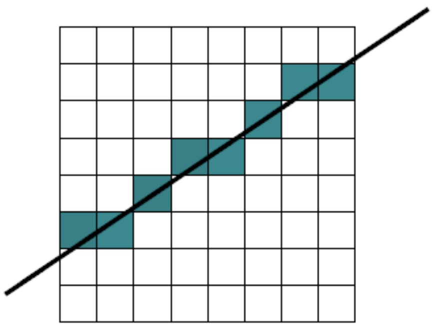
Rasterização

Gilda Aparecida de Assis
gildaaa1@gmail.com

Rasterização

- Gráficos são definidos através de primitivas geométricas como pontos,segmentos de retas, polígonos, etc
    - Representação vetorial
- Dispositivos gráficos são como matrizes de pixels (rasters)
    - Representação matricial
- Rasterização é o processo de conversão entre representações vetorial e matricial



Rasterização

- Dispositivos raster predominam no mercado

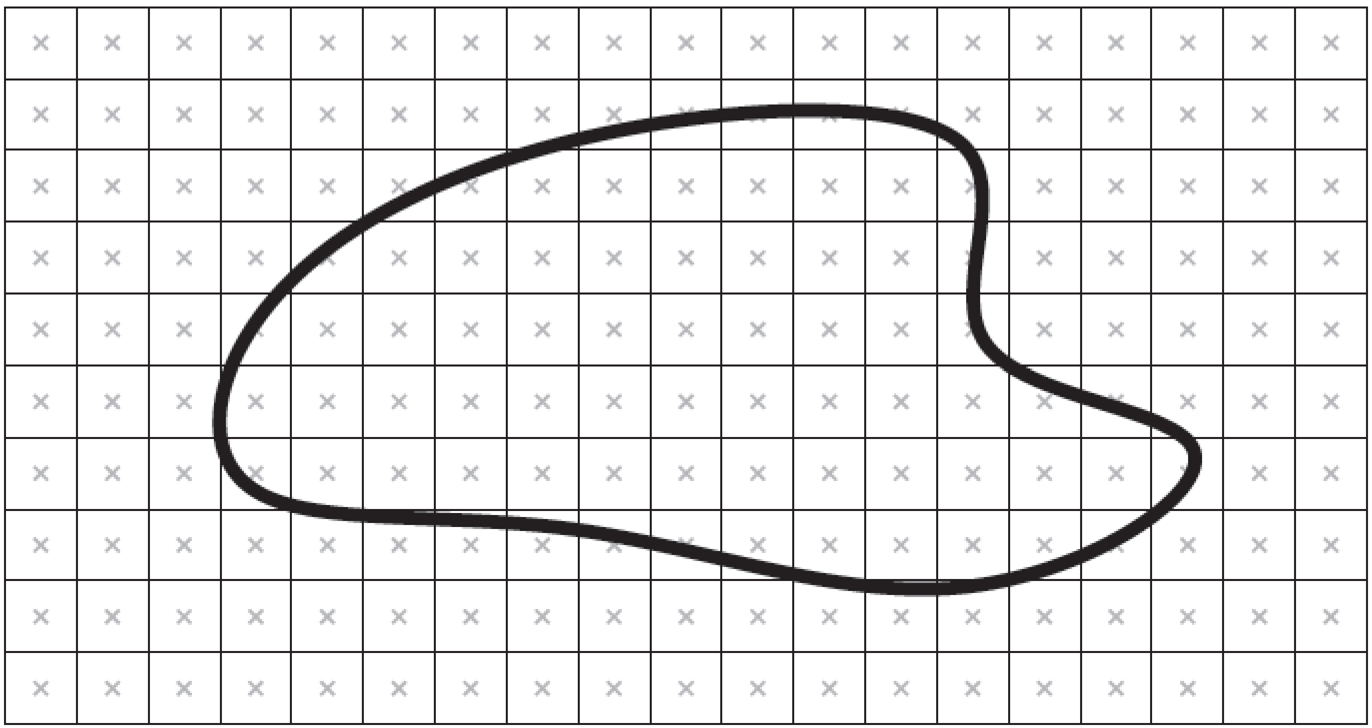
LCD

CRT

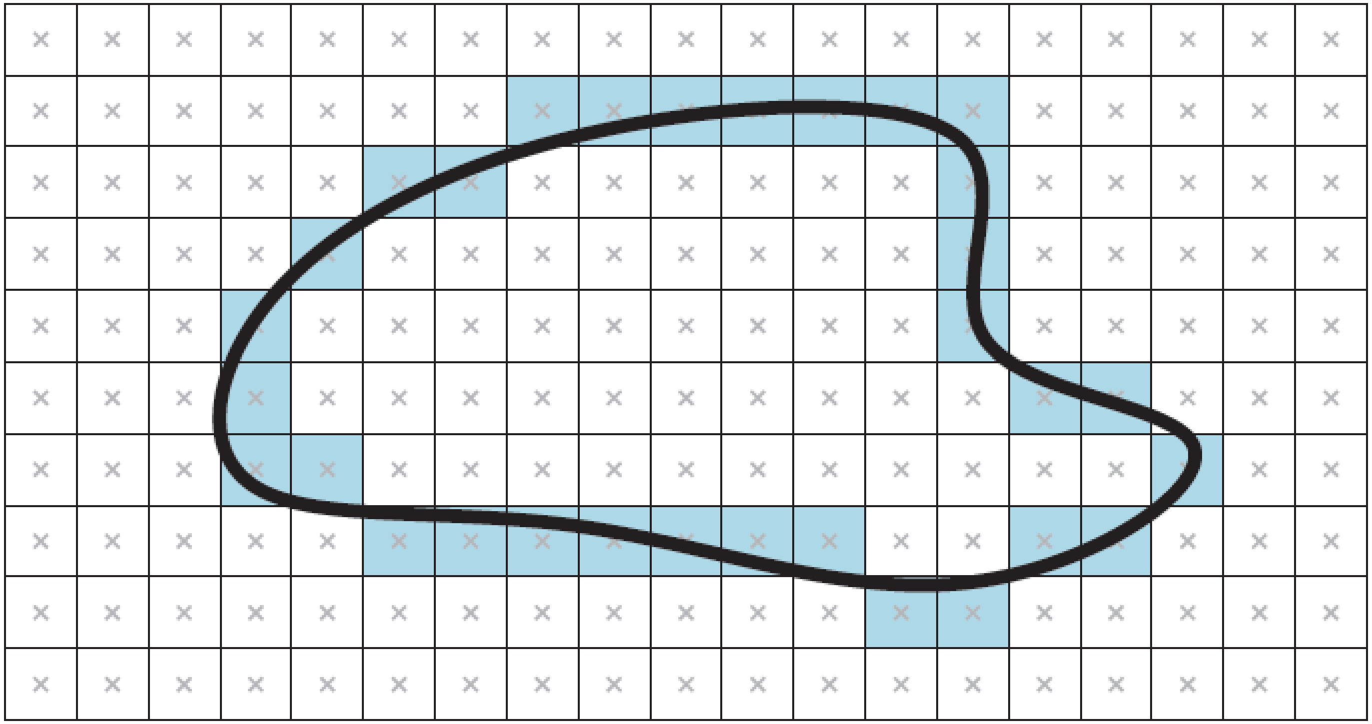
Problema da Rasterização

- Entrada:
  - Primitivas 2D no espaço
- Saída
  - Coleção de fragmentos de pixels a serem pintados junto com os atributos (cor, profundidade, etc.) interpolados para cada pixel

Exemplo de Rasterização 1/2

