Experiment No. 3
Topic: Implement Midpoint Circle Algorithm
Name: Vinith Shetty
Roll Number: 55
Date of Performance:
Date of Submission:

Experiment No. 3

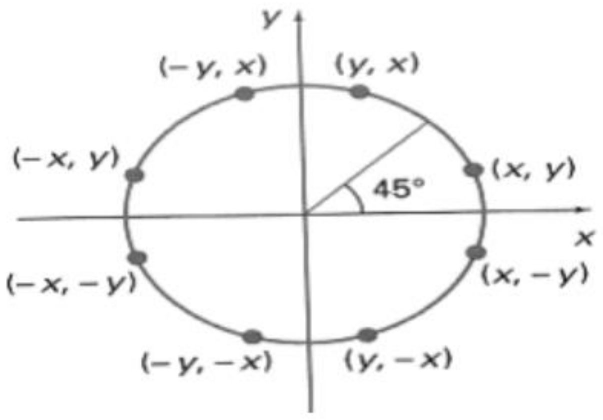
Aim: To implement midpoint circle algorithm.

Objective:

Draw a circle using mid-point circle drawing algorithm by determining the points needed for rasterizing a circle. The mid-point algorithm to calculate all the perimeter points of the circle in the first octant and then print them along with their mirror points in the other octants.

Theory:

The shape of the circle is similar in each quadrant. We can generate the points in one section and the points in other sections can be obtained by considering the symmetry about x-axis and y-axis.



The equation of circle with center at origin is x 2 + y 2 = r 2

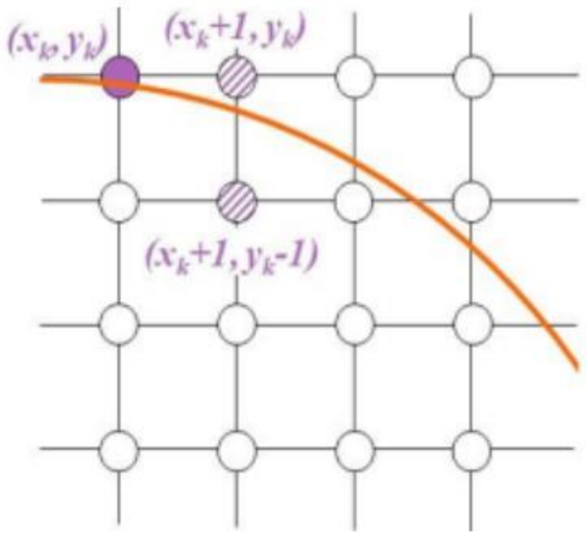
Let the circle function is f circle (x, y) -

 is &lt; 0, if (x, y) is inside circle boundary,

 is = 0, if (x, y) is on circle boundary,

 is &gt; 0, if (x, y) is outside circle boundary.

Consider the pixel at (xk, yk) is plotted,



Now the next pixel along the circumference of the circle will be either (xk + 1, yk) or (xk + 1,

yk – 1) whichever is closer the circle boundary.

Let the decision parameter pk is equal to the circle function evaluate at the mid-point between two pixels.

If pk &lt; 0, the midpoint is inside the circle and the pixel at yk is closer to the circle boundary.

Otherwise, the midpoint is outside or on the circle boundary and the pixel at yk – 1 is closer

to the circle boundary.

