Introduction to Computer Graphics
Assignment 6 – Texturing and Lighting

Handout date: 10.04.2018

Submission deadline: 19.04.2018, 13:00 h

Late submissions are not accepted

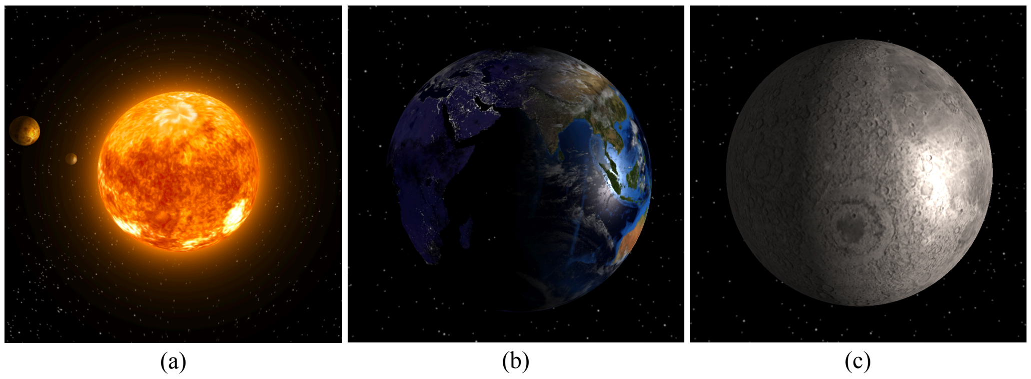

Figure 1: (a) The Sun with a glow implemented as a billboard, (b) the Earth rendered with a mixture of four textures and Phong lighting model and (c) the Moon rendered with Phong lighting model.

In this assignment, you will use the technique called *billboarding* to mimic the Sun's glow (see Figure 1 (a)). Furthermore, you will render all the planets and the space ship using the *Phong lighting model* (using the planet textures as the material parameters) to get nice specular effects (see Figure 1 (b) and (c)), and finally you will visualize the day and night appearance of the Earth as well as the clouds and water surface specularity (see Figure 1 (b)). Optionally, you can implement slight oscillations of the Sun's surface geometry so as to achieve a flow-like effect.

The framework code for this assignment extends the one from last week. "`todo`" comments have been inserted in `solar_viewer.cpp` to indicate where you need to add your implementations. If you already set up a GitHub repository to collaborate with your fellow group members, you can just copy the "`todo`" comments from `solar_viewer.cpp` over to your repository (or just note where your implementation needs to go and get started).

Overview

This exercise consists of 4 parts:

1. rendering the billboard
2. rendering the planets using the Phong lighting model
3. combining multiple textures to get day/night, clouds and water specularity effects for the Earth.
4. (optional) generating the Sun's surface geometry oscillations

Only the 2nd and 3rd parts depend on each other: the Earth shader builds on the Phong lighting model to add some special effects. Therefore, we suggest first implementing the Phong lighting model and only then moving on to texture the Earth.

Billboarding

The objective is to mimic the glow of the Sun. We will use the technique called *billboarding*, which involves placing a planar object in the scene with a given texture and transforming it to always be oriented perpendicularly to the eye. In our case, the planar object is a square consisting of two triangles; this is already provided to you (see files `billboard.[h|cpp]`) and variable `Solar_viewer::sunglow_`.

Your task is to create the image of a circle whose opacity attenuates with increasing distance from its center until it is fully transparent. Please read the `todo` notes and fill in the function `Texture::createSunBillboardTexture()`. Play around with the colors and attenuation factor until you get a visually pleasing result. The Figure 2 depicts an example of what happens when the opacity attenuation is not strong enough. Please note that each pixel $\mathbf{p_i}$ in the texture is defined as 4D vector $\mathbf{p_i} = [R, G, B, A]$, where the first three dimensions specify the RGB color and the last one the opacity $A \in [0, 255]$.

Furthermore, you need to transform the billboard according to the current position of the eye. The Figure 3 shows a default orientation of the billboard as well as the required orientation with respect to the eye. Please fill in the code in function `Solar_viewer::paint()` to correctly orient the billboard.

