Colour and Reflectance in Image Synthesis

by

Thomas Pflaum

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 1996

Abstract

In recent years substantial research has been done on physically based image synthesis, which includes the algorithms used to create synthetic images using models and algorithms based on the underlying physics of light. The work presented in this thesis focuses on two particular aspects of image synthesis that have not gained much attention in the past: how to represent colour and how to describe the reflection of light by surfaces.

A simulation system is presented that takes a model of the micro structure of a surface and creates a representation of the bidirectional reflectance function (BRDF) of a surface having that micro structure. In addition, a technique is presented that creates a colour space, used for rendering, that is adapted to the surface colours actually appearing in the scene.

Acknowledgments

The work presented in this thesis would not have been possible without the help, supervision and encouragement so kindly offered by many people. In particular, I would like to thank my supervisors Richard Bartels and William Cowan for their support and guidance. I would also like to thank my faculty readers Mike McCool and Bruce Simpson for there useful remarks that improved the quality of my thesis.

I am grateful for the support and friendship from the members of the the computer science department and especially the Computer Graphics Lab that let me have such a great time at Waterloo. In particular, I would like to Thank Rob Berks, Leith Chan, Leo Chan, Wilkin Chau, Matthew Davidchuk Ryan Gunther, Wolfgang Heidrich, Fabrice Jaubert, Rob Kroeger, Dan Milgram Andrew Park, Navid Sadikali and Julie Waterhouse.

Contents

List of Figures

Chapter 1

Introduction

In recent years photorealistic rendering has changed significantly. While the first approaches to image synthesis were limited by the available resources of computing power and memory, simple models were used to compute the interaction of light with surfaces. In addition, the computed light paths included only direct illumination from a light source. These simplifications limited the number of optical effects that could be generated. In particular the ray-tracing algorithm that is used by most commercially available rendering and animation packages limits the “global” lighting effects to shadows and perfect mirrors.

Another problem introduced by the simplified models is that they do not adequately simulate the physical behaviour of light. Some of the models used to describe how a surface reflects light, for example, do not follow the law of energy conservation, which means that “infinite” amounts of light could be generated by placing two surfaces close to each other.

The advantage of these simplifications is that they are easy to implement, and the time needed to generate an image is relatively short. But the lack of effects like soft shadows, caustics, and indirect illumination has lead to the introduction of a more sophisticated, physically based model that describe the interaction of light with surfaces.

The most widely used example was introduced by Kajiya in 1986 [26], who adopted an already known model from heat transfer theory [5]. It is an integral equation, called *the rendering equation* that describes the light leaving from a point on a surface as an integral of the light arriving at this location scaled by some function. This function is responsible for the appearance of the surface, and is usually called the *bidirectional reflectance distribution function* (BRDF). It will be described in more detail in the following sections of this thesis.

After the introduction of a mathematical formula describing the propagation of light between surfaces, many algorithms have been developed to solve this, or simplified versions of the integral equation. One of the algorithms trying to solve a simplified version is *radiosity* [7, 8], which assumes that all surfaces are perfectly diffuse reflectors. This allows the calculation of a solution for the interaction of all surfaces with each other by solving a system of linear equations.

Another approach adopts a well known mathematical method used to solve integral equations called Monte-Carlo integration [42]. Monte-Carlo integration computes samples of the domain at random locations, and uses these samples to derive an estimate of the solution. The more samples are taken the closer the estimate is to the actual solution. Unfortunately a large number of random samples are needed to get a good and noise free solution. On the other hand, the advantage of this algorithm is that it provides a complete solution of the rendering equation.

All these algorithms are usually referred to as *global illumination* algorithms, because they try to provide a global solution to the light transportation problem. They take the interaction of every surface with every other surface into account.

While all these algorithms solve the rendering equation (or a simplified version of it) to some approximation, most of the current implementations suffer from various problems. The first one is that the light is treated as simple RGB triples, thus omitting effects that arise from the fact that light is actually a continuous spectrum. Chapter 6 describes an alternative, efficient approach to this problem. Colour is represented in a way that depends on the colours actually appearing in the scene.

Another deficiency of common rendering systems is the lack of good models to describe the way light is reflected by surfaces, the lack of good BRDFs. While there are multiple models available, it is hard to come up with the correct parameters to match the reflectances of “real world” materials. This thesis addresses this problem by showing how to create and represent a BRDF from a geometric model of the surface micro structure.

This thesis is structured in the following way:

- The remainder of this chapter gives a more in-depth description of the rendering process and the techniques that are applied to it. It also describes how a BRDF may be obtained by creating a model of the structure of a surface and using a Monte-Carlo simulation to get an approximation of the reflectance properties.

- Chapter 2 first describes a BRDF from a physical point of view and then presents an overview of BRDF models that have previously been used in computer graphics, from a very simple heuristic model to complex models that are using simulation techniques to get the BRDF.
- Chapter 3 show how an efficient the Monte-Carlo simulation technique is implemented using importance sampling and stratification techniques. It also provides a discussion of the BRDF in radiometric terms to derive the correct scaling coefficients.
- Chapter 4 describes how the results of the simulation are represented in order to have a well filtered and reasonably fast data structure for use during the rendering process. It also shows some alternative representations that have been used in computer graphics.
- Chapter 5 shows some results of the BRDF-sampling and filtering process. For testing analytical models are taken and recreated using the simulation described in this thesis.
- Chapter 6 presents a way to represent spectral power distributions in a way that depends on the objects appearing in the scene. In order to be able to experiment with the effects of spectral rendering a viewer for spectral images is presented that allows changes to be made to the spectral power distribution of light source in an intuitive and interactive way.
- Chapter 7 concludes and shows future work.
- The Appendix presents the details of the implementation and the mathematical techniques used.

1.1 The Rendering Equation

In recent years the term “physically based rendering” has become more and more important in computer graphics. It means that instead of using “hacks” (“If it looks good enough, it is good enough”, [1]) a physically based model is used that is responsible for a particular effect in the real world. This section provides some motivation for such a model, the *rendering equation* [26].

The rendering equation describes the intensity of outgoing light [1] at a point on the surface, as the sum of the self emission of the surface (if it is a light source) plus the incoming light weighted by the reflectance properties of the surface:

$$L(\mathbf{s},\vec{\omega}) = L^e(\mathbf{s},\vec{\omega}) + \int_\Omega f(s,\vec{\omega}' \rightarrow \vec{\omega}) L(\mathbf{s},\vec{\omega}') \cos\theta' d\vec{\omega}' \tag{1.1}$$

$L(\mathbf{s},\vec{\omega})$ is the light leaving the surface in direction $\vec{\omega}$ from position $\mathbf{s}$. In the case of first generation rays $\vec{\omega}$ is the direction towards the camera, through a specific pixel of the final image. This outgoing light is the sum of the self emission of the surface $L^e(\mathbf{s},\vec{\omega})$ at position $\mathbf{s}$ in direction $\vec{\omega}$ plus the incoming light weighted by some function f. Light can arrive at location $\mathbf{s}$ from all directions Ω of the hemisphere above $\mathbf{s}$ (if the surface is opaque). "Above" $\mathbf{s}$ means that the normal of the surface goes through the pole of the hemisphere. Depending on the direction $\vec{\omega}'$ the incoming light has intensity $L(\mathbf{s},\vec{\omega}')$. For each incoming light direction $\vec{\omega}' \in \Omega$ the intensity is multiplied by a factor $f(s,\vec{\omega}' \rightarrow \vec{\omega})$ [2], which depends on the incoming and outgoing direction as well as the position on the surface. This function $f()$ is called the *bidirectional reflectance distribution function* (BRDF in short). The light is also multiplied by the cosine [3] of the angle between the incoming light direction and the surface normal θ'. Note that the integral is actually a double integral over the hemisphere.

To provide a more intuitive feeling of what is going on, we will present a short description on how this integral equation is solved using Monte-Carlo integration (Monte-Carlo tracing [42]).

Figure 1.1 shows how (naive) Monte-Carlo tracing works. We are interested in the light leaving a surface at position $\mathbf{s}$ in direction $\vec{\omega}$. Assuming that the surface is not a light source, the light leaving depends only on the incoming light. To get an estimate of the incoming light, N random rays are shot in directions $\vec{\omega}_i{}'$, sampling the hemisphere. Some of these rays hit other surfaces at positions s_i. Assuming that we know the intensity at s_i we know the intensity of the incoming light from direction $\vec{\omega_i}'$. This intensity is then multiplied by $f(s,\vec{\omega}' \rightarrow \vec{\omega}) \cos\theta'$. All these terms are added together and divided by the number of samples shot, assuming that the hemisphere was sampled uniformly. This normalized sum is used as an estimate for $L(\mathbf{s},\vec{\omega})$.

[1]By this we mean the outgoing *radiance*. A precise definition of units will be provided in Section 3.6. Until then we will talk about "intensity" which is proportional to the amount of energy, or the brightness.

[2]The $\rightarrow$ is used as a notation indicating that light is coming in from direction $\vec{\omega}'$ and leaving in direction $\vec{\omega}$

[3]This cosine term is due to the units being used. See Section 3.6.

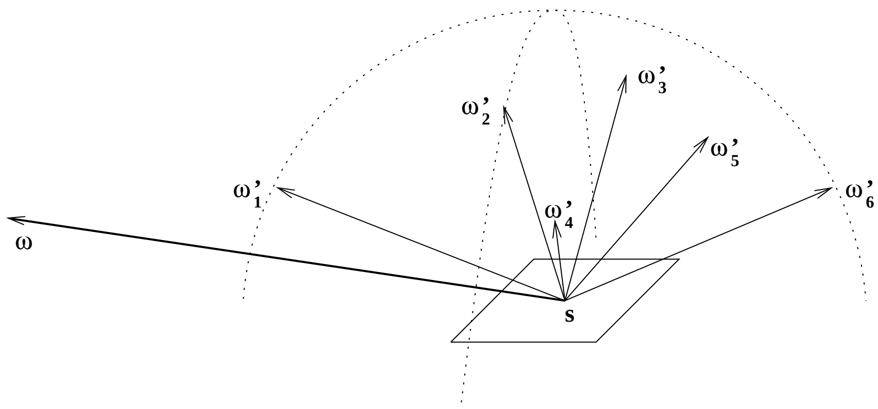


Figure 1.1: Monte-Carlo tracing.

$$\bar{L}(\mathbf{s},\vec{\omega}) = \sum_{i=0}^{N} \frac{f(s,\vec{\omega}' \to \vec{\omega})\bar{L}(\mathbf{s},\vec{\omega}')\cos\theta'}{N}$$

To calculate the light leaving s_i in direction $\vec{\omega_i}'$ the algorithm is applied recursively. (Care has to be taken that this does not result in an infinite loop). Many papers are available on how to optimize this process [26, 42, 43, 46].

1.2 Colour

The previous section showed how the propagation of light between surfaces is calculated. This section will show the basic concepts behind colour, and will show how colour can be represented.

Before talking about how to represent colour, we first have to define what we mean by colour. The perceived colour is determined largely by the *spectral power distribution* (SPD) of light entering the eye. The left part of Figure 1.2 diagram shows the light emitted by a fluorescent light source. The x-axis is the frequency ranging from 380nm to 780nm, which is the part of the electro-magnetic spectrum visible to humans. The y-axis shows the power emitted by the light source at a particular frequency. This is the first interpretation of colour: The sensation obtained from a particular SPD by the human eye.

But there is another interpretation of colour. The colour is not determinated as light we see,

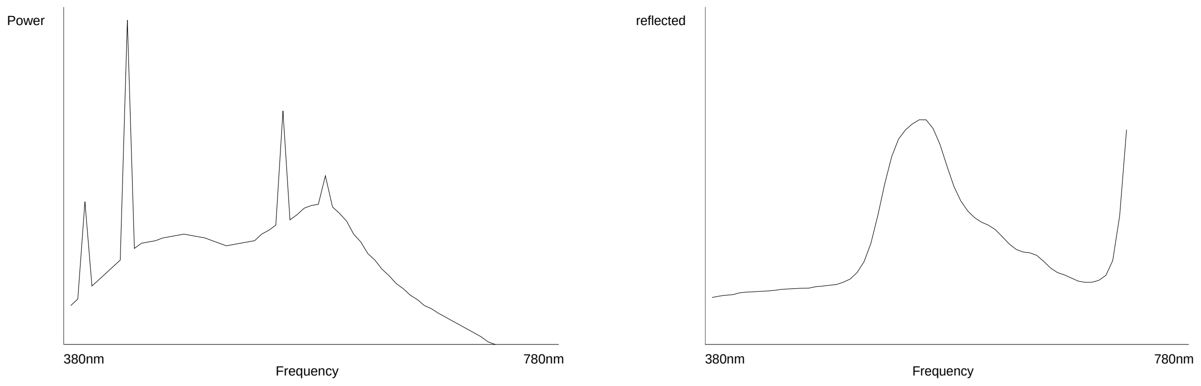


Figure 1.2: The left graph shows the spectral power distribution of a fluorescent light source, the right graph is the reflectance of pine needles.

but as a surface property. The colour of an object is defined by the way it reflects light. In this case colour is not defined by a SPD, but by the percentage of light that gets reflected at a specific wavelength. The right part of Figure 1.2 shows reflectance of a pine needle.

Both interpretations occur widely in human perception. While in the first definition the colour of an object depends on the lighting conditions (the colour of the light source), it defines colour as what we actually see. The second one treats colour as a material property, independent of any illumination.

An extreme example would be to have a green table in a totally dark room. The first interpretation would say the table is black (that is what we see if we would actually look at the table), the second interpretation would say the table is green because the actual illumination is not considered at all.

To calculate the SPD reflected by a surface with reflectance function $r(\lambda)$ when illuminated by a light source with SPD $i(\lambda)$ the product of two functions has to be computed.

$$o(\lambda) \;=\; r(\lambda)i(\lambda) \tag{1.2}$$

Since the power of the reflected light must be less than or equal to the incoming light, the integral of the reflectance function r must be less than or equal to 1, and $r(\lambda)$ must be ≥ 0 for all

λ [4].

$$\int_0^\infty r(\lambda)d\lambda \;\leq\; 1 \tag{1.3}$$

In the case of SPDs as well as in the case of reflectance functions, colour is defined as a continuous function of the frequency. To represent colour finitely approximations are necessary. One simple form is to sample the function uniformely in wavelength. The most accurate data that is generally available represents SPDs and reflectance properties as 401 samples at $1nm$ increments from $380nm$ to $780nm$. If, in the general case, n samples are used to represent a colour, both SPDs and reflectance values are n-dimensional vectors. To calculate the SPD $\vec{o}$ reflected by a surface with reflectance $\vec{r}$ illuminated by a SPD $\vec{i}$, a component by component multiplication of the two vectors $\vec{r}$ and $\vec{i}$ has to be performed. ($\circ$ is used to represent this component by component multiplication operator.)

$$\vec{o} \;=\; \vec{r} \circ \vec{i} = \begin{pmatrix} r_1 \cdot i_1 \\ r_2 \cdot i_2 \\ \cdots \\ r_n \cdot i_n \end{pmatrix} \tag{1.4}$$

These n-dimensional vectors could be used to represent colour in a rendering system. But the memory and computation time requirements would be rather high (n-multiplications to compute the reflected colour). Other more compact forms are needed. Fortunately, we can take advantage of the fact that the human visual system that has only three different types of colour receptors. This allows the representation of colour using only 3 coefficients. The RGB model is an example of such a colour representation. But this model has some deficiencies, which are explained in Chapter 6. An alternative is presented in Chapter 6 that takes advantage of the SPDs and reflectances that actually appear in the scene description.

[4]If $\int o(\lambda)d\lambda < \int r(\lambda)i(\lambda)d\lambda$ actually implies $\int r(\lambda)d\lambda < 1$ requires specification of the used measure. This thesis will not provide any details on these measures.

1.3 Frequency Dependent Rendering Equation

The rendering equation as stated in Section 1.1 does not deal with colour at all. In order to handle colour, the equation has to be extended to handle frequency dependent intensities. [5]

The rendering equation, as defined in Equation 1.1, computes the outgoing intensity in terms of the incoming light over the hemisphere above the surface. In order to handle frequency dependent intensity, the equation is extended to define the outgoing intensity as a function of direction and the wavelength:

$$L(\mathbf{s},\vec{\omega},\lambda) = L^e(\mathbf{s},\vec{\omega},\lambda) + \int_{\lambda_{\min}}^{\lambda_{\max}} \int_{\Omega} f(s,\vec{\omega}' \to \vec{\omega},\lambda' \to \lambda) L(\mathbf{s},\vec{\omega}',\lambda') \cos\theta' d\vec{\omega}' d\lambda' \qquad (1.5)$$

Since the intensity at wavelength λ_{out} may depend on incoming light at different wavelengths (due to effects of fluoresces and phosphorescence) [16] an additional integral over the wavelength is needed. Note that it is not sufficient to integrate only over the visible spectrum, since there are materials that reflect light from the ultra-violet part of the spectrum ($\lambda > 780nm$) into the visible spectrum (detergent whiteners and "black light sources"). The BRDF $f()$ needs two additional parameters, as well. The incoming and the outgoing wavelength are needed to express the fraction of light at wavelength λ' that is reflected as light of wavelength λ.

This complicates the rendering equation significantly. But for most materials the effect, that light get reflected at a different wavelength, is not very important and can be omitted without any visible difference. (Fluorescence and phosphorescent effects are exceptions.) In this case $f(s,\vec{\omega}' \to \vec{\omega},\lambda' \to \lambda)$ is $\neq 0$ only if $\lambda = \lambda'$. Thus different wavelengths can be computed independently, since there is no crosstalk between them. (The matrix that approximates the function is diagonal) As a consequence, the integral over the wavelength is not needed either, and the rendering equation can be rewritten in a simpler form:

$$L_\lambda(\mathbf{s},\vec{\omega}) = L^e_\lambda(\mathbf{s},\vec{\omega}) + \int_{\Omega} f_\lambda(s,\vec{\omega}' \to \vec{\omega}) L_\lambda(\mathbf{s},\vec{\omega}') \cos\theta' d\vec{\omega}' \qquad (1.6)$$

At each wavelength, the equation can be solved independently from the solutions of all the other equations. If an SPD is represented using n samples the equation can be rewritten in vector notation:

[5] Again the vague term "intensity" is used. A more precise description is presented in 3.6.

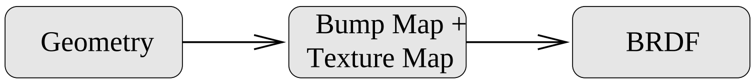


Figure 1.3: Levels.

$$\vec{L}(\mathbf{s},\vec{\omega}) = \vec{L}^e(\mathbf{s},\vec{\omega}) + \int_\Omega \vec{f}(s,\vec{\omega}' \to \vec{\omega}) \circ \vec{L}(\mathbf{s},\vec{\omega}') \cos\theta' d\vec{\omega}' \tag{1.7}$$

The range of $\vec{L}(\mathbf{s},\vec{\omega})$ and $\vec{f}(s,\vec{\omega}' \to \vec{\omega})$ is $\Re^n$ and the $\circ$ operator the component by component multiplication operator.

One common application of this equation is to use only three samples ($n = 3$) and use the RGB colour model. Shortcomings of this model are shown in Section 6.

1.4 The BRDF in the Rendering Process

The rendering equation provides a way to solve the global illumination problem by solving an integral equation. The solution, the way the final image looks, depends on the surfaces in the scene (the geometry), the way light is distributed by the light sources $L^e_\lambda(\mathbf{s},\vec{\omega})$ and the way light is reflected by surfaces, the BRDF $f_\lambda(s,\vec{\omega}' \to \vec{\omega})$. [6] In particular the BRDF is responsible for the colour and the "geometrical attributes" of appearance [23] such as gloss, haze, reflectiveness, shininess and diffuseness.

Figure 1.3 shows the different levels of detail at which images for photorealistic rendering are usually described. The highest level is the geometric representation of objects. This is the description that is closest to the actual physical model. Geometry is typically represented by specifying the boundary surfaces of objects. Effects that can be obtained from this representation include parallax shifts if the camera is moved and depth impression. But if very fine detail, such as a carpet with lots of different coloured pieces, should be represented geometric modeling becomes infeasible. The number of required polygons is very high, and good anti-aliasing is needed to get the desired effects.

[6] In the remainder of this thesis the vector representation of the rendering equation will be used. There has been some work done on frequency crosstalk in [16, 19], but the additional complexity increases rendering time so much, that it is not usable yet.

Thus, for fine detail, texture mapping and bump mapping are usually used. Texture mapping can be expressed by a BRDF that changes the colour of the reflectance as a function of the position on a surface. Bump mapping can also be expressed by a BRDF that changes the angular distribution of the reflected light. At the next level the frequency of colour changes is so high that it is not possible to distinguish different areas of a surface by its colour. In this case the reflected light, and thus the BRDF, does not depend on the position on the surface any more. It only specifies the way light gets reflected depending on the angle and colour of the incoming light and the direction from which the surface is viewed.

To choose between the different representations is difficult, and the choice changes as the position of the camera is moved. A a piece of woven cloth viewed from 5cm should be represented as a geometric model to look real, to be able to see the individual threads. If the camera move further away, a bump map might be indistinguishable from a geometric representation and is preferable because it requires less storage and computing time to render. If the camera moves even further away the positional change of the reflectance on the surface gets less and less important. If the camera is far enough away the reflectance properties can be represented by a function that is independent of the position on the surface. These three different levels are called *object scale*, *milliscale* and *microscale* [58].

While the object scale description can be obtained relatively easily by measuring real world objects, and milliscale maps can be obtained by taking photos of surfaces, it is hard to get the reflectance properties at the microscale level. The available physically based models used to represent the reflection properties take a large number of parameters that are not intuitive to the user. That means that it is hard to pick parameters that result in the desired reflectance properties. In addition these models have limitations that might not allow the representation of a specific material exactly. The other alternative, accurately measuring real world, objects is rather complicated; it requires expensive equipment and takes a long time.

Thus, the approach taken in this document to obtain the reflectance properties of materials is based on modeling the micro structure of the surface. That means that the surface details that are too small to be visible individually, such as the surface of brushed aluminum, are represented as a geometric model. This model is then used in a simulation that illuminates the model from various positions and computes the reflected light in various directions. The data gained in this way is stored, and used as a representation of the reflectance properties of a position-independent BRDF $f_\lambda(\vec{\omega}' \to \vec{\omega})$. This BRDF can then be applied to an object-scale surface, which should have reflection properties similar to those of the material being represented by the microscale model.

The simulation is implemented using Monte-Carlo ray-tracing to get an estimate of the BRDF. Since the domain of a position independent BRDF is still four dimensional (two angles for the incoming and two for the outgoing direction), a large number of samples as well as 4 dimensional filtering is needed to get a good representation of the BRDF. The following chapters outline the details of this approach.

Chapter 2

The Bidirectional Reflectance Distribution Function

The previous chapter presented a brief overview of the rendering process and the role of the BRDF in it. This chapter describes the BRDF and its properties in more detail. The first section gives a brief overview of the physics of light reflecting from a surface. Since the complexity of the physical world can never be completely simulated, the second section shows approximations that have been used in the past. Even though most of these models try to approximate the underlying physics, they are limited to a small variety of surface reflectance types, and even for those it is hard to chose the suitable model parameters. To solve this problem, we propose constructing a model of the surface micro structure. We then use this model in a simulation system that generates a BRDF from the model. Of interest is the extent to which complex BRDFs can be produced from simplified models of micro structure. We show through the example of grass that a simple micro-structure can produce a good approximation to a realistic BRDF (see Chapter 5).

2.1 Reflection from a Physical Perspective

Figure 2.1 shows a simple model of the interaction of light with an object. Light hitting the surface from a particular direction can either be reflected at the surface or it can be transmitted into the body of the object. When light is reflected at the surface (called *surface* or *specular* reflection) of a nonmetallic object, the colour of the light is almost unchanged. On the other hand, when light is

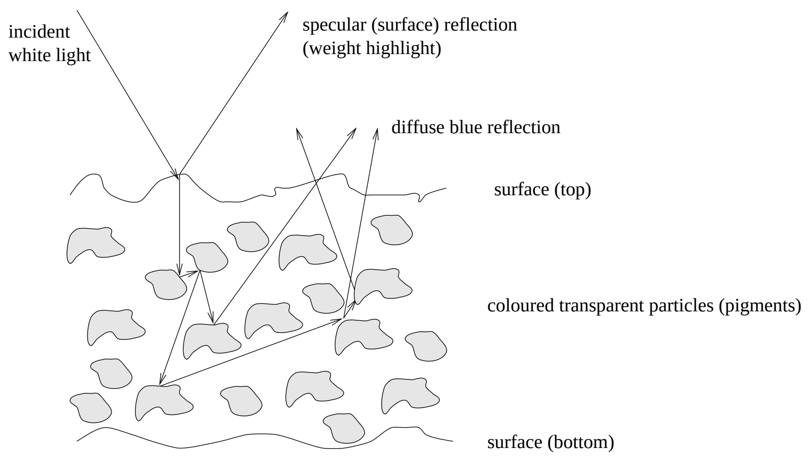


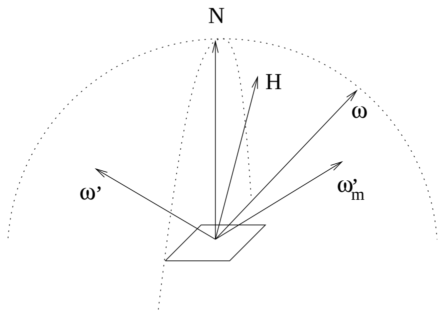
Figure 2.1: Specular and diffuse reflection.

reflected by a metallic surface, the specularly reflected light colour changes.

Most of the light striking rough, nonmetallic objects is refracted into the body of the object. The term refraction is used because the direction of the incident light beam is changed, when light is changing from one medium (usually air) to another (the body of the object), which have different indices of refraction.

The body typically contains a large number of pigmented particles. Light is scatter by these particles, with or without interacting with the colour centres in the particles. Colour centres interact differently with different wavelength so that colour of the light is changed. Because there are a large number of pigmented particles within the body, the light scatters many times. Consequently all dependence on the original direction of the light is lost. In other words, the amount of light leaving in a particular direction is independent of the direction of the incoming light. This type of reflection is usually referred to as *body* or *diffuse* reflection.

The amount of light reflected at the surface depends on the dielectric constant of the medium and the smoothness of the surface. The smoother the surface is the more light is reflected. In addition a rough surface causes diffusion of the incoming light. Thus, there are two ways diffuse reflection can be created, either by diffusion within the body, as just described, or by reflection at a rough surface. Reflection causes diffusion from rough surfaces by a simple stochastic effect. Photons hit the surface at positions of random surface orientation and bounce of, accordingly, in



N	surface normal
ω'	unit vector pointing towards the light source (θ, ϕ)
ω	unit vector indicating the outgoing direction (θ', ϕ')
ω'_M	unit vector indicating the mirror direction
s	position on the surface
H	halfway vector

Figure 2.2: Definition of angles and direction.

random directions. These two diffuse parts can be of different colour, since the light reflected at the surface does not interact with the pigment particles.

While this is a rather simple model of the way light gets reflected, it is still too complicated to derive an analytical solution. Thus even simpler models have been used in image synthesis (see Section 2.3.3).

2.2 Radiometric Derivation

The introduction of this thesis described the rendering equation using the intuitive term "intensity", which is roughly proportional to the brightness of the light or its energy. To create a BRDF from a microscale surface model, the physical units have to be defined properly. Otherwise the BRDF can be in consistent with physical laws, especially the energy conservation law.

The BRDF$f(\mathbf{s}, \vec{\omega}' \to \vec{\omega})$ is a 6 function of six variables, the position on the surface $\mathbf{s} \in \Re^2$, the outgoing direction $\vec{\omega} \in ([0, \frac{1}{2}\pi[\times[0, \frac{1}{2}\pi[)$ and incoming direction $\vec{\omega}'$. In the following discussion $\vec{\omega}$ will be used to represent a direction, as well as the location of a point on the unit hemisphere, with the normal of the surface pointing towards the pole of the hemisphere. In any case elevations are measured starting from the pole (see Appendix A).

As described in the introduction, this thesis deals only with BRDFs that are independent of the surface position. This will always be assumed in the remainder of this thesis, unless it is stated otherwise. Another assumption which is made in the first part of this chapter is that the range of the BRDFs are the real numbers $\Re$. This does not allow us to represent colour, but simplifies

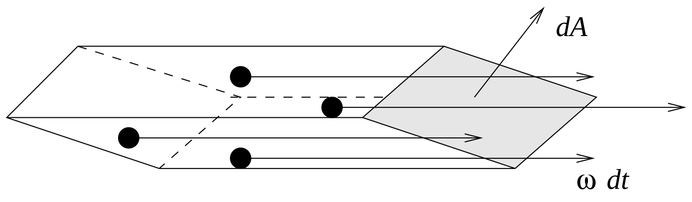


Figure 2.3: Particles flowing through a surface in time *dt*.

the description. The extension to support colour is rather straightforward and presented in Section 2.2.3.

