EXPERIMENT - 5

COMPUTER GRAPICS AND MULTIMEDIA

BRESENHAM'S CIRCLE DRAWING A LGORITHM

Implementation of Bresenham's circle drawing algorithm

Syeda Reeha Quasar
14114802719
Group - 3C7

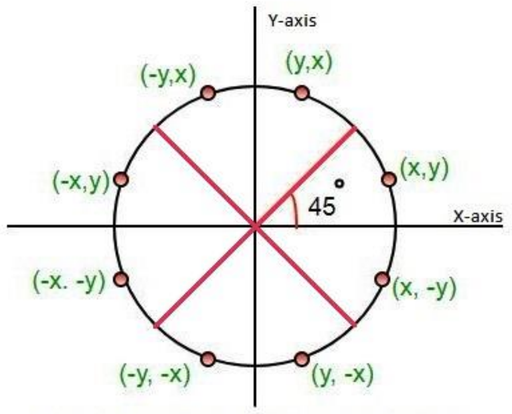
EXPERIMENT – 5

AIM:

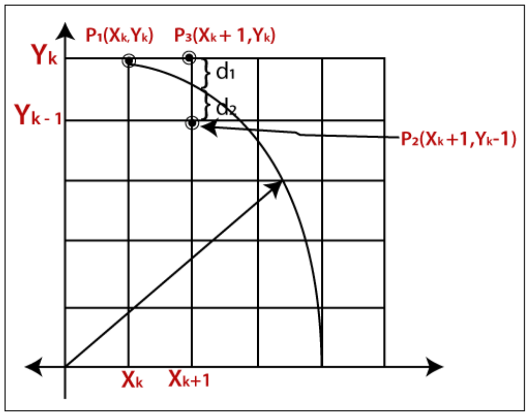
Implementation of Bresenham's circle drawing algorithm

THEORY:

- Bresenham's circle drawing algorithm selects the nearest pixel position to complete the arc.
- It uses only integer arithmetic which makes it faster than any other algorithms.
- The circle can be divided into 8 octants each of 45 degree where centre of the circle is (0,0)



For each pixel (x,y) all possible pixels in 8 octants



Bresenham Circle Algorithm:

Pseudocode:

Step 1: Set initial values of (xc, yc) and (x, y).

Where (xc,xy) are center of the circle

(x,y) are the starting point.

x= 0 and y= r (r is the radius of the circle)

Step 2: Calculate decision parameter d such that d = 3 – (2 * r).

Step 3: Call the display_Circle(int xc, int yc, int x, int y) function to display initial point (0,r).

Step 4: Repeat steps 5 to 8 until (x < = y)

Step 5: Increment value of x.

Step 6: If d < 0, set d = d + (4*x) + 6

Step 7: Else, set d = d + 4 * (x − y) + 10 and decrement y by 1.

Step 8: call display_Circle(int xc, int yc, int x, int y) method.

Step 9: Exit.

