2 ALGORITMOS PARA PRIMITIVAS

2.1 Algoritmos para el trazo de líneas

La ecuación de intersección de la pendiente cartesiana de una línea recta es

$$\mathbf{Y = m * x + b} \qquad \mathbf{(2\text{-}1)}$$

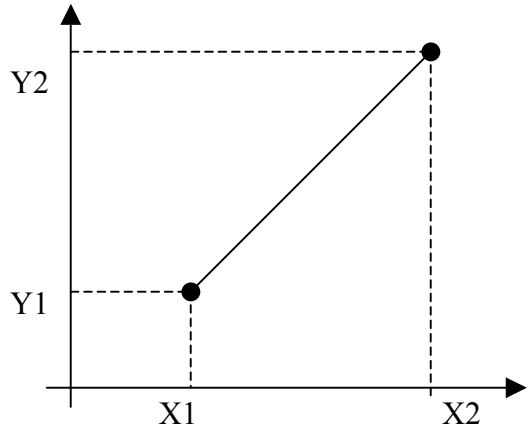


Figura 2.1 trayectoria de la línea entre las posiciones de extremos (x1,y1) (x2,y2)

donde m representa la pendiente de la línea y b la intersección de y dado que los dos extremos de un segmentos de líneas se especifica en las posiciones $(x_1,y_1)y(x_2,y_2)$, como se muestra en la figura podemos determinar los valores para la pendiente y la intersección de y en b con los cálculos siguientes

$$\mathbf{m = y_2 - y_1 / x_2 - x_1} \qquad \mathbf{(2\text{-}2)}$$

$$\mathbf{b = y_1 = m * x_1} \qquad \mathbf{(2\text{-}3)}$$

Los algoritmos para desplegar líneas rectas se basan en la ecuación de línea 2-1 y los cálculos que se dan en las ecuaciones 2-2 y 2-3.

Para cualquier x dentro del intervalo Δx a lo largo de una línea, se puede calcular el intervalo correspondiente Δy de y a partir de la ecuación 2-2 como

$$\mathbf{\Delta y = m\Delta x} \qquad \mathbf{(2\text{-}4)}$$

De modo similar, podemos obtener el intervalo Δx de x correspondiente a una Δy especifica como

$$\mathbf{\Delta x = \Delta y / m} \qquad \mathbf{(2\text{-}5)}$$

2.2 Algoritmo DDA

El analizador diferencial digital (DDA; digital diferencial analyzer) es un algoritmo de línea de conversión de rastreo que se basa en el calculo ya sea de Δy, o de Δx por medio de las ecuaciones 2-4 o 2-5. Efectuamos un rastreo de línea en intervalos unitarios de una coordenada y determinamos los valores enteros correspondientes mas próximos a la trayectoria de la línea para la otra coordenada. Considere primero una línea de pendiente positiva, como se ilustra en la figura anterior. Si la pendiente es menor o igual que 1, llevamos a cabo un muestreo de x en intervalos unitarios (Δx=1) y calcularemos cada valor sucesivo de y como

(2-6)

$$y_{k+1} = y_k + m$$

El subindice k toma valores enteros a partir de 1 y aumenta a razón de 1 hasta que se alcanza el valor final. Ya que m puede ser cualquier número real entre 0 y 1, los valores calculados de y deben redondearse al entero más cercano. Para líneas con pendiente positiva mayor que 1, se revierten las funciones de x y y. Es decir, se realiza un muestreo de y en intervalos unitarios (Δy = 1) y calculamos cada valor sucesivo de x como

$$x_{k+1} = x_k + \frac{1}{m}$$

(2-7)

Las ecuaciones 2-6 y 2-7 se basan en la suposición de que las líneas deben procesarse del extremo izquierdo al derecho si este procesamiento se revierte, de manera que sea el extremo derecho donde se inicia, entonces tenemos ya sea Δx = -1 y

$$y_{k+1} = y_k - m$$

(2-8)

o (cuando la pendiente es mayor que Δy = -1 con

$$x_{k+1} = x_k - \frac{1}{m}$$

(2-9)

Este algoritmo se resume en el procedimiento siguiente, que acepta como entrada las dos posiciones de píxel de los extremos. Las diferencias horizontal y vertical entre las posiciones de los extremos se asignan a los parámetros de dx y dy. Al iniciar con la posición de píxel (Xa,Ya), determinamos la compensación necesaria para generar la posición del píxel siguiente a lo largo de la trayectoria de la línea. Realizamos el ciclo de

este proceso **steps.** Si la magnitud de dx es mayor que la de dy y Xa es menor que Xb, los valores de los incrementos en las direcciones de X y Y son 1 y m, respectivamente. Si la variación mas alta se en la dirección de X pero Xa es mayor que Xb, entonces los decrementos –1y-m sirven para generar cada nuevo punto en la línea. De otra manera, utilizamos un incremento(o decremento) de 1/m en la dirección de X. Suponemos que los puntos se debe trazar en un sistema de intensidad de dos niveles, de modo que la solicitud de **setPixel** será con un valor de intensidad de 1.

```
procedure lineDDA(xa,ya,xb,yb:integer);
var

      crement,yIncrement,x,y:real;


         b-ya;
                 bs(dy)then steps:=abs(dx)
                  bs (dy);

                              ;


           xel(round(x),round(y),1);
             do steps do


               Increment;
                             ound(y),1);
```

El algoritmo DDA es un método para calcular posiciones de píxel, que es mas rápido que la aplicación directa de la ecuación 2-1

