Computação Gráfica:

Capítulo 11:

Raytracing/Raycasting/Pixel Shading
Algoritmos-Chave

(rev. 2020)

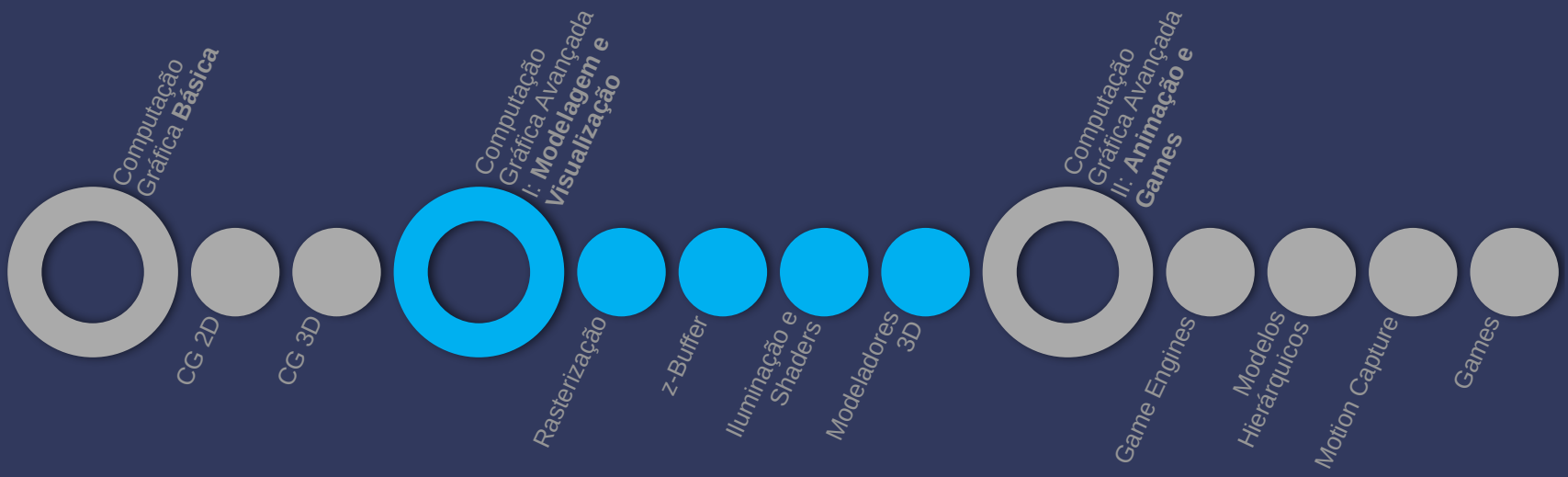
Prof. Dr. rer.nat. Aldo von Wangenheim

Departamento de Informática e Estatística - INE/CTC/UFSC

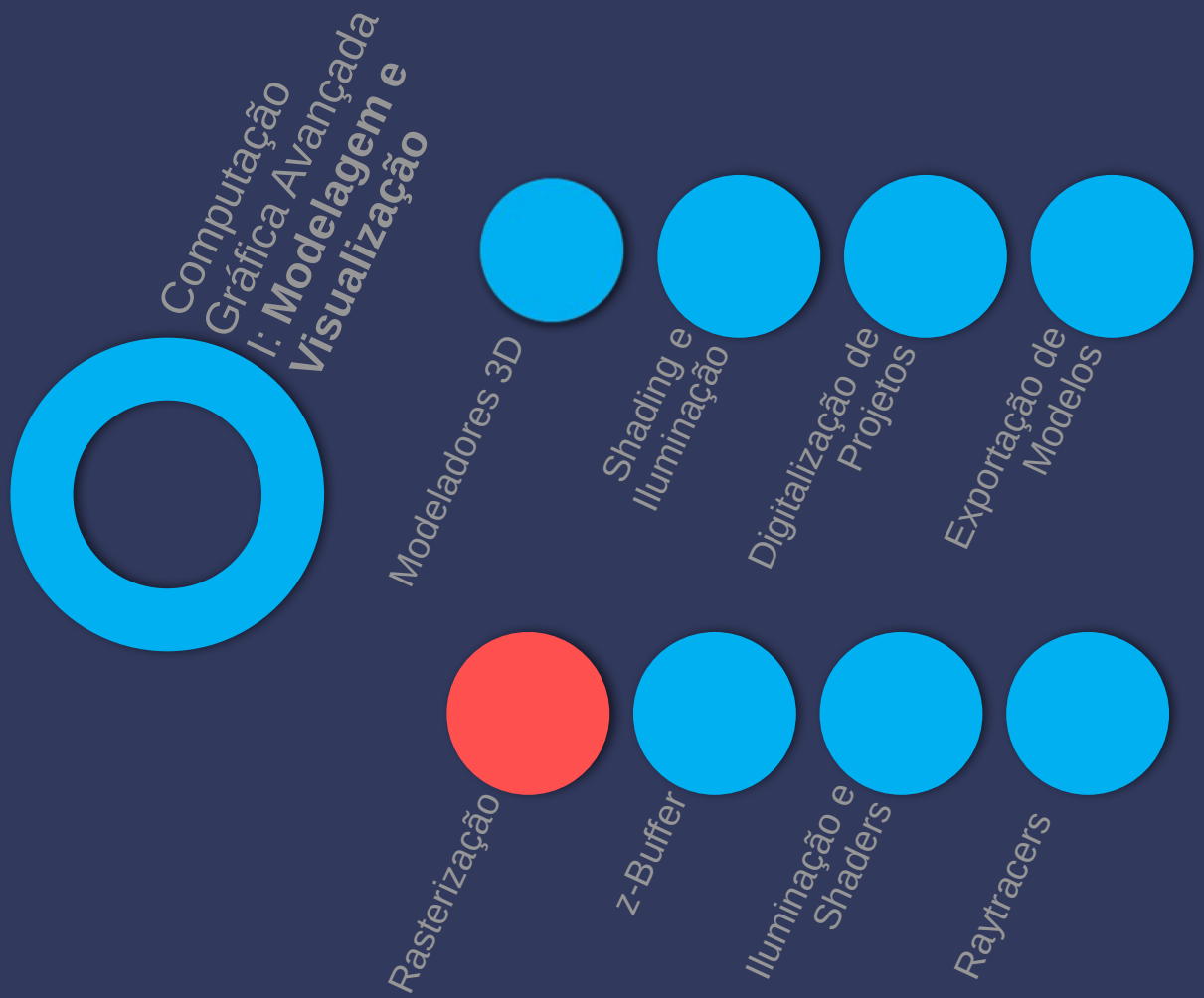
Conteúdo desta Aula

- O que é Raytracing ?
  - O que são Raytracers ?
  - Modelo de Funcionamento de Alto Nível de um Raytracer
- Principais Raytracers
- Conversão por Varredura (Scan Conversion)
- zBuffering
- Modelando a Iluminação de um Objeto
- Exercícios com Raytracing

Timeline da Disciplina



Estratégia dos Módulos Avançados



Conversão por Varredura ou Scan Conversion

ine5341 - Computação Gráfica

Capítulo 11

Parte 1

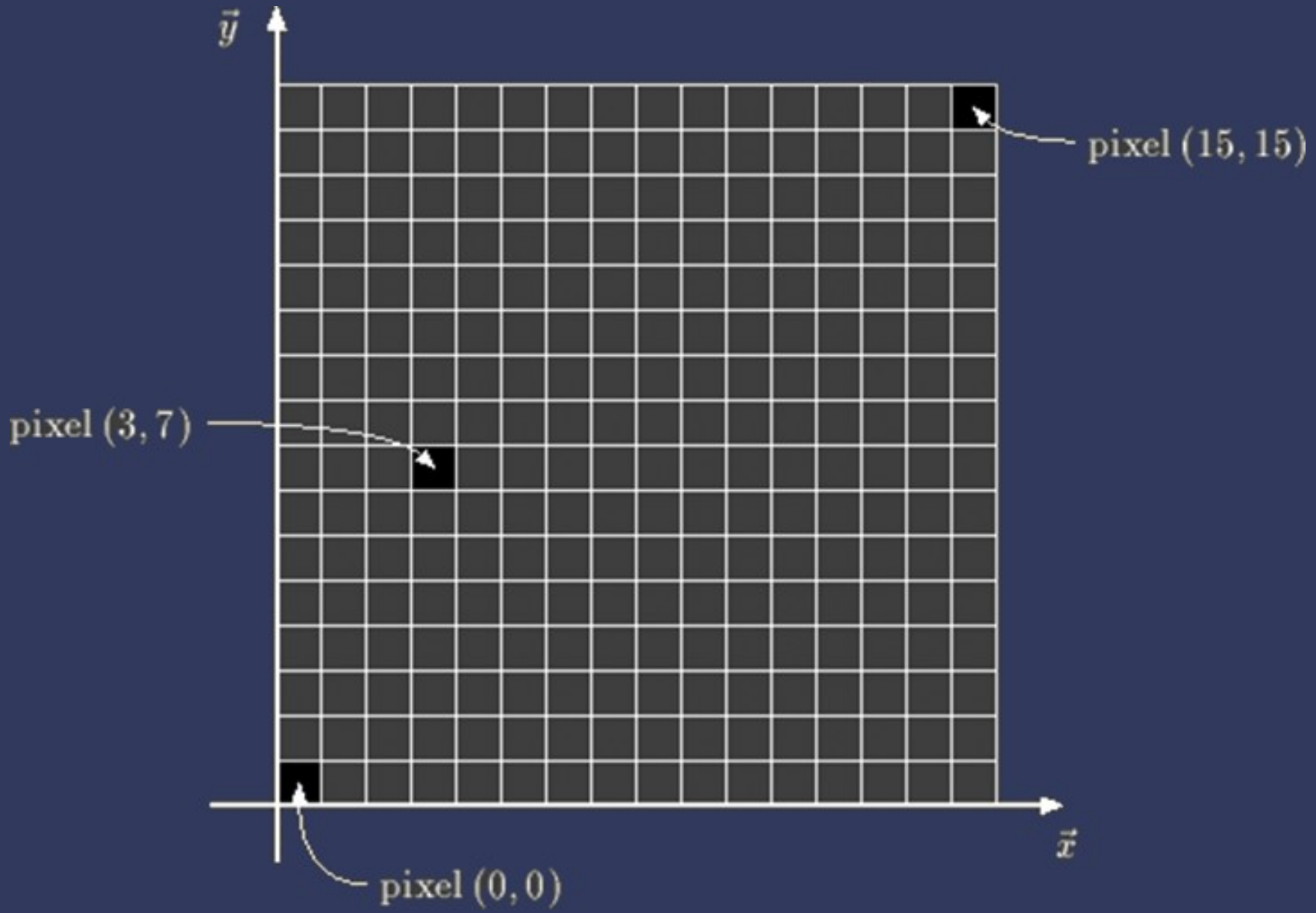
Passos para implementação do Raytracing no SGI:

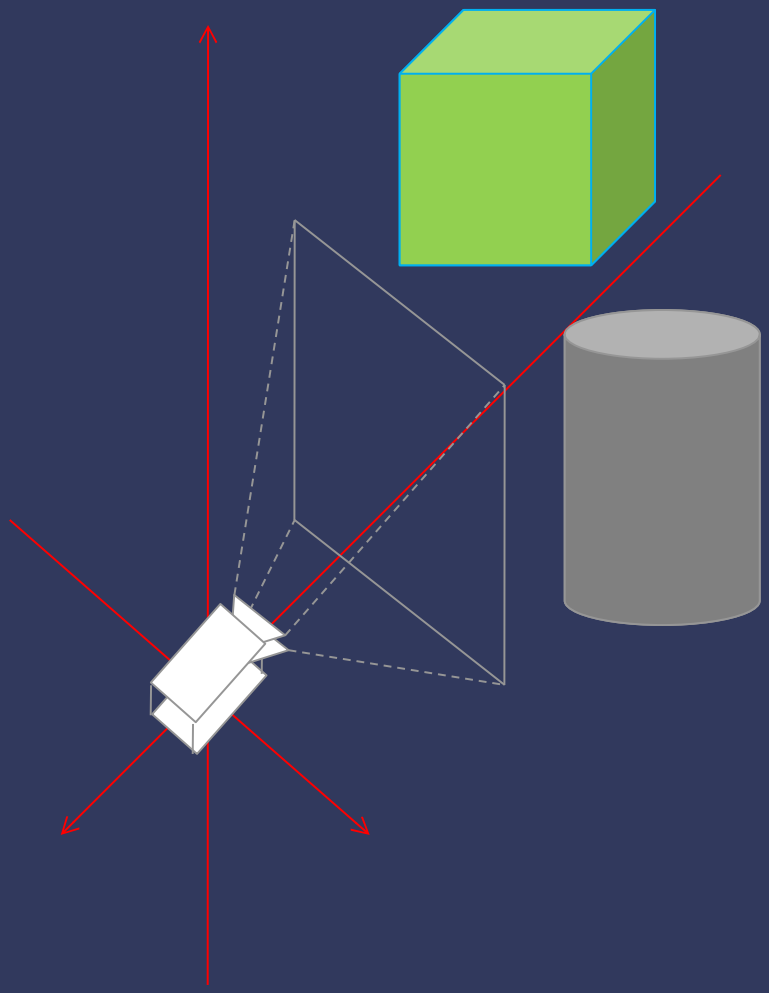
1. Criação da Window com uma orientação arbitrária
2. Transformação dos objetos do Mundo para SCN
3. Clipping *
4. Conversão por Varredura (*Rasterização*)
5. Z-Buffering, complementarmente à Conversão por Varredura (*Oclusão*)
6. Raytracing por Pixel (*Pixel Shading*)
7. Cópia do Z-Buffer para a Viewport

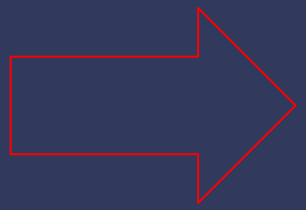
Conversão por Varredura

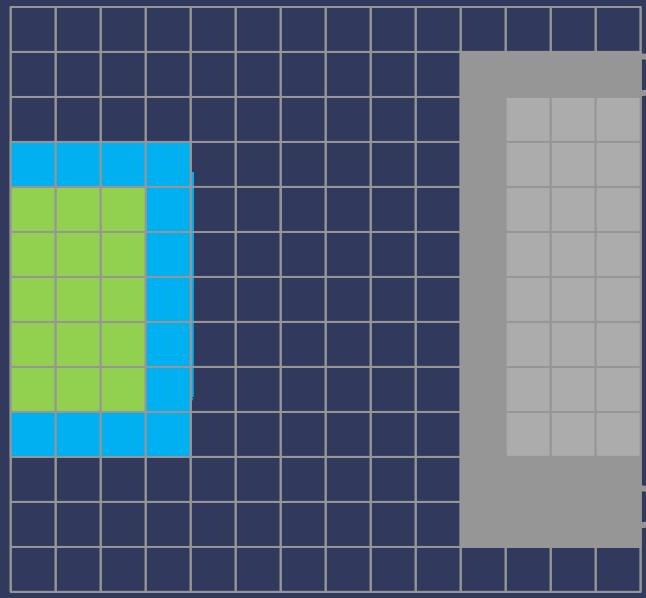
- Uma operação fundamental extensivamente utilizada para visualização em CG é o processo de conversão por varredura ou rasterização (*scan conversion* ou *rasterization*).
- Dado um polígono em Coordenadas do Plano de Projeção (espaço de imagem), este processo determina os pixels que interceptam este polígono em Coordenadas de Viewport.
- Em outras palavras, a Conversão por Varredura (juntamente com o *Z-Buffering*) determina qual objeto do mundo preencherá cada pixel da cena 2D final, vista pelo usuário.