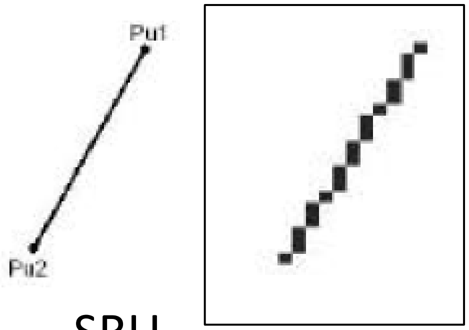


Exemplo de Rasterização 2/2



Exemplo de Rasterização

- Rasterização: converter a geometria para pixels (escolher os pixels)
- Último passo da pipeline de visualização



SRU

Raster na tela

- http://www.alexandre.eletrica.ufu.br/cg/primitivas.html

Rasterização

- Rasterização é um processo de amostragem
  - Domínio contínuo → discreto
  - Problemas de aliasing são esperados

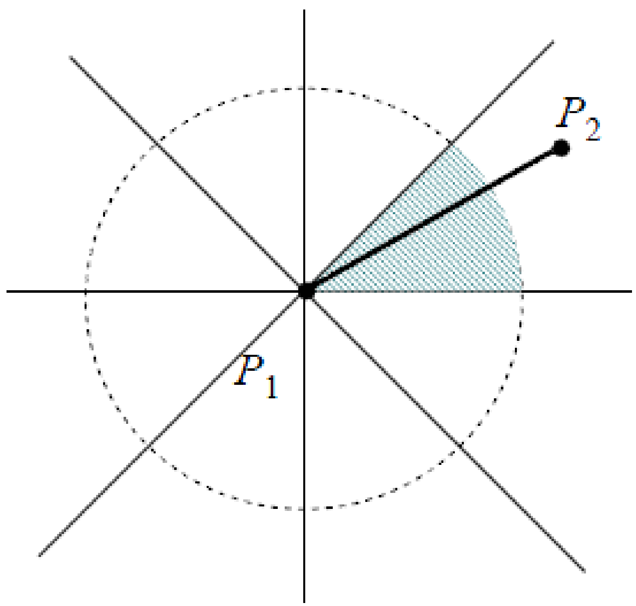
- Cada primitiva pode gerar um grande número de pixels
  - Rapidez é essencial
  - Em geral, rasterização é feita por hardware

Rasterização de linhas

- Segmento de reta entre P0= (x0, y0) e P1= (x1, y1)
- Objetivo é pintar os pixels atravessados pelo segmento de reta
- Na verdade, nem todos os pixels, apenas os mais próximos
- Segmentos com largura igual a 1.

