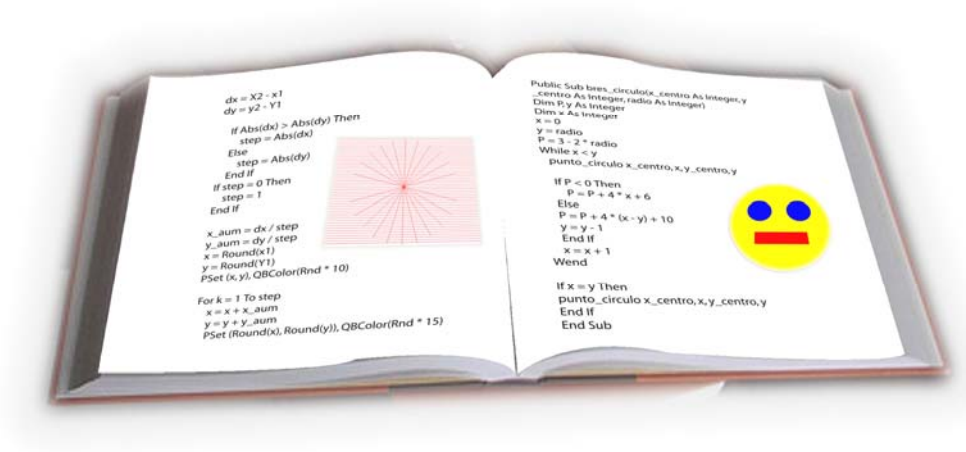
CAPITULO V

GRAFICANDO IMÁGENES POR COMPUTADORA



"Las matemáticas son el alfabeto con el cual Dios ha escrito el Universo."

Galileo Galilei

La graficación por computadora comprender los fundamentos de realizar dibujos en segunda y tercera dimensión utilizando lenguajes de programación para ejemplificar los algoritmos.

Los gráficos proveen uno de los medios más naturales y potentes de comunicarse. El procesamiento cerebral del aparato visual esta altamente desarrollado, y por lo tanto es capaz de reconstruir, procesar, interpretar, recordar y cotejar una enorme cantidad de información en un tiempo apenas perceptible. Por su parte, la evolución tecnológica y la rápida difusión de las computadoras determina que en la actualidad la Computación Gráfica es el medio de producción de imágenes más importante en la actualidad, superando a la fotografía, diseño y artes gráficas en general, y compitiendo con el cine y la televisión.

Tiene la ventaja adicional de poder reproducir imágenes virtuales, que no necesariamente existen o se pueden ver en la realidad.

Las graficas hechas por computadora cubren un amplio espectro de aplicaciones dentro de nuestro mundo, entre las más representativas encontramos las siguientes:

- **Interfases:** Actualmente la mayoría de los sistemas operativos, utilitarios, procesadores de texto, etc. tienen una interfase gráfica basada en la metáfora visual del escritorio, con íconos, menús descolgables, barras deslizantes y muchas otras facilidades.
- **Industria del entretenimiento:** Aquí tenemos la producción de video-juegos, películas y cortos de dibujos animados, posproducción y efectos especiales de películas, publicidad, etc.
- **Aplicaciones comerciales**: Son cada vez más comunes los sistemas para elaboración de presentaciones comerciales, incluyendo cartillas gráficas, diagramación automática, y publicación electrónica en general.
- **Diseño asistido**: El CAD en general, desde el dibujo de planos hasta el desarrollo de chips VLSI pasando por cientos de otras aplicaciones, también tiene un gran auge en la actualidad. En todos los casos la representación gráfica de la información es la clave del funcionamiento.

- **Aplicaciones Científicas:** Tenemos los sistemas de simulación hasta la visualización de fenómenos abstractos y su representación gráfica por medio de metáforas visuales adecuadas.
- **Cartografía y GIS:** Los gráficos por computadora son actualmente utilizados como soporte para los sistemas de información geográfica (GIS) y todas las aplicaciones relacionadas (turismo, geología, minería, clima, urbanismo, etc.).

5.1 PANORAMA GENERAL DE LOS SISTEMAS DE GRÁFICAS

Los sistemas de computación pueden adaptarse a aplicaciones de las gráficas en varias formas, conforme a los recursos de hardware y software de que disponen.

Las proposiciones de salida, como PRINT y WRITE, se san para producir la cadena de caracteres, a pesar de ser instrucciones simples estas pueden producir diseños complicados.

5.1.1 DISPOSITIVOS GRÁFICOS

Los resultados gráficos de una aplicación pueden mostrarse en una gran variedad de dispositivos de salida. Normalmente estos dispositivos son o bien de pantalla o bien de impresión. Sin embargo, desde el punto de vista de la Computación Gráfica, es importante otra clasificación, referida al modo en que los mismos son manejados por la computadora. Tenemos dos grupos bien definidos:

- **Dispositivos de vectores**, estos reciben de la computadora la información geométrica de la localización y tamaño de las primitivas que soportan, de las cuales producen una reproducción caligráfica".
- **Dispositivos de rastre,** estos reciben de la computadora la información de una serie de píxeles, los cuales son posicionados en forma contigua.

Los dispositivos de vectores fueron los primeros en desarrollarse, pero luego del vertiginoso descenso en el costo de la memoria volátil, a partir de la década de los 70 se hicieron más baratos los dispositivos de rastre, esto implica un cambio en la manera de representar las **primitivas gráficas** (usualmente dichas primitivas son el punto, el segmento de recta y la circunferencia o el círculo).

