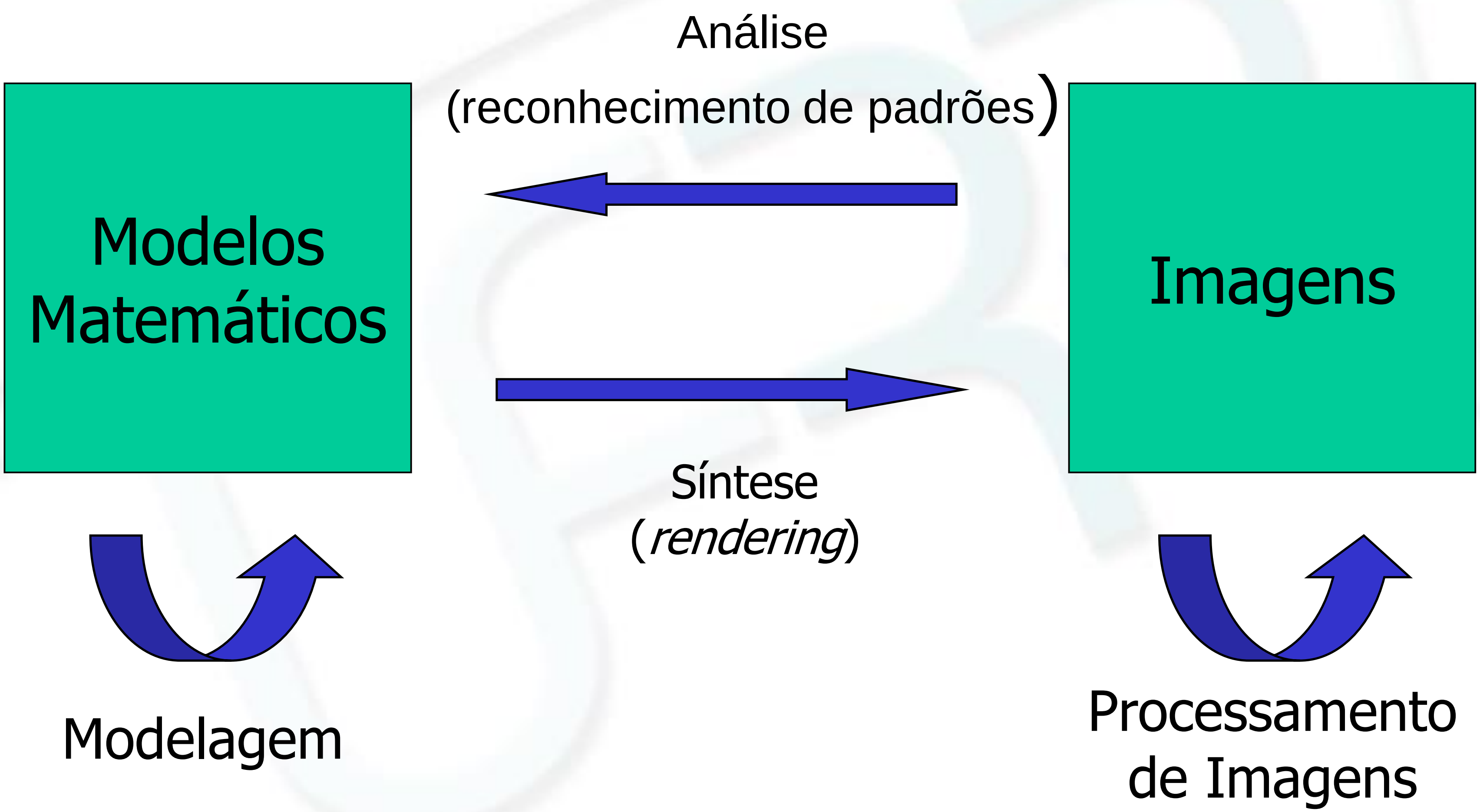
Computação Gráfica

Rasterização de Linhas

Professor: Luciano Ferreira Silva, Dr.

Aula 6

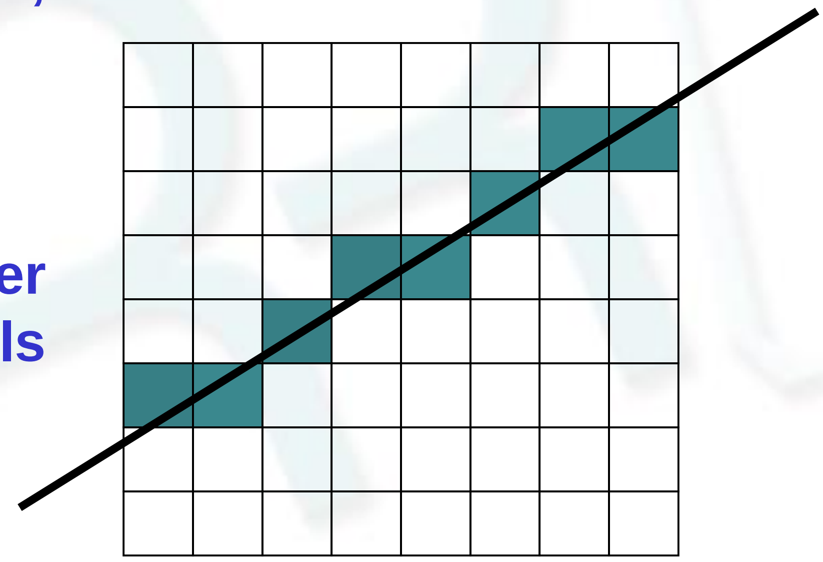
Definições


Análise
(reconhecimento de padrões)
Modelos Matemáticos
Imagens
Síntese
(*rendering*)
Modelagem
Processamento de Imagens

Pipeline de Visualização

Representação Vetorial x Matricial

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



Considerações Gerais

- **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**
- **Técnicas de *anti-aliasing* podem ser empregadas, usualmente extraindo um custo em termos de desempenho**

Geração de Primitivas

- **Geração de linhas:**

  - ✓ Utilização de dispositivos matriciais, com pontos discretos;
  - ✓ Processo: rasterização;
  - ✓ Não há problemas para linhas horizontais, verticais ou com inclinação de 45º;
  - ✓ Problema básico: qual o pixel ideal para uma linha desejada (?);
  - ✓ Soluções melhores para dispositivos com maior resolução.

