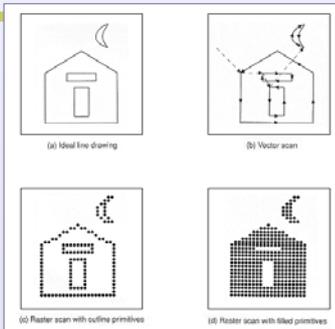


Conversão Matricial de Primitivas Gráficas

Maria Cristina F. de Oliveira
março 2009

Imagem Vetorial x Imagem Matricial



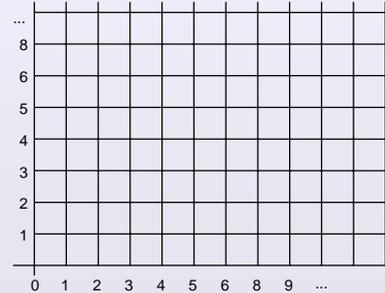
2

Problema

- # Traçar primitivas geométricas (segmentos de reta, polígonos, circunferências, elipses, curvas, ...) no dispositivo matricial
- # 'rastering' = conversão vetorial → matricial
- # Como ajustar uma curva, definida por coordenadas reais em um sistema de coordenadas contínuo, a uma malha de coordenadas inteiras cujos 'pontos' tem área associada

3

Sistema de Coordenadas do Dispositivo



4

Traçar uma reta pode ser trivial...

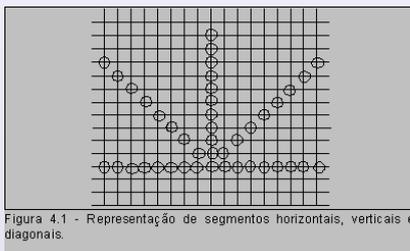


Figura 4.1 - Representação de segmentos horizontais, verticais e diagonais.

5

Diferentes critérios e resultados...

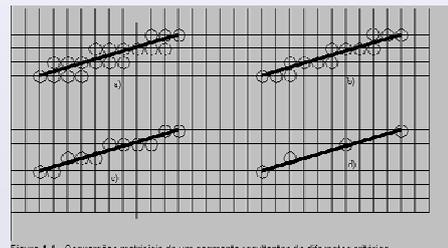


Figura 4.4 - Conversões matriciais de um segmento resultantes de diferentes critérios.

6

Conversão de Segmentos de Reta

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

7

Conversão de Segmentos de Reta

- # Dados pontos extremos em coordenadas do dispositivo:
 - P1(x₁,y₁) (inferior esquerdo)
 - P2(x₂,y₂) (superior direito)
- # Determina quais pixels devem ser "acesos" para gerar na tela uma boa aproximação do segmento de reta ideal

8

Conversão de Segmentos de Reta - Algoritmo DDA

- # Estratégia mais simples
- # Usa equação explícita da reta:
$$y = mx + B$$
$$m = (y_2 - y_1) / (x_2 - x_1) \text{ /*inclinação}$$
$$B = y_1 - m * x_1 \text{ /* intersecção eixo y}$$
$$y = mx + (y_1 - m * x_1) = y_1 + m * (x - x_1)$$

9

Algoritmo DDA

```
{ int x, x1, x2, y1, y2;
  float y, m;
  int valor;

  m = (y2-y1)/(x2-x1);      /* 0 < |m| < 1
  for (x = x1; x <= x2; x++) {
    /* arredonda y */
    y = y1 + m*(x-x1);
    write_pixel (x, round(y), valor);
  }
}
```

- # Cálculo em ponto flutuante: ineficiente

10

Otimização DDA

Na iteração i: $y_i = mx_i + B$
Na iteração i+1:
$$y_{i+1} = mx_{i+1} + B = m(x_i + \Delta x) + B$$
$$= mx_i + m\Delta x + B = y_i + m\Delta x$$

se $\Delta x = 1$, então
$$x_{i+1} = x_i + 1, \text{ e } y_{i+1} = y_i + m$$

Algoritmo incremental!!!

11

Algoritmo DDA

Na forma dada, funciona para segmentos em que $0 < m < 1$

Porque?

12

Exercício

Aplique o algoritmo (e adaptações) para fazer a conversão dos seguintes segmentos de reta:

- P1: (0,1) P2: (5,3)
- P1: (1,1) P2: (3,5)

13

Algoritmo DDA

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

Como tratar o traçado de uma reta com $m > 1$?