5.1.2 DISPOSITIVOS DE VECTORES

Actualmente estos dispositivos son mas caros, pero tienen ciertas ventajas que los hacen únicos. Por ejemplo, tienen mucha mejor resolución y precisión que los dispositivos de rastre, y requieren un ancho de banda de comunicación mucho menor dado que no reciben la discretización completa de las primitivas sino solamente su posición.

- **Plotters**: Grafican en una hoja (que en algunos casos puede ser de gran tamaño) sobre la cual se desliza una pluma movida por motores de pasos de gran precisión. En los plotters de tambor, la pluma se desliza en sentido horizontal y el papel en sentido vertical. En los plotters planos (más económicos), el papel esta fijo y la pluma realiza todos los movimientos. Son usuales las resoluciones del orden de los 10000 £ 10000. Es posible utilizar colores por medio de varias plumas. Son ideales para la graficación rápida y precisa de planos.

- **Displays de almacenamiento:** Al igual que televisores y monitores, estos dispositivos son pantallas de rayos catódicos, pero difieren en ciertos aspectos tecnológicos. Esencialmente, la pantalla tiene cierta memoria electrostática que mantiene visibles los elementos graficados con muy alta precisión y sin la necesidad de refresco. Por lo tanto, una imagen muy compleja a la cual se van agregando elementos en orden es idealmente representada por estos dispositivos. Un elemento se representa pintándolo por medio de una serie de recorridas del cañón electrónico. El borrado, sin embargo, no puede hacerse en forma selectiva, por lo que no se puede alterar la posición de un elemento sin tener que borrar y re-dibujar todos los demás. Sin embargo, su precisión y velocidad sin necesidad de memoria volátil los hace ideales para la representación de imágenes de radares.

5.1.3 DISPOSITIVOS DE RASTER

- **Impresoras de matriz:** Era hasta hace poco el dispositivo de impresión mas común. Recibe de la computadora la información gráfica

como una secuencia de líneas, las cuales va reproduciendo con una cabeza impresora (por medio del golpe de martillos o el rocío de tinta).

- **Impresoras Láser**: Recibe de la computadora la información gráfica como una secuencia de líneas, las cuales almacena en una memoria local. Dicha memoria es utilizada para comandar la intensidad de un haz láser que recorre línea por línea el papel, mientras es expuesto al contacto del toner. Donde el haz incide con gran intensidad, el papel se dilata por el calor y absorbe el toner.

- **Monitores**: Se han popularizado enormemente a partir del descenso en el precio de la memoria volátil y el incremento constante en la calidad de las prestaciones (resolución, color, precisión). Esencialmente se comportan de una manera similar a un receptor de televisión, excepto por el hecho de que reciben la señal de video y sincronismo en forma directa de la computadora y no a través de una portadora de radio. Al igual que con las impresoras de matriz, la imagen se construye línea por línea, en sentido horizontal primero (de izquierda a derecha) y vertical después (de arriba abajo). Debe existir un refresco de la imagen en memoria, la cual es recorrida por la tarjeta gráfica de la computadora para producir las líneas de barrido.

Por lo tanto, el elemento esencial en estos dispositivos es la tarjeta gráfica, la cual veremos en detalle mas adelante. Los monitores mas populares pueden tener resoluciones de hasta 1200 £ 1024 (aunque este límite avanza día a día), con una cantidad de colores limitada por las prestaciones de la tarjeta gráfica. Esto representa una calidad más que aceptable para la mayor parte de las aplicaciones.

5.1.4 HARDWARE GRÁFICO PARA MONITORES

Los dispositivos de rastre requieren un refresco permanente de la discretización de la salida gráfica. En el caso de los monitores, dicho refresco se realiza en un segmento de la memoria volátil de la computadora denominada frame búfer o búfer de pantalla, que usualmente se implementa

por medio de memoria RAM de alta velocidad localizada dentro de la tarjeta gráfica. El búfer de pantalla es accedido en forma rítmica por el generador de video, que es el encargado de componer" la señal de video que va hacia el monitor. Al mismo tiempo, al producirse una salida gráfica por parte de la CPU de la computadora, la misma debe ser discretizada y almacenada en el búfer de pantalla. Este acceso debe ser permitido solamente en los momentos en los que el generador de video no esta accediendo al búfer, y por lo tanto se requiere el uso de un árbitro que mantenga abierto el acceso al búfer solo en esos casos.

5.2 TÉCNICAS DE DISCRETIZACIÓN

A partir de este punto, y en lo que resta del Capítulo, analizaré el modelo de dispositivo de rastre. Para conseguir independencia de dispositivo, entonces, es necesario adoptar un conjunto de primitivas y establecer una serie de métodos que permitan representar dichas primitivas en nuestro dispositivo de rastre satisfaciendo un conjunto de especificaciones.

5.2.1.EL SISTEMA DE COORDENADAS FÍSICO

El primer paso para conseguir una representación adecuada de las primitivas es caracterizar matemáticamente el medio que nos permite representarlas. Las primitivas gráficas independientes de dispositivo (en la imagen mental del usuario) normalmente se representan en un espacio Euclídeo de una determinada dimensión. En dichas condiciones un punto es una entidad matemática p = (x; y), donde (x; y) 2 R2.

En el soporte aritmético de la computadora, dicha representación se efectúa con los tipos de datos provistos, que pueden ser números reales con punto rotante de simple o doble precisión. Este espacio se denomina espacio de la escena y es uno de los muchos espacios que se utilizaran para factorizar adecuadamente las diversas tareas de un sistema gráfico.

Por último, en el soporte gráfico del búfer de pantalla, un punto se representa con un píxel, y dicha representación se efectúa seteando una posición de memoria con un contenido dado.