Rasterização

- **Linhas (segmentos de reta):**
  - ✓Em CG linhas são representadas por extremos

$$P_1\ (x_1, y_1) \quad \leftarrow\rightarrow \quad P_2\ (x_2, y_2)$$

  - ✓Métodos/Algoritmos para rasterização:
    - **Analítico**
    - **DDA (Analisador Diferencial Digital)**
    - **Bresenham**

Método Analítico

- **Método mais simples e intuitivo**
- **Dados os extremos** $P_1(x_1, y_1)$ $P_2(x_2, y_2)$ **de uma linha:**

1. Descubra a equação da reta ($y = m\,x + b$);
   - $\boldsymbol{m = (y_2 - y_1) / (x_2 - x_1)}$
   - $\boldsymbol{b = y_1 - m.x_1}$
2. Varie x de $x1$ a $x2$ de 01 em 01 unidade;
3. Obtenha o y discreto correspondente por meio de arredondamento

Método Analítico

- *Dados $P_1(x_1, y_1)$ $P_2(x_2, y_2)$ temos o* algoritmo:

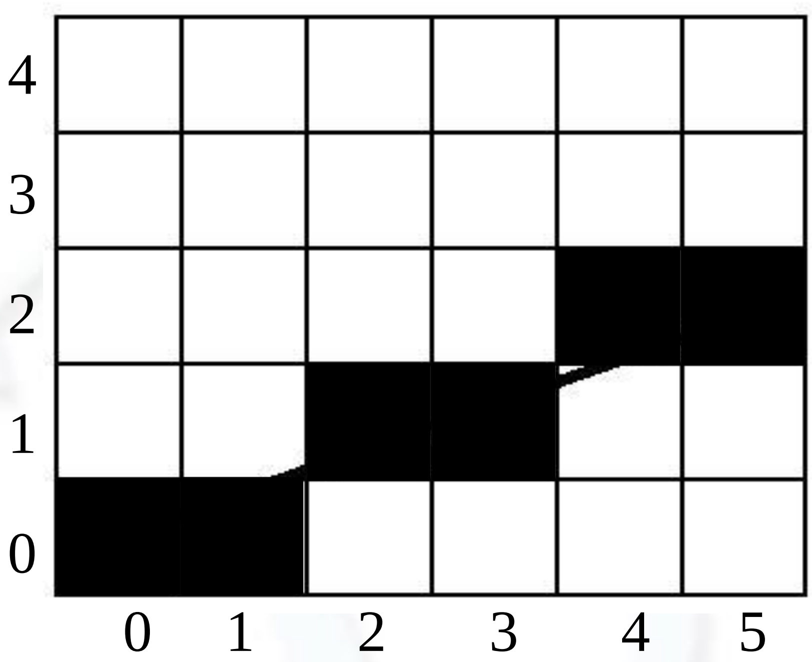
% reta vertical X1 = X2 Não	Sim

m = (Y2 - Y1)/(X2 - X1)
b = Y2 - m*X2
para X de X1 até X2
  Y = m*X + b
  liga_pixel(X, Y, Cor)
	para Y de Y1 até Y2
	  liga_pixel(X1, Y, Cor);

Método Analítico

- **Suponha a linha definida pelos pontos: $P_1(0,0)$ $P_2(5,2)$**

Pergunta Chave: Quais pixel são ligados?



Processando...
$m = (2 - 0)/(5 – 0) = 0.4$
$b = 0 – 0.4 * 0 = 0$

Processando... ($y = 0.4 * x$)

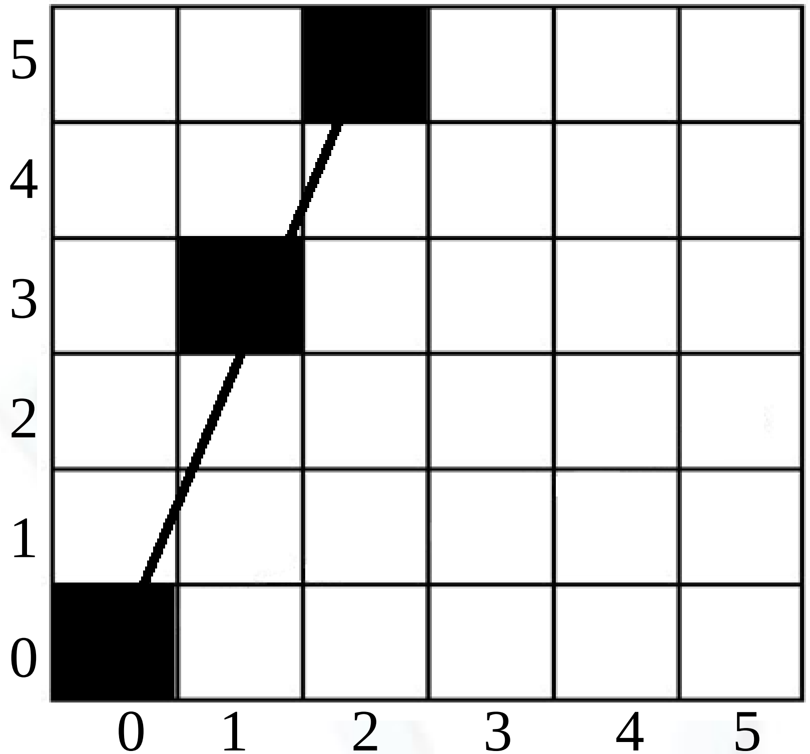
Para...

$x = 0 \rightarrow y = 0.0 \rightarrow$ Pixel (0, 0)
$x = 1 \rightarrow y = 0.4 \rightarrow$ Pixel (1, 0)
$x = 2 \rightarrow y = 0.8 \rightarrow$ Pixel (2, 1)
$x = 3 \rightarrow y = 1.2 \rightarrow$ Pixel (3, 1)
$x = 4 \rightarrow y = 1.6 \rightarrow$ Pixel (4, 2)
$x = 5 \rightarrow y = 2.0 \rightarrow$ Pixel (5, 2)

Método Analítico

- **Problema grave: $P_1(0,0)$ $P_2(2,5)$**

Pergunta Chave: Quais pixel são ligados?



Processando...