Finally, fill in the code in function `Solar_viewer::draw_scene()` to render the billboard using `color_shader_` (like you used for the planets in last week's assignment). Please note that, in order to make the transparency work, you need to explicitly enable blending and select an appropriate blending function. Please check the OpenGL reference entries for functions `glEnable` and `glBlendFunc`.

Phong lighting model

In this task, you will implement the Phong lighting model and use it to illuminate the planets. Recall the general Phong lighting model definition (as in assignment 2):

$$I = I_a m_a + I_l m_d (\mathbf{n} \cdot \mathbf{l}) + I_l m_s (\mathbf{r} \cdot \mathbf{v})^s, \tag{1}$$

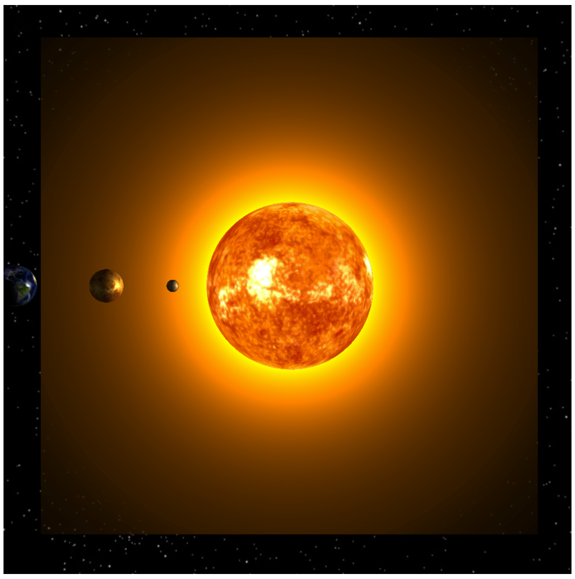

Figure 2: Insufficient glow opacity attenuation resulting in visible edges of the square billboard.

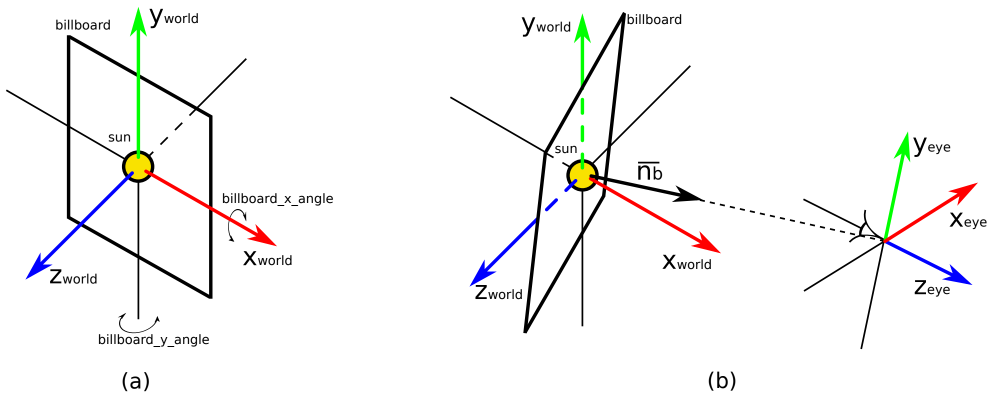

Figure 3: (a) The default orientation of the billboard in the world coordinate system and (b) the required orientation of the billboard with respect to the eye where $\mathbf{n_b}$ is the normal vector of the billboard.

where I is the final fragment's intensity, I_a is the ambient component of light source, I_l is diffuse/specular component of the light source, $m_{[a|d|s]}$ is the ambient/diffuse/specular component of the material, s is the shininess, and $\mathbf{n}, \mathbf{l}, \mathbf{r}, \mathbf{v}$ are the normal, light, reflected light and view vectors respectively. Recall that the diffuse component is only added if $\mathbf{n} \cdot \mathbf{l} > 0$ and the specular component is only added if both $\mathbf{n} \cdot \mathbf{l} > 0$ and $\mathbf{r} \cdot \mathbf{v} > 0$.

