Aula 5

Desenho de Linhas Retas
usando Pixels

Desenho de Linhas Retas usando Pixels

- usar a equação da reta:

- $y = ax + b$

Desenho de Linhas Retas usando Pixels

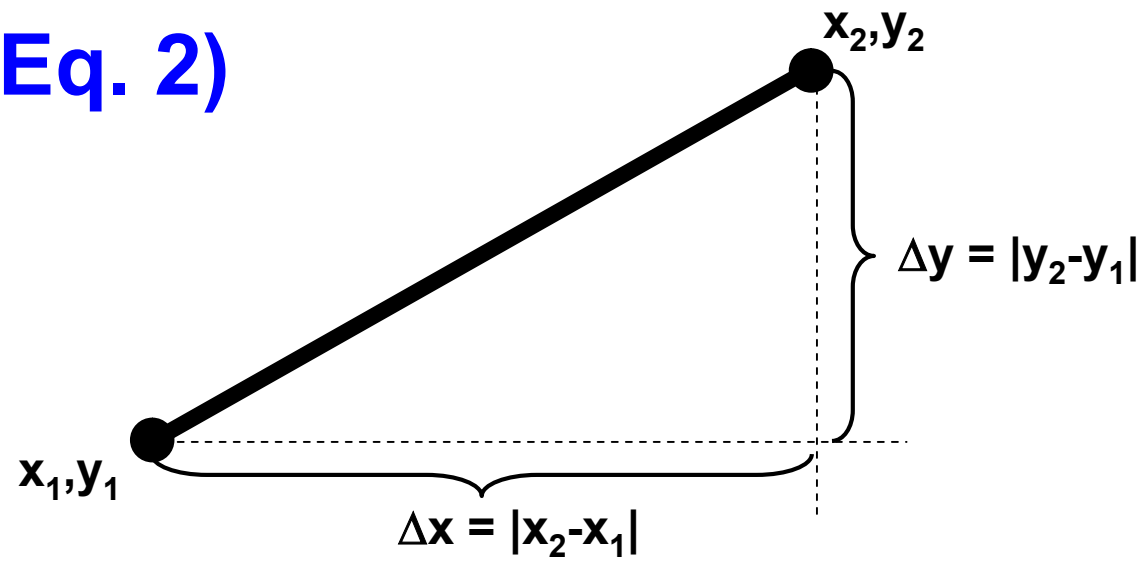
Dados dois pontos x_1,y_1 e x_2,y_2

acha-se $m = (y_2-y_1) / (x_2-x_1)$

em seguida, faz-se $m = (y-y_1) / (x-x_1)$ **(Eq. 1)**

e isola-se o y, ficando

$y = m(x-x_1)+y_1$ **(Eq. 2)**



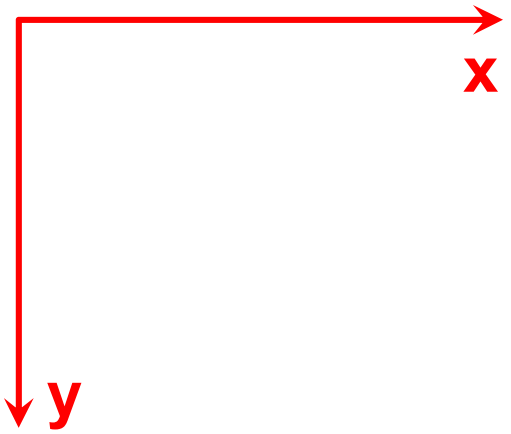
Usando a Equação 2, tem-se

$$y = m.(x-x1) + y1$$

logo, $y = \underbrace{m}_{a}.x \underbrace{- m.x1 + y1}_{b}$

logo, **$y = a.x + b$**

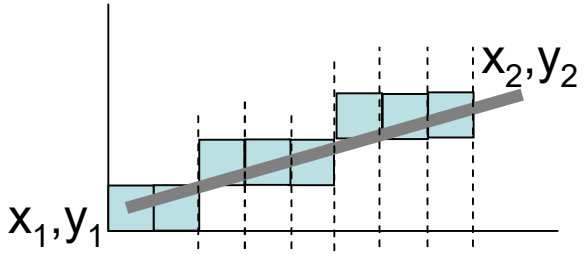
Lembre-se que a tela é assim:



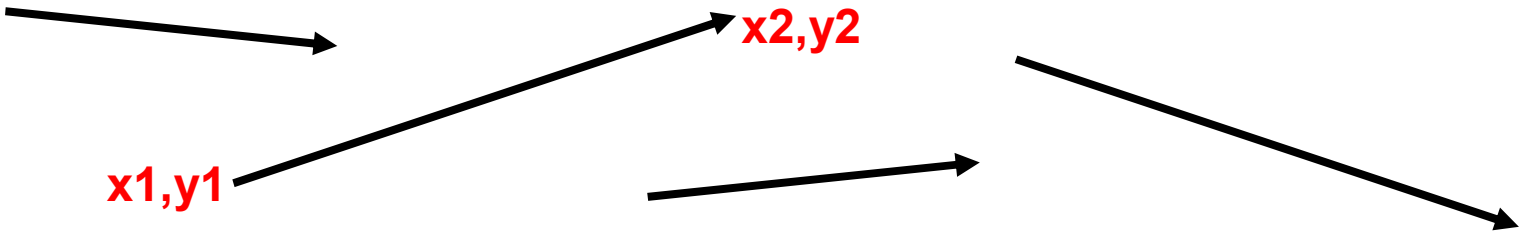
Usando a Eq. 2, o código ficará:

```
for x := x1 to x2 do          // x1 < x2
{ y:= m(x-x1)+y1
  pixel(x,y) := cor
```

para linhas mais horizontais que verticais ($\Delta x > \Delta y$), que vão da esquerda para a direita (**x1<x2**), isto funciona bem

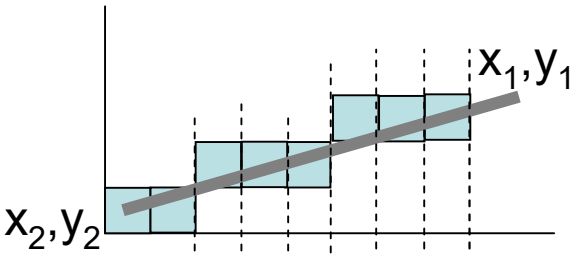


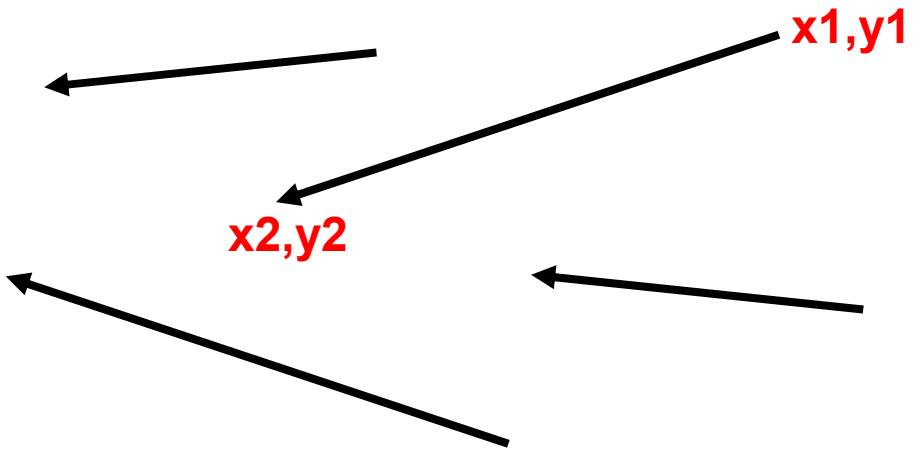
($\Delta x > \Delta y$)



Está avançando o x de um em um e calculando o y

para tratar também linhas que vão da direita para a esquerda (**x2<x1**), deve se inserir um código extra





o código ficará:

