COSC 4370 - Homework 1

Name: Juan Yanez PSID: 2029097

September 2022

1 Problem and Objective

This assignment wanted the rasterization of an ellipse. The ellipse will be defined as $(x/12)^2 + (y/6)^2 = 64^2$ where x >= 0. The x radius for this ellipse is 768 and the y radius is 384. The output image dimensions to fit the ellipse properly will be (800,800), and with that, the center of the image will be (400,400).

2 Method

To properly display the ellipse where x >= 0, only the right side of the ellipse must be created. In a theoretical x and y axis, the true center (0,0) will be (0,400). The x = 0 so that the ellipse is only x >= 0. The y axis midpoint is 400 which centers the ellipse vertically so that no points are positioned outside of the image. The ellipse was implemented with the midpoint ellipse algorithm, which was implemented for only the positive x pixels. This method was referenced from geeksforgeeks.

3 Implementation

All the implementation was done in main using functions from BMP.h. Given the equation for the ellipse, the x and y points were set to 0 and 384 which is the y radius. The start point of both pixel axes allowed the ellipse to stay within the edges of the image. Xx and Xy are variables to assist with the algorithm, which makes it easier to work with, they are initialized with 2 and x $radius^2$ * 768 respectively. The radii will be 768 and 384. By obtaining the center of the ellipse, and the other variables, the midpoint algorithm could be implemented.

3.1 Region 1

The first while loop creates the first region of the ellipse, while the second for loop creates the second region, where x = 0 and y = 384 initially. The initial decision parameter is q1 = y $radius^2$ + ¼ x $radius^2$ - x $radius^2$ * y radius. Before incrementing the region's parameters, the pixels are placed according to x and y + 400 with set_pixel from BMP.h. The addition of 400 to the y is due to the offset of the axis for the image, since the true center is not (0,0).Then for every position of

x after placing the pixel, if q1 is less than 0, then x is increased by one and Xx will increment by 2 * y radius^2 while q1 would be increased by Xx + y radius^2. If q1 is not less than 0, then x will be incremented by one and y will be reduced by one. Xx will increase by 2 * y radius^2, Xy will decrease by 2 * x radius^2, and q1 will increase by Xx - Xy + y radius^2.

3.2 Region 2

Region 2 was q2 which is y radius^2 * $(x_0 + ½)^2$ + x radius^2 * $(y_0 - 1)^2$ - x radius^2 * y radius^2. The second region follows a slightly different incrementation according to the algorithm, the placement of pixels was done with the same set_pixel function from BMP.h. For the increments, if q2 is more than 0, y will be decreased by one. Xy will also be decreased by 2 * x radius^2 and q2 will be incremented by x radius^2 - Xy. If q2 is not more than 0, then y will be decreased and x will be increased, both by one. Xx will be incremented by y radius^2 * 2, Xy will be decreased by x radius^2 * 2, and q2 will be incremented by Xx - Xy + x radius^2.

Once all pixels were placed in their proper positions through the midpoint algorithm, it was written to output.bmp in an image form.

4 Results

The output is output.bmp with the size of 800x800 which displays an ellipse where only x >=0, meaning that only the right half of the ellipse will be created, with pixels being white.