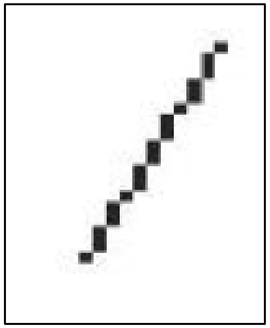
Rasterização de linhas

- Considerar segmentos de reta no primeiro octante, com os demais casos resolvidos de forma simétrica



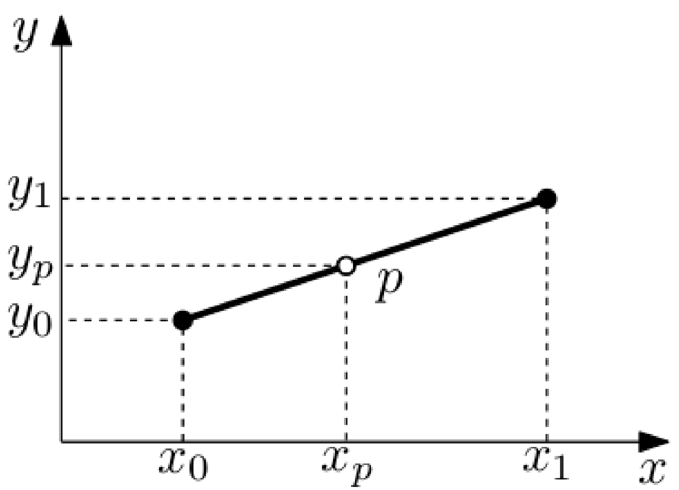
Rasterização de linhas

- Algoritmo Simples
- Algoritmo DDA -Digital Differential Analyzer
- Algoritmo de Bresenham



Algoritmo Simples

$m := (y_1 - y_0)/(x_1 - x_0)$
para $x := x_0$ **até** x_1, **faça**:
    $y := y_0 + m \times (x - x_0)$
    Pinte pixel $(x, \lfloor y + 0.5 \rfloor)$
fim para



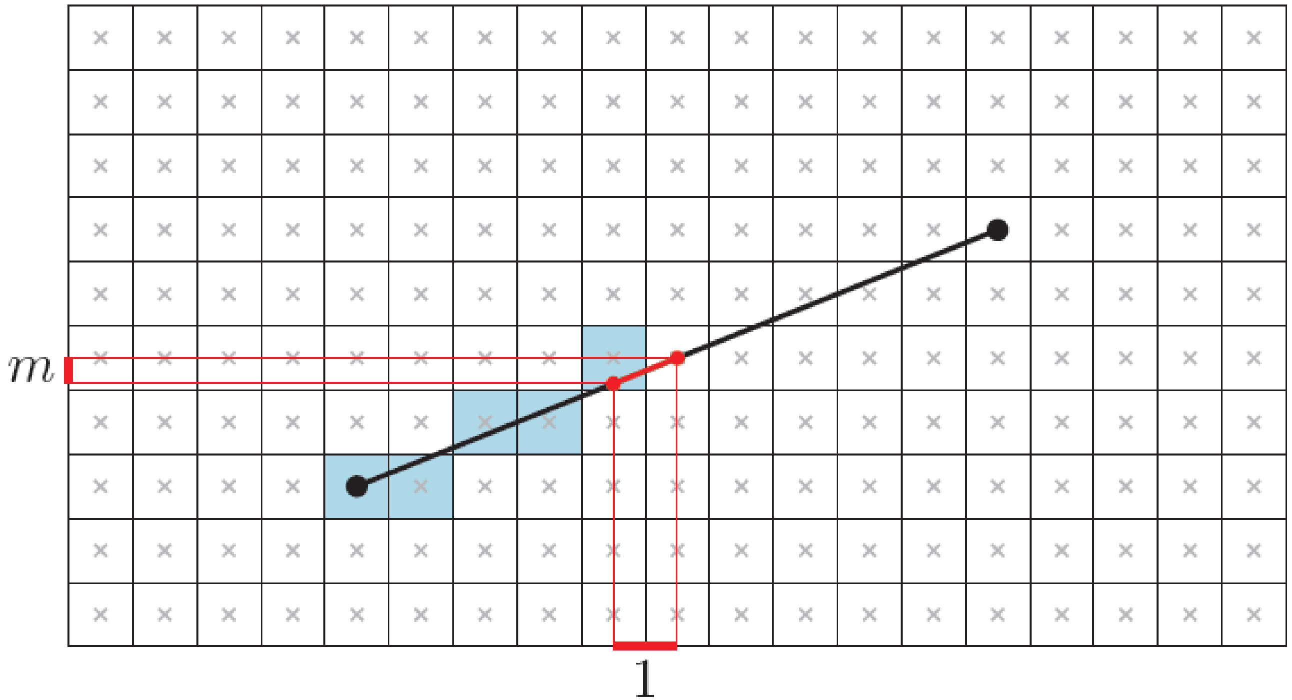
$$\frac{y_p - y_0}{x_p - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

