Conversão Matricial Algoritmos de Anti-Aliasing

Uéliton Freitas

Universidade Católica Dom Bosco - UCDB

freitas.ueliton@gmail.com

7 de outubro de 2014

Sumário

Introdução

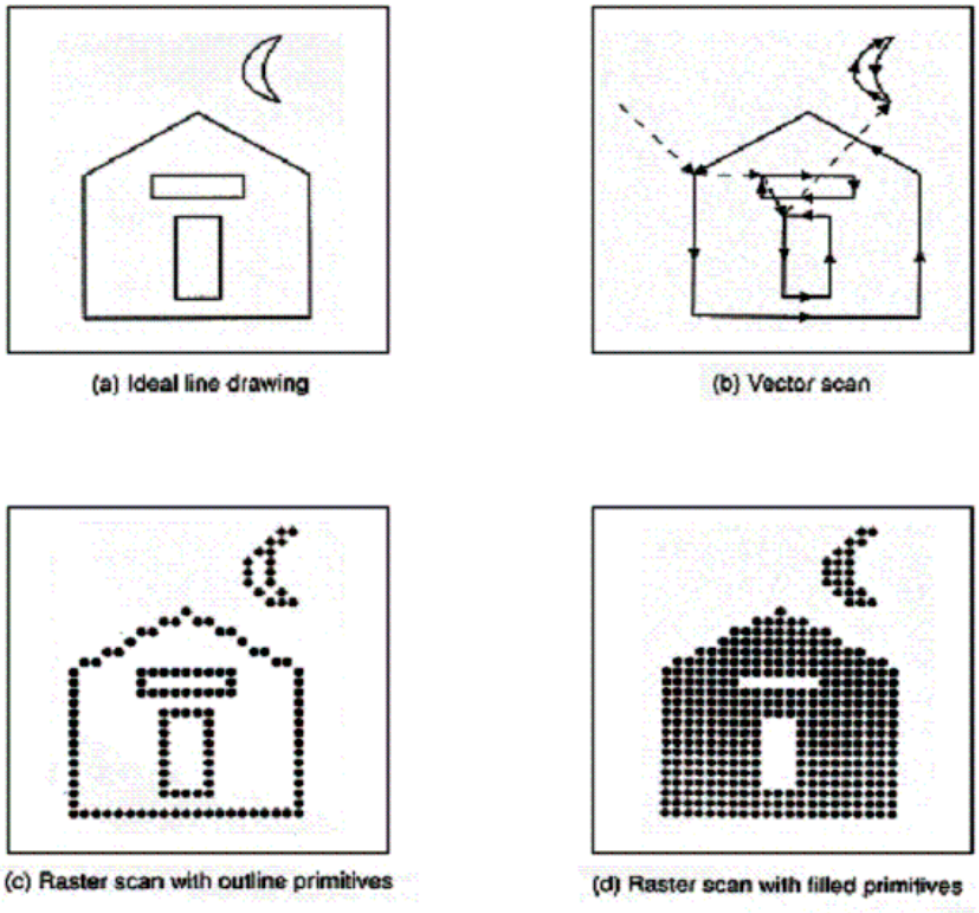


Figura : Imagen vetorial $\times$ Imagem Matricial

Introdução

Problemas

- Traçar primitivas geométricas (segmentos de retas, polígonos, circunferências, elipses, curvas,...) no dispositivo matricial.
- "rastering" = conversão vetorial $\rightarrow$ matricial.
- Como ajustar uma curva, definida como coordenadas reais em um sistema de coordenadas inteiras cujos "pontos" tem área associada.

Introdução

Problemas

- Traçar primitivas geométricas (segmentos de retas, polígonos, circunferências, elipses, curvas,...) no dispositivo matricial.
- "rastering" = conversão vetorial $\rightarrow$ matricial.
- Como ajustar uma curva, definida como coordenadas reais em um sistema de coordenadas inteiras cujos "pontos" tem área associada.

