

**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:**

**Theory of Ellipse Drawing**

An **ellipse** is mathematically defined as the set of all points in a plane where the sum of the distances to two fixed points (the **foci**) remains constant. Its standard equation, centered at the origin (0,0) with semi-major axis  $r_x$  and semi-minor axis  $r_y$ , is:

$$(x/r_x)^2 + (y/r_y)^2 = 1.$$

However, due to the **discrete nature of pixels** on a computer screen, a perfect ellipse cannot be rendered directly. Instead, we must approximate it by illuminating the closest pixels that best represent its path. In computer graphics, the **origin (0,0)** of the coordinate system is typically at the **top-left corner** of the display, with x-values increasing to the right and y-values increasing downward.

The **Midpoint Ellipse Drawing Algorithm (MEDA)** is an efficient method used for this approximation. It relies on **integer calculations** to minimize computational overhead and ensures **smooth, visually appealing ellipses**. MEDA achieves this by intelligently deciding which pixels to activate based on the ellipse's curvature, optimizing the representation on a pixel grid.

The initial coordinates are  $(0, r_y)$ .

For region 1:

The initial decision parameter is  $P_{10} = r_y^2 - r_x^2 r_y + (1/4) * r_x^2$

If  $P_{1k} < 0$ , the next pixel is at  $(x_k + 1, y_k)$ .

$$P_{1k+1} = P_{1k} + 2r_y^2 x_{k+1} + r_y^2$$

If  $P_{1k} \geq 0$ , the next pixel is at  $(x_k + 1, y_k - 1)$ .

$$P_{1k+1} = P_{1k} + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

For region 2:

The initial decision parameter is  $P_{20} = r_y^2 (x_0 + 1/2)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$

If  $P_{2k} < 0$ , the next pixel is at  $(x_k + 1, y_k - 1)$ .

$$P_{2k+1} = P_{2k} + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

If  $P_{2k} \geq 0$ , the next pixel is at  $(x_k, y_k - 1)$ .

$$P_{2k+1} = P_{2k} + 2r_y^2 x_{k+1} + r_x^2$$

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:

Step 1: Start

Step 2: Declare variables  $x_c, y_c, r_x, r_y, x_0, y_0, p_{1k+1}$ , and  $p_{2k+1}$

Step 3: Read values of  $x_c, y_c, r_x$ , and  $r_y$

Step 4: Initialize the x and y i.e. set the co-ordinates for the first point on the circumference of the ellipse centered at the origin as :

$$x_0 = 0;$$

$$y_0 = r;$$

Step 5: Calculate the initial decision parameter in region 1 as :

$$p_{10} = r_y^2 - r_x^2 r_y + (1/4) * r_x^2;$$

Step 6: At each  $x_k$  position, starting from  $k = 0$ , for region 1.

if  $p_{1k} < 0$

$$x_{k+1} = x_k + 1$$

$$y_{k+1} = y_k$$

$$p_{1k+1} = p_{1k} + 2r_y^2 x_{k+1} + r_y^2$$

else

$$x_{k+1} = x_k + 1$$

$$y_{k+1} = y_k - 1$$

$$p_{1k+1} = p_{1k} + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

and continue until  $2r_y^2 x \geq 2r_x^2 y$

Step 7: Calculate the initial decision parameter in region 2 using the last point  $(x_0, y_0)$  calculated in the region 1 as

$$p_{20} = r_y^2 (x_0 + 1/2)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

Step 8: At each  $x_k$  position, starting from  $k = 0$ , for region 1.

If  $p_{2k} < 0$

$$x_{k+1} = x_k + 1$$

$$y_{k+1} = y_k - 1$$

$$p_{2k+1} = p_{2k} + 2r_y^2 x_{k+1} + r_y^2$$

else

$$x_{k+1} = x_k$$

$$y_{k+1} = y_k - 1$$

$$p_{2k+1} = p_{2k} + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

Step 9: Determine the symmetry points in the other remaining quadrants.

Step 10: Move each calculated pixel position  $(x, y)$  onto the elliptical path centered on  $(x_c, y_c)$  & plot the coordinates values

$$x = x + x_c$$

$$y = y + y_c$$

Step 11: Repeat the steps for region 2 until  $y < 0$ .

