# Computer Graphics Coursework – Self Assessment Document

**Name:** *Tia Sherriff* **ID number:** *23647045*

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 | Vectors and matrices are used throughout my project. Vectors have been used, for example, within the camera class as well as matrices. |
| LO2: Application compiles and runs without alterations to the source code of CMake file. |  |
| LO3: Implementation of shaders to apply appropriate textures to objects. | Completed throughout the project (the floor, platform and obelisks all have textures). |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | Used throughout, specifically used to calculate the model matrix. |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | Used initially, created own that replaced these (see in further section). |
| LO2: 3D virtual world has been created using instances of a single object type. | Multiple object types have been used (see in further section). |
| LO3: Use of shaders to apply dynamic lighting from point light sources | Point lights used within project to form lights above obelisks. |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. | Can be seen within the camera class. |
| LO2: 3D world created using multiple object types. | Cubes, spheres and planes have all be used within the project (forming lights, the floor, obelisks and the platform). |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | Can be seen within the main program. |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. | Multiple light sources have been used (7 in total) – 1 spotlight and 8 point lights. |
| 72 75, 78 | LO1: Implementation of students own functions to replace glm functions (e.g., glm::length(), glm::dot(), glm::cross() etc.). | Can be seen within the maths class. Length was not used within my program. |
| LO1: Implementation of quaternions to calculate rotation matrix. | Can be seen within the camera class. |
| 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.). | The obelisks within my project move upwards when the player approaches them. The light in the centre of the world changes colour when the player stands in the centre of the world; other lights move upwards from the obelisks when the player does this, too. |
| LO3: Appropriate implementation of normal and specular maps. | The ground and obelisks use specular maps while the platform uses a normal map. |
| 85, 90, 100 | LO1: Use of quaternions to calculate view matrix. | Can be seen within the camera class. |
| LO1: Use of SLERP to smooth out changes in camera direction. | Can be seen within the camera class (refer to previous screenshot). |
| LO2: Implementation of a third person camera with the ability to switch between first and third period view. | The player is able to switch between a first and third person 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.). | The character/camera can’t pass through objects in some instances, but can in others (initially, the camera/character can’t pass through the platform but after jumping on it, it can). The code for this can be seen within the main program. The camera’s y is set to 0 so the player can’t hover. The screenshot below shows the “player” colliding with the platform, unable to move through it (I chose the player to be a cube rather than Suzanne as I felt it fit more with my theme). |
| LO3: Use of shaders to apply parameter driven effects within the scene, e.g., light properties controlled using camera/character position. | As mentioned previously, there are lights within the scene which move and change colour depending on the player’s position. |