# Computer Graphics Coursework – Self Assessment Document

**Name:** *Maira Kamran* **ID number:** *23670574*

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 | I have used vectors to position and colour my objects and matrices to transform them. For example, I set the position of the teapot statue by writing “glm::vec3 statuePosition = glm::vec3(15.0f, 1.0f, 10.0f);” and “glm::mat4 translate = Maths::translate(statuePosition);” ( a matrix) to translate it. |
| LO2: Application compiles and runs without alterations to the source code of CMake file. | Paste a screenshot of your application below |
| LO3: Implementation of shaders to apply appropriate textures to objects. | I used shaders to apply specular shading to the grass map I created, and to use normal mapping for the teapot. |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | I have used translation, rotating and scaling on each of the objects in the project. |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | I used my own “Maths” function to calculate the view and projection matrices. |
| LO2: 3D virtual world has been created using instances of a single object type. | There are multiple object types in the 3D world. |
| LO3: Use of shaders to apply dynamic lighting from point light sources | I’ve used the vertex and fragment shading to apply various shading types, including ambient, specular and diffuse. The plane and teapot each have their own constants set for each type of shading. There are multiple light sources available, including a spotlight, point light, and a directional light. I have coloured each differently. The spot light points at where the teapot has started off at. |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. | In the maths.cpp and maths.hpp sections you will see how I have implemented my own “perspective” and “lookAt” functions to calculate the view and projection matrices. |
| LO2: 3D world created using multiple object types. | There are 3 lights, a teapot and a flat plane. |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | The user can look around using the mouse and use WASD to move themselves around. |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. | There are 4 light sources, 3 of which are visible via sphere objects. They each have attenuation, position, and colour specified. One is a point light, another is a spot light, and another is a directional light. The objects have constants set for how specular, diffuse and ambient lighting affects them. |
| 72 75, 78 | LO1: Implementation of students own functions to replace glm functions (e.g., glm::length(), glm::dot(), glm::cross() etc.). | I have replaced the glm::cross() function with my own in maths.cpp, as I use it in camera.cpp and maths.cpp. |
| LO1: Implementation of quaternions to calculate 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.). | When the camera eye’s distance from the teapot is less than or equal to 8, the user can move the teapot around using WASD keys. At the bottom of coursework.cpp you can see how I have implemented this. |
| LO3: Appropriate implementation of normal and specular maps. | The teapot uses a normal map, and the grass plane uses a specular map. |
| 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. |  |