Winter 2018: CSI4130
Assignment 3

Due: Wednesday, March 22^{nd}, 2018, 11:00 pm on Virtual Campus
University of Ottawa - Université d'Ottawa

Jochen Lang

1 Lighting and Material [10 in total]

This assignment is based on a lit boxes laboratory (lab06) which can be downloaded from Virtual Campus. You should use this code as a start either in the C++/GLUT version or in the Java/JOGL version.

You will need to find a better way to light and calculate reflection. The current solution uses a single light source. We like to use multiple light sources and a different reflection model.

1.1 More Lights and Better Spheres [3]

Integrate the sphere with subdivision from Assignment 2 into this project instead of the boxes. Also use three separate light sources: a spot light (controllable as currently), a directional light source from the center of the upper-left edge of the viewing volume, and a point light source on the upper-right rear corner of the viewing volume. The directional and point light sources need not to be under user control except for switching them on and off. A single ambient term should also be included (only one rather than one for every light source).

1.2 BRDF Reflection [3]

The Blinn-Phong reflection model currently in the code is to be extended to use parameters estimated from measurements of real materials. The model is the same as we have been using except for a normalization in the specular time. It is given as

$$c_d + c_s \left(cos(\mathbf{N} \cdot \mathbf{H}) \right)^n \frac{n+2}{2\pi}$$

The parameter of the model are n, the diffuse colour c_d and the specular colour c_s. $\mathbf{N}$ and $\mathbf{H}$ are the normal and half-way vector, respectively.

Add the ability to change the intensity of the light sources. You should be able to toggle through the light sources with i and then use the up/down arrow keys to increase/decrease the value. The light source will need numbers larger than 1.0 as the measured models use radiance and not colours normalized from 0 to 1.

1.3 Lafortune Reflection [4]

Now we add a different reflection model switching between the two with b/B. Note, that it is likely easiest to use two different programs and switch between them.

Table 1: Blinn-Phong Model Parameters. Source: Addy Ngan et al. [1]

Material	c_d			c_s			n
Blue rubber	0.0425	0.0698	0.0957	0.00533	0.00471	0.00333	43.6
Brass	0.0382	0.0272	0.0119	0.0367	0.015	0.00537	3.16e+004
Metallic-silver	0.0695	0.0628	0.0446	0.0742	0.0615	0.0412	75

The Lafortune BRDF is a reflection model that is a generalization of the Blinn-Phong BRDF. It is commonly used when measured reflection need to be fit to a compact model.

The model is calculated as

$$c_d + c_s \left(K_{xy} \left(l_x v_x + l_y v_y\right) + K_z l_z v_z\right)^n \frac{n+2}{2\pi \left(max(|K_{xy}|, |K_y|)\right)^n}$$

Here the parameters of the model are K_{xy}, K_y and n, the diffuse colour c_d and the specular colour c_s. l_{xyz} and v_{xyz} are the components of the light and view vector, respectively.

Table 2: Lafortune Model Parameters. Source: Addy Ngan et al. [1]

Material	c_d			c_s			n	K_{xy}	K_z
Blue rubber	0.0464	0.0736	0.0986	0.291	0.239	0.159	32.6	-0.635	0.44
Brass	0.0387	0.0273	0.0123	0.118	0.0479	0.0172	1.07e+004	-0.577	0.577
Metallic-silver	0.0552	0.05	0.0359	0.434	0.363	0.243	21.4	-0.587	0.559

1.4 Combination of different materials [3] (Bonus)

As a bonus question (the bonus only counts towards the assignments), generate for each vertex a weight vector to combine the three materials, e.g., a vertex may be 0.2× blue rubber plus 0.5× brass plus 0.3× metallic silver. Load these values as an extra attribute in the vertex shader and pass it on to the fragment shader (not using the rendering instance to switch between materials anymore.) Calculate these values with some form of procedure for the vertices of the sphere (e.g., based on azimuth and evaluation, using alternating stripes, or similar).

2 Submission

Feel free to add additional source files as required but please do not submit project definitions or other IDE files to Virtual Campus. The files that you submit have to be sufficient for your program.

2.1 C++ Project

Your assignment submission must consist of a *zip archive*. The following is a (partial) list of files[1] You can duplicate `material.h` for Blinn-Phong and Lafortune if you like.

Do not introduce any non-standard C/C++ features (no windows includes!).

Filename	
`common/shader.h`	Loading and compiling shaders.
`common/shader.cpp`	
`assign3/render_shape.h`	The shape base class
`assign3/shape.h`	The shape interface definition
`assign3/subdivision_sphere.h`	The sphere with subdivison
`assign3/subdivision_sphere.cpp`	
`assign3/attributes.h`	The attributes.
`assign3/attributes.cpp`	
`assign3/light.h`	
`assign3/material.h`	
`assign3/lit_spheres.cpp`	The top-level glut main program (similar to interaction).
`assign3/blinnphong.vs`	Vertex shader
`assign3/blinnphong.fs`	Fragment shader
`assign3/lafortune.vs`	Vertex shader
`assign3/lafortune.fs`	Fragment shader

2.2 Java JOGL Project

In case of JOGL, I would like to receive your assignment in a zip archive with the following layout. You can duplicate `MaterialArray.java` for Blinn-Phong and Lafortune if you like.

[1] I use the forward slash for directories but windows use is fine.

Filename	
`Shader/src/shader/Shader.java`	Loading and compiling shaders.
`Assign3/src/RenderShape.java`	The shape base class
`Assign3/src/Shape.java`	The shape interface definition
`Assign3/src/SubdivisionSphere.java`	The sphere with subdivison
`Assign3/src/Attributes.java`	
`Assign3/src/LightArray.java`	The light array with inner class Light
`Assign3/src/MaterialArray.java`	The material array class inner class Material
`Assign3/src/LitSpheres.java`	
`Assign3/src/Main.java`	Top-level window and main.
`Assign3/shader/blinnphong.vs`	Vertex shader
`Assign3/shader/blinnphong.fs`	Fragment shader
`Assign3/shader/lafortune.vs`	Vertex shader
`Assign3/shader/lafortune.fs`	Fragment shader

References

[1] A. Ngan, F. Durand, and W. Matusik, “Experimental analysis of brdf models,” in *Proceedings of the Eurographics Symposium on Rendering.* Eurographics Association, 2005, pp. 117–226.