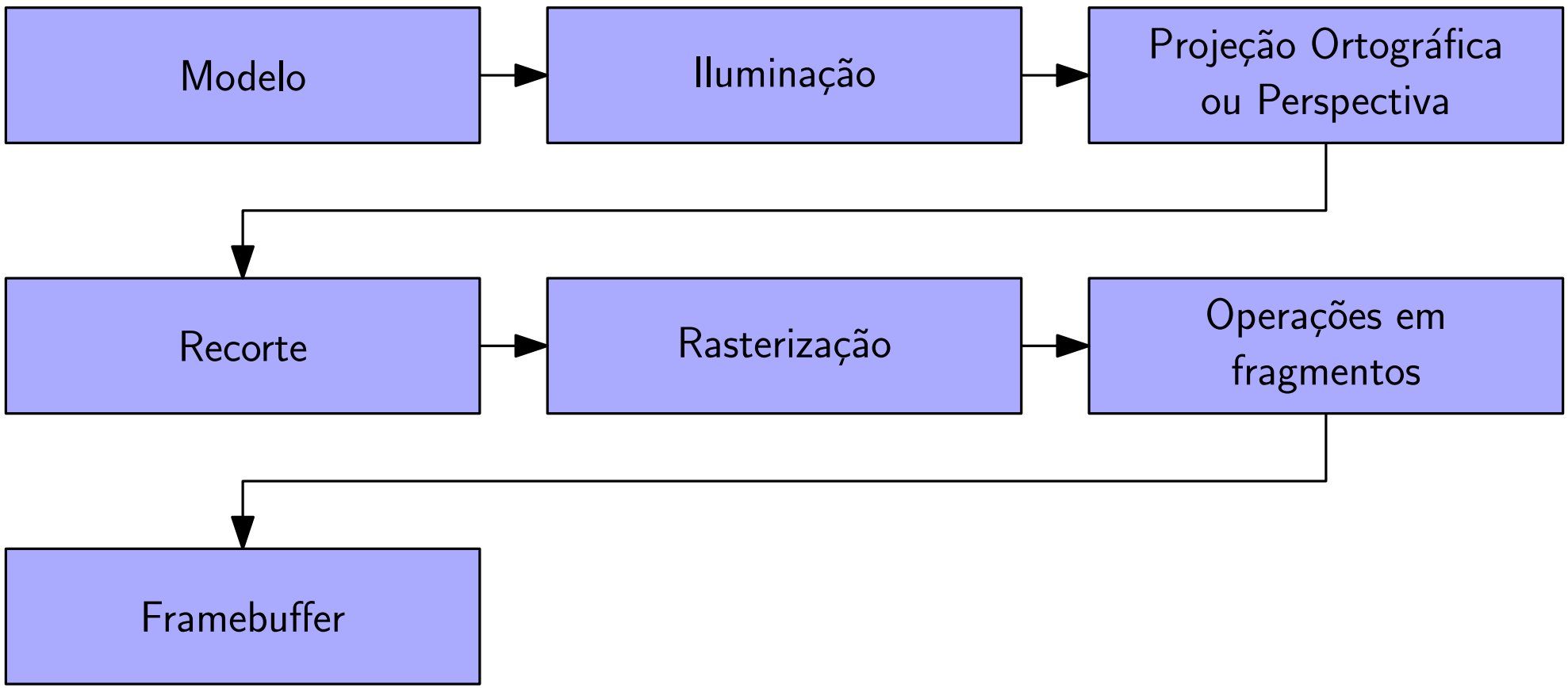
Computação Gráfica

Aula 27: Recorte

Vicente Helano Feitosa Batista Sobrinho
Faculdade Paraíso do Ceará
Sistemas de Informação
1o. semestre de 2011

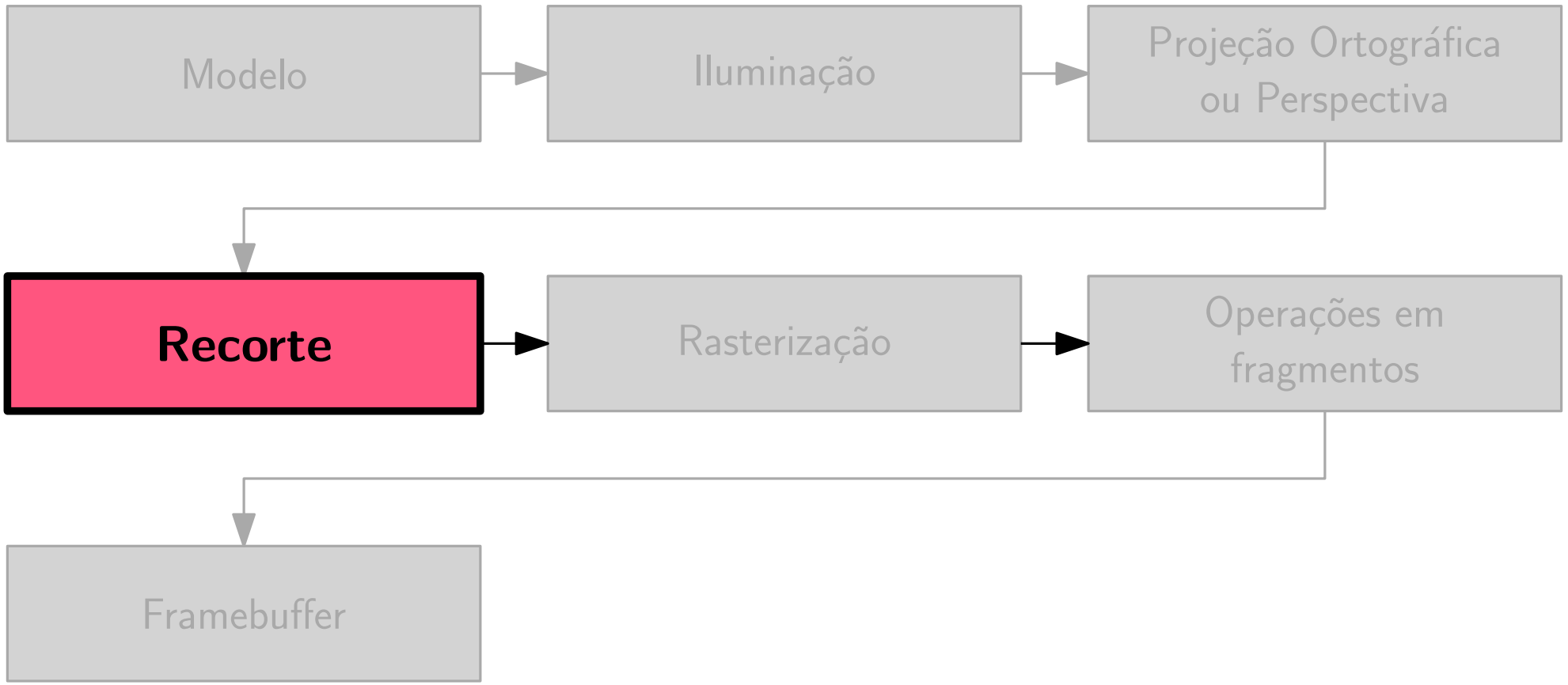
Faculdade Paraíso - CE

Pipeline gráfico (básico) de renderização por rasterização



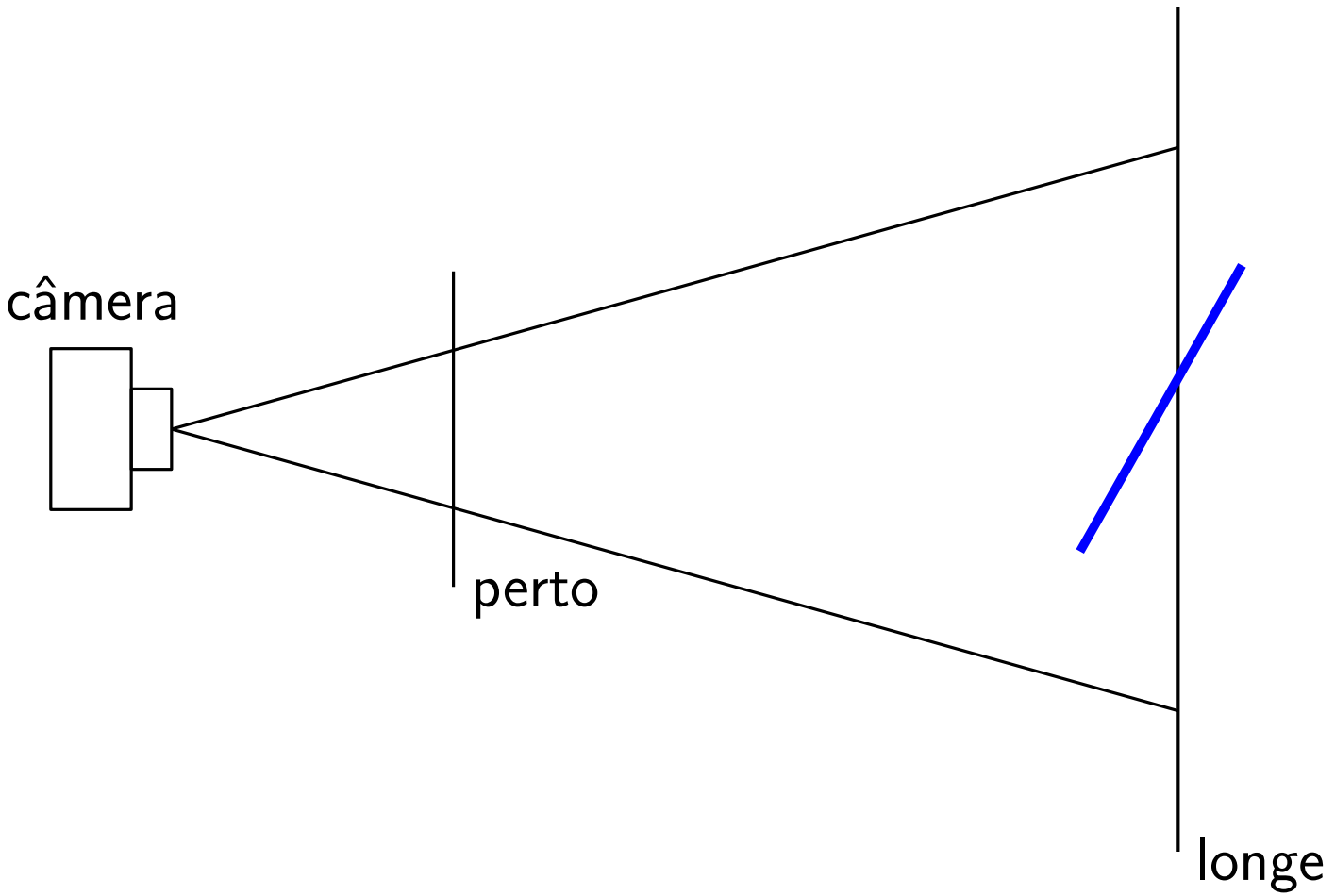
Posicionamento no pipeline

Pipeline gráfico (básico) de renderização por rasterização



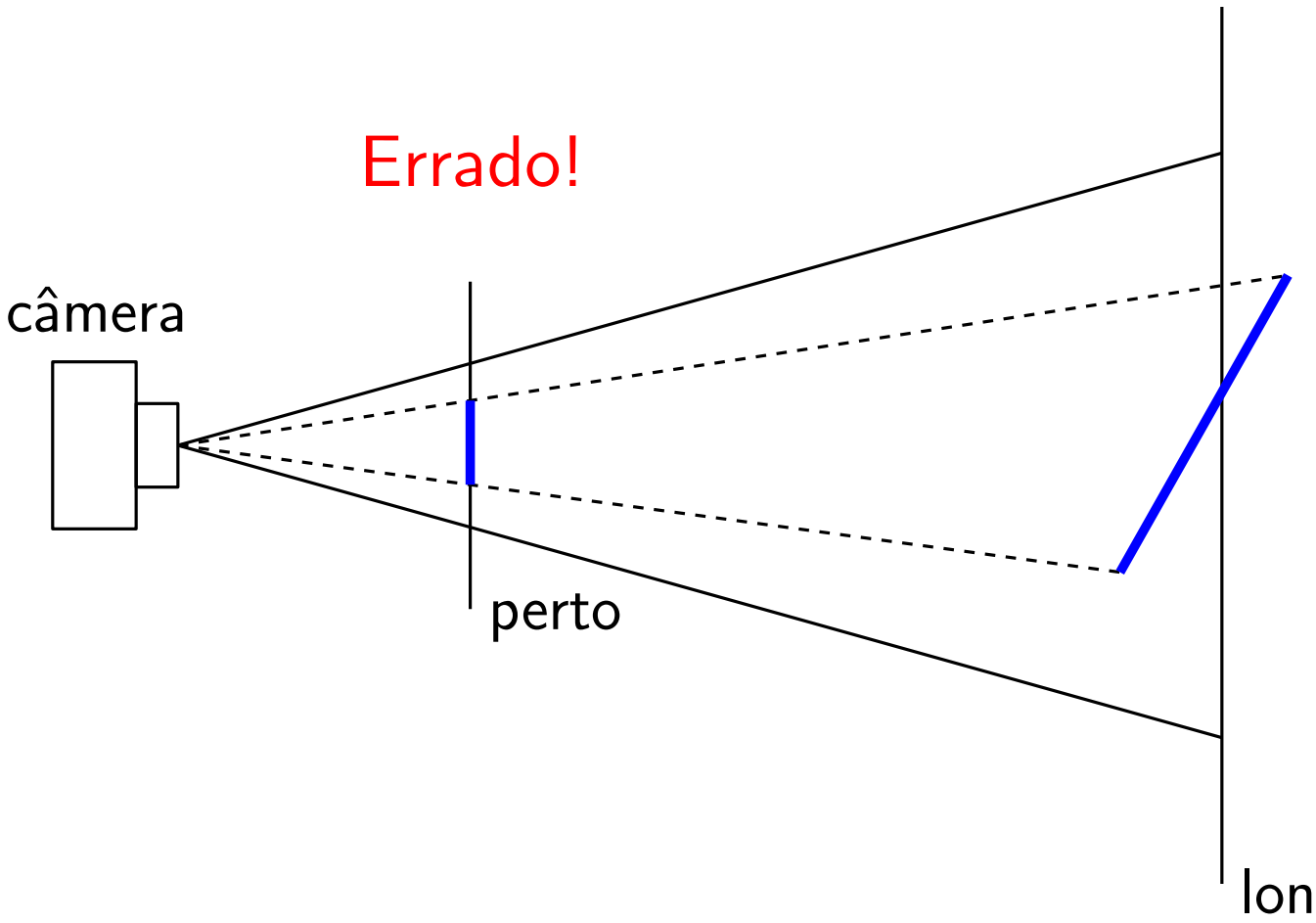
Por que o interesse em recortes?

- Rasterizar primitivas sem o devido cuidado pode levar a resultados indesejados



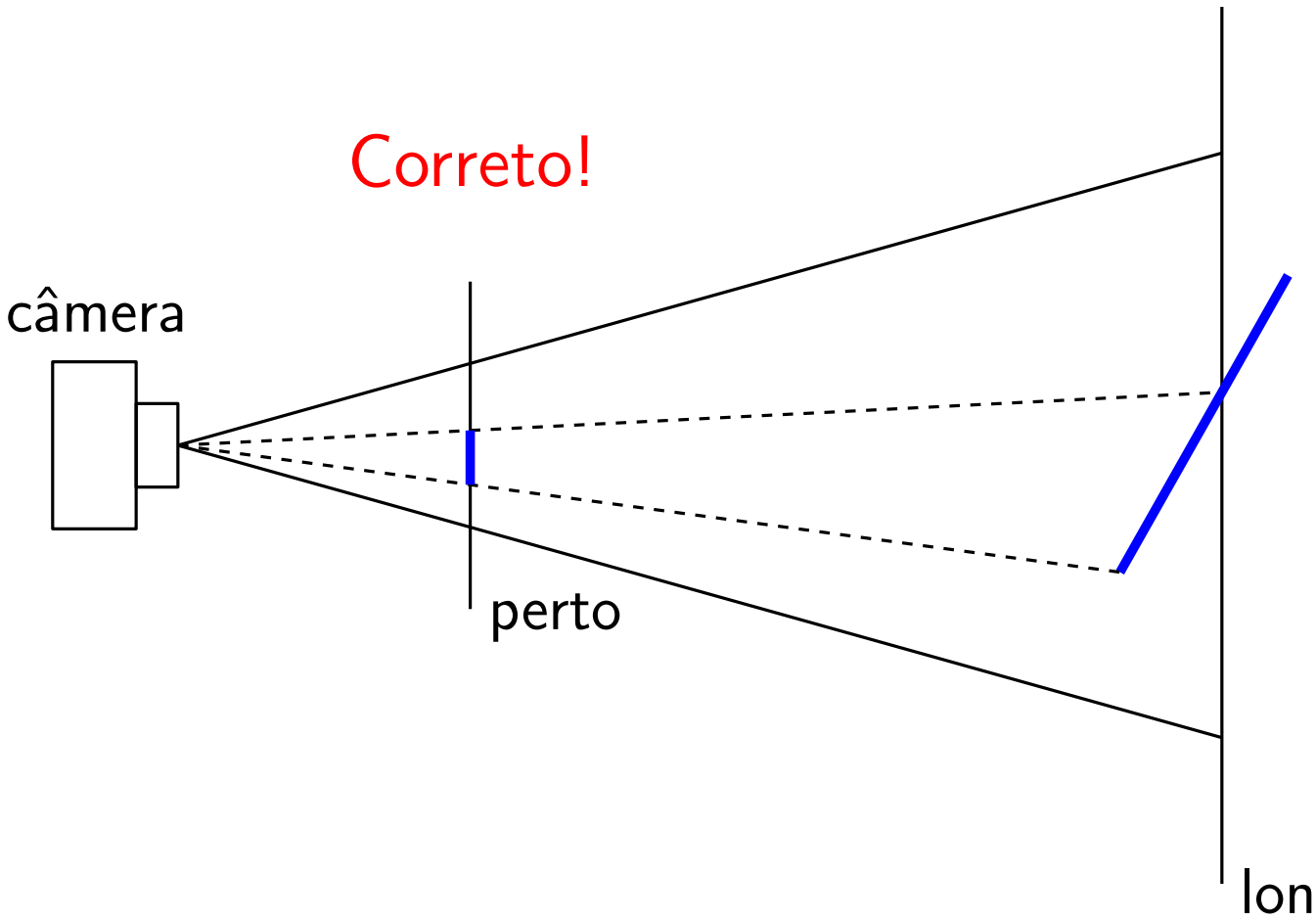
Por que o interesse em recortes?

- Rasterizar primitivas sem o devido cuidado pode levar a resultados indesejados



Por que o interesse em recortes?

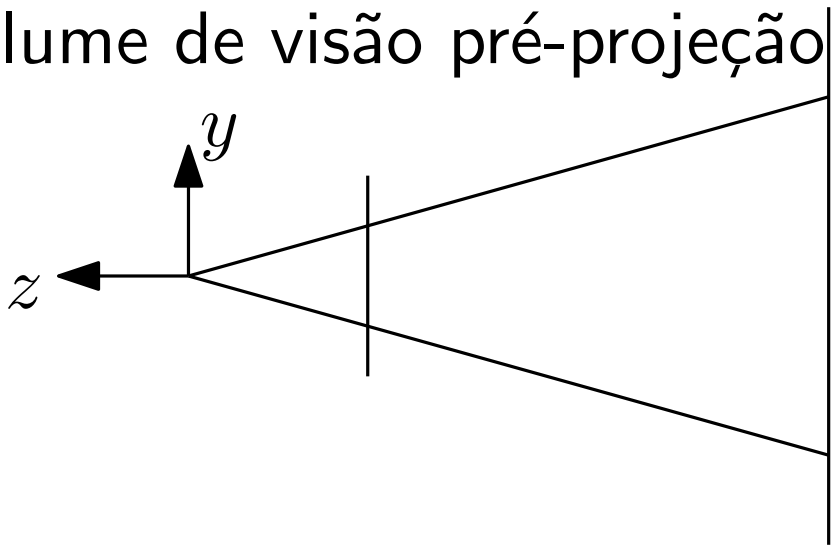
- Rasterizar primitivas sem o devido cuidado pode levar a resultados indesejados



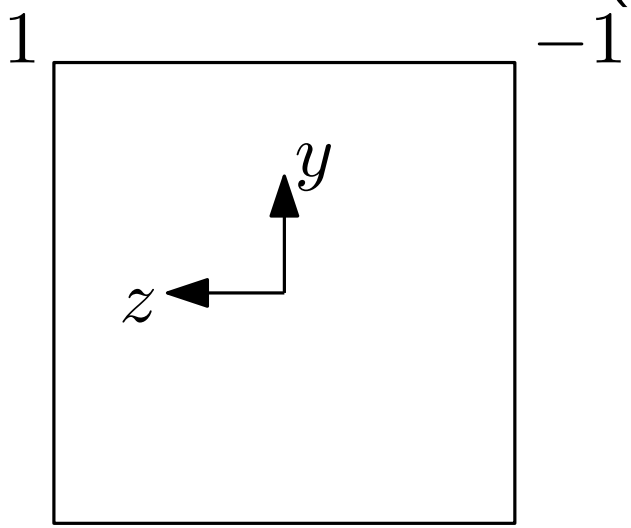
Espaços de recorte

O recorte pode ser implementado em dois espaços diferentes:

- Volume de visão pré-projeção



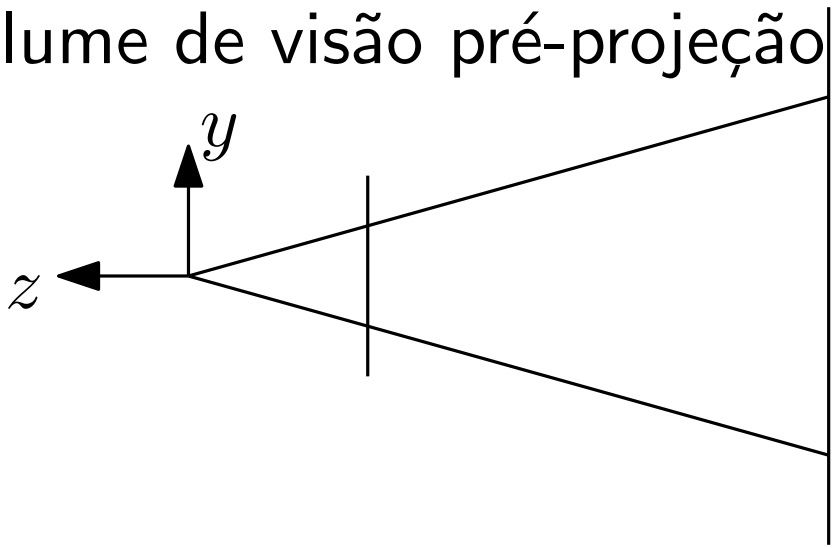
- Volume de visão normalizado (coordenadas homogêneas)



Espaços de recorte

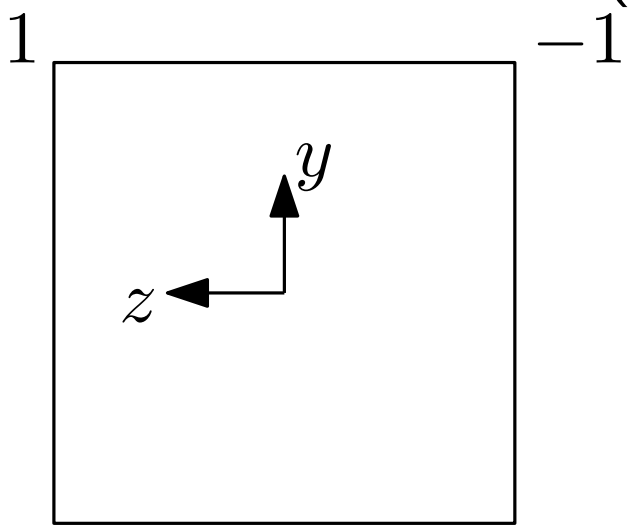
O recorte pode ser implementado em dois espaços diferentes:

- Volume de visão pré-projeção



$\Longrightarrow$ onde seções transversais definidas pelo usuário são calculadas com OpenGL

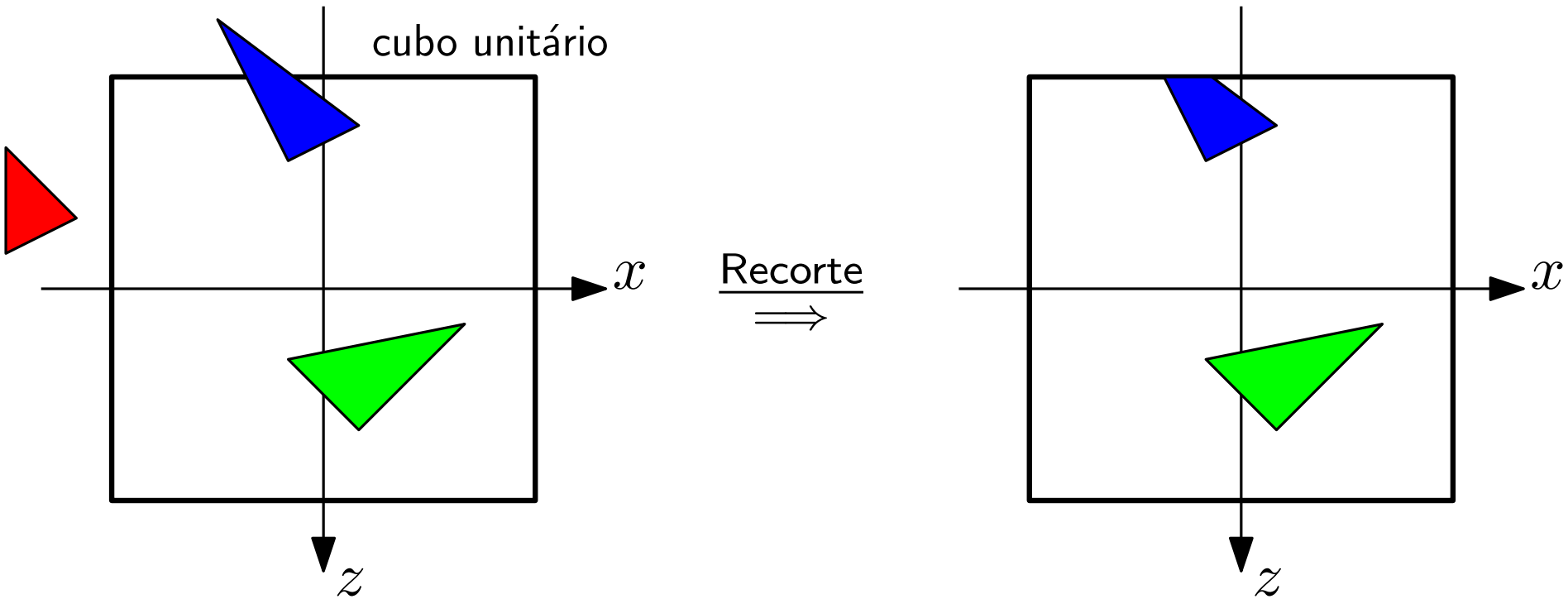
- Volume de visão normalizado (coordenadas homogêneas)



$\Longrightarrow$ OpenGL

Quem precisa ser recortado?

- Recortamos apenas primitivas parcialmente dentro do volume de visão



Outras utilidades para o recorte

- Seleção de objetos com auxílio do *mouse*
- Operações Booleanas em CSG (*Construtive Solid Geometry*)
- Sobreposição de janelas
- Anti-serrilhado

Definição geral do problema de recorte

Entrada.

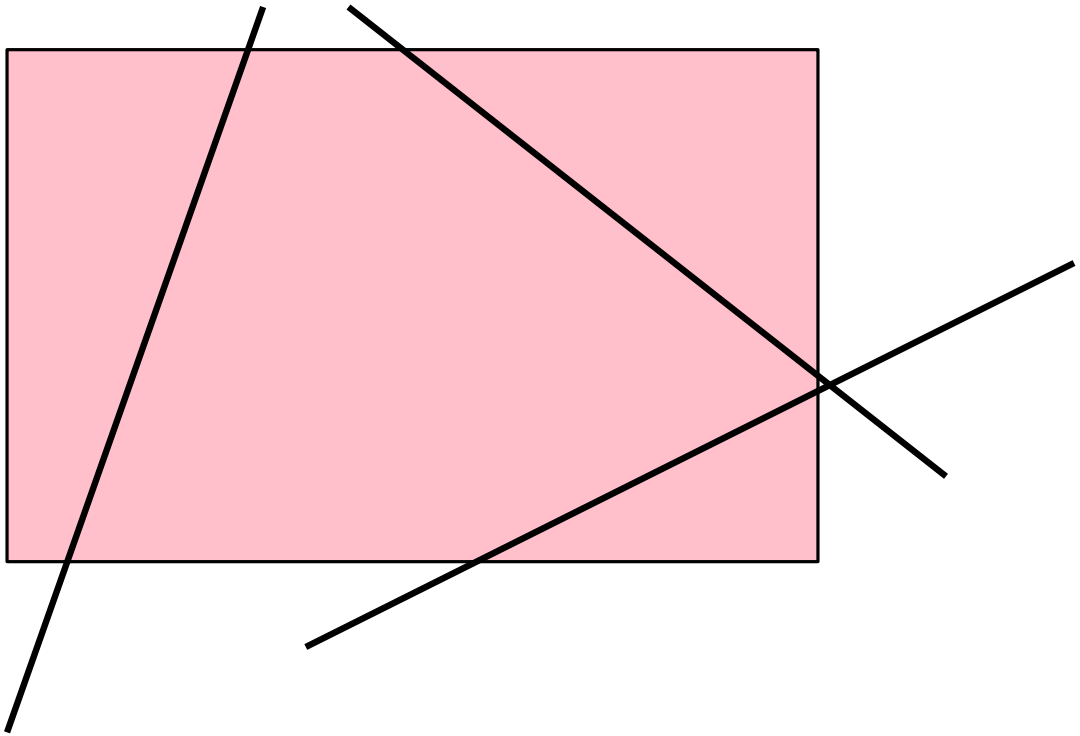
- *Objeto a ser recortado*: ponto, segmento de reta, polígono, curva, etc.
- *"Ferramenta" de recorte:* retângulo, polígono arbitrário, tronco de pirâmide ou paralepípedo (6 planos)

Saída.

