**Title**: To use **Midpoint Ellipse Drawing Algorithm** to draw an ellipse.

**Objective**:

1. To implement the Midpoint Ellipse Drawing Algorithm.
2. To evaluate how well the algorithm approximates an ellipse with pixels.
3. To analyze the efficiency and speed of the algorithm.
4. To visualize and display ellipses with varying parameters.

**Theory:**

An ellipse in a 2D plane is defined as the set of all points for which the sum of the distances to two fixed points (the foci) is constant. In computer graphics, similar to line drawing, we cannot directly render a perfect ellipse due to the discrete nature of pixels. Instead, we approximate the ellipse by determining and illuminating the closest pixels that best represent its elliptical path. The equation of an ellipse centered at the origin (0,0) with semi-major axis a and semi-minor axis b is given by (x/rx)2 + (y/ry)2 = 1.

In graphics programming, the output screen acts as a coordinate system with the origin (0, 0) at the top-left corner. The x-coordinate increases to the right, and the y-coordinate increases downward. The Midpoint Ellipse Drawing Algorithm (MEDA) is an efficient method to compute the pixel coordinates that form an ellipse. It utilizes integer calculations to minimize computational overhead and ensures smooth, visually appealing ellipses by deciding the optimal pixels to activate based on the ellipse's curvature.

The initial coordinates are (0,ry).

For region 1:

The initial decision parameter is P10 = ry2 - rx2ry + (1/4)\*rx2

If P1k < 0, the next pixel is at (xk + 1, yk).

P1k+1 = P1k + 2ry2xk+1 + ry2

If P1k ≥ 0, the next pixel is at (xk + 1, yk - 1).

P1k+1 = P1k + 2ry2xk+1 – 2rx2yk+1 + ry2

For region 2:

The initial decision parameter is P20 = ry2 (x0 + 1/2)2 + rx2(y0 - 1)2 - rx2ry2

If P2k < 0, the next pixel is at (xk + 1, yk - 1).

P2k+1 = P2k + 2ry2xk+1 – 2rx2yk+1 + rx2

If P2k ≥ 0, the next pixel is at (xk, yk - 1).

P2k+1 = P2k + 2ry2xk+1 + rx2

Using functions like `putpixel(x, y, color)` in C, MEDA allows us to render ellipse by illuminating the appropriate pixels on the screen, producing a precise and performance-friendly ellipse drawing.

****Midpoint Ellipse Drawing** Algorithm:**

1. Start
2. Declare variables xc, yc, rx, ry, x0, y0, p1k+1, and p2k+1
3. Read values of xc, yc, rx, and ry
4. Initialize the x and y i.e. set the co-oordinates for the first point on the circumference of the ellipse centered at the origin as :  
    x0 = 0;

y0 = r;

1. Calculate the initial decision parameter in region 1 as :  
    p10 = ry2 - rx2ry + (1/4)\*rx2 ;
2. At each xk position, starting from k = 0, for region 1.

If p1k < 0

xk+1  = xk  + 1

yk+1  = yk

p1k+1 = p1k + 2ry2xk+1 + ry2

else

xk+1  = xk  + 1

yk+1  = yk  - 1

p1k+1 = p1k + 2ry2xk+1 – 2rx2yk+1 + ry2

and continue until 2ry2x ≥ 2rx2y

1. Calculate the initial decision parameter in region 2 using the last point (x0, y0) calculated in the region 1 as

p20 = ry2 (x0 + 1/2)2 + rx2(y0 - 1)2 - rx2ry2

1. At each xk position, starting from k = 0, for region 1.

If p2k < 0

xk+1  = x k  + 1

yk+1  = yk - 1

p2k+1 = p2k + 2ry2xk+1 + ry2

else

xk+1  = xk

yk+1  = yk  - 1

p2k+1 = p2k + 2ry2xk+1 – 2rx2yk+1 + ry2

1. Determine the symmetry points in the other remaining quadrants.
2. Move each calculated pixel position (x, y) onto the elliptical path centered on (xc, yc) & plot the coordinates values

x = x + xc

y = y + yc

1. Repeat the steps for region 2 until y < 0.
2. Stop

**Source Code:**

// Midpoint Ellipse Drawing Algorithm to draw an ellipse.

