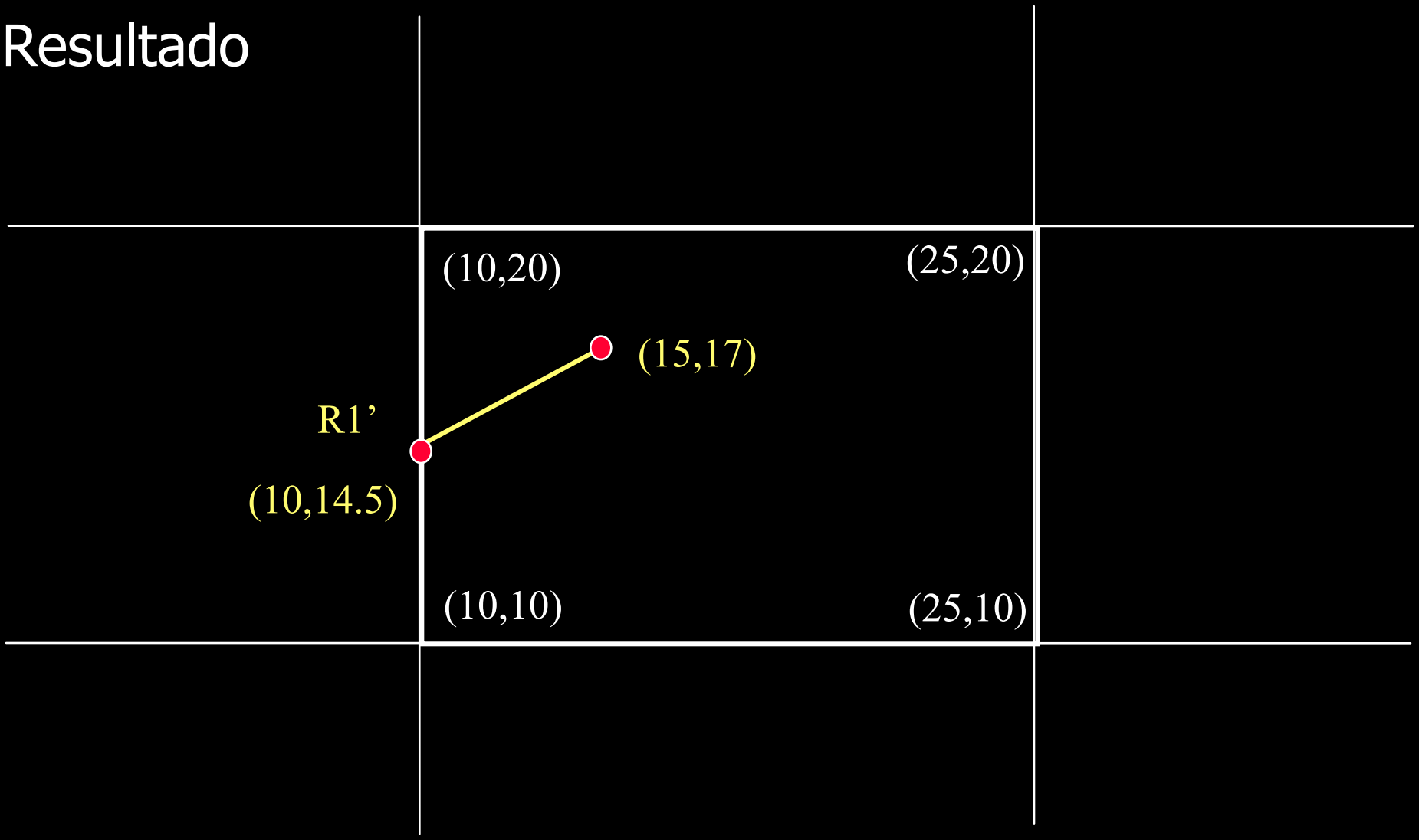
Clipando R1

- 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)

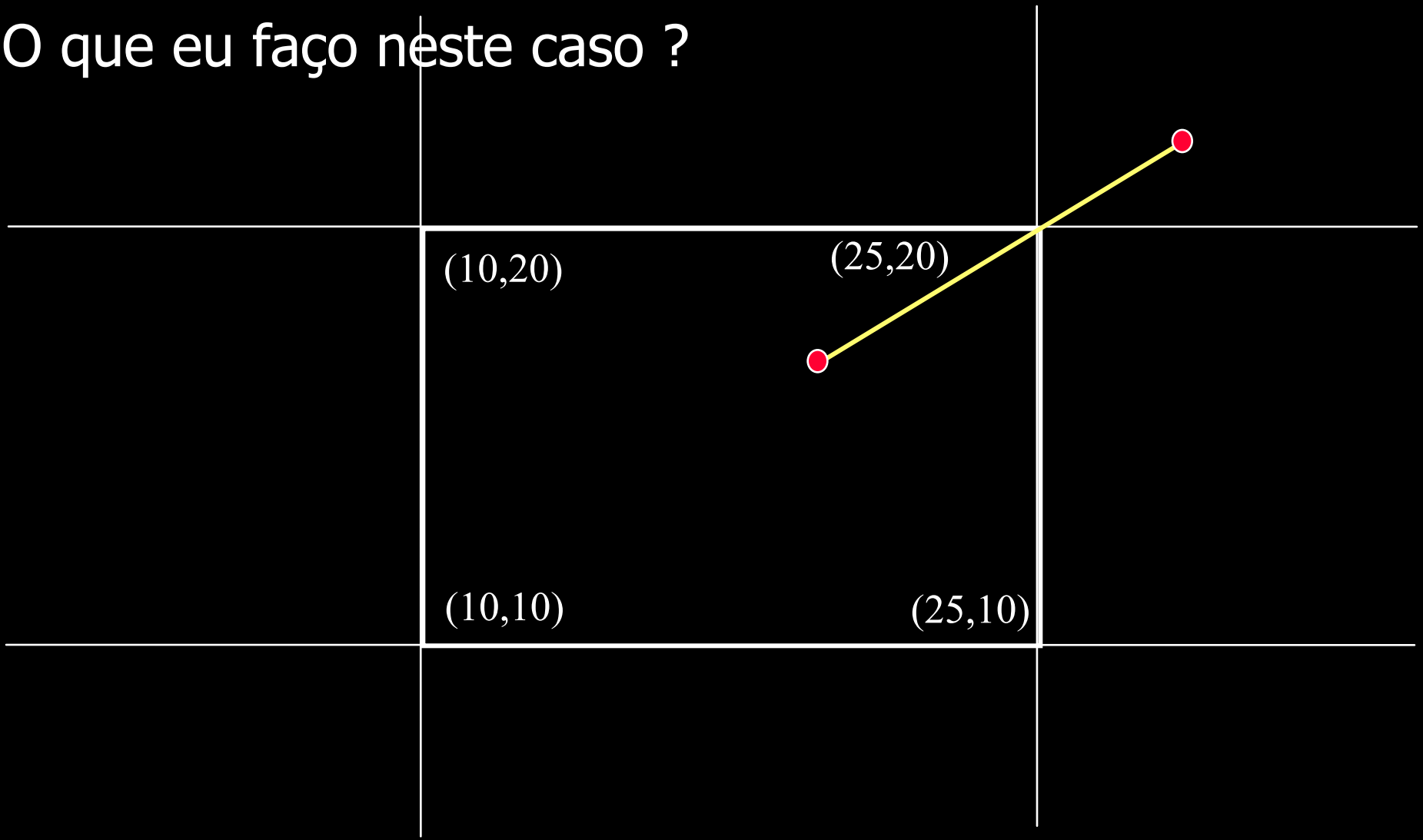
Clipando R2

- 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

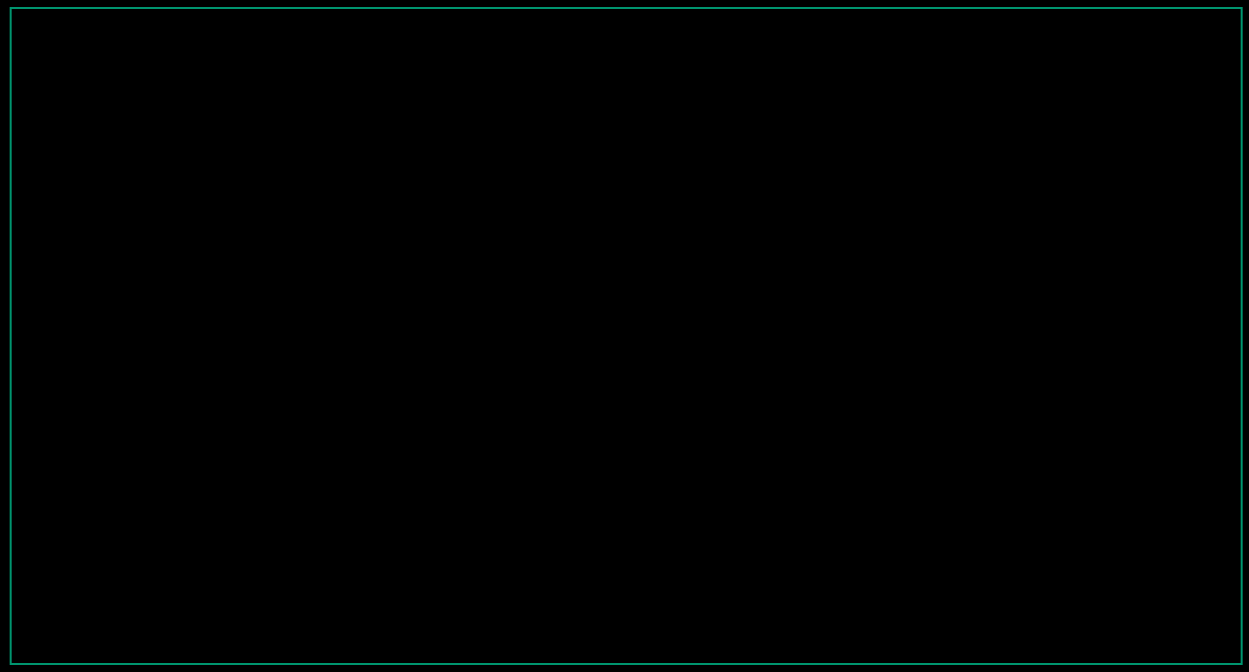
- Reescreva as equações paramétricas como segue:

