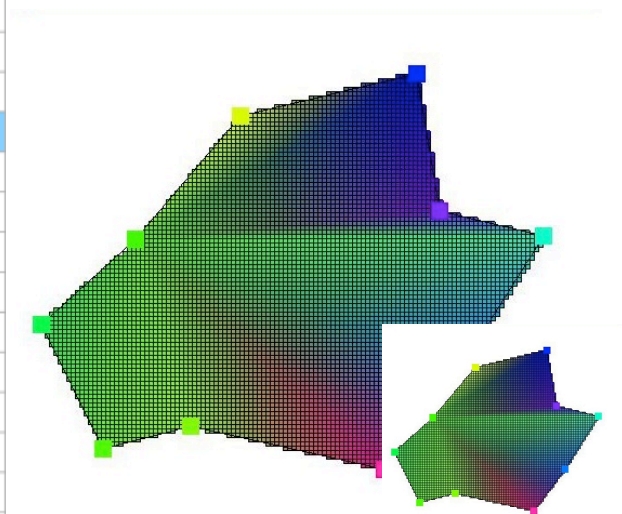
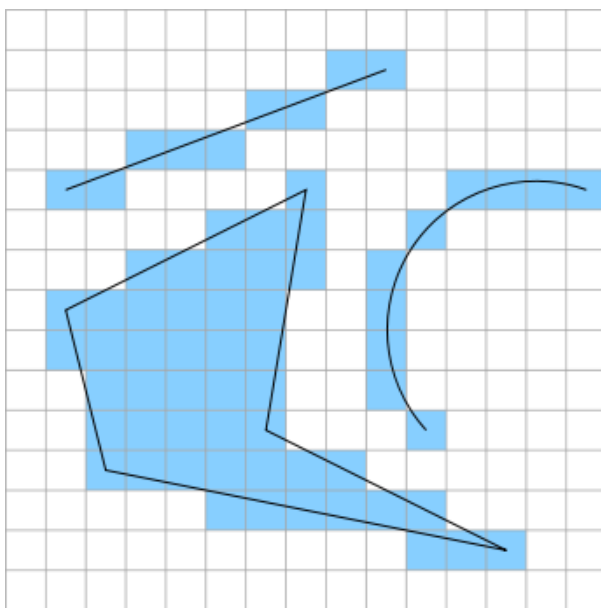


- INF01047 -

Rasterização de linhas no plano

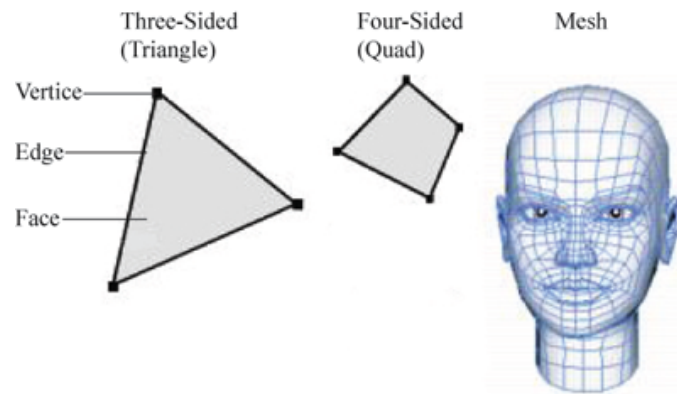
1

Como formamos as imagens na tela?



2

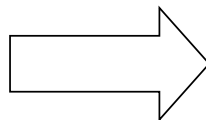
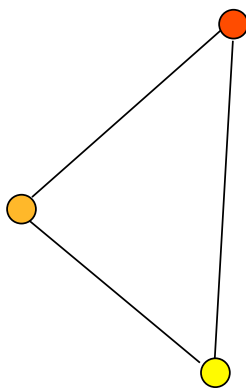
Hierarquia de Rasterização



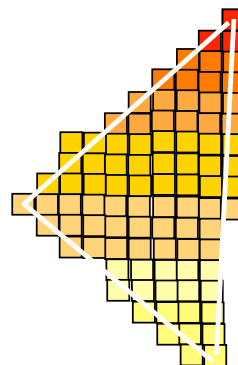
A rasterização de um objeto é feita rasterizando-se os polígonos que o compõem. A rasterização de cada polígono é feita a partir da rasterização das arestas definidoras do mesmo.

Rasterização = *Scan Conversion*

**Representação
Vetorial**



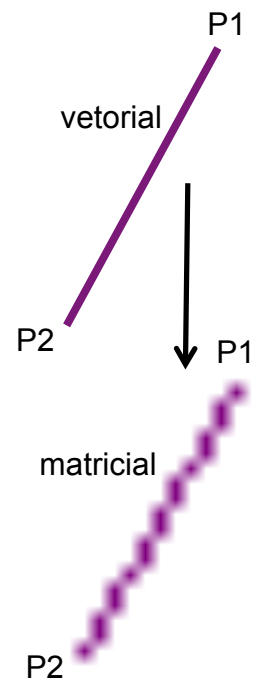
**Visualização
Matricial**



Rasterização de linhas

- Algoritmos de conversão de definição geométrica para pixels
- **Rasterizar = escolher pixels**
- Operação muito frequente
 - Deve ser eficiente!!

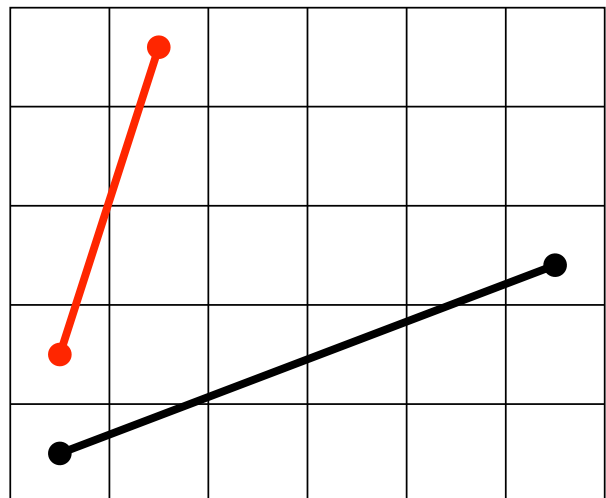
Aritmética de inteiros
Usar somas no lugar de multiplicações



5

Desenho de linhas

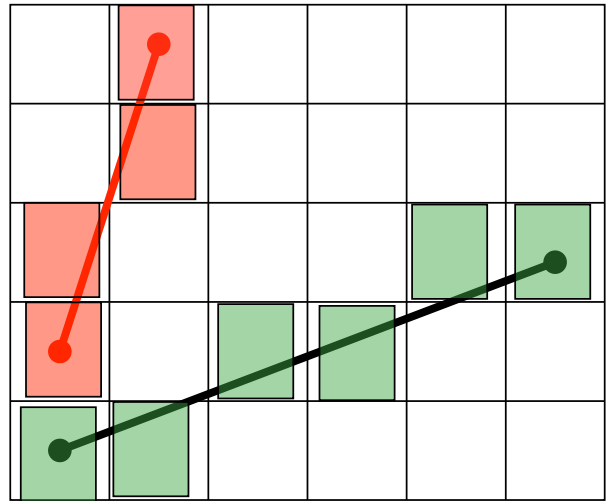
- Sejam as retas definidas por
 - $P1=(1,1)$ e $P2=(6,3)$
 - $P1=(0,1)$ e $P2=(1,4)$
 - Quais são os pixels que devem ser “ligados”?



