DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS
COMENIUS UNIVERSITY

MODELING BIDIRECTIONAL SCATTERING DISTRIBUTION FUNCTION

Bachelor Thesis

JANA HLINKOVÁ

Advisor:
Dr. Tomáš Plachetka

Bratislava, 2007

I hereby declare I wrote this thesis by myself, only with the help of referenced literature, under the careful supervision of my thesis advisor.

……………………………………….

ACKNOWLEDGEMENTS

Most of all, I would like to thank my advisor Dr. Tomáš Plachetka for his guidance, devoted time and many helpful suggestions during my work on this thesis.
A special thanks goes to my family and friends for their support and encouragement, and for keeping an eye on me, while I was writing the thesis.

Abstract

Realistic simulation of scene illumination is mathematically supported by rendering equation. One of the parameters of this equation is the bidirectional scattering distribution function (BSDF). BSDF is a function, which defines optical properties of surfaces. Computer graphics uses several not compatible material models that approximate BSDF. Different representations of BSDF cannot be automatically converted into one another. Moreover, the choice of material model influences the choice of rendering algorithm and vice-versa. Furthermore, if BSDF representation is updated, renderer has to be changed as well.

The main goal of this thesis is to present an abstract representation of BSDF, which would tackle the problem of incompatibility by providing a simple interface — an interface that would cover all representations of BSDF and allow the evolution of BSDF models independently of the renderers.

Keywords: BSDF, bidirectional scattering distribution funcion, Monte Carlo path tracing

Contents

Chapter 1

Introduction

Photorealistic scene synthesis is unmistakably one of the major goals of computer graphics. Not only film and computer game production sets higher demands on the quality of created images. Computer simulators for pilots, students of medicine or soldiers present an excellent supplement in the process of training, and thus, the better the simulation follows reality, the better the training is.

To create realistic images, accurate graphical representation of objects and good physical description of lighting effects are needed. Modeling colors and lighting effects, which can be seen on objects of the real world, is a complex process that involves principles of both physics and psychology. In order for an image on computer screen to appear realistic, it has to evoke approximately the same color sensation in human eye and brain, as a real world image would. Human color perception depends on the electromagnetic radiance reaching the eye and on the operation of the eye. The amount of electromagnetic radiance that is radiated from a scene towards the viewer is defined by objects the scene contains — their shapes and optical properties, and by lighting properties of light sources. Thus it is necessary to understand the physics of light propagation and the function of the eye, and to build a mathematical model of lighting on top of this knowledge.

Virtual scenes are represented by a 3D model, which contains the description of geometry of the virtual world, description of optical material properties and lighting. In order to obtain an image from a 3D model, distribution of light in the scene has to be calculated. This is done by simulating real-world optical phenomena based on laws of physics. Calculation of light distribution independent of the viewing direction is called the global pass of rendering. After this stage, the amount of light energy reflected by surface points into different directions is determined. Since light is an electromagnetic wave, its distribution can be represented by wavelength-dependent functions. Rendering algorithms usually evaluate this functions at several representative wavelengths. The intensity of monochromatic light is described by radiance. The radiance leaving a surface point in a particular direction is affected by the emission of this point, the illumination provided by other surface points and the optical material properties of this point. This dependence is formally characterized by rendering equation, which is a Fredholm type integral equation of the second kind. Thus from mathematical point of view, the result of view independent rendering stage is the solution of rendering equation for representative wavelengths.

When a camera or other measuring device is placed into the scene, light distribution is measured from a given location and orientation. This view dependent step is the local pass of rendering. Some rendering algorithms first calculate global light distribution and then measure it by a measuring device, other do this simultaneously.

The result of rendering is a color pixel map or equivalent discrete sampling of radiance function. The final step of image synthesis is tone mapping - the conversion of computed radiance to R, G, B intensities, that can be produced by color monitor.

* * *

Bidirectional scattering distribution function (BSDF) plays a significant role in the rendering equation. BSDF describes optical material properties and so gives a description of light-surface interaction. Mathematically, it is a probability density function, which determines the probability of light scatter into a particular direction. Unfortunately, precise analytic form of this function is not known for natural surfaces, thus models, which only approximate it, are used instead. Several different models have been proposed so far, e.g. the Phong model and its modifications, physical microfacet model or data interpolation model. Different representations bring up a problem of compatibility, when material descriptions want to be shared. Material described by a table of representative functional values may not be compatible with rendering program designed for Phong material description. Uniform interface and object serialization could be a possible way of solving this problem.

Chapter 2

Global illumination

Global illumination models are physical models, which do not consider light emitters as the only source of light in the scene, but take into account realistic reflection of light as well. These models, of course, produce more realistic pictures than direct illumination models (models that consider only direct illumination by a light source), but they make the problem of calculating light distribution harder. Mathematical model for this problem is the rendering equation, which defines how the amount of light radiated from a particular surface point in particular direction is calculated. There are several different algorithms for solving the rendering equation, one of these — the Monte Carlo path tracing will be described in this chapter.

2.1 Rendering equation

2.1.1 Solid angle

The direction of light energy emission from a surface point can be described in an illumination sphere Ω or illumination hemisphere Ω_H, which contain those solid angles, to where the surface point can emit energy. A surface point of an opaque material can emit or reflect light only to the hemisphere that is above the surface, whereas surfaces of transparent materials emit light to the entire sphere.

Direction of light propagation ω is defined by two angles θ, ϕ, where θ is the angle between ω and z-axis, and ϕ is the angle between the projection of ω into x-y plane and the x-axis.

