

Computação Gráfica - Amostragem

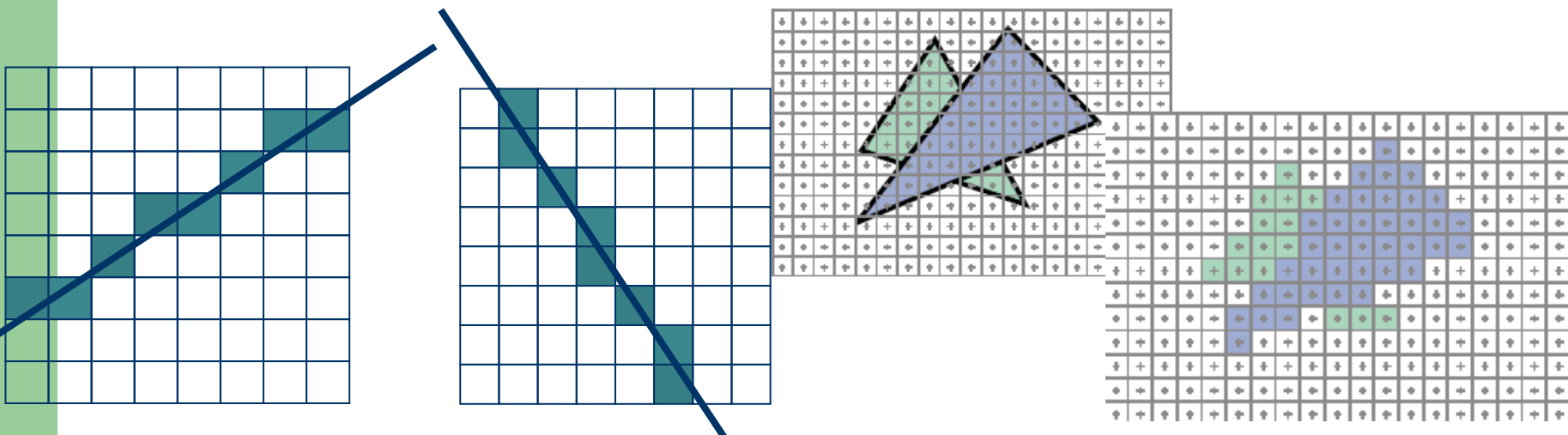
Profa. Mercedes Gonzales
Márquez

Tópicos

- Conceito de Amostragem
- Amostragem ou Rasterização de Segmentos

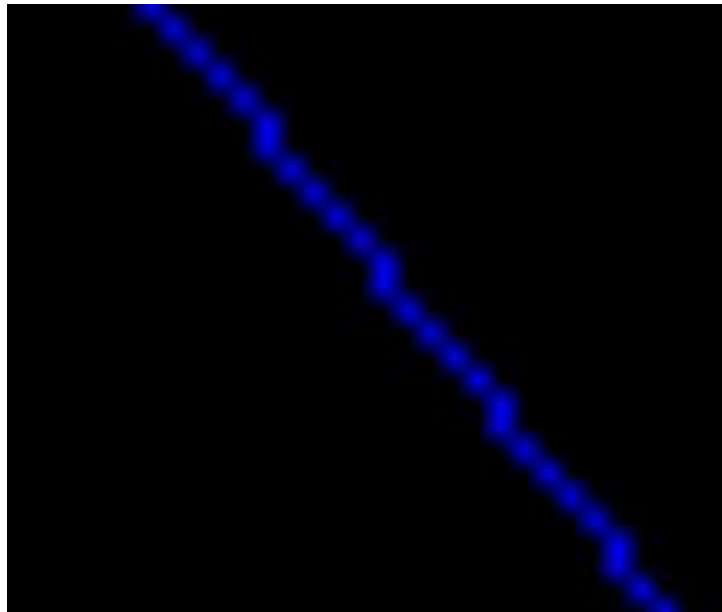
Amostragem - Problema

- As primitivas geométricas são contínuas e a tela é discreta (pixels).
- Solução: Transformar a imagem contínua ou vetorial de uma primitiva em um conjunto de amostras (i, j) , onde i, j são inteiros.