Algoritmo Simples

- Aritmética de ponto flutuante
- Sujeito a erros de arredondamento
- Utiliza multiplicação
- Lento

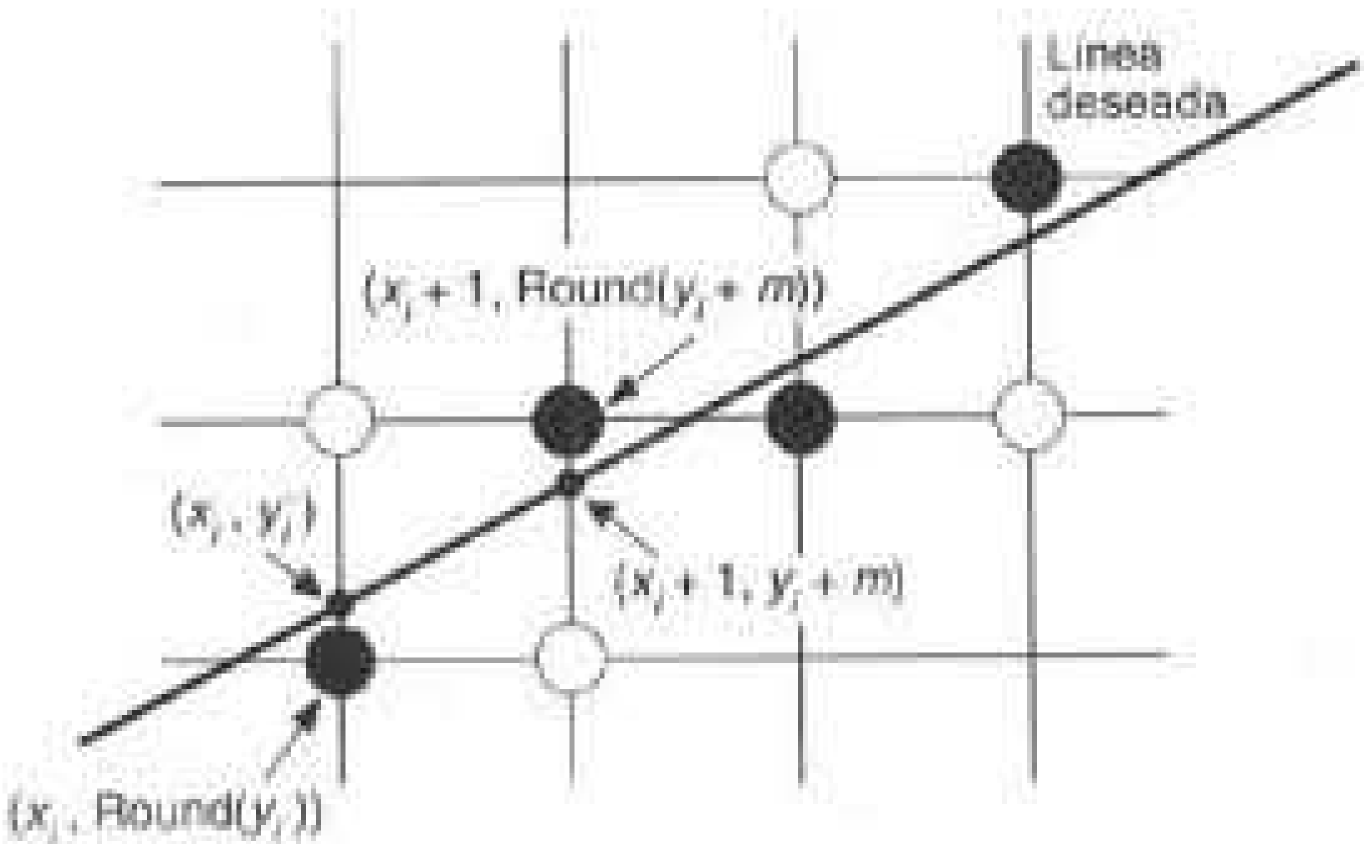
Algoritmo DDA -Digital Differential Analyzer

- Reduzir a aritmética de ponto flutuante
- Se observarmos que m é a variação em y para um incremento unitário de x, podemos fazer melhor



Algoritmo DDA -Digital Differential Analyzer

- Evita a multiplicação com ponto flutuante, mas continua com operações de adição e arredondamento