Este espacio se denomina espacio de pantalla y se direcciona a partir del sistema de coordenadas físico, cuyo origen es el vértice superior izquierdo. Es posible encontrar varias correspondencias posibles entre el sistema de coordenadas físico y un sistema de coordenadas arbitrario en el espacio de la escena. En la literatura normalmente se considera que un píxel es un \punto con extensión en el espacio de la escena, y por lo tanto el origen de dicho espacio coincide con el vértice superior izquierdo del píxel (0,0) (ver por ejemplo [33, 40, 66, 84]). Desde nuestro punto de vista, esto no es del todo correcto, y parece mas adecuado pensar que el origen del sistema de coordenadas de la escena esta en el centro del píxel (0,0) (ver Figura 5.1).

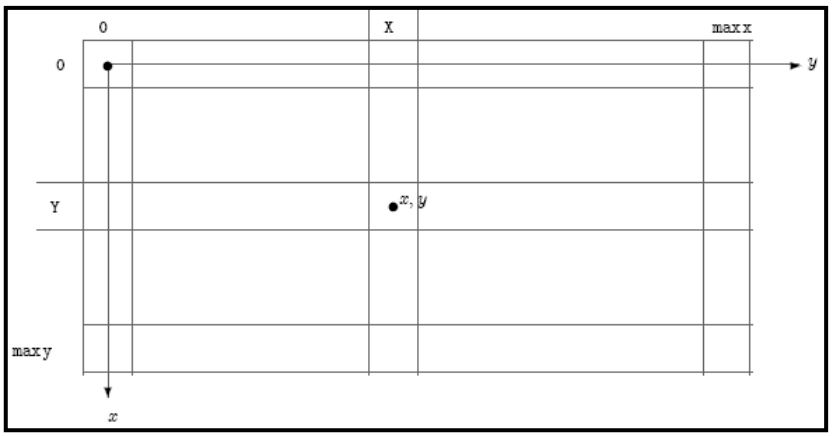


Figura 5.1 *Sistema de coordenadas físico junto al espacio de la escena.*

5.2.2.PRIMITIVAS GRÁFICAS

Es muy difícil escoger un conjunto de primitivas gráficas que sea adecuado para la representación de todo tipo de entidades gráficas. Sin embargo, el siguiente subconjunto en la práctica resulta suficiente:

- **Puntos**: Se especifican a partir de su localización y color. Su discretización es directa. (ver Figura 5.2).
- **Segmentos de recta:** Son esenciales para la mayor parte de las entidades. Se especifican a partir de un par de puntos que representan sus extremos.

Figura 5.2 *Segmento de recta.*

- **Circunferencias:** En algunos casos representar entidades curvadas con segmentos poligonales puede ser inadecuado o costoso, por lo que en la practica las circunferencias o círculos se adoptan como primitivas. Se especifican con la posición de su centro y su radio. (ver Figura 5.3).

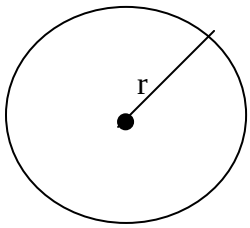


Figura 5.3 *Circunferencia.*

- **Polígonos**: Son indispensables para representar entidades sólidas. Se representan a partir de la secuencia de puntos que determina la poligonal de su perímetro.

5.2.3.ESPECIFICACIONES DE UNA DISCRETIZACIÓN

En el momento de escoger un método de discretización para una primitiva gráfica, es indispensable contar con criterios que permitan evaluar y comparar las ventajas y desventajas de las distintas alternativas. Entre todas las especificaciones posibles, podemos mencionar las siguientes:

- **Apariencia**: Es la especificación mas obvia, aunque no es fácil describirla en términos formales.

  Normalmente se espera que un segmento de recta tenga una apariencia recta más allá de que se hallan escogido los píxeles matemáticamente mas adecuados. Tampoco debe tener discontinuidades o puntos espureos. Debe pasar por la discretización del primer y ultimo punto del segmento. Debe ser uniforme, etc.

- **Simetría e invariancia geométrica**: Esta especificación se refiere a que un método de discretización debe producir resultados equivalentes si se modifican algunas propiedades geométricas de la primitiva que se esta discretizando. Por ejemplo, la discretización de un segmento no debe variar si dicho segmento se traslada a otra localización en el espacio, o si es rotado, etc.

- **Simplicidad y velocidad de computo**: Como los métodos tradicionales de discretización de primitivas se desarrollaron hace tres décadas, en momentos en que las posibilidades del hardware y software eran muy limitadas, los resultados eran muy sensibles al uso de memoria u operaciones aritméticas complejas. Por lo tanto, los métodos tienden a no depender de estructuras complejas y a ser directamente implementables en hardware específico de baja complejidad.

5.2.4.MÉTODOS DE DISCRETIZACIÓN

Dada una primitiva gráfica a discretizar, debemos encontrar los píxeles que la representen de la manera más correcta posible. Para ello, lo más adecuado es caracterizar matemáticamente a dicha primitiva, de modo que su discretización pueda efectuarse en forma sencilla. Entre los diversos métodos que pueden plantearse destacamos los dos siguientes:

- **Evaluar su ecuación diferencial a diferencias finitas**: Este método, denominado DDA (discrete diference analyzer) consiste en plantear la ecuación diferencial de la primitiva a discretizar, y luego evaluar dicha expresión a intervalos adecuados.
- **Análisis del error**: Estos métodos fueron desarrollados por Bressenham y se basan en analizar, dado un píxel que pertenece a la discretización de la primitiva, cual es el próximo píxel que minimiza una determinada expresión que evalúa el error que comete la discretización.

