UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

MAXON A NEMETSCHEK COMPANY | 3D FOR THE REAL WORLD
MAXON Computer GmbH

Interactive Physically-Based Hair Rendering

Masterarbeit

zur Erlangung des Grades eines Master of Science (M.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Jochen Hunz

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	Mauro van de Vlasakker , M.Sc. (MAXON Computer GmbH)

Koblenz, im März 2016

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....
(Ort, Datum) (Unterschrift)

Abstract

This work covers techniques for interactive and physically - based rendering of hair for computer generated imagery (CGI). To this end techniques for the simulation and approximation of the interaction of light with hair are derived and presented. Furthermore it is described how hair, despite such computationally expensive algorithms, can be rendered interactively. Techniques for computing the shadowing in hair as well as approaches to render hair as transparent geometry are also presented. A main focus of this work is the *DBK-Buffer*, which was conceived, implemented and evaluated. Using the *DBK-Buffer*, it is possible to render thousands of hairs as transparent geometry without being dependent on either the newest GPU hardware generation or a great amount of video memory. Moreover, a comprehensive evaluation of all the techniques described was conducted with respect to the visual quality, performance and memory requirements. This revealed that hair can be rendered physically - based at interactive or even at real - time frame rates.

Zusammenfassung

Die vorliegende Arbeit behandelt Techniken zur interakativen und physik - alisch basierten Darstellung von Haaren für Computer-Generated Imagery (CGI). Dafür werden Techniken zur Simulation und Approximierung der Interaktionen von Licht mit Haar hergeleitet und vorgestellt. Des Weiteren wird beschrieben, wie Haare, trotz solch berechnungsintensiver Algorithmen, sehr interaktiv dargestellt werden können. Verfahren zur Berechnung von Schatten in Haaren sowie Ansätze zur effizienten Darstellung von Haar als transparente Geometrie werden ebenfalls vorgestellt. Einen Hauptschwerpunkt der Arbeit bildet dabei der *DBK-Buffer*, welcher im Rahmen dieser konzeptioniert, implementiert und evaluiert wurde. Mit Hilfe des DBK-Buffers ist es möglich tausende von transparenten Haaren sehr effizient darzustellen ohne auf Funktionalitäten der neusten Grafikkarten-Generation, oder sehr viel Videospeicher, angewiesen zu sein. Darüber hinaus wurde eine umfassende Evaluierung der beschriebenden Techniken bezüglich der visuellen Qualität, der Performanz und des Speicheraufwandes durchgeführt. Dabei wurde gezeigt, dass Haare nicht nur mit interaktiven, sondern sogar mit echtzeitfähigen Bildwiederholungsraten physikalisch basiert dargestellt werden können.

Acknowledgments

I would first like to thank my thesis supervisor Prof. Dr. Stefan Müller of the University of Koblenz - Landau. Prof. Müller always guided me in the right direction when I ran into difficulties but also allowed me freedom for this thesis to be my own work. Besides this work, Prof. Müller played a significant role during my studies as my teacher and advisor. Without his talent to inspire and motivate his students, things would have been different.
I would also like to thank Mauro van de Vlasakker from the Maxon Computer GmbH as the second reader of this thesis. I am gratefully indebted to him for his valuable comments and contributions. Moreover, I would like to thank to everyone at the Maxon Computer GmbH who made this thesis possible. Especially, I would like to thank Maik Schulze, Stefan Minning and Fritz Kemmler as they always discussed algorithmic details with me, Edd Biddulph for his valuable help with OpenGL and Sebastian Jeckel for giving me deeper insights into the C++ programming language.
I would also like to acknowledge John Rees and Simon Wagner for proof reading, not only of this thesis, and for all their valuable comments. Furthermore, I would like to thank my fellow students, Dennis Schlösser and Christopher Krey, who enriched my last five years at university.
Finally, I must express my very profound gratitude to my parents, Brigitte and Rolf, my sister Anke and my brother Martin, as well as to my girlfriend Lisa. Neither this thesis nor my studies would have been possible without your unfailing support and continuous encouragement. Thank you!

Jochen Hunz

Contents

1 Introduction

The importance of hair and fur in computer graphics imagery (CGI) becomes more and more relevant. Obviously, the movie industry has a huge interest in hair and fur rendering for all kinds of movies as nearly every character has some kind of hair or fur. On the one hand, there are cartoonish movies such as *Tangled*, *Frozen*, and *Zoomania* from the *Walt Disney Animation Studios*. On the other hand, there are photo-realistic kinds of movies, where CGI augments real footage, as in recent movies such as *Planet of the Apes* by *20th Century Fox* (see figure 1). All these movies are typically rendered using offline rendering technologies like ray or path tracing. These techniques provide the best quality currently available in exchange for very long computation times.

Figure 1: Hair and fur rendering in production movies. Top: Cartoonish looking hair of Rapunzel and Flynn in *Tangled* from the *Walt Disney Animation Studios*. Bottom: full CGI Ceasar with his photo-realisitc fur beside James Franco playing as William Rodman in *Rise of the Planet of the Apes* from *20th Century Fox*.

For production movies, tools like *Shave and a Haircut* by Joe Alter [20] and *yeti*[*] by Peregrine Labs [40] are commonly used. Besides that, 3D content creation tools such as *Cinema 4D*, developed by the MAXON Computer GmbH, provide in-house solutions for creating, authoring, and rendering of hair (see figure 2).

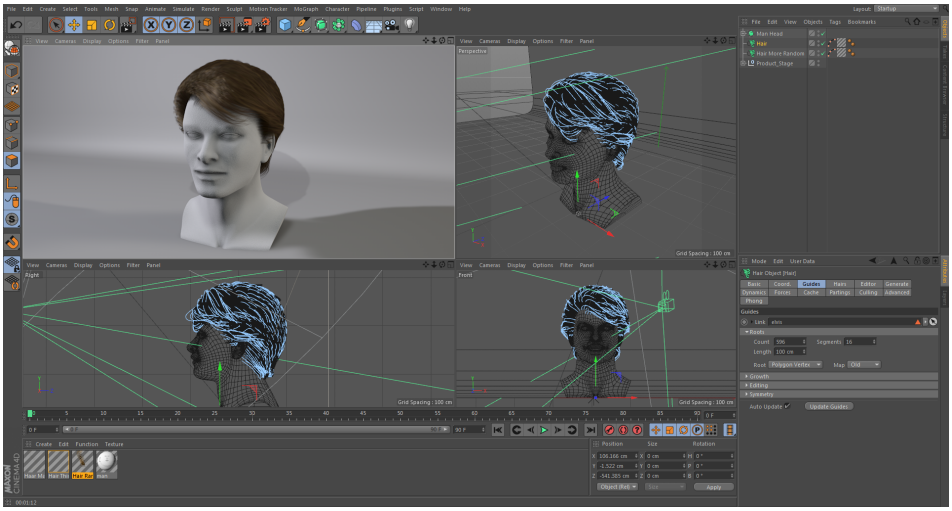


Figure 2: Screenshot of *Cinema 4D* showing the interface as well as several viewports. Here, the guide hairs of the hair model are highlighted with a blue color. The top left image shows a final render using the internal hair material system.

The power of graphics processing units (GPUs) is continuously growing which opens another market for hair and fur rendering: real-time applications such as video games and content creation tools which uses the GPU as a rasterizer. One of the first fully fledged real-time implementation of a physically-based hair model was done by Nvidia for their *Nalu Demo* in 2004 (see figure 3a) [11]. The effort Nvidia spent on hair and fur rendering resulted in a Software and SDK called *HairWorks* in 2014 [12]. *HairWorks* is a content creation and authoring tool (see figure 3b) which also provides an SDK in order to include *HairWorks* in other software. Nvidia's competitor AMD provides an open source system called *TressFX* [13] as shown in figure 4a. *TressFX* is a library for rendering and simulation of hair and fur. Beside that, *TressFX* comes with a plugin for *Maya* [3] in order to create and author hair and fur. *TressFX* was used for the reboot of *Tomb Raider* by SquareEnix in 2013 to render and simulate Lara's pony-tail (see figure 4b).

(a) Nalu demo. **(b)** HairWorks.

Figure 3: a) Real-time hair rendering in the *Nalu Demo* by Nvidia [11] and b) the hair creation and authoring tool *HairWorks* by Nvidia. [12]

(a) TressFX. **(b)** Tomb Raider.

Figure 4: a) Real-time hair rendering in *TressFX* by AMD [13] and b) *TressFX* used in the video game *Tomb Raider* by SquareEnix.

1.1 Motivation

This thesis was created in strong cooperation with the MAXON Computer GmbH, the developer of *Cinema 4D* and *BodyPaint 3D*. We asked several users of *Cinema 4D* to find out what they need and miss in terms of hair creation and rendering. As a result, the users came up with several very interesting ideas. Unfortunately, most ideas would not fit into the scope of this thesis or wouldn't offer good fields for research. However, the users do miss a good preview rendering of the hair shading in the OpenGL based viewport of *Cinema 4D* and other tools. For instance, the viewport of *Cin-*

ema 4D is only capable of previewing the geometry of the hair but not the actual shading as shown in figure 2. Because of this, we decided to dig into the great field of interactive physically-based hair rendering as it provides many resources and many open questions for scientific research. Nevertheless, it is worth noting that the implementation that was done during this thesis is a standalone project which is totally independent of *Cinema 4D*.

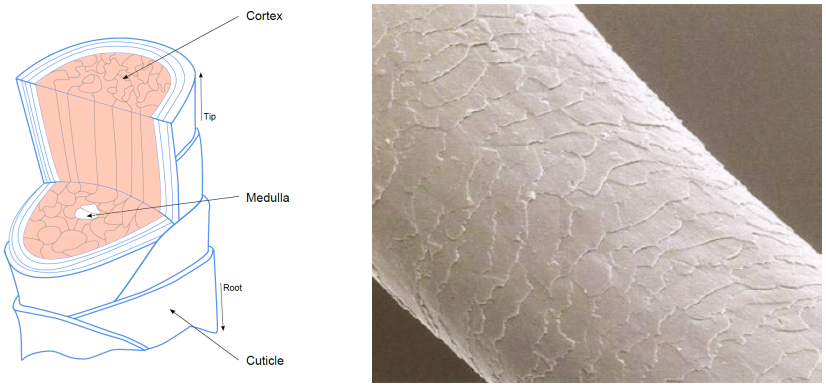
1.2 Organization of this Work

Chapter 2 introduces the fundamentals that are necessary to understand the theory of physically-based hair rendering. Therefore, an overview of the most important properties of human hair is given in chapter 2.1. Furthermore, chapters 2.2 - 2.5 introduce concepts of math, physics, and rendering that are used throughout this thesis. In chapter 3, a comprehensive discussion and derivation of the most important hair shading models is given. Thus, there is the phenomenological model of Kajiya and Kay (3.2), the physically-based model of Marschner et al. (3.3), and the artist friendly hair shading system of Sadeghi et al. (3.4) which was developed by the Walt Disney and Pixar Animation Studios for their production renderer *RenderMan* [43, 50]. As shadowing is very important for the appearance of hair, chapter 4 describes techniques suited for real-time hair rendering. As hair is a semi-transparent material, chapter 5 addresses techniques for rendering transparent geometry. Transparency for hair is difficult and expensive in terms of performance and memory as hair has a very complex and dense geometry. In chapter 5.3, the *DBK-Buffer* is introduced which is the main contribution of this thesis. The *DBK-Buffer* enables rendering of hair in a very efficient way without being dependent on recent hardware features that most GPUs don't even support. Furthermore, the *DBK-Buffer* is very efficient in terms of memory allocation which makes it even more suitable for GPUs. A comprehensive evaluation of the shading models and the shadowing and transparency techniques is given in chapter 6. Finally, chapter 7 concludes this thesis and anticipates possible future work.

2 Fundamentals

2.1 Properties of Human Hair

Hair is a filament consisting of layered keratin structures (cf. figure 5a). The outer hull of hair, called *cuticle*, is covered with tilted scales, growing from the root of the hair to its tip, as one can see in figure 5b. These scales are a very important for rendering as they shift reflected and transmitted light ray about several degrees. The innermost part of the hair is the *medulla*, which is surrounded by the *cortex*. The cortex and the medulla contain many pigment granules which define the color of the hair. The thickness of human hair varies between 50 and 120 μm based on the ethnic group [47]. For instance, the average diameter of European hair is $63.93\mu m$, the thickness of Asian hair can average around $120\mu m$. Moreover, Asian hair tend to be circular in cross section, where European hair has a significant eccentricity of approximately 0.67:1. A much more detailed overview of the geometric and physical properties of human hair is given by Sobottka and Weber [53].



(a) Illustration of a hair strand.

(b) Hair under a microscope.

Figure 5: a) Hair as a schematic illustration showing cuticle, cortex, and medulla [48]. b) Hair as seen under a microscope showing scales growing from the root to the tip [64].

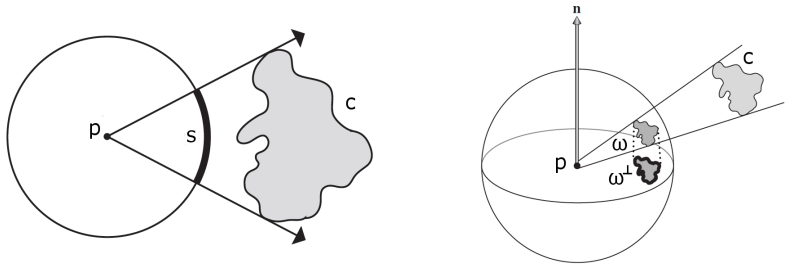
2.2 Solid Angles

Solid angles are a fundamental concept for physically-based rendering. Basically, they are an extension of the two-dimensional planar angles (cf. figure 6a) to an angle on a unit sphere (cf. figure 6b) [41]. While planar angles are measured in radians, solid angles are measured in *steradians* sr. The solid angle ω of an object c is the object's projected area on the unit sphere. Thus an entire sphere subtends a solid angle of $\omega = 4\pi$. Usually, directions within a sphere are measured in spherical coordinates (θ, ϕ), where ϕ

is the azimuth. Using spherical coordinates, the differential solid angle $d\omega$ is written as

$$d\omega = sin\theta d\theta d\phi. \tag{1}$$

Please see figure 7 for a visual derivation of this equation. In literature, one can often find the projected solid angle $d\omega^{\perp}$, which simply is the cosine - weighted solid angle: $d\omega^{\perp} = |cos\theta|\, d\omega$. Using the projected solid angle, several equations can be simplified.



(a) Planar angle. **(b)** Solid angle.

Figure 6: a) The planar angle of an object c is the length of the arc s on the unit sphere. b) The solid angle ω of an object c is the object's projected area on the unit sphere. The differential projected solid angle is the cosine - weighted solid angle: $d\omega^{\perp} = |cos\theta|\, d\omega$. Images from [41].

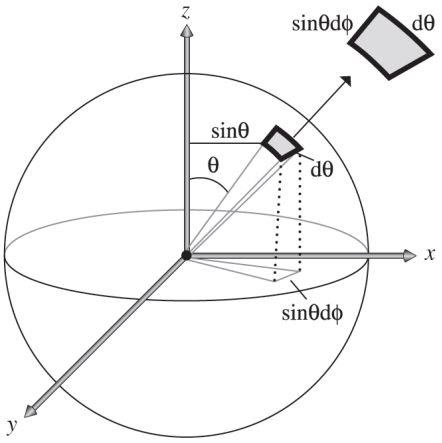


Figure 7: Derivation of equation 1: The differential solid angle $d\omega$ is the product of the two edges $sin\theta d\phi$ and $d\theta$. Image from [41].

2.3 Radiometry

The science of *radiometry* deals with the measurement of electromagnetic radiation and describes the flow of energy through space. Radiation is

made of a flow of photons which behaves like waves or particles. Each photon has an associated wavelength λ or frequency ν which affects the interaction between photons and matter. The wavelength and the frequency of a photon are related as

$$\nu = \frac{c}{\lambda}, \tag{2}$$

$$\lambda = \frac{c}{\nu}, \tag{3}$$

$$Q = h\nu, \tag{4}$$

where $h = 6.62620 * 10^{-34} J * s$ is Planck's constant and $c = 2.998 * 10^8 m/s$ is the speed of light [2, p. 202]. The radiant energy Q is the basic unit in radiometry and is measured in joules.
The electromagnetic spectrum covers the range of all possible wavelengths starting from extremely low frequency radio waves to extremely high frequency gamma rays. Thereby, the spectrum of wavelengths perceptible to human eye ranges roughly from 380 nm to 780 nm. In the following, the radiometric units and their relationships are described. For a total reference of all used quantities and units see table 1.