2.2.1 Flux and Radiance

In order to be able to talk about the BRDF in physical terms, some basic physical concepts have to be presented. The relevant field in physics is called *linear transport theory*. This deals with the motion of particles (photons in our case) through space, and tries to quantify the collective motion with terms like *flux* or *particle density*.

We define the particle density $p(\mathbf{x}, \vec{\omega})$ as the number of particle per unit volume $1/m^3$ traveling in direction $\vec{\omega}$. We also define the differential density, the number of particles per differential volume $P(\mathbf{x}, \vec{\omega}) = p(\mathbf{x}, \vec{\omega})dV$.

In order to be able to deal with a flow of particles in space, we examine the number of particle flowing through a differential surface area dA in direction $\vec{\omega}$, see Figure 2.3. Note that dA is actually a vector that points in the direction of the surface normal of the surface element A.

To find how many particles are flowing through dA we first have to note that the speed of the photons is always c, thus all particles travel $c \cdot dt$ in a differential time interval. Thus all particles moving in direction $\vec{\omega}$ within the volume shown in Figure 2.3 pass through the surface dA. The size of this volume depends not only on the size of the surface but also on the angle formed by the surface normal dA and the direction $\vec{\omega}$. Thus, the number of particles flowing trough dA is the size of the volume multiplied by the particle density $p(\mathbf{x}, \vec{\omega})$.

$$P(\mathbf{x}, \vec{\omega}) = p(\mathbf{x}, V\omega) cdt \cos\theta dA \tag{2.1}$$

Thus, $P(\mathbf{x}, \vec{\omega})$ is proportional to the time interval and the size of the surface area. θ is the angle between the flow direction $\vec{\omega}$ and the surface normal of dA. The quantity $p(\mathbf{x}, \vec{\omega})$ is called flux. It is common to rewrite this equation in terms of a differential solid angle $d\vec{\omega}$. $d\vec{\omega}$ is a vector pointing in the direction of the flow and its length is equal to the small differential solid angle of directions about $\vec{\omega}$. In this case, the length of the vector is $c\,dt$. The unit of a solid angle is called a steradian $[sr]$.

$$P(\mathbf{x}, \vec{\omega}) = p(\mathbf{x}, \vec{\omega}) \cos\theta d\vec{\omega} dA \tag{2.2}$$

Thus, the unit of the particle density $p(\mathbf{x}, \vec{\omega})$ is $\left[\frac{1}{m^2\ sr}\right]$.

So far we have only been concerned with particles. In rendering it is more usual to deal with energies. The energy of each photon depends only on its wavelength and can be computed using Plank's equation $E = hc/\lambda[W]$. This can be used to define the *radiance*. The radiance is the radiant energy per unit volume and is equal to the photon volume density in direction $\vec{\omega}$ times the energy of a single photon.

$$L(\mathbf{x}, \vec{\omega}) = \int p(\mathbf{x}, \vec{\omega}, \lambda) \frac{hc}{\lambda} d\lambda \tag{2.3}$$

Thus, the radiance is the integral over all wavelengths λ. If we want to represent colour, which means that different amounts of energy are at different wavelength we have to use spectral radiance which is defined as radiance per frequency.

$$L_s(\mathbf{x}, \vec{\omega}, \lambda) = p(\mathbf{x}, \vec{\omega}, \lambda) \frac{hc}{\lambda} = \frac{dL(\mathbf{x}, \vec{\omega})}{d\lambda} \tag{2.4}$$

Radiance at a surface position $\mathbf{x}$ from direction $\vec{\omega}$ is the power per unit projected area perpendicular to $\vec{\omega}$ (see Figure 2.4). Radiance is probably the most important quantity in rendering, as well as in the BRDF sampling approach presented in this thesis. This is a consequence of two important properties of the radiance.

- The radiance along the propagation direction of a ray does not change, which is a consequence of the energy conservation law and solid angle normalization [8]. This is a useful property in ray-tracing algorithms, since a particular radiance can be associated with a ray, and the radiance is the same at the start and end point of the ray.

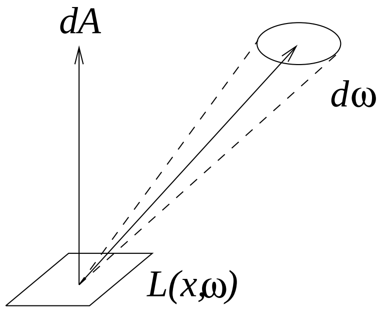


Figure 2.4: Definition of the radiance.

- The response of a sensor is proportional to the radiance impinging the surface visible by the sensor [8]. This make radiance the unit of choice to represent pixel values of an image, or the amount of light measured by a photometer when BRDF sampling is performed (Section 3.6).

Since radiance is defined per solid angle, it can not be used to describe the energy emitted by a point light source, which would have a solid angle of 0. Instead irradiance $E(\mathbf{x})$ has to be used.

$$E(\mathbf{x}) = \frac{\Phi}{4\pi} \frac{\cos\theta}{|\mathbf{x} - \mathbf{x}_s|} \tag{2.5}$$

Φ is the flux, the total energy emitted by the light source. The total flux is divided by the surface area of a sphere located at the position of the light source $\mathbf{x}_s$ passing through x. θ is the angle between the surface normal and the direction towards the light source.

The radiance is defined as the change in unprojected irradiance.

$$L(\mathbf{x}, \vec{\omega}) = \frac{dE(\mathbf{x})}{\cos\theta d\vec{\omega}} \tag{2.6}$$

Conversely, the irradiance is the integral over the unprojected radiance.

$$E(\mathbf{x}) = \int_\Omega L(\mathbf{x}, \vec{\omega}) \cos\theta d\vec{\omega} \tag{2.7}$$

2.2.2 BRDF Definition

The BRDF was introduced as part of the rendering equation. For non-emissive surfaces with a BRDF independent of the surface position (independent of $\mathbf{s}$), the rendering equation can be simplified:

$$L(\mathbf{s},\vec{\omega}) = \int_{\Omega} f(\vec{\omega}' \to \vec{\omega}) L(\mathbf{s},\vec{\omega}') \cos\theta' d\vec{\omega}' \tag{2.8}$$

This equation can be solved for $f(\vec{\omega}' \to \vec{\omega})$ which provides a definition for the BRDF:

$$f(\vec{\omega}' \to \vec{\omega}) = \frac{dL(\mathbf{s},\vec{\omega})}{L(\mathbf{s},\vec{\omega}') \cos\theta' d\vec{\omega}'} \quad \left[\frac{1}{sr}\right] \tag{2.9}$$

This also defines the unit of the BRDF to be sr^{-1}. In other words, the BRDF is the ratio of the incident light from a differential solid angle in direction $\vec{\omega}'$, to the light leaving in direction $\vec{\omega}$. The range of the BRDF is $[0,\infty)$. The reason why it is unbounded can be seen when representing a perfect mirror.

For a perfect mirror the light leaving in direction $\vec{\omega}_L$ depends only on light arriving from the mirror direction $\vec{\omega}'_M$ (according to the surface normal, see Figure 2.2). In addition the amount of reflected radiance is identical to the incoming radiance. This results in a simplified version of the rendering equation for perfect mirrors:

$$L(\mathbf{s},\vec{\omega_L}) = L(\mathbf{s},\vec{\omega}'_M) \tag{2.10}$$

In order to represent this reflectance behaviour a δ distribution has to be used as the BRDF.

$$f(\vec{\omega},\vec{\omega}') = \frac{\delta(\vec{\omega}' - \vec{\omega}'_M)}{\cos\theta'} \tag{2.11}$$

$\delta(\vec{\omega}' - \vec{\omega}'_M) = 0$ if $\vec{\omega}'$ is not equal to the reflection of the outgoing direction $\vec{\omega}'_M$ about the surface normal. The integral over $\int_{-\infty}^{\infty} \delta(\vec{\omega}' - \vec{\omega}'_M) = 1$ if $\vec{\omega}' - \vec{\omega}'_M$. Since $\int_{-\infty}^{\infty} \delta(x-y) f(x) dx$ equals $f(y)$ substituting the BRDF 2.11 into the rendering equation 2.8 results in equation 2.10.

The necessity of infinite values in the BRDF is responsible for the *distribution* part in the name *bidirectional reflectance distribution function.* There is an alternative to the BRDF which is called *biconical reflectance.* In contrast to the BRDF, biconical reflectance values have the advantage of being within $[0,1]$, but have the disadvantage that they depend on the distribution of the incoming light. Another advantage of the BRDF is that it is a part of the rendering equation, which makes it easy to use during the rendering process.

Some properties must be obeyed by any BRDF. The first is known as the *Helmholtz reciprocity principle.* It says that a BRDF is symmetric in terms of the incoming and outgoing direction.

$$f(\vec{\omega}' \rightarrow \vec{\omega}) = f(\vec{\omega} \rightarrow \vec{\omega}') \tag{2.12}$$

In other words, a photon traveling along a path from point A to point B would have the same energy if it were to travel the reverse path from B to A.

The second property is even more basic: energy conservation has to be maintained. That means that the amount of reflected light is less than or equal to the energy of the incoming light. This requires that the integral over the incoming hemisphere be less than or equal to 1 for all outgoing directions.

$$\int_\Omega f(\vec{\omega}, \vec{\omega}') d\vec{\omega}' \ \leq \ 1 \quad \forall \vec{\omega} \tag{2.13}$$

Note that this equation has to be true, but it is not sufficient to ensure an energy-conserving BRDF. In order to be energy conserving the total amount of reflected light has to be less than the total incoming light.

A BRDF is called *isotropic* if its value does not depend on the actual azimuth values of the incoming and outgoing direction, but only on their difference.

$$f_{\text{iso}}((\theta_i, \phi_i + \phi) \rightarrow (\theta_i, \phi_r + \phi)) = f_{\text{iso}}((\theta_i, \phi_i) \rightarrow (\theta_i, \phi_r)) \quad \text{for all } \phi \tag{2.14}$$

An isotropic BRDF can be written as a 3 dimensional function, because it only depends on azimuth difference.

$$f_{\text{iso}}{}'(\theta_i, \Delta\phi, \theta_r) = f_{\text{iso}}((\theta_i, \phi) \rightarrow (\theta_i, \phi + \Delta\phi)) \quad \text{for all } \phi \tag{2.15}$$

In other words, the reflected light at a surface position is unchanged if the surface is rotated around its normal. This is true for a large number of surfaces, but materials like brushed metal or cloth change their appearance when rotated. This is due to the directional micro-structure of the surface, like the brush scratches or the way threads are structured respectively.

2.2.3 BRDF and Colour

In the previous the section, the BRDF was defined in terms of Radiance L. To include colour, we have to define the BRDF in terms of the spectral radiance $L(\mathbf{s}, \vec{\omega}, \lambda)$.

$$f(\vec{\omega}' \rightarrow \vec{\omega}, \lambda' \rightarrow \lambda) = \frac{dL(\mathbf{s}, \vec{\omega}, \lambda)}{L(\mathbf{s}, \vec{\omega}', \lambda') \cos\theta' d\vec{\omega}'} \quad \left[\frac{1}{sr}\right] \tag{2.16}$$

The general spectral BRDF depends on the incoming as well as the outgoing wavelength. For a non fluorescent or phosphorescent surface, photons of a particular wavelength are not reflected as a photon of a different wavelength. This simplification allows us to define the spectral BRDF as a function of one wavelength:

$$f(\vec{\omega}', \rightarrow \vec{\omega}, \lambda) = \frac{dL(\mathbf{s}, \vec{\omega}, \lambda)}{L(\mathbf{s}, \vec{\omega}', \lambda) \cos\theta' d\vec{\omega}'} \quad \left[\frac{1}{sr}\right] \tag{2.17}$$

As described in Section 1.3, rendering systems usually use an n-dimensional vector to represent colour. Thus, a specular BRDF can be represented as a function with range $\Re^n$, $f_\lambda(\vec{\omega}' \rightarrow \vec{\omega})$. For discussions regarding the representation of a colour as an n-dimensional vector efficiently see Chapter 6.

2.3 Overview of BRDF Models

This section describes previous analytical approaches that have been used in the past. These techniques range from empirical approximations to sophisticated physical models that take anisotropic reflections into account.

2.3.1 Lambertian Surface

One of the simplest ways light can be reflected by an object is perfect diffuse reflection. This means that the reflected light is independent of the incoming light direction. While this model is very restrictive, it is the only model that can be used in combination with standard radiosity algorithms ([7, 45]).

$$L(\vec{\omega}) = k_d \sum_{i=1}^{n} L(\vec{\omega}_i{}')(\vec{\omega}_i{}' \cdot N)$$

Thus, the reflected light $L(\vec{\omega})$ is the sum of the incoming light from all (point) light sources $L(\vec{\omega}_i')$ in the scene, multiplied by the cosine of the angle between the surface normal N and the direction of the incoming light $\vec{\omega}_i'$ (see Figure 2.2).

A lambertian reflectance is represented by a constant BRDF.

$$f_{\text{lambertian}}(\vec{\omega}' \to \vec{\omega}) = k_d \tag{2.18}$$

2.3.2 Phong

One of the first empirical approaches to describe specular reflectance of surfaces is the *Phong*-model [34]. The model is fairly straightforward: The light reflected by a surface is either due to perfect lambertian body reflection or due to a specular reflection. For calculating the specular component a simple model is used. The cosine of the angle between the viewing direction and the mirror direction ($\vec{\omega} \cdot \vec{\omega}'_M$, see Figure 2.2) is raised to a power (usually between 2 and 100) which is then multiplied by the incoming light $L(\vec{\omega}_i')$. To compensate for the shortcomings of simple global illumination a constant ambient term is often added to the model. This term is used as a rough approximation of the indirect illumination.

$$L(\vec{\omega}) = k_a L_a + k_d \sum_{i=1}^{n} L(\vec{\omega}_i')(\vec{\omega}_i' \cdot N) + k_s \sum_{i=1}^{n} L(\vec{\omega}_i')(\vec{\omega}'_M \cdot \vec{\omega})^{k_e}$$

The closer the viewing direction is to the mirror direction, the brighter is the highlight. The size of the highlight is defined by the power k_e of the cosine. The constants k_a, k_d and k_s define the ratios between the ambient, diffuse and specular components.

To state Phong's model in terms of a BRDF is possible if the ambient term is omitted. Since the BRDF is always multiplied by the incoming light it is not possible to encode an additive constant into the BRDF.

$$f_{\text{phong}}(\vec{\omega}' \to \vec{\omega}) = k_d + k_s \frac{\vec{\omega}' \cdot \vec{\omega}'_M}{cos\theta'} \tag{2.19}$$

The Phong Model is purely heuristic. Even the law of energy conservation is not obeyed, it is possible that more light is reflected than is received. The reason for this is the cosine term in the denominator of Equation 2.19. For glancing angles (θ close to 90 degrees), the value of f gets arbitrarily large. If the outgoing direction is close to 90 degrees the numerator is close to one when the incoming direction is close to the same big angle. But in this case the denominator is very small, resulting in a big value for the BRDF. Thus, more light gets reflected than arrives.

This is not a problem in most rendering systems because most of them are ray-tracing based, which means that light sources are treated as points. In this case, the glancing angle problem is visible only when the viewpoint, the surface and the light source are almost in the same plane.

But despite its physically incorrect behaviour, Phong is still the most commonly used reflectance model today. This is mainly due to the following reasons:

1. Its computational simplicity. The cosine can be calculated by forming a scalar product of the incoming direction and the reflection vector. This makes it possible to build hardware implementation, which can be found in almost all 3D graphics workstations today [30].

2. The intuitive parameters. As we can see in the models described later in this section, the more advanced shading models have too many parameters. Many of these parameters have no intuitive meaning, making it hard to choose them well.

   In contrast the parameters of the Phong model are easy to understand. We have 3 coefficients k_a, k_d and k_s that define the ratio of ambient, to diffuse, to specular reflectance, and a parameter k_e that specifies the size of the specular highlight which can also be interpreted as the shininess of the surface.

But the Phong model is rather limited. Certain types of material can not be described properly. It works well for simulating plastic but later models will show that better results can be achieved using more sophistication.

2.3.3 Micro Facet Models

In order to compensate for some of the shortcomings of the Phong model, people like Blinn [4], Cook and Torrance [11] developed more complicated models based on an estimate of the surface micro structure. From such models analytical approximations are derived to be used in the BRDF.

One of these models assumes the surface to consist of a large number of small irregularly oriented V-shaped valleys (see Figure 2.5). The sides of these valleys are the *microfacets*, with normals N. The microfacets are assumed to be small enough that they are not visible individually. In order to

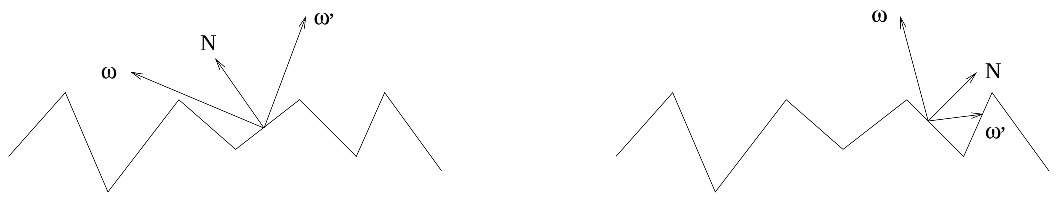


Figure 2.5: Torrance-Sparrow microfacet model.

compute the light reflected by a surface like this, it is first necessary to know how the microfacets are distributed. The right part of Figure 2.5 shows that it is also necessary to take care of the shadowing effects introduced by other facets. In addition the way a microfacet reflects light has to be specified.

Thus, microfacet models divide the BRDF into three parts [38]:

- $D(\vec{\omega}', \vec{\omega})$ The slope distribution function defines the distribution of the V-shaped microfacets. It defines the fraction of the facets that are oriented in the direction of the halfway-vector.
- $G(\vec{\omega}', \vec{\omega})$ is the geometric attenuation factor which expresses the ratio of light that is not self-obstructed by other microfacets of the surface.
- $F_\lambda(\vec{\omega})$ is the microfacet reflectance, which specifies the way light is reflected by the microfacets.

In all these models it is assumed that the microfacets are perfect mirrors; thus, the reflected light can be calculated from Fresnel's formula. Fresnel formula computes the reflected SPD as a function of the wavelength dependent index of refraction η_λ and the extinction coefficient κ_λ, for non conducting material $\kappa_\lambda = 0$. If the polarization of the light is neglected, the Fresnel function can be expressed as a real valued function.

$$F_{\eta_\lambda,\kappa_\lambda}(\vec{\omega}) = \frac{(g-c)^2}{2(g+c)^2}\left(1 + \frac{(c(g+c)-1)^2}{(c(g-c)+1)^2}\right) \tag{2.20}$$

with $c = \vec{\omega} \cdot \vec{\omega}_H$ ($\vec{\omega}_H$ is the halfway vector $\vec{\omega}_H = \vec{\omega} + \frac{\vec{\omega}'}{|\vec{\omega}+\vec{\omega}'|}$) and $g^2 = \eta_\lambda^2 + c^2 - 1$. In a conducting material the complex index of refraction $\bar{\eta}_\lambda = \eta_\lambda - \kappa_\lambda$ has to be used. The complete complex formula can be found in [17]. Fresnels formula can be derived from Maxwell's equations, as presented in [29].

The complete reflectance model, as presented by Cook and Torrance [10], combines the terms into a reflectance model. The original work includes a purely diffuse as well as an ambient part; that is not presented here. Adding this is straightforward and identical to way it is added to the specular part of the Phong model 2.3.2.

$$L(\vec{\omega}) = \sum_{i=1}^{n} L(\vec{\omega}_i{}')(N \cdot \vec{\omega}_i{}') \frac{1}{\pi} \frac{F_\lambda(\vec{\omega}) D(\vec{\omega}_i{}', \vec{\omega}) G(\vec{\omega}_i{}', \vec{\omega})}{(N \cdot \vec{\omega}_i{}')(N \cdot \vec{\omega})} \tag{2.21}$$

Again, this model can be rewritten as a BRDF according to Definition 2.9.

$$f_{\text{cook}}(\vec{\omega}', \vec{\omega}) = \frac{F_\lambda(\vec{\omega}) D(\vec{\omega}', \vec{\omega}) G(\vec{\omega}', \vec{\omega})}{(N \cdot \vec{\omega})} \tag{2.22}$$

The distribution D can be chosen to match the surface micro-structure. But the problem is that the geometric attenuation factor G has to be chosen according to the distribution D. Blinn [4] used a simple Gaussian distribution and provided an analytical solution for G. An alternative distribution was used by Beckmann [3] and introduced by Cook et. al. [11]. The Beckmann model takes frequency dependence into account. The disadvantage of this model is that it is not possible to come up with a good attenuation function. However, even more complicated microfacet distributions can be used to introduce anisotropic effects.

The advantage of this model compared to the Phong model is that a larger class of surfaces can be represented. Unfortunately to use this model in directly the micro structure of the surface has to be known, and a distribution function that matches this micro structure has to be found. In addition the geometric attenuation function has to be computed. All this is rather complicated and not very intuitive.

Schlick [38, 39] tried to solve this problem by fixing the distribution to an approximation of the Gaussian distribution. He also used an approximation of the Fresnel function and the geometric attenuation function. In addition he tried to add “intuitive” parameters to make it easy to use. This, of course, results in a more limited model that restricts the possible reflectance properties that can be represented by the reflectance model.

2.4 Measuring the BRDF

Another way to obtain a BRDF is to perform measurements on real world surfaces. The classical approach to do this is using a *goniophotometer*. A goniophotometer consists of a calibrated light-source that can be moved around a surface sample. The sample itself can be rotated around its center. Finally there is a measuring device that is located at a fixed position. To obtain the BRDF the light source is moved and the surface sample is rotated. This results in a number of data points that are then used to approximate the BRDF.

Unfortunately goniophotometer has some disadvantages. The first one is that goniophotometers are rather expensive. As a consequence there are not very many of these devices in existence. The second disadvantage is that it takes a long time to measure a BRDF. That makes the measurement of a BRDF even more expensive. In addition it is necessary that a sample of the surface is available that is small enough to be measured by the goniophotometer. The measurement of a grass BRDF, as described in Section 5.3, would not be possible with most goniophotometers, since the size of the specimen would have to be in the range of 1 square meter. Finally a goniophotometer measures only a small fraction of the light so accuracy is a problem.

Alternative approaches that use a CCD video camera in a known configuration are available. Ward et. al. [54] described a system that used a semi transparent hemispherical mirror and a CCD camera placed at the center of the hemisphere. The advantage of this approach is that it is much faster then measuring the BRDF with a goniophotometer, but it shares the disadvantage that the specimen has to be small. Additionally the accuracy of a CCD based approach is lower than the data obtained from a goniophotometer. A similar approach was taken by Karner et. al. [27].

The inherent problem of all direct measurement approaches is that the obtained data is final. It is not possible to change any of the parameters of the measured surface. This additional flexibility can be obtained using the simulation method described in Chapter 3.

2.5 Appearance Measurement in Other Areas

The way light is reflected by a surface is not only of interest in the field of computer graphics. Especially in the paint industry the reflectance properties of a particular paint or surface finish are important. While the BRDF is rarely used, other techniques which more directly correlate with the quality of the finish are extremely important.

The coating industry, for example in specifying object appearance needs a richer description than reflectance alone. When people talk about the way a surface looks they use such terms as “gloss”, “haze” and “luster”. Hunter [23] calls these attributes the *geometric properties* of appearance. Since these appearance properties are important in the description of what a specific coating looks like, methods have been developed to measure these quantities. To measure specular gloss for example, the light source is positioned at about a 45 degree angle and the measurement device is located along the reflection direction. Similar arrangements are defined for the other geometric properties (see [23]).

But these specific values can be derived easily if the complete BRDF of a surface is available. Thus, people in these areas are beginning to think about using complete BRDFs as a tool to describe the appearance properties of coatings.

Chapter 3

Sampling the BRDF

The previous chapter described the BRDF and models used in computer graphics to approximate the reflectance properties of real world objects. This chapter describes a simulation system that takes a model of the micro structure of a surface, simulates the interaction of light with the micro structure, and produces an approximation to the BRDF.

This chapter describes two different approaches for generating the BRDF from a model. The first uses a method similar to ray-tracing as used in traditional image synthesis. Rays are shot starting from the position of a virtual camera, or in this case the virtual spectro- radiometer. At the intersection point with a part of the micro structure the local illumination is computed.

The other approach uses inverse ray-tracing also called photon or light tracing. Rays are shot starting form the light source and are reflected one or multiple times until they leave the model. The position the photon leaves the model is recorded. This results in a distribution of photons, which is called the photon map [24, 25]. The photon map is then used to reconstruct the BRDF.

3.1 Micro Structure Model

All physically based models described in the previous section are difficult to use, because there is no simple and intuitive way to chose parameters. This makes it hard for a user to create desirable reflectance properties.

The micro structure model provides an alternative approach [58]. Instead of providing a fixed model with a number of parameters, the user specifies a geometric description of the micro structure of the model. Thus he uses standard modeling techniques, such as polygons, spline surfaces or CSG, to model the surface structure of a material.

The surface model is then used in a simulation that computes the BRDF of a surface with the given micro structure. In order to be able to represent the micro structure by a BRDF it is necessary that the light source as well as the observer be "far enough" away. Far enough means that the illumination does not change over the surface sample and that the observer cannot see the actual micro structure.

If, for example, the micro structure of cloth is modeled (the way the threads are woven), and a BRDF is created from this model, then it can be used if the actual threads are not visible. This is the case when the observer is more than 50cm away from the cloth (depending on the size of the threads).

One problem of the micro structure approach is choosing the proper size of the model. It has to be big enough to include all parts of the structure of the surface. But on the other hand, it has to be small in order to allow for a reasonably quick generation of the BRDF.

Another problem is choosing the reflectance properties of the micro geometry. At this level the problem is similar to the problem at the object level. Of course, a micro-micro structural approach could be used to generate the BRDF of the micro geometry. But this complicates the generation, and at the micro-micro level the same problem occurs. Thus, in this thesis a pragmatic approach has been taken. The Phong model is used to define the BRDF of the micro-geometry. This has been done because the details of the micro-BRDFs do not affect the final BRDF in a substantial way, as described in Section 5.3.2.

Figure 3.1 shows the basic concept used by the BRDF simulation system. In order to generate the BRDF from the micro geometry, the patch of micro geometry is placed in the center of a much larger hemisphere. For convenience we assume that the radius of the hemisphere is 1. Thus, for a point on the micro geometry, the direction of the incoming and outgoing light can are equivalently viewed as positions on the unit hemisphere.

The BRDF is a four dimensional function $f(\vec{\omega}' \rightarrow \vec{\omega})$. $\vec{\omega}'$ is the incoming light direction and $\vec{\omega}$ is the outgoing direction. Using the virtual hemisphere model, we can define the incoming direction as the position of a virtual point light source on the hemisphere. The outgoing direction is the position of the virtual radiometer.

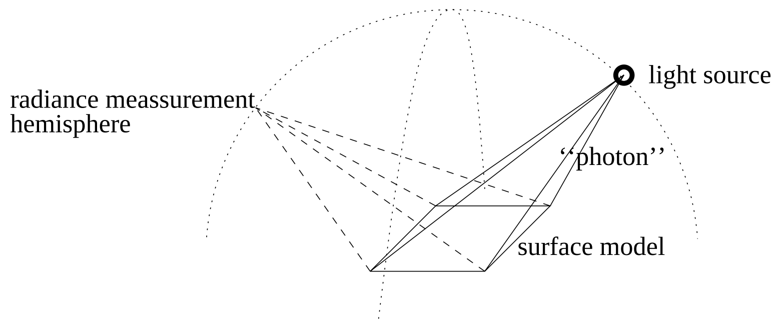


Figure 3.1: Creating the BRDF from a surface model.

For a fixed light source and radiometer position, the reflected light depends on the interaction of the light emitted by the light source with the micro-structural model. Thus, the interaction with the entire model has to be computed. This can be done by applying either ray-tracing (Section 3.2) or photon-tracing (Section 3.3).

To create an approximation to the BRDF, it is necessary to compute the light entering the virtual radiometer at various positions of the light source, as well as at various positions of the radiometer. After normalizing the brightness of the light source, the bidirectional reflectance can be computed for all radiometer and light source positions. Interpolation techniques can then be used to reconstruct the BRDF at positions close to the computed locations.

Since the BRDF is a four dimensional functions a large number of radiometer-light source pairs have to be computed to provide a good reconstruction of the BRDF. Thus it is useful to take advantage of symmetry (Helmholtz law see Equation 2.12) and any potential isotropy (see Equation 2.15). In the case of isotropic BRDFs the position of the light source has to be varied only over an arc instead of over the entire hemisphere. As we will show in chapter 5 of this thesis, this greatly simplifies the sampling and reconstruction process.