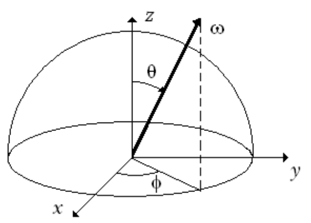

Figure 2.1: Solid angle ω defined by angles θ and ϕ

Sets of directions can be represented by solid angles. Differential (or infinitesimal) solid angle $d\omega$ is given by a vector $d\vec{\omega}$, where the vector equals to the direction of the differential set. It is possible to define differential solid angle by angles θ and ϕ. If θ is modified by $d\theta$ and ϕ by $d\phi$, the directional vector scans a differential rectangle with vertical size $d\theta$ and horizontal size $\sin\theta.d\phi$. Thus the size of the differential solid angle is:

$$d\omega = \sin\theta.d\phi d\theta. \tag{2.1}$$

Solid angle, in which a differential surface dA can be seen from point $\vec{p}$, is equal to the ratio of the area dA and the square of the distance r^2 between point $\vec{p}$ and the surface, if surface normal of dA is parallel to the directional vector from $\vec{p}$ to dA. If the angle between surface normal and the directional vector is θ, then the solid angle is:

$$d\omega = \frac{dA.\cos\theta}{r^2}. \tag{2.2}$$

2.1.2 Radiance

The most common metric used in computer graphics to characterize energy transfer is radiance (or intensity). Radiance $L(\vec{x},\omega)$ is defined as the differential light flux $d\Phi(\vec{x},dA,\omega,d\omega)$ leaving a surface element dA around point $\vec{x}$ in a differential solid angle $d\omega$ around ω, per the projected area of the surface element $dA.\cos\theta$ and the differential solid angle $d\omega$:

$$L(\vec{x},\omega) = \frac{d\Phi(\vec{x},dA,\omega,d\omega)}{dA.d\omega.\cos\theta}. \tag{2.3}$$

Light flux Φ (power) is a radiometric measure describing the amount of energy radiated through a boundary per unit time over a given range of spectrum. As light flux requires a definition of a boundary, it is more convenient to use radiance measure, which considers boundary in a differential way as a single surface point. Furthermore, as it is a standard to represent light as a combination of representative wavelengths of red, green and blue color in computer graphics, radiance is a suitable measure, as it characterizes monochromatic light.

2.1.3 Transfer probability density

The transfer probability density function describes the probability of reflection or refraction of light (photon) coming in from direction ω_{in} at point $\vec{x}$ into solid angle $d\omega_{out}$ around outgoing direction ω_{out}:

$$w(\vec{x},\omega_{in},\omega_{out}).d\omega_{out} = Pr\{\text{photon is reflected to } d\omega_{out} \text{ around } \omega_{out} | \text{ coming from } \omega_{in}\} \tag{2.4}$$

This density function does not refer to a pure continuous probability distribution, because the probability of light being reflected to the ideal reflection direction may be non-zero. (For continuous random variables, the probability, that the value of random variable is equal to some particular value, is zero.)

2.1.4 Rendering equation

The rendering equation determines the amount of light that is radiated from a surface point $\vec{x}$ into direction ω_{out}. This radiation consists of two components — the emitted radiation and reflected (or refracted) radiation, which can be dealt with separately.

Let $\Phi_{in}\left(\vec{x},dA,\omega_{in},d\omega_{in}\right)$ be the incoming light flux coming to area dA around point $\vec{x}$ from solid angle $d\omega_{in}$ around direction ω_{in} and let $w\left(\vec{x},\omega_{in,}\omega_{out}\right).d\omega_{out}$ be the probability of reflection into $d\omega_{out}$, if light arrives at $\vec{x}$ from direction ω_{in}. Then the reflected or refracted light flux leaving $\vec{x}$ in direction ω_{out} is:

$$\Phi_{out}\left(\vec{x},dA,\omega_{out},d\omega_{out}\right)=w\left(\vec{x},\omega_{in},\omega_{out}\right).d\omega_{out}.\Phi_{in}\left(\vec{x},dA,\omega_{in},d\omega_{in}\right) \tag{2.5}$$

The total amount of light reflected/refracted in $\vec{x}$ in direction ω_{out} is effected by all the incoming light power from every direction:

$$\Phi_{out}\left(\vec{x},dA,\omega_{out},d\omega_{out}\right)=\int_{\Omega}w\left(\vec{x},\omega_{in},\omega_{out}\right).d\omega_{out}.\Phi_{in}\left(\vec{x},dA,\omega_{in},d\omega_{in}\right). \tag{2.6}$$

Finally, the total light power leaving surface point $\vec{x}$ in direction ω_{out} is the sum of the emission and the reflected/refracted power:

$$\Phi_{out}\left(\vec{x},dA,\omega_{out},d\omega_{out}\right)=\Phi_{e}\left(\vec{x},dA,\omega_{out},d\omega_{out}\right)+\int_{\Omega}w\left(\vec{x},\omega_{in},\omega_{out}\right).d\omega_{out}.\Phi_{in}\left(\vec{x},dA,\omega_{in},d\omega_{in}\right). \tag{2.7}$$

The incoming light flux at point $\vec{x}$ from direction ω_{in} is equal to the light flux that is going out of a surface point $\vec{x'}$, if $\vec{x'}$ is a point visible from $\vec{x}$ in direction opposite to ω_{in}. This follows from the fundamental law of photometry:

$$d\Phi\left(\vec{x'},dA',\omega_{in},d\omega_{in}\right)=L\left(\vec{x'},\omega_{in}\right).\frac{dA'.\cos\theta'.dA.\cos\theta}{r^2}, \tag{2.8}$$

where dA is differential surface around $\vec{x}$, dA' differential surface around $\vec{x'}$, θ the angle between ω_{in} and dA surface normal, θ' the angle between $\omega_{in}$and surface normal of dA'. Using equation (2.2) the light flux leaving differential surface dA' and reaching differential surface dA is:

$$d\Phi_{out}\left(\vec{x'},dA',\omega_{in},d\omega_{in}\right)=L_{out}\left(\vec{x'},\omega_{in}\right).dA'.\cos\theta'.d\omega_{in}=L\left(\vec{x'},\omega_{in}\right).\frac{dA'.\cos\theta'.dA.\cos\theta}{r^2}=$$

$$=L_{in}\left(\vec{x},\omega_{in}\right).dA.\cos\theta.d\omega_{in}=d\Phi_{in}\left(\vec{x},dA,\omega_{in,}d\omega_{in}\right) \tag{2.9}$$