$m = (5 - 0)/(2 – 0) = 2.5$

$b = 0 – 0.4 * 0 = 0$

Processando... $(y = 2.5 * x)$

Para...

$x = 0 \rightarrow y = 0.0 \rightarrow$ *Pixel* $(0, 0)$

$x = 1 \rightarrow y = 2.5 \rightarrow$ *Pixel* $(1, 3)$

$x = 2 \rightarrow y = 4.0 \rightarrow$ *Pixel* $(2, 4)$

Esse resultado é bom para você?

Método Analítico

- **Outros problemas:**

  ✓ Operações com ponto flutuante, no entanto, pixel são inteiros;

  ✓ Muitos cálculos no processo – eficiência computacional (?)

  ✓ Escolha do pixel não é um fator considerado na elaboração da solução;

  - Pode ser qualquer um das redondezas do número obtido nas contas efetuadas ;

Método DDA

- **DDA: Analisador Diferencial Digital;**
- **Melhora os resultados do analítico;**
- **Técnica baseada no cálculo de Δy e de Δx.**

$m = \Delta y / \Delta x \rightarrow \quad \Delta y = m \cdot \Delta x \quad$ e $\quad \Delta x = \Delta y / m$

- **Qual é a ideia então?**

✓ **Se Δx > Δy (0° < θ < 45°)**: incrementa-se Δx em uma unidade e calcula-se os sucessivos valores para y:

$$\Delta y = m \cdot \Delta x \rightarrow y_k - y_{k-1} = m \cdot 1 \rightarrow y_k = y_{k-1} + m$$

✓ **Se Δx < Δy (45° < θ < 90°):** incrementa-se Δy em uma unidade e calcula-se os sucessivos valores para x:

$$\Delta x = \Delta y / m \rightarrow x_k - x_{k-1} = 1 / m \rightarrow x_k = x_{k-1} + 1 / m$$

Método DDA

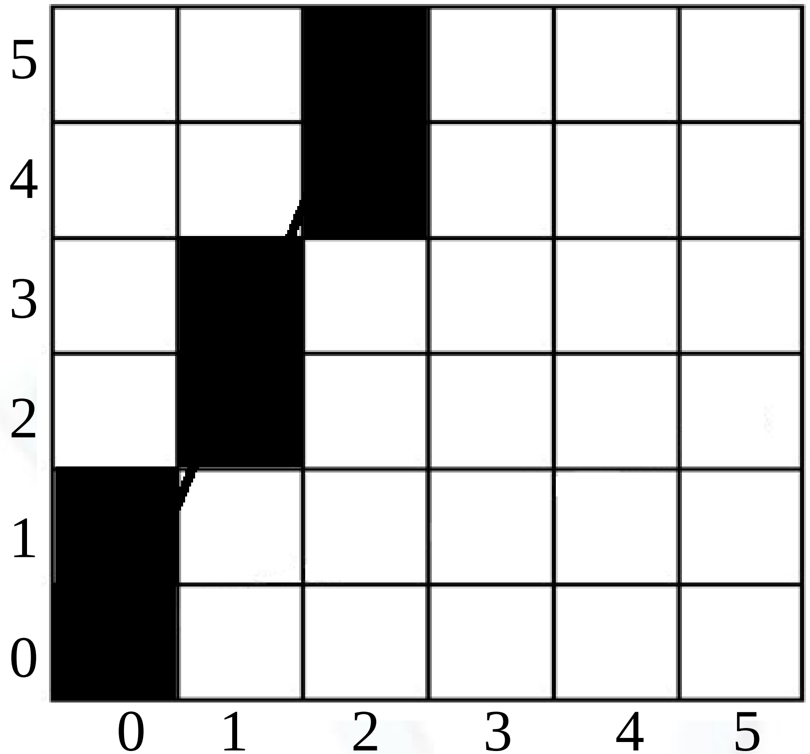
- Dados $P_1(x_1, y_1)$ $P_2(x_2, y_2)$ *temos o* algoritmo:

$\lvert x_2 - x_1 \rvert > \lvert y_2 - y_1 \rvert$	
Sim	Não
incremento = $y_2 - y_1 / x_2 - x_1$ $y = y_1$ p/ x de x_1 até x_2 faça dot (x,y,cor) y = y + incremento	incremento = $x_2 - x_1 / y_2 - y_1$ $x = x_1$ p/ y de y_1 até y_2 faça dot (x,y,cor) x = x + incremento

Método DDA

- **Problema grave: $P_1(0,0)$ $P_2(2,5)$**

Pergunta Chave: Quais pixel são ligados?



Processando...

$\Delta x = 2 - 0 = 2$

$\Delta y = 5 - 0 = 5$

Processando...

($\Delta x < \Delta y$ → varia y e calcula x)

$Inc = \Delta x / \Delta y = 2 / 5 = 0.4$

Para...

$y = 0 \rightarrow x = 0.0 \rightarrow$ *Pixel (0, 0)*

$y = 1 \rightarrow x = 0.0 + 0.4 = 0.4 \rightarrow$ *Pixel (0, 1)*

$y = 2 \rightarrow x = 0.4 + 0.4 = 0.8 \rightarrow$ *Pixel (1, 2)*

$y = 3 \rightarrow x = 0.8 + 0.4 = 1.2 \rightarrow$ *Pixel (1, 3)*

$y = 4 \rightarrow x = 1.2 + 0.4 = 1.6 \rightarrow$ *Pixel (2, 4)*

$y = 5 \rightarrow x = 1.6 + 0.4 = 2.0 \rightarrow$ *Pixel (2, 5)*

Método DDA

- Algoritmo simples;
- Resolve o problema de descontinuidade do Analítico;
- Ainda com vários problemas:
  - ✓ Utiliza aritmética de ponto-flutuante;
  - ✓ Sujeito a erros de arredondamento;
  - ✓ Pode ser lento em grande escala.

