Contents

Chapter 1

Introduction (draft)

Subsurface scattering (SS) is a physical phenomenon that naturally occurs in a wide range of natural materials. Some of the materials that exhibit a strong SS effect in everyday life are milk, human skin and marble. Subsurface scattering is that phenomenon that occurs when light is partially absorbed by an object, bounces inside ("scatters") and finally exits the surface on another point of the material (see Figure 1.1). The phenomenon that results is generally known as *translucency*. We can see some examples of translucency in Figure 1.2

Since the beginning of computer graphics, various attempts have been performed in order to physically model subsurface scattering. Some of these models involve Monte Carlo simulations of the light entering the medium [Pharr and Hanrahan, 2000], other focus on approximating the diffusion of light within the material using an analytical approach [Jensen et al., 2001].

The first model that proposed and analytical approach was the one by Jensen et al. [2001], as an approximation of the radiative transfer equation. This approximation has then been exploited by different authors, in order to account for multi-layered materials [Donner and Jensen, 2005], heterogeneous materials [Wang et al., 2010] and thin surfaces[Wang et al., 2010]. A recent analytical approximation, proposed by Frisvad et al. [2013], extends the approximation in order to account for the directionality of the incoming light.

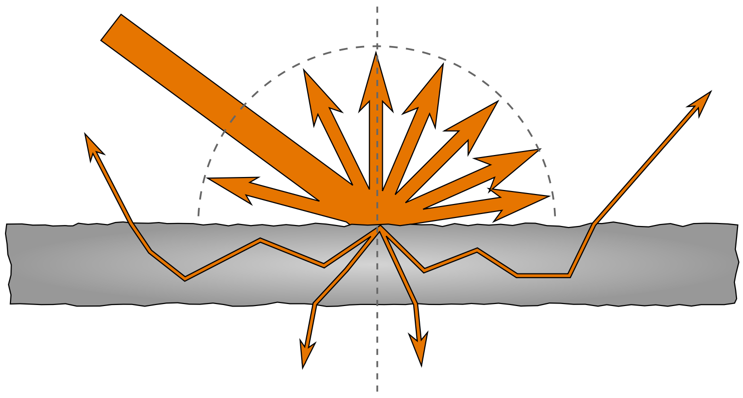


Figure 1.1: Diagram of subsurface scattering. Most of the incoming light gets reflected, but some of it enters the material and leaves it at a different point.

In recent years, with the advent of programmable graphics cards (GPU), it has become possible to exploit these algorithms and bring them to interactive frame rates, and in some cases even to real time rendering. Jensen and Buhler [2002] were the first to propose an efficient implementation (though not real time and on CPU) for rendering subsurface scattering using an octree. More recently, several methods have been proposed, including image-based splats, sum-of-Gaussians filtering, and grid-propagation based methods.

In this thesis we want to employ some cutting edge GPU techniques, with the aid of the programmable pipeline, to implement Frisvad et al. directional model in a real-time fashion. This method should achieve real time results (i.e. in the range of 30 to 60 frames per second) for a wide range of natural materials.

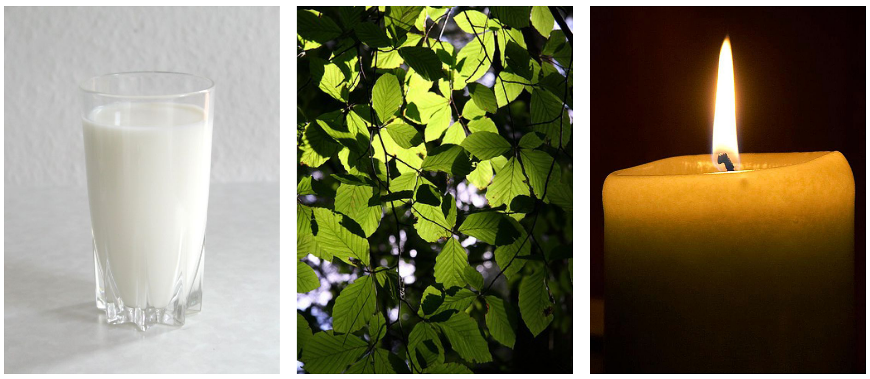


Figure 1.2: Some examples of translucent materials: milk, leaves and a candle. Images courtesy of Wikimedia Commons.

Chapter 2

Related Work

In rendering of subsurface scattering, all approaches rely on approximating correctly the *Radiative Transport Equation* (RTE). We identified two main approaches to the problem in literature:

Analytical One class of solutions consists of approximating the RTE or one of its approximations via an analytical model. These model can have different level of complexity and computation times, and are often adaptable to a wide range of materials. However, often they rely on assumptions on the scattering parameters that limit their applicability.

Numerical In this other class of solutions, a numerical solution for the RTE is actually computed. While providing an exact solution, the computation times are longer. When interactivity is needed, generally some heavy pre computation must be used.

2.1 Analytical techniques

In the analytical techniques, two different areas of research must be distinguished. The first area is the research on the actual models, while the second is research on how the actual models can be implemented efficiently. Each model is usually represented by a specific function called BSSRDF (*Bidirectional Subsurface Scattering Reflectance Distribution Function*), that describes how light propagates between two points on the surface. This function in the general case must be calculated between all the couple of points on the surface and then integrated over each point. Implementation techniques focus on efficiently implementing this integration step, often making assumptions for which points the computation can be avoided.

2.1.1 Models

Regarding the models, the first and most important is the dipole developed by Jensen et al. [2001]. The models relies on an approximation of the RTE called the *diffusion approximation*, that relies on the assumption on highly scattering materials. In this case, a BSSRDF for a planar surface in a semi-infinite medium can be obtained. The BSSRDF needs only the distance between two points to be calculated, and with some precautions can be also extended to arbitrary geometry. This model does not include any single scattering term, that needs to be evaluated separately. The model was then further extended in order to account for multi-layered materials[Donner and Jensen, 2005].

