Introduction to Computer Graphics
Assignment 2 – Lighting

Submission deadline: 09.10.2020, 12:00

Late submissions are not accepted

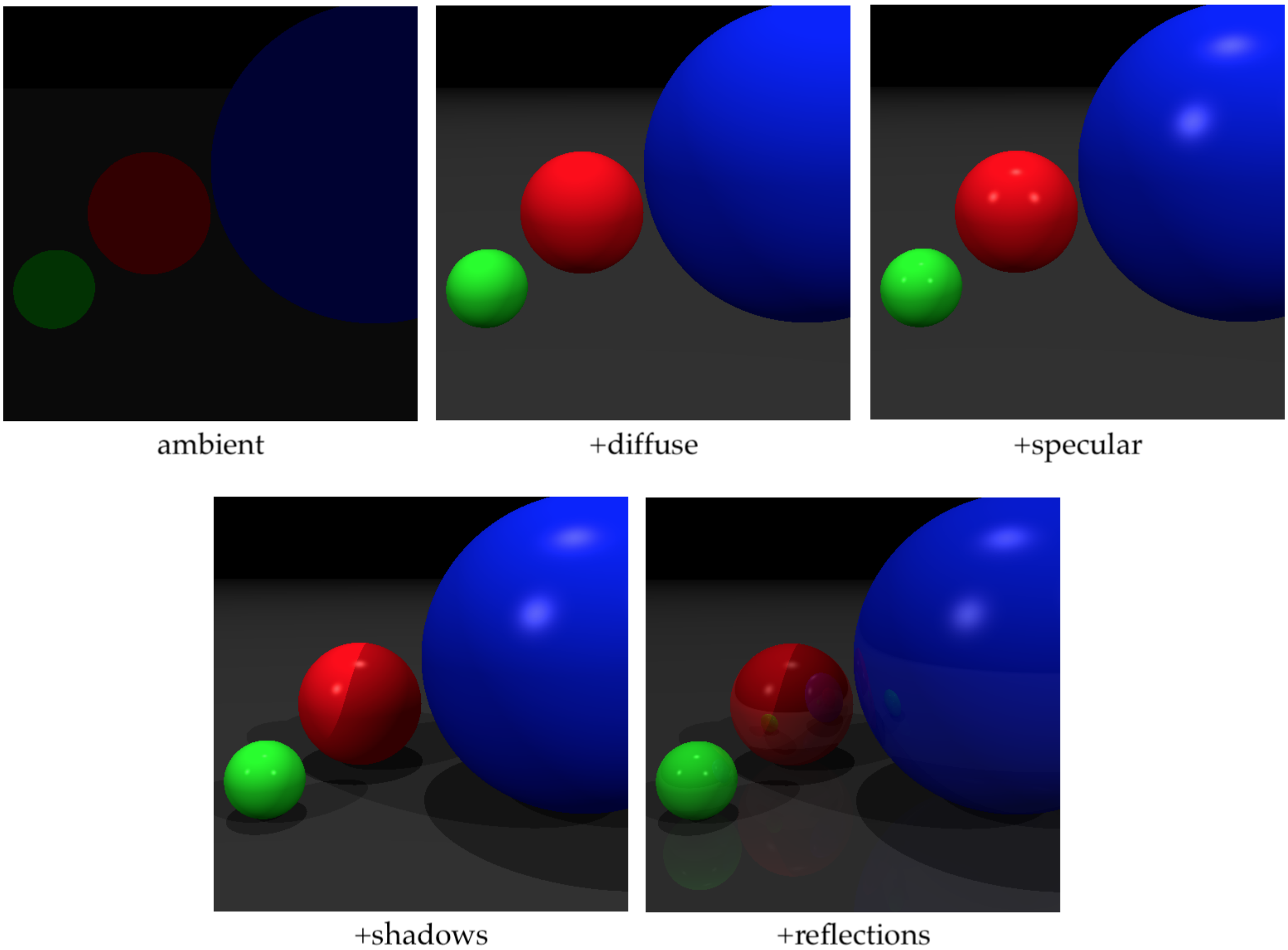


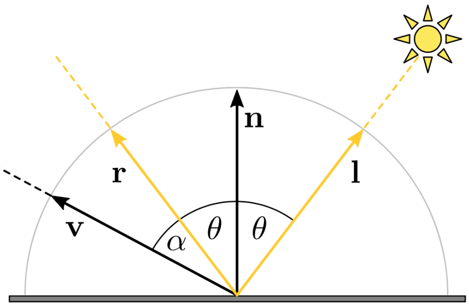
Figure 1: Expected result for Phong lighting model, shadows and reflections.

In this assignment, you will implement the Phong lighting model and add reflections to your scenes. The framework code for this assignment extends the one from last week; if you download a fresh copy from ILIAS you will need to copy your solutions for the plane and cylinder intersections into the files `Plane.cpp` and `Cylinder.cpp`. Furthermore,"`todo`" comments have been inserted in `Scene.cpp` to indicate where you need to add your implementations. If you already set up a git repository to collaborate with your fellow group members, you can just copy the TODO comments from `Scene.cpp` over to your repository (or just note where your implementation needs to go and get started).

In the `expected_results` directory, we provide the images you should expect your finished code to produce for a subset of the provided scenes. One such result is shown in Figure 1.

Phong Lighting Model and Shadows

The goal of the first part of this assignment is to implement the Phong lighting model. Follow the explanations from the lecture slides and the formula in Fig. 2. In the file `Scene.cpp`, you need to fill in the missing code in the `lighting()` function to compute the variable `color`.



$$\mathbf{I} = \mathbf{I}_a * \mathbf{m}_a + \mathbf{I}_l * \left(\mathbf{m}_d\,(\mathbf{n} \cdot \mathbf{l}) + \mathbf{m}_s(\mathbf{r} \cdot \mathbf{v})^s\right)$$

Figure 2: Phong lighting formula.

Start by computing the global ambient contribution. The result of this step is shown in Figure 1 on the top left. Then, for each light, add in its diffuse and specular contributions (see Fig. 1 top middle and top right). You will probably find it helpful to review the attributes stored in classes `Light` and `Material`. Feel free to use the existing vector functions in `vec3.h` e.g. `mirror`, `reflect`, `norm`, `dot`, `normalize`.