6

Desenho de linhas

- Sejam as retas definidas por
 - $P1=(1,1)$ e $P2=(6,3)$
 - $P1=(0,1)$ e $P2=(1,4)$
 - Quais são os pixels que devem ser “ligados”?



7

Desenho de linhas: algoritmo básico

- Dados os pontos extremos da linha na tela (já inteiros)

$Pt1 = (x1, y1)$, $Pt2 = (x2, y2)$

- Calcula coeficientes da equação da reta

- 2 operações com inteiros
- 3 operações de ponto flutuante

- Liga todos os pixels que pertencem à reta

$$y = mx + b$$

$$m = \frac{y2 - y1}{x2 - x1}$$

$$b = y1 - m \cdot x1$$

for $x = x1$ to $x2$ (assume incrementos unitários de x)

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

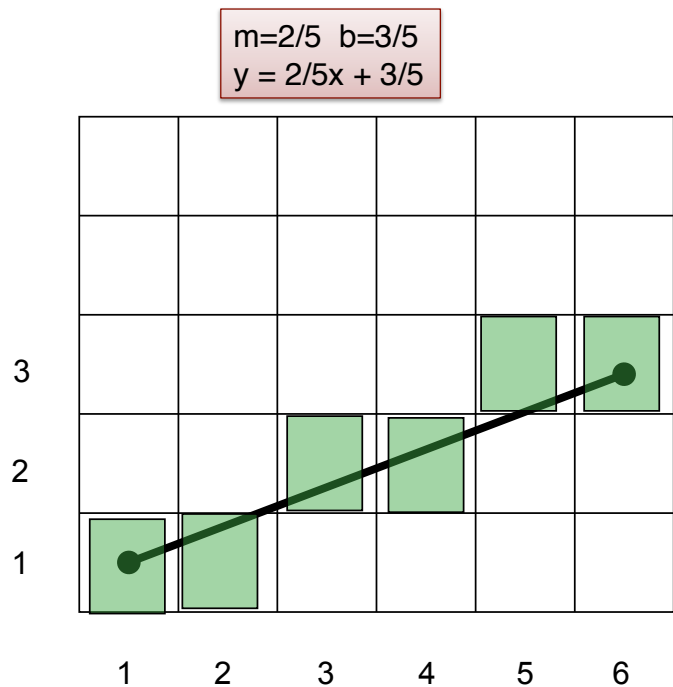
Desenha ponto(x, y) (x, y) inteiros! Como? Arredondamento

8

Para o exemplo anterior

x	y
1	1
2	$2/5 \cdot 2 + 3/5$ $7/5 = 1.4 = 1$
3	$2/5 \cdot 3 + 3/5$ $9/5 = 1.8 = 2$
4	$2/5 \cdot 4 + 3/5$ $11/5 = 2.2 = 2$
5	$2/5 \cdot 5 + 3/5$ $13/5 = 2.6 = 3$
6	$2/5 \cdot 6 + 3/5$ $15/5 = 3$

9



Problemas...

- Assume $0 \leq m \leq 1$
- 2 operações de ponto flutuante por pixel (multiplicação e soma)
- Porque trabalhar com float se pixels tem coordenadas inteiras?

Desenho de linhas: outra opção

- Usando a equação paramétrica da reta

$$P = Pt1 + t * (Pt2 - Pt1), \quad 0 \leq t \leq 1$$

$$dx = x2 - x1 \quad dy = y2 - y1$$

$$x = x1 + t * dx$$

$$y = y1 + t * dy$$

Ex: $Pt1=(1,1)$ $Pt2=(6,3)$

$$x(t) = 1 + t(6-1) = 1 + 5t$$

$$y(t) = 1 + t(3-1) = 1 + 2t$$

Para $t=1/2$ teríamos

$$x(0.5) = 1 + 5/2 = 7/2 = 3.5$$

$$y(0.5) = 1 + 1 = 2$$

Desenho de linhas

- “Pinta” todos os pixels que pertencem à reta

```
y = y1;  
x = x1;  
for t = 0 to 1  
    desenha ponto(x,y);  
    y = y1 + t * dy;  
    x = x1 + t * dx;
```

Continua com 2 multiplicações de ponto flutuante por pixel

Como saber o valor para incrementar em t ?

Desenho de linhas

- Resumo dos problemas
 - Inclinação das linhas
 - Desempenho
 - Número de operações
 - Operações com números reais x inteiros
 - Multiplicações x adições
- Soluções
 - eliminar ou reduzir operações com números reais
 - aproveitar **coerência espacial**
 - similaridade de valores referentes a pixels vizinhos
- Exemplos:
 - DDA: Digital Differential Analyzer
 - Bresenham (1965)

13



Desenho de linhas: DDA

- Dados pontos extremos de um segmento de reta
 - Pt1= (x1, y1)
 - Pt2= (x2, y2)
- Da álgebra elementar
$$m = (y_2 - y_1) / (x_2 - x_1)$$
$$m (x_2 - x_1) = y_2 - y_1$$
$$y_2 = m(x_2 - x_1) + y_1, \quad x \text{ variando de } x_1 \text{ a } x_2$$
- **Qual o incremento em y, se x varia unitariamente?**

14



Resolvendo ...

- $y = m(x - x_1) + y_1$
- $y' = m((x+1) - x_1) + y_1$
- $y' - y = ?$
- $[m((x+1) - x_1) + y_1] - [m(x - x_1) + y_1]$
- $m(x+1) - mx_1 + y_1 - mx + mx_1 - y_1$
- ~~$m(x+1) - mx_1 + y_1 - mx + mx_1 - y_1$~~
- ~~$mx + m - mx$~~

O incremento em y é m!