Introdução

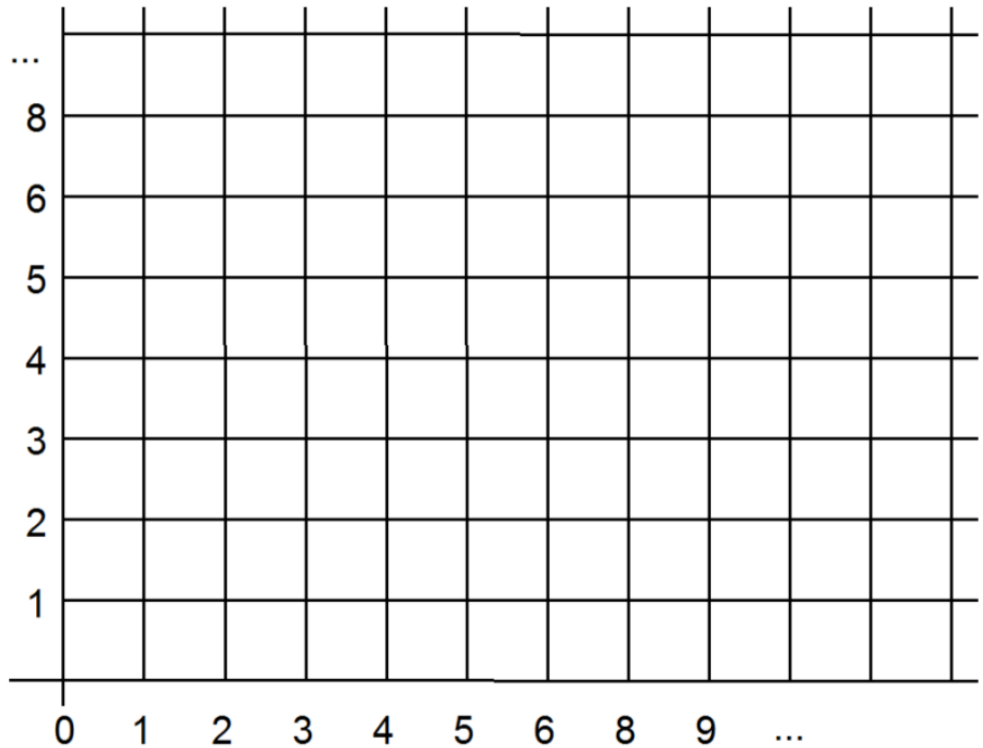


Figura : Sistema de coordenadas de dispositivo.

Conversão de Segmento de Reta

Conversão de Segmento de Reta

- Dados pontos extremos em coordenadas de dispositivo
  - $P_0(x_0, y_0)$.
  - $P_{end}(x_{end}, y_{end})$.
- Determinar quais pixels devem ser "acesos" para gerar uma boa aproximação do segmento de reta ideal.

Conversão de Segmento de Reta

Conversão de Segmento de Reta

- Características desejáveis:
  - Linearidade.
  - Precisão.
  - Espessura (Densidade Uniforme).
  - Intensidade independente de inclinação.
  - Continuidade.
  - Rapidez.

Conversão de Segmento de Reta

Equação da Reta

- Usar equação explícita da reta

$$y = m \cdot x + b$$

- m é a inclinação da reta e é dado por

$$m = \frac{y_{end} - y_0}{x_{end} - x_0}$$

- b é a intersecção do eixo y e dado por

$$b = y_0 - m \cdot x_0$$

Conversão de Segmento de Reta

Algoritmos Simples

- Varia-se x unitariamente de pixel em pixel, encontrando o valor de y.

```
1  {
2    int x, x0, xend, y0, yend;
3    float y, m;
4    int valor; //cor do pixel
5
6    m = (yend - y0)/(xend - x0);
7    for (x = x0; x <= xend; x++) {
8      y = y0 + m * (x - x0);
9      write_pixel (x, round(y), valor); //arredonda y
10   }
11 }
```

