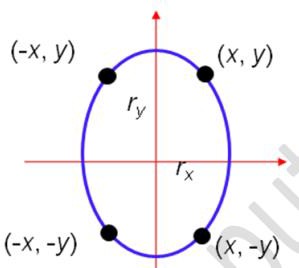
# Lab 4

**OBJECTIVE:**

TO IMPLEMENT MIDPOINT CIRCLE GENERATION ALGORITHM OR BRESENHAM’S CIRCLE ALGORITHM FOR DRAWING AN ELLIPSE OF GIVEN CENTER (x,y) AND MAJOR, MINOR AXIS rx, ry.

## THEORY:

In Ellipse, unlike circle, quadrants are Not symmetric between the two octants of a quadrant. Thus, we must calculate pixel positions along the elliptical arc through one quadrant and then we obtain positions in the remaining 3 quadrants by symmetry.



The next pixel is chosen based on the decision parameter. The required conditions are given in following algorithm.

# ALGORITHM:

1. Input radius rx and ry and ellipse center (xc,yc). Obtain the first point on an ellipse centered on the origion as then set the coordinates for the first point on the circumference of a circle centered on the origin as:

(x0,y0) = (0,ry)

1. Calculate the initial value of the decision parameter as:

1 2 2 1 2

P0 = ry

- rx ry+

4

𝑟x

1. At each xi position, starting at i=0, if P 1<0, the next point along the ellipse centered on (0,0) is

i

(xi+1,yi) and

1 1 2 2

P = P + 2 r x + r

i+1 i y i+1 y

Otherwise, the next point is (xi+1,yi-1) and

1 1 2 2 2

P = P + 2 r x - 2 r y + r

i+1 i y i+1 x i+1 y

And continue until 2 r 2x >= 2 r 2y

y x

1. (x0,y0) is the last position calculated in region 1. Calculate the initial parameter in region 2 as P2 = r 2(x +1/2)2 + r 2(y -1)2 - r 2r 2

0 y 0 x 0 x y

1. At each yi position, starting at i=0, if p2 >0, the next point along the ellipse centered on (0,0) is (xi,yi-1) and

i

2 2 2 2

P = P - 2 r y + r

i+1 i x i+1 x

Otherwise, the next point is (xi+1,yi-1) and

2 2 2 2 2

P = P + 2 r x - 2 r y + r

i+1 i y i+1 x i+1 x

Use the same incremental calculation as in region 1. Continue until y=0.

1. For both regions determine symmetry points in the other three quadrants.
2. Move each calculated pixel positon (x,y) onto the elliptical path centered on (xc,yc) and plot the coordinate values

X = x+xc y=y+yc

**SOURCE CODE:**

#include <stdio.h>

#include <graphics.h>

#include <stdlib.h>

typedef struct

{

int x;

int y;

} Coordinate;

void drawEllipse(int rX, int rY, int h, int k)

{

int x = 0, y = rY, a = 0, b = 0, i, j;

Coordinate ellipseCoords[8][64];

// For Region 1 and 4-way symmetry through Region 1!

int p = rY \* rY - rX \* rX \* rY + (rX \* rX) / 4;

while (2 \* rY \* rY \* x < 2 \* rX \* rX \* y)

{

ellipseCoords[0][a] = {.x = x + h, .y = y + k}; // R1

ellipseCoords[3][a] = {.x = x + h, .y = -y + k}; // R4

ellipseCoords[4][a] = {.x = -x + h, .y = -y + k}; // R5

ellipseCoords[7][a] = {.x = -x + h, .y = y + k}; // R8

if (p < 0)

{

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

}

else

{

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

}

++a;

};

// For Region 2 and 4-way symmetry through Region 2!

p = (rY \* rY) \* ((float)x + 1 / 2) \* ((float)x + 1 / 2) + (rX \* rX) \* (y - 1) \* (y - 1) - (rX \* rX) \* (rY \* rY);

while (y > 0)

{

ellipseCoords[1][b] = {.x = x + h, .y = y + k}; // R2

ellipseCoords[2][b] = {.x = x + h, .y = -y + k}; // R3

ellipseCoords[5][b] = {.x = -x + h, .y = -y + k}; // R6

ellipseCoords[6][b] = {.x = -x + h, .y = y + k}; // R7

if (p > 0)

{

p += -2 \* rX \* rX \* (--y) + rX \* rX;

}

else

{

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

}

++b;

};

for (i = 0; i < 8; i++)

{

printf("\n\nRegion %d:\n", i + 1);

j = 0;

if (i == 0 || i == 3 || i == 4 || i == 7)

{

while (j < a)

{

x = ellipseCoords[i][j].x, y = ellipseCoords[i][j].y;

putpixel(x, y, WHITE);

printf("(%d, %d)\n", x, y);

j++;

}

}

else

{

while (j < b)

{

x = ellipseCoords[i][j].x, y = ellipseCoords[i][j].y;

putpixel(x, y, WHITE);

printf("(%d, %d)\n", x, y);

j++;

}

}

}

}

int main()

{

int graphicsDriver = DETECT, graphicsMode;

int rX, rY, h, k;

printf("Enter the radius rX and rY of ellipse: ");

scanf("%d %d", &rX, &rY);

printf("Enter the center of ellipse: ");

scanf("%d %d", &h, &k);

initgraph(&graphicsDriver, &graphicsMode, (char \*)"");

drawEllipse(rX, rY, h, k);

printf("\n");

getch();

closegraph();

return 0x1337;

}

**OUTPUT:**