A significant improvement on the model was later given by D'Eon [2012], that improved the model to better fit path traced simulations without any extra computation cost. A more advanced model based on quantization was proposed by D'Eon and Irving [2011], that introduced a new physical foundation in order to improve the accuracy of the original diffusion approximation. Finally, some higher order approximation exist [Frisvad et al., 2013], in order to account for the directionality of the incoming light and single scattering. This allows a more faithful representation of the model at the price of extended computation times.

Finally, for real-time critical applications (such as games), translucency is often estimated as a function of the thickness of the material, that is used to modify a lambertian term [Tomaszewska and Stefanowski, 2012]. While not physically accurate, this technique allows to have a fast translucency effect that can be easily added to existing deferred pipelines.

2.1.2 Implementations

Most research on efficient implementations of a subsurface scattering analytical model has been made on the original model by Jensen et al. [2001]. The first efficient implementation was proposed by Jensen and Buhler [2002], based on a two-pass hierarchical integration approach. Samples on the model are organized in an octree data structure, that then is used to render the object. In the first step, the radiance from the light is stored in the points. In the second pass, using the octree, the contribution from neighboring points is computed, clustering far points in order to speed up calculations. In the original paper, the single scattering term is approximated with as a simple BRDF approximation.

Lensch et al. [2002] approached the problem by subdividing the subsurface scattering contribution into two: a direct illumination part and a global illumination part (i.e. the light shining through the object). The global illumination part is pre-computed as vertex-to-vertex throughput and then summed to the direct illumination term in real-time. Translucent shadow maps [Dachsbacher and Stamminger, 2003] use an approach similar to standard shadow maps: they render the scene from the light point of view, and then calculate the dipole contribution in one point only from a selected set of points, according to a specified sampling pattern. As in Lensch et al. [2002], the contribution is split into global and local to permit faster computations. Mertens et al. [2003b] propose a fast technique based on radiosity hierarchical integration techniques, that unlike the previous implementation can handle deformable geometry.

Another important category of methods is screen space methods. Mertens et al. [2003a] propose an image space GPU technique that pre-computes a set of sample points for the area integration and then performs the integral over multiple GPU passes. d'Eon et al. [2007] proposes a method in image-space, interpreting subsurface scattering as a sum of images to which a gaussian filter has been applied. The gaussians are then summed with weights that make them fit the diffusion approximation. Jimenez et al. [2009] improves further the technique, giving more precise results in case of skin. Shah et al. [2009] present a fast technique that render the object as a series of splats, using GPU blending to sum over the various contributions.

Regarding more advanced models, the better and the quantized dipole can be applied to any of the previous implementations, since they do not require additional information that the standard dipole. On the other hand, the directional dipole requires the direction of the incoming light as part of its calculations, so it is generally not applicable to the mentioned implementations.

2.2 Numerical techniques

Numerical techniques for subsurface scattering are often not specific, but come for free or as an extension of a global illumination numerical approximation, since the governing equations are essentially the same. Given their generality, they are usually slower that their analytical counterpart, and often rely on heavy pre-computation steps in order to achieve interactive framerates. Jensen's Photon Mapping[Jensen and Christensen, 1998] was originally developed to render anisotropic subsurface scattering. Classical approaches as a full Monte-Carlo simulation implementation of the light-material interaction[Dorsey et al., 1999], and finite-difference methods exist in literature[Stam, 1995].

Some less general methods have been introduced in order to devise more efficient approximations when it comes to the specific problem of subsurface scattering. Stam [1995] uses the diffusion approximation with the finite difference method on the object discretized on a 3D grid. Fattal [2009] uses as well a 3D grid, that is swept with a structure called light propagation map, that stores the intermediate results until the simulation is complete.

Wang et al. [2010], instead of performing the simulation on a discretized 3D grid, makes the propagation directly in the mesh, converting it into a connected grid of tetrahedrons called *QuadGraph*. This grid can be optimized to be GPU cache friendly, and provide a real-time rendering of deformable heterogeneous objects. The problem in this method is that the QuadGraph is slow to compute (20 minutes for very complex meshes) and has heavy memory requirements for the GPU.

Precomputed radiance transfer methods is another class of general global illumination methods, that generally pre-compute part of the lighting and store it in tables[Donner et al., 2009], allowing to retrieve it efficiently with an additional memory cost.

A recent method called SSLPV - Subsurface Scattering Light Propagation Volumes [Børlum et al., 2011] extends a technique originally developed by Kaplanyan and Dachsbacher [2010] to propagate light efficiently in a scene using a set of discretized directions on a 3D grid. The method allows real-time execution times and deformable meshes with no added pre-computation step, with the drawback of not being physically accurate.

Chapter 3

Theory

In this chapter, we give a theoretical introduction to the topic dealt with in this thesis. The ultimate goal of this chapter is to introduce and describe a analytical model for subsurface scattering. First, we will give a brief introduction to what light is, and how we physically describe it. Secondly, we will introduce the basic radiometric quantities that will be used throughout the chapter. Then, we will describe how this quantities are related and can be used to describe light-material interaction, using reflectance functions, of which BSSRDF functions are a special case. Finally, we will introduce subsurface scattering and the diffusion approximation, concluding with a description of two BSSRDF functions actually used to describe it, by Jensen et al. [2001] and Frisvad et al. [2013].

3.1 Light and Radiometry

Light is a form of electromagnetic radiation, a sinusoidal wave that propagates through space. Usually by light we generically refer to *visible light*, the small part of the electromagnetic spectrum the human eye is sensible to. This small window is between 380 nm of the infrared and 770 nm of the ultraviolet, but the precise boundaries vary according to the environment and the observer. Instead explicitly noted, we will use the terms light and visible light interchangeably.

The study of light is usually referred as optics. In computer aided image synthesis, we are interested in representing faithfully how visible light propagates though the scene and how interacts with materials. In addition, we are interested on effects that are noticeable at human scales. So, we are interested for example in subsurface scattering, absorption and emission but not in phenomena like diffraction, interference and quantum effects, that happen on a microscopic scale.

