Creation of Graphic Object Project

Project Summary

The project involved developing a software rasterizer. The software rasterizer was written in C++. The project focused on rasterization as a method for generating computer graphics, which involved decomposing 3D geometric shapes into a 2D array of pixels/dots. The alternative approach of ray casting/tracing, which involves tracing a path of light from the viewer and determining its interaction with 3D geometric shapes, was also discussed.

This project has multiple functionalities. The drawTriangle3D_Barycentric function takes a 3D triangle T and rasterizes it on the 2D screen. First, it checks if the triangle should be drawn, based on the boolean value of the shouldDraw property of the triangle. If it should not be drawn, the function returns without doing anything. If it should be drawn, the function calculates the bounding box of the triangle by finding the minimum and maximum x and y values among its three vertices. It then loops through all pixels within that bounding box and calculates the barycentric coordinates of the pixel with respect to the triangle. If the barycentric coordinates lie within the triangle, the function checks if the pixel is closer to the camera than any previously drawn pixel at that location. If it is, the function sets the color and depth values of the pixel to those of the triangle interpolated according to the barycentric coordinates. The drawModel function takes a Model object and loops through all of its triangles, calling the drawTriangle3D_Barycentric function on each one to rasterize it on the screen.

This project also consists of other functions such as the void Model::homogenize(), that takes the vertices of each triangle in the triangles vector of a Model object and converts them from homogeneous coordinates to Cartesian coordinates. This is done by dividing each vertex's x, y, and z values by its w value and setting w to 1. Along with that, it also has functions like the void Model::performBackfaceCulling, which performs backface culling on each triangle in the triangles vector of a Model object. It does this by calculating the surface normal of each triangle, determining if it is facing the viewer or not, and setting the shouldDraw flag of the triangle accordingly. The eye and spot vectors represent the viewer's position and the direction they are facing, respectively. If a triangle is facing the viewer, its color is set to blue in this implementation.

This project helped me gain a better understanding of how to create graphical objects. The use of matrices and performing mathematical operations on it using algorithms was a new addition to my arsenal of coding languages which was vital in furthering my understanding of how Mathematics and Computer Science can be applied together.