Università
della
Svizzera
italiana
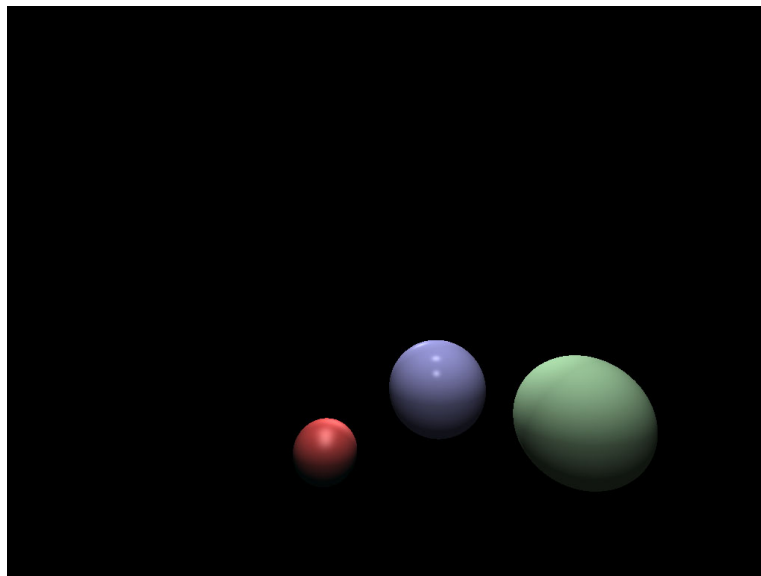
**Faculty
of Informatics**

Computer Graphics (Fall 2022)

Assignment 2: Phong lighting model

September 30, 2022

Your main task in this assignment will be to implement Phong lighting model to compute the above image. You will be given an updated framework with clearly marked places for implementing the required functionality. The updated framework is also the solution for the previous assignment.

Updated framework

We updated the framework by:

- adding *Material.h* file, which contains the structure for describing all the parameters of the Phong lighting model,

- extending the definition of the *Object* class by adding variable `material` to store the material information as well as functions `getMaterial()` and `setMaterial()`,

- adding a new constructor for the *Sphere* class, which takes as an argument the material structure,

- adding a new class *Light*, to represent a point light source, i.e., its position and the intensity,

- declaring additional variables for lights, i.e., array of point light sources, `lights`, and the intensity of the ambient light, `ambient_light`,

- declaring function `PhongModel` for implementing the lighting model.

The code is commented. Before continuing with the assignment, please familiarize yourself with the updates and check the code for comments indicating the places which should be modified.

Exercise 1 [10 points]

Implement the Phong lighting model as indicated by the comments in the code. Modify the scene definition such that it includes the following objects:

	Blue sphere	Red sphere	Green sphere
center	$(1.0, -2.0, 8.0)$	$(-1.0, -2.5, 6.0)$	$(3.0, -2.0, 6.0)$
radius	1.0	0.5	1.0
ρ_a	$(0.07, 0.07, 0.1)$	$(0.01, 0.03, 0.03)$	$(0.07, 0.09, 0.07)$
ρ_d	$(0.7, 0.7, 1.0)$	$(1.0, 0.3, 0.3)$	$(0.7, 0.9, 0.7)$
ρ_s	$(0.6, 0.6, 0.6)$	$(0.5, 0.5, 0.5)$	$(0.0, 0.0, 0.0)$
k	100.0	10.0	0.0

and three point light sources with positions at $(0.0, 26.0, 5.0)$, $(0.0, 1.0, 12.0)$, $(0.0, 5.0, 1.0)$, each emitting the light with intensity $(0.4, 0.4, 0.4)$. Upon completion of this exercise, you should be able to produce an image same as the one on top of this document.

In this exercise, you can use the in-built function `glm:reflect()` to compute reflected direction. If you do so, please read the documentation of this function and pay attention to the orientation of the directions you provide as an input to this function.

Exercise 2 [5 points]

Consider a shiny ground plane $y = 0$ illuminated by a directional light source. Assume that the direction towards the light is $(1, 2, 2)$ and the viewer/camera is at $(4, 6, 7)$. Note this is different from what we currently assume in our raytracer implementation where the camera is at $(0, 0, 0)$.

- Task 1 (3 points): Compute a position on the ground plane at which the viewer observes the peak of the highlight. You can assume that the peak occurs where a perfect mirror reflection takes the ray from the light source and reflects it directly towards the viewer.

- Task 2 (2 points): Let us model the appearance of the plane using Phone lighting model. Assume that the plane is reflecting half of the incoming light according to the diffuse reflection and half according to specular reflection. The shininess coefficient of the sphere is $k = 2$. The sphere does not emit any light, there is no ambient illumination, and the intensity of the directional light is $I = 1$. There is also no color in the scene, i.e., everything is grey. Compute the intensity of the light observed by a viewer at the peak of the highlight, i.e., the location of the plane computed in the previous task. Do you need to account for the distance between the light source and the plane? If so, assume for your calculations any distance you want.

Bonus Exercise 3 [2 points]

Imagine it is night and you see a full moon in the sky. Ignoring the shading artifacts caused by craters, it appears as a white disk with constant brightness rather than a sphere shaded according to the Phong illumination where the color is modulated by the cosine of the angle between the normal vector and the light direction. See the image on the right for the comparison. What could be the reason for this?

Phong model   Moon

Bonus Exercise 4 [3 points]

Use the solution to the Exercise 1 to create a short (2 seconds) animation where one of the light sources moves in a circle above the three spheres. To be able to localize the light sources, visualize them with small spheres. One way of modifying the code to generate an animation is to introduce an argument for the program, which can be set from the command line (Terminal) and is the time stamp or simply frame index. Using this additional argument, the raytracer can position the light source in the desired location and write the output to

a unique file which also contains the frame index, e.g., `result_0.ppm`, `result_1.ppm`,... You can write a short bash script to then render all the frames. Now, you only have to compile all the frames into the video sequence. One way of doing it is to use in macOS `sips` function from the terminal to first convert all the PPM files to PNG, for example, `sips -s format png result.ppm --out result.png`. Again you can use a bash script to convert all the files at once. Once you have PNG files that correspond to individual frames, you can load them as a sequence in QuickTimePlayer using *File → Open Image Sequence*. QuickTime-Player will allow you to save the sequence as a video file which you can later play. Alternatively, you can use FFmpeg[1].

Submission

You should submit one ZIP-file via iCorsi containing:

- one modified *main.cpp* file for exercises 1,

- a PDF or image file containing solution to exercises 2 and 3,

- if you solve also the Exercise 4, please also include the video file together with a separate *main.cpp* which solves this exercise.

Additionally, in the comment to your submission indicate which exercises you solved and what problems you encountered, if any.

<div style="text-align: right;">

Solutions must be returned on October 6, 2022 via iCorsi3

</div>

---

[1]https://ffmpeg.org