Expressing emitted, incoming and outgoing light power as:

$$\Phi_{e}\left(\vec{x},dA,\omega_{out},d\omega_{out}\right)=L_{e}\left(\vec{x},\omega_{out}\right).dA.\cos\theta_{out}.d\omega_{out}, \tag{2.10}$$

$$\Phi_{in}\left(\vec{x},dA,\omega_{in},d\omega_{in}\right)=L_{in}\left(\vec{x},\omega_{in}\right).dA.\cos\theta_{in}.d\omega_{in}, \tag{2.11}$$

$$\Phi_{out}\left(\vec{x},dA,\omega_{out},d\omega_{out}\right)=L_{out}\left(\vec{x},\omega_{out}\right).dA.\cos\theta_{out}.d\omega_{out}, \tag{2.12}$$

substituting into equation (2.7) and dividing both sides by $dA.\cos\theta_{out}.d\omega_{out}$ we get:

$$L_{out}\left(\vec{x},\omega_{out}\right)=L_{e}\left(\vec{x},\omega_{out}\right)+\int_{\Omega}L_{in}\left(\vec{x},\omega_{in}\right).\cos\theta_{in}.\frac{w\left(\vec{x},\omega_{in},\omega_{out}\right)}{\cos\theta_{out}}.d\omega_{in}. \tag{2.13}$$

As radiance $L_{in}(\vec{x}, \omega_{in})$ is equal to the radiance $L_{out}\left(\vec{x'}, \omega_{in}\right)$ if $\vec{x'}$ is a surface point visible from point $\vec{x}$ in direction $-\omega_{in}$, the equation can be rewritten in form:

$$L_{out}(\vec{x}, \omega_{out}) = L_e(\vec{x}, \omega_{out}) + \int_{\Omega} L_{out}(h(\vec{x}, -\omega_{in}), \omega_{in}) . \cos\theta_{in} . \frac{w(\vec{x}, \omega_{in}, \omega_{out})}{\cos\theta_{out}} . d\omega_{in}, \tag{2.14}$$

where $h(\vec{x}, -\omega_{in})$ is visibility function returning a point visible from point $\vec{x}$ in direction $-\omega_{in}$ $(h(\vec{x}, -\omega_{in}) = \vec{x'})$.

Defining bidirectional scattering distribution function (BSDF)

$$f(\vec{x}, \omega_{in}, \omega_{out}) = \frac{w(\vec{x}, \omega_{in}, \omega_{out})}{\cos\theta_{out}} \tag{2.15}$$

and substituting into (2.14), we obtain the *rendering equation*:

$$L_{out}(\vec{x}, \omega_{out}) = L_e(\vec{x}, \omega_{out}) + \int_{\Omega} L_{out}(h(\vec{x}, -\omega_{in}), \omega_{in}) . f(\vec{x}, \omega_{in}, \omega_{out}) . \cos\theta_{in} . d\omega_{in} \tag{2.16}$$

2.2 3D model

Having established the mathematical model for light transfer description, it is essential to choose a compatible representation of 3D scene elements. For instance, the representation of surface geometry should be able to provide a surface normal for any point of the surface and the optical material properties should be characterized by bidirectional scattering distribution function.

2.2.1 Geometry of surfaces

The most common representation of object surfaces is the polygon representation. Objects are approximated by polygon meshes (e.g. triangle mesh). The triangle mesh approximation of smoothly curved surfaces is called tessellation. Triangle meshes are very popular, as they reduce many problems that emerge when surfaces are approximated by polygons, which have more than three vertices. For instance, if a polygon has more than three vertices, than due to numerical errors, vertices may end up not lying in one plane, even if in reality they do belong to the same plane. This problem does not apply to triangles, as three vertices define a unique plane.

Polygon surfaces are specified with a set of vertex coordinates and associated attribute parameters. Usually, besides a vertex and a polygon-surface table for each object, additional information such as surface normal orientation is stored (normals that are usually stored are normals at triangle vertices, to avoid the discontinuity of normals at edges of neighboring triangles; the normals at inside points are then linearly interpolated). The polygonal surface representation allows conveniet texture mapping and conveniet definition of local material properties such as transparency or reflectivity.

A serious drawback of polygon representation is the fixed degree of approximation. The number of triangles that represent a surface is fixed. When resolution is increased, the smoothness of surface disappears.

Another possible way of representing three-dimensional objects presents constructive solid geometry (CSG). Complex objects are created from primitive 3D objects using boolean set operations (union, intersection, difference). Unary operations such as rotation, translation and scaling may be applied to any surface. CSG objects may be represented by a tree structure, where leaves represent object primitives and inner nodes store unary or binary operations. As algorithms that compute surface normals for object primitives exist, it is also possible to determine the normals form composite objects. The same applies to computation of object and line intersection.

2.2.2 Material properties

The ultimate objective of photorealistic virtual scene rendering is to capture the optical surface material properties, such as color, mirror like reflection or transparency. Optical properties of a material depend on the microstructure of the surface. Usually microstructure is not included in the object's geometry description, as it would be both memory consuming and computationally expensive. Instead, optical properties (characterizing reflection, refraction, etc.) are described by material functions.

Two basic types of surfaces are diffuse and specular surfaces. Surfaces that are rough or grainy tend to scatter (reflect or refract) light in all directions. The scattering of light equally into all directions is the diffuse scattering. As equal amount of light is radiated into each direction, these surfaces appear equally bright from every viewing direction. Specular reflection is typical for shiny surfaces, such as metals. Most of the incident light is reflected to a solid angle around the angle of perfect reflection. Different kinds of surfaces are often represented as some combination of these two extreme types (e.g. in Phong model).

