Primitivas Bidimensionales

Curso 06/07

Introducción

- Definiciones
  - Conversión al “raster”
    - Aproximar elementos geométricos continuos mediante un conjunto de puntos discretos (píxel)
    - Píxel: área de la pantalla del ordenador que tiene asociada una posición de memoria

Introducción

- Ejemplo: conversión de puntos discretos
  - Convertir al raster un punto definido mediante una coordenada almacenando el color con el que se va a visualizar
  - Un algoritmo sencillo de conversión punto-raster

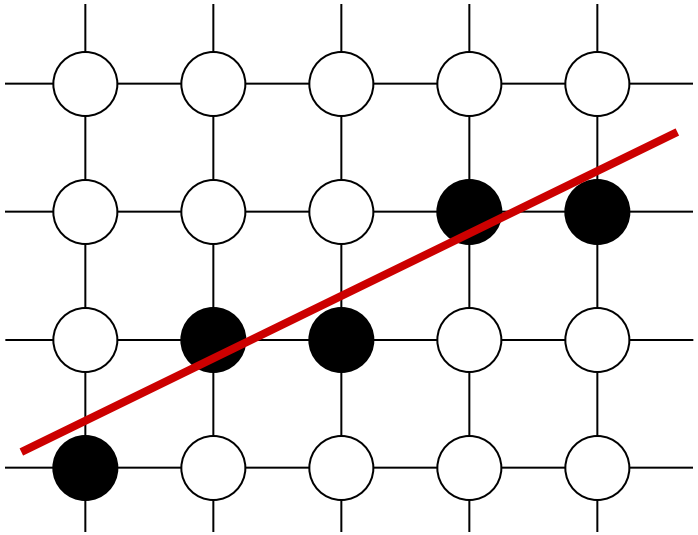
Sea (x,y) un punto real

(x',y')=(round(x), round(y))

  - Donde `(x',y')` es la posición en el raster
  - La función `round()` redondea el valor real al entero más cercano

Conversión de Rectas

- Problema
  - Calcular las coordenadas de los píxeles que representan una recta infinitamente delgada sobre una malla de un “raster” 2D



Conversión de rectas

- Algoritmo de fuerza bruta
  - Estrategia más simple
    - La ecuación de la recta es:

      $y=m*x+b$

    - Donde $m=dx/dy$
  - El algoritmo para pintar la recta entre los extremos (x_1,y_1)-(x_2,y_2) sería:

```
Para x desde x1 hasta x2
  (incremento/decremento unitario)
  y = x·m + b
  dibujar_pixel(x,round(y))
finPara
```

Conversión de rectas

- Algoritmo de fuerza bruta
  - Restricción
    - Los valores de `m` tienen que estar entre -1 y 1
  - Solución
    - Intercambiar la `x` por la y en los extremos de la recta, convertir al raster e intercambiar de nuevo antes de visualizar
    - Calcular la `x` en función de incrementos unitarios de `y`
  - Desventajas
    - Ineficiente, en cada iteración requiere de una multiplicación en coma flotante y de una invocación a la función `round()`

Trazado de líneas

- Algoritmo Diferencial Digital (DDA), incremental básico
  - Elimina el producto de coma flotante
  - Se basa en (si la pendiente es ≤1):

$$y_{i+1} = m \cdot x_{i+1} + b = m \cdot (x_i + \Delta x) + b = y_i + m \cdot \Delta x$$