Coordenadas de Dispositivo



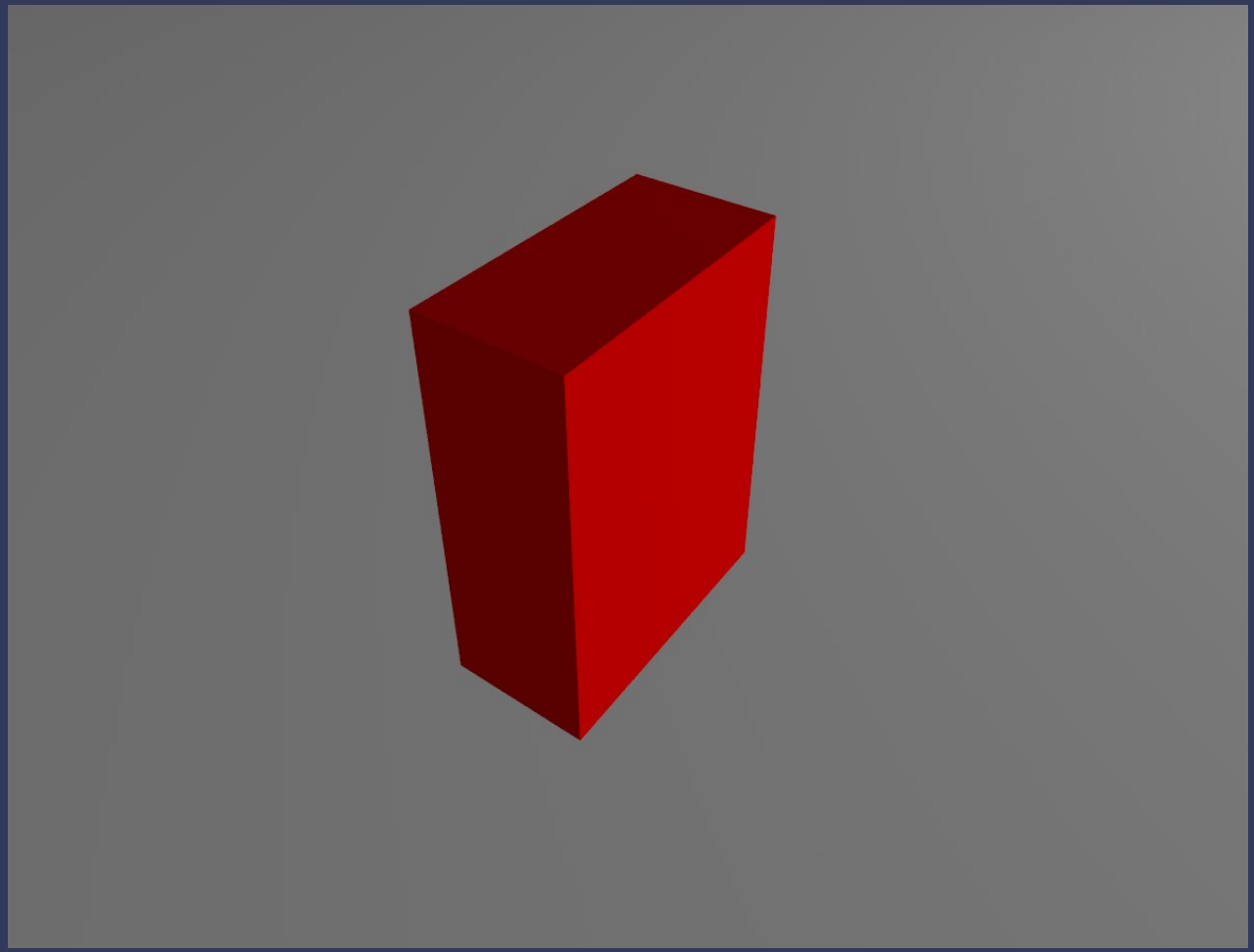




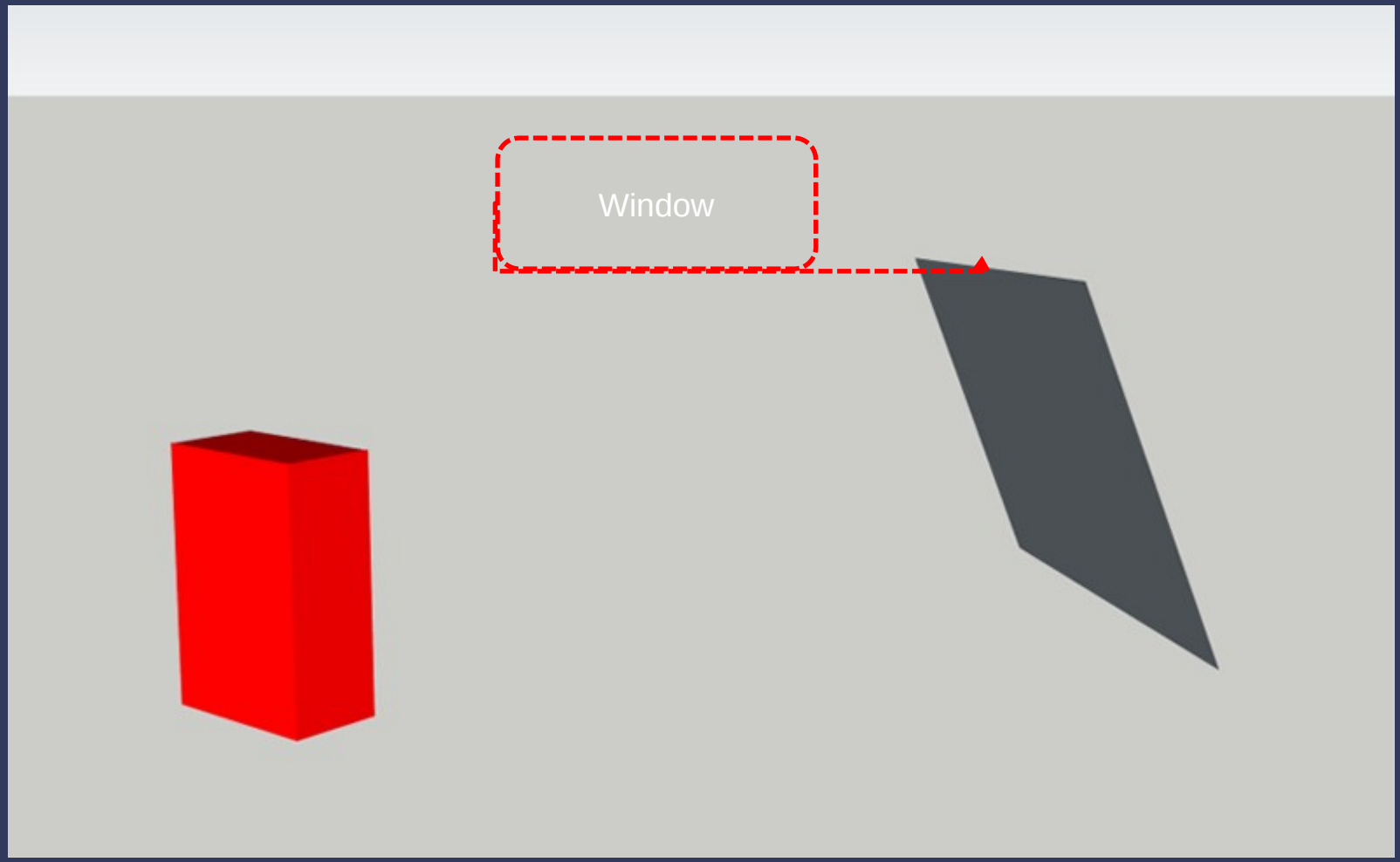


UM EXEMPLO...

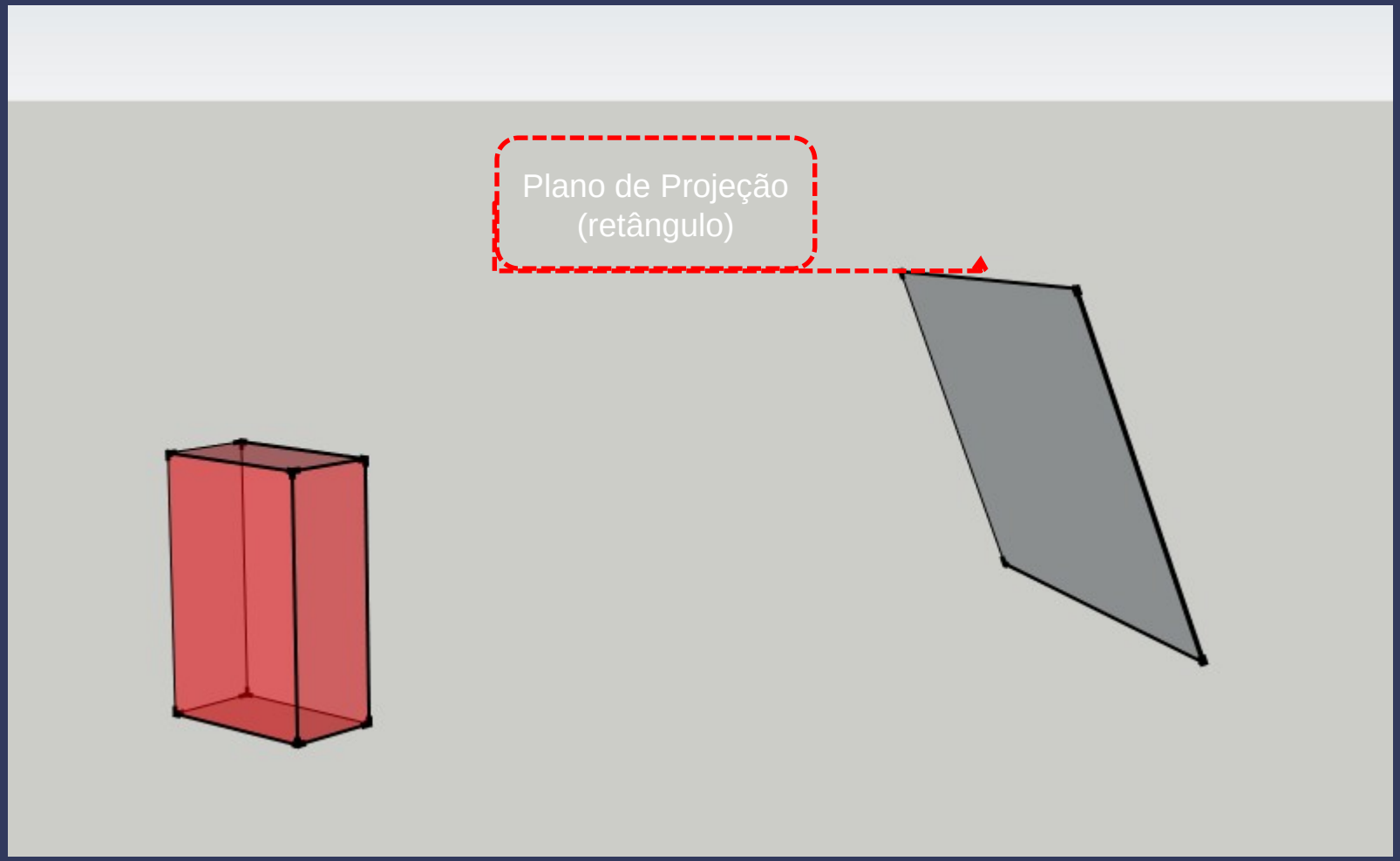
O que eu quero ver...



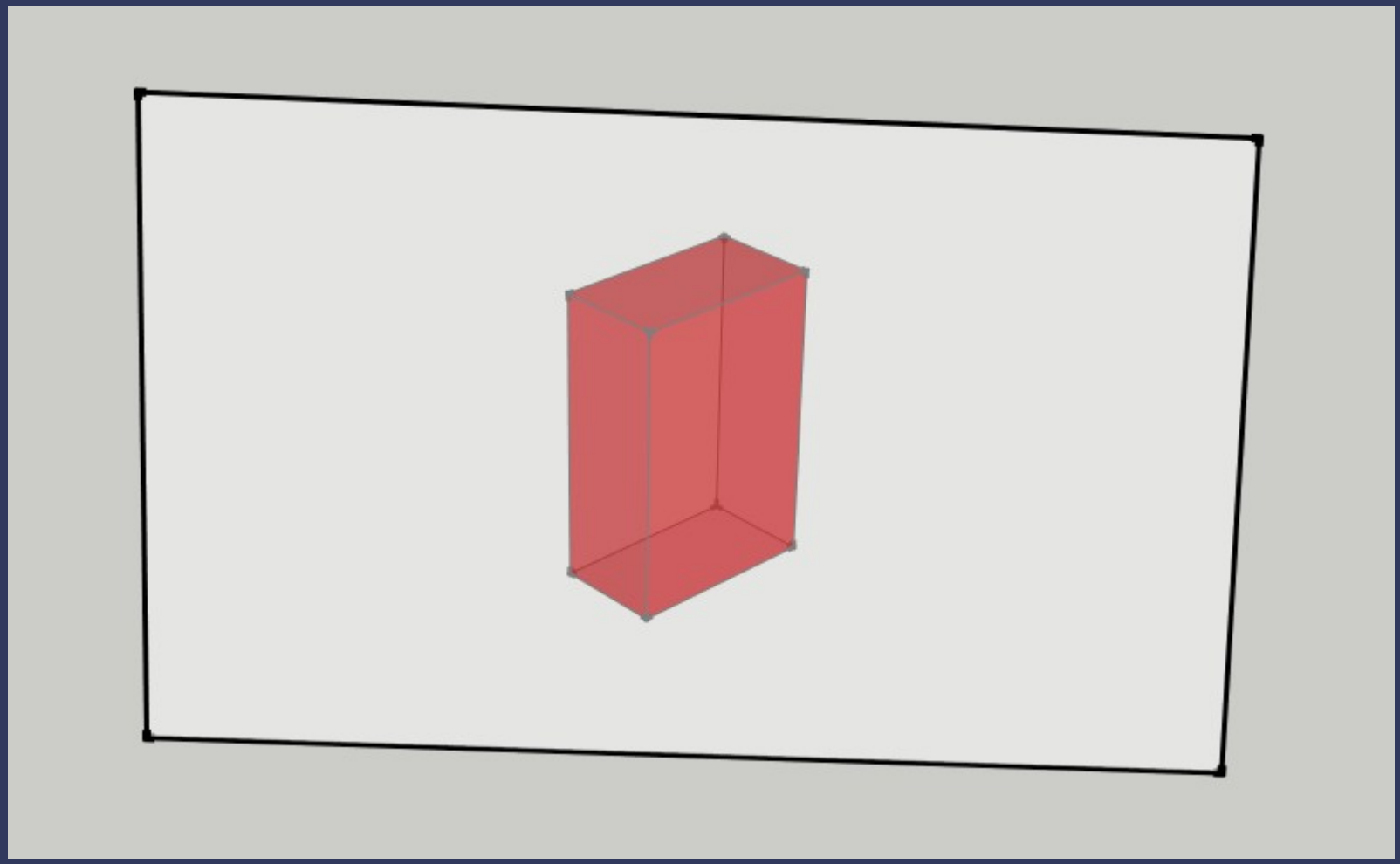
O que eu tenho...