- Sendo as condições de clipagem já vistas:

  - Cada uma dessas inequações pode então ser expressa como:

Liang-Barsky Line Clipping

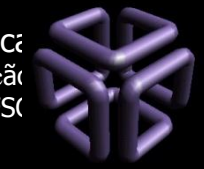
- Onde os parâmetros *p* e *q* são definidos como:



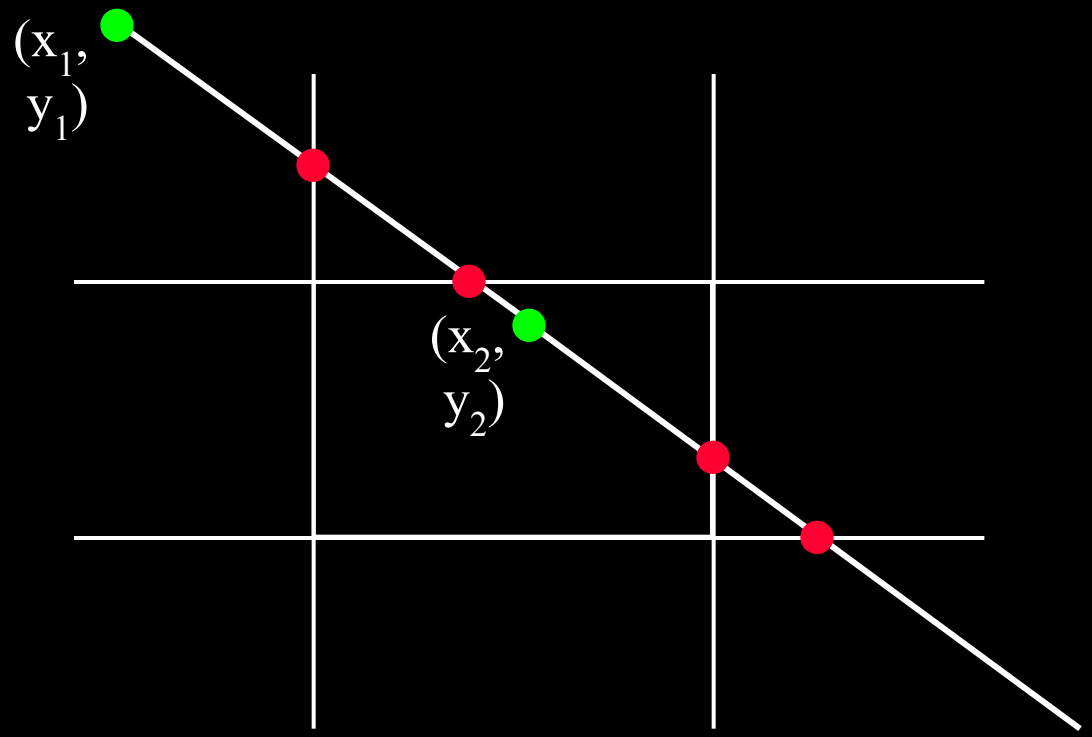
Liang-Barsky Line Clipping

Condições

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



Liang-Barsky Line Clipping



Qual a parte que está dentro ?

Liang-Barsky – Definição do Parâmetro ζ

- Os parâmetros ζ_1 and ζ_2 definem qual parte está dentro do retângulo:
  - O valor de ζ_1 : de fora para dentro (condição: $p_k < 0$)
  - O valor de ζ_2 : de dentro para fora (condição: $p_k > 0$)
  - Se $\zeta_1 > \zeta_2$, a linha está completamente fora.

Liang-Barsky Line Clipping

Calculamos ζ_1 para $p_1 < 0$, $p_3 < 0$

$\zeta_1 = \max(0, r_1, r_3)$

$= r_1$

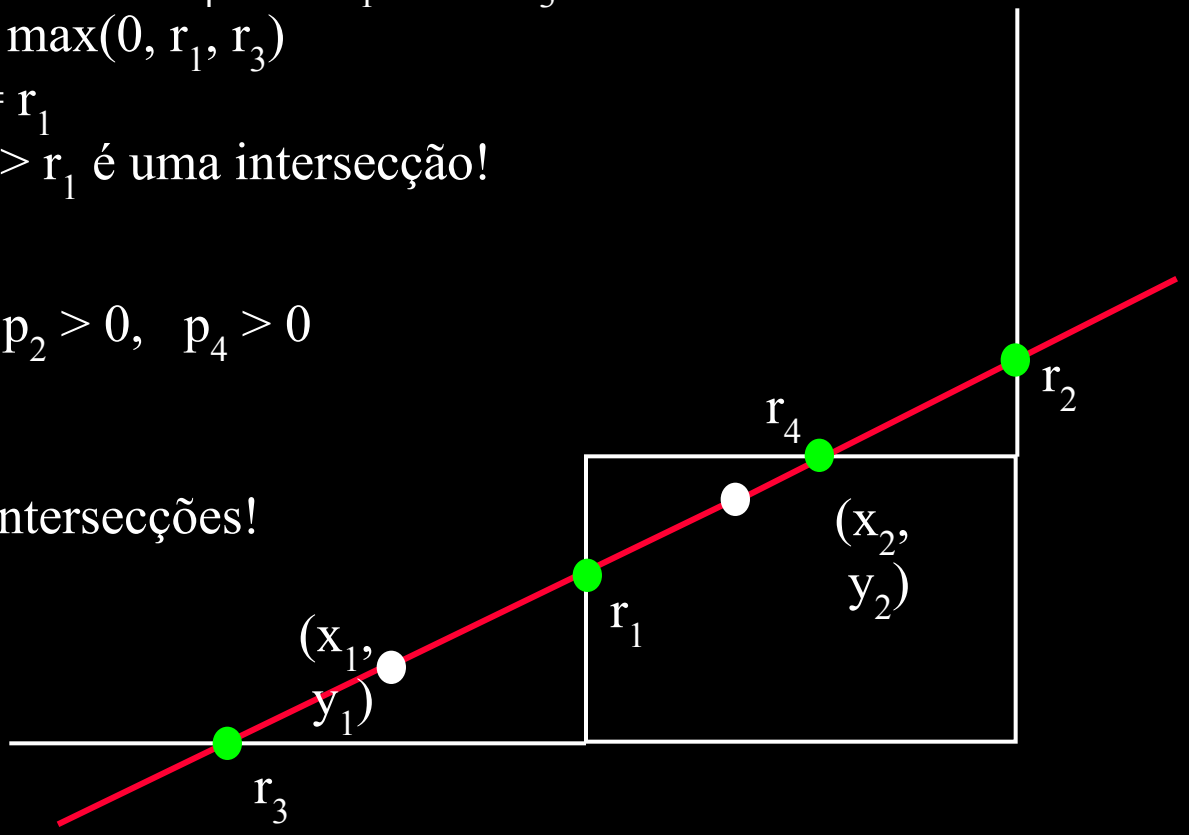
$\Rightarrow r_1$ é uma intersecção!

