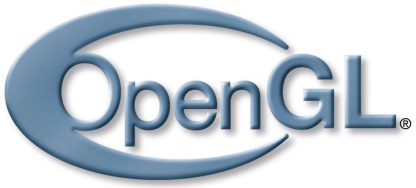
Computação Gráfica

TCC-00291

Assunto: Rasterização

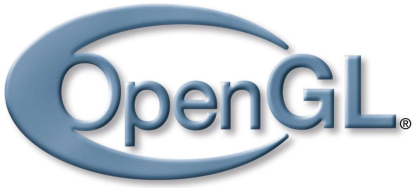


Rasterização

Introdução

Representação Vetorial x Matricial

Definimos os objetos da cena através de primitivas geométricas:
pontos, segmentos de retas, polígonos, etc.



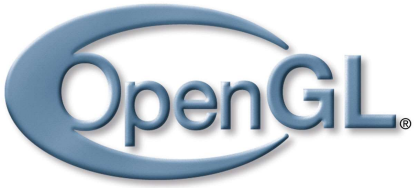
Rasterização

Introdução

Representação Vetorial x Matricial

Definimos os objetos da cena através de primitivas geométricas
: pontos, segmentos de retas, polígonos, etc.

Representação vetorial

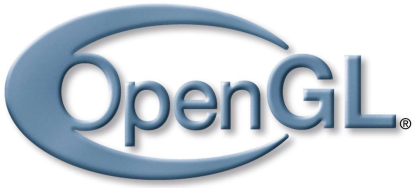


Rasterização

Introdução

Representação Vetorial x Matricial

Dispositivos gráficos podem ser pensados como matrizes de pixels .



Rasterização

Introdução

Representação Vetorial x Matricial

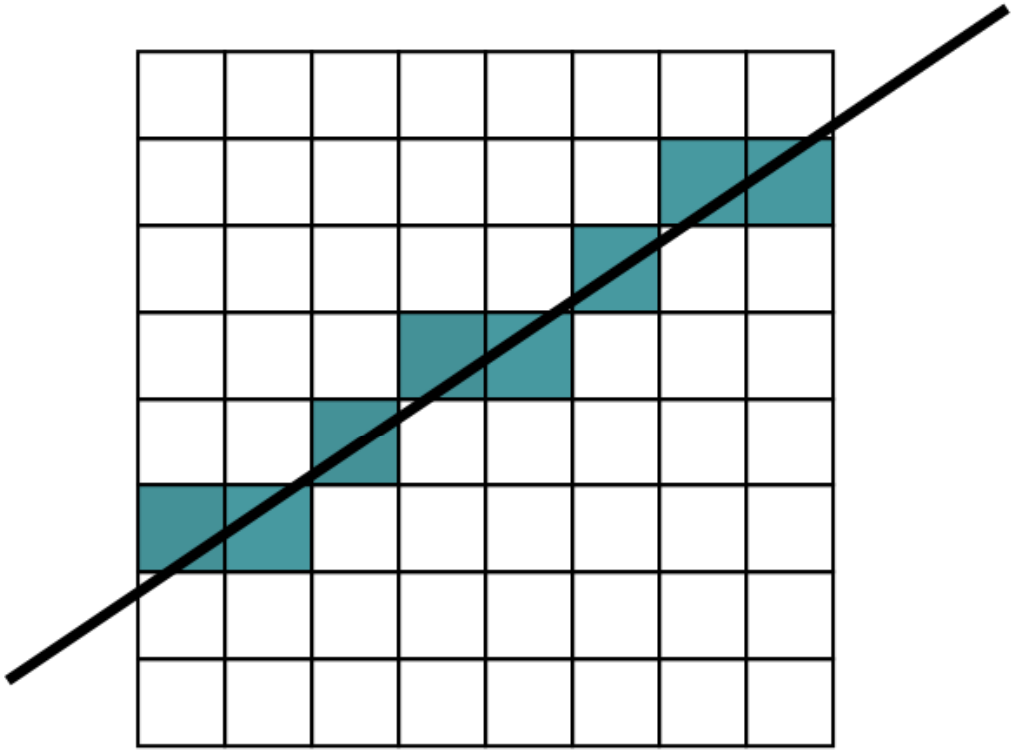
Dispositivos gráficos podem ser pensados como matrizes de pixels .

Representação matricial

Rasterização

Introdução

A rasterização é o processo de conversão entre representações vetorial e matricial.

