EXPERIMENT - 4

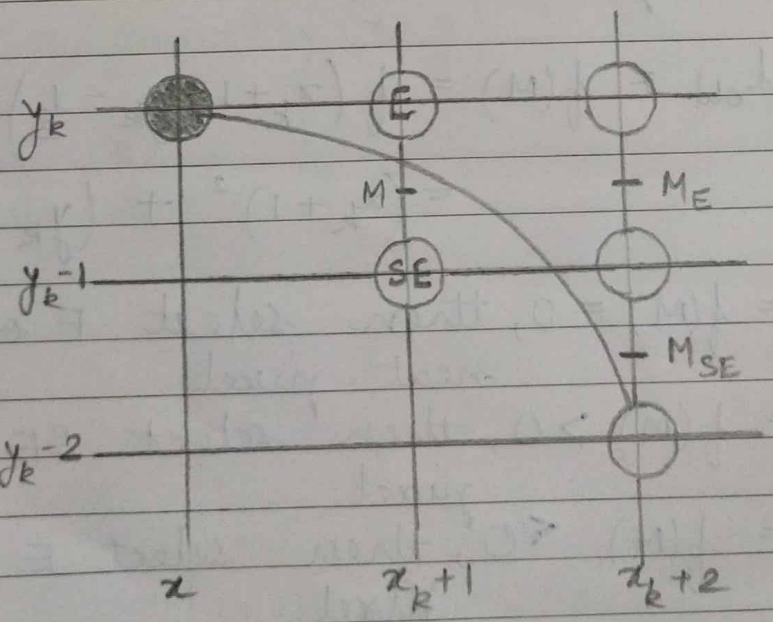
AIM: To implement midpoint circle drawing algorithm.

THEORY:

Principle of midpoint circle drawing approach is identical to the midpoint line - drawing approach.

Moving down from current pixel (x_k, y_k) possible point to be plotted is either East pixel (E) or South - East pixel (SE)

Co-ordinates of the midpoint M are $\left(x_k + 1, y_k - \frac{1}{2}\right)$



FOR EDUCATIONAL USE

We can write the circle equation in explicit form $f(x,y) = x^2 + y^2 - r^2$. We will put the value of midpoint co-ordinates in this function and evaluate its sign. If the function returns 0, mean midpoint is on the circle and we can select E or SE as the next pixel. If the function returns positive value, implies midpoint is inside the circle and the arc is closer to E pixel so select E as next pixel. And if function returns negative value, implies midpoint is outside circle and this arc is closer to SE pixel so select SE as next pixel.

Like line drawing algorithm, we will compute decision parameter d to decide the next pixel.

$$d = d_{old} = f(M) = f\left(x_k + 1, y_k - \frac{1}{2}\right)$$

$$= \left(x_k + 1\right)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2$$

If $d = f(M) = 0$, then select E or SE as a next pixel

If $d = f(M) > 0$, then select SE as a next pixel

If $d = f(M) < 0$, then select E as a next pixel.

Sundaram
FOR EDUCATIONAL USE

Advantages –

1. It is powerful and efficient.
2. No special programming skill is needed.
3. This algorithm is used to generate complex graphics on the raster system.
4. It involves integer calculations only.

Disadvantages –

1. It is time-consuming.
2. Circle generated using this algorithm is not so smooth due to uneven distance between pixels.

Algorithm

```
x ← 0
y ← r
d ← 1 - r
EightWaySymmetry (x, y)
while x < y do
    if d < 0 then
        d ← d + 2x + 3
    else
        d ← d + 2x - 2y + 5
        y ← y - 1
    end
    x ← x + 1
    EightWaySymmetry (x, y)
end
```

Sundaram
FOR EDUCATIONAL USE

EightWaySymmetry (x, y)