Algoritmo DDA -Digital Differential Analyzer

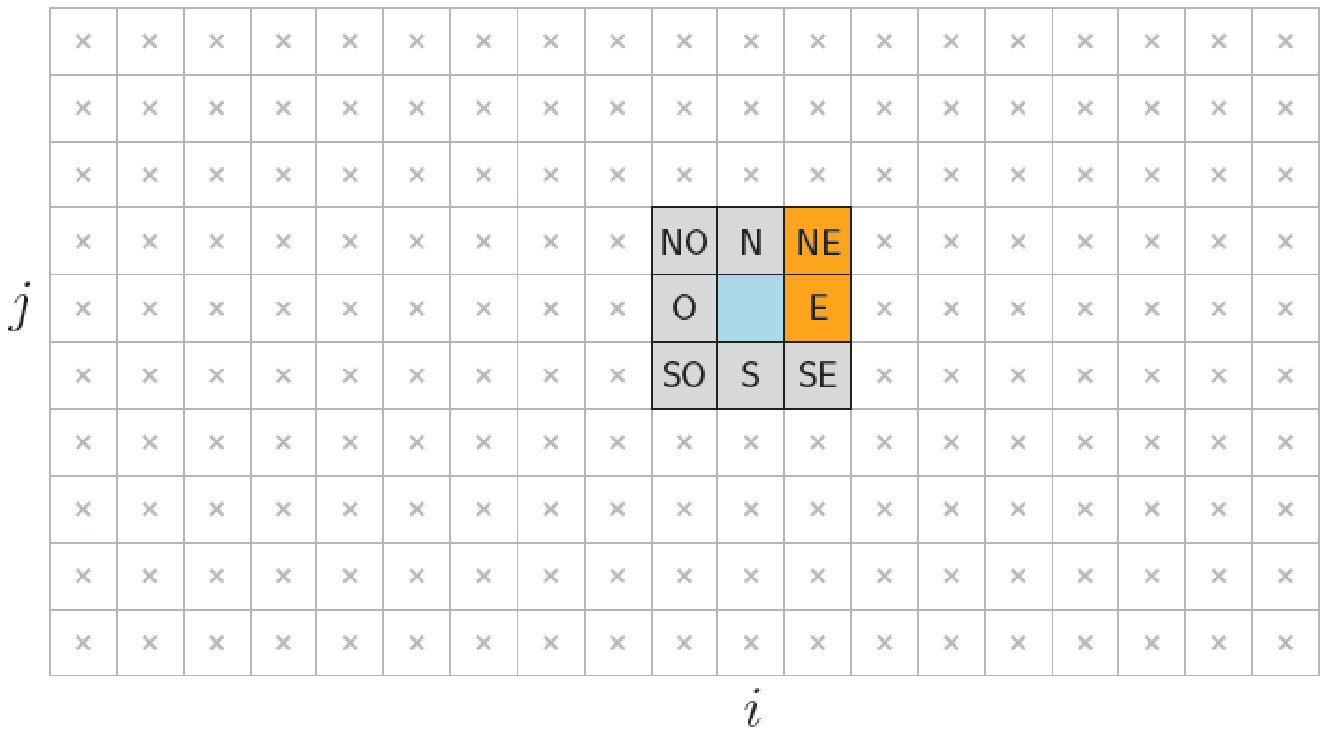
```
procedure Line (x0, y0, x1, y1, value);
int d;
float dy, dx, y, m;
{
    dy = y1-y0;
    dx = x1-x0;
    m = dy/dx;
    y = y0;
    for (x=x0 ; x<x1;  x++) {
         writepixel (x, round (y), value);
          y = y + m;     }
}
```

Algoritmo Bresenham

- Algoritmo clássico da computação gráfica
- Algoritmo incremental que utiliza apenas soma e subtração de inteiros
- Algoritmo padrão implementado em hardware
- Melhor aproximação para linhas e círculos, minimiza o erro (distância) para o objeto real

