# Computer Graphics Coursework – Self Assessment Document

**Name:** *Wendy Yu* **ID number:** *23686787*

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 the basic vectors such as gml::vec and glm::mat4 in my files, examples of these can be shown from lines **153-158** for cube positions and **257-265** for calculating model matrix in **coursework.cpp** |
| LO2: Application compiles and runs without alterations to the source code of CMake file. | A screenshot of a video game  AI-generated content may be incorrect. |
| LO3: Implementation of shaders to apply appropriate textures to objects. | I have implemented shaders such as diffuse textures and specular in multiple objects, examples are from lines **104-108** in **coursework.cpp** |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | The basic use of translation, rotation and scaling can be found in my world creation. Examples are from lines 211-255 and 257-260 in **coursework.cpp** |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | The implementation of glm library functions can be found in lines **263-264** in **coursework.cpp** and 10-18 in **camera.cpp** |
| LO2: 3D virtual world has been created using instances of a single object type. | Examples of this can be found from the file **vertexShader.glsl** and the creation of the objects from lines **268-279** in **coursework.cpp** |
| LO3: Use of shaders to apply dynamic lighting from point light sources | My world contains 5 point light sources and these can be found in lines 123-144 in coursework.cpp. The shaders used to apply can be found in the vertexShader.glsl and fragmentShader.glsl files. |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. | My calculation functions can be found in **maths.cpp** and **maths.hpp.** |
| LO2: 3D world created using multiple object types. | I have created multiple object types such as cubes, teapot, wall and floor in the 3D world. These can be found in the assets folder cube.obj, wall.obj… The creation of objects can be found in the **coursework.cpp** file. |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | I have implemented keyboard and mouse navigation inputs and code for this can be found in **coursework.cpp lines 300-332**. |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. | I have used different coloured light sources that can be seen in the world, there are 5-point light sources. This can be found in **coursework.cpp line 123-144**. |
| 72 75, 78 | LO1: Implementation of students own functions to replace glm functions (e.g., glm::length(), glm::dot(), glm::cross() etc.). | I have implemented some functions to replace the glm function in the **maths.hpp** and **maths.cpp** files. |
| LO1: Implementation of quaternions to calculate rotation matrix. | The quaternions can be found in the files maths.cpp and maths.hpp. These 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.). |  |
| LO3: Appropriate implementation of normal and specular maps. | I have implemented the normal and specular this can be found in **fragmentshader.glsl** |
| 85, 90, 100 | LO1: Use of quaternions to calculate view matrix. | I have used quaternions in **camera.cpp** to calculate the view matrix. **Lines 9-46.** |
| LO1: Use of SLERP to smooth out changes in camera direction. | The use of SLERP can be found in **maths.cpp** **lines 79-104.** |
| 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. |  |