Midpoint Circle Generation

Trisha Shah

SE CMPN A 20102A0004

**Algorithm:-**

Step 1:

Accept the radius r and center (xc, yc) of a circle. The first point of the

circumference of a circle with center as origine is (x0 , y0) = (0, r)

Step 2:

Calculate the initial decision parameter as P0= 5/4 – r ≈ 1 – r ( radius is

integer value)

Step 3:

At each xk position starting at k = 0 perform the following test. If Pk < 0 Then xk

+ 1= xk + 1 yk + 1= yk Pk + 1= Pk + 2xk + 3 Otherwise xk + 1= xk + 1 yk + 1= yk –

1 Pk + 1= Pk + 2 (xk – yk) + 5

Step 4:

Determine the symmetry points in other seven octants.

Step 5:

Translate each calculated pixel position by T (xk, yk) and display the pixel. x =

xk + 1 + xc y = yk + 1 + yc putpixel (x, y, WHITE)

Step 6:

Repeat step 3 through 5 until x  y.

Step 7:

Stop

**Program :**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void main()

{

int gd,gm;

int i,r,x,y,xc,yc;

float p;

detectgraph(&gd,&gm);

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

printf("Enter center of circle=");

scanf("%d %d",&xc,&yc);

printf("enter radius of circle=");

scanf("%d",&r);

x=0;

y=r;

p=1.25-r;

do {

if(p<0.0) {

x += 1;

p += (2\*x) + 3;

}

else

{ x += 1;

y -= 1;

p += 2\*(x-y) + 5;

}

putpixel(xc+x,yc+y,15);

putpixel(xc+x,yc-y,15);

putpixel(xc-x,yc+y,15);

putpixel(xc-x,yc-y,15);

putpixel(xc+y,yc+x,15);

putpixel(xc+y,yc-x,15);

putpixel(xc-y,yc+x,15);

putpixel(xc-y,yc-x,15);

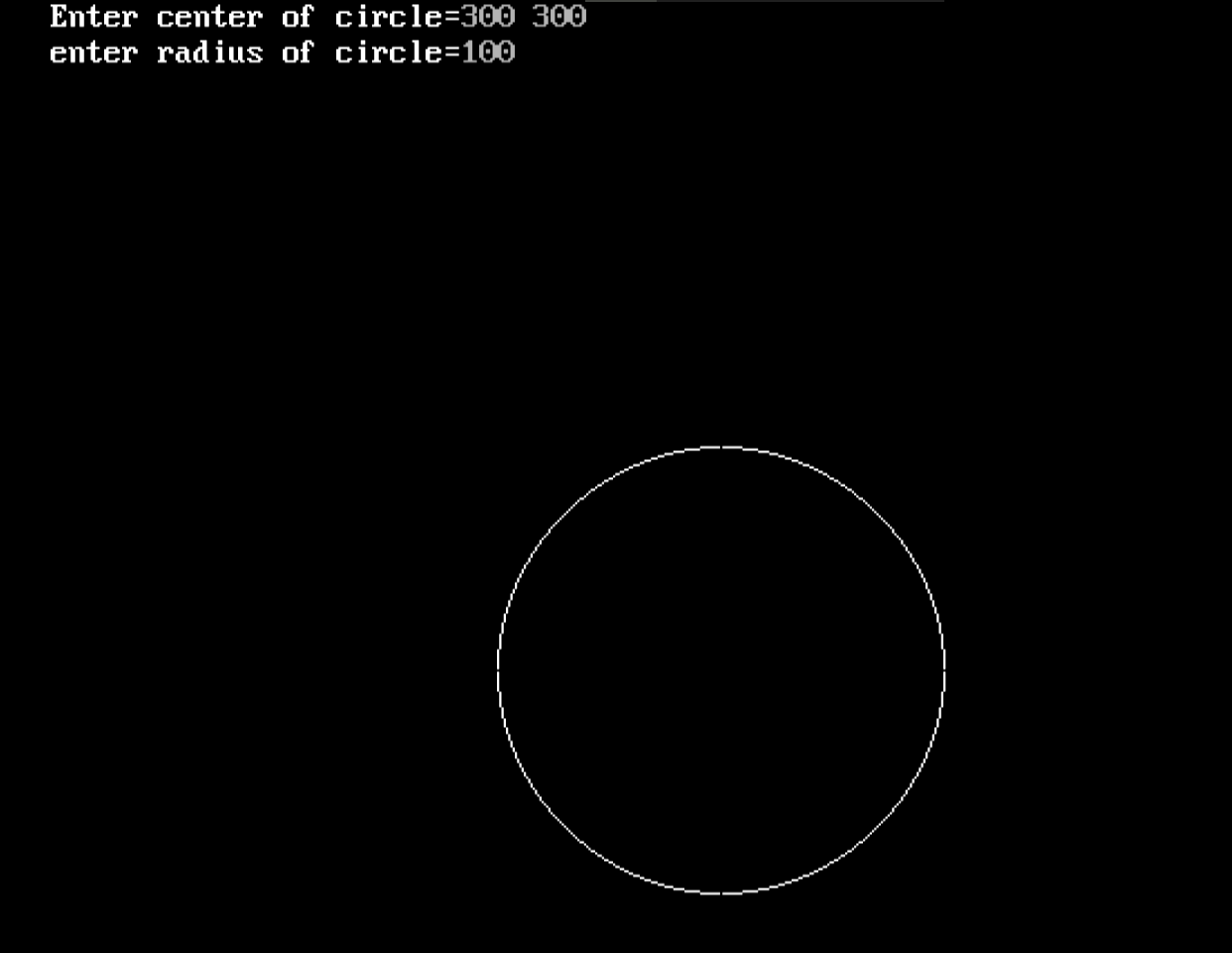
delay(10);