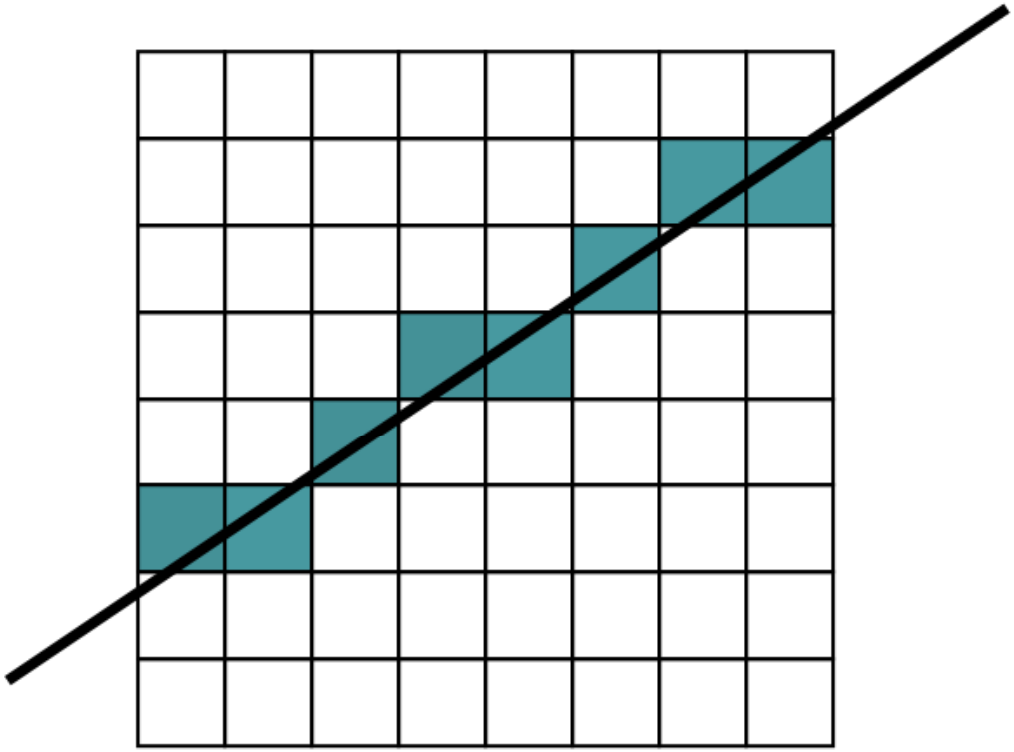


Rasterização

Introdução

A rasterização é um processo de amostragem

Domínio contínuo ⟶ Domínio discreto

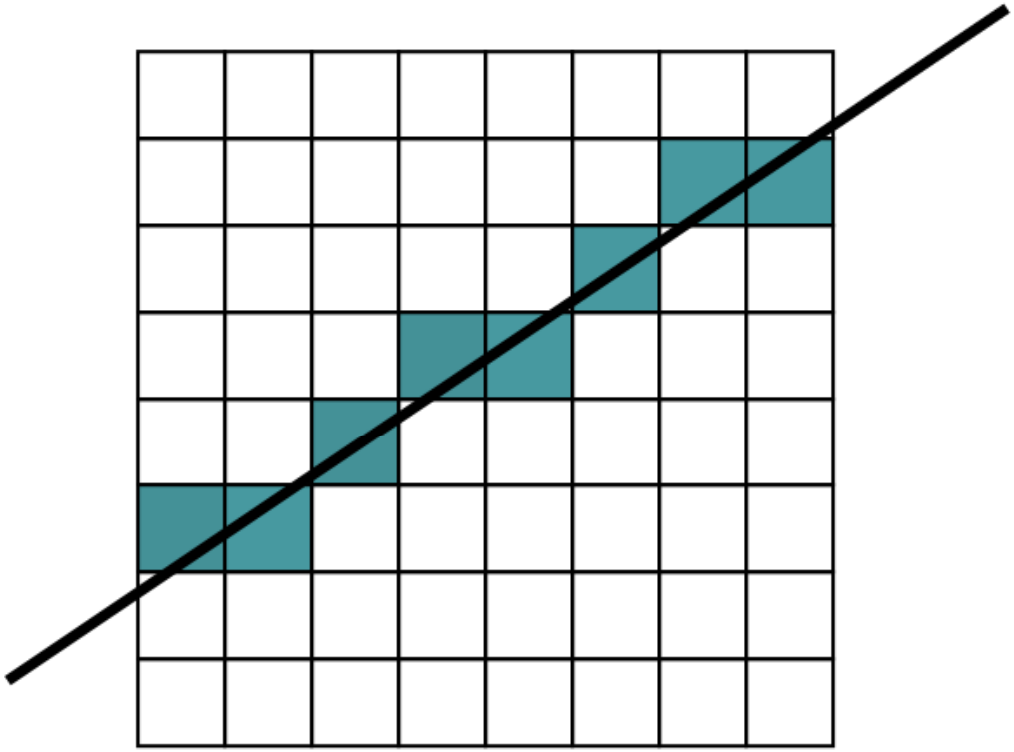


Rasterização

Introdução

A rasterização é um processo de amostragem

Problemas de aliasing são esperados

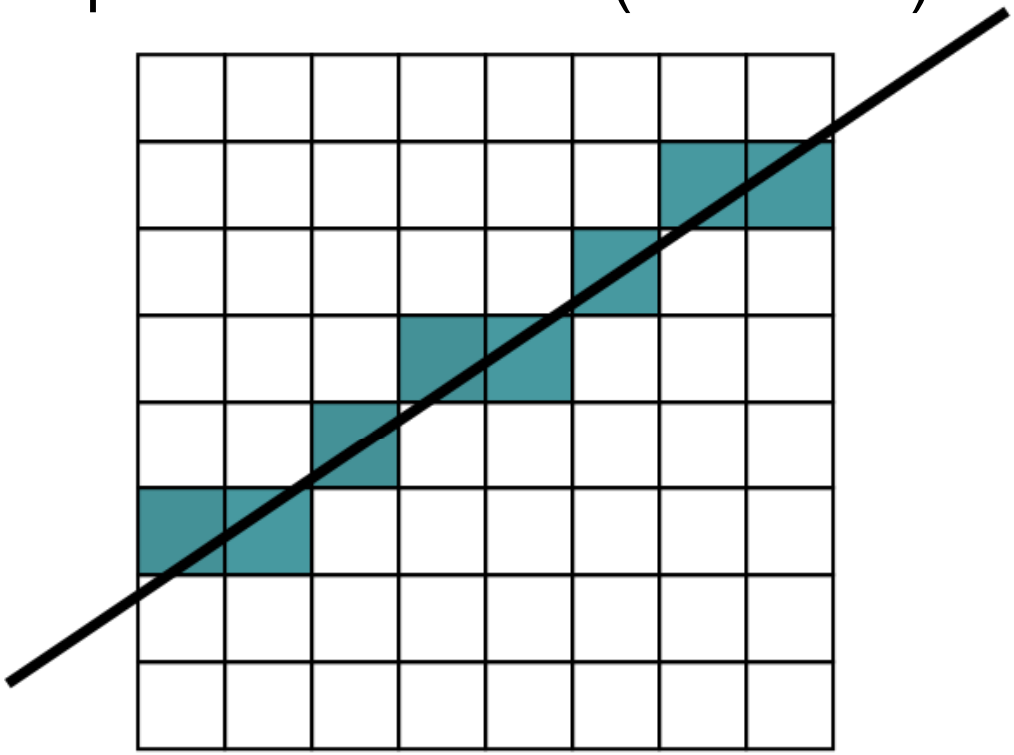


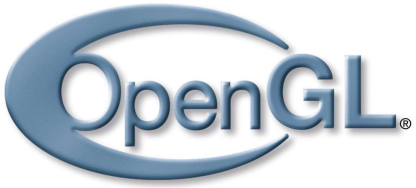
Rasterização

Introdução

A rasterização é um processo de amostragem

Cada primitiva pode gerar um grande número de pixels, logo rapidez é essencial (hardware)

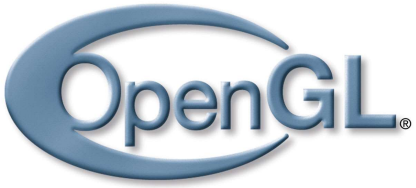




Rasterização

Introdução

Estudaremos apenas a rasterização de segmentos de reta e de polígonos.

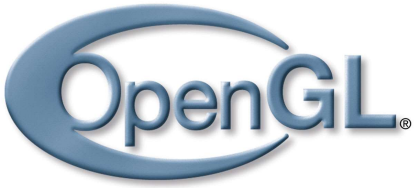


Rasterização

Introdução

Estudaremos apenas a rasterização de segmentos de reta e de polígonos.

Assumiremos que as primitivas já foram clippadas estão contidas no volume de visão.



Rasterização

Introdução

Estudaremos apenas a rasterização de segmentos de reta e de polígonos.

Assumiremos que as primitivas já foram clippadas estão contidas no volume de visão.

Assumiremos também que as coordenadas de um pixel são dadas por:

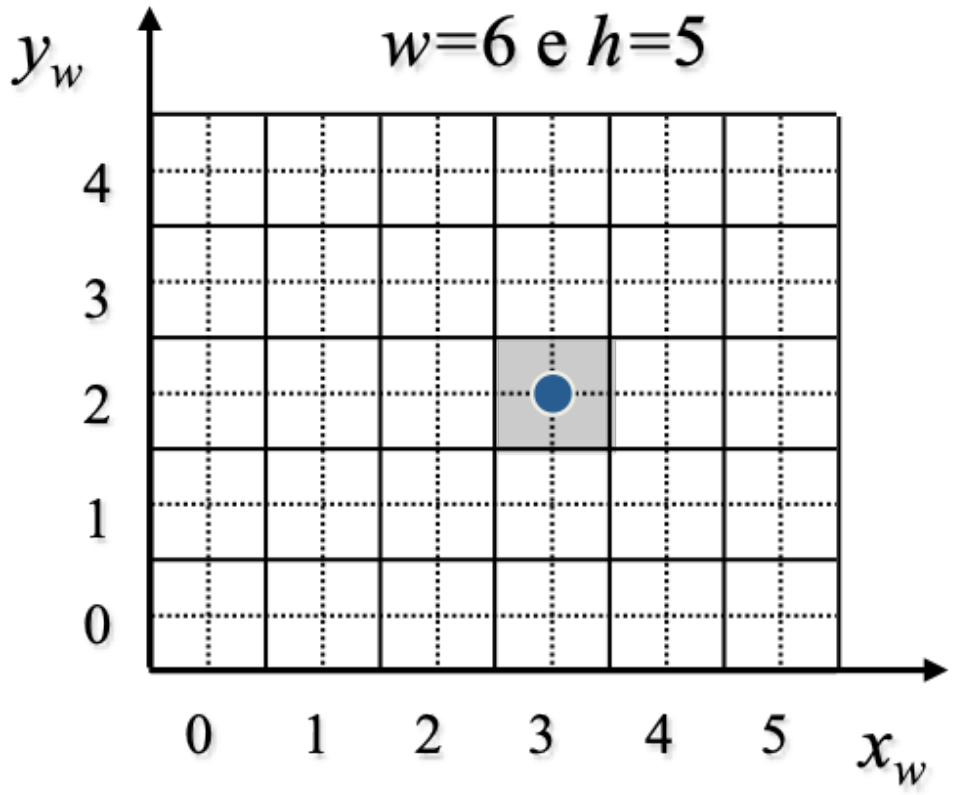
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \lfloor x \rfloor + \frac{1}{2} \\ \lfloor y \rfloor + \frac{1}{2} \end{pmatrix}$$

Rasterização

Introdução

Assumiremos também que as coordenadas de um pixel são dadas por:

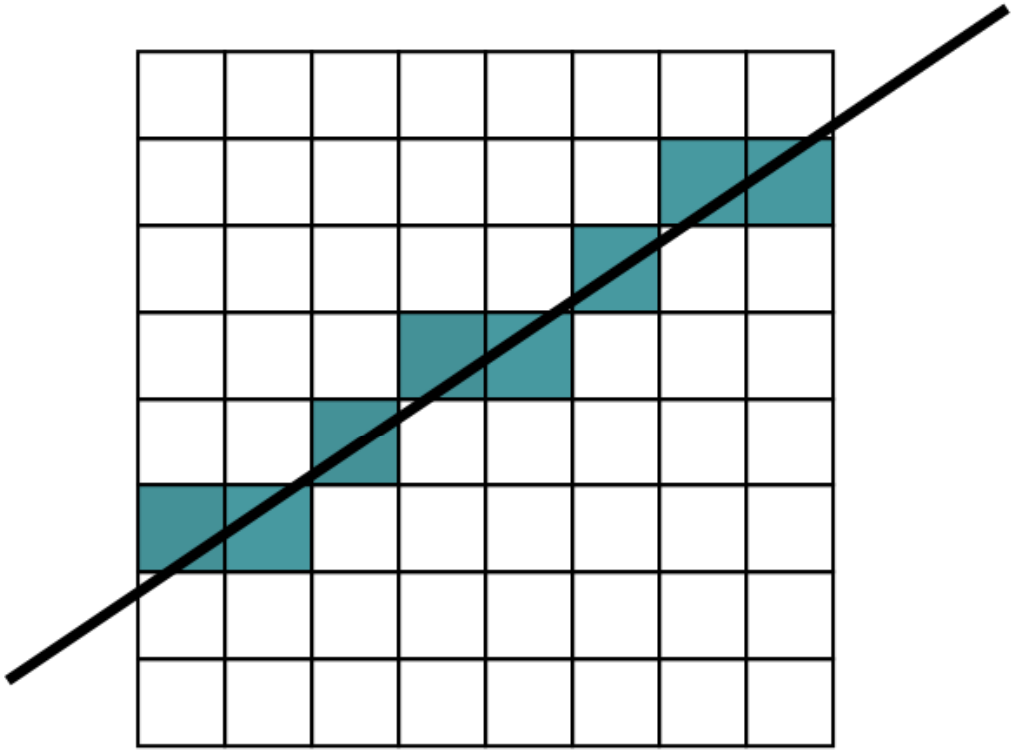
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \lfloor x \rfloor + \frac{1}{2} \\ \lfloor y \rfloor + \frac{1}{2} \end{pmatrix}$$



Rasterização

Segmentos

Como decidir quais pixels devem ser acesos para que seja traçada uma linha?

