Homework - 10

CS594

Introduction to lighting in OpenGL

While Homework-9 introduced different types of light sources and the visual effects which may be achieved therefrom, this assignment discusses how basic lighting calculations are performed in order to simulate these light sources and their interaction with the environment. In particular, different components of lighting such as ambient, diffuse and specular are covered. More advanced topics such as shadows are deferred until later assignments. You may read more on ambient lighting, diffuse reflection and specular reflection. Click on the link below to read the tutorial and complete the following exercise.

Basic lighting: Shows how to perform lighting calculations in OpenGL for simulating diffuse, specular and ambient lighting. Run and understand the code for "Tutorial 8: Basic shading". The type of light source simulated here is a point-light, which radiates equal amount of light in all the directions, like a candle. Most of the lighting computations are performed in the two shaders, viz., the vertex shader and the fragment shader. Recall the sequence of execution of shaders from the graphics pipeline. A vertex shader performs calculations for each vertex of the mesh. The data thus computed is passed on to the fragment shader, which performs calculations for each pixel on the screen. You may refer to previous tutorials to refresh how data is passed from C++ OpenGL code to the vertex shader and from vertex shader to the fragment shader.

Exercise: Edit the code of "Tutorial 8: Basic shading" to change the type of light source from point-light to a spot-light. A spot-light is a cone of light emanating from the apex of the cone. While a point-light is defined simply by its position, color and strength, to define a spot-light, one additionally

needs parameters such as the direction and the angle of cone. You may go back to Homework-9 and experiment with spot-lights in Blender. There may be a number of ways of simulating spot-lights. To do so, one needs to clip the light source outside the cone. Outputs from two methods are shown in Figure 1. In (a), the clipping calculation is performed in the fragment shader, while in (b), the same is performed in the vertex shader. Notice the sharp boundary of the cone in case of (a), which is smooth in case of (b). This is because the values computed in the vertex shader are interpolated in the fragment shader. So, if there are two adjacent vertices on a mesh, one of which is within the cone of the spot-light and the other is not, then the region between the two will be smoothly interpolated in case of (b).

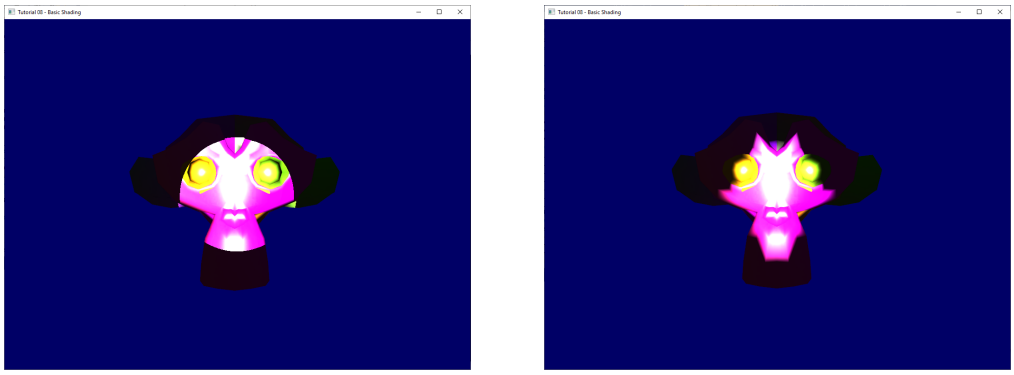

Figure 1: (a) Spotlight computation performed in fragment shader. (b) Spotlight computation performed in vertex shader.

Submit two versions of the code and output images. (i) Type-A, wherein the lighting calculation for cone-clipping is performed in the fragment shader, (ii) Type-B, wherein the lighting calculation for cone-clipping is performed in the vertex shader. Listed below are the files to be submitted for Type-A. Similarly for Type-B.

1. `.png` image which captures a screenshot of the output from Type-A code.
2. `.cpp` file for Type-A.
3. `StandardShading.fragmentshader` file for Type-A.
4. `StandardShading.vertexshader` file for Type-A.