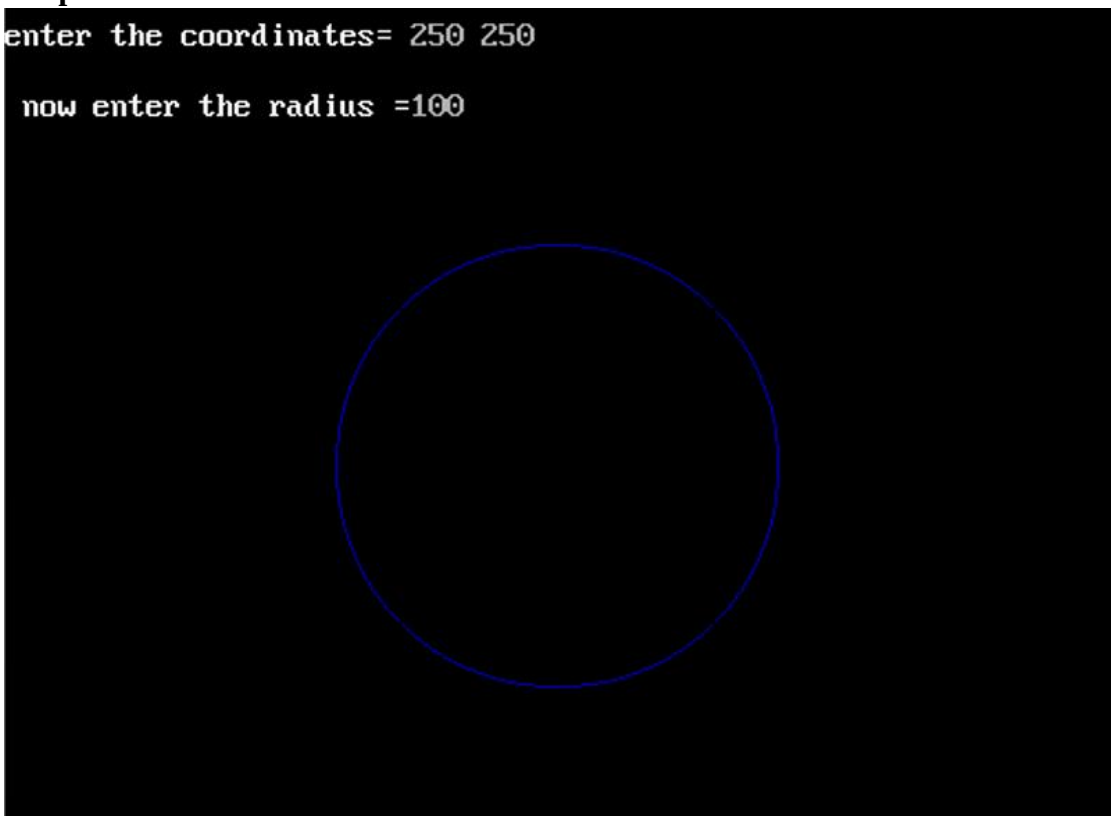
putPixel (x, y)
putPixel (x, -y)
putPixel (-x, y)
putPixel (-x, -y)
putPixel (y, x)
putPixel (y, -x)
putPixel (-y, x)
putPixel (-y, -x)

Experiment 4

Program:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int x,y,x_mid,y_mid,r,d;
int g_mode,g_driver=DETECT;
clrscr();
initgraph(&g_driver,&g_mode,"C:\\TURBOC3\\BGI");
printf("********** MID POINT Circle drawing algorithm ********\n\n");
printf("\nenter the coordinates= ");
scanf("%d %d",&x_mid,&y_mid);
printf("\n now enter the radius =");
scanf("%d",&r);
x=0;
y=r;
d=1-r;
do
{
putpixel(x_mid+x,y_mid+y,1);
putpixel(x_mid+y,y_mid+x,1);
putpixel(x_mid-y,y_mid+x,1);
putpixel(x_mid-x,y_mid+y,1);
putpixel(x_mid-x,y_mid-y,1);
putpixel(x_mid-y,y_mid-x,1);
putpixel(x_mid+y,y_mid-x,1);
putpixel(x_mid+x,y_mid-y,1);
if(d<0)  {
d+=(2*x)+1;
}
else{
y=y-1;
d+=(2*x)-(2*y)+1;
}
x=x+1;
}while(y>x);
getch();
}
```

Output:



Conclusion:

Implemented the Midpoint Circle Drawing Algorithm to create a graphical output.