# EXPERIMENT - 4

## COMPUTER GRAPICS AND MULTIMEDIA

### Midpoint Circle Generation Algorithm

To implement midpoint circle generation algorithm or bresenham's circle algorithm for drawing a circle of given center (x, y) and radius r

Syeda Reeha Quasar
14114802719
Group - 3C7

# EXPERIMENT – 4

## AIM:

To implement midpoint circle generation algorithm or bresenham's circle algorithm for drawing a circle of given center (x, y) and radius r.

## THEORY:

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

Bresenham's circle algorithm calculates the locations of the pixels in the first 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.

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

### Decision parameters:

1. In this the input radius r is there with a center **(xc , yc ).** To obtain the first point m the circumference of a circle is centered on the origin as **(x0 ,y0 ) = (0,r).**
2. Calculate the initial decision parameters which are:

   **p0 =5/4-r or 1-r**
3. Now at each xk position starting k=0, perform the following task.
   if pk < 0 then plotting point will be ( xk+1 ,yk ) and

   **Pk+1=pk +2(xk+1) +1**

   else the next point along the circle is (xk+1, yk-1 ) and

   **Pk+1=pk+2(xk+1) +1-2(yk+1)**
4. Determine the symmetry points in the other quadrants.
5. Now move at each point by the given center that is:

   **x=x+xc y=y+yc**
6. At last repeat steps from 3 to 5 until the condition **x>=y**

**Symmetric pixelpositions:**

- putpixel(xc+x,yc-y,GREEN); //For pixel(x,y)
- putpixel(xc+y,yc-x, GREEN); //For pixel(y,x)
- putpixel(xc+y,yc+x, GREEN); //For pixel(y,-x)
- putpixel(xc+x,yc+y, GREEN); //For pixel(x,-y)
- putpixel(xc-x,yc+y, GREEN); //For pixel(-x,-y)
- putpixel(xc-y,yc+x, GREEN); //For pixel(-y,-x)
- putpixel(xc-y,yc-x, GREEN); //For pixel(-y,x)
- putpixel(xc-x,yc-y, GREEN); //For pixel(-x,y)

## Midpoint Circle Generation Algorithm:

**Given -** Centre point of Circle = (X0 , Y0 )

Radius of Circle = R

The points generation using Mid-Point Circle Drawing Algorithm involves the following steps-

**Step-01:**

Assign the starting point coordinates (X0, Y0) as –

X0 = 0

Y0 = R

**Step-02:**

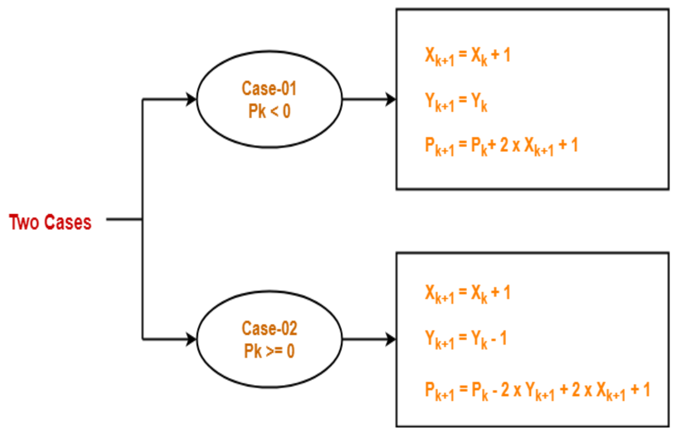
Calculate the value of initial decision parameter P0 as –

P0 = 1 – R

**Step3:**

Suppose the current point is (Xk , Yk ) and the next point is (Xk+1, Yk+1).

Find the next point of the first octant depending on the value of decision parameter Pk.



**Step4:**

If the given centre point (X0, Y0 ) is not (0, 0), then do the following and plot the point –

