MSc: Computational & Software Techniques in Engineering
Visualisation Assignment 2020

The assignment consists of two parts:

Part A The task is to build one Qt or Web based OpenGL/WebGL program containing a menu (or appropriate widget) with three items that allows the user to do the following:

1) Enter data using a dialog box (or similar widget that you would like to use) selected from a menu item entitled **"line rotation"** for the definition of a line in the form **b**+**μd** and an angle of rotation, *alpha*. The code should then display the line and perform the rotation of the cube model about the line through the specified angle.

- The 4×4 matrix representing the rotation by an angle alpha about a normalized axis (x,y,z) is:

$$\mathrm{M}_{\mathrm{rotation}} = \begin{bmatrix} (1-\cos\alpha)x\,x + \cos\alpha & (1-\cos\alpha)x\,y - z\sin\alpha & (1-\cos\alpha)z\,x + y\sin\alpha & 0 \\ (1-\cos\alpha)x\,y + z\sin\alpha & (1-\cos\alpha)y\,y + \cos\alpha & (1-\cos\alpha)y\,z - x\sin\alpha & 0 \\ (1-\cos\alpha)z\,x - y\sin\alpha & (1-\cos\alpha)y\,z + x\sin\alpha & (1-\cos\alpha)z\,z + \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- The 4×4 matrix representing the translation by a vector t is:

$$\mathrm{M}_{\mathrm{translation}} = \begin{bmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Once you have the data for **b** and **d** (each 3 floats) you will need to generate two translation matrices, with the **t** vector as the **-b** vector and the second, with the **t** vector as the **+b** vector.
You need to normalize the **d** vector to get the **(x,y,z)** used in the rotation matrix above and then generate the matrix to get the final rotation matrix.

2) Enter data using a dialog box (or similar) selected from a menu item entitled "view position" for an eye position **(eye: eye.x, eye.y, eye.z)** and a point you are looking at **(point: point.x, point.y, point.z)** giving the direction of view**.** On entering this data your program should redisplay the model as seen from this vantage point.

   The glm method you need to use is glm::lookAt:

```
view = glm::lookAt(vec3(eyex,eyey,eyez),
vec3(point.x,point.y,point.z),
vec3(0.0f,1.0f,0.0f));
```

3) Include a menu item entitled "**default position**" that returns the cube to the default viewing position with the eye at 0,0,2 looking towards the origin. This should be the view of the cube that first appears on the screen when the program is run.

Part B: Using the title "***GPU shaders for advanced visualisation***" give a summary account of the uses of shader technology in one or more aspects of computational engineering. **Max – 4 pages**

Part A: Working Code + documentation: **70%**
Part B: Account of GPU shaders + class diagram: **30%**

Deliverables: by TBC:

1) A zip file containing the source code, Qt project files and shader programs for Part A.

2) A class diagram illustrating the relationships between the classes in your application. This includes the existing classes and any that are generated when creating the application.

3) The summary account from Part B. Please include the class diagram from 2) in the same word/pdf document as the answer to part 3.