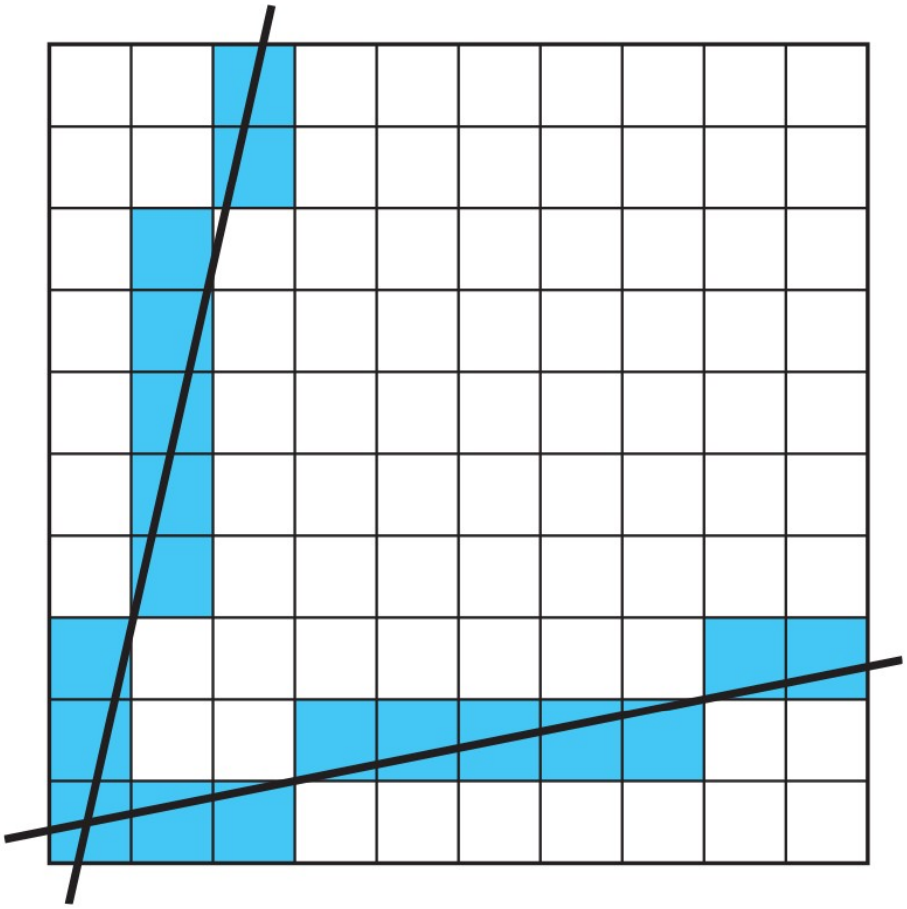


Rasterização

Critério de Bresenham

$$|x_2 - x_1| \geq |y_2 - y_1|$$

x dominante: um pixel por coluna



Rasterização

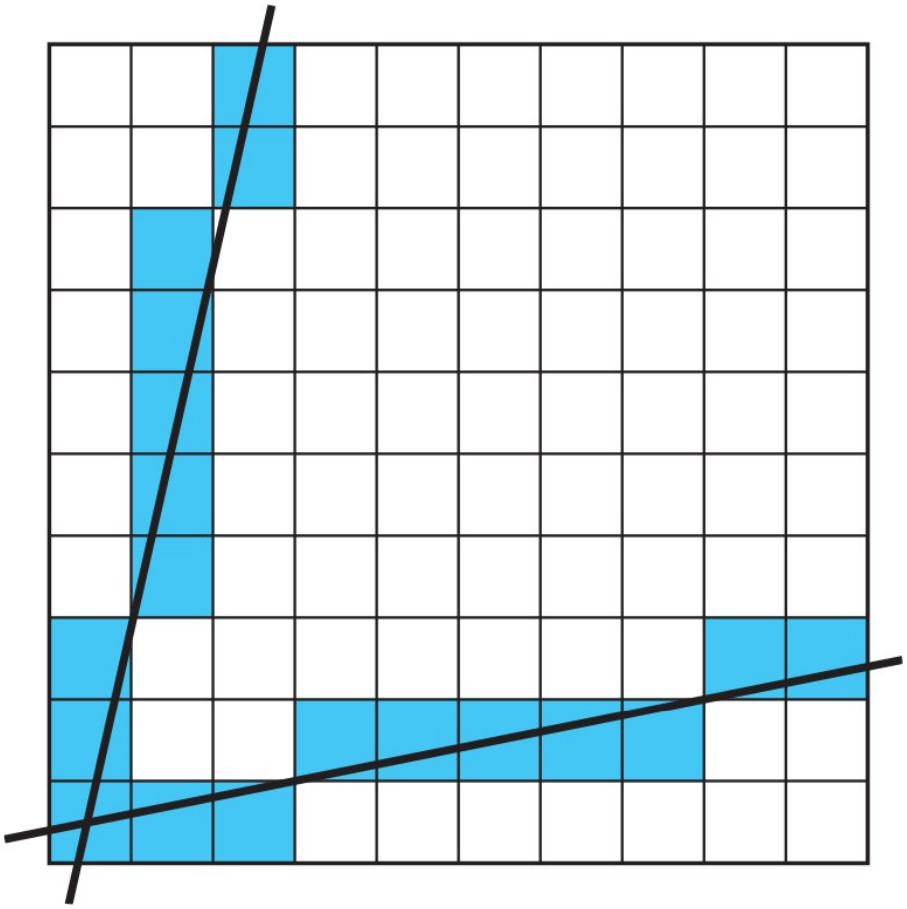
Critério de Bresenham

$$|x_2 - x_1| \geq |y_2 - y_1|$$

x dominante: um pixel por coluna

$$|x_2 - x_1| \leq |y_2 - y_1|$$

y dominante: um pixel por linha



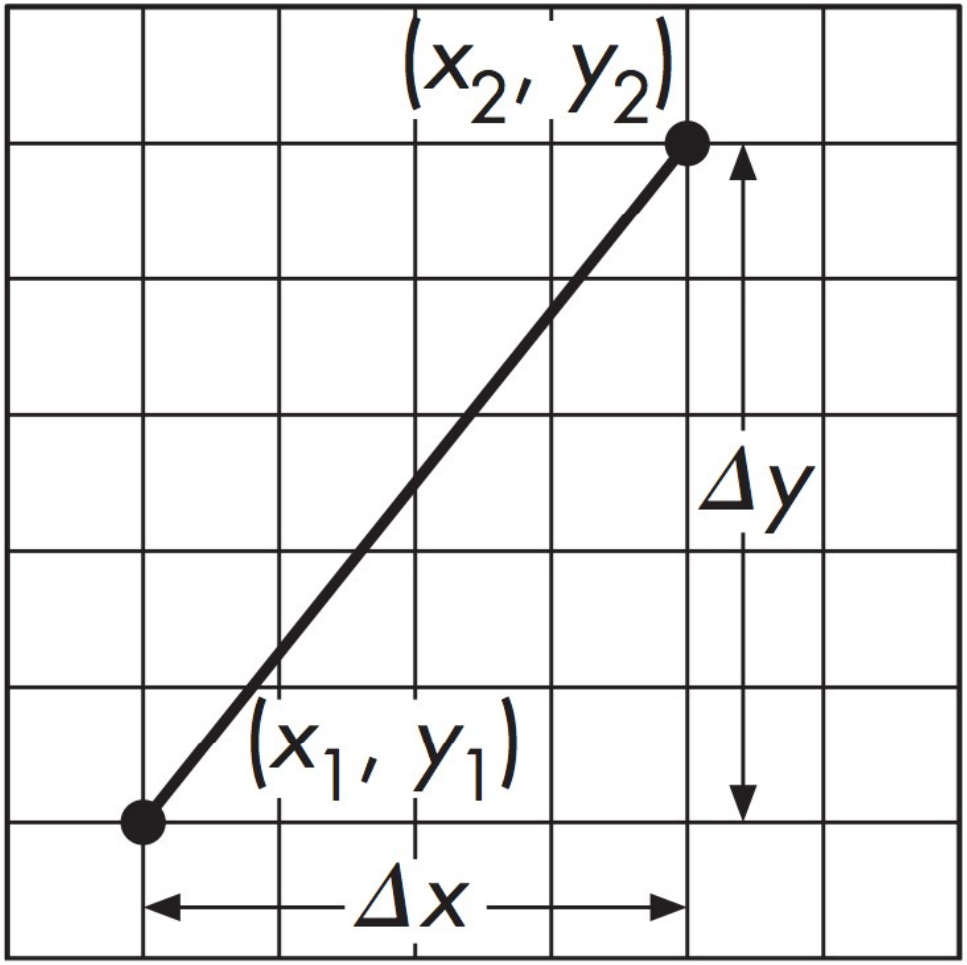
Rasterização

Algoritmo DDA

$$y_i = mx_i + b$$

$$\downarrow$$

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x}.$$



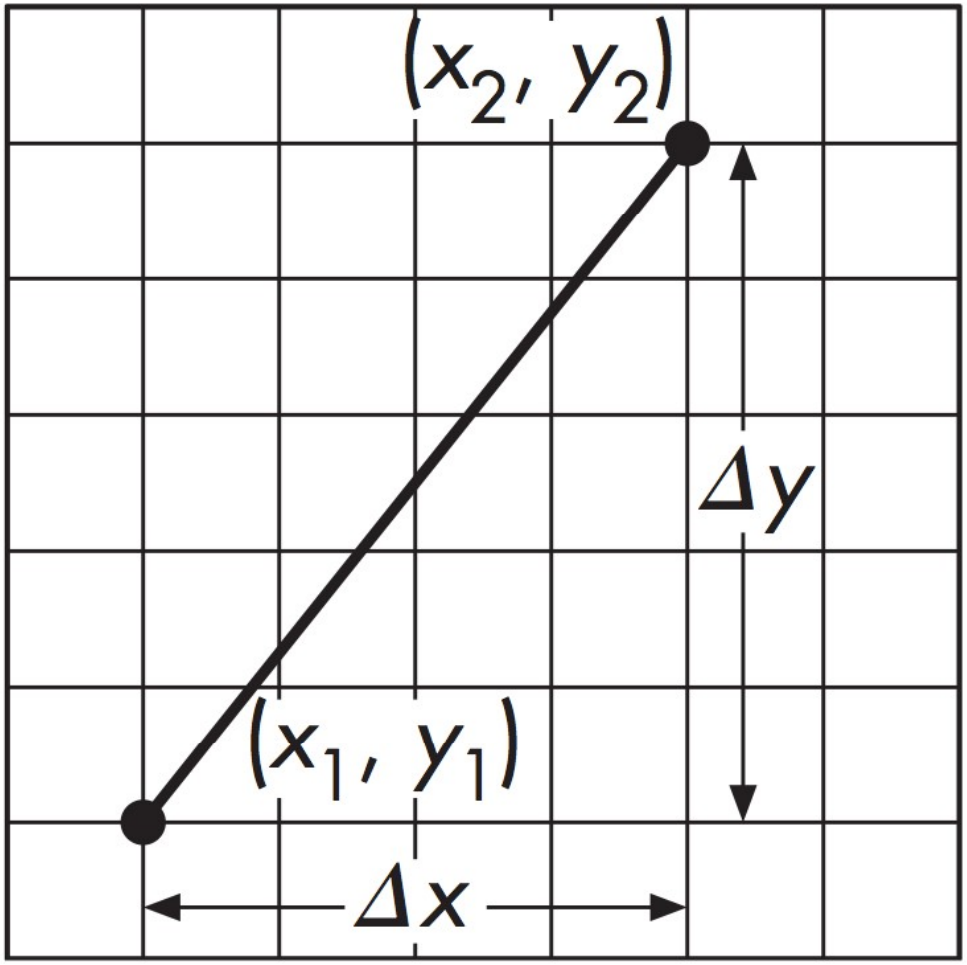
Rasterização

Algoritmo DDA

```
void linha(int x1, int y1,
           int x2, int y2)
{
 float m = (y2-y1)/(x2-x1);
 float b = y1 - m*x1;
 float y;