Mathematically, surface optical properties are described as a probability of light scatter (e.g. the transfer probability density function, BSDF). Different approaches to the representation of bidirectional scattering distribution function will be discussed in chapter 3.

2.2.3 Light sources

As light sources may be considered either objects that emit light energy (light bulb, sun) or objects that reflect light (walls). Later on, only self-emitting objects will be referred to as light sources.

The simplest model of a light emitter is a point source. Point source has a zero size and emits light radially to every direction. This type of source is used to model light sources that are small in comparison to the objects of the scene (light bulb), or that are located far from the scene (sun). Nearby sources of non-negligible sizes (long fluorescent light) are modeled by distributed light sources. This model considers accumulated illumination effects of point-sources approximating the surface of the source. Sky-light illumination model provides constant illumination at any point and in any direction.

Light sources in a virtual scene are characterized by their location, size and direction and intensity of emission (radiance L_e).

2.3 Bidirectional scattering distribution function (BSDF)

BSDF function characterizes optical material properties. It provides mathematical description of light-surface interaction. For every surface point $\vec{x}$ and every incomming direction ω_{in}, BSDF returns a value determining the portion of light energy $dL_{out}(\vec{x}, \omega_{out})$, that is scattered into outgoing direction ω_{out}:

$$f(\vec{x}, \omega_{in}, \omega_{out}) = \frac{dL_{out}(\vec{x}, \omega_{out})}{L_{in}(\vec{x}, \omega_{in}) . \cos\theta_{in} . d\omega_{in}} \tag{2.17}$$

($L_{in}(\vec{x}, \omega_{in})$is the total incomming radiance in direction ω_{in}, $d\omega_{in}$ is the solid angle around ω_{in}, θ_{in} is the angle between ω_{in} and the surface normal at $\vec{x}$.)

Again, BSDF is also dependent on the wavelength, but for simplicity, the wavelength argument is left out. In practice, BSDF functions are defined for the three representative wavelengths of R, G, B.

BSDF must satisfy laws of physics in order to be of any use in photorealistic rendering. Physically plausible BSDF's must satisfy the law of reciprocity and the energy conservation law. Helmholtz reciprocity describes the symmetry property of BSDF:

$$f(\vec{x}, \omega_{in}, \omega_{out}) = f(\vec{x}, \omega_{out}, \omega_{in}) .BSDF's \tag{2.18}$$

This symmetry property allows backward tracing of light in visibility ray-tracing algorithms (e.g. Monte Carlo path tracing algorithm). From the reversed point of view, BSDF determines the probability that the light, which is reflected in direction ω_{out} came from direction ω_{in}.

Energy conservation law states, that the amount of reflected energy cannot be higher than the amount of incoming energy. More precisely, the ratio of the total reflected energy and the incoming energy cannot be greater than 1. Equivalently, the probability of reflection to any direction cannot be greater than 1. These properties are defined as albedo (or total hemisphirical reflectivity) :

$$a(\vec{x}, \omega_{in}) = Pr\{\text{ photon is reflected} \mid \text{coming form } \omega_{in}\} = \int_{\Omega_H} f(\vec{x}, \omega_{in}, \omega_{out}) . \cos\theta_{out} . d\omega_{out} \leq 1. \tag{2.19}$$

2.4 Monte Carlo path tracing algorithm

The aim of Monte Carlo path tracing algorithm is to solve the rendering equation:

$$L_{out}(\vec{x}, \omega_{out}) = L_e(\vec{x}, \omega_{out}) + \int_{\Omega} L_{out}(h(\vec{x}, -\omega_{in}), \omega_{in}) . f(\vec{x}, \omega_{in}, \omega_{out}) . \cos\theta_{in} . d\omega_{in} \tag{2.20}$$

A more compact way of representing the rendering equation is the operator form:

$$L_{out} = L_e + K \circ L_{out}. \tag{2.21}$$

This operator equation, just like the rendering equation may be solved by iteration:

$$L_{out}^0 = L_e$$

$$L_{out}^1 = L_e + K \circ L_{out}^0 = L_e + K \circ L_e$$

$$L_{out}^2 = L_e + K \circ L_{out}^1 = L_e + K \circ (L_e + K \circ L_e) = L_e + K \circ L_e + K^2 \circ L_e$$

$$.......$$

$$L_{out}^n = \sum_{j=0}^{n} K^j \circ L_e, \tag{2.22}$$

where $K^0 = I$, there I is the identity operator. In case of infinite sum, the closed form is:

$$L_{out} = \sum_{j=0}^{\infty} K \circ L_e = (I - K)^{-1} \circ L_e, \tag{2.23}$$

presenting the solution to the rendering equation.

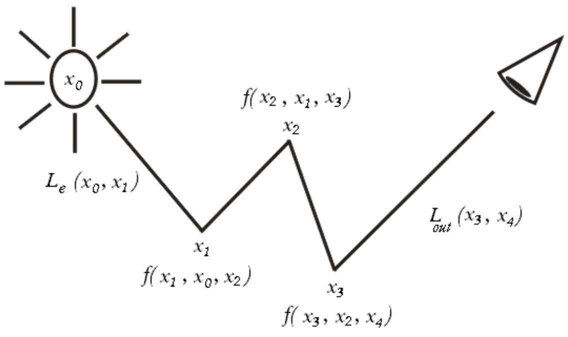

Figure 2.2: Light path

Relating the mathematical notation to the real world, the above formula describes journey of light from a light source to a measuring device. $K^n \circ L_e$ describes the intensity of light on the $(n+1)$st segment of the light path:

$$L_{out}(\vec{x}_n, \vec{x}_{n+1}) = K^n \circ L_e = \tag{2.24}$$

$$= \int_\Omega ... \int_\Omega L_e(\vec{x}_0, \vec{x}_1) . f(\vec{x}_1, \vec{x}_0, \vec{x}_2) ... f(\vec{x}_n, \vec{x}_{n-1}, \vec{x}_{n+1}) . \cos\theta_0 ... \cos\theta_{n-1} . d\omega_0 ... d\omega_{n-1}.$$