Amostragem - Objetivo

- Transformar a projeção de uma cena 3D em um padrão de matriz de pixels dos dispositivos de saída com um número mínimo possível de artefatos visuais (*aliasing*).



Amostragem de Segmentos

- Entrada: Pontos extremos do segmento
- Saída: Conjunto de pixels (x_i, y_i)
- Algoritmos de Rasterização de Segmentos
 - Algoritmo DDA (*Digital Differential Analyzer* -Analisador do diferencial digital)
 - Algoritmo Bresenham

Rasterização de Segmentos

- Algoritmo DDA (Analisador do diferencial digital – *digital differential analyzer*)
- Temos (x_k, y_k) e queremos obter o subsequente (x_{k+1}, y_{k+1})
- O incremento da coordenada x é trivial, isto é

$$x_{k+1} = x_k + 1$$

- E o incremento da coordenada y é obtido usando a equação da inclinação da reta

$$m = \frac{\Delta y}{\Delta x} = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}.$$

Rasterização de Segmentos

- Dai temos que

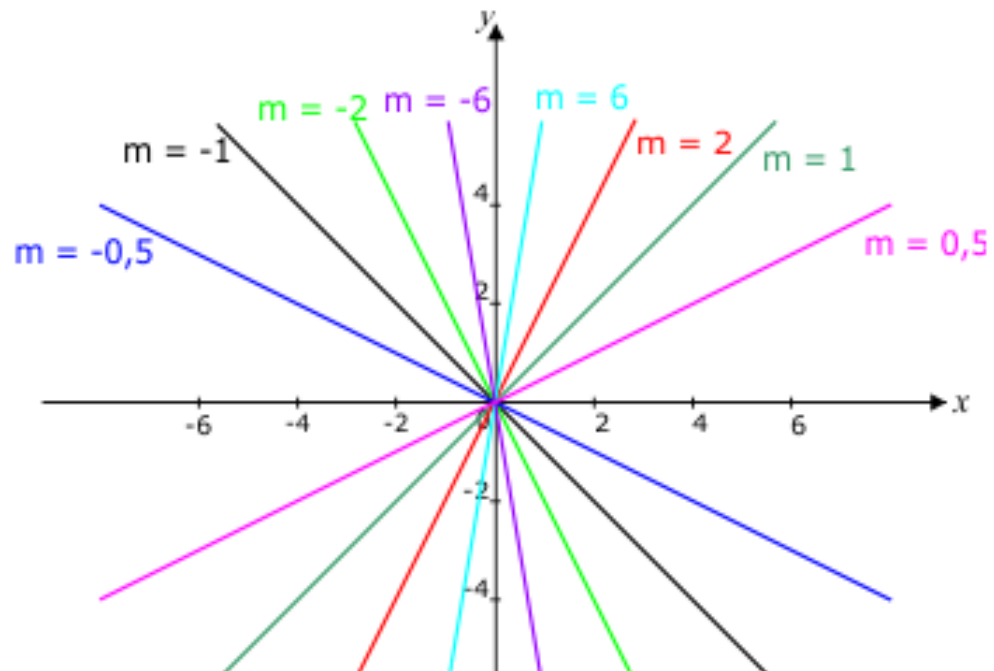
$$x_{k+1} = x_k + 1 \quad e \quad y_{k+1} = y_k + m$$

- Da mesma equação também podemos obter as seguintes relações:

$$y_{k+1} = y_k + 1 \quad e \quad x_{k+1} = x_k + \frac{1}{m}$$

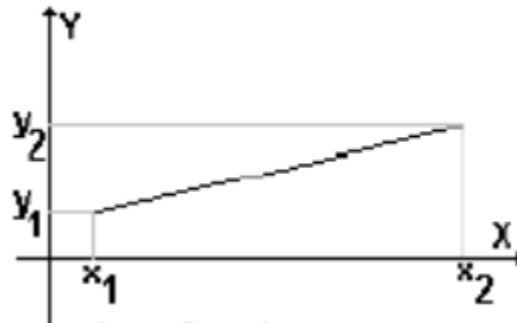
Rasterização de Segmentos

- Observe o comportamento das retas segundo sua inclinação m
 - $|m| < 1$ ($-1 < m < 1$) (formam um ângulo < 45 com o eixo x)
 - $|m| > 1$ ($-1 > m > 1$) (formam um ângulo > 45 com o eixo x)



Rasterização de Segmentos

CASO A) Para $|m| \leq 1$, as coordenadas x crescem mais rapidamente que as coordenadas y .

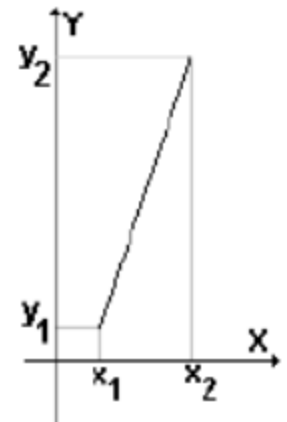


- Portanto, a amostragem é feita incrementando unitariamente na direção x .

$$\begin{aligned}x_{k+1} &= x_k + 1 \\y_{k+1} &= y_k + m\end{aligned} \quad (\text{Form.1})$$

Rasterização de Segmentos

CASO B) Para $|m| > 1$, as coordenadas y crescem mais rapidamente que as coordenadas x .



- Então faz-se incremento unitário na direção y .

$$y_{k+1} = y_k + 1 \quad (\text{Form.2})$$

$$x_{k+1} = x_k + \frac{1}{m}$$

Rasterização de Segmentos

- Para $|m| \leq 1$, // DDA line rasterizer.

```
void DDA(int i1, int j1, int i2, int j2) // Assume  $i2 > i1$ .{  
    float y = j1;  
    float m = float(j2 - j1)/(i2 - i1); // Assume  $-1 \leq m \leq 1$ .  
    glBegin(GL_POINTS);  
    for(int x = i1; x <= i2; x++) {  
        glVertex2i(x, round(y));  
        y += m;  
    }  
    glEnd();  
}
```

Rasterização de Segmentos

EXERCÍCIO:

(a) Acrescente no programa DDA.cpp o trecho de código que considere os segmentos de reta com inclinação $m > 1$ e $m < -1$ ($|m| > 1$).

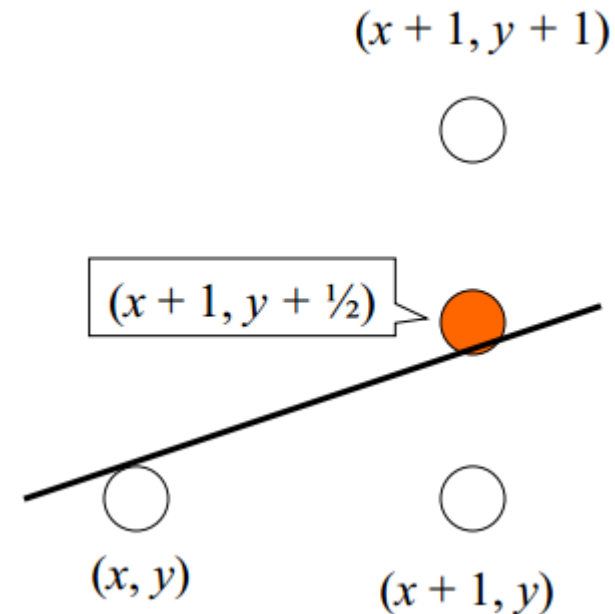
(b) O que acontece quando aplicamos a mesma formulação para todos os valores de m ?

Aplique a form.1 para os casos quando $|m| > 1$ e a form.2 para os casos quando $|m| < 1$. Relate os resultados.

(c) Como o algoritmo trata as retas verticais e horizontais?