3.2 Eye Tracing

One way to obtain the BRDF is similar to the ray-tracing technique [15] of conventional rendering. Instead of computing the path a photon takes from the light source to the measuring device, the inverse path is computed.

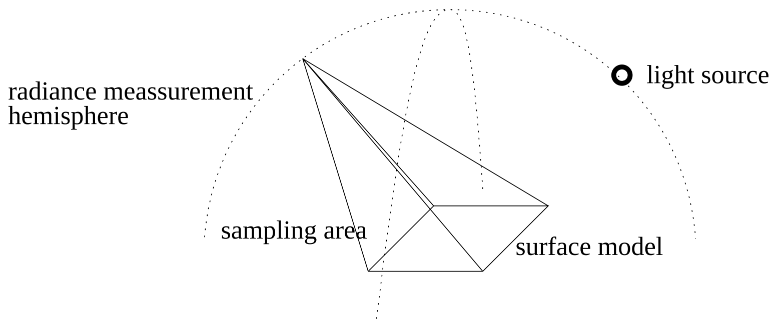


Figure 3.2: Sampling the BRDF by shooting rays from the virtual position of the measurement device.

From the position of the radiometer a ray is shot towards a random position on the micro model (see Figure 3.2). At the intersection point of the ray with the model the local illumination is computed. Local illumination means that only the direct illumination from the light source is used. The illumination due to multiple reflections of light is ignored.

For BRDF sampling it is assumed that the light source is far way with respect to the size of the surface model. That means that the illumination is constant on the surface patch. Since a point light source is used, the intensity of the illumination depends only on the cosine of the surface normal with the direction to the light source $N \cdot \vec{\omega}'$.

Shooting a single ray per pixel results in the light being reflected by one particular position on the micro facet model. In order to get an approximation of the whole model, multiple rays are shot distributed randomly over the model. An example of shooting 1, 2, 9 and 25 rays can be seen in Figure 3.3 (The example uses the grass model described in Chapter 5. All the images are filtered in the same way.)

A simple global illumination effect that is usually used in ray-tracers is shadow testing. This can be used in this setting, as well. At the intersection point of the ray with the model a *shadow ray* is traced towards the light source. If the ray intersects a surface, the current position is in shadow.

To get an approximation of the entire BRDF, for every light source position $\vec{\omega}'$ on the hemisphere, the reflected light into every direction $\vec{\omega}$ has to be computed. This can be done by fixing the light source position, and varying the outgoing direction. In order to do this the hemisphere of

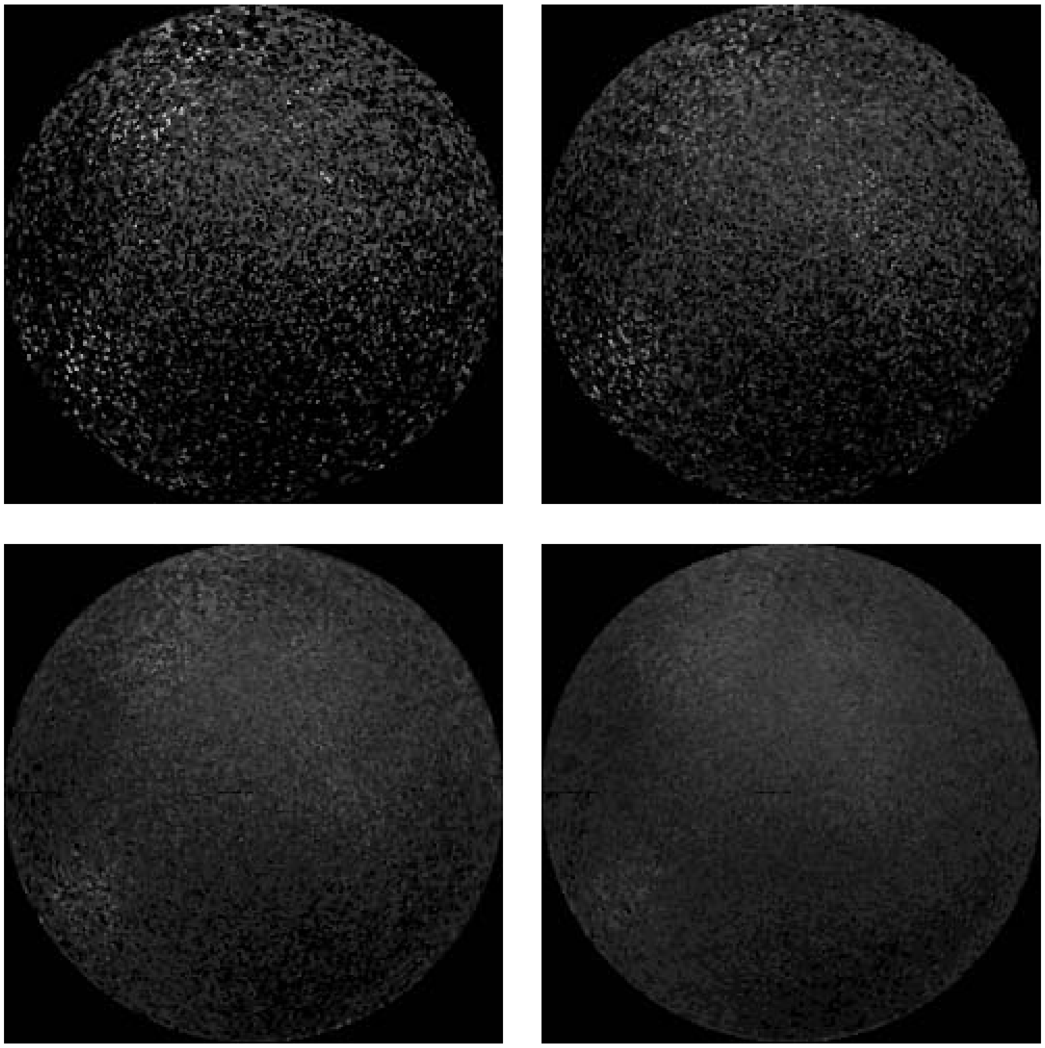


Figure 3.3: Comparison of 1, 2, 9 and 25 samples per pixel.

outgoing direction is subdivided uniformly. Uniformly means that for every zenith θ, the radius of the sphere is calculated ($r = \sin\theta$) and the same number of samples per unit distance are traced. This allows one to map from a regular Cartesian grid to points on the hemisphere.

$$\begin{array}{rcll} \theta & = & \frac{1}{2}\pi y & 0 \le y \le 1 \\ \phi & = & 2\pi \frac{x}{\sin\theta} & 0 \le x < \sin\theta \end{array} \tag{3.1}$$

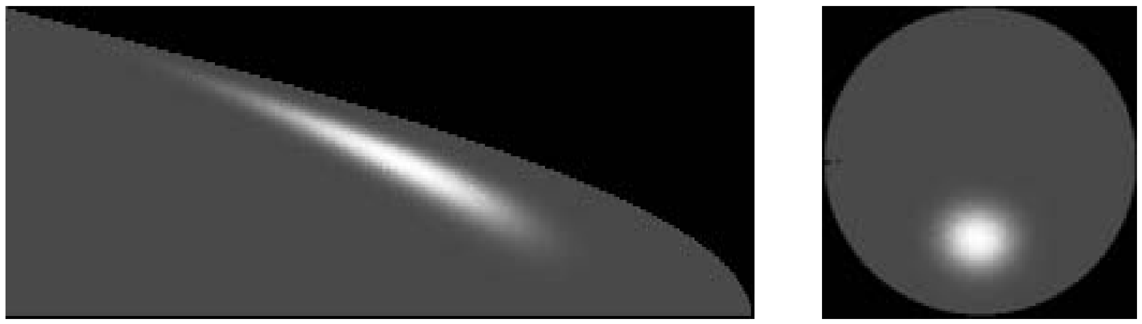


Figure 3.4: Mapping to a polar plot.

The domain of the mapping can be seen in the left part of Figure 3.4. The larger θ gets, the further away it is from the north pole, the bigger is the radius of the hemisphere and the more samples are taken. This allows a slice of the BRDF to be saved as a tiff image [14] with run-length encoding used to compress the black parts of the image.

In order to be able to visualize the BRDF for one light source position the BRDF can be projected into an angular plot. See for example, the right image in Figure 3.4.

So far the BRDF is computed for a single light source position. To obtain the entire 4D BRDF, a 2D array of these images has to be created. This requires a very large number of samples, which makes the process slow. In the case of isotropic BRDFs only a 1D array of images has to be created. But even in this case the time needed to render the whole BRDF takes a rather long time.

The major disadvantage of this approach results from the ray-tracing paradigm used to compute the illumination. No effects due to multiple scattering are handled by the simulation. This problem could be solved by using methods similar to those used by the path-tracing algorithm [26]. Path-tracing shoots a secondary ray from the intersection point into a random direction. At the position the secondary ray intersects with the model, the local illumination is computed again. The illumination at this position also effects the illumination at the first intersection point. This process can be repeated and thus multiple scattering of light have to be considered. The disadvantage of this approach is that it converges very slowly. Images look noisy unless a very large number of paths are computed for every primary ray intersection.

Another disadvantage of this approach is that neighboring radiometer and light source positions are treated independently. But actual BRDFs are rather smooth. Thus, it might be useful to propagate some information from the neighbours to the current pixel.

The technique described in the following section will take advantage of this coherence. It can simulate indirect illumination and will provide a simple way to filter the gathered data appropriately.

3.3 Photon Tracing

The technique described in Section 3.2 shoots rays from the position of the virtual photometer. Instead of doing this, we can shoot rays from the light source. This is closer to what is happening in the real world, where the light can be viewed as photons emitted by a light source. Shooting rays from the light source is simulating the path a photon takes.

3.3.1 Simulating Photon Flow

Shooting rays from the light source seems to be a straightforward thing to do. The reason this approach is not used in rendering (except for some special applications) is that most of the photons emitted by light sources do not hit the camera. Only a very small portion of the photons choose a path from the light source to the film of the camera. This makes it infeasible for rendering, since a very large number of photon paths would have to be created and only very few of them would actually account for the image.

When simulating bidirectional reflectance the setting is different. There is no camera in the traditional sense, but rather there is a photometer that measures the intensity at various positions on the hemisphere. Thus, the "film" that has to be hit by photons, in order to be useful, is the entire hemisphere surrounding the model. This makes photon tracing likely to be a good technique.

Furthermore, we only have to consider the photons leaving the light source in the direction of the model. Photons leaving in other directions can not affect the reflected light, since they would never hit the any surfaces at all. Thus, every photon emitted by the light source hits the model at least once. Depending on the micro BRDF, some energy of this photon is absorbed and the remainder is scattered into a random direction. A secondary ray is traced into the scattering direction. This ray might hit another surface of the model, in which case the same process is repeated. Eventually a direction is picked that does not intersect with any surface. In this case

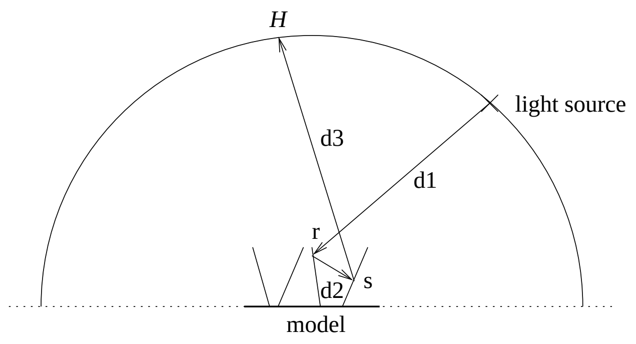


Figure 3.5: The path of a photon.

the ray intersects the hemisphere surrounding the micro model at a position $H = (\theta_H, \phi_H)$. This corresponds to one possible path a photon can take from the light source. The photon would be detected at a photometer located at H. Unless a photon gets stuck in a sequence of infinite reflection in the model (which is very unlikely, probability zero for realistic micro structure models), every photon eventually leave the model and thus intersect the sampling sphere.

Some photons might intersect the lower half of the hemisphere. This is a result of the micro structure model being too small. In the simulation system these photons are simply ignored. An alternative approach would use a constant diffuse term for the whole BRDF and add the photons intersecting the lower hemisphere to the diffuse term.

Figure 3.5 shows an example. The photon is leaving the light source in direction d_1 which is chosen randomly, but it is guaranteed that will intersect with the model. Thus, a ray is shot from the light source position in direction d_1. In the example the ray intersects the model at position r. At r it is scattered in a randomly picked direction d_2. The ray intersects again at position s, and is scattered in direction d_3. A third ray is created with origin s and direction d_3. This time, the model is not intersected. Instead the ray intersects the hemisphere at position H. At this location the energy carried by the photon is recorded.

3.3.2 Computing the Scattering Ray

Once a ray intersects with a surface S of the model at a point p, a scattering ray has to be computed. If the reflectance of the micro surface is entirely diffuse (its micro BRDF is constant), the photon is scattered in any direction with the same probability. Thus a random direction on the hemisphere around the surface normal has to be computed. Since every position on the hemisphere corresponds uniquely to a direction, a random position is picked. The distribution of the random position has to be constant. This means that the probability for every position on the hemisphere is the same.

A distribution like this can be created very easily. Assume the random numbers ξ_1 and ξ_2 $(0 \leq \xi_i \leq 1)$ are uniformly distributed, then uniformly distributed positions on the hemisphere can be computed using the following formulas:

$$\begin{aligned} \theta &= \arccos \xi_1 \\ \phi &= 2\pi\xi_2 \end{aligned} \tag{3.2}$$

3.4 From one Photon to a BRDF

So far we have only discussed how to compute the path one photon might take. To get a BRDF, multiple photons have to be computed. Every photon describes *one* possible path that a photon can take from the light source position $\vec{\omega}'$ to the photometer position $\vec{\omega}$. But of course there are an infinite number of paths a photon could take from $\vec{\omega}'$ to $\vec{\omega}$. Furthermore, for a full approximation of the BRDF the reflectance has to be computed for every light source and every photometer position.

3.4.1 Splatting

In the real world a photon can be viewed as a very narrow impulse (delta function) with a relatively small energy. Since there are a very large number of photons, we still perceive the light reflected by a surface to be continuous.

In a simulation the number of photon paths that can be computed is limited. As a consequence both the spatial width and the energy of a photon has to be higher than in the real world. If a photon with energy E hits the hemisphere at a location H, it is assumed that, with high probability, there might be a similar path, with similar probability, that places a photon close to the current photon. Taking this further leads to the idea that each photon hitting the hemisphere actually represents many photon, distributed in a normal distribution around H.

Figure 3.6 shows an example. Every ray represents a number of photons that are distributed according to a normal distribution. The further the position is away from the actual intersection

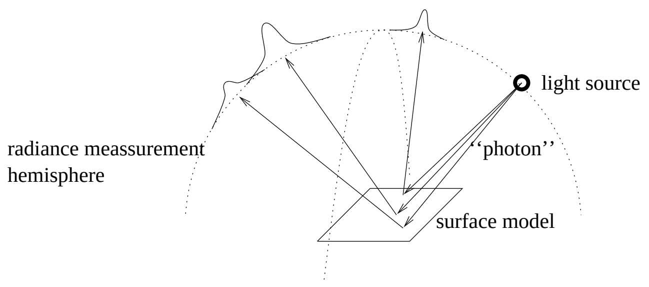


Figure 3.6: Each ray represents a whole number of photons distributed according to a normal distribution.

point, the fewer can be assumed and the less influence a photon has on the BRDF reconstruction at this location. This property is described by a distribution function $p_h(d)$, where d is the distance from position H.

Thus, to reconstruct the final BRDF at a position $P = (\vec{\omega}', \vec{\omega})$, the energy E_i of every photon, weighted by the function $p_i(d)$ is summed up:

$$f(\vec{\omega}' \rightarrow \vec{\omega}) = c \sum_{i=1}^{n} p_i(d) E_i \quad \text{where } d \text{ is the distance from photon } i \tag{3.3}$$

n is the total number of photons computed for the simulation, and c is a normalization constant. The value of this constant is derived in Section 3.6 and is necessary to ensure energy conservation. It has to be noted that different photons might have different energy distribution functions $p_i(d)$.

3.4.2 Distance

The problem is now how to define the distance from a photon. Assume we have a photon with origin $\vec{\omega}'$ and photometer position $\vec{\omega}$ ($P = (\vec{\omega}', \vec{\omega})$). If we want to know its influence on another position $P_2 = (\vec{\omega}_2', \vec{\omega}_2)$ we have to know the distance between P and P_2. This requires us to define a distance measure in the 4D space that is composed of the incoming light direction and the outgoing light direction.

A metric within this 4D space has to be defined. Of course there are various ways to define the distance. One simple way to define a distance measure is to compute the distance of the two points on the incoming sphere, and add it to the distance of the two points on the outgoing sphere. (For the definition of the distance of two points on the sphere see Appendix A.)

$$d_{L_1} = d_{\text{Sphere}}(\vec{\omega}', \vec{\omega}_2') + d_{\text{Sphere}}(\vec{\omega}, \vec{\omega}_2) \tag{3.4}$$

This defines some form of L_1 norm between the distance of the incoming directions, and the distances of the outgoing directions. In a similar way an L_2 and an L_∞ norm can be defined:

$$d_{L_2} = \sqrt{d_{\text{Sphere}}(\vec{\omega}', \vec{\omega}_2')^2 + d_{\text{Sphere}}(\vec{\omega}, \vec{\omega}_2)^2} \tag{3.5}$$

$$d_{L_\infty} = \max(d_{\text{Sphere}}(\vec{\omega}', \vec{\omega}_2'), d_{\text{Sphere}}(\vec{\omega}, \vec{\omega}_2)) \tag{3.6}$$

While the normal distribution seems to be a reasonable heuristic, other functions might be associated with a photon. A simple box function might be used for computational efficiency. In Chapter 4 a polynomial approximation to the Gauss curve is used.

The remainder of this chapter deals with techniques to compute only the "important" photon paths, thus omitting the ones that transport very little energy. This is driven by the idea that the paths with the highest energy transport are the most visible.

3.5 Monte Carlo Techniques

The technique described in the previous section can be viewed in the context of Monte Carlo integration. The term "Monte Carlo Techniques" is used for a number of techniques that use random samples of a function to generate an approximation of its integral or some other functional.

While Monte Carlo integration has been known in mathematics and numerical analysis for a long time, and has been used in other fields such as the approximation of heat transfer [22], it has been introduced to computer graphics fairly recently. It was used by Kajiya in [26] to solve the global illumination problem stated in terms of the "rendering equation". Since then Monte Carlo tracing has become more and more important for finding solutions to the global illumination problem. This is especially obvious in the work of Shirley [42, 41, 43, 44], who tries to improve the convergence of the Monte Carlo estimation using sophisticated importance sampling techniques, and in the work of Ward, who uses additional caching techniques [57, 53, 56] in his *Radiance* [55] system to lower the noise level in low frequency areas of the image. Such work increased the interest of the rendering community in the Monte Carlo approach, mainly because the images created by this technique contain the most sophisticated lighting effects seen in global illumination so far. In addition Monte Carlo integration provides a theoretically complete solution to the rendering-equation.

Since the simulation of bidirectional reflectance requires one to solve a subset of the rendering equation, Monte Carlo methods are used in this thesis.

3.5.1 Monte Carlo Integration

This section provides a short introduction to the theory of Monte Carlo integration. Monte Carlo integration provides a numerical approximation of a integral by sampling the integrand at random positions.

Suppose we wish to approximate the value of a definite integral:

$$I = \int_B f(x)dx \tag{3.7}$$

If we have a uniformly distributed random variable X we can compute an expectation of I. If N $x_1, \ldots x_N$ of X trials are taken, we get the following estimate:

$$I \approx \tilde{I} = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{3.8}$$

Monte Carlo techniques are especially useful when integrating over a high dimensional domain. If ρ is the number of dimension, then Monte Carlo integration converges with the rate $\rho/\sqrt{N}$. For traditional integration techniques the number of samples varies exponentially with the number of dimensions (scales with α^ρ for some α).

The convergence speed can be increased significantly by using *importance sampling* (see Section 3.7). Importance sampling takes advantage of properties that are known about the integrand. These properties are used to change the distribution of the random variable X. Assume that the density function of X is $p(x)$, then an approximation of I can be computed as follows:

$$I \approx \tilde{I} = \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \tag{3.9}$$

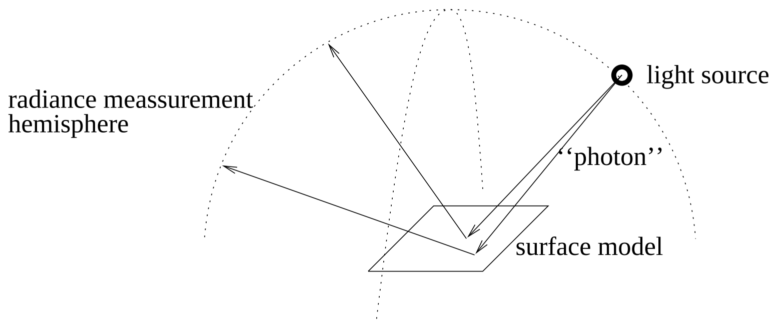


Figure 3.7: Sampling the BRDF by shooting rays from the light source.

The division by the density function is necessary to take care of the different number of samples taken at different parts of the sampling domain. If the density function is large around x, then a large number of samples of $f(x)$ are taken in the area around x. To compensate for this, the value of the function is divided by the density.

A good density function $p(x)$ for a function $f(x)$ is one that minimize the variance for a given number of samples. The variance is defined as

$$\sigma_p^2 = \int_B \frac{f^2(x)}{p(x)} dx - I^2 \tag{3.10}$$

Thus the optimal density $p(x)$ is

$$p_{\mathrm{opt}}(x) = \frac{|f(x)|}{\int_B |f(x)| dx} \tag{3.11}$$

Finding $p_{\mathrm{opt}}(x)$ is as hard as finding I. But the variance of the estimate I is smaller when $p(x)$ has a behaviour similar to $|f(x)|$ (see Equation 3.10). For more information see [60].

3.6 BRDF Sampling in Radiometric Terms

As described in Section 2.2.2 the BRDF is the ratio of the outgoing radiance to the incoming irradiance. For a point light source with unit intensity (flux), which is used in the simulation system described in this thesis, the BRDF is defined as:

$$f(\vec{\omega}' \to \vec{\omega}) = \frac{L(\mathbf{s}, \vec{\omega})4\pi}{\cos\theta'} \quad \left[\frac{1}{sr}\right] \tag{3.12}$$

θ' is the zenith of the light source. This equation also assumes that the light source is at unit distance. Since the model is small with respect to the size of the hemisphere, this can be assumed without introducing too much error.

The computation of $L(\mathbf{s}, \vec{\omega})$ involves solving the rendering equation on the micro model:

$$L(\mathbf{s}, \vec{\omega}) = \int_{\Omega} f_m(\vec{\omega}'' \to \vec{\omega}) L(\mathbf{s}, \vec{\omega}'') cos\theta'' d\vec{\omega}'' \tag{3.13}$$

The photon tracing technique described in Section 3.3 solves this integral by using path tracing, with paths starting from the light source. In other words, the integration is over all the paths a photon can take from the light source to the position $\vec{\omega}$ on the hemisphere. This provides an approximation for $f(\vec{\omega}' \to \vec{\omega})$ if $\vec{\omega}'$ and $\vec{\omega}$ are fixed.

With the splatting function $p(d)$ described in Section 3.4.1 we have to define a function $L_p(\mathbf{t}, \vec{\omega}' \to \vec{\omega})$ that describes the radiance in direction $\vec{\omega}$ with the light source located at $\vec{\omega}'$, and the emitted photon hitting the micro model at position $\mathbf{t}$.

Thus, we can express the total radiance at a position $\vec{\omega}$ on the hemisphere with the light source positioned at $\vec{\omega}'$.

$$f(\vec{\omega}' \to \vec{\omega}) = C\frac{4\pi}{\cos\theta'} \int_{\Omega}\int_{\Omega}\int_{A} p(d) L_p(\mathbf{t}, \vec{\omega}''' \to \vec{\omega}'') dt \cos\theta''' d\vec{\omega}''' d\vec{\omega}'' \tag{3.14}$$

d is the distance of $(\vec{\omega}'', \vec{\omega}''')$ from $(\vec{\omega}, \vec{\omega}')$ using one of the norms defined in Section 3.4.2. C is a constant which normalizes the BRDF properly, i.e. so that it fulfills the energy conservation property. This constant depends on the norm that is used, and on the distribution function $p(d)$ (see Section 4.2.3).

This multi dimensional integral integrates over the incoming direction $\vec{\omega}'''$, the outgoing direction $\vec{\omega}''$, the position of the first photon bounce $\mathbf{t}$ and all the paths from $\vec{\omega}'''$ to $\vec{\omega}''$ via $\mathbf{t}$ $L_p(\mathbf{t}, \vec{\omega}''' \to \vec{\omega}'')$.

As informally described in Section 3.3 this integral is solved by picking a random incoming direction, and a random location of the first bounce. The inter-reflections within the model are solved using path tracing. The outgoing direction is obtained as a result of the scattering within the model.

This sampling does not allow us to specify the outgoing direction of a photon. But as a consequence of the Helmholtz law (see Equation 2.12), which allows us to exchange the role of the incoming and outgoing direction, the incoming direction can also be defined as a function of the outgoing direction.

One more note on Equation 3.14. In the real world, it can be assumed that there is an almost infinite number of photons, and the distribution $p(d)$ is close to a delta function. In the simulation, a distribution with wider support is used. But, the more photons that are traced in the simulation, the smaller the support of the distributions can be chosen. In the limit, the distributions eventually become delta impulses, thus the technique described in this thesis converges to the actual solution of Equation 3.14.

3.7 Importance Sampling

Monte Carlo methods tend to converge rather slowly. One way to improve this is to use importance sampling, introduced in Section 3.5.1. Monte Carlo sampling takes *a priori* knowledge of the integrand into account.

We have little knowledge of the integrand. The micro structure model is defined by surfaces, thus getting analytical information is almost impossible (this is why we run a simulation in the first place). The only thing we know is the micro BRDF we have chosen for the surfaces in the model.

We can take advantage of this by adjusting the distribution of scattered rays. An improvement that has been used in Monte Carlo image synthesis is to scatter the rays according to the BRDF of the surface. i.e. more rays are shot in direction at which the BRDF is high. If we choose a Phong shading model as the micro BRDF, the scattering is highest in the mirror direction.

$$f_{\text{phong}}(\vec{\omega}' \to \vec{\omega}) = k_d + k_s \frac{\vec{\omega}' \cdot \vec{\omega}'_M}{cos\theta'} \tag{3.15}$$

Thus we choose the distribution of the random variable that is used to generate the scattered ray, according to the micro BRDF. That means that, more rays are generated close to the mirror direction $\vec{\omega}'_M$. Since the Phong BRDF is the cosine of the angle between the outgoing direction and the mirror direction taken to some power, the optimal distribution to sample a Phong BRDF is to use this cosine density.

If we assume that θ is the angle between the mirror direction and the outgoing direction, random numbers should be distributed according to the following density function.

$$p(\theta, \phi) = \frac{n+1}{2\pi} \cos^n \theta \tag{3.16}$$

A pair of uniform random numbers r_1 and r_2 in $[0, 1]$ can be transformed to be distributed according to 3.16 by the following transformations.

$$(\theta, \phi) = (\arccos((1 - r_1)^{\frac{1}{n+1}}), 2\pi r_2) \tag{3.17}$$

Using importance sampling reduces the variance significantly, because more paths that transport high energy are computed.

Once importance sampling is used, different photon paths have different probabilities, thus the probability has to be stored with every photon. This is necessary, because photons with a high probability have a lower weight in the final reconstruction of the BRDF than photons with a low probability (see Equation 3.9). The probability can also be used to modify the splat function $p(d)$.

Chapter 4

Representing the BRDF

One of the major problems when dealing with BRDFs is storing the four dimensional function in a compact way that is also easy to evaluate. This section first gives a short discussion of previous work that has been done to represent BRDFs, namely the approach taken by Westin et. al. [58] using spherical harmonics. The advantages and drawbacks of this approach are discussed. Then an alternative approach to represent the BRDF is presented. This approach uses the "splats" introduced in the previous chapter to represent the BRDF. It also shows how a spatial subdivision can be used to improve the performance of this representation.