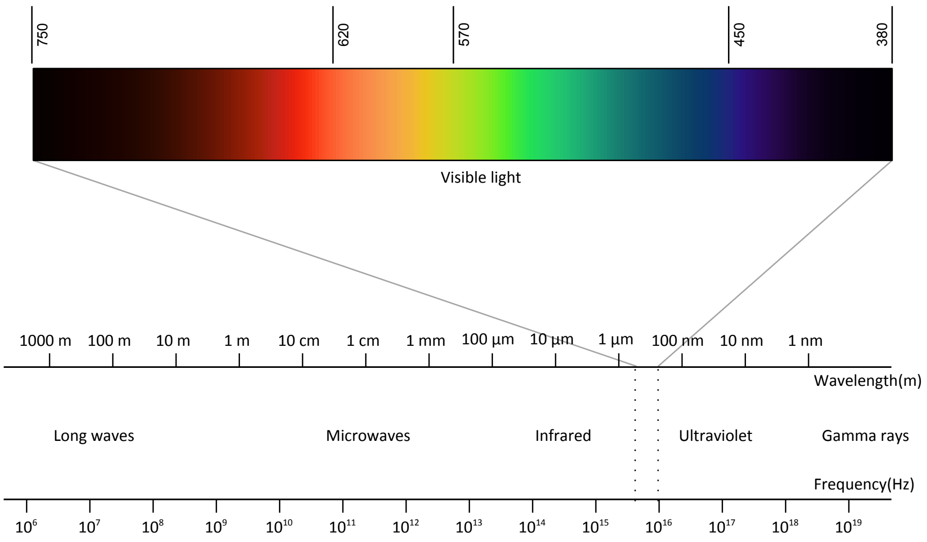


Figure 3.1: The electromagnetic spectrum.

The study of the measurement of electromagnetic radiation is called *radiometry*. The energy of light, like all the others forms of energy, is measured in *Joules (J)*, and its power in *Watts (W)*. *Photometry*, on the other hand, measures electromagnetic radiation as it is perceived from the human eye, and limits itself only to the visible spectrum, while radiometry spans all of it. The quantities for energy and power are called respectively *talbot* $(Cd \cdot s)$ and *candela (Cd)*.

In image synthesis radiometry, is employed, as its quantities are universal and can be easily converted to the photometric ones when necessary. The most important radiometric quantities used in computer graphics are radiant energy, radiant power, radiance, irradiance and intensity.

3.2 Radiometric quantities

3.2.1 Radiant flux

The radiant flux, also known as radiant power, is the most basic quantity in radiometry. It is usually indicated with the letter Φ and it is measured in joules per seconds (J/s) or Watts (W). The quantity indicates how much power the light irradiates per unit time. Given its nature, the radiant flux is constant irregardless of the distance from the light source.

3.2.2 Radiant energy

Radiant energy, usually indicated as Q, is the energy that the light carries in a certain amount of time. Like all the other SI units for energy, it is measured in joules (J). Radiant energy is usually obtained integrating the radiant flux along time:

$$Q = \int_{\Delta T} \Phi \; dt$$

Due to the dual nature of the light, the energy carried by the light can be derived both considering the flux of photons as particles, or considering light as a wave. We will not dig further into the topic, because for rendering purposes is not important how we characterize light.

3.2.3 Irradiance

Irradiance, usually defined as E, is the radiometric unit that measures the radiant flux per unit area *falling* on a surface. It is measuerd in Watts per square meter (W/m^2). It is obtained by further deriving the differential of the radiant flux by the differential area:

$$E = \frac{d\Phi}{dA}$$

Irradiance is usually the term using for the incoming power. The converse, i.e. the irradiance leaving a surface, it is usually referred as radiant exitance or radiosity, and indicated with the letter B.

3.2.4 Intensity

Intensity is often a misused term in the physics community, as it is used for a lot of different measures. Depending on the community, intensity may refer to irradiance or even to radiance (see following section). We will refer to intensity as it is generally interpreted by the optics community, i.e. radiant intensity. It is defined as the differential radiant flux per differential solid angle:

$$I(\vec{\omega}) = \frac{d\Phi}{d\omega}$$

Intensity is measured in Watts / steradian (W/sr) and it is indicated with the letter I.

3.2.5 Radiance

Radiance is the most important quantity in image synthesis. It is defined precisely as the differential flux per solid angle per projected surface area, and it is measured in Watt per steradian per square meter ($W/(sr \cdot m^2)$).

$$L(\vec{\omega}) = \frac{d^2\Phi}{d\omega dA \cos\theta}$$

Where θ is the angle between the surface normal and the incoming ray of light. Radiance is important in image synthesis because it is the natural quantity to associate with a ray of light (as it remains constant) and because is the quantity the human eye is actually sensible to. For a discussion on why radiance is related to the sensitivity of sensors and the human eye, see X.

All the other radiometric quantities can be derived from radiance:

$$
\begin{aligned}
E &= \int_{2\pi} L_i(\vec{\omega}) \cos\theta \, d\omega \\
I(\vec{\omega}) &= \int_A L(\vec{\omega}) \cos\theta \, dA \\
\Phi &= \int_A \int_{2\pi} L(\vec{\omega}) \cos\theta \, d\omega dA
\end{aligned}
$$

3.2.6 Radiometric quantities for simple lights

To help with the formulas used later in the report, we derive the above mentioned radiometric quantities for the two simplest types of light, i.e. directional and point lights.

- *Directional lights* simulate very distant light sources, in which all the rays of light are parallel (i.e. the sun). They are represented by a direction $\vec{\omega}_l$ and a constant radiance value, L.
- *Point light* simulate lights closer to the observer. Isotropic point lights are represented by a position of the light $\mathbf{x}_l$ and a constant intensity I. Point lights have a fall off that depends on the inverse square law, i.e. the radiance diminishes with the square of the distance.

Table 3.1 shows different radiometric quantities evaluated for point and directional lights, for a surface point $\mathbf{x}$ with surface normal $\vec{n}$.