Depende do objeto de entrada:

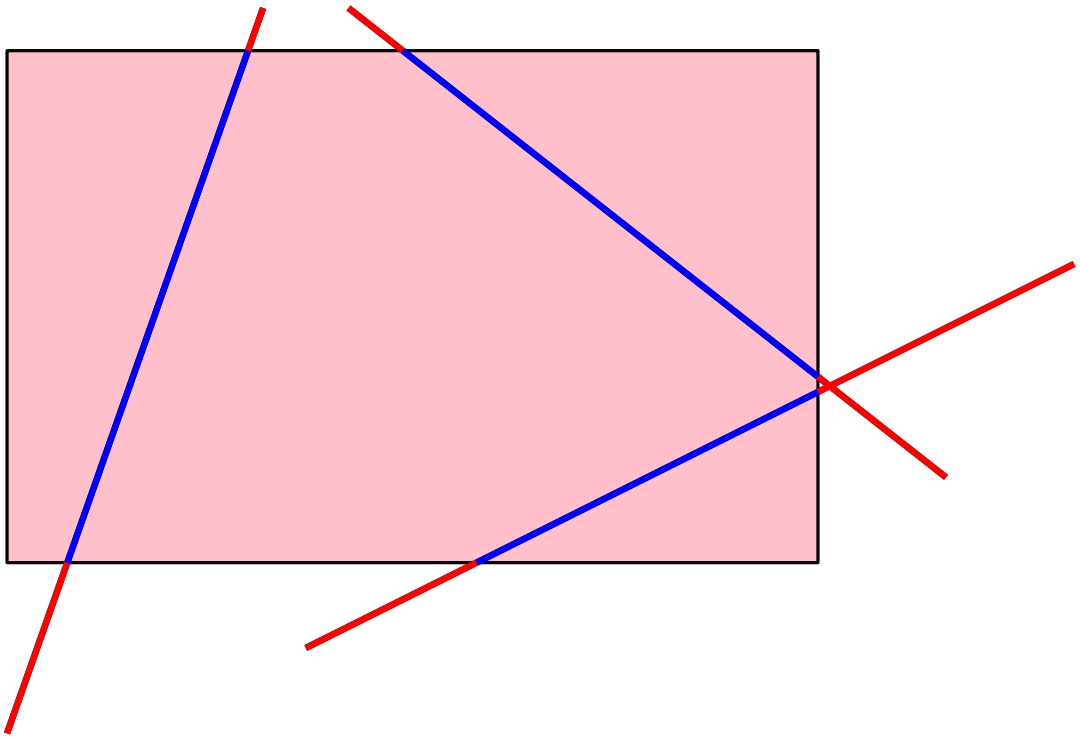
- Ponto $\Rightarrow$ sim ou não?
- Segmento de reta $\Rightarrow$ coleção de segmentos
- Polígono $\Rightarrow$ coleção de polígonos
- Curva $\Rightarrow$ coleção de curvas
- etc.

Recorte de segmentos de reta



Problema clássico em computação gráfica: recorte de segmento de reta contra retângulo alinhado com os eixos

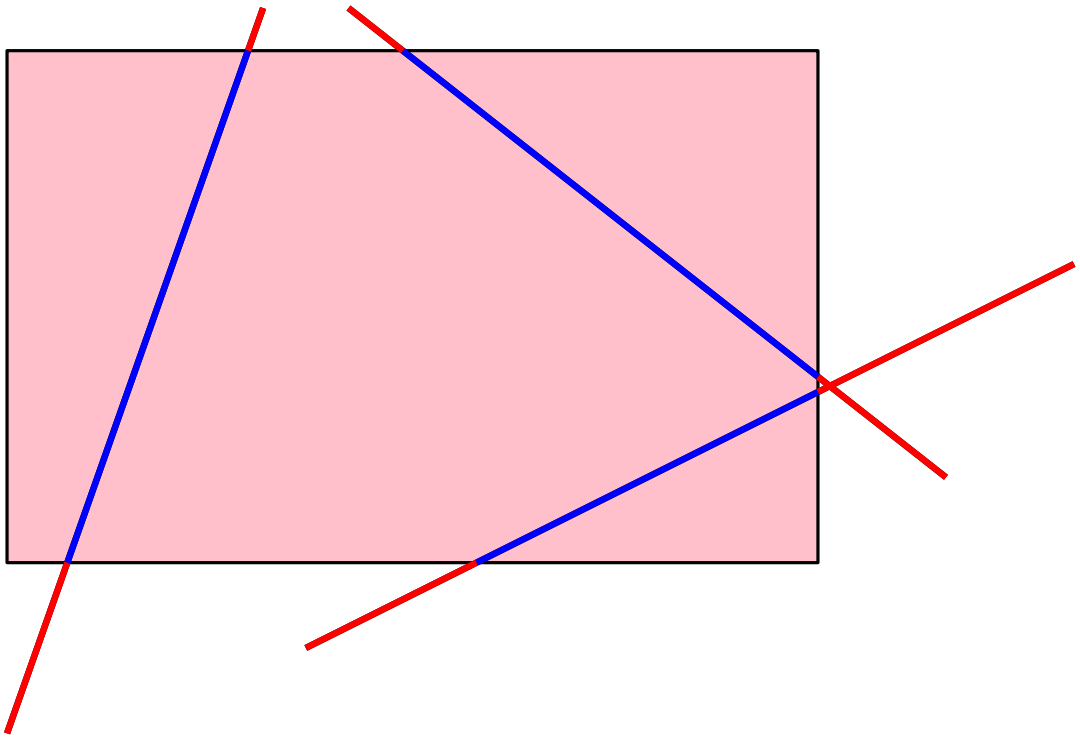
Recorte de segmentos de reta



Problema clássico em computação gráfica: recorte de segmento de reta contra retângulo alinhado com os eixos

Retângulo define região de interesse. As partes dos segmentos localizadas fora dessa região são removidas

Recorte de segmentos de reta



Problema clássico em computação gráfica: recorte de segmento de reta contra retângulo alinhado com os eixos

Retângulo define região de interesse. As partes dos segmentos localizadas fora dessa região são removidas

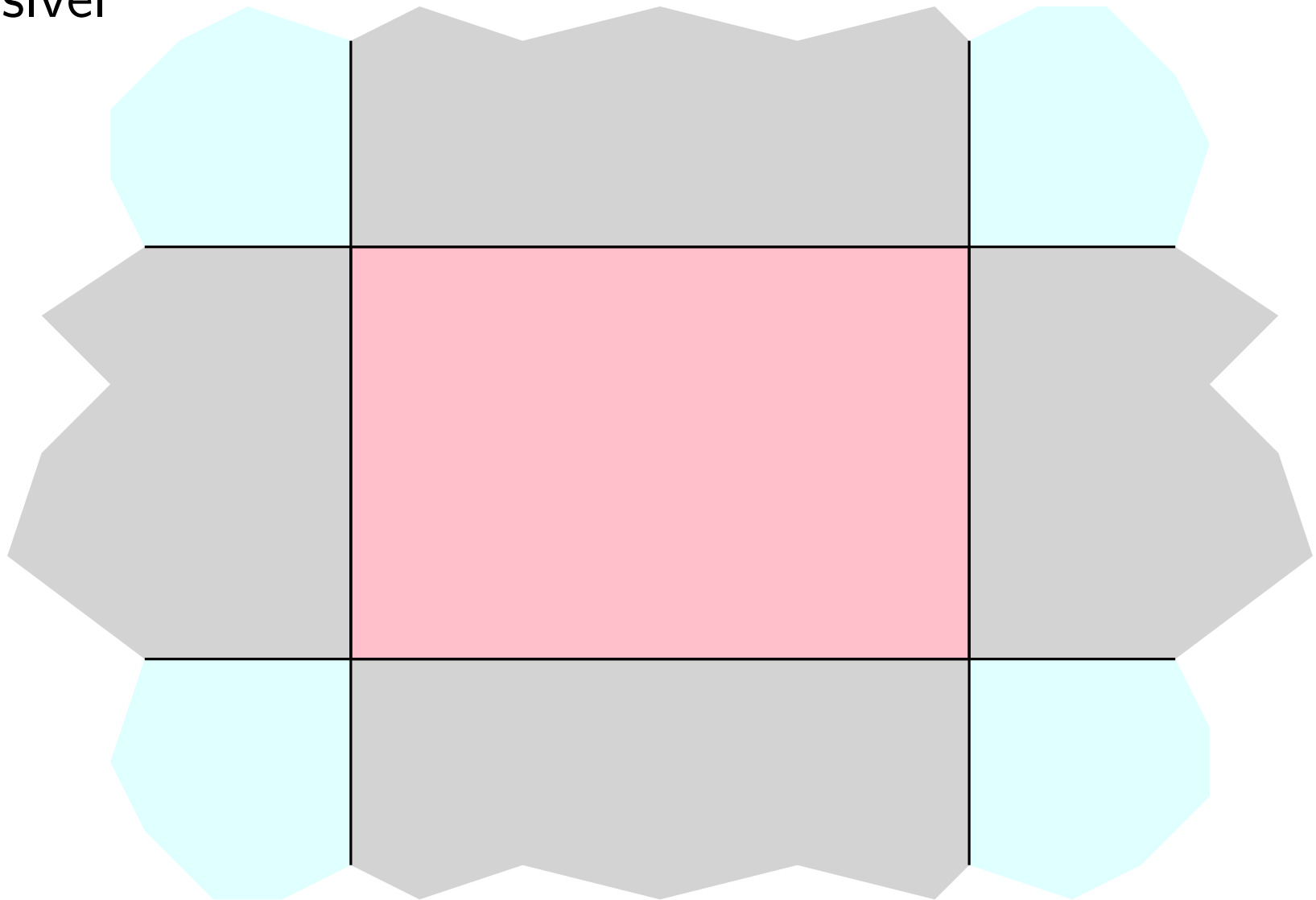
Os principais algoritmos são: Cohen-Sutherland, Liang-Barsky, Cyrus-Beck e Nicholl-Lee-Nicholl

Algoritmo de Cohen-Sutherland

- Decompõe o espaço em 9 regiões, das quais apenas uma é visível

Algoritmo de Cohen-Sutherland

- Decompõe o espaço em 9 regiões, das quais apenas uma é visível

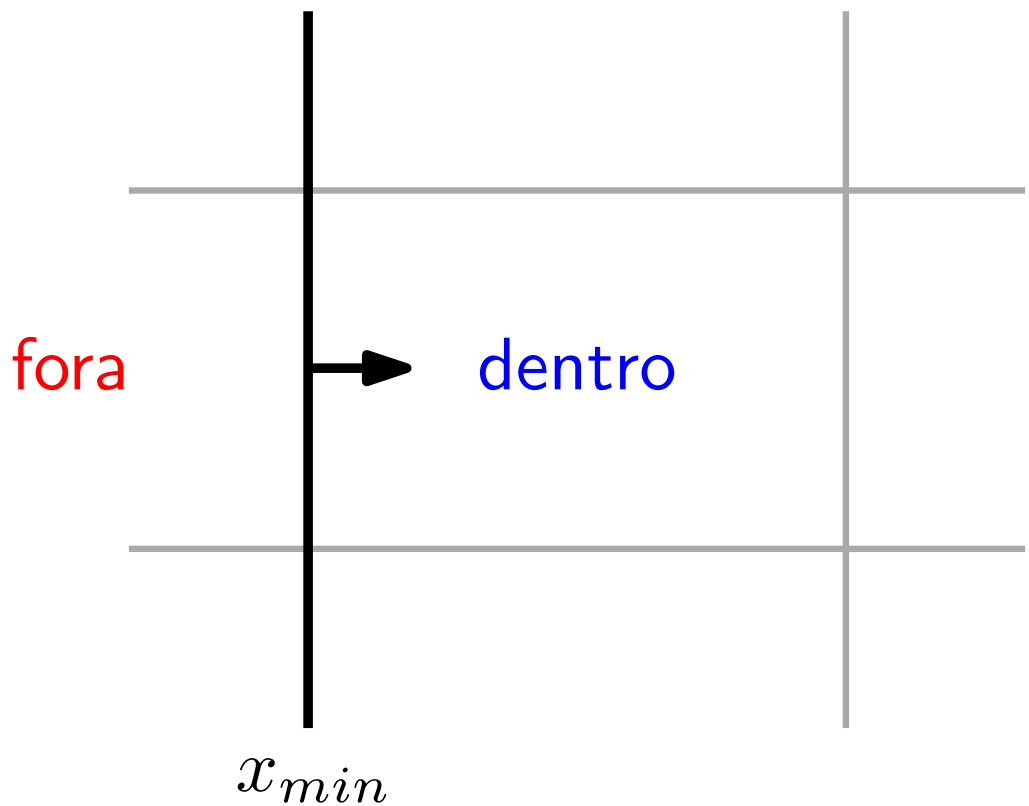


Algoritmo de Cohen-Sutherland

- Decompõe o espaço em 9 regiões, das quais apenas uma é visível
- Classifica os vértices do segmento (p, q) com relação a cada um dos 4 semi-planos que definem a região de interesse

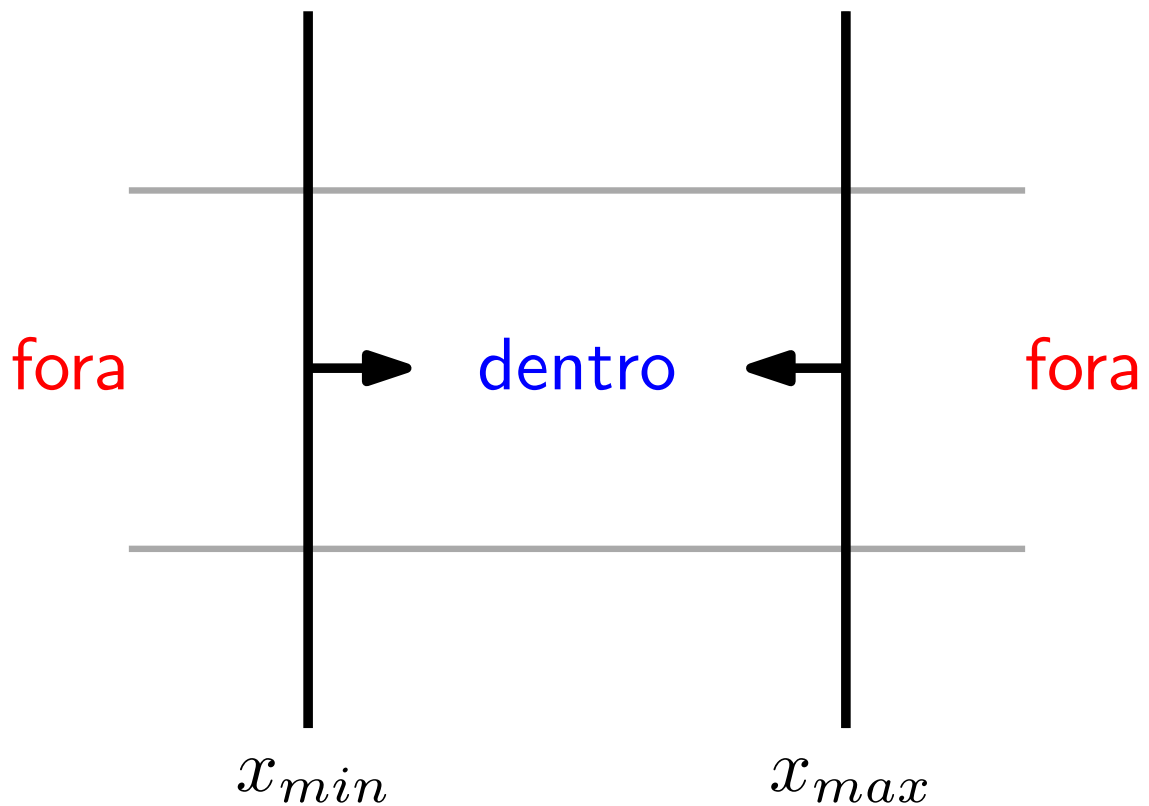
Algoritmo de Cohen-Sutherland

- Decompõe o espaço em 9 regiões, das quais apenas uma é visível
- Classifica os vértices do segmento (p, q) com relação a cada um dos 4 semi-planos que definem a região de interesse



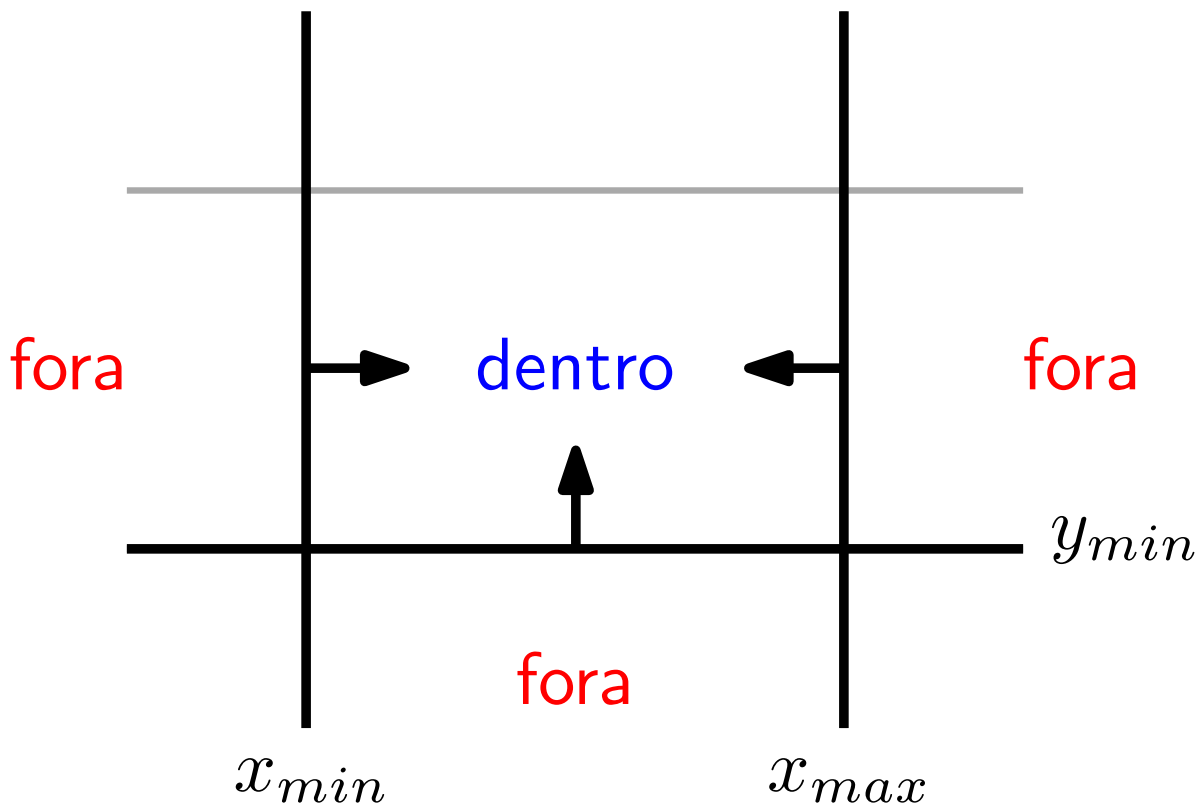
Algoritmo de Cohen-Sutherland

- Decompõe o espaço em 9 regiões, das quais apenas uma é visível
- Classifica os vértices do segmento (p, q) com relação a cada um dos 4 semi-planos que definem a região de interesse



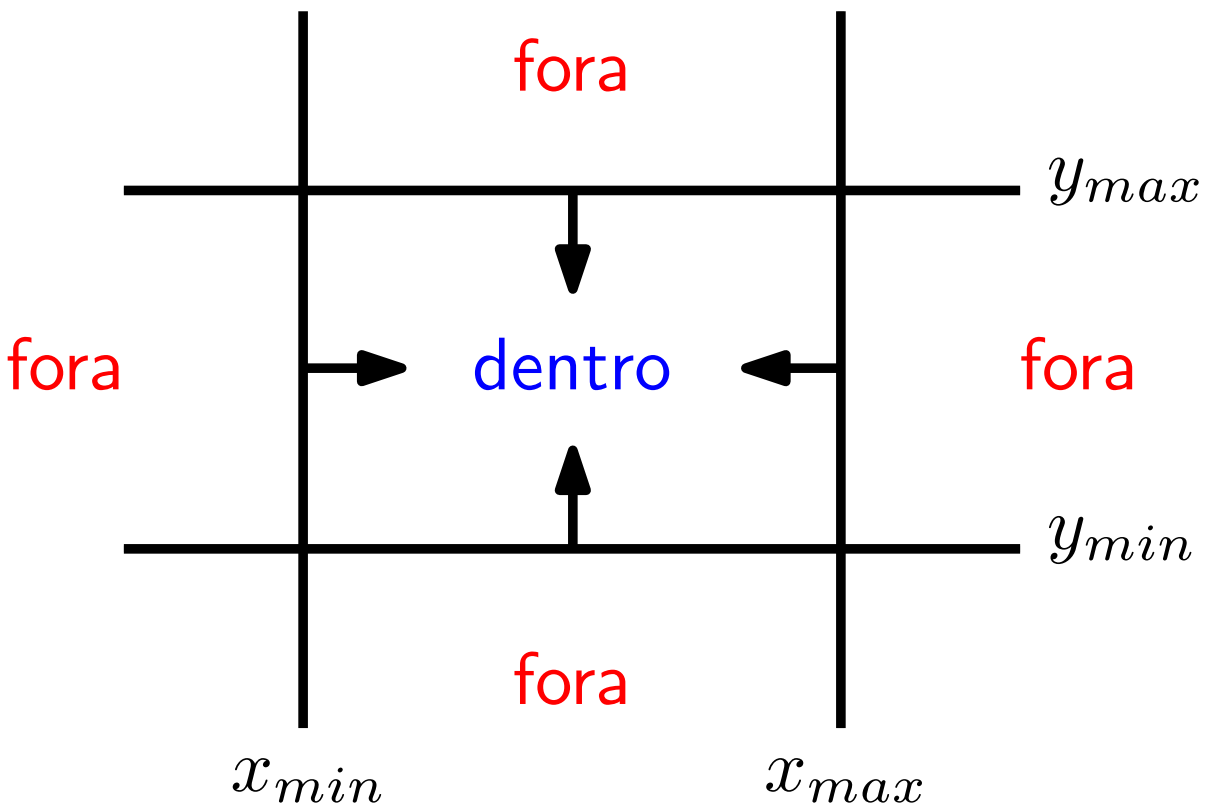
Algoritmo de Cohen-Sutherland

- Decompõe o espaço em 9 regiões, das quais apenas uma é visível
- Classifica os vértices do segmento (p, q) com relação a cada um dos 4 semi-planos que definem a região de interesse



Algoritmo de Cohen-Sutherland

- Decompõe o espaço em 9 regiões, das quais apenas uma é visível
- Classifica os vértices do segmento (p, q) com relação a cada um dos 4 semi-planos que definem a região de interesse



Algoritmo de Cohen-Sutherland

- Decompõe o espaço em 9 regiões, das quais apenas uma é visível
- Classifica os vértices do segmento (p, q) com relação a cada um dos 4 semi-planos que definem a região de interesse
- Para cada semi-plano, haverá 3 situações:
  1. Ambos os vértices estão *fora*: segmento não-visível
  2. Um vértice *dentro* e o outro *fora*: substituir o vértice *fora* pelo ponto de interseção; testar próximo semi-plano
  3. Ambos os vértices estão *dentro*: segmento visível

Algoritmo de Cohen-Sutherland

Dado um ponto p, utilizamos um código binário $W(p) = b_0 b_1 b_2 b_3$ para classificá-lo segundo sua região

Algoritmo de Cohen-Sutherland

Dado um ponto p, utilizamos um código binário $W(p) = b_0 b_1 b_2 b_3$ para classificá-lo segundo sua região

$$b_0 = \begin{cases} 1 & \text{se } p_y > y_{max} \\ 0 & \text{caso contrário} \end{cases}$$

$$b_1 = \begin{cases} 1 & \text{se } p_y < y_{min} \\ 0 & \text{caso contrário} \end{cases}$$

$$b_2 = \begin{cases} 1 & \text{se } p_x > x_{max} \\ 0 & \text{caso contrário} \end{cases}$$

$$b_3 = \begin{cases} 1 & \text{se } p_x < x_{min} \\ 0 & \text{caso contrário} \end{cases}$$

Algoritmo de Cohen-Sutherland

Dado um ponto p, utilizamos um código binário $W(p) = b_0 b_1 b_2 b_3$ para classificá-lo segundo sua região

$$b_0 = \begin{cases} 1 & \text{se } p_y > y_{max} \\ 0 & \text{caso contrário} \end{cases}$$

$$b_1 = \begin{cases} 1 & \text{se } p_y < y_{min} \\ 0 & \text{caso contrário} \end{cases}$$

$$b_2 = \begin{cases} 1 & \text{se } p_x > x_{max} \\ 0 & \text{caso contrário} \end{cases}$$

$$b_3 = \begin{cases} 1 & \text{se } p_x < x_{min} \\ 0 & \text{caso contrário} \end{cases}$$

1001	1000	1010
0001	0000	0010
0101	0100	0110

Observe que: $\begin{cases} 1 & \text{significa } \textit{fora} \\ 0 & \text{significa } \textit{dentro} \end{cases}$

Faculdade Paraíso - CE

Algoritmo de Cohen-Sutherland

Dado um ponto p, utilizamos um código binário $W(p) = b_0 b_1 b_2 b_3$ para classificá-lo segundo sua região

Para um segmento (p, q), teremos:

1001	1000	1010
0001	0000	0010
0101	0100	0110

Observe que: $\begin{cases} 1 & \text{significa } \textit{fora} \\ 0 & \text{significa } \textit{dentro} \end{cases}$

Algoritmo de Cohen-Sutherland

Dado um ponto p, utilizamos um código binário $W(p) = b_0 b_1 b_2 b_3$ para classificá-lo segundo sua região

Para um segmento (p, q), teremos:

- Rejeição trivial:
  $W(p)$ AND $W(q) \neq 0$

1001	1000	1010
0001	0000	0010
0101	0100	0110

Observe que: $\begin{cases} 1 & \text{significa } \textit{fora} \\ 0 & \text{significa } \textit{dentro} \end{cases}$

Algoritmo de Cohen-Sutherland

Dado um ponto p, utilizamos um código binário $W(p) = b_0 b_1 b_2 b_3$ para classificá-lo segundo sua região

Para um segmento (p, q), teremos:

- Rejeição trivial:
  $W(p)$ `AND` $W(q) \neq 0$
- Aceitação trivial:
  $W(p)$ `OR` $W(q) = 0$

1001	1000	1010
0001	0000	0010
0101	0100	0110

Observe que: $\begin{cases} 1 & \text{significa } \textit{fora} \\ 0 & \text{significa } \textit{dentro} \end{cases}$

Faculdade Paraíso - CE

Algoritmo de Cohen-Sutherland

Dado um ponto p, utilizamos um código binário $W(p) = b_0 b_1 b_2 b_3$ para classificá-lo segundo sua região

Para um segmento (p, q), teremos:

- Rejeição trivial: $W(p)$ `AND` $W(q) \neq 0$
- Aceitação trivial: $W(p)$ `OR` $W(q) = 0$
- Caso contrário, p (ou q) está fora de um semi-plano; substituí-lo pela interseção e reiniciar o algoritmo

1001	1000	1010
0001	0000	0010
0101	0100	0110

Observe que: $\begin{cases} 1 & \text{significa } \textit{fora} \\ 0 & \text{significa } \textit{dentro} \end{cases}$

Faculdade Paraíso - CE

Ilustração do algoritmo

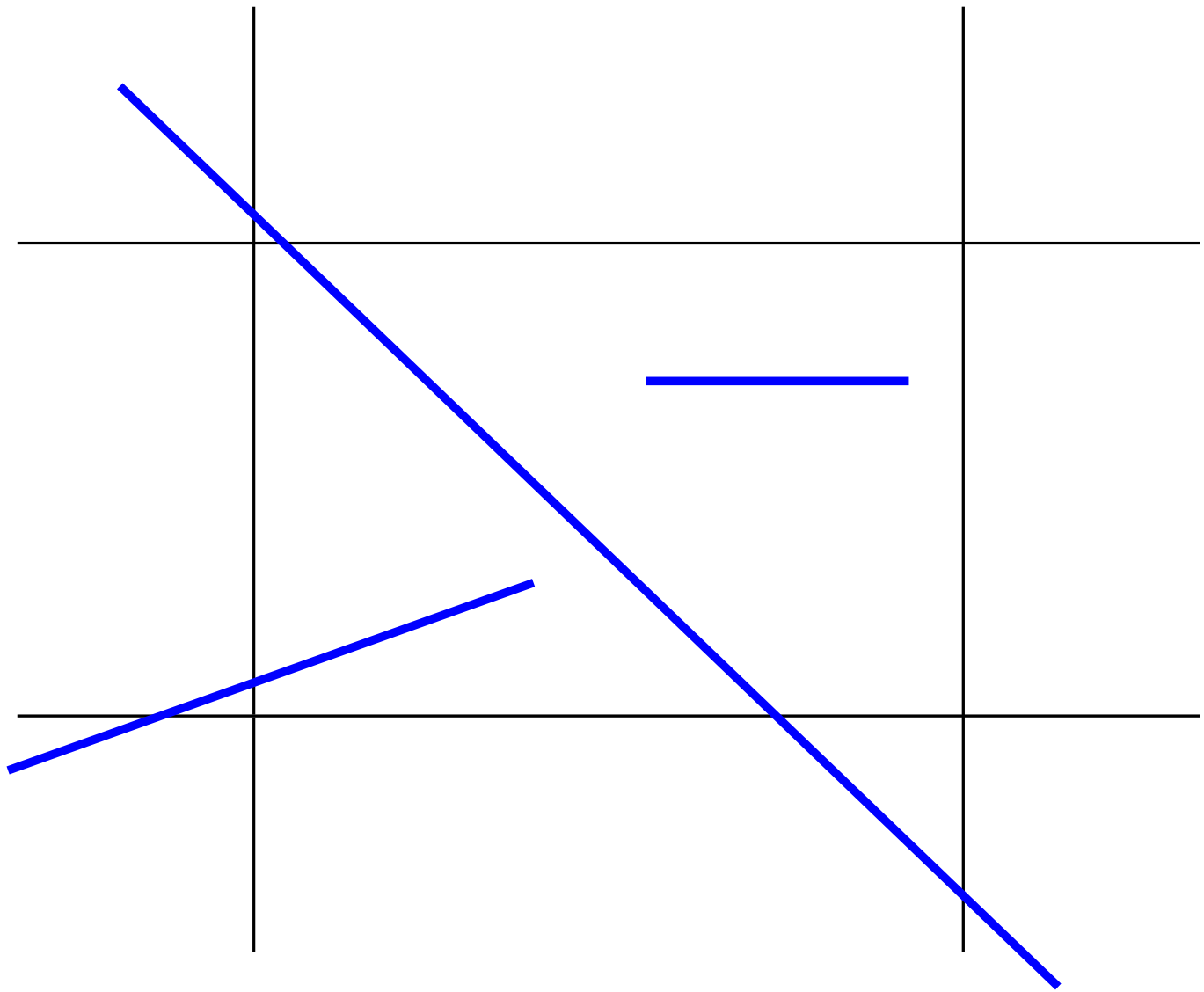
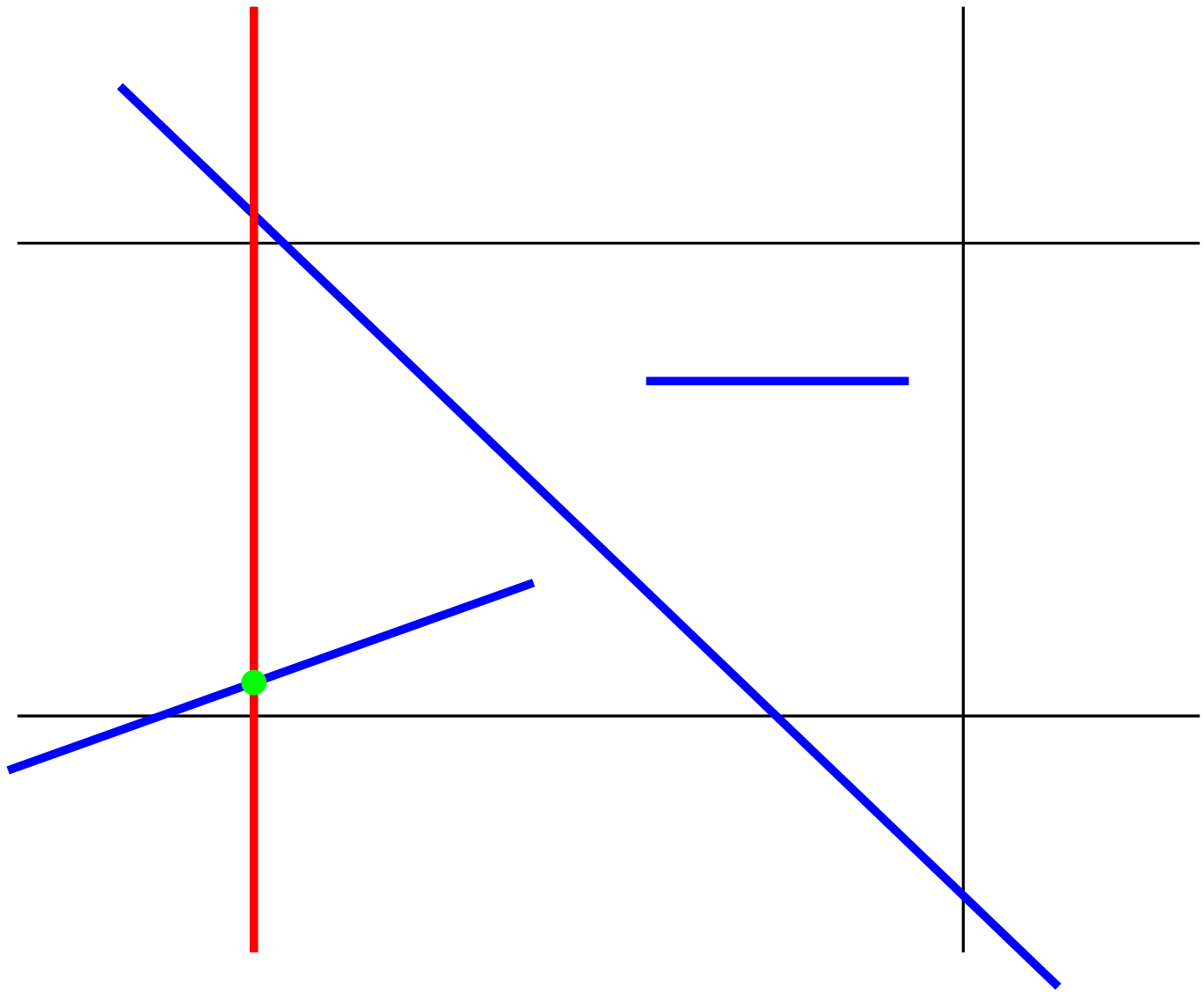


