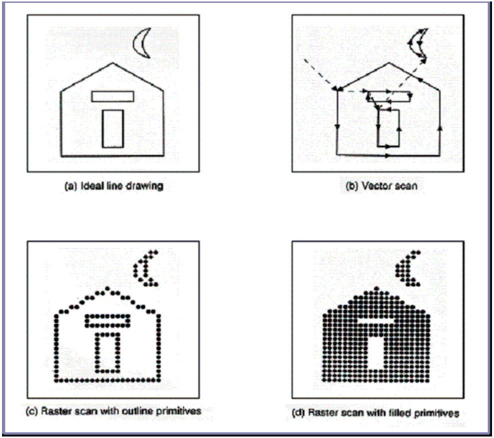
Conversão Matricial - Retas

Profa. Ana Luísa D. Martins Lemos

March 21, 2018

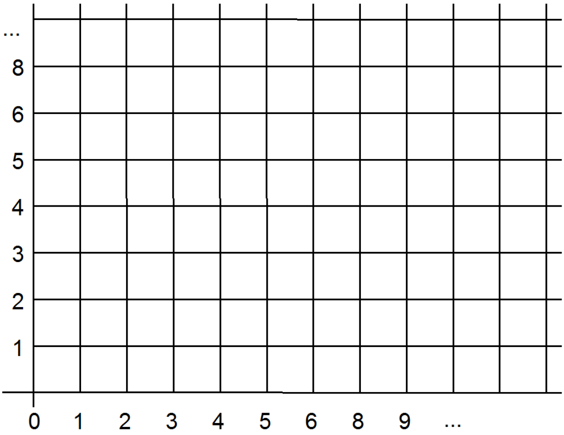
Imagem Vetorial × Imagem Matricial



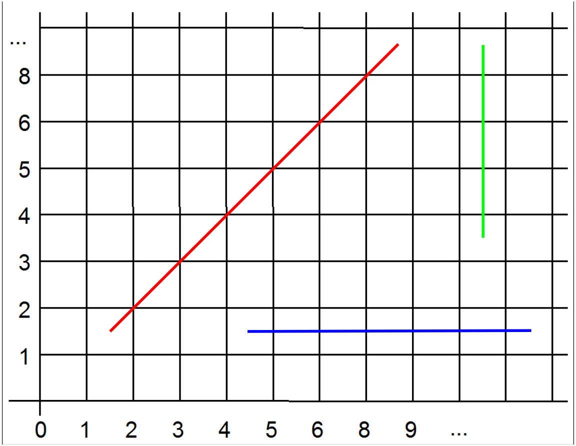
Problema

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

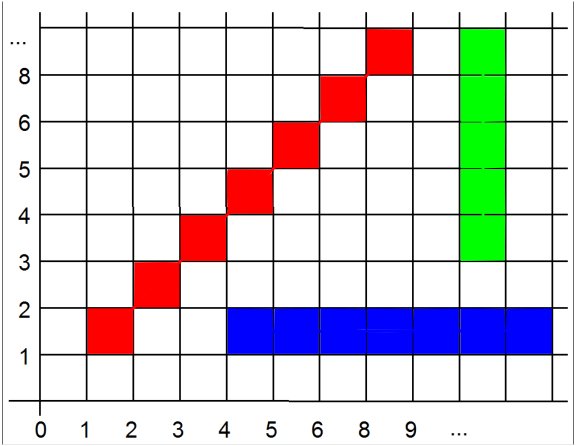
Sistema de Coordenadas do Dispositivo



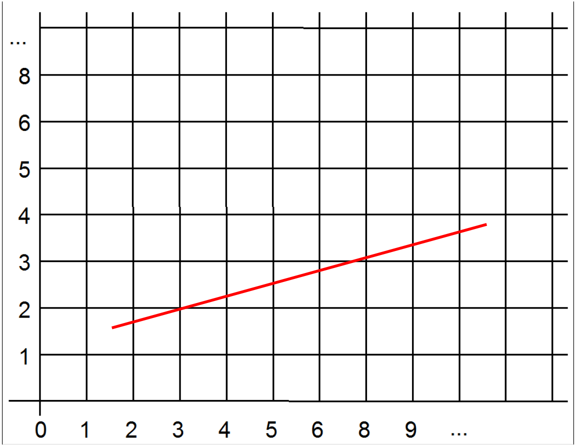
Sistema de Coordenadas do Dispositivo



Sistema de Coordenadas do Dispositivo



Sistema de Coordenadas do Dispositivo



Conversão de Segmentos de Reta

- Dados pontos extremos em coordenadas do dispositivo
  - $P_0 = (x_0, y_0)$ (inferior esquerdo)
  - $P_{\text{end}} = (x_{\text{end}}, y_{\text{end}})$ (superior direito)
- Determina quais pixels devem ser "acesos" para gerar na tela uma boa aproximação do segmento de reta ideal

Conversão de Segmentos de Reta

- Características desejáveis:
  - Linearidade
  - Precisão
  - Espessura (Densidade Uniforme)
  - Intensidade independente de inclinação
  - Continuidade
  - Rapidez no traçado

Conversão de Segmentos de Reta

- Usar equação explícita da reta

$$y = mx + b$$

- m é a inclinação da reta

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

- b é a interseção com o eixo y

$$b = y_0 - mx_0$$

Conversão de Segmentos de Reta

- Algoritmo Simples:
  - Variando-se x unitariamente de pixel em pixel, encontramos o valor de y

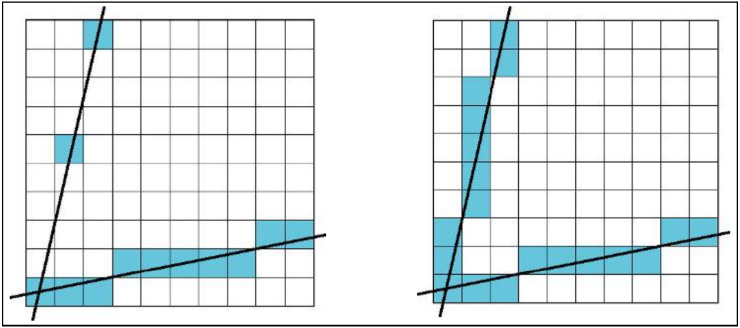
```
{
  int x, x0, xend, y0, yend;