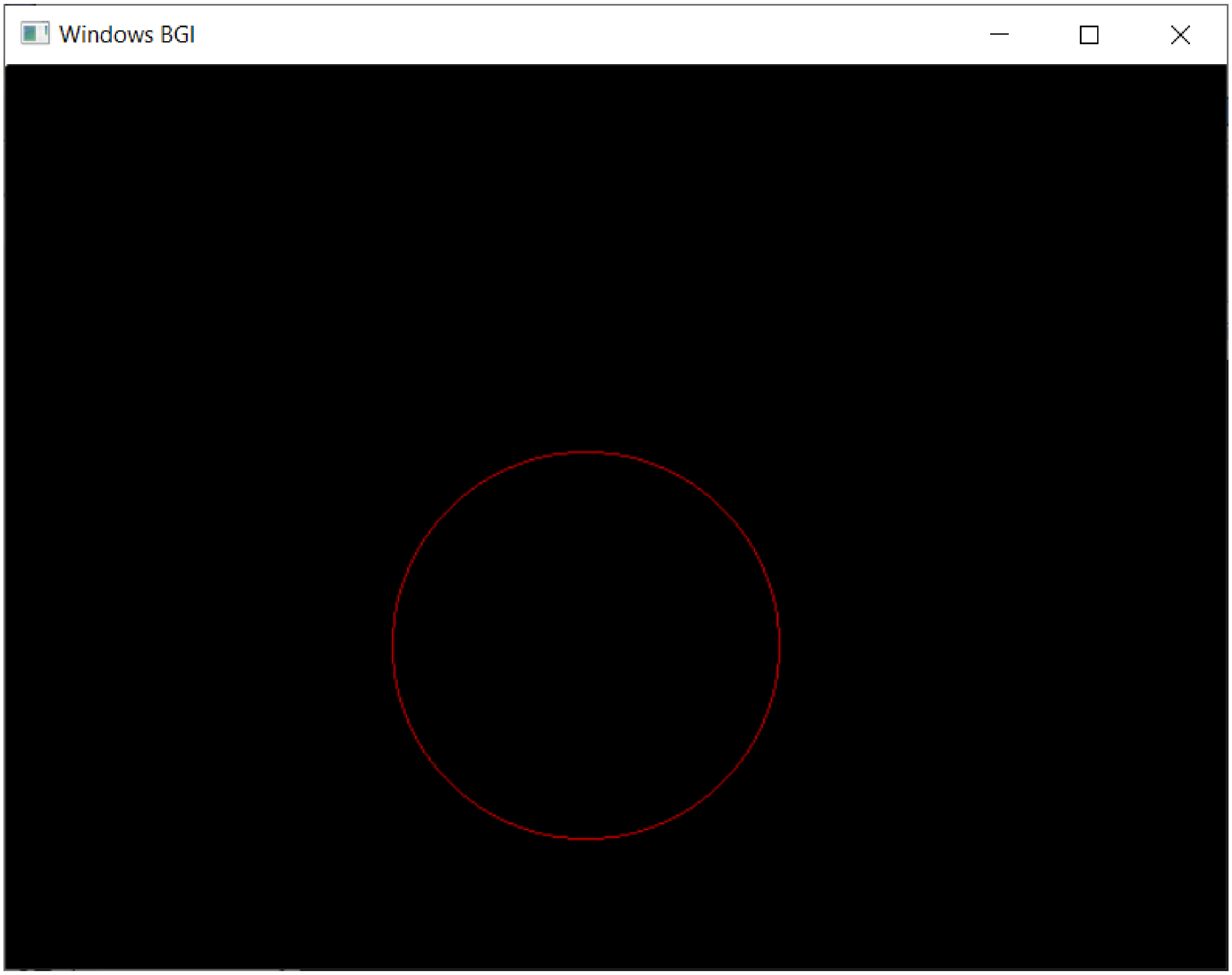
Algorithm implementation:

Source Code:

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>

void EightSymmetricPlot(int xc, int yc, int x, int y, int clr)
{
        putpixel(x + xc, y + yc, clr);
        putpixel(x + xc, -y + yc, clr);
        putpixel(-x + xc, -y + yc, clr);
        putpixel(-x + xc, y + yc, clr);
        putpixel(y + xc, x + yc, clr);
        putpixel(y + xc, -x + yc, clr);
        putpixel(-y + xc, -x + yc, clr);
        putpixel(-y + xc, x + yc, clr);
}

void BresenhamCircle(int xc, int yc, int r, int clr)
```

```
{
        int x = 0, y = r, d = 3 - ( 2 * r);
        EightSymmetricPlot(xc, yc, x, y, clr);

        while (x <= y)
        {
                if (d <= 0)
                {
                        d = d + (4 * x) + 6;
                }
                else
                {
                d = d + (4 * x) - (4 * y) + 10;
                y = y - 1;
                }
                x = x + 1;
                EightSymmetricPlot(xc, yc, x, y, clr);
                delay(100);
        }
}

int main(void)
{
        int xc, yc, r, c = 0, i, j, gdriver=DETECT, gmode;
        initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");

        printf("\n Enter the values of xc and yc: \n");
        scanf("%d %d", &xc, &yc);
        printf("\n Enter the value of radius: \n");
        scanf("%d", &r);