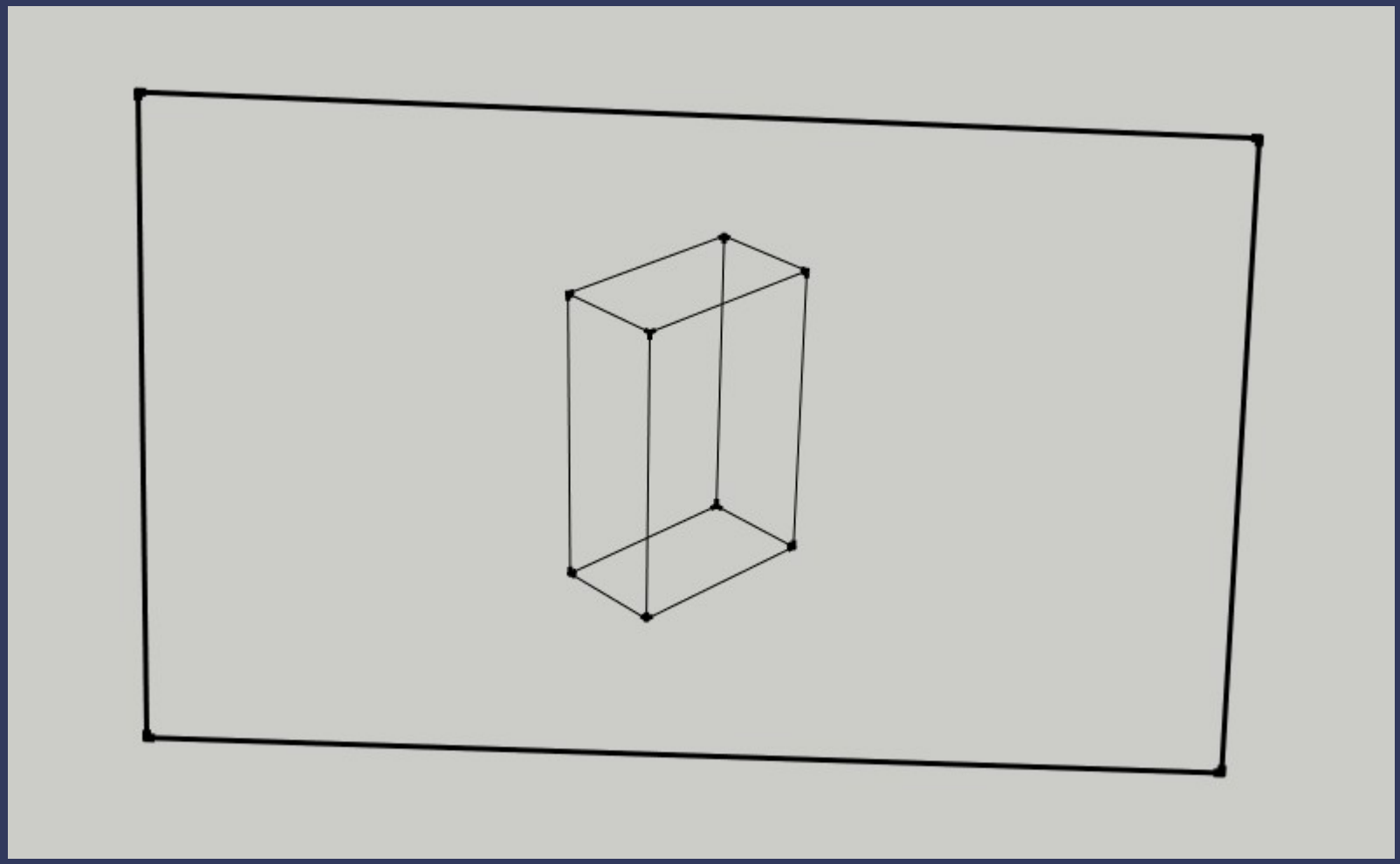
..com essa geometria...

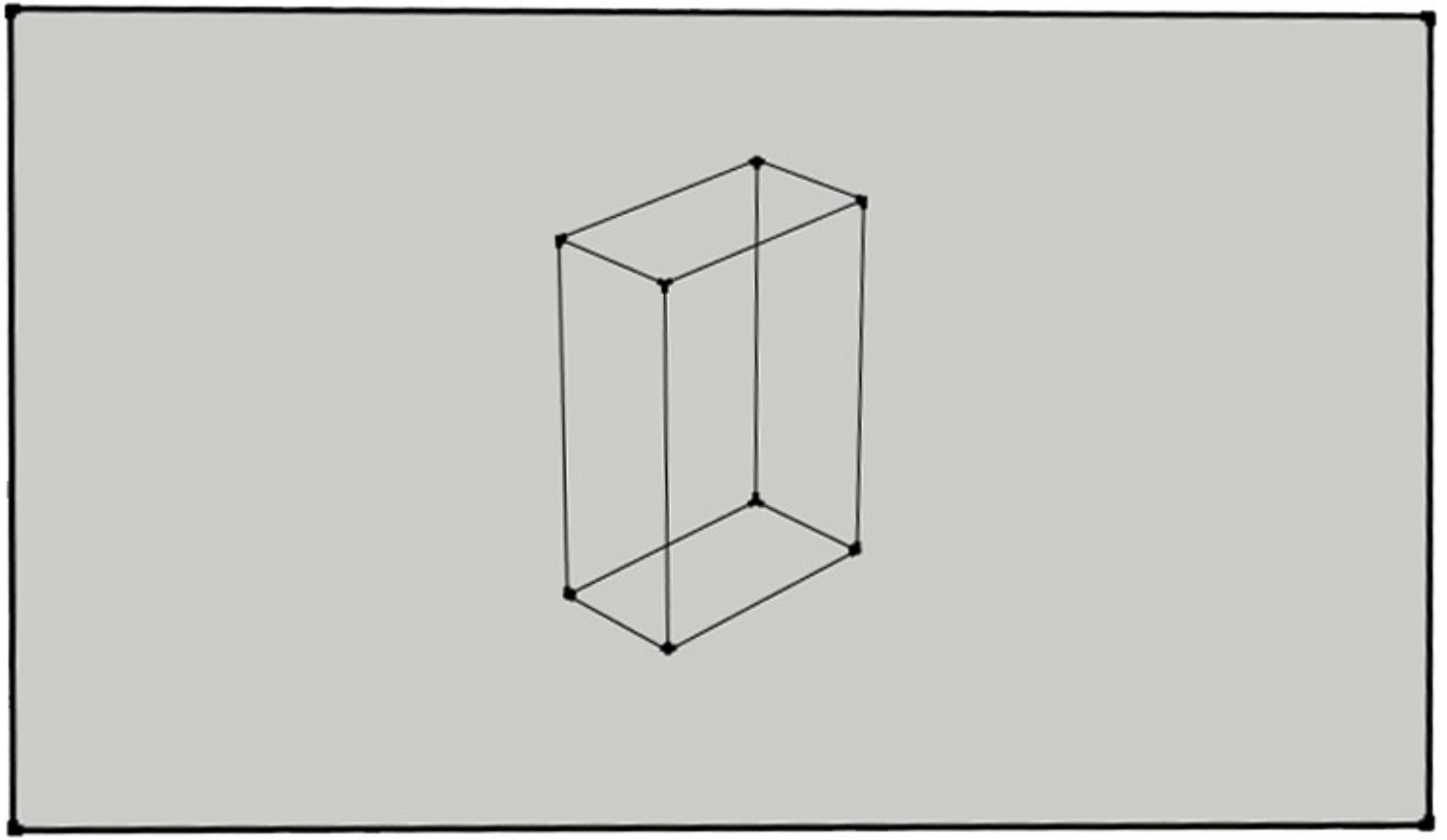


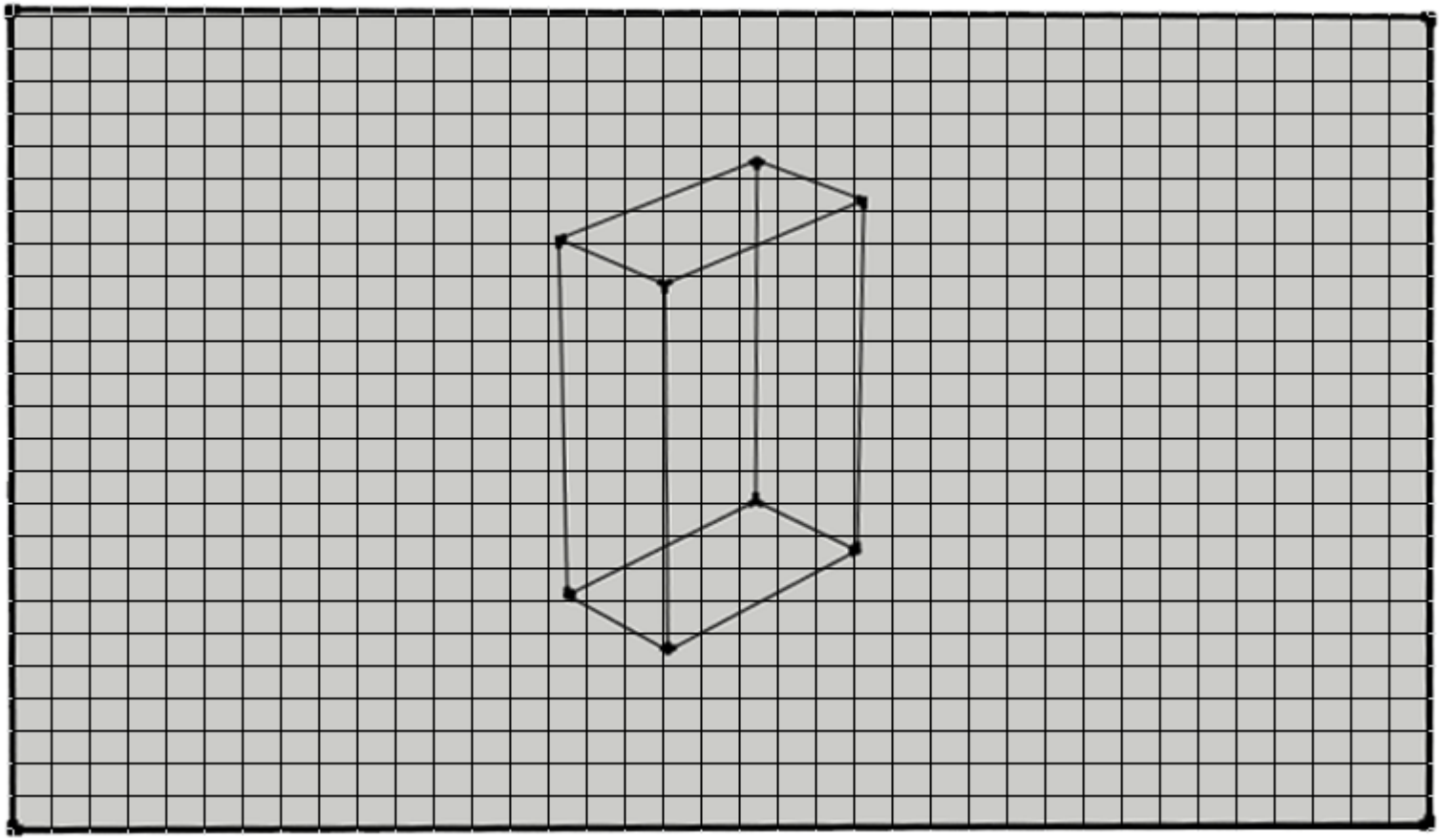
..e essa projeção...