Ilustração do algoritmo



Algoritmo de Cohen-Sutherland

Ilustração do algoritmo

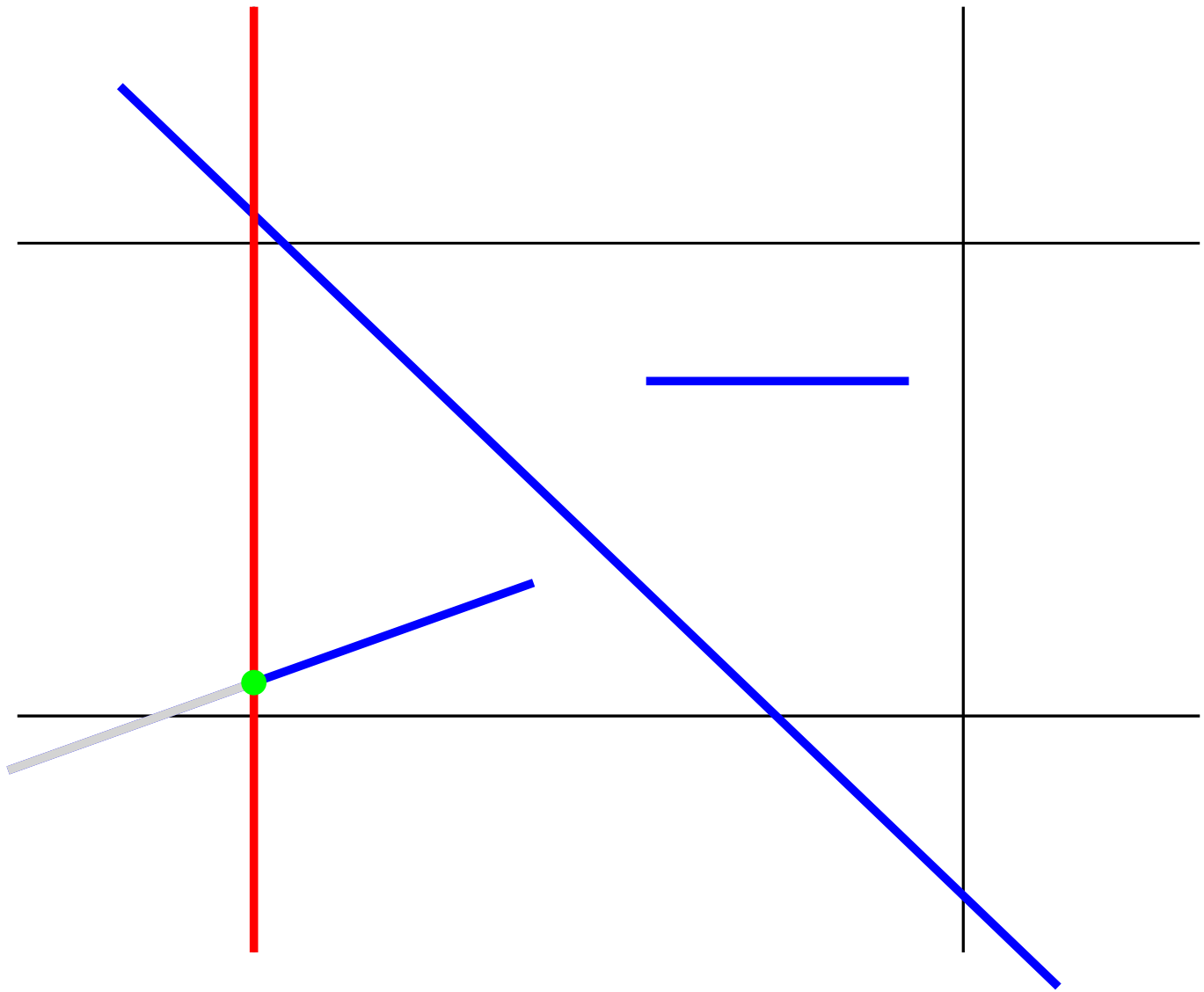


Ilustração do algoritmo

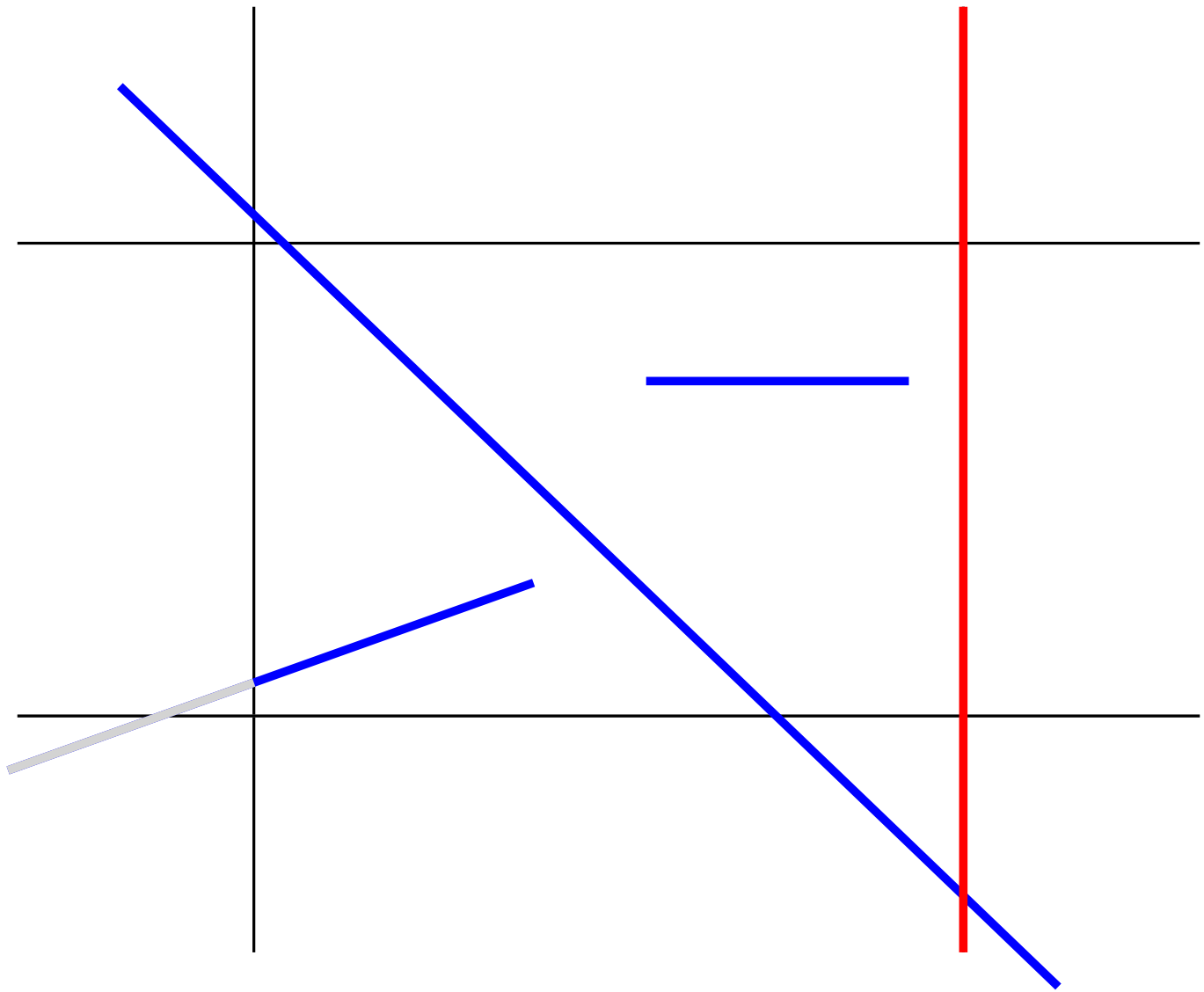


Ilustração do algoritmo

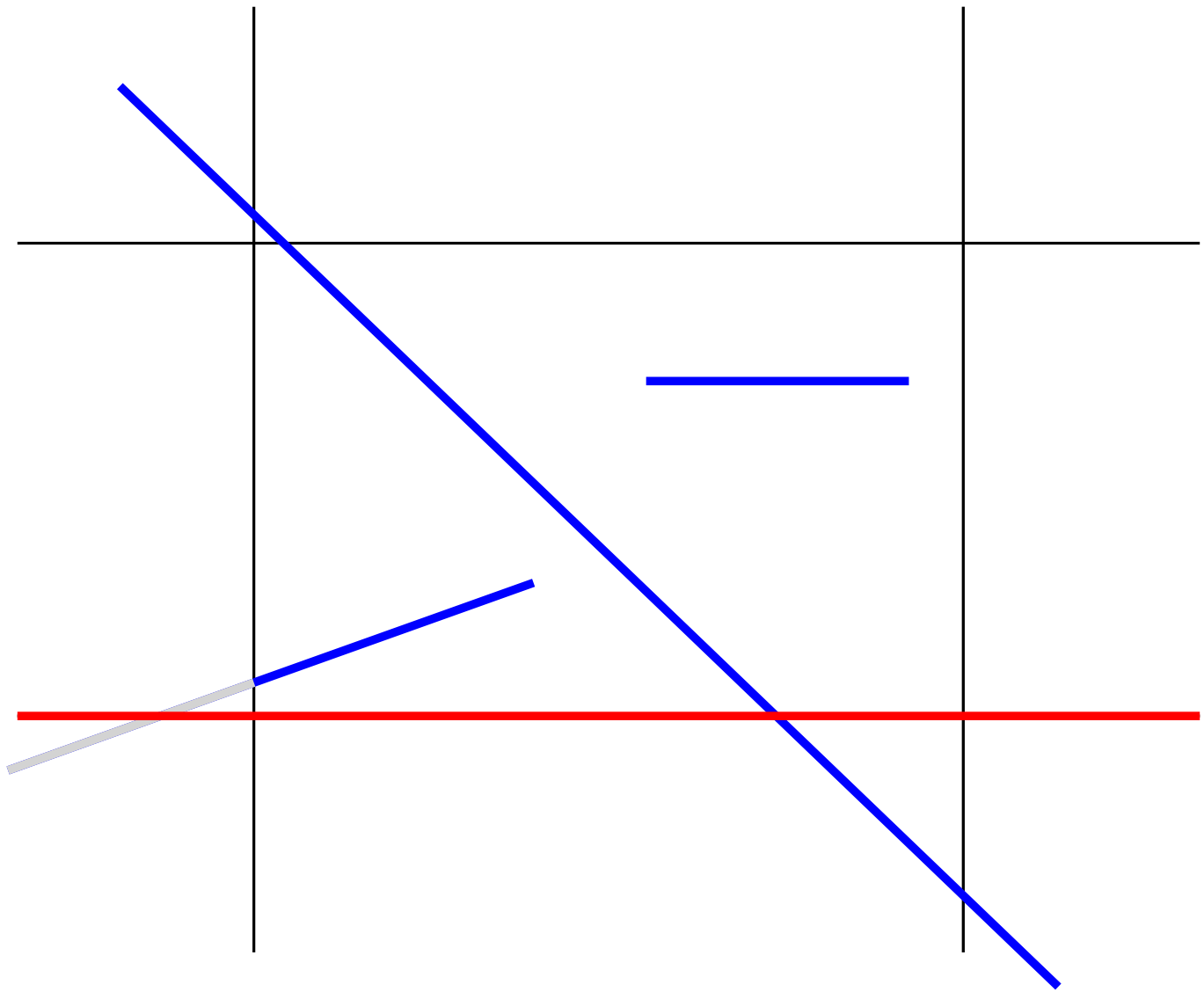


Ilustração do algoritmo

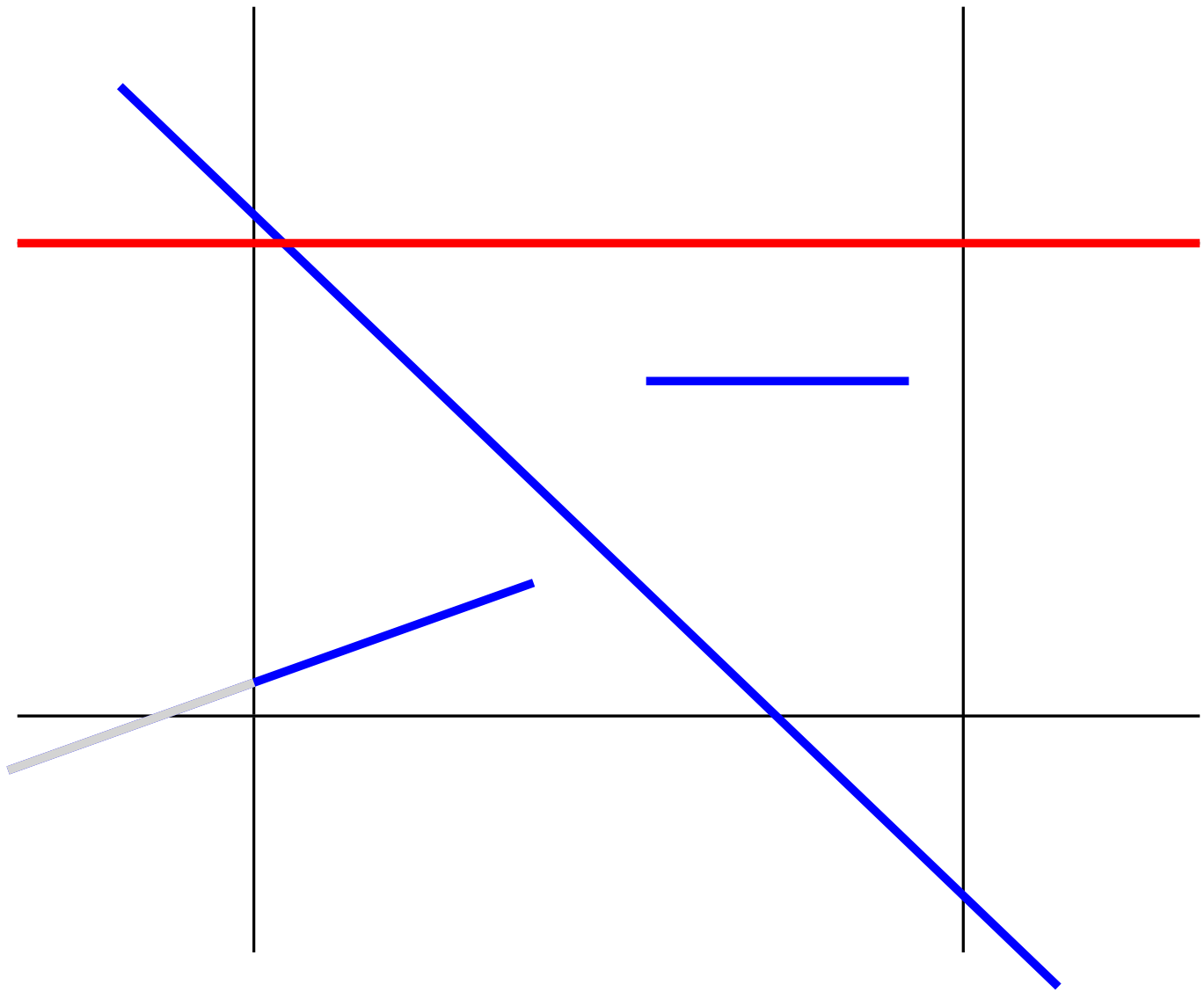


Ilustração do algoritmo

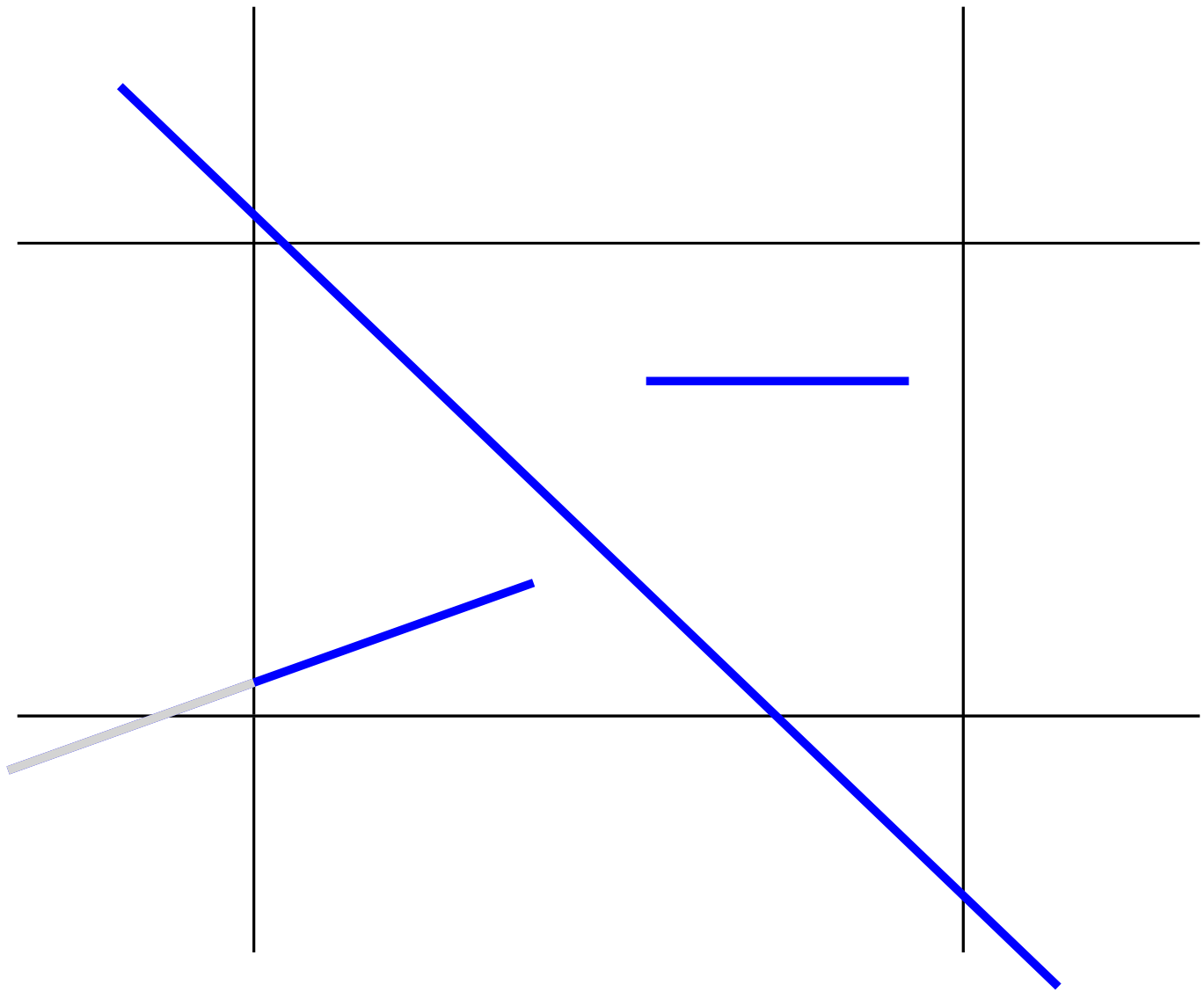


Ilustração do algoritmo

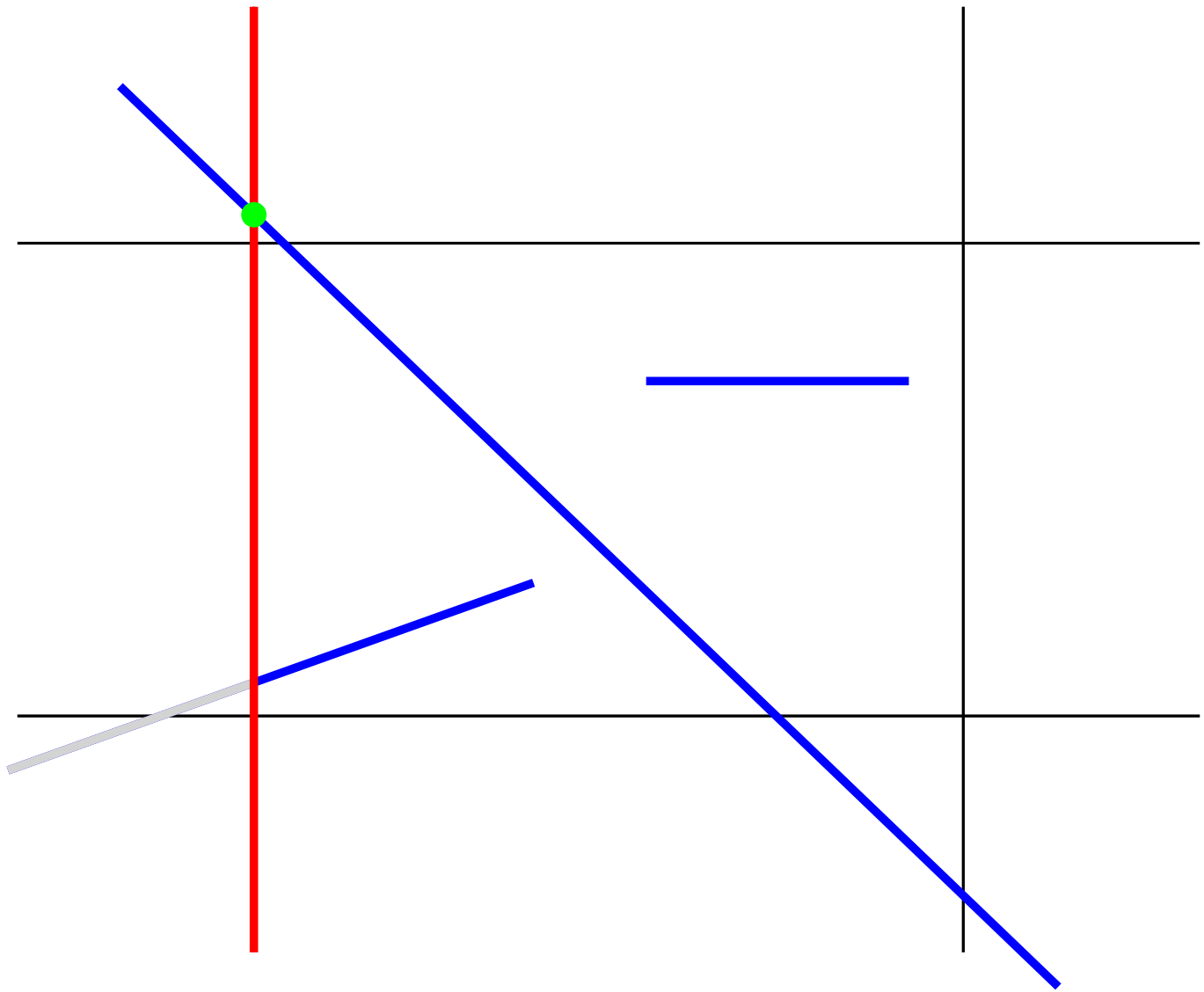


Ilustração do algoritmo

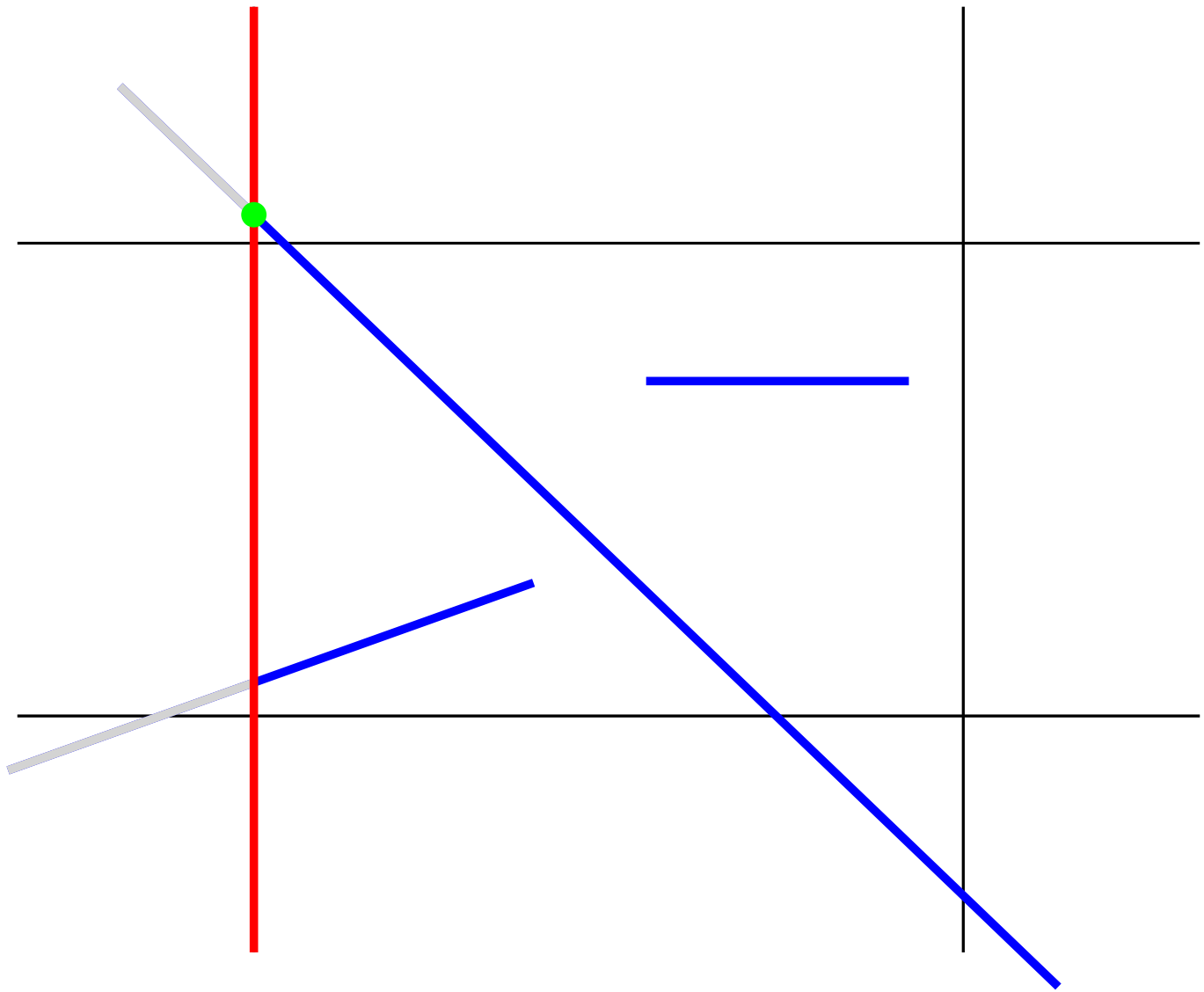


Ilustração do algoritmo

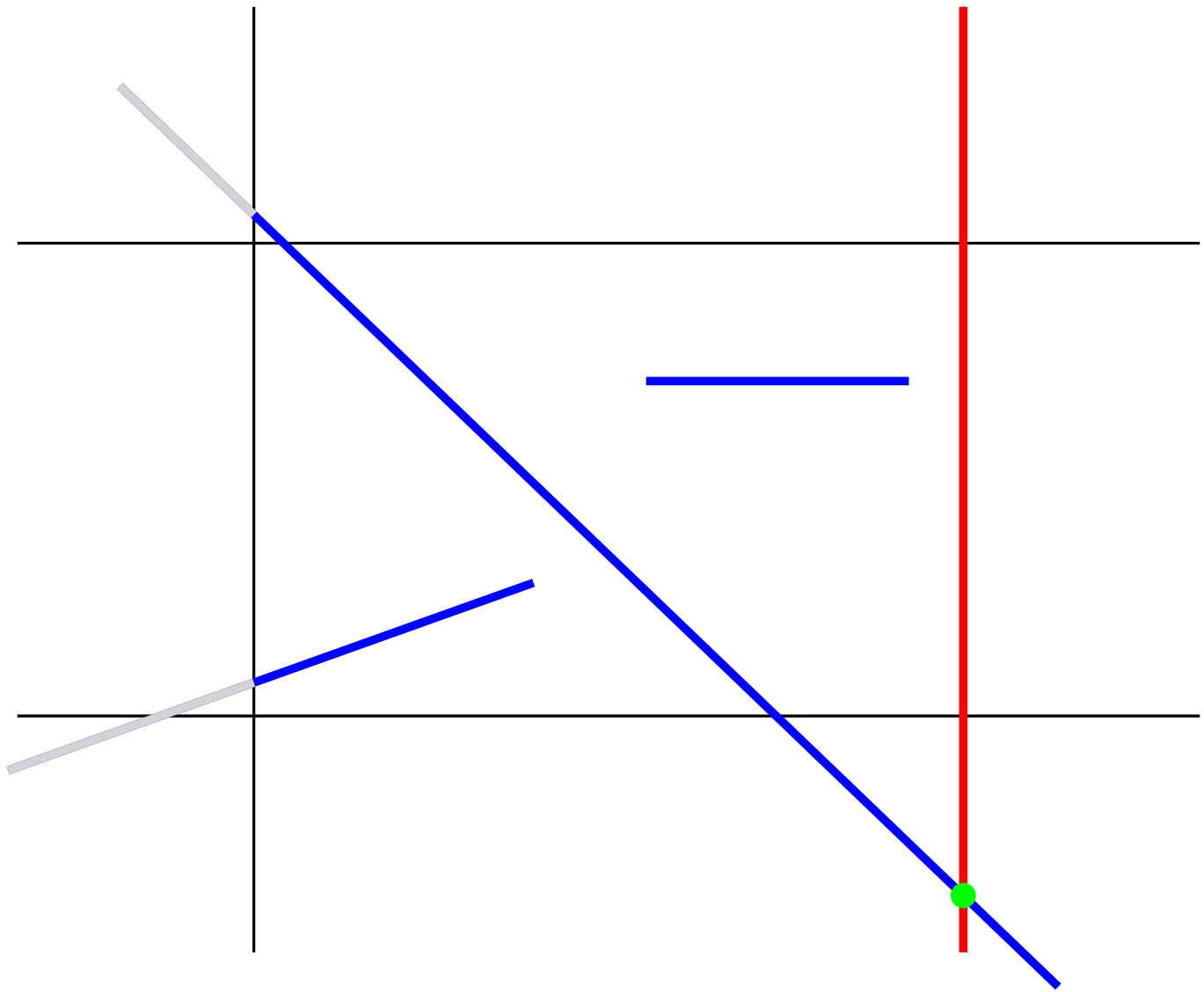


Ilustração do algoritmo

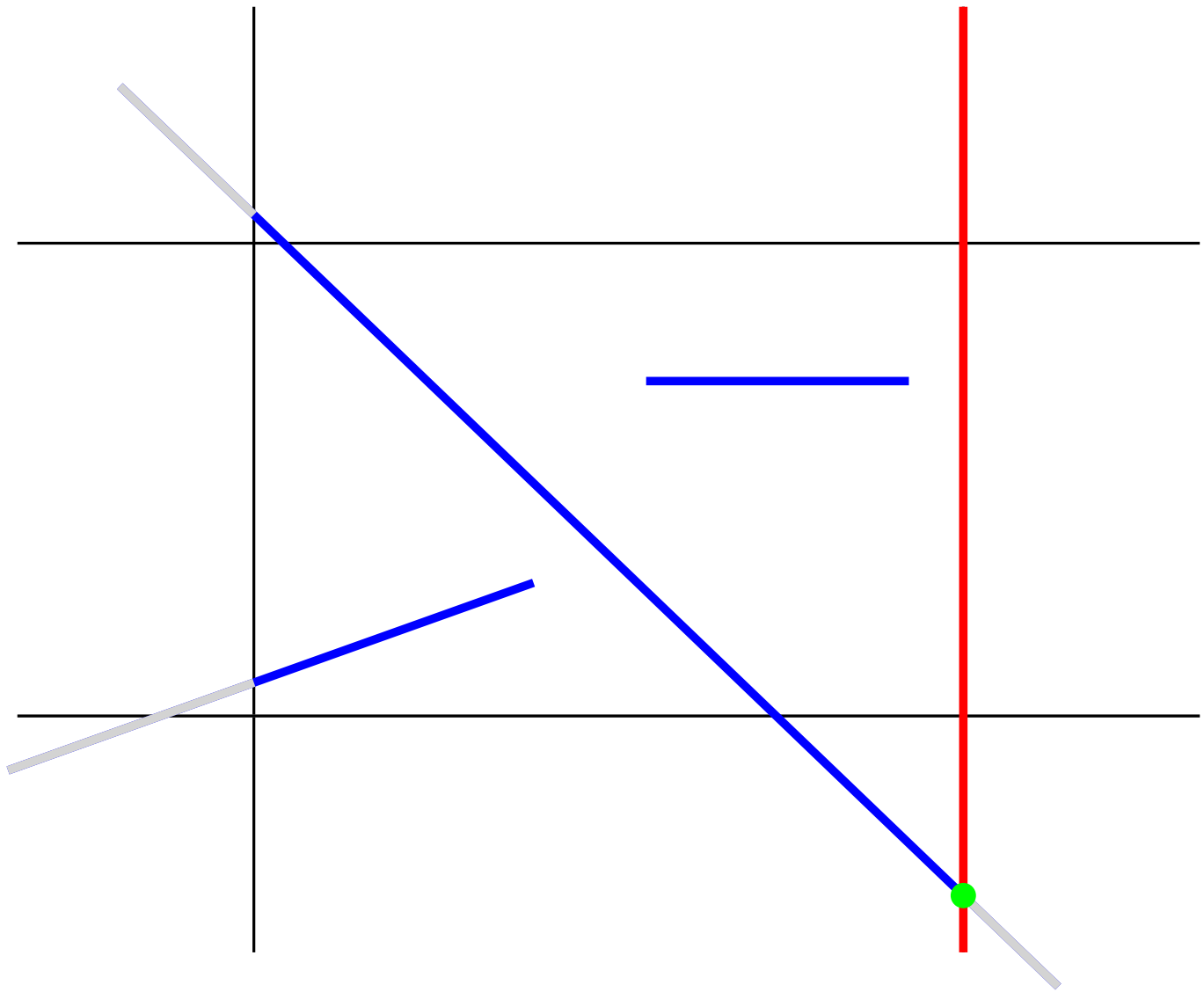


Ilustração do algoritmo

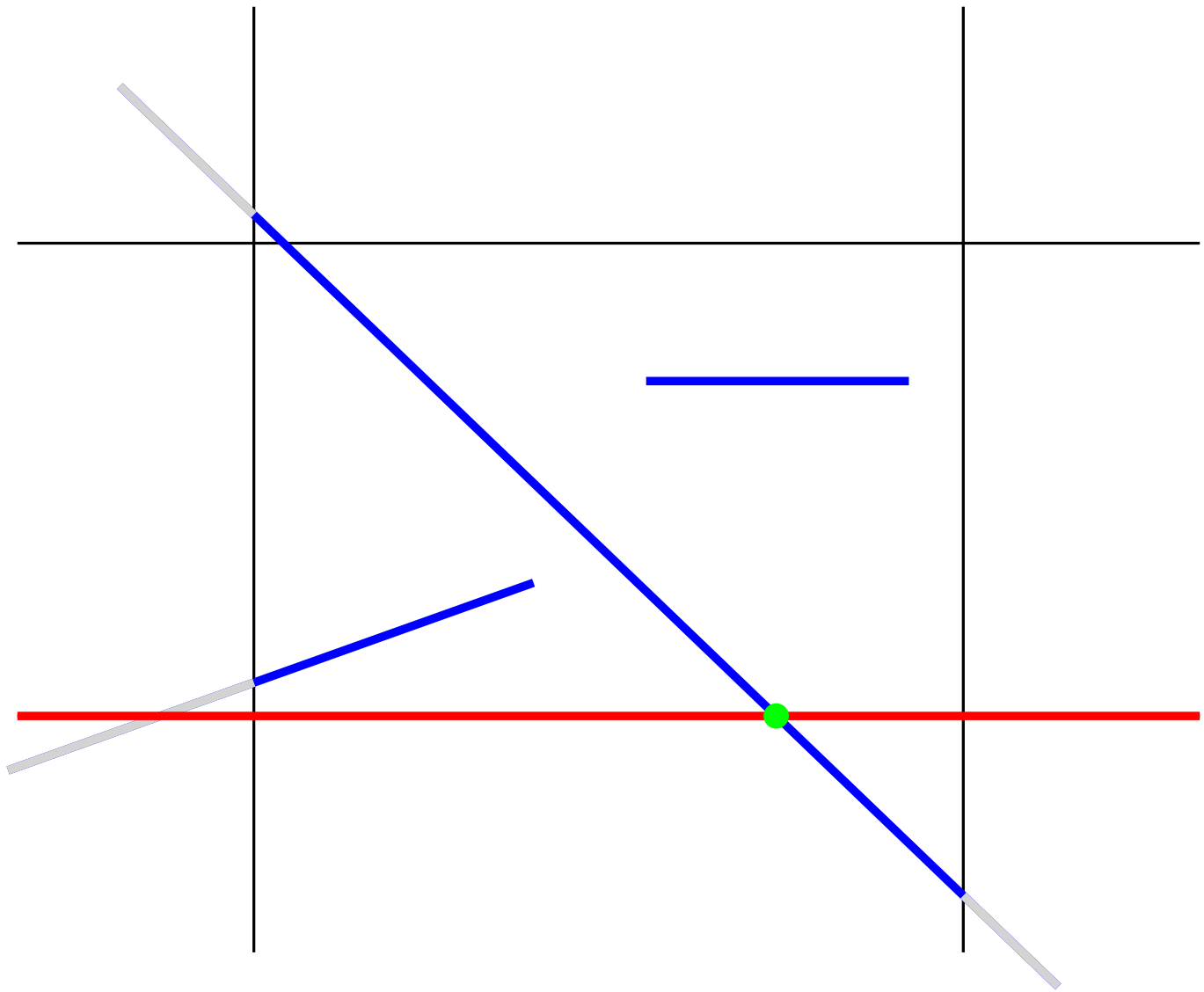


Ilustração do algoritmo

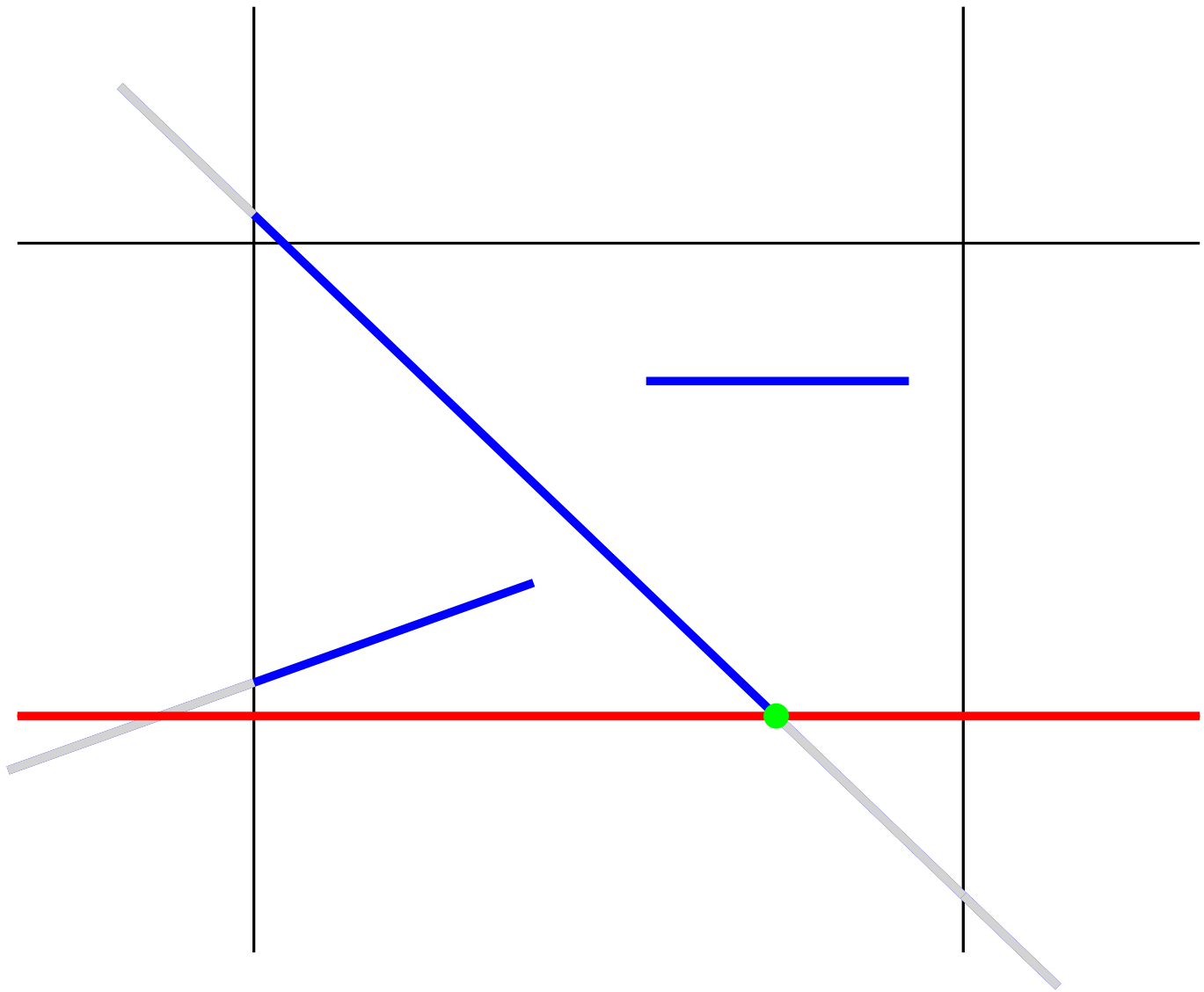


Ilustração do algoritmo

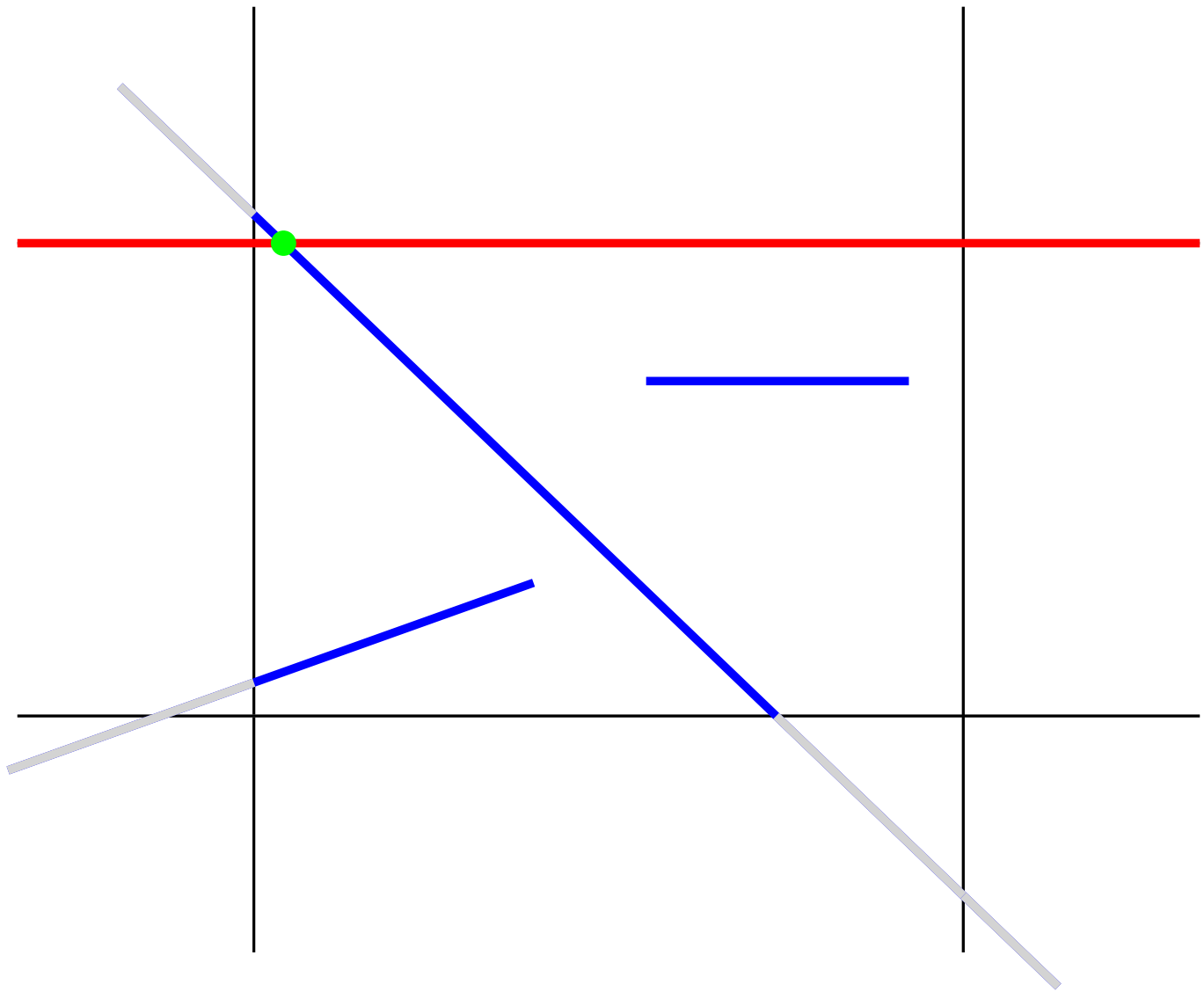


Ilustração do algoritmo

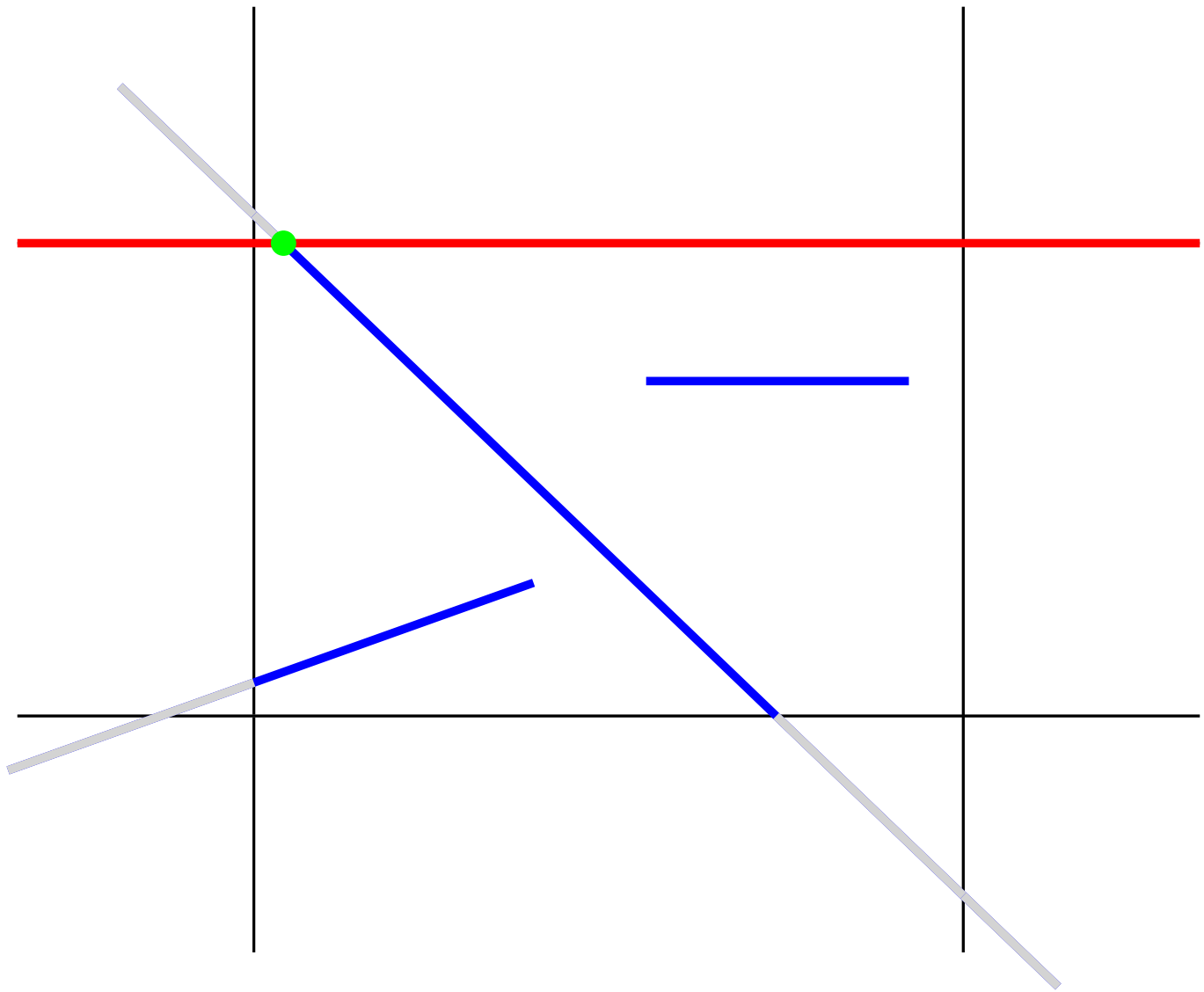


Ilustração do algoritmo

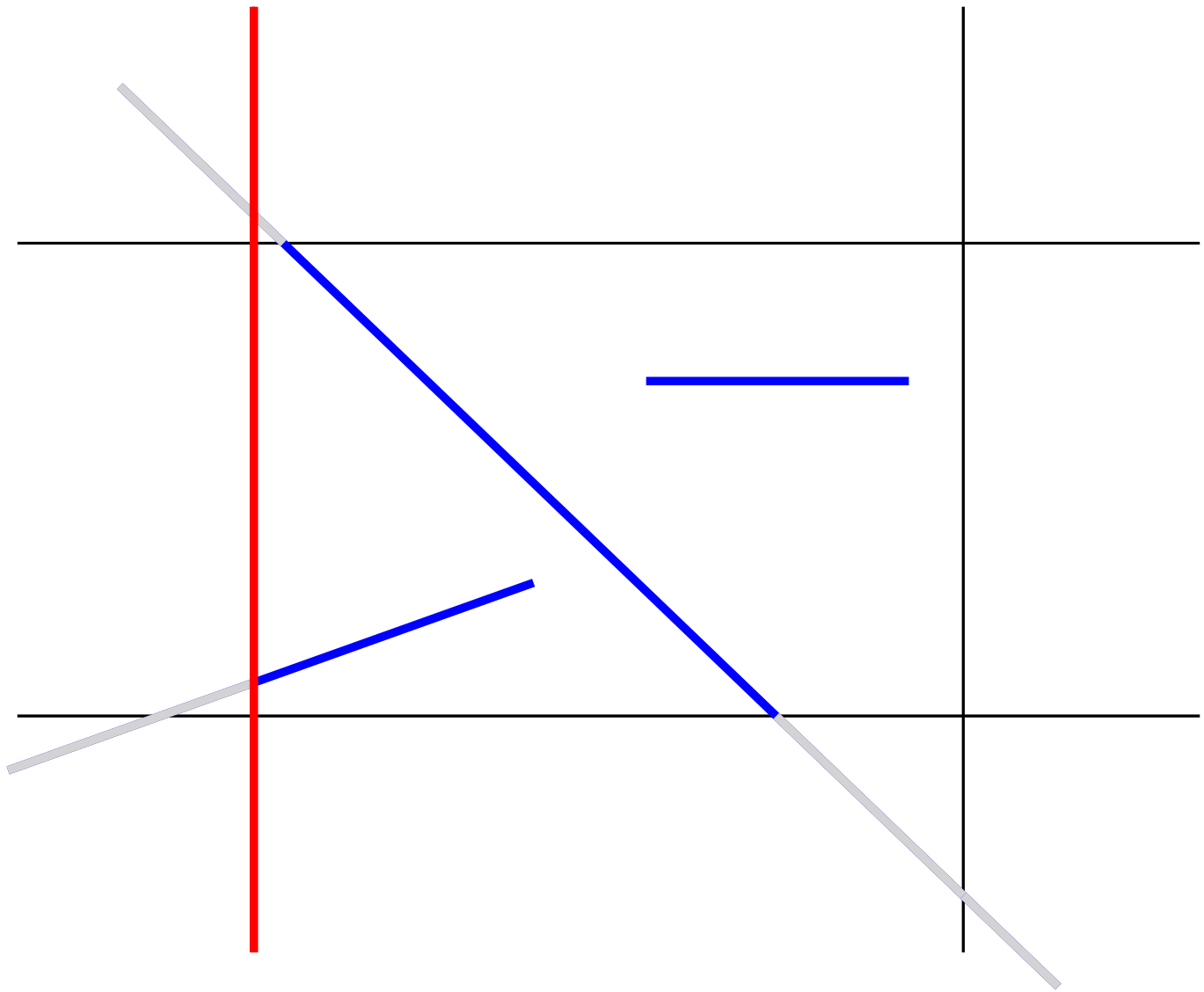


Ilustração do algoritmo

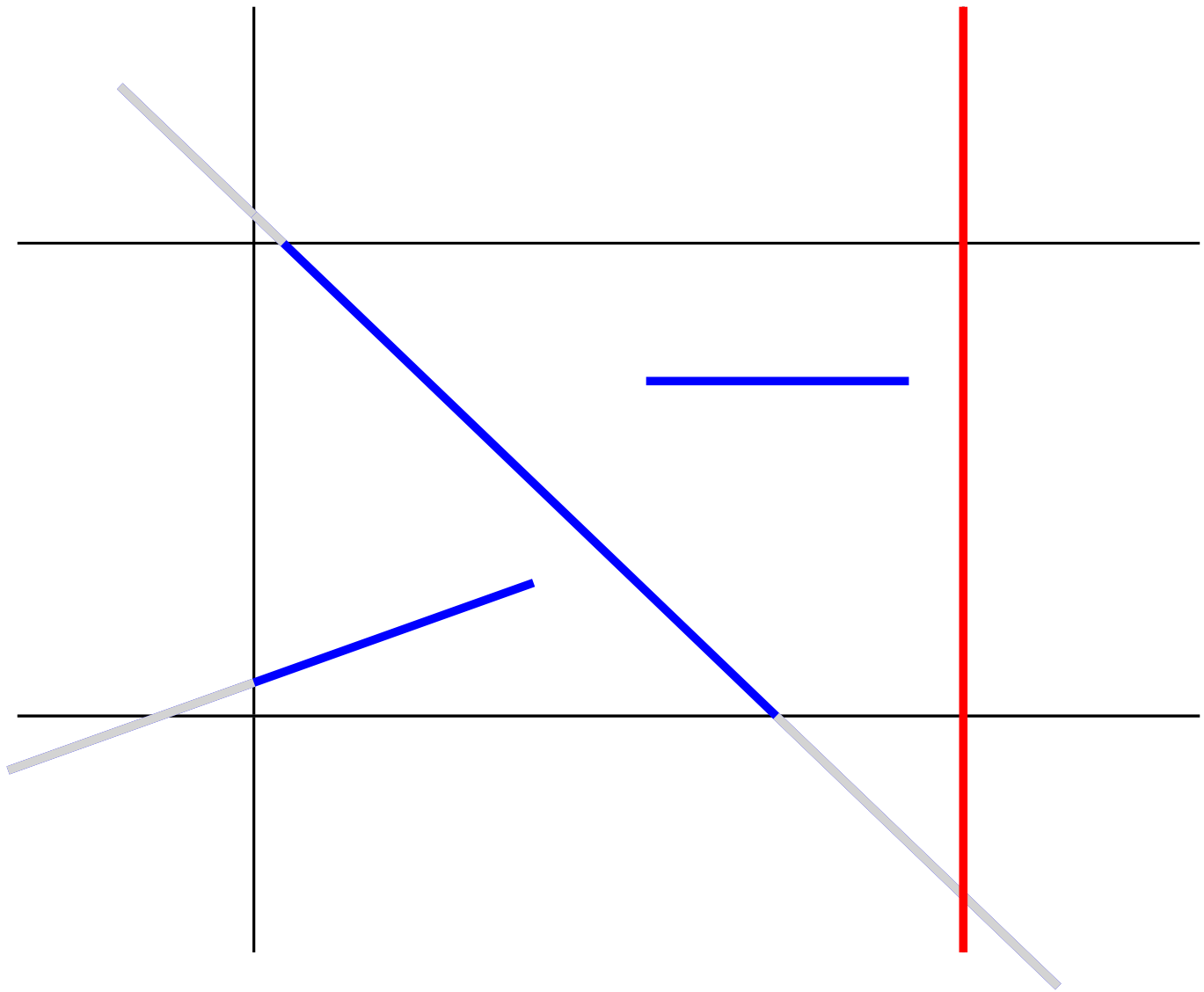


Ilustração do algoritmo

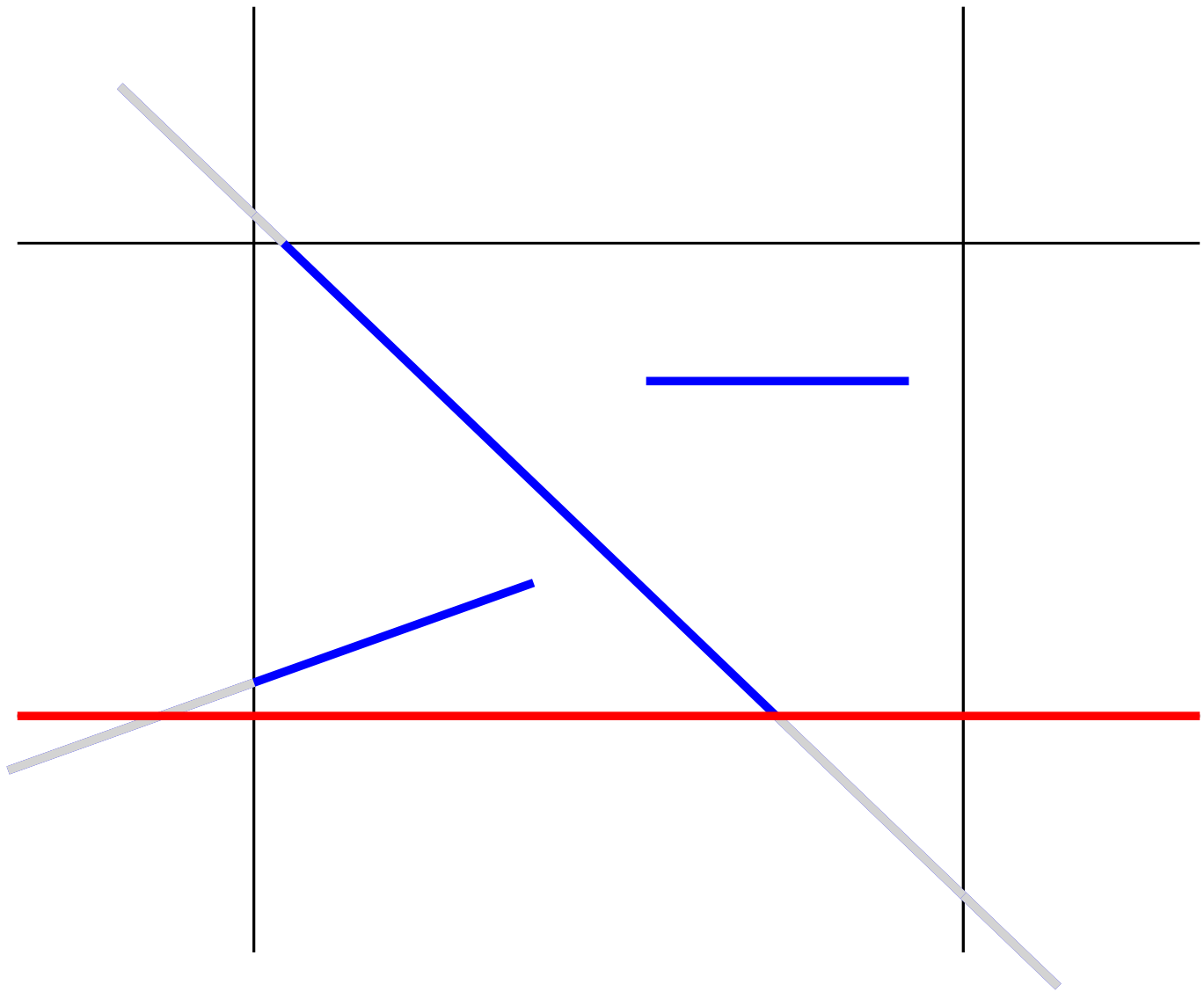
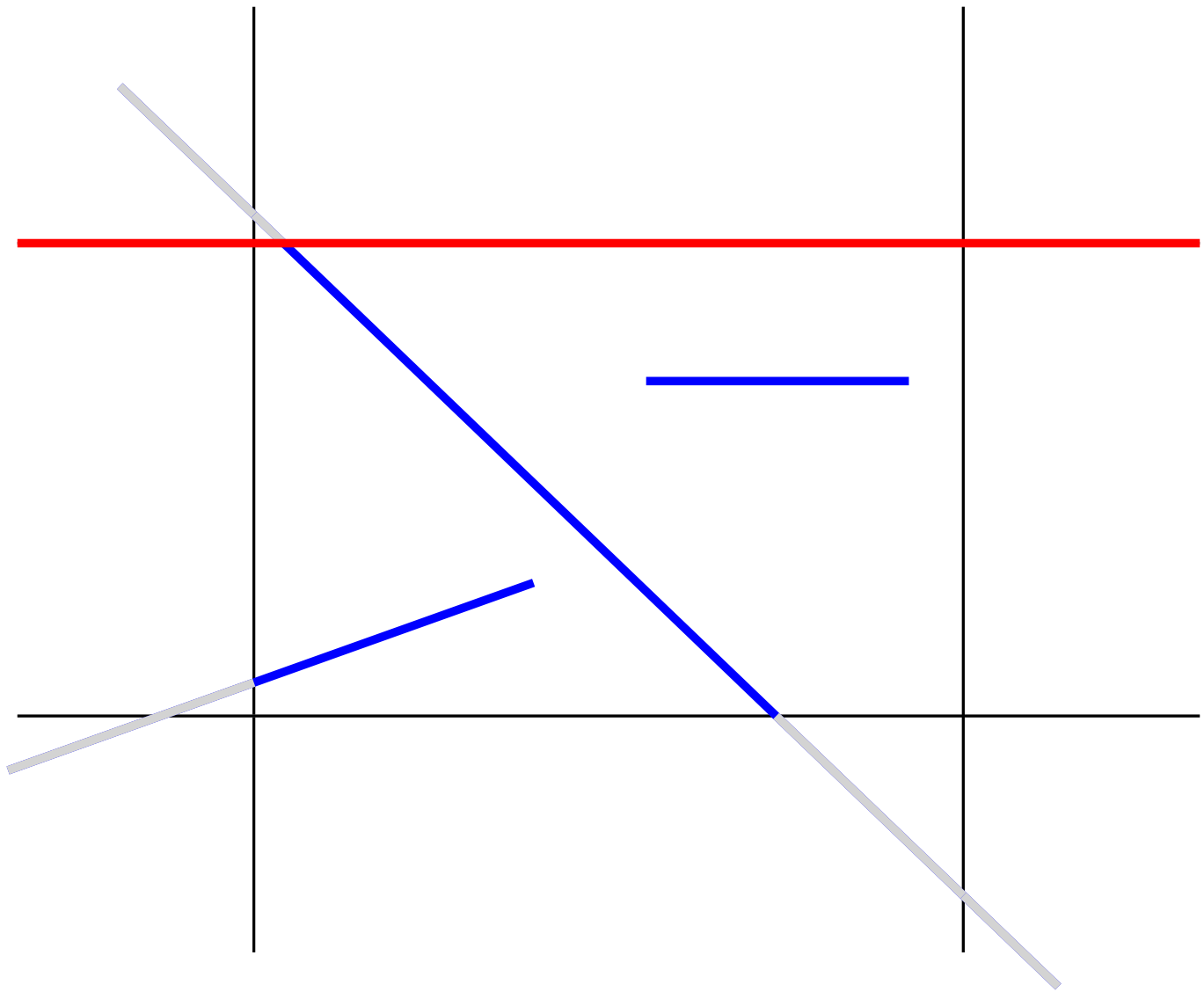
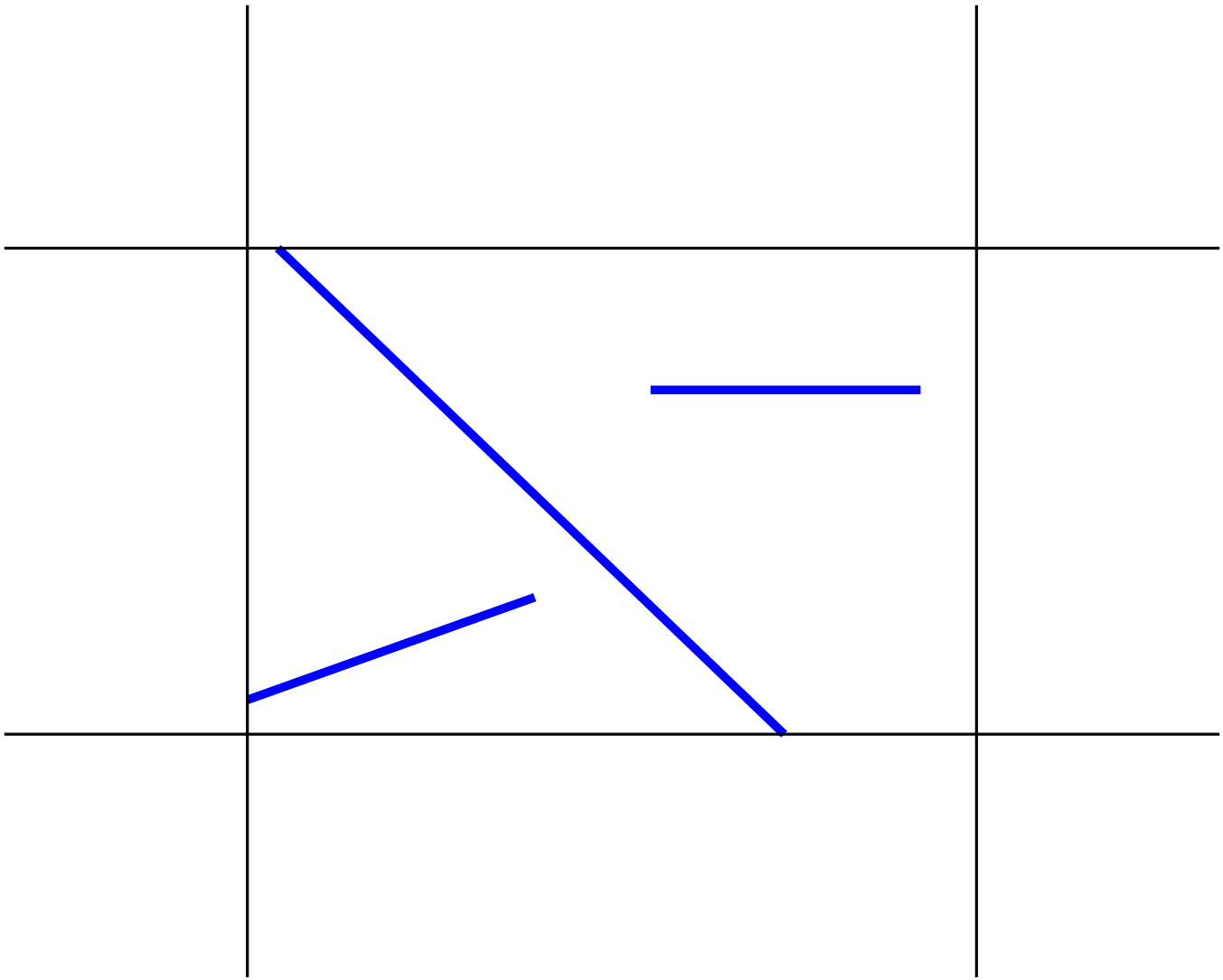


Ilustração do algoritmo



Algoritmo de Cohen-Sutherland

Resultado final:



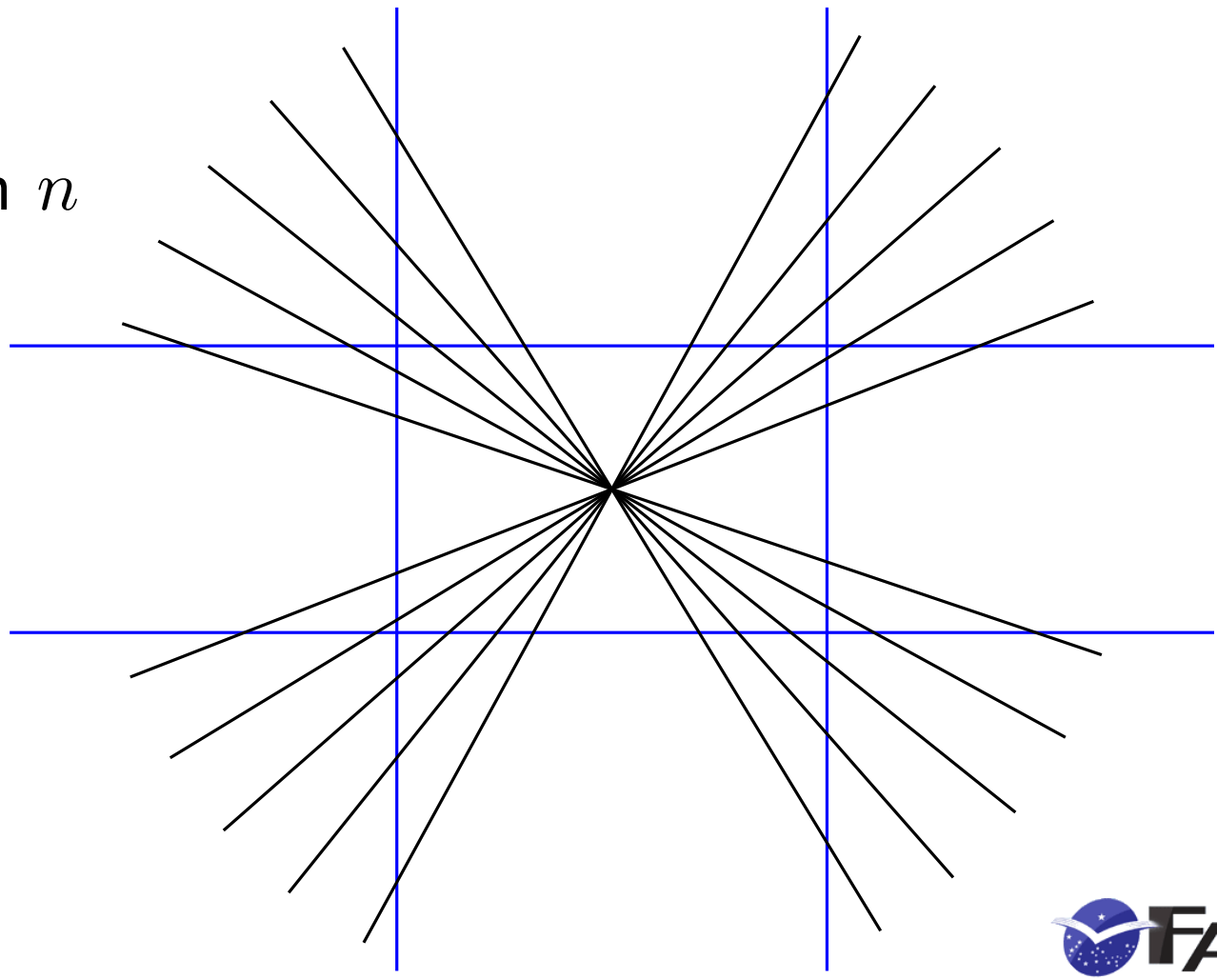
Algoritmo de Liang-Barsky

O cálculo excessivo de interseções pode tornar o algoritmo de Cohen-Sutherland ineficiente para algumas instâncias

Algoritmo de Liang-Barsky

O cálculo excessivo de interseções pode tornar o algoritmo de Cohen-Sutherland ineficiente para algumas instâncias

Exemplo de configuração com n segmentos

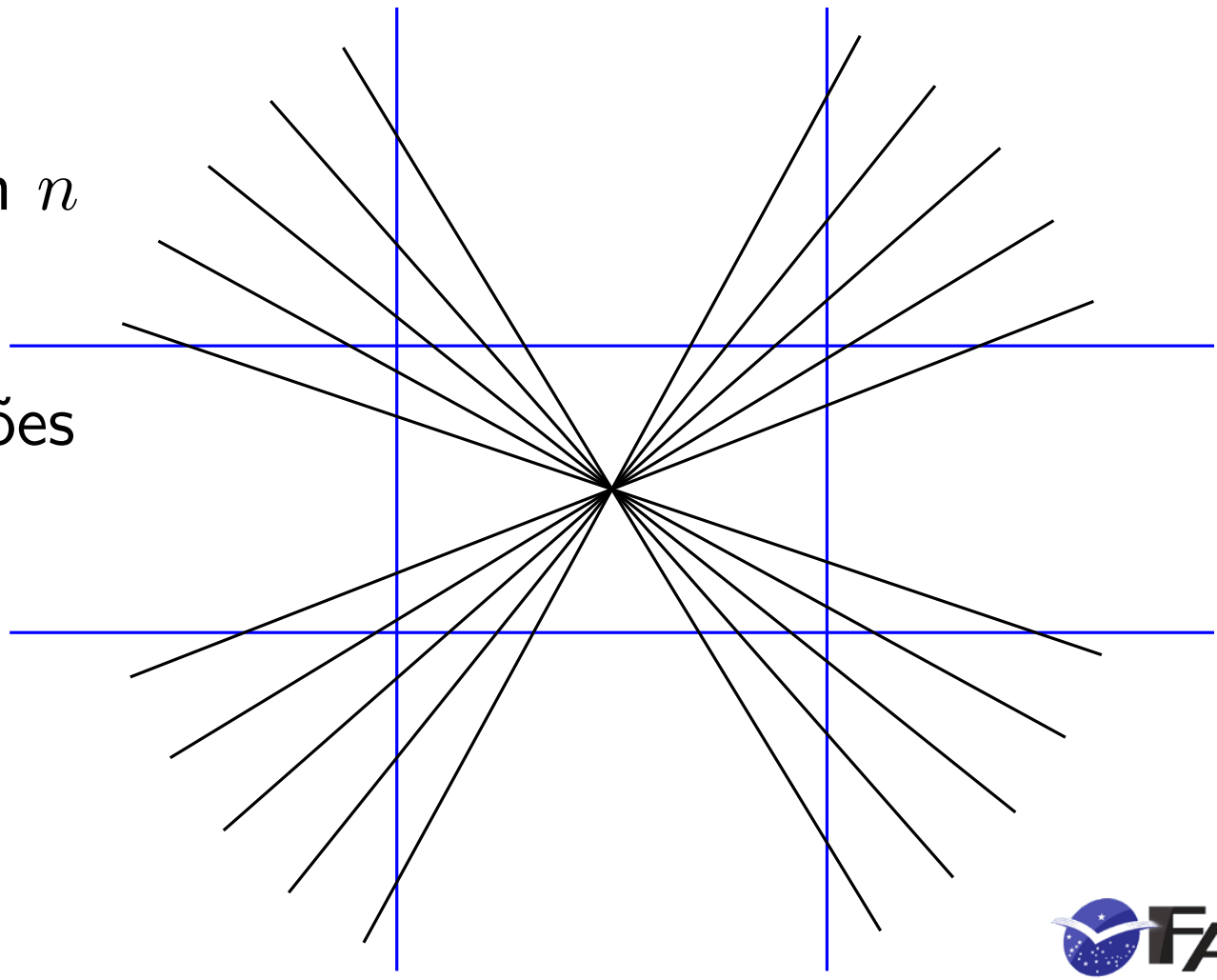


Algoritmo de Liang-Barsky

O cálculo excessivo de interseções pode tornar o algoritmo de Cohen-Sutherland ineficiente para algumas instâncias

Exemplo de configuração com n segmentos

Com $4n$ interseções



Algoritmo de Liang-Barsky