 fragmento(x1,y1);
 while( x1 < x2 )
 {
   x1++;
   y = m*x1 + b;

   fragmento(x1,Math.round(y));
 }
}
```



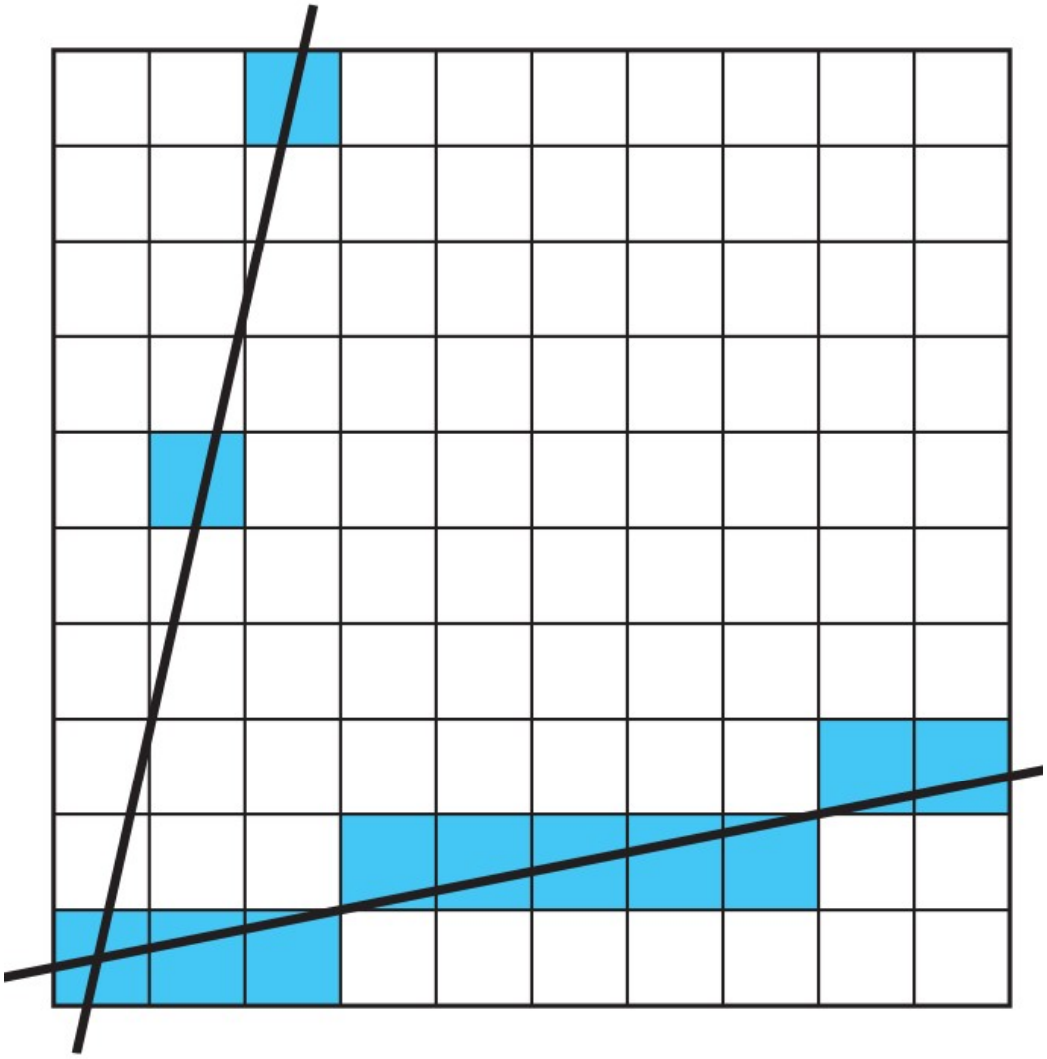
Rasterização

Algoritmo DDA

```
void linha(int x1, int y1,
           int x2, int y2)
{
 float m = (y2-y1)/(x2-x1);
 float b = y1 - m*x1;
 float y;

 fragmento(x1,y1);
 while( x1 < x2 )
 {
   x1++;
   y = m*x1 + b;

   fragmento(x1,Math.round(y));
 }
}
```



Obs. $0 \leq m \leq 1$

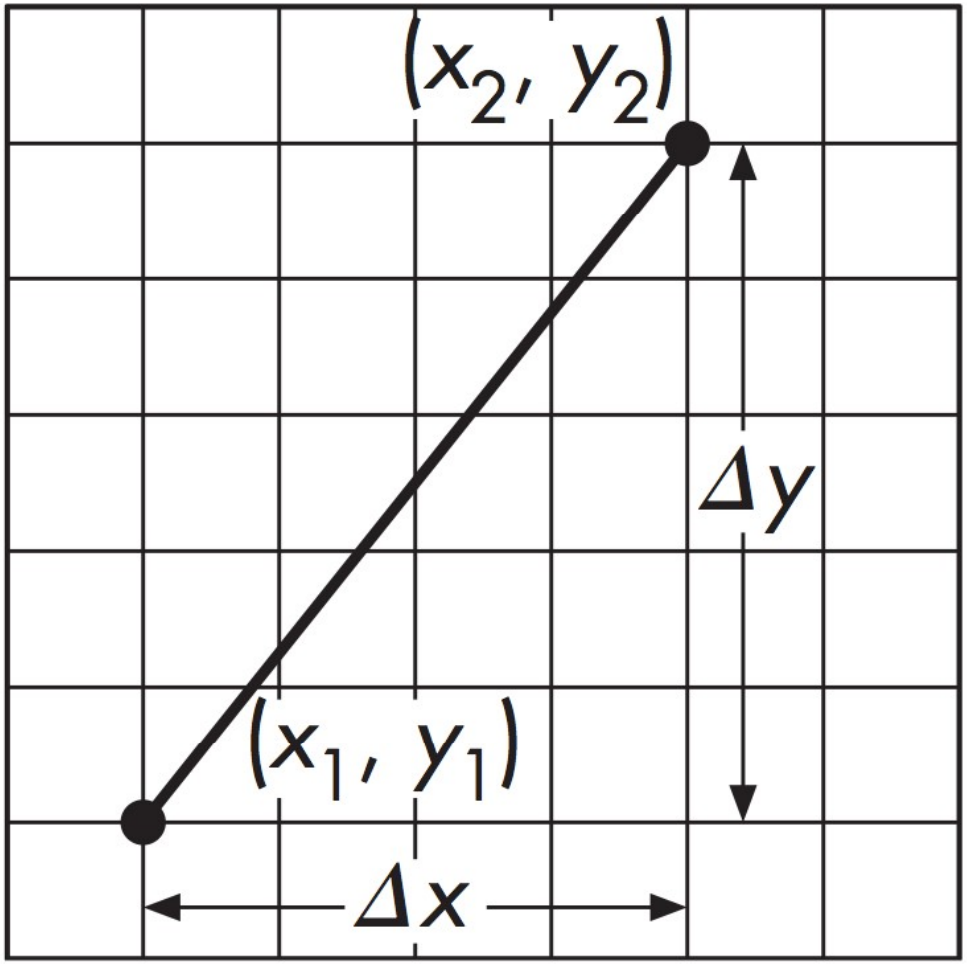
Rasterização

Algoritmo DDA

```
void linha(int x1, int y1,
           int x2, int y2)
{
 float m = (y2-y1)/(x2-x1);
 float b = y1 - m*x1;
 float y;

 fragmento(x1,y1);
 while( x1 < x2 )
 {
   x1++;
   y = m*x1 + b;

   fragmento(x1,Math.round(y));
 }
}
```