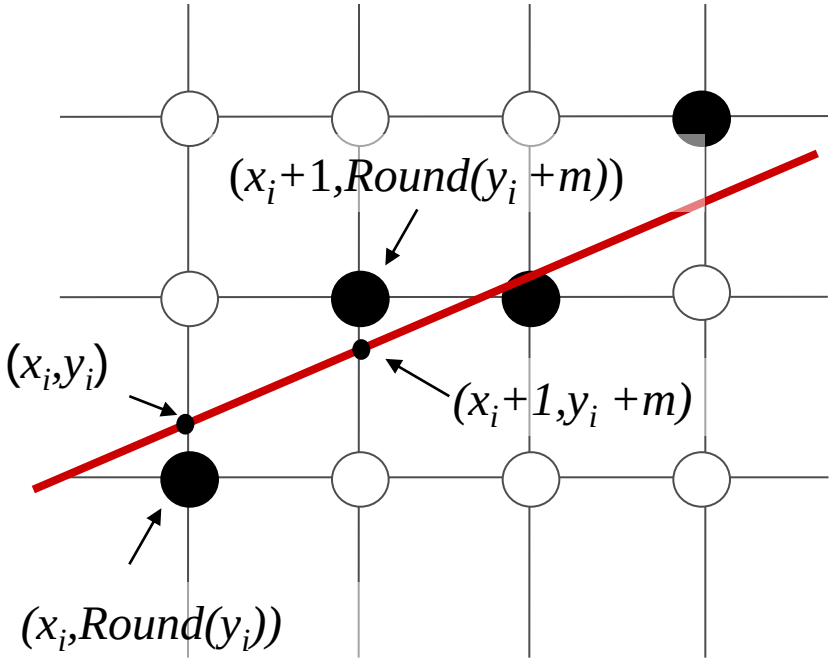
  - Como en las iteraciones $\Delta x = 1$ entonces resulta

$$y_{i+1} = y_i + m$$

Trazado de líneas

- Algoritmo Diferencial Digital
  - Los valores de y deben redondearse al entero más próximo
  - Algoritmo:

```
y = y1
Para x desde x1 hasta x2
   (incremento unitario)
   y = y + m
   dibujar_pixel(x,round(
   y))
finPara
```



Trazado de líneas

- Algoritmo Diferencial Digital
  - Cuando la pendiente es >1 entonces

    $x_{i+1} = x_i + (1/m)$

    - Los incrementos positivos implican representar la línea de izquierda a derecha, para el trazado inverso los incrementos deben ser negativos
  - Desventaja
    - Sigue manteniendo una llamada a la función `round()`

Trazado de líneas

- Algoritmo Diferencial Digital
  - Algoritmo completo para el segmento (x_1 , y_1)-(x_2 , y_2):

```
dx=x2-x1
dy=y2-y1
Si  abs(dx)>abs(dy)  entonces
   pasos=abs(dx)
sino
   pasos=abs(dy)
finSi
xInc=dx/pasos
yInc=dy/pasos
x = x1
y = y1
dibujar_pixel(round(x),round(y))
Para k desde 1 hasta pasos
   x = x + xInc
   y = y + yInc
   dibujar_pixel(round(x),round(y))
finPara
```

Trazado de líneas

- Algoritmo del punto medio de Bresenham (1967)
  - La ecuación del punto es:

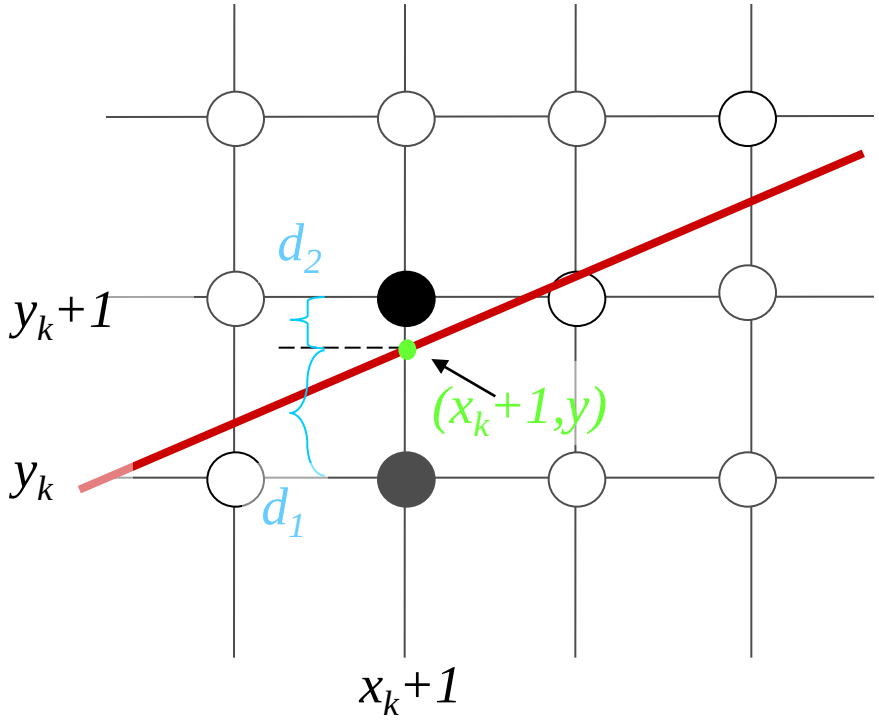
$$y = m \cdot (x_k+1) + b$$

  - Por tanto:

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

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

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



Trazado de líneas

- Algoritmo del punto medio de Bresenham
  - El signo del factor p_k determina el pixel:

$$p_k=\Delta x(d_1 - d_2)= \Delta x(2m(x_k+1)-2y_k+2b-1)=2\Delta y\, x_k+2\Delta y-2\Delta x y_k+\Delta x(2b-1)=2\Delta y\, x_k-2\Delta x\, y_k+\boxed{\Delta x(2b-1)+2\Delta y}$$

c

  - El factor p_k
    - Tiene el mismo signo que la diferencia $d_1 - d_2$
    - Todas las operaciones son con enteros
    - La constante c es independiente del pixel
    - Si p_k es negativo se pinta el pixel inferior

Trazado de líneas

- Algoritmo del punto medio de Bresenham
  - En el paso $k+1$ el parámetro p_{k+1}

    $p_{k+1}= 2\Delta y\, x_{k+1}-2\Delta x\, y_{k+1}+c$
  - Restando

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

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

    $p_{k+1}=p_k + 2\Delta y -2\Delta x\, (y_{k+1}-y_k)$
  - Con el término $y_{k+1}-y_k$ que toma el valor 0 o el valor 1
  - Si $p_0= 2\Delta y -\Delta x$ entonces

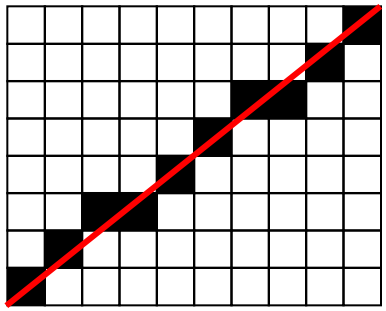
Trazado de líneas

- Algoritmo del punto medio de Bresenham
  1. Trazar el primer pixel (x_1,y_1)
  2. Calcular las constantes Δx, Δy, $2\Delta x$, $2\Delta y$
  3. Calcular $p_0 = 2\Delta y - \Delta x$
  4. Para cada x_k (comenzando en $k=0$)
     - si $p_k < 0$ se traza el punto (x_k+1, y_k) con $p_{k+1} = p_k + 2\Delta y$
     - otro caso se traza el punto (x_k+1, y_k+1) con $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
  5. Repetir el paso 4 Δx veces

Trazado de líneas

- Algoritmo del punto medio de Bresenham
  - Ejemplo: Trazar una recta entre (20,10)-(30,18)
    - $\Delta x=10$, $\Delta y=8$, $2\Delta y=16$, $2\Delta y-2\Delta x=-4$
    - $p_0= 2\Delta y -\Delta x=2\cdot 8-10=6$
    - $(x_0,y_0)=(20,10)$

k	p_k	(x_{k+1},y_{k+1})
0	6	(21,11)
1	2	(22,12)
2	-2	(23,12)
3	14	(24,13)
4	10	(25,14)
5	6	(26,15)
6	2	(27,16)
7	-2	(28,16)
8	14	(29,17)
9	10	(30,18)



Trazado de circunferencias

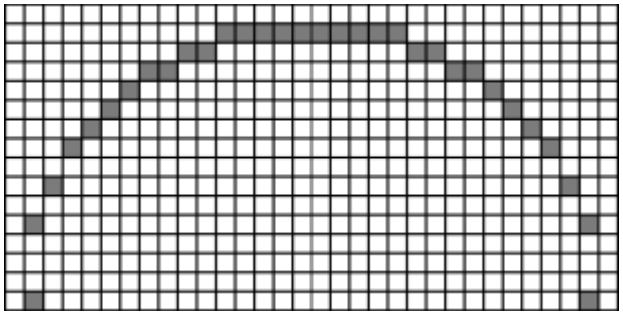
- La ecuación del círculo es:

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

- Podría dibujarse según:

$$y = y_c \pm \sqrt{r^2 - (x - x_c)^2}$$

- Inconveniente
  - Cálculo computacional
    - Coma flotante
    - Raíz cuadrada y redondeo
  - Distancia no uniforme entre píxeles