Quantity	Directional light	Point light
Cosine term	$\cos\theta = \vec{n} \cdot \vec{\omega}_l$	$\cos\theta = \frac{(\mathbf{x}-\mathbf{x}_l)\cdot\vec{n}}{\lvert\mathbf{x}-\mathbf{x}_l\rvert}$
$\Phi(\mathbf{x})$ Flux	∞	$4\pi I$
$E(\mathbf{x})$ Irradiance	$L\cos\theta$	$I\frac{\cos\theta}{\lvert\mathbf{x}_l-\mathbf{x}\rvert^2}$
$I(\mathbf{x},\vec{\omega})$ Intensity	∞	I
$L(\mathbf{x},\vec{\omega})$ Radiance	L	$\frac{I}{\lvert\mathbf{x}_l-\mathbf{x}\rvert^2}$

Table 3.1: Different radiometric values for simple light sources.

3.3 Reflectance Functions

After introducing the basic radiometric quantities, we still lack a way to describe light material interaction. More precisely, we need a way to relate the incoming and the outgoing radiance on a point of a chosen surface. As we have already discussed before, we use radiance since is the radiometric quantity that is directly proportional to what the human eye measures.

3.3.1 BRDF functions

One of the possible way to describe light-material interaction is by using a BDRF function, acronym for *Bidirectional Reflectance Distribution Function.* The BRDF function $f(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o)$ is defined on one point $\mathbf{x}$ of the surface as the differential ratio between the exiting radiance and the incoming irradiance:

$$f(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) = \frac{dL_o(\mathbf{x}, \vec{\omega}_o)}{dE_i(\mathbf{x}, \vec{\omega}_i)} = \frac{dL_o(\mathbf{x}, \vec{\omega}_o)}{L_i(\mathbf{x}, \vec{\omega}_i) \cos\theta_i d\vec{\omega}_i} \tag{3.1}$$

The BRDF states that the incoming and the outgoing radiance are proportional, so that the energy hitting the material at the point $\mathbf{x}$ is proportional to the energy coming out from the point. The BRDF function is generically reciprocal for the Hemholtz reciprocity principle ($f(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) = f(\mathbf{x}, \vec{\omega}_o, \vec{\omega}_i)$), and anisotropic (if the surface changes orientation and $\vec{\omega}_i$ and $\vec{\omega}_o$ are the same, the resulting BRDFs are different). In addition, it is positive($f(\mathbf{x}, \vec{\omega}_o, \vec{\omega}_i) \geq 0$), and conserves energy, so that the energy of the outgoing ray is no grater that the one of the incoming one ($\int_{2\pi} f(\mathbf{x}, \vec{\omega}_o, \vec{\omega}_i) \cos\theta_o d\vec{\omega}_o \leq 1$).

By inverting equation 3.1, we obtain the so-called *reflectance equation*:

$$L_o(\mathbf{x}, \vec{\omega}_o) = \int_{2\pi} f(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) L_i(\mathbf{x}, \vec{\omega}_i) \cos\theta_i d\vec{\omega}_i$$

That later we will extend to obtain the rendering equation. The BRDF function has some limitations, being not able to account for all phenomena. For example, with a BRDF it is not possible to account for subsurface scattering phenomena, because it assumes the light enters and leaves the material in the same point. To model these phenomena, more complicated functions are needed, like the BSSRDF function described later in this chapter.

3.3.2 Examples of BRDF functions

There are many examples of BRDF functions in literature. In this section, we will introduce three of the simplest ones: for a more exhaustive description of BRDF functions, refer to X. The three BRDFs are the lambertian or diffuse BRDF, the specular or mirror BRDF and the Blinn-Phong BRDF.

3.3.2.1 Lambertian BRDF

In the lambertian BRDF, the incoming radiance is distributed equally in all directions, regardless of the incoming directions. To do this, the BRDF must be constant:

$$f(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) = k_d$$

We can check that then the radiance is scattered equally in all directions by simple integration:

$$\begin{aligned}
L_o(\mathbf{x}, \vec{\omega}_o) &= \int_{2\pi} f_d L_i(\mathbf{x}, \vec{\omega}_i) \cos\theta_i d\vec{\omega}_i \\
L_o(\mathbf{x}, \vec{\omega}_o) &= k_d \int_{2\pi} L_i(\mathbf{x}, \vec{\omega}_i) \cos\theta_i d\vec{\omega}_i \\
L_o(\mathbf{x}, \vec{\omega}_o) &= k_d \; E(\mathbf{x})
\end{aligned}$$

The lambertian model is an ideal model, so very few material exhibit a lambertian diffusion, like unfinished wood or spectralon, a synthetic material created in order to have a lambertian diffusion.

3.3.2.2 Mirror BRDF

Another simple kind of BRDF is the perfectly specular BRDF, or mirror BRDF. In this function, all the incoming radiance from one direction $\vec{\omega}_i$ is completely diffused into the reflected direction $\vec{\omega}_r$, defined as $\vec{\omega}_r = \vec{\omega}_i - 2(\vec{\omega}_i \cdot \vec{n})\vec{n}$. The resulting BRDF is defined as follows:

$$f(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) = \frac{\delta(\vec{\omega}_o - \vec{\omega}_r)}{\cos\theta_i}$$

The function $\delta(\vec{\omega})$ is a hemispheric delta function. Once integrated over a hemisphere, the function evaluates to one only for the vector $\vec{\omega} = \mathbf{0}$. Putting the BRDF into the reflectance equation gives the following outgoing radiance:

$$L_o(\mathbf{x}, \vec{\omega}_o) = \begin{cases} L_i(\mathbf{x}, \vec{\omega}_i) & \text{if } \vec{\omega}_o = \vec{\omega}_r \\ 0 & \text{otherwise} \end{cases}$$

that is the expected result, as all the radiance is diffused into the direction $\vec{\omega}_r$.

3.3.2.3 Glossy BRDFs

As we can see from real life experience, rarely objects are completely diffuse or completely specular. These two models are idealized models, that represent an ideal case. So, to create a realistic BRDF model, we often need to combine the two terms and add an additional one, called glossy reflection. This term is often needed to model the behaviour of the surface more realistically, especially at grazing angles.