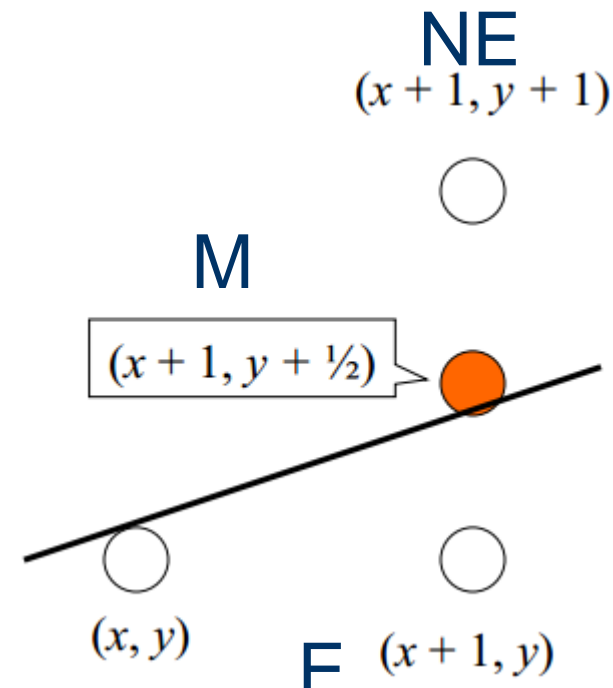
Rasterização de Segmentos

- Algoritmo Bresenham: Idéia básica:
- Temos (x, y) e queremos obter o subsequente, então o algoritmo decidirá entre $(x + 1, y)$ e $(x + 1, y + 1)$
- Decisão requer que se avalie se a linha passa acima ou abaixo do ponto médio $(x + 1, y + \frac{1}{2})$



Rasterização de Segmentos

- Mais especificamente, considere um pixel (x, y) sobre uma reta com inclinação no primeiro octante ($0 < m < 1$)
- O próximo pixel será E (o da direita $(x+1, y)$) se a reta passar abaixo do ponto médio M ou será NE (o da direita acima $(x+1, y+1)$) se a reta passar acima do ponto médio.



Rasterização de Segmentos

- A equação da reta em termos de sua inclinação pode ser escrita como: $y = \frac{\Delta y}{\Delta x} x + B$
- Para determinar um método que calcule de que lado da reta o ponto M se encontra, consideramos sua função implícita, $F(x, y) = ax + by + c = 0$ resultando em $F(x, y) = \Delta y \cdot x - \Delta x \cdot y + \Delta x \cdot B = 0$ onde $a = \Delta y$, $b = -\Delta x$ e $c = \Delta x \cdot B$
- Com isso verificamos que se :
 - $F(x, y) = 0$, o ponto está sobre a linha
 - $F(x, y) > 0$, o ponto está abaixo da linha
 - $F(x, y) < 0$, o ponto está acima da linha

Rasterização de Segmentos

- Em particular para o ponto-médio $M=(x+1,y+1/2)$, basta calcular $F(M) = F(x + 1,y + 1/2) = a(x + 1) + b(y + 1/2) + c$ e verificar o seu sinal.
- Chamamos $F(M)=d$ de **fator de decisão** para a escolha do próximo ponto E ou NE.
- Se $F(M) > 0$, escolhemos o pixel NE
- Se $F(M) < 0$, escolhemos o pixel E
- Se $F(M) = 0$ pode-se escolher qualquer um deles

Rasterização de Segmentos

- Calculamos o fator de decisão inicial d_{start} para o segmento de reta com pontos extremos (x_1, y_1) e (x_2, y_2) .

$$d_{start} = F\left(x_1 + 1, y_1 + \frac{1}{2}\right) = a(x_1 + 1) + b\left(y_1 + \frac{1}{2}\right) + c$$

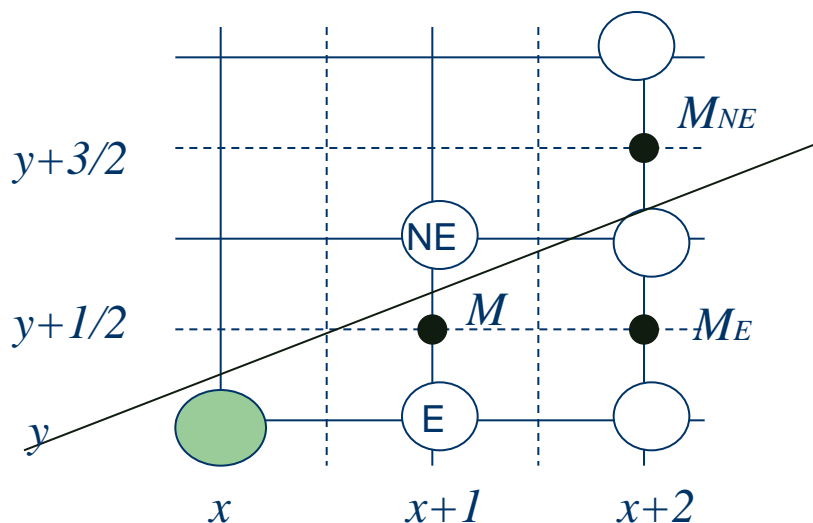
$$d_{start} = ax_1 + a + by_1 + b \cdot \frac{1}{2} + c = \underbrace{ax_1 + by_1 + c}_{F(x_1, y_1)} + a + \frac{1}{2}b$$

