Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Name:	PRIYANKA DEEPAK DHURI
Roll No:	
Class/Sem:	DSE/III
Experiment No.:	3
Title:	Midpoint Circle Drawing Algorithm
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Experiment No. 3

Aim :Write a program to implement Midpoint Circle Drawing Algorithm in C.

Objective :To implement midpoint circle drawing algorithm for drawing a circlewith radius (R) and centre (Xc, Yc)

Midpoint Circle Algorithm

Circles have the property of being highly symmetrical, which is handy when it comes to drawing them on a display screen.

· We know that there are 360 degrees in a circle. First we see that a circle is symmetrical about the x axis, so only the first 180 degrees need to be calculated.

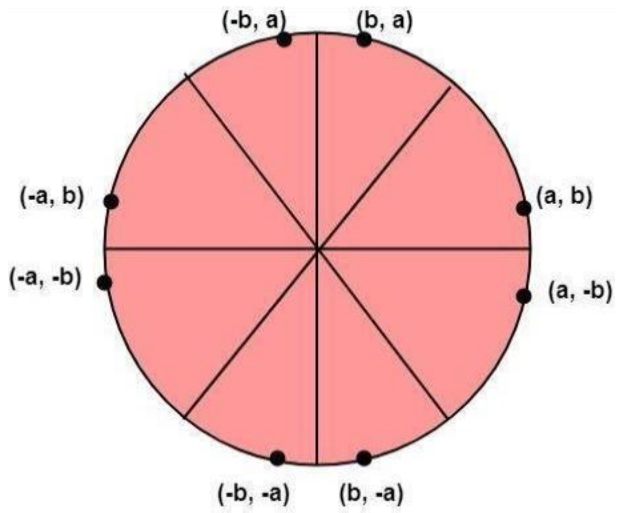
· Next we see that it's also symmetrical about the y axis, so now we only need to calculate the first 90 degrees.

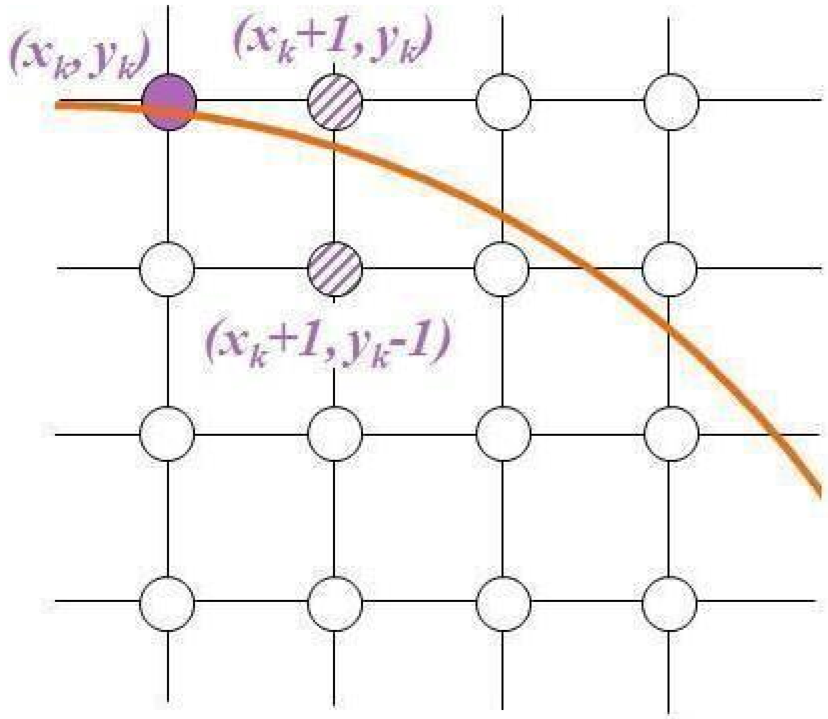
· Finally we see that the circle is also symmetrical about the 45 degree diagonal axis, so we only need to calculate the first 45 degrees. · We only need to calculate the values on the border of the circle in the first octant. The other values may be determined by symmetry

