CPSC 314
Assignment 3: Lighting and Shading

Due 11:59PM, Nov 3 2017

1 Introduction

The goal of this assignment is to explore lighting and shading of 3D models. You will be simulating four different shading models: Phong, Blinn-Phong, Anisotropic (like brushed metal), and Toon (cartoon-like). In this assignment you will be *completing* the four supplied pairs of shaders (both vertex and fragment shaders) to carry out these tasks, which includes editing necessary parts of the A3.js file.

1.1 Template

There are three scenes in the assignment, accessible with the number keys 1, 2, and 3 on your keyboard. You will be implementing the lighting models Phong and Blinn-Phong in Scene 1, Anistropic shading in Scene 2, and Toon shading in Scene 3. The template code is found in the main assignment directory, shader files can be found in the glsl folder, and the obj folder contains different objects at your disposal for the last two scenes. We have already defined the properties of a basic directional light in A3.js for the scene (defined in the lightColor, lightDirection and ambientColor variables). We have also defined suggested material properties for the objects you will be shading (defined in kSpecular, kDiffuse, kAmbient, shininess, alphaX, and alphaY).

1.2 Important Rules

The lighting models you will be implementing in this assignment are very common in graphics applications. Thus three.js has already implemented them in a number of the default materials provided with three.js. These can be used to test if you are getting reasonable results, but you must implement these shading models yourself in your own custom shaders.

2 Work to be done (100 pts)

1. **Part 1:** (70 pts) Shaders.

(a) **20 pts** Scene 1: Phong Reflection and Phong Shading. In Scene 1 you will see three spheres. Your goal is to shade them, from left to right, using Gouraud, Phong, and Blinn-Phong models. The Gouraud shaded sphere has already been given to you for visual comparison with the other two shading models, so you don't need to implement it. The Phong *shading* model improves on the Gouraud shader by computing the lighting per fragment, rather than per vertex. This is done by using the interpolated values of the fragment's position and normal.

The following image, taken from the Wikipedia article on the Phong reflection model, shows how the different components look individually and summed together:

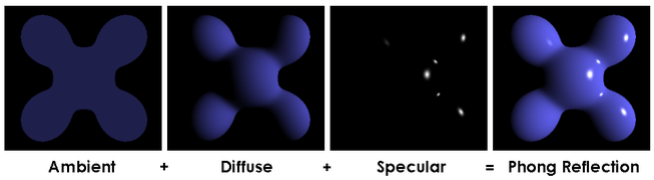

Figure 1: Phong Reflection Model

For this part implement the full Phong reflection model in a Phong shader, and apply it to the middle sphere. The main calculations should all go in the fragment shader (in file `phong.fs.glsl`). Note that you will still need a vertex shader (in `phong.vs.glsl`), to pass the appropriate information to your fragment shader. Remember that dot products should be clamped between 0 and 1, since we don't want to see reflections from the wrong side of the object.

(b) **15 pts** Scene 1: Blinn-Phong Reflection.

Instead of continuously recomputing the dot product between the reflected light vector and viewer, a dot product between the halfway vector between light and viewing direction, and the surface normal can be used. This is the essence of the Blinn-Phong reflection model.

Complete the supplied Blinn-Phong shaders (`phong_blinn.fs.glsl` and `phong_blinn.vs.glsl`) to apply the Blinn-Phong reflection model, per fragment, to the final sphere.

Once Scene 1 is complete, it should look similar to Fig. 2. You can also use the built in THREE.js Phong materials to test your shader work.

Note: the "Gouraud" THREE.js material does not have a specular component. *Hint:* Specular lighting shouldn't be static as you move the camera around the scene!

(c) **20 pts** Scene 2: Anisotropic Shading.

In Scene 2, you will start off with just a sphere for testing your implementation of an Anisotropic shading model. Anisotropic shading is similar to Phong and Blinn-Phong shading, except the specular term is direction-dependent, like many metal surfaces (see also Section 14.4 of the textbook). The provided material properties α_x and α_y refer to strength of the specular highlights along the principal

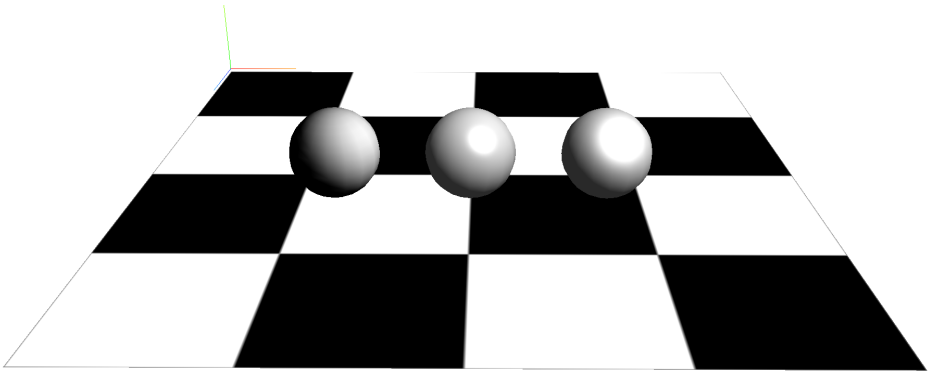

Figure 2: Example of completed Scene 1.

