4-2 Milestone Three: Algorithms and Data Structure

Maria Herring

CS 499 Computer Science Capstone

Instructor: Brooke Goggin

February 1, 2024

Revised Submission February 3, 2024

4-2 Milestone Three: Algorithms and Data Structure

The artifact I chose to enhance in the Algorithms and Data Structures category is an OpenGL 3D project. OpenGL is an application programming interface (API) that provides a large set of functions for rendering graphics. The artifact was developed in the C++ programming language. Visual Studio Integrated Development Environment (IDE) was the tool used to develop the application. The scene was originally developed as a final project in CS-330 (Computer Graphics and Visualization), which was completed in Winter 2023.

Initially, the development of this artifact presented many challenges. As I completed the project and rendered objects, I knew that I needed to expand my knowledge of OpenGL. I chose to enhance this artifact to demonstrate my greater understanding of how to implement algorithms and data structures in conjunction with OpenGL functions. Specifically, I will present skills that implement code to improve the structure and design of the project. I will demonstrate skills that implement the use of algorithmic functions to compute a surface normal. I will demonstrate my understanding of the OpenGL API and my understanding of the C++ programming language.

Previously, all the programming logic for the program was contained in main.cpp. The shader functions were refactored, and a separate shader class was created, and header files were created for the project to provide more segmentation and readability. The objects scale, rotation, and translation were corrected to reflect the appropriate values. I removed the use of the standard namespace from the application, opting to place std:: in front of functions from the standard library. While this may not be critical in this project, I used best practices to make it clear which namespace a function belongs to, to avoid collisions in projects that use multiple namespaces. An interleaved structure format was used to process the vertex data. The vertices, normals, and texture coordinate attributes were processed in a single buffer object instead of using multiple buffers. In this format, the graphics processing unit (GPU) processes this data from one location, improving performance, but also potentially using more memory.

Code for a surface normal calculation algorithm function has been added to the application. This function calculates the surface normal of a triangle. Calculating a surface normal allows us to convert topology problems into simpler linear programming. Surface normals are important properties of a surface for applying the correct light sources that generate shading and other visual effects. The logic of this function computes the cross product or vector product of two edges of the triangle is calculated. After the cross product is computed the normal is calculated, the normal is calculated by normalizing the unit length. The use of an algorithm improves the efficiency and accuracy of the surface normal calculations for the points on the surface.

Figure #1: Surface normal algorithm

A screenshot of a computer program

Description automatically generated

This function is called and applied to normalize and render the cylinders, with optimal lighting. In the fragment shader, normalization functions are used in a similar way to calculate the lighting positions. The fast and accurate computation of the surface normal from a point is also important to improve object recognition. The edge vector calculations, cross product, and normalization functions consist of constant-time operations, so the total time complexity of the function can be measured as O(1). The implementation of this function provides a computing solution that solves a given problem using algorithmic principles.

The enhancements made to this application allowed the objects to render correctly in configuration and scale. The structure and design of the application was improved by implementing additional classes and header files and by implementing standard coding principles. The algorithmic logic of the application was improved by applying functions to compute data, allowing the application to render properly.

This aligns with course outcomes that demonstrate the ability to use well-founded and innovative techniques skills, and tools in computing practices for the purposes of implementing computer solutions that deliver value and accomplish industry specific goals. It also aligns with outcomes which design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices.

Figure #2: before enhancements Figure #3: after enhancements

A group of objects on a table

Description automatically generated A group of objects on a table

Description automatically generated

When I completed CS-330 in the winter of 2023, I knew I still had a significant learning curve in properly using OpenGL to render graphics. I reviewed the course documentation and some additional tutorials on OpenGL. The knowledge gained from these additional resources allowed me to refactor my original project to properly render the graphics. Setting up the environment for the Visual Studio IDE was challenging. First, the project libraries for this program were downloaded to a separate folder and then linked into the project. I included those library folders containing all the include and library files in the project.

Linker errors were another challenge. Inputting the appropriate dependencies helped to resolve these errors. Compiling the shader program was another challenge. I had to rebuild the project to resolve the compile errors and create the link for the shader program to run. Resolving these errors helped me learn more about OpenGL and reminded me to be patient and work through problems methodically.