15

Desenho de linhas: DDA

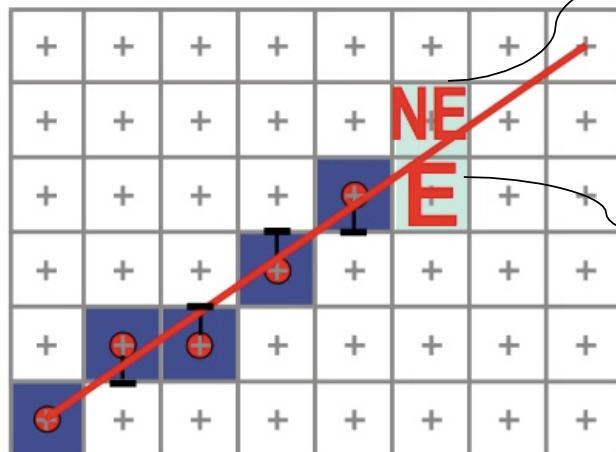
```
/* Interpolate values between start (xa, ya) and end (xb, yb) */
void DDA (int xa, int ya, int xb, int yb)
{
    int x;
    float m = (float) (yb - ya) / (float) (xb - xa); //m is the slope
    float y = ya;
    for (x=xa; x<=xb; x++) {
        output(x, round(y));
        y = y + m;
    }
}
```

- Ainda mantém uma operação de ponto flutuante por pixel
 $y = y + m$
- Necessidade do arredondamento

16

Desenho de linhas: Bresenham (1965)

- Idéia chave: a cada avanço **unitário** em x, é preciso escolher apenas entre 2 pixels



Pixel Northeast (NE)

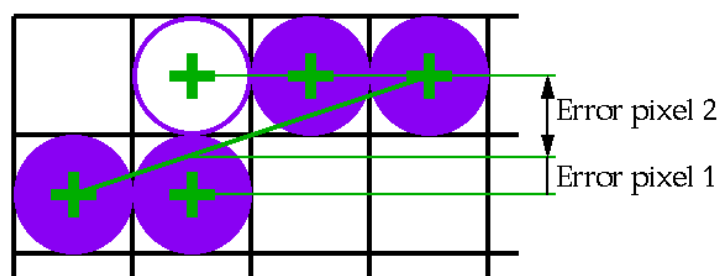
Pixel East (E)

17

$$0 \leq m \leq 1$$

Desenho de linhas: Bresenham

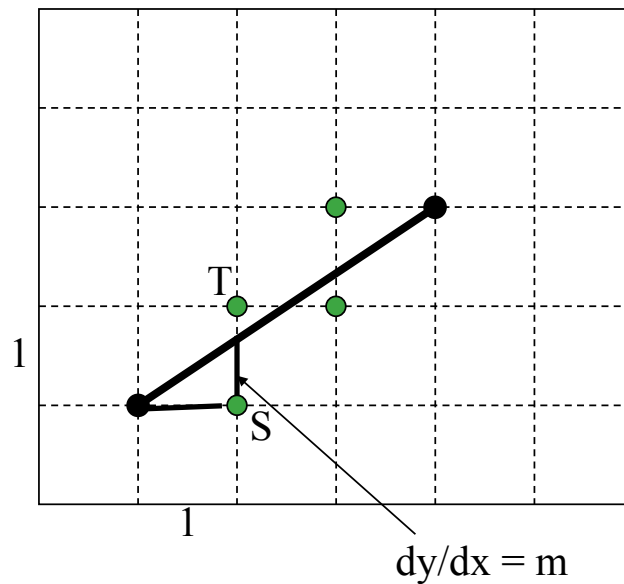
- Para incrementos unitários em x
 - A opção entre incrementar y ou não é determinada em função da distância do segmento de reta até o ponto na grade (raster).
 - Esta distância é chamada de **erro** (diferença para o y ideal, sobre a reta)
 - Minimizar este erro