4.1 Previous Work—Spherical Harmonics

Sillion et. al. [46] and Westin et. al. [58] use *spherical harmonics* to represent BRDFs. Spherical harmonics are naturally defined on a spherical domain. They are, in some sense, the equivalent of the Fourier basis functions on a spherical domain. Thus, the basis function have global support, and the coefficients of the spherical harmonic transform of a function provides some information about the frequency characteristics of the function.

$$Y_{l,m}(\theta,\phi) = \begin{cases} N_{l,m}P_{l,m}(\cos\theta)\cos(m\phi) & \text{if } m > 0 \\ N_{l,0}P_{l,0}(\cos\theta)/\sqrt{2} & \text{if } m = 0 \\ N_{l,m}P_{l,|m|}(\cos\theta)\sin(|m|\phi) & \text{if } m < 0 \end{cases} \tag{4.1}$$

The normalization constants $N_{l,m}$ are

$$N_{l,m} = \sqrt{\frac{2l+1}{2\pi}\frac{(l-|m|)!}{(l+|m|)!}} \tag{4.2}$$

The *associated Legendre polynomials* $P_{m,n}(x)$ are recursively defined:

$$\begin{aligned} P_{m,m}(x) &= (1-2m)\sqrt{1-x^2}P_{m-1,m-1}(x) \\ P_{m+1,m}(x) &= x(2m+1)P_{m,m}(x) \\ P_{l,m}(x) &= x\left(\frac{2l-1}{l-m}\right)P_{l-1,m}(x) - \left(\frac{l+m-1}{l-m}\right)P_{l-2,m}(x) \\ P_{0,0} &= 1 \end{aligned} \tag{4.3}$$

The fact that BRDFs are only defined on the upper hemisphere can be taken advantage of in the spherical harmonic decomposition. If it is assumed that the lower hemisphere is a mirror image of the upper one $p(\theta,\phi) = -p(\pi-\theta,\phi)$, then the $l+m =$ odd coefficients of the decomposition are 0, which allows for a faster decomposition and less storage usage.

There are several problems with this approach. The first one is that all basis functions have global support. Thus, to evaluate the spherical harmonic expansion at one location (θ,ϕ) it is necessary to evaluate all the basis function of the expansion. This is expensive.

In addition the way a function is decomposed does not seem to be ideal for most real world BRDFs. Most BRDFs are almost constant (low frequency) over most of their domain, but have some high frequency components in certain areas, so that a large number of coefficients is required. Personal communication with Stephen Westin [58] confirmed this. More than 300 coefficients are necessary to get a good approximation of such BRDFs.

Another disadvantage is that spherical harmonics are only defined on a spherical domain. The BRDF is a 3-dimensional function in the isotropic case and a 4-dimensional function in the anisotropic case. Thus, the spherical harmonics have to be expanded to higher dimensions. Sillion et. al. [46] suggests performing the spherical harmonic decomposition for a set of fixed incoming light direction $(\theta_i{}', \phi_i{}')$ and then using a cubic interpolation of the coefficients to get the value at any position in between. In terms of the Monte-Carlo simulation, this requires that the incoming direction can not be randomly chosen, or that the incoming directions have to be resampled to be uniform. Both approaches are not very good and introduce additional error.

The approach taken by Westin et. al. [58] uses a spherical harmonic decomposition of the coefficients to approximate the dependency on the incoming direction. This solution suffers from the same problems as the cubic interpolation approach, but it might result in a better approximation. The author did not comment on this.

4.2 Splat Basis

In Section 3.4.1 splats were introduced to represent photons. A splat is a four dimensional normal distribution centered around the position $P = (\vec{\omega}' \rightarrow \vec{\omega})$ of the light source and the position of the radiometer.

While a normal distribution makes intuitive sense, it is hard to compute the value of the BRDF for a particular incoming and outgoing direction. Because of the infinite support of the Gauss distribution, *all* the photons have to be evaluated and summed up. Another problem arises from the fact that the domain of a BRDF is not infinite, thus the Gauss distribution has to be trimmed. Furthermore the integral of the bounded distribution has to be computed to normalize the photon. This is necessary to ensure energy conservation. Since there is no analytical solution for the definite integral of a Gauss distribution available, numerical methods must be applied. All these restrictions make the Gauss distribution difficult to use.

4.2.1 Splats

To solve these problems, this thesis uses a polynomial approximation of the Gauss curve with similar shape, but a bounded domain.

$$f_{R_i}(r) = \begin{cases} C\left(-\frac{4}{9}\left(\frac{r}{R_i}\right)^6 + \frac{17}{9}\left(\frac{r}{R_i}\right)^4 - \frac{22}{9}\left(\frac{r}{R_i}\right)^2 + 1\right) & \text{if } -R_i \leq r \leq R_i \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

This function is used in geometric modeling to define the field function for meta-balls [59]. A comparison of the Gauss curve and the polynomial approximation can be seen in Figure 4.1. The dashed curve is Gauss and the solid curve is the approximation. In contrast to the Gauss distribution where the standard deviation is a parameter of the curve, this function allows the specification of the distance between the origin and the point where the function becomes 0. When

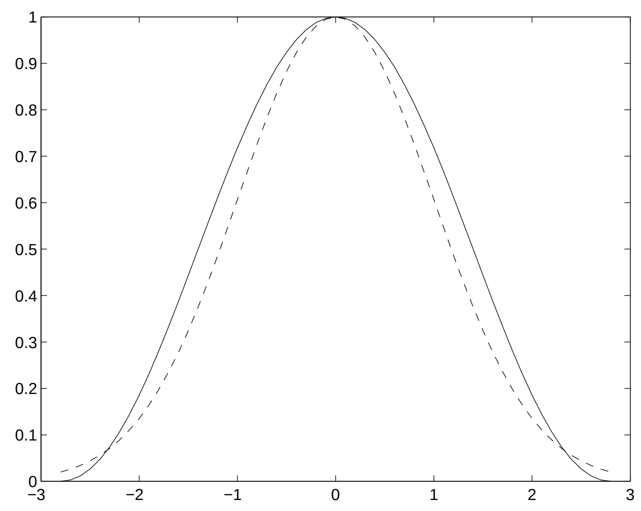


Figure 4.1: Polygon approximation of the Gauss-curve.

the function is used as a radial basis function, this distance is the radius of the splat, its area of influence measured from the center of the splat.

It is necessary to be able to limit the area of influence in order to allow convergence towards the correct solution. As stated in Section 3.6 the area of influence gets smaller as the number of traced photons is increased.

The function 4.4 has some nice properties. At position $r = R_i$ the value of the function is 0 and so is the tangent. At $r = 0$ the value of the function is 1 and the tangent is 0. C is a normalization constant.

In order to be able to normalize each splat properly (according to its probability and energy) the integral of the splat has to be computed.

When the radius of a splat is small compared to the radius of the hemisphere, as will be usually the case in our application, the integral can be approximated by a splat defined over a plane.

The Integral over the interval $[-R_i, R_i]$ is

$$F^{(1)}(R_i) = \int_{-R_i}^{R_i} f_{R_i}(r)dr = \frac{944}{945}R_i \tag{4.5}$$

The two dimensional rotation symmetric integral is:

$$F^{(2)}(R_i) = \int_0^{R_i}\int_0^{2\pi} f_{R_i}(r)d\phi dr = \int_0^{R_i} 2\pi r f_{R_i}(r)dr = \frac{8}{27}\pi R^2 \tag{4.6}$$

Since a rotation symmetric four-dimensional version of this distribution is used, the integral over the domain has to be computed. The domain depends on the 4D-norm (see Section 3.4.2). For the L_1 norm the integral is:

$$\begin{aligned} F_{L_1}(R) &= \int_0^R \int_0^{2\pi} \int_0^{R-r} \int_0^{2\pi} f_R(r+s)d\phi_2 ds d\phi_1 dr \\ &= \int_0^R 2\pi s \int_0^{R-s} 2\pi r f_R(r+s)\ dr\ ds \\ &= \tfrac{37}{1620}\pi^2 R^4 \approx 0.0228\pi^2 R^4 \end{aligned} \tag{4.7}$$

For the L_2 norm, the integral is:

$$\begin{aligned} F_{L_2}(R) &= \int_0^R \int_0^{2\pi} \int_0^{\sqrt{R^2-r^2}} \int_0^{2\pi} f_R(\sqrt{r^2+s^2})d\phi_2 ds d\phi_1 dr \\ &= \int_0^R 2\pi s \int_0^{\sqrt{R^2-s^2}} 2\pi r f_R(\sqrt{r^2+s^2})\ dr\ ds \\ &= \tfrac{37}{540}\pi^2 R^4 \approx 0.06851\pi^2 R^4 \end{aligned} \tag{4.8}$$

The previous two equations computed the integral under the splat if the area it covers does not intersect the boundary of the hemisphere. If it intersects, the integration domain can be split into a part that is entirely within the hemisphere and a part that is only partially within the hemisphere. If the splat on the outgoing hemisphere is D away from the boundary of the hemisphere, the integral can be split up as follows (for the L_1 norm):

$$\begin{aligned} G_{L_2}(R) &= \int_0^R \int_0^{2\pi} \int_0^{R-r} \int_0^{2\pi} f_R(r+s)d\phi_2 ds d\phi_1 dr \\ H_{L_2}(R) &= \int_0^R \int_0^{2\pi} \int_D^{R-r} \int_0^{2\pi - \arccos\frac{D}{s}} f_R(r+s)d\phi_2 ds d\phi_1 dr \\ F_{L_2}(R) &= G_{L_2}(R) + H_{L_2}(R) \end{aligned} \tag{4.9}$$

This can be solved, by taking advantage of the radial symmetry of the integrand (similar to Equations 4.7):

$$\begin{aligned} G(R) &= \int_{R-D}^R 2\pi s \int_0^{R-s} 2\pi r f_R(r+s)\ dr\ ds \\ H(R) &= \int_0^{R-D} 2\pi s \int_0^D 2\pi r f_R(r+s)\ dr\ ds \\ I(R) &= \int_0^{R-D} 2\pi s \int_D^{R-s} 2\pi r \arctan\left(\tfrac{D}{r}\right) f_R(r+s)\ dr\ ds \\ F_{L_1}(R) &= G(R) + H(R) + I(R) \end{aligned} \tag{4.10}$$

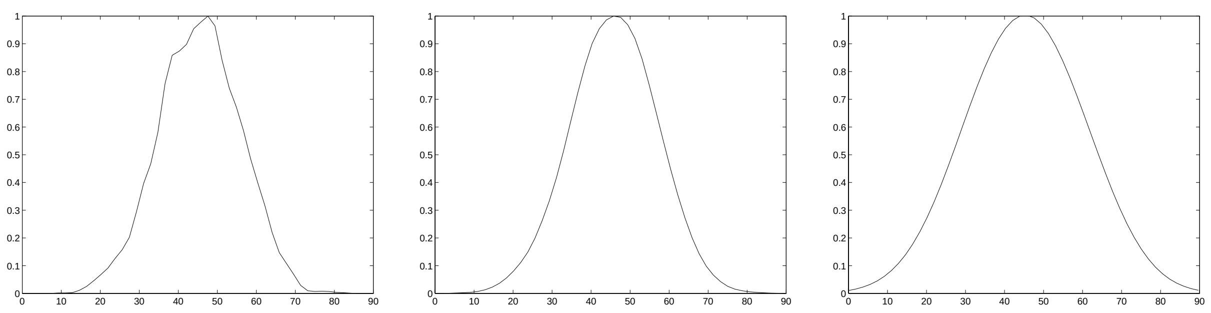


Figure 4.2: The effect of different splat widths.

All these integrals can be solved analytically using *Maple* [6]. Since the solutions are rather long they are not printed in this thesis. In the other cases, in which the splat intersects the boundary of the incoming hemisphere, or both hemispheres are intersected, the integral can be computed in a similar way. The same is true when the L_2 norm is used.

4.2.2 Splat Radius

While the integral of a splat has to be proportional to the energy of the photon, the radius of the splat can be chosen rather freely. In order to ensure convergence against the solution, the radius of the splat has to converge to 0 as the number of traced photon paths goes to infinity (In the limit, the splat will be a delta distribution).

But this criteria still allows us to chose the radius rather freely. For a fixed number of traced photons, widening the splat is similar to a low pass filtering of the BRDF data. The wider the splat, the less high frequency is in the BRDF. On the other hand, the smaller the splat is, the more high frequency components are in the BRDF, but the amount of noise is higher, too. To compensate for the noise, more photon paths have to be traced.

Figure 4.2 show a section of the BRDF of a phong surface, sampled with 100000 photons. The image on the left shows the section with a splat width of 0.08 radians, the middle one with 0.15 radians and the right one with a splat width of 0.3 radians.

Choosing the distribution automatically is rather difficult, since the knowledge of the micro model is limited. If we know that the micro-model is a single diffuse polygon, the radius of the photon can be largely independent of the number of photons shot. On the other hand if a highly specular surface is sampled we need a small radius, otherwise the sharp peak in the BRDF, typical for a specular surface, can not be represented appropriately (see Figure 4.2). If the reflectance is both specular and diffuse, a compromise has to be found.

4.2.3 Normalization

The splat $f_R(r)$ has to be normalized in order to provide energy consistency. This means that the value of C in Equation 4.4 has to be determined.

If only one light source position is considered, and the flux hitting the surface is 1, the reflected light has to be $1 \cdot O$, where O is the reflectance. Thus, if n rays are shot, the sum of all splats, before multiplying by O has to be 1.

$$C_o = \frac{1}{\sum_1^n F_i^{(2)}(R)} \tag{4.11}$$

The $F_i(R)$ are identical for all splats that do not intersect the boundary of the hemisphere. For splats intersecting the boundary, the integral is different depending on the distance from the boundary and can be computed according to Equation 4.9.

For multiple light source positions, the sum of all the splats on the incoming direction has to add up to 1 at any incoming direction. Since the total area of the unit hemisphere is 2π, and the splats are randomly distributed, the normalization constant on the incoming direction is:

$$C_i = \frac{2\pi}{\sum_1^n F_i^{(2)}(R)} \tag{4.12}$$

Since the incoming and outgoing direction are linked together by either the L_1 or the L_2 norm the combined normalization constant is $C_i \cdot C_o$.

$$C_i = \frac{2\pi}{\sum_1^n F_i(R)} \tag{4.13}$$

Note that $F(R)$ is different for the L_1 and the L_2 norm (see Section 4.2.1, Equation 4.8).

4.3 Spatial Subdivision

The splats introduced in the previous section have a bounded support. Thus, not every photon is relevant for a particular incoming and outgoing direction $P = (\vec{\omega}' \to \vec{\omega})$. In order to speed up the computation of the BRDF it has to be known which splats contribute to the BRDF at position P.

A similar problem occurs in rendering to solve the ray-object intersection problem. Spatial subdivisions are used that divide the space into cells [37]. For all surfaces that intersect the cell a pointer to this surface is stored. Thus, only the intersections with surfaces in the cell have to be computed when the ray travels through this cell.

The same spatial subdivision approach can be used to classify splats. The domain of the BRDF is subdivided into cells of approximately the same size. For each of these cells a list of pointers is stored that reference the splats whose support intersects the cell. Since the domain of the BRDF consists of points on two hemispheres, we will first describe how a hemisphere is subdivided and then extend the concept two four dimensions.

4.3.1 Subdividing the Hemisphere

The hemisphere is first subdivided along the parallels into *bands* of equivalent width (in radians). Depending on the circumference of each of these bands, it is subdivided into a number of spherical rectangles, or spherical triangles in the band that contains the north pole. Figure 4.3 shows an example. (A polar plot of the sphere is presented.)

The number of rectangles a band is subdivided into, is computed as a function of the length of the lower boundary of the band. The outermost is divided into a user specified number of rectangles $n_{\max}$. The inner bands are subdivided into cells of approximately the same area as the ones on the outermost band.

If the hemisphere is subdivided into n_{bands} then the bth band is subdivided into n_{rect} rectangles, which is computed according to the following formula.

$$n_{\text{rect}} = \max\left(\text{round}\left(\sin\left(\frac{\pi}{2}\frac{b}{n_{\text{bands}}}\right)n_{\max}\right), 1\right) \tag{4.14}$$

Thus, the number of rectangles is the product of the maximum number of rectangles $n_{\max}$ multiplied by the radius of the bottom of the current ring (which is the sine of the elevation

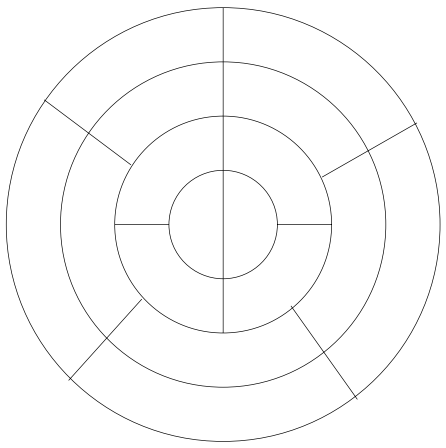


Figure 4.3: Spatial subdivision of a hemisphere.

measured from the north pole). The result is rounded to the nearest integer. The maximum ensures that the number of rectangles is at least 1. Thus, the further away a ring is from the pole, the more rectangles are used to subdivide it.

Figure 4.3 show an example of such a subdivision. The hemisphere is subdivided into 4 bands, with elevations (measured from the north pole) $0 - 22.5$ for the first ring, $22.5 - 45$ for the second, $45 - 75.5$ for the third and $75.5 - 90$ for the forth ring. Using formula 4.14, the first ring is subdivided into 2 triangles, and the other bands are subdivided in 4, 4 and 5 rectangles.

For each of these cells the intersecting splats have to be computed and stored with the cell. Thus, we have to be able to compute the distance d between two points on the sphere. This can be done using the following formula ($d_{\cos}$ is the cosine of the distance d):

$$d_{\cos} = \cos\theta_1 \cos\theta_2 + \sin\theta_1 \sin\theta_2 \cos(\phi_1 - \phi_2) \tag{4.15}$$

Since the domain of a splat $S = (\theta_s, \phi_s, r)$ consists of all the points within a circle of radius r around its center, the intersection of a circle on the sphere with the boundaries of the subdivision have to be computed.

Computing the intersection with the bands is rather straightforward. If $\theta_{\min}$ and $\theta_{\max}$ ($\theta_{\max} > \theta_{\min}$) are the elevations of the boundaries of a band, then a splat intersects a band when its center is closer than r to the top or to the bottom boundary.

$$\theta \geq \theta_{\min} - r \quad \text{and} \quad \theta \leq \theta_{\max} + r \tag{4.16}$$

The intersection test with a meridian is not quite as simple. The intersection points can be calculated by solving Equation A.1 for θ_1. More precisely, if $\theta_2 = \theta_s, \phi_2 = \phi_s$ identify the center of the circle, $\cos_d$ is the cosine of the radius r, and ϕ_r is the angle of the meridian for which the intersection should be calculated. Thus, the following parts of Equation 4.15 are constant:

$$a = \cos\theta_2$$
$$b = \sin\theta_2 \cos(\phi_1 - \phi_2)$$

Substituting a and b in Equation 4.15, leads to the following equation that has to be solved

$$cos_d = a\cos\theta_1 + b\sin\theta_1$$

solving for θ_1 results in:

$$\theta_{1,1} = 2\arctan\left(\frac{b + \sqrt{b^2 - cos_d{}^2 + a^2}}{cos_d + a}\right), \quad \theta_{1,2} = 2\arctan\left(\frac{b - \sqrt{b^2 - cos_d{}^2 + a^2}}{cos_d + a}\right) \tag{4.17}$$

This can be used to compute the intersection points I_1 and I_2 with the meridian. The intersection points are then checked to see if they are within the current band. If at least one of the intersection points is inside, the splat is inside the cell. If one is above the band and one is below the band, the splat is inside. Otherwise the splat is outside.

4.3.2 Subdividing the BRDF

The partitioning of a sphere can be extended in a straightforward way to subdivide the 4D domain of a BRDF. Every cell on the hemisphere of incoming directions contains a reference to the partitioned hemisphere of outgoing directions. (See Figure 4.4.)

The area of influence of one photon depends on the norm that is used to measure distances. But any sensible norm requires that the total distance is larger than or equal to the distance on one of the spheres. All the distances described in Section 3.4.2 follow this rule.

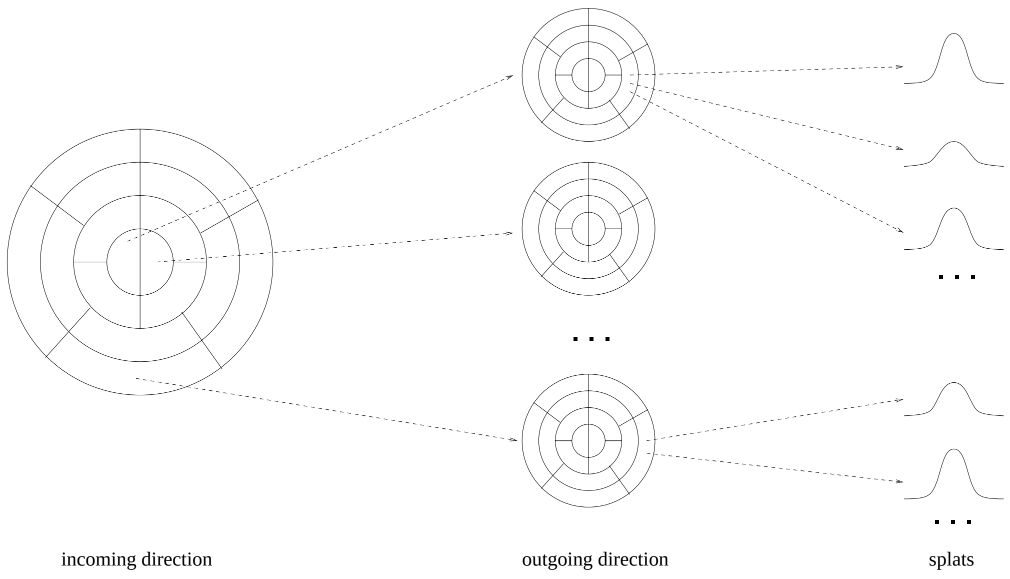


Figure 4.4: Spatial subdivision of the domain of a BRDF.

In order to get a norm independent subdivision, splats are assigned on the incoming side independently from the distance on the outgoing side and vice versa. That means that a splat is assigned to cells on the incoming hemisphere without looking at its outgoing direction. This defines a set of outgoing hemisphere cells the splat has to be assigned to. For every outgoing hemisphere the splat is assigned to the cells it intersects, again without looking at its incoming direction.

This cell assignment strategy works for every norm, but is not optimal. Let's assume a splat has radius r and barely intersects a cell C_I on the incoming hemisphere, after which it gets assigned to some cells on the outgoing hemisphere, where again it barely intersects a cell C_O. If its minimum distance from C_I is $3/4r$ and from C_O is $3/4r$ then its distance in 4D space according to the L_2 norm (Equation 3.5) is

$$d = \sqrt{\frac{9}{16}r^2 + \frac{9}{16}r^2} = \sqrt{\frac{18}{16}}r \quad \geq \quad r \tag{4.18}$$

Thus it should not be in cell (C_I, C_O). If, in contrast, the L_∞ norm (Equation 3.5) is used, it should be in cell (C_I, C_O).

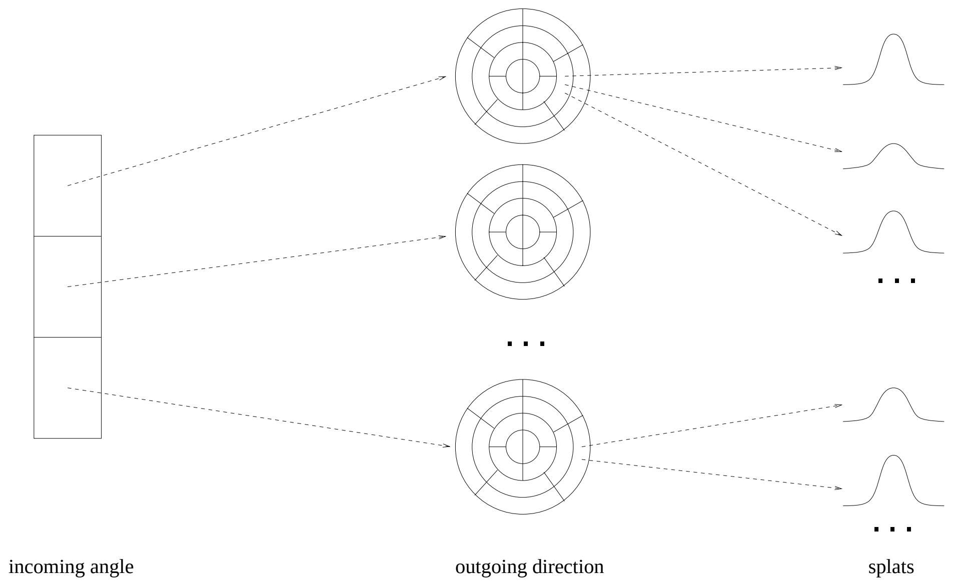


Figure 4.5: Spatial subdivision of the domain of an isotropic BRDF.

$$d \;=\; \max\left(\frac{3}{4}r, \frac{3}{4}r\right) \;=\; \frac{3}{4}r \quad < \quad r \tag{4.19}$$

4.3.3 Isotropic BRDFS

The subdivision of an isotropic BRDF is simpler. Since the domain is only three dimensional, the subdivision has one level less, see Figure 4.5. The incoming zenith is subdivided into ranges. For every range, a subdivided sphere is used as described in the previous section. The zenith of the outgoing direction is represented by the zenith on the sphere, and the difference in longitude between the incoming and outgoing direction is represented by the longitude on the sphere.

4.3.4 Results

The effectiveness of the method depends on the number of splats in a cell. The fewer splats there are in every cell, the faster the BRDF can be evaluated. The average evaluation time is proportional to the average number of splats in a cell.

The BRDFs described in Chapter 5 have 1/10th to 1/20th of their total number of splats in one

BRDF	nr photons	splat width	nr bands	nr rect max	average	ratio
lambertian	100,000	0.3	5	5	≈ 7000	0.07
phong	100,000	0.15	5	5	≈ 5000	0.05
brushed metal	300,000	0.5	5	5	$\approx 10,000$	0.033

Table 4.1: Effectiveness of the subdivision.

cell. Table 4.1 shows some examples. *nr splats* is the total number of photons in the BRDF, *splat width* is the width of a splat, *nr bands* is the number of bands the incoming and outgoing hemispheres are subdivided in, *nr rect max* is the maximum number of rectangles a band is subdivided in. *average* is the average number of splats in a cell, and *ratio* is the ratio of the average to the total number of splats, which is inverse proportional to the speedup.

Lambertian refers to the BRDF described in Section 5.1.1 which has a uniform distribution of photons. *Phong* is described in Section 5.1.2. The better speedup is due to the smaller splat width and the fact that most splats are concentrated in a few cells, which leads to a large number of almost empty cells. *Brushed metal* (Section 5.2) is an example for an anisotropic BRDF. The splat width is rather high, but since the BRDF is anisotropic there are more cells (4D subdivision).

The effectiveness of the method depends mainly on the radius of the splats. If splats with a large radius are used, even a finer subdivision of the domain will not result in a big improvement of the number of splat in a cell, since the large splats will intersect a large number of small cells. Experiments have shown that a cell width and hight of approximately the size of the splat is a good tradeoff between required memory and performance.

4.4 Future Research

In the past few years the interest in representing functions on the sphere has grown, resulting in the development of more and more techniques. Some of these techniques might be usable to represent BRDFs.

One promising idea is to use spherical wavelets as proposed by Schröder et. al. [40]. Spherical wavelets are an extension of the wavelet theory to support a more general topology of the domain.

Another idea is to use approaches similar to the hemi-cube algorithm use in radiosity [7] that project the hemisphere onto a number of rectangles. Standard wavelet techniques can then be used to compress the BRDF.

Chapter 5

Applications

This chapter shows applications of the BRDF sampling technique described in this thesis. The first two examples show the sampling of a Phong and Lambertian shaded plane. These examples are provided to demonstrate the effects of importance sampling and filtering. The third example shows an anisotropic BRDF for brushed metal that is created from a model of parallel cylinders.

The fourth example shows how this approach can be used to simulate the reflectance properties of grass. A comparison with measurements of a patch of real grass is given to verify the results obtained from the simulation.

5.1 Simple Models

This first section describes some very simple models, a plane with a standard shader applied to it. This is presented to show how the number of samples effects the generated BRDF, and to what extent high frequency effects can be reproduced by the model. The first example shows how a perfectly diffuse BRDF can be reproduced, while the second section shows how high frequency components are reproduced by the model.