- Realiza testes para evitar ao máximo o cálculo de interseções sem utilidade (evitar divisões em pt. flutuante)

Algoritmo de Liang-Barsky

- Realiza testes para evitar ao máximo o cálculo de interseções sem utilidade (evitar divisões em pt. flutuante)
- Utiliza a equação paramétrica da reta para representar os segmentos

$$f(t) = (1 - t)P_1 + t\,P_2, \text{ onde } t \in \mathbb{R}$$

Algoritmo de Liang-Barsky

- Realiza testes para evitar ao máximo o cálculo de interseções sem utilidade (evitar divisões em pt. flutuante)
- Utiliza a equação paramétrica da reta para representar os segmentos

$$f(t) = (1-t)P_1 + t\,P_2, \text{ onde } t \in \mathbb{R}$$

Separando as coordenadas:

$$x(t) = (1-t)x_1 + tx_2 = x_1 + t\Delta x$$
$$y(t) = (1-t)y_1 + ty_2 = y_1 + t\Delta y$$

Algoritmo de Liang-Barsky

Condições para um ponto estar contido no interior do retângulo:

$$x_{min} \leq x_1 + t\Delta x \leq x_{max}$$

$$y_{min} \leq y_1 + t\Delta y \leq y_{max}$$

Essas duas desigualdades podem ser desmembradas em 4 novas desigualdades do tipo:

$$tp_k \leq q_k, \quad k = 1, 2, 3, 4$$

onde:

$$p_1 = -\Delta x, \qquad q_1 = x_1 - x_{\text{min}}$$

$$p_2 = \Delta x, \qquad q_2 = x_{\text{max}} - x_1$$

$$p_3 = -\Delta y, \qquad q_3 = y_1 - y_{\text{min}}$$

$$p_4 = \Delta y, \qquad q_4 = y_{\text{max}} - y_1$$

Algoritmo de Liang-Barsky

- Se $p_k = 0$ para algum k, então o segmento é paralelo ao k-ésimo semi-plano
  - Se $q_k < 0$, então o segmento não é visível
  - Senão, o segmento deve ser recortado na direção perpendicular ao k-ésimo semi-plano

Algoritmo de Liang-Barsky

- Se $p_k = 0$ para algum k, então o segmento é paralelo ao k-ésimo semi-plano
  - Se $q_k < 0$, então o segmento não é visível
  - Senão, o segmento deve ser recortado na direção perpendicular ao k-ésimo semi-plano
- Se $p_k < 0$, o segmento está "vindo de fora para dentro", com relação ao k-ésimo semi-plano. Caso contrário, o sentido é "de dentro para fora"

Algoritmo de Liang-Barsky