Calculamos ζ_2 para $p_2 > 0$, $p_4 > 0$

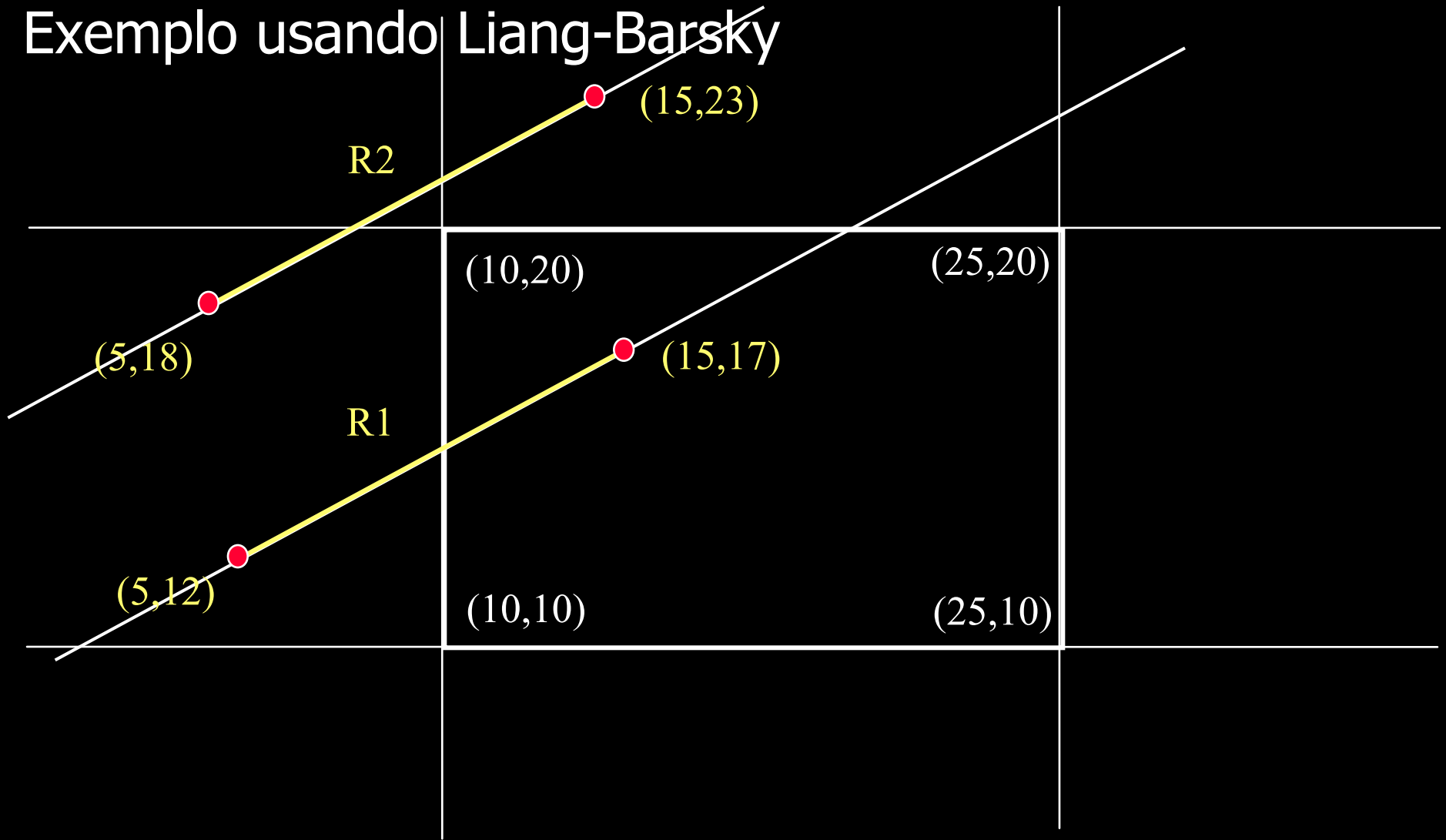
$\zeta_2 = \min(1, r_2, r_4)$

$= 1$

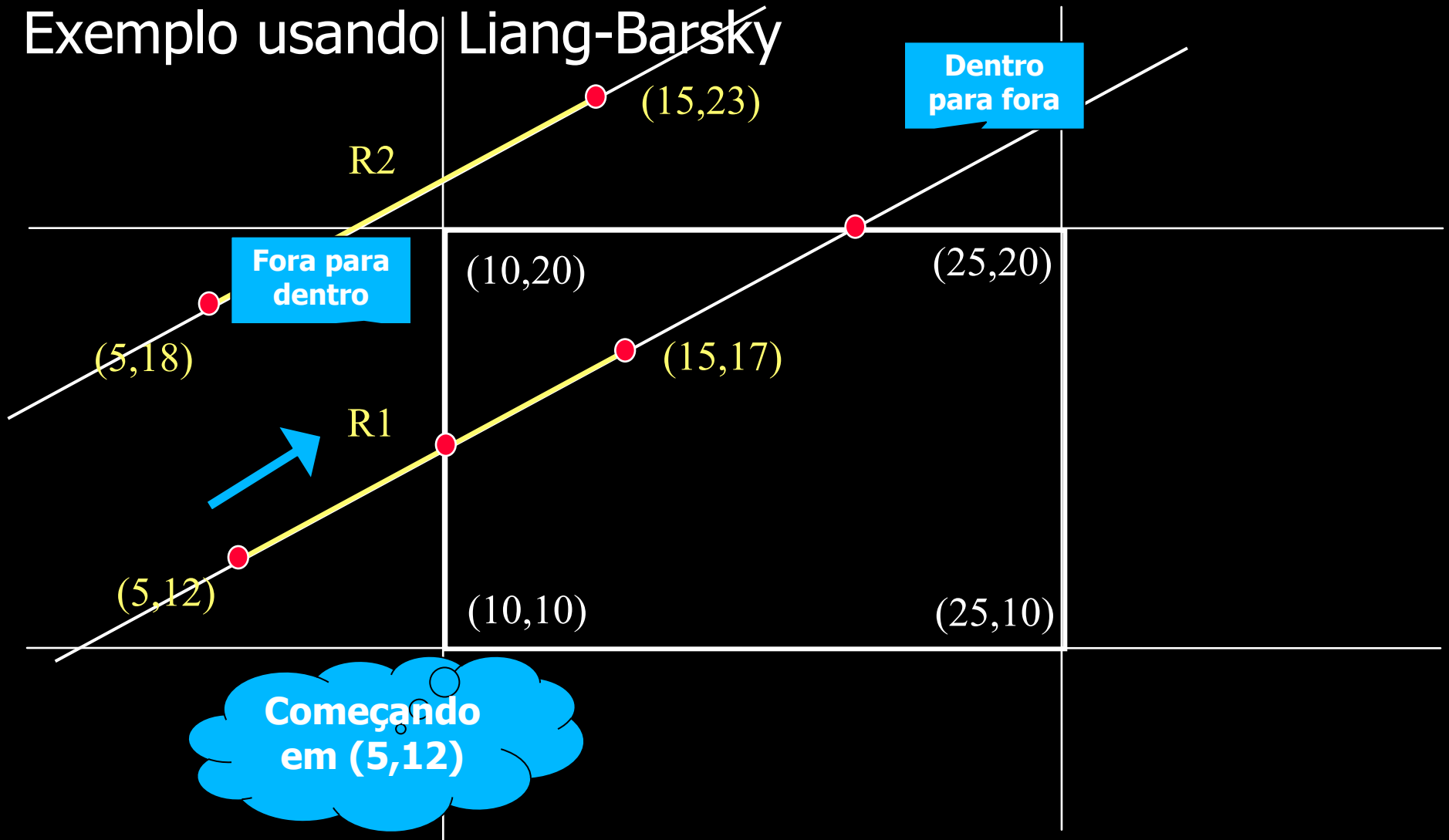
$\Rightarrow$ não há mais intersecções!



