Lab Report 3

Title: To use Midpoints algorithm to draw a circle.

Theory: The Midpoint Circle Drawing Algorithm is important in computer graphics because it efficiently and accurately draws circles using whole numbers. Unlike the Digital Differential Analyzer (DDA), which uses decimals, this algorithm works with integers, making it suitable for devices with limited computing power. This helps it work quickly, which is crucial for graphics that need to be updated in real-time.

One of the strengths of this algorithm is how it creates circles. It plots pixels around the circle's edge in all eight parts at once, making sure the circles look smooth and just right. By using a decision maker, it figures out the best places to put each pixel, adjusting them step by step as it goes around the circle. This method saves computer power and keeps the drawings precise, making it a top choice for software and systems that need to work fast and well.

Midpoint Algorithm:

Step 1: Start

Step 2: Initialize the midpoint circle algorithm with parameters: radius r, center coordinates (x_c, y_c)

Step 3: Set the values, x1 = 0 , y1 = radius

Step 4: Initialize the decision parameter, p = 1 - radius

Step 5: Draw a octant

Step 6: Use a loop to generate points symmetrically around all octants of circle

```
while (x1 <= y1) {
  if (p < 0) {
    p = p + 2 * x1 + 3;
    x1++;
  } else {
    p = p + 2 * x1 - 2 * y1 + 5;
    x1++;
    y1--;
  }

  putpixel(x_c + x1, y_c + y1, WHITE); // Octant 1
```

```
        putpixel(x_c + y1, y_c + x1, WHITE); // Octant 2
        putpixel(x_c - y1, y_c + x1, WHITE); // Octant 3
        putpixel(x_c - x1, y_c + y1, WHITE); // Octant 4
        putpixel(x_c - x1, y_c - y1, WHITE); // Octant 5
        putpixel(x_c - y1, y_c - x1, WHITE); // Octant 6
        putpixel(x_c + y1, y_c - x1, WHITE); // Octant 7
        putpixel(x_c + x1, y_c - y1, WHITE); // Octant 8
    }
    Step 7: Stop
```

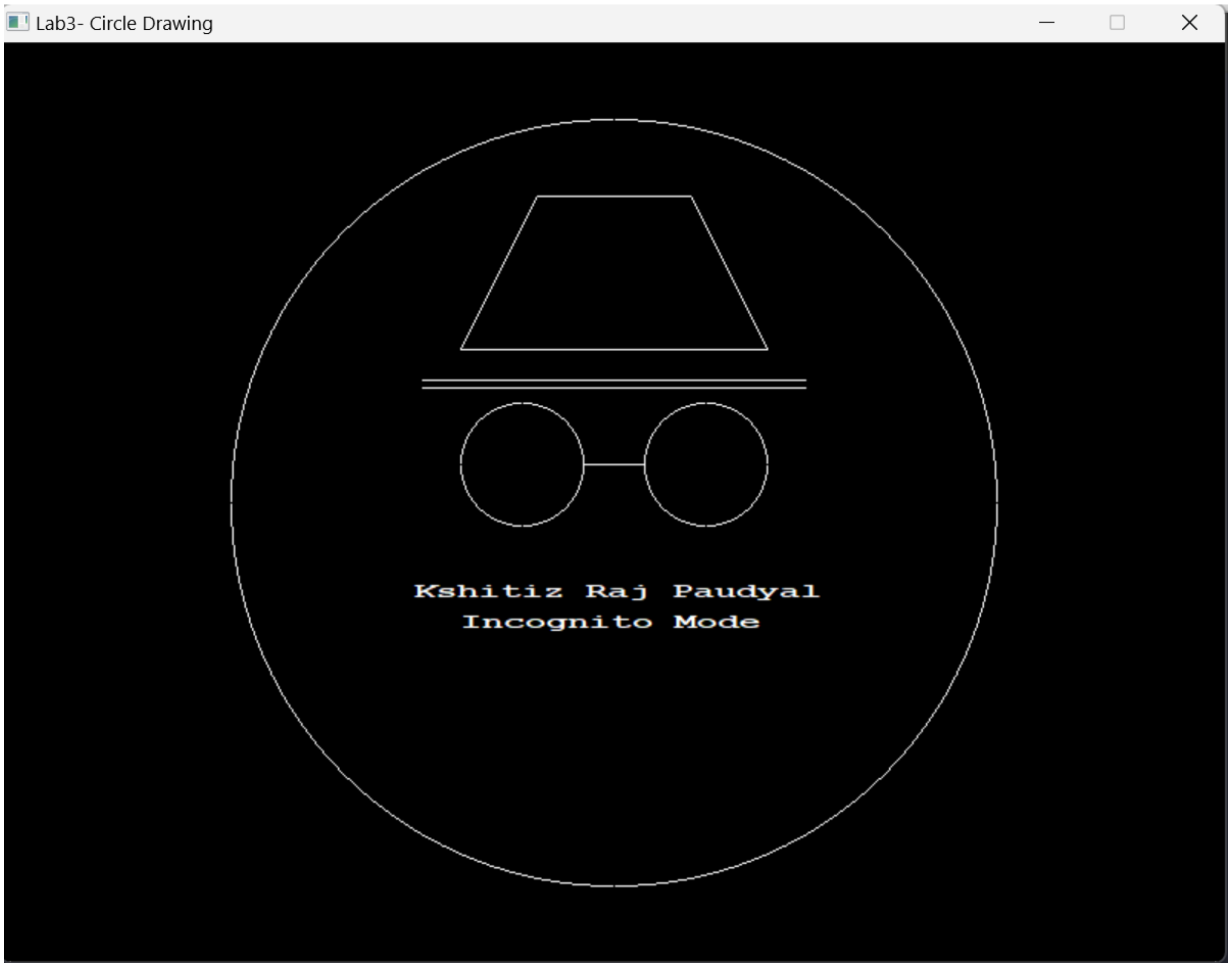
The code:

Using the application of drawing circle and a line, (Bresenham's & Midpoint algorithm in LAB 2 & 3) a simple incognito symbol is drawn using just line and circle, the code for it is given below:

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

using namespace std;

class midpoint_algorithm {
private:
    int radius;
    int x_c, y_c, x1, y1;

public:
    midpoint_algorithm(int r, int x, int y) {
        x_c = x;
        y_c = y;
        radius = r;
        x1 = 0;
        y1 = radius;
    }

    void draw_circle() {
        int p = 1 - radius;
        while (x1 <= y1) {
            if (p < 0) {
                p = p + 2 * x1 + 3;
                x1++;
            } else {
                p = p + (2 * x1) - (2 * y1) + 5;
                x1++;
                y1--;
            }

            putpixel(x_c + x1, y_c + y1, WHITE); // Octant 1
            putpixel(x_c + y1, y_c + x1, WHITE); // Octant 2
            putpixel(x_c - y1, y_c + x1, WHITE); // Octant 3
            putpixel(x_c - x1, y_c + y1, WHITE); // Octant 4
            putpixel(x_c - x1, y_c - y1, WHITE); // Octant 5
            putpixel(x_c - y1, y_c - x1, WHITE); // Octant 6
            putpixel(x_c + y1, y_c - x1, WHITE); // Octant 7
            putpixel(x_c + x1, y_c - y1, WHITE); // Octant 8
        }
    }
};

class bresenhams_algorithm {
private:
    int x, y, dx, dy;

public:
    void draw_line(int x1, int y1, int x2, int y2) {
        x = x1;
        y = y1;
        dx = abs(x2 - x1);
        dy = abs(y2 - y1);
        int sx = (x1 < x2) ? 1 : -1;
        int sy = (y1 < y2) ? 1 : -1;

        if (dy <= dx) {
            int p = 2 * dy - dx;
            while (x != x2 + sx) {
                putpixel(x, y, WHITE);
                x += sx;
                if (p < 0)
                    p += 2 * dy;
                else {
                    p += 2 * (dy - dx);
                    y += sy;
                }
            }
        } else {
            int p = 2 * dx - dy;
            while (y != y2 + sy) {
                putpixel(x, y, WHITE);
                y += sy;
                if (p < 0)
                    p += 2 * dx;
                else {
                    p += 2 * (dx - dy);
                    x += sx;
                }
            }
        }
    }
};

int main() {
    int gd = DETECT, gm;
    initwindow(800, 600, "Lab3- Circle Drawing");

    // outer circle
    midpoint_algorithm big_circle(250, 400, 300);
    big_circle.draw_circle();

    // Hats
    bresenhams_algorithm linedrawing;
    linedrawing.draw_line(350, 100, 450, 100);
    linedrawing.draw_line(350, 100, 300, 200);
    linedrawing.draw_line(450, 100, 500, 200);
    linedrawing.draw_line(300, 200, 500, 200);

    linedrawing.draw_line(275, 220, 525, 220);
    linedrawing.draw_line(275, 225, 525, 225);

    // Glasses
    int glasses_y = 275;
    midpoint_algorithm circle1(40, 340, glasses_y);
    circle1.draw_circle();

    midpoint_algorithm circle2(40, 460, glasses_y);
    circle2.draw_circle();

    //line
    linedrawing.draw_line(380, glasses_y, 420, glasses_y);

    setcolor(WHITE);
    settextstyle(DEFAULT_FONT, HORIZ_DIR, 2);
    outtextxy(270, 350, "Kshitiz Raj Paudyal");
    outtextxy(300, 370, "Incognito Mode");
    getch();
    closegraph();

    return 0;
}
```

The output:

The output of the asked program is given below:



```
Kshitiz Raj Paudyal
Incognito Mode
```

Figure : Application of Circle Drawing in creating Incognito Symbol

Conclusion:

The Midpoint circle drawing algorithm is efficient and accurate for rendering circle using integer arithmetic, unlike other methods that rely on floating-point operations. It's systematic pixel plotting across all the octants ensures smooth and precise circle rendering , make it suitable for real-time graphics applications.