18

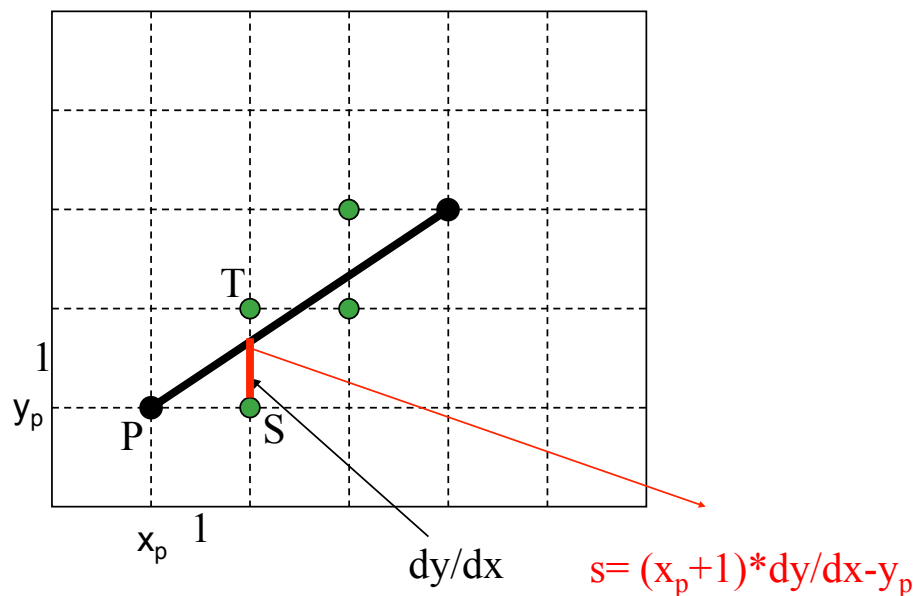
Desenho de linhas: Bresenham

- Na reta real, quando x tem incremento unitário, y é incrementado de m



19

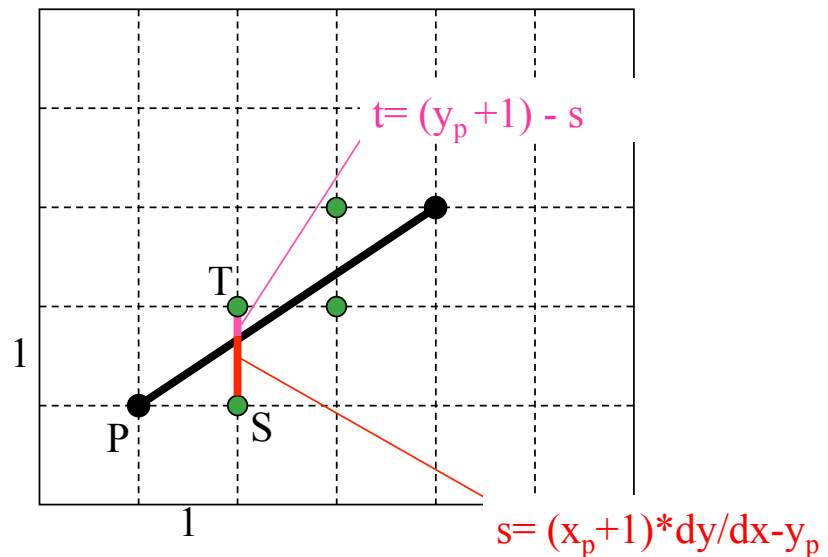
Desenho de linhas: Bresenham



20

Desenho de linhas: Bresenham

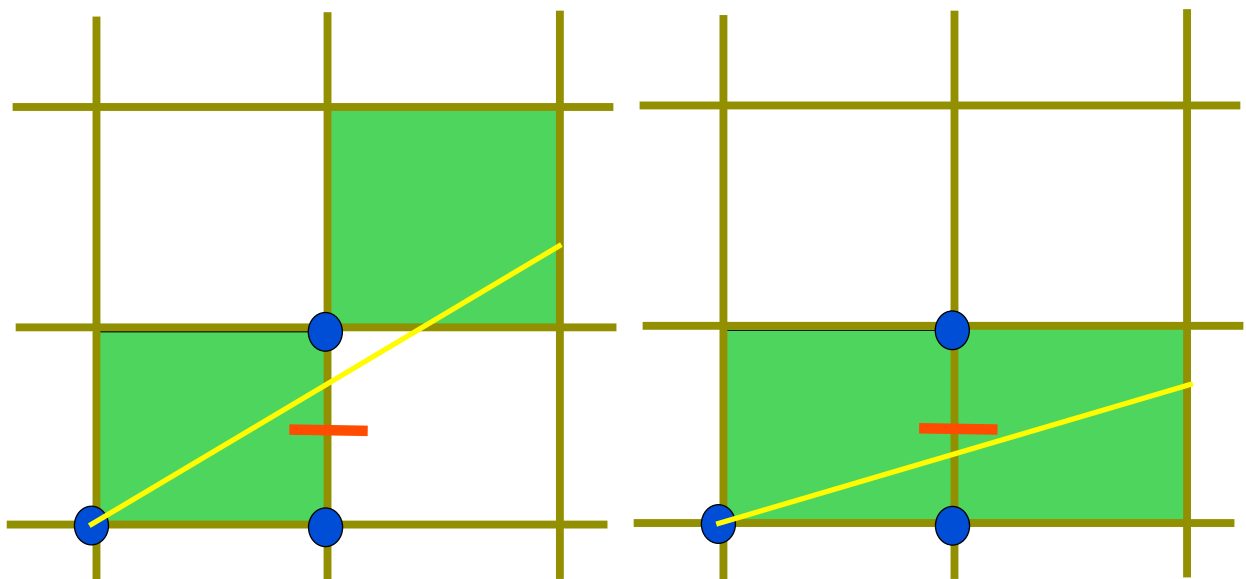
- Calcular **erro** = $s - t$ envolve avaliar números reais!



21

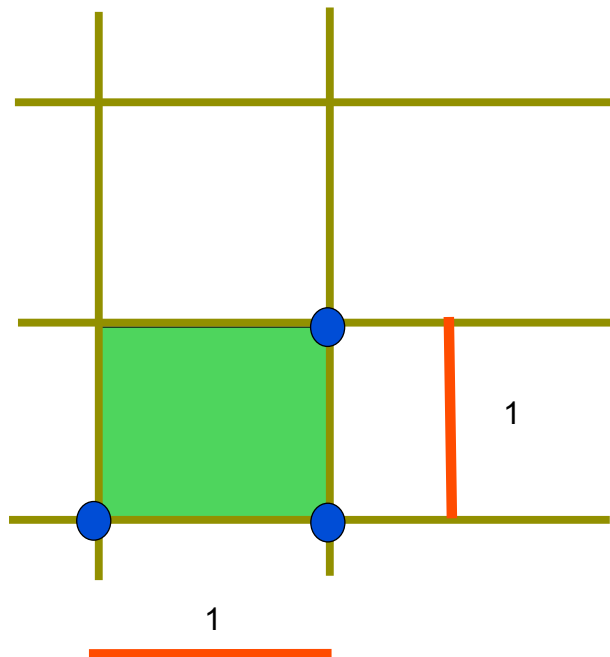
Bresenham

- Determinando o **erro** incrementalmente



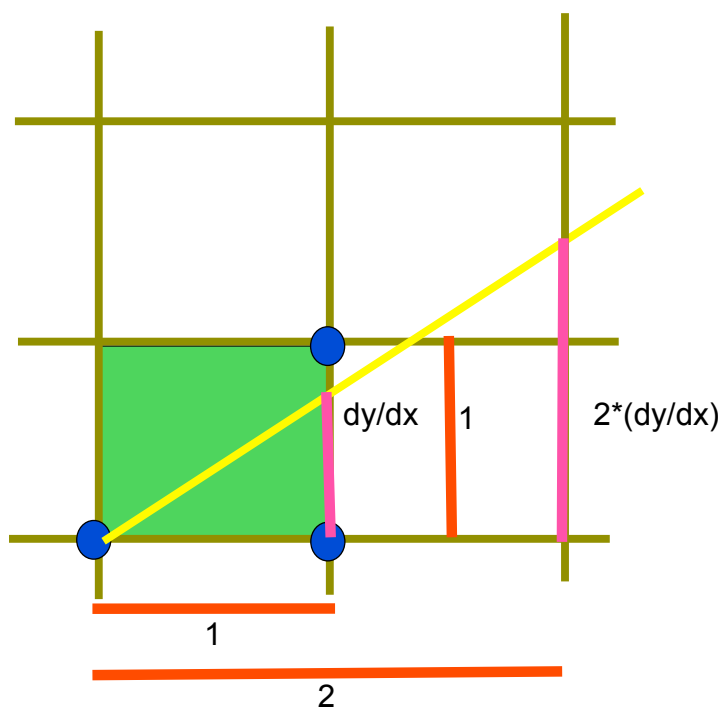
22

Bresenham (Definições)