The most used BRDF model used to model glossy reflections is based on microfacet theory. In this theory, the surface of an objecct is modeled as composed of small mirrors. In one of its classical formulation, the BRDF is represented as:

$$f(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) = \frac{DGF}{4\cos\theta_r\cos\theta_i} = \frac{GF}{4}\frac{(\vec{n}\cdot\vec{h})^s}{(\vec{n}\cdot\vec{r})(\vec{n}\cdot\vec{\omega}_i)}$$

D regulates how microfacets are distributed, and it is often modeled as $(\vec{n}\cdot\vec{h})^s$, where $\vec{h}$ is the half vector between the eye and the light, and s is an attenuation parameter. G accounts for the object self shadowing, while F is the Fresnel reflection term (more details in section Y). $\vec{r}$ is the reflection vector as defined in the previous section. See figure Z on how the vectors for the glossy reflection - $\vec{n}$, $\vec{h}$ and $\vec{r}$ - are defined.

Various alternative definitions exist for the D and G function, varying among the literature. Other glossy models exist, not based on microfacet theory.

3.3.3 The rendering equation

Given the reflectance equation, it is possible to generalize it in order to model all the lighting in an environment. In fact, this form of the reflectance equation does not account for two factors.

The first factor are emissive surfaces. We need to add an emissive radiance term $L_e(\mathbf{x}, \vec{\omega})$ that models how much radiance is a point on a surface emitting in a certain directions. This is useful to model lights as any other surface in the scene. Note that point lights have a singularity: they emit infinite radiance on the point they are placed.

The second factor is that the reflectance equation accounts only for direct illumination. In general, we want to model global illumination, i.e. to include also light that bounced onto another surface before reaching the current surface. To model this, we can replace the L_i term in the reflectance equation with another term L_r that accounts for reflected radiance. This term can be usually modeled as the product of the radiance of the light plus a visibility function $V(\mathbf{x})$.

Accounting for all the described factors, we reach the true form of the rendering equation:

$$L_o(\mathbf{x}, \vec{\omega}_o) = L_e(\mathbf{x}, \vec{\omega}) + \int_{2\pi} f(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) L_i(\mathbf{x}, \vec{\omega}_i) V(\mathbf{x}) \cos\theta_i d\vec{\omega}_i$$

This form of the rendering equation is still not completely general, since it is based on a BRDF, so no subsurface scattering effects or wavelength-changing effects (like iridescence) are impossible to model. We will extend later the rendering equation in order to account for these phenomena later on in this chapter.

3.3.4 Fresnel equations

Until now, on the considered models, we did consider only the reflected part of the radiance. When a beam of light coming from direction $\vec{\omega}_i$ hits a surface, only part of the incoming radiance gets reflected, while another part gets refracted into the material. As we can see from the setup from figure Z, we obtain the two vectors $\vec{\omega}_r$ and $\vec{\omega}_t$, the reflected and refracted vector, defined as follows:

$$\vec{\omega}_r = \vec{\omega}_i - 2(\vec{\omega}_i \cdot \vec{n})\vec{n}$$
$$\vec{\omega}_t = \eta((\vec{\omega}_i \cdot \vec{n})\vec{n} - \vec{\omega}_i) - \vec{n}\sqrt{1 - \eta^2(1 - (\vec{\omega}_i \cdot \vec{n})^2)}$$

Where $\eta = \frac{n_1}{n_2}$ is the relative index of refraction of the material. With this setup, we can use a solution to Maxwell's equations for wave propagation to describe how light spreads between reflected and refracted directions. what we obtain are called *Fresnel coefficients*. The coefficients are different according to the polarization of the incoming light, so there are two for the reflection (R_s, R_p) and two for transmission (T_s, T_p). The coefficients relate the incoming and outgoing power of the light, and by differentiation, the radiance as well.

$$R_s = \left| \frac{n_1 \cdot \cos\theta_i - n_2 \cdot \cos\theta_t}{n_1 \cdot \cos\theta_i + n_2 \cdot \cos\theta_t} \right|^2 \qquad R_p = \left| \frac{n_1 \cdot \cos\theta_t - n_2 \cdot \cos\theta_i}{n_1 \cdot \cos\theta_t + n_2 \cdot \cos\theta_i} \right|^2$$
$$T_s = \frac{n_2 \cdot \cos\theta_t}{n_1 \cdot \cos\theta_i} \left| \frac{2n_1 \cdot \cos\theta_i}{n_1 \cdot \cos\theta_i + n_2 \cdot \cos\theta_t} \right|^2 \qquad T_p = \frac{n_2 \cdot \cos\theta_t}{n_1 \cdot \cos\theta_i} \left| \frac{2n_1 \cdot \cos\theta_i}{n_1 \cdot \cos\theta_t + n_2 \cdot \cos\theta_i} \right|^2$$

In most computer graphics applications, we assume that the two polarizations are equally mixed. So, we will use the coefficient $R = \frac{R_s + R_p}{2}$ and $T = \frac{T_s + T_p}{2}$ in our calculations. Note that $R + T = 1$, so the overall energy is conserved.

3.4 Light transport and subsurface scattering

3.4.1 BSSRDF functions and generalized rendering equation

3.4.2 Scattering parameters

emission, absorption, scattering and phase functions

3.4.3 Standard dipole model

3.4.4 Directional dipole model

Chapter 4

Implementation

In this chapter, we describe the implementation details of our technique, using the approximation of the rendering equation introduced in the previous chapter. We start by giving a rather generic introduction of our algorithm, introducing then all the implementation details.

4.1 Requirements

Our algorithm, in order to be generic and applicable to a wide range of situations must meet some requirements.

- The algorithm should be *accurate*, so that the renderings are as close as possible to reference images generated using a offline Monte-Carlo simulation.
- The algorithm should be *fast*, executing at interactive frame rates.
- The algorithm should be *uv-independent*, not relying on a pre-made UV mapping of the object in order to being able to perform. The algorithm should work only if basic geometric features are provided (vertices and normals).

- The algorithm should be *adaptable*, handling in real-time dynamic light changes and object deformations. So it will not possible to use light baking or geometry form factors.