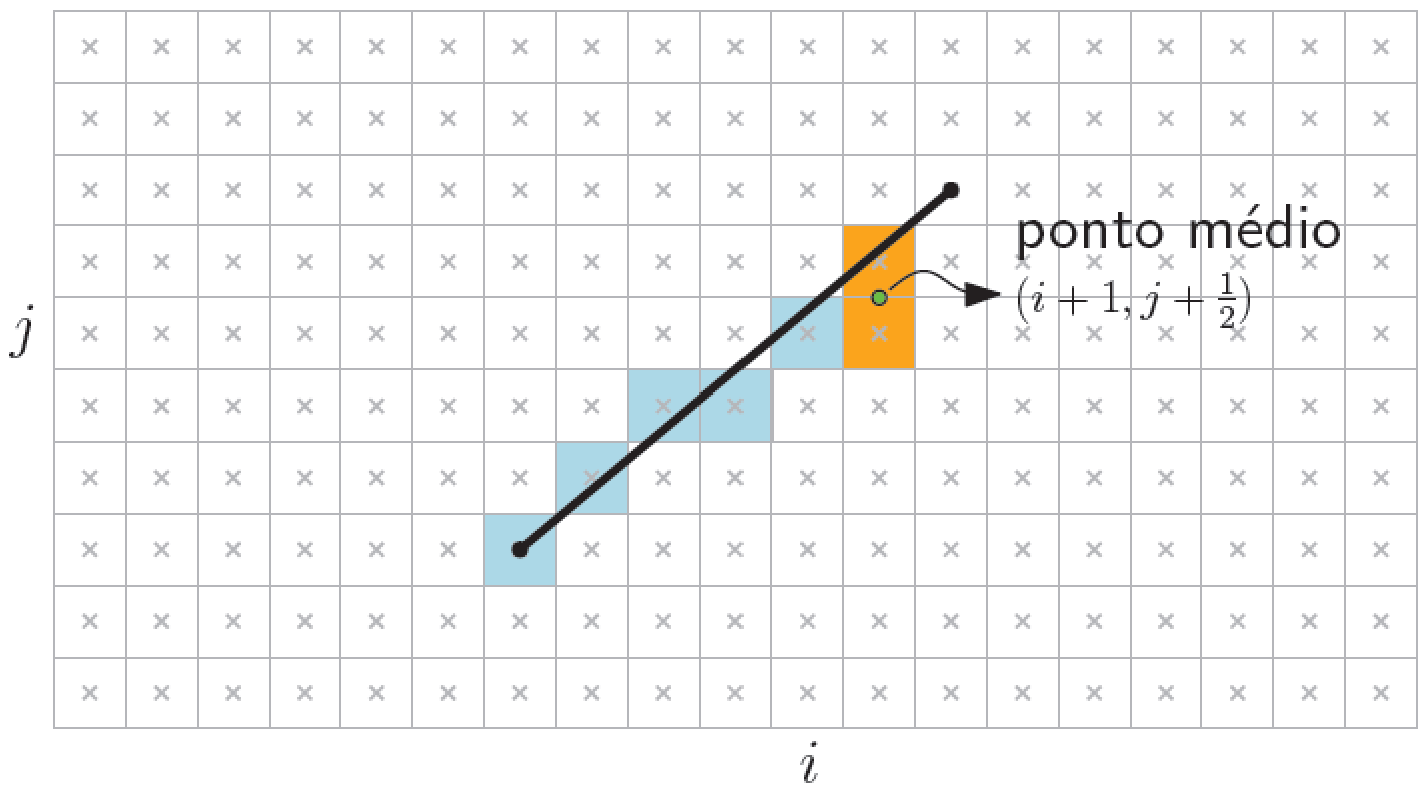
Algoritmo Bresenham

- Sempre que fazemos um incremento em x temos que fazer a escolha entre dois candidatos, ou seja, se um pixel (i, j) é pintado, então o próximo pixel é o E(i, j) ou o NE(i, j)
  - Segmentos de reta no 1º quadrante com m>0

