CS 499 Milestone Two – Enhancement One: Software Design & Engineering

Rafael V Canseco

SNHU

CS-499 Computer Science Capstone

Artifact Description
The artifact I selected for enhancement is a portion of the 3D graphics application originally created in CS-330: Computational Graphics and Visualization. This application uses OpenGL, GLM, GLEW, and GLFW to render a detailed 3D scene containing objects such as a table, backdrop, books, a percolator, and other decorative elements. The files I enhanced—SceneManager.cpp, ViewManager.cpp, and maincode.cpp—were originally developed in 2023 as part of my CS-330 final project. These files handle core responsibilities such as rendering the scene, managing camera transformations, loading shaders, and initializing the application window.

Justification for Inclusion in My ePortfolio
I selected this artifact because it is one of the strongest demonstrations of my software engineering abilities in a complex, real-world-style system. This application integrates multiple areas of computer science, including rendering pipelines, event-driven programming, memory management, defensive coding, and object-oriented design. Enhancing this artifact allowed me to showcase how I can take an existing codebase and elevate it through cleaner structure, reliability improvements, and modern coding practices.

Several components highlight my skills. I refactored duplicated rendering logic by introducing the helper function RenderBookSection(), reducing redundancy and improving maintainability in accordance with DRY principles. I strengthened the program with new defensive checks in maincode.cpp to ensure that GLFW, GLEW, shader compilation, and window creation all fail safely, improving the overall stability of the application. I updated documentation and naming conventions across the SceneManager and ViewManager files to improve readability, consistency, and clarity. I also improved memory safety by validating object creation and ensuring proper deletion during shutdown. These enhancements align closely with professional expectations for clean, modular, and safe software design, making this artifact an excellent addition to my ePortfolio.

Outcome Coverage
In Module One, I planned to update this artifact to demonstrate progress toward the software engineering outcome: “Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for implementing computer solutions.” I fully met this outcome by applying software design principles, refactoring core logic, adding defensive programming techniques, and improving reliability and clarity throughout the code.

These enhancements also demonstrate partial progress toward additional outcomes, such as designing and evaluating computing solutions, developing a security-minded approach through validation and safe memory management, and improving communication through clearer documentation. My original outcome-coverage plan remains accurate, and these updates provide strong evidence of growth in these areas.

Reflection on the Enhancement Process
Enhancing this artifact gave me valuable insight into how much can be learned by revisiting earlier work with more advanced skills. When I originally created this project in CS-330, my focus was mainly on rendering the scene correctly. Approaching the code again with a software

engineering mindset helped me see where the structure, readability, and safety of the application could be significantly improved.

During this process, I learned how to identify and consolidate duplicated logic into reusable helper functions, how defensive programming and validation contribute to a stable application, and how consistent naming and documentation make a major difference in long-term maintainability. I also reinforced the importance of proper initialization and cleanup in graphics applications that depend heavily on GPU resources and external libraries.

One of the biggest challenges I faced was locating the original CS-330 project files and making sure they still built correctly. Because the work was created years ago, I had to find the correct folders, verify shader paths, and ensure the OpenGL dependencies were still compatible. Stabilizing the old project environment was essential before making any enhancements. Another challenge was refactoring the book-rendering logic without changing the visual output of the scene. Every adjustment needed careful testing to ensure the rendered objects stayed identical. Enhancing initialization and shutdown routines also required careful attention to avoid introducing new issues.

Overall, this enhancement process strengthened my understanding of maintainability, modular design, and defensive programming. It elevated the artifact into a portfolio-quality example of my software engineering skills and demonstrated my ability to improve real-world codebases with clean, well-structured, and reliable solutions.

Code Enhancements:

MainCode.cpp

```
/***********************************************************
 *  main(int, char*)
 ***********************************************************/
int main(int argc, char* argv[])
{
    // Defensive check: GLFW initialization
    if (!InitializeGLFW())
    {
        std::cerr << "ERROR: Failed to initialize GLFW." << std::endl;
        return EXIT_FAILURE;
    }

    g_ShaderManager = new ShaderManager();
    g_ViewManager = new ViewManager(g_ShaderManager);

    // Create main window
    g_Window = g_ViewManager->CreateDisplayWindow(WINDOW_TITLE);

    // NEW CHECK – ensure window was actually created
    if (!g_Window)
    {
        std::cerr << "ERROR: Failed to create GLFW window." << std::endl;
        return EXIT_FAILURE;
    }

    // Defensive check: GLEW initialization
    if (!InitializeGLEW())
    {
        std::cerr << "ERROR: Failed to initialize GLEW." << std::endl;
        return EXIT_FAILURE;
    }

    // Load shader files
    g_ShaderManager->LoadShaders(
        "shaders/vertexShader.glsl",
        "shaders/fragmentShader.glsl"
    );
    g_ShaderManager->use();
```

