Computación Gráfica
Primitivas Gráficas (cont.)

-Círculo
-Area pintada

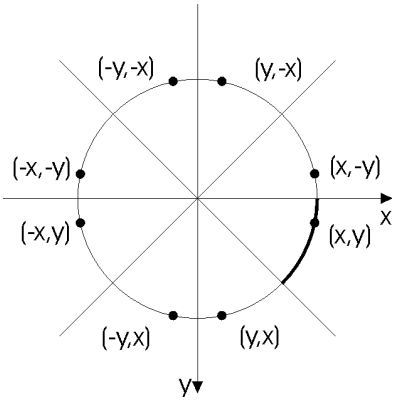
Algoritmos para graficar circunferencias

- DDA
- Bresenham

Una circunferencia queda determinada con el centro y el radio.

Algoritmos para graficar circunferencias

Simetría: se divide la circunferencia en 8 arcos simétricos.
Se desarrollan los pixel del arco definido por x≥0, y≥0, x≥y.
Quedan los pixels correspondientes en los otros 7 arcos simétricos



Algoritmo DDA

Partiendo de la ecuación de la circunferencia, y derivando.

$$x^2 + y^2 = r^2$$

Se obtiene su ecuación diferencial.

$$\frac{dy}{dx} = \frac{-x}{y}$$

Algoritmo DDA

Como en el caso de rectas este método evalúa la ecuación diferencial a intervalos finitos. Debe encontrar una secuencia de puntos de la pantalla (X^0 , Y^0), (X^1 , Y^1),..., (X^n , Y^n) que formen el arco de circunferencia. Entonces si tengo un punto de la discretización (X^k , Y^k), debe ser

$$X^{k+1} = X^k - \varepsilon Y^k$$

$$Y^{k+1} = Y^k + \varepsilon X^k$$

Aquí ε no es constante

Algoritmo DDA

Con esta forma de determinar los puntos de la discretización resulta

$$\frac{Y^{k+1} - Y^k}{X^{k+1} - X^k} = -\frac{X^k}{Y^k}$$

por lo tanto evalúa la ecuación diferencial a intervalos finitos.

El valor de ε determina la frecuencia de muestreo, si es muy pequeño, el cómputo será redundante y si es muy grande los puntos estarán muy separados.

El valor que se elige es $\varepsilon x=1$.

El algoritmo empieza en el pixel P=(r,0)

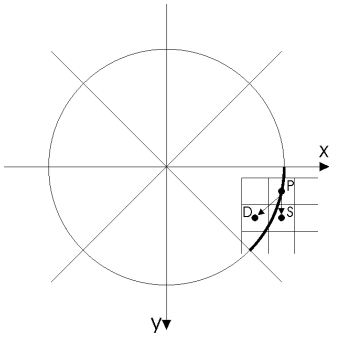
Algoritmo DDA

```
CirculoDDA ( radio , col )
{  rx = radio ;
   x = round ( rx ); y = 0;
   while ( y ≤ x ) {
      putpixel ( x , y , col );   putpixel ( y , x , col );
      putpixel ( − x , y , col );   putpixel ( − y , x , col );
      putpixel ( x , − y , col );   putpixel ( y , − x , col );
      putpixel ( − x , − y , col );   putpixel ( − y , − x , col );
      rx = rx − y / rx ;
      x = round ( rx );
      y = y + 1; }
}
```

Realiza sólo una operación de división en punto flotante en cada paso.

Algoritmo de Bresenham

Se basa en analizar el error entre la verdadera circunferencia y su discretización. En cada paso elige como próximo pixel a aquel que minimice el error.



Para cada pixel P en el arco definido por x≥0, y≥0, x≥y; el pixel siguiente sólo puede ser S (sur) o D (diagonal)

Algoritmo de Bresenham

En este caso se toma como error la distancia al cuadrado del pixel (x,y) a la circunferencia de centro en el origen y radio r:

$$e=x^2+ y^2-r^2.$$

Si elegimos S el proximo pixel es $P_S=(x, y+1)$ entonces:

$$e_S = x^2+ (y+1)^2-r^2 = e+2y+1$$

Si elegimos D, entonces el próximo pixel es $P_D=(x-1, y+1)$

$$e_D = (x-1)^2+ (y+1)^2-r^2 = x^2-2x +1+y^2+2y+1-r^2 = e_S -2x+1$$

Algoritmo de Bresenham

Entonces la elección del pixel P_S o P_D dependerá de cual de los dos errores tiene menor módulo

$$\text{Si } |e+2y+1| > |e+2y+1-2x+1| \Rightarrow D$$

Es lo mismo que

$$\text{Si } |e_S| > |e_S-2x+1| \Rightarrow D$$

Algoritmo de Bresenham

Además $-2x+1 \leq -1$ y $0 < e_S < 1$ entonces

Si $e_S > -e_S + 2x - 1 \Rightarrow$ D y por lo tanto

Si $2e_S > 2x - 1 \Rightarrow$ D

El algoritmo utiliza solo operaciones enteras

Algoritmo de Bresenham

```
CirculoBresenham(radio, col)
{  int x, y, e;
   x = radio; y = 0; e = 0;
   while(y ≤ x) {
      putpixel(x, y, col);     putpixel(y, x, col);
      putpixel(−x, y, col);    putpixel(−y, x, col);
      putpixel(x,−y, col);     putpixel(y,−x, col);
      putpixel(−x,−y, col);  putpixel(−y,−x, col);
      e = e + 2y + 1;
      y = y + 1;
      if (2e > (2x − 1)) {
         x = x − 1;
         e = e − 2x + 1;   }
   }
}
```

Discretización de Polígonos
Scan Conversion

- El objetivo es encontrar el conjunto de pixels que determinan el área sólida que dicho polígono cubre en la pantalla.

- El método que se presenta se basa en encontrar la intersección de todos los lados del polígono con cada línea de barrido, por eso se llama conversión scan del polígono.

- Todo polígono plano puede descomponerse en triángulos, luego veremos como dibujar el triángulo lleno.

Discretización de Polígonos

Para el algoritmo es necesario dimensionar dos arreglos auxiliares enteros

minx y maxx

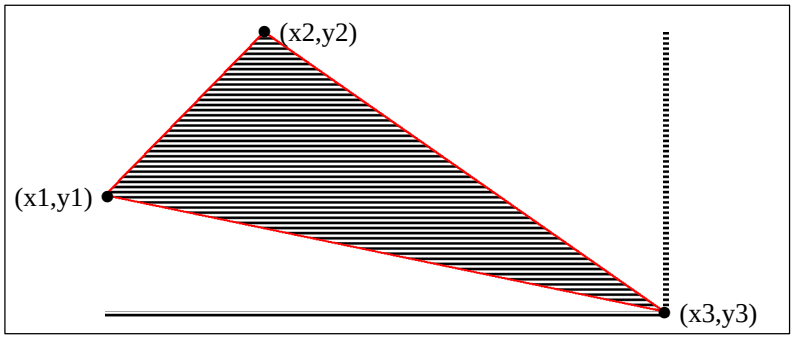
del tamaño del alto de la pantalla, que para cada línea de barrido almacenarán el menor y el mayor x respectivamente.

Discretización de Polígonos

Se inicializa minx en +∞ y maxx en -∞ y discretiza cada arista del triángulo con DDA o Bresenham reemplazando la sentencia putpixel por

If x>maxx[y] then maxx[y]=x;
If x<minx[y] then minx[y]=x;
Para cada ***y*** activo graficar una línea de minx[y] a maxx[y]

Discretización de Polígonos



Fin