Name: Nicholas Ung
NUID: 002336960

Lab 3 Report: Graphics Primitives

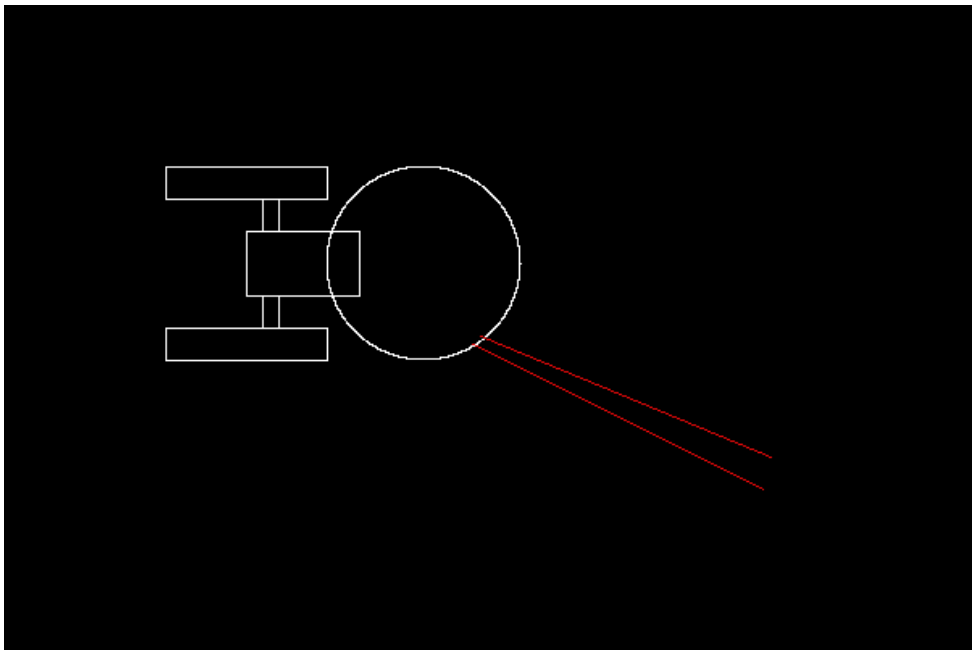
Summary

The purpose of this project was to implement five graphics primitives - **Point**, **Line**, **Circle**, **Ellipse**, and **Polylin**e - using specified algorithms. The implementations focused on drawing geometric figures accurately on a computer screen, handling coordinate issues and algorithmic efficiency. The project encompassed building and testing each primitive, optimizing performance, and creating visual test cases to demonstrate their functionality.

Required Images

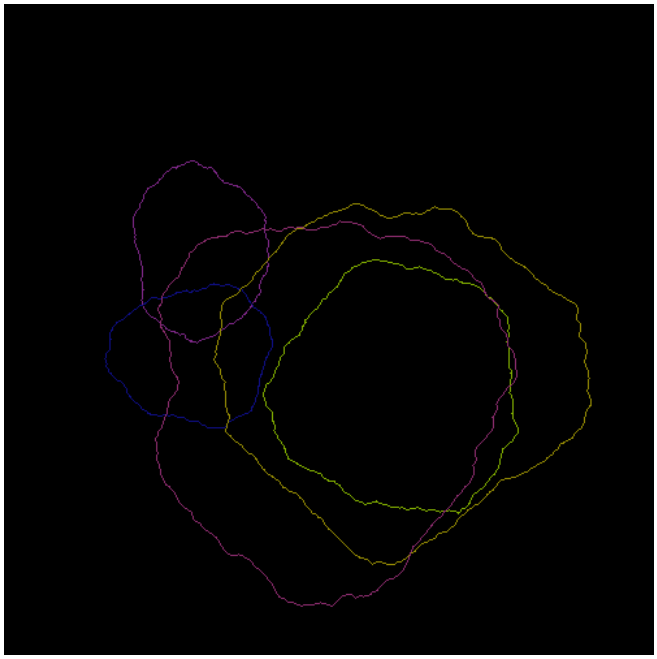
1. ***Test Image A***

   Description: I generated this crude rendering of what appears to be the USS Enterprise using ***test3a.c***. The proper image demonstrates that my line & circle drawing methods work properly and the lasers show that I implemented the color feature correctly.