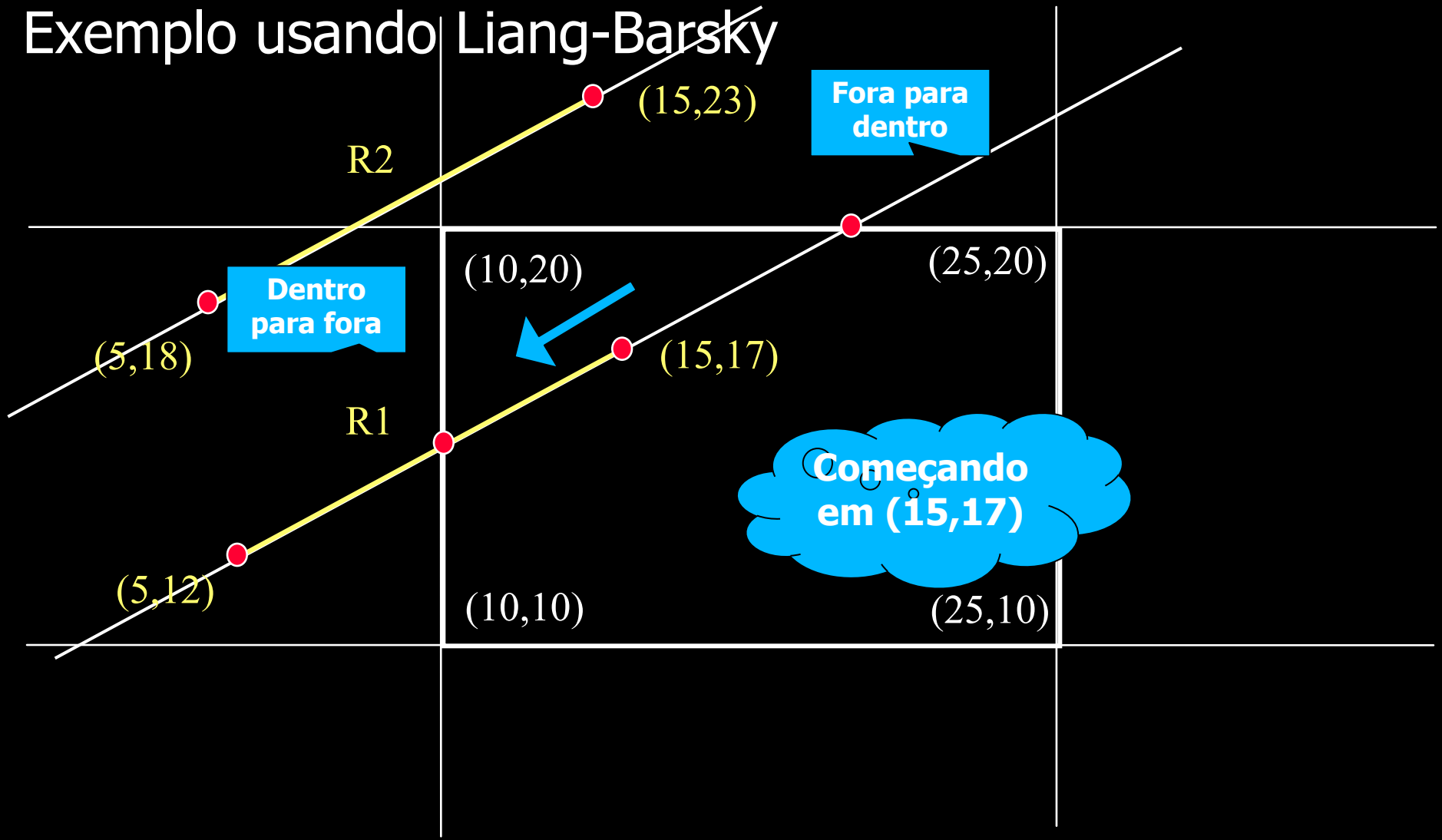
Exemplo usando Liang-Barsky



Exemplo usando Liang-Barsky



Exemplo usando Liang-Barsky



Calcular p e q para R1 (Reta inicia em (5,12))

- p1 = -Δx = -(15 - 5) = -10
- p2 = Δx = 15 - 5 = 10
- p3 = -Δy = -(17 - 12) = -5
- p4 = Δy = 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

Intersecção de Fora para Dentro:

Calcular $\zeta 1$ para R1 (Reta inicia em (5,12))

- $p_1 = -10 < 0$
- $p_3 = -5 < 0$

- $r_1 = q_1/p_1 = -5/-10 = 0.5$
- $r_3 = q_3/p_3 = 2/-5 = -0.4$
- $\zeta_1 = \max(0, r_1, r_3) = \max(0, 0.5, -0.4) = 0.5$

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)

Intersecção de Fora para Dentro:

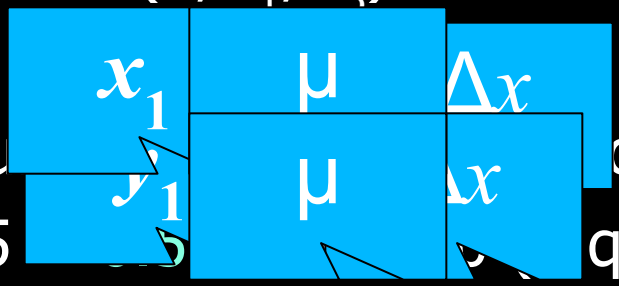
Calcular ζ1 para R1 (Reta inicia em (5,12))

- $p_1 = -10 < 0$
- $p_3 = -5 < 0$

Neste caso os valores negativos são p_1 e p_3

- $r_1 = q_1/p_1 = -5/-10 = 0.5$
- $r_3 = q_3/p_3 = 2/-5 = -0.4$
- $\zeta_1 = \max(0, r_1, r_3) = \max(0, 0.5, -0.4) = 0.5$

Substitu ca:



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

Intersecção de Dentro para Fora: Calcular ζ2 para R1 (Reta inicia em (5,12))

- $p_2 = 10 > 0$
- $p_4 = 5 > 0$

Neste caso os valores positivos são p_2 e p_4

- $r_2 = q_2/p_2 = 20/10 = 2$
- $r_4 = q_4/p_4 = 8/5 = 1.6$
- $\zeta_2 = \min(1, r_2, r_4) = \min(1, 2, 1.6) = 1$

- Como ζ_2 resulta 1, rejeitamos o cálculo de novos valores de dentro para fora.

Calcular p e q para R1 (Reta inicia em (15,17))

- p1 = -Δx = -(5 - 15) = 10
- p2 = Δx = 5 - 15 = -10
- p3 = -Δy = -(12 - 17) = 5
- p4 = Δy = 12 - 17 = -5
- q1 = x1 - x_{wmin} = 15 - 10 = 5
- q2 = x_{wmax} - x1 = 25 - 15 = 10
- q3 = y1 - y_{wmin} = 17 - 10 = 7
- q4 = y_{wmax} - y1 = 20 - 17 = 3

Calcular ζ1 para R1 (Reta inicia em (15,17))

- $p_2 = -10 < 0$
- $p_4 = -5 < 0$

Neste caso os valores negativos são p_2 e p_4

- $r_2 = q_2/p_2 = 10/-10 = -1$
- $r_4 = q_4/p_4 = 3/-5 = -0,6$
- $\zeta_1 = \max(0, r_2, r_4) = \max(0, -1, -0,6) = 0$

- Como ζ_1 resulta 0, rejeitamos o cálculo de novos valores de fora para dentro.

Calcular ζ2 para R1 (Reta inicia em (15,17))

- $p_1 = 10 > 0$
- $p_3 = 5 > 0$

Neste aso os valores positivos são p_1 e p_3

- $r_1 = q_1/p_1 = 5/10 = 0,5$
- $r_3 = q_3/p_3 = 7/5 = 1.4$
- $\zeta_2 = \min(1, r_1, r_3) = \min(1, 0,5, 1.4) = 0,5$

Substituindo na eq.paramétrica:

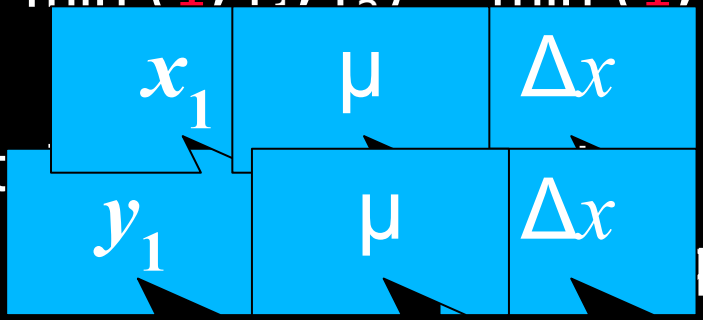
- x = 15 + 0.5 * -10 = 10 (o que nós já sabíamos)
- y = 17 + 0.5 * -5 = 14.5 (o que nós não sabíamos)

Calcular ζ2 para R1 (Reta inicia em (15,17))

- $p_1 = 10 > 0$
- $p_3 = 5 > 0$

Neste caso os valores positivos são p_1 e p_3

- $r_1 = q_1/p_1 = 5/10 = 0,5$
- $r_3 = q_3/p_3 = 7/5 = 1.4$
- $\zeta_2 = \min(1, r_1, r_3) = \min(1, 0,5, 1.4) = 0,5$



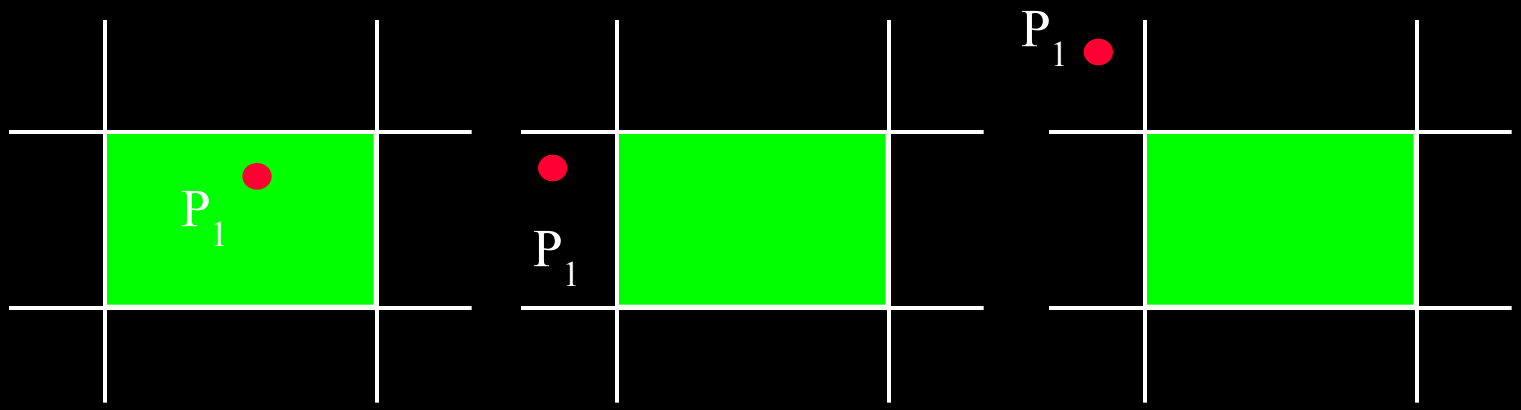
Substit : :

- x = que nós já sabíamos)
- y = 17 + 0.5 * -5 = 14.5 (o que nós não sabíamos)

