# Computer Graphics Coursework – Self Assessment Document

**Name:** *Mohammed Ashfak Anayet* **ID number:** *21336445*

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%) |  | 0 |
| 2. Develop a 3D graphics application (30%) |  | 0 |
| 3. Write shader code (30%) |  | 0 |
|  | Total | 0 |

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 | For making use of basic vector objects, I’ve defined each position, scale and rotation of my objects using vectors. Also, the camera makes use of vector objects.      During the calculation of model matrix, I make use of matrix objects. This is used for 3D transformations. |
| LO2: Application compiles and runs without alterations to the source code of CMake file. |  |
| LO3: Implementation of shaders to apply appropriate textures to objects. | I first compiled shaders by loading the vertex shaders and fragment shaders. To apply appropriate textures to objects I loaded the models and add their corresponding textures like this: |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | In the render loop where I calculate the model matrix, I make use of translations and transformations |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | In the render loop I made use of calculating view and projection matrices |
| LO2: 3D virtual world has been created using instances of a single object type. | I loaded the cube object and used it multiple times (the cube was made using Blender). |
| LO3: Use of shaders to apply dynamic lighting from point light sources |  |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. |  |
| LO2: 3D world created using multiple object types. | Yes, there’s a cube object and a plane object. |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | Yes. I had written the code so that the users can move around the world using the “WASD”, left shift and space bar to go up and down in the Y-axis. The user can also move and rotate the camera around using the mouse |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. |  |
| 72 75, 78 | LO1: Implementation of students own functions to replace glm functions (e.g., glm::length(), glm::dot(), glm::cross() etc.). |  |
| LO1: Implementation of quaternions to calculate rotation matrix. |  |
| 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.). |  |
| LO3: Appropriate implementation of normal and specular maps. |  |
| 85, 90, 100 | LO1: Use of quaternions to calculate view matrix. |  |
| LO1: Use of SLERP to smooth out changes in camera direction. |  |
| LO2: Implementation of a third person camera with the ability to switch between first and third period view. |  |
| 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.). |  |
| LO3: Use of shaders to apply parameter driven effects within the scene, e.g., light properties controlled using camera/character position. |  |