$$d_{start} = a + \frac{1}{2}b$$

- Como $F(x_1, y_1)$ está sobre a reta, temos que $F(x_1, y_1) = 0$, daí o resultado acima.

Rasterização de Segmentos

- A determinação da próxima amostra dependerá da determinação do próximo fator de decisão d (d_{new}), o qual será determinado de forma recorrente em função do fator de decisão do passo anterior (d_{old}).



- $d_{old} = F(M)$
- Se $F(M) \leq 0$ então Escolhe-se E e determina-se $d_{new} = F(M_E)$
- Senão Escolhe-se NE e determina-se $d_{new} = F(M_{NE})$

Rasterização de Segmentos

- Cálculo de $d_{new} = F(M_E)$.

$$d_{new} = F\left(x + 2, y + \frac{1}{2}\right) = a(x + 2) + b\left(y + \frac{1}{2}\right) + c$$

$$d_{old} = a(x + 1) + b\left(y + \frac{1}{2}\right) + c$$

Subtraindo d_{old} de d_{new} temos $d_{new} = d_{old} + a$. Isto é a diferença incremental é $difE = a$.

- Cálculo de $d_{new} = F(M_{NE})$.

$$d_{new} = F\left(x + 2, y + 1 + \frac{1}{2}\right) = a(x + 2) + b\left(y + \frac{3}{2}\right) + c$$

$$d_{old} = a(x + 1) + b\left(y + \frac{1}{2}\right) + c$$

Subtraindo d_{old} de d_{new} temos $d_{new} = d_{old} + a + b$. Isto é a diferença incremental é $difNE = a + b$.

Rasterização de Segmentos

- Podemos evitar a divisão por 2 multiplicando a , b e c por 2 (não altera a equação da reta).
- Então temos $d_{start} = 2a + b$, e $difE = 2a$, $difNE = 2a + 2b$

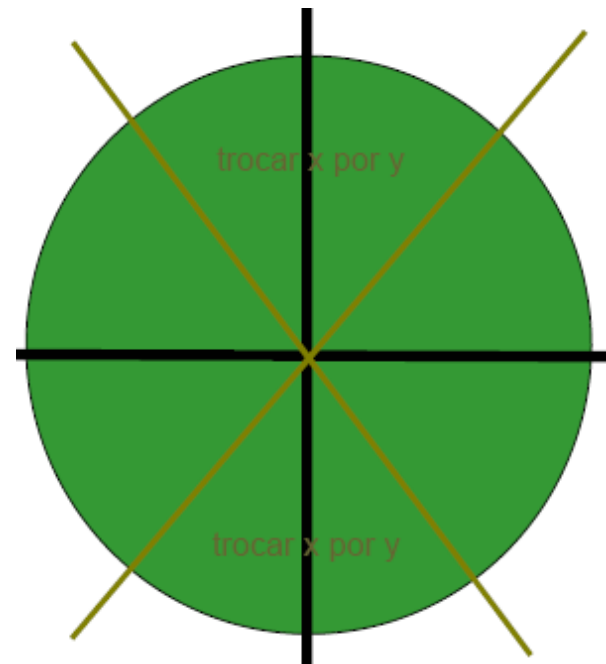
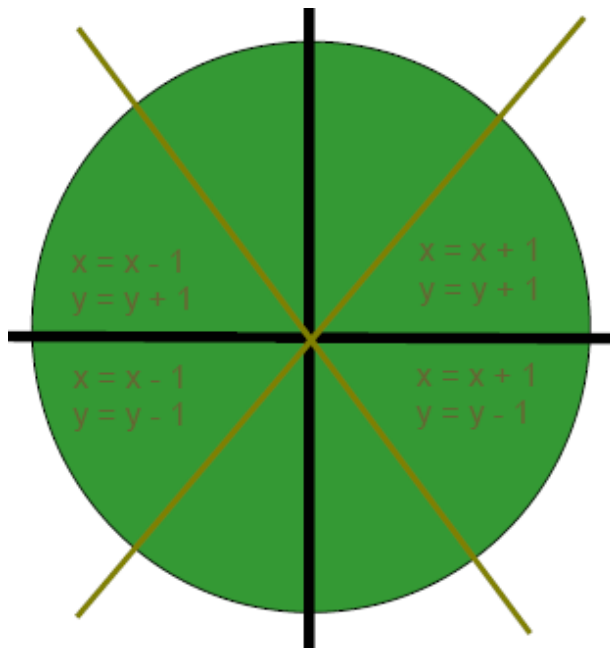
Rasterização de Segmentos

