LAB 3

OBJECTIVE: To implement and understand the Midpoint Ellipse Drawing Algorithm using C++ and graphics.h library.

THEORY:

The Midpoint Ellipse Algorithm is used to draw ellipses based on the properties of symmetry. It divides the drawing of the ellipse into two regions:

- Region 1: Where the slope of the ellipse is < 1
- Region 2: Where the slope is >= 1

The algorithm uses a decision parameter to decide the next pixel location. Because of symmetry, for each point (x, y), the algorithm plots points in all four quadrants of the ellipse.

ALGORITHM:

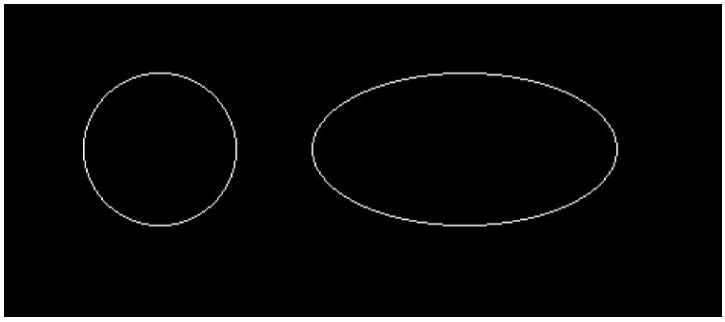
1. Input center (xc, yc) and radii (rx, ry)
2. Initialize x = 0, y = ry
3. Region 1:
   - Use the decision parameter to decide whether to move in x or y
4. Region 2:
   - Transition after slope becomes > 1
   - Adjust decision parameters accordingly
5. Plot 4 symmetrical points each iteration using putpixel()
6. Repeat until full ellipse is drawn

PROGRAMS

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

// Function to draw a circle using the Midpoint Algorithm
void drawCircle(int xc, int yc, int rad) {
    int pnc = 1 - rad; // Decision parameter
    int x = 0;
    int y = rad;

    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);

        x++;
        if (pnc < 0) {
            pnc = pnc + 2 * x + 1;
        } else {
            y--;
            pnc = pnc + 2 * (x - y) + 1;
        }
    }
}

// Function to draw an ellipse using the Midpoint Algorithm
void drawEllipse(int xc, int yc, int rx, int ry) {
    float x = 0, y = ry;

    float rxSq = rx * rx;
    float rySq = ry * ry;

    float dx = 2 * rySq * x;
    float dy = 2 * rxSq * y;

    float p1 = rySq - (rxSq * ry) + (0.25 * rxSq);

    // Region 1
```

```
    while (dx < dy) {
        putpixel(xc + x, yc + y, WHITE);
        putpixel(xc - x, yc + y, WHITE);
        putpixel(xc + x, yc - y, WHITE);
        putpixel(xc - x, yc - y, WHITE);

        x++;
        dx = 2 * rySq * x;

        if (p1 < 0) {
            p1 = p1 + dx + rySq;
        } else {
            y--;
            dy = 2 * rxSq * y;
            p1 = p1 + dx - dy + rySq;
        }
    }

    // Region 2
    float p2 = (rySq) * (x + 0.5) * (x + 0.5) +
(rxSq) * (y - 1) * (y - 1) - (rxSq * rySq);

    while (y >= 0) {
        putpixel(xc + x, yc + y, WHITE);
        putpixel(xc - x, yc + y, WHITE);
        putpixel(xc + x, yc - y, WHITE);
        putpixel(xc - x, yc - y, WHITE);

        y--;
        dy = 2 * rxSq * y;

        if (p2 > 0) {
            p2 = p2 - dy + rxSq;
        } else {
            x++;
            dx = 2 * rySq * x;
            p2 = p2 + dx - dy + rxSq;
        }
    }
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c:\\turboc3\\bgi");

    // Draw a circle
    drawCircle(200, 200, 50);

    // Draw an ellipse
    drawEllipse(400, 200, 100, 50);

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

Output:



RESULTS:

The Midpoint Ellipse Drawing Algorithm was successfully implemented and executed. The output matched the expected ellipse shape with smooth curvature in all four quadrants.

CONCLUSION:

This lab helped understand the working of the Midpoint Ellipse Algorithm, which is efficient for drawing ellipses using only integer calculations and symmetrical pixel plotting. The decision parameter technique minimizes computational overhead.