2-2 Milestone One: Code Review

Frank Smith

CS – 499

Southern New Hampshire University

**Category One: Software Design and Engineering**

The enhancement I have chosen for this category is derived from CS – 330 Computer Graphic and Visualization. Most of this code is plug and play regarding the deployment of modular classes that are included in the executable portions of the code that are manipulated to complete a 3D scene of your choosing. Header files such as SceneManager.h, ShaderManager.h, and ShapeMeshes.h are provided for the simple deployment of shapes within the chosen scene. Within those files are clearly commented functions that load, set, and render the objects desired which are then manipulated to have the correct locations within the scene, lighting, as well as textures and material make up to capture the realistic scene as a 3D replica. ViewManager.cpp specifies WINDOW\_WIDTH and WINDOW\_HEIGHT to create a viewing screen pixel size and then creates a camera object to be used for our view of that screen. In order to manipulate the scene, mouse coordinates are specified and tracked between frames. The ViewManager constructor initializes the camera object and sets it into a default position within the scene. Other constructors provide user input to manipulate the scene such as Mouse\_Position\_Callback() (consistently updates the position of the mouse input as x and y positions), and ProcessKeyboardEvents() (specifies keys as input that manipulate the camera position up, down, left, right, forwards, and backwards).

The portion of the code that I would like to enhance is the SetupSceneLights() method. The scene lights methods allow the developer to apply a single or a multitude of lights within the 3D environment and control how large they are, their position, the direction they are pointing, as well with how bright they are and what color they are. These lights are vectors that have 3 values assigned to them that will adjust the parameters mentioned above. To properly enhance this code, I plan on removing some of the lights as I believe they were excessive and didn’t bring much to the final product of the assignments while also applying better commenting to describe what each light and light vector is doing and how it applies to the 3D scene.

**Category Two: Algorithms and Data Structure**

The enhancement I have chosen for the second category is obtained from my repository for CS – 210 for the assignment ClockTime.cpp. This code was very basic that allowed us to create methods that added hour, minute, and seconds to a live clock. The hours can be taken from standard 0-12 or military time 0-24, minutes 0-59, and seconds 0-59. If the hour was equal to 11 or 23 when time to iterate the hour up, it would jump to 0. If the minute or second were to iterate up when at 59, it would jump to 0. Any other times for the hour, minute, or second add functions would iterate up by 1. The system when running would print to screen two clocks as stated above that will show default values for both clocks that a user can then manipulate with input. If 1 was entered to iterate the hour up, the addHour method is called to increase the hours from the defaults specified. This same action would occur if a 2 for add one minute is selected for the minute’s iteration the addMinute method would be called as well as the add one second option to add a second to the clock with the addSecond method.

A practical enhancement for this code would be to apply loop statements for the add functions for the hour, minute, and second methods that will automatically iterate up. Create a loop that increases the second value by 1 every second, the minute value up by 1 every 60 seconds that would reset the second value, and then iterate the hour up by 1 every 60 minutes that would reset the second value. That would provide a live clock. Then I would adjust the userInput loop to properly add hours, minutes, or seconds while the loop was running automatically.

**Category Three: Databases**

The third category I chose to reference the project in CS – 330 again to update and optimize the DefineObjectMaterial() function. The OBJECT\_MATERIAL struct is a vector containing values that specify the diffuse color, the specular color, the shininess of the object, and a tag to track the name of the material. My DefineObjectMaterials() method specified a wood material as well as a wall material. These properly captured the glossy look of the back wall that contained a little more shine than the wood would which the light was able to properly reflect off of the wall and show. Using the tag for materials I was then able to call those objects into the scene and apply them to different objects within the RenderScene() method. A practical enhancement for this code would be to specify another couple materials within the database that would better represent the objects within the scene. My plan is to create a metal material and a carpet material that will improve the look of the scene while sticking to best coding practices such as properly commenting what each setting is doing once applied.