2.3.1 Radiometric Quantities

The entire emitted energy per time unit is called the *radiant flux* Φ. It's unit is watt = J/s and it's given by

$$\Phi = dQ/dt. \tag{5}$$

Basically, it is a measure for the power of a light source. Derived from that, the *radiant intensity* I measures the emitted radiant flux per solid angle ω and therefore is measured in watts per steradiant W/sr:

$$I = d\Phi/d\omega. \tag{6}$$

For the purpose of physically-based rendering the terms *irradiance* and *radiosity* are very important. Thereby, the *irradiance* E is the arriving radiant flux per area, the *radiosity*, or *radiant exitance*, B is the leaving radiant flux per area:

$$E = d\Phi/dA_e, \tag{7}$$

$$B = d\Phi/dA_s. \tag{8}$$

The last important radiometric quantity is the *radiance* L. The unit of radiance is watts per square meter per steradiant. It measures the power per unit projected area $dA^{\perp} = dA * \overline{cos}\theta$ per unit solid angle $d\omega$ (cf. figure 8). Therefore, the radiance is given by:

$$\begin{aligned} L &= dI/(dA * \overline{cos}\theta) \\ &= (d^2\Phi)/(dA * \overline{cos}\theta * d\omega). \end{aligned} \tag{9}$$

The symbol $\overline{cos}$ represents a cosine clamped to zero. It is used because a negative cosine value only occurs when the projected area is 0 [2, p. 207]. A very important property of radiance is that it remains constant along a single light ray through empty space [41, p. 285]. This property is the basis of a rendering technique called *raytracing*.

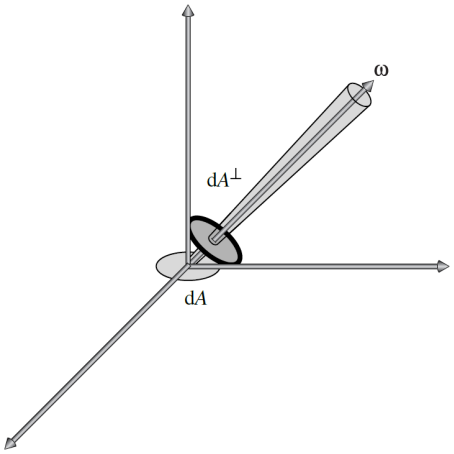


Figure 8: Radiance L is flux per unit projected area per unit solid angle. Image from [41].

2.4 BRDF Theory

In 1977, the physicist Fred Nicodemus defined a function that describes how a surface reflects light as the *bidirectional relectance distribution function* (BRDF) [39]. Thus, a BRDF defines the ratio between differential outgoing radiance and differential incoming irradiance:

$$f(d\omega_i, d\omega_o) = f(\theta_i, \phi_i, \theta_o, \phi_o) = \frac{dL_o(d\omega_o)}{dE_i(d\omega_i)}. \tag{10}$$

Given equation 7 and 9, the BRDF can also be expressed as:

$$f(d\omega_i, d\omega_o) = f(\theta_i, \phi_i, \theta_o, \phi_o) = \frac{dL_o(d\omega_o)}{dL_i(d\omega_i)\overline{cos}\theta_i d\omega_i} \tag{11}$$

Therefore, the units of the BRDF are inverse solid angles sr^{-1} as it is defined as radiance devided by irradiance. Accordingly, the BRDF value is the relative amount of energy reflected in direction ω_o given an incoming direction ω_i (cf. figure 9). A more detailed look at the notation is given in chapter 3.1.

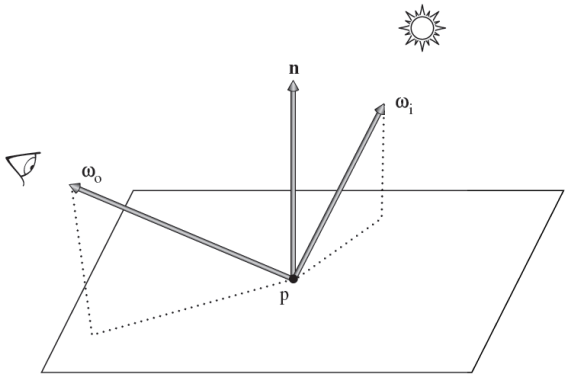


Figure 9: Bidirectional reflectance distribution function (BRDF). Image from [41].

In general, the BRDF is an approximisation of the more complex *bidirectional surface scattering reflectance distribution function* (BSSRDF), which also handles phenomena like subsurface scattering [39]. Depending on the surface's properties, light can scatter into the surface (refraction) or scatter away from it (reflection). In addition, refracted light can be absorbed or can be scattered multiple times within the subsurface. Because of to this, it is also possible that light can exit the subsurface again in more or less random directions. Figure 10 shows such a complex interaction of light with some surface. Here, the green and red circles are also noteworthy as they show the region covered by a pixel at two different scales of observations. Thereby, the green circle represents the case where each pixel covers a large region. Because of this, all complex scattering events can be approximated as happening at a single point. In contrast to this, the red circle covers only a very small area of the surface, which makes more complex algorithms necessary.

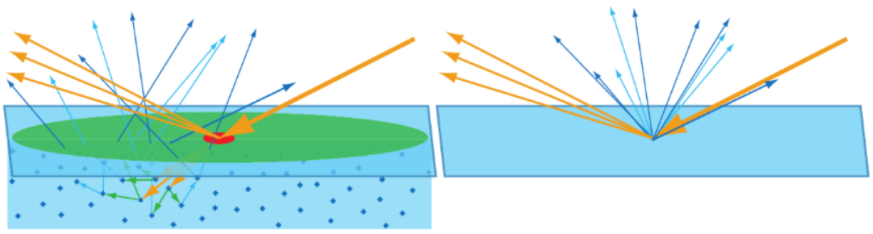


Figure 10: Complex interaction of light with a surface leading to subsurface scattering. The left image shows a full simulation of all scattered paths in the subsurface whereby the scattered light in the right image emits from the entry point only. Image from [2].

The ratio of all outgoing radiance L_o to all incoming radiance L_i is called the *directional reflectance* ρ:

$$\rho = \frac{\Phi_o}{\Phi_i} = \frac{\int_{2\pi} L_o(d\omega_o) cos\theta_o d\omega_o}{\int_{2\pi} L_i(d\omega_i) cos\theta_i d\omega_i} \tag{12}$$

Beyond that, the *directional-hemispherical reflectance* $\rho(\theta_i)$ measures the total reflected light, regardless of an outgoing direction, for a specific incoming direction [2, p. 226]. Therefore it is defined as:

$$\rho(\theta_i) = \int_{2\pi} f(d\omega_i, d\omega_o) cos\theta_o d\omega_o. \tag{13}$$

2.4.1 Physically Plausible BRDFs

A BRDF is called *physically plausible* when it fulfills three properties:

- the BRDF is not negative,
- the conservation of energy ($0 \leq \rho(\theta_i) < 1$) holds, and
- the *Helmholtz reciprocity* can be applied.

The conservation of energy is essential for light simulating algorithms as they would fail to converge otherwise. In real-time rendering approaches, a BRDF that is not energy conserving can still be used, but the shaded surface might appear much too bright. The term *Helmholtz reciprocity* is a principle of physics named after the German physicist Hermann von Helmholtz [57]. It basically states that the incoming and outgoing directions can be switched and the result of the BRDF remains the same:

$$f(d\omega_i, d\omega_o) = f(d\omega_o, d\omega_i). \tag{14}$$

However, it is worth noting that the Helmholtz reciprocity is often violated without noticeable artifacts in practice [2, p.226].

2.4.2 Rendering Equation

The *rendering equation*, or *light transport equation*, is a very prominent integral equation first introduced by James Kajiya [21] and by Immel and Cohen [19] in 1986. In order to solve this equation, various rendering techniques such as *path tracing* or *radiosity* were developed. The rendering equation is:

$$L_o(d\omega_o) = L_e(d\omega_o) + \int_{\Omega} f(d\omega_o, d\omega_i) L_i(d\omega_i) \left|cos\theta_i\right| d\omega_i. \tag{15}$$

It simply states that the total outgoing radiance L_o along a direction ω_o is the emitted radiance L_e in that direction plus the incident radiance L_i coming from all directions $d\omega_i$ scaled by the BRDF f and the cosine of θ_i.

2.5 Physics of Light Scattering

There are two phenomena in optics that are very important to simulate light scattering: *Snell's law* and the *Fresnel equations*. When a ray of light hits a surface with an angle of inclination θ_i, the reflected ray ω_r can be computed as

$$\omega_r = 2(n * \omega_i)n - \omega_i. \tag{16}$$

When taking transmission into account, the angle of transmission θ_t depends on the angle of inclination θ_i as well as on the two indices of refraction (IOR) η_i and η_t, whereby η_i is the IOR for the medium above the interface and η_t is the IOR for the medium below the interface (see figure 11). This dependence is known as Snell's law, which is:

$$\eta_i sin\theta_i = \eta_t sin\theta_t. \tag{17}$$

Using this law, θ_t can be easily computed. It is worth noting that in general, the IOR varies with the wavelength of light. Due to this, light may scatter in multiple directions. This effect, called *dispersion*, splits light into its spectral components when interacting with a prism, for instance. However, dispersion is mostly ignored in real-time rendering to simplify the rendering equation. When light interacts with an interface, it is either reflected or refracted.

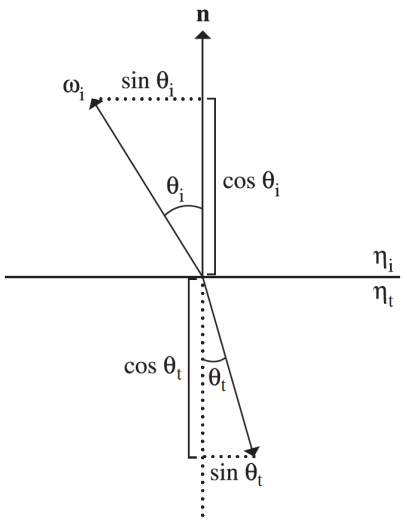


Figure 11: Geometry and terms used for Snell's law and the Fresnel equations [41].

The Fresnel equations, derived from the Maxwell equations, precisely describe the amount of light that gets reflected from a surface. There exist two different versions of Fresnel equations for *conductors* (materials that conduct electricity) and for *dielectrics*, respectively.

The Fresnel equations for dielectrics are:

$$F(\eta_i, \eta_t, \theta_i) = \frac{1}{2}(F_{\parallel}(\eta_i, \eta_t, \theta_i)^2 + F_{\perp}(\eta_i, \eta_t, \theta_i)^2), \tag{18}$$

where

$$F_{\parallel}(\eta_i, \eta_t, \theta_i) = \left| \frac{\eta_t cos\theta_i - \eta_i cos\theta_t}{\eta_t cos\theta_i + \eta_i cos\theta_t} \right|, \tag{19}$$

$$F_{\perp}(\eta_i, \eta_t, \theta_i) = \left| \frac{\eta_i cos\theta_i - \eta_t cos\theta_t}{\eta_i cos\theta_i + \eta_t cos\theta_t} \right|. \tag{20}$$

$F_{\parallel}$ accounts for parallel polarized light and $F_{\perp}$ for perpendicular polarized light. As hair is a dielectric material, the Fresnel equations for conductors are omitted here.

3 Light Scattering in Fibers

3.1 Terminology and Notation

The complex math and physics behind light scattering in fibers requires a well-considered notation and terminology as in figure 12. All terms and symbols used in this thesis are listed in table 1. The tangent of the hair is u, going from the root of the hair to its tip. It builds a right-handed orthonormal basis together with the vectors v and w, whereby v is the major and w is the minor axis. All vectors perpendicular to u lie in the $v - w$ plane, which is called the *normal plane*. The incoming light direction is noted as l in Cartesian coordinates and ω_i in spherical coordinates, whereby the outgoing view direction is noted as e and ω_o, respectively. The spherical coordinates are formed by the angles $\theta_X \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $\phi_X \in [0, 2\pi]$, where X is either i for the incoming direction or o for the outgoing one. The *longitudinal* inclination θ_X is an angle with respect to the normal plane, whereby the angle ϕ_X is the *azimuth* around the hair. The angles are measured in a way so that $\theta_X = 0$ is perpendicular to the hair, $\theta_X = \frac{\pi}{2}$ is u, $\theta_X = -\frac{\pi}{2}$ is $-u$, $\phi_X = 0$ is v and $\phi_X = \frac{\pi}{2}$ is w.
In addition to these angles several derived angles are used as well. Thus, the difference angle θ_d is defined as

$$\theta_d = \frac{\theta_o - \theta_i}{2}. \tag{21}$$

The relative azimuth ϕ is given by

$$\phi = \phi_o - \phi_i. \tag{22}$$

Beside that, the half angles θ_h and ϕ_h are simply

$$\theta_h = (\theta_i + \theta_o)/2, \tag{23}$$

and

$$\phi_h = (\phi_i + \phi_o)/2. \tag{24}$$

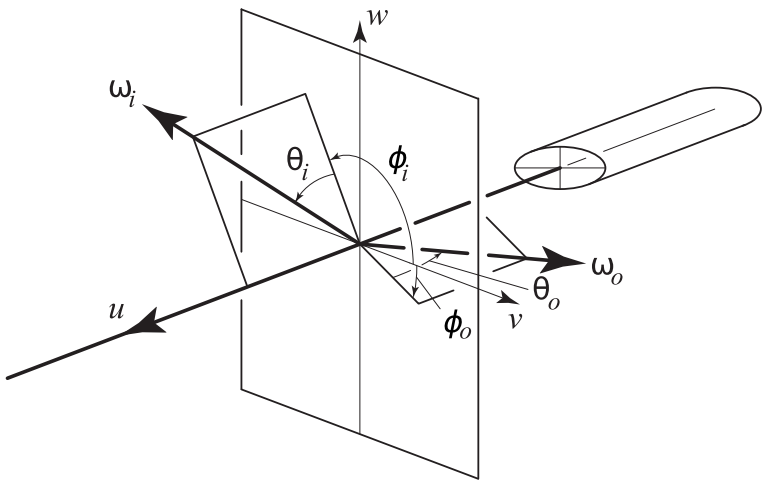


Figure 12: Geometry and notation. Image from [32]

3.2 Kajiya and Kay's Model

In 1989 Kajiya and Kay[22] presented the first technique to render furry and hairy surfaces. They separate the lighting model for a single hair into a diffuse Ψ_d and a specular component Ψ_s. The diffuse component is derived from Lambert's law [25], which describes the diffusely reflected radiance of an ideal matt surface as

$$k_d * n * l. \tag{25}$$

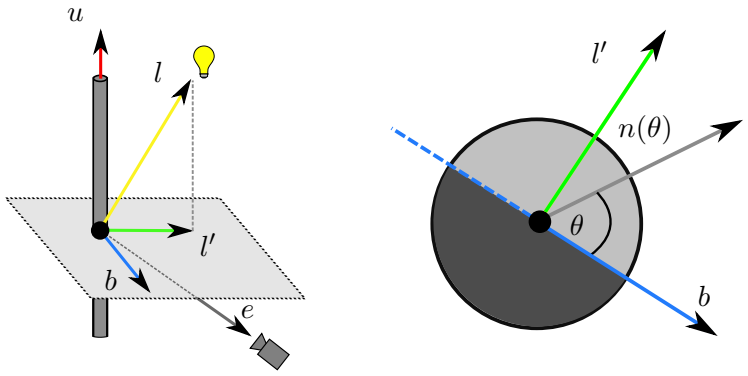
Here, k_d is the diffuse reflection coefficient. The derivation of Ψ_d is obtained by integrating the Lambert model over a cylindrical surface as in figure 13b. The orthonormal basis is given by the hair tangent u, the projection of the light direction on the normal plane l' and the orthogonal vector b (figure 13a). The vectors l' and b are given by

$$l' = norm(l - (u * l) * u) \tag{26}$$

$$b = l' \times u \tag{27}$$

Using these values, the diffuse component is computed as

$$\begin{aligned} \Psi_d =& k_d \int_0^\pi l * n \quad r d\theta \\ =& k_d r \int_0^\pi l * (b * cos\theta + l' * sin\theta) \quad d\theta \\ =& k_d r * l * l' \int_0^\pi sin\theta \quad d\theta \\ =& K_d * l * l' \end{aligned} \tag{28}$$



(a) Orthonormal basis. **(b)** Integration over the hemisphere.