Xplot = Xc + X0

Yplot = Yc + Y0

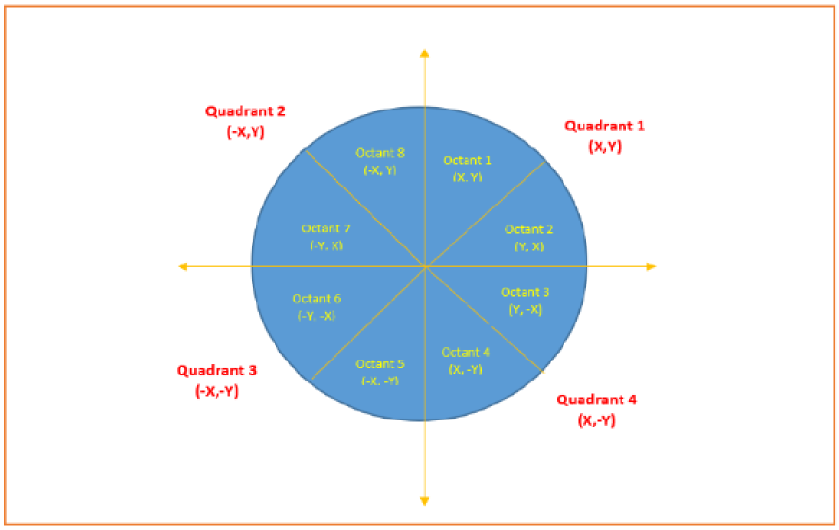
Here, (Xc , Yc ) denotes the current value of X and Y coordinates.

**Step5:**

Keep repeating Step-03 and Step-04 until Xplot >= Yplot.

**Step6:**

Step-05 generates all the points for one octant. To find the points for other seven octants, follow the eight symmetry property of circle