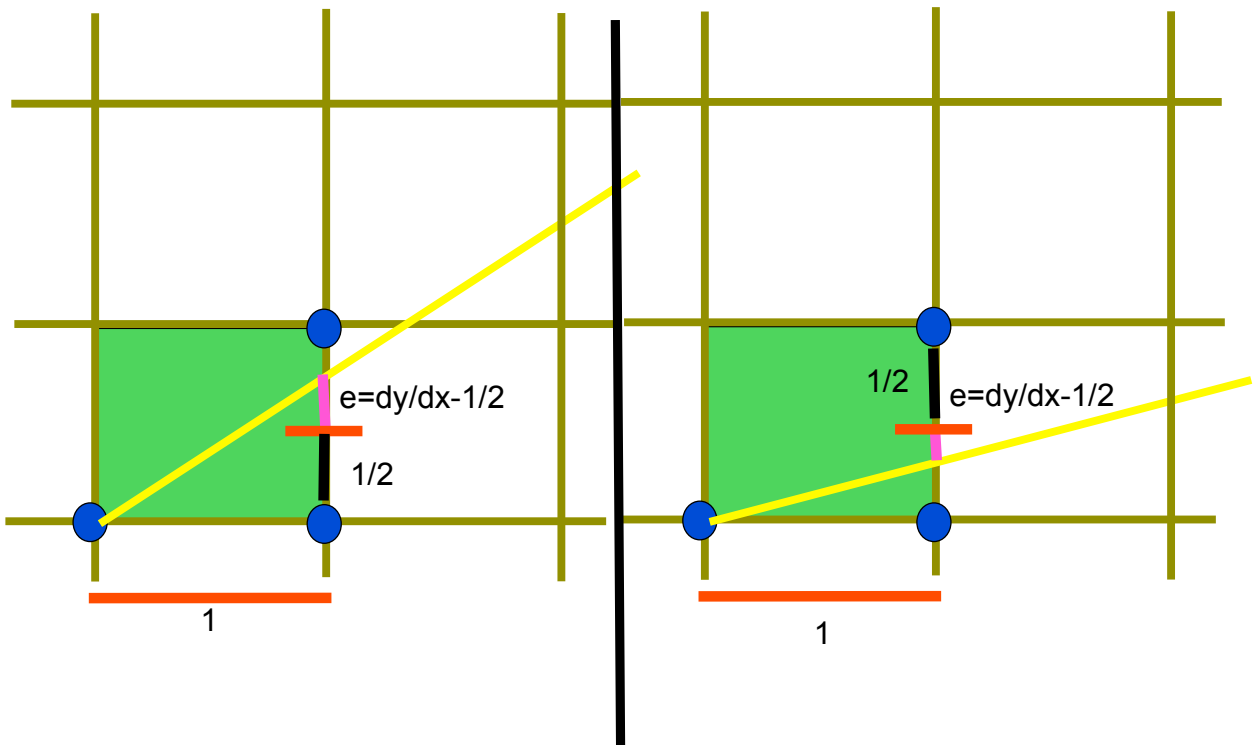
23

Bresenham (Definições)



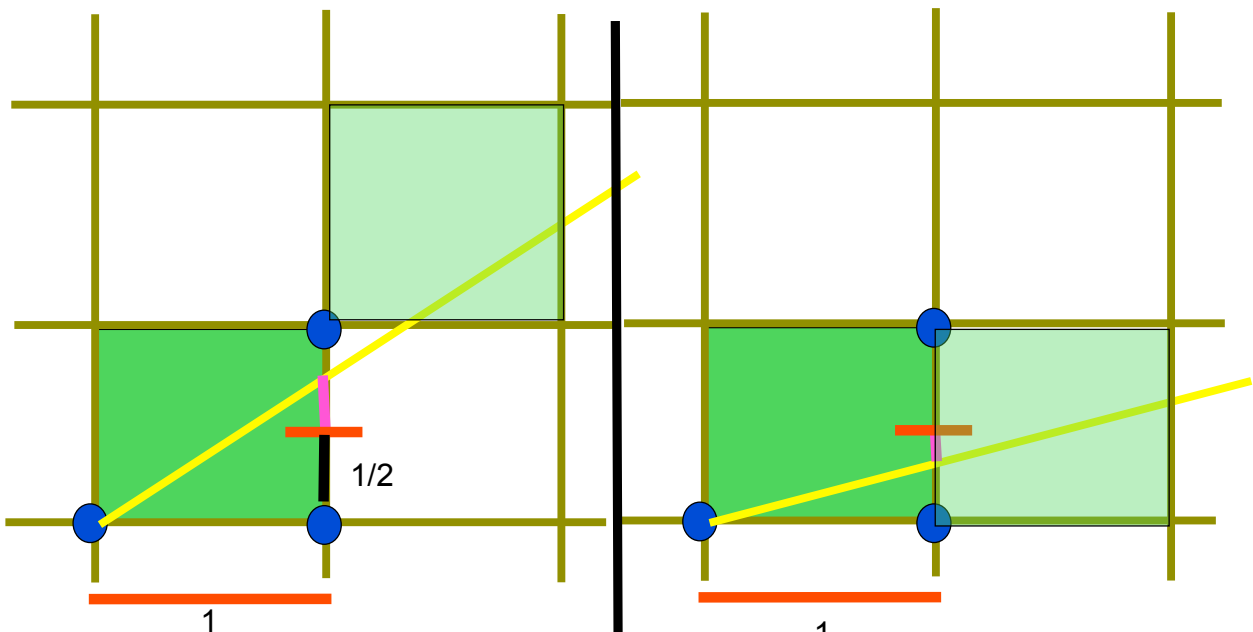
24

Bresenham (Casos Possíveis)



25

Bresenham (Casos Possíveis)



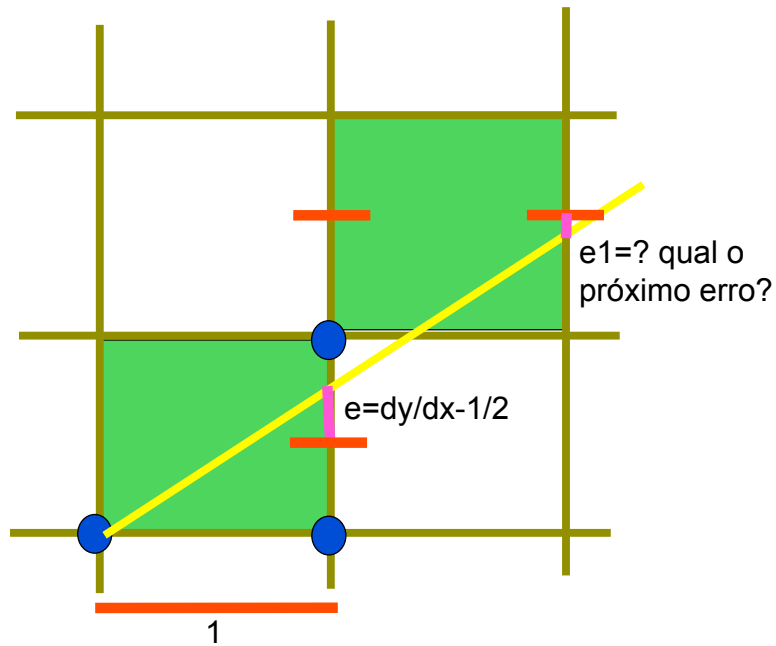
- Se o erro > 0
 - Desenhar o pixel $(x+1, y+1)$