Conversão de Segmento de Reta

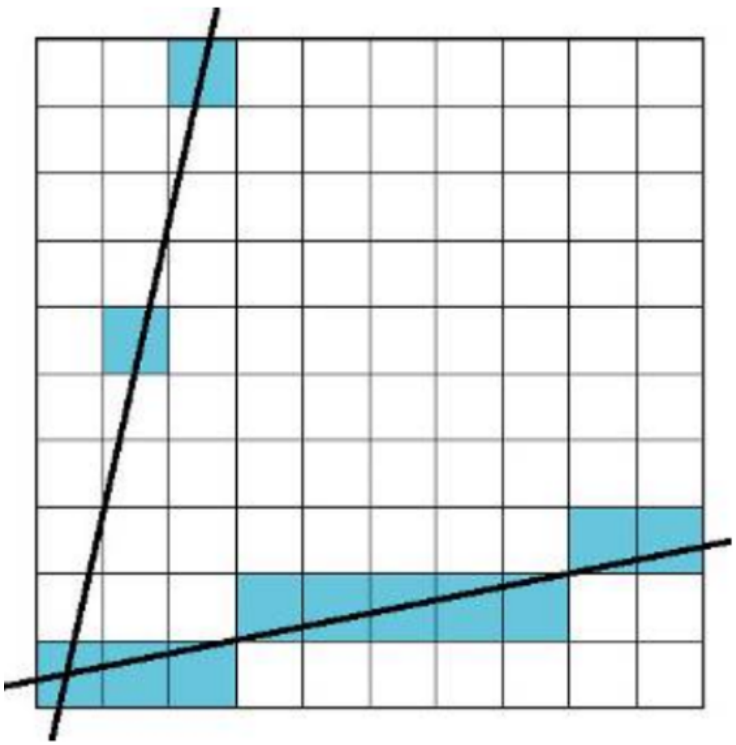
Algoritmos Simples

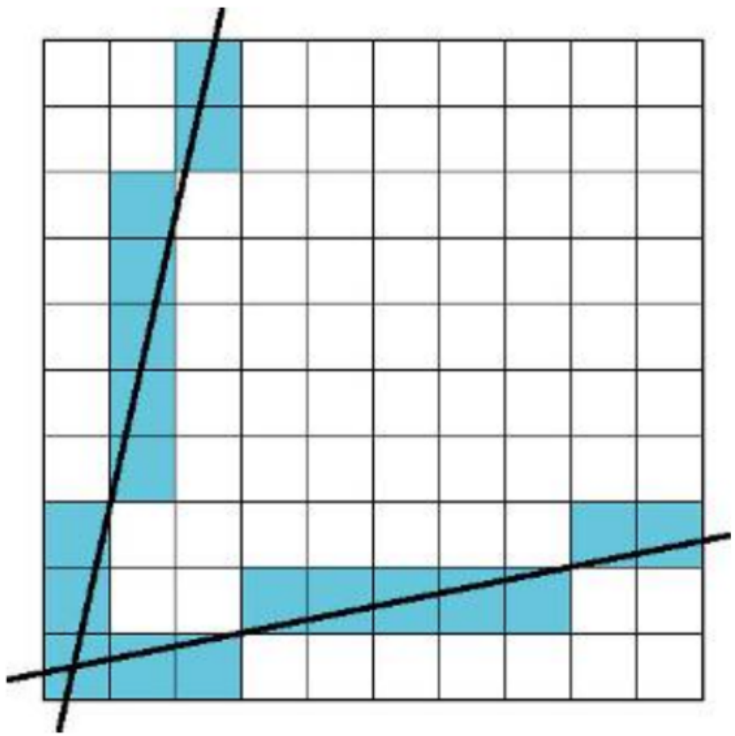
- Na forma dada funciona apenas com segmentos em que $0 < m < 1$. Por que?

Conversão de Segmento de Reta

Algoritmos Simples

- Se $0 < m < 1$ a variação em x é maior que em y. Caso não seja verdade, será traçado um segmento com buracos.





Conversão de Segmento de Reta

Algoritmos Simples

- Se $m > 1$ basta inverter os papéis de x e y, i.e, amostrar y em intervalos unitários, e calcular x.

$$x = x_0 + \frac{y - y_0}{m}$$

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Chamando δx uma variação em x, podemos encontrar a variação δy em y correspondente fazendo:

$$\delta y = m \cdot \delta x$$

- ou similarmente

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

- O algoritmo DDA se baseia no cálculo de δx e δy

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Para $|m| \leq 1$, na iteração i temos:

$$y_i = m \cdot x_i + b$$

- Sendo δ_x a variação na direção de x, na iteração $i + 1$ temos:

$$\begin{aligned} y_{i+1} &= m \cdot x_{i+1} + b \\ y_{i+1} &= m \cdot (x_i + \delta_x) + b \\ y_{i+1} &= m \cdot x_i + m \cdot \delta_x + b \\ y_{i+1} &= (m \cdot x_i + b) + m \cdot \delta_x \\ y_{i+1} &= y_i + m \cdot \delta_x \end{aligned}$$