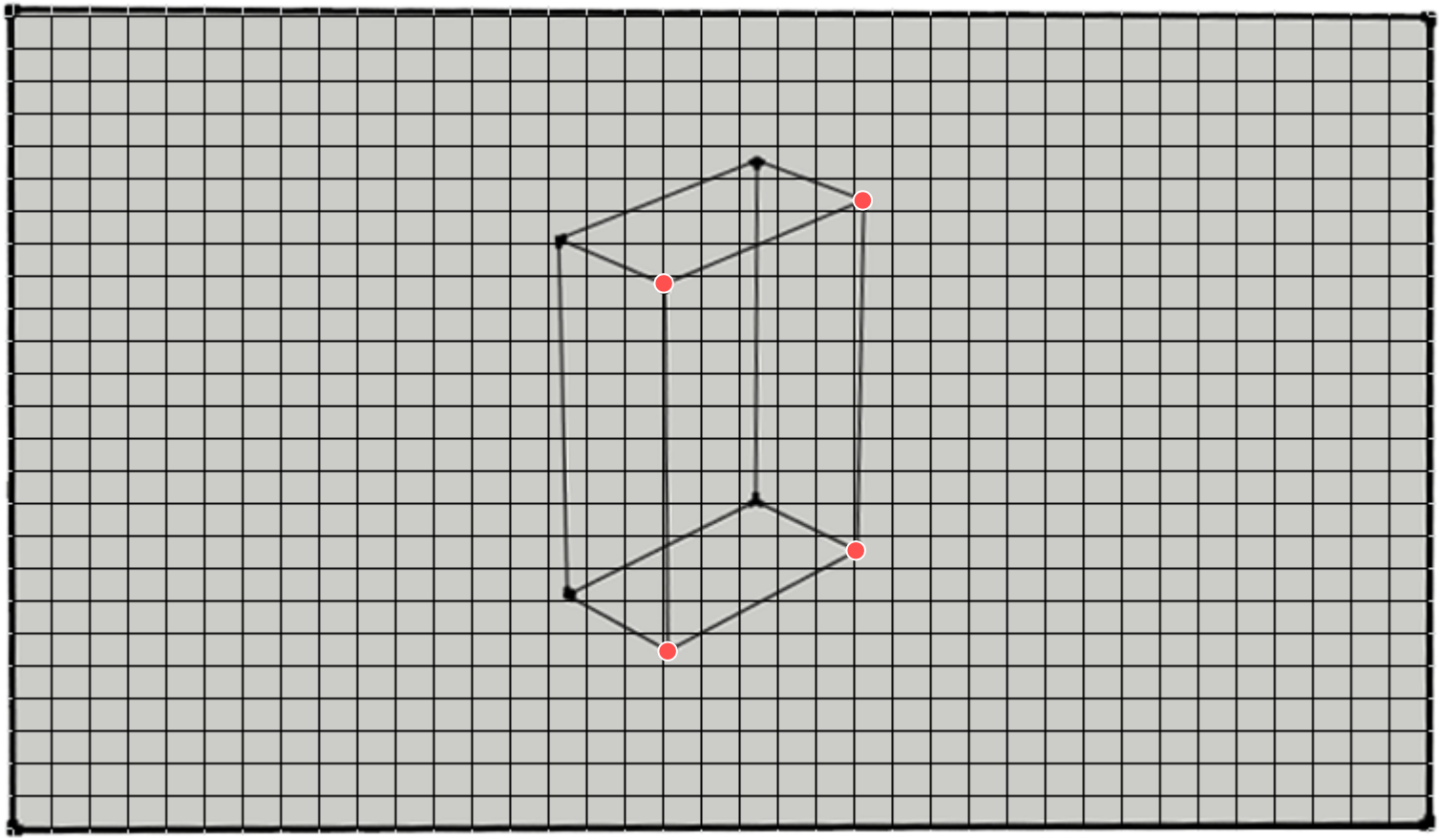


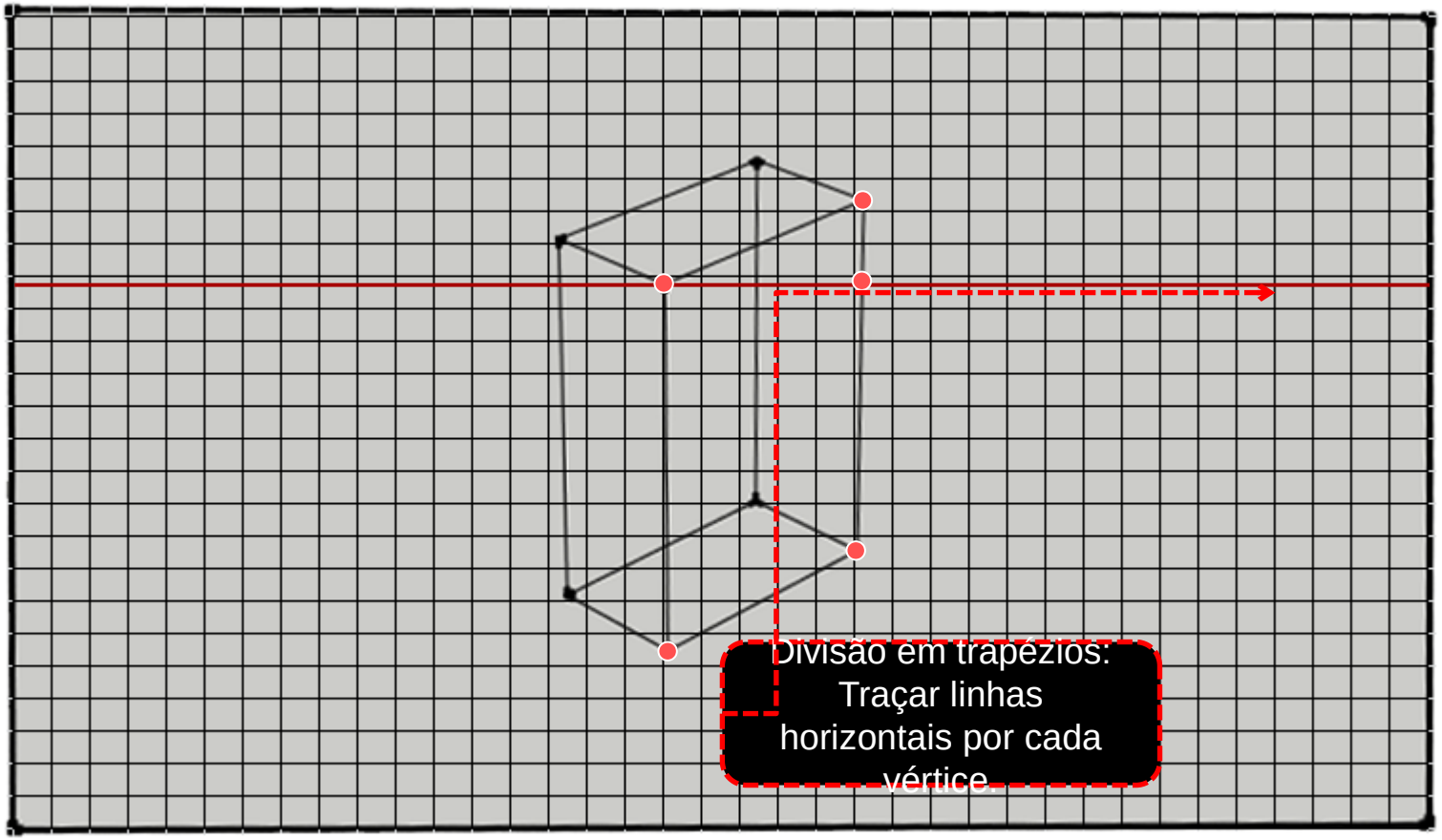
..vista como modelo de arame...