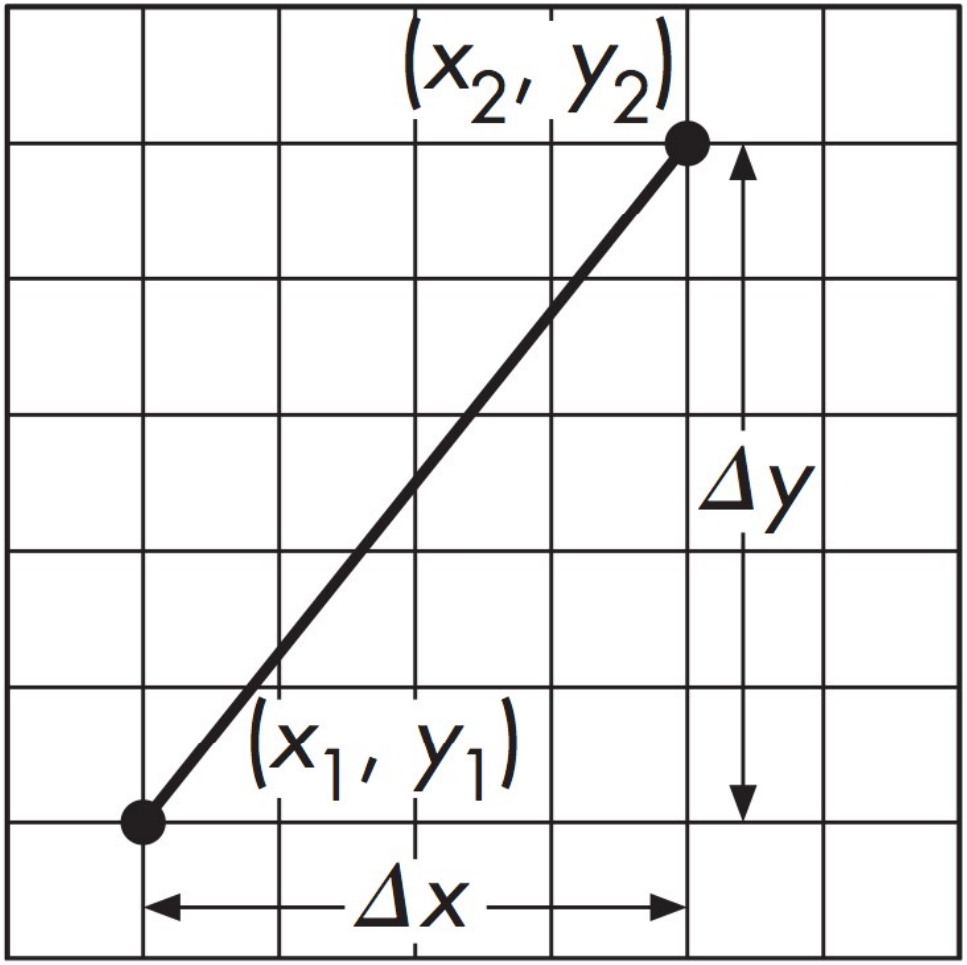
Como economizar?

Rasterização

Algoritmo DDA

```
void linha(int x1, int y1,
           int x2, int y2)
{
 float m = (y2-y1)/(x2-x1);
 float b = y1 - m*x1;
 float y;

 fragmento(x1,y1);
 while( x1 < x2 )
 {
   x1++;
   y += m;
   fragmento(x1,Math.round(y));
 }
}
```

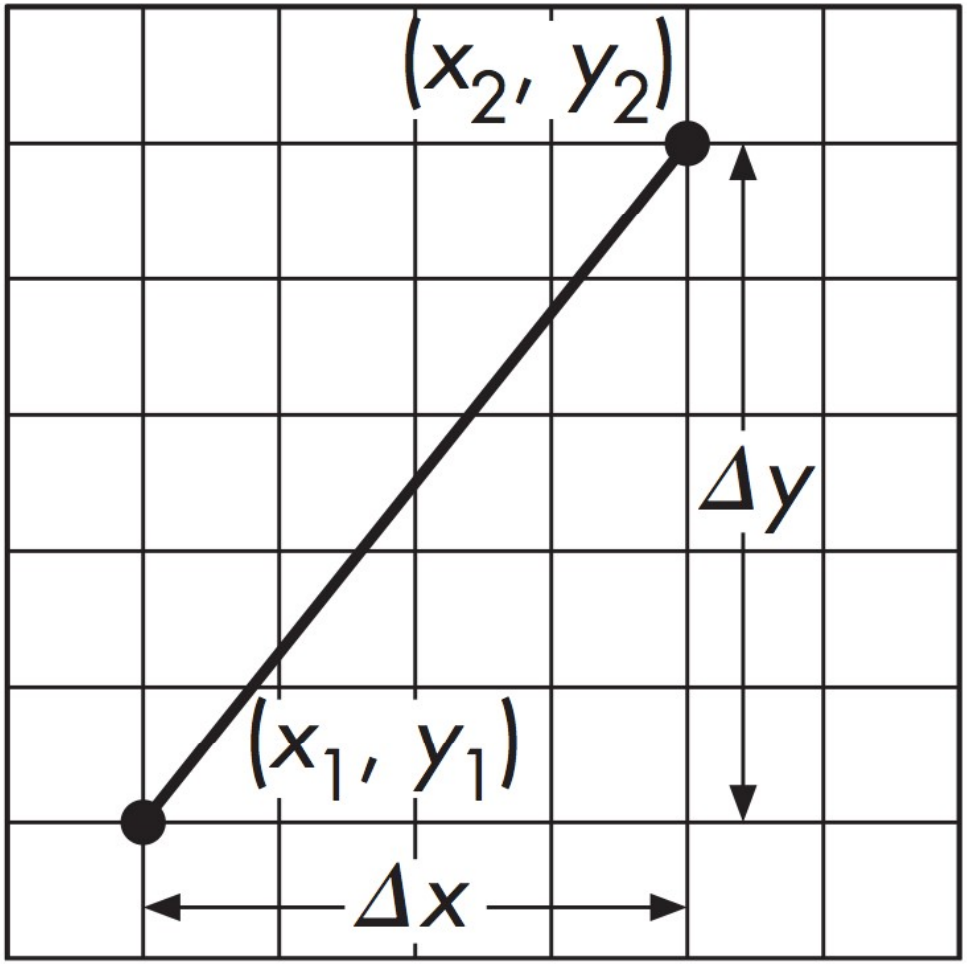


Rasterização

Algoritmo DDA

```
void linha(int x1, int y1,
           int x2, int y2)
{
 float m = (y2-y1)/(x2-x1);
 float b = y1 - m*x1;
 float y;

 fragmento(x1,y1);
 while( x1 < x2 )
 {
   x1++;
   y += m;
   fragmento(x1,Math.round(y));
 }
}
```

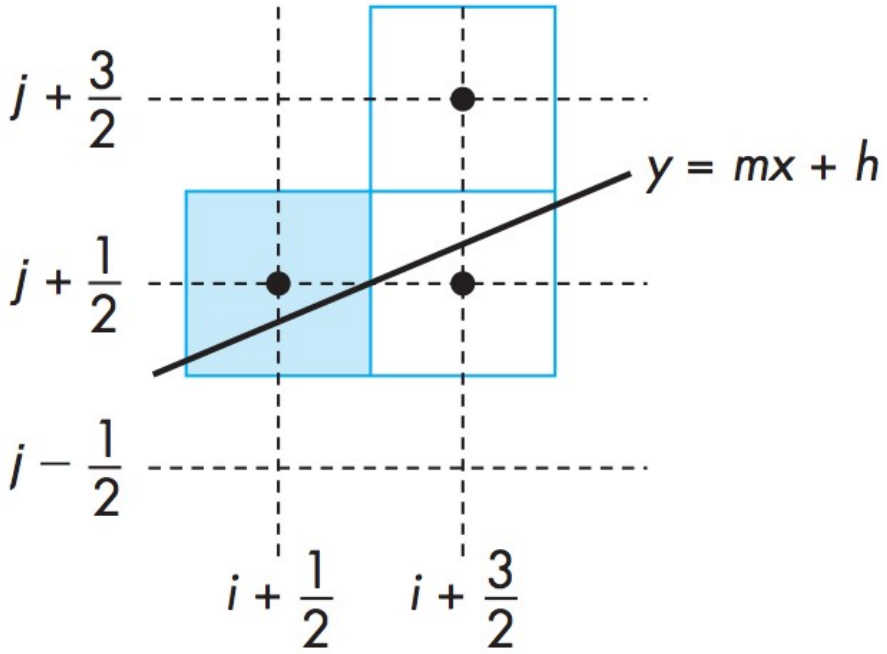


Como economizar?

Rasterização

Algoritmo de Bresenham

Algoritmo incremental que utiliza apenas soma e subtração de inteiros.



Rasterização

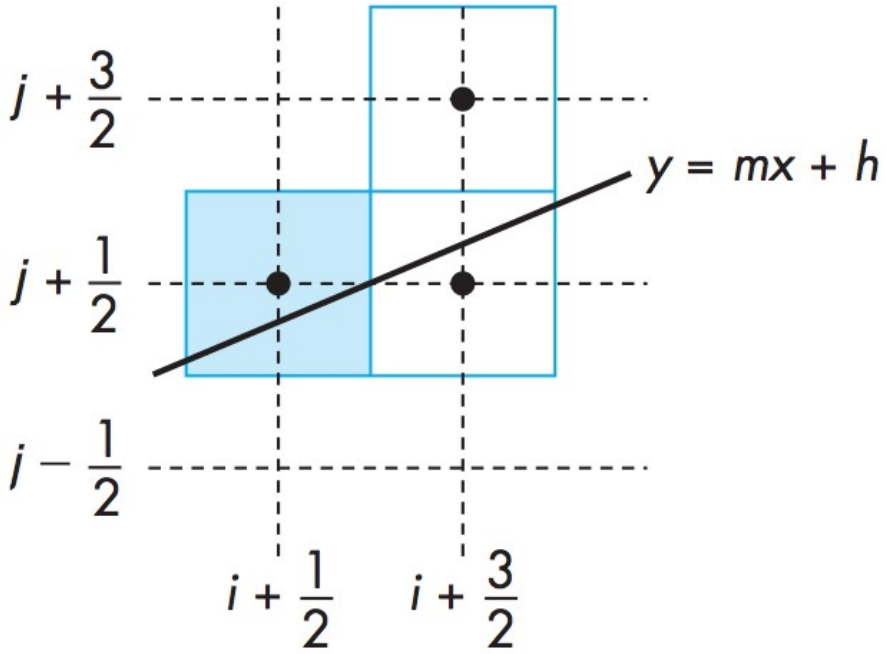
Algoritmo de Bresenham

Algoritmo incremental que utiliza apenas soma e subtração de inteiros.

