Gagujas, Michael

CS499 – Computer Science Capstone

July 14, 2024

***Outline***

EventTracker Code

Slight demonstration- talk about code history reason, purpose behind it,

CS330 – Comp Graphics 3D Scene

Software Engineering and Design

Enhancement 1 – Create Light/shader class to declutter main function

Enhancement 2 – Configuration file for magic numbers with shaders

Enhancement 3 – improve skybox performance and use encapsulation

Enhancement 4 – Add animated objects

Algorithms and Data Structures

Enhancement 1 – Implement Discrete Level of Detail (DLOD)

Enhancement 2 – Implement arrays/linked lists/hashmaps for DLOD

Databases

Enhancement 1 – Implement Android Keystore system, prevent brute force attacks from repeated login attempts

Enhancement 2 – Implement Firebase Authentication

Enhancement 3 – Implement third-party service login through Google and GitHub

***Script***

Hello, and thank you for taking the time to listen to my code review. My name is Michael Gagujas, and today I will be performing a code review on two of my artifacts, which are from CS330 – Computational Graphics and Visualizations and CS 360 – Mobile Architecture and Programming. The overall goal and purpose of this video is to analyze the current condition of the code in the artifacts that I’ve chosen for my final project, discuss my plans of enhancements, and demonstrate my abilities to perform a code review.

For the first artifact that we’ll be reviewing is my OpenGL Final project for CS330 – Computational graphics and visualizations, where I’ve recreated a 2D image into a 3D scene.

Start from main.cpp

First thing that I’ve noticed is that there can be

Better naming conventions, like for line 56, I can use MeshCreator meshManager rather than gMesh, since it is more descriptive of its purpose as it is used to create and manage different types of 3D meshes.  
Same for line 59 and 62 where gTexture can be more descriptive as textureManager and builder will be less vague if i use sceneBuilder.

And, also, Disclaimer:, An issue throughout the project that will be addressed is more use of error handling throughout the code and the necessity for better comments

Numerous comments restate what the code already clearly indicates, which will need to be fixed in a majority of the files as those comments were used to assist with learning the material at the time when it was relatively new. In general, it would be better to have more meaningful comments that adds value as to why something was done, or to provide context. If anything, I can group the related code together like the object creations and provide meaningful comments from there to make the code easier to read and maintain.

Looking at Line 65, it has magic numbers that would be better replaced with named constants like CAMERA\_INITIAL\_X , CAMERA\_INITIAL\_Y, and CAMERA\_INITIAL \_Z.

Line 66 and 67 can also change 2.0f to constant like SCREEN\_DIVISOR. Ultimately, I’m looking for consistency with the naming conventions, so that their names are clear and descriptive.

For lines 75-84, theres a common issue throughout the file where light settings and shaders are mixed in with the main program. My first enhancements that I will make that addresses the software engineering and design category of the project, is to encapsulate the light settings and controls, into a Light class, to make the code more organized, readable, and maintainable. Making the code more modular will make it easier to add, remove, or modify light settings in the future.

For lines 87-92, the function declarations can be greatly improved by adding comments that describe what the function does, its parameters, and its return value(if any), so that other developers can understand the purpose of the function without having to read through its implementation.

As we enter the main function, the following lines of code, initializes the GLFW library.

Line 110, I can remove the hard-coded window title of “7-1 Michael Gagujas” and make a variable like const char\* windowTitle = “7-1 Michael Gagujas” so that it’s easier to change a variable than a hardcoded string, especially if its placed in multiple locations.

For lines 117-121, I can add a group comment that the function calls configure the GLFW window and set up its necessary callbacks.  
  
For lines 139-156, they are essential for creating shader programs from vertex and fragment shader files, and they are creating the meshes and loading textures for giving the scene a detailed appearance.

After these tasks are finished with setting up the graphics, the main rendering loop begins.  
For the majority of the code in the rendering loop especially after line 177, I will implement better separation of concerns by creating a separate function renderScene(), that makes the main loop cleaner and easier to read. On top of encapsulating these calls in a separate lighting class, especially from 196-236, it improves the overall quality of my code and makes the lighting reusable for different parts of the code, or for different projects. The second enhancement that I will make is to use a configuration file to handle the numerous magic numbers in lines 196-236.

(Say this as skimming 237 – 287)

Again, for the past couple of blocks of code, the comments should be more meaningful and magic numbers should be better handled.

For lines 289-313, this block of code is part of my skybox and involves the third enhancement for my project. First, I’ll encapsulate the code into a separate function to make the code more maintainable and reusable. But my maintenance

enhancement involves improving its performance by using optimization techniques like back-culling and reducing the skybox’s texture size to improve its performance because as you can see \*\*Demonstrate the frame drops\*\* the frame rate drops significantly when it’s on.

For lines 316 to 337, it involves cleaning up and releasing all of the resource sthat were allocated during the execution of the graphics program.

After the main function, we have the functions that were declared earlier that handle specific user inputs.

With lines 341-378, it controls the camera movement and the movement of the light sources in the scene. To improve the code, I will use constants for key bindings instead of hardcoding the keys. The constants can be defined at the top of the file, which can make it easier to change the key bindings if necessary.

I will also change the string comparisons in lines 368-378 to enums to make the comparison less error prone and faster. In addition, better commenting regarding what “point light selection” and “animate point lights” means should be implemented here as well.

For lines 381 to 421, more in-depth comments, especially regarding parameters would improve the code, and using enums instead of string comparisons will improve the code.

For lines 423 to 447, it handles the mouse and mousewheel controls, and I will improve this code by removing the global variables firstMouse, lastX, lastY, and camera and having them by passed in as parameters.