- Algoritmo

```
int x1,x2, y1,y2, dx, dy, incE, incNE, d, x,y;
int valor;
{
    dx=x2-x1;
    dy=y2-y1;
    d=2*dy-dx; /* valor inicial para o fator de decisão */
    incE=2*dy; /* Incr. que move para E */
    incNE=2*(dy-dx); /* Incr. que move para NE */
    x=x1; y=y1;
    write_Pixel (x,y,valor); /* Pinta pixel inicial */
    while (x < x2) {
        if (d <= 0) {
            d=d+incE; /* Escolhe E */
            x=x+1;
        }
        else { /* Escolhe NE */
            d=d+incNE;
            x=x+1; y=y+1; /* pois é maior que 45° */
        }
        write_pixel (x,y,valor);
    } /* fim do while */
}
```

Rasterização de Segmentos

- Outros Octantes

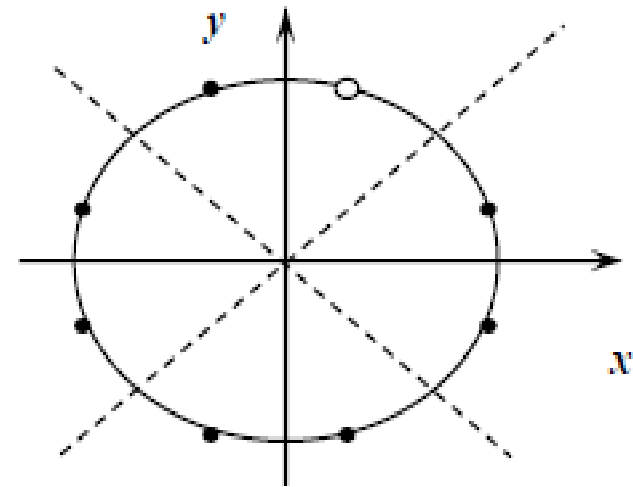


Rasterização de Segmentos

- Exercícios:
 - (1) Implemente o algoritmo
 - (2) Faça uma interface que considere a amostragem de um segmento cujos pontos são fornecidos de forma interativa. Considere a escolha de um dos dois algoritmos estudados.

Rasterização de Cônicas (circunferência)