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Para $|m| \leq 1$, na iteração i temos:

$$y_{i+1} = y_i + m \cdot \delta_x$$

- se $\delta_x = 1$ então:

$$x_{i+1} = x_i + 1$$
$$y_{i+1} = y_i + m$$

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Se $|m| > 1$, inverte-se os papéis, isto é, $\delta_y = 1$ e calcula-se x:

$$x_i = \frac{y_i - b}{m}$$

$$x_{i+1} = \frac{y_{i+1} - b}{m}$$

$$x_{i+1} = \frac{y_i + \delta_y - b}{m}$$

$$x_{i+1} = \frac{y_i - b}{m} + \frac{\delta_y}{m}$$

$$x_{i+1} = x_i + \frac{\delta_y}{m}$$

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Se $|m| > 1$, inverte-se os papéis, isto é, $\delta_y = 1$ e calcula-se x:

$$x_{i+1} = x_i + \frac{\delta_y}{m}$$

- se $\delta_y = 1$, então:

$$y_{i+1} = y_i + 1$$

$$x_{i+1} = x_i + \frac{1}{m}$$

Digital Differential Analyzer (DDA)

Digital Differential Analyzer (DDA)

- Assume-se que $x_0 < x_{end}$ e $y_0 < y_{end}$ (m positivo), processando da esquerda para a direita.
- Se não é o caso, $\delta_x = -1$ ou $\delta_y = -1$, a equação deve ser adaptada.
  - Fazer as adaptações como execícios.

Digital Differential Analyzer (DDA)

Exercício

- Aplica o algoritmo de DDA para a conversão dos seguintes segmentos de retas:
  - $P_1 = (0, 1), P_2 = (5, 3)$
  - $P_1 = (1, 1), P_2 = (3, 5)$

