CPSC 314
Assignment 3: Shaders

Due 11:59PM, March 11th, 2022

1   Introduction

In this Assignment, you will utilize your knowledge of lighting and shading to modify the appearance of 3D models. Here we will study how to use basic textures and implement various shading algorithms: Blinn-Phong, Toon, as well as some other fun shaders. This is a rather interesting assignment, so we hope you will have fun with this one.

1.1   Getting the Code

Assignment code is hosted on the UBC Students GitHub. To retrieve it onto your local machine navigate to the folder on your machine where you intend to keep your 'assignment code, and run the following command from the terminal or command line:

`git clone https://github.students.cs.ubc.ca/cpsc314-2021w-t2/a3-release.git`

1.2   Template

- The file `A3.html` is the launcher of the assignment. Open it in your preferred browser to run the assigment, to get started.

- The file `A3.js` contains the JavaScript code used to set up the scene and the rendering environment. You will need to make minor in it to answer the questions.

- The folder `glsl` contains the vertex and fragment shaders for the armadillo and other geometry. This is where you will do the rest of your coding.

- The folder `js` contains the required JavaScript libraries. You do not need to change anything here.

- The folder `obj` contains the geometric models loaded in the scene.

- The folder `images` contains the texture images used.

1.3 Execution

As mentioned above, the assignment can be run by opening the file `A3.html` in any modern browser. However, most browsers will prevent pages from accessing local files on your computer. If you simply open `A3.html`, you may get a black screen and an error message on the console similar to this:

<p style="color:red; text-align:center;">XMLHttpRequest cannot load... Cross origin requests are only supported for protocol schemes: http, data, https.</p>

Please see this web page for options on how to run things locally:

https://threejs.org/docs/#manual/en/introduction/How-to-run-things-locally

2 Work to be done (100 pts)

First, ensure that you can run the template code in your browser. See instructions in Assignment 1. The initial scene should look as in Figure 1. Study the template to get a sense of what and how values are passed to each shader file. There are four scenes, each corresponding to a different shader, you may toggle between them using the number keys 1, 2, 3 on your keyboard: 1: Blinn-Phong, 2: Toon, 3: Rolling Grid.

The default scene is set to 1. See

```
let mode = shaders.PHONG.key;
```

in A3.js. You may find it convenient during your development to change this default value to the scene containing the shader that you are currently working on (e.g., for question 1b, you could use `let mode = shaders.TOON.key;`).
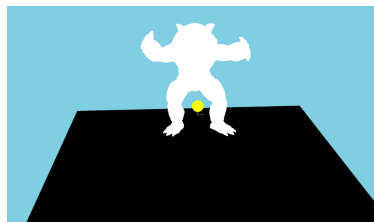

Figure 1: Initial configuration

Part 1: Required Features

(a) 15 pts Basic Texturing

For the first part of the assignment, you will implement a basic texturing. Your goal is to use the provided floorColorTexture to give the floor a rocky appearance. Use the provided floor vertex and fragment shaders, with the appropriate uniforms, to sample and apply the texture.
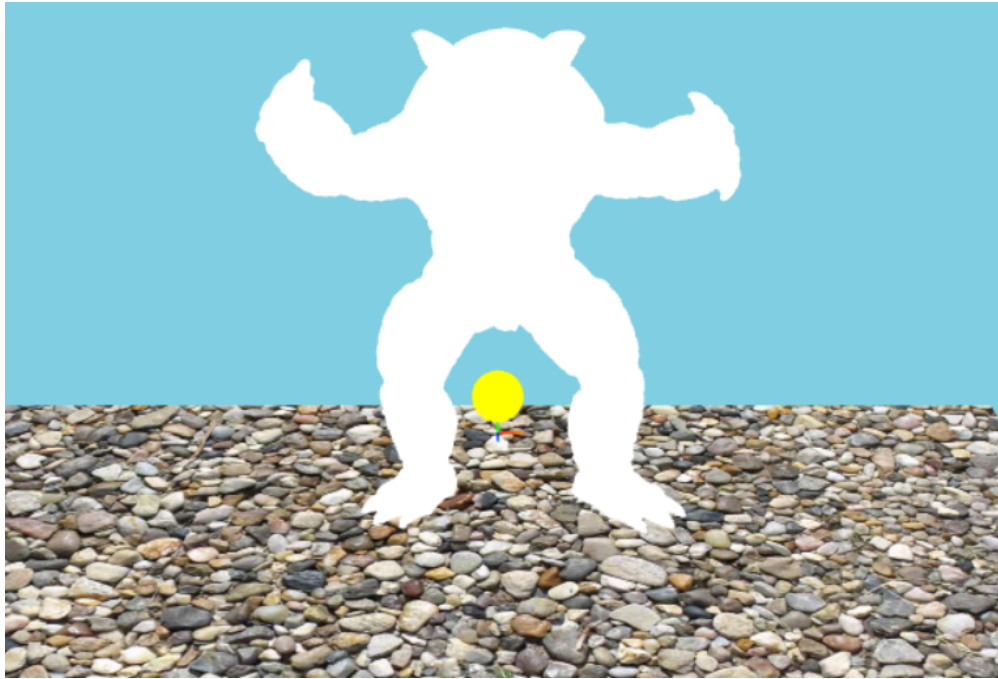