5.3 DISCRETIZACIÓN DE SEGMENTOS DE RECTAS

El análisis de los métodos de discretización de rectas parte de considerar el comportamiento esperado en determinados casos particulares. Dichos casos surgen de suposiciones específicas que simplifican el problema, pero que al mismo tiempo se pueden generalizar a todos los de más casos por medio de simetrías. Dado un segmento de recta que va de (x0; y0) a (x1; y1), se supone que

$$\Delta x = (x1 , x0) ¸ 0,$$

$$\Delta y = (y1 , y0) ¸ 0, y$$

$$\Delta x \geq \Delta y.$$

Esto equivale a trabajar en el "octavo" del espacio de pantalla sombreado en la Figura 5.4, donde el origen es el píxel que corresponde a la discretización del punto (x0; y0) y la zona sombreada a los lugares donde puede ubicarse el punto (x1;y1).

▪ SEGMENTOS DE RECTA DDA

Como ya se mencionara, los métodos DDA evalúan la ecuación diferencial de la primitiva a intervalos finitos. En el caso particular de los segmentos de recta, la ecuación diferencial es

$$\frac{\partial y}{\partial x} = \frac{\Delta y}{\Delta x} = m$$

El método busca encontrar una secuencia de n + 1 puntos tales que (x0;y0) = (x0;y0); (x1;y1); **....** ; (xn; yn) = (x1;y1). La discretización de cada uno de ellos son los píxeles de la discretización del segmento. Esta propiedad, si bien es trivial, es de gran importancia porque determina que la discretización de un segmento de recta es invariante frente a transformaciones afines. Esto significa que es equivalente transformar los extremos del segmento y discretizar el segmento transformado, o discretizar primero y transformar cada punto obtenido. Sin embargo, la primera alternativa es mucho más eficiente.

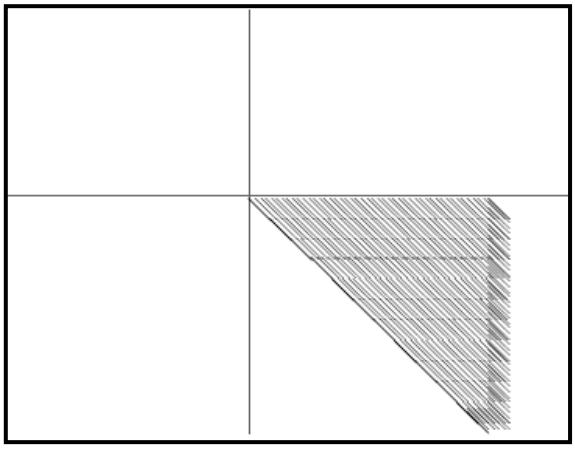


Figura 5.4 *Situación particular para derivar los algoritmos de discretización de segmentos de recta.*

Dada la ecuación diferencial y un incremento finito arbitrario $\xi = \frac{1}{n}$ podemos pasar de un píxel dado de la secuencia al siguiente por medio de la expresión

$$\begin{cases} x^{k+1} = x^k + \xi \, \Delta x \\ y^{k+1} = y^k + \xi \, \Delta y \end{cases}$$

ξ determina la "frecuencia" de muestreo del segmento. Un valor muy pequeño determina que muchas muestras producirán puntos que serán discretizados al mismo píxel. Por el contrario, un valor muy grande determinará que el

segmento aparezca "punteado" en vez de ser continuo como corresponde. Un valor práctico es elegir ξx = 1 y por lo tanto n = Δx, es decir, la discretización tiene tantos píxeles como longitud tiene el segmento en la variable que más varía (más uno, dado que la secuencia tiene n + 1 puntos). Al mismo tiempo es fácil ver que = $\xi \Delta y = \frac{\Delta y}{\Delta x} = m$

En la Figura 5.5 es posible ver el resultado de discretizar un segmento particular por el método DDA. Obsérvese que los puntos extremos (x0; y0) a (x1;y1) son en efecto puntos y por lo tanto están ubicados en cualquier lugar dentro del píxel que corresponde a su discretización. Un algoritmo sencillo que computa la discretización de un segmento de recta por el método DDA se muestra en la Figura 5.6. Obsérvese que es necesario computar las variables en punto flotante, y que además se requiere una división en punto flotante.

Para poder discretizar un segmento de recta en cualquiera de las posibilidades es necesario considerar las simetrías que se aplican. Si por ejemplo no se cumple que y = (y1 ; y0) ¸ 0, entonces hay que considerar pendientes negativas (simetría A), caso que el algoritmo de la Figura 5.6 realiza automáticamente. En cambio, si Δx = (x1; x0) ¸ 0, entonces es necesario decrementar a x en el ciclo e iterar mientras no sea menor que x1 (simetría B). Por último, si no se cumple que Δx ¸ Δy, entonces es necesario intercambiar los roles de las variables x e y (simetría C).