2.3 Algoritmo de línea de Bresenham

Para ilustrar el planteamiento de Bresenham primero consideramos el proceso de conversión de rastreo para líneas con pendiente positiva menor que 1. Al realizar el muestreo en la posición X_k+1, designamos las separaciones de píxel verticales de la trayectoria de la línea matemática como d_1 y d_2. La coordenada de Y en la línea matemática en la posición de la columna de píxel X_{k+1} se calcula como

$$\mathbf{Y = m\,(X_k + 1\,) + b} \qquad \textbf{(2-10)}$$

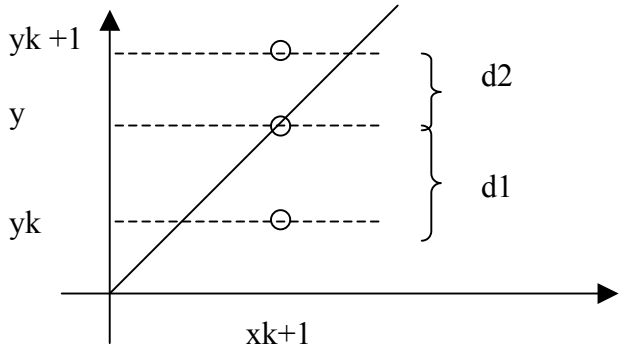


Figura 2.2 Distancia entre la posiciones de píxel y la coordenada de y de la linea en la posición xk + 1.

Entonces

$$\mathbf{d_1 = Y - Y_k}$$
$$\mathbf{= m\,(X_k + 1) + b - Y_k}$$

y

$$\mathbf{d_2 = (Y_k + 1) - Y}$$
$$\mathbf{= Y_k + 1 - m\,(X_k + 1) - b}$$

La diferencia entre estas dos separaciones es

$$\mathbf{d_1 - d_2 = 2\,m\,(X_k + 1) - 2\,Y_k + 2\,b - 1} \qquad \textbf{(2-11)}$$

Un parámetro de decisión P_k para el k-esimo paso en el algoritmo de línea se puede obtener al redondear la ecuación la ecuación 2-11, de modo que implique solo cálculos de enteros. Realizamos esto al sustituir $m=\Delta y/\Delta x$ donde Δy y Δx son las separaciones vertical y horizontal de las posiciones de los extremos y al definir.

$$\mathbf{P_k = \Delta x\,(d_1 - d_2)}$$
$$\mathbf{= 2\,\Delta y.\,X_k - 2\,\Delta x.\,Y_k + c} \qquad \textbf{(2-12)}$$

El signo de P_k es el mismo que el de d_1-d_2, puesto que $\Delta x > 0$ para nuestro Ej. El parámetro c es constante y tiene el valor $2\Delta y + \Delta x\,(2b - 1)$, que es el independiente del píxel si Y_k esta mas cerca de la trayectoria de la línea que el píxel $Y_k + 1$ (es decir, $d_1 < d_2$), entonces el parámetro de decisión P_k es negativo. En ese caso, trazamos el Píxel inferior; de otro modo, trazamos el píxel superior.

Los cambios de coordenadas a lo largo de la línea ocurren en pasos unitarios ya sea en la dirección de x o en la de y. Por tanto, es posible obtener los valores de parámetros de decisión sucesivos al utilizar cálculos incrementales en enteros. En el paso K+1, el parámetro de decisión se evalúa con base en la ecuación 2-12 como

$$P_{k+1} = 2\Delta y * X_k - 2\Delta x * Y_{k+1} + C$$

Al sustraer la ecuación 2-12 de la ecuación anterior, tenemos

$$P_{k+1} - P_k = 2\Delta y \ (X_{k+1} + X_k) - 2\Delta x \ (Y_{k+1} - Y_k)$$

Pero $X_{k+1} = X_k + 1$ de tal manera que

$$P_{k+1} = P_k + 2\Delta y - 2\Delta x \ (Y_{k+1} - Y_k) \qquad (2\text{-}13)$$

Donde el termino $Y_{k+1} - Y_k$ sea 0, o bien, 1dependiendo del parámetro P_k.

Este calculo recursivo de los parámetros de decisión entera de X, empezando en el extremo izquierdo de las coordenadas de la línea. El primer parámetro, se evalúa a partir de la ecuación 2-12 en la posición de píxel inicial (Xo,Yo) y con m evaluada como $\Delta y / \Delta x$:

$$P_o = 2\Delta y - \Delta x \qquad (2\text{-}14)$$

Podemos resumir el trazo de línea de Bresenham para una línea con una pendiente positiva menor que1 en los pasos que se indican en seguida. Las constantes $2\Delta y$ y $2\Delta y - 2\Delta x$ se calculan una vez para cada línea que se debe convertir mediante rastreo, de manera que el proceso aritmético solo implique la adición y sustracción de enteros de estas constantes.

2.3.1 Algoritmo de Bresenham para el trazo de líneas para |m| <1

1.- Se capturan de los dos extremos de la línea y se almacena el extremo izquierdo en(Xo,Yo).
2.- Se carga (Xo,Yo) en el búfer de estructuras; es decir, se traza el primer punto.
3.- Se calculan las constantes Δx, Δy, $2\Delta y$ y $2\Delta y - 2\Delta x$ y se obtiene el valor inicial para el parámetro de decisión como
$P_o = 2\Delta y - \Delta x$
4.- En cada X_k a lo largo de la línea, que indica en k=0 se efectuara la prueba siguiente: si $P_k < 0$,el siguiente punto que se debe trazar es (X_{k+1}, Y_k) y $P_{k+1} = P_k + 2\Delta y$
De otro modo, el siguiente punto que se debe trazar es (X_{k+1}, Y_{k+1}) y $P_{k+1} = P_k + 2\Delta Y - 2\Delta X$
5.-Se repite el paso $4\Delta X$ veces.