Algoritmo de Bresenham

Algoritmo de Bresenham

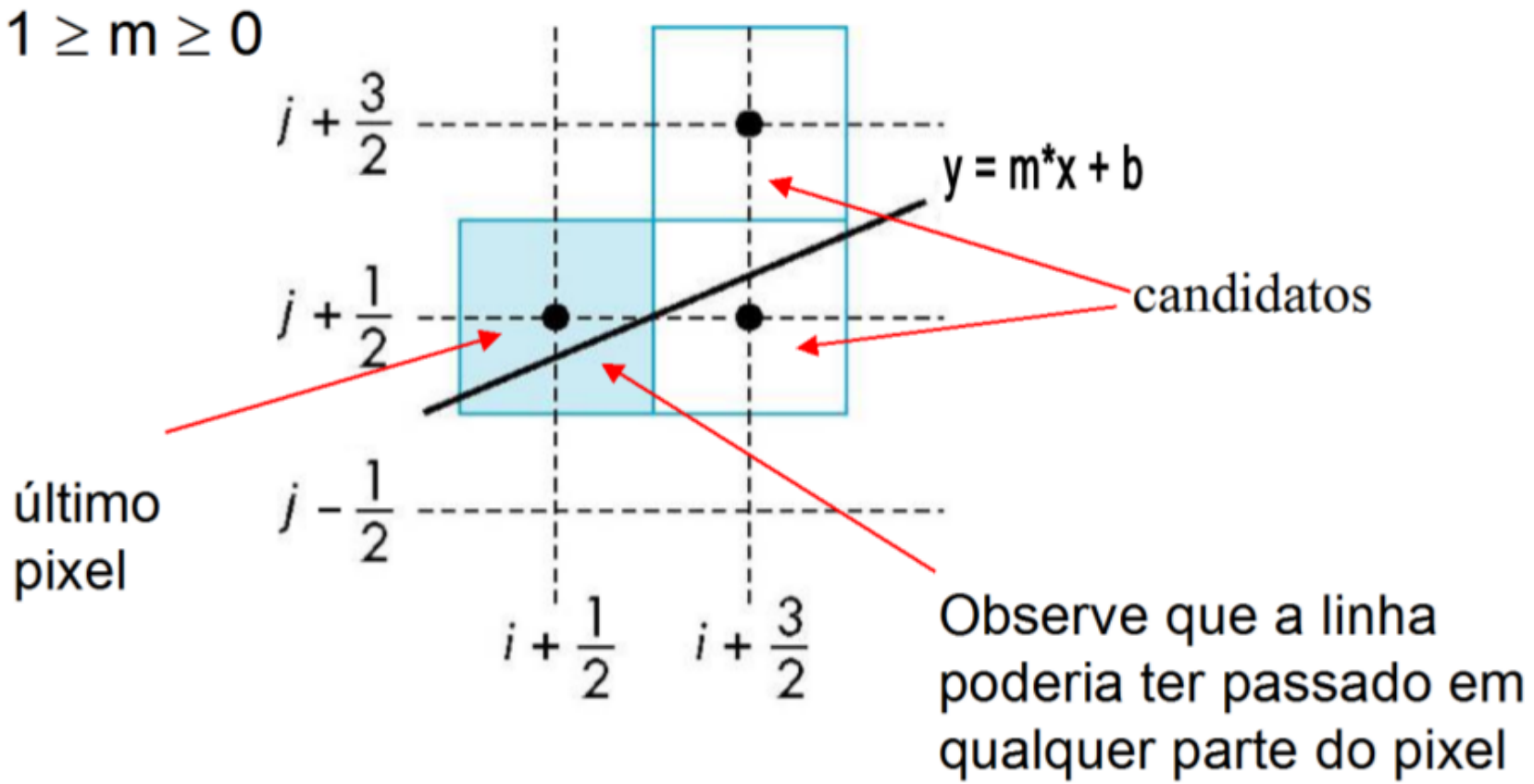
- O algoritmo DDA apesar de ser incremental, envolve cálculos com números flutuantes (cálculo de m) sendo ineficiente.
- O algoritmo de Bresenham trabalha apenas com inteiros sendo mais eficiente.

Algoritmo de Bresenham

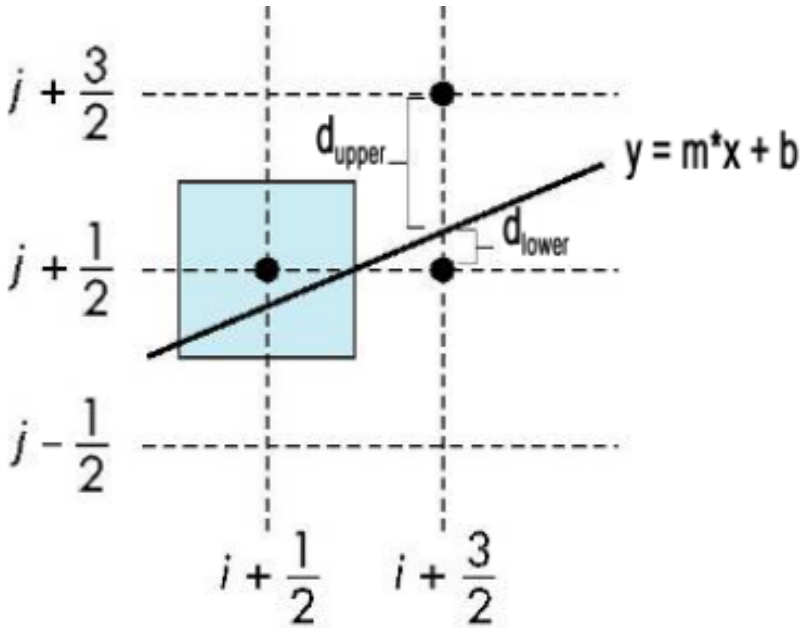
Algoritmo de Bresenham (Retas)

- Assume $0 < |m| < 1$.
- Incrementa x em intervalos unitários, calcula o y correspondente.
- Considera-se as duas possibilidades de escolha para y, decidindo qual a melhor.
  - $(x_k, y_k) \rightarrow (x_{k+1}, y_k)$
  - $(x_k, y_k) \rightarrow (x_{k+1}, y_{k+1})$

Algoritmo de Bresenham



Algoritmo de Bresenham



Algoritmo de Bresenham (Retas)

- $(d_{lower} - d_{upper} \geq 0) \rightarrow$ pixel superior.
- $(d_{lower} - d_{upper} < 0) \rightarrow$ pixel inferior.

