# Computer Graphics Coursework – Self Assessment Document

**Name:** *FengZhu* **ID number:** *21383082*

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 | 28.8 |
| 2. Develop a 3D graphics application (30%) | 85 | 25.5 |
| 3. Write shader code (30%) | 100 | 30 |
|  | Total | 84.3 |

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 | Use different translation matrix to move rocks(coursework.cpp)  Use vector operation when constructing camera coordinate space.(camera.cpp) |
| 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 several shaders to render rocks、man、sphere...use appropriate textures.(vertexShader.glsl/fragmentShader.glsl...) |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | Use translation、rotation and scaling matrix when rendering moving point light sphere.(coursework.cpp) |
| LO1: Implementation of glm library functions for calculating view and projection matrices. | Implemented custom lookAt and perspective function. (maths.cpp) |
| LO2: 3D virtual world has been created using instances of a single object type. | In my virtual world there are multiple objects such as rocks、terrain、sky、man and so on.(coursework.cpp) |
| LO3: Use of shaders to apply dynamic lighting from point light sources | Implemented a rotating point light.(light.cpp/sphere.cpp/coursework.cpp) |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. | Implemented custom lookAt and perspective function. (maths.hpp/maths.cpp) |
| LO2: 3D world created using multiple object types. | In my virtual world there are multiple objects such as rocks、terrain、sky、man and so on.(coursework.cpp) |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. | Users can press W/A/S/D or mouse scroll to move camera.  Use left mouse to rotate.(camera.cpp) |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. | Implemented a rotating point light.  Implemented a directional light(Users can press +/- to change direction)  (light.cpp/sphere.cpp/coursework.cpp) |
| 72 75, 78 | LO1: Implementation of students own functions to replace glm functions (e.g., glm::length(), glm::dot(), glm::cross() etc.). | Implemented custom cross and rotate function.(maths.cpp) |
| LO1: Implementation of quaternions to calculate rotation matrix. | Not implemented. |
| 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.). | Users can press p to start/stop the point light moveing.  Users can press +/- to change direction light direction.(coursework.cpp) |
| LO3: Appropriate implementation of normal and specular maps. | Render a sphere using normal and specular maps.(sphere.cpp)  Render a man using specular map.(model.cpp) |
| 85, 90, 100 | LO1: Use of quaternions to calculate view matrix. | Not Implemented. |
| LO1: Use of SLERP to smooth out changes in camera direction. | Not Implemented. |
| LO2: Implementation of a third person camera with the ability to switch between first and third period view. | Not Implemented. |
| 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.). | Users can pres m to change camera mode.(free camera or constrained camera which check collision with the terrain)(camera.cpp/terrain.cpp) |
| LO3: Use of shaders to apply parameter driven effects within the scene, e.g., light properties controlled using camera/character position. | Users can press c to apply a random color to the point light.(coursework.cpp)  Users can press +/- to change direction light direction.(coursework.cpp) |