2.3.2 Ejemplo 3-1 Trazo de líneas de Bresenham

Para ilustrar el algoritmo, digitalizamos la línea con los extremos(20,10) y (30,18).Esta línea tiene una pendiente de 0.8, con $\Delta x = 10$, $\Delta y = 8$

El parámetro de decisión inicial tiene el valor $P_o = 2\Delta y - \Delta x$ es = 6 y los incrementos para calcular parámetros de decisión sucesivos son: $2\Delta y = 16$, $2\Delta y - 2\Delta x = -4$

Trazamos el punto inicial (Xo,Yo) = (20,10) y determinamos las posiciones del píxel sucesivas a lo largo de la trayectoria de la línea a partir del parámetro de decisión como

K	PK	(Xk+1,Yk+1)	K	PK	(Xk+1,Yk+1)
0	6	(21,11)	5	6	(26,15)
1	2	(22,12)	6	2	(27,16)
2	-2	(23,12)	7	-2	(28,16)
3	14	(24,13)	8	14	(29,17)
4	10	(25,14)	9	10	(30,18)

Un trazo de píxeles se genera a lo largo de la trayectoria de esta línea, como se ilustra en la Fig.

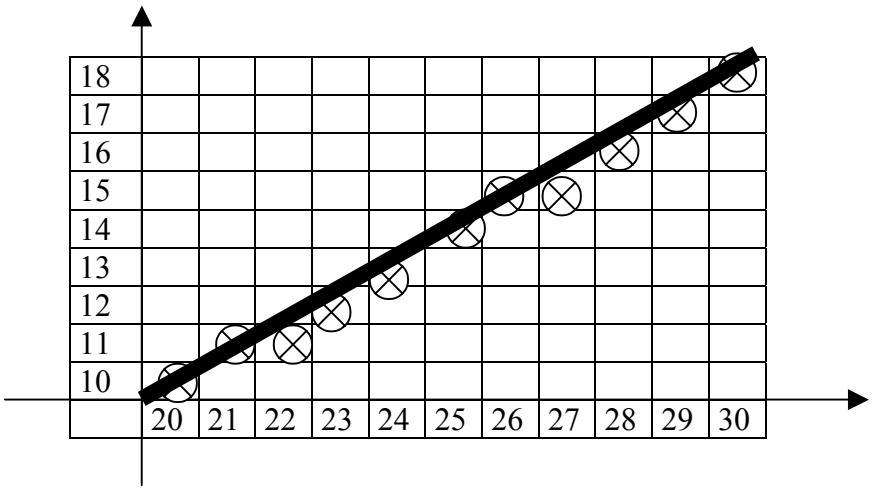


Figura 2.3 Posiciones de pixel a lo largo de la trayectoria de la línea entre (20,10) y (30,18) trazadas con el algoritmo de línea de Bresenham.

En el procedimiento siguiente, se presenta una implementación del trazo de líneas de Bresenham para pendiente en el rango 0<m<1. Las posiciones del píxel de los extremos para la línea se pasan a este procedimiento y los píxels se trazan del extremo izquierdo al

derecho. La solicitud de setPixel carga el valor de intensidad en el búfer de estructura en la posición especifica de píxel(x,y).

```
procedure lineBres(xa,ya,xb.yb:integer);
var
  dx,dy,x,y,xEnd,p:integer;
    begin
      dx:=abs(xa-xb);
      dy:=abs(ya-yb);
      p:=2*dy-dx;
      if xa>xb then
        begin
           x:=xb;
           y:=yb;
           xEnd:=xa;
        end  {if xa > xb}
       else
        begin
          x:=xa;
          y:=ya;
          xEnd:=xb;
        end;
       setPixel(x,y,1);
       while x< xEnd do
         begin
           x:=x+1;
           if p< 0 then p:=p+2*dy
           else
             begin
               y:=y+1;
               p:=p+2*(dy-dx);
             end;
           setpixel(x,y,1)
         end }
end;{lineBres}
```

2.4 Algoritmos De Generación De Circunferencias

Como la circunferencia es un componente que se utiliza con frecuencia en imágenes y graficas, la mayor parte de los paquetes de graficas incluyen un procedimiento para generar ya sea circunferencias completas o arcos circulares. De modo mas general, se puede ofrecer un solo procedimiento para desplegar ya sea curvas circulares o elípticas.

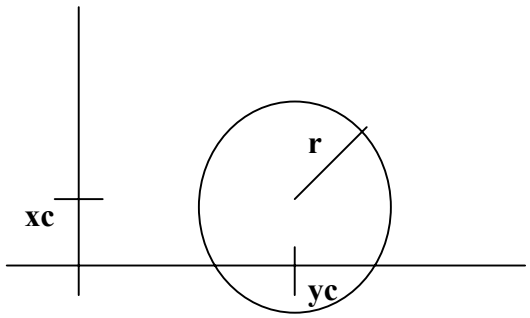


Figura 2.4 circunferencia con coordenadas de centro (xc,yc) y radio r.

2.4.1 Propiedades de las circunferencias

Una circunferencia se define como un conjunto de puntos que se encuentran, en su totalidad, a una distancia determinada **r** de una posición centrada (Xc,Yc). Esta relación de distancia se expresa por medio del teorema de Pitágoras en coordenadas cartesianas como:

$$(X - Xc)^2 + (Y - Yc)^2 = r^2 \qquad \textbf{(2-24)}$$

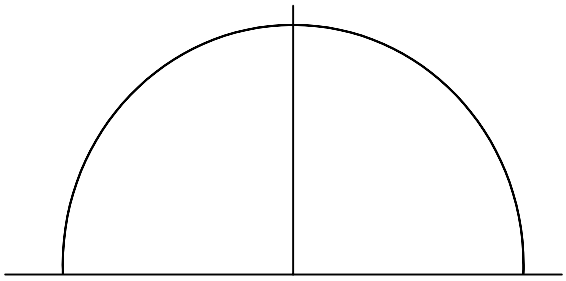


Figura 2.5 mitad positiva de una circunferencia trazada con la ecuación 2.26 y co (x,y)=(0,0).