Nicholl-Lee-Nicholl Line Clipping

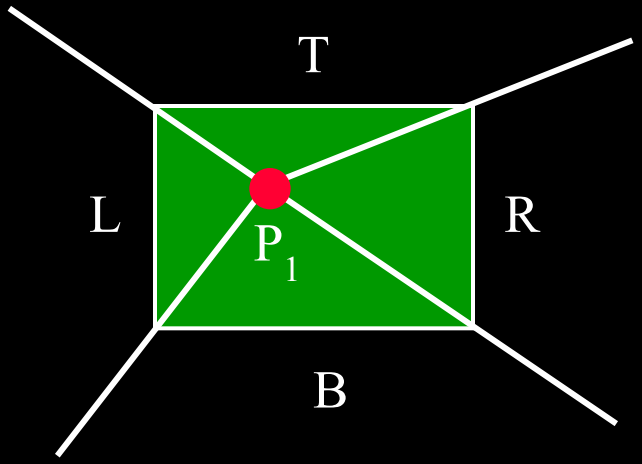
- Comparado aos algoritmos C-S e L-B:
  - NLN realiza menos comparações e divisões.
  - NLN pode ser aplicado somente a 2D clipping.
- O algoritmo NLN
  - Clipa uma linha com extremas P_1 e P_2
  - Passo 1: Determina P_1 para os nove quadrantes.
    - Somente três regiões são consideradas
    - Para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

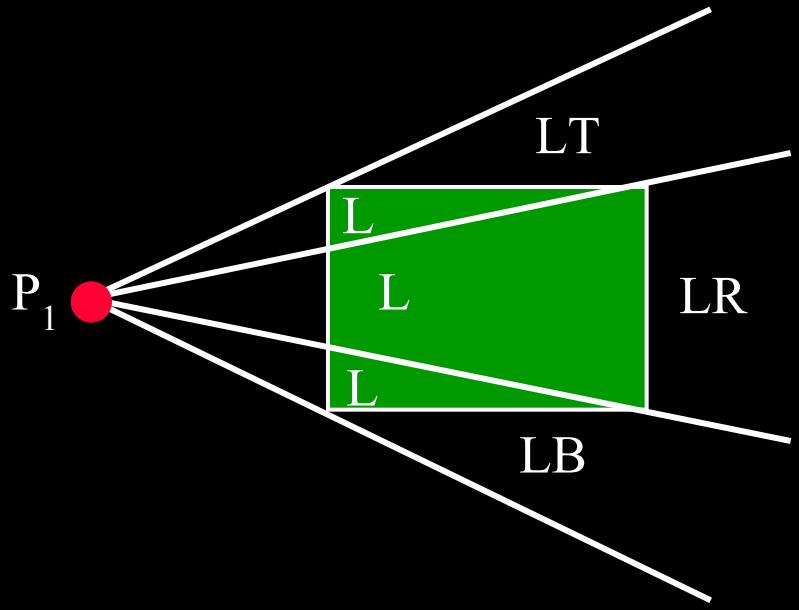
Nicholl-Lee-Nicholl Line Clipping: Passo 1a



