**COMPUTER GRAPHICS**

**SUMMATIVE ASSIGNMENT**

# Question 1a

We represent a 3D mesh as a collection of triangles. This is because we know that 3 points will always form a triangle, even in 3D space. However, this means that each point on the mesh will correspond to multiple triangles (presuming that the mesh is connected). Therefore, if we just list the vertex data in the vertex buffer object (VBO) then we will need to repeat the data for each position for each triangle. By using an index buffer object (IBO), we instead define the data for each vertex only once. We then use the IBO to reference these position for each triangle, greatly reducing the amount of memory needed to store the VBO.

# Question 1b

This fragment shader will not perform spot lighting correctly, and will instead colour every pixel in the mesh blue (#0000FF). *gl\_FragColour* determines the colour (and alpha) to use for each pixel. Because this is set to a constant *vec4(0.0, 0.0, 1.0, 1.0)* then the blue and alpha components will always be 1.0 (and the red and green components 0.0).

To support point lighting, we must first send the colour of each vertex from the vertex shader, to the fragment shader as the interpolated colour. To do this we use a varying in both the vertex and fragment shader and pass the colour of the vertex into the varying in the vertex shader. The varying in the fragment shader will then contain the interpolated colour for the given fragment (pixel). However, this just correctly sets up the colour, but does not actually support point lighting.

To support point lighting the fragment shader will need several other varyings given to it from the vertex shader. These are the VertexPosition and NormalMatrix, which are interpolated per fragment. We could also pass in uniforms to change the light position, light colour, light intensity, etc., but these could be hardcoded as constants and point lighting would still be possible (but not configurable).

# Question 1c

TODO

# Question 1d

i)

TODO

ii)

TODO