To add shadows to your scene, discard the diffuse and specular contributions from light sources that are blocked by another object. You can determine which light sources are blocked by generating a `shadow ray` (see lecture and exercise slides) and using the `intersection()` function. The expected result is shown in Fig. 1 lower left.

Reflections

The second part of this assignment is to add reflections to your scene. In the file `Scene.cpp` you need to fill in the missing code in the `trace()` function to update the variable `color`. Follow the recursive ray tracing algorithm explained in the lecture and the `todo` comments in the code.

Use the `material.mirror` to determine how reflective the material is and the function `reflect()` to compute the reflected ray. Compute the final returned color using linear interpolation:

$$\texttt{color} = (1 - \alpha) \cdot \texttt{color} + \alpha \cdot \texttt{reflected\_color}, \tag{1}$$

where `reflected_color` is computed by recursively tracing a ray reflected at the intersection point (see the exercise slides), and α is the `material.mirror` property.

Grading

Each part of this assignment is weighted as follows:

- Ambient contribution: 10%
- Diffuse contribution: 15%
- Specular contribution: 20%
- Shadows: 25%
- Reflections: 30%

What to hand in

A .zip compressed file with the following contents:

- Hand in **only** the files you changed (in this case, `Scene.cpp`) and the requested program output. It is up to you to make sure that all files that you have changed are in the zip.
- A `readme.txt` file containing a description on how you solved each exercise and the encountered problems.
- Other files that are required by your `readme.txt` file. For example, if you mention some screenshot images in `readme.txt`, these images need to be submitted too.

Submit solutions to ILIAS before the deadline. Late submissions receive 0 points!