Thus the overall radiance leaving $\vec{x}$ in direction ω_{out} can be calculated as the sum of integrals:

$$L_{out}(\vec{x}, \omega_{out}) = L_e + \int_\Omega L_e(h(\vec{x}, -\omega_{in}), \omega_{in}) . f(\vec{x}, \omega_{in}, \omega_{out}) . \cos\theta_{in} . d\omega_{in} +$$

$$+ \int_\Omega \int_\Omega L_e() . f() . f() . \cos() . \cos() . d\omega . d\omega \; +$$

$$.... + \int_\Omega ... \int_\Omega L_e() . f() .... f() . \cos() ..... \cos() . d\omega .... d\omega \; + ... \tag{2.25}$$

It is not possible to give an exact result of the rendering equation, due to the integration over entire illumination sphere and the infinite recurrence. Therefore, different kinds of simplification were presented and algorithms solving the simplified versions of rendering equation were developed. The Monte Carlo path tracing algorithm is based on the Monte Carlo integration — approximation of integral.

Path tracing is a Markov chain random walk technique for solving the rendering equation. For each pixel, the incoming radiance is estimated by backward tracking of rays. First, visible surface point is found as the intersection of a line defined by the position of the eye and the position of the pixel, and an object from 3D scene. To estimate the radiance $L_{out}\left(\vec{x}, \omega_{out}\right)$ radiated from the visible surface point in the direction of the intersecting line ω_{out}, the radiance contribution of the most influential incoming directions is summed. More precisely, instead of summing up the amount of light power reflected in direction ω_{out} from every incoming direction ω_{in}, only the most important incoming directions are chosen. For each chosen important incoming direction ω_{in} the incoming radiance is calculated as the outgoing radiance of a surface point $\vec{x'}$ in direction ω_{in}, such that $\vec{x'} = h\left(\vec{x}, -\omega_{in}\right)$. Again, the outgoing radiance $L_{out}\left(\vec{x'}, \omega_{in}\right)$ is the sum of the most influential incoming radiances for point $\vec{x'}$ and direction ω_{in}. This is recursively repeated until a light source is hit, or until the maximum number of steps is reached.

The determination of important incoming directions is done according to the bidirectional scattering distribution function (BSDF), which determines the probability, that radiance reflected at a surface point $\vec{x}$ into direction ω_{out}, came from direction ω_{in} (the reciprocal property of BSDF).

Chapter 3

Reflection models

Optical material properties are physically precisely described by bidirectional scattering distribution function. As it is not possible to use the exact definition of BSDF for rendering, computer graphics uses practical reflection models, which model some approximation of BSDF. Each model of BSDF should be physically plausible, i.e. it should satisfy the reciprocity property and the law of energy conservation.

3.1 Diffuse reflection model

Rough surfaces such as walls scatter incident light equally into all directions. This view and position independent type of reflection is called diffuse reflection. If the reflection is independent of viewing (outgoing) direction, then thanks to reciprocal property of BSDF, it is also independent of the incoming direction of light. The fractional amount of light diffusely reflected can be for each surface represented by diffuse-reflection coefficient, or diffuse reflectivity:

$$f\left(\vec{x}, \omega_{in}, \omega_{out}\right) = k_{\text{diffuse}}. \tag{3.1}$$

In order to be physically plausible, the law of energy conservation must be obeyed. Thus the albedo has to be less than 1:

$$a\left(\vec{x}, \omega_{in}\right) = \int_{\Omega_H} k_{\text{diffuse.}} \cos\theta_{out}.d\omega_{out} = \tag{3.2}$$

$$= \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\frac{\pi}{2}} k_{\text{diffuse}}. \cos\theta_{out}. \sin\theta_{out}.d\theta_{out}.d\phi_{out} = k_d.\pi \leq 1$$

and so:

$$k_{\text{diffuse}} \leq \frac{1}{\pi}. \tag{3.3}$$

3.2 Ideal reflection model

Ideal mirror-like reflection satisfies the reflection law of geometric optics — the angle of reflection θ_{out} is equal to the incoming angle θ_{in}, and the reflected beam stays in the same plane. Light is reflected only in the direction of ideal reflection. Thus BSDF can be defined as a Dirac-delta function:

$$f\left(\vec{x},\omega_{in},\omega_{out}\right)=k_{\text{ideal}}\left(\theta_{in}\right)\frac{\delta\left(\omega_{ideal}-\omega_{out}\right)}{\cos\theta_{in}}. \tag{3.4}$$

The albedo ideal mirror-like reflection is:

$$a\left(\vec{x},\omega_{in}\right)=\int_{\Omega_H}k_{\text{ideal}}\left(\theta_{in}\right)\frac{\delta\left(\omega_{ideal}-\omega_{out}\right)}{\cos\theta_{in}}.\cos\theta_{out}.d\omega_{out}=k_{ideal}\left(\theta_{in}\right)\leq 1,$$

thus the value $k_{\text{ideal}}\left(\theta_{in}\right)$ cannot be greater than 1.