Método de Bresenham

- **Ideia básica:**

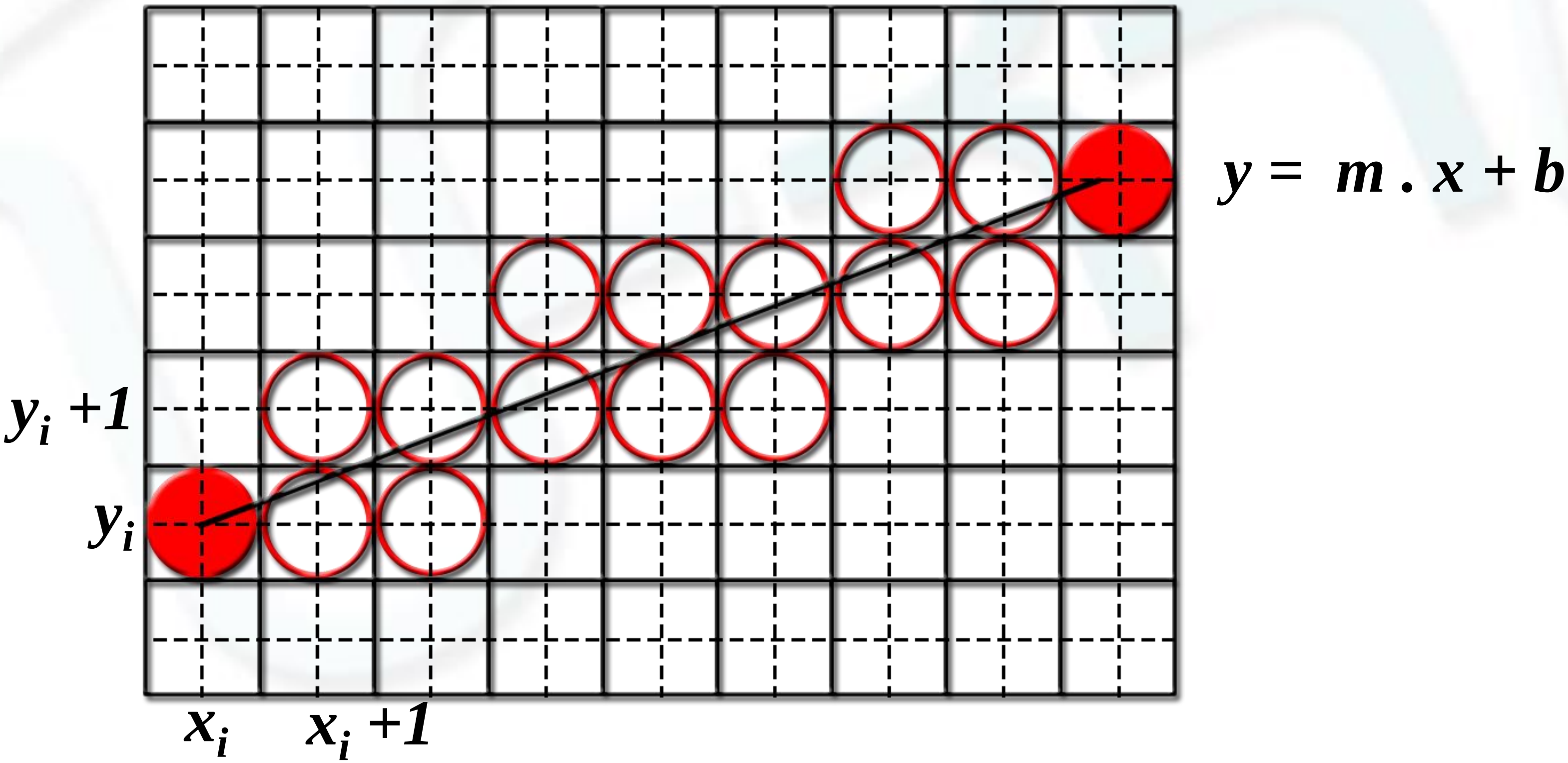
  ✓ Em vez de computar o valor do próximo y em ponto flutuante, decidir se o próximo pixel vai ter coordenadas $(x + 1, y)$ ou $(x + 1, y + 1)$

    - Considerando a reta no **primeiro octante**, a priori;

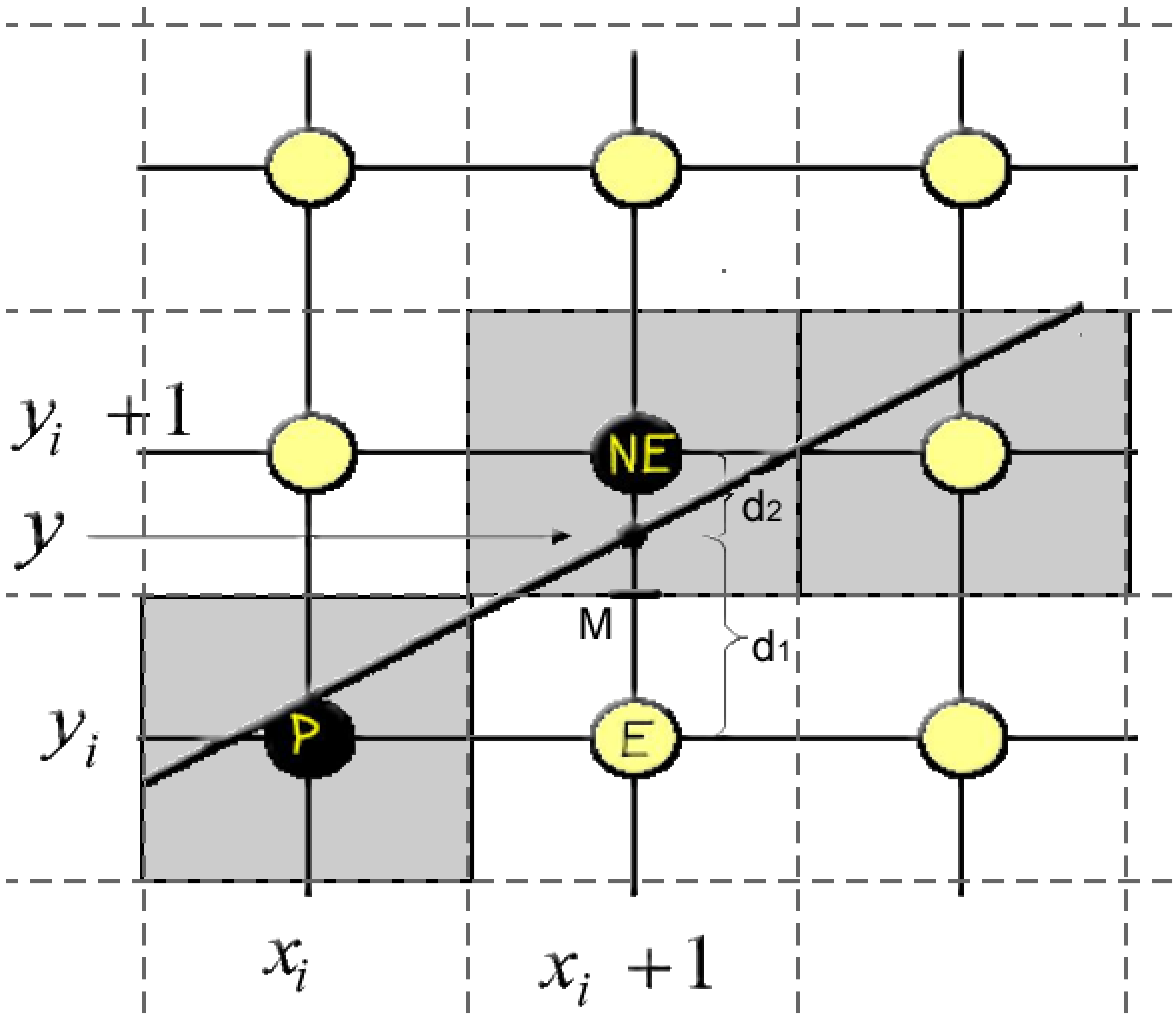
  ✓ Decisão requer que se avalie se a linha passa acima ou abaixo do ponto médio $(x + 1, y + ½)$;