Idéia

Decidir se o próximo pixel vai ter coordenadas

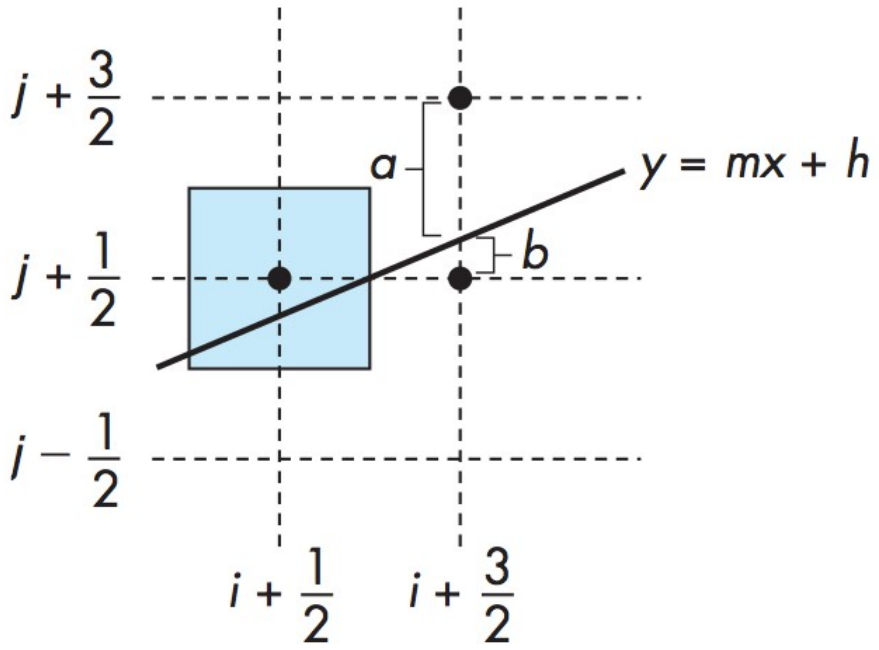
$(x + \frac{3}{2}, y + \frac{1}{2})$ ou $(x + \frac{3}{2}, y + \frac{3}{2})$



Rasterização

Algoritmo de Bresenham

Algoritmo incremental que utiliza apenas soma e subtração de inteiros.



Idéia

Decidir se o próximo pixel vai ter coordenadas

$$(x + \frac{3}{2}, y + \frac{1}{2}) \text{ ou } (x + \frac{3}{2}, y + \frac{3}{2})$$

Baseado na variável de decisão

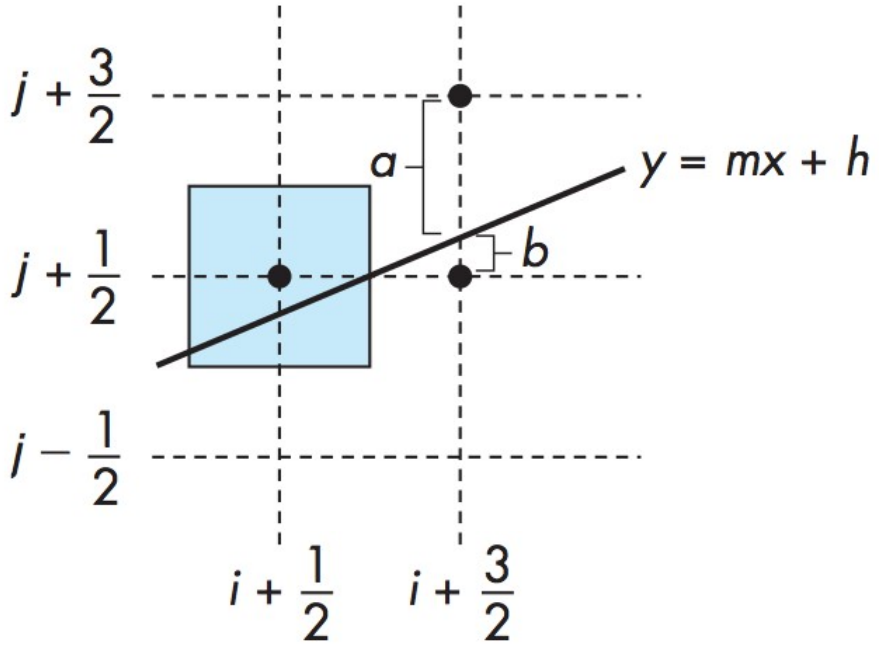
$$d = a - b$$

Rasterização

Algoritmo de Bresenham

Se a variável é positiva:

$$(x + \frac{3}{2}, y + \frac{1}{2})$$



Rasterização

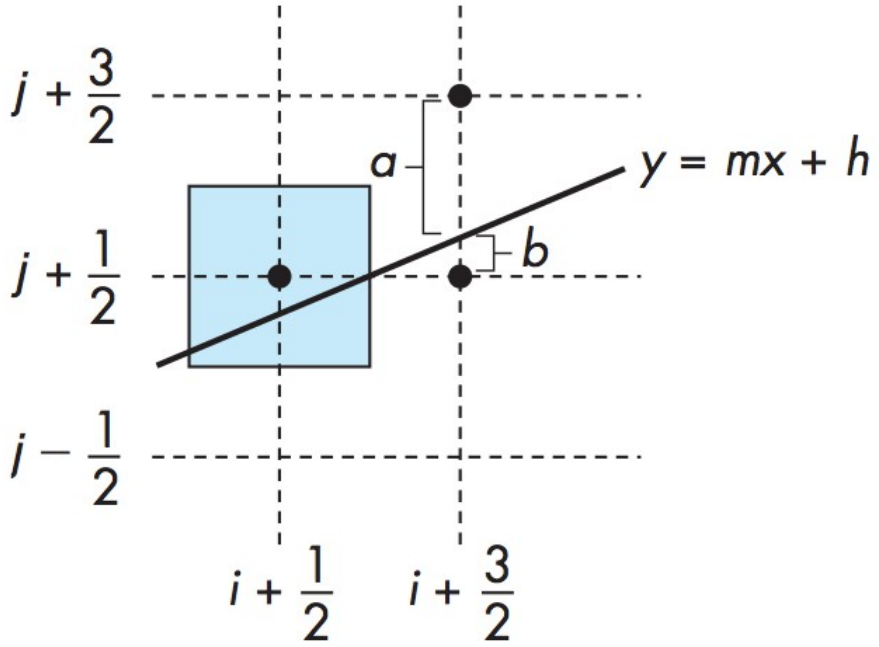
Algoritmo de Bresenham

Se a variável é positiva:

$$(x + \frac{3}{2}, y + \frac{1}{2})$$

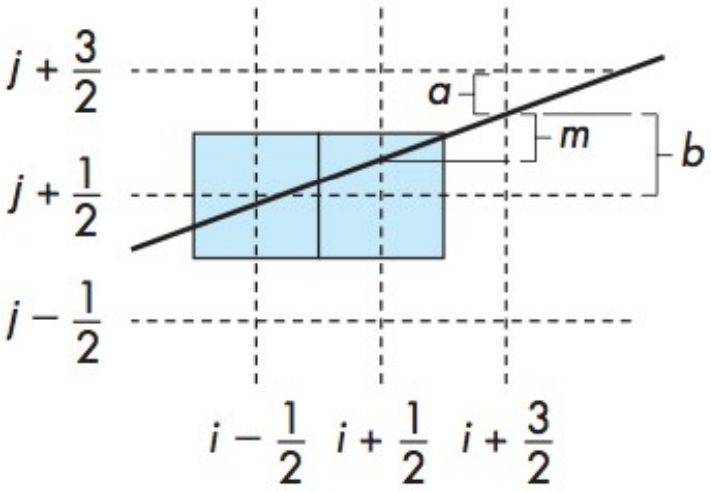
Se a variável é negativa:

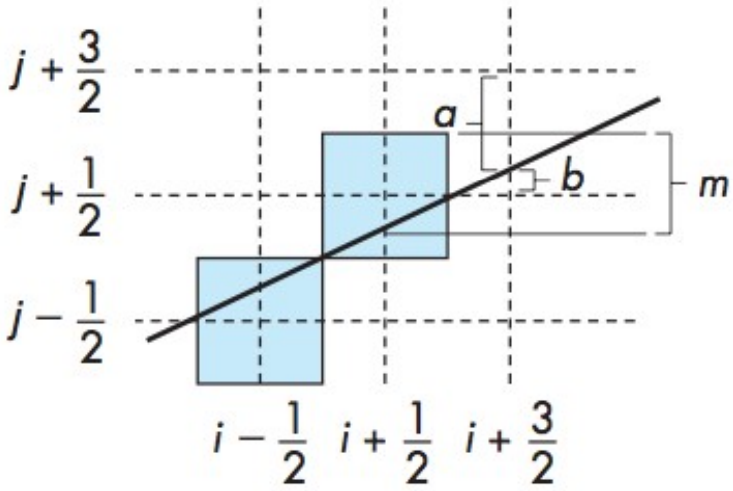
$$(x + \frac{3}{2}, y + \frac{3}{2})$$



Rasterização

Algoritmo de Bresenham





Calculando incrementalmente…

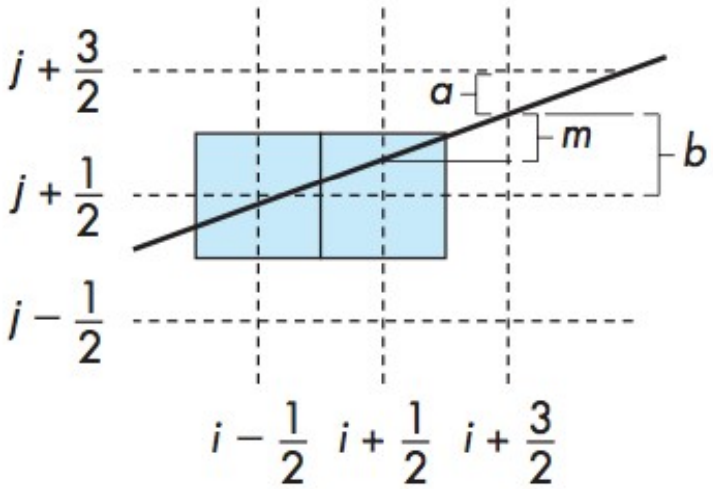
Se $d_k > 0$ →

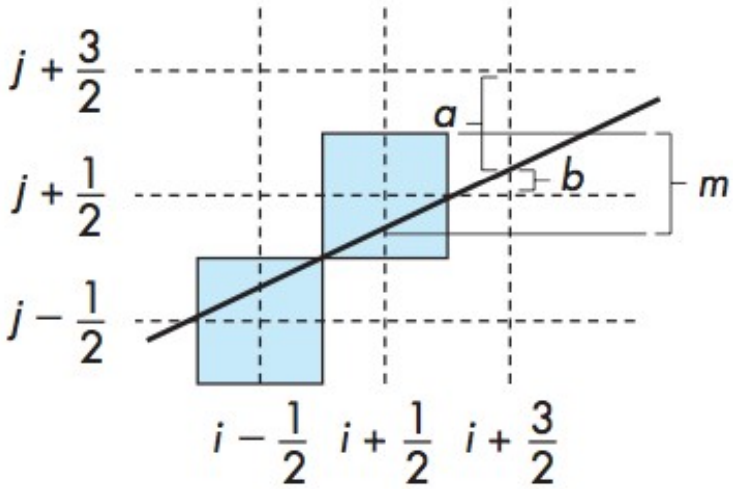
$$a_{k+1} = a_k - m$$

$$b_{k+1} = b_k + m$$

Rasterização

Algoritmo de Bresenham





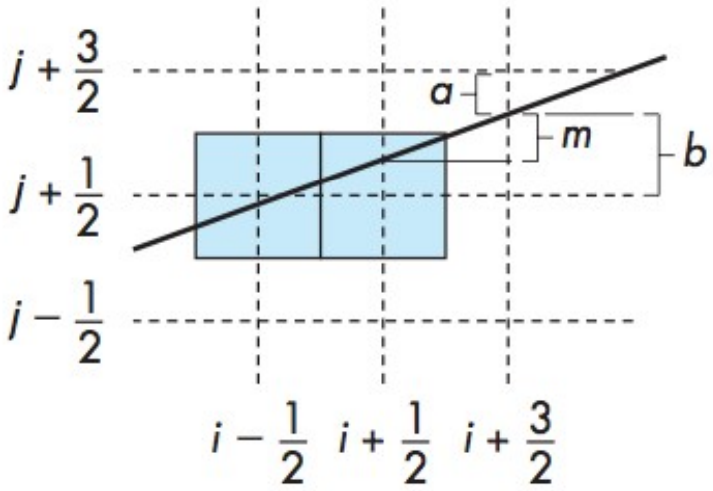
Calculando incrementalmente…

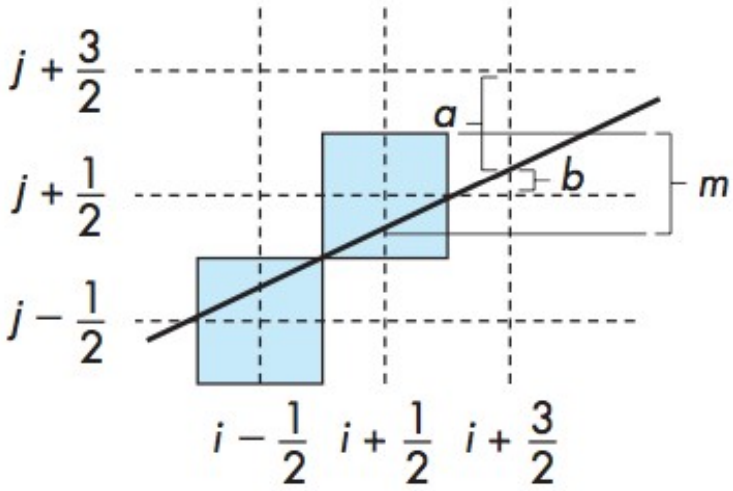
Se $d_k \leq 0$ →

$$a_{k+1} = a_k + (1 - m)$$
$$b_{k+1} = b_k - (1 - m)$$

Rasterização

Algoritmo de Bresenham





Calculando incrementalmente…

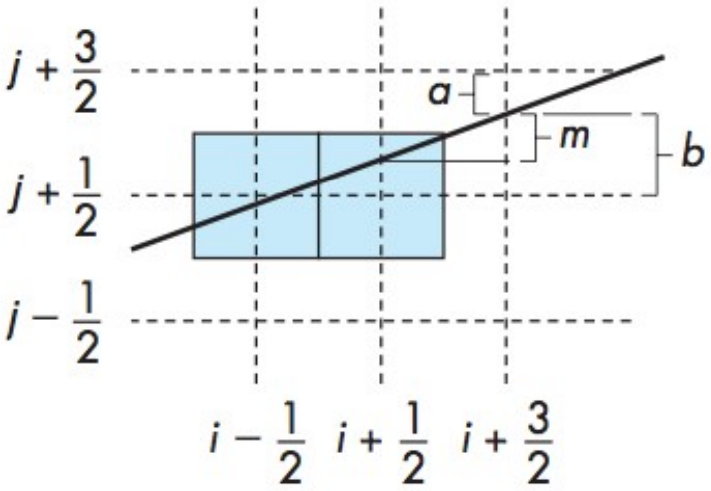
Lembrando →

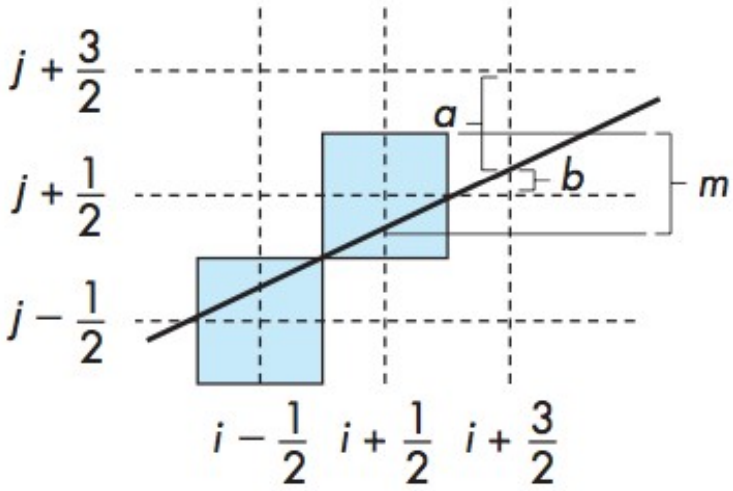
$$\Delta x = 1$$

$$m = \frac{\Delta y}{\Delta x}$$

Rasterização

Algoritmo de Bresenham





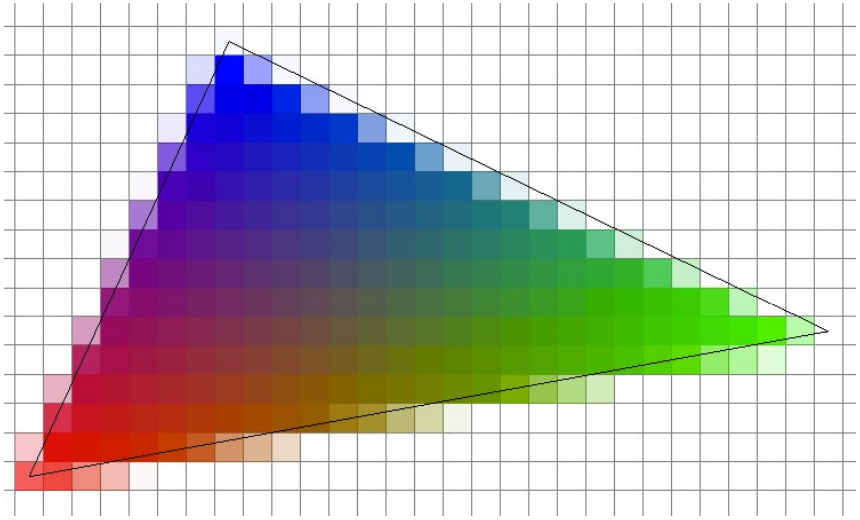
Calculando incrementalmente…

$$d_{k+1} = d_k - \begin{cases} 2\Delta y & \text{if } d_k > 0; \\ 2(\Delta y - \Delta x) & \text{otherwise.} \end{cases}$$

Rasterização

Polígonos

O processo de preenchimento de um polígono com um padrão de cores é equivalente à decidir quais pontos do plano estão em seu interior.

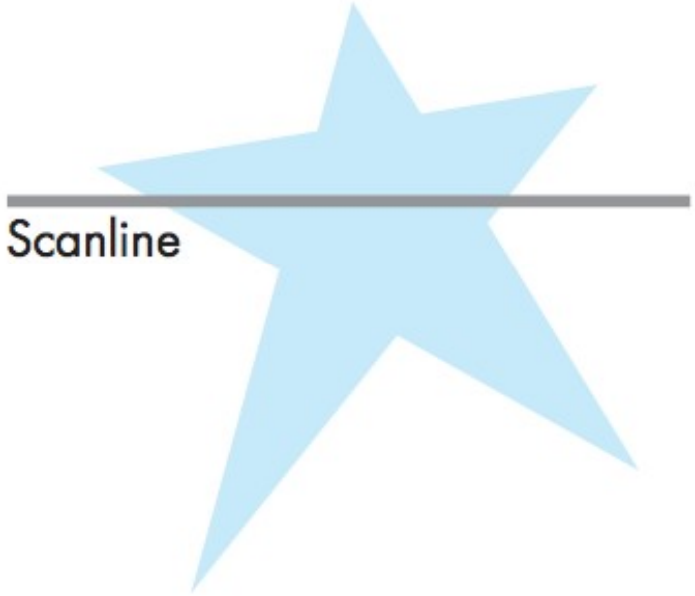


Rasterização

Polígonos

Teste par ou ímpar.