Podríamos utilizar esta ecuación para calcular la posición de los puntos de una circunferencia pasando a lo largo del eje de las X en pasos unitarios de Xc – R a Xc + R y calcular los valores correspondientes de Y en cada posición como

$$Y = Yc + \sqrt{r^2 - (Xc - X)^2} \qquad \textbf{(2-25)}$$

Otra manera de eliminar el espacio irregular que se presenta en la figura consiste en calcular los puntos a lo largo de la frontera circular utilizando las coordenadas polares **r** y θ. Al expresar la ecuación de la circunferencia en forma polar paramétrica, se obtiene el par de ecuaciones.

$$X = Xc + r * \cos\theta$$
$$Y = Yc + r * \operatorname{sen}\theta \qquad \textbf{(2-26)}$$

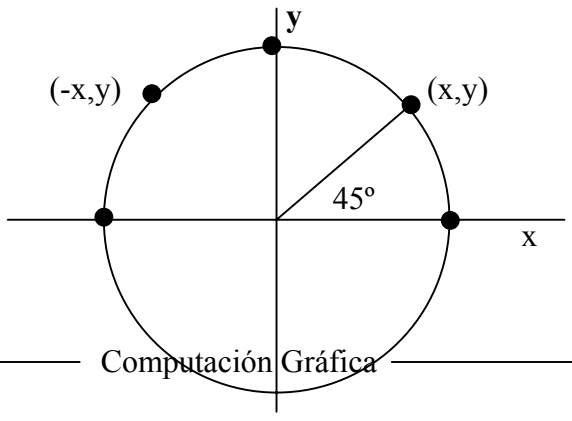


Figura 2.6 simetría de una circunferencia el cálculo de un punto de la circunferencia (x,y) en un octante da como resultado los puntos de la circunferencia que se ilustran para los otros siete octantes

2.4.2 Algoritmo de punto medio para la circunferencia

Al igual que en el algoritmo de línea de rastreo, efectuamos un muestreo en intervalos unitarios y determinamos la posición del píxel más cercano a la trayectoria específica de la circunferencia en cada paso. Para aplicar el método del punto medio, definimos una función de circunferencia como

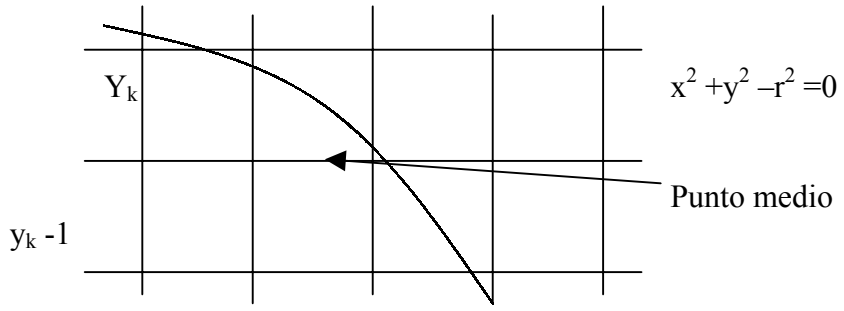
$$\mathbf{F(x,y) = x_2 + y_2 - r_2} \qquad \textbf{(2-27)}$$

Cualquier punto (x, y) en la frontera de la circunferencia con radio r satisface la ecuación f(x, y) = 0.

$$\begin{array}{rll} \mathbf{< 0} & \textbf{si (x,y) está dentro de la frontera de la circunferencia} & \\ \mathbf{F(x,y) = 0} & \textbf{si (x,y) está en la frontera de la circunferencia} & \textbf{(2-28)} \\ \mathbf{> 0} & \textbf{si (x,t) está fuera de la frontera de la circunferencia} & \end{array}$$

La figura muestra el punto medio entre los dos píxels candidatos en la posición de muestreo

$X_k + 1$. Suponiendo que acabamos de trazar el píxel en (x,y) en seguida necesitamos determinar si el píxel en la posición (X_k+1, Y_k) o aquel en la posición $(X_k +1, Y_k-1)$ está más cerca de la circunferencia. Nuestro parámetro de decisión es la función de circunferencia evaluada en el punto medio entre estos dos píxels:



x_k x_k+1 x_k+2

Figura 2.7 punto medio entre pixeles candidatos en la posición de muestreo xk + 1 a lo largo de la trayectoria circular

$$P_k = f(X_k + 1, Y_k - ½)$$
$$= (X_k + 1)^2 + (Y_k - ½)^2 - r^2 \quad \textbf{(2-29)}$$

Los parámetros de decisión sucesivos se obtienen al utilizar cálculos incrementales. Obtenemos una expresión recursiva para el siguiente parámetro de precisión cuando evaluamos la función de circunferencia en la posición de muestreo

$$X_{k+1} + 1 = X_k + 2$$

$$P_{k+1} = P_k + 2(X_k + 1) + (Y^2_{k+1} - Y^2_k) - (Y_{k+1} - Y_k) + 1 \quad \textbf{(2-30)}$$

El parámetro inicial de decisión se obtiene de evaluar la función de la circunferencia en la posición de inicio (Xo, Yo) = (0,r)

$$Po = F(1, r - ½)$$
$$= 1 + (r - 1/2)_2 - r_2$$
$$Po = 5/4 - r \quad \textbf{(2-31)}$$