Método de Bresenham

- **Considere a inclinação <u>$0 < m \leq 1$</u>;**
  - ✓ Ou seja, ela é crescente e $\Delta x \geq \Delta y$;
  - ✓ Logo se o pixel (x_i, y_i) está sobre a linha → o pixel mais próximo a linha será $(x_i + 1, y_i)$ ou (x_i+1, y_i+1);



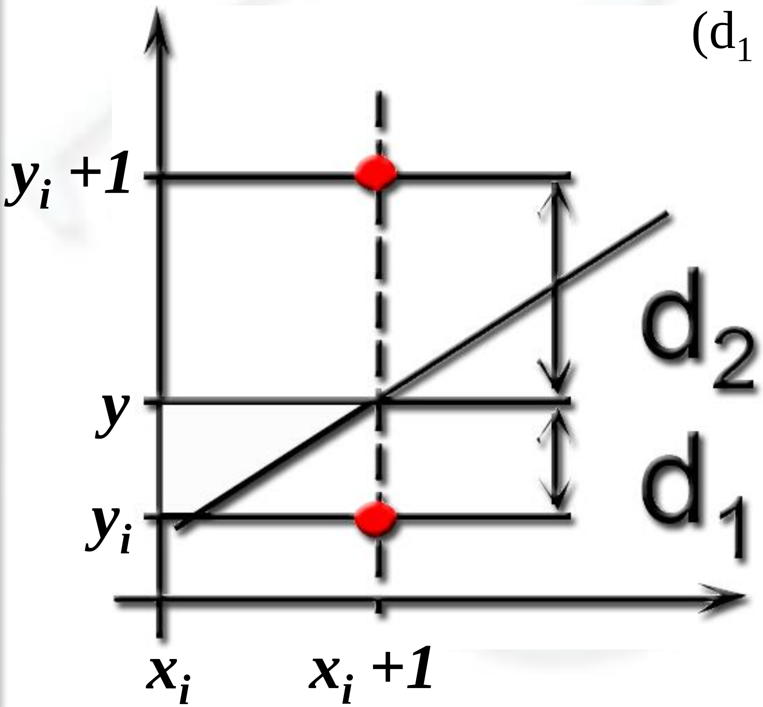
Linhas: Algoritmo de Bresenham



Método de Bresenham

- **Processo de decisão:**
  - ✓ Se $(d_1 – d_2) < 0$ → selecionar y_i
  - ✓ Se $(d_1 – d_2) \geq 0$ → selecionar y_i+1

  Onde:



$$
\begin{aligned}
(d_1 – d_2) &= (y – y_i) – (y_i + 1 – y) \\
&= 2.y – 2.y_i – 1 \\
&= 2.[m.(x_i+1) + b] – 2.y_i – 1 \\
&= 2.m.(x_i+1) + 2.b – 2.y_i – 1 \\
&= 2.m\,.\,x_i + 2.m + 2.b – 2.y_i – 1 \\
&= (2.\Delta y\,.\,x_i)/\Delta x + (2.\Delta y)/\Delta x + 2.b – 2.y_i – 1
\end{aligned}
$$

UFRR

Método de Bresenham

Parâmetro de decisão:

✓ Perceba, como estamos no primeiro octante $\Delta x > 0$

- $x_2 > x_1 \rightarrow x_2 - x_1 > 0 \rightarrow \Delta x > 0$
- Assim, se $(\mathbf{d_1} - \mathbf{d_2}) < 0 \rightarrow$ selecionar y_i
- Então, se $p_i = \Delta x \,.\, (\mathbf{d_1} - \mathbf{d_2}) < 0 \rightarrow$ selecionar y_i
- Se $(\mathbf{d_1} - \mathbf{d_2}) \geq 0 \rightarrow$ selecionar $y_i + 1$
- Então, se $p_i = \Delta x \,.\, (\mathbf{d_1} - \mathbf{d_2}) \geq 0 \rightarrow$ selecionar y_i+1

Método de Bresenham

- **Parâmetro de decisão:**

Sendo:

$$d_1 - d_2 = (2.\Delta y \,.\, x_i)/\Delta x + (2.\Delta y)/\Delta x + 2.b - 2.y_i - 1$$

E considerando: $\mathbf{p_i = \Delta x \,.\, (d_1 - d_2)}$

Então:

$$p_i = 2.\Delta y \,.\, x_i + 2.\Delta y + 2.\Delta x.b - 2.\Delta x.y_i - \Delta x$$

Método de Bresenham

Parâmetro de decisão

- ✓Ele norteará todas as nossas escolhas;
- ✓Estratégia agora:
  - Vamos descobrir o primeiro $p_i \rightarrow p_0$
  - Vamos encontrar uma maneira de calcular os próximos p_i (s)... $\rightarrow$ p_{i+1}

