COSC 4370 - Homework 1

Name: Jacob Rangel

PSID: 1997560

February 2024

1 - Problem

The primary goal of this assignment was to implement an algorithm for rasterizing arcs of an ellipse and a circle. Specifically, the ellipse is defined by the equation (x/16) ^2 + (y/8) ^2 = 45^2, and the circle is defined as x^2 + y^2 = R^2, where x > 0 and R = 100. The challenge was to effectively render these arcs onto a canvas, considering the given equations and constraints. The chosen dimensions for the image were based on a radius of 200, resulting in a 400 × 400-pixel canvas.

2 - Method

In addressing this problem, I made crucial modifications to two core functions: **writeEllipse** and **midpointEllipse**. The former is responsible for setting the color of pixels within the ellipse, ensuring symmetry across different quadrants. The latter implements the classic midpoint ellipse drawing algorithm, a key element in accurately rendering the ellipse. Additionally, I introduced a new function, **midpointCircleArc**, to handle the rendering of pixels for a circle arc.

To enhance the performance of the midpoint circle algorithm, I employed a later version of the incremental computation technique more extensively. Recognizing that the Ll functions are linear equations, and by directly computing them, I realized that any polynomial can be computed incrementally. This aligns with the approach we took with decision variables for both the line and the circle. In effect, we are calculating first- and second-order partial differences, a technique encountered again in Chapters II and 19. The strategy involves evaluating the function directly at two adjacent points, calculating the difference (which, for polynomials, is always a polynomial of lower degree), and applying that difference in each iteration.

This technique is later developed into a revised algorithm, where pixel selection is based on the sign of the variable 'd' computed during the previous iteration. The decision variable 'd' is then updated with either 'dE' or 'Ll8E,' using the value of the corresponding 'Ll' computed during the previous iteration. Furthermore, the 'Lls' are updated to consider the move to the new pixel,

utilizing the constant differences computed previously. The procedure incorporates the initialization of 'dE' and 'dsE' using the start pixel (0, R), and the revised algorithm is outlined in Fig. 3.18. This is what I used for developing the **midPointCircleArc** Function.

3 - Implementation

The implementation strategy involved a meticulous analysis of each coordinate within a 2D array. The goal was to determine whether a given point lies within the arcs of the ellipse and circle. A nested loop structure was employed to iterate through the coordinates. Pythagorean square calculations were crucial in assessing pixel placement accurately. To mitigate imprecision arising from integer multiplication, an error threshold was incorporated into the algorithm. Adjustments were made to the placement of arcs to ensure they fit seamlessly within the designated image boundaries.

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. A key insight was recognizing that a circle has eight symmetric counterparts for any point on its arc, which simplified the rendering process.

3.2 `midpointCircleArc`

This new addition, `midpointCircleArc`, tackled the rendering of pixels for a circle arc. The midpoint circle drawing algorithm was adapted to handle the unique characteristics of the circle defined by x^2 + y^2 = 100^2. It contributed to the overall visual representation of the assignment, providing a distinct element to the final image.

4 - Results

The program's output materialized in a .bmp file, encapsulating two distinctive semicircles. The image dimensions, pixel values, and color information were embedded in the file. A visual inspection of the image using standard image viewing tools revealed the successful implementation of the rasterization algorithm.