# Computer Graphics Coursework – Self Assessment Document

**Name:** *Michal Wasik* **ID number:** *23619282*

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 | Lines 240-256 for vectors and more below for different objects In coursework.cpp  Lines 69-82 in Maths.cpp file uses matrices in my own functions with one example below |
| LO2: Application compiles and runs without alterations to the source code of CMake file. | Screenshot of game running |
| LO3: Implementation of shaders to apply appropriate textures to objects. | Below my code for my vertexShader class in the vertexShader.glsl file |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | Lines 468 – 480 in coursework.cpp file showing translation, rotation and scaling |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | Lines 9 – 19 in Camera.cpp showing view and projection matrices calculated using glm functions |
| LO2: 3D virtual world has been created using instances of a single object type. | Lines 245 – 256 in coursework.cpp showing one of many object types I used in my scene below in the code |
| LO3: Use of shaders to apply dynamic lighting from point light sources | Point light added at Line 387 in coursework.cpp |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. | My camera.cpp file shows my using my own function where I changed the glm library functions to my own on lines 9 - 19 |
| LO2: 3D world created using multiple object types. | Other objects types can be seen at several instances such as at lines 240 – 256, 260-287, 290-304, 316-339 and 343-357 in the coursework.cpp file |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | Function at the bottom of the coursework.cpp file for keyboards and mouse Inputs |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. | Different types of light sources added such as point and spot light as seen in the coursework.cpp file at lines 387 – 398 |
| 72 75, 78 | LO1: Implementation of students own functions to replace glm functions (e.g., glm::length(), glm::dot(), glm::cross() etc.). | My whole maths.hpp and .cpp files with several functions used in project    Hope I haven’t missed any |
| LO1: Implementation of quaternions to calculate rotation matrix. | Quaternion rotation matrix used in my project in maths.cpp and lines 127 – 147 |
| 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.). | Placed pressure plates in my environment which controls a discoball in the sky if camera/player is within a certain distance of it i.e. standing on it |
| LO3: Appropriate implementation of normal and specular maps. | Normal and specular maps used in the coursework.cpp used for each object in the environment with one example below at line 241-243 |
| 85, 90, 100 | LO1: Use of quaternions to calculate view matrix. | Quaternions are used to calculate view matrix in the camera.cpp file at lines 29- 49 |
| LO1: Use of SLERP to smooth out changes in camera direction. | SLERP use shown in screenshot before and Is in my maths.cpp file at the lines 162-188 |
| LO2: Implementation of a third person camera with the ability to switch between first and third period view. | Third person camera added with button 1 and 2 toggling between first and third person camera, with the ability to adjust the camera with 3 and 4  Lines camera.cpp 51-82, coursework.cpp 443-462 & 602-607 respectively |
| 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.). | I’ve set the cameras eye Y coordinate at a constant 1.0f with prevents the camera/player flying around at line 579    Only way to change is by jumping with space thanks to the code shown below at lines 610- 627 |
| LO3: Use of shaders to apply parameter driven effects within the scene, e.g., light properties controlled using camera/character position. | :( |