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



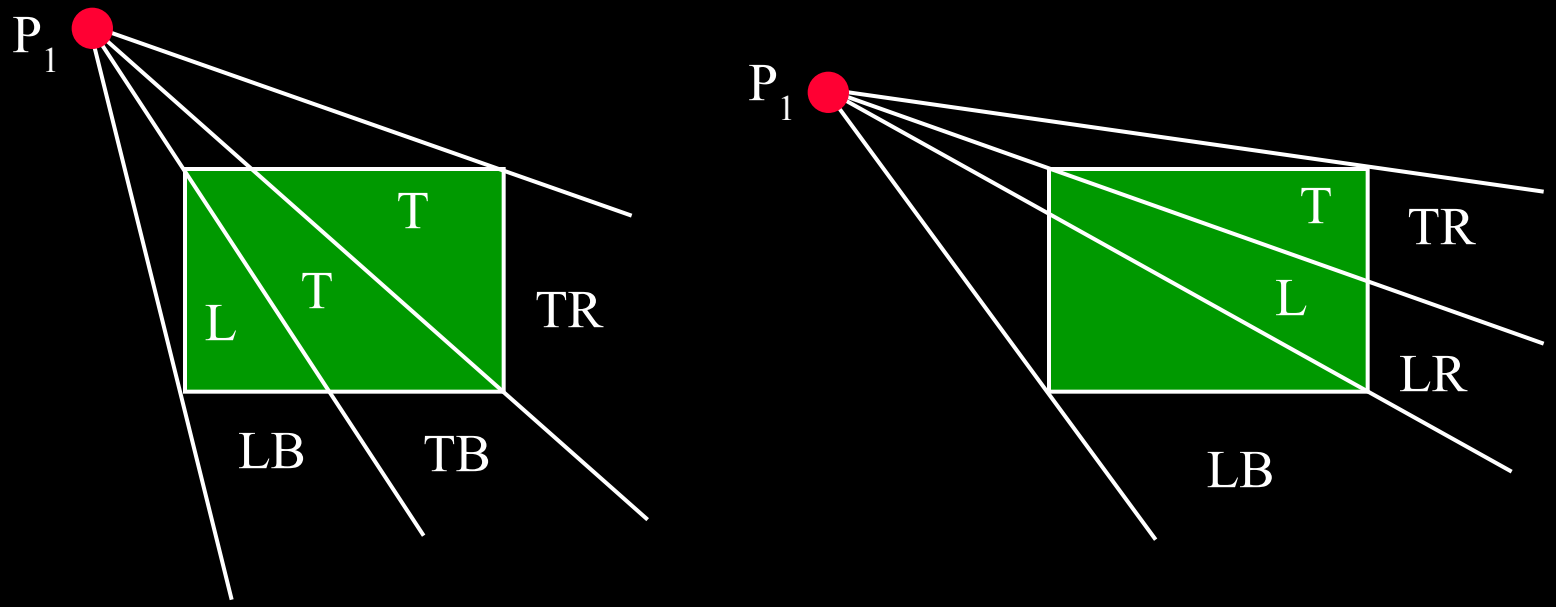


P_1 is inside the clip window and P_2 is outside

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

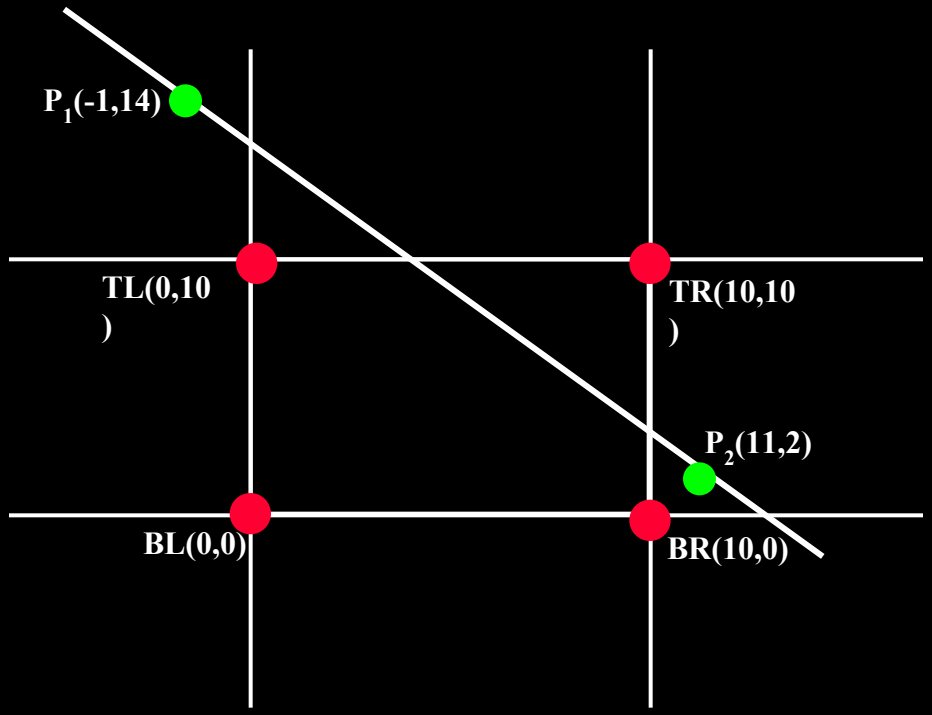


Nicholl-Lee-Nicholl Line Clipping

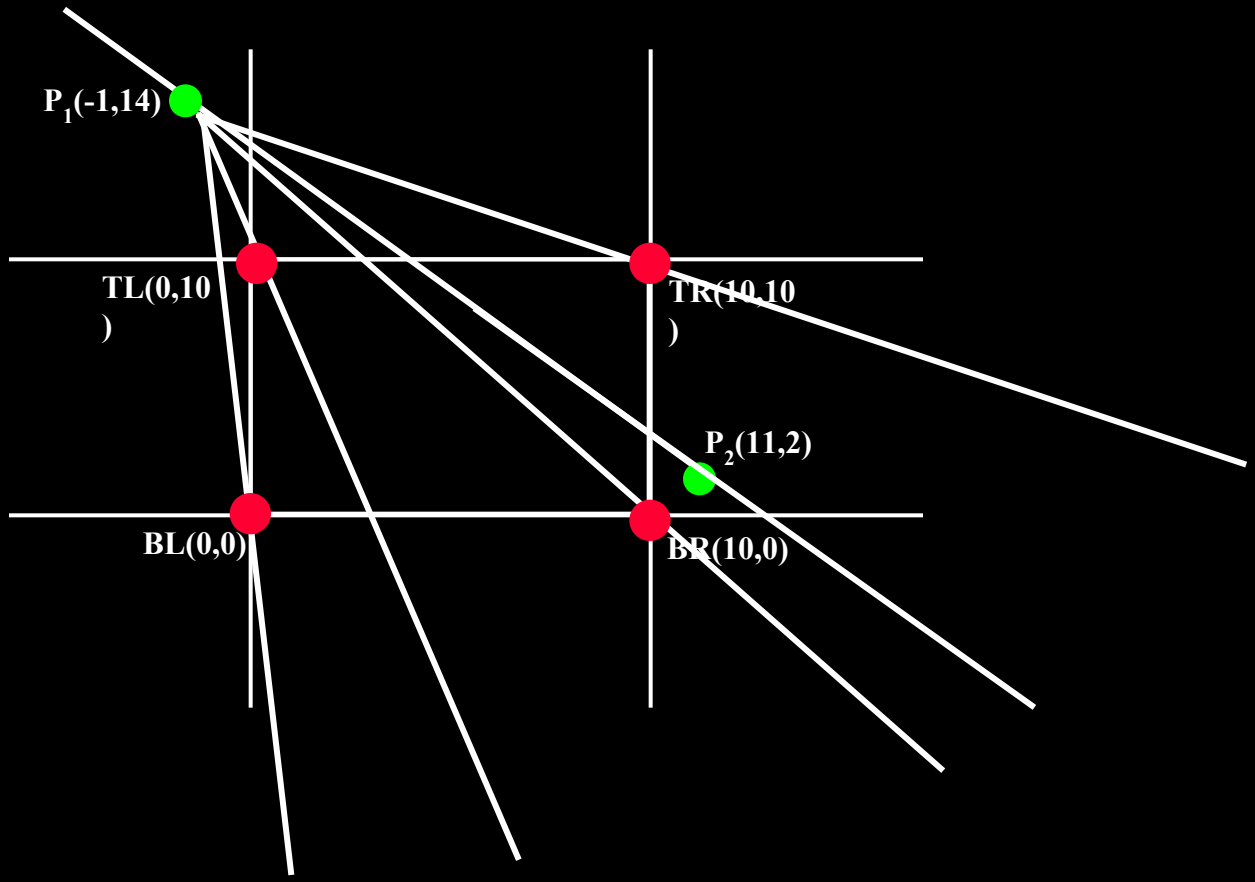
- Para determinar em qual região P_2 está:
  - Compare-se a inclinação da reta com as inclinações das retas hipotéticas calculadas.
    - Exemplo: P_1 está à esquerda, P_2 está em LT.

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

Exemplo usando Nicholl-Lee-Nicholl

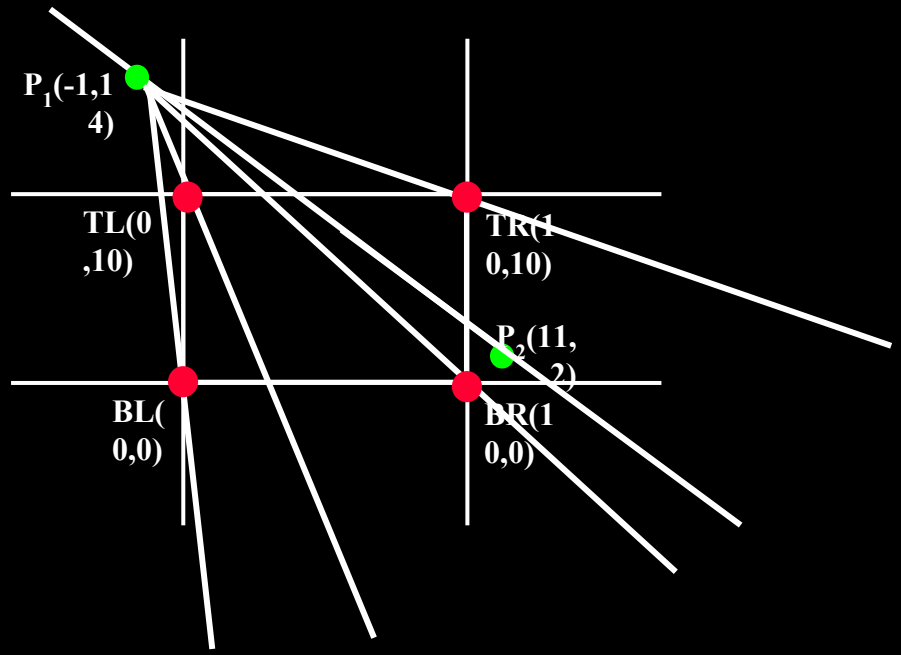


Exemplo usando Nicholl-Lee-Nicholl

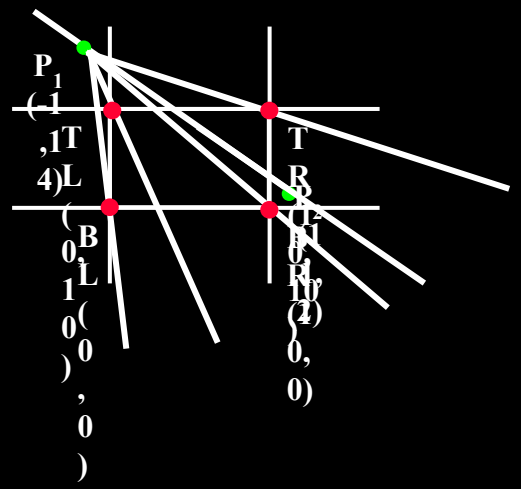


Exemplo usando Nicholl-Lee-Nicholl

- **Cálculo da inclinação das retas**
- **P1-BL =**
- **P1-TL =**
- **P1-TR =**
- **P1-BR =**
- **P1-P2 =**



Exemplo usando Nicholl-Lee-Nicholl



- Inclinação de **P1-P2** está entre
- **P1-TR** e
- **P1-BR**

logo haverá intersecções com

- **Norte** (y conhecido),
- **Leste** (x conhecido),

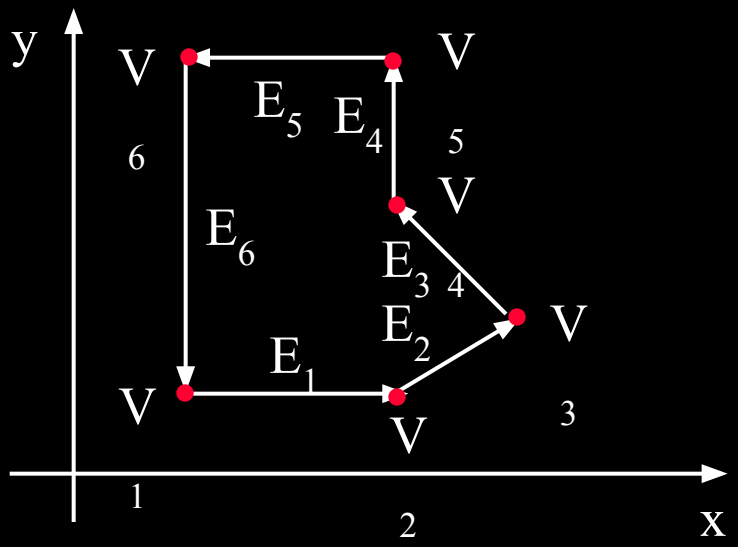
Obs.: necessitamos calcular

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

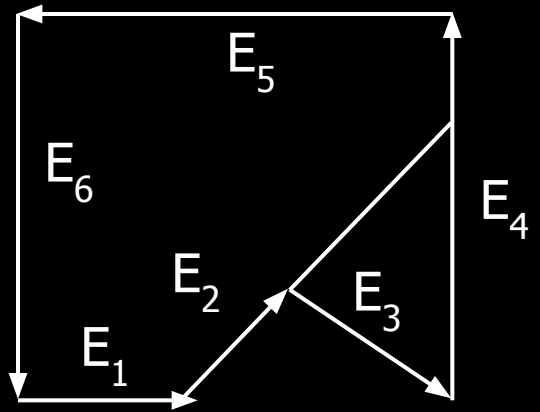
- Identifique se um polígono é Côncavo
  - Calcule o produto cruzado dos vetores de borda.



