# Computer Graphics Coursework – Self Assessment Document

**Name:** Sabeel Khan **ID number:** *23640569*

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 | Coursework.cpp, lines 23-29 use of glm:: vec3 in the object structure, lines 31-40 use of glm::vec3 in Light structure. Line 278 for multiple vector initialisations glm::vec3 positions[] = {, lines 438-442 using and combining matrices to compute a MVP Matrix |
| LO2: Application compiles and runs without alterations to the source code of CMake file. |  |
| LO3: Implementation of shaders to apply appropriate textures to objects. | Coursework.cpp line 239 shader compilation, lines 241-270 loading and binding textures, lines 447 to 465 applying the right texture depending on object type.  vertexShader.glsl receives and handles the UV coordinates, and outputting UV coordinates to the fragment shader  fragmentShader.glsl receives the UV coordinates, declares the 2d texture uniform on line 23 and samples the texture using the UV and outputs the final colours. |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | Courswork.cpp lines 438 – 444 and lines 476 – 482.`z |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | Please see LO1 for 62 – 68 Mark |
| LO2: 3D virtual world has been created using instances of a single object type. | Courswork.cpp lines 278 – 300 and lines 435 to 465 |
| LO3: Use of shaders to apply dynamic lighting from point light sources | Coursework.cpp lines 31-40, lines 322 – 346, lines 406 – 425 and lines 473 to 489  vertexShader.glsl line 6 and line 11 and lines 24-25  fragmentShader.glsl lines 23 – 32 and lines 50-56 and lines 60 - 85 |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. | Maths.cpp lines 46-70  Used in Camera.cpp lines 12 - 18 |
| LO2: 3D world created using multiple object types. | Coursework.cpp lines 99 – 102 and lines 302 – 308 and lines 457 – 460 |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | Coursework.cpp lines 511 – 541  Camera.cpp & Camera.hpp |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. | Coursework.cpp lines 333 – 358 and lines 405 – 418 and lines 472 – 488  FragmentShader.glsl lines 89 - 137 |
| 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. |  |
| 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. |  |
| 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. |  |