5.1.1 Lambertian

The first example places a perfectly diffuse plane in the center of the sampling hemisphere (normal pointing towards the pole of the hemisphere). As described in Section 2.3.1, the BRDF of a

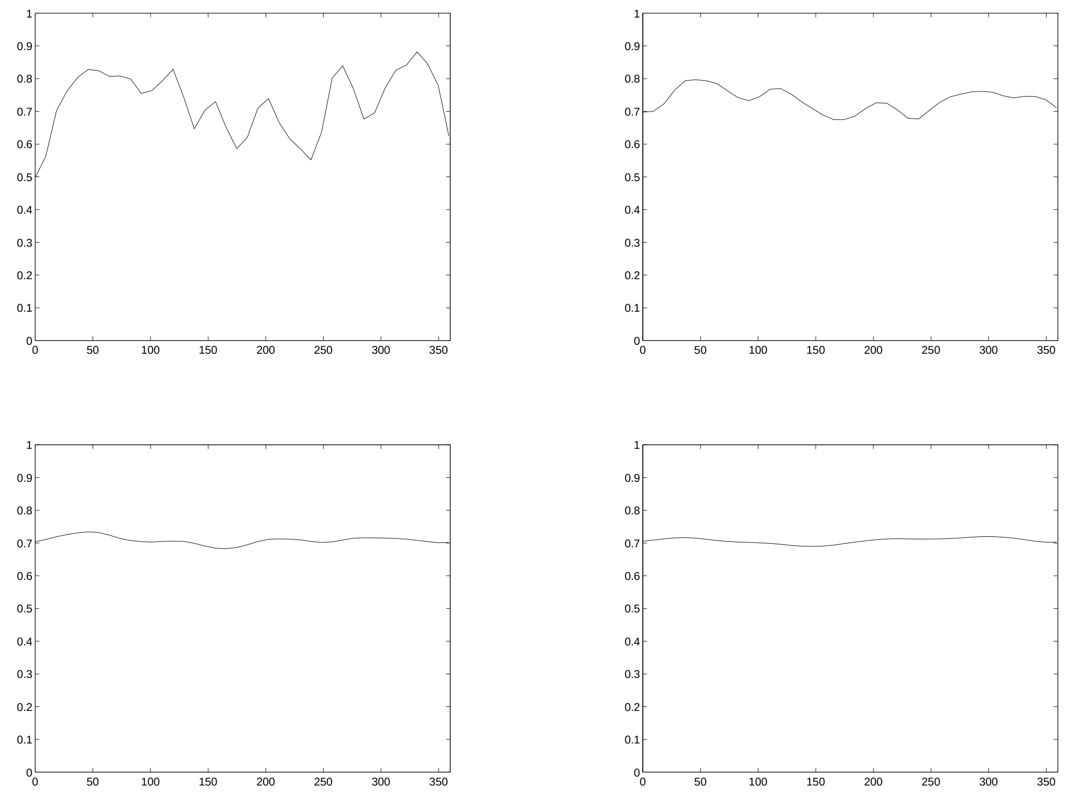


Figure 5.1: Different splat width for a lambertian BRDF.

lambertian surface is constant. For a constant BRDF importance sampling results in a uniform distribution of the outgoing rays over the hemisphere.

Since the micro geometry is a plane, with a constant reflectance, the BRDF generated by the sampling process should be a constant function again.

Figure 5.1 shows the variation of the generated BRDF with the azimuth. All the graphs show the BRDF with a incoming zenith angle of 45 degrees and an outgoing zenith angle of 45 degrees. The azimuth of the outgoing direction is plotted on the x-axis ranging from 0 to 360 degrees. Note that a lambertian BRDF is constant over its domain, thus every cross section should be a line.

To generate the BRDF 100,000 rays are shot, randomly distributed over the sampling area. Each ray is scattered in a random direction. Since the micro model is a simple plane, no multiple scattering occurs. The top left graph of Figure 5.1 shows the reconstructed BRDF with a splat

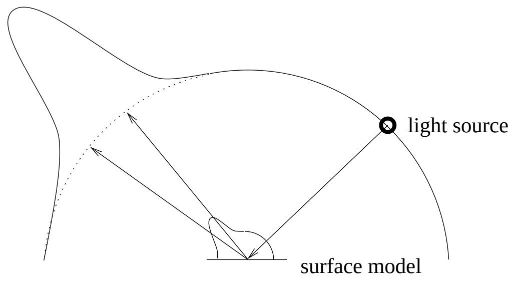


Figure 5.2: The reconstructed BRDF is similar to the Phong micro BRDF.

radius of 0.3 radians. The BRDF is still rather noisy. Increasing the splat radius to 0.5, which is similar to a low pass filtering, results in a significantly better approximation of a straight line (upper right graph). Increasing it even further to 0.7 (lower left) and 0.9 (lower right) results in BRDFs that are indistinguishable from a perfect diffuse BRDF when applied to a surface. It has to be noted that for an infinite splat radius, every splat distribution will result in a constant BRDF. But as described in the next section, specular effects require a small splat width which can not be captured with large splats.

It has to be noted that diffuse BRDFs are very hard to represent with the splatting technique. Since the shape of a splat is similar to a bell curve, a large number of splats, or a large splat width, is necessary to approximate a constant function. Thus, mainly diffuse surfaces with some high frequency components require a large number of samples when high frequency components are to be represented adequately.

5.1.2 Phong

The specular part of the Phong model (see Section 2.3.2) is another extreme BRDF. Light is only reflected in a small area around the mirror direction. As in the previous example, the micro BRDF of the plane should be recreated by the simulation system (see Figure 5.2).

Figure 5.3 shows the BRDF generated by the simulation in comparison with an actual Phong BRDF. The simulation used 100,000 photons and importance sampling.

Since importance sampling was used, most of the scattered rays where traced into directions with a high reflectance. Thus most of the paths carry high energy. This allows for a small filter

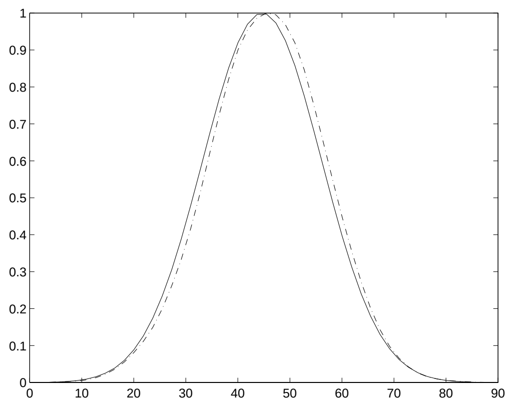


Figure 5.3: Comparison of a perfect phong with a reconstructed one.

width of 0.15 which results in an almost perfect reconstruction of the original BRDF.

The BRDF used a Phong exponent of 25, which results in small specular highlight and is usually considered high enough to get reflectance similar to specular real world objects. This example shows that the splatting technique is well suited to reconstruct high frequency phenomena in BRDFs.

5.2 Brushed Metal

One approach to model the BRDF of anisotropic surfaces is described in [36, 28]. This model uses a number of parallel cylinders that simulate the scratches created by brushing metal in one direction. Figure 5.4 shows an image of the micro structure. Since the model has a clear directional structure, the reflectance properties will not only depend on the relative angle between the incoming and outgoing direction but on the actual incoming and outgoing light directions.

The micro geometry used for the simulation consists of 50 parallel cylinders. The cylinders overlap by 50 percent (that means that cylinder n touches cylinder $n+2$). The BRDF used for the micro structure is a perfectly specular Phong model with an exponent of 25. Thus the cylinders are almost, but not quite perfect mirrors.

The simulation traced 300,000 rays. Since the BRDF is anisotropic, the light source position is chosen randomly on the entire hemisphere. A splat radius of 0.9 was used to reconstruct the BRDF from the recorded photons.

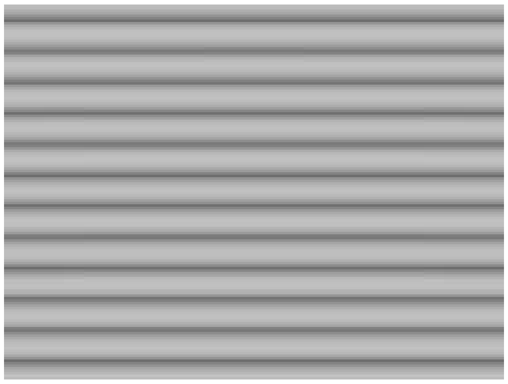


Figure 5.4: The micro structure model used for brushed aluminum.

Figure 5.5 shows some cross sections of the created BRDF. The topmost graph shows the BRDF as a function of the outgoing longitude. The incoming and outgoing zenith is 45 degrees and the incoming longitude is chosen such that it is perpendicular to the orientation of the cylinders. The graph shows that there is a large forward scattering (around 180 degrees) and an even bigger backward scattering (around 0 degrees). Taking a look at the details of the micro geometry this is not surprising. Since the light source is located at a zenith of 45 degrees more than 50 per cent of the photons will hit the cylinder at positions from which the photon is scattered backwards. The other, smaller part of the photons are scattered forward. A few photons are scattered to the side, which is due to the fact that the cylinders are not perfect mirrors.

The middle graph in Figure 5.5 shows the BRDF if the micro model (or the incoming direction) is rotated by 45 degrees. In this case two effects contribute to the BRDF: forward scattering and scattering in the direction of the cylinders. This leads to a rather wide maximum at about 225 degrees (180 + 45 degrees).

Rotating the model even further, causing the cylinders to be parallel to the incoming light direction, results in a narrow specular highlight at 180 degrees. In this case the only effect that can be observed is forward scattering of the photons. Some photons are scattered away from the 180 degrees direction because they hit the sides of the cylinders, but the dominant effect is a relatively sharp, Phong like maximum at 180 degrees. (See Figure 5.3) for a comparison to the Phong BRDF.)

The graphs plotted in Figure 5.6 show the BRDF in a configuration similar to the top graph in

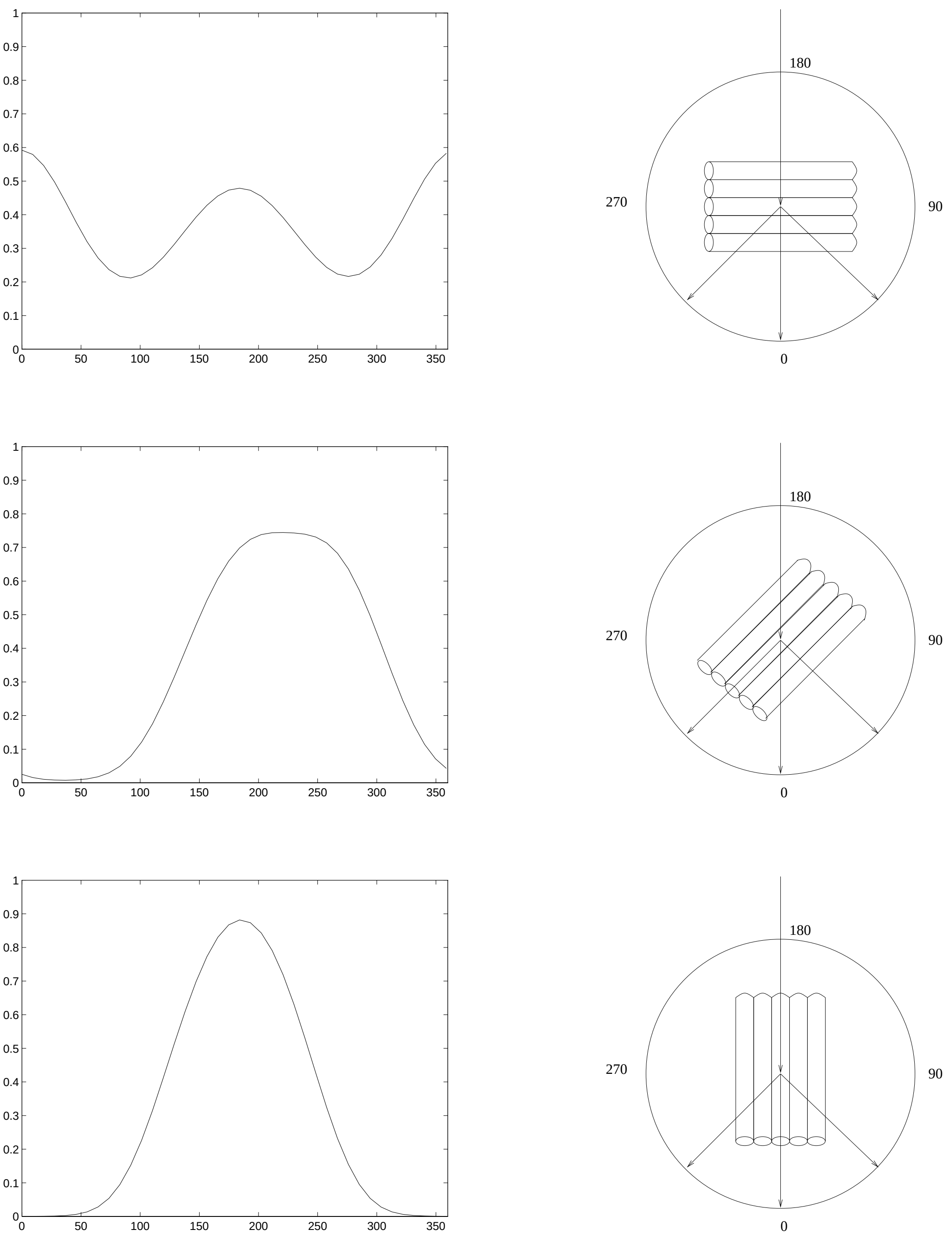


Figure 5.5: Some cross sections of the BRDF.

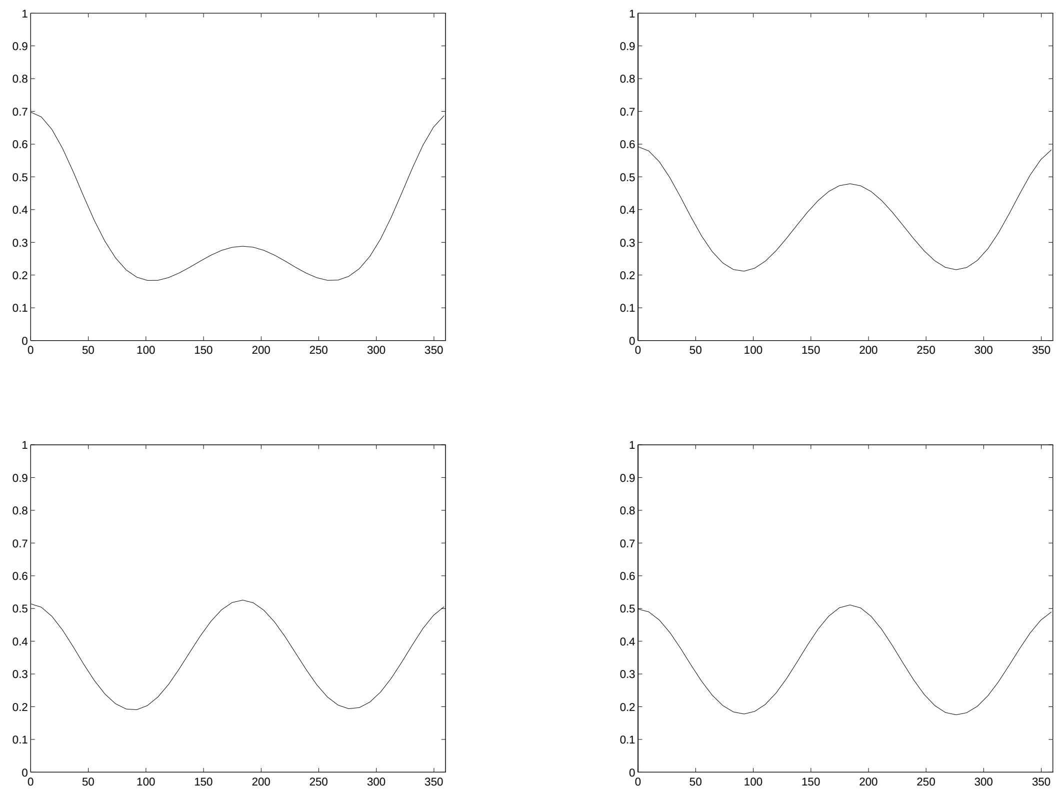
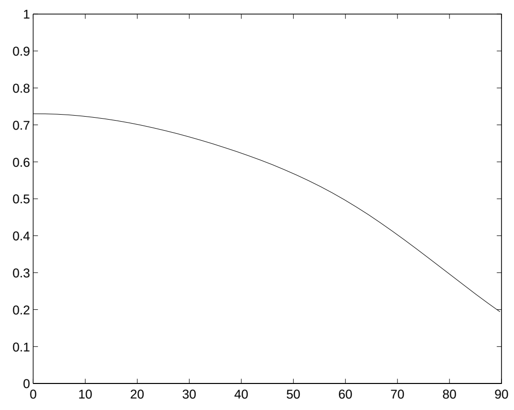


Figure 5.6: Some cross sections of the BRDF.

Figure 5.5. The outgoing direction is 45 degrees and the cylinders are perpendicular to the incoming direction. The top left graph shows the BRDF plotted as a function of outgoing longitude, with the zenith of the incoming light set to 70 degrees. The top right image has an incoming zenith of 45 degrees, the lower left of 30 degrees, and the lower right a zenith of 0 degrees.

As can be seen, the further the light goes down the larger is the difference between forward and backward scattering becomes. At an zenith of 70 degrees the backward scattering is more than twice as big as the forward scattering, at a zenith of 45 degrees there is still more backward scattering. As the position of the light source moves closer to the pole of the hemisphere, the amount of forward and backward scattering becomes almost identical.

This is because only a part of the cylinders is visible from the light source. At an angle of 70 degrees only the backfacing parts are visible, thus most of the light gets scattered backwards. At 0



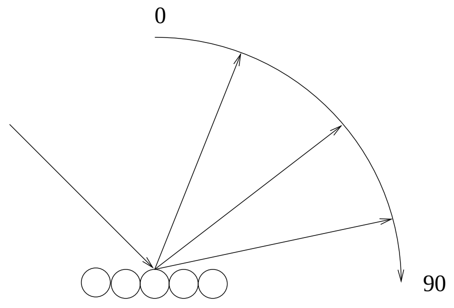


Figure 5.7: A cross section of the BRDF, with varying θ.

degrees the probability that a front or backfacing part of a cylinder is hit by a photon is identical, thus the amount of forward and backward scattering is identical.

Figure 5.7 shows the dependence of the forward scattering (longitude 180 degrees) as a function of the zenith of the outgoing direction. The light source is located at $(45, 0)$.

The picture 5.8 shows the BRDF applied to the bottom of a frying pan. The direction of the scratches is radial around the center of the pan. Thus, the micro structure that is introduced by the BRDF is a number of concentric tori. The pan is illuminated from a light source to the right of the viewer. The BRDF creates two V-shaped highlights starting at the center and going up and down wards, an effect that can not be created by an isotropic BRDF.

5.3 Grass

While looking for an interesting model of a micro structure that can be used by the simulation system, we came across the reflectance properties of grass, viewed from a distance sufficiently far away so that individual pieces of grass are not visible by the viewer. Grass is supposed to have a high amount back scattering [12].

This section describes how we [1] measured the reflectance of grass. Then a simple model of grass is created which is used by the simulation system to create a BRDF. This BRDF is then compared

[1]Thanks to Wiliam Cowan and Wilkin Chau for their help in taking these measurements.

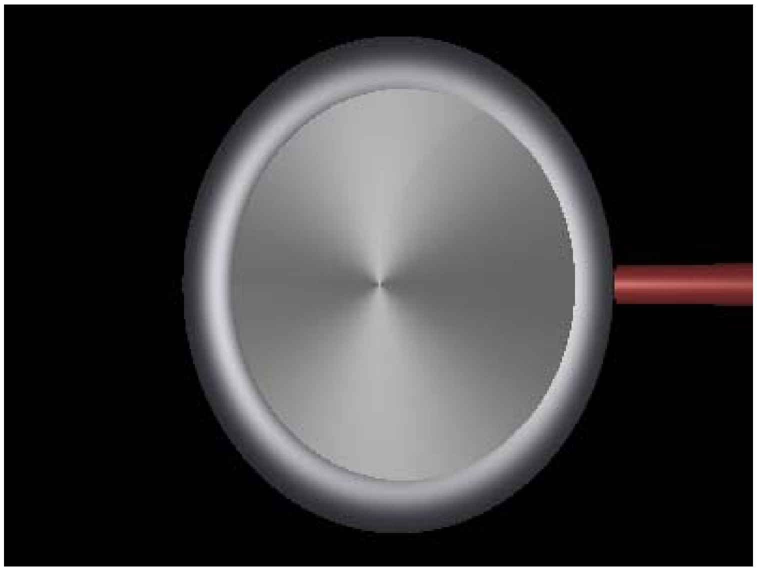


Figure 5.8: A pan rendered with the brushed metal BRDF.

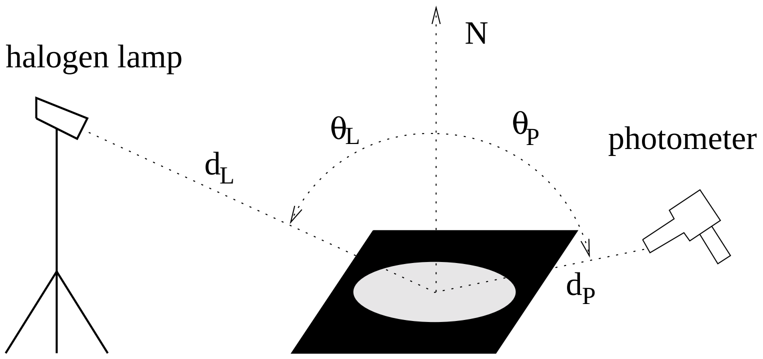


Figure 5.9: Setup for measuring the reflectance of grass.

to the measured reflectance values.

5.3.1 Measuring Grass

To measure the reflectance properties of grass we started looking for a nice uniform piece of short cut grass. After finding one, we set up a light source and the photometer as illustrated in Figure

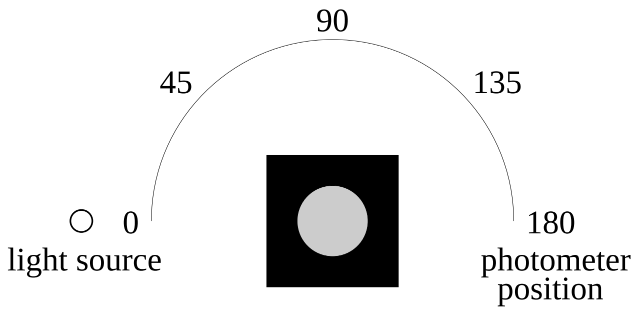


Figure 5.10: Longitude of samples.

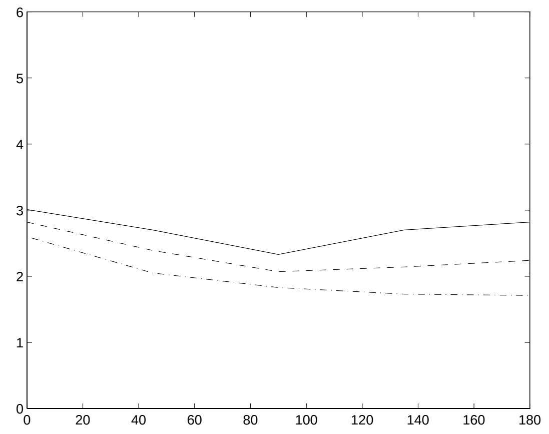
5.9. We used a square board of wood and cut a circular whole in the middle with a radius of 1 foot. The remaining board was painted flat black. We placed this board on the grass, the grass visible through the hole was our measuring area. The black painted board was used to eliminate light originating from sources outside of the measuring area. All the measurements where done at night.

A halogen lamp was placed on a tripod at a distance of 10 feet from the center of the measuring area. The reflected light was measured using a photometer at a distance of 10 feet.

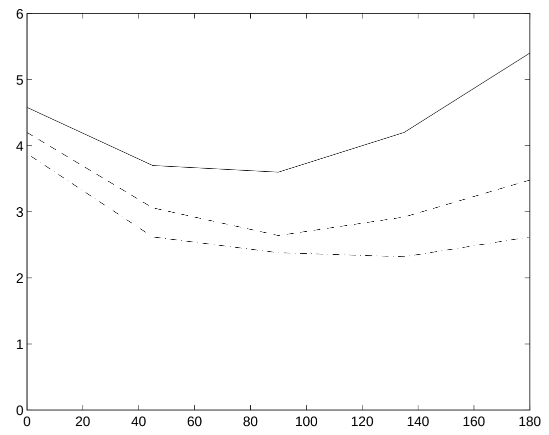
Since we assumed the grass to be isotropic, we only changed the longitude of the light source. We took measurements with the light source positioned at a zenith of 45, 60 and 70 degrees. For each light source position we measured the reflected light at zenith of 60, 70 and 80 degrees. For each pair of incoming and outgoing zenith, measurements were taken at longitudes of 0 degrees (right underneath the light source), 45 degrees, 90 degrees, 135 degrees, and 180 degrees opposite of the light source (see Figure 5.10).

Figure 5.11 shows the results of the measurement. The first diagram shows the reflectance of grass for the light source positioned at an zenith of 45 degrees. The solid line shows the change of the reflectance as a function of the longitude (see Figure 5.10) with the photometer positioned at a zenith of 80 degrees. The photometer was positioned at 70 degrees for the dashed line and at 60 degrees for the dotted line.

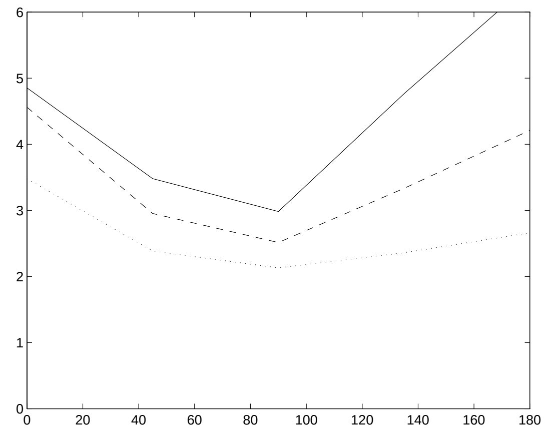
The second diagram shows the same sequence of measurements but with the light source at a zenith of 60 degree. The third diagram has the light source positioned at 70 degrees. In order to be able to compare the different diagrams, the measured values have been normalized such that the



light source:	45 degrees	
photometer:	solid:	80 degrees
	dashed:	70 degrees
	dotted:	45 degrees



light source:	60 degrees	
photometer:	solid:	80 degrees
	dashed:	70 degrees
	dotted:	45 degrees



light source:	70 degrees	
photometer:	solid:	80 degrees
	dashed:	70 degrees
	dotted:	45 degrees

Figure 5.11: Reflectance of grass.

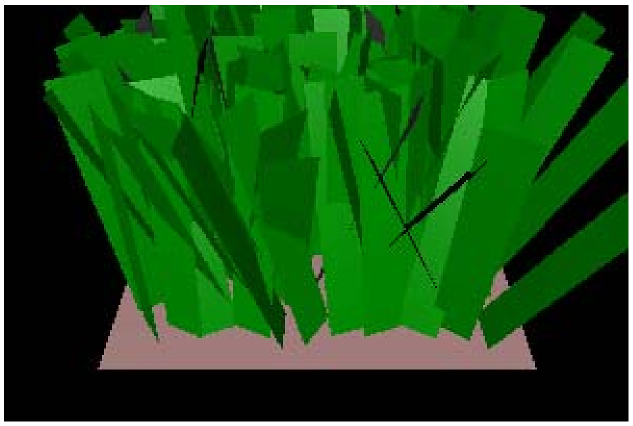


Figure 5.12: A model for grass.

incident flux on the measurement area is the same for all light source positions. That means that the measured values are divided by the the cosine of the zenith of the light source.

The reflectance of grass shows some unusual qualitative properties. At low angles of the light source there is a high backward and forward scattering (0 degrees and 180 degrees) compared to the light that gets reflected into the perpendicular directions around 90 degrees. This property becomes more and more pronounced as the light source is moved closer to the ground.

Another interesting property is that more light gets reflected towards low outgoing angles. The further down the photometer is, the more light is measured. This is true for all light source positions and all outgoing zenith.

These extraordinary properties make grass an interesting test case for the BRDF simulation system.

5.3.2 Grass Model

Figure 5.12 shows the first attempt to create a model for grass. The individual blades are rectangles that are rotated around their center (left diagram in Figure 5.13) by an angle α and around their base by an angle β (see right diagram in Figure 5.13).

For the model in Figure 5.12 the angle α was chosen randomly between 0 and 360 degrees. The angle β was between 0 and 20 degrees. The width was 0.01 and the height was 0.06. There were

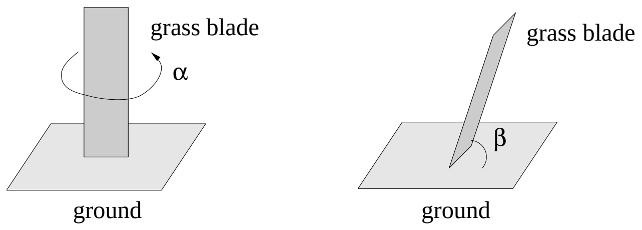


Figure 5.13: Creating of the grass model.

