

Implementation of Mid-Point Ellipse Drawing Algorithm

Objectives

- To study the concept of raster graphics and ellipse generation.
- To understand the working principle of the Mid-Point Ellipse Drawing Algorithm.
- To implement the Mid-Point Ellipse Drawing Algorithm using a programming language.
- To draw an ellipse efficiently using integer calculations.
- To observe symmetry properties of an ellipse in computer graphics.

Software(s) Required

Python 3 ,matplotlib

Theory

In computer graphics, an **ellipse** is a closed curve that represents the set of points for which the sum of distances from two fixed points (foci) is constant. Drawing an ellipse on a raster display requires selecting pixel points that best approximate the mathematical shape.

The **Mid-Point Ellipse Drawing Algorithm** is an efficient rasterization algorithm used to draw an ellipse using **incremental calculations**. It is an extension of the Mid-Point Circle Algorithm and avoids floating-point arithmetic, making it faster and suitable for real-time graphics.

An ellipse centered at (xc, yc) with semi-major axis r_x and semi-minor axis r_y is defined by the equation:

$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1$$

The algorithm divides the first quadrant of the ellipse into **two regions** based on slope:

Region 1

- Slope magnitude is **less than 1**
- Movement is mainly in the **x-direction**
- Decision parameter determines whether the next pixel is chosen horizontally or diagonally

Decision parameter for Region 1:

$$p_1 = r_y^2 - r_x^2 r_y + 14r_x^2$$

Region 2

- Slope magnitude is **greater than 1**
- Movement is mainly in the **y-direction**
- Decision parameter determines whether the next pixel is chosen vertically or diagonally

Decision parameter for Region 2:

$$p_2 = r_y^2(x+0.5)^2 + r_x^2(y-1)^2 - r_x^2r_y^2$$

The Mid-Point Ellipse Algorithm is widely used because it:

- Uses only integer arithmetic
- Is fast and efficient
- Produces smooth and accurate ellipses

Algorithm: Mid-Point Ellipse Drawing Algorithm

Step 1: Start

Begin the program.

Step 2: Input

Read the following values:

- Center of the ellipse (x_c, y_c)
- Semi-major axis r_x
- Semi-minor axis r_y

Step 3: Initialize

- Set initial point:

$$x = 0, y = r_y$$

- Compute the initial decision parameter for **Region 1**:

$$p_1 = r_y^2 - r_x^2r_y + \frac{1}{4}r_x^2$$

Step 4: Plot Initial Points

Plot the four symmetric points of the ellipse using:

$$(x_c \pm x, y_c \pm y)$$

Step 5: Region 1 Processing

Repeat while:

$$2r_y^2x \leq 2r_x^2y$$

- If $p_1 < 0$:
 - Increment $x = x + 1$
 - Update decision parameter:

$$p_1 = p_1 + 2r_y^2x + r_y^2$$

- Else:
 - Increment $x = x + 1$
 - Decrement $y = y - 1$
 - Update decision parameter:

$$p_1 = p_1 + 2r_y^2x - 2r_x^2y + r_y^2$$

- Plot symmetric points for each new (x, y)

Step 6: Initialize Region 2

Calculate the decision parameter for **Region 2**:

$$p_2 = r_y^2(x+0.5)^2 + r_x^2(y-1)^2 - r_x^2r_y^2$$

Step 7: Region 2 Processing

Repeat while:

$$y > 0$$

- If $p_2 > 0$:
 - Decrement $y = y - 1$

- Update decision parameter:

$$p_2 = p_2 - 2r_x^2y + r_x^2$$

- Else:
 - Increment $x = x+1$
 - Decrement $y = y-1$
 - Update decision parameter:

$$p_2 = p_2 + 2r_y^2x - 2r_x^2y + r_x^2$$

- Plot symmetric points for each new (x, y)

Step 8: Stop

Terminate the program after the ellipse is completely drawn.