Figure 13: The diffuse model of Kajiya - Kay. Images from [26]

where K_d represents all quantities independent of l and l' including the diffuse hair color. Substituting equation 26 into equation 28 yields:

$$\begin{aligned} \Psi_d =& K_d * l * norm(l - (u * l) * u) \\ =& K_d * sin(u, l) \end{aligned} \tag{29}$$

In addition to the diffuse component, Kajiya and Kay derive a specular component from the ad-hoc phong specular model [42]. The reflected light only depends on the longitudinal component of ω_o as the normal of a hair strand points in all directions perpendicular to u. The specular component L_s using the terms of figure 14 is given by

$$\begin{aligned} \Psi_s =& K_s * cos(e, e')^m \\ =& K_s * cos(\theta - \theta')^m \end{aligned} \tag{30}$$

where m is the specular exponent from the Phong lighting model. Here, the angles θ and θ' can be computed as:

$$\begin{aligned} \theta &= acos(u * l) \\ \theta' &= \pi - acos(u * e) \end{aligned} \tag{31}$$

Even today, the model of Kajiya and Kay is widely used in computer graphics, for example in the game Ryse by Crytek [52]. However, the model is extended using the approach of Scheuermann [51]. Scheuermann computes two highlights using the specular model of Kajiya and Kay, whereby the

first highlight is shifted towards the tip of the hair and the second highlight is shifted towards the root of the hair. The theory of this approach is based on the shading model of Marschner et al., which will be adressed in the next chapter.

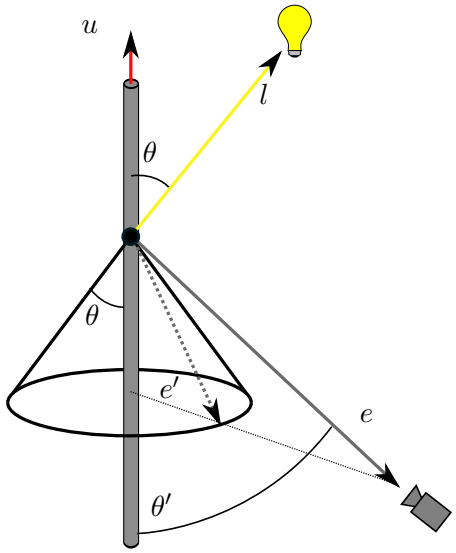


Figure 14: The specular model of Kajiya - Kay. Image from [26]

3.3 Marschner's Model

The first fully physically-based light scattering model for hair fibers was developed by Marschner et al. in 2003 [32]. They derive a *bidirectional scattering function* f_s from the general BRDF (see chapter 10) as

$$f_s(d\omega_i, d\omega_o) = f_s(\theta_i, \phi_i, \theta_o, \phi_o) = \frac{d\overline{L}_o(d\omega_o)}{d\overline{E}_i(d\omega_i)}, \tag{32}$$

where $\overline{L}$ is *curve intensity* and $\overline{E}$ is *curve irradiance*. Both values are normalized per unit length and are analogous to irradiance E and radiance L:

$$d\overline{E} = D * dE, \tag{33}$$

$$d\overline{L} = D * dL, \tag{34}$$

$$d\overline{E}_i(d\omega_i) = DL_i(d\omega_i)cos\theta_i d\omega_i, \tag{35}$$

where D is the diameter of the fiber. Given these equations the curve radiance is:

$$\overline{L}_o(\omega_o) = D \int_{4\pi} f_s(\omega_i, \omega_o) L_i(\omega_i) cos\theta_i d\omega_i. \tag{36}$$

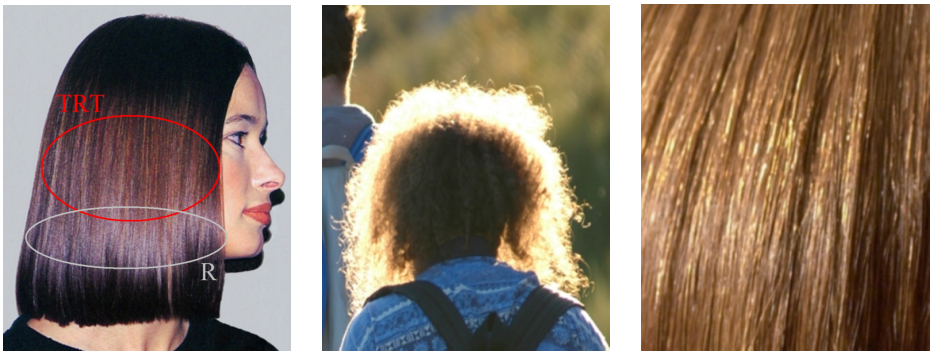
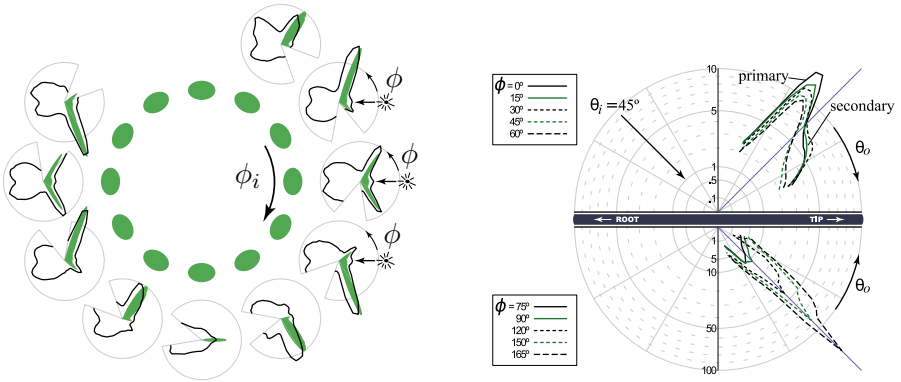


Figure 15: Left: Primary highlight (R) and secondary highlight (TRT). Middle: Strong forward scattering of light. Right: Glints / caustics. Images from [32].

Please note that the integral to compute $\overline{L}_o$ is defined over the entire sphere, while the rendering equation introduced in chapter 2.4.2 is defined over a hemisphere.

3.3.1 Scattering Measurements

Light scattering in the incidence plane of hair fibers was investigated first by Stamm et al. in 1977 [54]. Incidence plane means that the light source and the light detector are coplanar with the fiber. They measured that a primary specular peak occurs which is shifted several degrees away from the specular direction due to the outer cuticle surface of hair fibers (cf. figure 17). Moreover, a secondary specular peak in non black hair appears on the other side of the specular direction. Likewise, a study conducted by Bustard and Smith in 1991[8] reports that the primary peak preserves polarization, whereby the secondary peak is depolarized. They also observed azimuthal scattering in the normal plane producting strong peaks. Additionally, both studies show that the secondary peak is due to internal reflections within the fiber. Motivated by light scattering measurements of hair fibers by Stamm et al. and by Bustard and Smith, Marschner et al. conducted their own experimental and theoretical study. In particular, they measured light scattering in the incidence plane, the normal plane, and in the full 3D hemisphere. The measurements in the incidence plane follow the measurements conducted by Stamm et al. as well as those conducted by Bustard and Smith, and they were able to verify earlier results. Most noteworthy is that the specular highlight occurs approximately when $\theta_o = -\theta_i$. Scattering within the normal plane was measured in order to understand the complexity of the secondary highlight. Thus, the light source and the light detector were oriented perpendicular to the fiber. As a result, it can be stated that hair has a 180° rotational symmetry and is bilaterally sym-



(a) Scattering in normal plane.

(b) Scattering in hemisphere.

Figure 16: a) Scattering in the normal plane from blonde hair with eccentricity around 0.7:1. Green plots show the result of a Monte Carlo simulation. All illumination comes from the right side. b) Scattering in one hemisphere for a fixed incidence direction $\theta_i = 45°$. Images from [32]

metric in cross section [32]. Furthermore, there are two bright peaks which are called *glints*. Figure 16a shows plots of the measurement in the normal plane. All illumination comes from the right side. The green ellipses indicates the hair orientation and that hair fibers are not generally circular in cross section [47]. Here, the *eccentricity* a is around 0.7:1. Green plots show the result of a Monte Carlo simulation on the fiber model.

A full 3D hemispherical scattering measurement was also conducted during the experiments (see figure 16b). The angle of incidence was fixed at 45°: Three main components can be derived:

- Primary highlight (R): The primary specular peak shifted toward the root due to the tilted surface scales of the outer cuticle surface of hair fibers.
- Transmittance (TT): The strong transmittance component due to forward scattering from light colored hair.
- Secondary highlight (TRT): The colored secondary peak shifted toward the tip which strongly depends on ϕ and on the eccentricity a of the hair.

The letters R (for reflection) and T (for transmission) are used to describe the paths the light travels in order to produce these effects as shown in figure 17. Scattering angles without the tilted surface scales are indicated with dashed lines. Figure 15 shows some photographs of these effects.

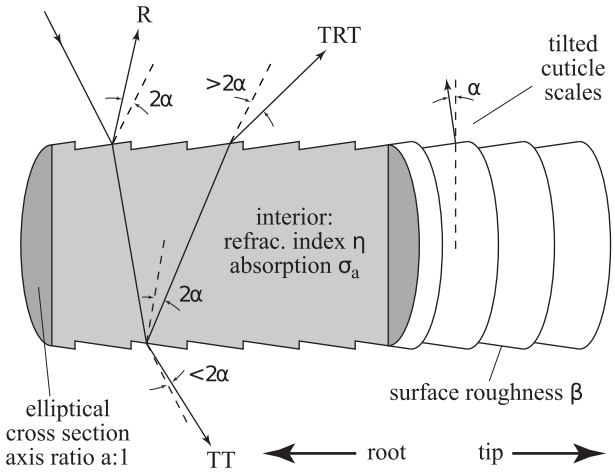


Figure 17: Longitudinal light scattering from hair. The scattering angles are shifted to certain degrees due to the tilted cuticle scales. All three light paths R, TT, and TRT are visible. Image from [32]

3.3.2 A Physically-Based Shading Model

Based on the introduced measurements, Marschner et al. developed the first fully physically-based hair shading model. As hair is a dielectric material, a strand is treated as a dielectric cylinder. The optical properties of the fiber are approximated using an index of refraction $\eta = 1.55$ and an uniform color absorption coefficient σ_a. In the following, a circular cross section of the hair is assumed which simplifies the math due to symmetry and other properties. However, chapter 3.3.3 gives a description of how to extend the model for elliptical cross sections.
The 4-dimensional scattering equation 32 can be expressed as a product of two 2-dimensional functions $M_p(\theta_i, \theta_o)$ and $N_p(\theta_d, \phi)$, where $p \in \{R = 0, TT = 1, TRT = 2\}$. The function $M_p(\theta_i, \theta_o)$ captures the θ dependence and is therefore called the *longitudinal scattering function* whereby $N_p(\theta_d, \phi)$ is called the *azimuthal scattering function* as it captures ϕ dependence. Thus, the scattering function f_s can be written as:

$$f_s(\theta_i, \phi_i, \theta_o, \phi_o) = \frac{1}{cos^2\theta_d} \sum_{p=0}^{2} M_p(\theta_i, \theta_o) N_p(\theta_d, \phi). \tag{37}$$

The division by $cos^2\theta_d$ accounts for the projected solid angle of the specular cone [32]. In addition, earlier work on scattering from fibers [1, 31, 37] has shown two important properties:

- A ray that enters a dielectric cylinder will always exit at the same inclination. Thus, a bundle of parallel incident rays coming from direction ω_i will scatter in a cone in direction $-\omega_i$. This property is

strongly related to the fact that scattering only occurs when $\theta_o = -\theta_i$ as described in chapter 3.3.1. This effectively reduces f_s from 4D to 3D.

- Moreover, it is sufficient to do a 2D analysis in the normal plane. This follows as a consequence of *Bravais's law*: The law of Bravais says that if the incident and refracted rays at a dielectric interface are projected in the normal plane, the projected rays still obey Snell's law (see equation 17) when using a corrected index of refraction $\eta'(\eta, \theta_d) > \eta$ [32]. Thereby, η' is given by

$$\eta'(\eta, \theta_d) = \frac{\sqrt{\eta^2 - sin^2\theta_d}}{cos\theta_d}. \tag{38}$$

The measurements by Marschner et al. revealed that reflected light will stay exactly in the specular cone. Additionally, the scattered distribution gets blurred to different degrees depending on the specific path p . The cuticle scales of the outer hair surface cause a tilt of the surface normals. As a result, the scattered lobes will not be centered on the specular cone. In order to account for these effects, Marschner et al. approximate the longitudinal scattering function with a unit-integral gaussian function

$$g(\beta, x, \mu) = \frac{1}{\beta\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\beta})^2} \tag{39}$$

with a standard deviation β and mean μ. As the different paths p cause different lobe shifts and widths, the longitudinal scattering functions M_p can be defined as:

$$\begin{aligned} M_p(\theta_i, \theta_o) &= g(\beta_p, \frac{\theta_i + \theta_o}{2} - \alpha_p, 0) \\ &= g(\beta_p, \theta_h - \alpha_p, 0). \end{aligned} \tag{40}$$

The terms α_p and β_p are the longitudinal shift and the longitudinal width for $p \in \{R, TT, TRT\}$. Typical values that are directly derived from the measurements can be found in table 1. In literature, one can find the longitudinal scattering function often simply expressed as $M_p(\theta_h)$ because M_p only depends on θ_h on the right side of the equation. The derivation of the azimuthal scattering function N_p is much more sophisticated. Figure 18 shows the geometry for scattering from a dielectric circle. As described before, the cross section of the hair is assumed to be circular and is extended for elliptical cross sections in chapter 3.3.3. According to the law of Bravais, it is sufficient to analyze azimuthal scattering in the normal plane only, when using the adjusted index of refraction η'. This simplifies the derivation as the math of scattering from a circular cross section is well known. It was first studied by Descartes to understand the appearance of

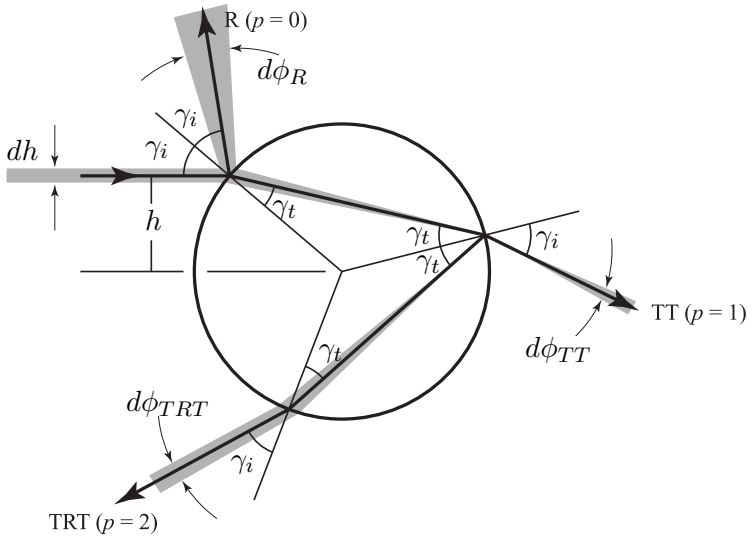


Figure 18: Scattering from a cross section. Image from [32]

rainbows (see Humphreys' *physics of the air* [18]). Thus, the angle of incidence in circular cross section is

$$\gamma_i = asin(h), \tag{41}$$

where $-1 < h < 1$ is the offset from the center of a unit circle. Then the angle γ_t of the refracted ray can be computed straightforwardly using equation 17 (Snell's law) and η':

$$1\gamma_t = asin(\frac{h}{\eta'}). \tag{42}$$

The exit angle $\phi(p, h)$ can be computed by following a ray and its interactions with the cross section as in figure 18:

$$\begin{aligned} \phi(p,h) &= 2p\gamma_t - 2\gamma_i + p\pi \\ &= 2p * asin(\frac{sin\gamma_i}{\eta'}) - 2\gamma_i + p\pi. \end{aligned} \tag{43}$$

