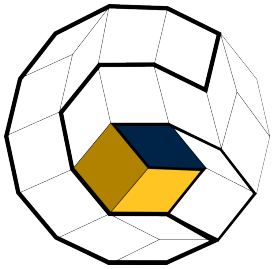
TU BRAUNSCHWEIG
Dr. Georgia Albuquerque
Institut für Computergraphik
Matthias Überheide

May 18, 2018



Real-time Computer Graphics, Summer 2018
Assignment 4

Present your solution to this exercise on Thursday, May 31, 2018.

The exercises are going to take place in the CIP pool, room G40 in Mühlenpfordstraße 23. Please make sure your solutions compile and run on the CIP pool computers. Note that you need a y-number, which can be obtained at the Gauß-IT-Zentrum, to use the computers. If for some reason you are not able to attend the exercise, you may send your solution to ecg@cg.cs.tu-bs.de instead.

This exercise and the corresponding version of the framework can be found on the lectures website (`http://www.cg.cs.tu-bs.de/teaching/lectures/ss18/ecg/`).

Theoretical Tasks

4.1 Lighting (30 Points)

$$L = M_e + M_a \cdot L_a + \sum_i (M_a \cdot L_{a,i} + M_d \cdot L_{d,i} \cdot \cos\varphi + M_s \cdot L_{s,i} \cdot \cos^n \Psi)$$

Briefly explain the variables in the equation and give an example value. Are all possible values physically plausible? Which values make sense?

4.2 Short Presentation (Extra credit 10 Points)

Prepare a 5min talk about an interesting topic related to the current lecture. Send your topic proposal via email to ecg@cg.cs.tu-bs.de at least 24 hours before the presentation.

Practical Tasks

4.3 Camera space normal shader (20 Points)

In this task you will implement a shader that visualizes the objects' normals in camera space. This helpful tool can be used later to identify problems with lighting and textures. Complete the `normal.vert` to transform the normals into camera space and render the interpolated per pixel normals in the fragment shader `normal.frag`. Note that you have to map normals $[-1, 1]^3$ to color $[0, 1]^3$.
Note that for this exercise the `OGLNormalMeshObject` will be used which is intended for the later exercises. The program will output warnings about not being able to find some of the uniforms. These warnings can be ignored here.

4.4 Per Pixel Lighting (40 Points)

In this task you will implement per pixel phong lighting. The necessary pipeline to upload lightsources and materials to the GPU is provided. Make yourself familiar with the code providing these uniforms (classes `OGLMaterial`, `OGLLightSource` and `OGLNormalMeshObject`). Implement the missing parts of `multiLight.vert` and `multiLight.frag`.

The vertex shader file defines a structure for light sources. `LightSource` encloses all important parameters of a point light source, being its position and the color value for the ambient, diffuse and specular light components. The very first thing to do in your shader code is to define output variables, which will be passed to the fragment program. Follow the steps to provide the necessary vectors. Note that the given light source positions are already in camera space as the camera is used as a root node in the scene.
With the vertex shader set up you have to complete the fragment shader. The material is again available through a struct. Define compatible input values and complete the steps in the code. Using all given input vectors and colors, compute the different color components for this fragment, being an ambient component, a diffuse component and a specular component. Finally combine these color components for the final fragment color and assign it to the provided output variable `color`. Note that the color has a fourth component, the alpha value. Since we do not use transparency, simply set this value to 1.0. Do not forget to remove the the last line which overrides the color output with a fixed debug color.

4.5 Animated Lights (10 Points)

Create a small animation with a light source. Start by rotating the light source around the scene. Try to find other interesting animations.

4.6 Coding (Extra credit 10 Points)

Present one of the following:

- a bug in the framework
- a helpful test case to solve an exercise
- the implementation of an additional feature
- a nice demo using the current framework

Note that a short documentation is required and code has to be handed in. Please report bugs immediately (via email) so they can be fixed as soon as possible.