# Computer Graphics Coursework – Self Assessment Document

**Name:** Zoe Ive **ID number:** *23611589*

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%) | 70 | 28 |
| 2. Develop a 3D graphics application (30%) | 70 | 21 |
| 3. Write shader code (30%) | 70 | 21 |
|  | Total | 70 |

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 | In camera.hhp, I have used vectors to define world positions for the camera in terms of x, y and z coordinates. The matrices are used to find the viewing direction of the camera in the 3d world.  (camera.hpp, ln 21 – 32) |
| LO2: Application compiles and runs without alterations to the source code of CMake file. | Application runs with no errors |
| LO3: Implementation of shaders to apply appropriate textures to objects. | Multiple 3D objects created with shader to apply a box texture to all the faces  (coursework.cpp, ln 257 – 261)  (coursework.cpp, ln 368 – 385) |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | These transformations translate, rotate and scale the 3d objects based on the value of the iterable int “i” so that all the boxes are in different positions.  (coursework.cpp, ln 304 – 308) |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | (camera.hpp, ln 30 – 32)  (camera.cpp, ln 14 – 18) |
| LO2: 3D virtual world has been created using instances of a single object type. | Creates a 3D cube in the environment  (coursework.cpp, ln 301 – 320) |
| LO3: Use of shaders to apply dynamic lighting from point light sources | I created a model object from the cube asset and applied the crate texture to it, then created a point light source to shade it.  (coursework.cpp, ln 251 – 268) |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. |  |
| LO2: 3D world created using multiple object types. | I have used the sphere.obj file to create a sphere object alongside the boxes with it’s own texture, lighting and independent vector 3 coordinates. When drawing the model, I have used a selection statement to specify which object is being drawn so that they do not overlap but are instead discreet. |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | This code allows the user to control the camera and move around the 3D space as they wish, only being restricted on the y axis which they can navigate by jumping  (coursework.cpp, ln 342 – 378) |
| 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. | (maths.cpp, ln 58 – 78) |
| 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. | (camera.cpp, ln 28 – 49) |
| 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. |  |