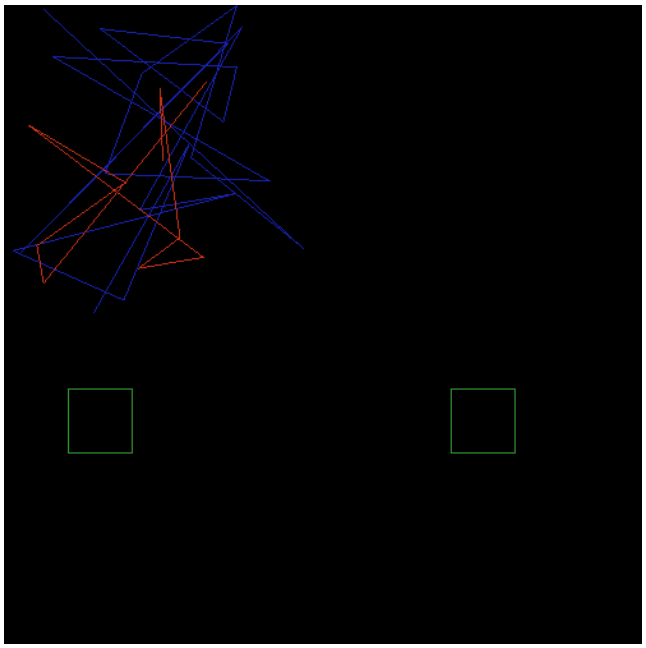
2. *Test Image B*

Description: This picture was produced with ***test3b.c*** and matches up with the sample image. It has multiple polylines produced by recursive subdivision of lines as expected.



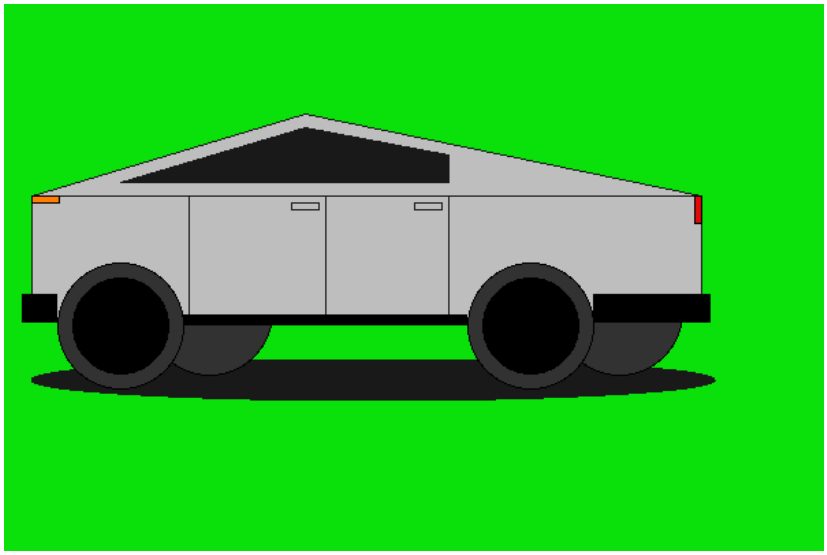
3. *Test Image C*

Description: The third image was generated with ***test3c.c*** and shows a series of red and blue polylines in the top-right corner as well as two green squares, drawn clockwise and counter-clockwise with polylines. It matches up with expectations and seems to show that graphics methods are working as intended.