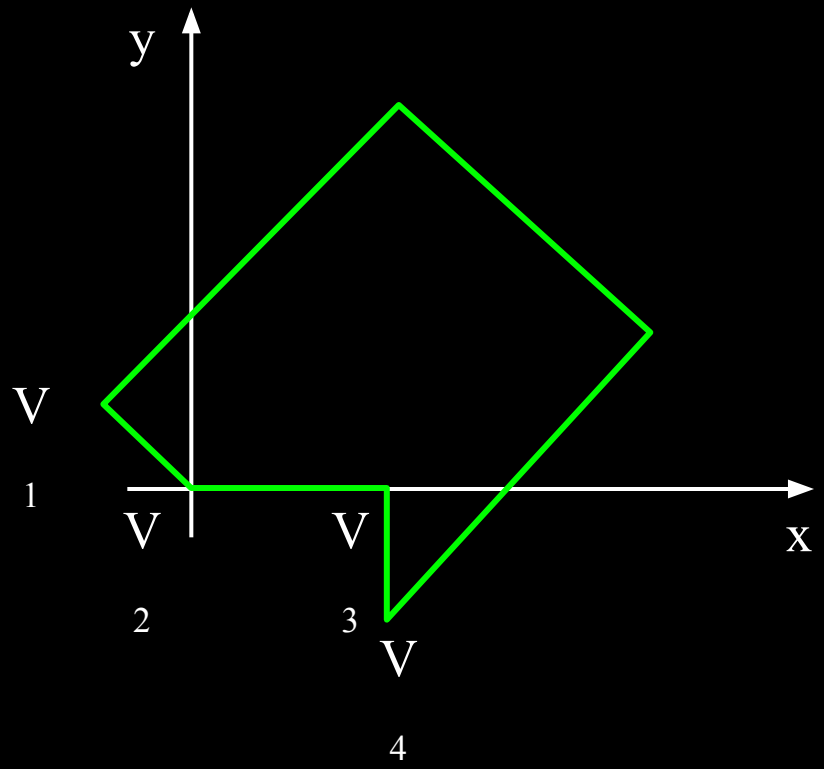
  - Uma componente z negativa resultante da multiplicação posicionada entre componentes positivas indica uma concavidade local.

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

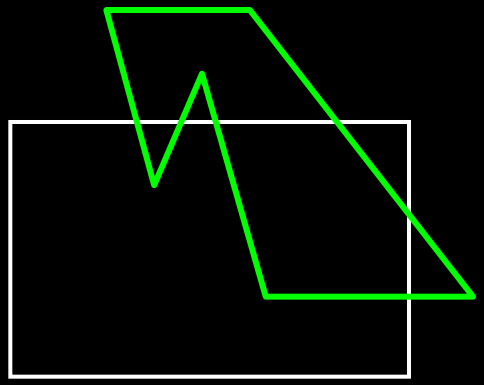


After rotating V_3 onto the x axis, V_4 is below the x axis.

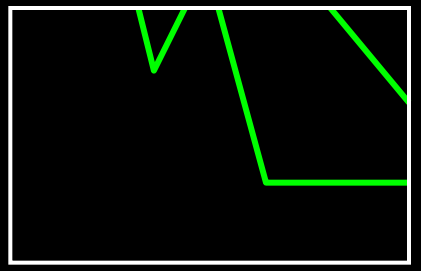
Split the polygon along the line of $\overline{V_2V_3}$.

Clipping de Polígonos

Clipping de Linhas



Antes do clipping



Depois do clipping

Resultam 2 polígonos distintos:

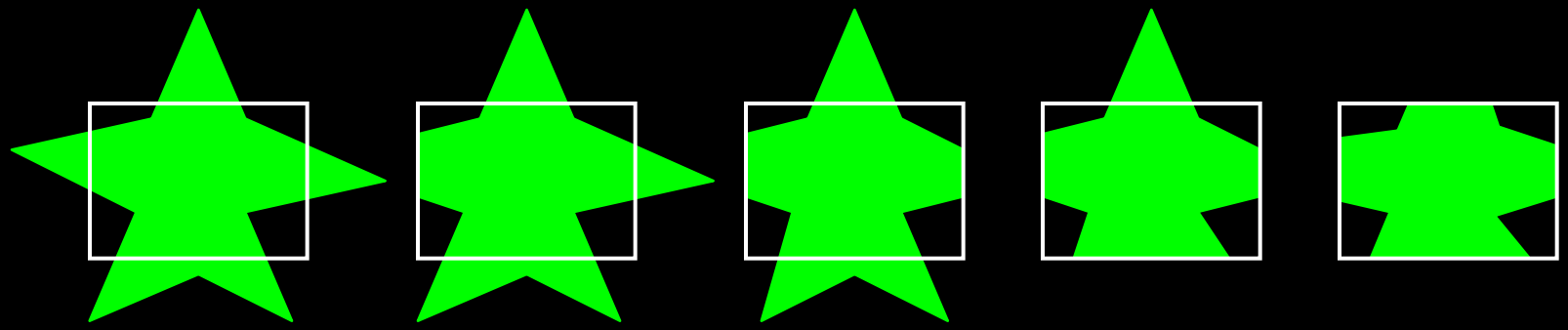
Clipping de Polígonos

Antes do clipping

Depois do clipping

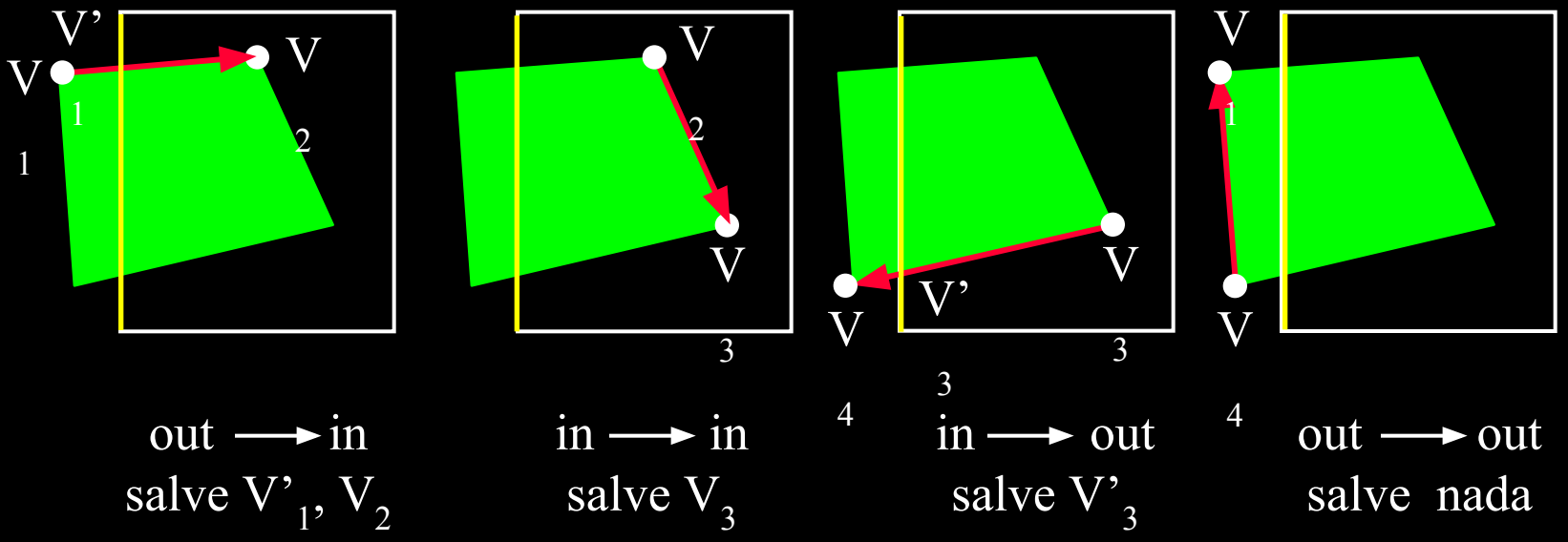
Clipping de Polígonos de Sutherland-Hodgeman

- 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:



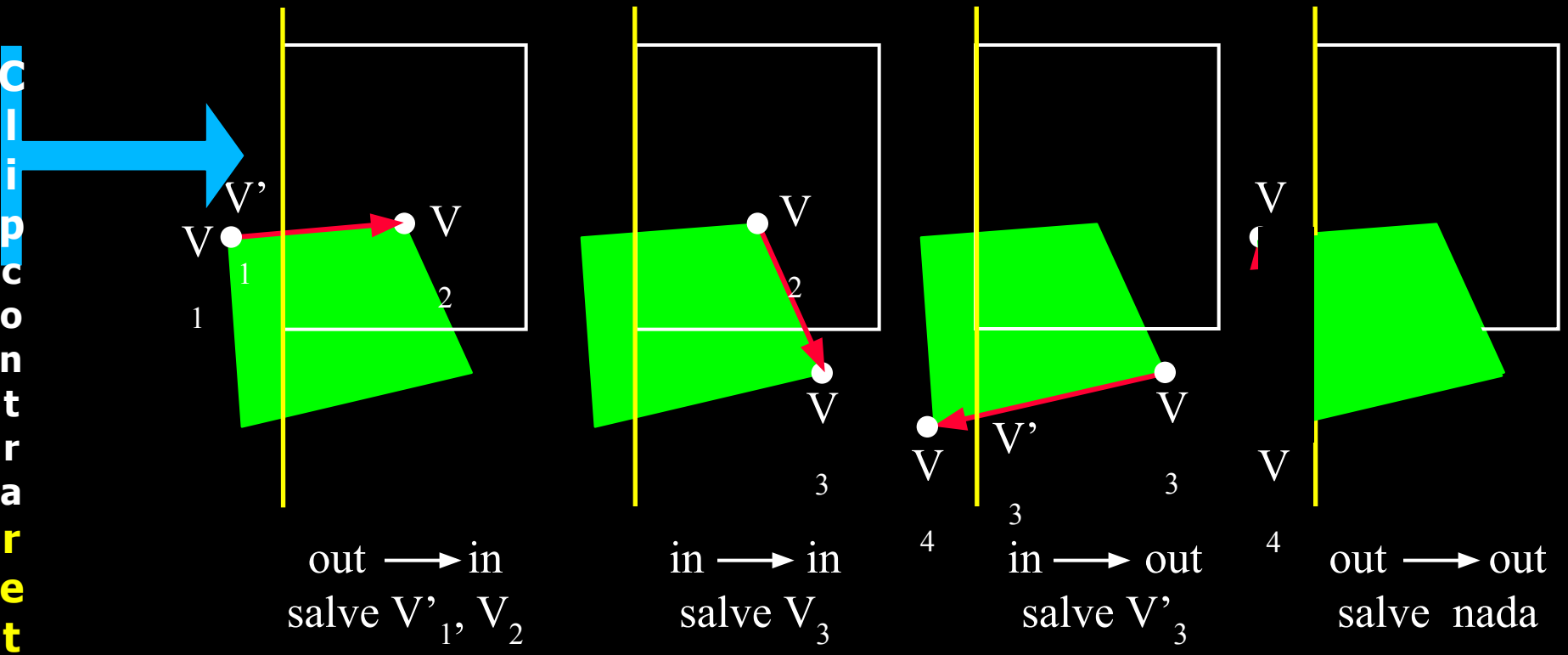
Clipping de Polígonos de Sutherland-Hodgeman

