# Computer Graphics Coursework – Self Assessment Document

**Name:** *Roshan Patel* **ID number:** *23669475*

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 | Coursework.cpp lines 142-148, 294-297  Regular use of vectors and matrices throughout the code. Provided examples show vectors to display teapots and matrices for the model matrix. |
| LO2: Application compiles and runs without alterations to the source code of CMake file. |  |
| LO3: Implementation of shaders to apply appropriate textures to objects. | Coursework.cpp lines 107-113  Have added crate.jpg texture to blank cube.obj to visualise 3D crates. |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | Coursework.cpp lines 294-297  Used translate, scale and rotate as part of the model matrix. |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | Have implemented my own functions. |
| LO2: 3D virtual world has been created using instances of a single object type. | Coursework.cpp lines 173-182, 306-307  Used for loop to display instances of teapot object and are displayed by being called in the render loop. |
| LO3: Use of shaders to apply dynamic lighting from point light sources | Coursework.cpp lines 116-119  Added a point light source to the scene for lighting effects. |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. | Camera.cpp lines 40-43  Used orientation.matrix multiplied by Maths::translate to calculate the view matrix and Maths::perspective for the projection matrix. |
| LO2: 3D world created using multiple object types. | Coursework.cpp lines 173-189, 306-313  Multiple teapot and crate objects displayed in the scene. |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | Coursework.cpp lines 15-16, 333-371  Captured and enabled mouse and keyboard inputs to allow user movement. |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. | Coursework.cpp lines 116-132  Used multiple point lights alongside a spotlight and directional light to light the room. |
| 72 75, 78 | LO1: Implementation of students own functions to replace glm functions (e.g., glm::length(), glm::dot(), glm::cross() etc.). | Maths.cpp lines 3-66  In the Maths class have replaced all of the currently used glm functions. |
| LO1: Implementation of quaternions to calculate rotation matrix. | Maths.cpp lines 92-112  Have used a quaternion matrix to calculate the 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.). | Not complete. |
| LO3: Appropriate implementation of normal and specular maps. | Coursework.cpp lines 107-110, 193-195, 222-224  Have used normal and specular maps on objects. |
| 85, 90, 100 | LO1: Use of quaternions to calculate view matrix. | Camera.cpp line 40  Used quaternions to calculate the view matrix. |
| LO1: Use of SLERP to smooth out changes in camera direction. | Camera.cpp line 34  Used SLERP to smooth out camera (can disable natural camera lock as later explained in the self-assessment grid to verify if needed) |
| LO2: Implementation of a third person camera with the ability to switch between first and third period view. | Not complete. |
| 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.). | Camera.cpp line 37 and Coursework.cpp lines 364-367  Locked the y level of the camera at a natural eye level so the user can no longer fly around midair and also locked the camera pitch at an appropriate level so the user cant turn all the way up/down to simulate what a person would actually be able to do. |
| LO3: Use of shaders to apply parameter driven effects within the scene, e.g., light properties controlled using camera/character position. | Not complete. |