# OpenGL Advanced:
## GLSL Shaders

CSC4029Z Assignment 2

## 1 Introduction

This assignment is designed to extend your OpenGL skills. You **should reuse the original code you submitted for Assignment 1**. You are asked to demonstrate your competence with OpenGL shaders, lighting and texturing techniques.

## 2 Requirements

Take the setup provided by the first practical: with the loaded object centred on the world space origin. Note that for this assignment you only need to support a **single model**.

You should support (1) scaling of the model along each world space axis, as well as (2) x/y/z rotation (all about the origin, where the model is centred – the model should remain centred on (0,0,0)). You do not need to support other modelling transformations. These transformations are sufficient demonstrate that your normals have been transformed correctly from model space into camera space (required for lighting).

The camera must be constrained to look at the world space centre but must also be able to orbit around this fixed point, based on key press input. This will allow you to circle the view around the object and see it from different angles as the light falls on the surface. Next, you will need to add some additional steps, as outlined below.

### 2.1 Phong-Shading (per fragment lighting)

You are required to add light sources to the world (at least 2). Each of these needs to have different colour values to show up effectively on your object. Per-fragment lighting is required for this and the accumulated value of all the lights falling on a fragment will determine its final colour. While the objects will not have a colour assigned, the texture values (see later) will effectively provide a 'material' colour per fragment – this will then be modified by the colour of the light falling onto it, as determined by the Phong Reflection Model (PRM). You can assume ONE colour per light (so ambient, diffuse and specular light colour is all the same). Experiment to see what gives the most pleasing appearance (we need to see clear diffuse/specular interactions on the model surface).

### 2.2 Shader files

You will need to update both the vertex and fragment shader files for this assignment. This is where you will put the code necessary to implement Phong shading and texturing. You may have additional shader files for your solution, if this makes sense for your implementation (it is not necessary though).

### 2.3 Moving Lights

The lights that are added to the scene need to move: this can either be done by transforming them individually (using for example key presses to move them) or having them continuously orbit in either a clockwise or anti-clockwise direction around the model (make sure they do not move inside the model!). This provides another method to show how the light falls on the object's surface. We simply treat the point light source positions as regular 3D points that can be transformed (via a 'modelling' transformation) from their

initial positions. *These new positions must be mapped to camera space for lighting calculations*. Remember: lighting calculations are done in camera space.

### 2.4 Texturing

You need to apply a texture to the surface. To ease the task of loading a texture image file in C++, we suggest that you use the freely available single header file stb_image.h. You can find this header file at https://github.com/nothings/stb. You can assume that the OBJ model you load contains texture coordinates if it does not, you should display a message indicating this and simply ignore the texturing stage. For Python, please refer to the original assignment and the resources noted there. The main difficulty with textures is ensuring the bytes are loaded correctly (the images are generally not in a simple format, and you need a 2D array of pixel data packed into memory correctly). Any good OpenGL support API will provide support for image loading.

### 2.5 Bump Mapping or Advanced texturing feature

The final 15% available is for adding bump-mapping to the surface of the object **OR** adding some alternative feature, such as animated textures.
Bump mapping is achieved through shader techniques that perturb the surface of the model. This requires not only the texture file but a secondary image that is a bump-map. Note: depending on the format of the bump map, you may need (or choose) to compute the normal at a texel sample using "bitangents" etc. YOU MUST CITE THE CODE YOU USED. *We did not cover this, and as usual, we only award credit for your own work.* Simpler bump maps directly store a normal vector (as 3 floats per pixel, instead of RGB) and do not required these complex calculations. Please note what you implement (bump mapping or something else) in the README and be clear to acknowledge other sources where appropriate.

## 3 Marking

This practical should be accompanied by compilation/installation instructions for either Windows 10/11 or Ubuntu/Linux. Failure to ensure this, will lead to a 30% penalty up front. These instructions should be contained in a README file, included as part of the submission. Please also note any additional functionality you may have built, and be sure to include citations for all sources and indicate where this code was used. Most marks are allocated to the basic Phong shader which is easy to implement from the Slides and other (properly acknowledged) sources.

## 3.1 Marking Guide

| Feature | Marks |
|---|---|
| **Phong Shading** | **18** |
| - Specular highlight present and moves with viewpoint | 0 - 5 |
| - Diffuse lighting, shadows fixed relative to light | 0 - 5 |
| - Scaling on axis - lighting remains correct | 0 - 3 |
| - Rotation – lighting correct | 0 - 5 |
| **Shader Files provided** | **5** |
| **Lights** | **12** |
| - Lights move | 0 – 6 |
| - Two lights, different colours | 0 – 6 |
| **Texturing** | **7** |
| - Texture appear on model surface | 0 - 3 |
| - Texture correctly mapped | 0 - 2 |

| | |
|---|---|
| (remains fixed under model transformations) | |
| - Texture lookup correct in GLSL | 0 - 2 |
| **Bump Mapping/Advanced feature** | **8** |
| - Visible point light sources | 0 - 2 |
| - Others | variable |

## 4 Submission Deadline

Hand in the assignment by 10:00AM, 26 May 2023.

## 5 Useful Resources

1. https://learnopengl.com/
2. http://www.opengl-tutorial.org/