Método de Bresenham

- **Descobrindo p_0:**

$$p_i = 2.\Delta y \,.\, x_i \; + 2.\Delta y + 2.\Delta x.b - 2.\Delta x.y_i - \Delta x$$

Então:

$$p_0 = 2.\Delta y \,.\, x_0 \; + 2.\Delta y + 2.\Delta x.b - 2.\Delta x.y_0 - \Delta x$$

Perceba $y_0 = (\Delta y/ \Delta x).x_0 + b$

Substituindo temos:

$$p_0 = 2.\Delta y \,.\, x_0 \; + 2.\Delta y + 2.\Delta x.b - 2.\Delta x.[(\Delta y/ \Delta x).x_0 + b] - \Delta x$$

Método de Bresenham

- **Descobrindo p_0:**

$$p_0 = 2.\Delta y \,.\, x_0 + 2.\Delta y + 2.\Delta x.b - 2.\Delta x.[(\Delta y/\Delta x).x_0 + b] - \Delta x$$

Expandindo:

$$p_0 = \cancel{2.\Delta y \,.\, x_0} + 2.\Delta y + \cancel{2.\Delta x.b} - \cancel{2.\Delta y \,.\, x_0} - \cancel{2.\Delta x.b} - \Delta x$$

Simplificando:

$$\boxed{p_0 = 2\Delta y - \Delta x}$$

Método de Bresenham

- **Obtendo p_{i+1} a partir de p_i:**

$$p_i = 2.\Delta y\ .\ x_i + 2.\Delta y + 2.\Delta x.b - 2.\Delta x.y_i - \Delta x$$

$$P_{i+1} = 2.\Delta y\ .\ x_{i+1} \cancel{+ 2.\Delta y} \cancel{+ 2.\Delta x.b} - 2.\Delta x.y_{i+1} \cancel{- \Delta x}$$

$$- p_i = -2.\Delta y\ .\ x_i \cancel{- 2.\Delta y} \cancel{- 2.\Delta x.b} + 2.\Delta x.y_i \cancel{+ \Delta x}$$

Logo:

$$p_{i+1} - p_i = 2.\Delta y\ .\ (x_{i+1} - x_i) - 2.\Delta x\ .\ (y_{i+1} - y_i)$$

Como $x_{i+1} = x_i + 1 \rightarrow x_{i+1} - x_i = 1$. Portanto:

$$\boxed{p_{i+1} = p_i + 2.\Delta y - 2.\Delta x.(y_{i+1} - y_i)}$$

Método de Bresenham

- **Parâmetro de decisão incremental:**

$$p_{i+1} = p_i + 2.\Delta y - 2.\Delta x.(y_{i+1} - y_i)$$

Se $p_i < 0 \rightarrow$ selecionar $y_i \rightarrow y_{i+1} = y_i \rightarrow y_{i+1} - y_i = 0$

$$\boxed{\boldsymbol{p_{i+1} = p_i + 2.\Delta y}}$$

Se $p_i \geq 0 \rightarrow$ selecionar $y_i + 1 \rightarrow y_{i+1} = y_i + 1 \rightarrow y_{i+1} - y_i = 1$

$$p_{i+1} = p_i + 2\Delta y - 2\Delta x$$

$$\boxed{\mathbf{p_{i+1} = p_i + 2(\Delta y - \Delta x)}}$$

UFRR

Método de Bresenham

$\Delta x = x_2 - x_1$; $\Delta y = y_2 - y_1$; $y = y_1$

Parâmetro = $p = 2 . \Delta y - \Delta x$

para x de x_1 até x_2 faça:

liga pixel (x, y, cor)