3.3 Phong specular reflection model

Shiny objects, which are not ideal reflectors, reflect most of the light into a solid angle around the direction of ideal reflection. Bidirectional scattering distribution function in Phong modeled is defined as:

$$f\left(\vec{x},\omega_{in},\omega_{out}\right)=k_{\text{specular}}\frac{\cos^n\Phi}{\cos\theta_{in}}, \tag{3.5}$$

where k_{specular} is the specular-reflection coefficient and Φ is the angle between the angle of viewing direction ω_{out} and the angle of ideal reflection. However, this model violates the reciprocal property of BSDF, therefore following reciprocal model is preferred by photorealistic algorithms:

$$f\left(\vec{x},\omega_{in},\omega_{out}\right)=k_{\text{specular}}\cdot\cos^n\Phi. \tag{3.6}$$

To satisfy the energy conservation law, the following restriction for the specular-reflection coefficient k_{specular} must not be violated:

$$k_{\text{specular}}\leq\frac{n+2}{2\pi}. \tag{3.7}$$

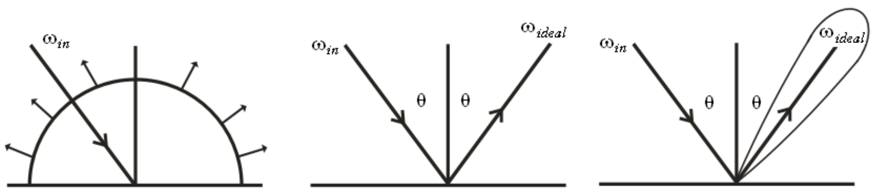

Figure 3.1: Diffuse reflection, ideal reflection, specular reflection

3.4 Combined Phong reflection model

Phong model was the first reflection model developed in computer graphics and is still most commonly used. It divides reflectivity into a diffuse and a specular component. Bidirectional scattering distribution function is defined as:

$$f\left(\vec{x},\omega_{in},\omega_{out}\right)=f_d\left(\vec{x},\omega_{in},\omega_{out}\right)+f_s\left(\vec{x},\omega_{in},\omega_{out}\right)=k_{\text{diffuse}}+k_{\text{specular}}\cos^n\Phi,$$

where k_{diffuse} is the fraction of energy reflected diffusely, k_{specular} the fraction reflected specularly, and Φ is the angle between direction of ideal reflection and viewing direction ω_{out}.

3.5 Physical microfacet model

In the physical model, surface roughness is modeled by randomly scattered and oriented microfacets. Each microfacet is a perfect mirror reflector. The scattered light can be again divided into diffuse and specular component. The diffuse component is simulated by multiple reflections on the microfacets (as it is thought to be in reality), the specular reflection is produced by direct reflection from a microfacet. This model gives very good results — high quality of rendered pictures, but computationally it is very expensive.

3.6 Data interpolation

Data interpolation model is a general model that can be used for any type of light scatter. Exact values of BSDF are measured for representative incoming and outgoing directions, and representative wavelengths, using a spectrometer. Values for desired incoming and outgoing direction are calculated by interpolation. As interpolation is not a time-consuming operation, desired values of BSDF can be calculated quickly. The disadvantage of this solution is the high memory demand.

Chapter 4

Importance sampling

Importance sampling of directions plays a significant role in the Monte Carlo path tracing algorithm. In the estimation of outgoing radiance $L_{out}(\vec{x}, \omega_{out})$ form point $\vec{x}$ in direction ω_{out}, only the contribution of radiance from most influential incoming directions is taken into account. The most important incoming directions are generated according to bidirectional scattering distribution function (e.g. in case of specular reflection, most of the generated directions should belong to a close neighborhood of the ideal reflection direction).

Ideal sampling should follow probability distribution function defined as:

$$pdf(\omega_{out}) = \frac{f(\vec{x}, \omega_{out}, \omega_{in}) \,.\, \cos\theta_{out}}{\int_{\Omega_H} f(\vec{x}, \omega_{out}, \omega_{in}) \,.\, \cos\theta_{out} . d\omega_{out}},$$

in practice though, approximations of *pdf* function are used instead.

4.1 Modified Phong model

In this section, a method of sampling based on [LW94] is described.

The modified Phong model defines BSDF as:

$$f(\vec{x}, \omega_{in}, \omega_{out}) = k_d \frac{1}{\pi} + k_s \frac{n+2}{2\pi} \cos^n \alpha, \tag{4.1}$$

where k_d is diffuse reflectivity coefficient, k_s specular reflectivity coefficient, n is specular exponent and α is the angle between ideal reflection direction and outgoing direction.

This model is physically plausible. Reciprocity property is obviously satisfied, as α remains the same ($\alpha = |\omega_{in} - \omega_{out}|$), and k_d, k_s and n are constants.

Energy conservation law is satisfied if $k_d + k_s \leq 1$:

$$a(\vec{x}, \omega_{in}) = \int_{\Omega_H} \left(k_d \frac{1}{\pi} + k_s \frac{n+2}{2\pi} \cos^n \alpha \right) \cos\theta_{out} . d\omega_{out} = \tag{4.2}$$

$$= k_d + k_s \frac{n+2}{2\pi} \int_{\Omega_H} \cos^n \alpha . \cos\theta_{out} . d\omega_{out} = a_d(\vec{x}, \omega_{in}) + a_s(\vec{x}, \omega_{in}) \leq 1,$$

the maximum of $\int_{\Omega_H} \cos^n \alpha . \cos\theta_{out} . d\omega_{out}$ is $\frac{2\pi}{n+2}$, and so $k_d + k_s \leq 1$.

As the model is given in closed form, we are able to evaluate BSDF for every point, incoming and outgoing direction, what is important for sampling.

Sampling of incoming directions ω_{in} for outgoing direction ω_{out} is done as follows: first is decided, whether a diffuse direction or specular direction is sampled. A stochastic variable ξ is uniformly sampled over the interval $[0, 1]$. Then if:

- $0 \leq \xi < a_d(\vec{x}, \omega_{out})$: diffuse sample is sampled according to diffuse distribution,
- $a_d(\vec{x}, \omega_{out}) \leq \xi < a_d(\vec{x}, \omega_{out}) + a_s(\vec{x}, \omega_{out})$: specular sample is sampled according to specular distribution,
- else: another $\xi$is sampled.

The value of $a_d(\vec{x}, \omega_{out})$ is k_d. The value of $a_s(\vec{x}, \omega_{out})$ is approximated as $k_s \frac{n+2}{n+1}. \cos\theta_{out}$.