If it is not possible to satisfy all the previous within the strict constraints of real-time rendering, we want to create an algorithm that gets as close as possible to the requested result. The algorithm should progressively progressively improve and converge to an accurate result whenever possible (e.g. when the light in the scene is not changing).

4.2 Algorithm overview

By keeping the limitations presented in the previous section in mind, we introduce our four pass algorithm.

Step 1 - Light buffer
In the first step, positions and normals of the object are rendered into a texture from the light point of view. In addition, for each light a conversion matrix is computed and stored, in order to convert the position from world space to texture space.

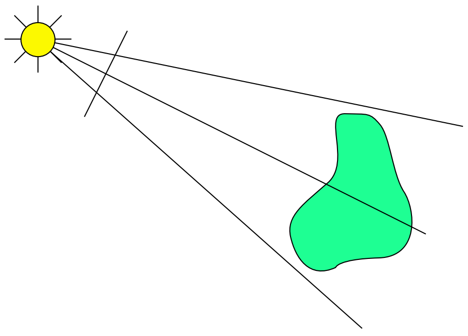


Figure 4.1: Render to G-buffer.

Step 2 - Render to array texture
In the second step, we render the object from different directions into an array texture. Each layer of the texture represents a direction from which the object is rendered to. If we are able to cover all the surface of the model, Since the BSSRDF model we are using is view-independent (apart from a Fresnel term) we can combine the results of the cubemap in a final combination step.

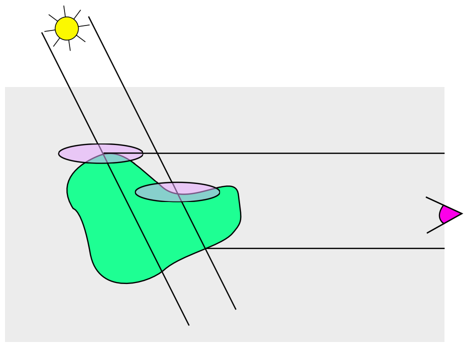


Figure 4.2: Render to cubemap.

When we are rendering a point from the orthographic camera that represents a side of the cubemap, we do as illustrated in Figure 4.3. For each fragment, we calculate the closest point to the light, sampling the texture calculated in the previous step. In Figure 4.3 we have two examples, of when the two points coincide (directly lit) and when the two points does not (light passing through the object).

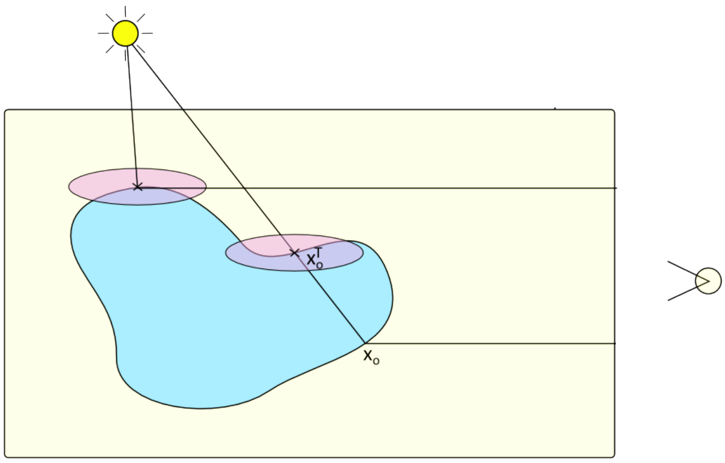


Figure 4.3: Render to cubemap, side view. Note that the fragment $\mathbf{x}_o^T$ is not rendered in this step, but only the point $\mathbf{x}_o$ is.

Then, we place a disk in the texture and accumulate the BSSRDF from the neighboring points, as shown in Figure 4.4. The points are chosen on the disk according to a pre-determined sampling scheme (discussed later). In order to accumulate the points and perform the right area integral, we need to assume that all the points on the disk cover the same area. So, we accumulate the point according to this formula:

$$C^f(\mathbf{x}_o) = \sum_{i=1}^{k} L_i(\mathbf{x}_i, \vec{\omega}_i) S_i(\mathbf{x}_i, \mathbf{x}_o, \vec{\omega}_i, \vec{\omega}_o)\ (\vec{n_i} \cdot \vec{\omega}_i)\ F_t(\vec{n}_i, \vec{\omega}_i)$$

where $\mathbf{x}_o$ is the exiting point C^f is the cubemap on face f, L_i is the incoming radiance, S_i is the BSSRDF, $\mathbf{x}_i$ and $\vec{n_i}$ are the position and the normal sampled from a point of the disc, k is the number of samples, and F_t is the incoming Fresnel term.

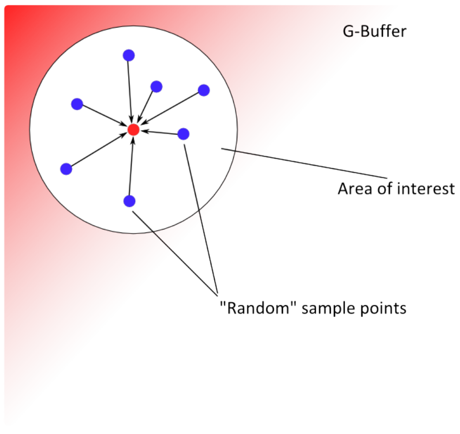


Figure 4.4: Render to cubemap, gbuffer view.

Step 3 - Combination

In this step, for each fragment on the surface we sample all the six cubemap sides as illustrated in Figure 4.5. In order to do this, we need a depth cubemap that inform us if the fragment was visible when we rendered the cubemap face. Then, each point is divided by the number of visible faces to average it. In formulas, to get the final luminance:

$$L(\mathbf{x}) = \frac{\sum_{i=1}^{6} V_i C(\mathbf{x}^i_{proj})}{\sum_{i=1}^{6} V_i}$$

where V_i is a visibility function that is zero if the point is not visible on the face i and 1 if it is visible. $C(\mathbf{x})$ is the sample from the cubamap obtained in the previous step, and $\mathbf{x}^i_{proj}$ is the point $\mathbf{x}$ projected on the i-th cubemap face.