}

while(x<y);

getch();

}



**Advantages:**

The midpoint method is used for deriving efficient scan-conversion algorithms to draw geometric curves on raster displays.

The method is general and is used to transform the nonparametric equation f(x,y) = 0, which describes the curve, into an algorithms that draws the curve.

**Disadvantages:**

-time consumption is high

-the distance between the pixels is not equal so we won’t get smooth circle.

**Ellipse Generation**

**Theory:**

Midpoint ellipse algorithm plots(finds) points of an ellipse on the first quadrant by dividing the quadrant into two regions.

Each point(x, y) is then projected into other three quadrants (-x, y), (x, -y), (-x, -y) i.e. it uses 4-way symmetry.

**Mid-Point Ellipse Algorithm :**

1. Take input radius along x axis and y axis and obtain center of ellipse.
2. Initially, we assume ellipse to be centered at origin and the first point as : (x, y0)= (0, ry).
3. Obtain the initial decision parameter for region 1 as: p10=ry2+1/4rx2-rx 2ry
4. For every xk position in region 1 :

If p1k<0 then the next point along the is (xk+1 , yk) and p1k+1=p1k+2ry2xk+1+ry2

Else, the next point is (xk+1, yk-1 )

And p1k+1=p1k+2ry2xk+1 – 2rx2yk+1+ry2

1. Obtain the initial value in region 2 using the last point (x0, y0) of region 1 as: p20=ry2(x0+1/2)2+rx2 (y0-1)2-rx2ry2
2. At each yk in region 2 starting at k =0 perform the following task.

If p2k>0 the next point is (xk, yk-1) and p2k+1=p2k-2rx2yk+1+rx2

Else, the next point is (xk+1, yk -1) and p2k+1=p2k+2ry2xk+1 -2rx2yk+1+rx2

1. Now obtain the symmetric points in the three quadrants and plot the coordinate value as: x=x+xc, y=y+yc
2. Repeat the steps for region 1 until 2ry2x>=2rx2y

**Implementation**

#include<stdio.h>

#include<graphics.h>

void Drawellipse(float rx,float ry,float x1,float y1)

{

float x=0;

float y=ry;

float slope=(-ry\*ry\*rx)/(rx\*rx\*y);

float p1=ry\*ry-rx\*rx\*ry+(1/4)\*rx\*rx;

float p2=ry\*ry\*(x+1/2)+rx\*rx\*(y-1)\*(y-1)-rx\*rx\*ry\*ry;

while(2\*ry\*ry\*x<2\*rx\*rx\*y)

{

putpixel(x1+x,y1+y,MAGENTA);

delay(30);

putpixel(x1-x,y1+y,MAGENTA);

delay(30);

putpixel(x1-x,y1-y,MAGENTA);

delay(30);

putpixel(x1+x,y1-y,MAGENTA);

delay(30);

x=x+1;

if(p1<0)

{

p1=p1+2\*ry\*ry\*x+ry\*ry;

}

else

{

y=y-1;

p1=p1+2\*ry\*ry\*x-2\*rx\*rx\*y+ry\*ry;

}

}

while(y>0)

{

putpixel(x1+x,y1+y,MAGENTA);

delay(30);

putpixel(x1-x,y1+y,MAGENTA);

delay(30);

putpixel(x1-x,y1-y,MAGENTA);

delay(30);

putpixel(x1+x,y1-y,MAGENTA);

delay(30);

y=y-1;

if(p2>0)

{

p2=p2-2\*rx\*rx\*y+rx\*rx;

}

else

{

x=x+1;

p2=p2+2\*ry\*ry\*x-2\*rx\*rx\*y+rx\*rx;

}

}

}

void main()

{

float rx,ry,x1,y1,x,y,p1,p2,slope;

int i,gd,gm;

initgraph(&gd,&gm,” C:\\TURBOC3\\BGI”)

printf("ENTER RX");

scanf("%f",&rx);

printf("ENTER RY");

scanf("%f",&ry);

printf("ENTER x-coordinate of centre:");

scanf("%f",&x1);

printf("ENTER y-coordinate of centre");

scanf("%f",&y1);

Drawellipse(rx,ry,x1,y1);

getch();

closegraph();

}