In order to find all paths that contribute in a given direction ϕ one need to solve the equation

$$\phi(p,h) - \phi = 0. \tag{44}$$

The roots r of this equation yield γ_i. There is one root for $p = 0$ and for $p = 1$ and there are one or three roots for $p = 2$. However, solving this equation 44 is a non trivial and expensive task. Because of this, Marschner et al. approximated Snell's law with a cubic polynomial in order to compute γ_t. Hence γ_t can be expressed as:

$$\gamma_t = \frac{3c}{\pi}\gamma_i - \frac{4c}{\pi^3}\gamma_i^3 \tag{45}$$

where $c = asin(1/\eta')$. By using equation 45 in equation 43 , the exit angle $\phi(p,h)$ can be approximated with the cubic polynomial

$$\begin{aligned}\hat{\phi}(p,\gamma_i) &= 2p(\frac{3c}{\pi}\gamma_i - \frac{4c}{\pi^3}\gamma_i^3) - 2\gamma_i + p\pi \\ &= -\frac{8pc}{\pi^3}\gamma_i^3 + (\frac{6pc}{\pi} - 2)\gamma_i + p\pi.\end{aligned} \tag{46}$$

Finally, one can get all paths contributing in direction ϕ by solving for the roots of equation $\hat{\phi}(p,\gamma_i) - \phi = 0$. The next step is to compute the intensity of the scattered light. As the unit cross section represents a fiber diameter of 2, according to equation 33 the uniform irradiance that falls on the cross section is given by:

$$E(h)dh = \overline{E}/2dh. \tag{47}$$

Here, the width of the incident beam is noted as dh. Given the fact that the incident beam is scattered into the angular interval $d\phi_p$ (cf. figure 18), the curve radiance $\overline{L}_p$ for a specific path p, taking into account the path attenuation $A(p,h)$, can be computed by [32]

$$\overline{L}_p(\phi(h))d\phi_p = A(p,h) * \overline{E}/2dh, \tag{48}$$

which yields

$$\begin{aligned}\overline{L}_p(\phi(h)) &= A(p,h) * \frac{1}{2}\left|\frac{dh}{d\phi_p}\right|\overline{E} \\ &= A(p,h) * \left|2\frac{d\phi_p}{dh}\right|^{-1}\overline{E}.\end{aligned} \tag{49}$$

The derivative $\frac{d\phi_p}{dh}$ is given by:

$$\frac{d\phi_p}{dh} = \frac{-\frac{24pc}{\pi^3}\gamma_i^2 + \frac{6pc}{\pi} - 2}{\sqrt{1-h^2}}. \tag{50}$$

Attenuation $A(p,h)$ occurs because of volume absorption within the fiber and Fresnel reflection at the interfaces. Volume absorption strongly depends on the path length inside the fiber. Fortunately, the length per path projected to the normal plane s can be derived from figure 18:

$$s = 2 * cos\gamma_t. \tag{51}$$

Using this inner path length s, the attenuation due to volume absorption is:

$$T(h) = e^{-2*\sigma_a * cos\gamma_t}. \tag{52}$$

The attenuation due to Fresnel reflection is computed using the Fresnel equations (see chapter 2.5). However, the Fresnel equations have to use

the corrected index of refraction η' due to Bravais's law (see equation 38). Furthermore, Marschner et al. have shown that η' has to be used for the perpendicular component $F_\perp$ and another index η'' has to be used for the parallel component $F_\parallel$. The index η'' is:

$$\eta''(\eta, \theta_d) = \frac{\eta^2 cos\theta_d}{\sqrt{\eta^2 - sin^2\theta_d}} = \frac{\eta^2}{\eta'}. \tag{53}$$

Putting all this together yields the definition of $A(p, h)$:

$$\begin{aligned} A(0,h) &= F(1, \eta', \gamma_i) \\ A(1,h) &= (1 - F(1, \eta', \gamma_i)) * (1 - F(\eta', 1, \gamma_t)) * T(h) \\ A(2,h) &= (1 - F(1, \eta', \gamma_i)) * F(\eta', 1, \gamma_t) * (1 - F(\eta', 1, \gamma_t)) * T(h)^2. \end{aligned} \tag{54}$$

Finally, Marschner et al. conclude the azimuthal scattering function based on equation 49 to be:

$$N_p(\theta_d, \phi) = \sum_r A(p, h) \left| 2 \frac{d\phi_p}{dh} \right|^{-1} \tag{55}$$

where r is the roots of equation 44.
This approach works well for the R- and TT - components. However, the TRT-component produces singularities in f_s with infinite intensity which are called *caustics*. In order to remove these unrealistic phenomena, Marschner et al. proposed algorithm 1 that first removes the caustic from N_{TRT} and then replaces it by Gaussians representing the same energy centered over the caustics positions. Caustics appear because the function $\phi(p, h)$ is smooth which results in a fold at the transition from one to three roots [32, 64]. Descartes has shown that this fold occurs when $\frac{d\phi}{dh} = 0$, hence if

$$h^2 = (4 - \eta'^2)/3. \tag{56}$$

Obviously this equation is symmetric, resulting in two extrema. Using this knowledge, the angle γ_c at which the caustic appears can be computed as:

$$\begin{aligned} \frac{d\phi}{dh} = 0 \Leftrightarrow \\ \frac{-\frac{24*2*c}{\pi^3}\gamma_c^2 + \frac{6*2*c}{\pi} - 2}{\sqrt{1 - h^2}} = 0 \Leftrightarrow \\ \frac{48 * c}{\pi^3}\gamma_c^2 = \frac{12 * c}{\pi} - 2 \Leftrightarrow \\ \gamma_c = \pm\sqrt{\frac{\frac{12*c}{\pi} - 2}{\frac{48*c}{\pi^3}}}. \end{aligned} \tag{57}$$

Algorithm 1 is the original algorithm from Marschner et al. adjusted with some additions to make it more easy to implement. The terms $w_c, k_G, \Delta\eta'$

and Δh_M are user adjustable parameters. They represent the azimuthal width of caustic, glint scale factor, fade range for caustic merge, and the caustic intensity limit. Common values are listed in table 1. Line 2 accounts for the fact that equation 56 is only defined for values $\eta' < 2$. For greater values, the Gaussian lobe is inserted at $\phi_c = 0$ (line 10) and from there on smoothed out over a short range of incidence angles $\Delta\eta'$. The function *smoothstep(a,b,x)* from line 9 implements a cubic hermite interpolation between a and b.

Algorithm 1 Smooth the caustic in N_{TRT} (cf. [32]

1: **procedure** $N_{TRT}(\theta_d, \phi, \eta', w_c, k_G, \Delta\eta', \Delta h_M)$
2: **if** $\eta' < 2$ **then**
3: Compute h_c and γ_c using equation 41 and 57
4: Δh = min(Δh_M, $2\sqrt{2w_c/|\frac{d^2\phi}{d^2h}(h_c)|}$)
5: t = 1
6: $\phi_c = \hat{\phi}(2, \gamma_c)$
7: **else**
8: $\Delta h = \Delta h_M$
9: $t = 1 -$ smoothstep$(2, 2 + \Delta\eta', \eta')$
10: $\phi_c = 0$
11: **end if**
12: $L = N_{TRT}(\theta_d, \phi)$
13: $L = L * (1 - t * g(\phi - \phi_c, w_c, 0)/g(0, w_c, 0))$
14: $L = L * (1 - t * g(\phi + \phi_c, w_c, 0)/g(0, w_c, 0))$
15: $L = L + t * k_G * A(2, h_c) * \Delta h * (g(\phi - \phi_c, w_c, 0) + g(\phi + \phi_c, w_c, 0))$
16: return L
17: **end procedure**

3.3.3 Scattering in Elliptical Fibers

This chapter describes how the physically-based shading model presented, that is based on smooth circular cylinders, can be extended to elliptical cross sections. Unfortunately, a simple analytical solution is not possible [32]. However, Marschner et al. have shown that a change of the index of refraction can have a qualitative effect similar to changing the eccentricity a. They have stated that for mild eccentricities $0.85 \leq a \leq 1$ it is sufficient to use an index of refraction η_1^* when ϕ_h is aligned with a principal axis. In total, the corrected index of refraction for the TRT - component is:

$$\eta^*(\phi_h) = \frac{1}{2}((\eta_1^* + \eta_2^*) + cos(2\phi_h)(\eta_1^* - \eta_2^*)) \tag{58}$$

with

$$\begin{aligned} \eta_1^* &= 2(\eta - 1)a^2 - \eta + 2 \\ \eta_2^* &= 2(\eta - 1)a^{-2} + \eta + 2 \end{aligned} \tag{59}$$

3.3.4 Implementation

The implementation of the model of Marschner et al. follows the ideas of Nguyen and Donnelly [38]. As the computations of the longitudinal scattering functions M_p and the azimuthal scattering functions N_p are very expensive, they suggest pre-computing them in three 2-dimensional look up textures. Thereby the first texture stores the one dimensional functions M_R, M_{TT}, and M_{TRT} in the RGB channels. Therefore, the texture is parameterized with $sin(\theta_i)$ and $sin(\theta_o)$. Using the sine instead of the angle itself helps avoiding inverse trigonometric functions within the GLSL shader. For the azimuthal functions N_p, two lookup textures are required as N_{TT} and N_{TT} are 3-dimensional functions due to absorption. Thus, N_R and N_{TT} are stored in the second texture and N_{TRT} in the third. As N_p depends on θ_d and ϕ, the textures are parameterized with $cos(\theta_d)$ and $cos(\phi)$. In addition to that, θ_d can also be stored in the first lookup texture as it is a function of θ_i and θ_o. Finally, the model of Marschner et al. can be applied within GLSL as shown in algorithm 2. Lines 2 and 3 simply compute the sine of θ_i and θ_o whereby lines 3 and 4 projects them into the normal plane. With those projected vectors, line 6 can easily compute the cosine of ϕ. Lines 8 - 10 perform the actual lookups using the computed values. Thereby they transform the sine and cosine values $\in [-1, 1]$ to uv space $\in [0, 1]$ by multiplying with 0.5 and adding 0.5. Finally, line 12 computes equation 37. The term $1/cos^2\theta_d$ is already applied to the M lookup table in order to save computation time in the shader.

Algorithm 2 Applying the shading model of Marschner et al. using precomputed lookup textures.

procedure MARSCHNER(ω_i, ω_o, u, M, N_{LUT0}, N_{LUT1})
  $sin(\theta_i) = \omega_i * u$
  $sin(\theta_o) = \omega_o * u$
  $\omega_i^{\perp} = \omega_i - sin(\theta_i) * u$
  $\omega_o^{\perp} = \omega_o - sin(\theta_o) * u$
  $cos(\phi) = (\omega_o^{\perp} * \omega_i^{\perp}) / \sqrt{(\omega_o^{\perp} * \omega_o^{\perp}) * (\omega_i^{\perp} * \omega_i^{\perp})}$

  $(M_R, M_{TT}, M_{TRT}, cos(\theta_d)) = M(sin(\theta_i) * 0.5 + 0.5, sin(\theta_o) * 0.5 + 0.5)$
  $(N_R, N_{TT}) = N_{LUT0}(cos(\phi) * 0.5 + 0.5, cos(\theta_d) * 0.5 + 0.5)$
  $N_{TRT} = N_{LUT1}(cos(\phi) * 0.5 + 0.5, cos(\theta_d) * 0.5 + 0.5)$

  return $M_R * N_R + M_{TT} * N_{TT} + M_{TRT} * N_{TRT}$
end procedure

3.4 An Artist Friendly Hair Shading Model

The model of Marschner et al. adds important detail to the overall appearance of hair in rendering. However, the model is difficult to control for artists as it can only be controlled using physically-based parameters. In addition, these parameters strongly depend on each other, which is undesirable from an artist's point of view. As an example, the intensity of a highlight must be reduced by the model in order to preserve energy if the artist increases the width of a highlight (see figure 19). It is difficult for artists to guess parameters of such a shading model [36]. Because of this, much production work in motion pictures uses older ad-hoc approaches as they offer more control to artists [49]. To account for this, Sadeghi et al. [49] developed an artist friendly hair shading system which uses meaningful artist friendly controls. These controls were defined in strong collaboration with a team of artists working for the Walt Disney Animation Studios. They came up with the following control parameters for the shading model:

R: Color, intensity, longitudinal position, longitudinal width.

TT: Color, intensity, longitudinal position, longitudinal width, azimuthal width.

TRT minus glints: Color, intensity, longitudinal position, longitudinal width.

Glints: Color, intensity, frequency of appearance.

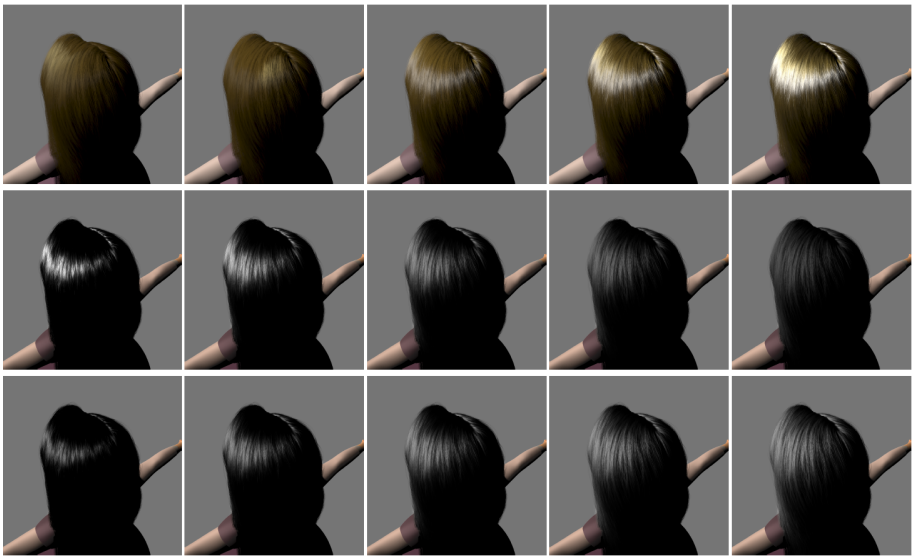


Figure 19: Top row: Varying the index of refraction affects all visual components of the appearance (coupled parameters). Middle row: The intensity and the width of the highlight are coupled. Thus, the intensity of the highlight decreases as its width increases. Bottom row: The same but with decoupled parameters from the artist friendly hair shading model. Image from [49].

It can easily be seen that these artist friendly controls align nicely with the physically-based model of Marschner et al. (cf. chapter 3.3.1). The overall goal of Sadeghi et al. was to approximate the scattering function f_s (see equation 37) with a pseudo scattering function f'_s as:

$$f'_s(\theta_i, \phi_i, \theta_o, \phi_o) = \frac{1}{cos^2\theta_d} \sum_{p=0}^{2} C_p I_p M'_p(\theta_i, \theta_o) N'_p(\phi), \tag{60}$$

where $p \in \{R, TT, TRT\}$ and C_p and I_p are the color and intensity of component p, respectively. The longitudinal pseudo scattering function M'_p is very similar to the one used by Marschner et al. M'_p is

$$M'_p(\theta_i, \theta_o) = g'(\beta_p, x, 0), \tag{61}$$

where $g'(\beta, x, \mu)$ is the Gaussian function g but with unit height. Because of their complexity, the azimuthal pseudo scattering functions N'_p need to be defined separately. The azimuthal scattering function for the primary highlight N'_R follows the work of Kim [23, p. 84]:

$$N'_R = cos(\phi/2). \tag{62}$$

Thus when plotted, N'_R is shaped like a flipped heart as shown in figure 20.