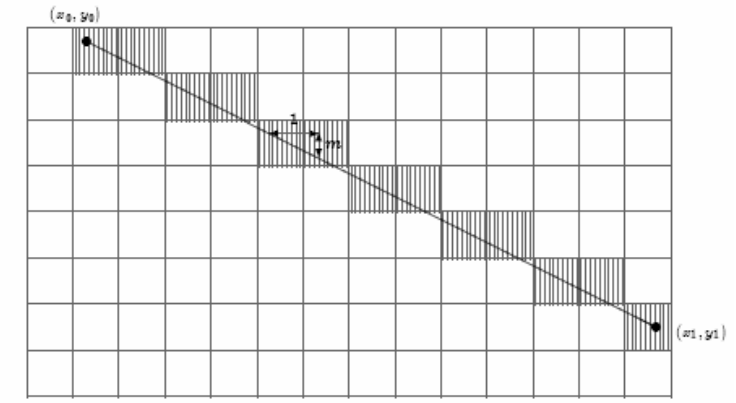


Figura 5.5 *La discretización de un segmento por DDA*

```
Public Sub dda(x1 As Integer, Y1 As Integer, X2 As Integer, y2 As Integer)
Dim dx, dy, step, k As Integer
Dim x_aum, y_aum, X, Y As Integer
dx = X2 - x1
dy = y2 - Y1

   If Abs(dx) > Abs(dy) Then
      step = Abs(dx)
   Else
      step = Abs(dy)
   End If

x_aum = dx / step
y_aum = dy / step
X = Round(x1)
Y = Round(Y1)
PSet (X, Y), QBColor(Rnd * 10)

For k = 1 To step
   X = X + x_aum
   Y = Y + y_aum
   PSet (Round(X), Round(Y)), QBColor(Rnd * 15)
Next k
End Sub
```

Figura 5.6 *Algoritmo DDA para segmentos de recta en visual Basic 6.0.*

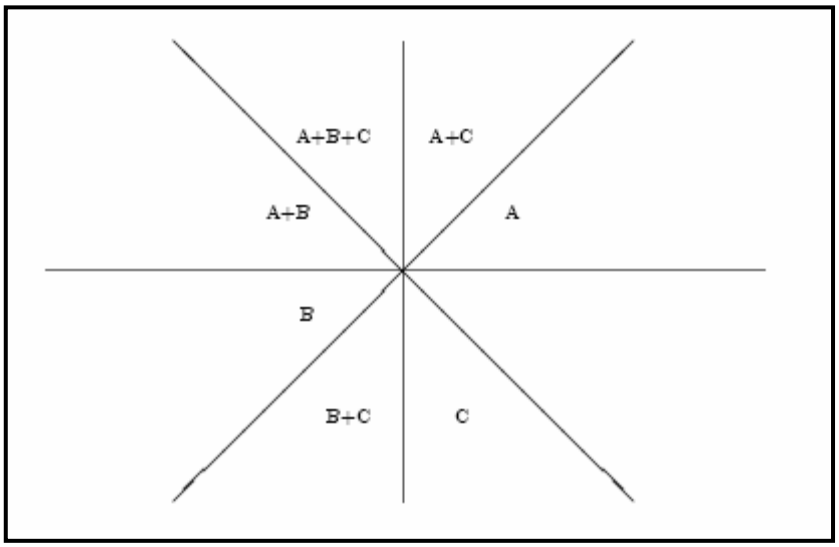


Figura 5.7 *Simetría para los demás casos*

▪ SEGMENTOS DE RECTAS POR BRESSENHAM

En el algoritmo DDA para segmentos de recta es necesario computar sumas entre las variables en punto flotante, y además se requiere una división en punto flotante para computar la pendiente.

El mérito del algoritmo que vamos a presentar consiste en que todas las operaciones se realizan en aritmética entera por medio de operaciones sencillas, y por lo tanto, su ejecución es más rápida y económica, y es de fácil implementación con hardware específico.

El punto de partida del análisis es el siguiente. Si la discretización de los puntos extremos del segmento debe pertenecer a la discretización del segmento, entonces es conveniente efectuar la llamada al algoritmo luego de discretizar los extremos. Esto significa que (x0;y0) y (x1; y1), y por lo tanto Δx y Δy son enteros. Luego, si p es un píxel que pertenece a la discretización del segmento, entonces en las condiciones particulares mencionadas más arriba, el próximo píxel solamente puede ser el ubicado a la derecha (E o \hacia el este"), o el ubicado en diagonal hacia la derecha y hacia abajo (D o \en diagonal", ver Figura 5.8).

La decisión de ir hacia el paso E o D se toma en función del error que se comete en cada caso.

En este algoritmo se considera que el error es la distancia entre el centro del píxel elegido y el segmento de recta, medida en dirección del eje Y positivo del espacio de pantalla (es decir, hacia abajo). Si el error en p fuese cero, entonces al ir hacia E el error pasa a ser m (la pendiente del segmento), y en D el error pasa a ser m - 1 (ver Figura 5.9).

En general, si en p el error es e, la actualización del error es

Paso E

e := e + m

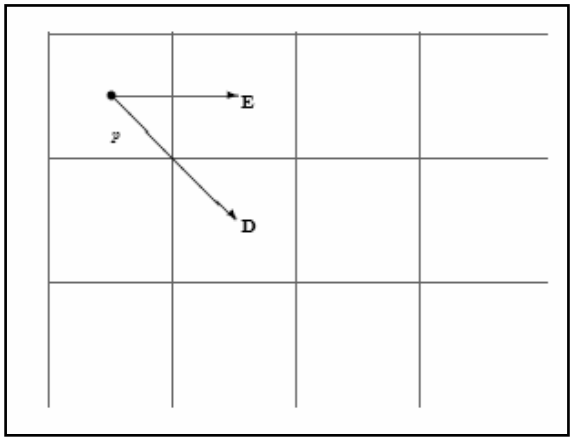


Figura 5.8 *Si p pertenece a la discretización, el próximo píxel solo puede ser E o D.*