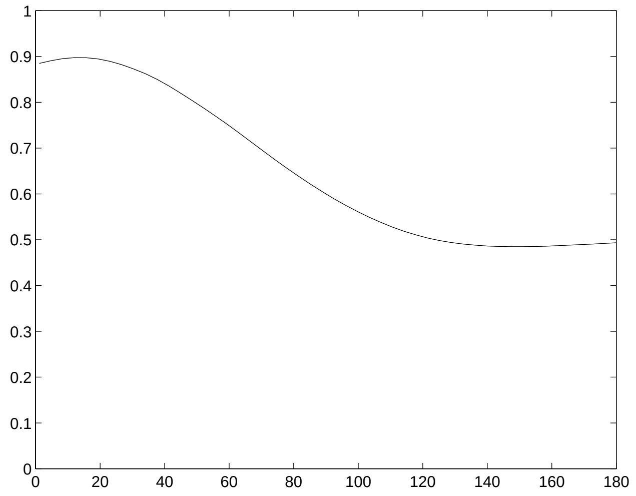


Figure 5.14: BRDF of the first try for grass.

20×20 grass blades placed uniformly placed on a 0.5.5 piece of "ground".

The BRDF that resulted from this simulation was not satisfying. While there was a high back scattering, the increase in forward scattering compared to a diffuse BRDF was not visible. Figure 5.14 shows a cross section of the BRDF with the light source positioned at a zenith of 30 degrees and the measuring device has a zenith of 20 degrees.

Thus the model had to be changed. The new model can be seen in Figure 5.15. The blades of grass are significantly shorter (0.03) and a little bit narrower. This results in less dense grass. In addition the grass was flatter. That means that the bending angle β (see Figure 5.13 ranged from 0 to 85 degrees. An image of the micro model can be seen in Figure 5.15.

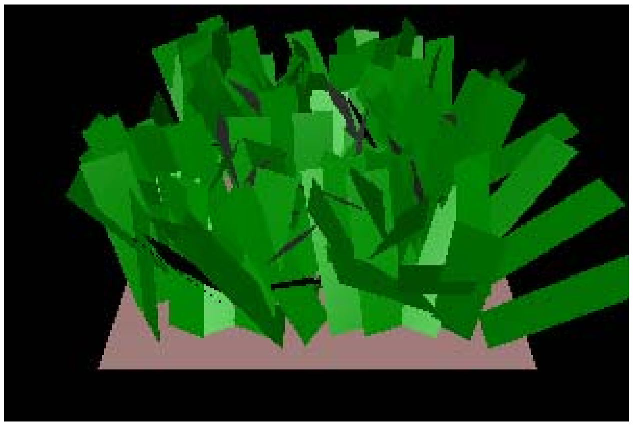


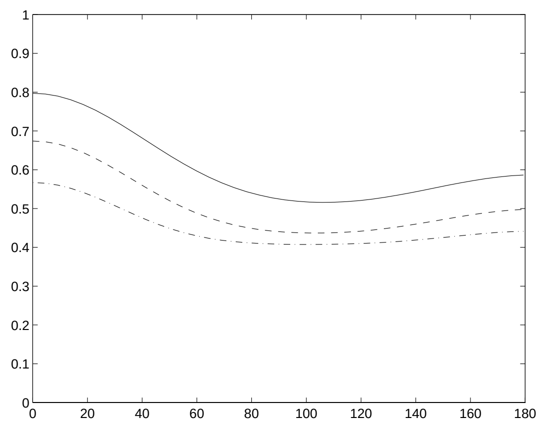
Figure 5.15: A better model for grass.

The resulting BRDF was much closer to the measured data. Figure 5.16 shows a plot of the BRDF at different incoming and outgoing angles. The first graph shows the reflected light with the light source positioned at a zenith of 45 degrees. The solid, dashed and dotted lines correspond to an outgoing zenith of 80, 70 and 60 degrees.

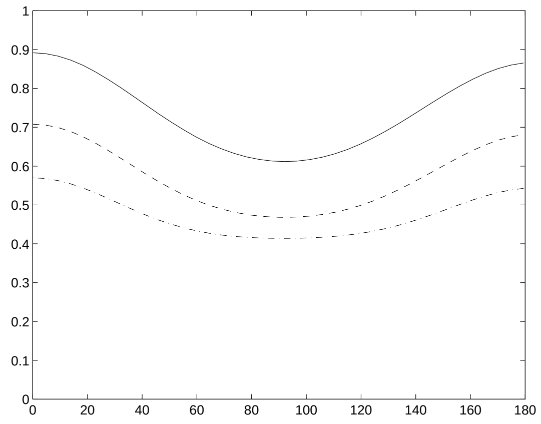
Most of the effects that are visible in the measured are reproduced by the simulation. At small zenith angles of the light source, the BRDF is mainly diffuse except for high backscattering. At a larger light source angles the forward and backward scattering is much higher compared to the diffuse reflectance in other direction. The simulation also recreated the higher reflectance at lower outgoing direction.

An image of the grass BRDF applied to a flat surface can be seen in Figure 5.17. A light source is placed right in the center of the grass, about 25 per cent of the width of the field above the plane. The image shows two highlights that are created by the grass BRDF; one because of the high back scattering and one because of the hight forward scattering. The one further away from the observer is brighter because of the higher back scattering component of grass compared to the forward scattering.

The sequence of images presented in Figure 5.18 shows a similar effect. In the image on left, the light source is right behind the observer ($\phi_{in} = 0$). The light reflected from the grass is rather bright because of the high backscattering component. The image in the middle has the light source



light source:	45 degrees	
photometer:	solid:	80 degrees
	dashed:	70 degrees
	dotted:	45 degrees



light source:	60 degrees	
photometer:	solid:	80 degrees
	dashed:	70 degrees
	dotted:	45 degrees

Figure 5.16: Simulated reflectance of grass.

placed right of the observer ($\phi_{in} = 90$). The image is darker compared to the left image, because grass scatters less light perpendicular to the incoming light direction. In the image on the right the light source is place opposite of the observer ($\phi_{in} = 180$), thus the light scattered forward can be seen. The image is darker than the left one but brighter than the middle one.

Different Micro BRDF

The next example examined the the influence of the micro BRDF on the BRDF of the whole model. The BRDF simulation was run on two models of grass. For the first one, a micro BRDF was used with a specular component of 0.5 and a diffuse component of 0.5. This was compared to the BRDF created by a micro model with the same geometry but a specular component of 0.7 and a diffuse

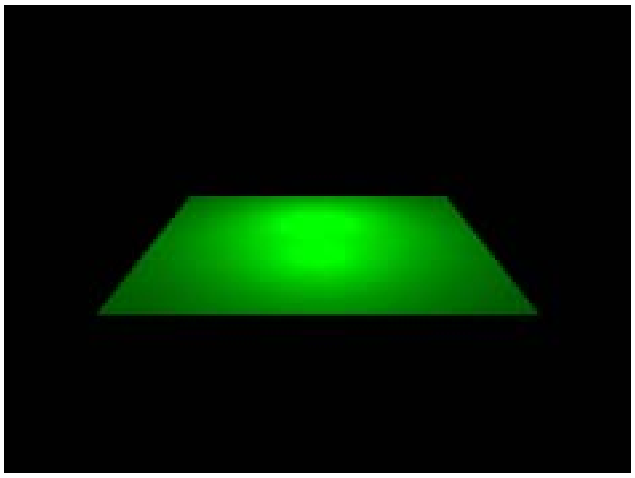


Figure 5.17: The two highlights of the grass BRDF.

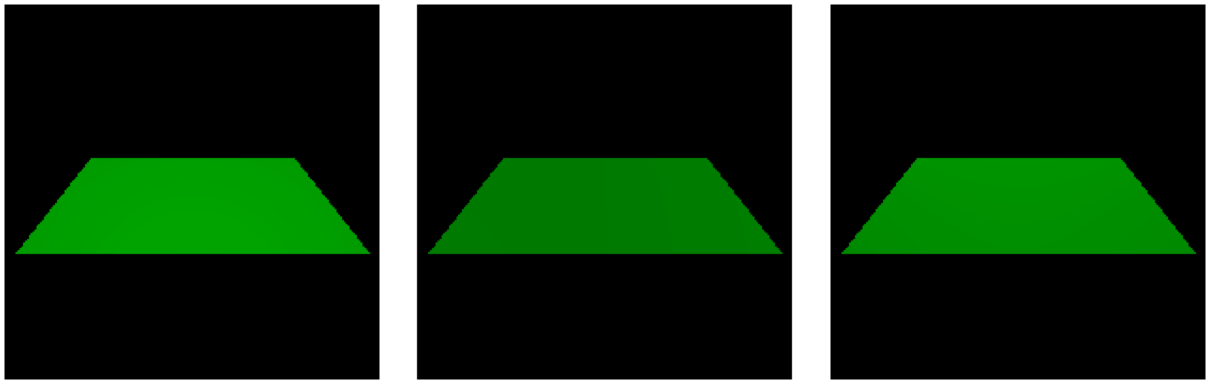


Figure 5.18: Grass with the light source at 0, 90 and 180 degrees.

component of 0.3.

The result can be seen in Figure 5.19. The cross section is plotted is for a light source position of 60 degrees and an photometer position of 80 degrees. The solid line represents the BRDF with a micro BRDF of similar diffuse and specular components.

The difference in the two BRDF is rather small for a substantial change of the micro BRDF. This is mainly due to the fact that details of the reflectance of an individual blade of grass are

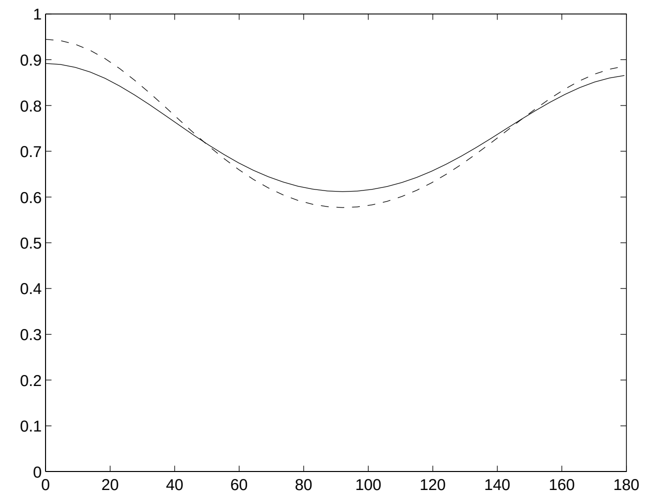


Figure 5.19: Reflection of grass depending on the micro brdf.

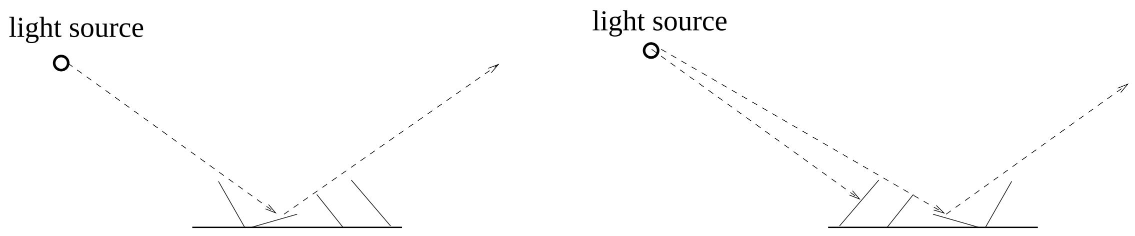


Figure 5.20: Two light source positions, relative to the mowing direction.

averaged out because of shadowing and multiple reflections. Thus the choice of a micro BRDF is not as critical as it might appear to be at first glance.

Anisotropic Grass

One of the most noticeable effects of grass can be seen in sports arenas after the grass has been mowed. If it is mowed in different directions, the colour of the grass appears different depending on the direction of the mowing. This is a consequence of the mower bending the grass in different directions. Note that the BRDF for mowed grass is no longer isotropic.

Figure 5.20 shows an example. The left diagram shows what happens when the light source

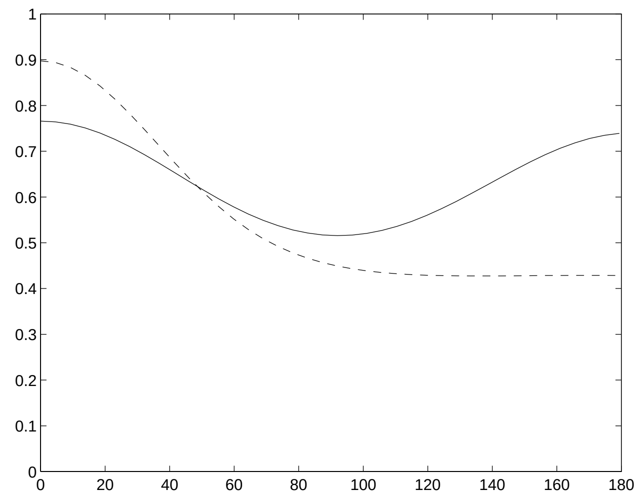


Figure 5.21: The reflectance of mowed grass depending on the light source position.

is in the direction towards which the blades are bending. In this case there is only a significant forward scattering. The backward scattering is similar to the diffuse reflectance (see the dashed graph in Figure 5.21). If the light source is on the other side, there is both backwards scattering from straight blades and forward scattering from blades that are bending (see solid graph in Figure 5.21).

Figure 5.22 shows this anisotropic grass BRDF applied to a surface. The light source is placed behind the observer at an incoming angle of 45 degrees. The different colour of the stripes is due to the way the grass is bending.

The model used for this BRDF simulation is similar to the one presented in Figure 5.15, but the rotation angle α only varied between -10 and 10 degrees (see Figure 5.13).

5.4 Conclusion

This section showed applications of the reflectance simulation system. First two case are presented (Lambertian and Phong) that show how the system handles “extreme” BRDFs. The aluminum example showed that anisotropic BRDFs can be handled using the simulation system. The final example showed how the complicated reflectance properties of grass are correctly reproduced.

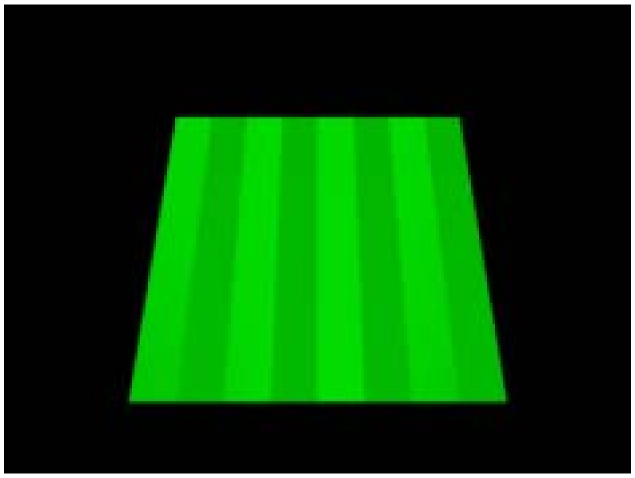


Figure 5.22: An example of mowed grass.

Chapter 6

Colour Representation

The previous chapter showed how the reflection properties of materials can be described. Separate from this, it is necessary to represent the light itself in an appropriate way.

This section shows some of the shortcomings of the commonly used RGB model, and presents an alternative solution. This alternative approach uses a colour space that depends on the lights and material colours that actually appear in the scene.

In order to be able to experiment with multi-spectral representations, an editor for spectral power distributions (SPD) is presented. These SPDs can then be used to interactively specify the illuminant of a pre-rendered reflectance image.

6.1 Colour Spaces

As explained in the introduction to this thesis, colour can be either an SPD or the reflectance property of a material. In both cases, colour is a continuous function of the wavelength. Therefore the remainder of this chapter uses the term *colour* when something is valid for both SPDs and reflectances.

Since general continuous functions are impossible to represent in a digital computer, some other approach has to be used. The representation described in the introduction is straightforward. The continuous function is sampled at uniformly spaced locations, and the value of these samples are used to approximate the colour. Thus, each SPD or reflectance is an n-dimensional vector.

$$c = (c_1, c_2, \ldots, c_n)^\top \in \Re^n$$

Every element of the vector represents the colour at a particular wavelength. In order to increase the accuracy of the approximation, the number of samples can be increased. But as described in the introduction, this is an expensive approach in both computation memory.

An alternative approach to representing a continuous function is a linear combination of basis functions. Thus, a colour is represented as a weighted sum of functions:

$$c(\lambda) = \sum_{i=1}^{m} k_i b_i(\lambda) \tag{6.1}$$

To compute the coefficients k_i the inner product of the colour $c(\lambda)$ with the dual basis function $\tilde{b}_i(\lambda)$ has to be computed:

$$k_i = \int_{\lambda_{\min}}^{\lambda_{\max}} c(\lambda)\tilde{b}_i(\lambda) d\lambda \tag{6.2}$$

The dual basis functions are the unique functions $\tilde{b}_i(\lambda)$ that satisfy the following condition:

$$\int_{\lambda_{\min}}^{\lambda_{\max}} \tilde{b}_i(b_j(\lambda))\, d\lambda = \delta_{ij} \tag{6.3}$$

where d_{ij} is the Kronecker delta.

In the case of an orthonormal set of basis function, the basis functions are their own duals. Thus, the projection of a colour into the space defined by the basis B is the inner product of the SPD with the basis function b_i.

$$k_i = \int_{\lambda_{\min}}^{\lambda_{\max}} c(\lambda) b_i(\lambda) d\lambda \tag{6.4}$$

There are many types of orthonormal basis functions that can be used. Examples include the Fourier basis and wavelet basis. The approach taken in this thesis is to use point sampled basis functions. Thus, every basis function is a vector $b \in \Re^n$. The whole set of basis functions can be written as a $m \times n$ matrix.

$$B = \begin{pmatrix} b_1^\top \\ b_2^\top \\ \dots \\ b_m^\top \end{pmatrix} = \begin{pmatrix} b_{1,1}, & b_{1,2}, & b_{1,3} & \dots & b_{1,n} \\ b_{2,1}, & b_{2,2}, & b_{2,3} & \dots & b_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{m,1}, & b_{m,2}, & b_{m,3} & \dots & b_{m,n} \end{pmatrix} \in \Re^{m \times n} \tag{6.5}$$

Rewriting Equation 6.1 in matrix notation.

$$c = B^\top k_B \tag{6.6}$$

In this equation c is a n element, point sampled colour, while k_B is the coefficient vector. In general m is between 3 and 12 [13] and thus much smaller than n which is usually between 300 and 400 (one sample per nm in the visible spectrum between $380nm$ and $780nm$). Thus, Equation 6.6 converts from a representation in terms of the basis B to a point sampled representation.

To project a point sampled colour c into the space defined by the matrix B, a dot product of c with the dual basis functions has to be computed: $k_i = c \cdot \tilde{b}_i$. As stated before, if the basis functions are orthonormal, the dual basis vectors are identical to the basis vectors. In this case, the projection can be done by computing the dot product of the colour with the basis functions: $k_i = c \cdot \tilde{b}_i$.

This can be expressed in matrix notation:

$$k_B = Bc \tag{6.7}$$

In the rest of this chapter we will assume that all basis functions are orthonormal.

To calculate the reflected light o_B in terms of the basis B of an incoming SPD i_B and a reflectance r_B, we first convert i_B and r_B into point sampled colours, perform component by component multiplication, and project the result back into the basis B:

$$o_B = B(B^\top r_B \circ B^\top i_B) \tag{6.8}$$

This is rather complicated and computationally intensive. It can be simplified if we rewrite the reflectance r_B as a $m \times m$ diagonal matrix with the elements of r_B on the diagonal. Then equation 6.8 can be written as:

$$o_B = RBB^\top i_B \tag{6.9}$$

Since we assume that the basis vectors b_i are orthonormal, the dot product $b_i \cdot b_j$ is 1 if $i = j$ and 0 otherwise. In this case $BB^\top$ is the identity matrix. Since R is a diagonal matrix, o_B can be computed by a component by component multiply of r_B and i_B. Thus, if a m dimensional basis is used, m multiplications are necessary to compute the reflected SPD.

Now that we have defined how colours are converted to basis representations, how we can compute the reflected light and how we can convert back to a point sampled vector, the question remaining is: What is a good basis and how many basis vectors are necessary?

6.1.1 Tri-Stimulus Colour Spaces

As described in the introduction we can take advantage of the shortcomings of the human visual system. The human eye has three types of cones that are used to perceive colour. Roughly speaking one type of cone is especially responsive to red light, one is responsive to green light, and one is responsive to blue light. The colour sensation perceived depends on the output signal generated by these sensors [20]. To get a specific colour sensation, we have to create the receptor response that is specific for this sensation. Thus each colour sensation can be described by three values that are in some form related to the output of the three types of cones. Since the visible light is a continuous spectrum, there are arbitrarily many spectral power distributions that create the same colour sensation. Those SPDs are called metamers.

In other words: A colour can be characterized by three values. Each spectral power distribution corresponds to a triple of these values. The question is now what three basis functions are used.

This leads to the definition of tri-stimulus colour spaces. There is a well defined three-value colour space, called XYZ space. In physiological and psychological experiments the mapping form SPDs to XYZ values has been defined. This resulted in the definition of three, so called, *colour matching functions* $X(\lambda)$, $Y(\lambda)$ and $Z(\lambda)$. The x, y and z components of a SPD can be calculated by convolving the SPD with the appropriate colour matching function.

$$x = \int_{380nm}^{780nm} s(\lambda) X(\lambda) df$$

The XYZ colour components y and z can be computed in the same way..

Any (sensible) three-component colour description can describe any colour visible to a human and it can describe this colour uniquely. One example of such a colour space is the RGB colour space used by computer-monitors and by the television. An XYZ value can be converted into RGB using a simple matrix multiplication:

$$(r, g, b)^\top = M(x, y, z)^\top$$

The inverse of M converts from RGB to XYZ.

Since RGB is sufficient to represent each colour, and all currently available rendering systems use RGB, why isn't the problem solved? The problem is that RGB can represent the light source correctly and it can represent the colour reflected by a surface correctly, but not the reflection process. For example two light sources that have the same RGB-colour might have totally different spectra. If they get reflected by the same surface, the RGB representation of the reflected colours may no longer be the same. This problem is visible if fluorescent light sources are used. The fact that a fluorescent light's SPD has three sharp peaks (see Figure 1.2) is totally lost in the RGB representation, yet those peaks can have a significant interaction with certain surfaces.

6.1.2 Optimal Colour Space

Clearly RGB and XYZ define colour spaces with $n = 3$ basis functions. But as we have shown in the previous section, these colour spaces have problems handling the light emitted by a fluorescent light source. Increasing the number of samples does not necessarily improve the result, since important aspects might still be hidden in one sample, unless we go up to a very large number of samples, in which case computation gets expensive again.

The solution followed in this thesis tries to use a basis that depends on the light sources and surfaces that actually appear in the scene [32].

A scene contains a (usually small) number of light sources $l_i \in R^n$ and a (larger) number of different surfaces with reflectance vectors $r_j \in R^n$. The colours that have to be represented during the rendering process are all the SPDs from the light sources reflected by the surfaces in the scene. Since the light can bounce multiple times, all SPDs that appear (modulo scaling) are created by one, two, ... bounces. If the SPDs l_i and o_ja as well as the reflectances r_k are point sampled, the SPDs appearing during the rendering can be computed as follows:

$$o_1 \;=\; l_i \circ r_{j_1}$$

$$\begin{aligned} o_2 &= l_i \circ r_{j_1} \circ r_{j_2} \\ &\dots \\ o_k &= l_i \circ r_{j_1} \circ r_{j_2} \circ r_{j_3} \dots \end{aligned} \tag{6.10}$$

Now if we pick the m "most important" o_k, especially the ones that have been created by bouncing once, we can define a matrix $B \in R^{m \times n}$, $O = (o_1, o_2, \dots o_m)$.

We want an "optimal" low dimensional basis for these vectors. This can be done using the singular value decomposition of O ([18]).

$$[U, S, V] = \mathrm{svd}(O)$$

The matrix $S \in R^{m \times n}$ contains the eigenvalues of $O^\top O$ on the diagonal of its upper $m \times m$ part. The eigenvalues on the diagonal are sorted in decreasing order. The first k ($k \leq m$) columns of $U \in R^{m \times m}$ contain the basis vectors for the optimal k-dimensional colour space to represent all vectors in O ($V \in R^{m \times m}$ is irrelevant in this application. For more information on singular value decomposition see [18]).

Optimal here means that if we represent all the vectors in O using only the first k rows of U, and calculate the difference to the actual value using the L_2 norm, than this basis is the one that minimizes the sum of all the L_2 errors (in the SPDs, not in perception).

The running-time for this calculation is $O(n^3)$ but if the size of the matrix O is reasonable (for example 400 by 400) the time needed to render the actual image is significantly higher.

6.1.3 Reducing the Number of Vectors in the SVD

How do we keep the matrix O this small? There are an infinite number of vectors o_k: every light source reflected by every surface (one bounce) each of these reflected by every surface again (two bounces) and so on. One thing we can do is to use only the "one bounce" vectors. Because they contribute the largest amount of energy to the final image this is a reasonable approach. Even in this case the number of vectors is the product of the number of (different) light sources and the number of (different) surfaces in the scene. But one property helps us here, namely that rendering obeys the rule of superposition [31], which means that we can render an image independently for each light source and then simply add the generated images together. The result that we get is identical to rendering the scene with all light source turned on at once.

In the case of finding an optimal colour space this property proves useful. Now we can find an optimal colour space for each light source individually. Thus, if we use only first generation rays, the number of o_k used in the singular value decomposition is linear in the number of different surface properties.

Of course, we don't have to render the image for each light source independently. It can be rendered for all light sources at once if we store the representation of each surface reflectance in any of the used colour space. The time needed to combine the individual components to get the colour of the final pixel is very small compared to the time needed to actually trace a ray.

6.2 Viewer for Spectral Images

Full spectral rendering is not used very much, because most people don't have an intuitive feeling for the effects that cannot be created with an RGB colour representation. People are used to specifying colours in a tri-stimulus colour representation such as RGB or HSV. This section describes a viewer for spectral images that allows one to apply a spectral illuminant to the image. This can be used as a platform for experimenting with spectral specification of light sources.

6.2.1 Reflectance Images

In a classical rendering systems, a scene is composed of surfaces with certain surface properties and light sources that illuminate the scene. The image is then rendered, and the results of the computation is an image that can be viewed on the CRT. Thus the final images are a description of the SPD at each pixel on the screen.

An alternative approach can be used if there is only one light source in the scene. Instead of storing the SPD at each pixel, the reflectance coefficient of the visible object is stored.

In the simplest case, there is one light source and only direct illumination is considered. At pixel $p(x, y)$ the object A is visible with a BRDF $f(\vec{\omega}' \rightarrow \vec{\omega})$. In this case the reflectance image contains a scaled version of the BRDF, with the outgoing direction $\vec{\omega}$ set to the view direction, and the incoming direction $\vec{\omega}'$ set to the direction of the light source. The scaling factor depends on the distance of the surface from the light source.

Images like this can be created by rendering the image with a neutral light source. Neutral means that the power emitted is independent of frequency.

Having the reflectance image, it is now possible to change the SPD s of the light source. Since the reflectance coefficient, scaled by the relative brightness at every pixel is known, the light actually entering the eye, can be calculated by scaling s with the reflectance at every pixel.

If not only direct illumination is considered, but a global illumination solution is computed, the scaling depends on the geometry of the entire scene and not only on the distance from the light source. This is due to indirect illumination and shadows.

Even in this more complicated case, storing the reflectance values instead of the actual SPDs makes it possible to specify the SPD emitted by the light source after the rendering is done. This is possible because the rendering equation is linear in terms of the light emitted by light sources.

The viewer presented in this section allows viewing of this type of reflectance image. To give the user a feeling of how different SPDs effect the appearance of an image, an interactive SPD editor is included that allows the user to specify the light source.

6.2.2 SPD editor

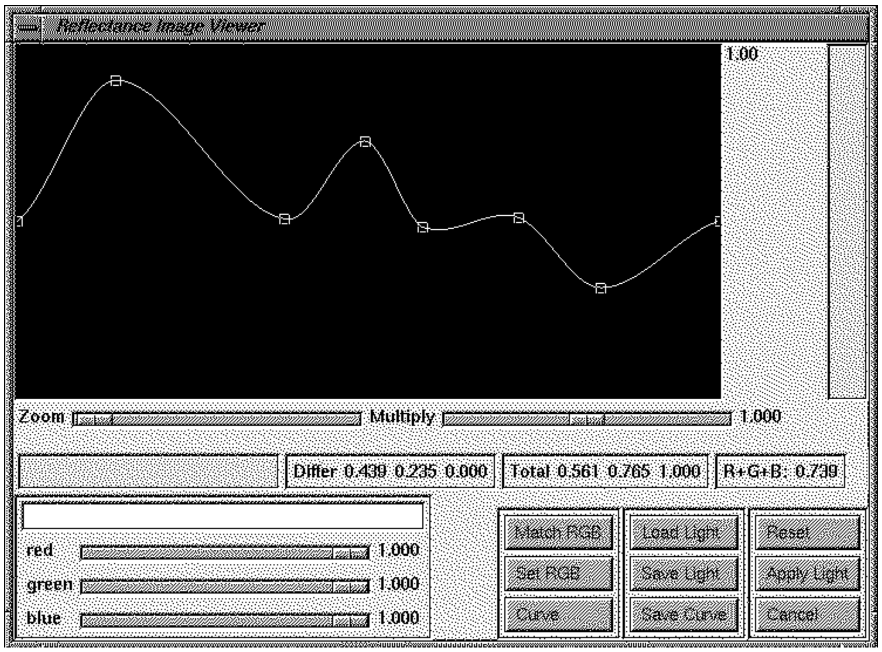
The idea behind the SPD editor is to provide an easy to use interface for editing SPDs. The user interface of the editor can be seen in Figure 6.1.