- Se $p_k = 0$ para algum k, então o segmento é paralelo ao k-ésimo semi-plano
  - Se $q_k < 0$, então o segmento não é visível
  - Senão, o segmento deve ser recortado na direção perpendicular ao k-ésimo semi-plano
- Se $p_k < 0$, o segmento está "vindo de fora para dentro", com relação ao k-ésimo semi-plano. Caso contrário, o sentido é "de dentro para fora"
- Para $k = 1, 2, 3, 4$, calcular $t_k = q_k/p_k$

Algoritmo de Liang-Barsky

- Se $p_k = 0$ para algum k, então o segmento é paralelo ao k-ésimo semi-plano
  - Se $q_k < 0$, então o segmento não é visível
  - Senão, o segmento deve ser recortado na direção perpendicular ao k-ésimo semi-plano
- Se $p_k < 0$, o segmento está "vindo de fora para dentro", com relação ao k-ésimo semi-plano. Caso contrário, o sentido é "de dentro para fora"
- Para $k = 1, 2, 3, 4$, calcular $t_k = q_k/p_k$
- Os vértices do segmento recortado são obtidos por:
  - $t_{\min} = \max\{0, t_i\}$, $\forall i \in \{k : p_k < 0\}$
  - $t_{\max} = \min\{1, t_i\}$, $\forall i \in \{k : p_k > 0\}$

Algoritmo de Liang-Barsky

- Se $p_k = 0$ para algum k, então o segmento é paralelo ao k-ésimo semi-plano
  - Se $q_k < 0$, então o segmento não é visível
  - Senão, o segmento deve ser recortado na direção perpendicular ao k-ésimo semi-plano
- Se $p_k < 0$, o segmento está "vindo de fora para dentro", com relação ao k-ésimo semi-plano. Caso contrário, o sentido é "de dentro para fora"
- Para $k = 1, 2, 3, 4$, calcular $t_k = q_k/p_k$
- Os vértices do segmento recortado são obtidos por:
  - $t_{\min} = \max\{0, t_i\}$, $\forall i \in \{k : p_k < 0\}$
  - $t_{\max} = \min\{1, t_i\}$, $\forall i \in \{k : p_k > 0\}$
- Se $t_{\min} > t_{\max}$, então o segmento não é visível

Ilustração do algoritmo

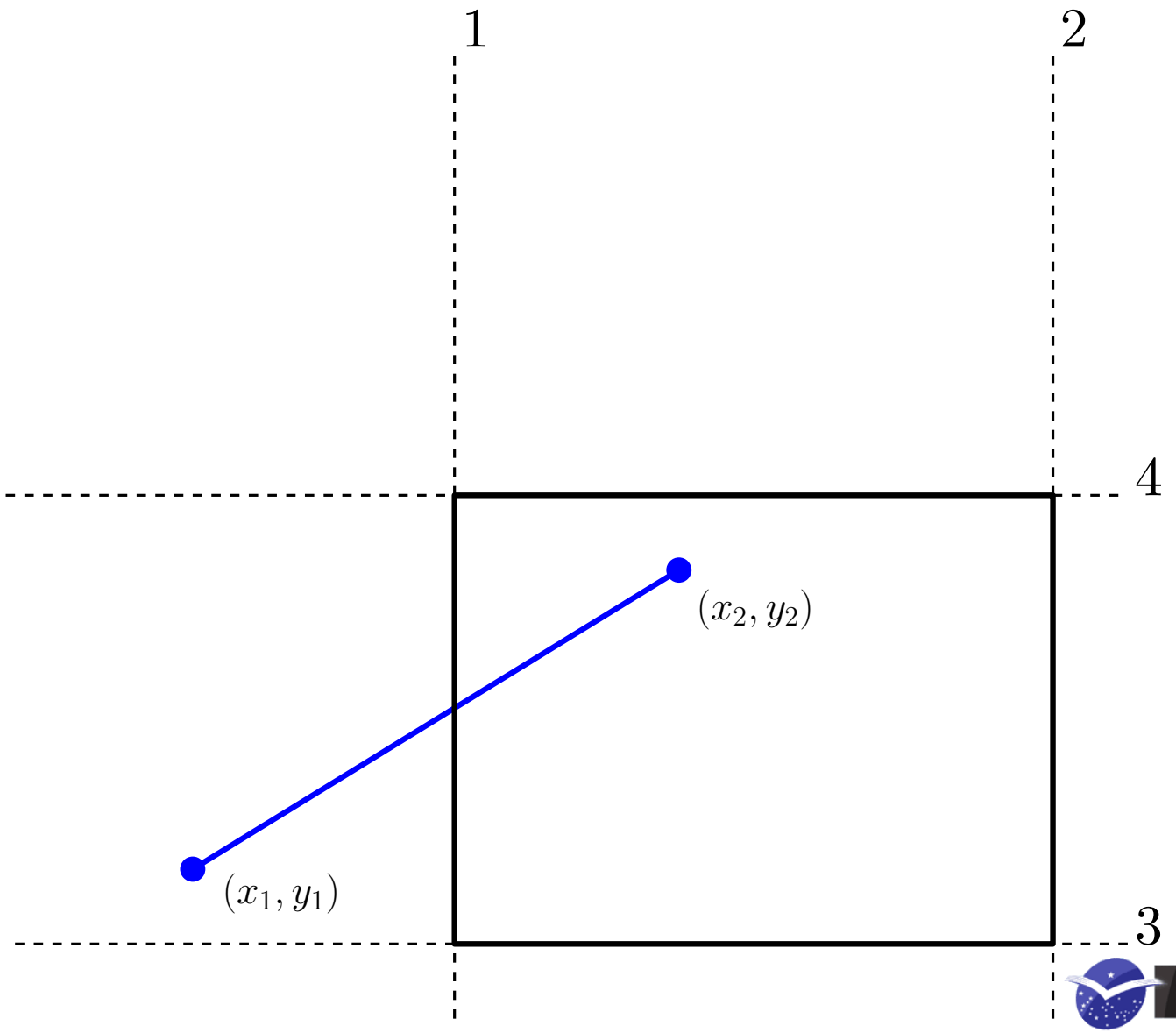


Ilustração do algoritmo

Observe que:

- $p_1 < 0$ e $p_3 < 0$
- $p_2 > 0$ e $p_4 > 0$

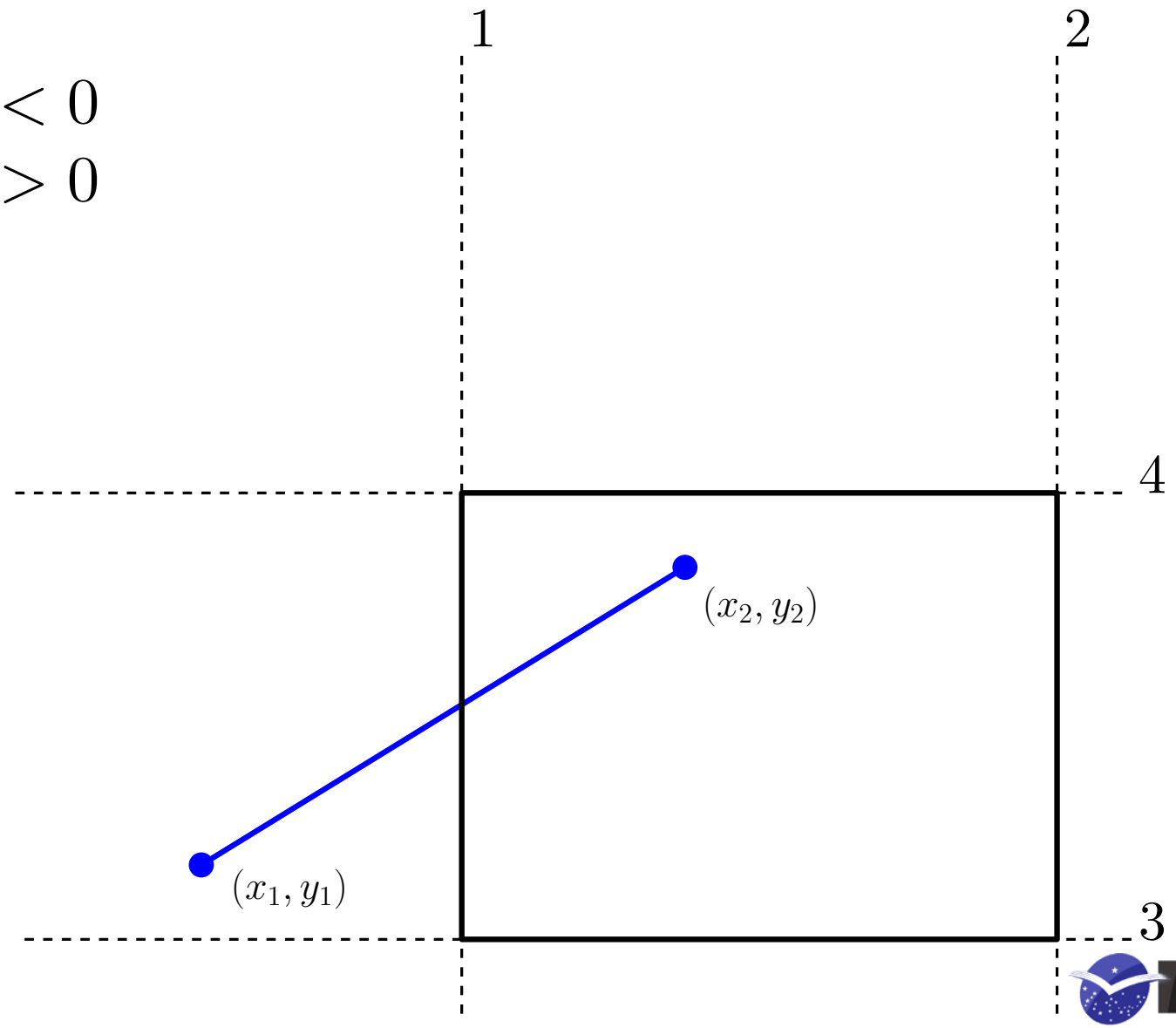


Ilustração do algoritmo

Observe que:

- $p_1 < 0$ e $p_3 < 0$
- $p_2 > 0$ e $p_4 > 0$

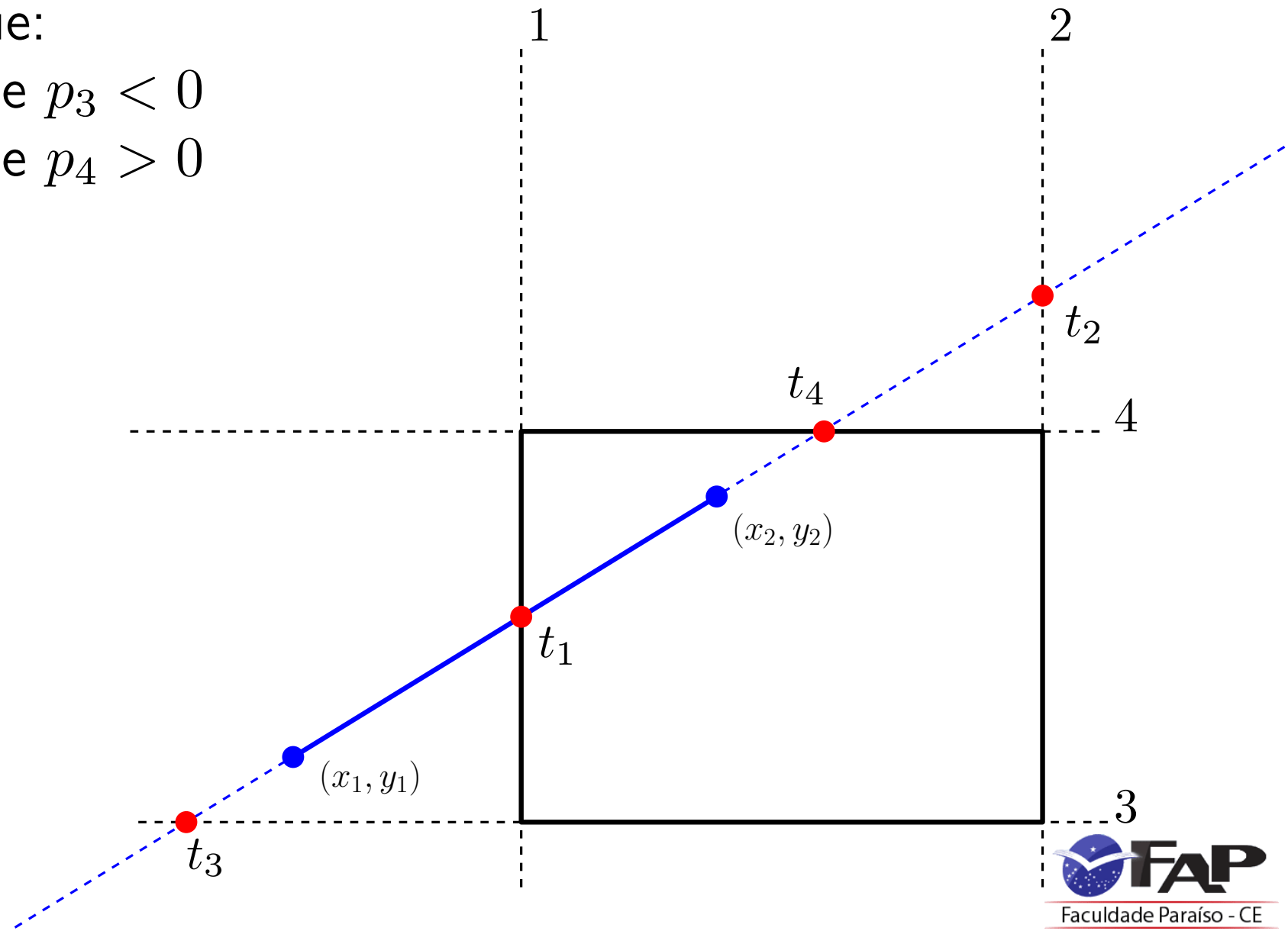


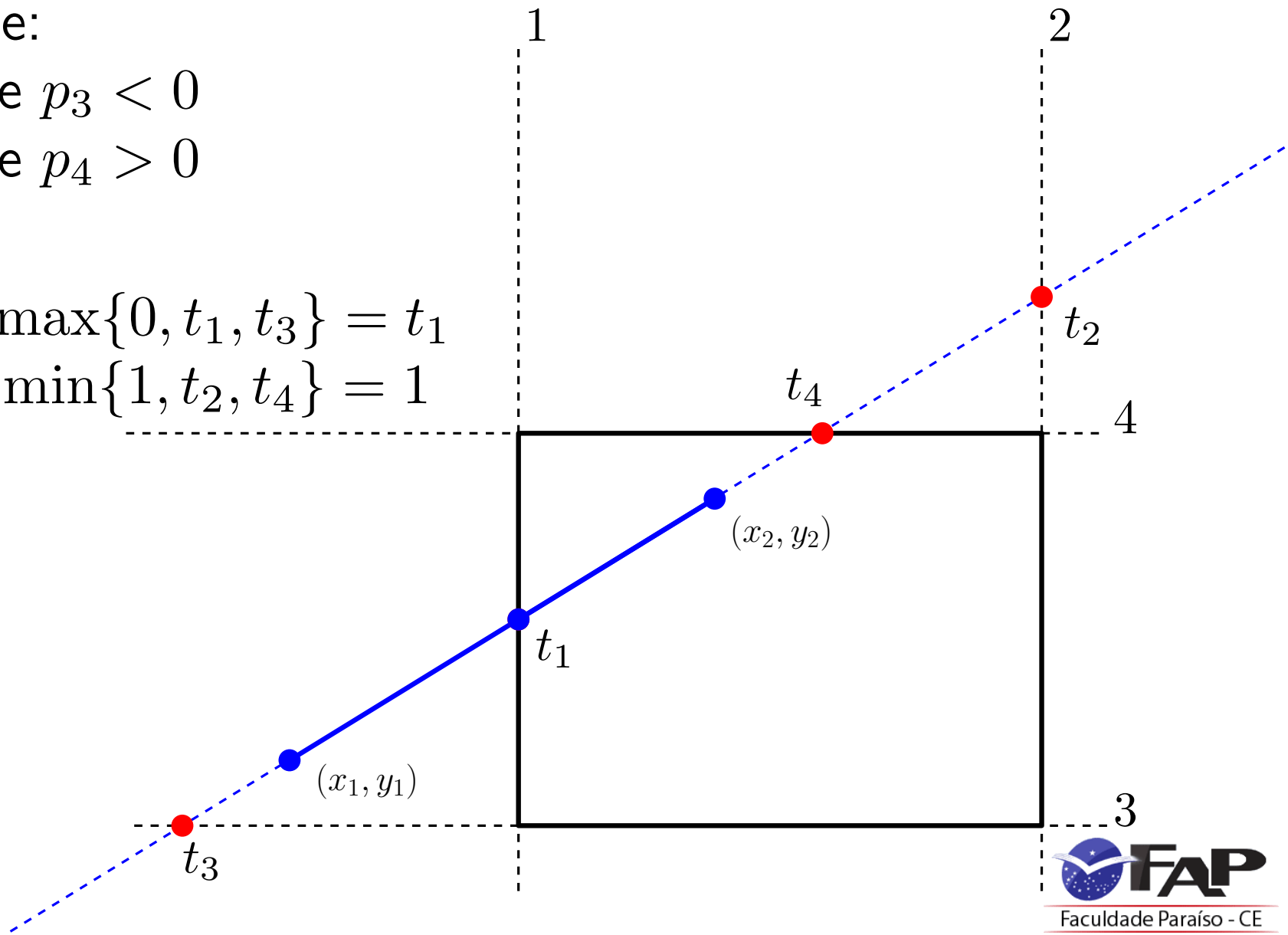
Ilustração do algoritmo

Observe que:

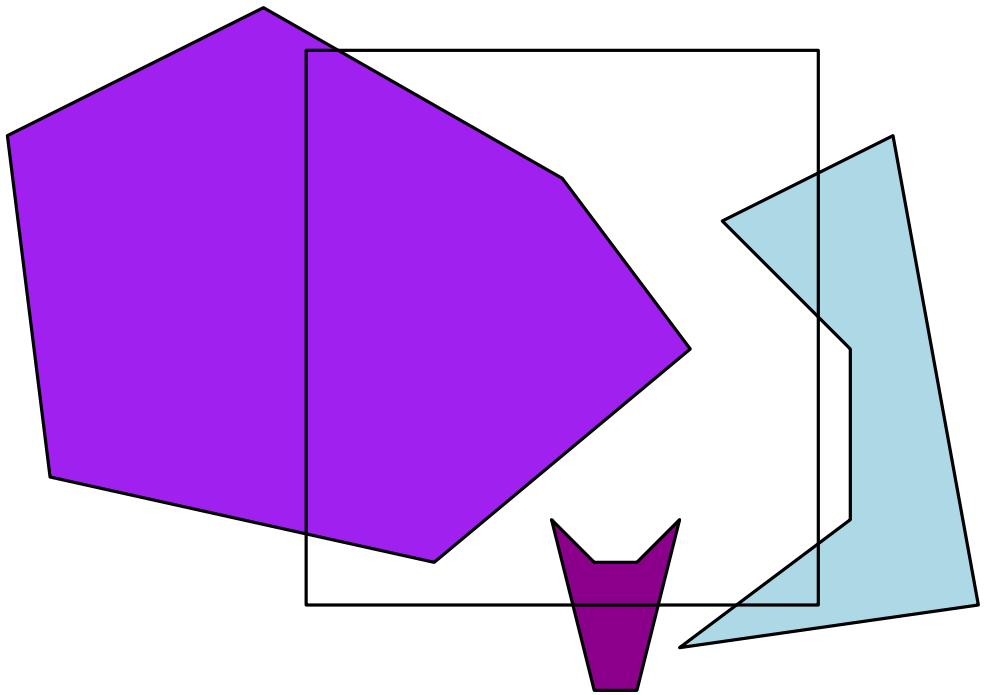
- $p_1 < 0$ e $p_3 < 0$
- $p_2 > 0$ e $p_4 > 0$

Então:

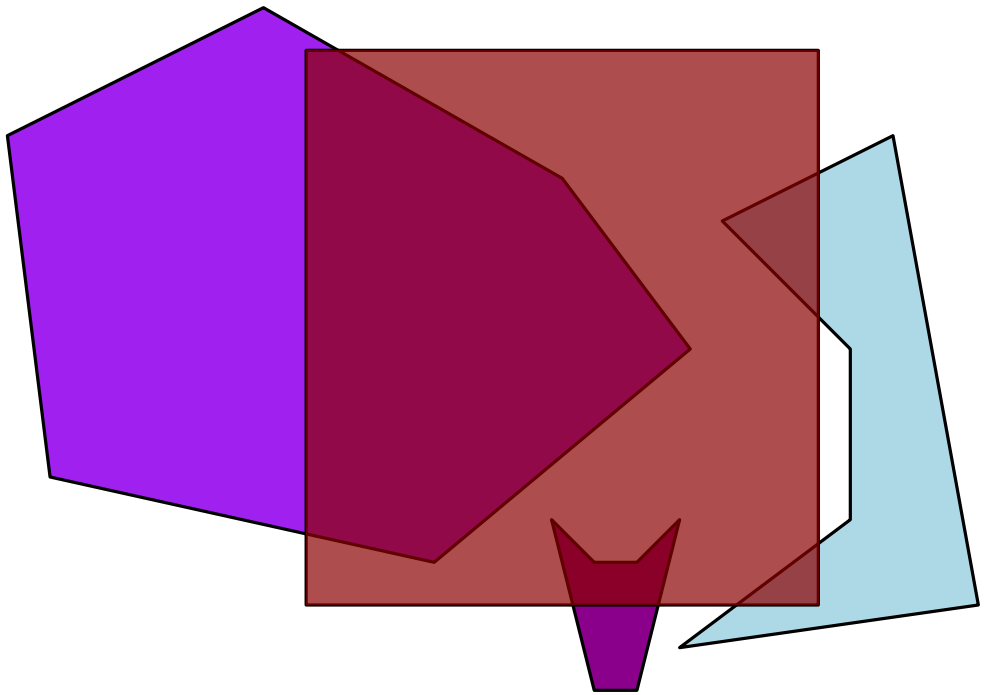
- $t_{\min} = \max\{0, t_1, t_3\} = t_1$
- $t_{\max} = \min\{1, t_2, t_4\} = 1$



Recorte de polígonos

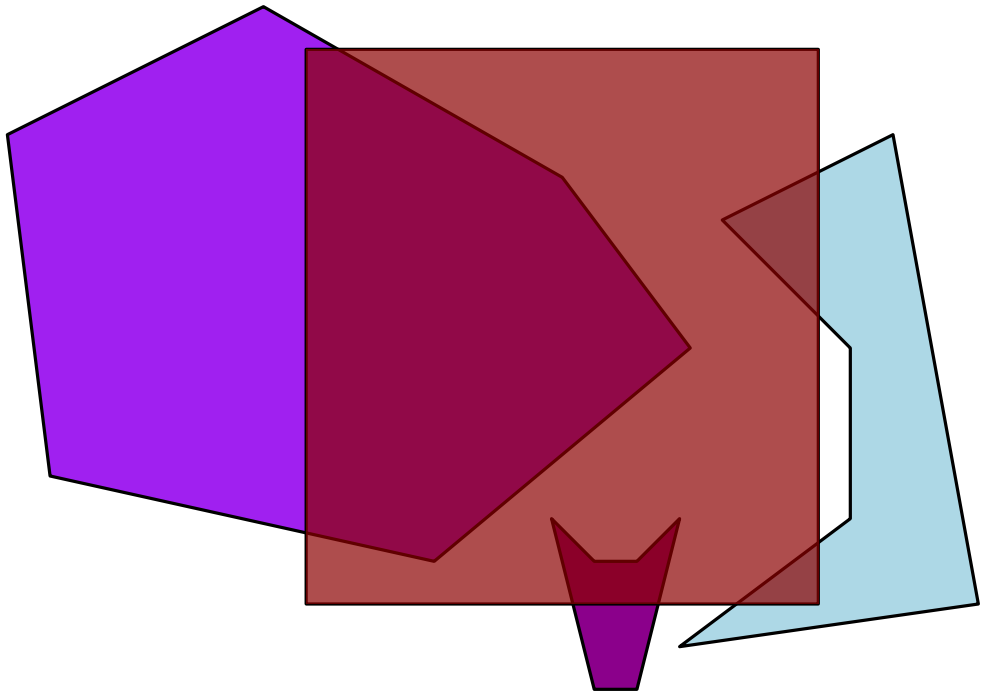


Recorte de polígonos



Pode gerar um único polígono convexo, múltiplos polígonos não-conexos ou um polígono côncavo

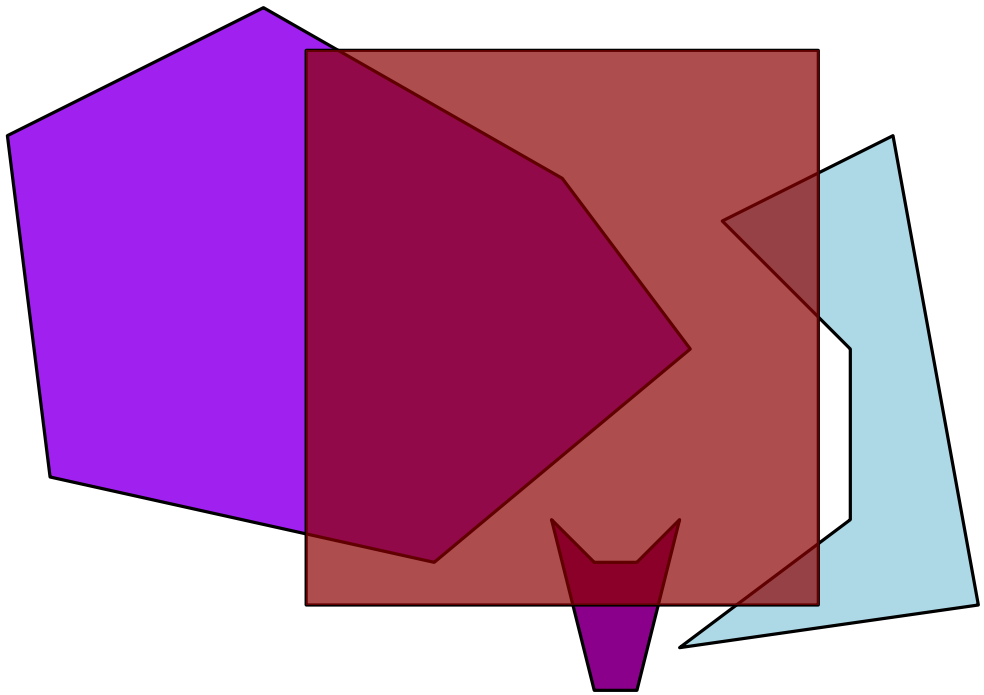
Recorte de polígonos



Pode gerar um único polígono convexo, múltiplos polígonos não-conexos ou um polígono côncavo

Se a região de recorte tiver forma arbitrária, então obteremos também uma ferramenta para realizar operações boolianas entre polígonos

Recorte de polígonos



Pode gerar um único polígono convexo, múltiplos polígonos não-conexos ou um polígono côncavo

Se a região de recorte tiver forma arbitrária, então obteremos também uma ferramenta para realizar operações boolianas entre polígonos

Principais algoritmos: Sutherland-Hodgman, Weiler-Atherton

Algoritmo de Sutherland-Hodgman

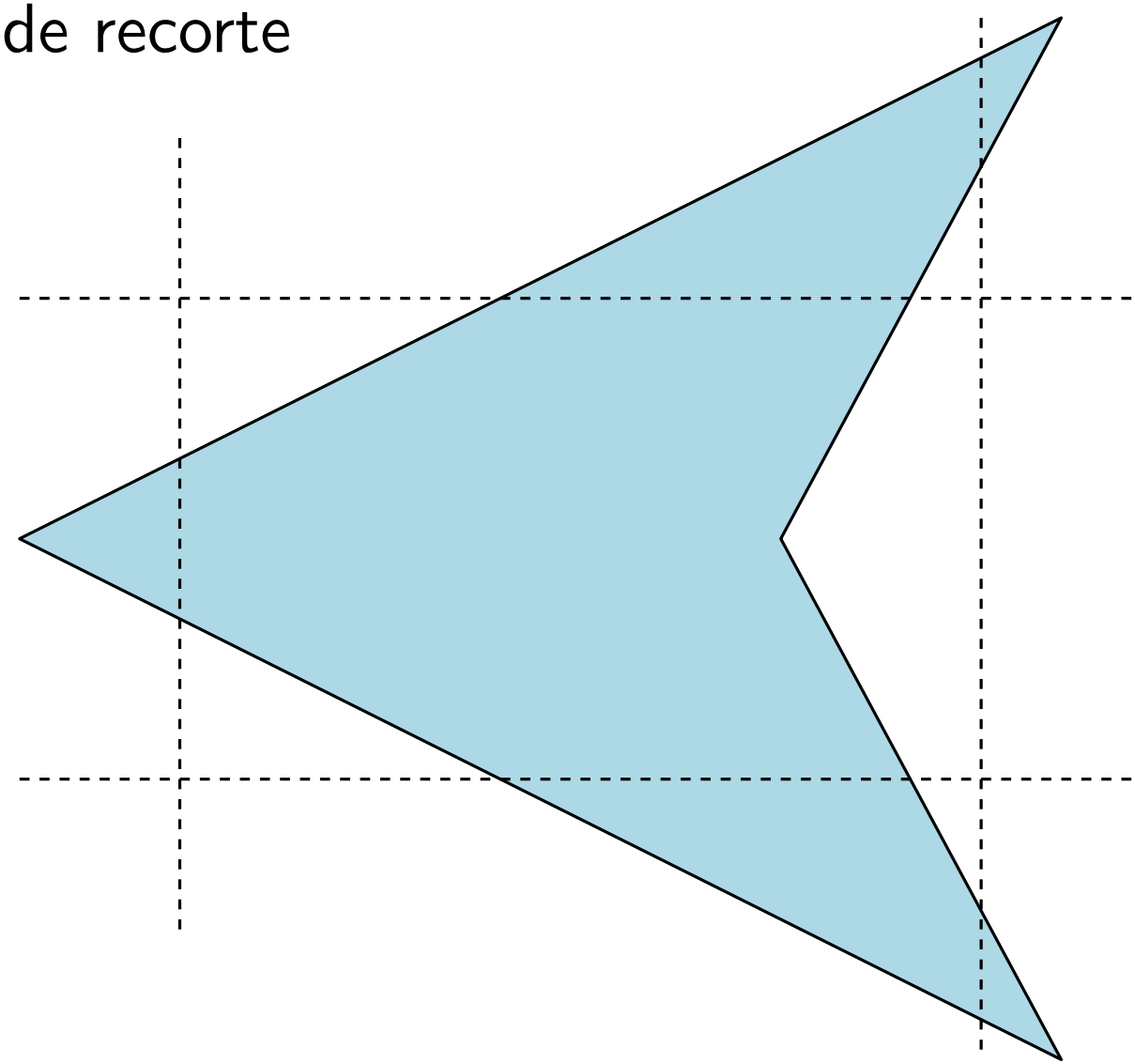
Emprega a mesma ideia do algoritmo de Cohen-Sutherland

- Recortar o polígono com cada semi-plano definido pela região de recorte

Algoritmo de Sutherland-Hodgman

Emprega a mesma ideia do algoritmo de Cohen-Sutherland

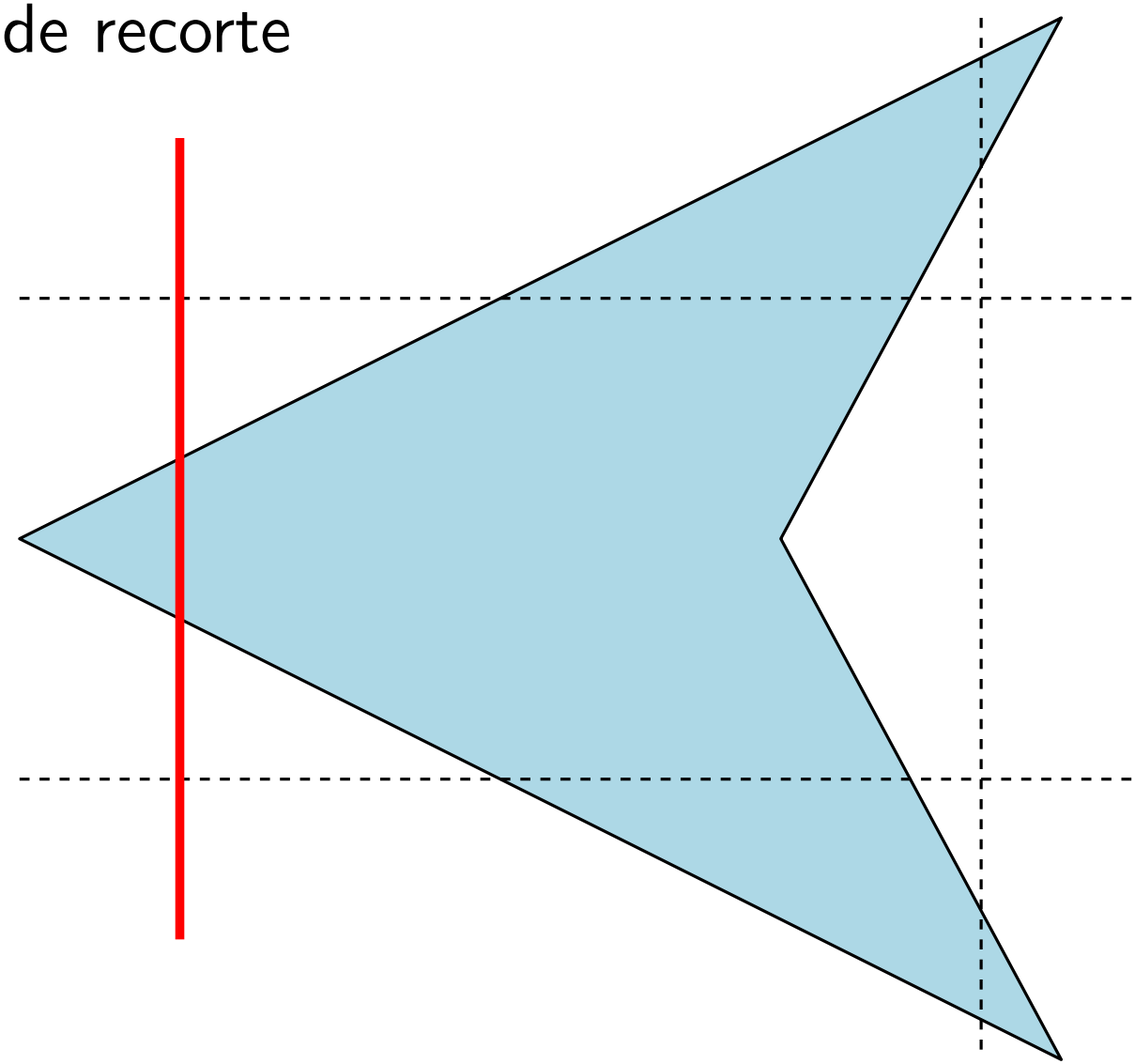
- Recortar o polígono com cada semi-plano definido pela região de recorte



Algoritmo de Sutherland-Hodgman

Emprega a mesma ideia do algoritmo de Cohen-Sutherland

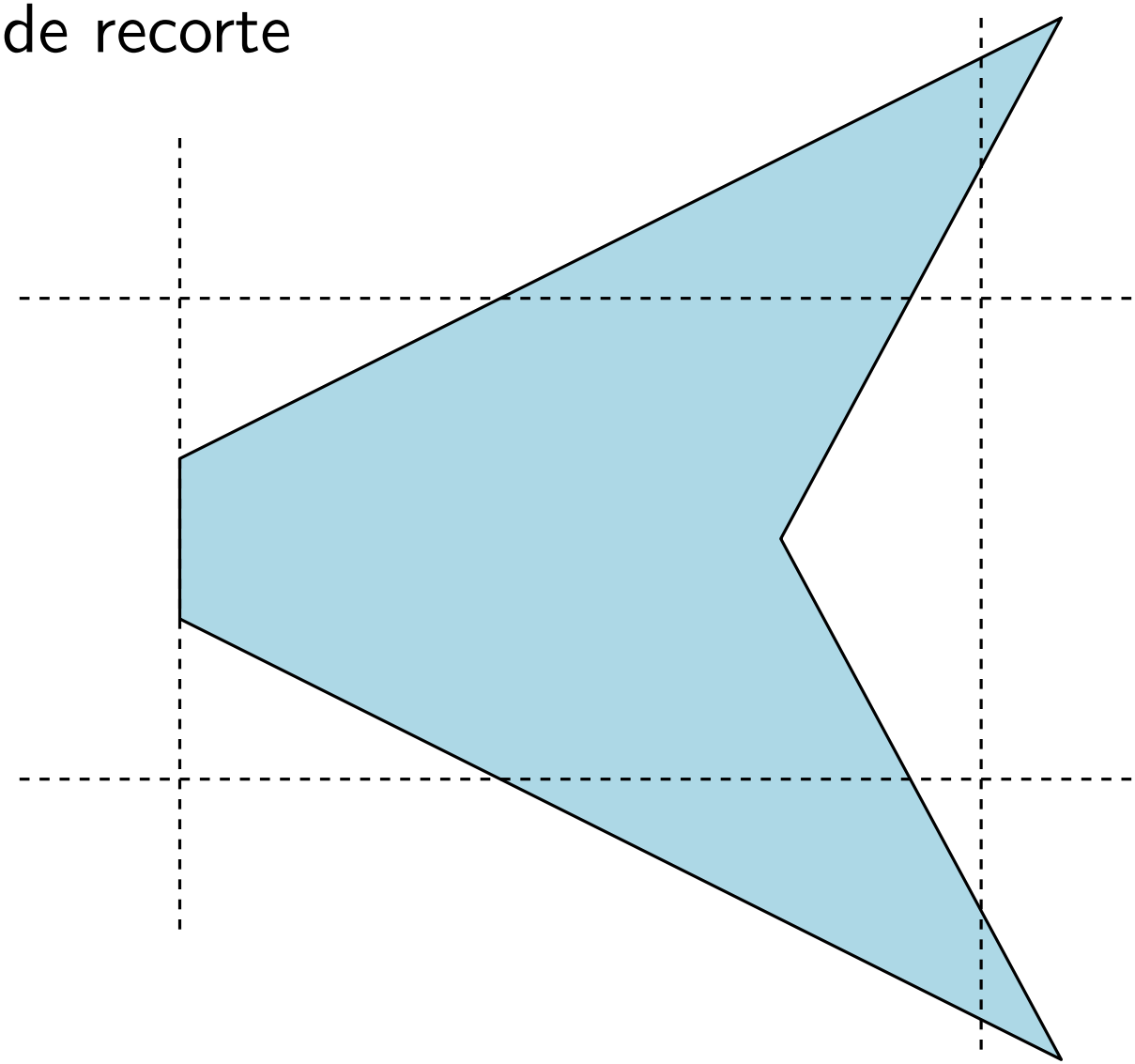
- Recortar o polígono com cada semi-plano definido pela região de recorte



Algoritmo de Sutherland-Hodgman

Emprega a mesma ideia do algoritmo de Cohen-Sutherland

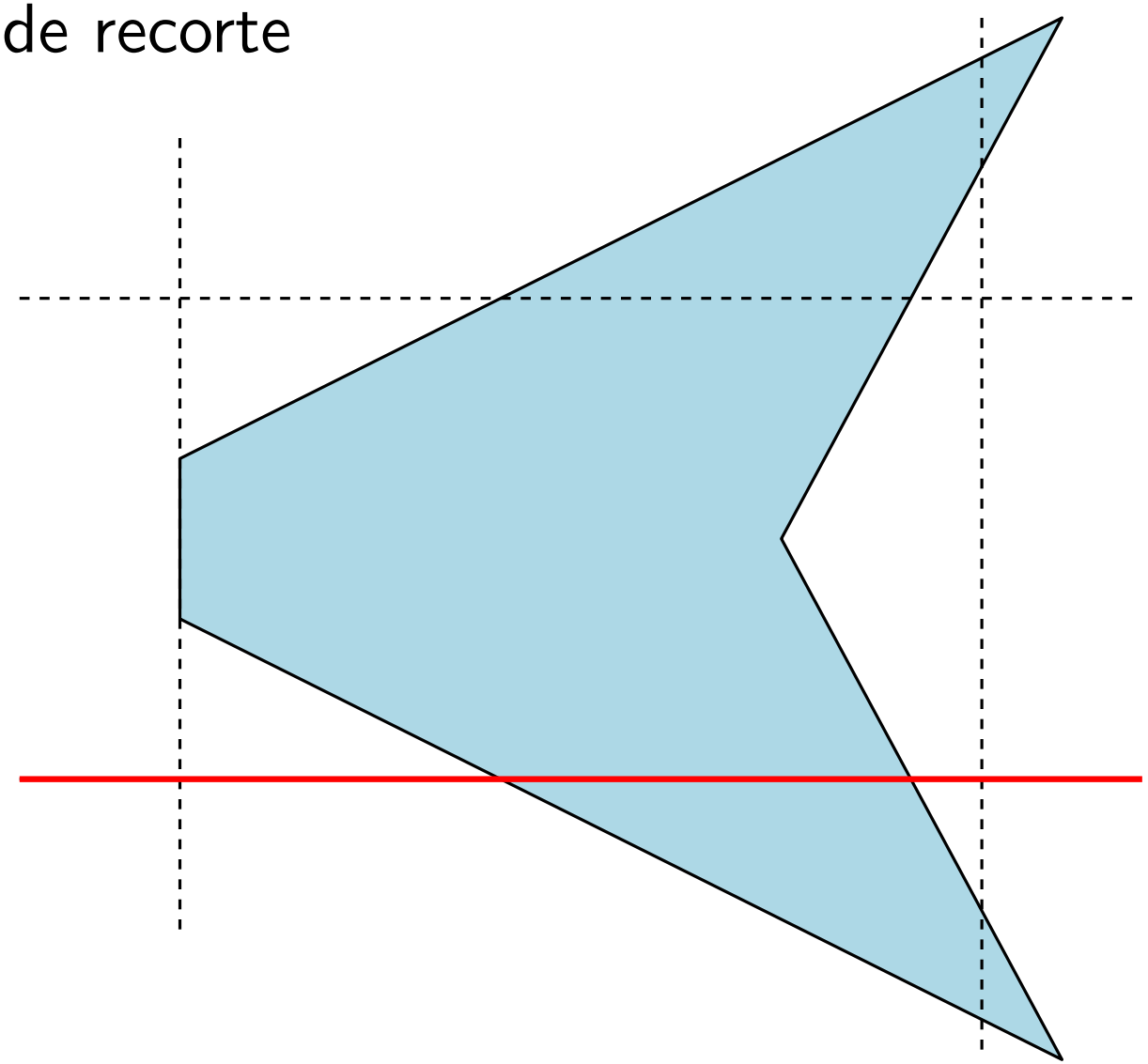
- Recortar o polígono com cada semi-plano definido pela região de recorte



Algoritmo de Sutherland-Hodgman

Emprega a mesma ideia do algoritmo de Cohen-Sutherland

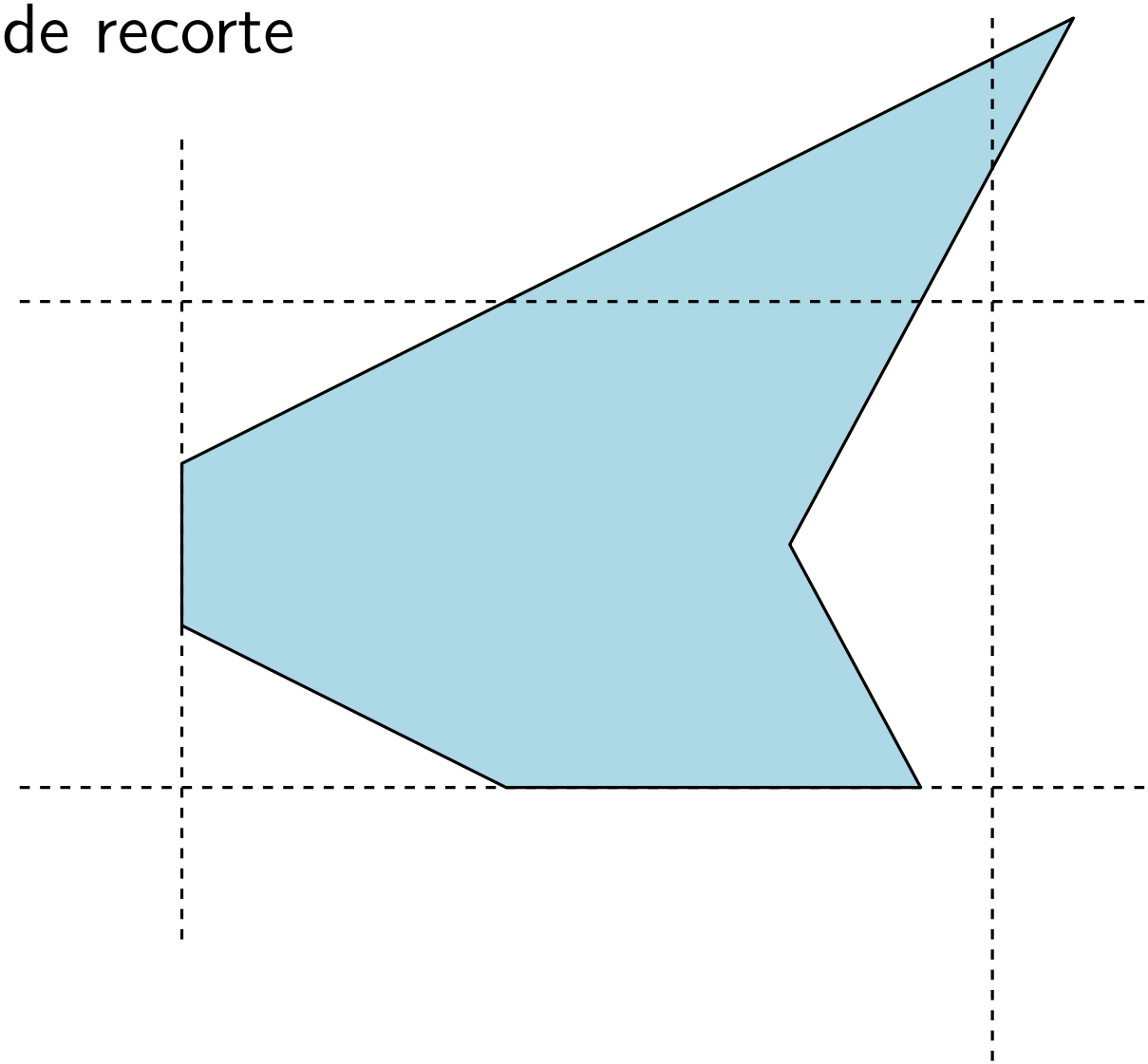
- Recortar o polígono com cada semi-plano definido pela região de recorte