tangent direction and the binormal direction (orthogonal to both the normal and tangent) respectively. We will be using Ward's model of Anisotropic reflection[1], where the specular term is

$$k_{specular}\sqrt{\frac{L\cdot N}{V\cdot N}} * \exp\left(-2\frac{((H\cdot T)/\alpha_x)^2+((H\cdot B)/\alpha_y)^2}{1+H\cdot N}\right)$$

Assume that the "brush direction," T, is the horizontal tangent to the surface, which we can estimate with the cross product of the surface normal with an "up" vector $(0, 1, 0)$.

Begin by implementing the anisotropic shaders (`anisotropic.?s.glsl`) for the sphere, and once the sphere looks correct, build an interesting scene (e.g., as in Fig. 3) using the provided objects in the obj folder of the assignment, and the object loader in A3.js.

(d) **15 pts** Scene 3: Toon Shading with silhouettes.

In Scene 3, you will start off with just a sphere for testing your implementation of a Toon shading model. This is an example of a "non-photo realistic" (or NPR) shader. It emulates the way cartoons use very few colors for shading, and the color changes abruptly, while still providing a sense of 3D for the model. This can be implemented by quantizing the light intensity across the surface of the object. Instead of making the intensity vary smoothly, you quantize this variation into a number of steps for each "layer" of toon shading. Also, draw approximate silhouette edges of the object by detecting fragments on the silhouette and coloring these fragments with a dark color (think about the relationship between the surface normal and the viewing direction for these fragments).

Like before, begin by implementing the Toon shaders (`toon.fs.glsl` and `toon.vs.glsl`) for the sphere, and once the sphere looks

[1]For more details, see, for example, `https://en.wikibooks.org/wiki/GLSL_Programming/Unity/Brushed_Metal`

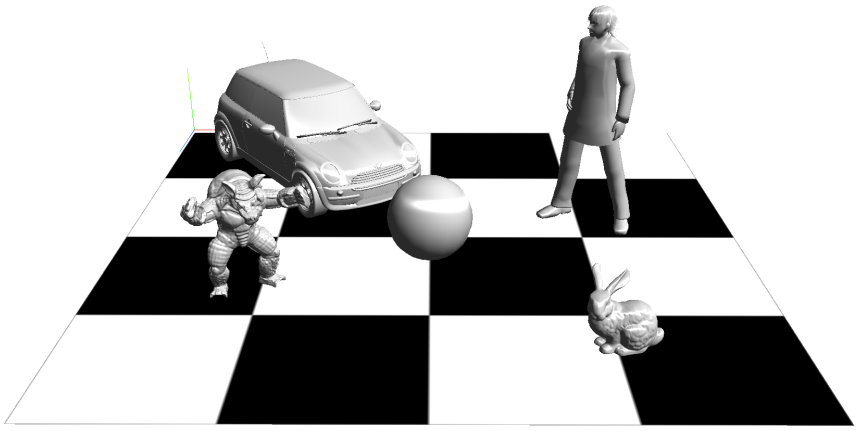

Figure 3: Example of completed Scene 2.

correct, build an interesting scene using the provided objects in the obj folder of the assignment, and the object loader in A3.js. See Fig. 4.

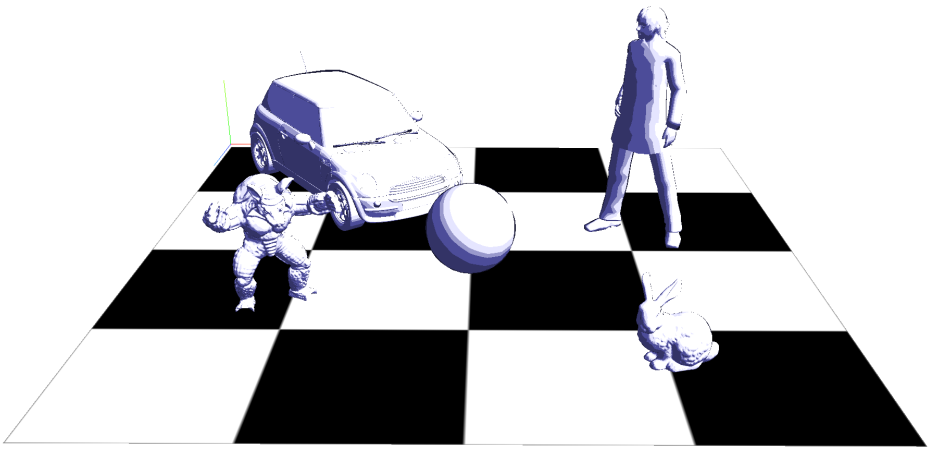

Figure 4: Example of completed Scene 3.