14

Algoritmo DDA

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

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

15

Algoritmo DDA

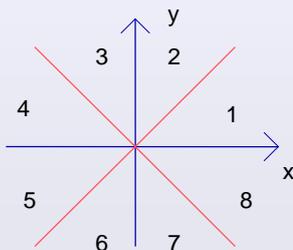
Assume $x_1 < x_2$ e $y_1 < y_2$ (m positivo), processamento da esquerda para a direita

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

16

Octantes do Sistema de Coordenadas Euclidiano



17

Algoritmo de Bresenham (Retas)

Assume $0 < m < 1$

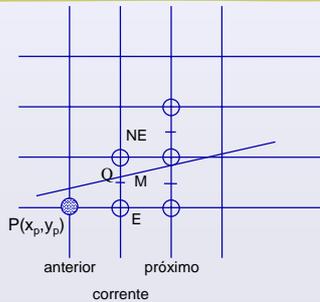
Incrementa x em intervalos unitários, calcula o y correspondente

Abordagem considera as duas possibilidades de escolha de y , decidindo qual a melhor...

Qual é?

18

Algoritmo de Bresenham (Retas)



19

Teste do Ponto Médio (Retas)

Decide com base no valor de uma Variável de Decisão d , computada pela equação implícita

$$F(x,y) = ax + by + c = 0$$

Em termos da equação explícita:

$$y = dy/dx * x + B$$

$$F(x,y) = dy * x - dx * y + B * dx, \text{ i.e.}$$

$$a = dy, b = -dx, c = B * dx$$

20

Subdivisão do espaço pela reta

Dada a reta $F(x,y) = ax + by + c = 0$, e um ponto (x,y) :

Se $F(x,y) = 0$: (x,y) pertence à reta

Se $F(x,y) < 0$: (x,y) está na região acima da reta

Se $F(x,y) > 0$: (x,y) está na região abaixo da reta

Verifique, por exemplo, para $F(x,y) = x - y = 0$

21

Algoritmo do Ponto Médio (Retas)

A cada passo, para determinar o pixel na iteração atual

- escolhe entre 2 alternativas (E ou NE), com base no sinal da variável d calculada na iteração anterior

Se $F(M) = d < 0$, E mais próximo (M está acima da reta ideal)

Se $F(M) = d = 0$, M está na reta ideal

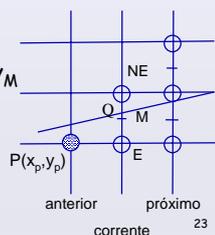
Se $F(M) = d > 0$, NE mais próximo (M está abaixo da reta ideal)

22

Algoritmo do Ponto Médio (Retas)

O que acontece com a localização de M e com o valor de d na iteração seguinte?

- Se escolheu NE, x_M e y_M aumentam em 1
- Se escolheu E, x_M aumenta em 1



23

Algoritmo do Ponto Médio (Retas)

Depois de usar o valor de d para decidir por E ou NE, a variável de decisão pode ser atualizada

- Ao invés de avaliar explicitamente $F(M)$ para determinar o novo valor de d no próximo passo, calcula por diferenças finitas!

24

Algoritmo do Ponto Médio (Retas)

Depois de usar o valor de d para decidir por E ou NE, a variável de decisão pode ser atualizada

- adiciona o incremento adequado, ΔE ou ΔNE , ao valor anterior

Isso evita que a equação tenha que ser computada explicitamente para obter $F(M)$!!

25

Algoritmo do Ponto Médio (Retas)

```
void MidPointLine (int x1,int x2, int y1,int y2)
int dx,dy, incE, incNE, d, x, y;
int valor;
{
  dx= x2-x1; dy= y2-y1;
  d= 2*dy-dx; /* fator de decisão: valor inicial */
  incE= 2*dy; /* Incremento para E */
  incNE=2*(dy-dx); /* Incremento para NE */
  x= x1; y= y1;
  write_Pixel (x,y,valor); /* Pinta pixel inicial */
```

26

Algoritmo do Ponto Médio (Retas)

```
while (x < x2) {
  if (d <= 0) { /* Escolhe E */
    d= d+incE; }
  else { /* Escolhe NE */
    d= d+incNE;
    y++;} /* maior que 45° */
  x++;
  write_pixel (x, y, valor);
} /* fim do while */
} /* fim do algoritmo */
```

27

Algoritmo do Ponto Médio (Retas)

Exercício: aplique o algoritmo para

- P1: (5,8) P2: (9,11)

- Qual o valor inicial de d ?
- Quantos passos (iterações) são executados?
- Quais os valores de x , y e d a cada passo?