Algoritmo de circunferencia de punto medio

1. Se captura el radio R y el centro de la circunferencia (x, y) y se obtiene el primer punto de una circunferencia centrada en el origen como:
   (Xo, Yo) = (0, r)
2. Se calcula el valor del parámetro de decisión como:
   Po = 5 / 4 - r
3. En cada x. Posición, al iniciar en k = 0, se realiza la prueba siguiente. Si $P_k < 0$, el siguiente punto a lo largo de la circunferencia centrada en el punto (0, 0) es (X_k, Y_k) y
   $P_{k+1} = P_k + 2X_{K+1} + 1$

2.4.3 Ejercicio

Dado el radio de una circunferencia r = 10, demostramos el algoritmo de la circunferencia de punto medio al determinar las posiciones a lo largo del octante de la circunferencia en el primer cuadrante desde x = O hasta x = y. El valor inicial del parámetro de decisión es

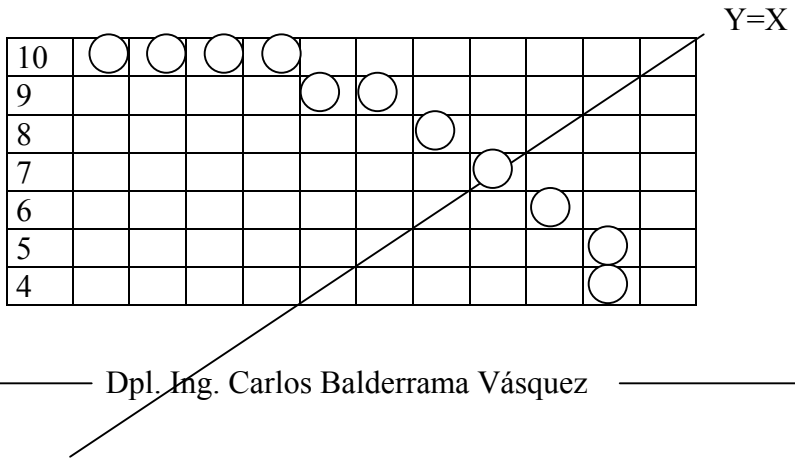
$$Po = 1 - r = -9$$

En el caso de la circunferencia centrada en el origen de las coordenadas, el punto inicial es $(X_0, Y_0) = (0, 10)$ y los términos de incremento iniciales para calcular los parámetros de decisión son

$$2x_0 = 0, \qquad 2y_0 = 20$$

Los valores sucesivos del parámetro de decisión y las posiciones a lo largo de la trayectoria de la circunferencia se calculan mediante el método del punto medio como:

k	P_k	(X_{k+1}, Y_{k+1})	$2X_{k+1}$	$2Y_{k+1}$
0	-9	(1,10)	2	20
1	-6	(2,10)	4	20
2	-1	(3,10)	6	20
3	6	(4,9)	8	18
4	-3	(5,9)	10	18
5	8	(6,8)	12	16
6	5	(7,7)	14	14



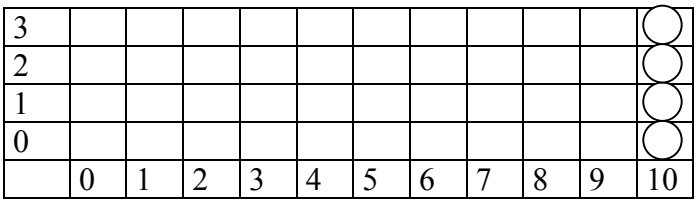


Figura 2.8 Posiciones de los pixeles seleccionados a lo largo de la trayectoria de una circunferencia con radio r=10 y centro en el oprigen

```
procedure plotpuntos;
begin
putpixel(xc+x,yc+y,3);
putpixel(xc-x,yc+y,3);
putpixel(xc+x,yc-y,3);
putpixel(xc-x,yc-y,3);
putpixel(xc+y,yc+x,3);
putpixel(xc-y,yc+x,3);
putpixel(xc+y,yc-x,3);
putpixel(xc-y,yc-x,3);
end;
```

```
procedure
circunferencia(xc,yc,ra:integer);
var p,x,y:integer;
begin
x:=0;y:=ra;
plotpuntos;p:=1-ra;
while x<y do
begin
inc(x);
if p>=0 then dec(y);
if p<0 then p:=p+2*x+1
else p:=p+2*(x-y)+1;
plotpuntos;
end;
end;
```

2.5.1 Algoritmos De Generación De Elipses

Expresado en forma ambigua, una elipse es una circunferencia alargada. Por todas las curvas elípticas se pueden generar al modificar los procedimientos para el trazo de circunferencias con el fin de considerar las diversas dimensiones de una elipse a lo largo de los ejes mayor y menor.

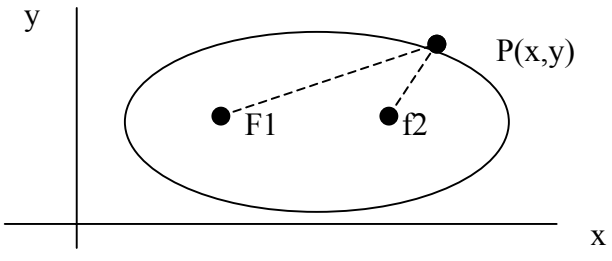


Figura 2.9 elipse generada alrededor de dos posiciones fijas (focos) f_1 y f_2

2.5.2 Propiedades de las elipses

Una elipse se define como el conjunto de puntos en que la suma de las distancias desde dos posiciones fijas *(focos)* sea la misma para todos los puntos. Si las distancias a los dos focos desde cualquier punto $P = (x, y)$ en la elipse se representan como d_1 y d_2 entonces la ecuación general de una elipse puede expresarse como

$$\mathbf{d_1 + d_2 = constante} \qquad \textbf{(2-32)}$$

Al expresar las distancias d_1 y d_2 , en términos de distancias focales

$$\sqrt{(x - x_1)^2 + (y - y_1)^2} + \sqrt{(x - x_2)^2 + (y - y_2)^2} \qquad \textbf{(2-33)}$$

