Advanced Computer Graphics
Exercise 2 - BRDF Sampling

Handout date: 28.09.2015
Submission deadline: Monday, 05.10.2015, 11:00pm

Note

Undeclared copying of code or images (either from other students or from external sources) is strictly prohibited! Any violation of this rule will lead to expulsion from the class. No late submission allowed.

What to hand in

Solve the exercise in groups of 3. Hand in a `.zip` compressed file renamed to `Exercise2_#id.zip` where `#id` is your group number. It should contain the following files:

- The BRDF implementations:
  - `diffuse.cpp` : the diffuse BRDF
  - `phong.cpp` : a BRDF based on the Phong shading model
  - `hemisampling.cpp` : cosine hemisphere sampling
- A `readme.pdf` containing a description of your solution approach, your rendered images, how much time you needed and encountered problems (use the same numbers and titles), and providing answers to the questions.

2.1 Before Starting

Open the files `src/ex2/diffuse.cpp` and `src/ex2/phong.cpp` and put your group number in `#define GROUP_NUMBER`.

2.2 Local Illumination

2.2.1 The Reflection Equation

The reflection equation is an integral equation to estimate the radiance L_r leaving a point $\mathbf{x}$ in a certain direction $\mathbf{w}_r$. This equation is based on the law of conservation of energy. At a certain point $\mathbf{x}$ on the surface, the radiance emitted in a certain direction $\mathbf{w}_r$ is the sum of the emitted light L_e and reflected light in that direction. The reflected light is the sum of the incoming light L_i from all directions multiplied by the surface reflection and the cosine of the incident angle $\cos(\theta_i)$. The surface reflection at a point $\mathbf{x}$ can be defined by a *bidirectional reflectance distribution function* (BRDF) $fr(\mathbf{x}, \mathbf{w}_i, \mathbf{w}_r)$ that represents the ratio of reflected radiance exiting along a direction $\mathbf{w}_r$ to the irradiance incident L_i on the surface from a certain direction $\mathbf{w}_i$.

$$L_r(\mathbf{x}, \mathbf{w}_r) = L_e(\mathbf{x}, \mathbf{w}_r) + \int_\Omega fr(\mathbf{x}, \mathbf{w}_i, \mathbf{w}_r) L_i(\mathbf{x}, \mathbf{w}_i) \cos(\theta_i) d\mathbf{w}_i$$

2.2.2 Monte Carlo Integration

In the lecture, you have learned about Monte-Carlo Integration, which is a numerical method to evaluate an integral $I = \int f(x)dx$. This method reformulates the integration into an equivalent expected value computation problem, and estimates this expected value by generating a large number of sample points $X_1, X_2, \ldots, X_M$ following a *probability density function* (pdf) $p(x)$

$$I = \int \frac{f(x)}{p(x)} p(x)dx = E[\frac{f(x)}{p(x)}] \approx \frac{1}{M} \sum_{i=1}^{M} \frac{f(X_i)}{p(X_i)}.$$

One question you may ask is what would be the best pdf in order to approximate the integral with as few samples as possible. As explained during the lecture, the best pdf turns out to be proportional to $f(x)$.

2.2.3 Rejection Sampling of a Hemisphere

Using MC methods, to estimate the radiance $L_r(\mathbf{x}, \mathbf{w}_r)$, we can uniformly sample the incoming direction $\mathbf{w}_i$ on a hemisphere centered at the surface normal. In the lecture, you have learned a rejection sampling approach to generate these samples.

```
Vector3D w_i;
do {
        w_i.x = 1-2*drand48();
        w_i.y = 1-2*drand48();
        w_i.z = 1-2*drand48();
} while(w_i.length2() > 1 || w_i.z < 0)

// Project onto hemisphere
w_i /= w_i.length();
```

Listing 1: Rejection Sampling of a Hemisphere

Question 2.2.3. What is the expected number of `do ...while` loops this approach needs to generate one sample? Explain your answer. **(5 points)**

2.2.4 Naive Implementation

To avoid the shortcomings of the rejection sampling approach, we can directly sample the hemisphere. In `src/naive.cpp` we introduce a naive implementation of this algorithm. Take a look at the implementation of the function `hemisphereSampling`. This function uniformly samples a hemisphere using spherical coordinates. Now take a look at the `Li` function and try to understand the code and how it relates to the reflection equation and the Monte Carlo integration.

For this part, use the scene **diffuse-naive.xml** by setting **Command Arguments** to `diffuse-naive.xml groupId`, where `groupId` is the id of your group.

2.3 Directional Sampling & BRDF Sampling

The naive implementation uniformly samples the hemisphere to estimate the radiance at a point $\mathbf{x}$. As we have briefly mentioned in Section 2.2.2 the best pdf to estimate the integral is proportional to the function that we are integrating:

$$h(\mathbf{x}, \mathbf{w}_i, \mathbf{w}_r, \mathbf{n}) = fr(\mathbf{x}, \mathbf{w}_i, \mathbf{w}_r) L_i(\mathbf{x}, \mathbf{w}_i) \cos(\theta_i)$$

As $L_i(\mathbf{x}, \mathbf{w}_i)$ is unknown, we can sample $\mathbf{w}_i$ according to $g(\mathbf{w}_i) = fr(\mathbf{x}, \mathbf{w}_i, \mathbf{w}_r) \cos(\theta_i)$.