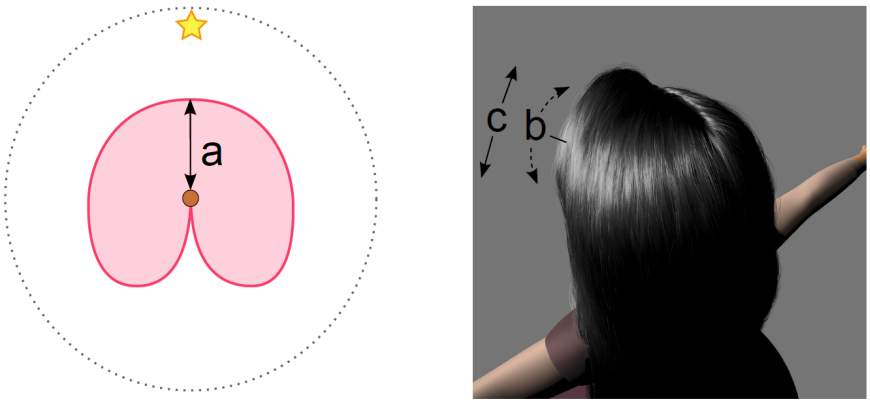


Figure 20: Primary highlight's azimuthal control parameters (left) and a frontlit rendering (right) where (a) is the intensity I_R, (b) is the longitudinal shift α_R, and (c) is the longitudinal width β_R^2. Images from [49].

Transmission N'_{TT} is a sharp forward directed lobe which is approximated using a Gaussian with unit height:

$$N'_{TT}(\phi) = g'(\delta_{TT}^2, \pi - \phi, 0). \tag{63}$$

Here, δ_{TT}^2 is the azimuthal width. See figure 21 for a visualization of N'_{TT} and all its parameters.

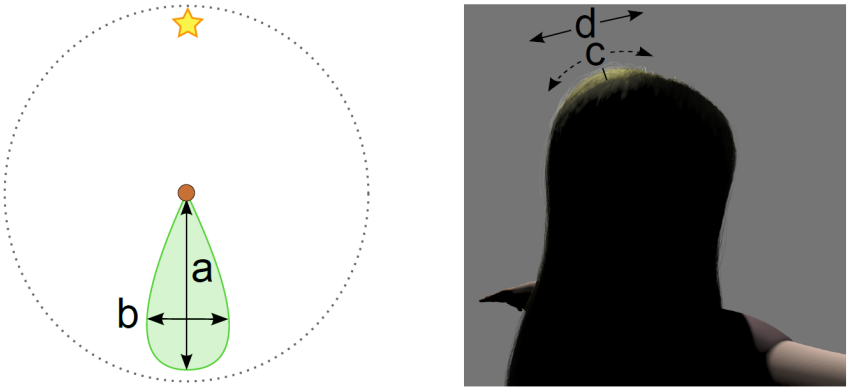


Figure 21: Transmission's azimuthal control parameters (left) and a backlit rendering (right) where (a) is the intensity I_{TT}, (b) is the azimuthal width δ_{TT}^2, (c) is the longitudinal shift α_{TT}, and (d) is the longitudinal width β_{TT}^2. Images from [49].

Finally, the azimuthal scattering function for the secondary highlight N'_{TRT} is split into two functions N'_{TRT-G} and N'_G to account for the glints. As

glints can be seen as two sharp peaks, they are simply modeled as a Gaussian with a random shift. In total, these functions are defined as:

$$\begin{aligned} N'_{TRT}(\phi) &= N'_{TRT-G}(\phi) + N'_G(\phi), \\ N'_{TRT-G}(\phi) &= cos(\phi/2), \\ N'_G(\phi) &= I_G g'(\delta_G^2, G_{angle} - \phi, 0), \end{aligned} \tag{64}$$

where δ_G^2 is the azimuthal width of the glints, I_G is the glint intensity, and the half angle between two glints $G_{angle} \in \{30°, 45°\}$ is randomized per hair strand to give some random appearance to the glints. Figure 22 gives an overview over N'_{TRT} and its parameters.

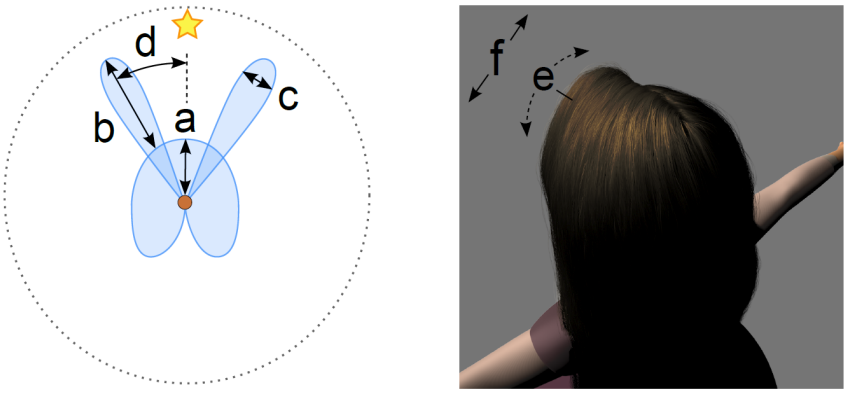
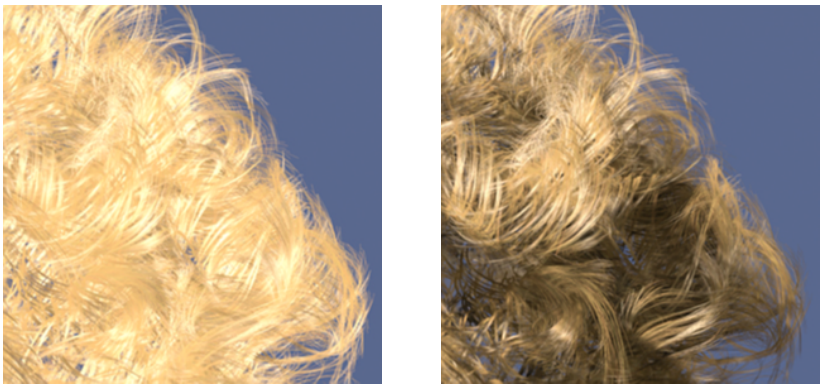


Figure 22: Secondary highlight's azimuthal control parameters (left) and a frontlit rendering (right) where (a) is the intensity I_{TRT}, (b) is the glint intensity I_G, (c) is the azimuthal width of glints δ_G^2, (d) is the half angle between two glints G_{angle}, (e) is the longitudinal shift α_{TRT}, and (f) is the longitudinal width β_{TRT}^2. Images from [49].

4 Shadowing



(a) Without shadow. **(b)** With shadow.

Figure 23: Hair rendered without and with proper shadowing. Image from [29]

Shadows are a crucial factor for all kinds of volumetric objects such as hair, fur, clouds, or smoke. They have a major influence on their appearance and illumination. Figure 23 shows hair rendered with and without proper shadows. One very common technique for computing shadows in a real-time computer graphics environment is the use of depth-based shadow maps [58]. However, shadow maps only provide a binary decision whether or not an object receives light. This yields to unpleasant results when rendering hair (see chapter 6.2). Instead, when dealing with such semi-transparent geometry, the transmittance function $\tau(x)$ has to be evaluated. The transmittance function measures how much light is transmitted from the light source to a point x within the hair volume. For this, it computes an exponential falloff using the opacity $O(x)$. The values are given by [24]:

$$\tau(x) = e^{-O(x)}, \tag{65}$$

$$O(x) = \int_0^{x_z} \rho(z')dz', \tag{66}$$

where ρ is a density (or extinction) function along the path [6]. This density function can be simply expressed as the transparency value α_s of the geometry at point x.

4.1 Opacity Shadow Maps

Kim and Neumann introduced a two pass algorithm called *opacity shadow maps* [24], one of the first real-time capable approaches to render appropriate shadows for hair [14, p. 246]. As in figure 25a, they use a set of parallel

opacity maps oriented perpendicular to the light's direction. However, it is worth noting that the slicing does not have to be uniform. In contrast to a shadow map, an opacity map stores $O(x)$ per Texel. For rendering the first pass, the depth buffer is disabled in order to not lose information due to early depth tests and additive hardware blending is enabled. Using this, the integral of equation 66 can be approximated with a sum over α_s. Then, each fragment contributes its transparency α_s to the nearest opacity map along the shadow ray. In the second pass, the opacity values from adjacent opacity maps are sampled using common shadow map sampling techniques like percentage closer filtering [46]. It is then straight forward to do a linear interpolation of these two values to get an approximation of $O(x)$ from which the transmittance function 65 can be computed. The opacity reaches infinity for opaque geometry. Because of this, Kim and Neumann introduced a scaling constant κ such that $e^{-\kappa} = 2^{-d}$, where d is the number of bits per pixel. As an example, using a 8 bit alpha buffer yields to $\kappa \approx 5.56$. Consequently, equation 65 becomes:

$$\tau(x) = e^{-\kappa * O(x)}. \tag{67}$$

It is obvious that the quality of opacity shadow maps highly depend on the number of slices one uses. Using too few slices will produce strong layering artifacts as in figure 24. Due to this, an implementation should use a least 16 layers in order to achieve convincing results. The implementation of Nguyen and Donnelly is able to generate all 16 opacity maps in a single render pass by using the GPU in an efficient way [38]: As $O(x)$ is a one dimensional value, one can store 4 opacity maps within one RGBA texture. In addition, using a GPU feature called multiple render targets allows rendering to 4 (or more) different textures simultaneously.

4.2 Deep Opacity Maps

The goal of deep opacity mapping is to reduce the layering artifacts produced by opacity shadow maps and to reduce the number of layers and thereby the performance of the algorithm (see figure 24). To achieve this, Yuksel and Keyser [63] propose an algorithm that warps opacity layers to the shape of the hair structure as in figure 35a. Their algorithm has three steps:
First, the hair mesh is rendered to a depth map as seen from the light source to capture the shape of the hair. As the depth map is only used to compute the shape of the opacity maps, an 8-bit depth map is sufficient. Using the entry z_0 from the depth map, one can introduce K layers z_0+d_{k-1} to z_0+d_k, where $d_0 = 0, d_{k-1} < d_k$ and $1 \leq k \leq K$. As with opacity shadow maps, the slicing does not have to be uniform.

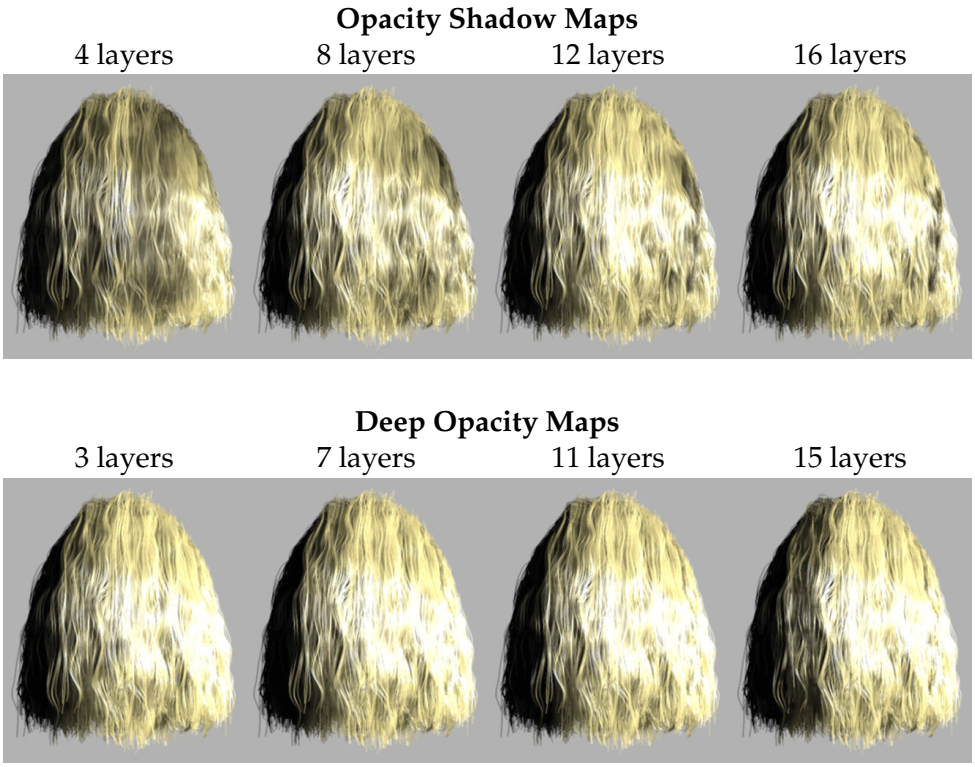
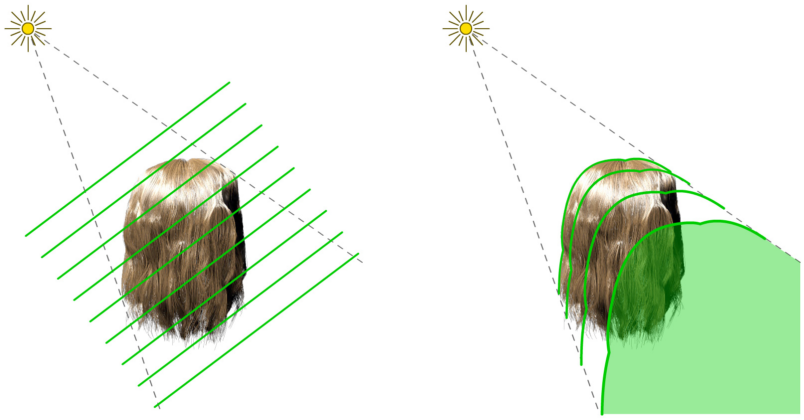


Figure 24: Top row: Using too few layers with opacity shadow maps generates strong layering artifacts. Bottom row: Use of deep opacity maps produces good quality with just 3 layers. Images from [63].

Second, the deep opacity map is generated. For this, additive hardware blending is enabled and the depth test is disabled. The layer in which a fragment falls can be computed efficiently on the fly within the fragment shader using the depth map from the first pass. Now, in order to compute the integral of the transmittance function 67, the fragment can contribute its transparency α_s to its layer and to the layers behind that one. It is essential that texture filtering for the depth map is disabled during this pass. As before, one can store up to 4 layers in one RGBA texture and can compute multiple layers using multiple render targets. However, figure 24 shows that using just three layers is sufficient to produce a pleasant result. One problem is that using a small number of layers makes it difficult to assign every hair to a layer. For instance, the point in the green area of figure 35a lies behind the last layer. The authors propose three different options if points lie beyond the last layer: Ignoring the points so that they don't cast shadows at all, add them to the last layer, or increase the last layer to cover the complete hair volume. Unfortunately, increasing the layer size yields to less accuracy within that layer. As transmittance at that point should be near to zero anyway, adding them to the last layer usually give reasonable

results. The *third* and last step is to render the hair using the deep opacity map. Here, the algorithm is just the same as for opacity shadow maps.



(a) Opacity Shadow Map. **(b)** Deep Opacity Map.

Figure 25: Opacity shadow maps using a set of parallel opacity maps, whereby deep opacity maps uses fewer layers matching the shape of the hair. The green area lies behind the last layer and is in total shadow. Image from [63]

4.3 Approximated Deep Shadow Mapping

Deep opacity as well as opacity shadow maps are suitable techniques to compute proper shadows for hair rendering. However, when it comes to real-time rendering of hair in video games, these techniques might be much to expensive. Eventually, hair is only a minor part of the overall video game and must be rendered as fast as possible in just a few milliseconds. Because of this, AMD came up with *approximated deep shadow mapping* for their SDK *TressFX* [28, 33]. The first video game that uses this approach was *Tomb Raider* by Square Enix from 2013 [27]. The basic idea is rather simple as it just uses a standard depth based shadow map. While shading, each fragment shader invocation estimates the number of hairs n that lie in between the current fragment and the light source by:

$$n = \frac{max(0, d_f - d_s)}{w}, \tag{68}$$

where w is the width of the hair, d_f is the distance from the fragment to the light source, and d_s is the distance from the hair surface to the light source, respectively. The hair surface is just given by the depth based shadow map. However, as the distance between the current hair fragment and the hair surface $d_f - d_s$ involves floating point arithmetic, it is recommended to increase n by one as soon as the distance becomes greater than some ϵ. In the

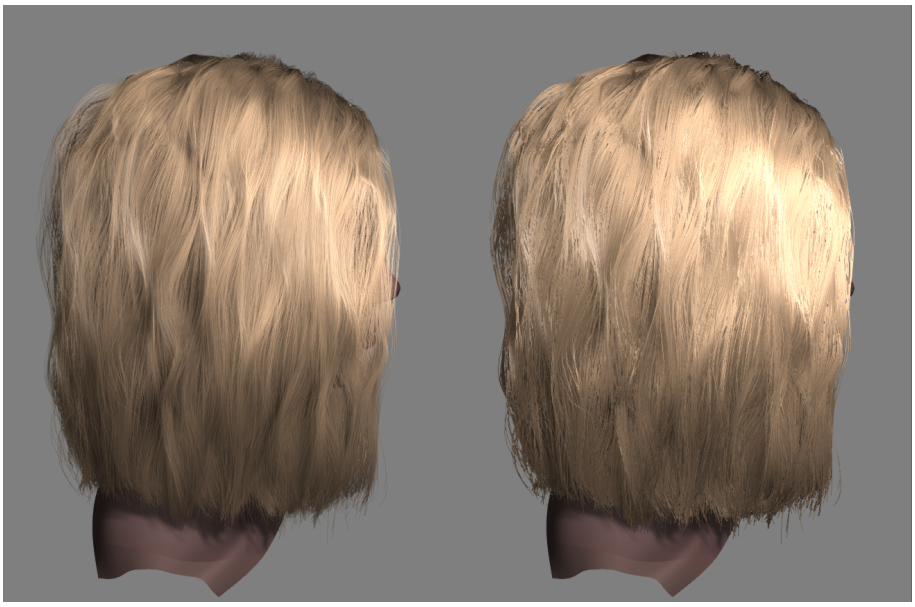
end, one can compute the attenuation of light, and therefore the strength of the shadow by:

$$inShadow = clamp((1 - \alpha_s)^n, 0, 1). \tag{69}$$

Here, the function $clamp(x, a, b)$ returns the value as $a \leq x \leq b$.

