COSC 4370 - Homework 1

Name: Gokce Yilmaz

PSID: 2168280

September 2024

1 - Problem

The primary goal of this assignment was to implement an algorithm for rasterizing arcs of an ellipse. Specifically, the ellipse is defined by the equation $(x/9)$ ^2 + $(y/16)$ ^2 = 24^2, where y <= 0. The challenge was to effectively render this half-ellipse onto a canvas, considering the given equation and constraints. The chosen dimensions for the image were based on a radius of 24*24, resulting in an 800 × 800- pixel canvas.

2 - Method

To deal with this problem, I concentrated on two functions: writeEllipse and midpointEllipse. The former is plotting the pixels, responsible for the drawing of the lower half of the ellipse - symmetrical across quadrants; the latter implements the midpoint algorithm for an ellipse, customized to just draw the lower half of the ellipse, that is, y ≤ 0.

Instead of recalculating every point from the beginning, the algorithm builds on the previous steps by making small adjustments, which makes it faster and more efficient, especially for real-time drawing. It only plots points for the lower half of the ellipse, and to make sure the shape is symmetrical, it mirrors these points across the x-axis. The writeEllipse function handles this by reflecting the pixels into the third and fourth quadrants, so the lower half of the ellipse is drawn correctly on both sides.

One of the most well-known methods for drawing shapes on a grid is based on finding the next pixel step-by-step using decision parameters. This approach balances the accuracy of more precise floating-point math with the faster speed of using simpler, whole-number calculations. The ellipse is split into two parts: Region 1, which handles the flatter, more horizontal section, and Region 2, which deals with the steeper, more vertical part. Each region is handled separately to keep the drawing smooth and accurate.

3 – Implementation

The implementation strategy revolved around carefully iterating over the pixel coordinates and determining whether each point fell within the lower arcs of the ellipse. The method employed the ellipse equation:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

where a and b represent the semi-major and semi-minor axes, respectively. For each pixel, we calculated the corresponding decision parameters (d1 for region 1 and d2 for region 2), updating the x and y coordinates incrementally.

3.1 `midpointEllipse`

This function played a significant role in the assignment, employing the midpoint ellipse drawing algorithm. The algorithm systematically iterates through coordinates, considering the ellipse equation, and renders pixels where necessary. The function begins at the top of the ellipse (x = 0, y = b) and initializes the decision variables, which help decide whether to move horizontally or diagonally to plot the next point.

Region 1: The algorithm begins by plotting points where the slope is less than 1, so it moves more sideways (horizontally) than downward (vertically). It uses a decision value (d1) to decide if the next pixel should move directly to the right (x++) or down and to the right (x++, y--). The algorithm keeps checking and updating d1 to make this choice as it continues plotting points.

When the slope gets steeper and the vertical movement becomes more important, the algorithm moves into Region 2. Here, it uses a second decision value (d2) to decide whether to move straight down (y--) or diagonally down and to the right (x++, y--). It keeps adjusting this decision until the entire lower half of the ellipse is drawn.

4 – Results

The program generated a .bmp file as its output, containing the image's dimensions, pixel data, and color information. Upon viewing the file with standard image viewers, the image confirmed the successful execution of the rasterization algorithm.

Source: https://www.geeksforgeeks.org/midpoint-ellipse-drawing-algorithm/