  float y, m;
  int valor; //cor do pixel

  m = (yend - y0)/(xend - x0);
  for (x = x0; x <= xend; x++) {
    y = y0 + m * (x - x0);
    write_pixel (x, round(y), valor); //arredonda y
  }
}
```

Problema

- Na forma dada, só funciona para segmentos em que $0 < m < 1$. Porque?

Problema

- Se $0 < m < 1$ a variação em x é superior à variação em y. Se esse não for o caso, vai traçar um segmento com buracos!!



Solução

- Se $m > 1$, basta inverter os papéis de x e y, i.e., amostra y a intervalos unitários, e calcula x

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

Algoritmo DDA

- *Digital Differential Analyzer*
- Chamando de δ_x uma variação na direção de x, podemos encontrar a variação δ_y em y correspondente fazendo

$$\delta_y = m\delta_x$$

- ou similarmente

$$\delta_x = \frac{\delta_y}{m}$$

- O algoritmo se baseia no cálculo de δ_x ou δ_y

Algoritmo DDA

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

$$y_i = mx_i + b$$

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

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

- Se $\delta_x = 1$, então

$$\begin{aligned}
x_{i+1} &= x_i + 1, \text{ e} \\
y_{i+1} &= y_i + m
\end{aligned}$$

Algoritmo DDA

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

$$
\begin{aligned}
x_i &= \frac{y_i - b}{m} \\
x_{i+1} &= \frac{y_{i+1} - b}{m} \\
x_{i+1} &= \frac{y_i + \delta_y - b}{m} \\
x_{i+1} &= \frac{y_i - b}{m} + \frac{\delta_y}{m} \\
x_{i+1} &= x_i + \frac{\delta_y}{m}
\end{aligned}
$$

