COSC 4370 - Assignment 1

Name: Nimet Ozkan | ID: 1961233

February 2022

1 Objective

Assignment requires the implementation of an algorithm to rasterize the ellipse. The ellipse is defined as $(x/12)$^2+$(y/6)$^2=64^2 where y>=0. This equation will be converted to the standard elliptical equation form (xh)^2/a^2 + (yk)^2/b^2 = 1. Here (h, k) is the center, a and b being the center. the lengths of the semi-major and semi-minor axes. Since the x-intercept will be 786 and the y-intercept will be 384, the x-radius will be 786 and the y-radius will be 384.

2 Method

There are two extra functions that need to be added in the code provided: *DrawPoints* and *DrawEllipse*. No need to modify any other part of the existing code. DrawPoints takes a BMP shape object that references the original BMP shape, the radius center values of x and y, in a long/float data type format. Using them within the BMP::set_pixel function, which requires the x and y coordinates of the pixel point and the pixel values of the color. By the functionality of this function, the program can draw points in left, right, up, and down directions. DrawEllipse also takes a BMP shape object that references the original BMP shape, the radius center values of x and y in the form of a long/floating data type. With this function we will be able to work with horizontal and vertical ellipses in the coordinate plane, we consider two cases: those centered at the origin and those centered at a point other than the origin. To achieve this, we will apply the midpoint ellipse algorithm and in the DrawEllipse function we will use DrawPoints to divide the points on the first quadrant of the ellipse into two regions. Then, each point (x, y) will be reflected to the other three quadrants (-x, y), (x, -y), (-x, -y) i.e., 4-way symmetry will be used.

3 Implementation

Given the input radius, x == 786 and y == 384, we were able to obtain the center of the ellipse. If we want to make an ellipse within the window size, we must determine what the window's maximum limit should be. Therefore, in terms of x-radius, we can easily see that the window must be at least x-radius*2. If we round from 786 to 800, we can easily say that the width of 600*2 = 1600 is sufficient for this ellipse. We needed to divide this window width and height in half to get the central points. However, this ellipse must appear above 0 because y >= 1. That's why we added the half height of the ellipse to the midpoints and the central points. So we obtained x_central = 896, y_central = 1400 in a window with width = 1792 and height = 1792.

3.1 DrawPoints

The given object was used to draw points based on the 4-way symmetry of the center of the BMS. The white colored dots at (x-radius,y-radius) position, with the given pixel values of 255, 255, 255, 0, were added together with x_rad+x_central, y_rad+y_central and moved around the center in this way. Uses negative value for x to move left, positive x for right side, negative y for down and positive y for up direction.

3.2 DrawEllipse

This function contains derivatives and finds endpoints. We used the square of the y-radius with the x-radius to find the end/stop value of the x-radius. We also used x-radius squared times y-radius to find the end/stop of the y-radius. We checked the first quarter by checking these stop values of the x and y radius in a while loop. The first quadrant is designated as a variable quad, and the second quadrant as a quad. The first quad is determined by1 pow(rad_y,2) - (pow(rad_x,2)*rad_y) + (0.25 * pow(rad_x,2)), this quad represents our midpoint. We checked whether this midpoint is inside the circumference or on the circumference (quad1 < 0), or whether it is outside the circumference. We created the point and its reflection by calling the DrawPoints function to draw our ellipse in both quadrants.

4 Results

The output of the program was a .ppm file. An ellipse is easily visible when viewed through an image viewer. The white ellipse is rasterized in a window without going out of frame.