# Computer Graphics Coursework – Self Assessment Document

**Name:** *Your Name* **ID number:** *12345678*

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 | Used glm::vec multiple times across the project, prominent example can be found coursework.cpp |
| LO2: Application compiles and runs without alterations to the source code of CMake file. | Paste a screenshot of your application below (Screenshot has been provided below) |
| LO3: Implementation of shaders to apply appropriate textures to objects. | I’ve applied diffuse textures for multiple objects which can be found in coursework.cpp |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | To build the 3D environment I had to use translation, rotation and scaling transformations on the plane objects to build the floor, four walls and the ceiling. |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | I have implemented code using the glm library functions to make a working camera for my scene, examples can be found in camera.cpp and camera.hpp |
| LO2: 3D virtual world has been created using instances of a single object type. | In coursework.cpp I used a for loop to put multiple crates into the virtual world. See Line 166 |
| LO3: Use of shaders to apply dynamic lighting from point light sources | I have used shaders to calculate multiple light sources as can be seen in light.cpp and the fragmentShader.glsl |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. | Had to create my own lightFragmentShader.glsl and lightVertex.glsl to get the shaders to work as they were not originally included in the coursework files. |
| LO2: 3D world created using multiple object types. | The world features different objects like crates and a teapot as well as the 2D planes to build the room. |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | Yes, keyboard and mouse functionality can be found in coursework.cpp |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. | Spotlights and point lights have been implemented into the 3D environment |
| 72 75, 78 | LO1: Implementation of students own functions to replace glm functions (e.g., glm::length(), glm::dot(), glm::cross() etc.). | n/a |
| LO1: Implementation of quaternions to calculate rotation matrix. | n/a |
| 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.). | n/a |
| LO3: Appropriate implementation of normal and specular maps. | n/a |
| 85, 90, 100 | LO1: Use of quaternions to calculate view matrix. | n/a |
| LO1: Use of SLERP to smooth out changes in camera direction. | n/a |
| LO2: Implementation of a third person camera with the ability to switch between first and third period view. | n/a |
| 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.). | n/a |
| LO3: Use of shaders to apply parameter driven effects within the scene, e.g., light properties controlled using camera/character position. | n/a |

A screenshot of a video game

AI-generated content may be incorrect.