- Se $\delta_y = 1$, então

$$
\begin{aligned}
y_{i+1} &= y_i + 1, \text{ e} \\
x_{i+1} &= x_i + \frac{1}{m}
\end{aligned}
$$

Algoritmo DDA

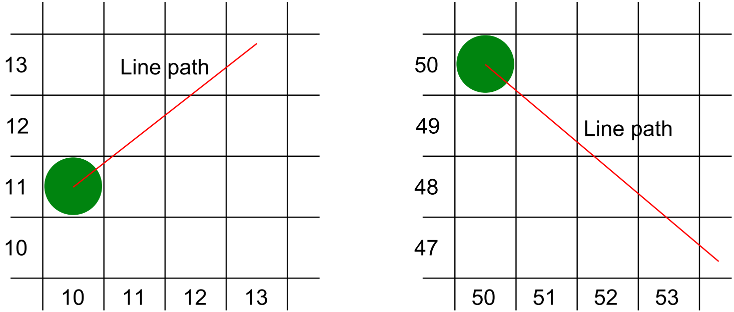
- Assume $x_0 < x_{end}$ e $y_0 < y_{end}$ (m positivo), processamento da esquerda para a direita
- Se não é o caso, então $\delta_x = -1$ ou $\delta_y = -1$, e a equação de traçado deve ser adaptada de acordo
  - Exercício: Fazer a adaptação em cada caso

Exercício

- Aplique o algoritmo (e adaptações) para fazer a conversão dos seguintes segmentos de reta
  1. $P_1 = (0, 1)$ e $P_2 = (5, 3)$
  2. $P_1 = (1, 1)$ e $P_2 = (3, 5)$

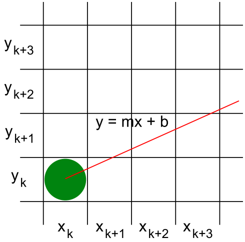
Algoritmo de Bresenham

- Algoritmo DDA apesar de ser incremental, envolve cálculo com números flutuantes (cálculo de m): ineficiente
- O algoritmo de **Bresenham** trabalha somente com inteiros: muito mais eficiente



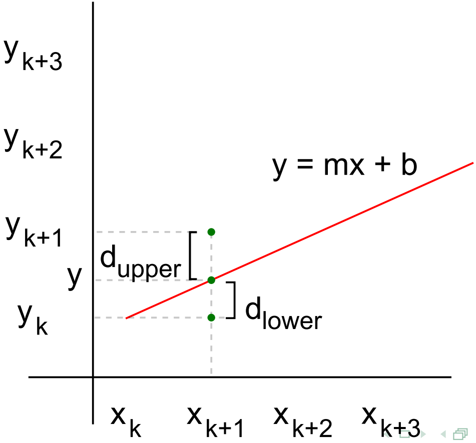
Algoritmo de Bresenham (Retas)

- Assumindo $0 < m < 1$
- Incrementa x em intervalos unitários, calcula o valor de y correspondente
- Abordagem considera as duas possibilidades de escolha de y, decidindo qual a melhor
  - $(x_k, y_k) \to (x_k + 1, y_k)$
  - $(x_k, y_k) \to (x_k + 1, y_k + 1)$



Algoritmo de Bresenham (Retas)

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



Algoritmo de Bresenham (Retas)

- Com base na equação da reta ($y = mx + b$), na posição $x_k + 1$, a coordenada y é calculada como

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

- Então

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

e

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

Algoritmo de Bresenham (Retas)

- Um teste rápido para saber a proximidade

$$p_k = d_{\text{lower}} - d_{\text{upper}}$$
$$p_k = 2m(x_k + 1) - 2y_k + 2b - 1$$

