Problem Statement

Write a program that uses the Midpoint Ellipse Drawing Algorithm to draw an ellipse in a graphics window. The user should provide the center coordinates and the horizontal and vertical radii (rx and ry) of the ellipse as input. The program must compute all points on the ellipse using the Midpoint Ellipse Drawing Algorithm and render the shape using the graphics library. The implementation should utilize symmetry and decision parameters to efficiently plot the ellipse with pixel precision.

Background Theory

The Midpoint Ellipse Drawing Algorithm is a foundational technique in computer graphics used to draw ellipses accurately and efficiently using integer arithmetic. Unlike traditional methods that rely on complex mathematical functions and floating-point operations, this algorithm uses a decision parameter to determine the optimal pixel positions, thereby improving performance and precision.

An ellipse differs from a circle in that it has two radii: a horizontal radius (rx) and a vertical radius (ry). The ellipse is divided into two regions based on the slope of the curve. Region 1 handles the portion where the slope is less than 1, and Region 2 handles the portion where the slope is greater than 1. In each region, a different decision parameter is used to determine whether to move horizontally, vertically, or diagonally to the next pixel.

The algorithm takes advantage of the symmetry of ellipses, similar to circles. By calculating points in the first quadrant and reflecting them across the x- and y-axes, the complete ellipse can be drawn with minimal computation.

Algorithm

Step 1: Start

Step 2: Declare variables:

1. xc, yc as Center of ellipse
2. rx, ry as Semi-major and semi-minor radii
3. x, y as Current pixel position
4. p1, p2 as Decision parameters for region 1 and region 2

Step 3: Read values of xc, yc, rx, ry

Step 4: Initialize starting point on the ellipse centered at origin:

x = 0

y = ry

Step 5: Calculate the initial decision parameter for Region 1:

p1 = ry² - rx²·ry + (1/4)·rx²

Step 6: For Region 1 (while 2·ry²·x < 2·rx²·y):

Repeat:

If p1 < 0:

x = x + 1

p1 = p1 + 2·ry²·x + ry²

Else:

x = x + 1

y = y - 1

p1 = p1 + 2·ry²·x - 2·rx²·y + ry²

Plot symmetric points using 4-way symmetry

Until 2·ry²·x ≥ 2·rx²·y

Step 7: Calculate the initial decision parameter for Region 2:

p2 = ry²·(x + 0.5)² + rx²·(y - 1)² - rx²·ry²

Step 8: For Region 2 (while y ≥ 0):

Repeat:

If p2 > 0:

y = y - 1

p2 = p2 - 2·rx²·y + rx²

Else:

x = x + 1

y = y - 1

p2 = p2 + 2·ry²·x - 2·rx²·y + rx²

Plot symmetric points using 4-way symmetry

Step 9: For each (x, y), determine 4 symmetric points:

(xc + x, yc + y)

(xc - x, yc + y)

(xc + x, yc - y)

(xc - x, yc - y)

Step 10: Plot these pixel positions using `putpixel()`

Step 11: Repeat until Region 2 is complete (y < 0)

Step 12: Stop