$$e = dy/dx - 1/2$$

- Senão
 - Desenhar $(x+1, y)$
- Erro é atualizado

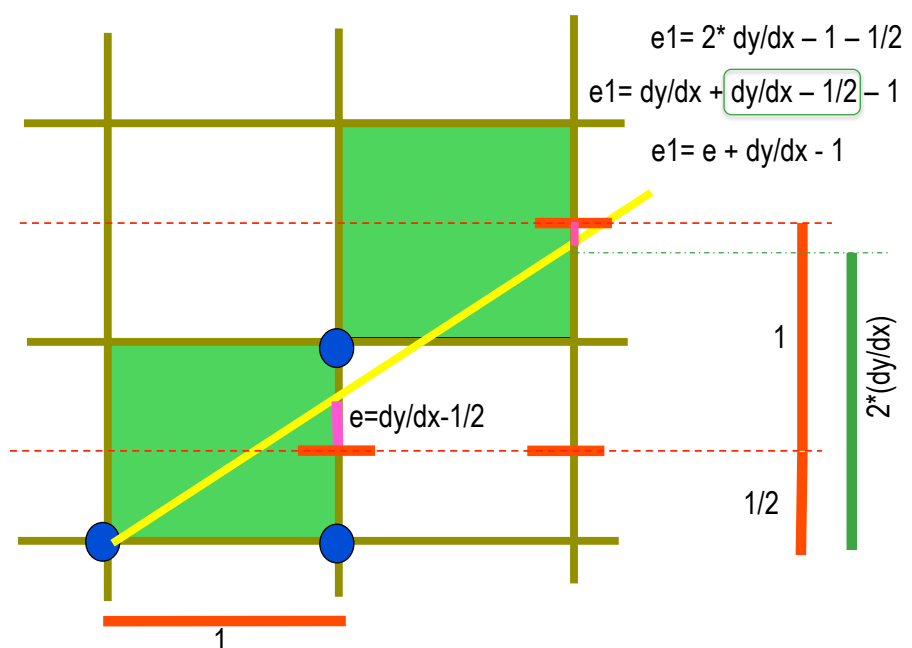
26

Bresenham (Atualização do erro)



27

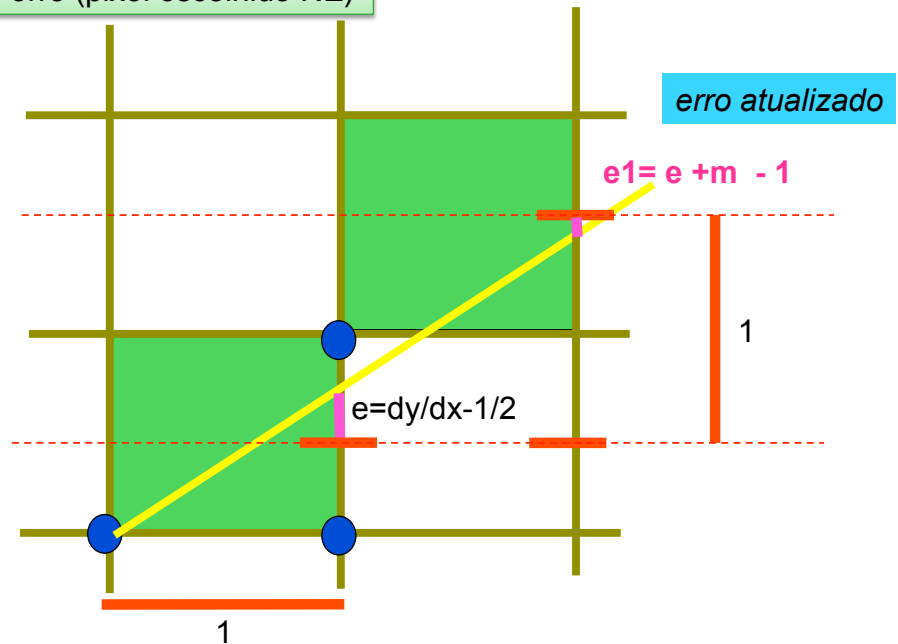
Bresenham (Definições)



28

Bresenham (Definições)

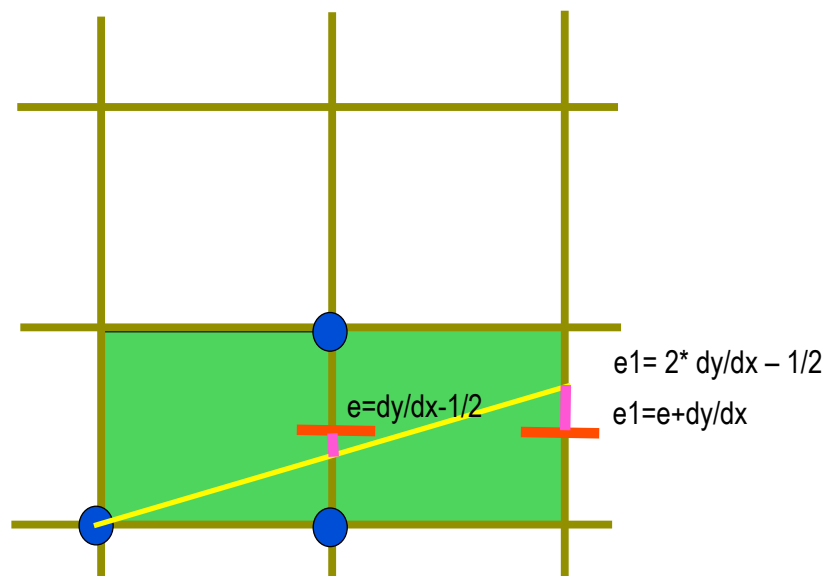
Caso 1 do próximo erro (pixel escolhido NE)



29

Bresenham (Definições)