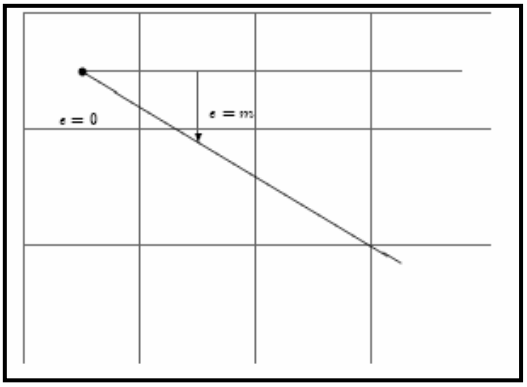


Figura 5.9 *Actualización del error.*

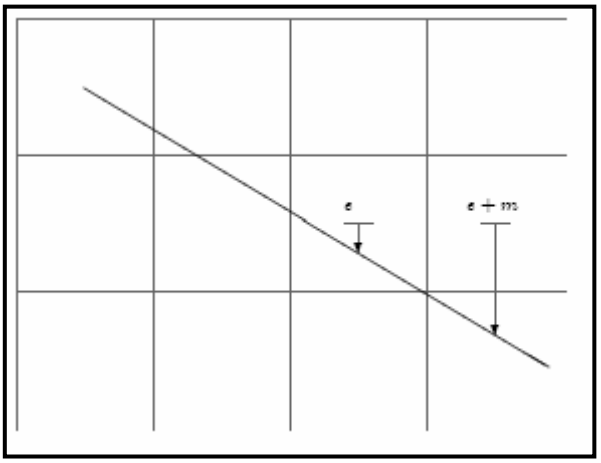


Figura 5.10 *Elección del próximo píxel.*

- Paso D

e := e + m i 1

Por lo tanto, la elección del paso E o D depende de que el valor absoluto de e+m sea o no menor que el valor absoluto de e + m - 1. Expresado de otra manera, sea e el error en un determinado píxel. Si e +m> 0:5 entonces el segmento de recta pasa más cerca del píxel D, y si no, pasa más cerca del píxel E (ver Figura 5.10).

Una de las economías de cómputo del método proviene de poder testear el error preguntando por cero. Es fácil ver que si se inicializa el error en

$$e_o = m - 0:5;$$

Entonces en cada paso hay que chequear e >0 para elegir D. La otra economía proviene de realizar manipulaciones algebraicas para efectuar un cómputo equivalente pero en aritmética entera. Como se testea el error por cero, multiplicar el error por una constante no afecta el resultado. Por lo tanto, multiplicamos el error por 2Δx. A partir de dicho cambio, se constatan las siguientes igualdades:

$e_o = 2\Delta x(\frac{\Delta y}{\Delta x} - 0.5) = 2\Delta y - \Delta x.$

Paso E

e := e + 2Δy.

Paso D

e := e + 2(Δy - Δx).

Todas las operaciones, entonces, se efectúan en aritmética entera.

```
procedure linea(x0,x1,y0,y1,col: integer);
var x,y,dx,dy,e,ix,iy:integer;
begin
dx := x1 - x0; dy := y1 - y0;
ix := 2*dx; iy := 2*dy;
y := y0; e := iy - dx;
for x := x0 to x1 do begin
```

```
putpixel (x,y,col);
e := e+iy;
if e>0 then do begin
y := y+1;
e := e-ix;
end;
end;
end;
```

Figura 5.11 *Algoritmo de Bressenham para segmentos de recta.*

La implementación del algoritmo de Bressenham para segmentos de recta se muestra en la Figura 5.11.

5.4 DISCRETIZACIÓN DE CIRCUNFERENCIAS

Como en el caso de los segmentos de recta, en la discretización de circunferencias o círculos trabajaremos solamente en una parte de la misma, y obteniendo las demás partes por simetría.

Supongamos que la circunferencia a discretizar está centrada en el origen, y que p = (x;y) es un píxel de su discretización tal que

$x \geq 0$,

$y \geq 0$,

$x \geq y$.

Dicha situación describe un octavo de la circunferencia, pero las partes faltantes pueden encontrarse por medio de las simetrías mostradas en la Figura 5.12

▪ DISCRETIZACIÓN DE CIRCUNFERENCIAS POR DDA

Sea una circunferencia de radio r centrada en el origen y sea p = (x;y) un punto que pertenece a la misma. Entonces, la ecuación diferencial de la circunferencia en dicho punto (ver Figura 2.16) es

$$\frac{\partial y}{\partial x} = -\frac{x}{y}$$

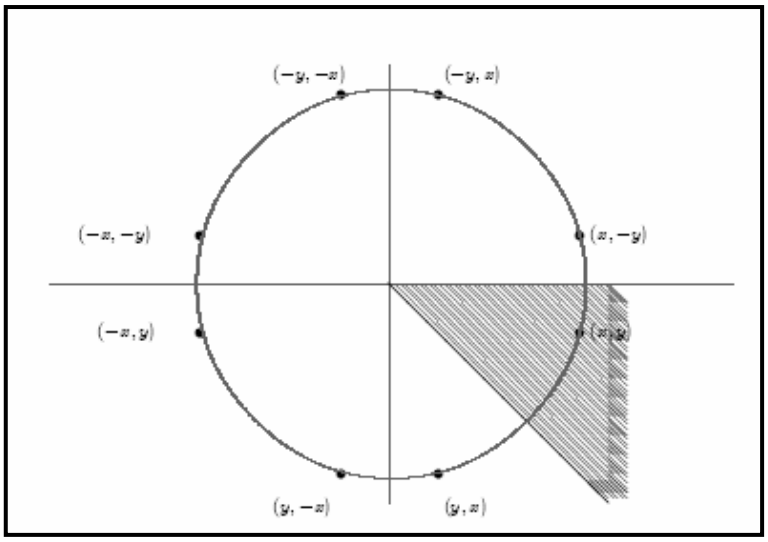


Figura 5.12 *Simetrías para la discretización de circunferencias.*

La evaluación de la ecuación diferencial por diferencias finitas, como en el caso de los segmentos de recta, consiste en encontrar una secuencia de valores, de modo que dado un píxel de la discretización (xk;yk), el próximo píxel se encuentra con

$$\begin{cases} xk + 1 = xk - \xi\, y\, k \\ yk + 1 = y\, k + \xi\, x\, k \end{cases}$$

ξ determina la "frecuencia" de muestreo de la circunferencia. Valores pequeños determinan un cómputo redundante, y valores muy grandes determinan un cubrimiento desparejo. Un valor práctico es elegir "x = 1 y por lo tanto ξ y = $\frac{y}{x}$ (ver Figura 5.13). El primer píxel de la secuencia es po = (r;0), el cual pertenece a la discretización de la circunferencia y además se computa en forma directa.

