The artifact I enhanced is my 3D Kitchen Scene project from CS 330: Computational Graphics and Visualization. This was my first real attempt at 3D graphics programming, where I created a virtual kitchen with objects like a cutting board, teapot, and fruit bowl using OpenGL. When I initially developed this project, I was focused primarily on getting the graphics to display correctly rather than implementing proper software design principles. The result was functional but limited in its implementation.

The original code used a simple procedural approach with direct function calls for rendering. All objects were hard-coded in the renderScene() method, making it impossible to add new objects without modifying the core rendering code. There was no proper resource management - textures were loaded without any caching mechanism, potentially leading to memory issues with larger scenes. Perhaps most limiting was the flat structure that made it difficult to create complex scenes with related objects.

I chose this artifact for enhancement because it perfectly demonstrates my growth in software design skills. Looking back at this code after completing several more advanced courses in software engineering, I could clearly see opportunities to apply the principles I've learned. The simple implementation provided a clean slate for demonstrating how proper software engineering practices can transform basic functional code into a robust, maintainable system.

The first major enhancement I implemented was redesigning the entire application using object-oriented principles. I created a proper class hierarchy with the SceneObject base class and used polymorphism to allow different types of objects to implement their own rendering behavior. This alone made the code much more maintainable and extensible - adding new object types no longer required modifying existing code, following the Open-Closed Principle. I was particularly proud of how clean the inheritance hierarchy became, with clear separation of responsibilities between different classes.

The second major enhancement was implementing a scene graph system that allows objects to have parent-child relationships with proper transformation inheritance. In the original flat design, positioning objects relative to each other was extremely difficult. With the scene graph, I can now create complex hierarchical structures (like a table with items on top) where moving the parent automatically updates all children. This required a deep understanding of transformation matrices and hierarchical data structures, skills I gained in my data structures and algorithms courses.

Resource management was another key focus of my enhancement. I added a texture caching system that prevents duplicate loading of the same textures, significantly improving memory usage and performance. This system uses a map to store textures by name, allowing them to be referenced throughout the application without redundant loading. I also implemented proper resource cleanup in destructors to prevent memory leaks - something my original code didn't address at all.

One of the more advanced enhancements was integrating SQLite database functionality for saving and loading scenes. This feature allows users to save their work and return to it later, something the original application couldn't do. Implementing this required designing a database schema that could represent the complex scene graph structure and writing code to serialize and deserialize the scene objects. This enhancement demonstrates my understanding of data persistence and database integration - skills crucial for real-world applications.

Performance optimization was another area where I applied my software engineering knowledge. I implemented frustum culling to avoid rendering objects outside the camera's view, significantly improving performance for complex scenes. I also added performance metrics tracking to monitor render times and object counts, allowing for data-driven optimization. These enhancements show my ability to think about efficiency and scalability, not just basic functionality.

These enhancements directly address the course outcomes I planned to meet. I employed strategies for collaborative environments by implementing a modular design with clear interfaces that would make it easier for a team to work on different components simultaneously. I demonstrated my ability to design and evaluate computing solutions by using more efficient data structures and algorithms. I showcased well-founded techniques by implementing industry-standard design patterns like component-based architecture. I also partially addressed the security mindset issue through proper error handling and input validation in the database operations, though I plan to expand this in future work by adding user authentication and better input sanitization.

The enhancement process transformed not just the code but also my understanding of software engineering. What started as a simple graphics assignment became a comprehensive application of software design principles. The most satisfying aspect was seeing how concepts from multiple courses - object-oriented design, data structures, algorithms, database systems, and software engineering - all came together in this project. This experience has dramatically improved my skills in software architecture and design, preparing me for the challenges I'll face in my professional career working on complex, large-scale systems that require thoughtful architecture and robust implementation.