Algoritmo de Sutherland-Hodgman

Emprega a mesma ideia do algoritmo de Cohen-Sutherland

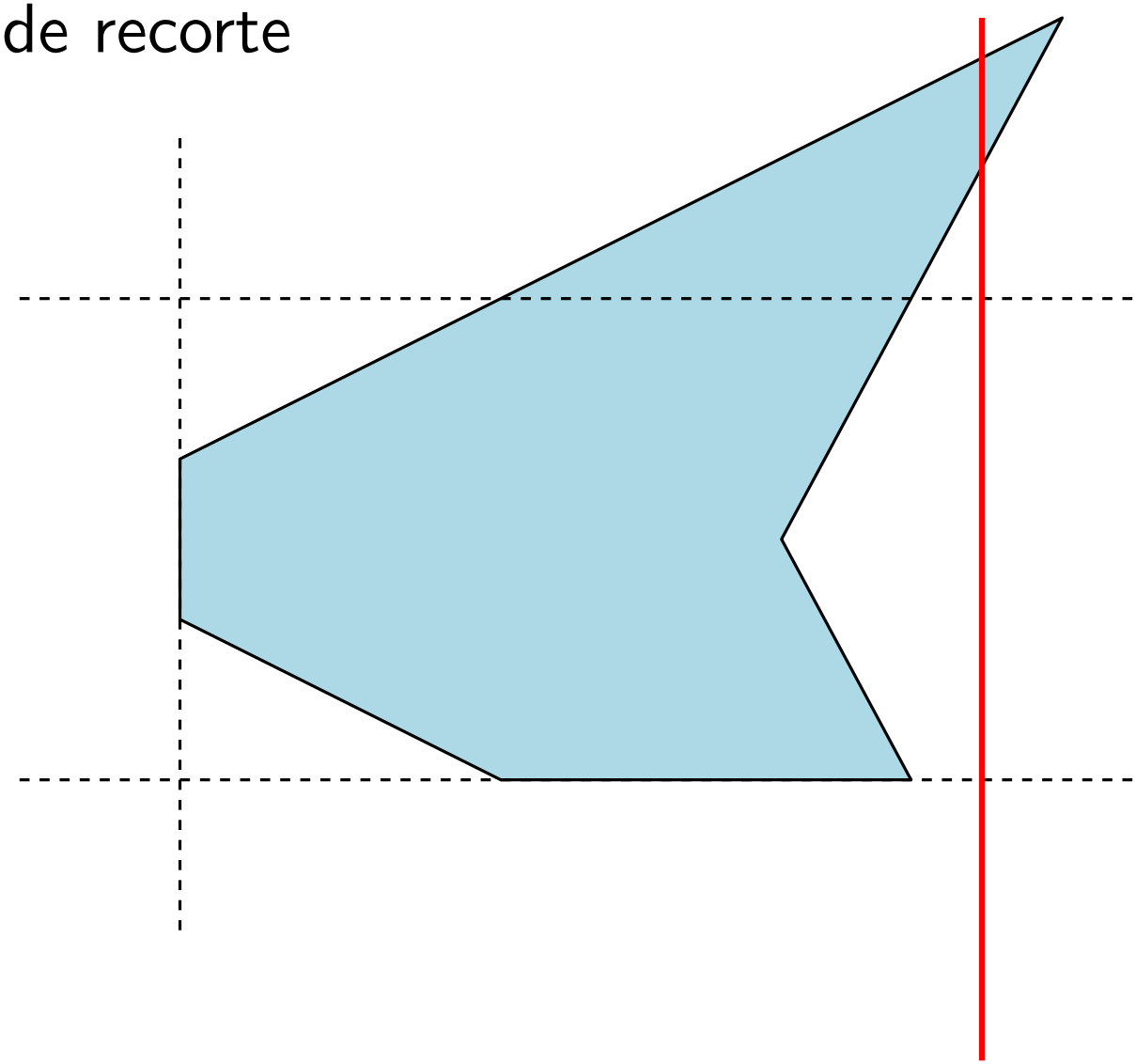
- Recortar o polígono com cada semi-plano definido pela região de recorte



Algoritmo de Sutherland-Hodgman

Emprega a mesma ideia do algoritmo de Cohen-Sutherland

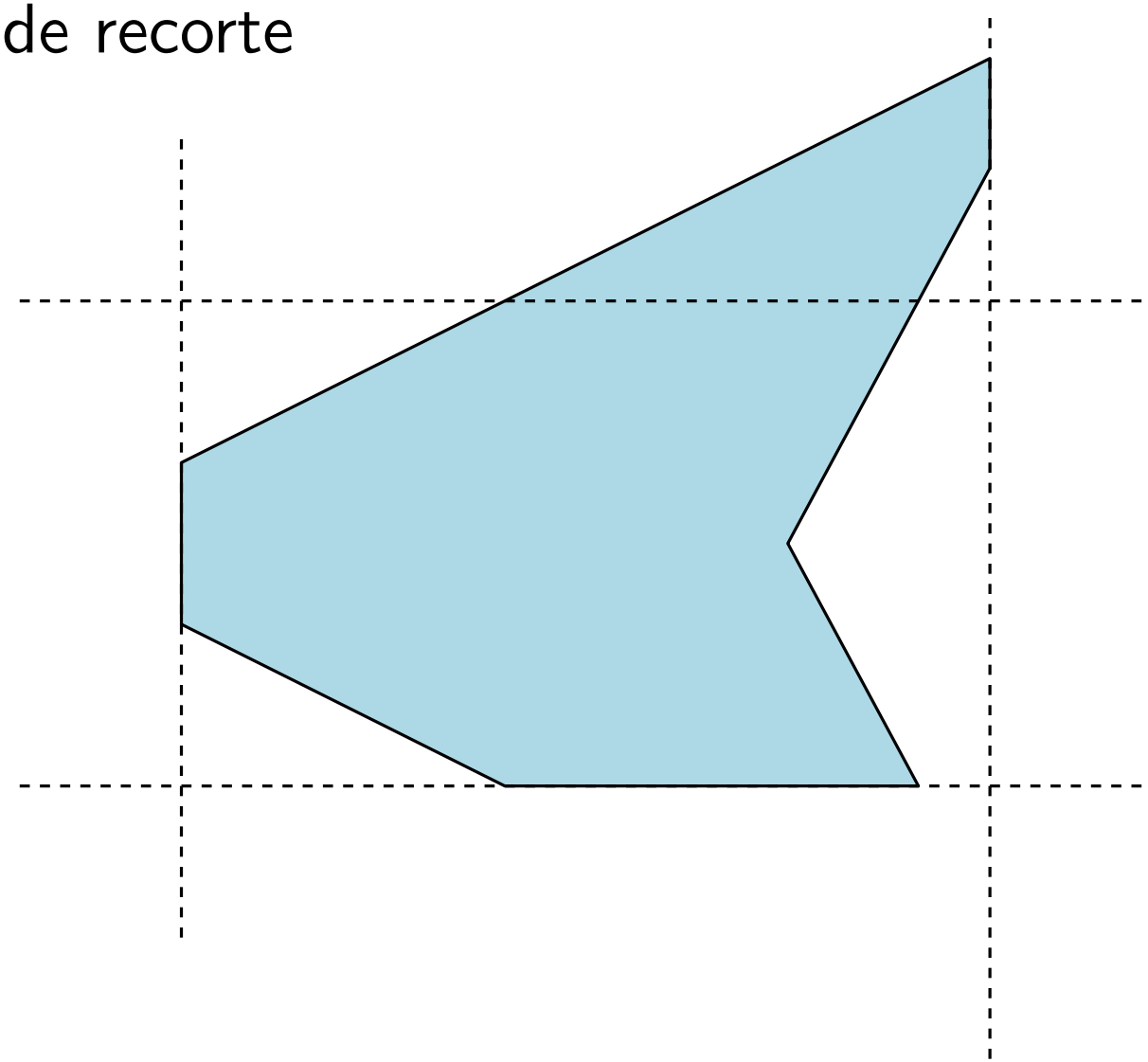
- Recortar o polígono com cada semi-plano definido pela região de recorte



Algoritmo de Sutherland-Hodgman

Emprega a mesma ideia do algoritmo de Cohen-Sutherland

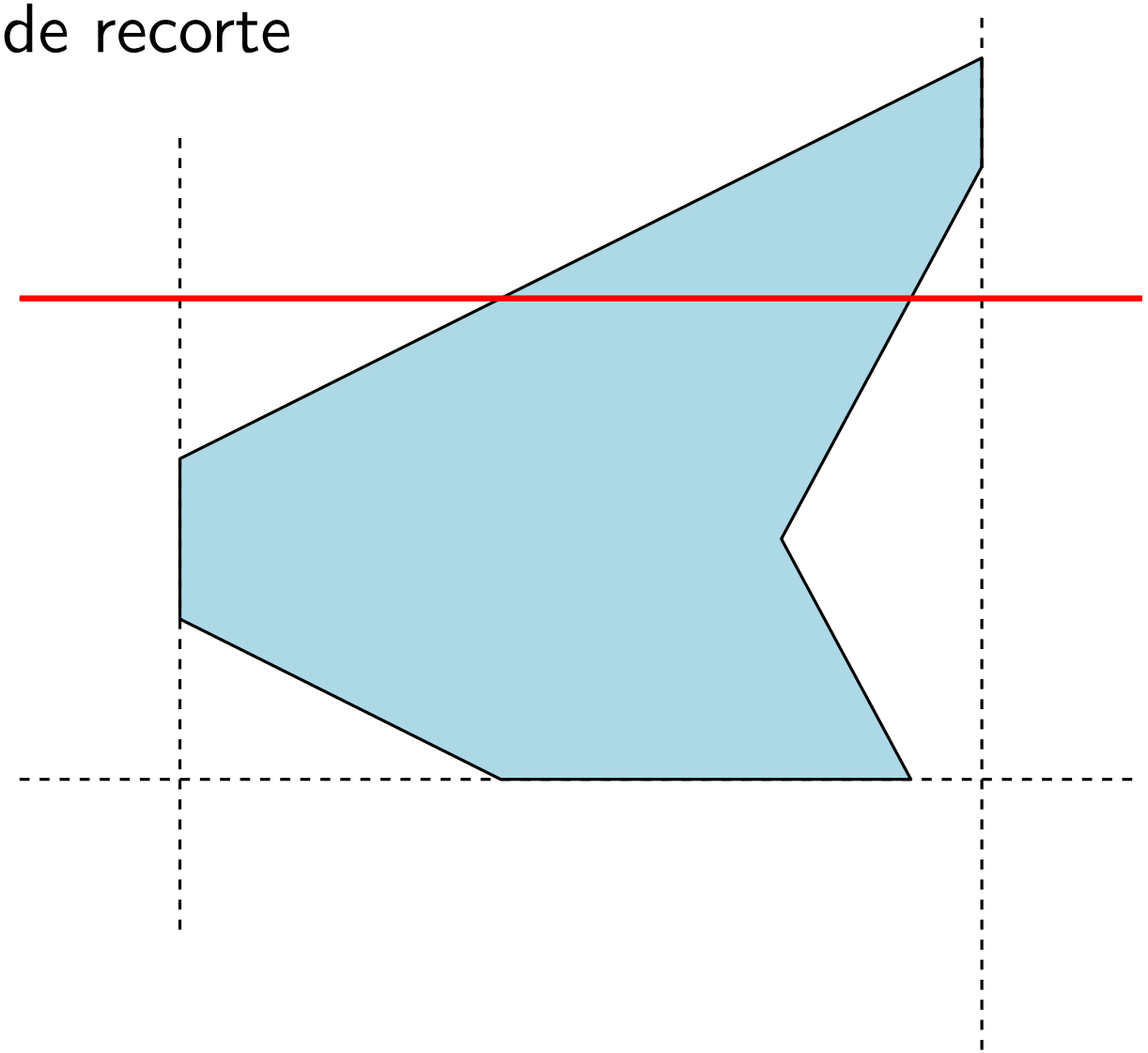
- Recortar o polígono com cada semi-plano definido pela região de recorte



Algoritmo de Sutherland-Hodgman

Emprega a mesma ideia do algoritmo de Cohen-Sutherland

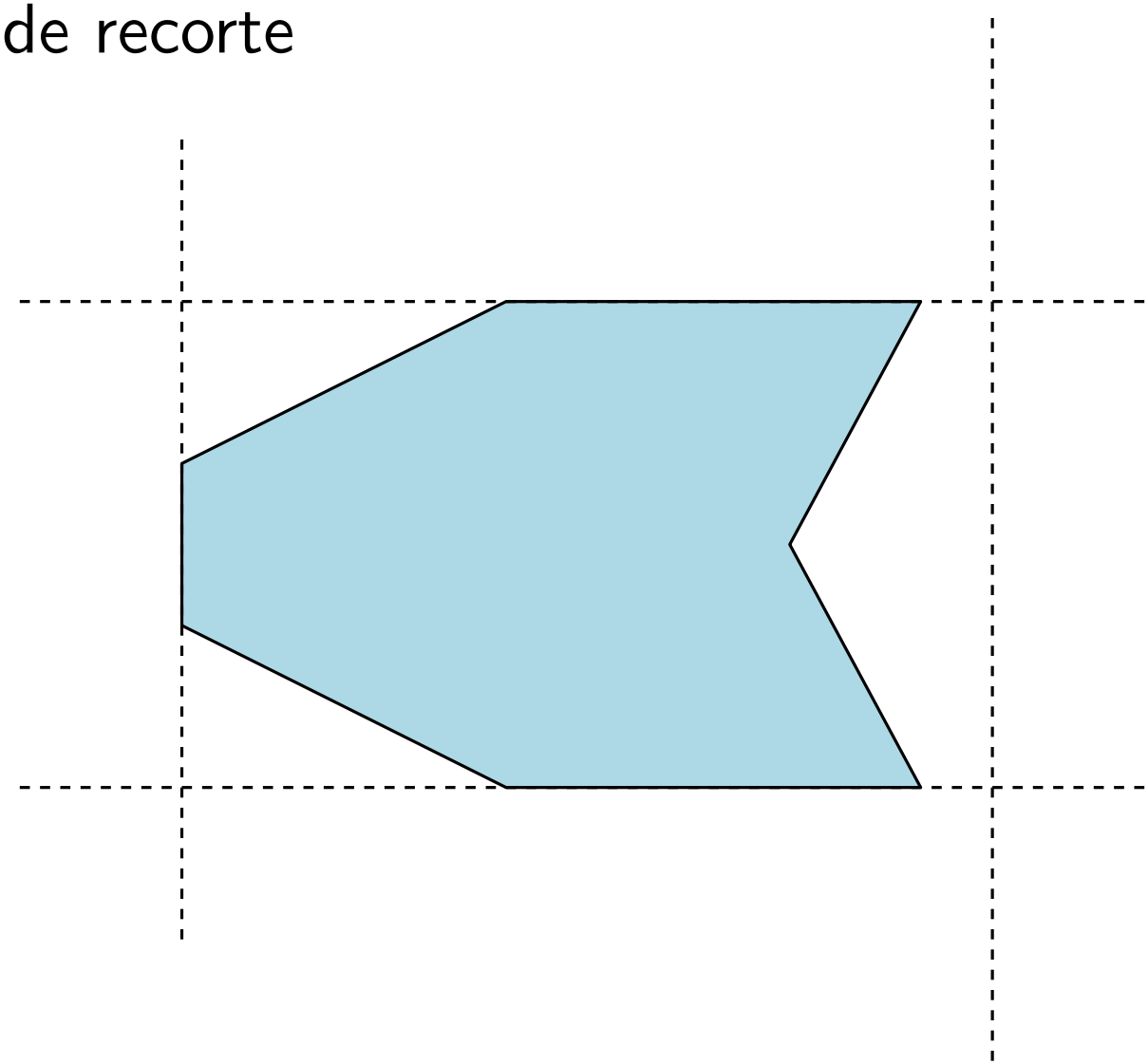
- Recortar o polígono com cada semi-plano definido pela região de recorte



Algoritmo de Sutherland-Hodgman

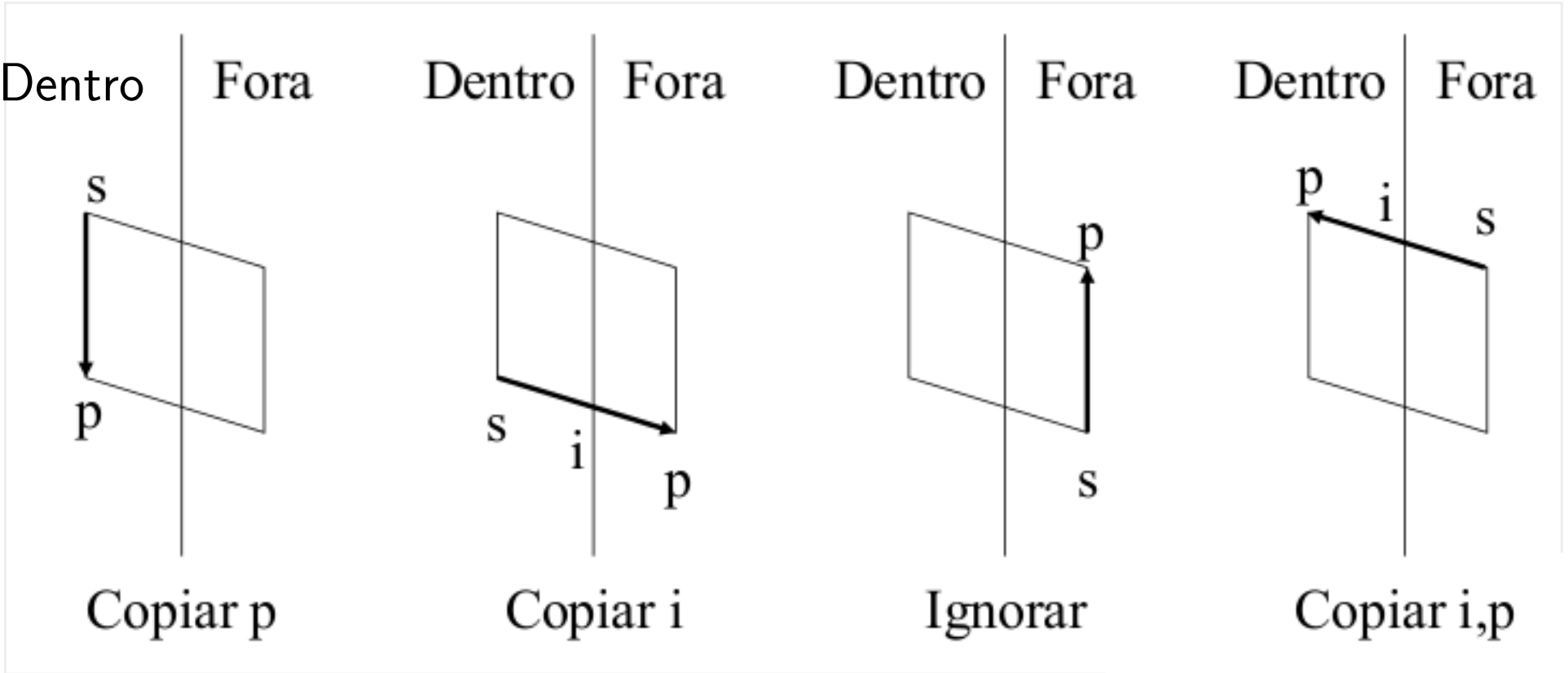
Emprega a mesma ideia do algoritmo de Cohen-Sutherland

- Recortar o polígono com cada semi-plano definido pela região de recorte



Algoritmo de Sutherland-Hodgman

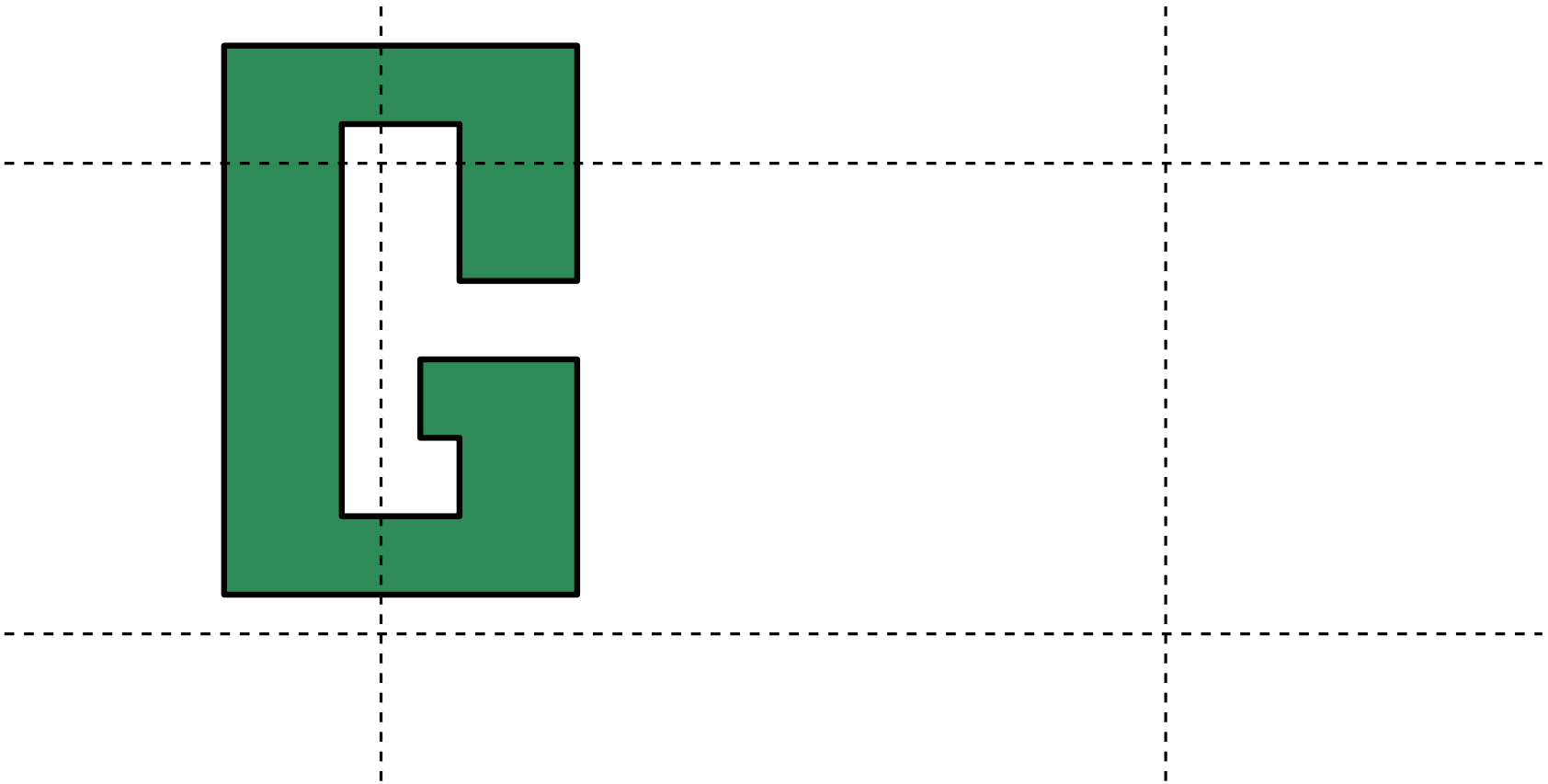
Existem quatro casos a serem tratados:



[© Esperança e Roma, 2011]

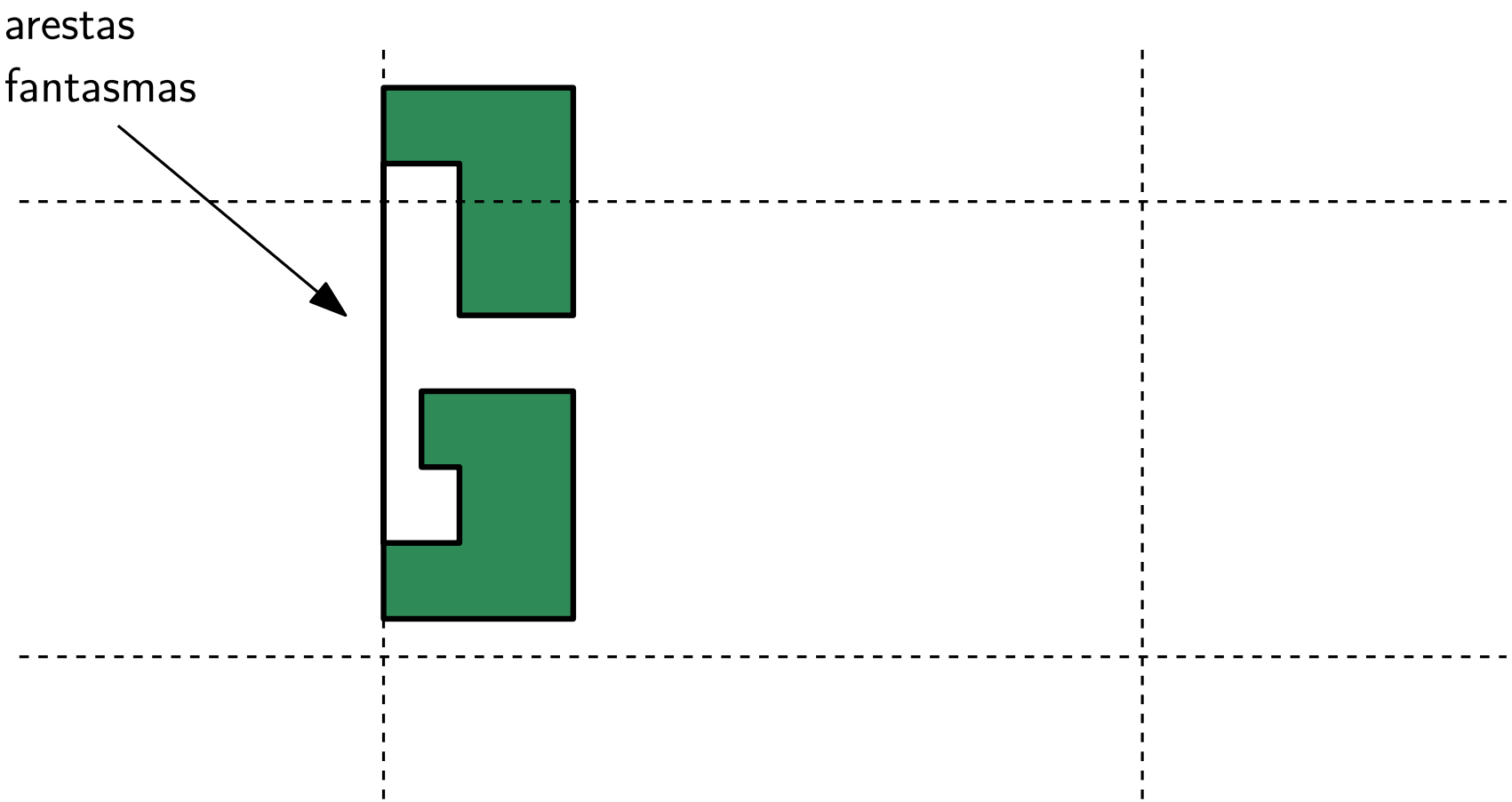
Algoritmo de Sutherland-Hodgman

Resulta em arestas “fantasmas”: não é crítico para rasterização



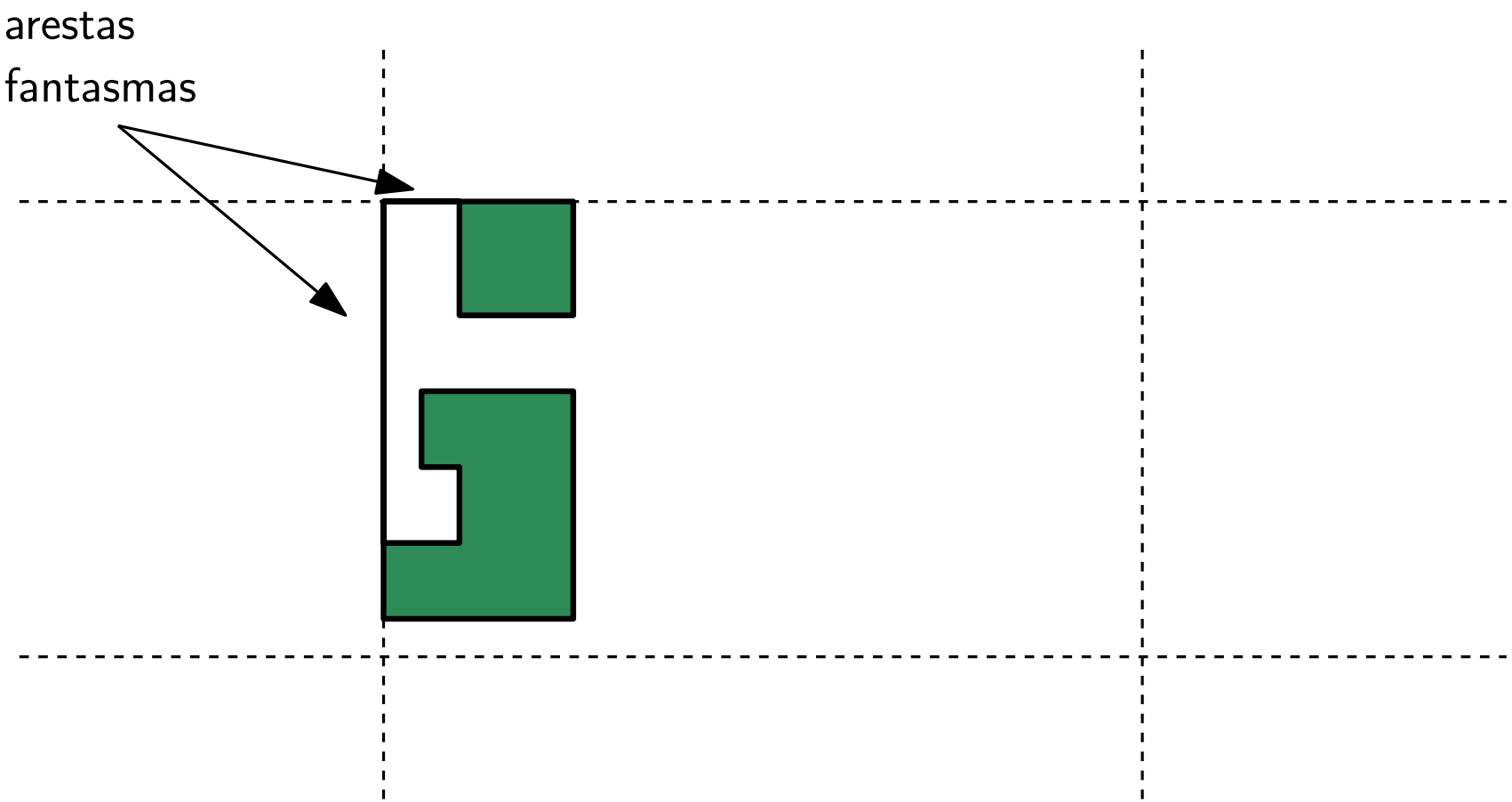
Algoritmo de Sutherland-Hodgman

Resulta em arestas "fantasmas": não é crítico para rasterização



Algoritmo de Sutherland-Hodgman

Resulta em arestas "fantasmas": não é crítico para rasterização



Algoritmo de Weiler-Atherton

- Permite recortar polígonos arbitrários contra regiões de recorte arbitrárias

Algoritmo de Weiler-Atherton

- Permite recortar polígonos arbitrários contra regiões de recorte arbitrárias
- Pode ser utilizado para realizar operações boolianas entre polígonos arbitrários