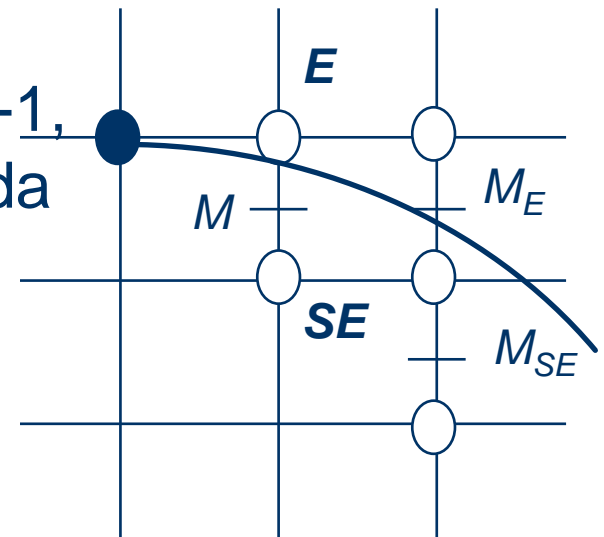
- Algoritmo: Idéia básica:
- Tira-se proveito da simetria da circunferência: basta calcular um arco de circunferência de 45° para obter a circunferência toda.
- Assim como no caso das linhas, a estratégia é selecionar entre 2 pixels aquele que está mais próximo da circunferência, utilizando o sinal da função implícita no ponto intermediário entre os dois possíveis pixels (E ou SE).



Rasterização de Cônicas (circunferência)

- Mais especificamente, considere apenas um arco de 45° da circunferência (o 2° octante) do pixel $(0,R)$ até o $x=y= R/(2)^{1/2}$.
- O próximo pixel será E (o da direita $(x+1, y)$) se o ponto médio M estiver dentro da circunferência ou será SE (o da direita abaixo $(x+1, y-1)$) se M estiver fora da circunferência.
- Equação implícita da circunferência

$$F(x,y) = x^2+y^2-R^2$$



Rasterização de Cônicas (circunferência)

- Para o ponto-médio $M=(x+1,y-1/2)$, basta calcular $F(M) = F(x + 1, y - 1/2) = (x+1)^2 + (y-1/2)^2 - R^2$ e verificar o seu sinal.
- Chamamos também $F(M)=d$ de **fator de decisão** para a escolha do próximo ponto E ou SE.
- Se $F(M) > 0$, escolhemos o pixel SE
- Se $F(M) < 0$, escolhemos o pixel E
- Se $F(M) = 0$ pode-se escolher qualquer um deles

Rasterização de Cônicas (circunferência)

- Calculamos o fator de decisão inicial d_{start} para o próximo ponto médio $(1, R-1/2)$ iniciando do ponto $(0, R)$

$$d_{start} = F(1, R-1/2) = 5/4 - R.$$

Rasterização de circunferência

$$d_{old} = F(x+1, y-1/2) = (x+1)^2 + (y-1/2)^2 - R^2$$

- Cálculo de $d_{new} = F(M_E)$.

$$d_{new} = F(x+2, y-1/2) = d_{old} + (2x+3)$$

- Cálculo de $d_{new} = F(M_{SE})$.

$$d_{new} = F(x+2, y-1/2-1) = d_{old} + (2x-2y+5)$$

Rasterização de circunferência

- Algoritmo

```
void pontomedio(int raio, int valor) {  
    int x = 0;  
    int y = raio;  
    double d = 5/4 - raio;  
    ponto_circulo (x, y, valor);  
    while (y > x) {  
        if (d < 0) /* escolhe E */  
            d += 2.0*x + 3.0;  
        else { /* escolhe SE */  
            d += 2.0*(x - y) + 5;  
            y --;  
        }  
        x ++;
```

```
        ponto_circulo (x, y, valor);  
    } /* while*/  
} /*pontomedio*/  
void ponto_circulo(int x, int y, int valor)  
{  
    writepixel(x, y, valor);  
    writepixel(y, x, valor)  
    writepixel(y, -x, valor)  
    writepixel(x, -y, valor)  
    writepixel(-x, -y, valor)  
    writepixel(-y, -x, valor)  
    writepixel(-y, x, valor)  
    writepixel(-x, y, valor)  
} /*ponto_circulo*/
```