        BresenhamCircle(xc, yc, r, 4);
        getch();
        closegraph();
        return 0;


}
```

OUTPUT:

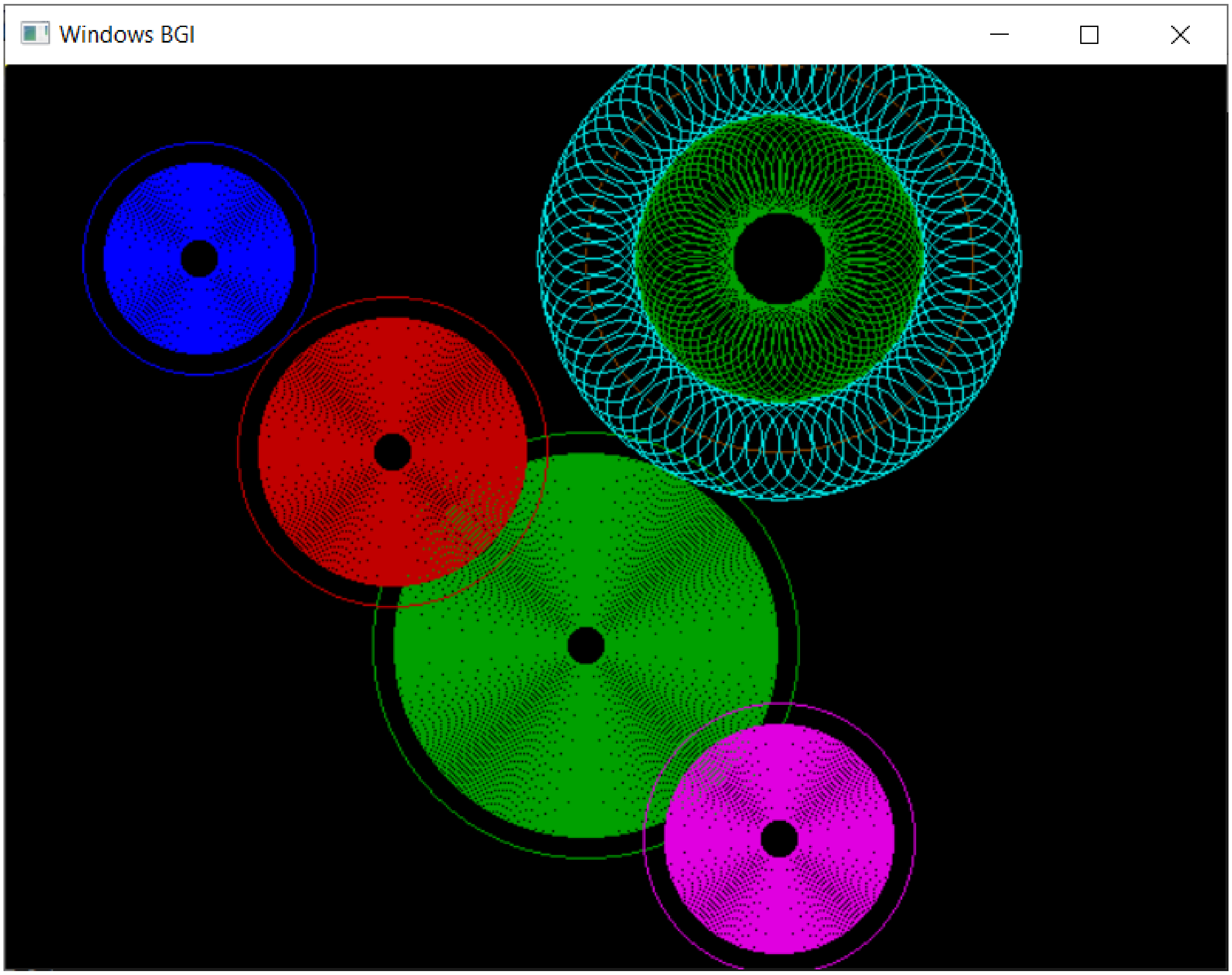


Pattern and design using the algorithm:

SOURCE CODE:

```
#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <math.h>


void EightSymmetricPlot(int xc, int yc, int x, int y, int clr)

