Sham Hintolay
1618608

Homework 1 Report

1. Problem

In this algorithm, we are rasterizing an eclipse with when y >= 0 using the following formula:

$$(\frac{x}{6})^2 + (\frac{y}{12})^2 = 64^2$$

The provided reading, textbook, and GeeksForGeeks page on midpoint algorithm were helpful for the assignment.

2. Method

In order to succeed in the assignment, we only made changes in the main.cpp file. We needed to create a ellipse-creating function that is called from the main, which takes in the BMP object initialized in main(). The mid point algorithm was essential to creating the half-ellipse. It works by splitting the first quadrant into 2 regions, and fills the ellipse points in each region separately. With the points in the first quadrant decided, we can just mirror those points along the x and y axis to get the points for the rest of the quadrants. Since we are looking for a ellipse when y >= 0, we only flip points along the y axis. Using the x and y we get we plug them into the set_pixel() method for the BMP object. In order to center the ellipse, we also need to translate the points with the same fixed number for all (width/2 and height/2 make sense). After this the changes to the BMP object are made, and within the main function, we can call the write method to output our ellipse.

3. Implementation Details

$$\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1$$

This is the equation of an ellipse in standard form, in order to get a and b, we have to multipy them by 64, which are then the values we use to plug into the ellipse function. Using these values we split the quadrant into 2 regions, based on where dy/dx = -1, and a prediction function (d1) is used to determine when our first region is filled and when to split and move to region 2. With each iteration of the loop we increment x and decrement y to move “down the ellipse” and d1 constantly checks to see if we should still keep on “setting pixels”. We follow the same logic for the loop filling region 2, however we can just set d2 as 0 because we are already on the other side of region 1 and don’t have to worry about hitting another region, since we just need to keep going down and forming the ellipse until y >= 0 is no longer true.

4. Results

The results were a bmp file named “output.bmp”, which produced a centered, white half ellipse.