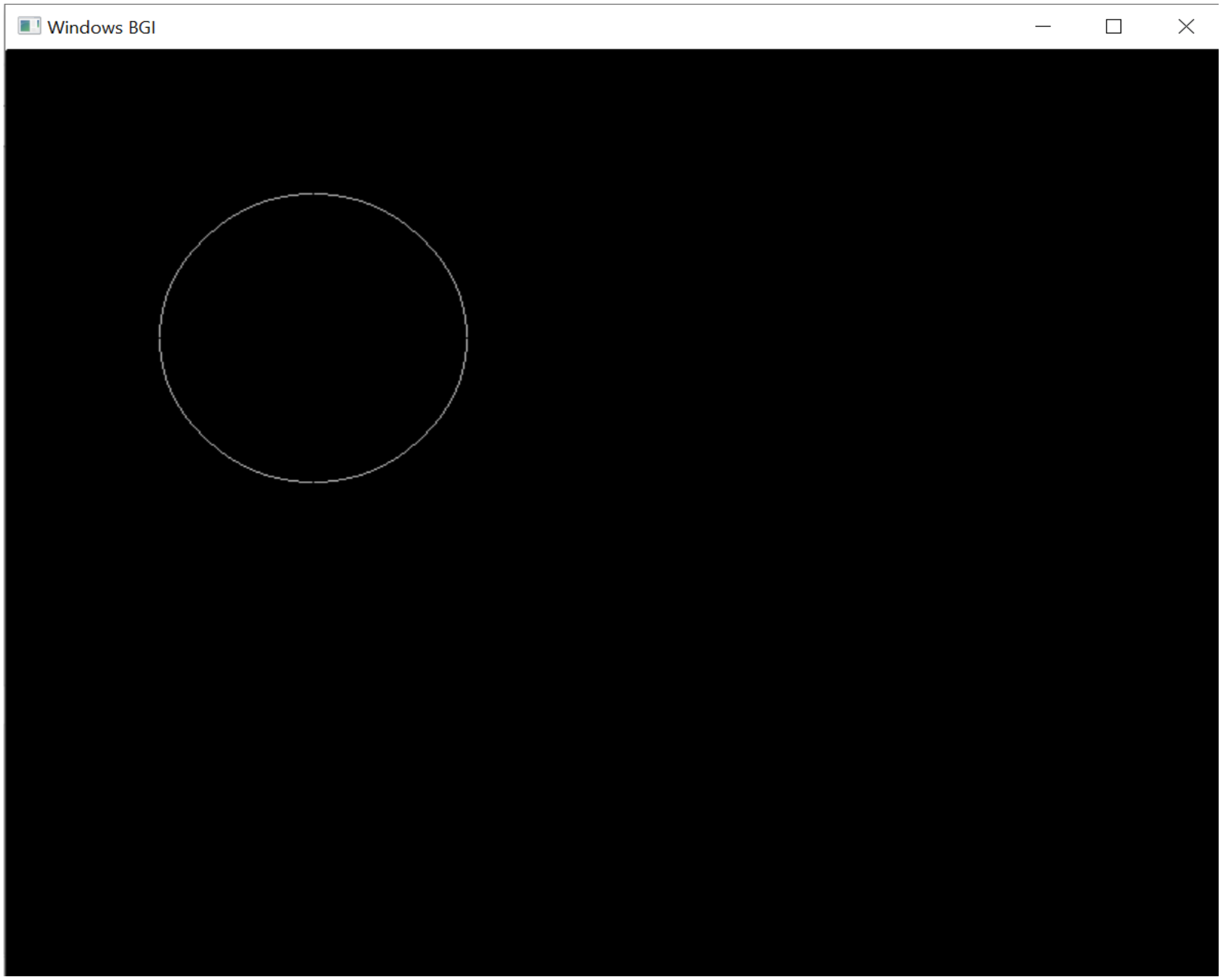
**Step7:** End Algorithm

## Algorithm implementation:

```
#include<graphics.h>

#include<conio.h>

#include<stdio.h>
```

```
main(){

        initwindow(800,800);

        int i, r=100, x=0,y,xc=200,yc=200;

        float d;

        d=1.25-r;

        y=r;

        do{

                if(d<0.0){

                        x=x+1;

                        d=d+2*x+1;

                }else{

                        x=x+1;

                        y=y-1;

                        d=d+2*x-2*y+10;

                }

        putpixel(xc+x,yc+y,7);

        putpixel(xc-y,yc-x,7);

        putpixel(xc+y,yc-x,7);

        putpixel(xc-y,yc+x,7);

        putpixel(xc+y,yc+x,7);

        putpixel(xc-x,yc-y,7);

        putpixel(xc+x,yc-y,7);

        putpixel(xc-x,yc+y,7);

        }

        while(x<y);

        getch();

}
```

