Foundations of Graphics

Universität Bonn
Institut für Informatik II
October 29, 2019

Winter term 2019
Prof. Dr. Matthias Hullin

Sheet 3 - Raytracing

Please upload your solutions to
`https://uni-bonn.sciebo.de/s/XfRkWhpWDEeR8Y7`
by
So, 3.11.2019.
Make sure to **list all group members** on all pages / source files.
Theoretical solutions must be texed or scanned, **photos will not be accepted**.

Practical Part

Assignment 1) Implementing ray tracing algorithm *(2+2+2=6Pts)*

In this exercise, we will implement a simple ray tracer that computes Phong shading, perfect reflections and hard shadows. An overview of the basic principle is given in Figure 1.

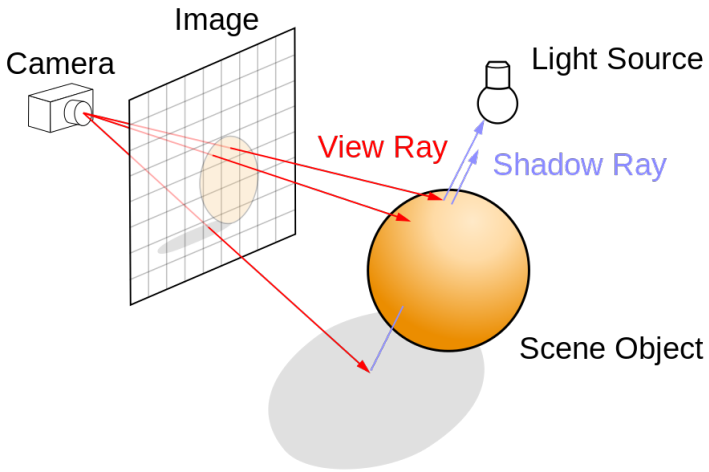

Figure 1: Ray tracing (source: wikipedia)

Running the unmodified framework will result in a very dark image, where the balls are barely visible. The image does not show the shadowing, reflection and shading effects. In the following tasks, you need to add them to the ray tracer.

a) Implement hard shadows inside the `rayTracing()` function by checking whether any objects are in between your intersection point with the geometry and the light source.
b) Implement reflections. Save screen shots with and without reflection.
Hint: If the new ray starts exactly at the previous intersection point, it might directly intersect with the same offset again. To prevent this, add a small offset (in the right direction!) to its origin.
c) Modify `rayTracing()` function to include Phong shading with diffuse and specular terms. Now your final result should be similar to Figure 2.

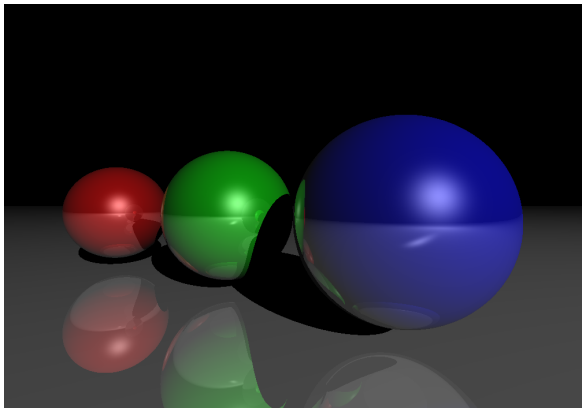

Figure 2: Result of the ray tracing algorithm with Phong shading model with reflection

Theoretical Part

Assignment 2) Rendering Complexity *(0.5+1+0.5+1=3Pts)*

You just wrote a raytracer that computes images for an indoor scene using the Whitted raytracing algorithm. Each ray coming from the camera is reflected three times in the scene (all materials are reflective) before it terminates. Refraction is disabled, but **shadow testing is enabled** for all reflections.

Let's assume that each ray intersection test with the scene (for primary, secondary and shadow rays) takes 0.0001 seconds and your output image has a resolution of 640×480 pixels (using one primary ray per output pixel).

Unsatisfied with the look of the perfect reflections, you decide to implement a naïve global illumination algorithm. Instead of generating a single reflection ray at each intersection, you generate 100. The recursion depth remains fixed at 4 (one primary ray, three reflected rays).

a) How long does the image with the perfect reflections render?

b) How long does the naive global illumination algorithm render?

c) What is the asymptotic runtime for n reflections? (Remember: Previously the recursion depth was 3.)

d) How could this simple algorithm be improved to reduce render time without affecting the quality too much? Name at least one idea!

Assignment 3) Plane reflection *(2+1+1=4Pts)*

We are in 2D land. You render a landscape with a big lake and a plane at position $P = \begin{pmatrix} 375 & 194 \end{pmatrix}^T$ from a camera position $C = \begin{pmatrix} 0 & 10 \end{pmatrix}^T$ (see Figure 3).

a) Using the formula for reflection from the lecture, at which position $L = \begin{pmatrix} x & 2 \end{pmatrix}^T$ on the lake will the reflection of the plane show up? You can assume the normal vector of the lake to be $N = \begin{pmatrix} 0 & 1 \end{pmatrix}^T$.

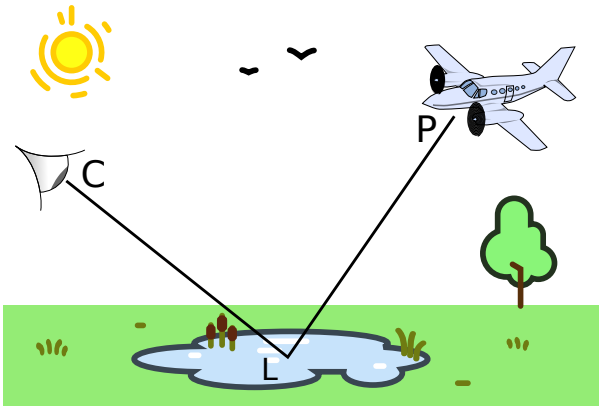

Figure 3: A landscape with a plane reflecting in a lake.

b) In the lake is also a sunken ship which is visible at the surface point $L_1 = \begin{pmatrix} 10 & 2 \end{pmatrix}^T$. The lake is uniformly 10 units deep. The refractive index of the air is $n_1 = 1.000277$ and the refractive index of the water is $n_2 = 1.330$. At which position on the bottom $B = \begin{pmatrix} x & -8 \end{pmatrix}$ is the ship wreck?

c) How does this observation change if the lake is frozen in the winter to crystal clear ice with a refractive index of $n_3 = 1.31$? How about a diamond sea ($n_4 = 2.417$)? A qualitative answer is sufficient.

Good luck!