Es importante tener en cuenta que el sistema de ecuaciones 2.1 puede representarse como una transformación: ·

$$\begin{bmatrix} x_k + 1 \\ y_k + 1 \end{bmatrix} = \begin{bmatrix} 1 & \xi \\ -\xi & 1 \end{bmatrix} * \begin{bmatrix} x_k \\ y_k \end{bmatrix}$$

La cual tiene determinante 1+ ξ^2. Esto significa que la discretización es levemente espiralada hacia afuera. Una forma de solucionar este problema es utilizar una recurrencia cuya transformación sea

$$\begin{bmatrix} 1 & \xi \\ -\xi & 1 - \xi^2 \end{bmatrix}$$

Esta transformación representa en realidad una elipse y no una circunferencia, pero la excentricidad de la misma es del orden de ξ^2 y por lo tanto la diferencia es más reducida. Utilizar dicha expresión altera la evaluación de yk+1:

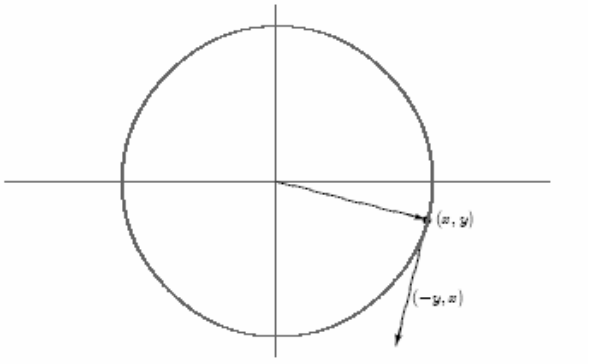


Figura 5.13 *Interpretación geométrica de la ecuación diferencial de la circunferencia centrada en el origen.*

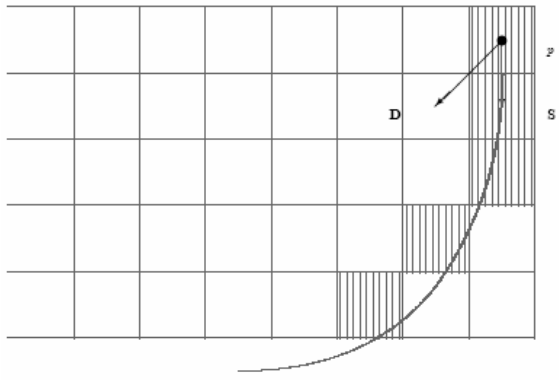


Figura 5.14 *Si p pertenece a la discretización, el próximo píxel solo puede ser S o D.*

$$yh + 1 = y\,k\,(1 - \varepsilon^2) - \varepsilon x_h$$
$$yh + 1 = y\,k - \varepsilon(\varepsilon yk - xh)$$
$$yk + 1 = yh - \varepsilon x_{h+1}$$

Por lo tanto, la recurrencia para computar la discretización de una circunferencia centrada en el origen con DDA queda modificada a

$$\begin{cases} x_{k+1} = x_k - \varepsilon y_k \\ y_{k+1} = y_k + \varepsilon x_{k+1} \end{cases}$$

De todas maneras, la desventaja más importante de este algoritmo es que debe realizar una división en punto flotante para cada paso.

▪ DISCRETIZACIÓN DE BRESSENHAM PARA CIRCUNFERENCIAS

Como en el caso planteado para segmentos de recta, este método se basa en analizar el error entre la verdadera circunferencia y su discretización. Si p pertenece a la discretización, el próximo píxel de la discretización solo puede ser S (\ir hacia el sur") o D (\ir en diagonal", ver Figura 5.14).

El error (en este caso la distancia al cuadrado) de un píxel p = (x;y) a una circunferencia de radio r con centro en (0; 0) es:

$\xi = x^2 + y^2 - r^2$:

```
Public Sub bres_circulo(x_centro As Integer, y_centro As Integer, radio As Integer)
Dim P, Y As Integer
Dim X As Integer
X = 0
Y = radio
P = 3 - 2 * radio
While X < Y
  punto_circulo x_centro, X, y_centro, Y
   If P < 0 Then
      P = P + 4 * X + 6
   Else
   P = P + 4 * (X - Y) + 10
   Y = Y - 1
   End If
```

```
   X = X + 1
Wend
If X = Y Then
punto_circulo x_centro, X, y_centro, Y
End If
End Sub
```

Figura 5.15 *Algoritmo de Bressenham para circunferencias.*

Si elegimos el paso S entonces el próximo píxel es pS = (x;y +1), y el error pasa a ser

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

Si elegimos el paso D entonces el próximo píxel es ps = (x, y +1), y el error pasa a ser

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

La elección de un paso S o D dependerá de cuál error tiene menor módulo:

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

Teniendo en cuenta que tanto en D como en S se incrementa y, entonces se actualiza el error a e_S antes de la comparación.