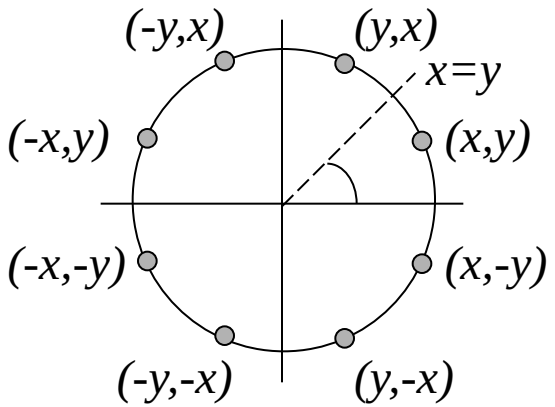
Trazado de circunferencias

- Ecuación en polares

$$x = x_c + r\cos\theta$$

$$y = y_c + r\sin\theta$$

- Inconvenientes
  - Hay cálculos de funciones trigonométricas
- Puede mejorarse teniendo en cuenta la simetría del círculo
  - Sólo requiere dibujarse en [0,π/4]



Trazado de circunferencias

- Algoritmo de punto medio
  - Siguiendo un razonamiento similar al empleado para líneas se traslada para dibujar circunferencias
  - Utiliza aritmética entera
  - Utiliza la simetría del círculo
  - Si el último punto dibujado es (x_k,y_k) entonces el siguiente será (x_{k+1},y_k) o (x_{k+1},y_{k-1})
  - El parámetro de decisión es el punto medio de ambos

Trazado de circunferencias

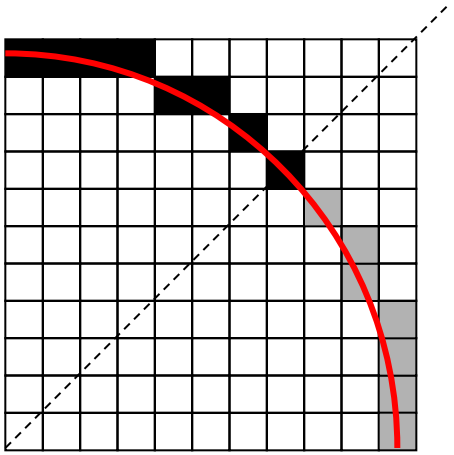
- Algoritmo de punto medio (La discusión completa del algoritmo está en Foley y en Hearn)
  - Algoritmo
    1. Primer punto $(x_0,y_0)=(0,r)$
    2. Si r es:
       - Real, $p_0=5/4-r$
       - Entero, $p_0=1-r$
    3. Si p_k
       - <0 entonces (x_{k+1},y_k), $p_{k+1}=p_k+2x_{k+1}+1$
       - ≥0 entonces (x_{k+1},y_{k-1}), $p_{k+1}=p_k+2x_{k+1}+1-2y_{k+1}$

       Con $2x_{k+1}=2x_k+2$

       $2y_{k+1}=2y_k-2$
    4. Calcular los puntos simétricos
    5. Desplazar los puntos (calculados para una circunferencia centrada en el origen) a la circunferencia centrada en (x_c,y_c)
    6. Repetir los pasos 3-5 hasta que $x \geq y$

Trazado de circunferencias

- Algoritmo de punto medio
  - Ejemplo, circunferencia $r=10$ centrada en $(0,0)$
    - $p_0 = 1-r = -9$
    - $(x_0,y_0)=(0,10)$



k	p_k	(x_{k+1},y_{k+1})
0	-9	(1,10)
1	-6	(2,10)
2	-1	(3,10)
3	6	(4,9)
4	-3	(5,9)
5	8	(6,8)
6	5	(7,7)

Trazado de otras figuras

- El algoritmo de punto medio se generaliza a
  - Elipses
  - Cónicas
  - Splines
    - Curvas generadas a partir de la composición de segmentos de polinomios cúbicos

$$x = a_{x0} + a_{x1}u + a_{x2}u^2 + a_{x3}u^3$$

$$y = a_{y0} + a_{y1}u + a_{y2}u^2 + a_{y3}u^3$$

    - Las constantes se resuelven imponiendo continuidad y “suavidad”