If you need some inspiration the following movies and video games were rendered with toon shading (also called cell shading) techniques:

`http://en.wikipedia.org/wiki/List_of_cel-shaded_video_games`.

2. **Part 2:** (30 pts) Creative License.

   Create a new shader or modify one of the above shaders to implement some interesting lighting effects similar to the complexity of the ones above. Some recommendations are:

   - Implement cross-hatching in your NPR toon shader (see Figure 5).
   - Explore other non-photo realistic shaders, such as Gooch shading for technical illustration.

- Simulate fog.
- Implement a more physically based shader, based on SIGGRAPH course material (`http://blog.selfshadow.com/publications`). For example, parts of `https://goo.gl/iJ4moh`.

Bonus marks may be given at the discretion of the marker for particularly noteworthy explorations.

2.1 Hand-in Instructions

You do not have to hand in any printed code. Create a README.txt file that includes your name, student number, and login ID, and any information you would like to pass on to the marker. Create a folder called "a3" under your "cs314" directory. Within this directory have two subdirectories named "part1," and "part2", and put all the source files, your makefile, and your README.txt file for each part in the respective folder. Do not use further sub-directories. The assignment should be handed in with the exact command:

```
handin cs314 a3
```

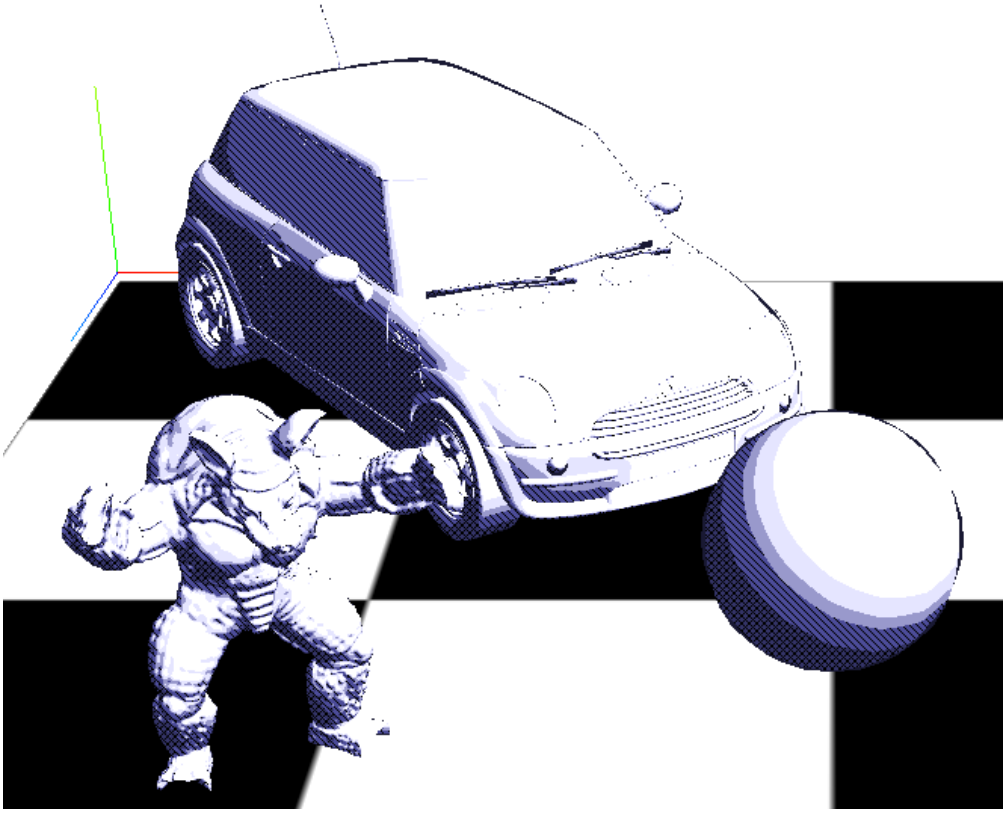

Figure 5: Example of hatching.