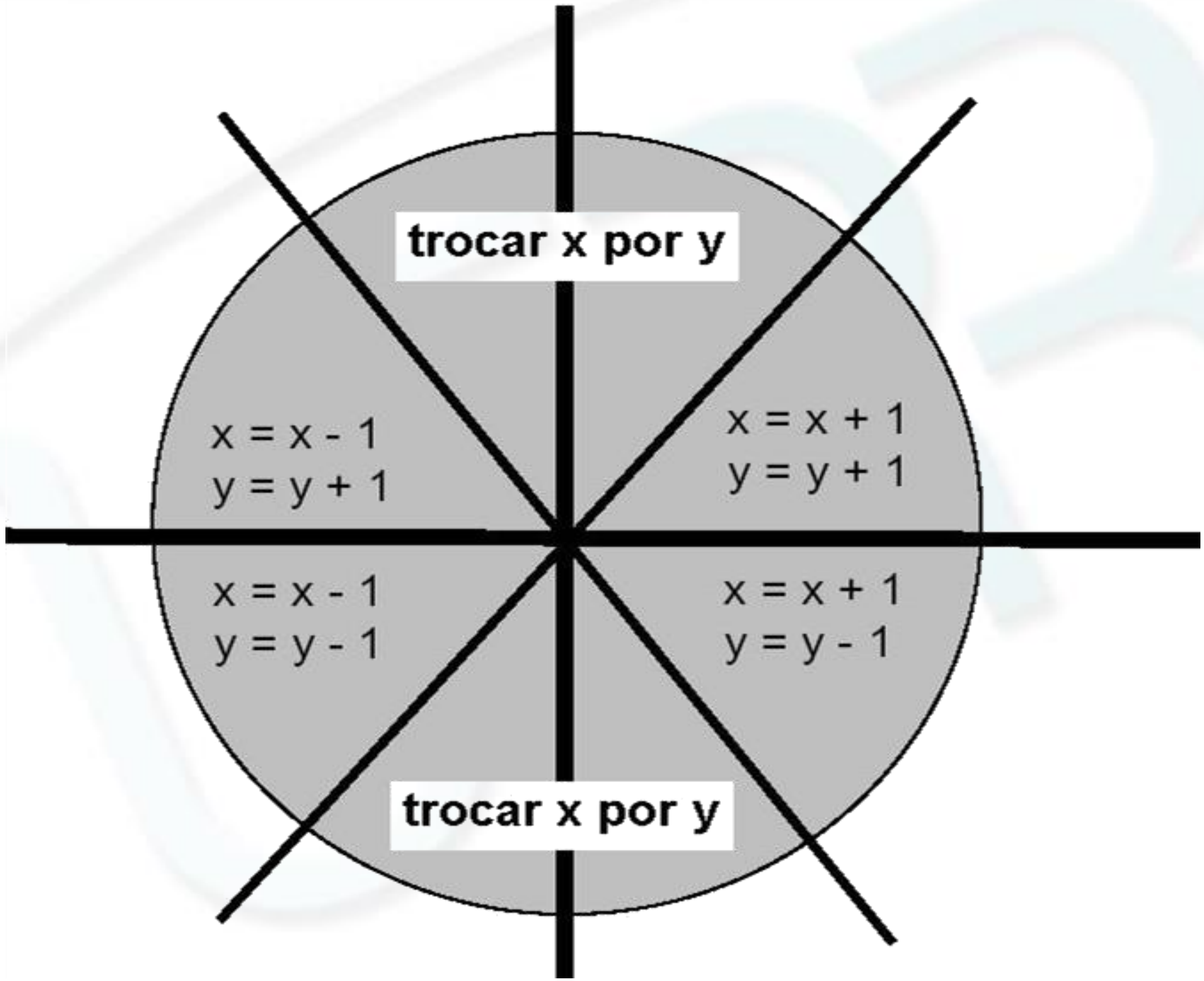
$p \geq 0$

Sim	Não
$y = y+1$ $p = p + 2.(\Delta y - \Delta x)$	não altera y $p = p + 2.\Delta y$

Método de Bresenham

```
ALGORITMO BRES_INT (x1, y1, x2, y2)
    dy = y2 – y1; dx = x2 – x1;  y = y1;
    p = 2dy – dx;
    FOR (x = x1 TO x2) {
        WritePixel(x, y);
        IF (p ≥ 0) {
            y = y + 1;
            p = p – 2(dy – dx); }
        ELSE {p = p+2dy;}
    }
```

Bresenham - Outros Octantes



Extensão para os demais octantes

- **Se $x_2 < x_1$**
  - ✓ Trocar P_1 com P_2
- **Se $y_2 < y_1$**
  - ✓ $y_1 \leftarrow -y_1$
  - ✓ $y_2 \leftarrow -y_2$
  - ✓ Pintar pixel $(x, -y)$
- **Se $|y_2 - y_1| > |x_2 - x_1|$**
  - ✓ Repetir o algoritmo trocando "y" com "x"

Método de Bresenham

- **Principais vantagens:**
  - ✓ Método bastante veloz;
  - ✓ Utiliza somente aritmética inteira;
  - ✓ Usa um incremento unitário;
  - ✓ Evita operações caras em pontos flutuantes: multiplicação e divisão;
  - ✓ As multiplicações que ele realiza são por 2 (deslocamento de bit):

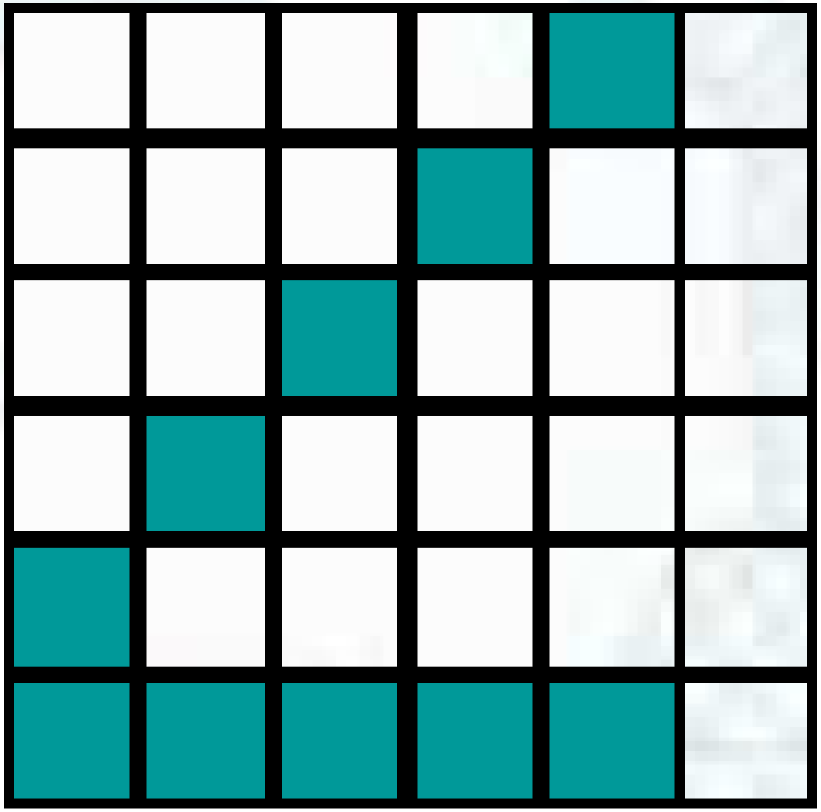
Ex.:	3: 11	24:11000
	6: 110	48:110000
	12:1100	....

Método de Bresenham

- **Principais vantagens:**
  - ✓ Evita operações de arredondamente;
  - ✓ Aproveita a coerência espacial: similaridade de valores referentes a pixel vizinhos;
  - ✓ Escolha entre dois valores de pixel vizinhos;

