# Computer Graphics Coursework – Self Assessment Document

**Name:** *Taimur Ali Khan* **ID number:** *23728715*

Complete the self-assessment grid below by writing a short explanation of how you have satisfied the requirement and how it has implemented in your code.

|  |  |  |
| --- | --- | --- |
| **Learning outcome** | **Mark** | **Weighted mark** |
| 1. Use appropriate mathematical tools (40%) | 62 | 24.8 |
| 2. Develop a 3D graphics application (30%) | 68 | 20.4 |
| 3. Write shader code (30%) | 68 | 20.4 |
|  | Total | 65.63 |

Your mark for each Learning Outcome (LO) is the highest mark achieved based on the criteria specified in the self-assessment grid. Note that you will need to have satisfied all criteria at the lower mark bands to be awarded marks in the higher mark bands, e.g., to get a mark in the 70 - 80 band for a learning outcome you will have needed to have satisfied all criteria in the 40 – 50 and 50 – 60 mark bands.

## Learning Outcomes:

**LO1** Select and use appropriate mathematical tools for constructing and manipulating geometry in 3D space.

**LO2** Develop an interactive 3D graphics application using an industry-standard API.

**LO3** Write shader code for the programmable pipeline on modern graphics hardware using an industry standard shader language.

## Self-assessment Grid

|  |  |  |
| --- | --- | --- |
| **Mark** | **Criterion** | **Comments (state how and where you have achieved the criterion)** |
| 42, 45, 48 | LO1: Basic use of vector and matrix objects | I used glm::vec3 for positions (camera, eyeball) and glm::mat4 for transformations (view/projection matrices). Calculations like glm::length() and glm::normalize() were used for distances and directions. |
| LO2: Application compiles and runs without alterations to the source code of CMake file. | A screenshot of a video game  AI-generated content may be incorrect.A screenshot of a computer  AI-generated content may be incorrect. |
| LO3: Implementation of shaders to apply appropriate textures to objects. | While no image textures were used, shader uniforms were applied to dynamically color objects in the scene (walls, bulbs, eyeball). |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | I used glm::translate() and glm::scale() to position and size the stand, eyeball, and bulbs. No object rotation was needed. |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | I used glm::lookAt() for view matrix and glm::perspective() for projection matrix setup. |
| LO2: 3D virtual world has been created using instances of a single object type. | The room and light bulbs use a single cube VAO drawn with different transformations and colors. |
| LO3: Use of shaders to apply dynamic lighting from point light sources | Only one light source on the eye ball. |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. | View and projection were handled using glm::lookAt() and glm::perspective(). |
| LO2: 3D world created using multiple object types. | A cube VAO is used for room walls and bulbs. The eyeball is imported as a model (.obj) using the model loader. |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | The camera is controlled via WASD keys for movement and mouse for orientation. |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. | No only light cube has been shown . |
| 72 75, 78 | LO1: Implementation of students own functions to replace glm functions (e.g., glm::length(), glm::dot(), glm::cross() etc.). | I did not implement custom vector functions; instead, I used glm for all vector calculations. |
| LO1: Implementation of quaternions to calculate rotation matrix. | Not implemented. All vector calculations (distance, direction) were done using glm library.  . |
| LO2: Interactive dynamic aspects of the virtual word and controllable by the user (e.g., position of objects, location and function of light sources etc.). | Implemented keyboard and mouse-based navigation, and proximity-based color of ceiling bulbs near the table. |
| LO3: Appropriate implementation of normal and specular maps. | Only basic diffuse color effects using uniform variables in shaders were used. |
| 85, 90, 100 | LO1: Use of quaternions to calculate view matrix. | Not implemented |
| LO1: Use of SLERP to smooth out changes in camera direction. | Not implemented |
| LO2: Implementation of a third person camera with the ability to switch between first and third period view. | Not implemented. |
| LO2: The position of the camera or character obeys the constraints of the physical space (e.g., can’t pass through objects, can’t hover in midair etc.). | Collision detection prevents camera movement through the eyeball and limits movement within room bounds. |
| LO3: Use of shaders to apply parameter driven effects within the scene, e.g., light properties controlled using camera/character position. | Not implemented |