5 Transparency

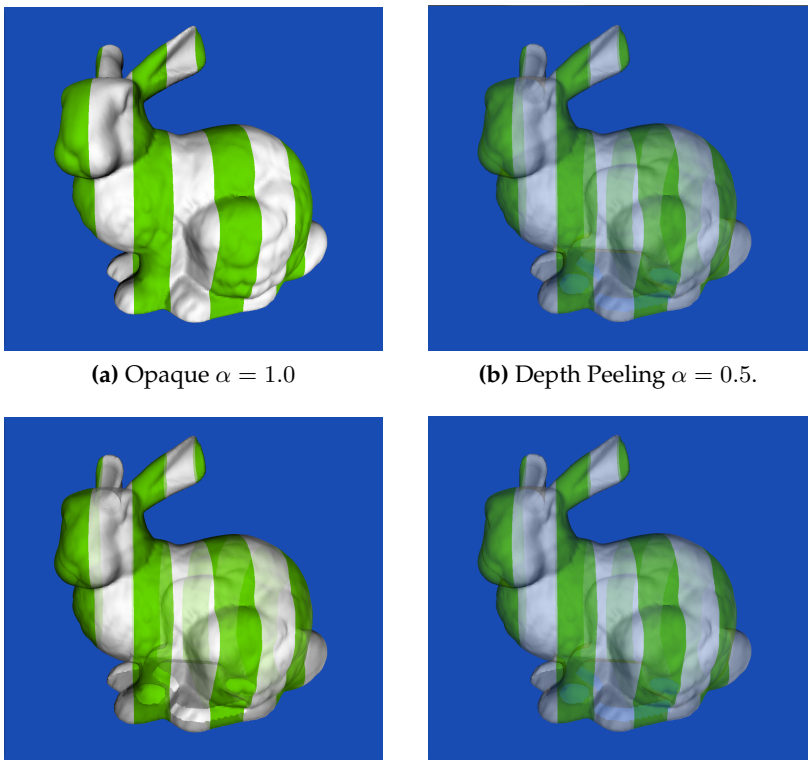
The discussion about physically-based hair rendering has already shown that hair is semi-transparent. Thus, light scatters in the hair volume strongly effecting lighting and shadowing. Beside that, the geometry of hair has to be rendered taking transparency into account. Without a proper transparency technique, hair appears dull and aliased (cf. figure 26).



$\alpha = 0.5$ $\alpha = 1.0$

Figure 26: Hair rendered with and without transparency. Transparency increases the smoothness of the hair so that it appears less dull and aliased.

In real-time computer graphics, the common approach to solving transparency is *alpha blending*. The order of the scene is very important for appropriate alpha blending. So either the scene is rendered from back to front while using the *over-operator* or it is rendered from front to back using the *under-operator* [44]. For both approaches, the scene must be sorted every frame along the view direction. As such sorting might be suitable for a few transparent objects, it becomes unacceptably expensive when it comes to hair rendering. Additionally, it is not sufficient to sort all hair strands along the camera direction only because the individual strands are highly overlapping. Thus, sorting per fragment becomes necessary. Techniques related to the term *order independent transparency* (OIT) try to solve the transparency problem without the need to sort the scene. A prominent approach to solve OIT is *depth peeling* [15, 30] and related techniques like *dual depth peeling* [5]. For depth peeling, the scene is rendered multiple times. The first render stores only the foremost fragments (the first *peel*). Using the depth



(a) Opaque $\alpha = 1.0$

(b) Depth Peeling $\alpha = 0.5$.

(c) WBOIT $\alpha = 1.0$.

(d) WBOIT $\alpha = 0.5$.

Figure 27: Renderings of the *Stanford bunny*: (a) fully opaque, (b) depth peeling produces the correct result, (c) WBOIT fails to render the bunny opaque, and (d) WBOIT renders a good approximation of depth peeling. All images were generated using the NVIDIA GameWorks™ OpenGL samples [17].

buffer from the first render, the second render can discard the foremost fragments while storing the next deeper fragments, and so on. For scenes with a moderate depth complexity, depth peeling is a sensible approach (cf. figure 27b). However, a great number of depth peels is necessary for hair as it has a very high depth complexity. Due to this, depth peeling is not suitable for hair rendering. Another area of OIT is approximation techniques like *sort-independent alpha blending* [35] or *weighted blended order independent transparency* (WBOIT) [34]. These techniques can render transparent geometry very quickly and the quality might be acceptable for several use cases (cf. figure 27d). Unfortunately, WBOIT lacks quality and consistency while moving the camera. Moreover, even a fully opaque object is rendered with a high transparency as shown in figure 27c.

5.1 A-Buffer

Yang et al. presented the first real-time concurrent *per-pixel linked list* (PPLL) construction on the GPU [60] in 2010 which heavily relies on modern GPU features like *atomic counters* and (*image load store*) for arbitrary read and write to images. It basically is a hardware implementation of the old *A-Buffer*, which provides a list of fragments per pixel used for effects like transparency or anti-aliasing [10]. To construct a PPLL, two different data structures are necessary: the *head pointer buffer* and the *node buffer*. Thereby, the head pointer buffer is a 2D texture as big as the render target storing pointers inside the node buffer. The node buffer itself is some GPU based storage type big enough to store as many fragments as needed (3D textures or shader storage buffer objects). Using the PPLL approach for hair rendering yields to a three step algorithm: In the *first step*, the head pointer buffer is cleared with the value -1, which basically indicates a *null pointer*. *Secondly*, the node buffer gets filled as in algorithm 3. For both of these, depth and color writing are disabled, since all outputs are routed into the node and head pointer buffer. In addition, depth testing against the depth map of the opaque scene is enabled in order to handle occlusion correctly. An atomic counter tracks the current global position in the head pointer buffer (line 2). This is possible because, as the name suggests, atomic counters provide functionality to atomically read, increment, and decrement an unsigned integer in a shader program. For each fragment, the current entry in the head pointer buffer is atomically exchanged with the incremented atomic counter (line 6). This yields the link to the next element in the linked list. The link is -1 if no element got inserted yet, thus if the current fragment is the first element in the linked list of that pixel. Lines 3 and 4 are necessary to avoid an overdraw of the node buffer. This can happen if more fragments are written than memory was allocated. In the end, the tangent, depth and the link are stored in the node buffer.

Algorithm 3 Fill the A-Buffer in the Fragment Shader

```
1: procedure FILL A-BUFFER(vector2 p, uint maxIndex)
2:     index = atomicIncrement(counter)
3:     if index > maxIndex then
4:         return
5:     end if
6:     link = atomicExchange(headBuffer,p,index)
7:     Store link, tangent, and depth in nodeBuffer[index]
8: end procedure
```

The *last step* is to resolve the PPLL to render the hair. For this, a full screen quad is rendered where each fragment shader invocation accesses one entry in the head pointer buffer. If the entry is not equal to -1, at least one

entry in the underlying linked list is stored. Now, the PPLL can be easily traversed using the link information stored in the node buffer. However, as the overall goal is to render the hair as transparent geometry, the PPLL have to be sorted on the GPU first. Afterwards, the fragments can be blended using common alpha blending.

5.2 K-Buffer

Complex and dense hairstyles can have dozens of fragments per pixel. Rendering all of those using a PPLL is very costly and most often not suitable. Anyhow, hair is only semi-transparent and only the K foremost fragments might be visible. Thus, only the K foremost fragments must be blended together and the rest can be ignored. Techniques and data structures taking this idea into account are denoted as a *K-Buffer*. The term K-Buffer was introduced first by Callahan et al. [4, 9]. However, their approach assumes already partially sorted geometry which makes it useless for hair rendering. Fortunately, a K-Buffer can be easily created on the fly using the A-Buffer as done by Yu et al. [61]. For this, the first K entries of the A-Buffer are stored in a local array on the GPU. Afterwards, the complete A-Buffer is traversed whereby each element is compared to the local array. If any smaller element is found, it is exchanged with the largest element in the local array. As this technique still requires the creation of an A-Buffer, memory consumption is still very high. Therefore, Vasilakis et al. introduced the K$^+$-buffer which stores the K nearest fragments [55, 56] only. To do so, they propose to use one of two array-based data structures: *max-array* and *max-heap*. Both track the depths of all currently stored fragments. The first entry in the max-array is always the fragment with the greatest depth whereby the rest is unordered. In contrast to this, the max-heap is a binary tree where each node is larger or equal to the depth of its child nodes. In their paper, an easy array based implementation is given [55]. As graphic devices may process fragments in parallel, even if they relate to the same fragment coordinates, *read-modify-write* memory-hazards might appear. To solve this issue, modern GPU functionalities such as *INTEL_fragment_shader_ordering* [16] for Intel GPUs or *NV_fragment_shader_interlock* [7] for Maxwell GPUs can be used. For other hardware, the authors propose per-pixel *binary semaphores* using a 32-bit unsigned integer texture initialized with 0.

Algorithm 4 Mutex in OpenGL using binary semaphores (black), Intel's fragment shader ordering (red), or NVIDIA's fragment shader interlock (blue) [56]

```
1:  procedure MUTEX
2:      beginFragmentShaderOrderingINTEL()
3:      beginInvocationInterlockNV()
4:      while true do
5:          if !imageAtomicExchange(t,p,1) then
6:              {enter critical section}
7:              imageStore(t,p,0)
8:              break
9:          end if
10:     end while
11:     endInvocationInterlockNV()
12: end procedure
```

Algorithm 4 shows the implementation of a mutex in OpenGL using either Intel's fragment shader ordering, NVIDIA's fragment shader interlock, or the binary semaphores as proposed by [55]. Here, line 5 atomically exchanges the current value stored in the unsigned integer texture with 1. The if statement only becomes true if the returned value is 0. Thus, if no other invocation has entered the critical code section yet. All other invocations continue spinning until the lock is free again. Please note that the critical code in line 6 is executed for all three methods.

5.3 A Novel Approach: DBK-Buffer

During this thesis, a novel approach for rendering semi-transparent hair was developed in order to compensate the drawbacks of other methods: the depth-based K-Buffer (*DBK-Buffer)*. The main goals were:

- Find only the foremost fragments to be more memory efficient than the A-Buffer.
- Do not use a GPU mutex as they either need special hardware or are very inefficient (as binary semaphores).
- Preserve a quality similar to other approaches while not being slower.

The idea is to introduce a layer which works a bit like the layering approach of deep opacity mapping (see chapter 4.2). The hair is first rendered into a depth map from the camera's point of view. Afterwards, the depth map is shifted along the view direction by a user controllable amount. Every fragment that lies within that layer is stored in the PPLL using algorithm 3. All

other fragments are just discarded. Because of this, a huge amount of memory is saved as shown in chapter 6.3.3. One drawback is that the technique might miss important fragments that do not lie within that layer. For instance, a single hair strand might stick out of the hair volume in a way that following fragments have a distance too far away to lie within the layer. Because of this, the hair is rendered once more as opaque geometry in such a way that every fragment that does lie within the layer is discarded. The PPLL is then blended over this opaque layer. Another drawback is that the user has to use a proper depth shift and must allocate a sufficient amount of memory in order to avoid overdraws. However, the A-Buffer also has the challenge of allocating enough memory. Nevertheless, the advantage of significantly less memory in comparison to the A-Buffer in combination with a better performance than the mutex based K-Buffer does make this novel approach very valuable for certain scenarios. A much more sophisticated and detailed evaluation is done in chapter6.3.

5.4 Random Subsets

Older GPUs that don't support at least OpenGL version 4.2, and therefore don't support important features like atomic counters or image load store, are not able to execute PPLL in any of the presented forms. For systems using such hardware, or when a more easy implementation is necessary, blending random subsets might be a good way to go. Therefore the hair strands are split in K random subsets. Each of those is then rendered into textures using deferred rendering. Then it is possible to shade each subset and blend them from front to back or vice versa. One can also split the hair strands into layers starting from the skull to the outer hair. However, this approach might not work for arbitrary hairstyles and meshes. For this reason random subsets were implemented instead.

6 Results and Evaluation

In this chapter results of the implementation are shown and evaluated. For each described feature, a comprehensive analysis of the visual quality and the computational performance was conducted. OpenGL timer queries are used for performance measurements, as computations are mainly done on the GPU. The test system consists of an Intel® Core™ i7-4790K quad-core CPU running at 4 GHz, 16 GB RAM, and an NVIDIA GeForce® GTX 960 GPU with 4 GB of VRAM. The viewport resolution for all images is 656×861 pixels, which was embedded within a Qt user interface as shown in figure 45. The hair models used are courtesy of Cem Yuksel and can be downloaded on his web page [62]. They are basically a set of splines that can be loaded using the library provided. Figure 28 gives an overview of the hair models used in this thesis. In addition to the hair models, the female head model is courtesy of Murat Afshar and can also be downloaded on the web page.

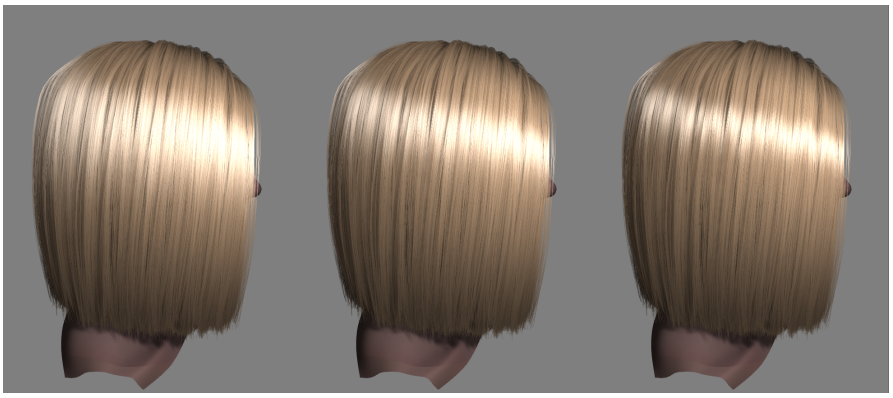
Straight	*Wavy*	*Curly*
50.000 strands	50.000 strands	50.000 strands
1.250.000 vertices	2.450.000 vertices	3.441.580 vertices

Figure 28: The different hair models (*Straight*, *Wavy*, *Curly*) used throughout this thesis rendered with the shading model of Marschner et al. All three consist of 50.000 hair strands whereby the number of segments per hair strand does not have to be uniform. Because of this, the total number of vertices per hair mesh varies. All hair models are courtesy of Cem Yuksel and the female head model is courtesy of Murat Afshar [62].

6.1 Shading Models

6.1.1 Visual Results

This chapter shows visual results of the three implemented and described shading models: Kajiya and Kay (3.2), Marschner et al. (3.3), and Sadeghi et al. (3.4). All figures of this sub-chapter are rendered transparent and are using Deep Opacity Mapping with PCF filtering.
The phenomenological model of Kajiya and Kay does not provide many parameters in order to control the appearance of the hair. There is the diffuse color and intensity as well as the specular intensity and the specular exponent m. Figure 29 shows varying specular exponents with fixed intensities and a fixed diffuse color. It can be seen that decreasing the specular exponent does broaden the specular highlight. This follows directly from the phong model from which the model of Kajiya and Kay was derived. However, the model does not observe the law of energy conservation (cf. 2.4.1), implying that the model of Kajiya and Kay is not physically-plausible. Figure 46 shows more renderings using the model of Kajiya and Kay.



