Primitivas de Saída

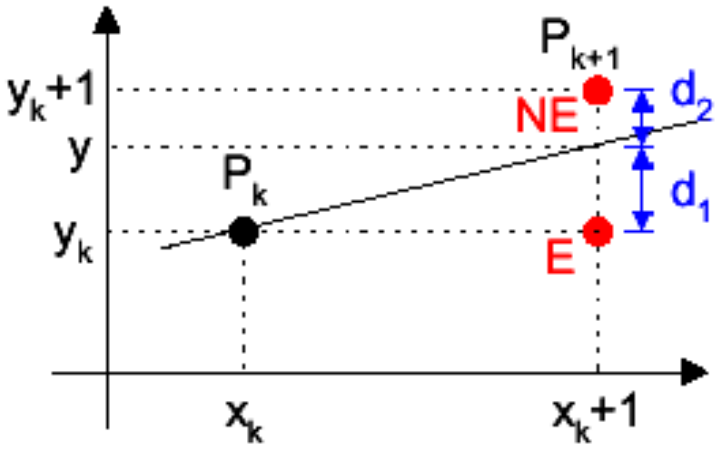
Computação Gráfica

Algoritmo de Bresenham

- Apenas realiza cálculos com inteiros
- Considerando o ponto (P_k), e para o caso em que $0 \leq m \leq 1$, o próximo ponto (P_{k+1}) apenas poderá ser o ponto à direita (E), ou o ponto acima e à direita (NE).
- A escolha do próximo ponto depende das duas distâncias d_1 e d_2.

  $d_1 - d_2 < 0 \Rightarrow$ ponto E

  $d_1 - d_2 \geq 0 \Rightarrow$ ponto NE



Algoritmo de Bresenham

- Considerando apenas o caso em que

  $0 \leq m \leq 1$, temos $y = m (x_k + 1) + b$

- Calcula-se as duas distâncias:

  $d_1 = y - y_k = m (x_k + 1) + b - y_k$

  $d_2 = (y_k + 1) - y = y_k + 1 - m (x_k + 1) - b$

- Determina-se a diferença entre ambas:

  $d_1 - d_2 = 2m (x_k + 1) - 2y_k + 2b - 1$

Algoritmo de Bresenham

- Substitui-se a inclinação pelo quociente $\Delta y/ \Delta x$, o que leva a utilização de apenas operações inteiras.
- Desta forma obtém-se a variável de decisão p_k:

  $p_k = \Delta x\ (d_1 - d_2) = 2\Delta y.x_k - 2\Delta x.y_k + c$

  onde $c = 2\Delta y + \Delta x\ .\ (2b - 1)$

Algoritmo de Bresenham

- Pode-se agora calcular p_k de forma incremental:

  $p_{k+1} - p_k = 2\Delta y\,(x_{k+1} - x_k) - 2\Delta x\,(y_{k+1} - y_k)$

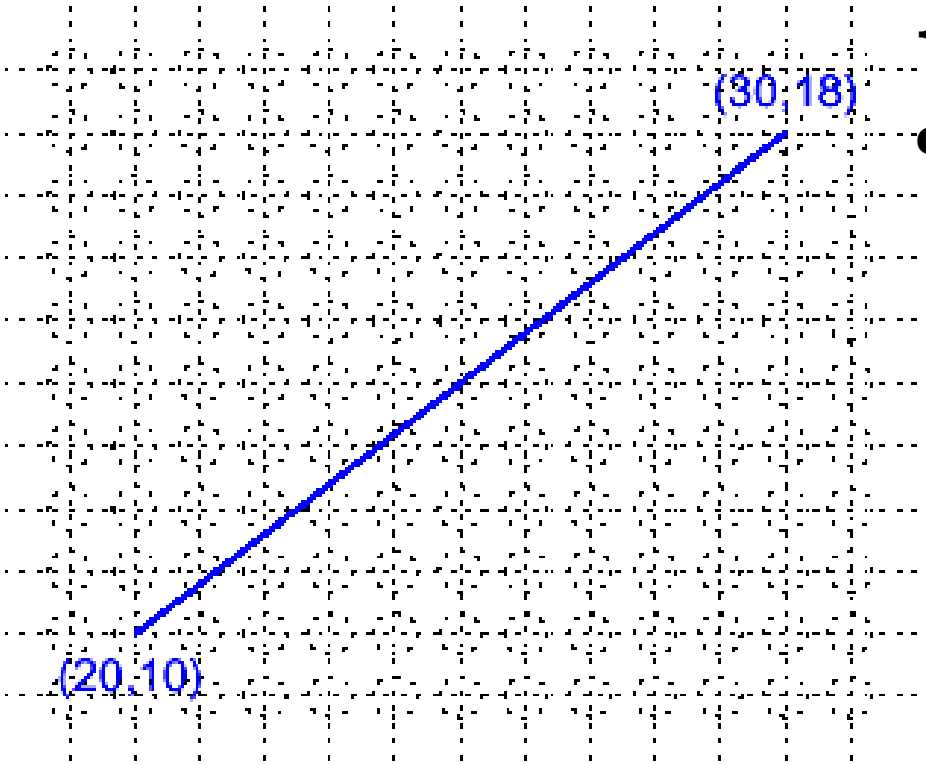
- Como $x_{k+1} = x_k + 1$, temos que:

  $p_{k+1} = p_k + 2\Delta y - 2\Delta x\,(y_{k+1} - y_k)$

- Então para $y_{k+1} - y_k = \begin{cases} 0 \Leftarrow ponto\_E \\ 1 \Leftarrow ponto\_NE \end{cases}$

$$p0 = 2\Delta y - \Delta x$$

Algoritmo de Bresenham




```
void alg_bresenham(int
x1,y1,x2,y2)
{
    int dx, dy, x, y, const1,
const2, p;
    dx = abs (x2-x1);
    dy = abs (y2-y1);
    p = 2*dy – dx;
    const1 = 2*dy;
    const2 = 2*(dy-dx);
    x = x1; y = y1;
    colora_pixel (x,y);
    while (x < x2) {
        x = x + 1;
        if (p < 0)
         p += const1;
        else {
           p+= const2; y++;
        }
        colora_pixel (x,y);
    }
}
```

Algoritmo de Bresenham – Genérico

```
void bres_gen(int x1,y1,x2,y2)
{    int dx, dy, x, y, i,
         const1, const2, p;
     dx = x2-x1;
     dy = y2-y1;
     if (dx >= 0)
        incrx = 1;
     else{ incrx= -1; dx= -dx;}
     if (dy >= 0)
        incry = 1;
     else{ incry= -1; dy= -dy;}
     x = x1; y = y1;
     colora_pixel (x,y);
     if (dy < dx){
        p =2*dy – dx;
        const1 = 2*dy;
        const2 = 2*(dy-dx);
        for (i=0; i < dx; i++) {
            x += incrx;
            if (p < 0)
                p += const1;
            else { y += incry;
                    p+= const2;}
            colora_pixel (x,y);
        }
}
```

Algoritmo de Bresenham – Genérico

```
else {
   p =2*dx – dy; const1 = 2*dx; const2 = 2*(dx-dy);
   for (i=0; i < dy; i++) {
       y += incry;
       if (p < 0)
          p += const1;
        else { x += incrx;
                  p+= const2;}
       colora_pixel (x,y);
   }
}
```