Figure 2: Question 1(a). Textured floor

(b) 25 pts Blinn-Phong Reflection

First of all, note that the Phong reflection model is a different type of thing than the Phong shading model; they just happen to be named after the same person. The latter improves on the Gouraud shading 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. We'll be using Phong shading throughout this assignment except for part 1 (d). The following image, taken from the Wikipedia article on this model, shows how the different components look individually and summed together:
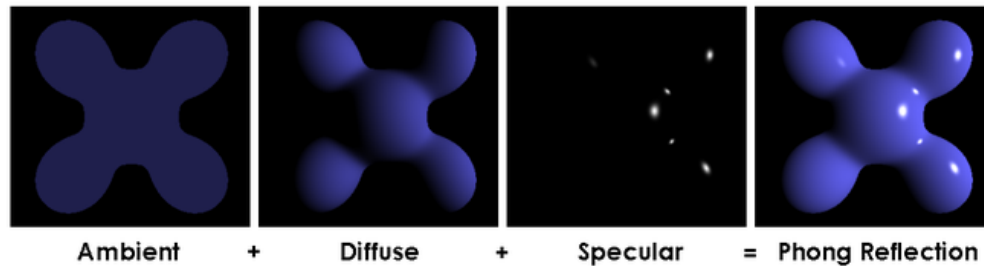
Figure 3: Phong reflection model.

In this scene, you'll implement the Blinn-Phong reflection model, which is a slight modification of the specular component of Phong's original reflection model to make it more compatible with physically-based rendering (we will cover that later in the course). It is also simpler, and described in detail (with code) in the textbook, Section 1.3; it uses the "halfway vector," instead of the "bounce" vector.

Hint 1: This part can be done entirely in `phong.fs.glsl` and `phong.vs.glsl`.

Your task here is to complete the code in phong.vs.glsl and phong.fs.glsl to shade the Armadillo in Scene 1 using the Blinn-Phong reflection algorithm. The main calculations should all go in the fragment shader, but you will still need a vertex shader to pass the appropriate information to your fragment shader. Your resulting armadillo should look something like Figure 4.

Figure 4: Question 1(b). Blinn-Phong'd armadillo

(c) 25 pts Toon

Send the armadillo to the realm of action and superheros! Unlike the smooth, realistic shading in the previous questions, Toon shading gives a non-photorealistic result. 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. Use two blue colour tones for the armadillo (lower light intensity will be a darker shade), as seen in Figure 4.

This is most easily done by interpolating between two predefined colours. Lastly, draw a dark blue silhouette on the armadillo. You can use the cosine of the normal and the viewing direction to compute whether the fragment should be an outline: fragments that are "edgy" enough should be outlines, and recall how you can obtain the cosine for two vectors.

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`

Hint 1: Use the surface normal and the viewing direction of a fragment to determine whether it is on the silhouette of the armadillo.

Hint 2: Since the silhouette is determined using the surface normal and the viewing direction, and does not depend the light direction, moving the light source will not affect its location.
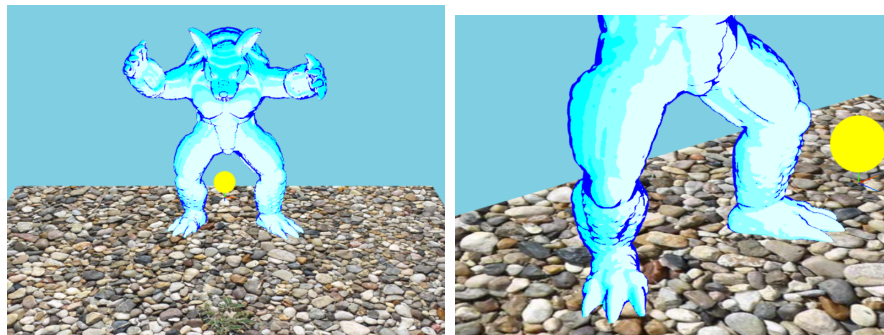

Figure 5: Question 1(c). Toon armadillo

(d) 35 pts Scene 3: Grid pattern

The possibilities are endless! Here, we will (i) make the armadillo out of squares by checking whether a fragment is close enough to planes on a regular grid in the 3D space around the armadillo; (ii) make the grid squares smoothly "roll" down the body over time, as in Figure 6, and (iii) interpolate the color of the fragments by the light intensity. To do so, you will need to write a function in the fragment shader that shades the fragment depending on a time "ticks" and the local vertex position, discarding fragments that you don't want to render. Notice the empty space between the grid squares. Complete the code in squares.vs.glsl and squares.fs.glsl.