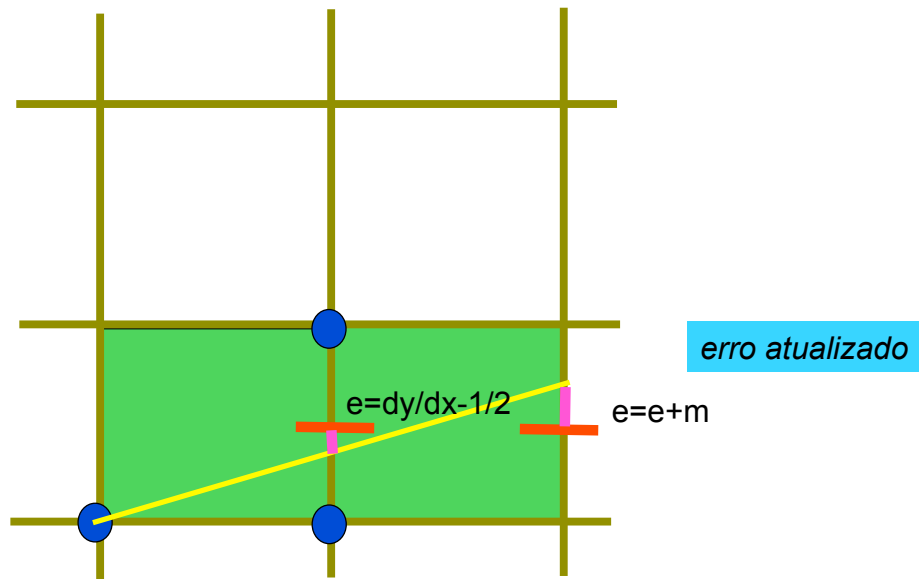
Caso 2 do próximo erro (pixel escolhido E)



30

Bresenham (Definições)

Caso 2 do próximo erro (pixel escolhido E)



31

Algoritmo Bresenham Real

ALGORITMO BRES_REAL (x_0, y_0, x_1, y_1)

1. $x = x_0; y = y_0;$
2. $dy = y_1 - y_0; dx = x_1 - x_0;$
3. $m = dy/dx;$
4. $e = m - 0.5;$
5. FOR $i=0$ TO dx
6. WritePixel(x, y);
7. IF ($e > 0$) {
8. $y = y + 1;$
9. $e = e - 1;$ }
10. $e = e + m;$
11. $x = x + 1;$
12. ENDFOR

32

Substituindo e por $e_i = 2e \cdot dx$

ALGORITMO BRES_REAL (x_0, y_0, x_1, y_1)

1. $x = x_0; y = y_0;$
2. $dy = y_1 - y_0; dx = x_1 - x_0;$
3. $m = dy/dx;$
4. $e = m - 0.5;$
5. FOR $i=0$ TO dx
6. WritePixel(x, y);
7. IF ($e > 0$) {
8. $y = y + 1;$
9. $e = e - 1;$ }
10. $e = e + m;$
11. $x = x + 1;$
12. ENDFOR

$$\rightarrow e_i = 2 * (m - 0.5) * dx$$

$$\rightarrow e_i = (2 * dy/dx - 1) * dx$$

$$\rightarrow e_i = 2 * dy - dx$$

$$\rightarrow 2 * (e - 1) * dx$$

$$\rightarrow 2edx - 2dx$$

$$\rightarrow e_i - 2dx$$

$$\rightarrow 2 * (e + dy/dx) * dx$$

$$\rightarrow 2edx + 2dy$$

$$\rightarrow e_i + 2dy$$

Bresenham inteiro

ALGORITMO BRES_INT (x_0, y_0, x_1, y_1)

1. $x = x_0; y = y_0;$
2. $dy = y_1 - y_0; dx = x_1 - x_0;$
- 3.
4. $e = 2dy - dx;$
5. FOR $i=0$ TO dx
6. WritePixel(x, y);
7. IF ($e > 0$) {
8. $y = y + 1;$
9. $e = e - 2dx;$ }
10. $e = e + 2dy;$
11. $x = x + 1;$
12. ENDFOR

$$\rightarrow e_i = 2 * (m - 0.5) * dx$$

$$\rightarrow e_i = (2 * dy/dx - 1) * dx$$

$$\rightarrow e_i = 2 * dy - dx$$

$$\rightarrow 2 * (e - 1) * dx$$

$$\rightarrow 2edx - 2dx$$

$$\rightarrow e_i - 2dx$$

$$\rightarrow 2 * (e + dy/dx) * dx$$

$$\rightarrow 2edx + 2dy$$

$$\rightarrow e_i + 2dy$$

Bresenham inteiro

ALGORITMO BRES_INT (x_0, y_0, x_1, y_1)

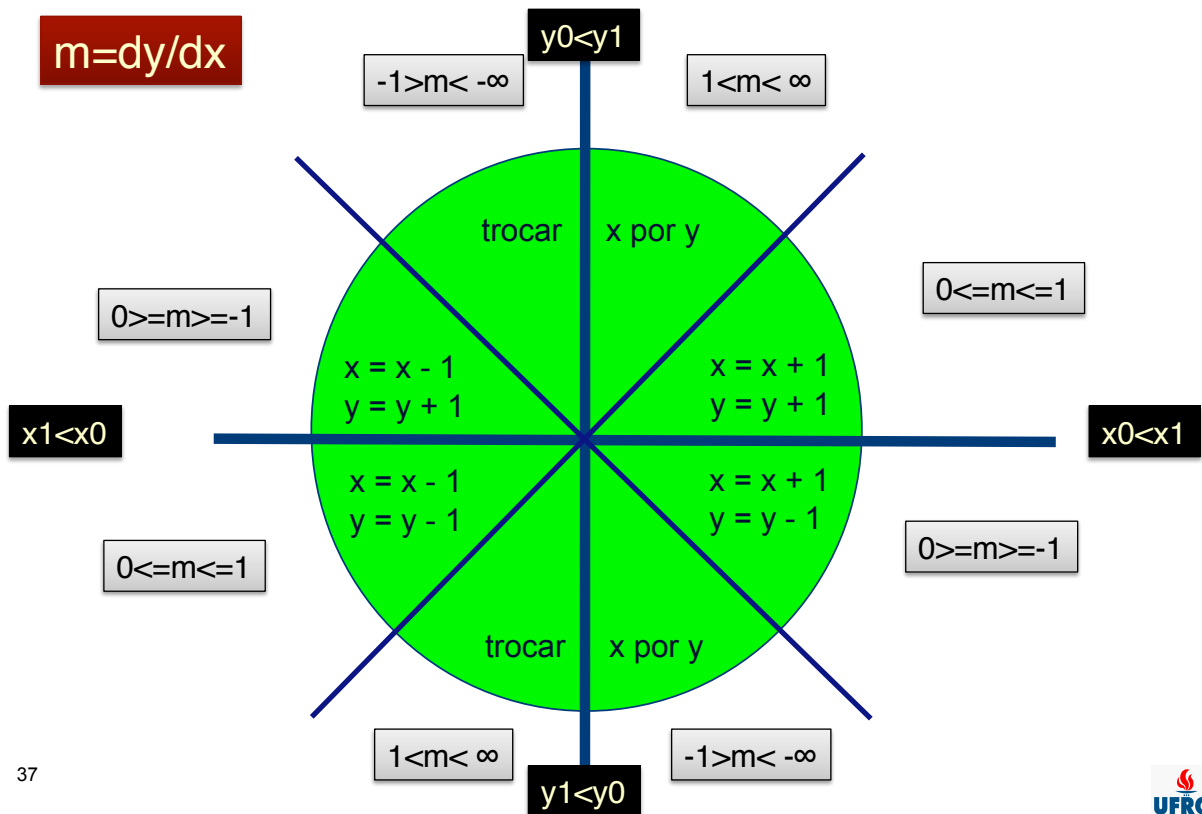
```
1.  x = x0; y = y0;
2.  dy = y1-y0; dx = x1-x0;
3.
4.  e = 2dy-dx;
5.  FOR i=0 TO dx
6.    WritePixel(x, y);
7.    IF (e > 0) {
8.      y = y+1;
9.      e = e - 2dx; }
10.   e = e+2dy;
11.   x = x+1;
12.  ENDFOR
```