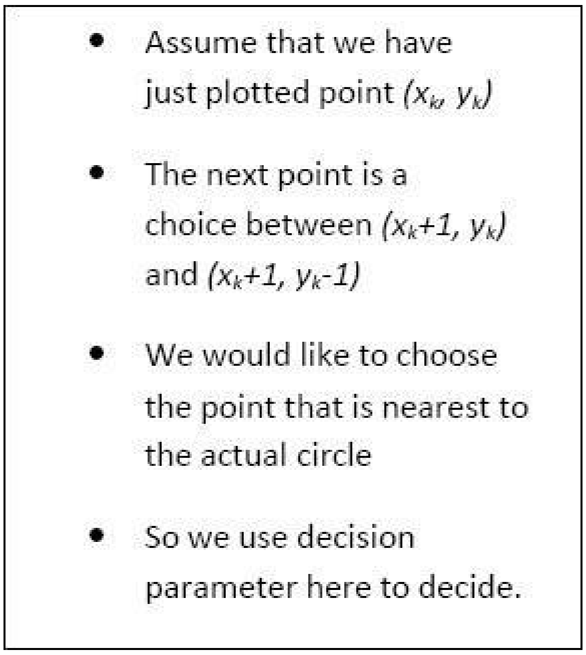
Midpoint circle algorithm calculates the locations of the pixels in thefirst 45 degrees. It assumes that the circle is centered on the origin. So for every pixel (x, y) it calculates, we draw a pixel in each of the eight octants of the circle. This is done till when the value of the y coordinate equals the x coordinate. The pixel positions for determining symmetry are given in the below algorithm.

It is based on the following function for testing the spatial relationship between the arbitrary point (x, y) and a circle of radius r centered at the origin:

$$f(x, y) = x^2 + y^2 - r^2 \qquad \begin{bmatrix} < 0 \text{ for } (x, y) \text{inside the circle} \\ = 0 \text{ for } (x, y) \text{on the circle} \\ > 0 \text{ for } (x, y) \text{outside the circle} \end{bmatrix} \text{.....equation 1}$$


(-b, a)
(b, a)
(-a, b)
(a, b)
(-a, -b)
(a, -b)
(-b, -a)
(b, -a)


(x_k, y_k)
(x_k+1, y_k)
(x_k+1, y_k-1)



- Assume that we have just plotted point *(x_k, y_k)*
- The next point is a choice between *(x_k+1, y_k)* and *(x_k+1, y_k-1)*
- We would like to choose the point that is nearest to the actual circle
- So we use decision parameter here to decide.

Algorithm :

Plot the initial (x_k , y_k) i.e. $x_k = 0$ and $y_k = r$
$x_{k+1} = x_k + 1$ $y_{k+1} = y_k$
If $P_k >= 0$, then choose $y_{k+1} = y_k$
Repeat 3 and 4 until x becomes greater than or equal to y
To plot the entire circle, use the 8-way symmetry

