COSE436 Interactive Visualization (Fall 2024)
Instructor: Prof. Won-Ki Jeong
Due date: Nov 3, 2024, 11:59 pm.

Assignment 2: GLSL Shaders (100 pts)

In this assignment, you will learn how to use GLSL shaders for surface rendering. Below is the list of required functions you need to implement.

1. Triangular mesh file I/O and Phong Illumination Model

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.

Once triangular mesh I/O is iplemented, You need to implement per-fragment shading (Phong illumination model) using vertex and fragment shaders. I provide a part of Phong illumination model 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`.

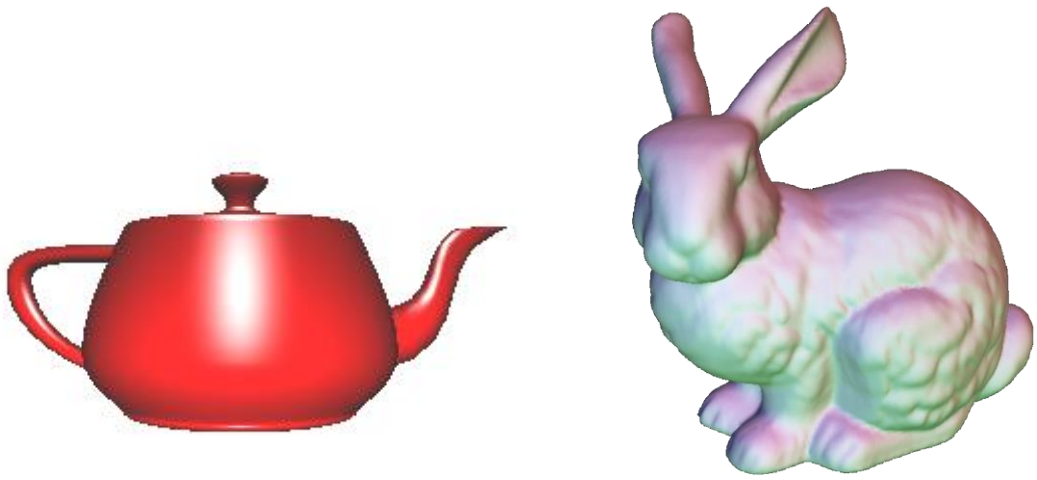

Figure 1. Left: Phong shading of the Utah teapot model using a single light source. Note the highlight reflection on the surface. Right: Diffuse rendering of bunny triangluar mesh using two light sources (purple on top, green on bottom).

2. Cartoon shader

In this assignment, you need to implement piecewise-linear shading that mimics cartoon-like rendering with silhouette as shown in Figure 2.

Figure 2. Example of cartoon shading with silhouette of Torus and Teapot models

The above shading effect was generated via two render passes – a silhouette render pass and a cartoon render pass.

Silhouette render pass Cartoon render pass

Figure 3. Silhouette & Cartoon Render passes

2.1 Silhouette render pass

To render the silhouette of the object (as shown Figure 3 left), you can draw the back-face polygons of the slightly larger object in black and then draw the front-face polygons of the original sized object in white. To make the object slightly larger, you can move each vertex along its vertex normal in your vertex shader. The silhouette thickness can be controlled by how much each vertex is moving along the vertex normal direction. Implement a keyboard callback so that pressing + increases (- decreases) the silhouette thickness interactively.

You can render front-faced or back-faced triangles only by using glCullFace().

2.2 Cartoon render pass

For this effect, you need to compute the angle between the surface normal and

the light direction, and then assign a constant color value for a certain angle range (the higher the angle, the darker the color). The number of ranges can be interactively chosen using the number key (from 2 to 9). Then your shader will assign different shades of the color.

You already know how to compute per-fragment surface normal from the previous phong shader, so this will be easy to implement. Make sure that you normalize light direction and normal vectors in your fragment shader.

2.3 Combine silhouette & cartoon shading

Since you combine silhouette rendering with cartoon rendering, you need to draw only back-faced polygons with black color (first render pass), and then front-faced polygons can be drawn using cartoon rendering (second render pass).

You need to make two shader programs (one per each render pass), and render the object twice by switching between the programs.

3. Etc.

Test your code with glut 3D models first (glutSolidTorus(), glutSolidTeapot(), etc). Note that the vertex normal of glutSolidTeapot() is pointing inside the model (so you should use glFrontFace(GL_CW) to invert the orientation). Once it works fine, then try with triangular meshes (make sure per-vertex normal is calculated correctly).

Implement keyboard callback 'p' so that by pressing 'p' toggles between phong shader and cartoon shader. Note that some keys are assigned to both phong and cartoon shader (such as numeric keys), so you need to distinguish the keyboard callback correctly depending on the current rendering mode.

4. Submission

You should modify the skeleton code, and submit **main.cpp, *.frag and *.vert** 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!

Score breakdown (total 100)

1. Triangular mesh I/O (15)
   1.1. Loading triangular mesh and passing the data to GPU using buffers (10)
   1.2. Calculate per-vertex normal correctly (5)
2. Phong shader (25)
   2.1. Correctness of per-pixel Phong illumination implementation (15)
   2.2. Ability to change diffuse/ambient/specular/shininess parameters interactively (10)
3. Cartoon shader (50)
   3.1. Correctness of silhouette rendering (15)
   3.2. Ability to interactively change the silhouette thickness by pressing + or - (10)
   3.3. Correctness of cartoon rendering (15)
   3.4. Ability to change cartoon shading ranges interactively using number keys (10)
4. Common (10)
   4.1. Ability to correctly render triangular meshes with shaders (5)
   4.2. Toggle between phong and cartoon shader by pressing p (5)