- Assim
  - $p_k \geq 0$: pixel superior
  - $p_k < 0$: pixel inferior

Algoritmo de Bresenham (Retas)

- Mas calcular m envolve operações de ponto flutuante

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

então, substituindo m por Δ_y/Δ_x, e multiplicando tudo por Δ_x, temos

$$p_k = \Delta_x(d_{\text{lower}} - d_{\text{upper}})$$

como $\Delta_x > 0$, o sinal de p_k não é alterado, então

$$p_k = 2\Delta_y x_k - 2\Delta_x y_k + c$$

com $c = 2\Delta_y + \Delta_x(2b - 1)$, parâmetro constante independente da posição do pixel

Algoritmo de Bresenham (Retas)

- No passo $k+1$ temos

$$p_{k+1} = 2\Delta_y x_{k+1} - 2\Delta_x y_{k+1} + c$$

subtraindo p_k dos dois lados tempos

$$p_{k+1} - p_k = (2\Delta_y x_{k+1} - 2\Delta_x y_{k+1} + c) - p_k$$

$$p_{k+1} - p_k = 2\Delta_y(x_{k+1} - x_k) - 2\Delta_x(y_{k+1} - y_k)$$

mas $x_{k+1} = x_k + 1$ (incremento unitário de x), então

$$p_{k+1} = p_k + 2\Delta_y - 2\Delta_x(y_{k+1} - y_k)$$

Algoritmo de Bresenham (Retas)

- Nessa equação

$$p_{k+1} = p_k + 2\Delta_y - 2\Delta_x(y_{k+1} - y_k)$$

$y_{k+1} - y_k$ será 0 ou 1 dependendo do sinal de p_k

- Se $p_k < 0$, então o próximo ponto é $(x_k + 1, y_k)$ então

$$y_{k+1} - y_k = 0 \text{ e}$$
$$p_{k+1} = p_k + 2\Delta_y$$

- Caso contrário o ponto será $(x_k + 1, y_k + 1)$ então

$$y_{k+1} - y_k = 1 \text{ e}$$
$$p_{k+1} = p_k + 2\Delta_y - 2\Delta_x$$

Algoritmo de Bresenham (Retas)

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

$$p_0 = 2\Delta_y - \Delta_x$$

Algoritmo de Bresenham (Retas)

1. Entre com os dois pontos (x_0, y_0) e $(x_{\text{end}}, y_{\text{end}})$
2. Entre com a cor para a posição no frame-buffer (x_0, y_0) e plote o primeiro ponto
3. Calcule as constantes Δ_x, Δ_y, $2\Delta_y$, e $2\Delta_y - 2\Delta_x$ e obtenha o valor inicial do parâmetro de decisão
$$p_0 = 2\Delta_y - \Delta_x$$
4. A cada x_k ao longo da linha, começando em $k = 0$, execute o seguinte teste
   - Se $p_k < 0$, o próximo ponto a ser plotado é $(x_k + 1, y_k)$ e
$$p_{k+1} = p_k + 2\Delta_y$$
     Caso contrário, o próximo ponto a ser plotado é $(x_k + 1, y_k + 1)$ e
$$p_{k+1} = p_k + 2\Delta_y - 2\Delta_x$$
5. Execute o passo 4 $\Delta_x - 1$ vezes

Linhas

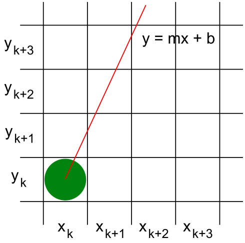
```
void bresenham (int x1,int x2, int y1,int y2){
  int dx,dy, incSup, incInf, p, x, y;
  int valor;

  dx = x2-x1; dy = y2-y1;
  p = 2*dy-dx; /* fator de decisão: valor inicial */

  incInf = 2*dy; /* Incremento inferior */
  incSup = 2*(dy-dx); /* Incremento superior */
  x = x1; y = y1;
  write_Pixel (x,y,valor); /* Pinta pixel inicial */
  while (x < x2) {
    if (p < 0) { /* Escolhe Inferior */
      p = p + incInf;}
    else { /* Escolhe Superior */
      p = p + incSup;
      y++;} /* maior que 45o */
    x++;
    write_pixel (x, y, valor);
  } /* fim do while */
} /* fim do algoritmo */
```