Todo raio saindo de **p** para o infinito deve cortar um número ímpar de segmentos do polígono, se o ponto for de interior e um número par se o ponto for de exterior.

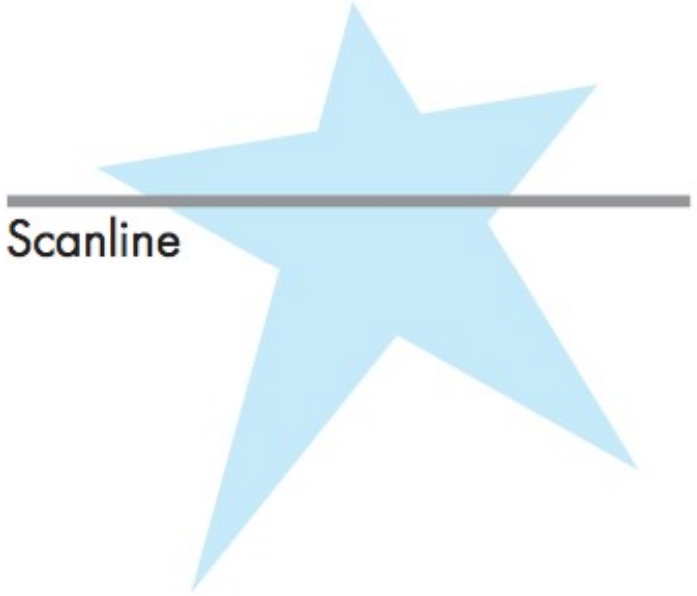


Rasterização

Polígonos

Teste par ou ímpar.

Usualmente, trabalhamos com scanlines para definir os segmentos e o ponto **p**.

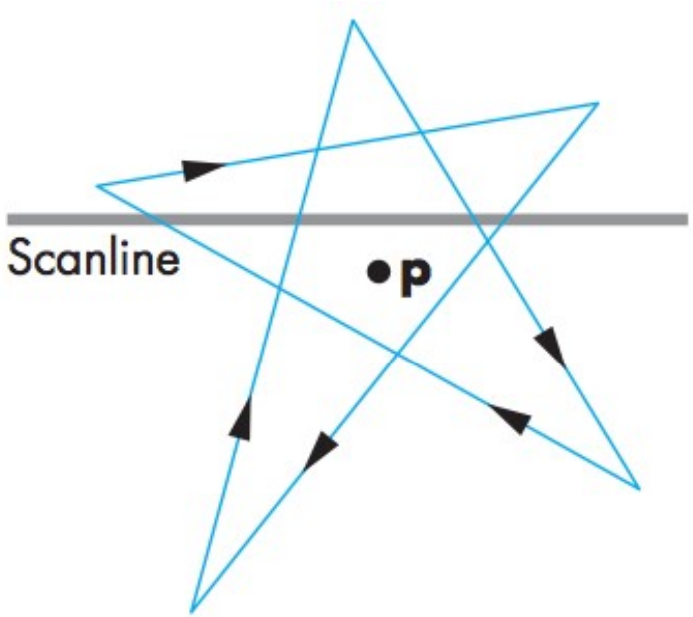


Rasterização

Polígonos

Teste par ou ímpar.

O algoritmo funciona para polígonos simples, como vemos no exemplo abaixo:

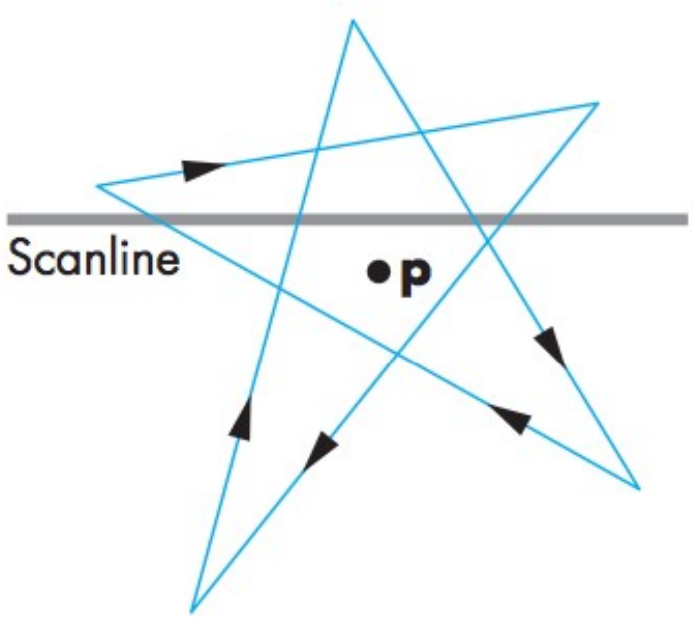


Rasterização

Polígonos

Teste winding number.

Supõe que as arestas do polígono são percorridas em uma direção fixa, a partir de um ponto inicial.

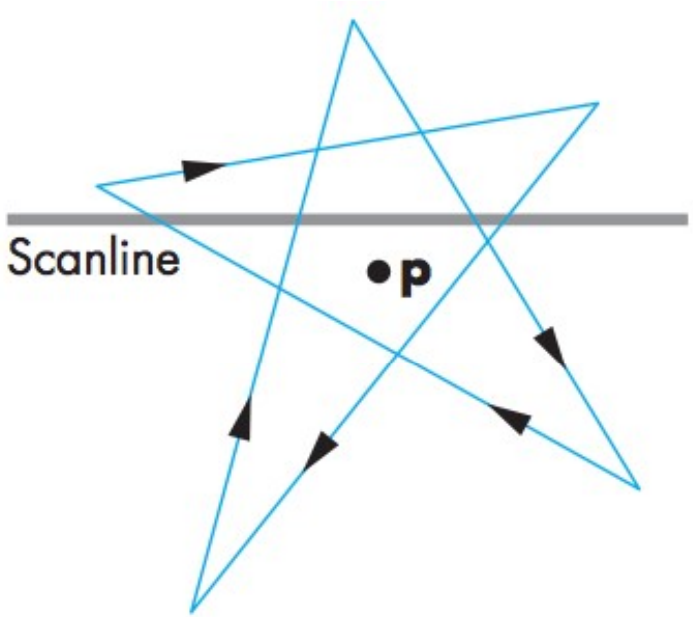


Rasterização

Polígonos

Teste winding number.

O winding number é o número de vezes que o ponto é envolvido por um ciclo de segmentos.

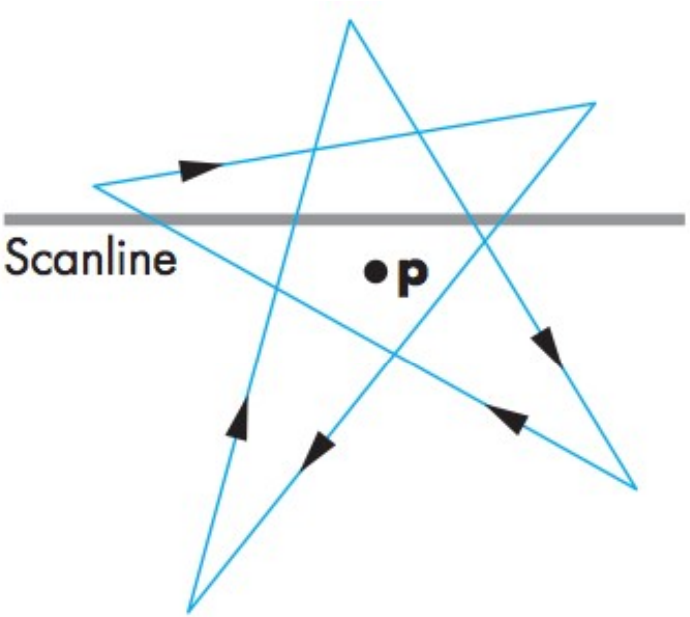


Rasterização

Polígonos

Teste winding number.

Se o ponto tiver winding number igual a zero, então é exterior.

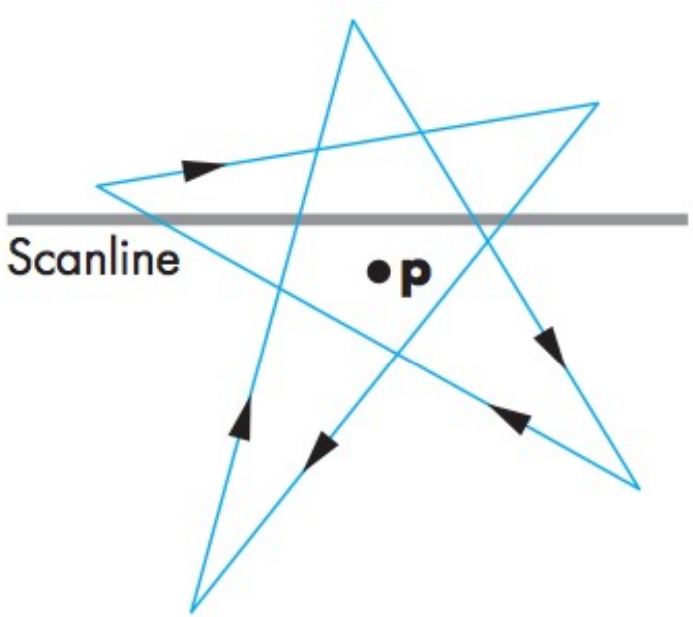


Rasterização

Polígonos

Teste winding number.

Exercício: Calcule o winding number de **p**.



Computação Gráfica

TCC-00291

Assunto: Rasterização