#include <stdio.h>

#include <graphics.h>

void drawEllipsePoints(int xc, int yc, int x, int 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);

}

void midpointEllipse(int xc, int yc, int rx, int ry) {

int x = 0;

int y = ry;

// Initial decision parameter of region 1

long long rxSq = (long long)rx \* rx;

long long rySq = (long long)ry \* ry;

long long twoRxSq = 2 \* rxSq;

long long twoRySq = 2 \* rySq;

long long px = 0;

long long py = twoRxSq \* y;

// Region 1

long long p = rySq - (rxSq \* ry) + (0.25 \* rxSq);

while (px < py) {

drawEllipsePoints(xc, yc, x, y);

x++;

px += twoRySq;

if (p < 0) {

p += rySq + px;

} else {

y--;

py -= twoRxSq;

p += rySq + px - py;

}

delay(20);

}

// Region 2

p = rySq \* (x + 0.5) \* (x + 0.5) + rxSq \* (y - 1) \* (y - 1) - rxSq \* rySq;

while (y >= 0) {

drawEllipsePoints(xc, yc, x, y);

y--;

py -= twoRxSq;

if (p > 0) {

p += rxSq - py;

} else {

x++;

px += twoRySq;

p += rxSq - py + px;

}

delay(20);

}

}

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

int xc, yc, rx, ry;

printf("Created by Kushal Shah\nMid-Point Ellipse Drawing Algorithm\n");

printf("Enter center of the ellipse (xc yc): ");

scanf("%d %d", &xc, &yc);

printf("Enter radius along x-axis (rx): ");

scanf("%d", &rx);

printf("Enter radius along y-axis (ry): ");

scanf("%d", &ry);

outtextxy(20, 20, "Kushal Shah");

midpointEllipse(xc, yc, rx, ry);

delay(50000);

closegraph();

return 0;

}

**Output:**

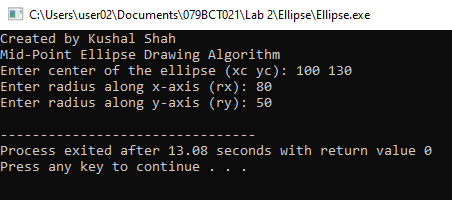


Figure 1: Inserting center and radius of the ellipse.

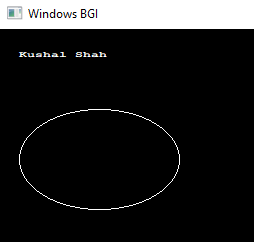


Figure 2: Drawing ellipse using MEDA

**Discussion**:

The Midpoint Ellipse Drawing Algorithm is an efficient method for rasterizing ellipses in computer graphics. By leveraging the symmetry of ellipses, the algorithm computes only one-quarter of the ellipse and mirrors the results to the other quadrants, significantly reducing the number of calculations required. This approach not only simplifies the implementation but also enhances performance, making it suitable for real-time applications where computational resources are limited.

The algorithm operates using integer arithmetic, which minimizes rounding errors and improves efficiency. It employs an incremental decision parameter to determine the next pixel to illuminate, allowing for smooth transitions along the ellipse's perimeter. However, while the algorithm excels in rendering smooth ellipses, it may struggle with precision in lower resolutions, potentially leading to a pixelated appearance. Overall, the Midpoint Ellipse Drawing Algorithm remains a fundamental technique in computer graphics, balancing simplicity and efficiency in ellipse rendering.

**Conclusion**:

The Midpoint Ellipse Drawing Algorithm is a robust technique for ellipse rendering in computer graphics. By utilizing symmetry and decision parameters, it efficiently computes pixel locations, minimizing computational load while ensuring visual accuracy. This algorithm is foundational in graphical applications, providing a reliable method for ellipse generation across various digital platforms. Its efficiency makes it suitable for real-time applications where performance is critical. The implementation stages outlined provide a clear pathway for effective coding and application of the algorithm in graphics programming.