2.3.1 Lambertian Model and Cosine Hemisphere Sampling

The brdf $fr(\mathbf{x}, \mathbf{w}_i, \mathbf{w}_r)$ of Lambertian surfaces is uniform. Therefore, the function $g(\mathbf{x}, \mathbf{w}_i, \mathbf{w}_r, \mathbf{n})$ is proportional only to $\cos(\theta_i)$ (which is the cosine of the incident angle).

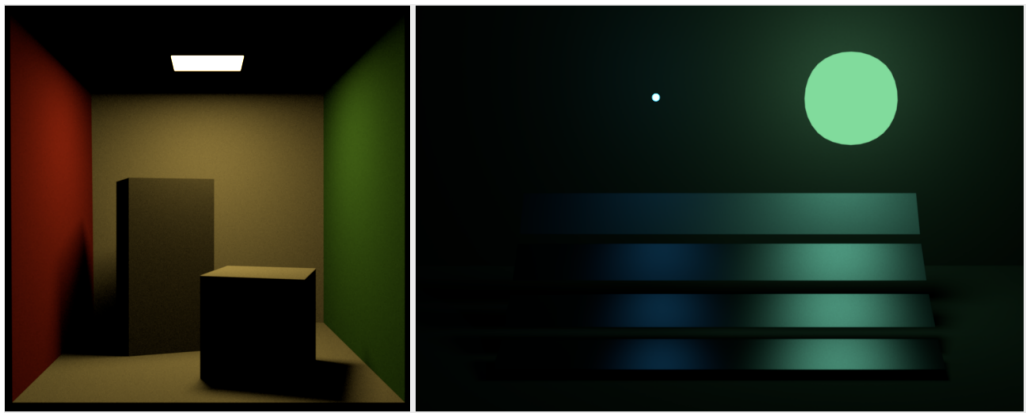

Figure 1: Rendering with the Lambertian model (left) and Phong model (right)

Question 2.3.1. Implement the function

```
Vector3f squareToCosineHemisphere(const Point2 &sample)
```

which receives a uniformly distributed sample over $[0;1] \times [0;1]$ and returns a unit vector over the hemisphere, sampled using cosine-weighted hemisphere sampling. The common way to generate a cosine weighted hemisphere sampler is to generate uniform points on a disk, and then project them up to the hemisphere. **(10 points)**

For this exercise, use the scene **diffuse-brdf.xml**.

2.3.2 Phong Sampling

As seen during the course, the Phong model takes into account a diffuse component as well as a specular one. Refer to the document `pdf/ex2/phong_importance_sampling.pdf` for a formal presentation of the Phong BRDF model. The reference may also give you important hints on the implementation details.

Question 2.3.2a. Implement the three methods of `phong.cpp`, which are:

- `Color3f eval(const BSDFQueryRecord &bRec)` **- 10 points**
  which evaluates the BRDF function $fr(\mathbf{x}, \mathbf{w}_i, \mathbf{w}_o)$ given incoming and outgoing directions.
- `float pdf(const BSDFQueryRecord &bRec)` **- 10 points**
  which returns the conditional probability $p(\mathbf{w}_i|\mathbf{x}, \mathbf{w}_o)$ of an incoming direction given the outgoing direction at a point $\mathbf{x}$. Explain in your report how you find such probability.
- `Color3f sample(BSDFQueryRecord &bRec, const Point2f &sample)` **- 10 points**
  which samples an incoming direction following the pdf. Then the output is $\frac{fr(\mathbf{x},\mathbf{w}_i,\mathbf{w}_o)\cos(\theta_i)}{p(\mathbf{w}_i|\mathbf{x},\mathbf{w}_o)}$.

Hint: Since the Phong model is a linear combination of a diffuse term (weighted by k_d) and a specular term (weighted by k_s), we can randomly choose to sample *either* the diffuse term or the specular term based on these weights (see the above document). These weights are later used to mix corresponding pdfs for an unbiased sampling. Explain why the combined pdf is unbiased.

Note: As the framework implements a forward path-tracer that shoots rays from the eye, the convention for outgoing and incoming rays are inverted ($\mathbf{w}_o$ is `bRec.wi` and $\mathbf{w}_i$ is `bRec.wo`).

For this exercise, use the scene **phong-brdf.xml**.

Question 2.3.2b. In order to evaluate the Phong BRDF model, you need to find the angle between the perfect specular reflective direction and the outgoing ray direction. How can you compute this angle, without explicitly calculating the reflected direction, given the incoming direction $\mathbf{w}_i$ and the surface normal $\mathbf{n}$ in world coordinates? **(5 points)**