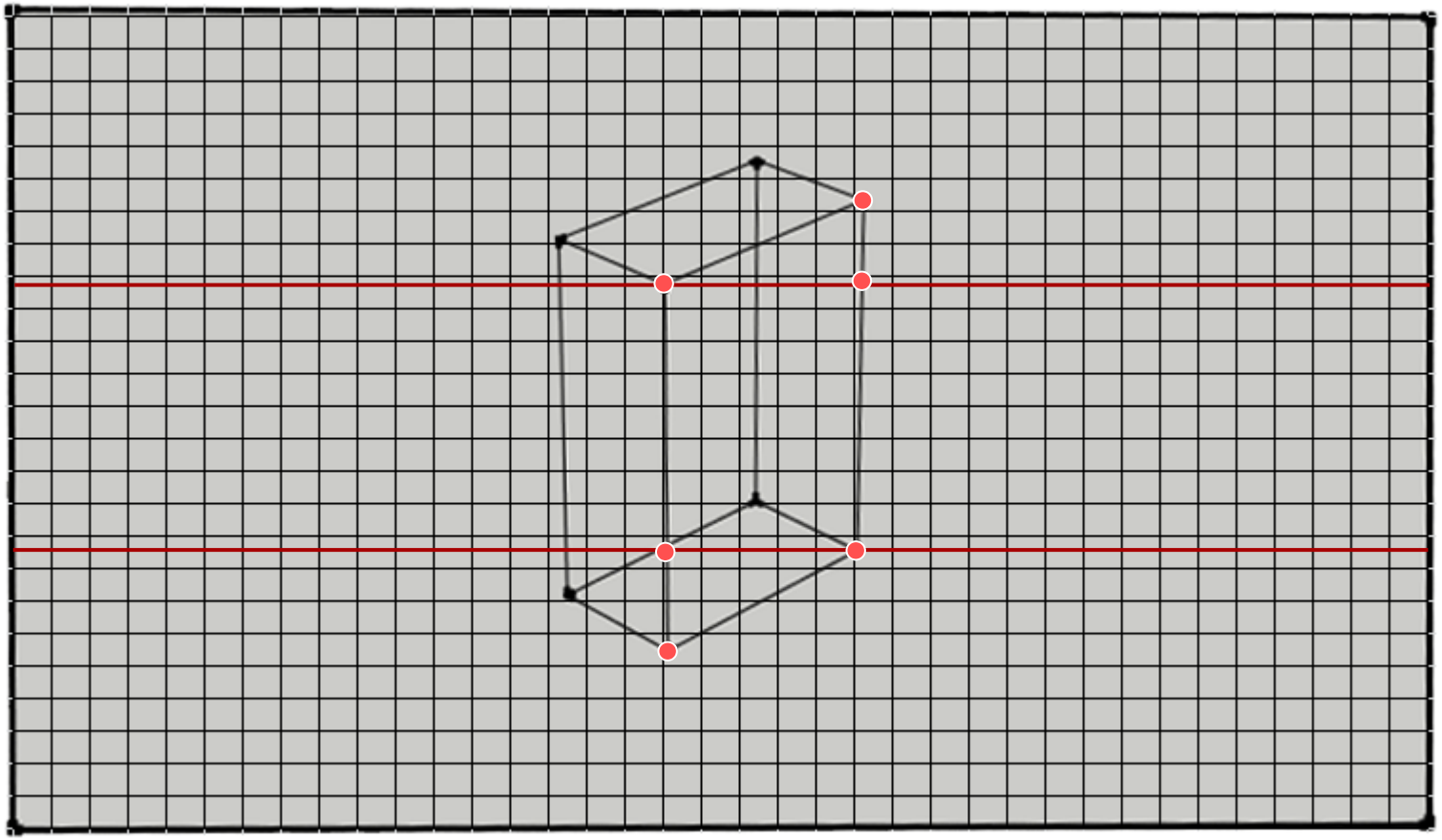


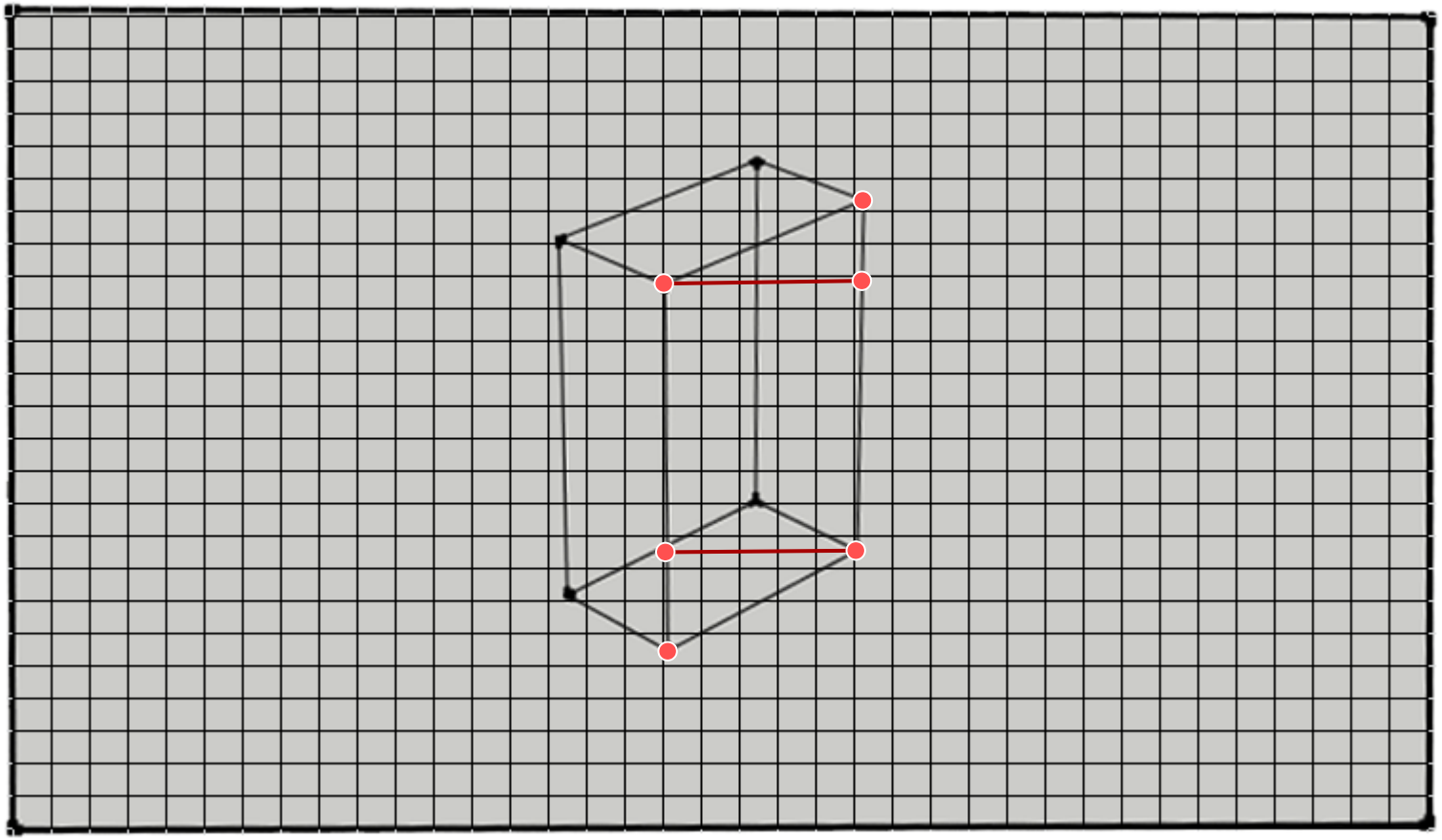


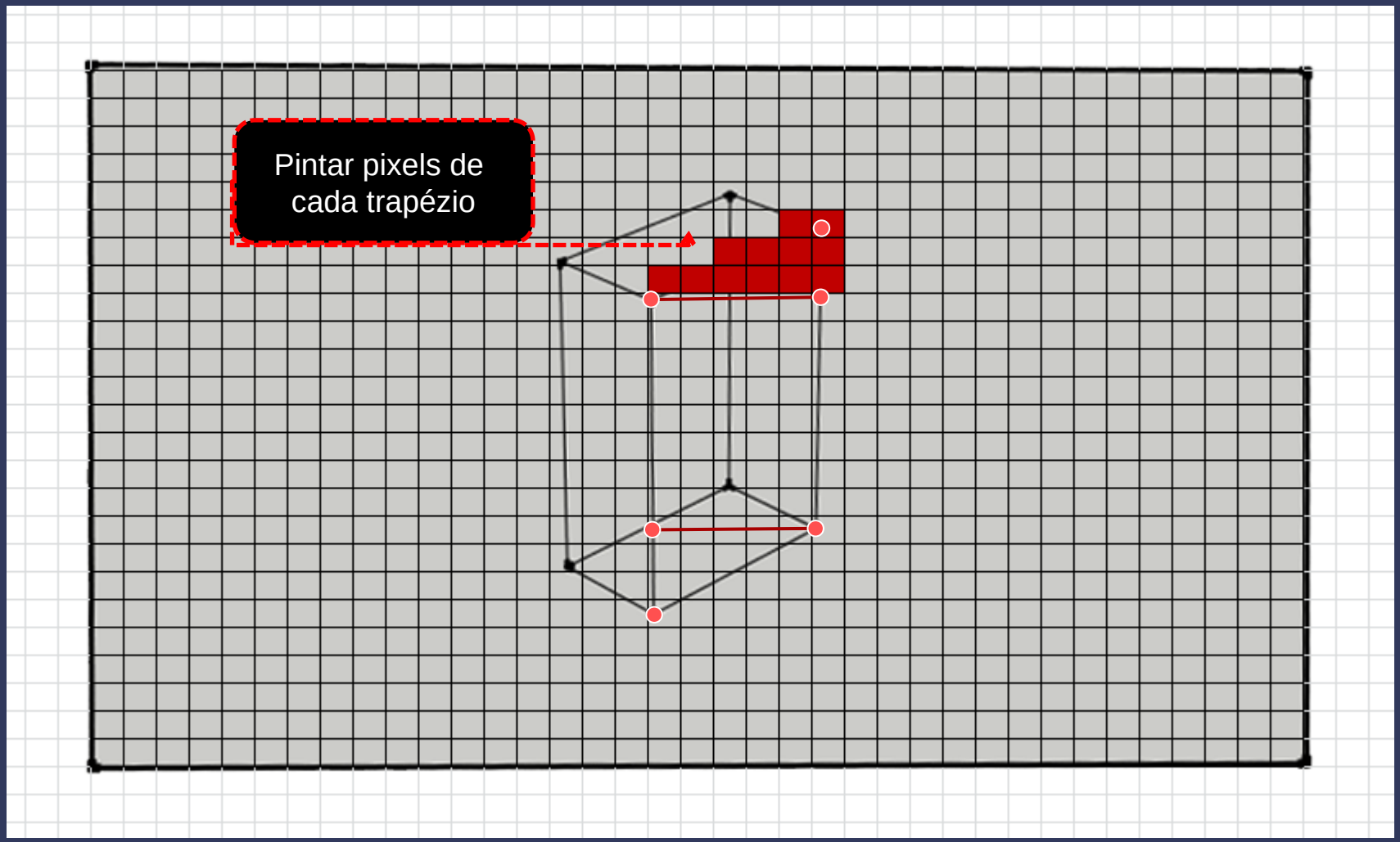


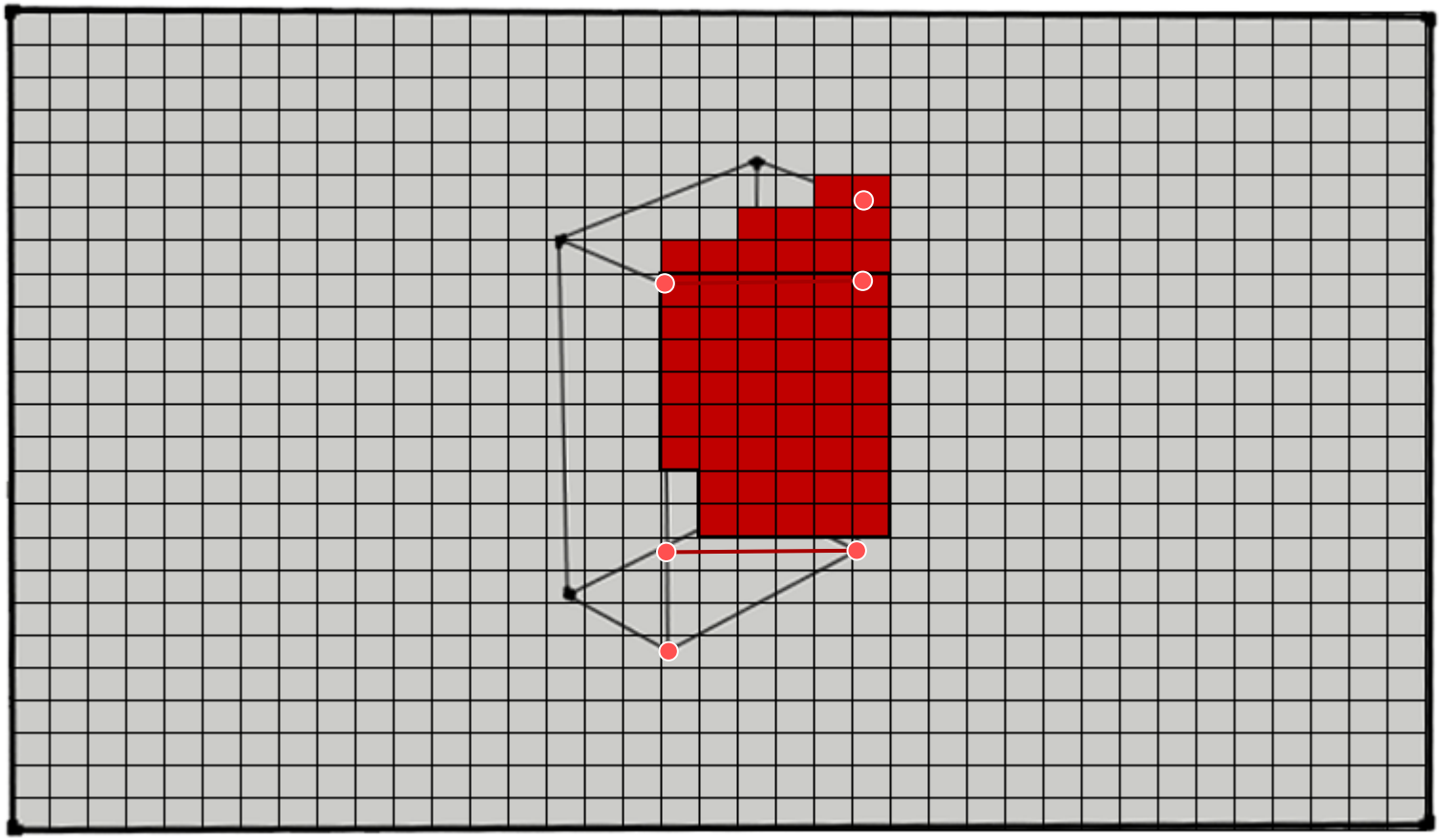


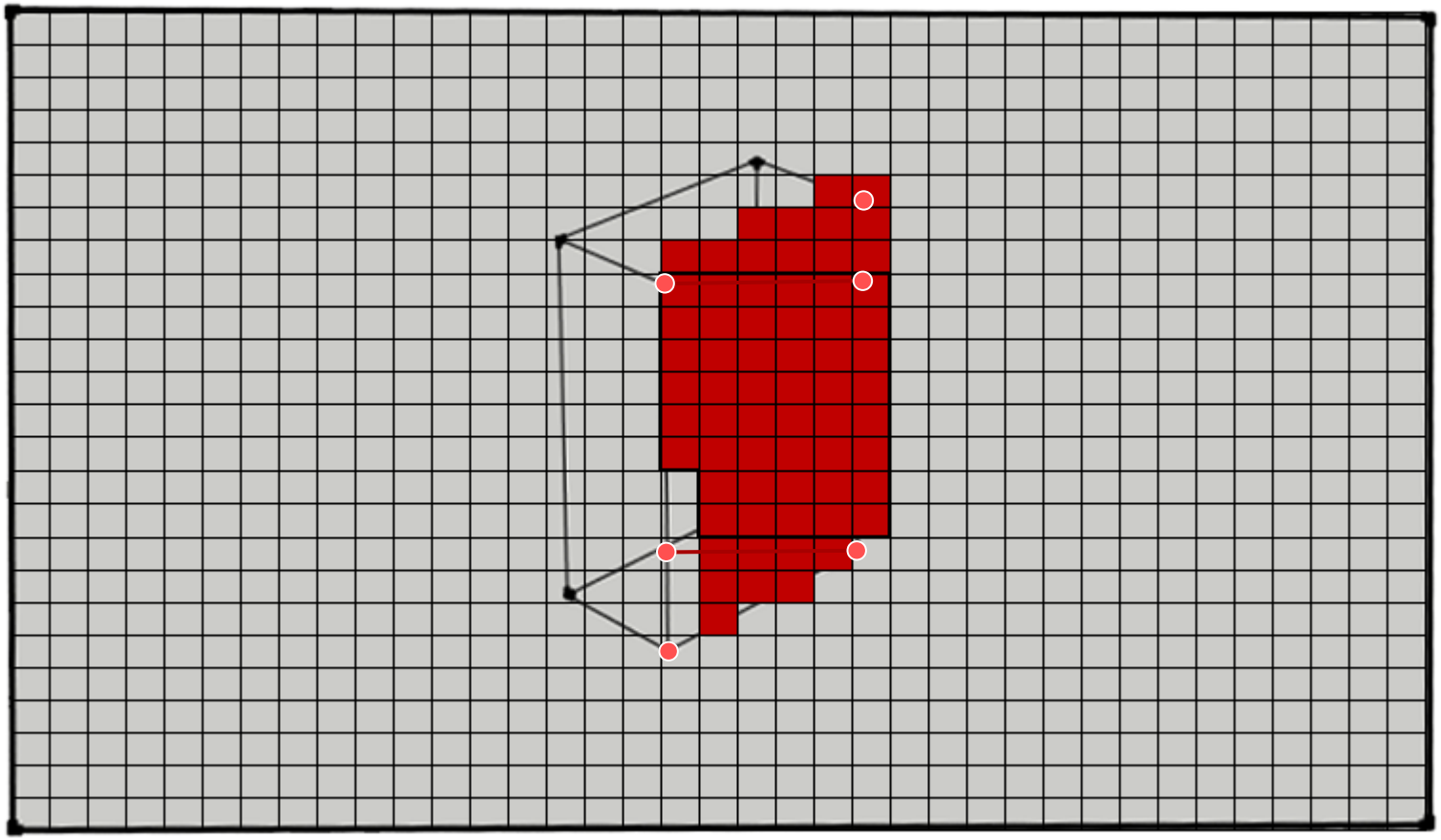

Divisão em trapézios:
Traçar linhas
horizontais por cada
vértice.

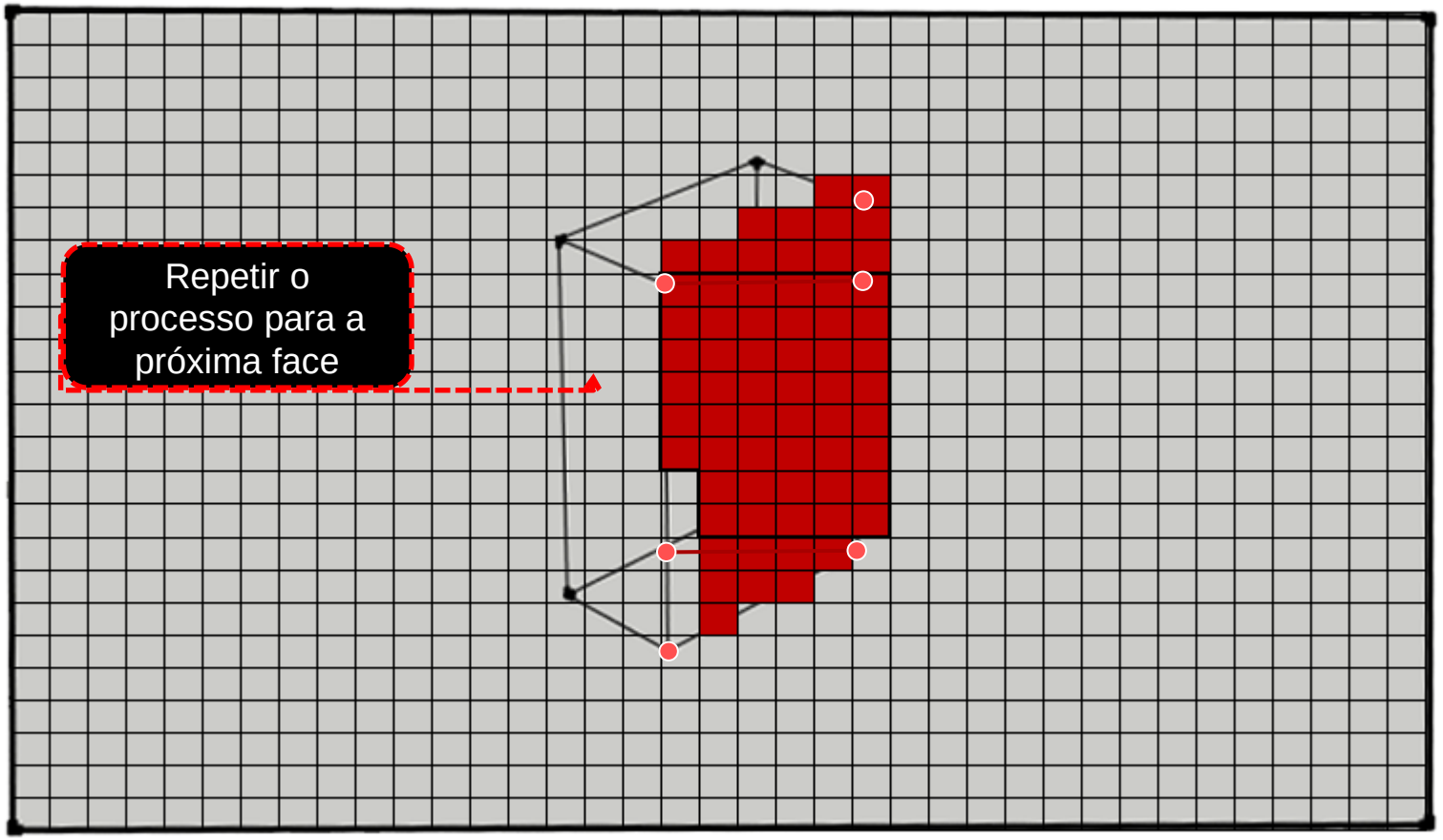





Pintar pixels de
cada trapézio

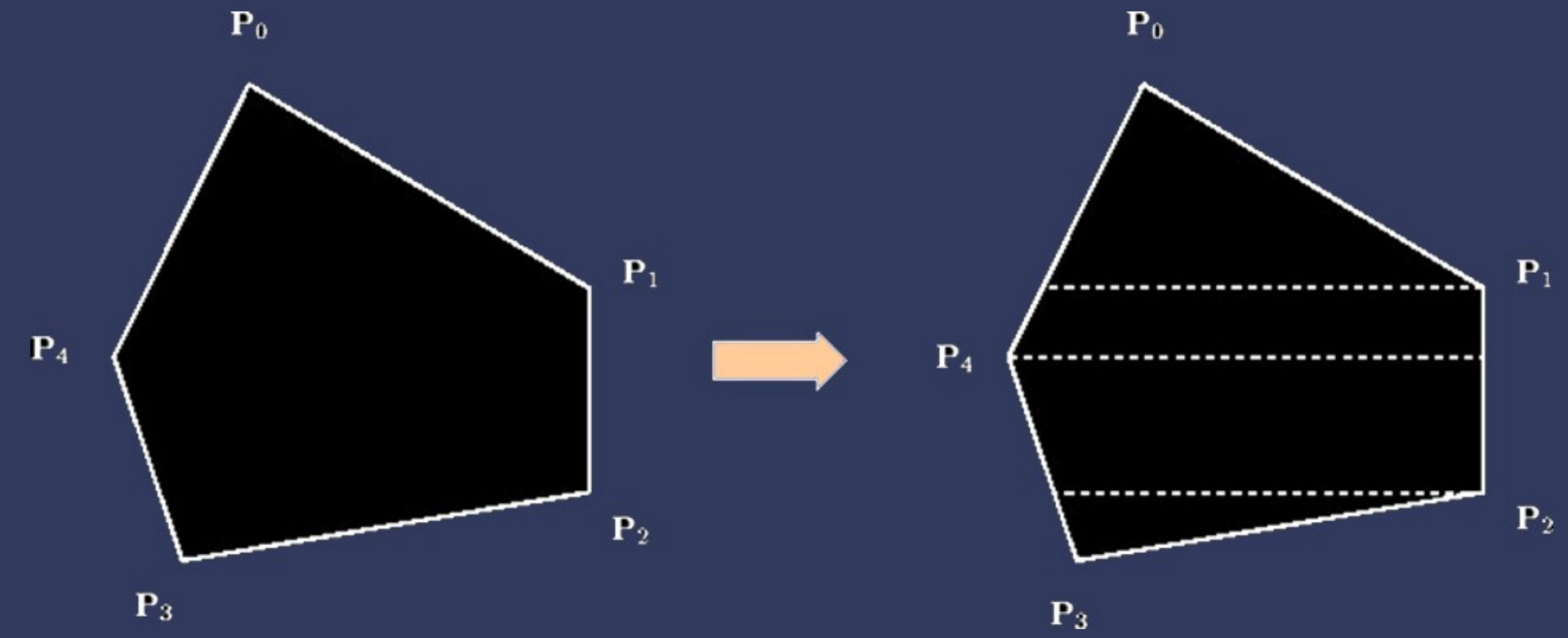





Repetir o processo para a próxima face

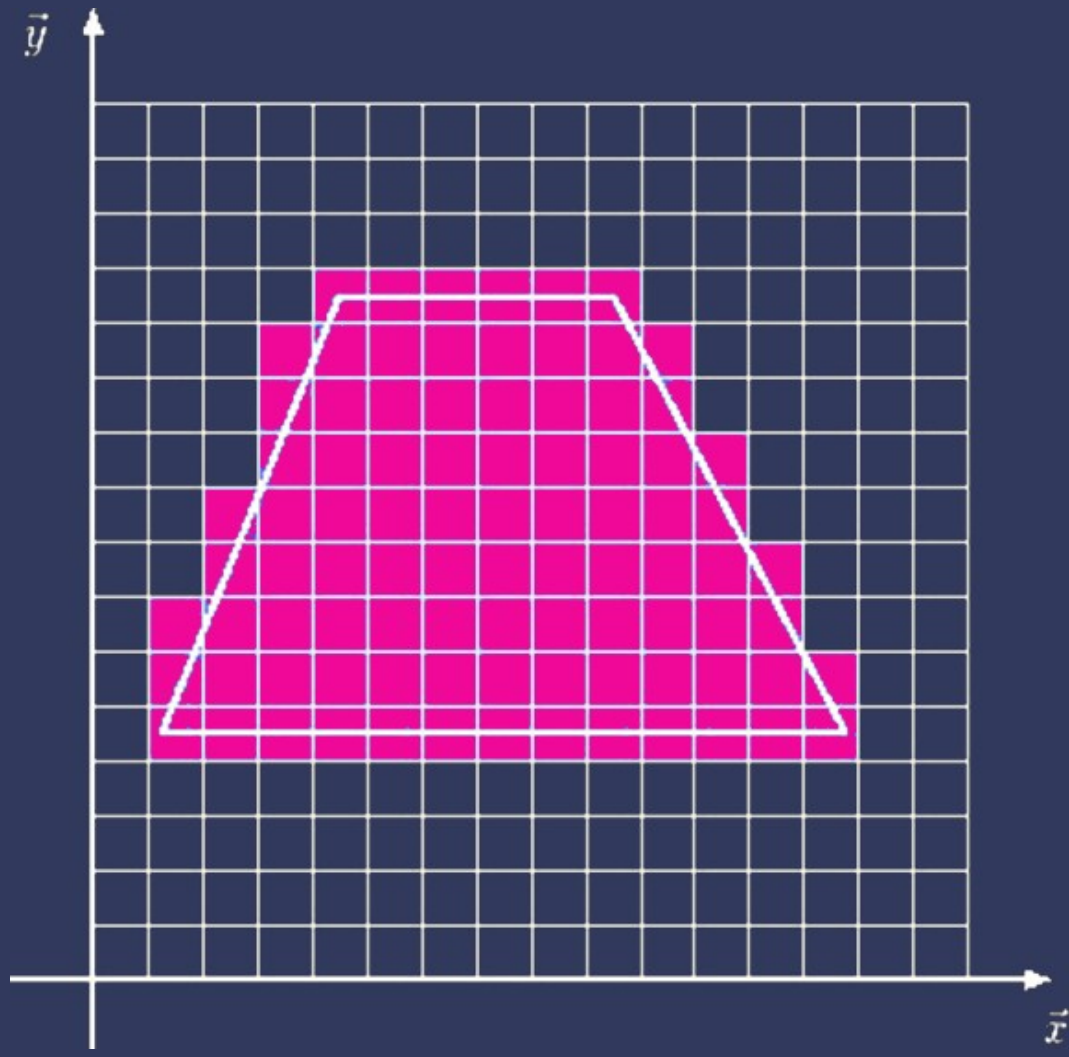
Conversão por Varredura

- Para a conversão por varredura de um polígono vamos dividir o polígono em uma série de trapézios e então converter por varredura cada um destes trapézios.
  - Cada trapézio será gerado de forma que as bordas superior e inferior do trapézio sejam paralelas às linhas de varredura, possuindo “y” constante nesta região.
  - Também levamos em conta trapézios degenerados sob a forma de triângulos, que possuem o topo ou a base com comprimento zero.
  - Na figura, os trapézios superior e inferior do desenho são de fato triângulos.