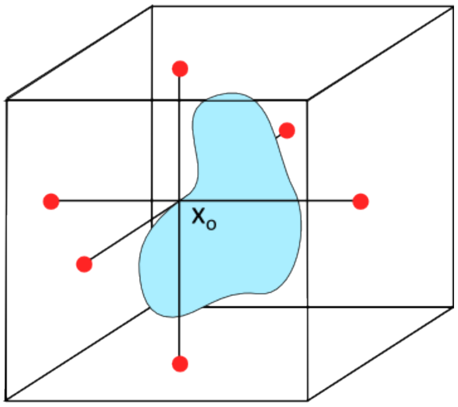


Figure 4.5: Final combination step. Note that the point is generally not visible from all the cubemap faces.

4.3 Implementation details

4.4 Artifacts

A simple implementation of the method described until now is not completely correct. In fact, since we are discretizing the surface on a texture, some sampling artifacts are inevitable. During our implementation we identified three different types of artifacts:

- Incorrect sampling of the G-buffer
- Incorrect sampling of the cubemap
- Shadow bias (when sampling the depth of the texture).

That are described in detail in the following sections.

4.4.1 Incorrect sampling of the G-buffer

In order to sample the G-buffer correctly, we need to modify the world coordinate of the point $\mathbf{x}_o$ with normal $\vec{n}_o$ in order to "shrink" a little bit towards the inside of the object according to the light direction $\vec{\omega}_l$:

$$\mathbf{x}'_o = \mathbf{x}_o - \epsilon_g(\vec{n}_o - \vec{\omega}_l(\vec{\omega}_l \cdot \vec{n}_o))$$

4.4.2 Incorrect sampling of the cubemap

Since we are sampling a cubemap, we do not need to account for the light direction in this case. In addition, the cubemap needs a 3D vector to be sampled with. So, instead of using $\mathbf{x}_o$, we use a point slightly intruded (i.e. displaced along the normal) in the calculations, according to:

$$\mathbf{x}'_o = \mathbf{x}_o - \epsilon_c\vec{n}_o$$

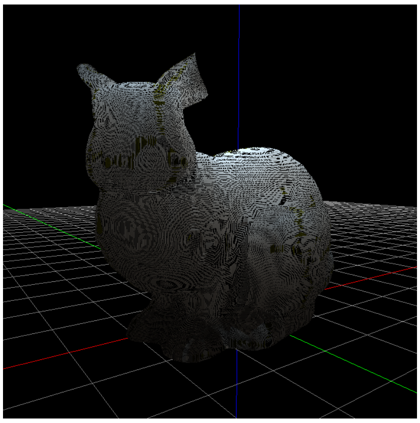
4.4.3 Shadow bias

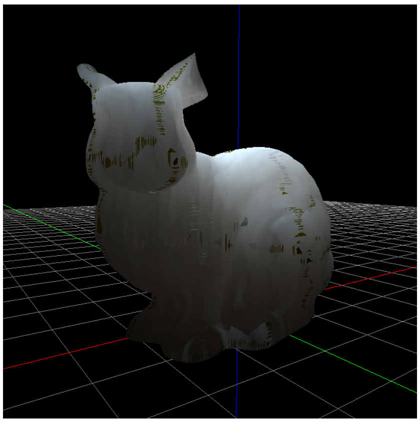
In order to avoid artifacts such as shadows acne, we use a bias when comparing the z values of a point to determine if the point is in shadow or not. This implies that we need to convert the depth value from texture space ($\mathbf{z}_{tex}$) to world space again ($\mathbf{z}_{world}$). Since we are using an orthographics camera, the z value is the same in clip coordinates and in normalized device coordinates. Then, we simply use the camera projection properties ($\mathbf{z}_{far}$, $\mathbf{z}_{near}$) to convert the depth into the camera local space, in order to finally add the camera position transformed z value ($\mathbf{z}_{camera}$) and reconstruct the depth in world space:

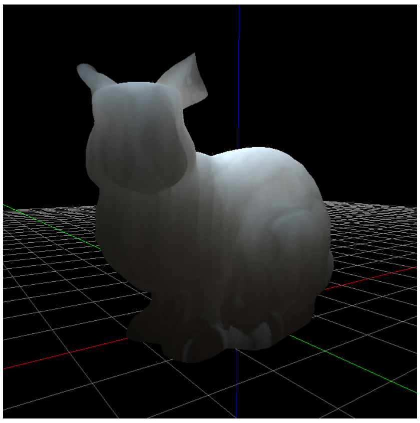
$$\mathbf{z}_{world} = \mathbf{z}_{camera} - \frac{\mathbf{z}_{far} - \mathbf{z}_{near}}{2}\left(2\mathbf{z}_{tex} - 1 + \frac{\mathbf{z}_{far} + \mathbf{z}_{near}}{\mathbf{z}_{far} - \mathbf{z}_{near}}\right)$$

And finally compare the obtained z with the z position in world space of the point ($\mathbf{z}$) using the bias ϵ_b, so a point is lit iff:

$$\mathbf{z}_{world} - \epsilon_b < \mathbf{z}$$


(a) Without shadow bias and sampling fixes


(b) With shadow bias and without sampling fixes


(c) With both shadow bias and sampling fixes

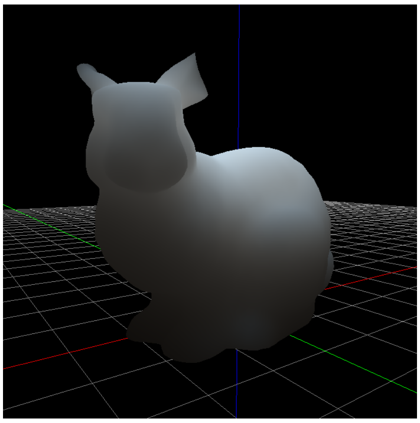

(d) Reference

Figure 4.6: Progressively removing artifacts to get to the final image.