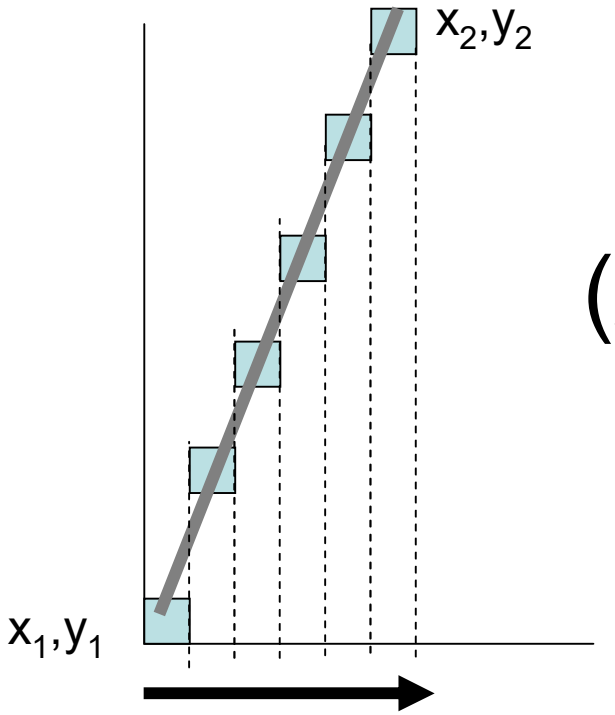
```
if (x1 < x2) then inc =1
else inc = -1
x=x1
y=y1
while (x <> x2) do
   { y = m(x-x1)+y1
     plota(x,y)
     x = x+inc
```

($\Delta x > \Delta y$)

ou o código poderá:

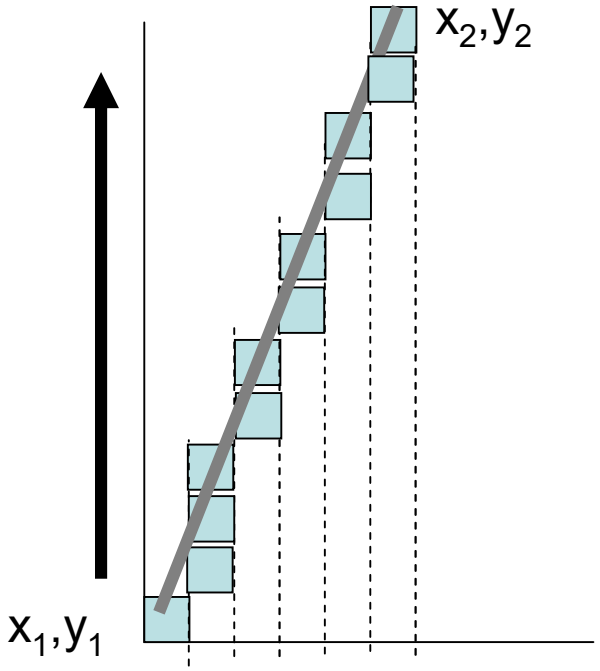
```
for x := x2 to x1 do
  {y:= m(x-x1)+y1
   pixel(x,y) := cor

                    // x2<x1
```

No caso de linhas mais verticais que horizontais (**Δy>Δx**), usando este código, surgirão muitos **buracos** na linha



(Δy>Δx)

Precisa avançar o y de um em um e calcular o x



Não pode avançar o x de um em um e calcular o y

(Eq. 1)

assim, a partir de m = (y-y1)/(x-x1)

isola-se o x, ficando:

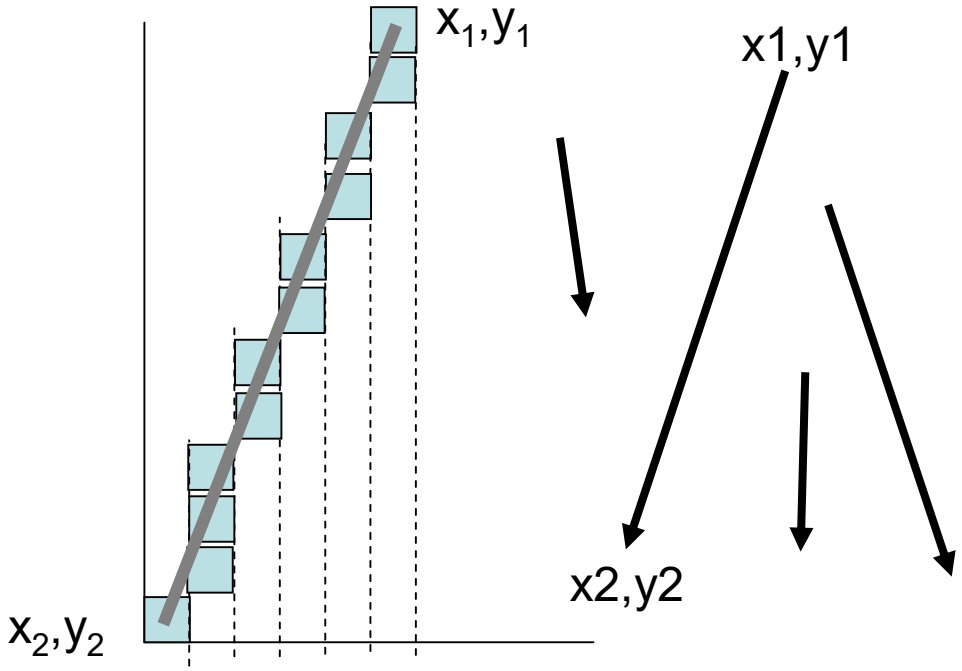
x = (y – y1)/m + x1

o código ficará:

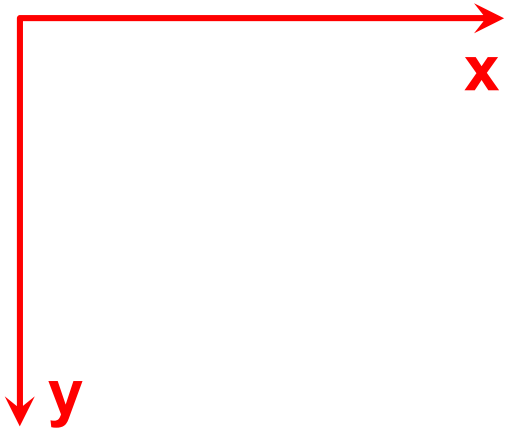
```
for y:=y1 to y2 do          // y1 < y2
   x = (y – y1)/m + x1
   pixel(x,y) := cor
```

($\Delta y > \Delta x$)

para linhas mais verticais que horizontais, que vão de cima para baixo (y2>y1), isto funciona bem



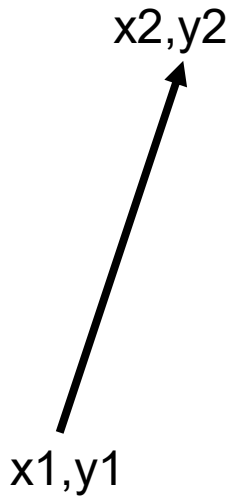
Lembre-se que a tela é assim:



Está avançando o y de um em um e calculando o x

para tratar também linhas que vão de baixo para cima (y1>y2)

($\Delta y > \Delta x$)