P0
P1
P4
P2
P3
P0
P1
P4
P2
P3

Conversão por Varredura

- A união de todos os pixels que interceptam o conjunto de trapézios será o conjunto de pixels que intercepta o polígono.

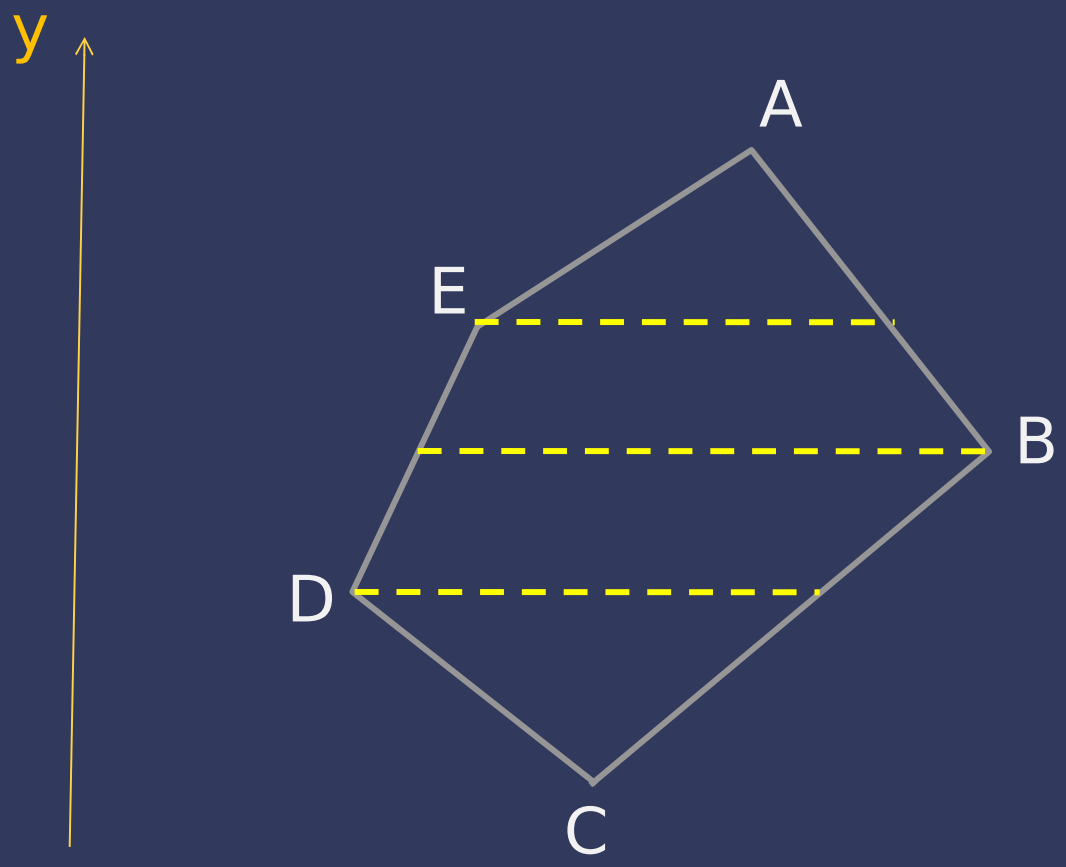


Gerando os Trapézios

- Quebrar um polígono em trapézios é simples. Para isto, primeiramente obtemos os vértices extremos do polígono no eixo y, isto é, os vértices que possuem o maior e o menor valor em y.
  - Todos os outros vértices possuem y dentro do intervalo criado por estes vértices extremos.
- Ordenamos então os vértices restantes em ordem decrescente de y, e para cada vértice deste traçamos uma linha horizontal.

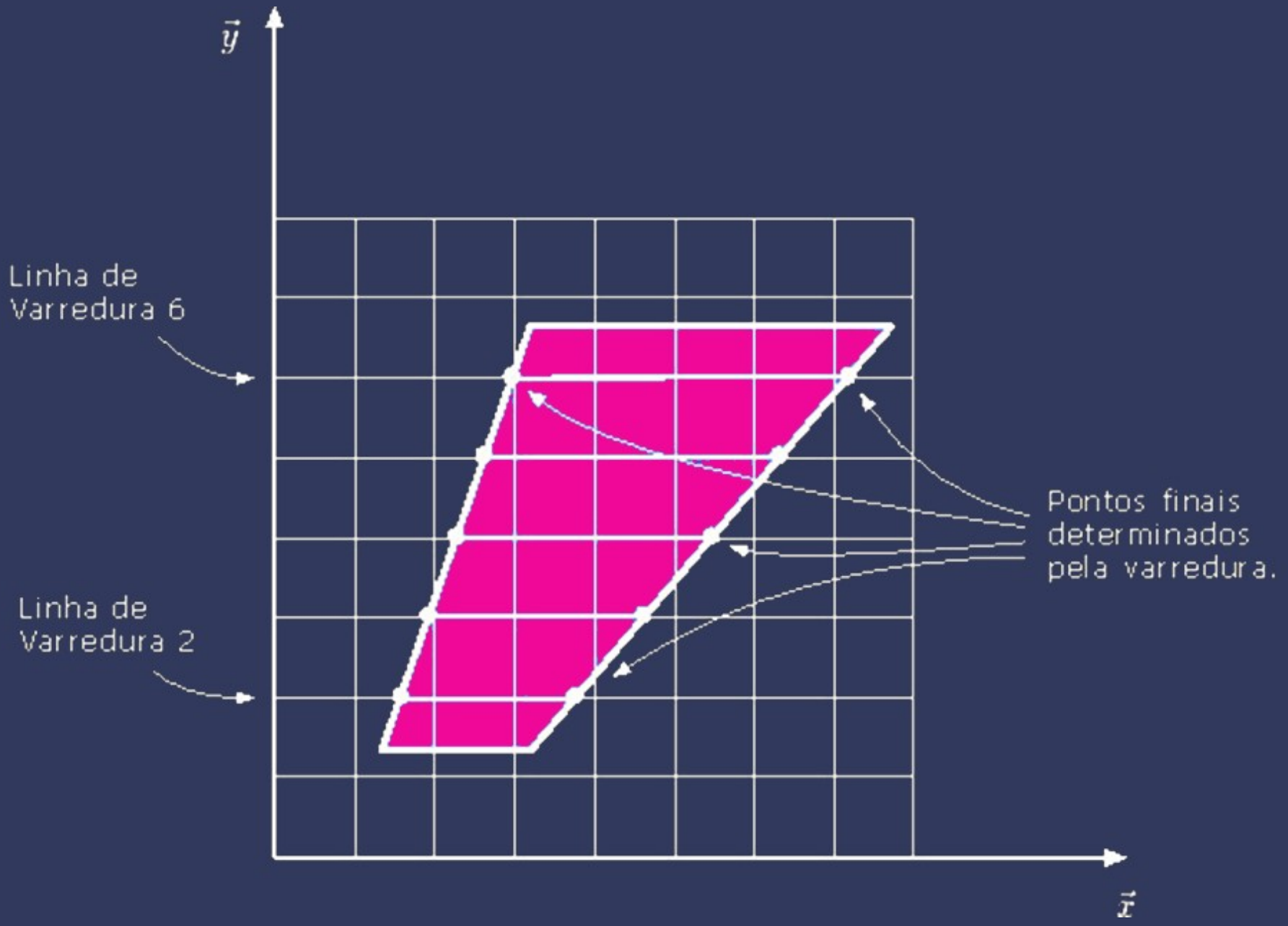
Gerando os Trapézios

Vértices: { A , E , B , D , C }



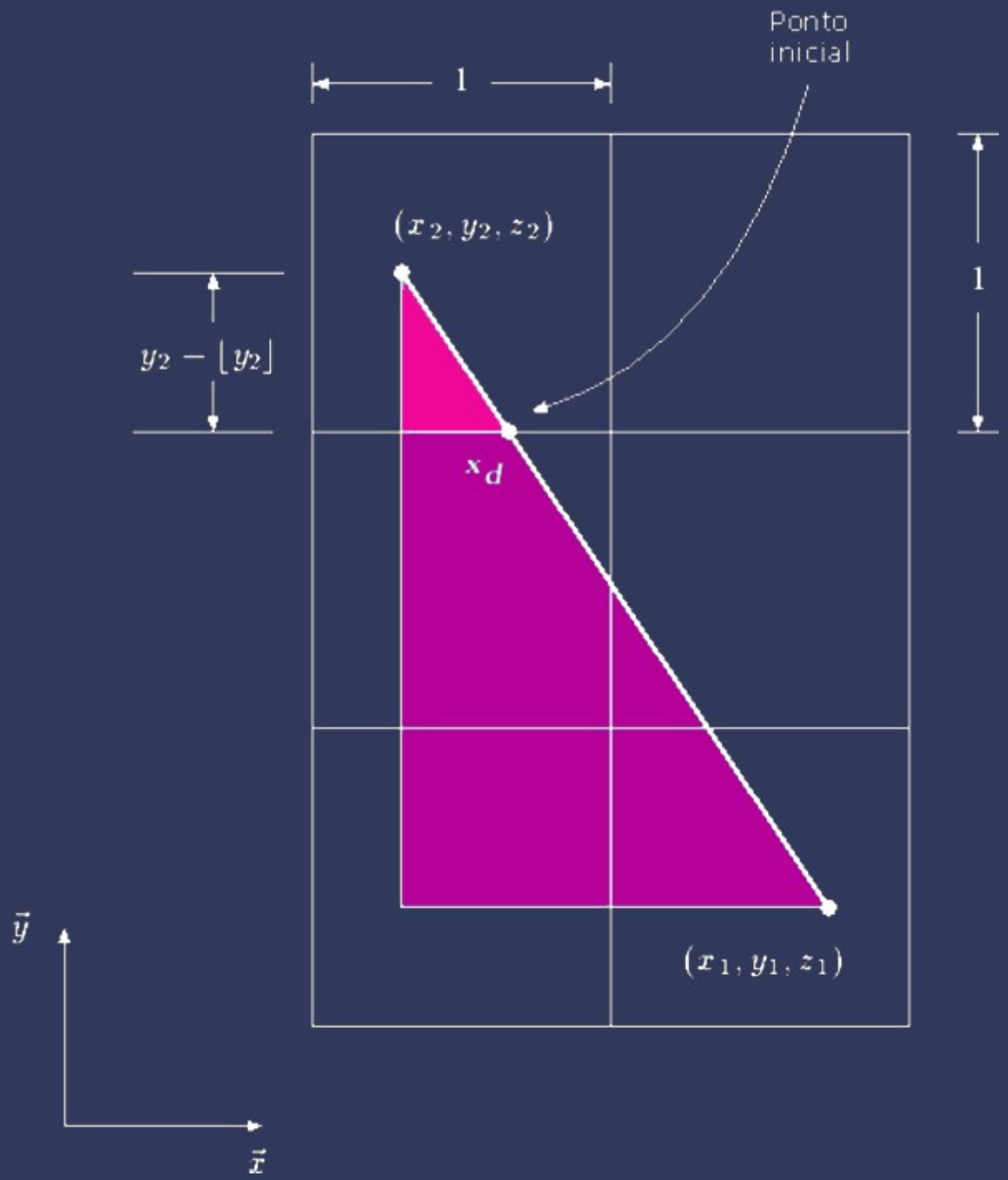
Conversão por Varredura

- A idéia aqui é simples. Vamos criar um rastreador de bordas que segue os pontos finais das linhas formadas pela intersecção de cada linha de varredura com o trapézio.
- Este rastreador de bordas pode ser definido com facilidade como uma estrutura de dados simples que é inicializada para a linha de varredura no topo de cada trapézio e atualizada para cada linha de varredura subseqüente.


$\vec{y}$
Linha de Varredura 6
Linha de Varredura 2
Pontos finais determinados pela varredura.
$\vec{x}$

Inicializando o Rastreador de Bordas

- Suponha que seja dada uma borda definida pelos pontos $P_1 = (x_1, y_1, z_1)$ e $P_2 = (x_2, y_2, z_2)$, definida em espaço de dispositivo (Coordenadas de Viewport).
- Precisamos inicializar o rastreador de bordas calculando a intersecção desta borda com a mais alta linha de varredura que intercepta a mesma.
- O rastreador vai seguir percorrendo a borda e calculando sua intersecção com as linhas de varredura subseqüentes.
- Se definirmos $Dx = x_2 - x_1$, e $Dy = y_2 - y_1$, então a figura adiante mostra como calcular o ponto inicial (somente os valores de x e y são mostrados por questão de simplicidade).