Hint 1: : The `discard` statement in GLSL throws away the current fragment, so it is not rendered.

Hint 3: : The armadillo's model coordinates may be smaller than you think. If your armadillo disappears completely (not because of some syntax error) try using smaller numbers when computing regions to discard.
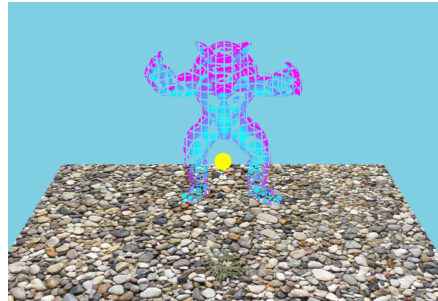

Figure 6: Question 1(d). Grid-pattern armadillo

Part 2: Creative License (Optional)
You have many opportunities to unleash your creativity in computer graphics! In this **optional** section, and you are invited to extend the assignment in fun and creative ways. We'll highlight some of the best work in class. A small number of exceptional contributions may be awarded bonus points. Some possible suggestions might be:

- Create a new interesting shader for the armadillo or chest. E.g., Make the armadillo's skin out of an anisotropic material, like brushed metal. e.g., see using Heidrich-Seidel Anisotropic distribution model
  `https://en.wikipedia.org/wiki/Specular_highlight#Heidrich%E2%80%93Seidel_anisotropic_distribution`

- Add a new object that uses texture mapping.

- Generate new animated effects using the `discard` function in GLSL.

3 Submission Instructions

3.1 Directory Structure

Under the root directory of your assignment, create two subdirectories named "part1" and "part2", and put all the source files and everything else required to run each part in the respective folder. Do not create more sub-directories than the ones already provided.
You must also write a clear README.txt file that includes your name, student number, and CWL username, instructions on how to use the program (keyboard actions, etc.) and any information you would like to pass on to the marker. Place README.txt under the root directory of your assigment.

3.2 Submission Methods

Please compress everything under the root directory of your assignment into `a3.zip` and submit it on Canvas. You can make multiple submissions, but we will grade only the last one.

4 Grading

4.1 Point Allocation

Each assignment has 100 points for Part 1. Part 2 is optional and you can get bonus points (0-10 points) at the instruction team's discretion. Percentage wise, we use Part 1's total points as the denominator: e.g. if you get 95 out of 100 points from Part 1, but no points from Part 2, then your percentage grade would be 95/100. If you get full points from both Parts, then your percentage grade would be 110/100.

4.2 Face-to-face (F2F) Grading

For each assignment, you are required to meet face-to-face with a TA on Zoom or in person to demonstrate that you understand why your program works. Details regarding how to sign up a grading session with a TA will be announced on Canvas and on Piazza.

4.3 Penalties

Aside from penalties from incorrect solution or plagiarism, we may apply the following penalties to each assignment:

Late penalty. You are entitled up to three grace (calendar) days in total throughout the term. No penalties would be applied for using them. However once you have used up the grace days, a deduction of 10 points would be applied to each extra late day. Note that

1. The three grace days are given for all assignments, **not per assignment**, so please use them wisely;

2. We consider the time of only your last submission;

3. We do not consider Part 1 and Part 2 submissions separately. Say if you submitted Part 1 on time but updated your submission for Part 2 one day after the deadline, we would count one late day.

No-show penalty. You are required to sign up a grading slot at least one day before F2F grading starts, and show up at your slot on time. So a 10-point deduction would be applied to each of the following circumstances:

1. Not signing up a grading slot before the sign-up period closes;

2. Not showing up at your grading slot.

Note that we would not apply the no-show penalty if you are unable to sign up/show up on time due to an emergency, or if you cannot sign up because none of the slots work for you. In those cases, please contact the course staff on Piazza before a sign-up closes. If you have not signed up on time or if you have missed your slot, please follow the steps outlined in this Piazza post: https://piazza.com/class/ky4qwpnvrqa5pq?cid=190