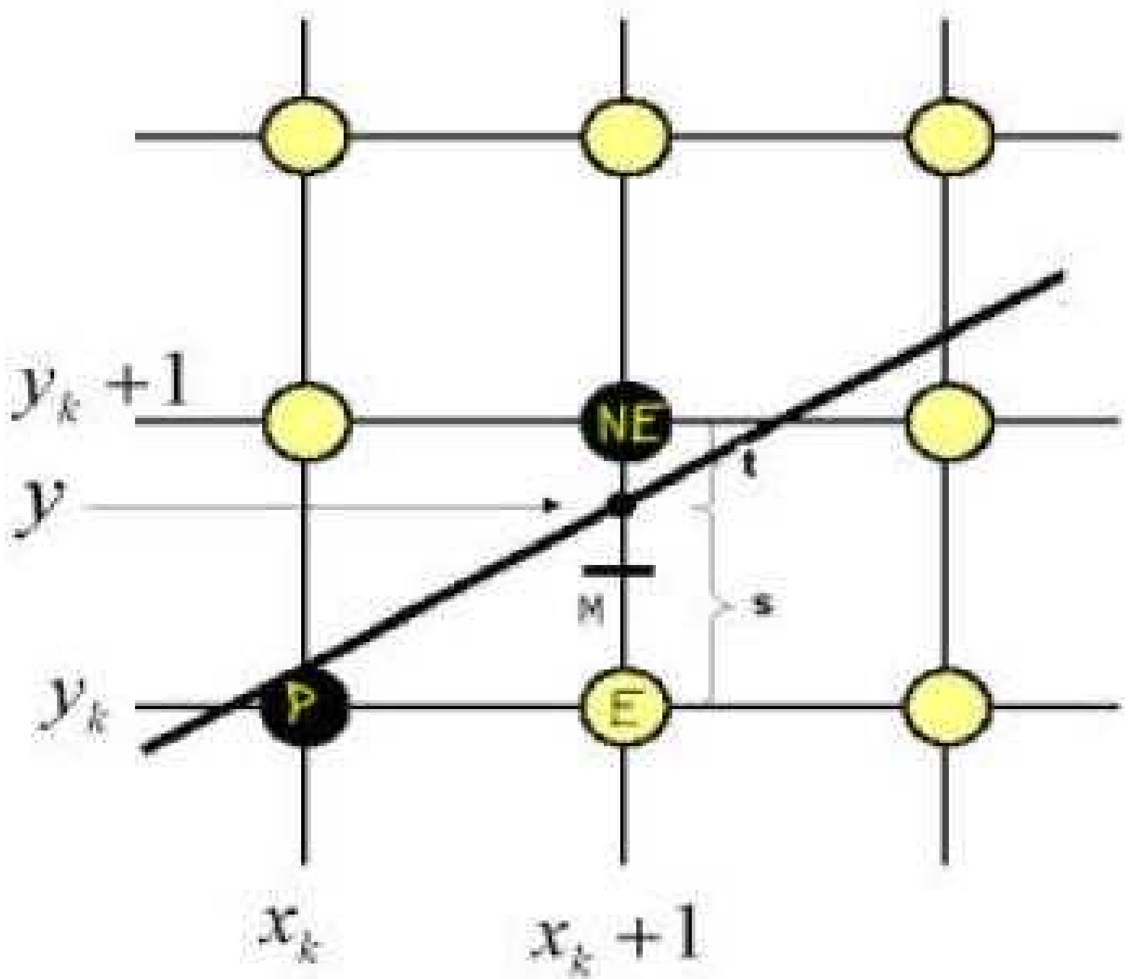


Algoritmo Bresenham

- Podemos usar a posição da reta em relação ao ponto médio entre os candidatos (i+1, j) e (i+1, j+1) para fazer a escolha entre E e NE

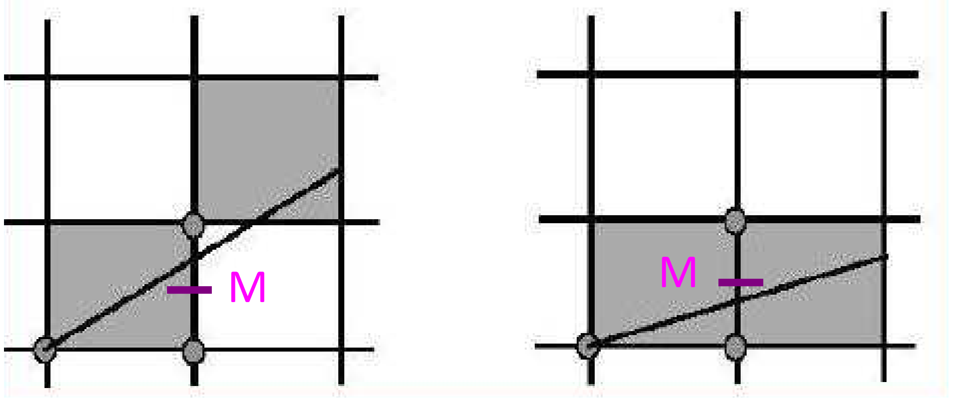


Algoritmo Bresenham



Algoritmo Bresenham

- Aritmética de números inteiros
- Escolha entre dois pixels vizinhos (coerência espacial): (x+1, y) ou (x+1, y+1)
- Escolha baseada na avaliação se a linha passa acima ou abaixo do ponto médio (x+1, y+0.5)



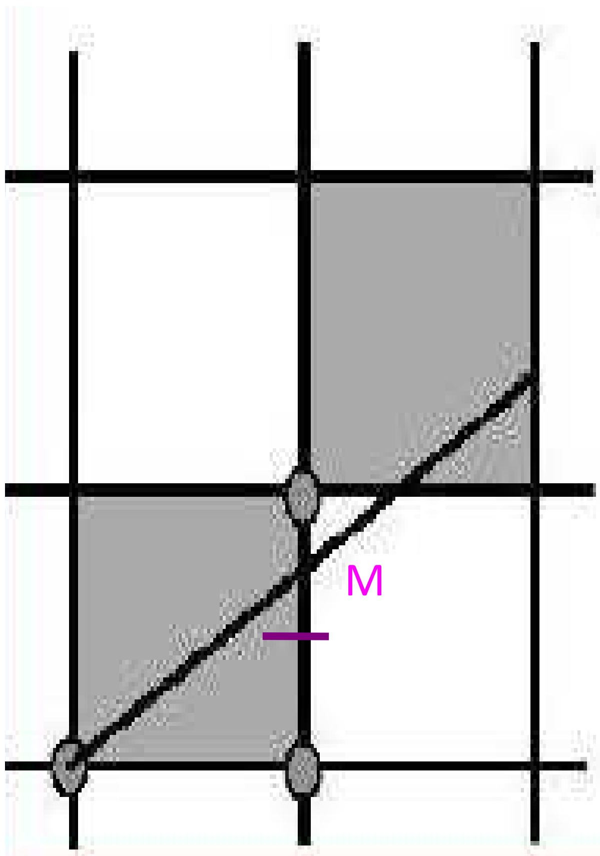
Algoritmo Bresenham

- Equação da reta na forma implícita:
- $F(x,y) = ax + by + c = 0$
- Se $dy = (y1 - y0)$ e $dx = (x1- x0)$, a forma explícita da linha é $y = (dy/dx).x + B$, ou seja, $(dy/dx).x + B – y = 0$. Se multiplicar por (dx) fica $dy.x -dx.y + dx.B = 0$. B é obtido substituindo (x0, y0) em (x,y) na quação. Fazendo a correspondência em F(x, y) fica:
  - $F(x,y) = dy.x – dx.y + B.dx = 0$, onde $a = dy$, $b = -dx$, $c = B.dx$
  - $F(x,y) == 0$, se (x,y) pertence à linha
  - $F(x,y) < 0$, se (x,y) está acima da linha
  - $F(x,y) > 0$, se (x,y) está abaixo da linha