28

Solução para P1: (5,8) P2: (9,11)

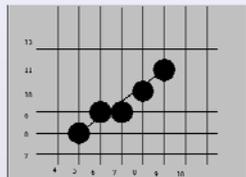


Figura 4.0 - Representação inicial do segmento de reta entre o ponto (5,0) e (9,11), pelo algoritmo de Bresenham.

29

Traçado de Circunferências

Circunf. com centro na origem e raio R :

$$x^2 + y^2 = R^2$$

$$y = \pm \sqrt{R^2 - x^2}$$

$$x = R \cdot \cos \theta, y = R \cdot \sin \theta$$

- Partindo de P1: (0,R), porque não usar diretamente a equação explícita acima para traçar um arco de $\frac{1}{4}$ da circunf.?
- Porque não usar a forma paramétrica?

30

Algoritmo do Ponto Médio (Circunferência)

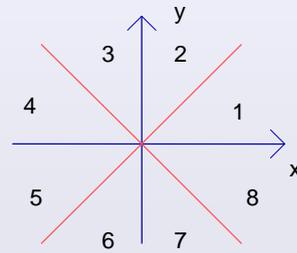
Traçado de arco de 45° no segundo octante, de $(0,R)$ a $x = y = R/\sqrt{2}$

O restante da curva pode ser obtido por simetria

- Se o ponto (x,y) pertence à circunferência, outros 7 pontos sobre ela podem ser obtidos de maneira trivial...

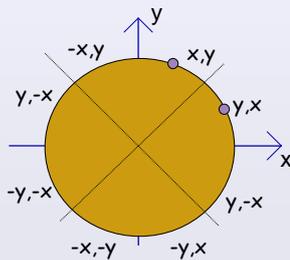
31

Octantes do Sistema de Coordenadas Euclidiano



32

Simetria de Ordem 8



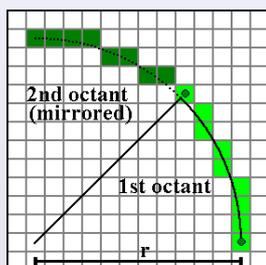
33

Simetria de Ordem 8

```
void CirclePoints (int x, int y, int value)
{
    write_pixel( x,y,value);
    write_pixel( x,-y,value);
    write_pixel(-x,y,value);
    write_pixel(-x,-y,value);
    write_pixel( y,x,value);
    write_pixel( y,-x,value);
    write_pixel(-y,x,value);
    write_pixel(-y,-x,value);
}
```

34

Simetria



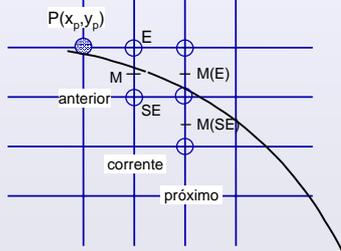
35

Algoritmo do Ponto Médio (Circunferência)

- Mesmo princípio de escolher entre os 2 pixels possíveis
- No caso, escolhe entre E ou SE, analisando a posição do ponto médio M em relação à circunferência
 - Se $d < 0$, M está dentro da circ. ideal (E)
 - Se $d = 0$, M está na circunf.
 - Se $d > 0$, M está fora da circ. ideal (SE)

36

Algoritmo do Ponto Médio (Circunferência)



37

Teste do Ponto Médio (Circunferência)

Variável de Decisão d , na iteração atual, i.e., determinando (x_{p+1}, y_{p+1})

$$F(M) = F(x_p + 1, y_p - \frac{1}{2}) = (x_p + 1)^2 + (y_p - \frac{1}{2})^2 - R^2 = d$$

38

Algoritmo do Ponto Médio (Circunferências)

Para determinar o pixel na iteração corrente

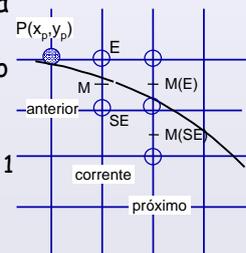
- escolhe entre as duas alternativas, com base no sinal da variável d , calculada na iteração anterior
- atualiza a variável d com o valor correspondente ao pixel escolhido

39

Algoritmo do Ponto Médio (Circunferências)

O que acontece com a localização de M e o valor de d na iteração seguinte?