Desarrollando tenemos:

$$\boldsymbol{Ax^2 \pm By^2 \pm CXy \pm Dx + Ey + E = 0} \qquad \textbf{(2-34)}$$

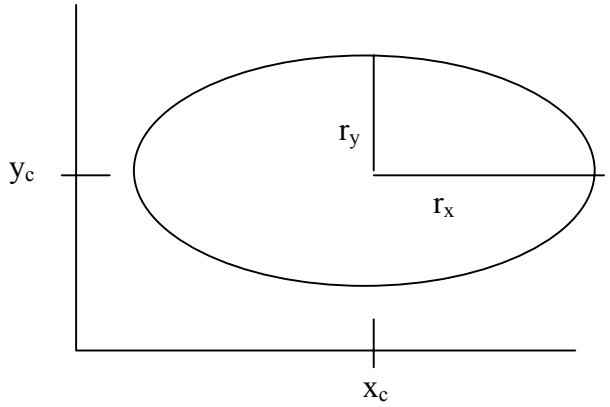


Figura 2.10 Elipse centrada en ($x_c y_c$) con semi eje mayor r_x y semi eje menor r_y

donde los coeficientes *A, B, C, D, E* y *E* se evalúan en términos de las coordenadas focales y las dimensiones de los ejes mayor y menor de la elipse.

La ecuación de la elipse que aparece puede expresarse en términos de las coordenadas del centro de la elipse y los parámetros $r_x r_y$

$$\mathbf{(x - x_c / r_x)^2 + (y - y_c / r_y)^2} \qquad \textbf{(2-35)}$$

Se pueden aplicar consideraciones sobre la simetría para reducir aún más los cálculos.

$$\mathbf{X = X_c + R_x * \cos \theta} \qquad \textbf{(2-36)}$$
$$\mathbf{Y = Y_c + R_y * sen\ \theta}$$

2.5.3 Algoritmo de la elipse de punto medio

El método de punto medio para elipse se aplica a lo largo del primer cuadrante en dos partes. La figura presenta la división del primer cuadrante de acuerdo con la pendiente de una elipse con $\mathbf{r_x} < r_y$. Procesamos este cuadrante tomando pasos en la dirección de x donde la pendiente de la curva tiene una magnitud menor que 1 y tomando pasos unitarios en la dirección de y donde la pendiente tiene una magnitud mayor que 1.

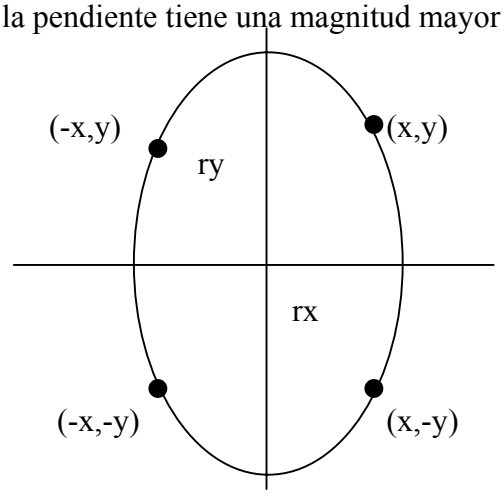


Figura 2.11Simetria de una elipse, el cálculo de un punto de la elipse en un cuadrante da como resultado los puntos de la elipse que se quiere ilustrar para los otros tres cuadrantes.

Las regiones 1 y 2 pueden procesarse de varias maneras. Podemos iniciar en la posición (O, $r_{y,}$) y pasar en el sentido del reloj a lo largo de la trayectoria elíptica en el primer cuadrante, al alternar de pasos unitarios en los pasos unitarios en y cuando la pendiente adquiere un valor menor que -1. De modo alternativo, podemos iniciar en *(rs,* 0) y seleccionar puntos en el sentido contrario al de las manecillas del reloj, alternando de pasos unitarios en y a pasos unitarios en cuando la pendiente adquiere un valor mayor que -1. Con procesadores paralelos podríamos calcular en forma simultánea las posiciones de píxel en las dos regiones.

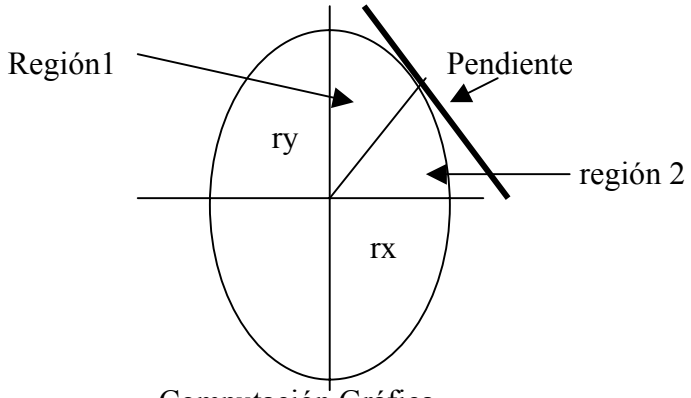


Figura 2.12 regiones de procesamiento de la elipse. En la región 1 la magnitud de la pendiente de la elipse es menor que 1; en la región 2, la magnitud de la pendiante de la elipse es mayor que 1.