Algorithm –

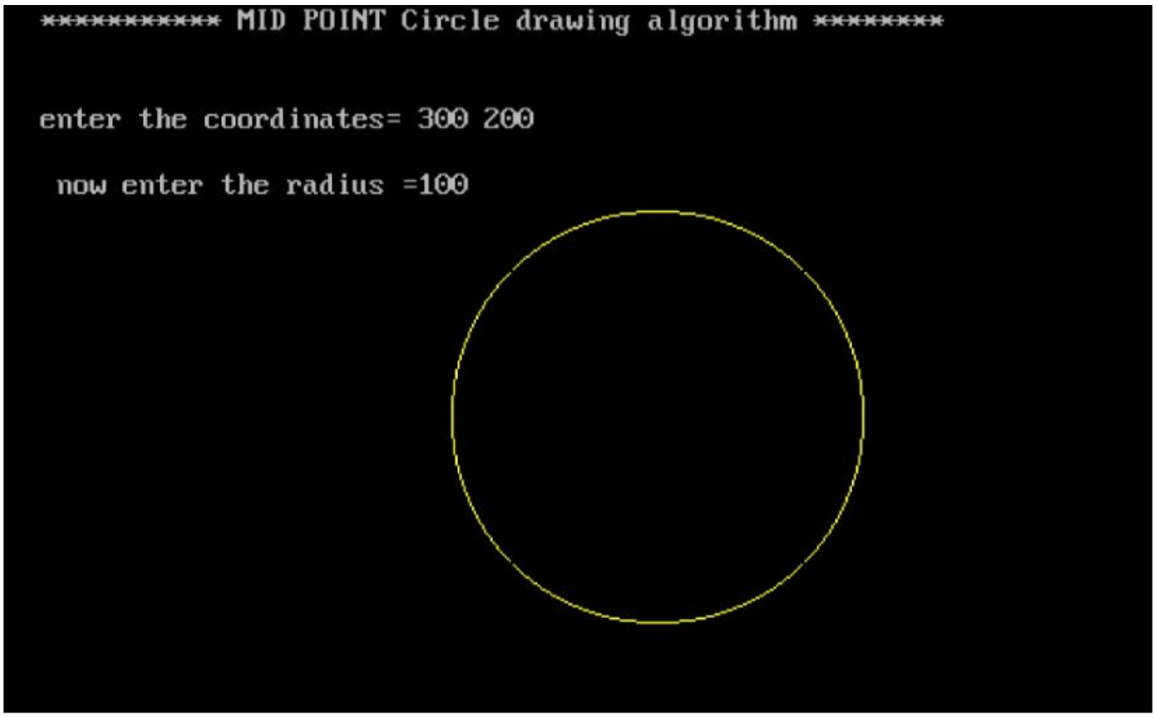
```
Step1: Put x =0, y =r in equation 2
            We have p=1-r
Step2: Repeat steps while x ≤ y
            Plot (x, y)
            If (p<0)
Then set p = p + 2x + 3
Else
            p = p + 2(x-y)+5
            y =y - 1 (end if)
            x =x+1 (end loop)
Step3: End
```

Program –

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
int x,y,x_mid,y_mid,radius,dp;
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",&radius);
x=0;
y=radius;
dp=1-radius;
do
{
putpixel(x_mid+x,y_mid+y,YELLOW);
putpixel(x_mid+y,y_mid+x,YELLOW);
putpixel(x_mid-y,y_mid+x,YELLOW);
putpixel(x_mid-x,y_mid+y,YELLOW);
putpixel(x_mid-x,y_mid-y,YELLOW);
putpixel(x_mid-y,y_mid-x,YELLOW);
putpixel(x_mid+y,y_mid-x,YELLOW);
putpixel(x_mid+x,y_mid-y,YELLOW);
if(dp<0) {
```

```
dp+=(2*x)+1;
}
else{
y=y-1;
dp+=(2*x)-(2*y)+1;
}
x=x+1;
}while(y>x);
getch();
}
```

output –



```
************ MID POINT Circle drawing algorithm *********

enter the coordinates= 300 200

 now enter the radius =100
```

Conclusion: Comment on

1. Fast or slow

2. Draw one arc only and repeat the process in 8 quadrants

3. Difference with line drawing method

Fast or slow: The Midpoint Circle Algorithm is fast and particularly efficient when compared to other circle-drawing algorithms due to its simplicity and use of integer calculations. The Midpoint Circle Algorithm provides a relatively quick and efficient way to draw circles, making it suitable for various applications in computer graphics, games, simulations, and other fields where circle rendering is required.

Draw one arc only and repeat the process in 8 quadrants: The Midpoint Circle Algorithm typically draws a circle in one octant (one-eighth of the circle) and reflects it across the x and y axes to complete the circle. However, to draw only an arc in one quadrant and replicate it in all eight quadrants, the process will be slightly modified to draw an arc in the first quadrant.

Difference with line drawing method: The Midpoint Circle Algorithm focuses on plotting points on the circumference of a circle, while line drawing algorithms concentrate on determining pixels to form straight lines between two given points. The Midpoint Circle Algorithm for circles and line drawing algorithms for straight lines. Their underlying mathematical concepts and computational approaches are distinct due to their respective geometric shapes and applications.