COSE436 Interactive Visualization (Fall 2023)
Instructor: Prof. Won-Ki Jeong
Due date: Nov 5, 2023, 11:59 pm.

Assignment 2: Triangle Mesh Viewer (100 pts)

In this assignment, you will implement an OpenGL triangle mesh viewer based on the basic virtual trackball viewer implemented in assignment 1. Below is the list of required functions you need to implement.

1. Triangular mesh file I/O (40 pts)

Several simple and complex triangular mesh data are given for this assignment (bunny, dragon, fandisk under 'mesh-data' directory). First step for the assignment is provide I/O and data management class for triangular meshes. The input data format name is 'off', ASCII text file containing the list of vertex coordinates and per-face connectivity information. The format of off file is as follows:

```
OFF                 : first line contains the string OFF only
#v #f 0             : total number of vertices, faces, and 0
vx1 vy1 vz1         : x/y/z coordinate for vertex 1
vx2 vy2 vz2         : x/y/z coordinate for vertex 2
...
#v_f1 f1v1 f1v2 f1v3   : # of total vertices and each index for face 1
#v_f2 f2v1 f2v2 f2v3   : # of total vertices and each index for face 2
...
```

All the example files provided for this assignment will be triangular meshes, so the total number of vertices per face is always 3. Below is an example of an off file containing 34835 vertices and 69473 triangles:

```
OFF
34835 69473 0
-0.0378297 0.12794 0.00447467
-0.0447794 0.128887 0.00190497
-0.0680095 0.151244 0.0371953
...
3 20463 20462 19669
3 8845 8935 14299
3 15446 12949 4984
```

Once you read an 'off' file from the disk, you should store the mesh in memory using three arrays – vertex coordinates, vertex normals, and indices. Use these arrays as input to your **buffer objects (vertex (VBO) and index buffer objects (IBO))** and use **glDrawElements()** to render them. Note that per-vertex normal is not given in off file, so you need to compute it on your own when the mesh is loaded.

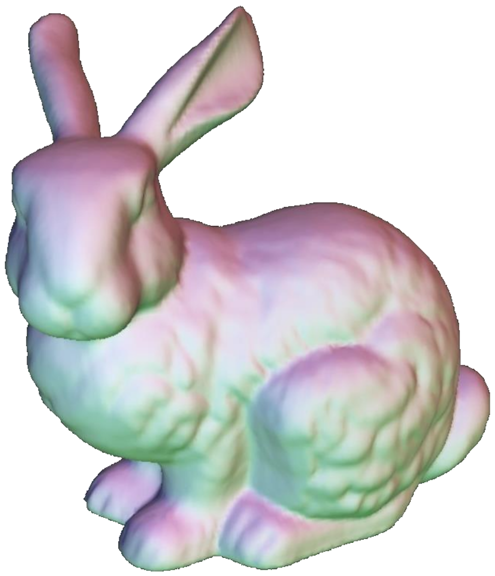

Figure 1. Rendering example of bunny triangluar mesh. In this example, two light sources are used (purple on top, green on bottom), and only diffuse term is used for smoooth shading.

2. Per-fragment smooth shading using Phong illumination model (60 pts)

You need to implement per-fragment shading technique using vertex and fragment shader. You need to implement the Phong illumination model. I provide a part of Phong shader source code in my lecture notes, so you can freely use it as a starter.

You are required to create multiple light sources in different locations and colors, which are pre-defined. However, you should implement functions to change diffusion, ambient, and specular parameters interactively (k_a, k_d, k_s in Phong equation) using the keyboard, as follows:

```
s      : switch between lights.
1 or 3 : decrease/increase diffusion parameter (k_d)
4 or 6 : decrease/increase ambient parameter (k_a)
7 or 9 : decrease/increase specular parameter (k_s)
- or + : decrease/increase shininess parameter (alpha in
         Phong equation)
```

Make sure k_a, k_d, and k_s are between `0` and `1`.

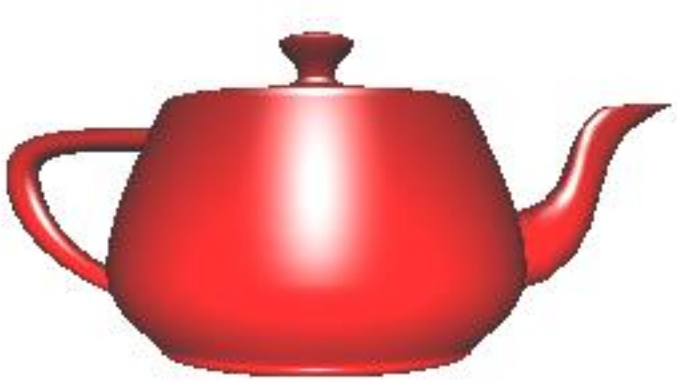

Figure 2. Phong shading of the Utah teapot model using a single light source. Note the highlight reflection on the surface.

3. Submission

You should modify the skeleton code, and submit **main.cpp, fshader.frag and vshader.vert only** in a single zip file. The code must be compiled without additional external library other than the provided ones. Make sure your code reads in mesh data from the same relative directory location as given in the skeleton file (i.e., ‘build’ and ‘mesh-data’ directories are at the same level, and the source files (.cpp, shaders) are located one level higher than them in the directory structure).

Good luck and have fun!