Definimos la función de una elipse con base en la ecuación 2-35 con *(x,* y_0)

$$\mathbf{F(x,y) = r^2_y x^2 + r^2_x y^2 - r^2_x r^2_y} \qquad \textbf{(2-37)}$$

Que tiene las propiedades siguientes

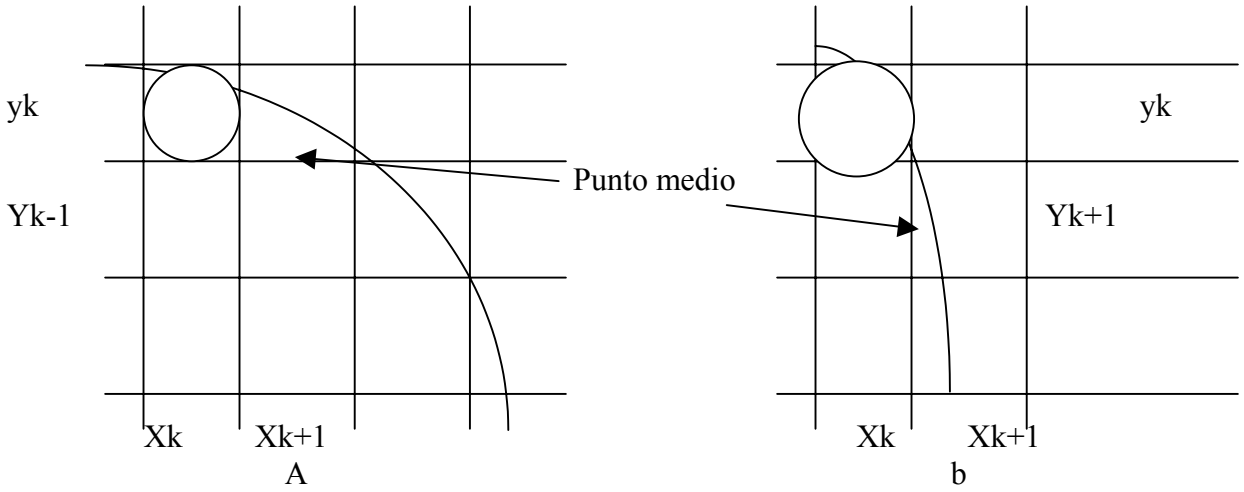
$$F(x,y) \begin{cases} < 0 \text{ si (x,y) está adentro de la frontera de la elipse} \\ = 0 \text{ si (x,y) está en la frontera de la elipse} \\ > 0 \text{ si (x,y) está afuera de la frontera de la elipse} \end{cases} \qquad \textbf{(2-38)}$$

Los parámetros de decisión se incrementan con las cantidades siguientes

Incremento

$$\mathbf{2r^2_y x_{k+1} + r^2_y} \qquad \textbf{si } \mathbf{p1_k < 0}$$

$$\mathbf{2r^2_y x_{k+1} + r^2_y - 2r^2_x y_{k+1}} \qquad \textbf{si } \mathbf{p1_k >= 0}$$



Fugura 2.13 A) Punto medio entre píxels candidatos en la posición de muestreo xk+1 a lo largo de la trayectoria elíptica. B) Punto medio entre píxels candidatos en la posición de muestreo xk-1 a lo largo de la trayectoria elíptica.

Algoritmo de la elipse de punto medio

. Se capturan r_x , r_y y el centro de la elipse *(*x_c , y_c*)* y se obtiene el primer punto de

. Se calcula el valor inicial del parámetro de decisión en la región 1 como

$$P\ o = r_y - r_x * r_y + \tfrac{1}{4}(r\ x)$$

. En cada posición x_k en la región 1, al iniciar en $k = 0$, se realiza la prueb

$$P^1_{k+1} = P^1_k + 2r^2_y * x_{k+1} + r^2_y$$

2.5.4 Ejercicio

Dados los parámetros de entrada de la elipse $r_x = 8$ y $r_y = 6$ realizar el rastreo de la elipse en el primer cuadrante, donde los valores iniciales del parámetro de decisión son:

$2r^2_y x = 0$ con incrementos de $2r^2_y = 72$

$2r^2_x y = 2r^2_x r^2_y$ con incrementos de $-2r^2_x = -128$

en el caso de la región 1 el punto inicial para la elipse centrada en el origen es $(x_0,y_0) = (0,6)$ y el valor del parámetro inicial del parámetro de decisión es

$p1_0 = r^2_y - r^2_x r_y + ¼\ r^2_x = -332$

los valores sucesivos son los siguientes

K	$P1_k$	(x_{k+1},y_{k+1})	$2r^2_y x_{k+1}$	$2r^2_x y_{k+1}$
0	-332	(1,6)	72	768
1	-224	(2,6)	144	768
2	-44	(3,6)	216	768
3	202	(4,5)	288	640
4	-108	(5,5)	360	640
5	288	(6,4)	432	512
6	244	(7,3)	504	384

Ahora se sale de la región 1 puesto que $2r^2_y x > 2r^2_x y$

En el caso de la región 2 el punto inicial es (x_0,y_0) =(7,3) y el parámetro de decisión inicial es

$$P2_0 = f(7 + 1/2 * 2) = -151$$

Las posiciones restantes a lo largo de la trayectoria de la elipse son:

K	$P1_k$	(x_{k+1},y_{k+1})	$2r^2_y x_{k+1}$	$2r^2_x y_{k+1}$
0	-151	(8,2)	576	256
1	233	(8,1)	576	128
2	745	(8,0)	-	-

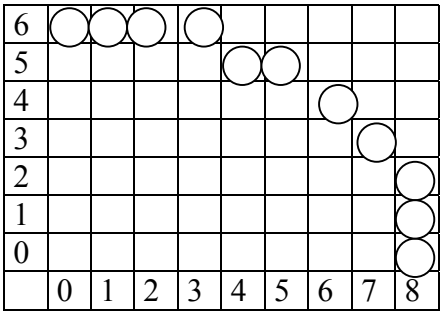


Figura 2.14 posiciones a lo largo de la trayectoria elíptica centrada en el origen