Step 12: 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 Shishir Timilsina\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(30, 100, "Shishir Timilsina");

midpointEllipse(xc, yc, rx, ry);

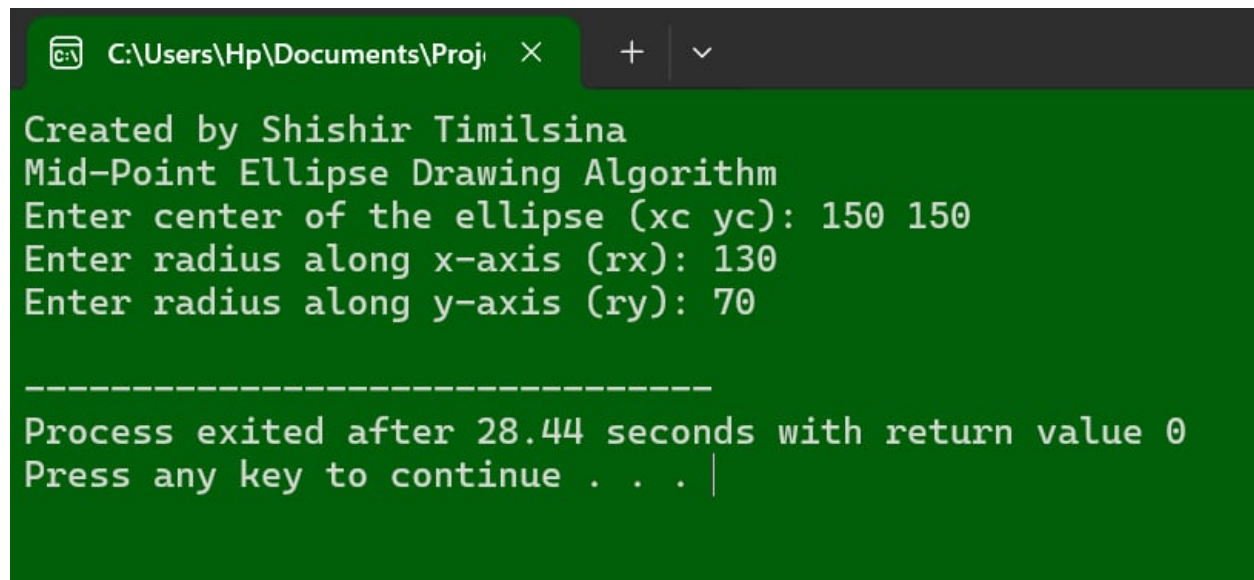
delay(50000);

closegraph();

return 0;

}
```

**Output:**



```
C:\Users\Hp\Documents\Proj >
Created by Shishir Timilsina
Mid-Point Ellipse Drawing Algorithm
Enter center of the ellipse (xc yc): 150 150
Enter radius along x-axis (rx): 130
Enter radius along y-axis (ry): 70

-----
Process exited after 28.44 seconds with return value 0
Press any key to continue . . . |
```

*Figure 1: Inserting center and radius of the ellipse.*



Figure 2: Drawing ellipse using MEDA

## Discussion

The **Midpoint Ellipse Drawing Algorithm (MEDA)** stands out as a highly efficient and foundational technique for rasterizing ellipses in computer graphics. Its core strength lies in leveraging the inherent **symmetry of ellipses**. By computing pixel coordinates for only one-quarter of the ellipse, the algorithm drastically reduces computational load, mirroring these results to the other three quadrants. This not only simplifies implementation but also significantly enhances performance, making MEDA particularly well-suited for **real-time applications** where computational resources are often constrained.

MEDA's operational efficiency is further bolstered by its reliance on **integer arithmetic**. This design choice minimizes the accumulation of rounding errors, which can be problematic in floating-point operations, thus improving overall precision and speed. The algorithm employs an **incremental decision parameter** to intelligently select the next pixel to illuminate. This ensures a smooth visual transition along the ellipse's perimeter, contributing to a high-quality rendering. While MEDA excels at producing visually smooth ellipses, a potential limitation arises in **lower resolutions**, where the discrete nature of pixels might lead to a more pixelated or less



precise appearance. Despite this, its robustness in balancing simplicity with efficiency makes it a go-to method.

## **Conclusion**

Thus, as shown in the program above, We can draw a ellipse by plotting individual pixel using Mid-point Ellipse Drawing Algorithm using graphics functions provided in the graphics.h header file.