Ponto
inicial
1
1
(x_2, y_2, z_2)
$y_2 - \lfloor y_2 \rfloor$
x_d
(x_1, y_1, z_1)
$\vec{y}$
$\vec{x}$

Rastreando Bordas

- Podemos ver que o ponto inicial é dado por:
  ($x_2 + x_d$, $\lfloor y_2 \rfloor$, $z_2 + z_d$) EQ. 10.1
- onde $\lfloor y_2 \rfloor$ é o valor discretizado de y_2, na posição da linha de varredura-base mais próxima abaixo de y_2, ou seja, valor inteiro da linha de pixels onde o valor y_2 cairá dentro (discretização-piso).
- Para calcular x_d,

  (semelhança de triângulos)

$$\frac{x_d}{y_2 - \lfloor y_2 \rfloor} = \frac{x1 - x2}{y2 - y1} = -\frac{dx}{dy}$$

- ou:

$$x_d = -(y_2 - \lfloor y_2 \rfloor)\frac{\Delta x}{\Delta y}$$

Atualizando o Rastreador de Bordas para Linhas de Varredura Subseqüentes

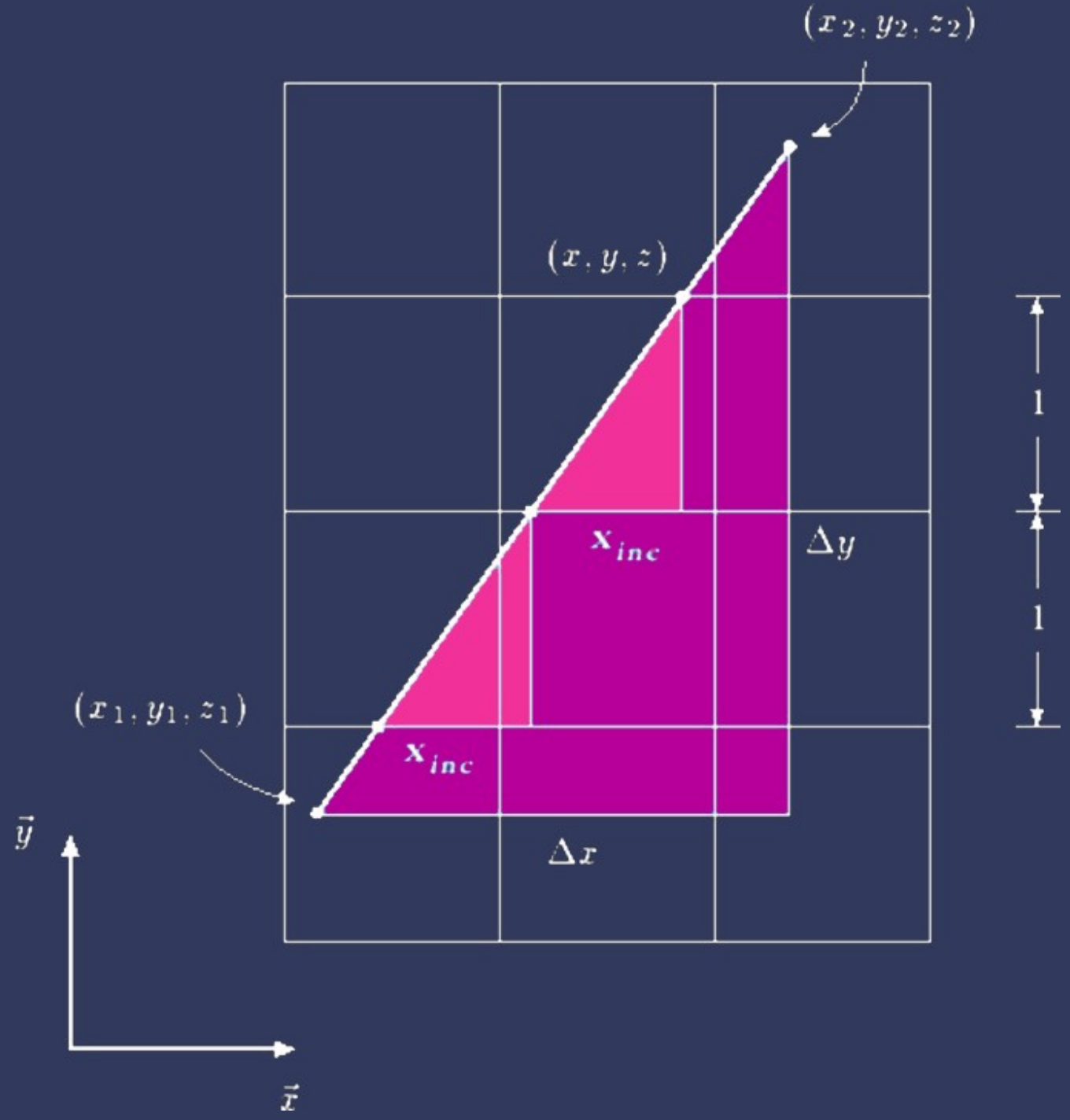
- Suponha que seja dada uma borda definida por dois pontos
  $P_1 = (x_1, y_1, z_1)$ e $P_2 = (x_2, y_2, z_2)$, definidos em espaço de dispositivo.
- A equação anterior mostra como calcular o ponto inicial do rastreador.
- Se definirmos

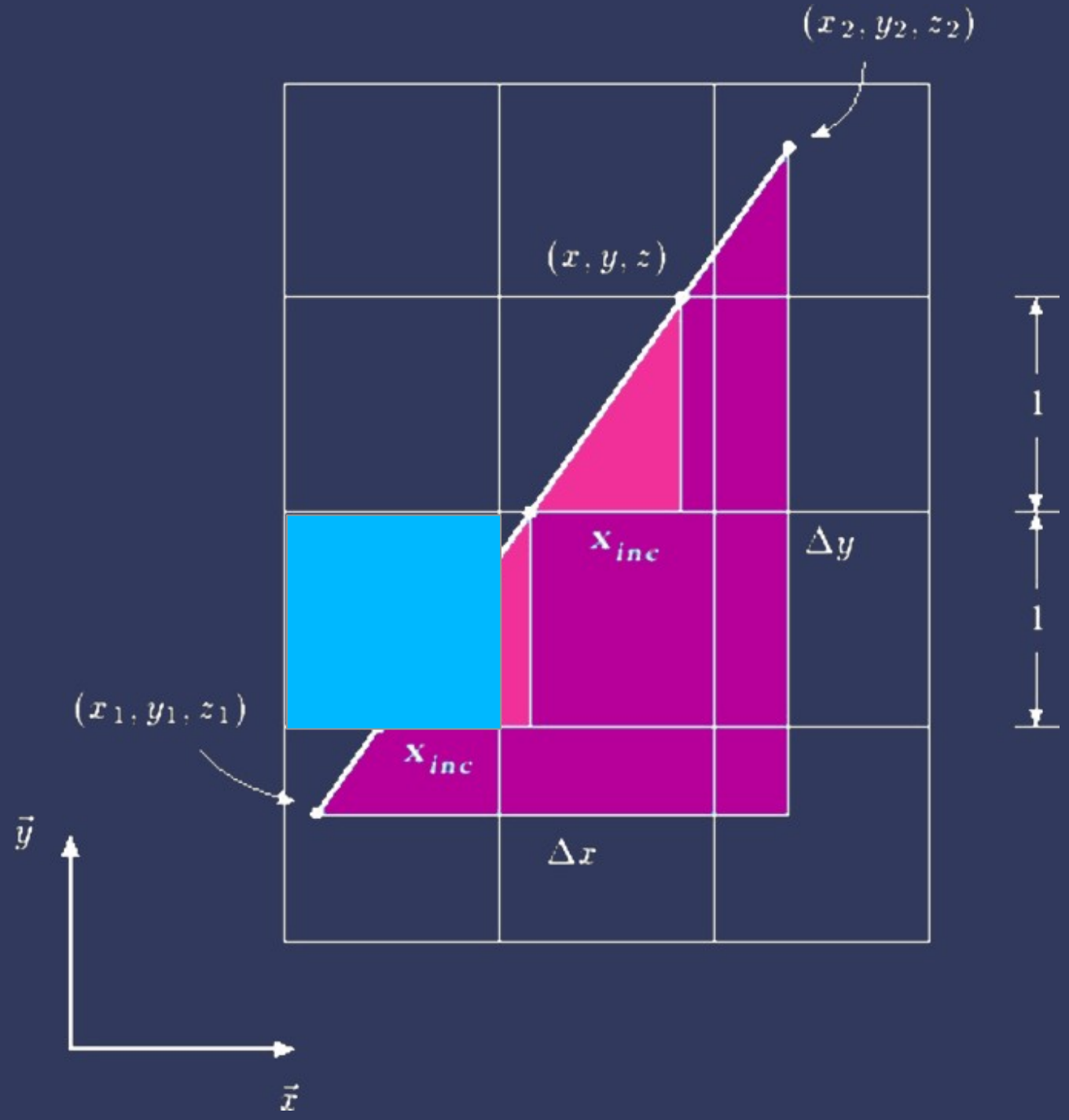
  $\Delta x = x_2 - x_1$,

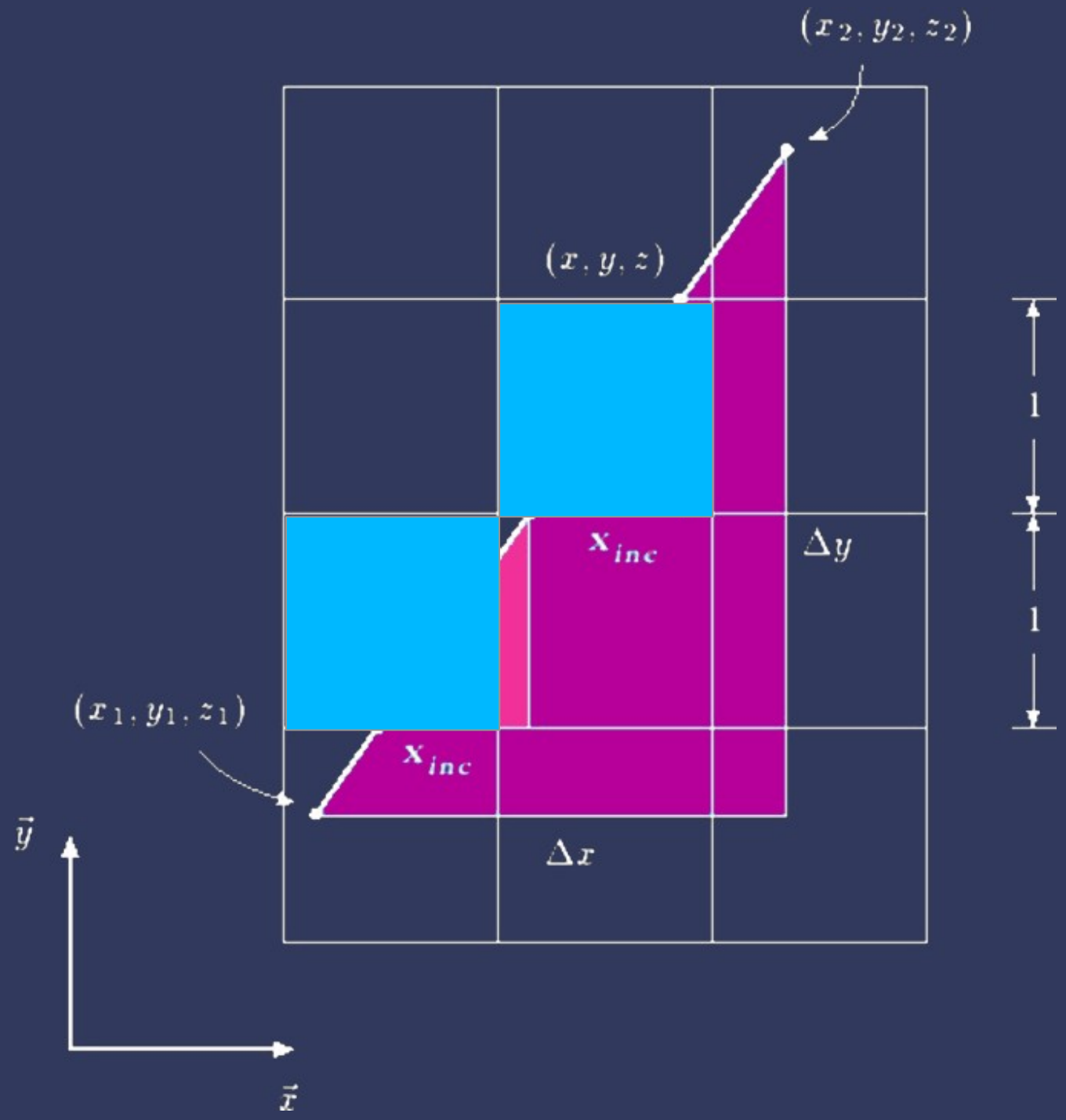
  $\Delta y = y_2 - y_1$, e

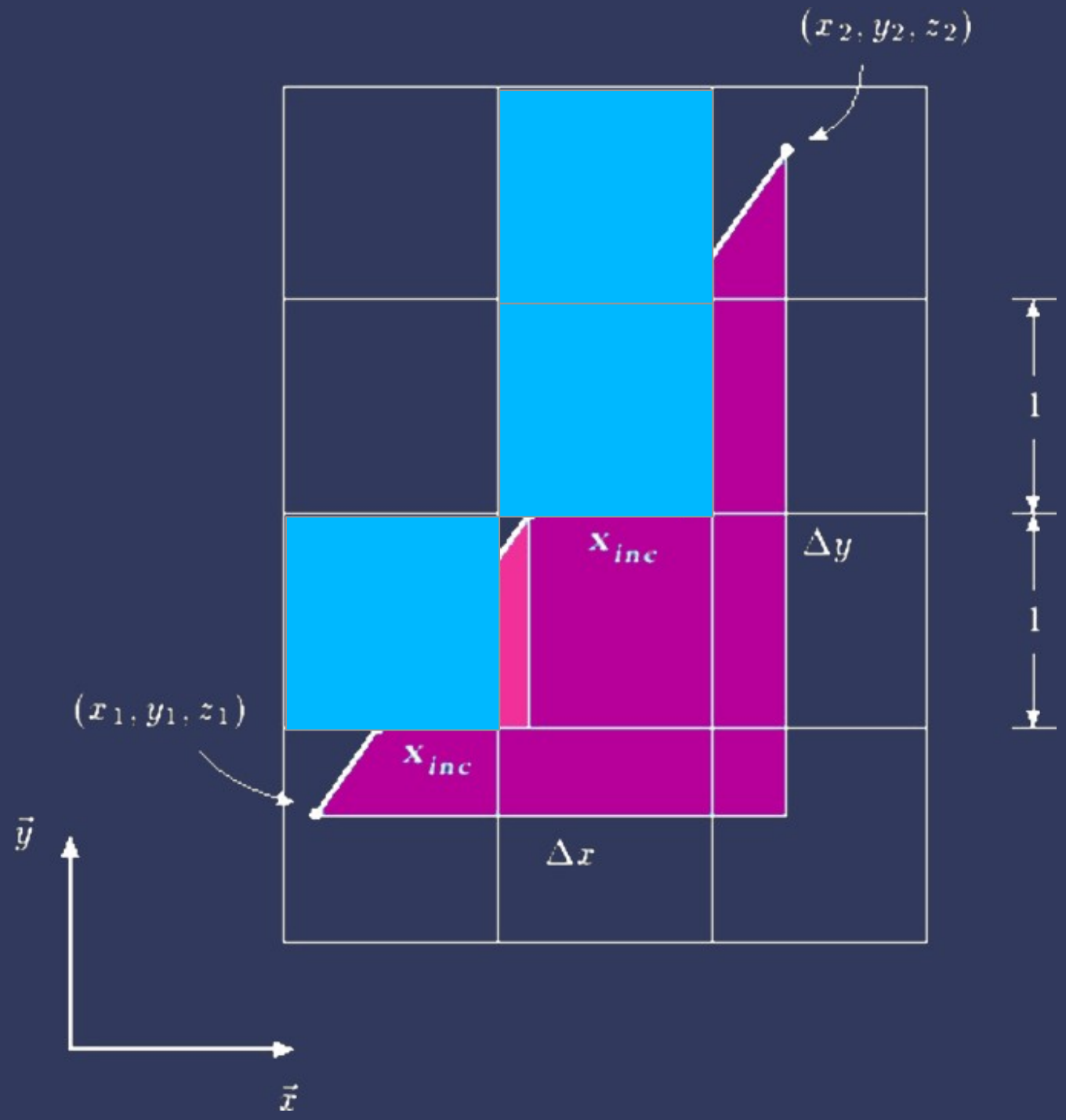
  $\Delta z = z_2 - z_1$,

  então a figura adiante mostra como atualizar o rastreador de bordas:


(x_2, y_2, z_2)
(x, y, z)
x_{inc}
Δy
1
1
(x_1, y_1, z_1)
x_{inc}
$\vec{y}$
Δx
$\vec{x}$


(x_2, y_2, z_2)
(x, y, z)
x_{inc}
Δy
1
1
(x_1, y_1, z_1)
x_{inc}
Δx
$\vec{y}$
$\vec{x}$


(x_2, y_2, z_2)
(x, y, z)
1
1
x_{inc}
Δy
(x_1, y_1, z_1)
x_{inc}
$\vec{y}$
Δx
$\vec{x}$


(x_2, y_2, z_2)
x_{inc}
Δy
1
1
(x_1, y_1, z_1)
x_{inc}
Δx
$\vec{y}$
$\vec{x}$

Atualizando o Rastreador de Bordas para Linhas de Varredura Subseqüentes

- Da figura, vemos que o ponto inicial subseqüente (na linha de varredura de baixo) ao ponto P = (x , y , z) é:

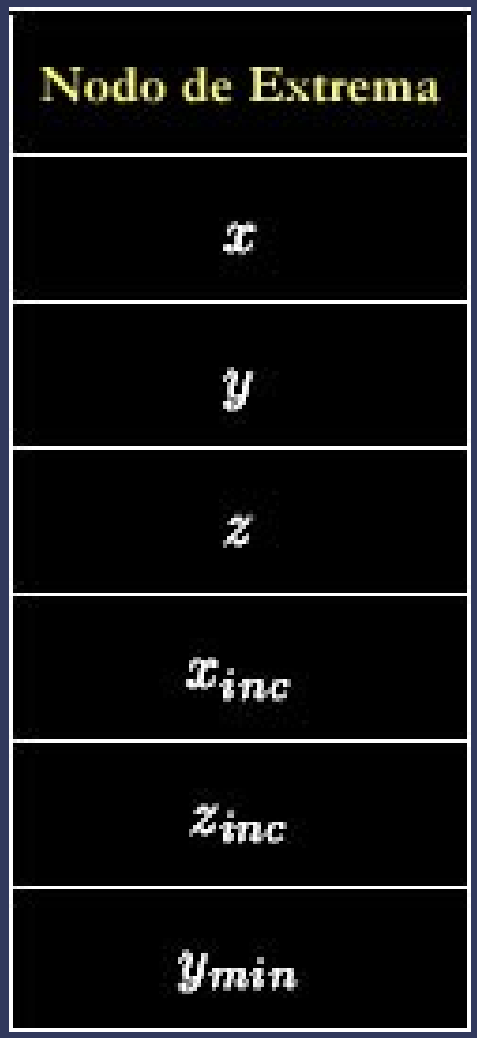
$$P' = (x + x_{inc} , y - 1 , z + z_{inc})$$

- E, de novo por semelhança de triângulos, é fácil calcular o valor de x_{inc} (e z_{inc}):

$$\frac{x_{inc}}{y_{inc}} = \frac{\Delta x}{\Delta y} \qquad x_{inc} = \frac{\Delta x}{\Delta y}$$

EQ 10.2

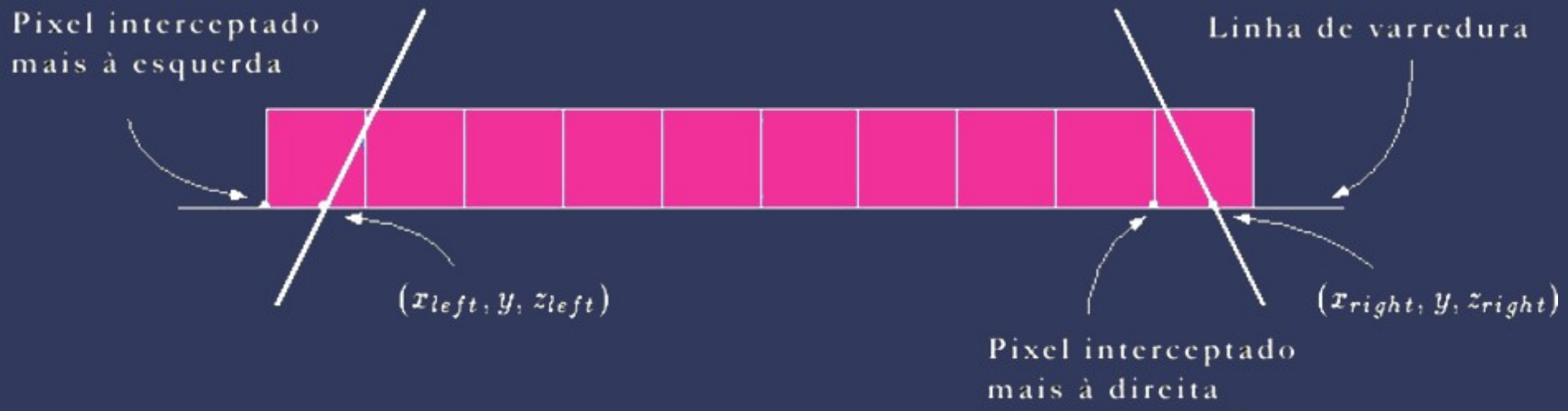
A Estrutura de Dados de Pontos Finais



- onde (x,y,z) é o ponto inicial,
- x_{inc} é o incremento adicionado à coordenada x para ir de uma linha de varredura à outra,
- z_{inc} é o incremento adicionado à coordenada z para ir de uma linha de varredura à outra e
- y_{min} é o limite inferior do trapézio que indica ao algoritmo quando parar de

Identificando os Pixels Interceptados a cada Linha de Rastreamento

- Para encontrar os pixels que interceptam um trapézio, criamos dois rastreadores de bordas e percorremos o trapézio no sentido y, linha de varredura por linha de varredura.
- Isto nos permite determinar, para cada linha de varredura, quais são os pontos extremos de um segmento de reta que forma a intersecção da linha de varredura com as bordas laterais do trapézio.
- Para encontrar quais são os pixels que interceptam o trapézio em uma determinada linha de varredura, nós precisamos apenas determinar as coordenadas do canto esquerdo inferior de cada pixel.


Pixel interceptado
mais à esquerda
Linha de varredura
(x_{left}, y, z_{left})
$(x_{right}, y, z_{right})$
Pixel interceptado
mais à direita

Identificando os Pixels Interceptados a cada Linha de Rastreamento

- Vemos que os pixels afetados são aqueles entre $(\lfloor x_{esq} \rfloor, y)$ $(\lfloor x_{dir} \rfloor, y)$
- Para encontrar estes pixels basta incrementar x de 1 em 1, iniciando em $\lfloor x_{esq} \rfloor$ e terminando em $\lfloor x_{dir} \rfloor$
- O valor de y é constante porque as linhas de varredura são horizontais.

Estabelecendo um Valor de Profundidade para Cada Pixel

- Agora já sabemos como encontrar os valores de x e de y para cada pixel que compõe o polígono.
- Falta apenas encontrar o valor de z. Este valor é necessário para a próxima etapa (o Z-Buffering) e também para a iluminação.

Estabelecendo um Valor de Profundidade para Cada Pixel

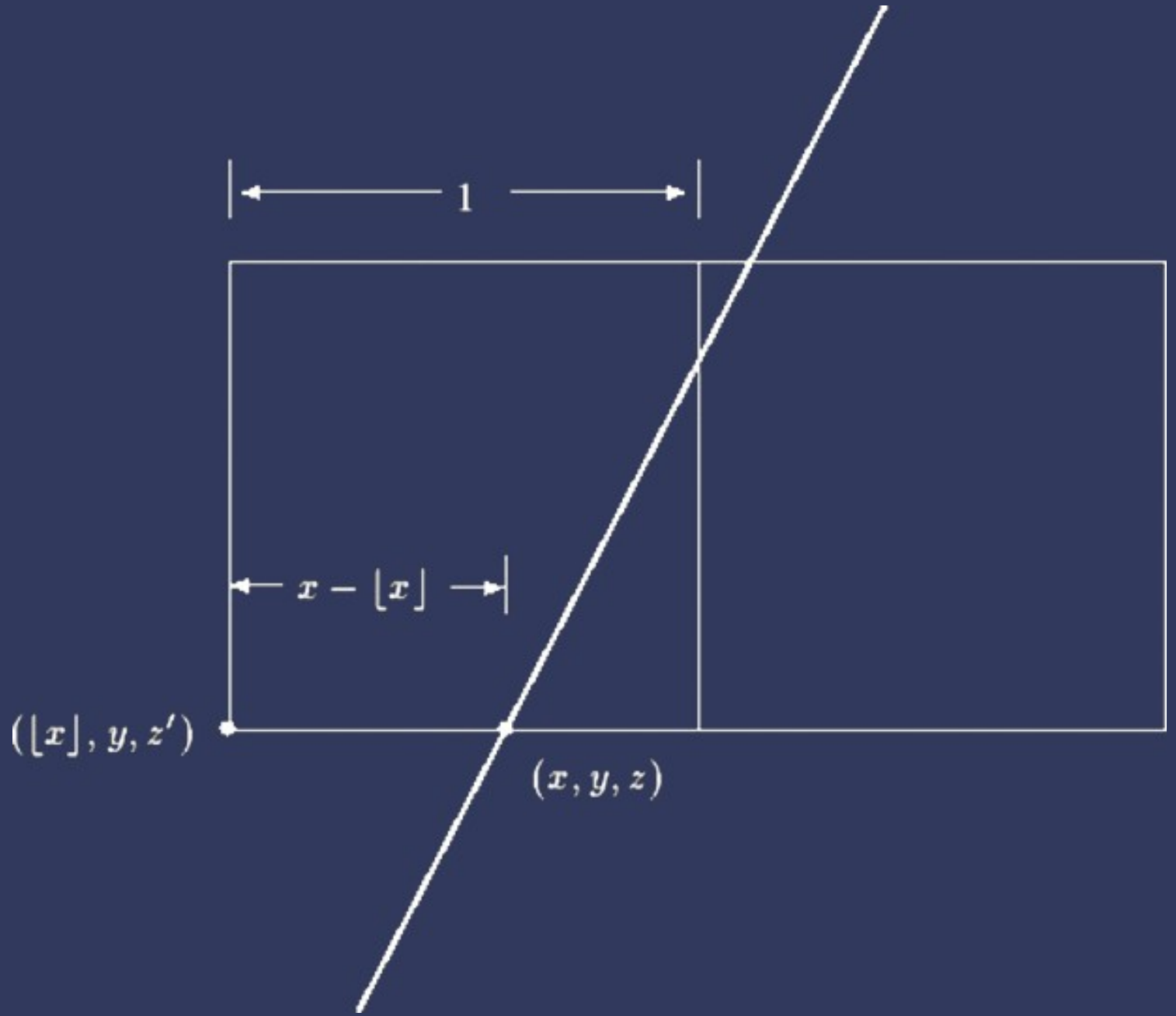
- Este valor vem direto da normal do polígono, n:

$$z_{hinc} = \frac{-x_n}{z_n} \qquad \text{EQ 10.3}$$

- Onde z_{hinc} é o incremento em z entre dois pixels consecutivos horizontalmente e x_n e z_n são as componentes x e z do vetor normal do polígono.
- Como nossos pixels são indexados em sua ponta inferior esquerda, o último passo é calcular o valor de z para o primeiro pixel da linha:

$$z' = z - (x - \lfloor x \rfloor) z_{inc} \qquad \text{EQ 10.4}$$

Estabelecendo um Valor de Profundidade para Cada Pixel



Resumindo (Algoritmo de Pixel Shading v.1):

Para cada Polígono Q:

Para cada Trapézio T que forma Q:

Aplique 10.1 para calcular o ponto inicial de ambas bordas de T.

Aplique 10.2 para calcular os incrementos em x e em z entre as linhas de varredura.

Para cada par de pontos extremos das bordas esquerda e direita de T:

Aplique 10.3 e 10.4 para calcular o valor de Z do primeiro pixel da linha e o

incremento de z entre os pixels.

Preencha a linha entre os pontos extremos incrementando 1 em x e o valor de 10.3

em z.

fim-para

Incremente x, y e z dos pontos iniciais para a próxima linha de varredura.

fim-para

Fim-para

Resumindo (Algoritmo de Pixel Shading v.1):

Para cada Polígono Q:

Para cada Trapézio T que form

Aplique 10.1 para calcular

Aplique 10.2 para calcular

varredura.

Para cada par de pontos ex

Aplique 10.3 e 10.4 para

incremento de z entre

Preencha a linha en

de 10.3

em z.

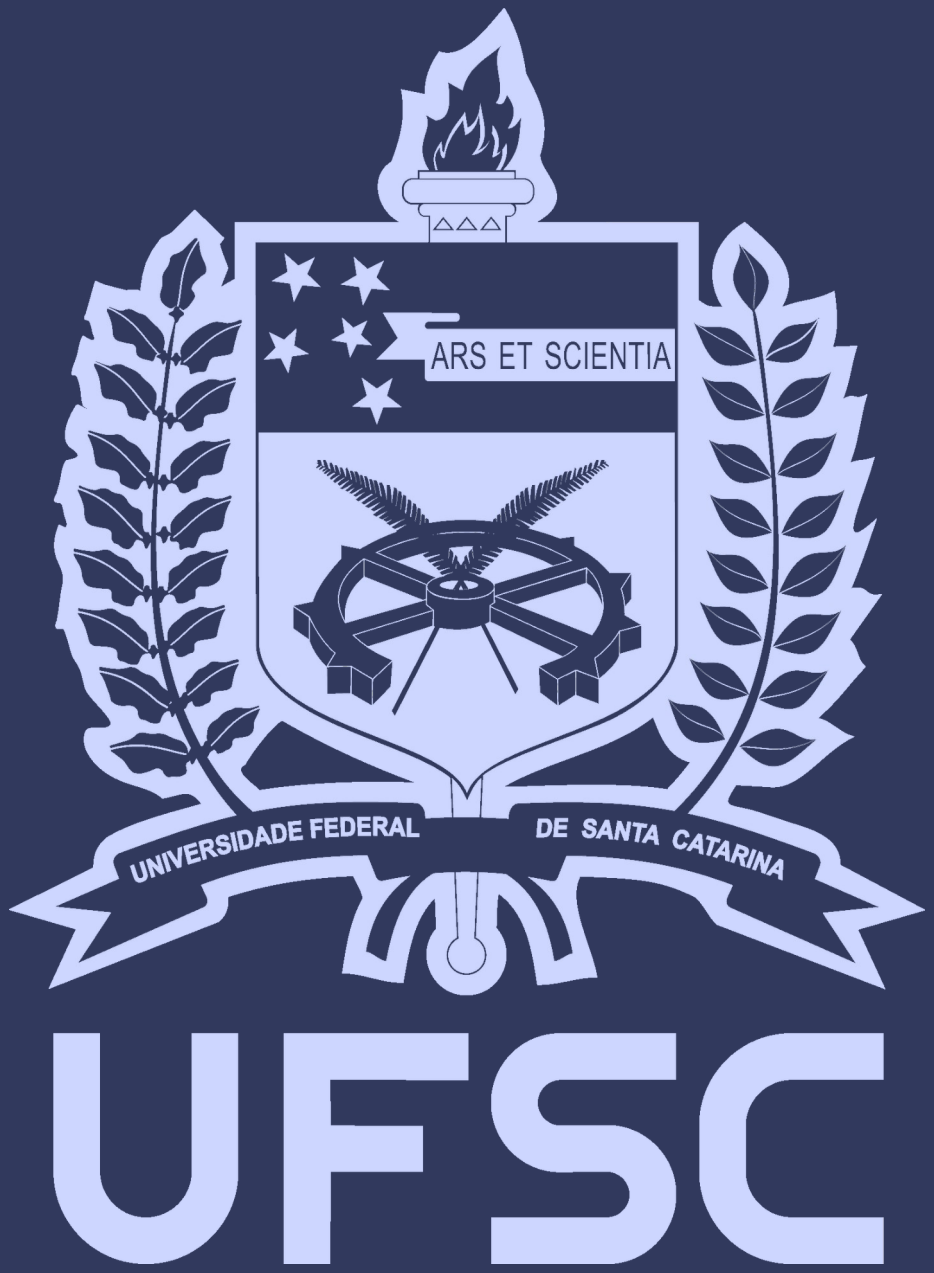
fim-para

Incremente x, y e z dos pont

fim-para

Fim-para

Observe que isto vai fornecer x e y em coordenadas de *viewport* e z em *coordenadas*


ARS ET SCIENTIA
UNIVERSIDADE FEDERAL
DE SANTA CATARINA
UFSC