Algoritmo de Bresenham (Retas)

- Exercício: aplique o algoritmo para
  1. $P_1 = (5, 8)$ e $P_2 = (9, 11)$
  2. $P_1 = (20, 10)$ e $P_2 = (30, 18)$

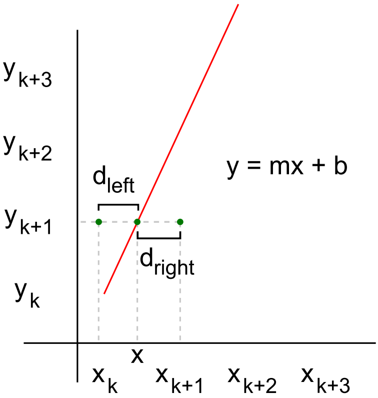
Algoritmo de Bresenham (Retas)

- Assumindo $m > 1$
- Incrementa y em intervalos unitários, calcula o valor de x correspondente
- Abordagem considera as duas possibilidades de escolha de x, decidindo qual a melhor
  - $(x_k, y_k) \rightarrow (x_k, y_k + 1)$
  - $(x_k, y_k) \rightarrow (x_k + 1, y_k + 1)$



Algoritmo de Bresenham (Retas)

- $(d_{\text{left}} - d_{\text{right}}) \geq 0 \rightarrow$ usar o pixel da direita
- $(d_{\text{left}} - d_{\text{right}}) < 0 \rightarrow$ usar pixel da esquerda



Algoritmo de Bresenham (Retas)

- Com base na equação da reta ($y = mx + b$), na posição $y_k + 1$, a coordenada x é calculada como

$$x = \frac{y_k + 1 - b}{m}$$

- Então

$$d_{\text{left}} = x - x_k$$
$$d_{\text{left}} = \frac{y_k+1-b}{m} - x_k$$

e

$$d_{\text{right}} = (x_k + 1) - x$$
$$d_{\text{right}} = x_k + 1 - \frac{(y_k+1-b)}{m}$$

Algoritmo de Bresenham (Retas)

- Um teste rápido para saber a proximidade

$$p_k = d_{\text{left}} - d_{\text{right}}$$
$$p_k = 2\frac{(y_k+1-b)}{m} - 2x_k - 1$$

- Assim
  - $p_k \geq 0$: pixel da direita
  - $p_k < 0$: pixel da esquerda

Algoritmo de Bresenham (Retas)

- Mas calcular m envolve operações de ponto flutuante

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

então, substituindo m por Δ_y/Δ_x, e multiplicando tudo por Δ_y, temos

$$p_k = \Delta_y(d_{\text{left}} - d_{\text{right}})$$

como $\Delta_y > 0$, o sinal de p_k não é alterado, então

$$p_k = 2\Delta_x y_k - 2\Delta_y x_k + c$$

com $c = 2\Delta_x(1 - b) - \Delta_y$, parâmetro constante independente da posição do pixel

Algoritmo de Bresenham (Retas)

- No passo $k+1$ temos

$$p_{k+1} = 2\Delta_x y_{k+1} - 2\Delta_y x_{k+1} + c$$

subtraindo p_k dos dois lados tempos

$$p_{k+1} - p_k = (2\Delta_x y_{k+1} - 2\Delta_y x_{k+1} + c) - p_k$$

$$p_{k+1} - p_k = 2\Delta_x(y_{k+1} - y_k) - 2\Delta_y(x_{k+1} - x_k)$$

mas $y_{k+1} = y_k + 1$ (incremento unitário de y), então

$$p_{k+1} = p_k + 2\Delta_x - 2\Delta_y(x_{k+1} - x_k)$$

Algoritmo de Bresenham (Retas)