- Se escolheu SE, x_M aumenta 1, y_M diminui 1
- Se escolheu E, x_M aumenta 1



40

Algoritmo do Ponto Médio (Circunferências)

- # No algoritmo de traçado de retas a atualização de d requer um incremento constante.
- # No algoritmo de traçado de circunferências (v. 1) a atualização de d requer o cálculo de uma função linear do ponto de avaliação (x_p, y_p)
- # Valor inicial de d é dado por $F(1, R-1/2)$, pois $(x_1, y_1) = (0, R)$

41

Algoritmo do Ponto Médio (Circunferências) (v. 1)

```
void MidpointCircle (int raio, int valor)
/* Assume circunferência centrada na origem */
{ int x, y;
  float d;

  /* Inicialização das variáveis */
  x= 0;  y= raio;
  d= 5/4 - raio; /* variável de decisão: valor inicial */
  CirclePoints (x, y, valor);
```

42

Algoritmo do Ponto Médio (Circunferências) (V. 1)

```

while (y > x) {
  if (d < 0) {          /* Seleciona E */
    d = d + 2*x + 3;
    x++;}
  else {               /* Seleciona SE */
    d = d + 2*(x-y) + 5;
    x++; y--;}
  CirclePoints (x, y, valor);
} /* Fim while */
} /* Fim MidpointCircle */

```

43

Algoritmo do Ponto Médio (Circunferências)

- # Problema: usa aritmética real para o cálculo de d , devido ao valor inicial
- # Solução: definir $h = d - 1/4$, e substituir d por $h + 1/4$ no código
- inicialização passa a ser $h = 1 - R$, e a comparação $d < 0$ torna-se $h < -1/4$, ou simplesmente $h < 0$ (h é inteiro!)
- # No algoritmo, chamamos novamente h de d !!

44

Algoritmo do Ponto Médio (Circunferências) (v. 2)

```

void MidpointCircle (int raio, int valor)
/* Assume circunferência centrada na origem */
/* Utiliza apenas aritmética INTEIRA */
{ int x, y, d;

  /* Inicialização das variáveis */
  x = 0; y = raio;
  d = 1 - raio;
  CirclePoints (x, y, valor);
}

```

45

Algoritmo do Ponto Médio (Circunferências) (v. 2)

```

while (y > x) {
  if (d < 0) {          /* Seleciona E */
    d = d + 2*x + 3;
    x++;}
  else {               /* Seleciona SE */
    d = d + 2*(x-y) + 5;
    x++; y--;}
  CirclePoints (x, y, valor);
} /* Fim while */
} /* Fim MidpointCircle */

```

46

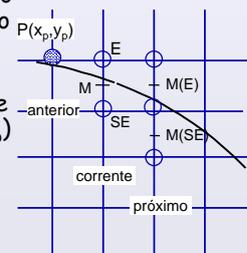
Algoritmo do Ponto Médio (Circunferências) (v. Final)

- # Desempenho pode ser melhorado substituindo a computação direta da variável d pelo cálculo incremental

47

Algoritmo do Ponto Médio (Circunferências) (v. Final)

- # Se escolhe E na iteração atual, ponto de avaliação move de (x_p, y_p) para $(x_p + 1, y_p)$
- # Se escolhe SE, ponto de avaliação move de (x_p, y_p) para $(x_p + 1, y_p - 1)$
- # ΔE e ΔSE computados a partir do pixel inicial $(0, R)$



48

Algoritmo do Ponto Médio (Circunferências) (v. Final)

- # escolhe pixel corrente com base no sinal da variável d calculada na iteração anterior;
- # atualiza variável d somando ΔE ou ΔSE , calculados na iteração anterior;
- # atualiza os Δ s para considerar o movimento para o próximo pixel, utilizando as diferenças constantes previamente calculadas; e
- # move para o próximo pixel.