$$\text{Si } |e| > |e - 2x + 1| \text{ entonces D:}$$

Por último, teniendo en cuenta que -2x + 1 es siempre negativo (recordar que x > 0; x >y), entonces:

Si 2e >2x -1 entonces D:

De esa manera, el algoritmo queda planteado exclusivamente con operaciones enteras y de aritmética sencilla (ver Figura 5.15)

5.5 DISCRETIZACIÓN DE POLÍGONOS

El objetivo de la discretización de polígonos es encontrar el conjunto de píxeles que determinan el área sólida que dicho polígono cubre en la pantalla. Si bien existen diversos métodos, se presenta el más difundido, que se basa en encontrar la intersección de todos los lados del polígono con cada línea de barrido (a y constante), por lo que el método se denomina conversión scan del

polígono. Este método es de gran importancia porque se generaliza a una clase de algoritmos denominados scan-line para resolver determinados problemas de iluminación y sombreado en tres dimensiones.

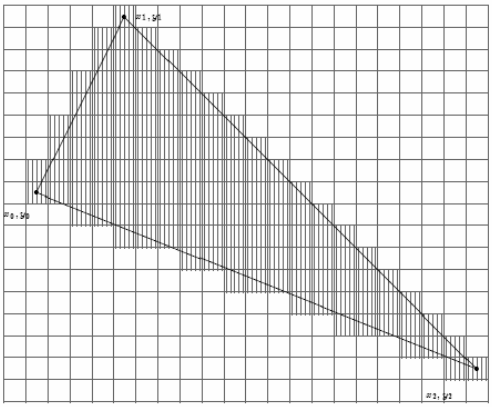


Figura 5.16 *Conversión scan de un polígono.*

Todo polígono plano puede descomponerse en triángulos. Por lo tanto el triángulo sería la base del análisis de la conversión scan de polígonos en general. Para computarla es necesario dimensionar dos arreglos auxiliares de enteros minx, maxx que para cada línea de barrido almacenarán el menor y mayor x respectivamente (ver Figura 5.16).

Falta algoritmo de discretización de polígonos.

5.6 RELLENOS

Antes de definir los rellenos, hablaré un poco de lo que son los sistemas de rastreo.

"Varios registros se ponen a disposición de los procesadores en un sistema rastreador a fin de almacenar posiciones coordenadas y diversas instrucciones. Puede utilizarse cuatro registros para contener valores coordenados de

extremos de líneas, parámetros de circunferencias y elipses, y posiciones de cadenas de caracteres o marcadores."[1]

La ventaja de estos sistemas es su facultad de almacenar fácilmente, así como de desplegar áreas llenas de color o de un modelo de sombreado. Los modelos de llenado de estas áreas se almacenan como valores de color o de intensidad en un buffer de cuadros. El despliegue de áreas sombreadas en un sistema vectorial es considerablemente más complicado, ya que el llenado de un área requiere el trazado de segmentos de líneas dentro de la frontera del área durante cada ciclo de renovación.

Varios algoritmos nos permiten realizar esta propiedad, uno de ellos es el método que hace uso de la definición de la frontera con el fin de identificar qué píxeles pertenecen al interior de un área. Otros métodos comienzan desde una posición en el interior del área y pintan hacia fuera.

1 Graficas por Computadora, pag. 77

RESUMEN

Son cada vez más comunes los sistemas para elaboración de presentaciones comerciales, incluyendo cartillas gráficas, diagramación automática, y publicación electrónica en general.

Los sistemas de computación pueden adaptarse a aplicaciones de las gráficas en varias formas, conforme a los recursos de hardware y software de que disponen.

Los resultados gráficos de una aplicación pueden mostrarse en una gran variedad de dispositivos de salida.

Dispositivos de rastre**,** estos reciben de la computadora la información de una serie de píxels, los cuales son posicionados en forma contigua.

Actualmente estos dispositivos son mas caros, pero tienen ciertas ventajas que los hacen únicos. Son ideales para la graficación rápida y precisa de planos.

Primitivas gráficas

Es muy difícil escoger un conjunto de primitivas gráficas que sea adecuado para la representación de todo tipo de entidades gráficas. Su discretización es directa. **Segmentos de recta, Circunferencias, Polígonos**

En el momento de escoger un método de discretización para una primitiva gráfica, es indispensable contar con criterios que permitan evaluar y comparar las ventajas y desventajas de las distintas alternativas. Debe pasar por la discretización del primer y último punto del segmento. Debe ser uniforme, etc.

Por ejemplo, la discretización de un segmento no debe variar si dicho segmento se traslada a otra localización en el espacio, o si es rotado, etc.

Dada una primitiva gráfica a discretizar, debemos encontrar los píxeles que la representen de la manera más correcta posible.

Análisis del error: Estos métodos fueron desarrollados por Bressenham y se basan en analizar, dado un píxel que pertenece a la discretización de la

primitiva, cual es el próximo píxel que minimiza una determinada expresión que evalúa el error que comete la discretización.

El análisis de los métodos de discretización de rectas parte de considerar el comportamiento esperado en determinados casos particulares.

BIBLIOGRAFÍA:

LIBROS:

Donald Eran / M. Pauline Baker; "GRÁFICAS POR COMPUTADORA",

Caludio Delriexus ; "INTRODUCCIÓN A LA COMPUTACIÓN GRÁFICA"

INTERNET:

http://articulos.conclase.net/libreriabgi/LibreriaBGI1.html

http://galia.fc.uaslp.mx/~medellin/Gr1/dda.pas

http://www.cannes.itam.mx/Alfredo/Espaniol/Cursos/Grafica/Linea.pdf

http://www.udlap.mx/~is115728/graf-importancia.html