Lab Report -1

1. scan conversion of a point

Code:

```
#include <iostream>
#include <graphics.h>


int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    setcolor(RED);
    int x = 100;
    int y = 100;
    putpixel(x, y, getcolor());
    delay(5000);
    closegraph();
    return 0;
}
```

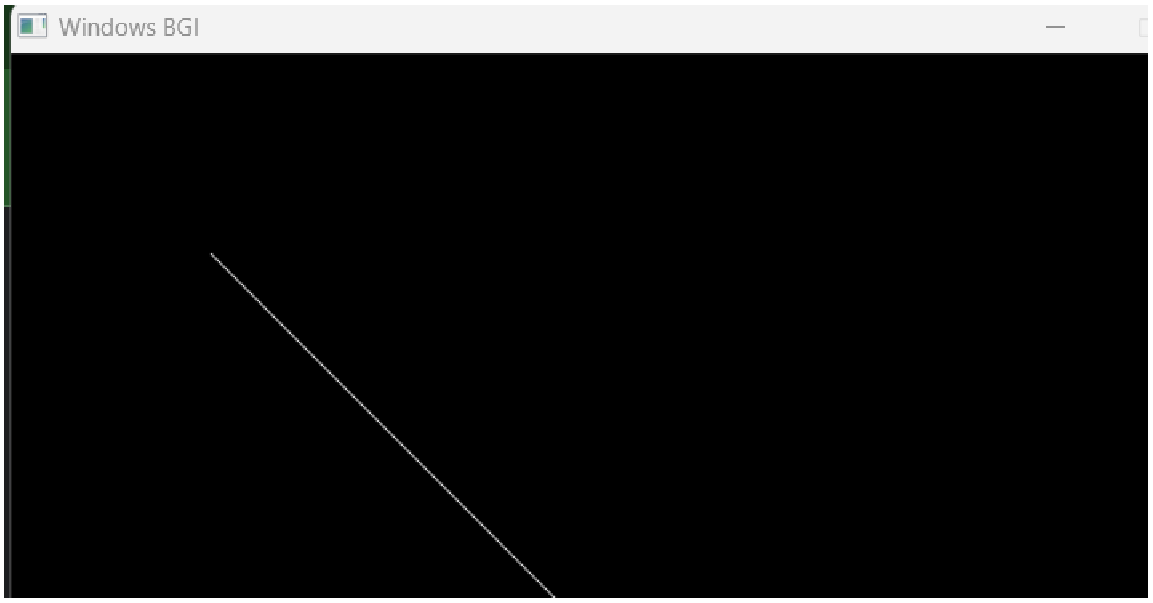
OUTPUT:

2. **scan conversion of a straight line using DDA algorithm**

Code:

```
#include <iostream>

#include <graphics.h>


void drawLineDDA(int x1, int y1, int x2, int y2) {

   int dx = x2 - x1;

   int dy = y2 - y1;


   int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);


   float xInc = static_cast<float>(dx) / steps;

   float yInc = static_cast<float>(dy) / steps;
}



   float x = x1, y = y1;


   for (int i = 0; i <= steps; i++) {
       putpixel(static_cast<int>(x + 0.5), static_cast<int>(y + 0.5), WHITE);


       x += xInc;
       y += yInc;
   }
}


int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");


   int x1 = 100, y1 = 100;
   int x2 = 300, y2 = 300;


   drawLineDDA(x1, y1, x2, y2);


   delay(5000);
   closegraph();
   return 0;
}
```

OUTPUT:

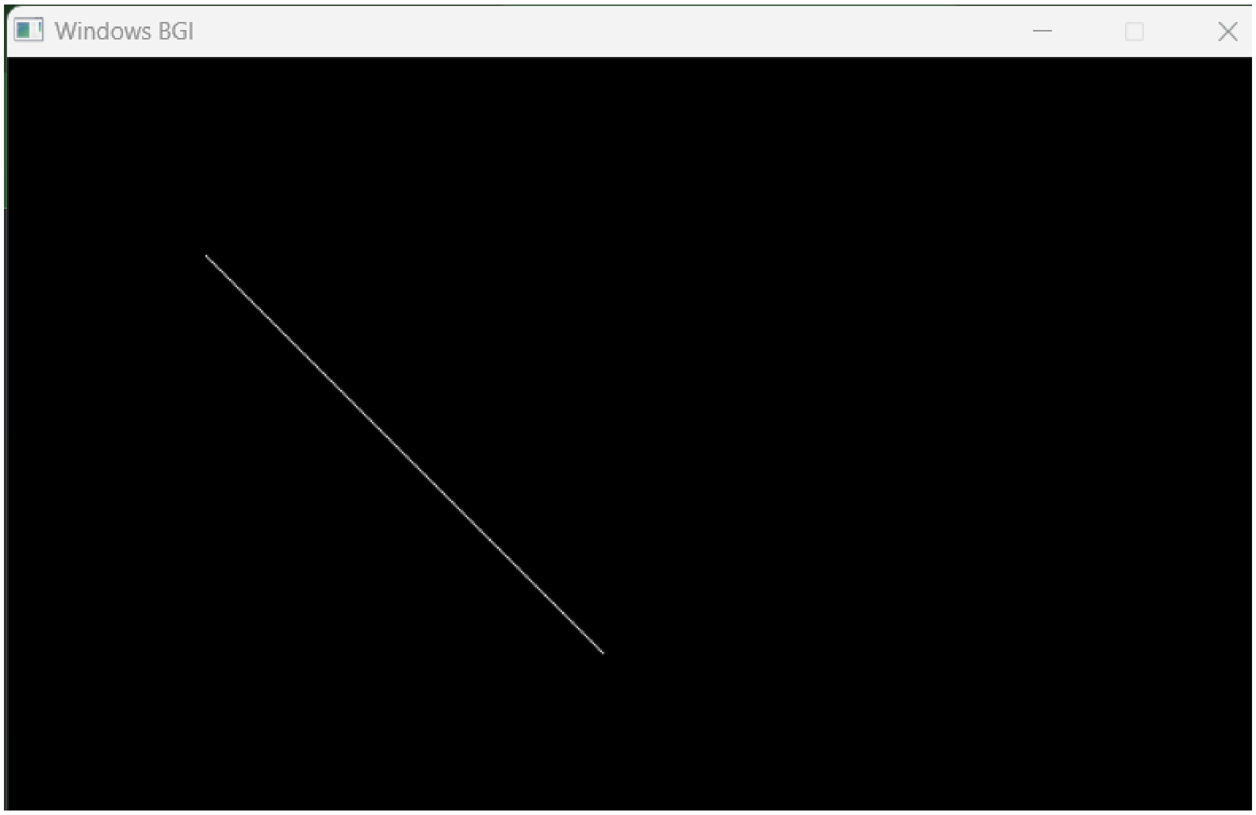


3. **scan conversion a straight line using the Bresenham's line algorithm**

Code:

```
#include <iostream>

#include <graphics.h>

void drawLineBresenham(int x1, int y1, int x2, int y2) {

    int dx = abs(x2 - x1);

    int dy = abs(y2 - y1);

    int sx = (x1 < x2) ? 1 : -1;

    int sy = (y1 < y2) ? 1 : -1;

    int err = dx - dy;


    while (true) {

        putpixel(x1, y1, WHITE);
        if (x1 == x2 && y1 == y2)
            break;

        int e2 = 2 * err;

        if (e2 > -dy) {
            err -= dy;
            x1 += sx;
        }

        if (e2 < dx) {
            err += dx;
            y1 += sy;
        }
    }
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    int x1 = 100, y1 = 100;
    int x2 = 300, y2 = 300;
    drawLineBresenham(x1, y1, x2, y2);


    delay(5000);
    closegraph();
    return 0;
}
```

OUTPUT:

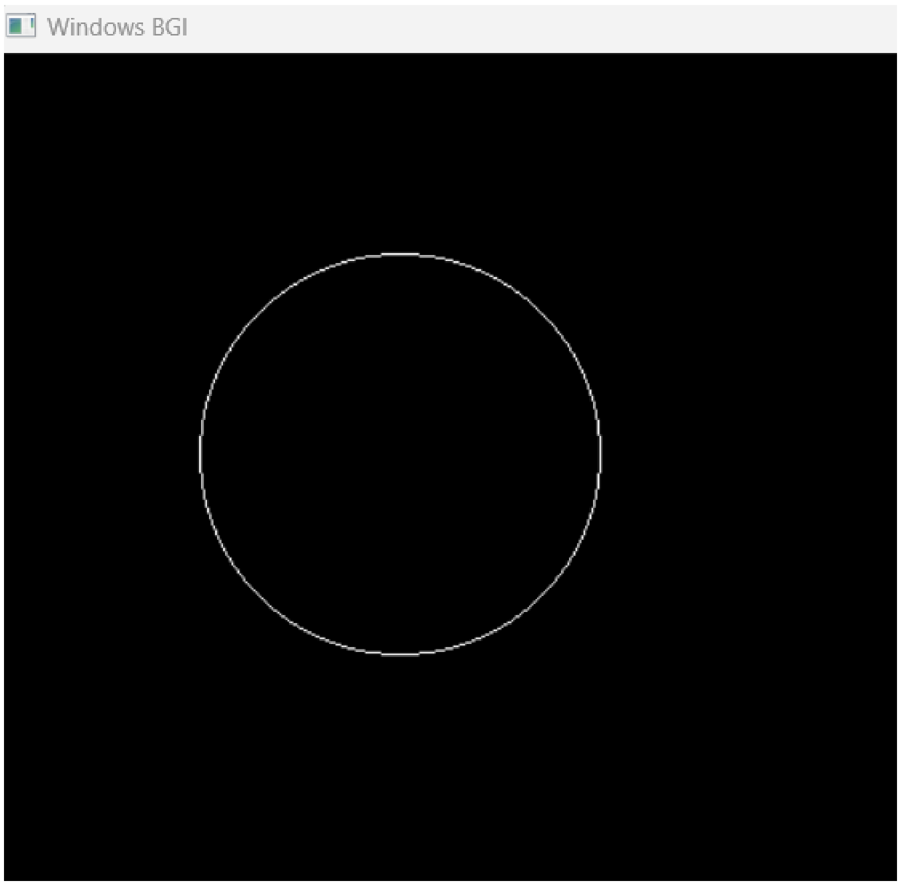


4. **scan conversion a circle using the Bresenham's line algorithm**

CODE:

```
#include <iostream>

#include <graphics.h>


void drawCircleBresenham(int xc, int yc, int radius) {

    int x = 0;

    int y = radius;

    int d = 3 - 2 * radius;


    while (x <= y) {

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

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

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

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

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

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

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

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


        if (d <= 0) {

            d += (4 * x) + 6;

        } else {

            d += (4 * (x - y)) + 10;

            y--;

        }
        x++;
    }
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    int xc = 200, yc = 200;
    int radius = 100;

    drawCircleBresenham(xc, yc, radius);

    delay(5000);
    closegraph();
    return 0;
}
```

OUTPUT:


Windows BGI