ScenceManger.cpp

```cpp
/***********************************************************
 * Helper: RenderBookSection()
 * CS-499 Enhancement – removes duplicated boilerplate
 ***********************************************************/
void SceneManager::RenderBookSection(
    const glm::vec3& scaleXYZ,
    const glm::vec3& positionXYZ,
    const std::string& materialTag,
    const std::string& textureTag)
{
    SetTransformations(scaleXYZ, 0.0f, 0.0f, 0.0f, positionXYZ);
    SetShaderMaterial(materialTag);
    SetShaderTexture(textureTag);
    SetTextureUVScale(1.0f, 1.0f);
    m_basicMeshes->DrawBoxMesh();
}



/***********************************************************
 * RenderBook()
 ***********************************************************/
void SceneManager::RenderBook()
{
    float gap = 0.02f;   // Small gap between books
    float tableHeight = 0.2f;    // Height of the table
    float diagonalOffset = 0.5f;  // Amount to shift diagonally (x and z)

    // First book (bottom)
    RenderBookSection(
        glm::vec3(3.5f, 0.3f, 2.5f),
        glm::vec3(-6.0f + diagonalOffset, tableHeight, 5.0f + diagonalOffset),
        "cover",
        "book");

    RenderBookSection(
        glm::vec3(3.53f, 0.23f, 2.3f),
        glm::vec3(-6.0f + diagonalOffset, tableHeight + 0.01f, 4.89f + diagonalOffset),
        "pages",
        "pages");
```

ScenceMangerH.H

```cpp
    // Methods for rendering the individual objects in the 3D scene
    void RenderTable();       // Rectangular table
    void RenderPercolator();      // Coffee Percolator
    void RenderBackdrop();    // Backdrop
    void RenderBook();       // Books on the round table
    void RenderCoffeeCup();   // Coffee cup on the round table
    void RenderTray();          // Tray for Pot and Mug
    void RenderFlowerPot();    // Small flower pot
    void RenderBookSection(
        const glm::vec3& scaleXYZ,
        const glm::vec3& positionXYZ,
        const std::string& materialTag,
        const std::string& textureTag);



};
```

ViewManger.cpp

```
// CS-499 Enhancement: Documented projection modes
// 1 = Front orthographic
// 2 = Side orthographic
// 3 = Top orthographic
// 4 = Perspective view

if (glfwGetKey(m_pWindow, GLFW_KEY_1) == GLFW_PRESS)
{
    bOrthographicProjection = true;
    g_pCamera->Position = glm::vec3(0.0f, 4.0f, 10.0f);
    g_pCamera->Up = glm::vec3(0.0f, 1.0f, 0.0f);
    g_pCamera->Front = glm::vec3(0.0f, 0.0f, -1.0f);
}
if (glfwGetKey(m_pWindow, GLFW_KEY_2) == GLFW_PRESS)
{
    bOrthographicProjection = true;
    g_pCamera->Position = glm::vec3(10.0f, 4.0f, 0.0f);
    g_pCamera->Up = glm::vec3(0.0f, 1.0f, 0.0f);
    g_pCamera->Front = glm::vec3(-1.0f, 0.0f, 0.0f);
}
if (glfwGetKey(m_pWindow, GLFW_KEY_3) == GLFW_PRESS)
{
    bOrthographicProjection = true;
    g_pCamera->Position = glm::vec3(0.0f, 7.0f, 0.0f);
    g_pCamera->Up = glm::vec3(-1.0f, 0.0f, 0.0f);
    g_pCamera->Front = glm::vec3(0.0f, -1.0f, 0.0f);
}
if (glfwGetKey(m_pWindow, GLFW_KEY_4) == GLFW_PRESS)
{
    bOrthographicProjection = false;
    g_pCamera->Position = glm::vec3(0.0f, 5.5f, 8.0f);
    g_pCamera->Front = glm::vec3(0.0f, -0.5f, -2.0f);
    g_pCamera->Up = glm::vec3(0.0f, 1.0f, 0.0f);
}
```

Project Solution Compiled: