# Computer Graphics Coursework – Self Assessment Document

**Name:** *Alicja Starzyk* **ID number:** *22505222*

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 used basic matrix and vectors using the glm library which I linked at the top(line 14,15) of the coursework.cpp.    I used vectors and matrix logic within the maths class.      In order to achieve one of the later on requirements of translating, rotating and scaling it was easier to make a method that calculates all of the matrixes needed for the transformations within maths.cpp within each of the brackets.  glm::vec3 is a 3D vector used with specific x,y,z components. |
| LO2: Application compiles and runs without alterations to the source code of CMake file. | The code runs successfully, I used GitHub to clone and fork the repository to my own Git and then after downloading the files I used CMake to choose the source code, that being Computer-Graphics-Coursework and building the binaries into a build folder I added. The configuration was successful and I was able to generate it successfully. When running my program it is successful with no errors. |
| LO3: Implementation of shaders to apply appropriate textures to objects. | I specified the texture coordinated with in the coursework.cpp file as seen above.    On lines 106 I complied the shader programs, the vertex shader and fragment shader. From line 122 onwards I chose the image/ texture that will be on said triangle. Since it was PNG file that was in the assets file I had to change the code 133 that specifies the 2d texture as png uses the RGBA colour model.  Below are screen shots of the code within the 2 shader files that applied the texture. |
| 52, 55, 58 | LO1: Basic use of translation, rotation and scaling transformations. | I used basic transformation of all the mentioned movements in the coursework.cpp file to animate the triangle. The glfwGetTime() allows for the triangle to rotate a full 360 every 2 seconds.      The objects position is moved by altering the shader by the transformation matric in coursework.cpp    The transformation now calculates the composite transformation matrix to have all the basic transformations work at once which was achieved on line 184 of the coursework.cpp |
| LO1: Implementation of glm library functions for calculating view and projection matrices. |  |
| LO2: 3D virtual world has been created using instances of a single object type. |  |
| LO3: Use of shaders to apply dynamic lighting from point light sources |  |
| 62, 65, 68 | LO1: Implementation of students own functions for calculating view and projection matrices. |  |
| LO2: 3D world created using multiple object types. |  |
| LO2: Users can navigate the virtual world using keyboard and mouse inputs. |  |
| LO3: Use of shaders to apply dynamic lighting from different types of light sources. |  |
| 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. |  |