Chapter 5

Results

Bibliography

Jesper Børlum, Brian Bunch Christensen, Thomas Kim Kjeldsen, Peter Trier Mikkelsen, Karsten Østergaard Noe, Jens Rimestad, and Jesper Mosegaard. Sslpv: Subsurface light propagation volumes. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, HPG '11, pages 7–14, New York, NY, USA, 2011. ACM. URL `http://doi.acm.org/10.1145/2018323.2018325`.

Carsten Dachsbacher and Marc Stamminger. Translucent shadow maps. In *Proceedings of the 14th Eurographics Workshop on Rendering*, EGRW '03, pages 197–201, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. URL `http://dl.acm.org/citation.cfm?id=882404.882433`.

Eugene D'Eon. A better dipole (a publicly available manuscript). Technical report, -, 2012. URL `.http://www.eugenedeon.com/papers/betterdipole.pdf.`

Eugene D'Eon and Geoffrey Irving. A quantized-diffusion model for rendering translucent materials. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, pages 56:1–56:14, New York, NY, USA, 2011. ACM. URL `http://doi.acm.org/10.1145/1964921.1964951`.

Eugene d'Eon, David Luebke, and Eric Enderton. Efficient rendering of human skin. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, pages 147–157, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. URL `http://dx.doi.org/10.2312/EGWR/EGSR07/147-157`.

Craig Donner and Henrik Wann Jensen. Light diffusion in multi-layered translucent materials. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages

1032–1039, New York, NY, USA, 2005. ACM. URL `http://doi.acm.org/10.1145/1186822.1073308`.

Craig Donner, Jason Lawrence, Ravi Ramamoorthi, Toshiya Hachisuka, Henrik Wann Jensen, and Shree Nayar. An empirical bssrdf model. *ACM Trans. Graph.*, 28(3):30:1–30:10, July 2009. URL `http://doi.acm.org/10.1145/1531326.1531336`.

Julie Dorsey, Alan Edelman, Henrik Wann Jensen, Justin Legakis, and Hans Køhling Pedersen. Modeling and rendering of weathered stone. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 225–234, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. URL `http://dx.doi.org/10.1145/311535.311560`.

Raanan Fattal. Participating media illumination using light propagation maps. *ACM Trans. Graph.*, 28(1):7:1–7:11, February 2009. URL `http://doi.acm.org/10.1145/1477926.1477933`.

J. R. Frisvad, T. Hachisuka, and T. K. Kjeldsen. Directional dipole for subsurface scattering in translucent materials. Technical report, DTU Compute, 2013. URL `http://www2.imm.dtu.dk/pubdb/p.php?6646`. Unpublished manuscript.

Henrik Wann Jensen and Juan Buhler. A rapid hierarchical rendering technique for translucent materials. *ACM Trans. Graph.*, 21(3):576–581, July 2002. URL `http://doi.acm.org/10.1145/566654.566619`.

Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 311–320, New York, NY, USA, 1998. ACM. URL `http://doi.acm.org/10.1145/280814.280925`.

Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 511–518, New York, NY, USA, 2001. ACM. URL `http://doi.acm.org/10.1145/383259.383319`.

Jorge Jimenez, Veronica Sundstedt, and Diego Gutierrez. Screen-space perceptual rendering of human skin. *ACM Trans. Appl. Percept.*, 6(4):23:1–23:15, October 2009. URL `http://doi.acm.org/10.1145/1609967.1609970`.

Anton Kaplanyan and Carsten Dachsbacher. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, pages 99–107, New York, NY, USA, 2010. ACM. URL http://doi.acm.org/10.1145/1730804.1730821.

Hendrik P. A. Lensch, Michael Goesele, Philippe Bekaert, Jan Kautz, Marcus A. Magnor, Jochen Lang, and Hans-Peter Seidel. Interactive rendering of translucent objects. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, PG '02, pages 214–, Washington, DC, USA, 2002. IEEE Computer Society. URL http://dl.acm.org/citation.cfm?id=826030.826632.

Tom Mertens, J. Kautz, P. Bekaert, F. Van Reeth, and H.-P. Seidel. Efficient rendering of local subsurface scattering. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on*, pages 51–58, Oct 2003a.

Tom Mertens, Jan Kautz, Philippe Bekaert, Hans-Peter Seidelz, and Frank Van Reeth. Interactive rendering of translucent deformable objects. In *Proceedings of the 14th Eurographics Workshop on Rendering*, EGRW '03, pages 130–140, Aire-la-Ville, Switzerland, Switzerland, 2003b. Eurographics Association. URL http://dl.acm.org/citation.cfm?id=882404.882423.

Matt Pharr and Pat Hanrahan. Monte carlo evaluation of non-linear scattering equations for subsurface reflection. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 75–84, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. URL http://dx.doi.org/10.1145/344779.344824.

M.A. Shah, J. Konttinen, and S. Pattanaik. Image-space subsurface scattering for interactive rendering of deformable translucent objects. *Computer Graphics and Applications, IEEE*, 29(1):66–78, Jan 2009.

Jos Stam. Multiple scattering as a diffusion process. In PatrickM. Hanrahan and Werner Purgathofer, editors, *Rendering Techniques 1995*, Eurographics, pages 41–50. Springer Vienna, 1995. URL http://dx.doi.org/10.1007/978-3-7091-9430-0_5.

Anna Tomaszewska and Krzysztof Stefanowski. Real-time spherical harmonics based subsurface scattering. In Aurélio Campilho and Mohamed Kamel, editors, *Image Analysis and Recognition*, volume 7324 of *Lecture Notes in Computer Science*, pages 402–409. Springer Berlin Heidelberg, 2012. URL http://dx.doi.org/10.1007/978-3-642-31295-3_47.

Yajun Wang, Jiaping Wang, Nicolas Holzschuch, Kartic Subr, Jun-Hai Yong, and Baining Guo. Real-time rendering of heterogeneous translucent objects with arbitrary shapes. *Comput. Graph. Forum*, 29(2):497–506, 2010. URL http://dblp.uni-trier.de/db/journals/cgf/cgf29.html#WangWHSYG10.