$m = 10$ $m = 50$ $m = 100$

Figure 29: Shading model of Kajiya and Kay. The specular exponent varies from left to right from $m = 10$, $m = 25$, to $m = 50$, whereby the specular intensity is always set to 1 and the diffuse color is $(0.72, 0.59, 0.47)$

The model of Marschner et al. comes with much more sophisticated, physically - based parameters. In total, the model can be split into three components: R, TT, and TRT (see 3.3.1). All three components heavily depend on the longitudinal shift α_p and the azimuthal shift β_p, whereby the shifts of the TT and TRT component are based on the ones from the R component (see table 1). Using the longitudinal shift α_p, one can control *where* the component tends to appear on the hair model. In addition to that, the azimuthal shift β_p controls the *width* of the component. Figure 30 shows the R and TRT component where the azimuthal shift β_R varies from $5°$ to $10°$ and

the longitudinal shift α_R is set to $-5°$. The last column of figure 30 eventually shows the full shading model together with the diffuse component of Kajiya and Kay as suggested by [32]. In contrast to the model of Kajiya and Kay, the model of Marschner et al. is physically plausible as it is energy conserving. This can be easily seen as the highlight intensity decreases when the highlight width increases. The color of the secondary highlight only depends on the absorption coefficient σ_a as one can see in the top row of figure 32. Here, different absorption coefficients are shown for different hair colors. Beside the R and TRT components, the strong TT component contributes much to the appearance of hair as shown in figure 31. In this figure, the hair is lit by two light sources whereby one light source serves as a backlight. As for the model of Kajiya and Kay, other examples of the Marschner shading model can be found in the appendix (figure 46).

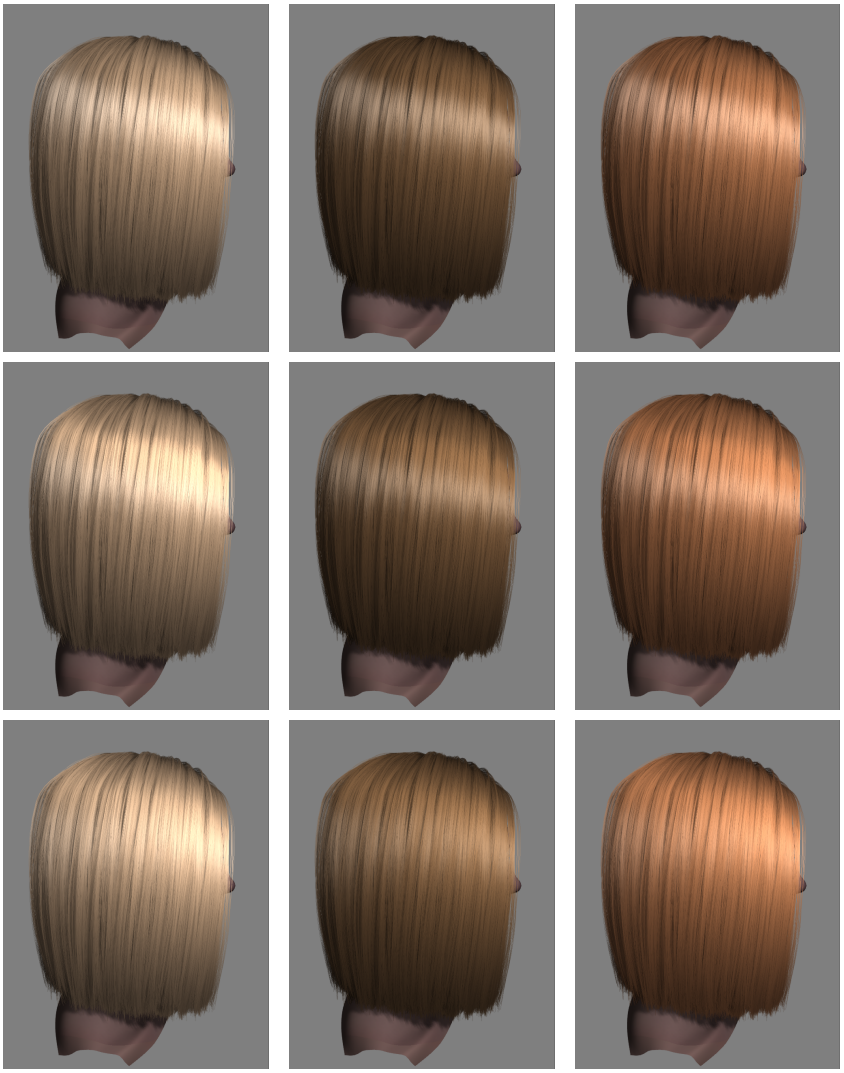
R component TRT component R + TRT + diffuse

Figure 30: Shading model of Marschner et al. showing the R and TRT component as well as the complete model including the diffuse component of Kajiya and Kay's shading model. The longitudinal width β_R is $5°$ for the first row and $10°$ for the second.

Marschner et al. Sadeghi et al.

Figure 31: Hair lit with two point lights, whereby one serves as a stronger backlight.

Sadeghi et al. approximated the model of Marschner et al. to provide more artist friendly controls. Chapter 3.4 already provides a good overview of all parameters. The longitudinal scattering functions M_p are unit-height Gaussians. Due to this, the intensity of the highlights is independent of the highlight's width β_p. However, intensity control parameters I_p were introduced to provide artists full control over each visual feature. Given all these parameters, it is feasible to match the model of Sadeghi et al. to the Marschner shading model as shown in figure 32. Here, the bottom row shows fine tuned renderings using the model of Sadeghi et al. The main difference is due to the lack of Fresnel equations in Sadeghi's model. However if needed, one can easily incorporate Fresnel equations in their model. For this thesis, the half angle between two glints G_{angle} is fixed to $45°$. To allow some more variation in the TRT component, G_{angle} should vary between $30°$ and $45°$ per hair strand. More results are shown in the appendix (figure 46).
Beside the primary and the secondary highlight, the model of Sadeghi et al. also provides a simple approximation for the TT forward scattering component of Marschner et al (see figure 31 right). Again, it is feasible to get a similar appearance of the TT component using the approximation of Sadeghi et al. However, a perfect match is not possible.



$\sigma_a = (0, 0.06, 0.17)$ $\sigma_a = (0.22, 0.3, 0.4)$ $\sigma_a = (0.11, 0.36, 0.4)$

Figure 32: Top: Model of Kajiya and Kay. Center: Sadeghi shading model ($I_R = 0.45, I_{TRT} = 0.3$) trying to match the Marschner based shading. Bottom: Marschner shading model ($\alpha_R = -5°, \beta_R = 7.5°$) with varying absorption coefficients σ_a.

6.1.2 Performance

The performance for the three shading models was measured using the three different hair models. However, all shading models perform nearly the same on the test system due to the fact that most of the math is precomputed. Nevertheless, the performance varies for the three hair models simply because of different amounts of geometry being rendered. Figure 33 shows the performance in milliseconds for the *Straight*, *Wavy*, and

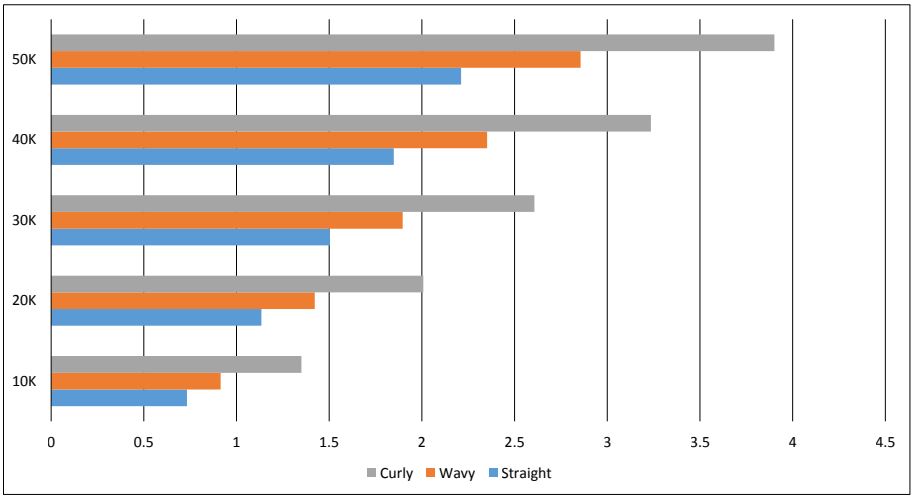


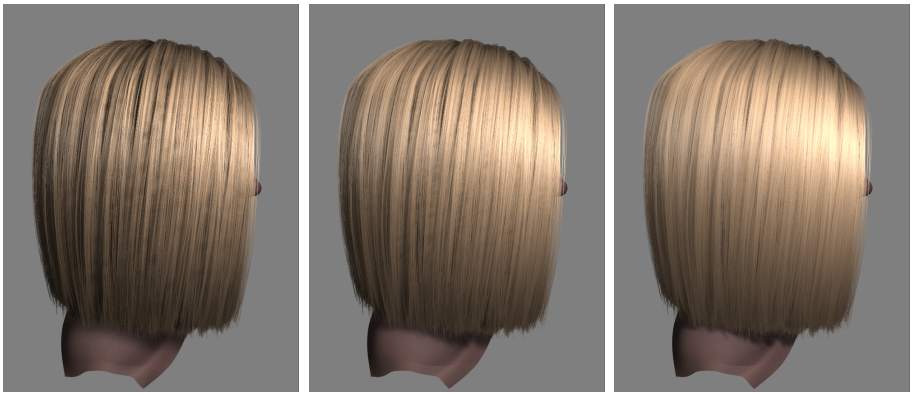
Figure 33: Performance in milliseconds for the three different hair models (*Straight*, *Wavy*, *Curly*) with varying hair density (10.000 - 50.000 hairs). The hair is illuminated by one light, without shadows and without transparency taken into account. Different shading models are not compared as they all perform very similarly on the test system used.

Curly hair model where the number of rendered hair strands varies between 10.000 and 50.000 strands. The hair was rendered as opaque geometry without any shadowing as GL_LINES. When using deferred rendering instead of plain forward rendering, the performance is slightly worse. However, further evaluation will show that deferred rendering is highly recommended for rendering hair with shadows.

6.2 Shadowing

6.2.1 Visual Results

Shadows are a crucial factor when rendering hair. They highly increase the voluminous appearance and the realism of the hair. For this work, three different real-time shadow techniques were implemented: Shadow Mapping (SM), Approximated Deep Shadow Mapping (ADSM), and Deep Opacity Mapping (DOM). The quality and the performance of all approaches highly depends on the shadow map and deep opacity map resolution (cf. figure 34). For this evaluation, the resolution is set to 512x512 texels.
The common approach to achieving proper shadows whit real-time rendering is SM. However, SM only provides a binary decision whether or not a fragment is in shadow. This yields to very noisy and unrealistic shadows when used with hair (cf. figure 37a). However, applying PCF - filtering helps to smooth the shadow significantly as shown in figure 37b. Here, a 5x5 PCF filter was applied. Nevertheless, the hair still appears much too

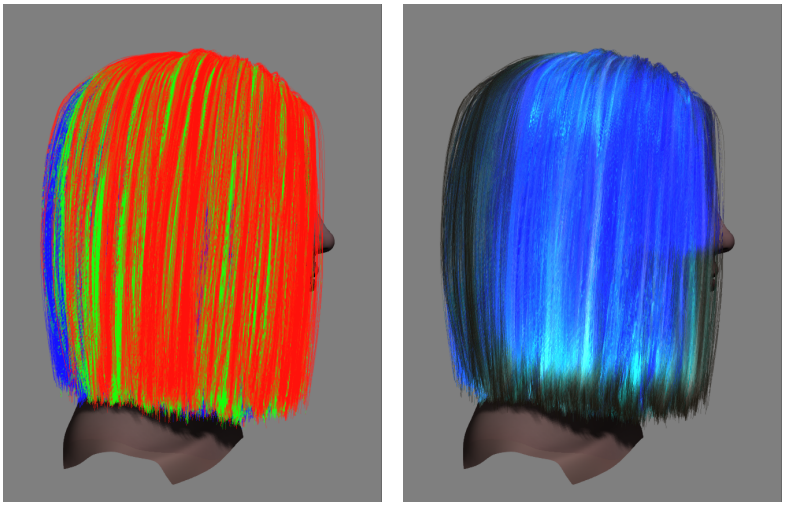


a) 128x128 b) 256x256 c) 512x512

Figure 34: Shadow rendered with DOM using a opacity map resolution of either a) 128^2, b) 256^2, or c) 512^2 texels using a PCF kernel size of 3x3.

dark when using SM. For this, the shadow strength s can be decreased yielding brighter shadows as in figure 37c. Here, s was decreased to 0.6. Unfortunately, this leads to an artificial look of the hair as it then appears way too bright in general, even in full shadowed regions.
To solve this, AMD came up with ADSM for *TressFX* [28, 33]. ADSM is a simple extension to normal SM as it just increases the shadow strength for fragments that lie deeper in the hair volume. In direct comparison to SM, ADSM provide shadows that are much more realistic (see figure 37f). Without proper filtering, ADSM also struggles with aliasing and noise as one can see in figure 37d. For artists, ADSM doesn't offer much control. There are basically two parameters that can be controlled: The transparency α_s and the width of the hair w. As shown in figure 37e and 37f, α_s affects the strength of the shadow. In addition to that, the hair width w also controls the strength of the shadow because w is used to guess the number of fibers between the current shading point and the light source. Because of this, a lower width increases the shadow strength. A more sophisticated but also more accurate technique is DOM. DOM tries to approximate the transmittance function $\tau(x)$ (see equation 67) in order to measure how much light is transmitted to a certain point x using K opacity layers aligned with the shape of the hair model. In this implementation, $K = 3$ layers are used as they fit best in one OpenGL texture while providing good results as described in chapter 4.2. Figure 35a shows the three layers as seen from the light source. Here, the red, green, and blue channels correspond to the first, second, and third layer, respectively. The depth of each layer is controllable by the user. However, the initial layering is based on the hair boundaries such that the first layer covers approximately 5%, the second layer 15%, and the third layer the rest of the hair model. The resulting opacity map is shown in figure 35b. It stores the accumulated opacity values of each layer

in the red, green, and blue channels. Due to the fact that the accumulated opacity values increase from layer to layer, the third layer contributes most to this visualization. Therefore, the most prominent color in figure 35b is blue. DOM also suffers from aliasing and noise without proper filtering. With proper filtering however, DOM gives the most plausible result. In direct comparison to ADSM, DOM provides smoother shadows and better nuances, especially for darker shadows. For brighter shadows however, ADSM gives a good approximation for volumetric shadows.



(a) DOM Layers **(b)** Opacity Map

Figure 35: a) Layers as seen from the light source used for DOM. The first layer is red, the second green, and the third is blue. b) The deep opacity map as seen from the light source. The red, green, and blue channels store the accumulated opacity values of the first, second, and third layers, respectively.

6.2.2 Performance

Shadows do have a huge impact on the performance of the rendering. Because of this, a comprehensive measurement was conducted. Figure 36 provides an overview of those measurements. It differs between three different resolutions of shadow and opacity maps (128x128, 256x256, and 512x512) as well as between the different shadow techniques using different filtering (none, 3x3, and 5x5 PCF). One can easily see that ADSM comes with nearly no cost in contrast to SM and that DOM is the most expensive technique. PCF filtering does have a slight impact on the performance of SM and ADSM and a greater impact on the performance of DOM. Noteworthy, the biggest performance impact on DOM ic caused by the size of the opacity map. However, chapter 6.2.1 has shown that a resolution of 256x256 can

be sufficient. Then, DOM using a 3x3 PCF filter requires ≈ 6.3 ms to compute whereby ADSM requires ≈ 3.3 ms. When transparency is not an issue, one can use deferred rendering in order to further speed up the rendering. With deferred rendering, DOM with 3x3 PCF filtering requires ≈ 5.0 ms to compute which is a speedup of $\approx 25\%$. However, deferred rendering does not speed up SM or ADSM significantly. Because of this and the good quality that ADSM is capable of offering, ADSM should be used in performance critical applications like video games. In any case, ADSM should be preferred to SM. With high PCF filtering and 512x512 shadow map resolution, ADSM is only $\approx 5\%$ slower than SM while providing much more pleasant results. See figure 37 for visual results.