49

Algoritmo do Ponto Médio (Circunferências) (V. Final)

```
void MidpointCircle (int raio, int valor)
/* Utiliza diferenças parciais de 2º ordem para
  calcular o valor da variável de decisão  $d$  */
/* Assume circunferência centrada na origem */
/* Utiliza apenas aritmética INTEIRA */

{ int x, y, d, deltaE, deltaSE;

  /* Inicialização das variáveis */
  x= 0; y= raio; d= 1 - raio;
  deltaE= 3; deltaSE= -2*raio + 5;
  CirclePoints (x, y, valor);
```

50

Algoritmo do Ponto Médio (Circunferências) (V. Final)

```
while (y > x) {
  if (d < 0) { /* Seleciona E */
    d= d + deltaE;
    deltaE= deltaE + 2; deltaSE= deltaSE + 2;
    x++;}
  else { /* Seleciona SE */
    d= d + deltaSE;
    deltaE=deltaE + 2; deltaSE=deltaSE + 4;
    x++; y--;}
  CirclePoints (x, y, valor)
} /* Fim while */
} /* Fim da rotina MidpointCircle */
```

51

Conversão matricial de elipses

Algoritmo do Ponto Médio: mesmos princípios, com alguns complicadores...

Precisa traçar o arco em um quadrante completo, por exemplo, o primeiro

Tarefa: estudar algoritmo e sua derivação

52

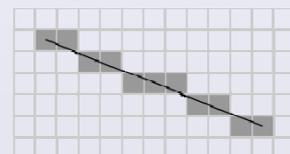
Correção no traçado

- # Distorção devido à razão de aspecto não uniforme do dispositivo
 - Razão de aspecto gráfica: relação entre resolução vertical e resolução horizontal
 - $(\text{resolução vertical})/(\text{resolução horizontal})$
 - Área de exibição não é quadrada, pixels não são quadrados, e densidades de pixels na horizontal e na vertical são diferentes
- # Correção: transformação de escala nos dados da curva a ser traçada

53

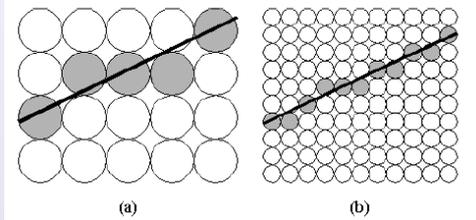
Correção no traçado

- # Antialiasing
 - Primitiva (e.g., segmento de reta) tem espessura (área)



54

Correção no traçado



Aliasing: efeito escada, ou serrilhado. (a) é uma ampliação da região central em (b)

55

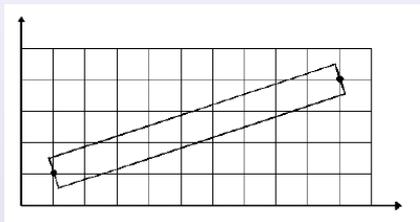
Correção no traçado

Antialiasing: possíveis abordagens

- Amostragem de áreas ponderada
 - Intensidade atribuída ao pixel é proporcional à sua área ocupada pela primitiva

56

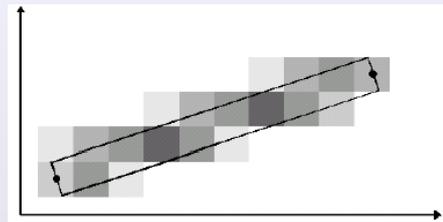
Correção no traçado



Segmento de reta com espessura

57

Correção no traçado



Intensidade do pixel proporcional à área coberta pela 'primitiva ideal' (com espessura)

58

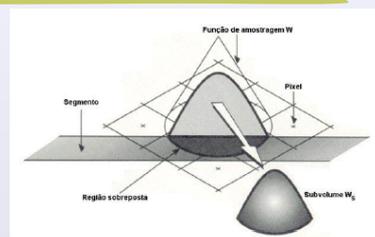
Correção no traçado

Antialiasing: possíveis abordagens

- Amostragem de áreas não ponderada
 - Mesmo princípio, mas considera também a proximidade da área ocupada ao centro do pixel: a intensidade do pixel é proporcional ao 'peso' da área, mas o peso da área varia em função sua distância ao centro do pixel

59

Correção no traçado



Contribuição da área coberta pela primitiva é proporcional à área *multiplicada* por uma função peso, que tem valor máximo no centro do pixel

60

Bibliografia

Livro Hearn & Baker, Cap. 3.5; 3.9;
3.10

Apostila CG em
<http://www.gbdi.icmc.usp.br/?q=node/360>