Further information concerning the assignment (Part A)

The starting point for the assignment is the code we looked at and modified for the visualization of the cube (Example 3 from the OpenGL examples folder of the Monday directory). Basically, you have the cube model program converted to a Qt widget application from which you can add the necessary widgets to allow the user to input the relevant data (in the class we did an example where we added three menu items to rotate the cube about the three axes and return to the default view.)

You will need two methods added to the **Scene** class, one for rotation and one for the view. These methods should take in the relevant data for the rotation and the view.

```
virtual void updateView(float eX, float eY, float eZ, float
directX, float directY, float directZ) = 0;

virtual void rotateModel(float bX, float bY, float bZ,
float dX, float dY, float dZ, float phi) = 0;
```

These will now need to be added and implemented in the **Scenebasic** class which inherits **Scene**.

The implementation of **rotateModel** should compute the matrix

$$\mathbf{M2}_{\mathbf{translation}} * \mathbf{M}_{\mathbf{rotation}} * \mathbf{M1}_{\mathbf{translation}}$$

You can then set the model variable, **model**, equal to this matrix.

For the view operation you take in the eye position (eye) and direction of view (direct). You can then set the view variable, **view**, equal to the result of calling **glm::LookAt**

You need to get the rotation and view parameters from the user into the **MainWindow** class and then you can pass this data into one of two methods you can create in your **MainView** class for accepting this data (there is a pointer to the view in the **MainWindow** class which you can use to do this)

Once the data has been passed into the appropriate method of the **MainView** class you can then in these methods call the **rotateModel** or **updateView** methods using the scene pointer.

```
scene->rotateModel(bx,by,bz,dx,dy,dz,phi)
```

or

```
scene->updateView(ex,ey,ez,dx,dy,dz)
```

For displaying the line of rotation in addition to the cube there are two possible approaches. The first is to use two VAO's, one attached to the cube data and one for the line data (two points). By binding first the cube VAO and drawing and then binding the line VAO and drawing you will be able to display both models on the screen at once.

The alternative method is to use the GL function `glBufferSubData`. This allows you to store both the cube data and the line data in the same VBO buffer (cube data first then line data) and hence use only one VAO (an index is set for the offset to each data set within the buffer). You can then display each model using the appropriate offset.