$$P_{k+1} = P_k + 2x_k + 3$$

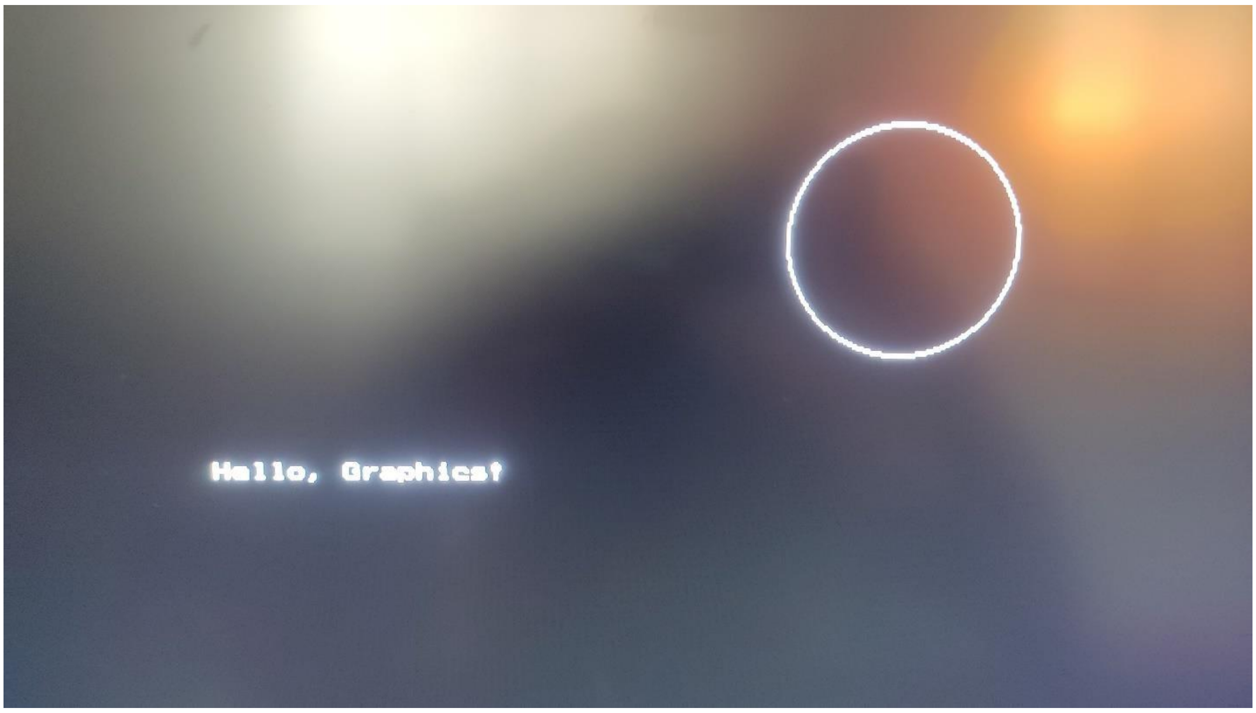
$$P_{k+1} = P_k + 2x_k - 2y_k + 5$$

Code :

```
#include <stdio.h> #include <graphics.h> void
drawCircleMidpoint(int xc, int yc, int radius)
{     int gd = DETECT, gm;     initgraph(&gd, &gm,
NULL);     int x = radius;     int y = 0;     int p
= 1 - radius;     putpixel(x + xc, y + yc, WHITE);
if (radius > 0) {         putpixel(x + xc, -y + yc,
WHITE);         putpixel(y + xc, x + yc, WHITE);
putpixel(-y + xc, x + yc, WHITE);
    }     while (x > y)
{         y++;         if (p <= 0)
p = p + 2 * y + 1;         else
{             x--;             p = p
+ 2 * y - 2 * x + 1;
        }         if (x < y)
break;         putpixel(x + xc, y + yc,
WHITE);         putpixel(-x + xc, y + yc,
WHITE);         putpixel(x + xc, -y + yc,
WHITE);         putpixel(-x + xc, -y + yc,
WHITE);         if (x != y)
{             putpixel(y + xc, x + yc,
WHITE);             putpixel(-y + xc, x + yc,
WHITE);             putpixel(y + xc, -x + yc,
WHITE);             putpixel(-y + xc, -x +
yc, WHITE);
        }     }
delay(5000);
closegraph();
} int main() {     int xc, yc, radius;
printf("Enter the center of the circle (xc, yc): ");
scanf("%d %d", &xc, &yc);
```

```
    printf("Enter the radius of the circle: ");     scanf("%d", &radius);
drawCircleMidpoint(xc, yc, radius);     return 0;
}
```

Output:



```
Hello, Graphics!
```

Conclusion:

The Midpoint Circle Drawing Algorithm is a simple and efficient algorithm for drawing circles. It is based on the concept of symmetry and reduces the computational load by taking advantage of the symmetry properties of a circle. The algorithm uses integer arithmetic and avoids the need for expensive floating-point operations.

One notable advantage of the Midpoint Circle Drawing Algorithm is its efficiency in terms of both time and space. It generates circle points using only integer addition and subtraction operations, making it suitable for systems with limited computational resources.

However, it's important to note that this algorithm is specifically designed for integer coordinates and may not produce accurate results for circles with very large radii. In such cases, other algorithms, such as the Bresenham Circle Algorithm, might be preferred. Overall, the Midpoint Circle Drawing Algorithm is a fundamental and widely used technique in computer graphics for rendering circular shapes efficiently.