Overall, the main C++ file has numerous enhancements that I plan to implement, and it has a lot of comments and error handling that need to be implemented.

To further improve my project, my 4th enhancement for software engineering and design will involve my SceneObjects.cpp file, where I will add objects that are animated like a firefly that’s constantly moving in the scene. This specific file uses meshes, textures and shaders to transform and render shapes into complete objects that are in the scene.

For this specific file, there is a tremendous amount of repetition with long function lengths that need to be broken down into more smaller and manageable functions. Helper functions will be instrumental for reducing the duplication. In addition, just like with the main.cpp file, comments need to be more descriptive, error handling must be performed, and better variable naming conventions need to be implemented.

For example, lines 28-35, where a shape is scaled, rotated, and translated can be made into a separate function since it is frequently used. I will also make a separate function for setting up the textures like lines 44-47 to make the code more modular and manageable.

Overall, the rest of the code repeats for different scene objects, and for the sake of simplicity I will transition to my 1st enhancement for the algorithms and data structures category, which involves my MeshCreator.cpp file. For my 1st enhancement, I will implement Discrete Level of Detail, to optimize the rendering of my 3D scene, by having different versions of my 3D models with varying levels of detail like low poly and high poly versions. This is related to the MeshCreator.cpp file because this file creates and manages the different types of 3D meshes for defining the shape of a 3D object. By using a distance-based algorithm that calculates the distance between the camera and each object in the scene, I can enhance the performance of my project and still maintain the visual integrity of my 3D scene. In addition, to manage these different versions of each model, I will implement my 2nd enhancement to algorithms and data structures by utilizing a combination of arrays, linked lists, and hash maps to different versions of each model. **The hash map will be used to initially render the scene while linked lists will store the different versions of each model and make it easier for switching based on camera distance. Currently, only arrays are used to store the vertices and polygons of meshes, which can be seen in the continuation of our code review.**

For lines 16-40, I have designated functions that create the mesh data and releases the mesh data during initialization and termination.  
For our first function, it creates a plane mesh in OpenGL and uses the verts array to contain the vertex data for the plane. For this variable, a better variable name can be planeVertices to make the code more clear. In addition, similarly to the previous file, SceneObjects.cpp, code is repeated numerous times and would be better as smaller and reusable functions. Generating the vertex array like line 58-59, generating the buffers like line 61-64, setting the indices like 70-72, and setting the attribute pointers like lines 74-85 can make the code easier to understand and maintain. Especially because it is used frequently in other shapes.

For the next shape, where it creates a pyramid shape with a square base, it follows a similar structure with specifying the vertices, normals, and texture coordinates in a single array. Like the previous function for the plane mesh, reusable smaller functions for generating the vertex array, generating the buffers, and setting the vertex attribute pointer should be used to make the code more readable and maintainable. Essentially, the other seven shapes follow the same structure, so I will transition to my 2nd artifact, an event tracking mobile application that was used for my final project in CS360 – mobile architecture and programming. For this artifact, I will be enhancing aspects of its database to fulfill the databases category of the project.

**For this project, I had to develop a fully functional mobile application that allowed a user to login, add events to a database, and, after receiving user permission, send SMS messaging notifications. This artifact aligns with the category of databases because it uses SQLite to store event information and user credentials like salted and hashed passwords. The first file that I will enhance involves the LoginActivity Java file where users can login and signup, where their credentials like username, salt, and salted passwords are stored securely in SQLite.**

**In the beginning of the code, starting from line 32, a new class named Loginactivity extends the AppCompatActivity class to support library action bar features. Private variables that align with the Model-View-ViewModel architectural pattern can be observed along with variables to hold the email Input and password Input from users. So far, the code follows Android coding best practices, but more detailed comments can be added. For line 53, there is deprecated code for the flag translucent status, and that should be updated or secured to address any potential security issues.**

**For lines 63-81, it sets an onclick listener to retrieve the text entered by a user and calls the isValidCredentials method to check if the credentials are valid. When the credentials are valid, it starts a new main activity to represent a successful login, but when the credentials are invalid, it shows a Snackbar message that tells the user their username or password is incorrect. Overall, the code separates the validation logic from the UI logic, but it does not follow secure coding practices as it has a few potential security issues. Because the code sends the email as an extra with the intent to the MainActivity, it can potentially by intercepted by malicious appos or be read if the device is compromised. In addition, the password is handled as a String, which is immutable and cant be removed from memory until garbage is collected, allowing an attacker to potentially read it from memory. Moreover, the code doesnt show any rate limiting or account lockout after multiple failed login attempts.**

**For my first enhancement, I will improve the database design and handling of credentials by suggesting code changes like limiting the number of failed login attempts to mitigate brute force attacks, handle passwords as characters, which can be cleared after use, and use the Android Keystore system to securely store and retrieve emails. Similarly, lines 84-111 handles the sign up process and has the same issues as the login process, where it is receiving inputs and is vulnerable to potential attackers**

**For lines 115-133, it handles the credentials validation process with retrieving the password salt and hashed password, but there are better methods for ensuring that credentials are not compromised. Although for my 2nd enhancement for the databases category, I will be implementing Firebase Authentication to add an extra layer of security to the application, and ensure that only authenticated users can access its database.**  
  
**Moving on, we have the final part of LoginActivity which consists of an onresult listener that creates a callback for the results of a validation process, and a hideKeyboard function that hides the device’s softkeyboard.**

Essentialy that is the majority of the files that I will be enhancing with exception to the ones that I will have to create wholely. But I will also be implementing a 3rd database enhancement, where I connect my mobile application to Firebase services and allow users to login with third party services like Google and GitHub.

Thank you very much for your time.