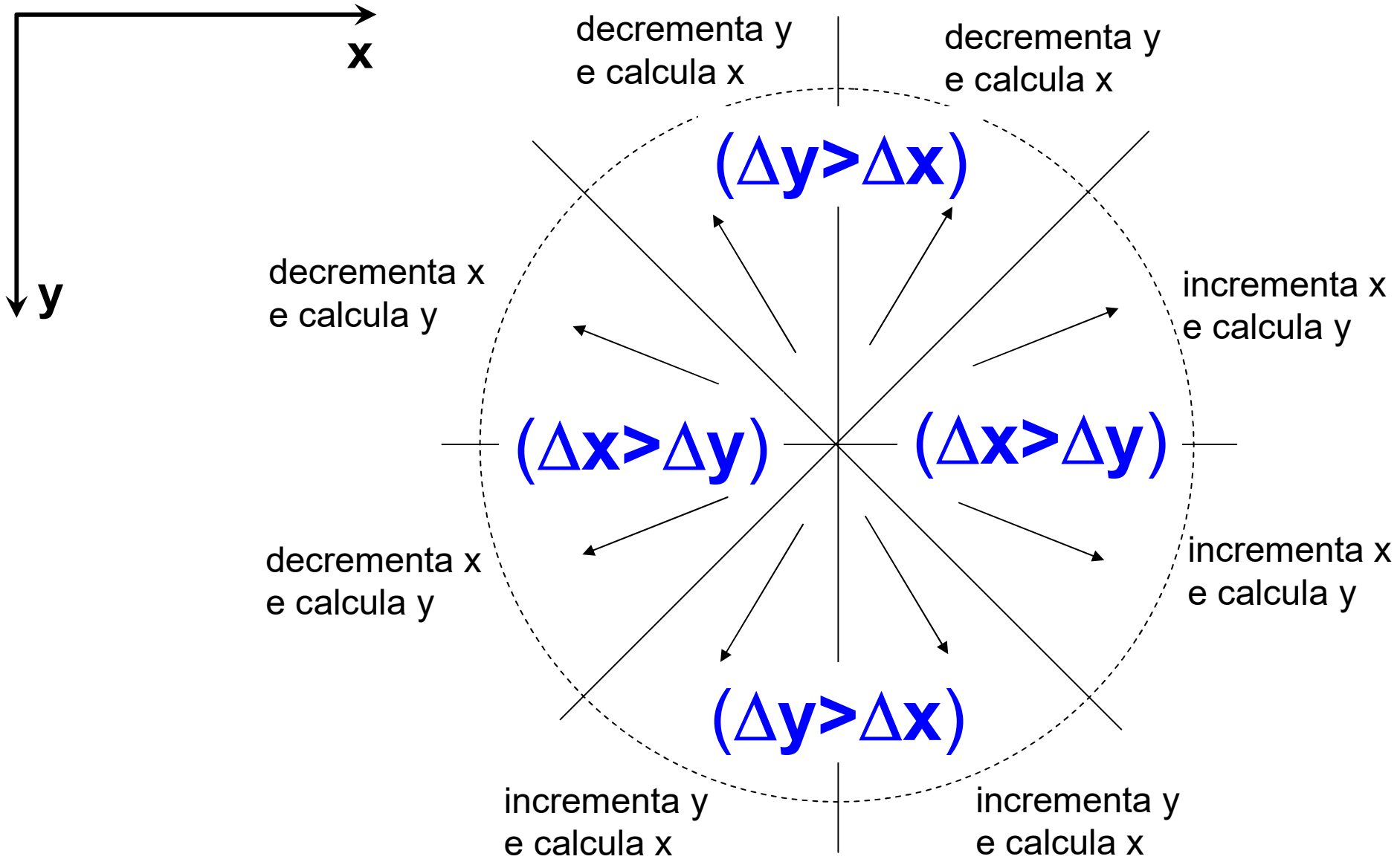
o código ficará:

```
if (y1 < y2) then inc =1
else inc = -1

x=x1
y=y1
while (y <> y2) do
      x = (y – y1)/m + x1
      plota(x,y)
      y = y + inc
```

ou ainda

```
for y:=y2 to y1 do     // y2 < y1
   x = (y – y1)/m + x1
   pixel(x,y) := cor
```

Em resumo:
Considerando que a tela cresce para a direita e para baixo, vale o seguinte:



Um problema ainda a ser resolvido é que o cálculo de **m = (y2-y1)/(x2-x1)** não pode ser feito, se a reta é vertical, ou seja, se x1 = x2 tem-se uma divisão por zero

Assim, o código precisa incluir um teste inicial e, se x1 = x2, o código pode ser:

```
if (x1 = x2)
  if (y1<y2)    for y = y1 to y2 do pixel(x1,y) = cor
  else    for y=y2 to y1 do pixel(x1,y) = cor
```

Outra opção é usar a **Forma paramétrica**

em que, dados dois pontos P1=(x1,y1) e P2=(x2,y2),

calcula-se o vetor **V** = P2 – P1

ou seja, Vx = x2-x1 e Vy = y2-y1

Assim, a equação da reta ficará:

$$\mathbf{p = p1 + V \,.\, t \quad para\ t \in [0,1]}$$

ou seja

$$\mathbf{x = x1 + (x2 - x1) \,.\, t}$$

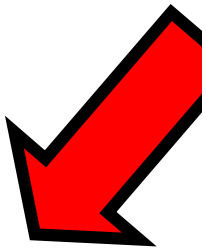
$$\mathbf{y = y1 + (y2 - y1) \,.\, t \quad para \quad t \in [0,1]}$$

Neste caso, o código ficaria:

```
for t = 0 to 1 do
    x = x1 + (x2-x1) . t
    y = y1 + (y2-y1) . t
```

Com esta abordagem não é preciso ficar vendo se vai para cima ou para baixo ou para esquerda ou direita

mas, **t** precisa ser incrementado bem devagar para não ficar buracos, por exemplo, de 0.01 em 0.01



O uso de floats leva a programas mais lentos e, difíceis de serem implementados diretamente em hardware ou programados em processadores mais simples