# **Milestone Three Narrative**

**CS 499 – Computer Science Capstone**

### **Artifact Description**

The artifact I selected is my **CS 330 Computational Graphics and Visualization final project**, originally developed in Spring 2025. This C++/OpenGL application renders a 3D desk scene with objects such as a keyboard, mouse, wrist rest, and glasses. The artifact leverages multiple managers (SceneManager, ViewManager) to load textures, handle lighting, and manage the rendering pipeline. While originally intended to demonstrate my knowledge of graphics programming, the project also provided a strong opportunity to enhance its **algorithms and data structure usage**.

### **Justification for Inclusion in ePortfolio**

I chose this artifact because it gave me the chance to demonstrate applied improvements to **data structures and algorithms** in a real-world graphics context. In the original code, the program relied on **linear searches** through fixed-size arrays (m\_textureIDs[16]) to look up textures and slots. This approach had two limitations:

1. **Inefficient lookups** – every search required scanning through multiple elements.
2. **Hard-coded capacity** – texture management was limited to 16 textures.

For my enhancement, I refactored the texture management system to use a **std::unordered\_map<std::string, GLuint>** for O(1) average-time lookups and removed the artificial texture limit. This directly showcases my ability to apply efficient data structures in practice.

### **Course Outcomes Coverage**

In Module One, I planned to demonstrate progress toward the outcome:

**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.**

I successfully achieved this outcome by applying algorithmic reasoning to reduce lookup time and scalability constraints in the rendering pipeline. The trade-off was slightly higher memory overhead due to hash tables, but this was justified by the performance and flexibility gains.

I also continued progress toward:

* **Using innovative techniques and tools** – by applying modern C++ STL containers (unordered\_map) in place of manual arrays.
* **Professional communication** – through documentation in comments explaining why the change improves both complexity and maintainability.

No updates to my outcome-coverage plan were necessary, as this enhancement aligned well with my original plan.

### **Reflection on the Enhancement Process**

Enhancing the artifact emphasized the importance of **choosing the right data structure for the job**. While the original code worked, it depended on arrays and linear scans which was not sustainable as the project grew. By replacing these with hash maps, I gained hands-on practice in balancing **time complexity, memory trade-offs, and practical software engineering constraints**.

The main challenge was ensuring compatibility with the rest of the rendering pipeline and untouched code. Functions such as CreateGLTexture, FindTextureID, and FindTextureSlot had to be refactored carefully so that the new map-based logic was still integrated seamlessly with the OpenGL bindings. Testing was critical: I validated that textures were still loaded correctly and that performance improved during scene rendering.

Ultimately, this process demonstrated how applying data structures and algorithmic principles can make a meaningful impact on the efficiency and scalability of software, even in domains like computer graphics.