4. ***Creative Image (3-D Shape)***

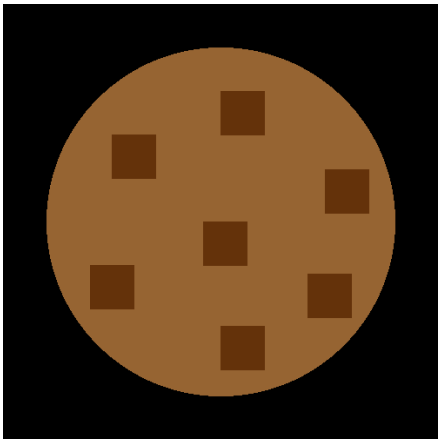
   Description: The final task was to create a 3-D looking image with all the features implemented in this lab so I chose to produce a mock-up of a Tesla Cybertruck. I made the body out of lines, the wheels out of circles, and the shadow out of an ellipse. In making this, I realized the order in which assets are built in the code can help influence perspective and I was also able to use my flood-fill algorithm to color in the shapes.



Extensions

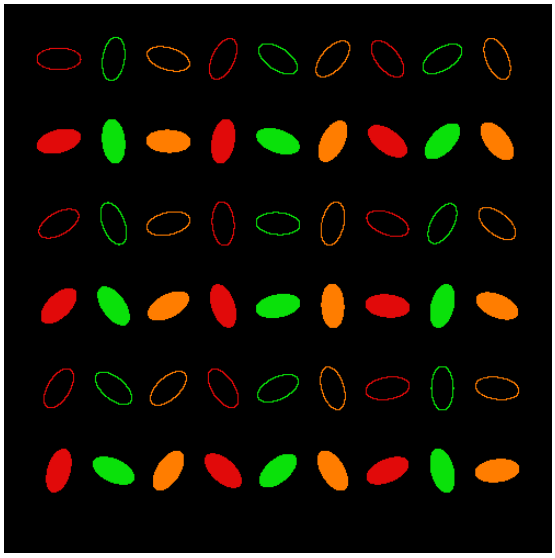
1. ***Flood-Fill Algorithm***

   Description: I developed a 4-way flood fill algorithm to enable the filling of polygons and circles. My method involved starting at a seed coordinate pair and having that point color neighboring pixels until it reached already colored pixels or the border of the shape. I created a cookie using my ***circle_drawFill*** method, then created the chips out of boxes made with polyline and filled with my ***floodFill*** method.



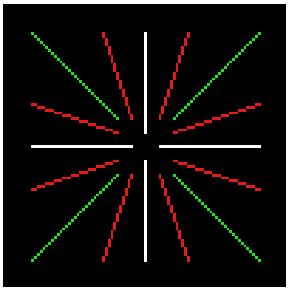
2. ***Arbitrary Ellipse Orientations***

   Description: My second extension included adding functionality to display ellipses in different orientations. I simply added a parameter in the ***ellipse_set*** method to take the angle at which the ellipse is oriented. I used a for loop to create a nice pattern of filled and unfilled multicolored ellipses at different angles.



Bresenham's Line Algorithm

My implementation of the line algorithm resulted in metrics of: **Average line length = 103.9; 4,681,647.60 lines per second**. I am using an Apple MacBook Pro with 8 GB memory and an M2 Chip. I implemented the pattern as we discussed in class and it seemed to work well in producing the desired images and functionality although my results are slower than Professor Maxwell's reported numbers.



Name: Nicholas Ung
NUID: 002336960

Reflection

This project deepened my understanding of computer graphics fundamentals, particularly how geometric primitives are implemented and optimized for performance. Handling screen coordinate discrepancies and optimizing algorithms for better performance were key objectives that I accomplished.

Acknowledgements

I would like to acknowledge online resources such as W3schools for providing useful material for improving my C programming. The code samples from the Hearn and Baker textbook website helped serve as a reference for algorithm implementations for the circle and ellipse methods. Aside from those, I primarily consulted the lecture notes and recordings from Professor Maxwell to understand the techniques for solving the problems in this assignment.