Diffuse samples are sampled according to cosine distribution:

$$pdf\left(\omega_{out}\right) = \frac{1}{\pi}. \cos\theta_{out}. \tag{4.3}$$

Specular samples are sampled according to distribution:

$$pdf\left(\omega_{out}\right) = \frac{n+1}{2\pi} \cos^n \alpha. \tag{4.4}$$

Two stochastic variables ξ_1 and ξ_2 are sampled uniformly over the interval $[0, 1]$. Angles θ_{in} and ϕ_{in} of the sampled incoming direction are determined as:

For diffuse distribution $pdf\left(\omega_{out}\right) = \frac{1}{\pi}. \cos\theta_{out}$:

$$\left(\theta_{in},\ \phi_{in}\right) = \left(\arccos\sqrt{\xi_1},\ 2\pi\xi_2\right). \tag{4.5}$$

If direction is described with vector coordinates $(x,\ y,\ z)$, then:

$$\begin{aligned}(x,\ y,\ z) &= \left(\sin\theta_{in}. \cos\phi_{in}, \sin\theta_{in}. \sin\phi_{in}, \cos\theta_{in}\right) = \\ &= \left(\sqrt{1-\xi_1} cos\left(2\pi\xi_2\right), \sqrt{1-\xi_1} sin\left(2\pi\xi_2\right), \sqrt{\xi_1}\right).\end{aligned} \tag{4.6}$$

Sampling the specular distribution $pdf\left(\omega_{out}\right) = \frac{n+1}{2\pi} \cos^n \alpha$, at first $(\alpha,\ \alpha_\phi)$ is determined:

$$\left(\alpha,\ \alpha_\phi\right) = \left(\arccos \xi_1^{\frac{1}{n+1}},\ \arccos \xi_2^{\frac{1}{n+1}}\right), \tag{4.7}$$

then $\left(\theta_{in},\ \phi_{in}\right) = \left(\theta_{out} \pm \alpha,\ \phi_{out} + \pi \pm \alpha_\phi\right)$.

In terms of direction vector coordinates $(x,\ y,\ z)$:

$$(x,\ y,\ z) = \left(\sin\theta_{in}. \cos\phi_{in},\ \sin\theta_{in}. \sin\phi_{in},\ \cos\theta_{in}\right). \tag{4.8}$$

Chapter 5

Uniform material interface

The variety of material descriptions brings up the problem of compatibility. A material, which is described by constants k_d, k_s and n, as it is in Phong model, does not have to fit with a rendering program designed for microfacet model. Often it would be appreciated, if compatibility was possible. One would be able to render scenes, where materials are described by different BSDF models, with a single program. Also, the quality of rendering could be easily regulated with the choice of material description, while using the same rendering program.

Let's consider the Monte Carlo path tracing algorithm. In the determination of outgoing radiance $L_{out}(\vec{x}, \omega_{out})$, directions of incoming radiance are sampled according to BSDF. Since the only role of a material is to provide BSDF value $f(\vec{x}, \omega_{in}, \omega_{out})$ and to specify optimal directions for path tracing, it would be enough, if the direction generation was implemented by material itself, the generator would be presented as an interface, and the reflectance model would be hidden for the rest of the program.

5.1 Object serialization

Objects that are created during an application exist only when the application is running. Object serialization allows objects created during the runtime of a program to be stored and reused again. Serialized objects, flattened to a stream of bites, can be sent via networks and used by different computers, even running a different operating system. There is no problem in reusing serialized objects even if the code of the class has been changed in the meantime.

Serialization proposes benefits for class hierarchies as well. Let's say there is an abstract class animal, which has abstract methods draw() and sound(). Then, classes sheep, goldfish and marmot extend this class, and implement methods draw() and sound() each in it's own way, and maybe define new methods. During an application, object instances of sheep, goldfish and marmot are created and serialized. In another application, these objects may instance the abstract class animal, and still, each of the objects will be drawn in a correct way. Furthermore, let's say a buddy in Africa creates new extension of animal class, e.g. giraffe, implements draw() and sound(), creates an object instance of giraffe, serializes it, and sends it to his friend from Slovakia. Then the Slovak friend may listen to the sound of giraffe, even though he does not have the source code of giraffe class.

5.2 Implementation

Based on principles that were mentioned earlier in this thesis, uniform material interface was proposed and partially implemented. The interface was designed as follows.

Abstract class Material presents the uniform interface for every BSDF representation. This class contains three abstract methods: a method that returns BSDF value for a given point, incoming direction and outgoing direction, a method that generates directions according to BSDF and returns direction defined by two angles θ and ϕ, and a method that generates vectors according to BSDF. For every BSDF representation, class that extends the abstract class Material can be implemented. Every extension of abstract class Material implements methods BSDF, Generate_direction and Generate_vector according to the chosen BSDF representation. The specific design of methods is:

- public *double* BSDF(Point x, Direction in, Direction out) — takes vector coordinates of a point, incoming direction and outgoing direction as arguments, returns value $f\,(\vec{x}, \omega_{in}, \omega_{out})$
- public *Direction* Generate_direction(Point x, Direction out) — takes vector coordinates of a point, outgoing direction as arguments, returns incoming direction generated according to BSDF defined by angles $\theta$and ϕ
- public *Vector* Generate_vector(Point x, Vector out) — takes vector coordinates of a point, vector coordinates of outgoing direction as arguments, returns vector coordinates of incoming direction generated according to BSDF

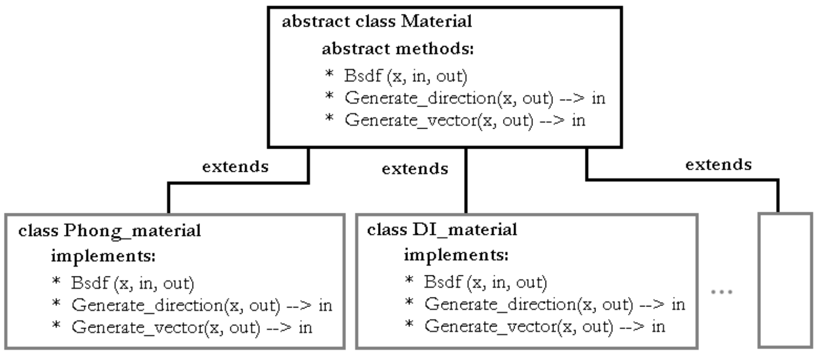
abstract class Material
abstract methods:
* Bsdf (x, in, out)
* Generate_direction(x, out) --> in
* Generate_vector(x, out) --> in
extends
extends
extends
class Phong_material
implements:
* Bsdf (x, in, out)
* Generate_direction(x, out) --> in
* Generate_vector(x, out) --> in
class DI_material
implements:
* Bsdf (x, in, out)
* Generate_direction(x, out) --> in
* Generate_vector(x, out) --> in
...

