Nar Bhattarai

CS-499

4-2 Milestone Three

**Artifact Description**

This artifact is a 3D viewer application developed using OpenGL as part of CS-330 (Computational Graphics and Visualization). The initial code was produced in [month/year]. It showcases a scene composed of basic geometric shapes, including a plane, box, cylinder, and cone, each with distinct textures. The application also permits camera movement.

From an algorithms and data structures perspective, this project demonstrates how vector and matrix data structures (via GLM) are used to represent 3D positions, orientations, and transformations. Algorithms for camera control, including input handling, vector normalization, cross products, and matrix multiplication, enable smooth navigation and dynamic scene rendering.

**Reason for Inclusion**

I selected this project because it demonstrates excellent object-oriented design. It successfully differentiates between scene management, view management, shader control, and mesh loading. This project illustrates the utilization of graphics APIs, matrix mathematics (GLM), and resource management.

In addition, it highlights how fundamental data structures (vectors, matrices, and classes) are combined with algorithms (linear algebra operations, camera transformations, and input-to-motion mapping) to solve real-time rendering problems in a computationally efficient manner.

**Enhancements Made**

For Milestone Two, I introduced several enhancements:

* **Robust Startup and Cleanup:** Incorporated checks for glfwInit, window creation, and GLEW initialization, ensuring that all allocated objects are appropriately cleaned up.
* **Resource Management Fixes:** Corrected the DestroyGLTextures() function to properly delete GPU textures, which previously had a bug.
* **Prevented Data Growth:** Moved the material setup outside of the rendering loop to stop materials from being allocated every frame.
* **Defensive Camera and Timing:** Added checks in camera callbacks and constrained deltaTime to prevent erratic camera movement during spikes. These improvements enhance the underlying camera algorithms, ensuring reliable translation from input to motion.
* **Improved Logging and Maintainability:** Included logs for texture loading and added comments to clarify the purpose of various functions.

These enhancements reflect my skills in software design and engineering, with a focus on developing reliable code, effective resource management, and maintaining high standards of code quality. They align with program outcomes, particularly relating to technical communication, designing computing solutions, and utilizing established tools and practices.

**Reflection**

By implementing these enhancements, I gained an understanding of the importance of careful resource management in graphics applications and how even minor errors—such as selecting the wrong GL function—can lead to challenging, hard-to-trace problems. I also expanded my knowledge of how algorithms (like adjusting orientation through yaw/pitch changes) and data structures (including vectors and matrices) are fundamental to real-time rendering systems. I encountered difficulties in making sure that my changes did not disrupt compatibility with other support classes (like ShaderManager and ShapeMeshes). To address this, I applied targeted fixes while maintaining the integrity of the overall system to ensure build compatibility. This experience strengthened my capacity to utilize algorithmic problem-solving and organized data management in practical coding situations.