## OUTPUT:

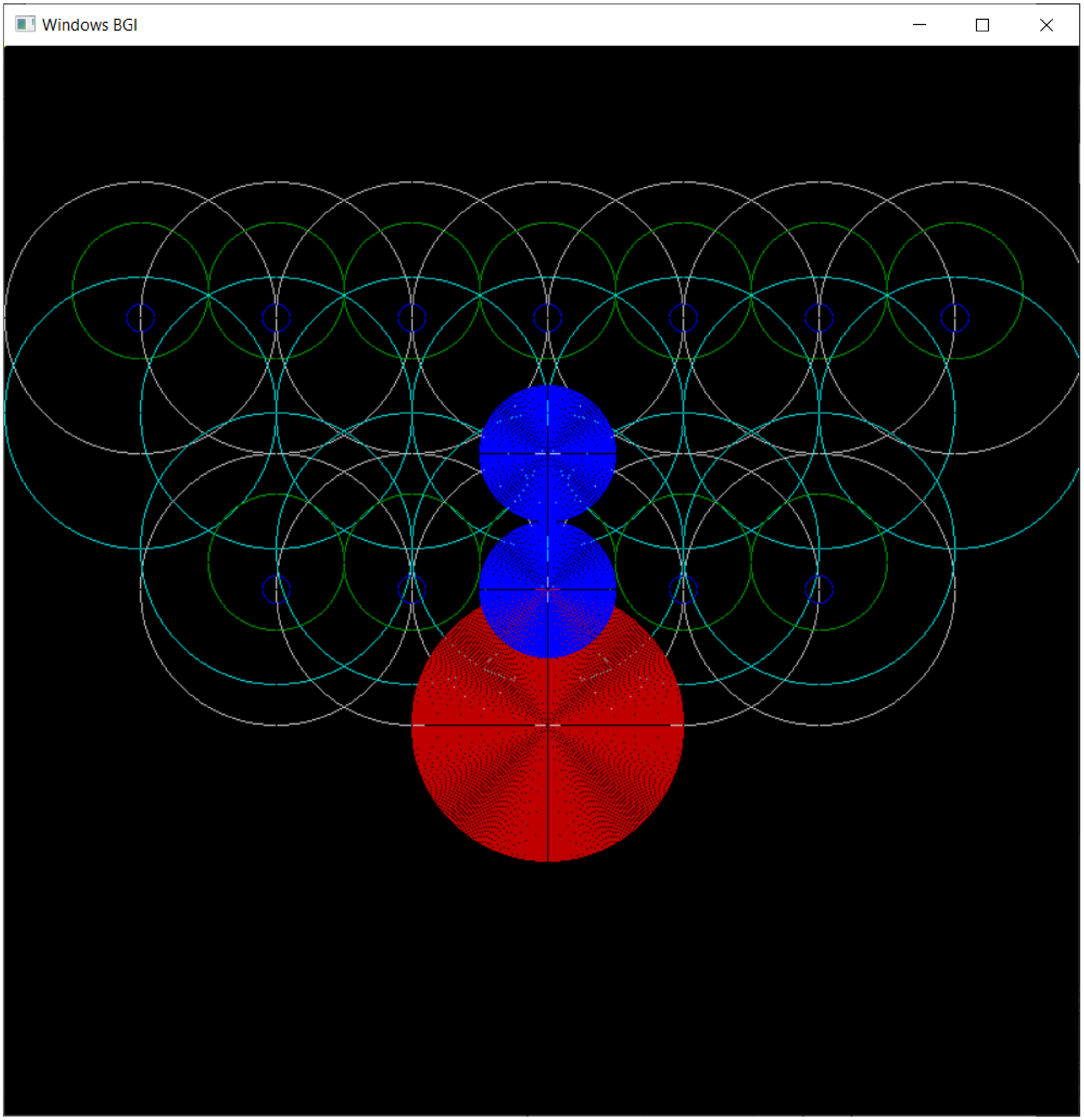


## Source Code:

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

circle_func(int xc,int yc,int r, int clr )

{

        int i,y,x=0;

        float d;


        d=1.25-r;

        y=r;

        do{

                if(d<0.0){

                        x=x+1;

                        d=d+2*x+1;

                }

                else{

                        x=x+1;

                        y=y-1;

                        d=d+2*x-2*y+1;

                }

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

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

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

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

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

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

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

        }

        while(x<y);

}


main()

{

        initwindow(800,800);

        int i = 100;

        while (i <= 700){

                circle_func(i,200,10, 1);

                circle_func(i,200,100, 7);

                circle_func(i,180,50, 2);

                circle_func(i,270,100, 3);

                i += 100;

                delay (100);

        }


        i = 200;

        while (i < 700){

                circle_func(i,400,10, 1);

                circle_func(i,400,100, 7);

                circle_func(i,380,50, 2);

                circle_func(i,370,100, 3);

                i += 100;

                delay (100);
```

```
	}


		for (i = 1; i <= 100; i++)

	{

		circle_func(400,500,i, 4);

	}

	delay (100);

	for (i = 1; i <= 50; i++)

	{

		circle_func(400,300,i, 1);

		circle_func(400,400,i, 1);

	}

	getch();

}
```

## OUTPUT:



## VIVA QUESTIONS:

### Q1. How many parameters are required to draw the circle?

Ans.

**3** Parameters are required to draw a circle that are – coordinates of the center x, y and the radius of the circle.

Syntax : circle(x, y, radius); where, (x, y) is center of the circle. 'radius' is the Radius of the circle.

### Q2. Why we calculate the initial value of the decision parameter as P0=5/4 – r ≈ 1 – r?

Ans. $P_0 = 5/4 - r \approx 1 - r$

For initial point, $(x_0, y_0) = (0, r)$

$P0 = f_{circle}\ (1, r - {}^1/_2)$

$= 1 + (r - {}^1/_{2)})^2 - r^2$

$= 5/4 - r$

$\sim 1 - r$

### Q3. What are symmetry points?

Ans.

Point Symmetry is when every part has a matching part: the same distance from the central point. but in the opposite direction. A circle have infinite points of symmetry. Circle drawing algorithms take the advantage of 8 symmetry property of circle. Every circle has 8 octants and the circle drawing algorithm generates all the points for one octant. The points for other 7 octants are generated by changing the sign towards X and Y coordinates. implementation.

### Q4. What is plot points?

Ans.

A plot is a graphical technique for representing a data set, usually as a graph showing the relationship between two or more variables and the points we plot are called the plot points. Point plotting is an elementary mathematical skill required in analytic geometry. Plot points vary as per the conditions given. In the mid-point algorithm we calculate the $P_k$ and according to that we decide the next plotting points.

**Q5. What is circle midpoint?**

Ans.

The midpoint circle algorithm is an algorithm used to determine the points needed for rasterizing a circle. Bresenham's circle algorithm is derived from the midpoint circle algorithm. This is an algorithm which is used to calculate the entire perimeter points of a circle in a first octant so that the points of the other octant can be taken easily as they are mirror points; this is due to circle property as it is symmetric about its center. In this technique algorithm determines the midpoint between the next 2 possible consecutive pixels and then checks whether the midpoint in inside or outside the circle and illuminates the pixel accordingly.