{

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

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

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

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

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

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

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

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

}

void BresenhamCircle(int xc, int yc, int r, int clr)

{

        int x = 0, y = r, d = 3 - ( 2 * r);

        EightSymmetricPlot(xc, yc, x, y, clr);


        while (x <= y)

        {

                if (d <= 0)

                {

                        d = d + (4 * x) + 6;

                }
                else
                {
                d = d + (4 * x) - (4 * y) + 10;
                y = y - 1;
                }
                x = x + 1;
                EightSymmetricPlot(xc, yc, x, y, clr);
//              delay(100);
        }
}

int main(void)
{
        int xc, yc, r, c = 0, i, j, gdriver=DETECT, gmode;
        initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");



        for (i = 10; i < 100; i++)
        {
                BresenhamCircle(300, 300, i, 2);
        }
        BresenhamCircle(300, 300, 110, 2);
        for (i = 10; i < 50; i++)
        {
                BresenhamCircle(100, 100, i, 1);
        }
        BresenhamCircle(100, 100, 60, 1);
        for (i = 10; i < 70; i++)
        {
                BresenhamCircle(200, 200, i, 4);
        }
        BresenhamCircle(200, 200, 80, 4);
        for (i = 10; i < 60; i++)
        {
                BresenhamCircle(400, 400, i, 5);
        }
        BresenhamCircle(400, 400, 70, 5);

        BresenhamCircle(400, 100, 100, 6);

        //rangoli pattern circle
        int x, y, color, angle = 0;
  struct arccoordstype a, b;
        while (angle <= 360)
  {
    setcolor(BLACK);
    arc(400, 100, angle, angle + 2, 50);
    setcolor(RED);
    getarccoords(&a);
    BresenhamCircle(a.xstart, a.ystart, 25, 2);
    setcolor(BLACK);
    arc(400, 100, angle, angle + 2, 100);
    getarccoords(&a);
    setcolor(GREEN);
    BresenhamCircle(a.xstart, a.ystart, 25, 3);
    angle = angle+5;
    delay(50);
  }

        getch();
        closegraph();
        return 0;
}
```

OUTPUT:



VIVA QUESTIONS:

Q1. Explain the concept of 8-way symmetry?

Ans.

Circle is an eight-way symmetric figure. The shape of circle is the same in all quadrants. In each quadrant, there are two octants. If the calculation of the point of one octant is done, then the other seven points can be calculated easily by using the concept of eight-way symmetry.

Any circle follows 8-way symmetry. This means that for every point (x, y) 8 points can be plotted. These (x, y), (y, x), (-y, x), (-x, y), (-x, -y), (-y, -x), (y, -x), (x, -y). For any point (x + a, y + b), points (x ± a, y ± b) and (y ± a, x ± b) also lie on the same circle.

Q2. How is the nearest pixel position calculated to complete the arc using bresenham's algorithm?

Ans.

In Bresenham's algorithm at any point (x, y) we have two option either to choose the next pixel i.e. (x+1, y) ;(x+1, y-1). And this can be decided by using the decision parameter d as:

- If d > 0, then (x+1, y-1) is to be chosen as the next pixel as it will be closer to the arc.
- else (x+1, y) is to be chosen as next pixel.

Q3. Bresenham's algorithm uses which type of arithmetic to find the pixel position?

Ans.

Bresenham's algorithm implemented entirely with integer numbers and the integer arithmetic. It only uses addition and subtraction and avoids heavy operations like multiplication and division.

Q4. How is the bresenham's circle drawing algorithm different from mid-point circle drawing algorithm?

Ans.

Bresenham's line algorithm determines the points of a raster that have to be selected to form an approximate straight line between two points. It is used in a bitmap image to draw line primitives. It uses integer addition, bit shifting and subtraction which are cheap operations in computer architectures. It is one of the earliest algorithms and is important even in the present because of its speed and simplicity. Its extension can be used for drawing circles.

Midpoint circle algorithm is used in computer graphics to determine the points required for rasterizing a circle. The Bresenham's circle algorithm is derived from it. The midpoint circle algorithm may be generalized to conic sections.