- 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 Sutherland-Hodgeman

- O que acontece nesta situação ?
  - Igual aos 4 casos anteriores?

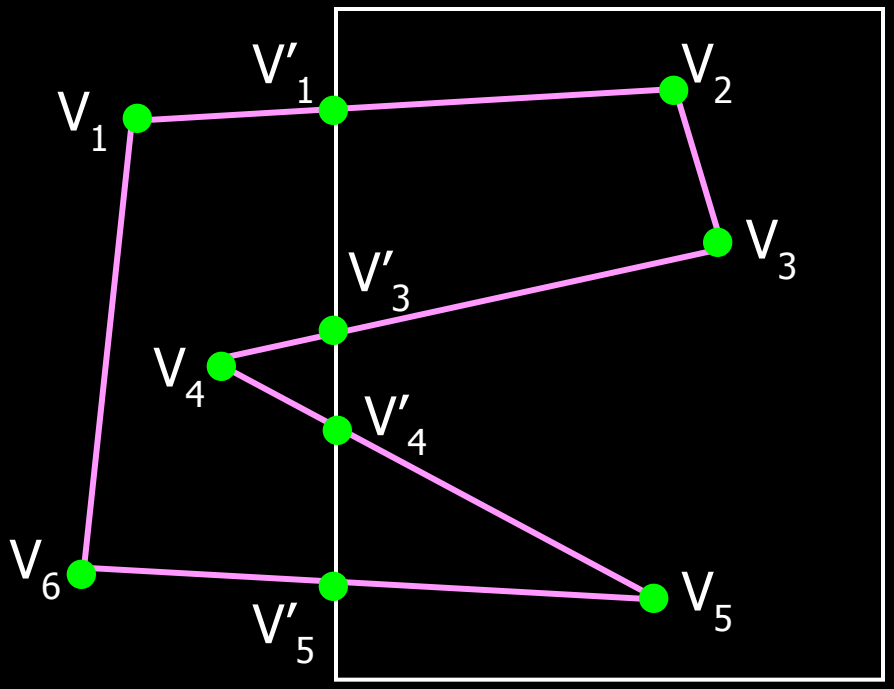
Clip contra eta o

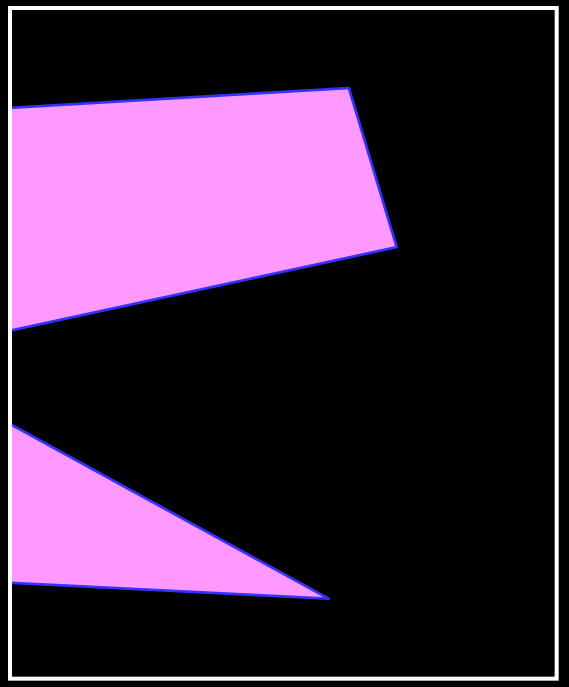


Clipping de Polígonos de Weiler-Atherton

- 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:
  - Para um par de vértices de fora para dentro, siga a borda do polígono;
  - Para um par de vértices de dentro para fora, siga a borda da window no sentido horário.

Clipagem de Polígonos de Weiler-Atherton





Depois do Clipping

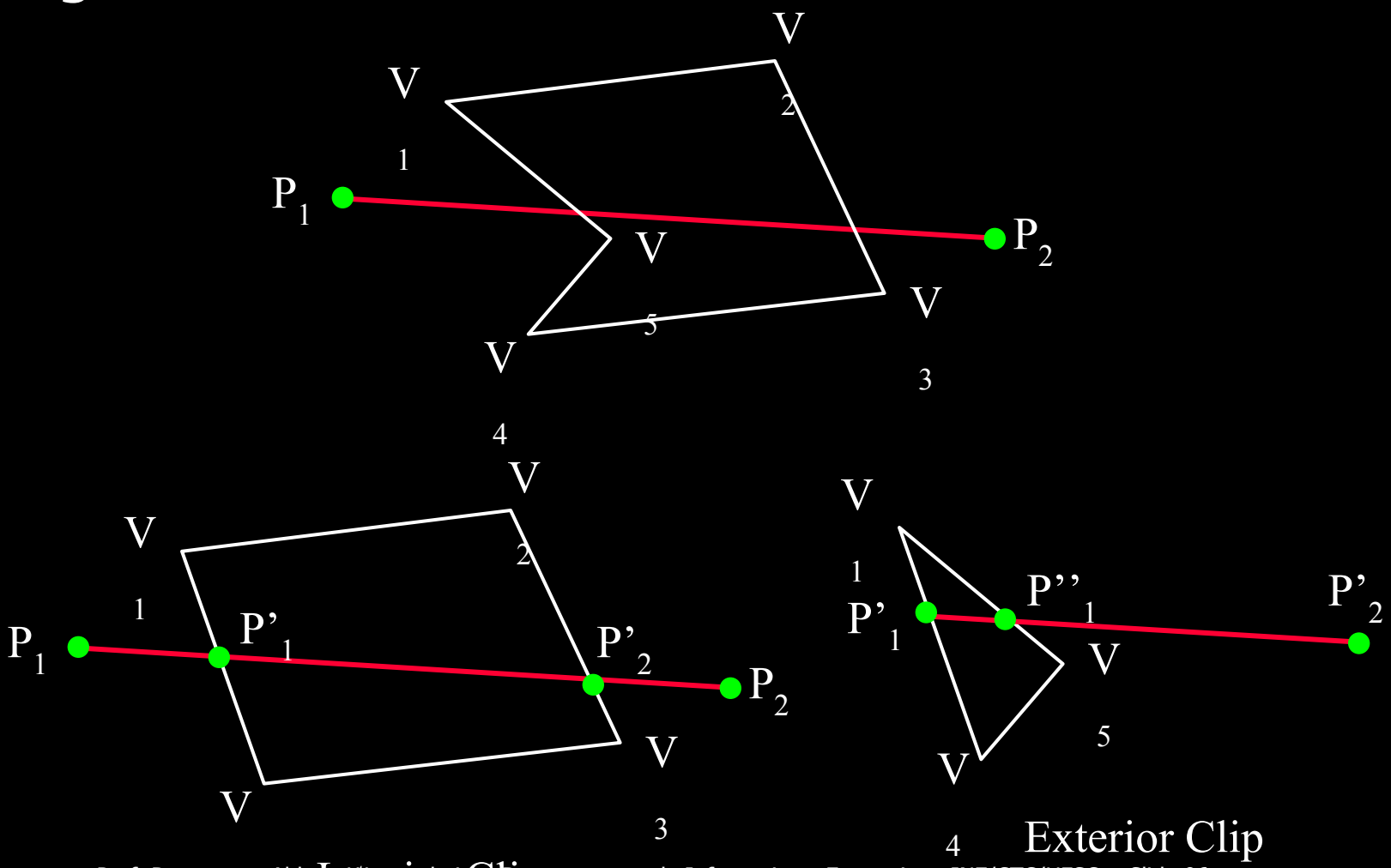
Outros Clippings

- 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



Trabalho: Clipping - parte #1

- Extenda seu sistema 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 sistema 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