Milestone Four Narrative

Jason Kremhelmer

Southern New Hampshire University

CS-499: Computer Science Capstone

Dr. Maciosek

12/01/2024

# Briefly describe the artifact. What is it? When was it created

The artifact that I am using is a scene from CS330. It is a basic 3D scene rendered from a 2D image. The initial code was created by Brian Battersby – SNHU Instructor, Nov 1st, 2023. My work with this code was completed Oct 19th, 2024.

# Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?

I am including this artifact in my ePortfolio, as this is what I would like to do with my CS knowledge. I would like to create 3D renderings for people to enjoy. The components that I will be using to showcase my abilities are in the SceneManager.cpp, MainCode.cpp, and ViewManager.cpp. This is where the meat and potatoes are for the scene. I have improved efficiency in the code, removed redundant coding, corrected errors in some of the codework, as well as commented on the sections of improvement.

# Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

I am working on meeting my plans outlined in module 1. By the completion of this course, I will have either completed the planned updates or revised the plan to coincide with the updates I have made.

# Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

While I was improving on the code I am learning that some of the planned improvements are going to be more complicated that originally thought. As I am going through, I am learning new techniques to improve and modify code. Some of the challenges I faced were just the way to update the code. I have the concept in my head, just putting it into action is harder than anticipated.

# Updates to the base code

The updates that I have completed to this point in the MainCode.cpp are:

* // GLM Math Header inclusions

#include <glm/glm.hpp>

#include <glm/gtc/matrix\_transform.hpp> //updated to include OpenGL Mathmatics to improve 3D processing

#include <glm/gtc/type\_ptr.hpp>

#include "SceneManager.h"

#include "ViewManager.h"

#include "ShapeMeshes.h"

#include "ShaderManager.h"

* int main(int argc, char\* argv[])

{

// if GLFW fails initialization, then terminate the application

if (InitializeGLFW() == false)

{

return(EXIT\_FAILURE);

}

// try to create a new shader manager object

g\_ShaderManager = new ShaderManager();

// try to create a new view manager object

g\_ViewManager = new ViewManager(

g\_ShaderManager);

// try to create the main display window

g\_Window = g\_ViewManager->CreateDisplayWindow(WINDOW\_TITLE);

// if GLEW fails initialization, then terminate the application

if (InitializeGLEW() == false)

{

return(EXIT\_FAILURE);

}

// load the shader code from the external GLSL files

g\_ShaderManager->LoadShaders(

"../../Utilities/shaders/vertexShader.glsl",

"../../Utilities/shaders/fragmentShader.glsl");

g\_ShaderManager->use();

// try to create a new scene manager object and prepare the 3D scene

g\_SceneManager = new SceneManager(g\_ShaderManager);

g\_SceneManager->PrepareScene();

// loop will keep running until the application is closed

// or until an error has occurred

while (!glfwWindowShouldClose(g\_Window))

{

// Enable z-depth

glEnable(GL\_DEPTH\_TEST);

// Clear the frame and z buffers

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

// convert from 3D object space to 2D view

g\_ViewManager->PrepareSceneView();

// refresh the 3D scene

g\_SceneManager->RenderScene();

// Flips the the back buffer with the front buffer every frame.

glfwSwapBuffers(g\_Window);

// query the latest GLFW events

glfwPollEvents();

}

// clear the allocated manager objects from memory

if (NULL != g\_SceneManager)

{

delete g\_SceneManager;

g\_SceneManager = NULL;

}

if (NULL != g\_ViewManager)

{

delete g\_ViewManager;

g\_ViewManager = NULL;

}

if (NULL != g\_ShaderManager)

{

delete g\_ShaderManager;

g\_ShaderManager = NULL;

}

// Terminates the program successfully

exit(EXIT\_SUCCESS);

* }

At this point I am working on updating MainCode.cpp. The updates I am making in this code are to improve the databases used, and to open more options for better scene rendering.

# References

Quilty, D. (2024, April 18). *How to Improve Database Performance: The Ultimate Guide*. Retrieved from Percona: https://www.percona.com/blog/ultimate-guide-to-improving-database-performance/