In our scene, we consider a single point light source—the Sun (see `sunlight` variable defined in `phong.frag`)—which provides the I_l component. I_a is usually thought of as the joint (and dispersed) contribution of all the light sources in the scene; in our case, just use: $I_a = 0.2 * \texttt{sunlight}$. For all planets except the earth, we'll assume the material components $m_{[a|d|s]}$ are the same and correspond to the RGB color sampled from the texture.

Compute the required vectors in the vertex shader (`phong.vert`), which you will then use in the fragment shader (`phong.frag`) (please check the `todo` notes in both source files). The vertex shader relies on a couple of uniform variables, namely the modelview matrix, MVP matrix, normal matrix and light position, which you must set before you render each planet within function `Solar_viewer::draw_scene()`. Please note that the normal matrix defines the transformation of the object's normals from the model to the view coordinate system.

Shading the Earth

Here the task is to combine multiple textures to visualize the day and night appearance of the Earth, to visualize the clouds, and to restrict the specular component of the Phong lighting model to the water surfaces which are not covered by clouds (we want the mainland to act as a purely diffuse surface).

Please start by copying your Phong lighting model vertex shader code from `phong.vert` to `earth.vert`. Then fill in the fragment shader code in `earth.frag` to combine the textures and to apply the Phong lighting.

The gloss and cloud textures are 1-channel (grayscale), whereas the day and night textures are 3-channel (RGB). The gloss and cloud textures together define the amount of specularity we want to use for a given pixel. The gloss texture is binary mask: it indicates whether or not there should be any specular component. The cloud texture allows for partial specularity (if permitted by the gloss mask): changing the cloudiness from 1 to 0 should linearly interpolate the specular component from zero to its full value. Please combine these two textures to compute the appropriate amount of specularity for a given pixel.

You will compute two different colors—a "day color" and a "night color"—and mix them together (linearly interpolate between them) based on the amount of sunlight reaching the fragment; the day color should appear on the hemisphere facing the sun and the night color on the opposite side and they should blend nicely on the border. Hint: you can use the diffuse component coefficient $\mathbf{n} \cdot \mathbf{l}$ from the Phong lighting model to determine this amount of daylight.

The first step in computing the day color is to apply the Phong lighting model. As in the previous subtask, model the ambient light component as $I_a = 0.2 * \texttt{sunlight}$. The ambient and diffuse components of the material, m_a and m_d, are given by the RGB color sampled from the day texture, but the specular component m_s should be pure white (i.e. $m_s = [1,1,1]$). Please do not forget to modulate the specular component of the Phong lighting model by the specularity value computed from the gloss and cloud textures (remember that mainland acts as diffuse surface and cloudiness reduces the specularity). Next, compute the cloud color by applying the Lambertian lighting model (i.e. the Phong lighting model without the specular component) where the material corresponds to RGB color sampled from cloud texture. Finally, mix the color from the Phong lighting calculation with the cloud color by linear interpolation (using the cloud texture grayscale value as the interpolation weight) to obtain the day color.

The night color is computed by simply linearly interpolating between the color sampled from night texture and black based on the cloudiness level.

Please refer to the Figure 1 (b) for an example of the expected visual outcome.

Please do not forget to pass all the textures as the uniform variables to the shaders (fill in the code in `Solar_viewer::draw_scene()`).

Solar surface disturbance

(NOTE: this section is optional and it is NOT graded.) In this task you will implement the gradual disturbance of the geometry of the sphere representing the Sun such that it would resemble a fluid. Please see the `todo` notes in `sun.vert` vertex shader for further instructions.

Grading

Each part of this assignment is weighted as follows:

- billboarding: 30%
- Phong lighting model: 30%
- shading the Earth: 40%

What to hand in

A compressed .zip file renamed to `Exercise5-Group`*i*`.zip`, where *i* is your group number. It should contain:

- Hand in **only** the files you changed (in this case, `solar_viewer.cpp`, `texture.cpp` `phong.[vert|frag]`, `earth.[vert|frag]`).
- A couple of screenshots clearly showing that you have correctly implemented the billboard, the Phong lighting model and multitexturing of the Earth (e.g. as in the Figure 1).
- A `readme.txt` file containing a description of how you solved each part of the exercise (use the same numbers and titles) and whatever problems you encountered. Indicate what fraction of the total workload each project member contributed.