# Computer Graphics Coursework – Self Assessment Document

**Name:** *Michael Kihara* **ID number:** *20040382*

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%) | 72 | 0 |
| 2. Develop a 3D graphics application (30%) | 70 | 0 |
| 3. Write shader code (30%) | 50 | 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 basic vector objects to create vertex arrays (coursework.cpp ln 136), used matrix objects to write transformation functions (Maths.cpp ln 6) |
| LO2: Application compiles and runs without alterations to the source code of CMake file. |  |
| LO3: Implementation of shaders to apply appropriate textures to objects. | Implemented shaders to apply textures in vertexShader.glsl, fragmentShader.glsl, appropriately applied various textures to different objects and object types (coursework.cpp lines: 348, 353, 411, 425, 558) |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | Used translation, scaling and rotation transformations while calculating the model matrix (coursework.cpp line 461), used basic translation when moving objects in the 3d world (coursework.cpp line 540, 541). |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | Implemented view and perspective matrix calculations in Maths.cpp class (maths.cpp lines 52 and 82), also had previously used glm functions to do so (camera.cpp line 30) |
| LO2: 3D virtual world has been created using instances of a single object type. | 3d world has been created using instances of Object class (cousework.cpp lines 38, 499) |
| LO3: Use of shaders to apply dynamic lighting from point light sources | Shaders have been written to apply dynamic lighting from a point light source (fragment shader 17-20, 30-36) |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. | Implemented view and perspective matrix calculations in Maths.cpp class (maths.cpp lines 52 and 82), camera.cpp lines 25,30 is where this is implemented. |
| LO2: 3D world created using multiple object types. | 3d world is created using more than one object type, the first being a basic Object struct, and the second being an elongated diamond with separate buffer objects and vertex arrays to the prior class. They are defined at (obj type 1: courework.cpp 38 ,283, type 2: line 317) |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | I have implemented the keyboard and mouse input functions at line 588 (coursework.cpp) and line 645 (courework.cpp) |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. | Fell short of implementing this. |
| 72 75, 78 | LO1: Implementation of students own functions to replace glm functions (e.g., glm::length(), glm::dot(), glm::cross() etc.). | Implemented glm::length, glm::dot, glm::cross in maths.cpp at lines 98, 109 and 114. |
| LO1: Implementation of quaternions to calculate rotation matrix. | Fell short of this. |
| 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.). | Dynamic aspects of the world were implemented in the form of the position of an object being manipulated by the user moving into the object. The object moves to a randomly determined part of the 3d world’s bounds. (it cant move out of the ~50 radius zone bound by the mountains. This functionality can be observed in coursework.cpp at lines 34, 540, and 623. |
| 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. |  |