Catmull-Rom-Editor

To specify the spectral power distribution of the light source, a piecewise polynomial curve is used. The curve is specified using Catmull-Rom-splines (see [15]). The cubic Catmull-Rom-splines are defined as follows:

$$Q(t) = \frac{1}{2}\left(t^3, t^2, t, 1\right)\begin{pmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 1 \\ 0 & 2 & 0 & 0 \end{pmatrix}\begin{pmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{pmatrix}$$

Because Catmull-Rom splines are interpolating, it is possible for the spline curve to go below the lowest control point of the curve. Since values smaller than 0 are not meaningful in describing the SPD of an illuminant, they are clamped to 0.



```
Multiply 1.000
Differ 0.439 0.235 0.000   Total 0.561 0.765 1.000   R+G+B: 0.739
red    1.000
green  1.000
blue   1.000
```

Figure 6.1: The SPD editor.

The user can modify the curve by moving the control points. It is also possible to add new control points and to remove control points. As the SPD is changed, the numerical RGB values of the SPD are displayed and the colour represented by the SPD is shown immediately. To adjust the over all brightness, the SPD can be scaled.

Matching Colours

In order to be able to experiment with different SPDs that represent the same colour, colour matching functionality is incorporated into the editor. The user specifies the desired RGB colour and provides a rough outline of the desired SPD. The matching functionality then changes the SPD so that it matches the specified RGB colour. The change to the SPD is done in a way that changes the overall shape of the SPD as little as possible, meaning the L_2 distance is as small as possible.

The approach was to use a constrained least-squares algorithm. If $s \in \Re^n$ is the specified SPD, $M \in \Re^{3\times n}$ is the matrix converting to RGB, then $c = Ms$ is the RGB representation of s. $\bar{c}$ is the desired RGB-triple s should map to. Thus, s should be changed to $\bar{s}$ s.t.

$$M\bar{s} = \bar{c} \tag{6.11}$$

Since this is a linear equation it can be rewritten in terms of changes to the SPD $\Delta s = s - \bar{s}$. This also requires to define $\Delta c = c - \bar{c}$.

$$M\Delta s = \Delta c \tag{6.12}$$

Since the solution space of this equation is $n - 3$ dimensional, an additional constraint has to be applied to get a unique solution. As stated before, the change Δs to the SPD should be L_2 minimal. Additionally none of the elements of the final SPD $\bar{s}$ can be negative, since an SPD is always greater or equal to 0.

Thus, a linearly constrained least squares problem has to be solved. This can be done using the algorithm described in [21], which solve the following problem:

$$Ex = f \quad \text{equations to be exactly satisfied} \tag{6.13}$$

$$Ax = b \quad \text{equations to be approximately satisfied in the sensethat } ||Ax - b||_2 = \min \tag{6.14}$$

$$Gx \geq h \quad \text{inequality constraints} \tag{6.15}$$

with $E \in \Re^{e \times b}$, $x \in \Re^b$, $A \in \Re^{a \times b}$ and $G \in \Re^{g \times b}$. The algorithm tries to satisfy Equation 6.13 exactly. If this is possible, x is chosen s.t. it approximates Equation 6.14 in an L_2 minimal sense under the constraints imposed by Equation 6.15. $Gx \geq h$ means that every component of Gx has to be larger than the corresponding component in h.

We can use this function to solve the colour matching problem stated above. By setting $E = M$ and $f = \Delta c$ we specify the equality constraints. Thus, the solution x to this equation is Δs. Since equation 6.13 is linear, the new SPD $\bar{s} = s - \Delta s$ solves the equation $M\bar{s} = \bar{c}$. Since none of the elements of $\bar{s}$ can be negative, the inequality constraints matrix $G \in \Re^{n \times n}$ is set to the identity matrix and h is set to $-c$. In addition the solution vector $x = \Delta c$ should be as small as possible, according to the L_2 norm. Thus the approximation equation A is set to identity and b is set to the null vector.

Another problem of the colour matching process is to handle the overall intensity of the SPD. While the user specifies an illuminant, the overall power depends on the brightness of the reflectance

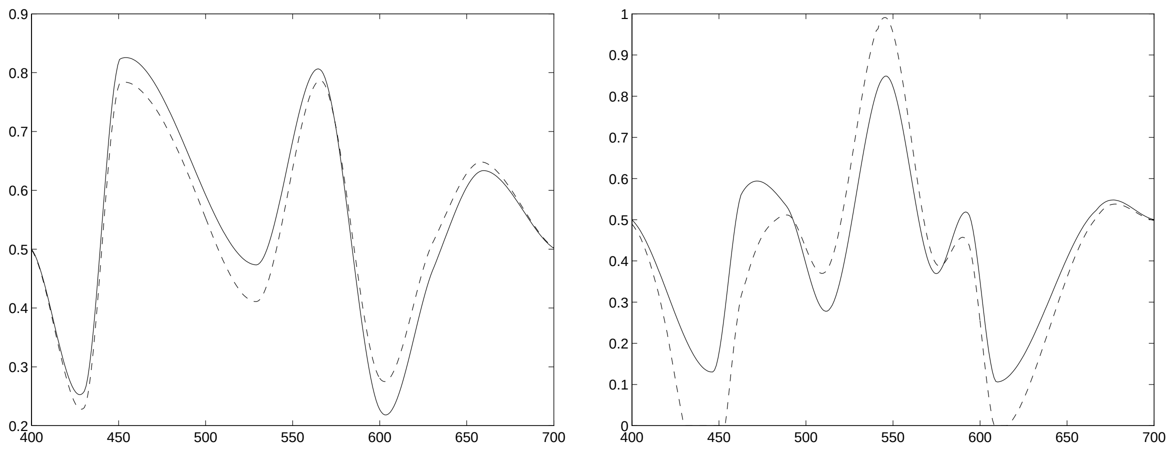


Figure 6.2: Examples of original and matched SPDs.

image. If the reflectance values are low, the power of the illuminant has to be high to be displayable on a CRT and vice versa.

Since the user specifies a colour, the matching process should be independent of the over all SPD brightness. This is achieved by first normalizing the RGB colour that should be matched. The normalized colour is used as $\bar{c}$. For the SPD, the RGB value c' it represents is computed. The L_2 of this value is then used to scale the SPD $s' = ||c'||s$. This results in a brightness independent, hue and saturation matching of the SPD s and the RGB colour $\bar{c}$.

Figure 6.2 shows an example of the colour matching functionality. The left graph shows a user specified graph that represents RGB $(0.79, 1.0, 0.99)$ (solid line). The dashed line is the graph after it has been matched to the user specified colour RGB $(1, 1, 1)$. The plot shows that the general shape of the user defined SPD is well preserved.

The graph on the right shows a different graph, mapping to RGB $(0.2, 1.0, 0.18)$. The dashed graph shows it matched to RGB $(0.52, 1.0, 0.54)$. Since the difference between the colour the graph maps to and the colour specified by the user is bigger than in the previous examples, the difference between the two graphs is bigger. But still, the shape of the original SPD is preserved.

6.2.3 Apply the Light Source and Displaying

To calculate the light reflected (assuming ambient illumination and perfect diffuse reflection) by a pixel, with both light and reflection coefficient represented in terms of different basis functions (each basis itself is orthonormal), Equation 6.9 can be used.

$$o = RB_{ref}B_{light}^{\top}i \tag{6.16}$$

B_{ref} is the matrix containing the basis vectors used to represent the reflectance. R is a diagonal matrix containing the coefficients of the reflectance. B_{light} contains the basis vectors used to represent the illumination, and i is the coefficient vector describing the illumination ($B_{light}^{\top}i$ is a n elements column vector representing the spectral distribution of the light source). o is a column vector describing the reflected light in terms of the colour space B_{ref}.

In the case of the reflectance image viewer, the light source is already a point sampled representation of the SPD. Thus, the multiplication of i with B_{light} can be omitted.

The vector o calculated in Equation 6.16 is represented in the same colour space as the reflectance values. Thus, to display it on the screen a conversion to RGB is necessary:

$$c = M_{ref\_to\_rgb}o$$

If this conversion matrix is not available, it can be calculated by multiplying the matrix converting from n-samples representation to RGB, with B_{ref}.

$$M_{ref\_to\_rgb} = M_{spect\_to\_RGB}B_{ref}$$

6.2.4 Efficient Image Conversion

It is necessary to find an efficient conversion technique from the reflectance space of the image to RGB space, because it has to be applied to every pixel of the image. This conversion depends on the SPD of the illuminant. We will show how this conversion can be done by a single multiply of the reflectance coefficient $r \in \Re^m$ with a $m \times 3$ matrix.

B is the $m \times n$ matrix converting a pixel to its full spectral representation. To convert a pixel r in basis-space to full spectrum, $B^{\top}$ is multiplied by r:

$$s = B^{\top}r$$

To calculate the colour reflected by r when hit by an SPD i, s is multiplied by i. This can be written as a matrix multiplication:

$$s_r = IB^\top r$$

where I is a diagonal $n \times n$ matrix containing i on the diagonal. To convert the result to XYZ space, it is multiplied by the $3 \times n$ matrix X containing the XYZ basis functions:

$$p_{xyz} = XLB^\top r$$

This can be converted to RGB-space by multiplying it with the 3×3 XYZ to RGB conversion matrix M. This results in:

$$p_{rgb} = MXLB^\top r$$

The matrix C defined as $C = MXLB^\top$ is the $m \times 3$ matrix converting from basis to RGB representation.

This matrix depends on the light source. So in order to display an image on the screen, this matrix has to be calculated once and $p_{xyz} = Cr$ has to be calculated for each pixel.

6.2.5 Examples

Figure 6.3 show two images, which are illuminated by light source with the same RGB values, but different SPDs. The colour are clearly different. The two different SPD can be seen in Figure 6.4.

The second example shows the use of the SVD approach in rendering. The image on the left in Figure 6.5 is rendered with full spectral information (61 samples). To render the image on the right, the SPD of the light source and the reflectance vectors where converted to RGB, and the rendering was performed in RGB space. The colours are clearly different.

The image in Figure 6.6 is rendered using the colour space created by a singular value decomposition. The image is almost indistinguasible from the full spectral solution.

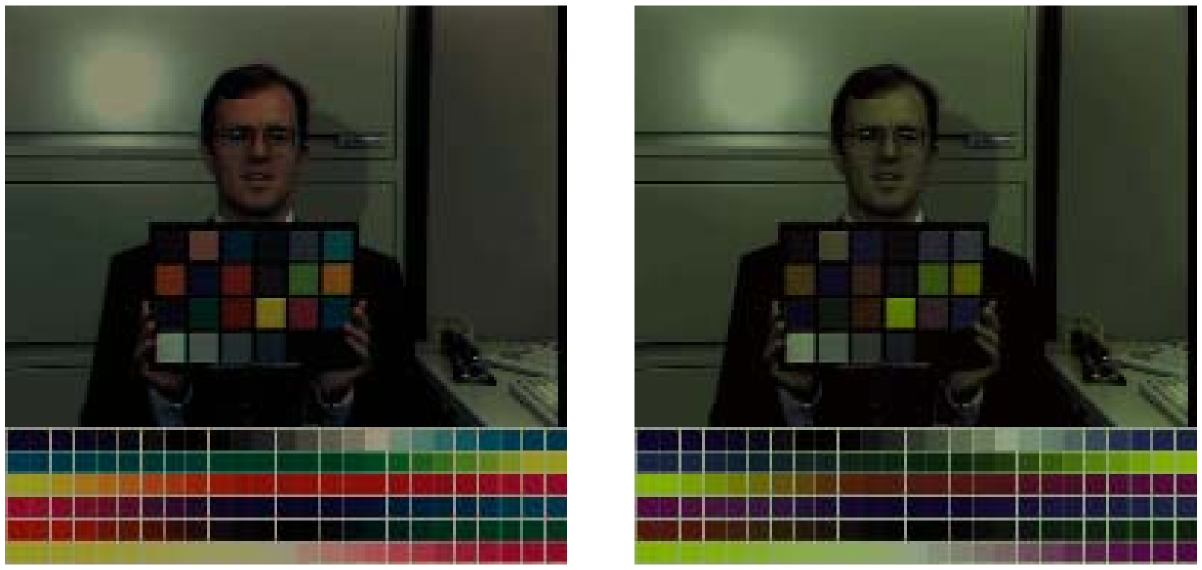


Figure 6.3: Image illuminated with different SPDs.

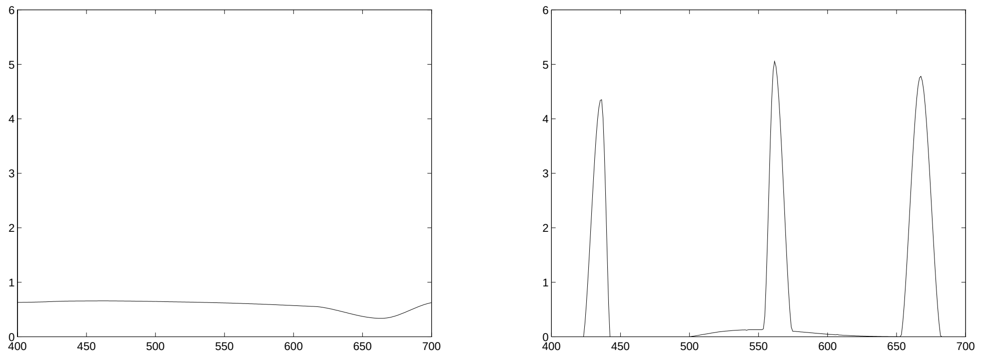


Figure 6.4: SPD used as light sources in Figure 6.3.

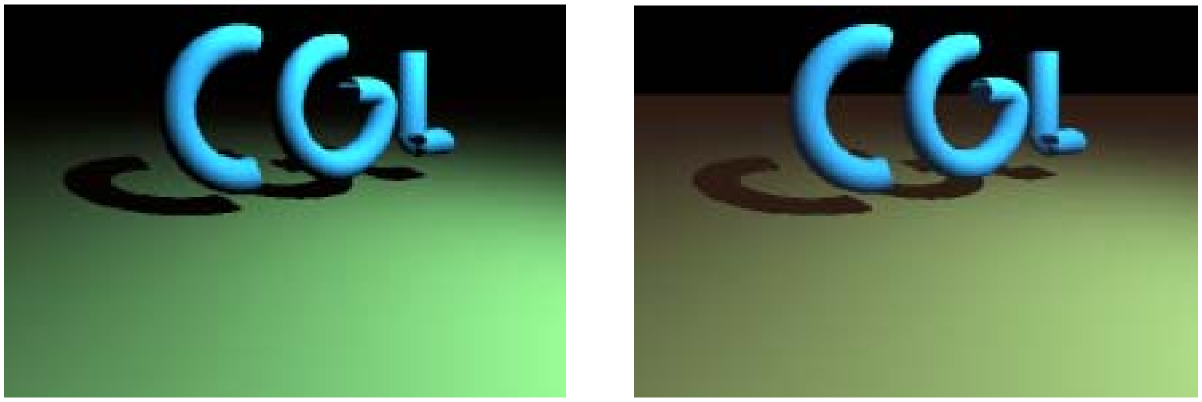


Figure 6.5: Comparison of RGB and full spectral rendering.

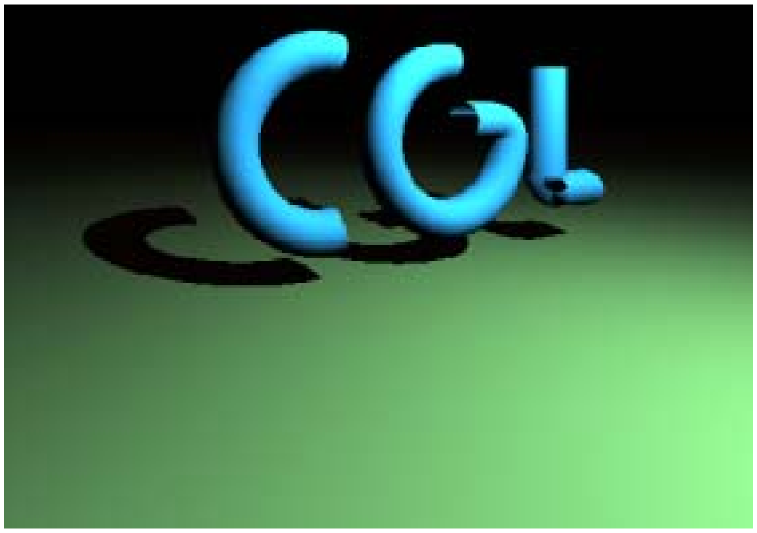


Figure 6.6: CGL image render with SVD colour space.

Chapter 7

Conclusions and Future Work

This chapter presents an overview of the results presented in this thesis. It also suggests some future research directions which might extend and improve the results obtained in this thesis.

7.1 Summary

This thesis addresses two problems with current rendering systems. The way they deal with colour and the way reflectance properties of materials are represented.

To model the reflectance of a material, an approach is presented that uses a geometric model of the surface micro structure of the material. The micro model is used by a simulation system that creates a BRDF from this model. This is done using photon tracing techniques. Photon tracing shoots rays from the position of a virtual light source to a random position on the micro model. The ray intersects with a surface of the micro model and gets reflected into some direction, losing some energy. After possibly multiple reflections the ray does not intersect the model anymore. The direction in which the photon leaves the model is recorded. A large number of photons are shot from random light source positions. To allow for a fast convergence of the simulation, importance sampling is used.

Photon tracing results in an large collection of photons, all with different incoming directions, outgoing direction and a different energy. These photons are used to create a smooth reconstruction of the BRDF. This is done by associating an area of influence with each photon, on both the incoming and outgoing direction. The influence of a photon is normally distributed around its center to form a “splat”.

In order to make the evaluation of the reconstructed BRDF faster, the normal distribution is approximated by a polynomial. The polynomial has a finite support and is faster to evaluate. Furthermore, the integral of the polynomial is easy to compute, which allows for an easy normalization of the BRDF, which is necessary to ensure that energy is conserved.

In addition the finite support of the splats makes it possible to use a spatial subdivision of the domain of the BRDF, which amounts to storing a list of splats that intersect each cell of the subdivision. This results in a faster evaluation of the BRDF, which makes the splat representation usable in rendering systems.

Some sample BRDFs were created using this simulation system. The first one simulates anisotropic reflectance. A surface model of parallel cylinders was used to approximate the micro structure of brushed metal. The BRDF that is created shows the desired anisotropic effects, similar to those of brushed metal. The second example tries to simulate the reflectance of grass. A simple model of grass is created. The results of the simulation are then compared to results from a measurement of grass. Most of the effects obtained from the measurement can be reproduced by the simulations.

After describing a way to create the reflectance properties of materials, adequate representations for colour are examined. Shortcomings of the widely used RGB model are outlined. To assist the user in understanding the effects of different spectral power distributions that are represented by the same RGB triple, an editor for SPDs is presented. The editor allows a user to specify an SPD and a RGB colour. The SPD is then changed in a least squares minimal manner to map to the specified RGB colour.

As an alternative to the usual RGB model, a scene dependent colour representation is used. Scene dependent means that the actual colour space is created from the colour of the light emitted by light sources and the colour of the surfaces. In particular it computes all the spectral power distributions that might appear in the rendering process and uses a singular value decomposition to create an optimal basis for this set of vectors. This basis is orthogonal and is used as the colour space by the renderer.

The problem that there are too many SPDs that might appear during the rendering process is addressed by using the “most important”, namely those that are due to direct illumination. In addition the superposition property of linear light transport theory is applied to further reduce the number of SPDs. This principle allows us to render the image independently for each light source in the scene, creating the optimal colour space for each of the light sources. The individual images are then added together. The images created using this technique are visually indistinguishable from images rendered using 401 samples to represent a colour.

7.2 Future Work

The BRDF simulation system could be improved by adding stratification techniques to the Monte Carlo integration system. This might result in a faster convergence of the simulation. To improve the results even further, the photon tracing techniques could be combined with the eye-tracing based technique described in Section 3.2. The results of these combined approaches as used in image synthesis [51, 52] might be a starting point for future research.

In addition it might be interesting to use the current simulation system to compute the BRDFs of different materials. The use of transparency in the model might give better results for certain types of materials.

A major area of research is to find a better representation for the BRDF. As described in Section 4.4 spherical wavelets [40] provide an interesting new approach. An alternative approach would be to examine the splats more closely. They are similar to radial basis functions that have been used in approximation theory for a while. Results from this area could be applied to splats on the hemisphere and allow for a more compact representation of the bidirectional reflectance.

Appendix A

Spherical Geometry

This section describes some of the basic properties of spherical geometry. It also describes the terminology and symbols used in this theses.

A.1 Spherical Coordinates

In this thesis the angle ϕ describes the longitude of a position on the sphere, while θ is the angular distance from the north pole of the sphere (zenith). Every time when spherical coordinates are used in combination with a Cartesian coordinate system the north pole of the sphere ($\theta = 0$) lies on the z-axis. The 0-meridian is in the x-z plane.

To convert from spherical and Cartesian coordinates the following formulas can be used (assuming the radius of the sphere is 1):

$$x = \sin\theta\cos\phi$$
$$y = \sin\theta\sin\phi$$
$$z = \cos\theta$$

To convert from Cartesian coordinates to spherical coordinates, the transformations look like this:

$$\phi = \arctan\frac{y}{x}$$
$$\theta = \arctan z$$

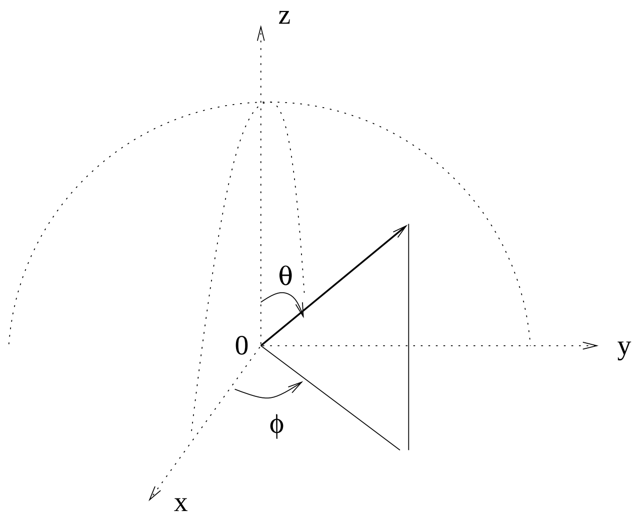


Figure A.1: Definition of the Spherical Coordinates.

A.2 The Spherical Triangle and Distance

The distance between two points $P_1 = (\theta_1, \phi_1)$, $P_2 = (\theta_2, \phi_2)$ on a unit sphere is measured along the great circle (a circle with a radius of the sphere) that goes through the two points). This distance d can be calculated using the following formula:

$$\cos_d = \cos\theta_1 \cos\theta_2 + \sin\theta_1 \sin\theta_2 \cos(\phi_1 - \phi_2) \tag{A.1}$$

The distance in radians is the arccosine of $\cos_d$. If the standard definition of the range of the arccos is used $[-1, 1] \rightarrow [0, \pi]$, $\arccos(\cos_d)$ is the shortest distance between P_1 and P_2 on the surface of the sphere.

The area of a spherical triangle $\triangle_s(P_1, P_2, P_3)$ on a unit sphere can be computed using the following formula:

$$A = \epsilon = \alpha + \beta + \gamma - \pi \tag{A.2}$$

For the unit sphere the area is equivalent to the excess ϵ of the triangle.

The angles can be computed from the length of the sides of the triangle a, b, c.

$$\cos\alpha = \frac{\cos a - \cos b \cos c}{\sin b \sin c} \tag{A.3}$$

$$\cos\beta = \frac{\cos b - \cos c \cos a}{\sin c \sin a} \tag{A.4}$$

$$\cos\gamma = \frac{\cos c - \cos a \cos b}{\sin a \sin b} \tag{A.5}$$

Appendix B

Grass BRDF Measurement

This Appendix contains a table of the grass reflectance measurement.

Light is the elevation angle in degrees measured from the ground (parallel to ground is 0 degrees). *Meter* is the elevation angle of the photometer. *Angle* is the azimuth angle of the meter. $angle = 0$ means that the meter is behind the light source, at $angle = 90$ the meter is perpendicular to the illumination direction, and at $angle = 180$ the meter is on the opposite side of the light source.

In all measurements the distance of the light source and the radiometer from the model was constant 10 feet.

At every set of angles 5 measurements where taken: One measuring the center of the piece of grass, one measuring to the right side, one to the left side, one on the far side and one on the near side. All this is relative to the position of the radiometer.

Note that the measured intensities have to be corrected for the incoming flux. With the light source positioned at a low angle, the flux hitting the sample is lower than at a higher angle. To compensate this effect the intensities have to be multiplied by the sine of the azimuth (or the cosine of the zenith) of the light source.

Light	Meter	Angle	Center	Left	Right	Far	Near
45	30	0	1.84	1.87	1.98	2.09	1.63
45	30	45	1.45	1.61	1.30	1.42	1.55
45	30	90	1.30	1.15	1.04	1.37	1.21
45	30	135	1.22	1.19	1.06	1.21	1.13
45	30	180	1.21	1.20	1.17	1.10	1.33

Light	Meter	Angle	Center	Left	Right	Far	Near
45	20	0	2.00	1.89	2.08	2.06	1.75
45	20	45	1.69	1.83	1.54	1.60	1.69
45	20	90	1.47	1.43	1.29	1.43	1.52
45	20	135	1.51	1.57	1.42	1.50	1.51
45	20	180	1.59	1.48	1.51	1.48	1.62
45	10	0	2.13	2.14	2.14		
45	10	45	1.91	1.82	1.79		
45	10	90	1.65	1.80	1.30		
45	10	135	1.91	1.83	1.80		
45	10	180	2.00	2.02	1.96		
30	30	0	1.94	2.01	1.84	2.02	1.82
30	30	45	1.31	1.39	1.16	1.14	1.38
30	30	90	1.19	1.09	1.06	1.14	1.27
30	30	135	1.16	1.00	1.34	1.28	1.12
30	30	180	1.31	1.18	1.43	1.25	1.15
30	20	0	2.10	2.33	2.02	2.19	1.89
30	20	45	1.53	1.71	1.41	1.37	1.59
30	20	90	1.32	1.19	1.31	1.20	1.42
30	20	135	1.46	1.25	1.54	1.49	1.36
30	20	180	1.74	1.56	1.78	1.70	1.61
30	10	0	2.29	2.40	2.21		
30	10	45	1.85	1.83	1.75		
30	10	90	1.80	1.67	1.66		
30	10	135	2.10	1.91	2.20		
30	10	180	2.70	2.50	3.00		
20	30	0	1.19	1.16	1.10	1.00	1.14
20	30	45	0.85	0.85	0.78	0.75	0.97
20	30	90	0.73	0.58	0.69	0.73	0.73
20	30	135	0.81	0.52	0.92	0.94	0.67
20	30	180	0.91	0.88	0.98	1.08	0.77
20	20	0	1.56	1.63	1.42	1.47	1.56
20	20	45	1.01	0.96	0.93	0.82	1.17
20	20	90	0.86	0.74	0.87	0.81	0.99
20	20	135	1.14	0.93	1.21	1.24	0.96
20	20	180	1.44	1.19	1.40	1.59	1.02

Light	Meter	Angle	Center	Left	Right	Far	Near
20	10	0	1.66	1.70	1.58		
20	10	45	1.19	1.17	1.20		
20	10	90	1.02	1.06	1.20		
20	10	135	1.63	1.44	1.80		
20	10	180	2.20	2.30	2.20		

Appendix C

Implementation - Photon Tracing

This section describes the implementation of the reflectance simulation system and the data structures used to store the generated BRDF. Since the simulation uses ray-tracing techniques similar to those used in image syntheses ([9]) the Vision rendering architecture [49, 47] was used as a framework for the implementation.

The data-structures used to represent the BRDF were implemented as part of the GP project [33, 2] and are called the `GPBRDF` library.

The sample images where created by using the Vision system again. The system was extended by including the GP classes and thus having access to the sampled BRDFs. In order to provide a simple way to use these BRDFs the RenderMan Shading Language [35, 50], that is supported by Vision was extended.

C.1 Overview of the Implementation

Figure C.1 provides an overview of the implementation. In order to get from a model of the surface, to an image which has this surface-type applied to some of its objects, the following components are involved:

- The first step is to use the geometric model of the micro structure of the material and run it through the Monte-Carlo simulation. This simulation is implemented within the Vision frame work. (see Section C.5)

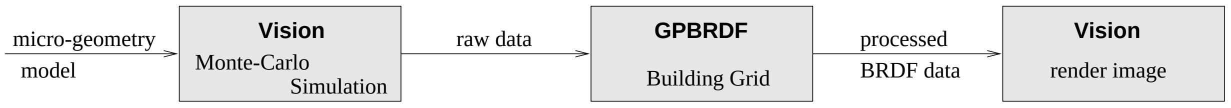


Figure C.1: Components of the implementation.