- Nessa equação

$$p_{k+1} = p_k + 2\Delta_x - 2\Delta_y(x_{k+1} - x_k)$$

$x_{k+1} - x_k$ será 0 ou 1 dependendo do sinal de p_k

- Se $p_k < 0$, então o próximo ponto é $(x_k, y_k + 1)$ então

$$x_{k+1} - x_k = 0 \text{ e}$$
$$p_{k+1} = p_k + 2\Delta_x$$

- Caso contrário o ponto será $(x_k + 1, y_k + 1)$ então

$$x_{k+1} - x_k = 1 \text{ e}$$
$$p_{k+1} = p_k + 2\Delta_x - 2\Delta_y$$

Algoritmo de Bresenham (Retas)

- Esse cálculo iterativo é realizado para cada posição y começando de baixo para cima
- O ponto de partida é calculado como

$$p_0 = 2\Delta_x - \Delta_y$$

Algoritmo de Bresenham (Retas)

- Outros casos:
  - Caso ambos x e y decresçam de $P_0 = (x_0, y_0)$ para $P_{\text{end}} = (x_{\text{end}}, y_{\text{end}})$
    - Trocar ponto inicial por final.
  - Caso m seja negativo
    - Procedimento semelhante aos discutidos anteriormente, porém uma coordenada decresce enquanto a outra cresce
  - Tratar casos especiais à parte
    - Linhas horizontais $\Delta_y = 0$
    - Linhas verticais $\Delta_x = 0$
    - Linhas diagonais $|\Delta_x| = |\Delta_y|$
    - Em todos esses casos não é preciso nenhum processamento para identificar o próximo pixel

Algoritmo de Bresenham (Retas)

Lista p/ 28/03

1. Crie um programa com duas funções: a de imprimir uma linha entre dois pontos quaisquer e a de alterar a cor da linha. Uma linha inicial entre os pontos (0,0) e (0,0) azul deve ser traçada. A linha muda de posição de acordo com as coordenadas dadas por um clique inicial e um clique final com o botão esquerdo do mouse. Utilize o algoritmo de Bresenham para traçar a linha (não utilizar `GL_LINES`). Para mudar a cor, o usuário digitará as teclas de 0 a 9. Cada tecla deverá ter cores indexadas previamente escolhidas. Escolha as cores que desejar. A reta deve ser

Algoritmo de Bresenham (Retas)

UNIFESP
Universidade Federal de São Paulo
1933

Lista p/ 28/03

2. Escreva um programa que contenha três funções: as funções de traçar uma linha e de mudar a cor do exercício 1 e uma função de traçar triângulos (apenas as linhas que formam o triângulo). Inicialmente, deve ser impressa uma linha azul de coordenadas (0,0) - (0,0). Apenas uma figura deve ser apresentada de cada vez na tela. As figuras anteriores são apagadas. Caso o usuário clique a tecla 'r' ou 'R', a função de traçar retas é ativada. Caso o usuário clique a tecla 't' ou 'T' a função de traçar triângulos é ativada. O traçado da reta continua da mesma maneira do exercício 1. No traçado de triângulos, os três vértices são determinados por três cliques seguidos com o botão esquerdo do mouse. Retas e triângulos devem ser traçados utilizando o algoritmo de Bresenham.

Algoritmo de Bresenham (Retas)

Lista p/ 28/03

3. Crie um programa com quatro funções: traçar retas, traçar triângulos e mudar de cor, como do exercício 2, e uma função para alterar a espessura do traçado. As funções de mudar a espessura do traçado e de alterar a cor são selecionáveis pelas teclas 'e' ou 'E', e 'k' ou 'K', respectivamente. Quando a função de mudar a cor estiver ativa, os números de 0 a 9 indicam o índice da cor a ser utilizada, como no exercício 2. Na função de alterar a espessura do traçado, as teclas de 1 a 9 indicarão o valor desejado.