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- Com base na equação da reta ($y = m \cdot x + b$), na posição $x_k + 1$, a coordenada y é calculada da seguinte forma:

$$y = m \cdot (x_k + 1) + b$$

- Então:

$$d_{lower} = y - y_k$$
$$d_{lower} = m \cdot (x_k + 1) + b - y_k$$

- e:

$$d_{upper} = (y_k + 1) - y$$
$$d_{upper} = y_k + 1 - m \cdot (x_k + 1) - b$$

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- Um teste rápido pode ser feito da seguinte forma para saber a proximidade:

$$p_k = d_{lower} - d_{upper} \quad (1)$$

$$p_k = 2m(x_k + 1) - 2y_k + 2b + 1 \quad (2)$$

- Assim:
  - $p_k > 0 \rightarrow$ pixel superior.
  - $p_k < 0 \rightarrow$ pixel inferior.

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- Mas calcular m é uma operação que ainda envolve ponto flutuante:

$$m = \frac{y_{end} - y_0}{x_{end} - x_0} = \frac{\Delta_y}{\Delta_x}$$

- Substituindo m por $\frac{\Delta_y}{\Delta_x}$, e multiplicando tudo por Δ_x em (1) temos:

$$p_k = \Delta x(d_{lower} - d_{upper})$$

- Como $\Delta x > 0$, o sinal de p_k não é alterado, assim:

$$p_k = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

- com $c = 2\Delta y + \Delta x(2b - 1)$ que é um parâmetro constante e independente da posição do pixel.

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- No passo $k+1$ temos:

$$p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c$$

- subtraindo p_k em ambos os lados temos:

$$p_{k+1} - p_k = (2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c) - p_k$$
$$p_{k+1} - p_k = 2\Delta y \cdot (x_{k+1} - x_k) - 2\Delta x \cdot (y_{k+1} - y_k)$$

- e como $x_{k+1} = x_k + 1$ (incremento unitário em x), então:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x \cdot (y_{k+1} - y_k)$$

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- Nesta equação:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x \cdot (y_{k+1} - y_k)$$

- $y_k + 1 - y_k$ será 0 ou 1 dependendo do sinal de p_k
- Se $p_k < 0$, então o próximo ponto $(x_k + 1, y_k)$ então:

$$y_{k+1} - y_k = 0$$
$$p_k + 1 = p_k + 2\Delta y$$

- caso contrário o ponto será $(x_k + 1, y_k + 1)$, assim:

$$y_{k+1} - y_k = 1$$
$$p_k + 1 = p_k + 2\Delta y - 2\Delta x$$

Algoritmo de Bresenham

Algoritmo de Bresenham (Retas)

- Este cálculo iterativo é realizado para cada posição de x começando da esquerda para a direita.
- O ponto de partida é calculado como sendo

$$p_0 = 2\Delta y - \Delta x$$

Algoritmo de Bresenham

```
1  void bresenham (int x1,int x2, int y1,int y2)
2  int dx,dy, incSup, incInf, p, x, y;
3  int valor;
4  {
5    dx = x2-x1; dy = y2-y1;
6    p = 2*dy-dx; /* fator de decisão: valor inicial */
7
8    incInf = 2*dy; /* Incremento Superior */
9    incSup = 2*(dy-dx); /* Incremento inferior */
10
11   x = x1; y = y1;
12   write_Pixel (x,y,valor); /* Pinta pixel inicial */
13
14   while (x < x2) {
15     if (p <= 0) { /* Escolhe Inferior */
16       p = p + incInf;}
17     else { /* Escolhe Superior */
18       p = p + incSup;
19       y++;} /* maior que 45o */
20     x++;
21     write_pixel (x, y, valor);
22   } /* fim do while */
23 } /* fim do algoritmo */
```

Algoritmo de Bresenham

Exercício

- Aplique o algoritmo para a reta composta pelos seguintes pontos em seu extremo:
  - $P_0(0, 0)$
  - $P_1(4, 3)$