Figure 5.1: Class hierarchy

Now it will be discussed, how the uniform material interface solves the problem of compatibility. Let's consider the Monte Carlo path tracing algorithm. As was already stated above, material properties play a role only in the determination of the direction of light path for backward tracing, and in the calculation of contribution of incident radiance. Thus, the interface provided by abstract class Material is fully sufficient. If Monte Carlo algorithm is implemented using this simple interface, compatibility of material models is guaranteed.

Say one person implements class Phong_material that extends abstract class Material. This implementation is bound with Phong reflection model, material optical properties are described by constants k_d , k_s and n. He creates an instance of Phong_material class and serializes it.

An instance of Phong_material is a concrete material type, let's say silver. Another person uses data interpolation material model and implements DI_material also extending abstract class Material. As an object instance of DI_material class, cotton cloth material is defined and again serialized. Now both of them can render a scene containing silver coins on cotton table cloth, even though they do not know, how the other material was implemented.

Based on the Modified Phong model and the work of [LW94], class Phong_material, extending abstract class Material, was implemented in Java. The key ideas essential for the implementation are described in chapter 4. The abstract class Material was implemented in Java as well.

Chapter 6

Conclusion

In this thesis, one of the important parameters of rendering equation — the bidirectional scattering distribution function (BSDF) has been studied. Different possible ways of BSDF representation have been described and a method of unifying them, based on abstract BSDF representation, designed for Monte Carlo path tracing algorithm, was proposed. The abstraction was based on the reduction of the concept of material to a simple interface. This interface was implemented as an abstract class which could be later on used as a plugin for rendering programs. Basic ideas of implementing a subclass extension of the abstract material class, using Modified Phong model, were described, and this subclass was implemented.

One possible direction of future work could definitely be the implementation of the numerous BSDF representations, and so finally the essence of the presented ideas would be fully demonstrated. Another possibility could be to adjust the abstraction to other rendering algorithms.

List of Figures

Bibliography

[LW94] Lafortune, E. P., Willems, Y. D., *Using the Modified Phong Reflectance Model for Physically Based Rendering*, Report CW 197, Department of Computing Science, K. U. Leuven, 1994.

[HD97] Hearn, D., Baker, M. P., *Computer Graphics - C Version*, 2nd ed., Prentice Hall, Inc., New Jersey, 1997.

[WW92] Watt, A., Watt, M., *Advanced Animation and Rendering Techniques - Theory and Practice*, ACM Press, 1992.

[P03] Plachetka, T., *Event-Driven Message Passing and Parallel Simulation of Global Illumination*, University of Paderborn, 2003.

[L00] László, S. K., *Monte-Carlo Methods in Global Illumination*, Institute of Computer Graphics, Vienna University of Technology, 2000.

[JAVA] http://java.sun.com/developer/technicalArticles/Programming/serialization

Abstrakt

Tvorba realisticky vyzerajúcich scén je nepochybne jedným z hlavných cieľov počítačovej grafiky. Nielen vo filmovej produkcii či počítačových hrách sa kladú vyššie a vyššie nároky na kvalitu vytvorených obrázkov. Počítačové simulátory sa využívajú na tréning pilotov, študentov medicíny, či pri vojenskom výcviku. Prirodzene, čím lepšie dokáže simulátor navodiť dojem reality, tým kvalitnejší je tréning. Kľúčovým aspektom pri syntéze realistických scén je simulácia distribúcie svetla – simulácia interakcie svetla a objektov v scéne.

Realistická simulácia osvetlenia virtuálnej scény je matematicky popísaná renderovacou rovnicou. Jedným z parametrov tejto rovnice je rozptylová funkcia – bidirectional scattering distribution function (BSDF). Táto funkcia popisuje optické povrchové vlastnosti a tak definuje rôzne typy materiálov.

V počítačovej grafike sa používa niekoľko rôznych reprezentácií funkcie BSDF – rôzne reflekčné modely (Phongov reflekčný model a jeho modifikácie, fyzikálny mikroplôškový model, interpolačný model, a.i.). Tieto modely však nie sú navzájom kompatibilné. Nie je možné automaticky konvertovať jednu reprezentáciu na druhú. Problémom je taktiež úzka väzba medzi reflekčným modelom a renderovacím programom. Program navrhnutý pre jednu reprezentáciu BSDF nemusí akceptovať materiál popísaný inou reprezentáciou. Tak isto, zmeny v reprezentácii BSDF si vyžiadajú zmeny v renderovacom programe a naopak.

Hlavným cieľom tejto práce je navrhnúť abstraktnú reprezentáciu funkcie BSDF, ktorá by obišla problém nekompatibility pomocou jednoduchého jednotného rozhrania. Vďaka tomuto rozhraniu by sa reprezentácia BSDF stala pre zvyšok renderovacieho programu neviditeľná. Renderer by sa tým pádom stal nezávislý od materiálového popisu, čo by umožnilo robiť zmeny v programe, či popise materiálu nezávisle (pri rýchlosti napredovania počítačovej grafiky je táto nezávislosť vítaná).