- Usa simetria para reduzir a complexidade
 - Linhas em outros quadrantes
- Escolha se limita a dois pixels
- Usa função de erro como critério de escolha
- Trabalha em aritmética de inteiros
- Ótimo para implementar em hardware

Outros tipos de retas

- Caso for horizontal ou vertical tratar como caso especial
- Caso $x_1 > x_2$ inverter ordem dos pontos
- Para as outras inclinações de segmentos extensões do caso principal

Bresenham (outros octantes)



37

Exemplos

- $P_0 = (1, 1)$ $P_1 = (3, 5)$

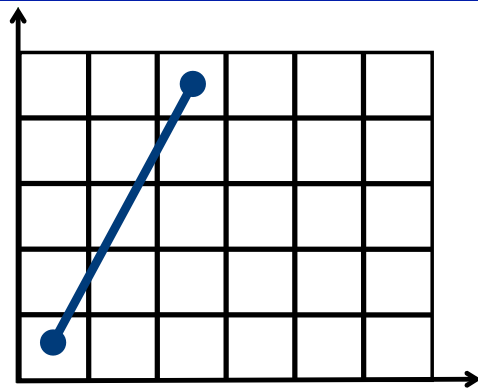
$$m = (5 - 1) / (3 - 1)$$

$$= 4 / 2$$

$$= 2$$

- $y_0 < y_1$

2o. octante



- $P_0 = (-1, -1)$ $P_1 = (-5, -3)$

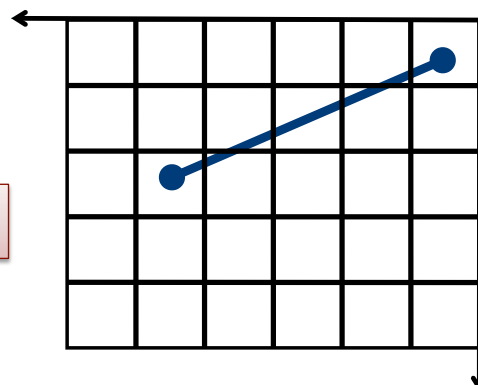
$$m = (-3 + 1) / (-5 + 1)$$

$$= -2 / -4$$

$$= 0.5$$

- $x_1 < x_0$

5o. octante



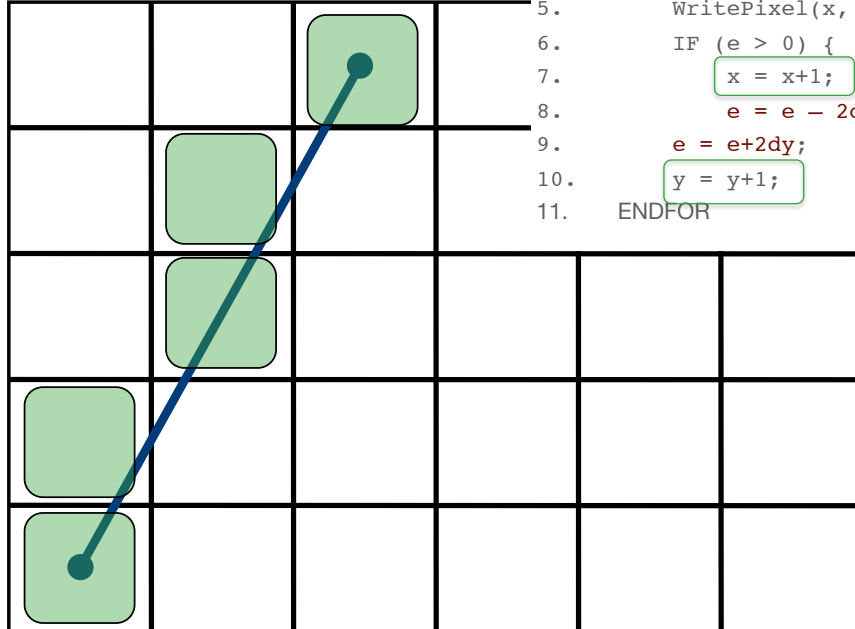
38

Bresenham (2o octante)

m=2
y0<y1

Trocar x por y

P0=(1,1) P1=(3,5)



ALGORITMO BRES_INTEIRO (x0, y0, x1, y1)

```
1. x = y0; y = x0;  
2. dy = x1-x0; dx = y1-y0;  
3. e = 2dy - dx;  
4. FOR i=0 TO dx  
5.   WritePixel(x, y);  
6.   IF (e > 0) {  
7.     x = x+1;  
8.     e = e - 2dx; }  
9.   e = e+2dy;  
10.  y = y+1;  
11. ENDFOR
```

39



Outros objetos gráficos...

- Círculos
 - Equação do círculo: $(x-x_c)^2 + (y-y_c)^2 = r^2$
 - Outra opção: forma paramétrica
 - $x = x_c + r \cos \theta$
 - $y = y_c + r \sin \theta$
 - Custo alto computação
 - Desenho não-uniforme do círculo
- Equivalente Bresenham para círculos

40



Outros objetos gráficos...

- Simetria: precisamos rasterizar apenas um octante. Os outros 7 são encontrados por simetria