Algoritmo de Weiler-Atherton

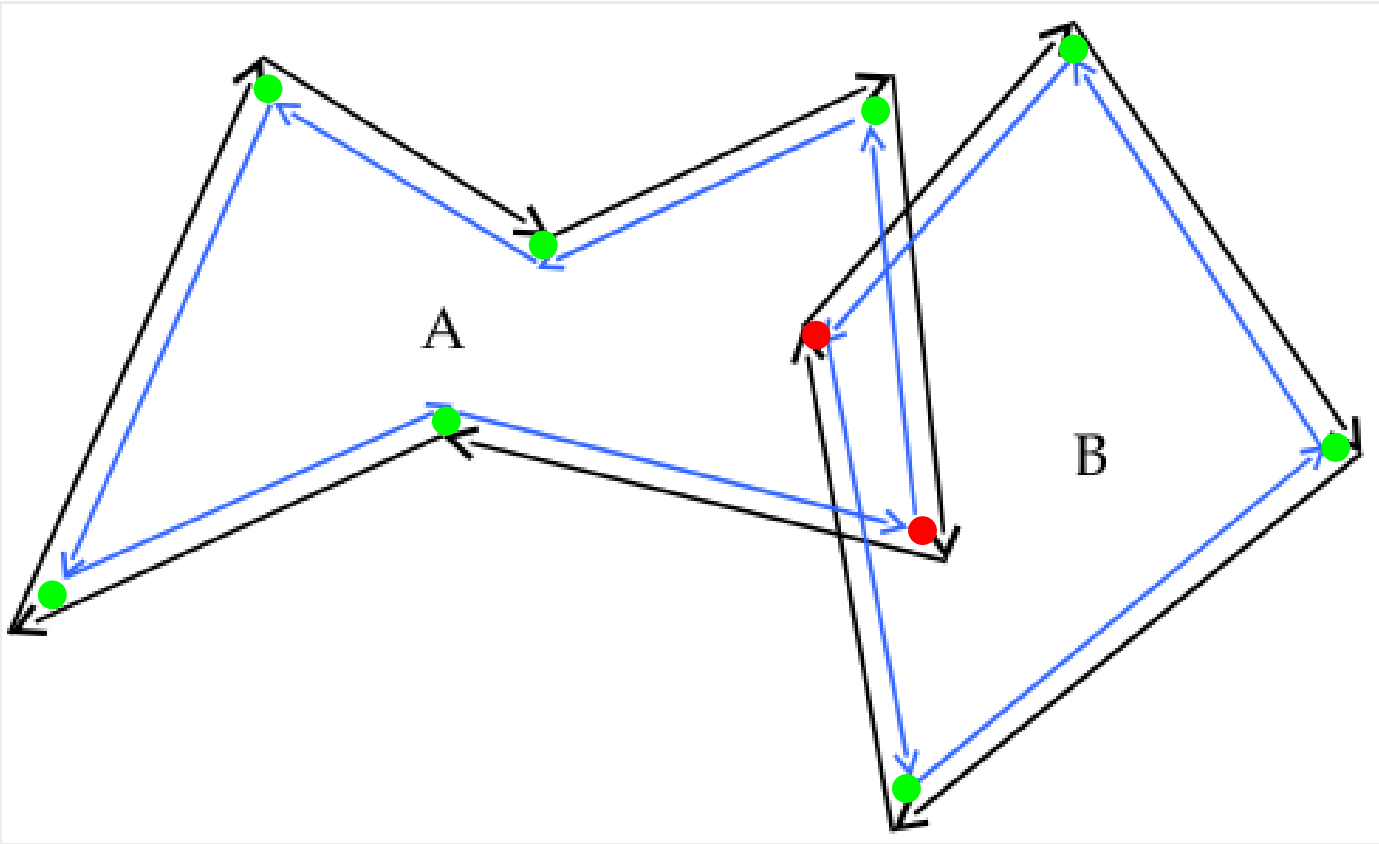
- Permite recortar polígonos arbitrários contra regiões de recorte arbitrárias
- Pode ser utilizado para realizar operações boolianas entre polígonos arbitrários
- As arestas dos polígonos são duplicadas (interior $\Longrightarrow$ sentido anti-horário, exterior $\Longrightarrow$ sentido horário)

Algoritmo de Weiler-Atherton

- Permite recortar polígonos arbitrários contra regiões de recorte arbitrárias
- Pode ser utilizado para realizar operações boolianas entre polígonos arbitrários
- As arestas dos polígonos são duplicadas (interior $\Longrightarrow$ sentido anti-horário, exterior $\Longrightarrow$ sentido horário)
- Ausência de arestas fantasmas

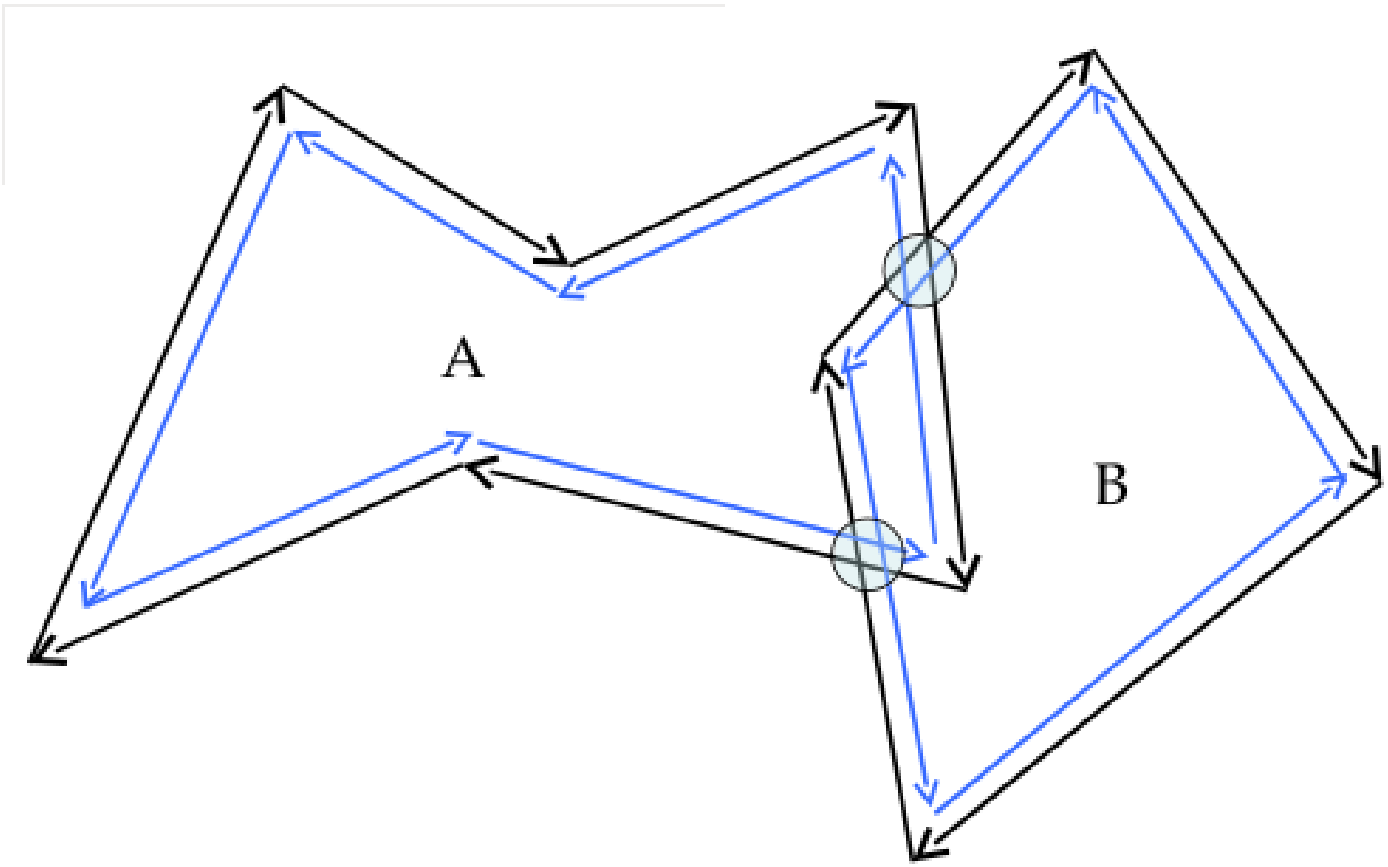
Algoritmo de Weiler-Atherton

Classificamos os vértices de A (resp. B) como dentro ou fora de B (resp. A)



[© Esperança e Roma, 2011]

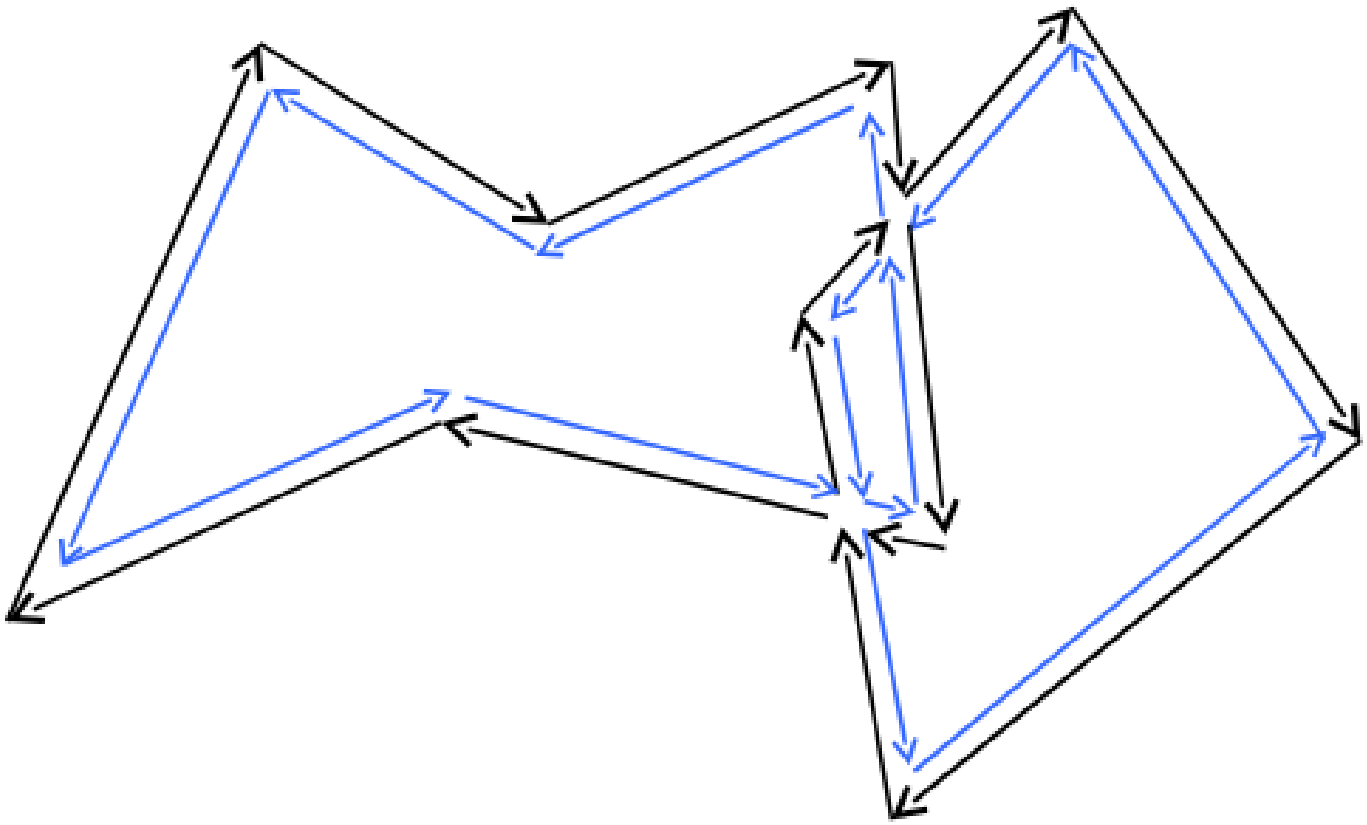
Cálculo dos pontos de interseção entre A e B



[© Esperança e Roma, 2011]

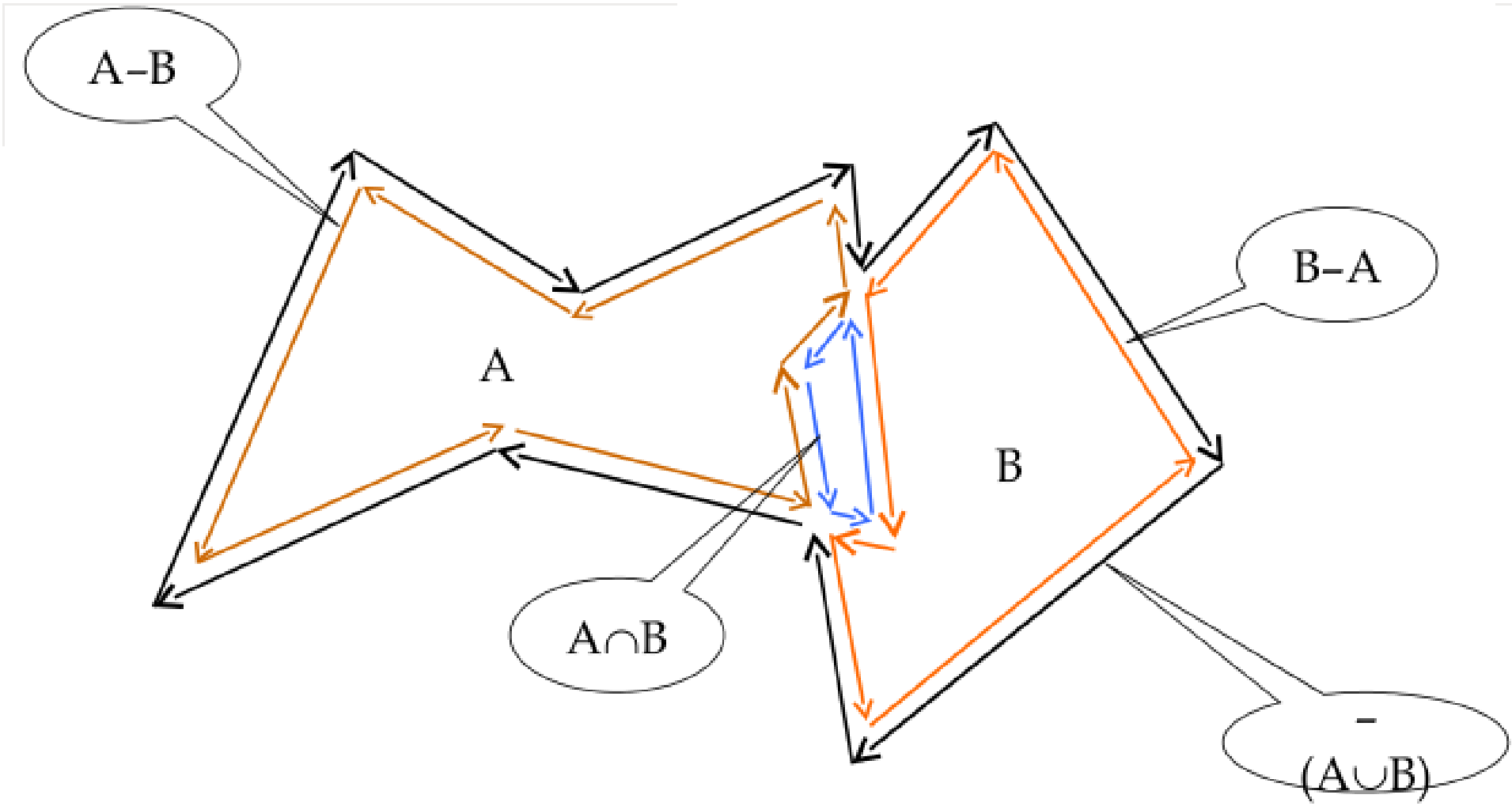
Algoritmo de Weiler-Atherton

Inserção de pontos de interseção nas listas de vértices de A e B



[© Esperança e Roma, 2011]

Classificação das novas regiões



[© Esperança e Roma, 2011]

Recorte em três dimensões

- Em OpenGL, polígonos são representados por um conjunto de triângulos

Recorte em três dimensões

- Em OpenGL, polígonos são representados por um conjunto de triângulos
- Recorte é realizado após projeção, mas antes da divisão perspectiva

Recorte em três dimensões

- Em OpenGL, polígonos são representados por um conjunto de triângulos
- Recorte é realizado após projeção, mas antes da divisão perspectiva
- Logo, precisamos saber recortar triângulos contra os seis planos do volume de visão canônico $[-1, 1]^3$

Recorte em três dimensões

- Em OpenGL, polígonos são representados por um conjunto de triângulos
- Recorte é realizado após projeção, mas antes da divisão perspectiva
- Logo, precisamos saber recortar triângulos contra os seis planos do volume de visão canônico $[-1, 1]^3$
- O resultado deverá ser um polígono, mas definido por sua triangulação

Recorte em três dimensões

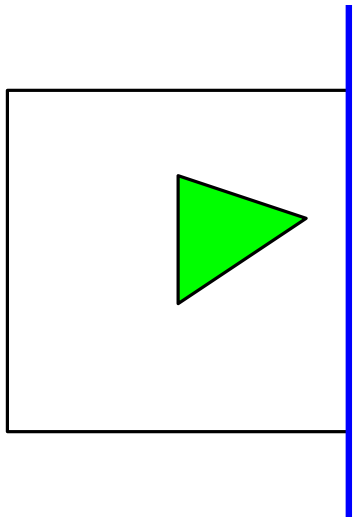
- Em OpenGL, polígonos são representados por um conjunto de triângulos
- Recorte é realizado após projeção, mas antes da divisão perspectiva
- Logo, precisamos saber recortar triângulos contra os seis planos do volume de visão canônico $[-1, 1]^3$
- O resultado deverá ser um polígono, mas definido por sua triangulação
- Os algoritmos de Sutherland-Hodgman e Liang-Barsky podem ser extendidos ao $\mathbb{R}^3$

Recorte de triângulos

Para cada um dos 6 planos do cubo canônico, temos 4 casos:

Recorte de triângulos

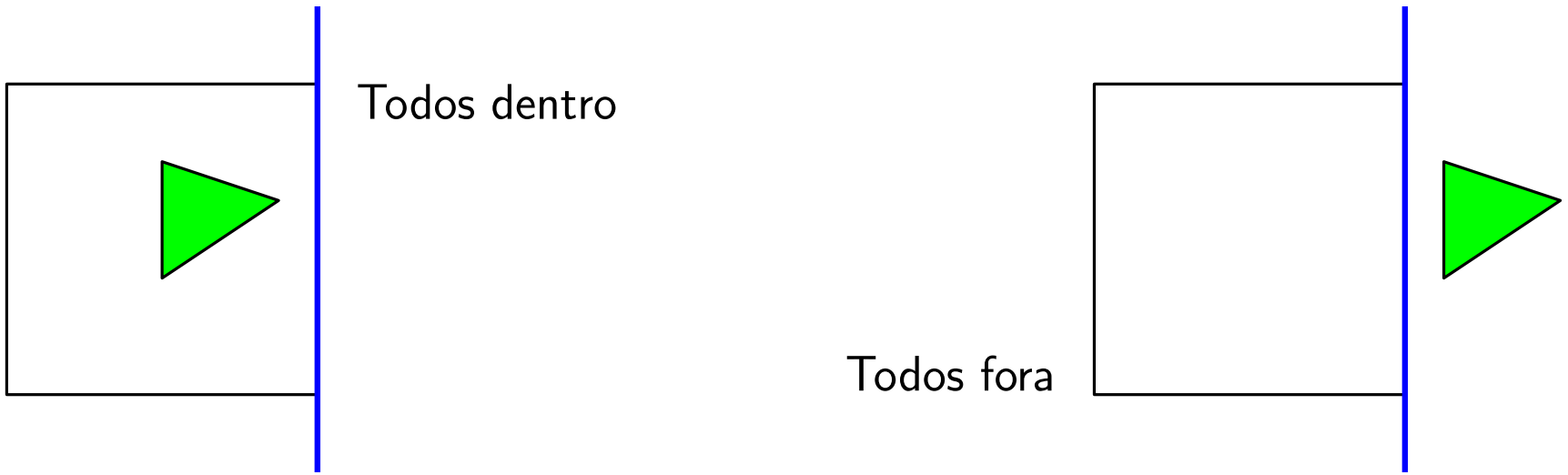
Para cada um dos 6 planos do cubo canônico, temos 4 casos:



Todos dentro

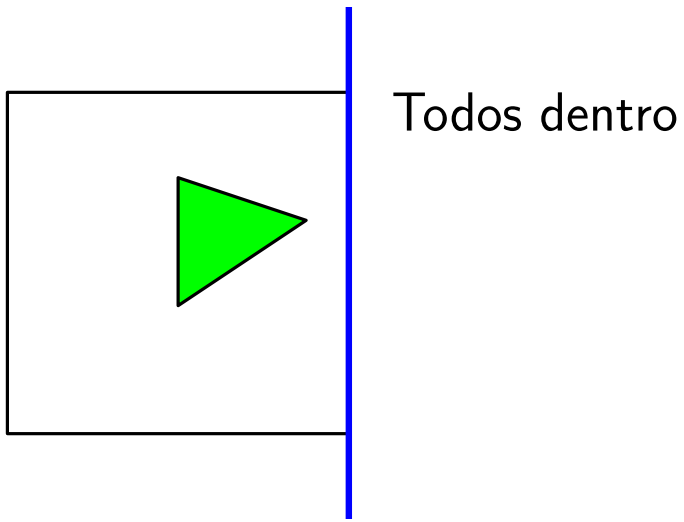
Recorte de triângulos

Para cada um dos 6 planos do cubo canônico, temos 4 casos:

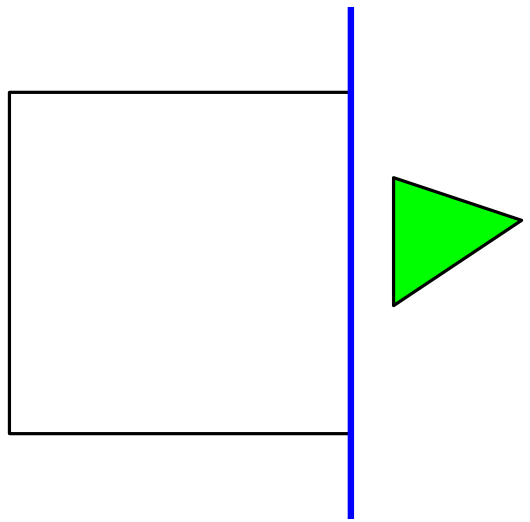


Recorte de triângulos

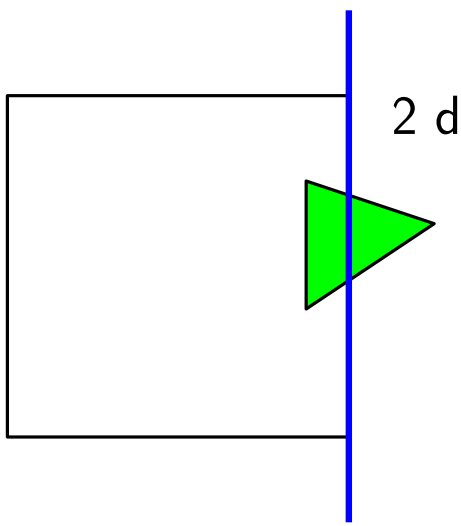
Para cada um dos 6 planos do cubo canônico, temos 4 casos:



Todos dentro



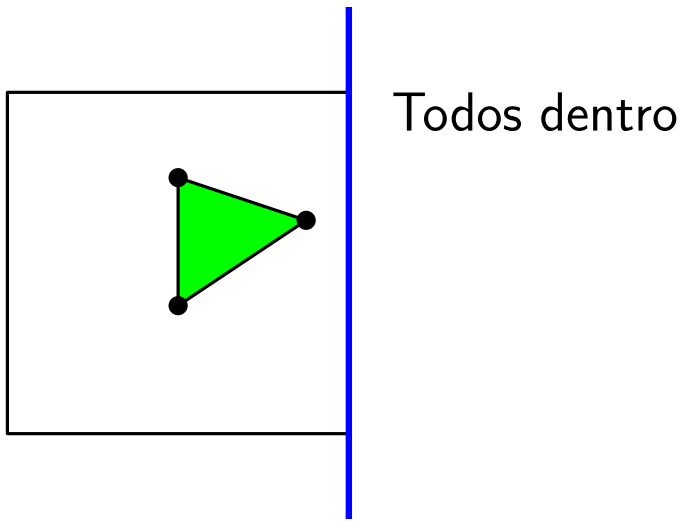
Todos fora



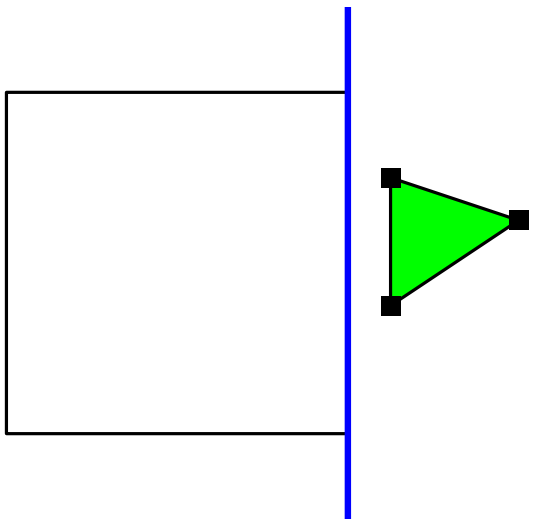
2 dentro

Recorte de triângulos

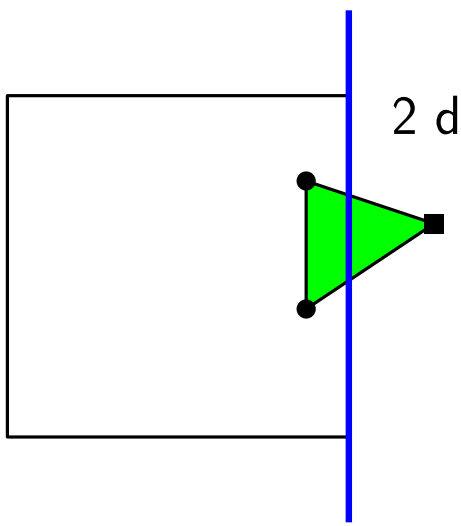
Para cada um dos 6 planos do cubo canônico, temos 4 casos:



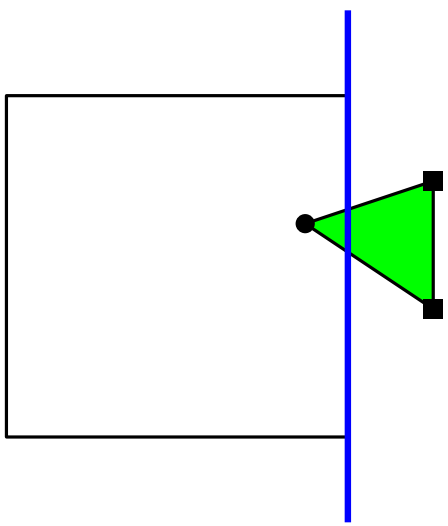
Todos dentro



Todos fora



2 dentro



2 fora

Recorte de triângulos

Para cada um dos 6 planos do cubo canônico, temos 4 casos:

Observação.

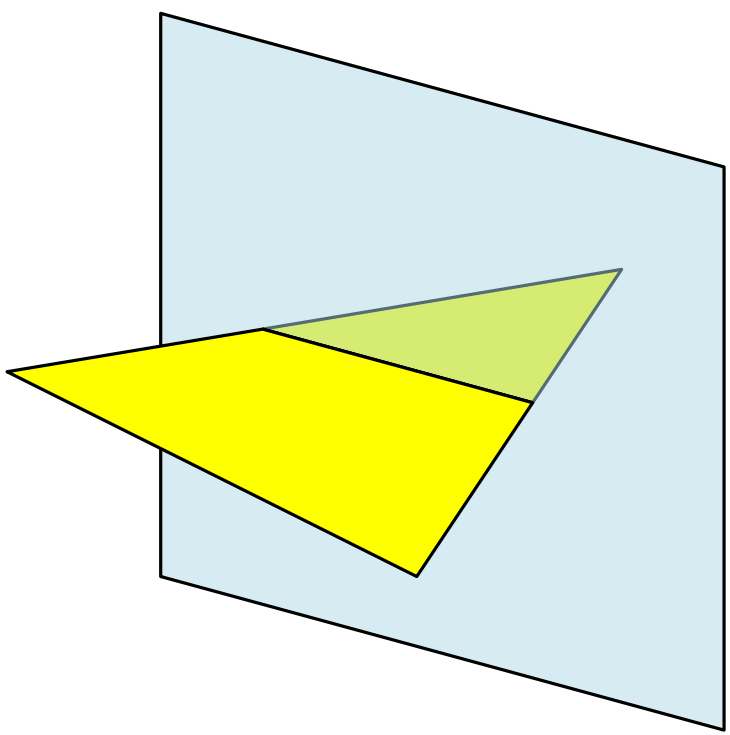
A interseção de um plano com um triângulo sempre resulta em um quadrilátero $\Longrightarrow$ dois triângulos

Recorte de triângulos

Para cada um dos 6 planos do cubo canônico, temos 4 casos:

Observação.

A interseção de um plano com um triângulo sempre resulta em um quadrilátero $\Longrightarrow$ dois triângulos



Recorte de triângulos

Para cada um dos 6 planos do cubo canônico, temos 4 casos:

Observação.

A interseção de um plano com um triângulo sempre resulta em um quadrilátero $\Longrightarrow$ dois triângulos