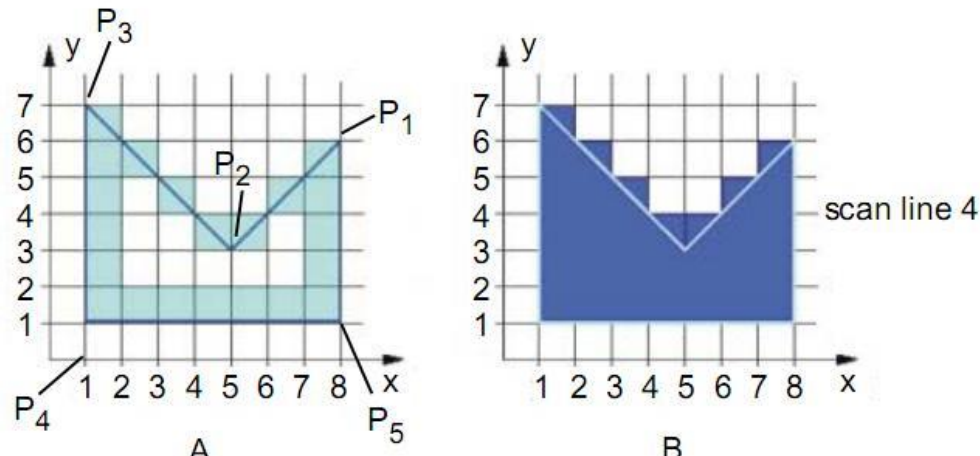
Rasterização de circunferência

- Exercícios:

(1) Acrescente o algoritmo de rasterização de circunferência na interface do trabalho anterior. Considere o centro na origem e o raio fornecido de forma interativa.

Rasterização de Polígonos

- O processo de determinar quais pixels serão desenhados no preenchimento é chamado conversão de varredura (scan conversion). A scan line 4 na figura abaixo, por exemplo, pode ser dividida nas regiões $x < 1$ (fora do polígono), $1 \leq x \leq 4$ (dentro do polígono), $4 < x < 6$ (fora do polígono), $6 \leq x \leq 8$ (dentro do polígono) e $x > 8$ (fora do polígono).

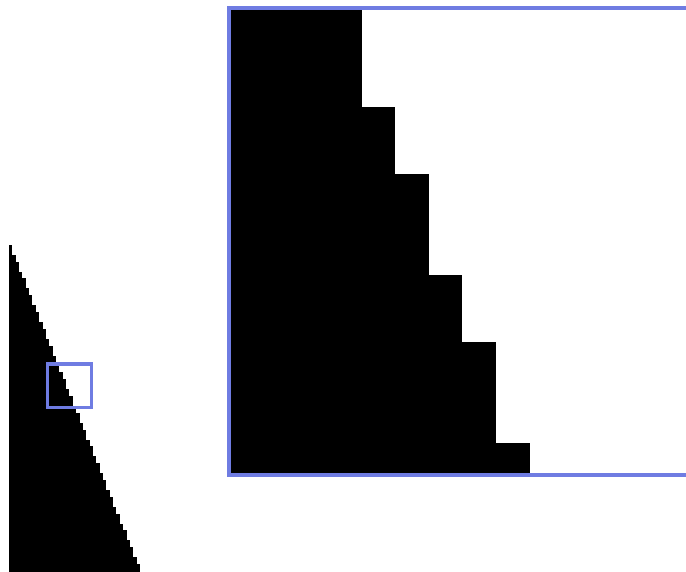


Rasterização de Polígonos

- Detalhes sobre o algoritmo de conversão de linhas ou varredura por linha (scan line) pode ser encontrado no livro no formato eletrônico [Síntese de Imagens: Uma Introdução ao Mundo de Desenho e Pintura dos Sistemas Digitais](#) pag.218.

Rasterização de Polígonos

- Um dos problemas encontrados nos processos de rasterização apresentados é a presença de bordas serrilhadas, *jagged ou stair step pattern* em inglês, nas imagens como mostrado na figura abaixo.



Rasterização de Polígonos

- Como podemos atenuar tais artefatos para gera imagens visualmente mais agradáveis? A idéia se baseia em uma observação simples: quando estivermos muito afastados da tela de exibição, não distinguimos as bordas serrilhadas. Isso decorre da nossa limitada acuidade visual. O que percebemos de fato é o resultado da combinação de cores de vários pixels em torno do pixel da borda quando a imagem estiver muito distante.
- Portanto, uma solução seria emular esta “combinação”, atenuando as fortes transições de luminâncias nas bordas, com uso de mais de uma amostra por pixel.

Rasterização de Polígonos

Efeito Anti Aliasing

