

Computación Gráfica

Algoritmos raster

Dibujo de líneas y llenado de polígonos

- Prof. María Cecilia Rivara
- mcrivara@dcc.uchile.cl
- Semestre 2004/2

Algoritmos raster

Objetivos: Rasterizar (asignar intensidad a píxeles) para dibujar / pintar primitivas gráficas

- líneas
 - polígonos
 - círculos, etc
- Incluye algoritmos de clipping

Algoritmos raster para dibujar líneas

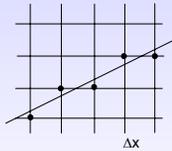
1. Algoritmo incremental (método de diferencias similar a métodos para EDP)

- Aritmética entera

$$y = mx + B \quad \text{pendiente } m = \frac{\Delta y}{\Delta x}$$

Supuesto: $|m| \leq 1$

Se avanza de izquierda a derecha



Algoritmo incremental básico

$$y_{i+1} = mx_{i+1} + B = m(x_i + \Delta x) + B = \underbrace{(m x_i + B)}_{y_i} + m \Delta x$$

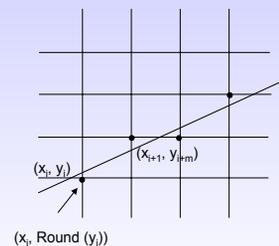
$$y_{i+1} = y_i + m \Delta x$$

y si $\Delta x = 1 \Rightarrow y_{i+1} = y_i + m$ (se redondea)

Cambio unitario en $x \Rightarrow$ cambio de y en m

Observaciones

- x_i y y_i definidos en función de valores previos en el paso anterior
- No se necesita calcular intersección en y
- No se ocupan multiplicaciones
- Si $|m| > 1$, invertir roles de x e y (paso unitario en y e incremento de x en $1/m$)
- Se puede generalizar a aritmética de punto flotante



Algoritmo básico

```

procedure Line(           {Asuma  $-1 \leq m \leq 1, x_0 > x_1$ }
    x0, y0                 {Punto izquierdo}
    x1, y1,                 {Punto derecho}
    value : integer;        {Valor a poner en los pixeles de la línea}

var
    x : integer;           {x varía entre x0 y x1 en incrementos unitarios}
    dy, dx, y, m : real;

begin
    dy := y1 - y0;
    dx := x1 - x0;
    m := dy/dx;
    y := y0;
    for x := x0 to x1 do
        begin
            WritePixel (x, Round(y), value); {Asigne value al pixel}
            y := y + m                       {Incremente y en la pendiente m}
        end
    end;

```

MCRivara/CG2004/2

7

Desventajas del algoritmo incremental básico

- La función Round toma mucho tiempo
- y, m son reales

Algoritmo de Bresenham

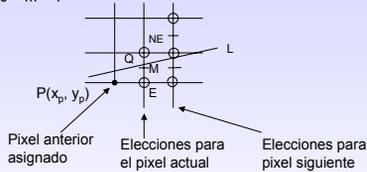
- Bresenham desarrolló un algoritmo incremental clásico que usa solo aritmética entera evitando la función Round
- La técnica se generaliza al dibujo de círculos
- Bresenham demostró que sus algoritmos generan las aproximaciones que minimizan el error (distancia)

MCRivara/CG2004/2

8

Algoritmo de la línea media (variación de la idea de Bresenham)

Supuesto: $0 < m < 1$



Q: punto de intersección de la línea L con la línea $x = x_p + 1$

MCRivara/CG2004/2

9

Idea

Se calculan las diferencias NE-Q y E-Q y el signo de se usa para seleccionar el pixel más cercano a Q

- En el algoritmo de la línea media se observa a que lado de la línea L queda M
 - M por debajo \Rightarrow Q más cerca de NE
 - M por arriba \Rightarrow Q más cerca de E

MCRivara/CG2004/2

10

¿Cómo calcularlo?

$$y = \frac{dy}{dx}x + B, \text{ con } dy = y_1 - y_0, dx = x_1 - x_0$$

Ec. implícita de L

$$F(x, y) = dyx - dxy + Bdx = 0 = ax + by + c$$

$$F(x, y) = \begin{cases} 0 & \text{punto } (x, y) \text{ sobre la línea} \\ > 0 & \text{punto bajo la línea} \\ < 0 & \text{punto arriba de la línea} \end{cases}$$

Luego basta calcular

$$d = F(M) = F(x_p + 1, y_p + 1/2) \text{ y testear signo}$$

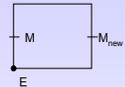
- $d > 0$, se elige pixel NE
- $d < 0$, se elige pixel E
- $d = 0$, cualquiera (elijamos E)

MCRivara/CG2004/2

11

¿Cómo seguimos al pixel siguiente? (1)

- Si se eligió E



$$M_{\text{new}}(x_p + 2, y_p + 1/2) \text{ (se incrementa } x \text{ en } 1)$$

$$d_{\text{new}} = F(x_p + 2, y_p + 1/2) = a(x_p + 2) + b(y_p + 1/2) + c$$

$$d_{\text{old}} = a(x_p + 1) + b(y_p + 1/2) + c$$

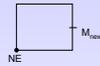
$$\text{Restando } d_{\text{new}} - d_{\text{old}} = \Delta_{NE} \text{ (incremento)}$$