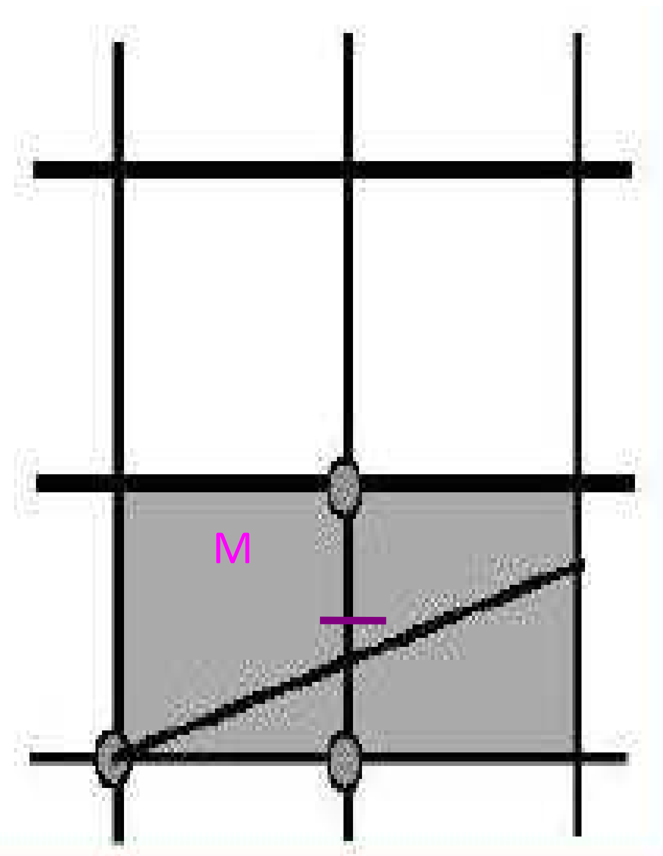
    - Calcula $d = F(M)$ (variável de decisão), onde M é o ponto médio e decide com base no sinal de d

Algoritmo Bresenham

$d > 0$



$d < 0$



Algoritmo Bresenham

- A variável de decisão no próximo passo (x_k+2) também pode ser calculada de forma incremental
- Se pixel escolhido foi E (leste) então
    - $d_{novo} = d_{velho} + a$, onde $a = dy$
- Se pixel escolhido foi NE (nordeste) então
    - $d_{novo} = d_{velho} + a + b$, onde $a = dy$, $b = -dx$

Algoritmo Bresenham - Resumo

A cada passo, o algoritmo escolhe entre dois pixels baseado no sinal da variável de decisão da iteração anterior.

Então, ele atualiza a variável de decisão adicionando $delta_E$ ou $delta_{NE}$ ao valor antigo, dependendo da escolha do pixel.

O primeiro ponto médio é $F(x_0+1, y_0+0.5)$, com $d= F(x,y) =$ $F(x,y) = dy.x – dx.y + B.dx$, substituindo o ponto médio fica $d_{inicial} = a+(b/2)$, onde $a = dy$, $b = -dx$

Para eliminar a fração em $d_{inicial}$ a função $F(x,y)$ é redefinida como $F(x,y) = 2(ax + by + c)$ e $d_{inicial} = 2a+b$

Algoritmo Bresenham - Resumo

Se d = F(M) > 0, então M está acima da linha, escolhe NE

Se d = F(M) < 0, então M está abaixo da linha, escolhe E

Se pixel escolhido foi E (leste) então

- $d_{novo} = d_{velho} + 2*a$, onde $a = dy$

Se pixel escolhido foi NE (nordeste) então

- $d_{novo} = d_{velho} + 2*(a + b)$, onde $a = dy$, $b = -dx$

Algoritmo Bresenham

Exercício

Definir os pixels para o segmento de reta , P0 = (20,10) e P1 = (30,18)

$\Delta x = 10$, $\Delta y = 8$, $d_{inicial} = 2\Delta y - \Delta x = 6$

$a = \Delta y = 8$, $b = -\Delta x = -10$

$a + b = -2$

Algoritmo Bresenham

Exercício

k	d	(x,y)	k	d	(x,y)
0	6	(21,11)	5	6	(26,15)
1	2	(22,12)	6	2	(27,16)
2	-2	(23,12)	7	-2	(28,16)
3	14	(24,13)	8	14	(29,17)
4	10	(25,14)	9	10	(30,18)