2.6 Programa en pascal de primitivas

```
Program primitivas;
uses graph,crt;
var xe,ye,rx,ry:integer;
    grd,grm,erco:integer;

Procedure lineaDDA(xa,ya,xb,yb:integer);
var dx,dy,paso,k:integer;
```

```
    xincre,yincre,x,y:real;
begin
        dx:=abs(xb-xa);
        dy:=abs(yb-ya);
        if dx>dy then paso:=dx
                else paso:=dy;
        xincre:=dx/paso;
        yincre:=dy/paso;
        x:=xa;y:=ya;
        for k:=1 to paso do
                begin
                        x:=x+xincre;
                        y:=y+yincre;
                        putpixel(round(x),round(y),7);
                end;
end;

procedure lineaBresenham(xa,ya,xb,yb:integer);
var dx,dy,x,y,xfin,p:integer;
begin
        dx:=abs(xb-xa);
        dy:=abs(yb-ya);
        p:=2*dy-dx;
        if dx>dy then
                begin
                        x:=xb;y:=yb;
                        xfin:=xa;
                end
        else
                begin
                        x:=xa;y:=ya;
                        xfin:=xb;
                end;
        while x<xfin do
                begin
                        inc(x);
                        if p<0 then p:=p+2*dy
                        else
                                begin
                                        inc(y);
                                        p:=p+2*(dy-dx);
                                end;
                        putpixel(x,y,2);
                end;
end;

procedure circunferencia(xc,yc,ra:integer);
```

```
var p,x,y:integer;
        procedure plotpuntos;
        begin
                putpixel(xc+x,yc+y,3);
                putpixel(xc-x,yc+y,3);
                putpixel(xc+x,yc-y,3);
                putpixel(xc-x,yc-y,3);
                putpixel(xc+y,yc+x,3);
                putpixel(xc-y,yc+x,3);
                putpixel(xc+y,yc-x,3);
                putpixel(xc-y,yc-x,3);
        end;
begin
        x:=0;y:=ra;
        plotpuntos;p:=1-ra;
        while x<y do
                begin
                        inc(x);
                        if p>=0 then dec(y);
                        if p<0 then p:=p+2*x+1
                        else p:=p+2*(x-y)+1;
                        plotpuntos;
                end;
end;

procedure elipse(xc,yc,rx,ry:integer);
var p,px,py,x,y:integer;
    ry2,rx2,rx22,ry22:integer;
        procedure plotear;
        begin
                putpixel(xc+x,yc+y,4);
                putpixel(xc-x,yc+y,4);
                putpixel(xc+x,yc-y,4);
                putpixel(xc-x,yc-y,4);
        end;
begin
        ry2:=ry*ry;
        rx2:=rx*rx;
        ry22:=2*ry2;
        rx22:=2*rx2;
        x:=0;y:=ry;
        plotear;
        p:=round(ry2-rx2*ry+(0.25*rx2));
        px:=0;py:=rx22*y;
        while px<py do
                begin
                        inc(x);px:=px+ry22;
```

```
                    if p>=0 then
                        begin
                            dec(y);py:=py-rx22;
                        end;
                    if p<0 then p:=p+ry2+px
                    else p:=p+ry2-py;
                    plotear;
                end;
        p:=round(ry2*sqr(x+0.5)+rx2*sqr(y-1)-rx2*ry2);
        while y>0 do
                begin
                    dec(y);
                    py:=py+rx22;
                    if p<=0 then
                        begin
                            inc(x);
                            px:=px+ry22;
                        end;
                    if p>0 then p:=p+rx2-py
                    else p:=p+rx2-py+px;
                    plotear;
                end;
end;

procedure presentacion;
begin
        clrscr;
        gotoxy(20,5);writeln('PROGRAMACION GRAFICA **** SIS - 1201 ****  ');
        gotoxy(1,8);writeln('          TRAZO DE LAS PRIMITIVAS          ');
        gotoxy(10,12);writeln('-Trazo de Linea algoritmo DDA             ');
        gotoxy(10,13);writeln('-Trazo de Linea algoritmo Bresenham       ');
        gotoxy(10,14);writeln('-Trazo de Circunf. algoritmo del Pto. Medio');
        gotoxy(10,15);writeln('-Trazo de Elipse algoritmo Pto. Medio     ');
        readln;
end;

procedure menu;
var op1:char;sw:byte;
begin
        cleardevice;sw:=1;
        repeat
                setcolor(1);
                rectangle(440,0,590,120);
                settextstyle(2,0,4);
                setcolor(13);
                outtextxy(449,4,'***** MENU *******');
                outtextxy(470,20,'1.- Linea DDA');
```

```
                outtextxy(470,35,'2.- Linea Bresenham');
                outtextxy(470,50,'3.- Circunferencia');
                outtextxy(470,65,'4.- Elipse');
                outtextxy(470,80,'5.- Salir');
                outtextxy(450,100,'ELIJA SU OPCION');
                gotoxy(72,7);op1:=readkey;cleardevice;
                case op1 of
                        '1':lineadda(30,50,500,300);
                        '2':lineabresenham(100,100,300,500);
                        '3':circunferencia(200,200,150);
                        '4':elipse(320,180,3000,20);
                        '5':sw:=0;
                end;
        until sw=0;
        closegraph;
end;

Begin
        presentacion;
        initgraph(grd,grm,'c:\tp\bgi');
        erco:=graphresult;
        if erco=grok then menu
        else
                begin
                        writeln('Error de grafico', GraphErrorMSG(erco));
                        readln;
                end;
end.
```