Milestone Three Narrative

Jason Kremhelmer

Southern New Hampshire University

CS-499: Computer Science Capstone

Dr. Maciosek

11/22/2024

# 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 ViewManager.cpp are.

* void ViewManager::Mouse\_Position\_Callback(GLFWwindow\* window, double xMousePos, double yMousePos)

{

// when the first mouse move event is received, this needs to be recorded so that

// all subsequent mouse moves can correctly calculate the X position offset and Y

// position offset for proper operation

if (gFirstMouse)

{

gLastX = xMousePos;

gLastY = yMousePos;

gFirstMouse = false;

}

// calculate the X offset and Y offset values for moving the 3D camera accordingly

float xOffset = xMousePos - gLastX;

float yOffset = gLastY - yMousePos; // reversed since y-coordinates go from bottom to top

// set the current positions into the last position variables

gLastX = xMousePos;

gLastY = yMousePos;

// Sensitivity factor

float sensitivity = 0.2f; //increased float sensitivity. This will all the camera to respond quicker to smaller mouse movements

xOffset \*= sensitivity;

yOffset \*= sensitivity;

// move the 3D camera according to the calculated offsets

g\_pCamera->ProcessMouseMovement(xOffset, yOffset);

}

* void ViewManager::ProcessKeyboardEvents()

{

// close the window if the escape key has been pressed

if (glfwGetKey(m\_pWindow, GLFW\_KEY\_ESCAPE) == GLFW\_PRESS)

{

glfwSetWindowShouldClose(m\_pWindow, true);

}

// if the camera object is null, then exit this method

if (NULL == g\_pCamera)

{

return;

}

float velocity = g\_pCamera->MovementSpeed \* gDeltaTime; // Frame rate independent movement

// process camera zooming in and out

if (glfwGetKey(m\_pWindow, GLFW\_KEY\_W) == GLFW\_PRESS)

{

g\_pCamera->ProcessKeyboard(FORWARD, velocity);

}

if (glfwGetKey(m\_pWindow, GLFW\_KEY\_S) == GLFW\_PRESS)

{

g\_pCamera->ProcessKeyboard(BACKWARD, velocity);

}

// process camera panning left and right

if (glfwGetKey(m\_pWindow, GLFW\_KEY\_A) == GLFW\_PRESS)

{

g\_pCamera->ProcessKeyboard(LEFT, velocity);

}

if (glfwGetKey(m\_pWindow, GLFW\_KEY\_D) == GLFW\_PRESS)

{

g\_pCamera->ProcessKeyboard(RIGHT, velocity);

}

// process camera panning up and down

if (glfwGetKey(m\_pWindow, GLFW\_KEY\_Q) == GLFW\_PRESS)

{

g\_pCamera->ProcessKeyboard(UP, velocity);

}

if (glfwGetKey(m\_pWindow, GLFW\_KEY\_E) == GLFW\_PRESS)

{

g\_pCamera->ProcessKeyboard(DOWN, velocity);

}

* void ViewManager::ProcessInput()

{

ProcessKeyboardEvents();

// Other input handling code

}

The implementation of the above code will improve the camera’s response, as well as move smoother to the inputs of the user.

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

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

# References

*DSA Tutorial - Learn Data Structures and Algorithms*. (2024, Dec). Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/dsa-tutorial-learn-data-structures-and-algorithms/