MCRivara/CG2004/2

12

¿Cómo seguimos al pixel siguiente? (2)

Si se eligió NE



$M_{new}(x_p + 2, y_p + 3/2)$ (se incrementan x, y en 1)

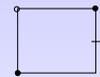
Análogamente al caso anterior

$$d_{new} = d_{old} + \underbrace{a+b}_{\Delta_{NE}}$$

Obs: No es necesario calcular F(M) basta incrementar d por Δ_E o Δ_{NE}

¿Cómo partimos?

Línea desde (x_0, y_0) a (x_1, y_1)



$M(x_0 + 1, y_0 + 1/2)$

(x_0, y_0)

$$d_{start} = a + b/2 = dy - dx/2$$

Algoritmo de Bresenham (variación)

```

procedure MidpointLine (x0, y0, x1, y1, value: integer);
var
  dx, dy, incrE, incrNE, d, x, y : integer;
begin
  dx := x1 - x0;
  dy := y1 - y0;
  d := 2 * dy - dx;           {valor inicial para d}
  incrE := 2 * dy;           {incremento usado para moverse a E}
  incrNE := 2 * (dy - dx);   {incremento usado para moverse a NE}
  x := x0;
  y := y0;
  WritePixel (x, y, value);   {Pixel de partida}
  while x < x1 do
    begin
      if d <= 0 then
        begin
          d := d + incrE;     {Elija NE}
          x := x + 1;
        end
      else
        begin
          d := d + incrNE;    {Elija NE}
          x := x + 1;
          y := y + 1;
        end
      WritePixel (x, y, value) {Elige pixel más cercano a la línea}
    end (while)
  end; (MidpointLine)
  
```

Observaciones:

- Para eliminar la fracción se redefine $\tilde{F} = 2F$ (no afecta el signo)
- El algoritmo trabaja solo para $0 < m < 1$
Otras pendientes se manejan por reflexiones con respecto a ejes del sistema de coordenadas.
- No se han considerado casos particulares líneas verticales y horizontales de pendientes ± 1

Obs: Distinto grosor dependiendo de la pendiente

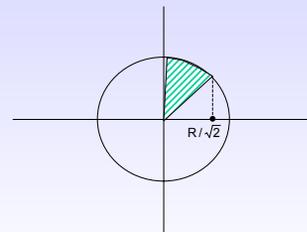
¿Cómo tratarlo?

- Variación de la intensidad en función de la pendiente
- Tratando la línea como rectángulo



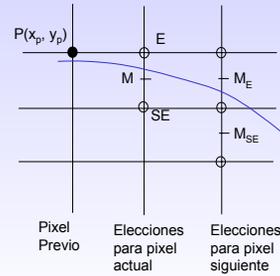
Círculo: Algoritmo del punto medio

- Se calcula solo un 1/8 del círculo



Círculo

- Algoritmo del punto medio
Función implícita $F(x, y) = x^2 + y^2 - R^2$
Variable de decisión $d = F(M)$
 Δ_E y Δ_{SE} son ahora funciones de (x_p, y_p)
- Algoritmo con diferencias de segundo orden



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

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

- Si $d_{old} < 0$ se elige E \Rightarrow

$$d_{new} = F(x_p + 2, y_p - 1/2) = (x_p + 2)^2 + (y_p - 1/2)^2 - R^2$$

$$d_{new} = d_{old} + \underbrace{(2x_p + 3)}_{\Delta_E}$$

Círculo (cont.)

Si $d_{old} \geq 0$ se elige SE \Rightarrow

$$d_{new} = F(x_p + 2, y_p - 3/2) = (x_p + 2)^2 + (y_p - 3/2)^2 - R^2$$

$$d_{new} = d_{old} + \underbrace{(2x_p - 2y_p - 5)}_{\Delta_{SE} \text{ (incremento)}}$$

Círculo (cont.)

Condición inicial: pixel en $(0, R)$

Punto medio siguiente en $(1, R - 1/2)$

Luego $F(1, R - 1/2) = 5/4 - R$

Tarea: Escribir el algoritmo

Llenado de rectángulos y polígonos

(1) ¿Qué pixeles llenar? Relacionado con clipping?

(2) ¿Con qué valor se llena?

- Polígonos sin clipping

Se explota coherencia: primero se calculan los extremos y después los interiores

Conversión scan de un rectángulo

for y desde y_{min} hasta y_{max} {por cada scan line}

for x desde x_{min} to x_{max} {por cada pixel}

WritePixel ($x, y, value$)

Rectángulos que comparten arista



Convención para no dibujar arista dos veces. Un pixel no se considera parte de la primitiva si el semiplano que contiene a la primitiva está por debajo o a la izquierda de la arista

Dificultades: slivers

Polígonos "agujas"
Pueden no incluir
píxeles en su interior

