COSC 4370: Interactive Computer Graphics

Instructor: Prof. Zhigang Deng

Name: NHAT NGUYEN

UID: 1652765

HW1 REPORT

Assignment problem:

Rasterize 2 half circles. One with radius 100 where x >= 0 and one with radius 150 where y >= 0.

Sub-problems:

- Get OpenGL to work.
- Read, understand, and figure out how to run the given code and get the output.
- Figure out what algorithm to use to render the circles.
- Understand the algorithm by reading the given paper.
- Implement the algorithm to get the correct output.

First thought:

The principle Rasterization, summarized, is to get a pixel (or pixels) to display as closely as possible to where we wanted. To achieve visually satisfactory, this process must execute as rapidly as possible, therefore, we need to come up with an efficient algorithm to calculate the coordinate of the point and display it.

Process:

At first, an attempt to calculate y value as I increment x, using the circle equation. But that would require using type double and doing multiple multiplies and divinations repeatedly. This would prove to be very inefficient.

After a deeper look into the reading, the "Midpoint circle scan conversion algorithm using second-order differences" was used and proven to be much faster than the previous method. This algorithm taken advantage of how incrementation of second-order differences of x and y affect x and y values, and only use increment and deduction in its calculations, result in a much faster execution time.

Using the chosen algorithm, the program generated an eight of a circle. This was also for efficiency sake, because the remaining part can be generated through mirroring the generated image.

Some testing was done to ensure it rendered the image in the right quadrant.

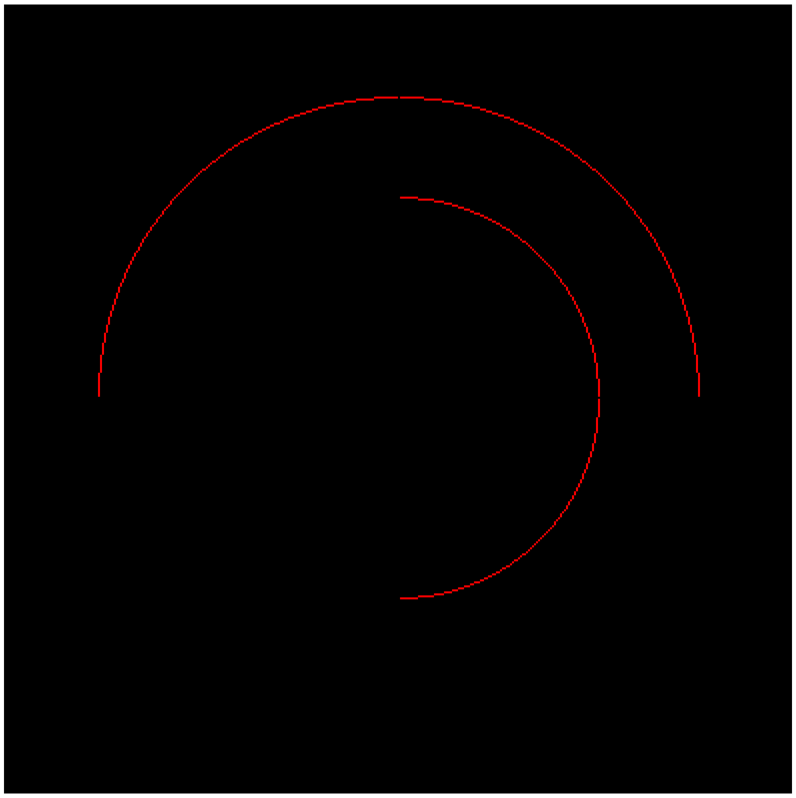
The initial thought was to use the renderPixel() function to generate the remaining part of the circle. After some consideration, it does not seem like the right approach, so the operations was moved to rasterizeArc() instead.

Finalize and generate the image.

Method using:

"Midpoint circle scan conversion algorithm using second-order differences" because of its efficiency.

Result:



Conclusion:

Through the reading and lecture videos, I was able to use the given algorithm and provided codes to generate the wanted image. It let me review some of my memory on second-order differences and C++ coding, also reinforce my understanding on rasterization.