COSC 4370 - Homework 1

Name: LUJIA WU PSID:1854803

February 2021

1 Problem

In this assignment we require to implement an algorithm for rasterizing eclipse using c++ programing language. The eclipse which we require to rasterize is:

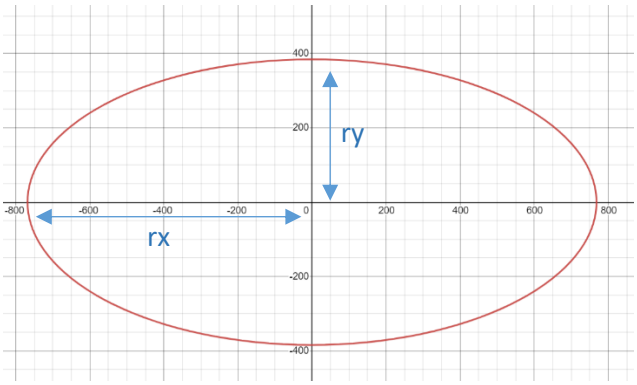
$$\left(\frac{x}{12}\right)^2 + \left(\frac{y}{6}\right)^2 = 64^2 \text{ that } y >= 0 .$$

2 Method

In order to rasterize the function above we require to make it more general and simple. So we can do:

$$\left(\frac{x}{12}\right)^2 + \left(\frac{y}{6}\right)^2 = 64^2 \quad => \div 64^2 => \quad \frac{\left(\frac{x}{12}\right)^2 + \left(\frac{y}{6}\right)^2}{64^2} = \frac{64^2}{64^2} \quad => \quad \left(\frac{x}{768}\right)^2 + \left(\frac{y}{384}\right)^2 = 1$$

In fact, this is an ellipse function and the tuple (x,y) is the perimeter of that ellipse. Based on the formula our ellipse radius toward x is $rx = 768$ unit and toward y is $ry = 384$ unit. Also, the center of ellipse is $c = (0,0)$.



Here, we also have a constraint of $y >= 0$ which shows that we are interested in the upper side of the ellipse to make it as an eclipse. So, based on the midpoint circle implementation we had so far in the attached documents, our shape was consists of 8 sections called octets. If we can draw the first four octets this will lead us to the shape of eclipse.

3 Implementation

In our application we have considered a 2000x2000 pixel bmp image using *BMP struct* from the *BMP.h* header file and we fill the surface with black RGB values as zero for each color-channel. Next we call our only function to return the BMP variable. The function we have implemented is:

```
BMP rasterizeEclipse(BMP BMPImage, int xc, int yc, int rx, int ry, int r, int g, int b);
```

It receives BMP variable, coordination of center of the eclipse as (xc,yc), radiuses toward x-y axis as (rx,ry) and RGB color values as (r,g,b).

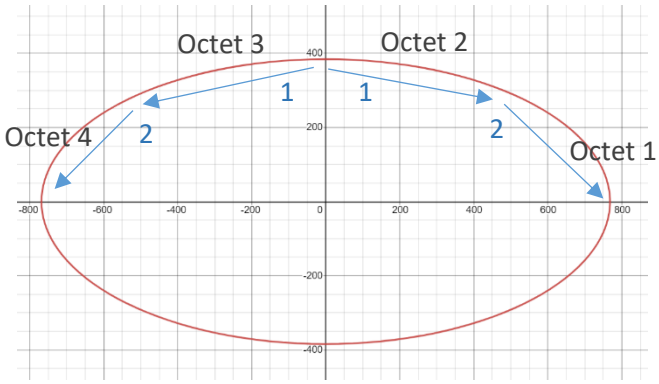
When the center of the ellipse is (0,0) we can write the function of ellipse as below:

$$\frac{(x-x_c)^2}{r_x^2}+\frac{(y-y_c)^2}{r_y^2}=1 \quad => \quad p(x,y)=r_y^2x^2+r_x^2y^2-r_x^2r_y^2$$

And $\frac{dy}{dx}=-\frac{2r_y^2x}{2r_x^2y}=0$ => which shows the boundary of 2nd octet if we start from the point $(0,r_y)$. In fact, toward the path of 2nd octet to 1st octet we need to check that if we have reached the boundary or not. In other words to check for $2r_y^2x < 2r_x^2y$ which we have considered in our while conditions.

Each time in our while loop we check the boundary and increment toward x and decrease the y. Also we set the pixel for (x,y) and update the new p value. Based on the formula above. Simultaneously, we do this for the 3rd octet since it does the same but in reverse order. In other words, the same steps are considered for x and y but the only difference is with the direction of the x that we have considered it as –x to get the same result.

In the next part the similar steps are done to till our y variable reaches 0 and x reaches its highest amount in both directions. Below, we have depicted the direction of setting pixels.



Also, for the color of the line we set the RGB values as (100, 250, 50) which are used directly for setting pixels in our function.

Finally, we return the output BMP image variable to the caller function in order to be saved in the local-memory.

4 Results

The output of the program is a bmp-image which can be downloaded and viewed in a simple image-viewer applications. Based on the formula, the image dimension is required to be 2000x2000 pixels. Below is the output of the application.

As a conclusion the zip-file related to the assignment #1 consists of a *main.cpp* file, *BMP.h* header file, *report.pdf* file and *output.bmp* image.