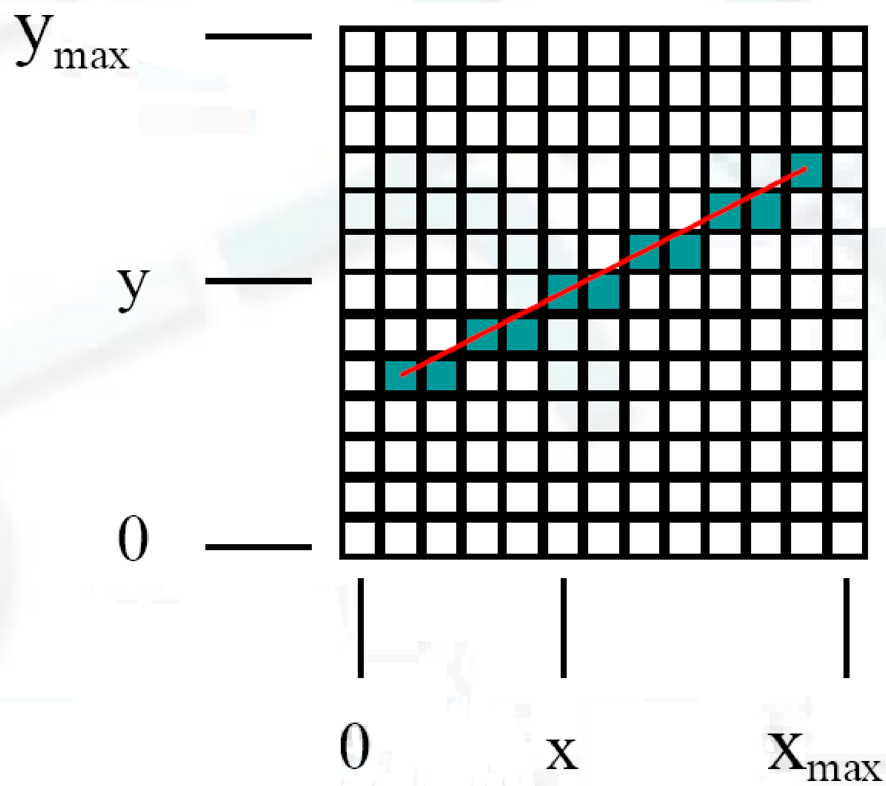
Antialiasing de linhas

- Linhas de comprimentos diferentes podem ter o mesmo número de pixels;
- Ambas as linhas têm 5 pixels, mas a linha diagonal é maior (fator √2);
- Linhas diagonais aparentam mais apagadas do que linhas horizontais e verticais



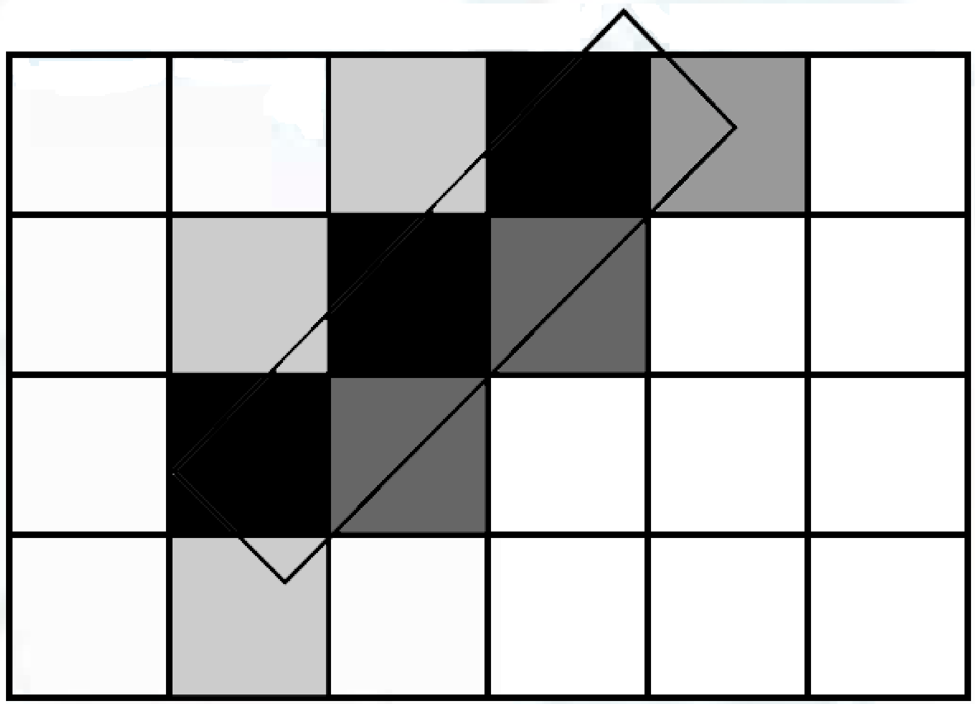
Antialiasing de linhas

- **Ajusta as intensidade de pixels ao longo de uma reta;**
  - ✓ Compensa o efeito-degrau produzido pelo processo de rasterização;



Antialiasing de linhas

- **Exemplo:**
  - ✓ Considerar a linha como um retângulo de largura 1 pixel;
  - ✓ Acender cada pixel com intensidade proporcional à área encoberta;



Antialiasing de linhas

- **Alternativa mais barata;**
  - ✓ Aproxima percentual de área coberta pela distância do centro do pixel parcialmente coberto à linha central da área retangular;
- **Aumenta o tempo de cálculo do algoritmo de geração de linhas, mas produz um resultado melhor;**

Antialiasing de linhas

- **Para compensar esse problema:**

  ✓Ajustar a intensidade de uma linha em função de sua inclinação;

  ✓Maior intensidade para $|m| = 1$;

  ✓Menor intensidade para $m = 0$ e $m = \infty$

Primeiro Trabalho

- Neste trabalho você deve (INDIVIDUALMENTE):

1. Desenvolver um programa que permita desenhar retas por meio dos algoritmos: **Analítico**, **DDA** e **Bresenham**.

2. Construir um relatório que descreva a construção e os resultados de maneira comparativa.

3. Apresentar o programa desenvolvido e entregar o relatório digital na sala do professor;

   - Obviamente você será arguido nesse momento.

Primeiro Trabalho

- **OBSERVAÇÕES:**
  - ✓ Objetivo do trabalho é que você veja os algoritmos trabalhando e os compare, desse modo, implemente de forma que isso aconteça;
  - ✓ Trabalhos entregues sem defesa não serão aceitos (receberão nota zero (0)).
    - Dessa forma, **NÃO** adianta apenas fazer o trabalho e enviar por e-mail;
    - **Muita atenção:** não sou uma PJ ou PF precisando de software de rasterização, com isso em mente apresente com o intuito de:
      - – Provar que você entende os algoritmos;
      - – Provar que você realmente é o autor do programa.

Primeiro Trabalho

- **OBSERVAÇÕES (Cont...):**
  - ✓ Trabalhos entregues sem o relatório serão avaliados em cinquenta porcento (50%) da nota;
  - ✓ Trabalhos entregues após a data final estabelecida não serão aceitos, a não ser com a apresentação de atestado médico ou declaração de serviço militar;
- **DICAS:**
  1. Escolha a linguagem de programação que você mais domina e que você entende que **não** será um fator limitante no desenvolvimento do trabalho;
  2. Comece a desenvolver o trabalho hoje e não deixe para apresentar no último dia.

Dúvidas