- The resulting data is then used and processed by small programs built on top of the GPBRDF library. (see Section C.6)
- In order to be able to apply the BRDF to a surface the Vision system has been extended to be able to use these sampled BRDFs. This is done by using the GPBRDF library. (see Section C.6)

Additionally tools to create the micro structure for grass and aluminum are available, as well as tools to analyze the BRDF.

C.2 Usage of the Simulation

The BRDF simulation system is included into the Vision architecture. It can be enabled by specifying the `BRDFLight` option in the RenderMan interface:

```
Option "BRDFLight" "texturename" "grass.pm"
                   "rays" 100000
```

The `texturename` token specifies the name of the created splat file and the number after `rays` sets the number of traced paths. The micro geometry can be specified using standard RenderMan modeling operations.

The sampling hemisphere has its pole on the positive z-axis. Thus, the base plane of the micro geometry has to be in the x-y plane. The size of the hemisphere is 1, and the default sampling area is a 0.1×0.1 square centered at the origin. The sampling area can be changed using `"area" width`.

When `BRDFLight` is enabled, the renderer will produce a *photon map* instead of an image. A photon map is an ASCII file with one line for each photon. The first line contains the number of photons in the file. Each line consists of the following information:

```
theta_in phi_in theta_out phi_out probability colour1 colour2 colour3
```

`theta_in` and `theta_out` are the position of the light source for the particular photon, and `theta_out` and `phi_out` are the outgoing direction of the photon. `probability` is the probability of the path taken by the photon. The following numbers specify the energy transfered by the computed path. If RGB is used as a colour model, `colour1` is the red, `colour2` is the green, and `colour3` is the green part. If a different colour space with a different number of basis functions is used, there might be a different number of colour values.

C.3 Tools

This section presents the tools used to optimize and analyze a BRDF. In addition to optimization, tools are provided to compute cross section and angular plots of a BRDF, as well as tools that optimize a BRDF.

C.3.1 GPBRDFSectSlow

`GPBRDFSectSlow` reads in a photon map and computes a cross section of the BRDF. The syntax to run the program is the following:

```
GPBRDFSectSlow <spread> <energy_min> <nr_samples> <input_pm>
```

`spread` is the radius of the photons. `energy_min` specifies the minimum energy a photon should have to be taken into account. This option allows to speed up the computation of a cross section of the BRDF. The problem is of course that the threshold has to be low enough to leave all the important photons within the computation. `nr_samples` specifies the number of data points in the cross section, and `input_pm` is the file containing the photon map.

C.3.2 GPBRDFOpt

As described in Section 4.3.1 a subdivision of the hemisphere is used to improve the evaluation speed of the BRDF. This optimization is done by `GPBRDFOpt`. It takes a photon map, creates the subdivision and writes it to disk.

```
GPBRDFOpt <rings> <rects> <spread> <energin_min> <input.pm> <output.grd>
```

`rings` is the number of rings on the created output subdivision. `rects` is the maximum number of cells in a ring. Both the number of rings and the maximum number of rectangles in a ring is the same on the input and output hemisphere. `spread` and `energy_min` are the same as in `GPBRDFSectSlow`, `input.pm` specifies the input photon map and `output.grd` specifies the name of the generated optimized BRDF.

C.3.3 GPIBRDFOpt

`GPIBRDFOpt` opt is similar to `GPBRDFOpt` except that it works for isotropic BRDFs (see Section 4.3.3). For isotropic BRDFs it is assumed the `phi_in` of the incoming direction is always 0.

```
GPIBRDFOpt <rings> <rects> <spread> <energin_min> <input.pm> <output.grd>
```

The options are similar to `GPBRDFOpt`, except that the number of rectangles does not apply for the incoming direction.

C.3.4 GPBRDFSection

`GPBRDFSection` is similar to `GPBRDFSectSlow` except that it takes a `.grd` file (a file created by `GPBRDFOpt`) instead of a photon map.

```
GPBRDFSection <nr_samples> <grid.grd>
```

Since most parameters are already specified by running `GPBRDFOpt` only the number of samples and the file name have to be specified.

For all files that create cross sections of the BRDF the incoming direction and outgoing direction are specified in the source file. This is due to the large number of possibilities to cut a section trough a BRDF, which would have required a lot of parameters (section along θ_{in}, section along ϕ_{out}, ...).

C.3.5 GPBRDFMap

`GPBRDFMap` creates an angular plot of the BRDF. The user specifies θ_{in} and ϕ_{in}, as well as a scale factor. The angular plot is saved as a tiff file.

```
GPBRDFMap <size> <energy_scale> <spread> <theta_light> <phi_light>
          <input.grd> <output.tif>
```

C.4 Using the BRDF

In order to be able to use the BRDF in a rendering system, it has to be applied to a surface. In the Vision rendering system the surface properties are described by a *surface shader*. Surface shaders are procedural and written in the Pixar Shading Language [35, 50].

The shading language has been extended to provide access to BRDFs. This is done by the `brdf()` command:

```
point N, O, I, D;
color c = brdf("name.grd"; N, O, I, D);
```

The name of the `.grd` file is specified as well as the current surface normal `N`, the outgoing direction `O`, the incoming direction `I`. The `D` is a direction on the surface that specifies the direction of anisotropy. The access of the BRDF data is implemented using the `GPBRDF` library.

C.5 Monte-Carlo Simulation

The Monte Carlo simulation is implemented as part of the Vision rendering system [49, 47]. The actual tracing of photons is performed in the `VisBRDFLighting` class. It is performed as part of the `prepareIllumination` method. The actual simulation works as follows:

1. A random position for the light source is chosen. If an anisotropic BRDF is sampled, this position can be anywhere on the hemisphere. For an isotropic BRDF the longitude is 0.

2. Choose a random position on the sampling area. Shoot a ray from the light source position towards this point (direction `D`).

3. Shoot a ray with direction `D` and compute the closest intersection.

4. At the intersection point, get the micro BRDF and compute a random direction `E` on the hemisphere around the intersection point, distributed according to the micro BRDF. This is done using the technique described in [48].

5. Set `D = E` and return to step (2) until no intersection point can be found.

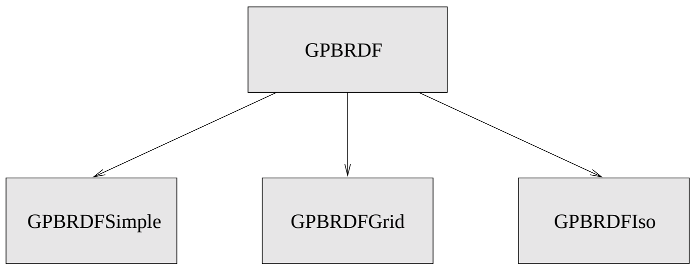


Figure C.2: Class hierarchy for representing BRDFs.

6. If a scattered ray does not intersect the micro model, it intersects the intersection of this ray with the sampling hemisphere is computed. This intersection point is the outgoing direction of the photon.

All the created photons are then stored in an ASCII file, the photon map as described in C.2.

C.6 Storing and Representing the BRDF

The implementation of the classes representing the splats and BRDFs is implemented as part of the GP-Project [2].

The core of this part of the implementation is the `GPBRDF` object. `GPBRDF` is the base class for implementation of BRDFs. It provides an abstraction for a BRDF. It contains a pure virtual member `getReflectance()` that takes the incoming and outgoing direction as parameters and returns the value of the BRDF.

Derived from `GPBRDF`, there are a number of actual implementations (see Figure C.2). `GPBRDFSimple` is a simple implementation of a BRDF. It only acts as a collector for `GPSplat4D` objects.

A `GPSplat4D` object is a representation of a single photon. It contains the incoming and outgoing direction, its probability and its energy (colour). The most important method of `GPSplat4D` is `getRadiance()`. The method `getRadiance()` takes an incoming and outgoing direction as parameters and returns the contribution of the splat at this location. Thus the splat object defines the drop off function of a photon. In this thesis the only drop off function being used is the polynomial approximation of a gauss splat, thus there is no other splat type. If a different type of splat should be used, a class can be derived from `GPSplat4D`.

One other important method is provided by `GPSplat4D`, `configure()`. This method takes the radius of the splat and the total number of splats in the BRDF as parameters, and normalizes the splat. This function is usually called by a `GPBRDF` object, before the BRDF is actually used. The normalization has to be done only once.

`GPBRDFSimple` implements a simple array of splat objects. If `getReflectance()` is called, it simply calls `getRadiance()` for every splat and sums up all the radiance values of the splats. This is rather slow if the number of splats is large.

`GPBRDFGrid` implements the subdivision scheme described in Section 4.3.1. For each cell it test which splats intersect the cell and stores a list of pointers to all intersecting splats. If `getReflectance()` is called, the cell for the incoming and outgoing direction is determined, `getRadiance()` for all intersecting splats is called and the returned values are summed up.

`GPBRDFIso` is a simpler version `GPBRDFGrid`, that works with isotropic BRDFs only. Although the BRDF is isotropic, it uses the same splats, but the longitude of the incoming direction is always 0.

Both `GPBRDFGrid` and `GPBRDFOpt` provide a `write()` method that saves the optimized BRDF to a file. A corresponding `read()` reads it back form disk.

Appendix D

Implementation - Reflectance Image Viewer

This appendix describes the usage and the implementation of the reflectance image viewer, which includes the spectral power distribution editor described in Section 6.2.

D.1 Program Usage

This section describes how to use the reflectance image viewer.

D.1.1 Main Window

After the program is started by typing

```
  riv
```

a window containing a menu bar pops up. The `File` menu contains the following items:

- "`Load Image`" Loads an reflectance image stored in the extended tiff format.
- "`Save Basis Image`" Saves the image in the extended tiff format.
- "`Save RGB`" Save the currently displayed RGB image in tiff format.
- "`Edit Light Source`" Opens a window containing the light source editor (see Section D.1.2).

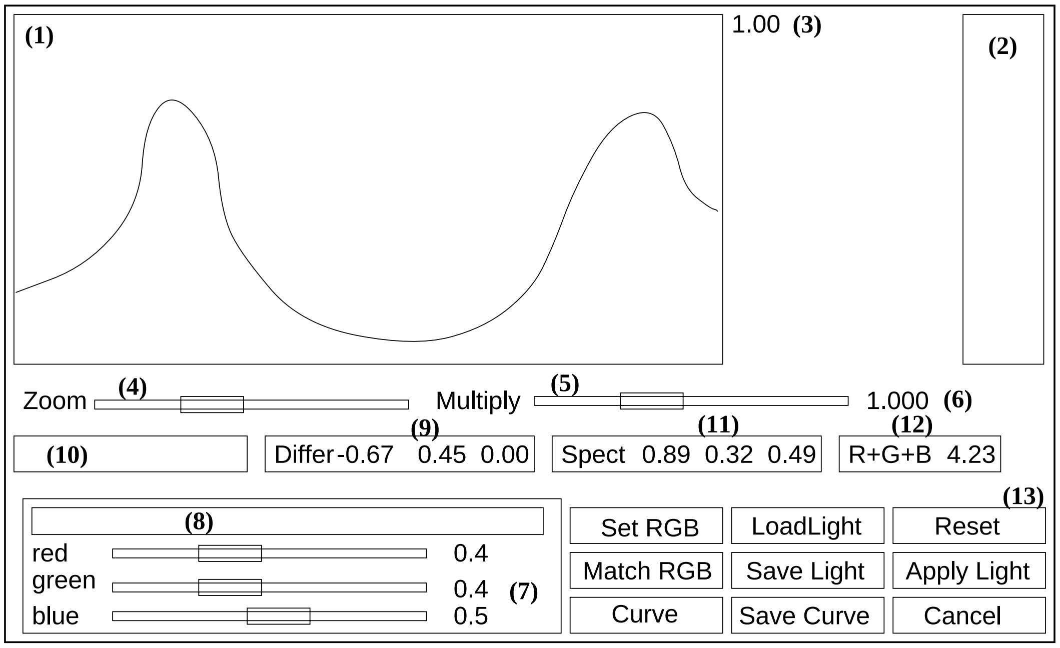


Figure D.1: Overview of the light source editor

.

The `RGB Conversion` menu specifies the colour conversion matrix that is used to convert from spectral to RGB. The default is `Box` which means that the RGB basis functions are three non overlapping, box functions of the same width. `CIE` uses the CIE RGB basis functions.

After the user loads an image, the image is displayed in this window. The window can be resized, in which case the image is scaled to match the new dimensions of the window. The window always displays the entire image.

D.1.2 Light Source Editor

After the user chooses "`Edit Light Source`" from the menu the light source editor pops up. Image D.1 shows a sketch of the editor. The numbers "(x)" are not part of the actual editor. Instead they are used in the following to explain the task of the individual elements of the editor.

Editing an SPD

Area **(1)** (see Figure 6.1) contains the spline based SPD editor. The window displays the power distribution in terms of the wavelength. The left most column represents 400nm and the right most column 700 nm. An SPD can be modified by adding, deleting and moving control points.

The control points are identified by small squares. These squares can be selected using the left mouse button and moved, while the button is pressed. The curve changes its shape immediately as the mouse is moved. The spline cure is interpolating, i.e. it passes through the control point. It is not possible to move control points passed the lower edge of the editor.

It is not possible to move a control point passed any other control point. Thus, if a control point p is right of p_l and left of p_r it is not possible to move it left of p_l or right of p_r. The left and the right most control point can only be moved in y-direction.

New control points can be added using the middle mouse button. The control point is added at the current x-position of the mouse. The y coordinate is determined by the value of the curve at this position. It is also possible to remove a control point using the right mouse button.

Since the curve represents the SPD of a light source, the colour represented by the SPD is shown in area **(2)** (see Figure D.1). The colour is normalized in a way, that the maximum component of the resulting RGB-vector is 1.0. This normalization is necessary in order to be able to display the colour of light sources which would otherwise map the RGB-values larger then 1.0. The RGB vector of the colour displayed in **(2)** is shown in **(11)** and the total power (the sum of the RGB components) is displayed in **(12)**.

Initially, the maximum value which can displayed in the SPD-spline editor is 1.0. This can be changed using the `Zoom` slider **(4)**. This slider changes the maximum value (the power represented by the top most row of the SPD-editor), which is displayed by the widget **(3)**. A logarithmic scale is used to map the slider position to a value. If the slider is on its leftmost position, it represents a zoom factor of 1.0.

The `Multiply` slider **(5)** can be used to scale the current SPD. The curve is multiplied by the value displayed by **(6)**, which is calculated form the position of the slider using a logarithmic scale. If the slider is in the middle, it represents a multiplication factor of 1.0.

The RGB-Editor

The RGB-editor **(8)** can be used to edit a tristimulus colour in terms of its RGB values. The value of any of the three slider is displayed as values ranging from 0.0 to 1.0. The colour represented by the current position of the sliders is displayed as well.

The RGB-editor has no effect on the SPD. It is just used as an reference colour which should be match by a SPD.

Matching of Colours

The editor provides support for matching a spectral power distribution to a RGB-colour. After the user specified an SPD and the colour it is supposed to be matched to, the **Match** button runs the constraint least squared matching algorithm.

The Buttons

The editor contains 8 buttons divided into three groups **(13)**. The first groups deals with the general operations:

Apply Light This button takes the current SPD and applies it to the image displayed by the main window. Thus it calculates the reflected light of the image hit by the light source. The resulting image is displayed in the main main window.

Reset Resets the light source editor to its initial state.

Cancel Closes the light source editor.

The second group of buttons allows to load and save light sources:

Load Light This is used to load a light source from disc. The light source can be either represented as a number of samples or as coefficients of the Catmull-Rom spline. The format is determined automatically. If a light source defined in terms of point samples is loaded, it can not be edited using the spline editor.

Save Light Saves the current SPD as a sequence of 300 point samples (ranging from 400 to 700nm at 1nm increments).

Save Curve Saves the current SPD as Catmull-Rom control points.

The third group is used to automatically match the SPD to a RGB value and to get the editor back into curve editing mode:

Match RGB Modifies the current SPD to match the colour currently specified by the RGB editor.

After automatic matching the curve can not be edited using the spline editor any more. Clicking the `Curve` button undoes the matching.

Curve If the current SPD is represented as a number of samples (either by using the "`Match RGB`" button or by loading a light source) this button gets the editor back into curve editing mode. The last spline curve is displayed in **(1)**.

D.2 Implementation

This section gives a brief overview of the implementation of the project. An outline can be seen in Figure D.2.

D.2.1 Overview

The viewer is divided into the following modules.

- `main.cc` Contains all the user interface implemented using Motif.
- `GPLightEdit.cc` Contains the Catmull-Rom spline editor, as well as the functions converting from spectral representation to RGB. The matrix converting from Basis to RGB-representation (which is depending on the light source) is computed in this module too. It also contains the routines for loading and saving light sources.
- `GPLoadFile.cc` performs the loading and saving of images. The actual conversion form Basis to RGB space is performed here as well.
- `GPData.cc` Contains data for the XYZ tristimulus basis functions as well as other conversion matrices.

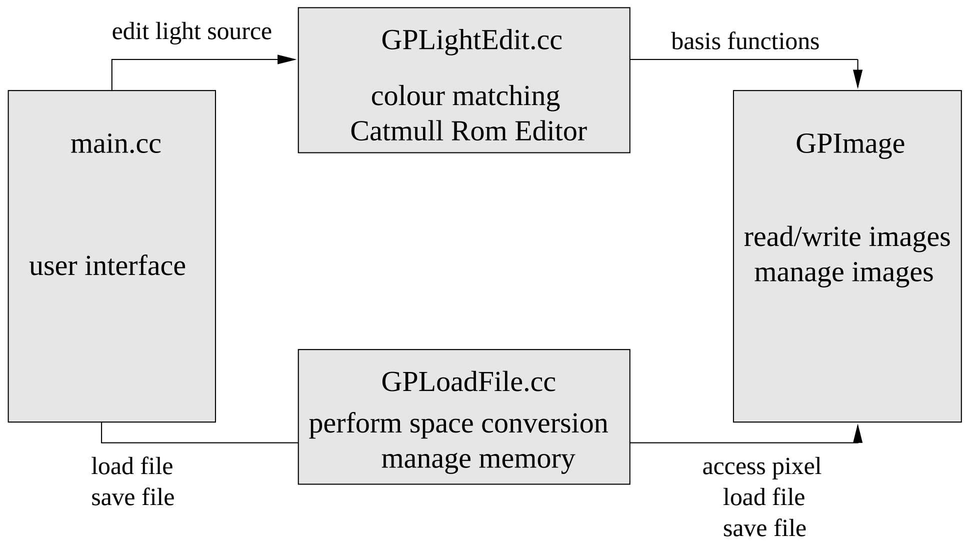


Figure D.2: An overview of the implementation of the viewer.

- `GPImage` This library is used to perform the actual loading and saving of images. In addition it provides an interface for accessing the pixel information stored in an image. This interface is independent on the structure and the "physical" data type of the underlying image data.

An overview of the data and control flow between the individual modules can be seen in Figure D.2 as well.

D.2.2 Colour Matching

The colour matching is performed in `GPLightEdit::matchRGB`. This method calls a fortran function taken from [21].

D.2.3 Light Source File Format

Light sources are stored as either a number of uniform point samples or as Catmull-Rom spline coefficients. In both cases a simple ASCII format is used. Short examples of both formats can be seen in Figure D.3.

```
# fluorescent, point samples
cspec 400 700 1
30.000000
31.000000
32.000000
33.000000
34.000000
35.000000
50.000000
65.000000
80.000000
...
```

```
# riv illumination Catmull Rom parameters
curve 6 400 700
0.000000 0.500000
0.000000 0.500000
0.203333 0.916667
0.713333 0.100000
1.000000 0.500000
1.000000 0.500000
```

Figure D.3: Examples of light source definition files.

Point Samples Comments are identified by a line beginning with "#". The actual definition of a point sampled light source starts with the token "cspec" followed by the wavelength of the first and the last sample. The last number on this line identifies the wavelength increment of the samples.

Catmull-Rom Control Points Comments are identified by a line beginning with "#". The actual definition of a SPD starts with the token "curve" followed by the number of control points followed by the wavelength of the first and the last sample. Each control point is defined by its t-parameter and the value of the curve at this position.

Bibliography

[1] Tony Apodaka and Darwyn Peachey. Writing renderman shaders. *SIGGRAPH '92 Course Notes 21*, 1992.

[2] Richard Bartels. The GP project. Technical report, University of Waterloo, July 1996.

[3] Petr Beckmann and Andre Spizzichino. *The Scattering of Electromagnetic Waves from Rough Surfaces.* MacMillian, 1963.

[4] James F. Blinn. Models of light reflection for computer synthesized pictures. *Computer Graphics (SIGGRAPH '77 Proceedings)*, 11(2):192–198, July 1977.

[5] S. Chandrasekar. *Radiative Transfer.* Oxford University Press, 1950.

[6] B. W. Char, K. O. Geddes, G.H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt. Maple V *First Leaves: A Tutorial Introduction to* Maple V. Waterloo Maple Publishing, 1992.

[7] Michael Cohen and D. P. Greenberg. The hemi-cube: A radiosity solution for complex environments. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):31–40, August 1985.

[8] Michael F. Cohen and John R Wallace. *Radiosity and Realistic Image Synthesis.* Academic Press, 1993.

[9] Robert L. Cook, Tom Porter, and Loren Carpenter. Distributed ray tracing. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3):137–145, July 1984.

[10] Robert L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1:7–24, January 1982.

[11] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *Computer Graphics (SIGGRAPH '81 Proceedings)*, 15(3):301–316, July 1981.

[12] William B. Cowan and Colin Ware. Colour perception tutorial. *SIGGRAPH '83 Course Notes*, 1983.

[13] James L. Dannemiller. Spectral reflectance of natural objects: how many basis functions are necessary? *JOSA A*, 9(4):507–515, April 1992.

[14] Aldus Developers Desk. *TIFF – Revision 6.0.* Aldus Corporation, June 1992.

[15] J. D. Foley, Andries van Dam, S. K. Feiner, and J. F. Huges. *Computer Graphics.* Addison-Wesley, 1990.

[16] Andrew Glassner. A model for fluorescence and phosphorescence. In *Fifth EUROGRAPHICS Workshop on Rendering*, pages 57–68, Darmstadt, June 1994.

[17] Andrew S. Glassner. *Principles of Digital Image Synthesis*, volume 1. Morgan Kaufmann Publishers, 1995.

[18] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore Maryland 21218, 1983.

[19] Jay S. Gondek, Gary W. Meyer, and Jonathan G. Newman. Wavelength dependent reflectance functions. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28:213–220, August 1994.

[20] Roy Hall. *Illumination and Color in Computer Generated Imagery.* Monographs in Visual Computing. Springer-Verlag, 1988.

[21] R. J. Hanson and K. H. Haskell. Algorithm 587. *ACM Transaction on Mathematical Software*, 8(3):323, September 1982.

[22] J. R. Howell and M. Perlmutter. Monte carlo solution of thermal transfer through radiant media between gray walls. *Journal of Heat Transfer*, pages 116–122, February 1964.

[23] Richard S. Hunter and Richard W. Harold. *The Measurement of Appearance, Second Edition.* John Wiley & Sons, New York, 1987.

[24] Henrik Wann Jensen. Importance driven path-tracing using the photon map. In P.M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*, pages 326–335, Toronto, Ontario, 1995.

[25] Henrik Wann Jensen. Rendering caustics on non-lambertian surfaces. In *Proceedings of Graphics Interface '96*, pages 116–121, Toronto, Ontario, May 1996.

[26] James T. Kajiya. The rendering equation. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):143–150, August 1986.

[27] Konrad F Karner, Heinz Mayer, and Michael Gervautz. An image based measurement system for anisotropic reflection. *Computer Graphics Forum (EUROGRAPHICS '96 Proceedings)*, 15(3):119–128, August 1996.

[28] Gavin S. P. Miller. From wire frames to furry animals. In *Proceedings of Graphics Interface '88*, pages 138–145, Edomonton, Ontario, June 1988.

[29] K. D. Möller. *Optics.* University Science Books, 1988.

[30] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide.* Addison Wesley, 1993.

[31] Jeffry S. Nimeroff, Eero Simoncelli, and Julie Dorsey. Efficient re-rendering of naturally illuminated environments. *Proceedings of Fifth Eurographics Workshowp on Rendering*, pages 359–373, June 1994.

[32] Mark S. Peercy. Linear color representation for full spectral rendering. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 191–198, August 1993.

[33] Thomas H. Pflaum. The GPImage Library. Technical report, University of Waterloo, July 1996.

[34] Bui-Tuong Phong. Illumination for computer generated pictures. *CACM*, 18(3):311–317, July 1975.

[35] Pixar. *The RenderMan Interface, Version 3.1.* Pixar, San Rafael, CA, September 1989.

[36] Pierre Poulin and Alain Fournier. A model for anisotropic reflection. *Computer Graphics (SIGGRAPH '90 Proceedings)*, pages 273–282, August 1990.

[37] H. Samet. Implementing ray tracing with octrees and neighbor finding. In *Computer Graphics*, volume 23(4), pages 445–460, August 1989.

[38] Christophe Schlick. Customizable reflectance models for everyday rendering. In *Fourth EUROGRAPHICS Workshop on Rendering*, pages 73–83, Paris, June 1993.

[39] Christophe Schlick. A survey of shading and reflectance models. *Computer Graphics Forum*, 13(2):121–131, June 1994.

[40] Peter Schröder and Michael Sweldon. Spherical wavelets. *Computer Graphics (SIGGRAPH '95 Proceedings)*, 29(2):249–252, August 1995.

[41] Peter Shirley. *Physically Based Lighting Calculations for Computer Graphics.* PhD thesis, Dept. of Computer Science, U. of Illinois, Urbana-Champaign, November 1990.

[42] Peter Shirley. A ray tracing algorithm for global illumination. In *Graphics Interface '90*, pages 205–212, May 1990.

[43] Peter Shirley. Monte carlo simulation and integration. In Paul Heckbert, editor, *Global Illumination (SIGGRAPH '93 Course Notes 42)*, pages 9.1–9.23, 1993.

[44] Peter Shirley. Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, January 1996.

[45] François Sillion and Claude Puech. *Radiosity & Global Illumination.* Morgan Kaufmann, 1994.

[46] François X. Sillion, James R. Arvo, Stephen H. Westin, and Donald P. Greenberg. A global illumination solution for general reflectance distributions. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):187–196, July 1991.

[47] Philipp Slusallek. *Vision – An Architecture for Physically Based Rendering.* PhD thesis, University of Erlangen, IMMD IX, Computer Graphics Group, April 1995.

[48] Philipp Slusallek, Thomas Pflaum, and Hans-Peter Seidel. Using procedural RenderMan shaders for global illumination. In *Computer Graphics Forum (Proceedings EUROGRAPHICS '95)*, 1995. accepted for publication.

[49] Philipp Slusallek and Hans-Peter Seidel. Vision: An architecture for global illumination calculations. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):77–96, March 1995.

[50] Steve Upstill. *The RenderMan Companion.* Addison Wesley, 1990.

[51] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Fifth EUROGRAPHICS Workshop on Rendering*, pages 147–162, Darmstadt, June 1994.

[52] Erik Veach and Leonidas J. Guibas. Optimally combining techniques for monte carlo rendering. *Computer Graphics (SIGGRAPH '95 Proceedings)*, 29:419–428, August 1995.

[53] Gregory J. Ward. Adaptive shadow testing for ray tracing. In *Second Eurographics Workshop on Rendering (Part. Edition)*, May 1991.

[54] Gregory J. Ward. Measuring and modelling anisotropic reflection. *Computer Graphics (SIGGRAPH 92 Proceedings)*, pages 265–272, July 1992.

[55] Gregory J. Ward. The RADIANCE lighting simulation and rendering system. *Computer Graphics (SIGGRAPH 94 Proceedings)*, pages 459–472, July 1994.

[56] Gregory J. Ward and Paul S. Heckbert. Irradiance gradients. In Alan Chalmers and Derek Paddon, editors, *Third EUROGRAPHICS Workshop on Rendering*, pages 85–98, Bristol, May 1992.

[57] Gregory J. Ward and Francis Rubinstein. A ray tracing solution for diffuse interreflection. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):85–92, August 1988.

[58] H. Westin, James R. Arvo, and Kenneth E. Torrance. Predicting reflectance functions from complex surfaces. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(4):255–264, August 1992.

[59] B. Wyvill, C. McPheeters, and G. Wyvill. Animating soft objects. *The Visual Computer*, 2:227–234, 1986.

[60] Daniel Zwillinger. *Handbook of Integration.* Jones and Bartlett Publishers International, 1992.