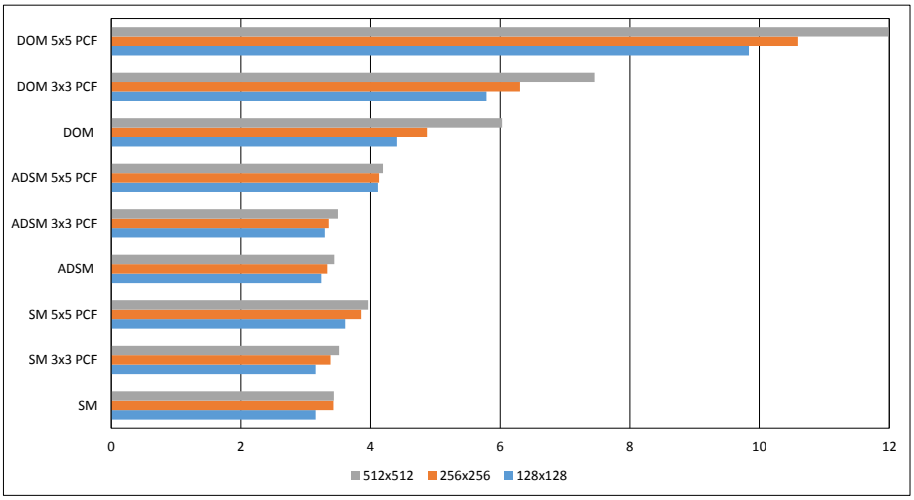


Figure 36: Performance in milliseconds for different shadow techniques (SM, ADSM, DOM) with varying shadow map / opacity map resolutions and varying filtering (none, 3x3, and 5x5 PCF). The resulting images are shown in figure 37.

(a) SM $s = 1$ **(b)** SM 5x5 PCF $s = 1$ **(c)** SM 5x5 PCF $s = 0.6$

(d) ADSM $\alpha_s = 0.4$ **(e)** ADSM 5x5 PCF $\alpha_s = 0.4$ **(f)** ADSM 5x5 PCF $\alpha_s = 0.2$

(g) DOM $\alpha_s = 0.03$ **(h)** DOM 5x5 PCF $\alpha_s = 0.03$ **(i)** DOM 5x5 PCF $\alpha_s = 0.01$

Figure 37: Comparison of SM (a-c), ADSM (d-f), and DOM (g-i). The first column shows the techniques using one sample only, whereby a 5x5 PCF filter was applied in the second column. In the third column, the shadow strength s and the shadow transparency α_s have changed in order to get brighter shadows.

6.3 Transparency

6.3.1 Visual Results

Transparency is a huge and difficult topic in hair rendering, simply because of the sheer amount of fragments that need to be blended. Furthermore, as hair is a translucent material, transparency plays a huge role when it comes to plausible, physically-based hair rendering. Without proper transparency, hair appears much to dull and aliased as in figure 38.

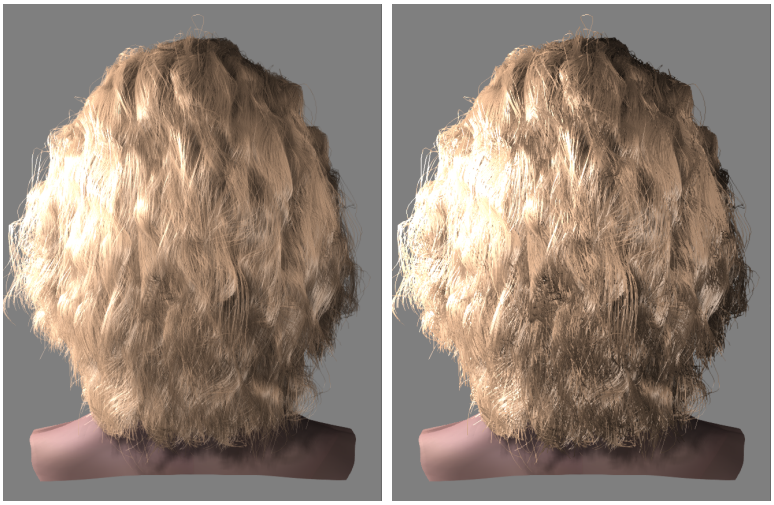
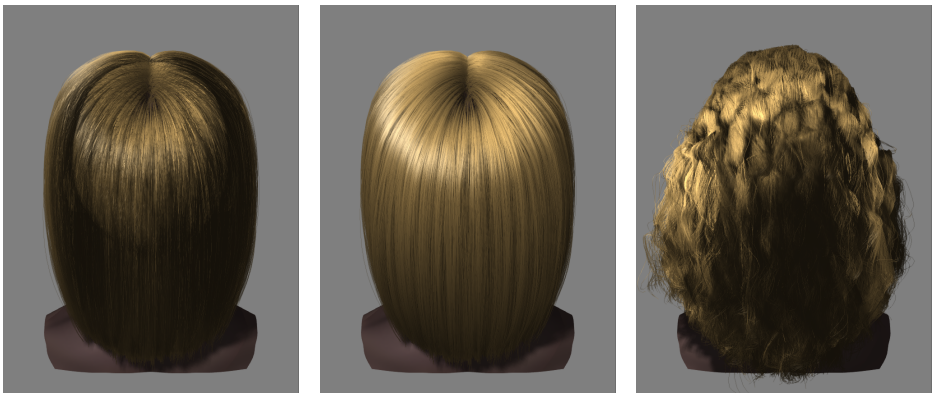


Figure 38: Transparent hair in contrast to opaque hair. Without transparency, the hair appears much more dull and aliased.

For this thesis, five different techniques have been implemented: *Blending*, *Random Subsets*, *A-Buffer*, *K⁺-Buffer*, and our new depth-based K-Buffer approach called: *DBK-Buffer*. *Blending* simply blends the hair from back to front using additive hardware blending. Because of this, the hair strands are sorted along the view direction on the CPU beforehand (cf. figure 39a and 39b). Beside that, *Blending* can produce wrong results, especially for complex hair models (cf. figure 39c). This is due to the fact that hair strands might overlap, which do make a sort per fragment necessary.



(a) Unsorted **(b)** Sorted **(c)** Complex model

Figure 39: Additive hardware blending of a) unsorted and b) sorted hair strands. For complex models as in c), sorting per hair strand is not sufficient.

Because of this, more sophisticated strategies using some kind of per pixel linked list (PPLL) approach should be used. Using the *A-Buffer*, all fragments are routed in the PPLL. This allows a full sorting and blending of all fragments. As the hair models used are very dense, one should avoid such a scenario. Moreover, only the k foremost fragments contribute to the final image when using a specific transparency α. Thus for $\alpha = 0.5$, only the $k = 6$ foremost fragments need to be blended, for instance. Using this, the *A-Buffer* still stores all fragments but sorts and blends only the k foremost ones. To solve this, the *K⁺-Buffer* was implemented which uses fragment shader interlocks in order to store the k foremost fragments only. The *DBK-Buffer* stores fragments that lie within a user-defined layer. All PPLL approaches, thus the *A-Buffer*, *K⁺-Buffer*, and the *DBK-Buffer*, are capable of producing the same visual result when used correctly. Because of that, a visual comparison is neglected. They only differ in terms of performance and memory consumption. Beside the sophisticated PPLL approaches, an easy to implement and fast technique called *Random Subsets* was implemented. It simply renders the hair in k random subsets in a deferred manner and sorts and blends them while compositing. Figure 47 shows $k = 8$ random subsets and the corresponding result.

(a) *Blending* **(b)** *PPLL* **(c)** *Random Subsets*

Figure 40: Visual results of a) *Blending* with sorted strands, b) PPLL approaches (*A-Buffer*, *K⁺-Buffer*, *DBK-Buffer*), and c) *Random Subsets*.

6.3.2 Performance

Handling transparency for dense hair meshes is challenging and expensive. The performance measurements of the techniques presented vary strongly as shown in figure 42. Here, measurements in milliseconds for all approaches implemented using either SM, ADSM, DOM, or without shadows (None) are shown. The *Straight* hair model, viewed from the side as in figure 37, is used again. It can be seen that the *A-Buffer* performs worst. That is because the complete PPLL needs to be traversed several times in order to find the k foremost fragments. The PPLLs can become very large, as the hair models with 50.000 strands each are very dense. The faster, but still slow *K⁺-Buffer* uses the OpenGL extension *NV_fragment_shader_interlock*. For this extension, a huge number of control flow barriers are introduced within a massive parallel algorithm. Due to this, the number of operations the GPU can execute in parallel is reduced, yielding a bad performance. Out novel *DBK-Buffer* outperforms the outer PPLL approaches. With DOM and 3x3 PCF filtering, the *DBK-Buffer* approach is capable of rendering the hair at a minimum of 50 frames per second (FPS). Using ADSM instead, nearly 58 FPS are attainable. For performance critical applications like video games, even 58 frames per second is much too slow. For this, the *Random Subset* ($k = 8$) approach is very suitable as it renders the hair with 91 FPS (DOM) and 125 FPS (ADSM), respectively. Considering that video games will not use such dense hair models, those techniques are well suited for real-time rendering. The hardware *Blending* approach might be fairly fast for SM and ADSM. However as mentioned before, *Blending* should be avoided as it yields wrong results. In contrast to all transparency techniques, the performance measurement also reveals that deferred rendering is not always faster than forward rendering. It is noteworthy, that

the performance of the *DBK-Buffer* strongly depends on the user defined depth shift. This depth shift depends on the hair model used and the transparency α and needs to be adjusted appropriately. For the measurements of figure 42, the depth shift was chosen such that $k \geq 8$ fragments are stored and thus the result is equal to the other approaches.

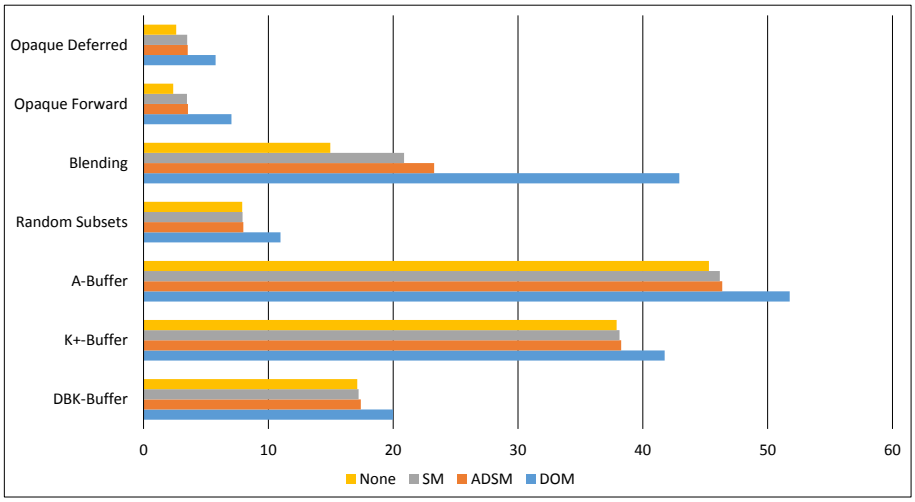


Figure 41: Performance in milliseconds for different transparency approaches using either Deep Opacity Mapping (DOM), Approximated Deep Shadow Mapping (ADSM), or just Shadow Mapping (SM), including a 3x3 PCF filter.

6.3.3 Memory Requirements

In addition to the technique's performance, its memory consumption is crucial, especially because GPUs have a limited amount of memory available. Figure 42 shows the GPU memory consumption of the transparency techniques described in megabytes (MB). Please note that *Blending* is omitted here as it doesn't consume any additional memory. The viewport is, as before, 656 x 861 = 564.816 pixels in size. As the *Random Subset* approach uses $k = 8$ framebuffer objects (FBOs), where each FBO has one *RGB16* color (for the tangent) and one 16 bit depth texture attached, a total of 34.47 MB is used. The *K⁺-Buffer* allocates memory for $k = 8$ fragments per pixel of size RGBA16 (tangent and depth). In addition, the counter buffer has to store one integer (4 byte) per pixel. Thus, the *K⁺-Buffer* allocates 36.63 MB in total. In contrast to this, the *A-Buffer* stores not only k but all fragments. For the *Straight* hair model as in figure 37, $12.959.208$ fragments are stored in the *A-Buffer*. Thereby each entry consists of one *RGBA16* for the tangent and the depth and one *R32* for the link. Additionally, the head buffer also occupies around 2.16 MB. Thus, a total of 150.46 MB is stored. However, as this is only true for a particular point of view and the *Straight* hair model,

the *A-Buffer* has to be much bigger in order to avoid overdrawing. For instance, the *A-Buffer* implementation of AMD in *TressFX* allocates around 276 MB of video memory for the given viewport size. For a full HD viewport (1920 x 1080), 930 MB are allocated. In order to avoid such huge memory consumption, the *DBK-Buffer* was developed. The huge advantage of this novel approach is the massive amount of saved memory combined with the very good performance it provides. Beside that, the *DBK-Buffer* does not depend on hardware features like the fragment shader interlock which is only available for the newest GPU generation. For memory measurements, the depth shift of the *DBK-Buffer* was chosen in such a way that the result looks equal to the other PPLL implementations. Then the *DBK-Buffer* stores 4.518.344 fragments which results in 53.86 MB of memory including the head buffer. In addition to this, the hair that lies behind the depth layer is rendered as opaque geometry as described in chapter 5.3. Thus, a total of 59.25 MB is occupied by the *DBK-Buffer*.

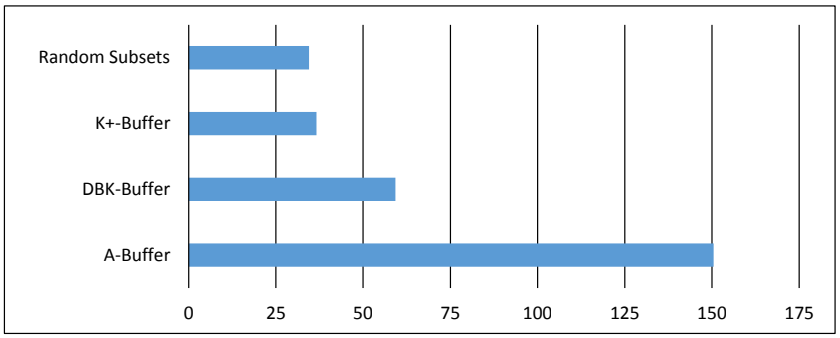


Figure 42: Video memory requirements of different transparency approaches in megabytes.

7 Conclusion

7.1 Future Work

Beyond this thesis some more papers can be taken into account. For instance, Yan et al. recently developed a physically-accurate reflectance model for fur [59]. Their model approximates the fiber with two cylinders whereby the outer cylinder accounts for the cortex and the inner one accounts for the medulla (cf. figure 43). They state that there is a structural difference between hair and fur strands. The medulla of fur is significantly larger than that of hair (see chapter 2.1) and therefore the medulla is approximated using one additional cylinder. Additionally, the cuticle of fur is usually rougher than that of hair.

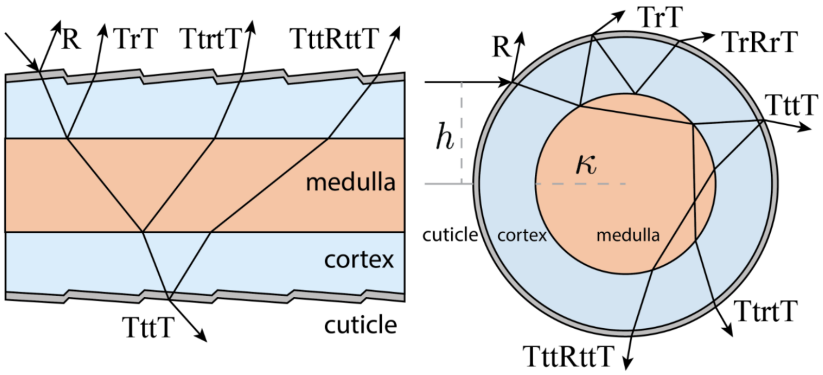


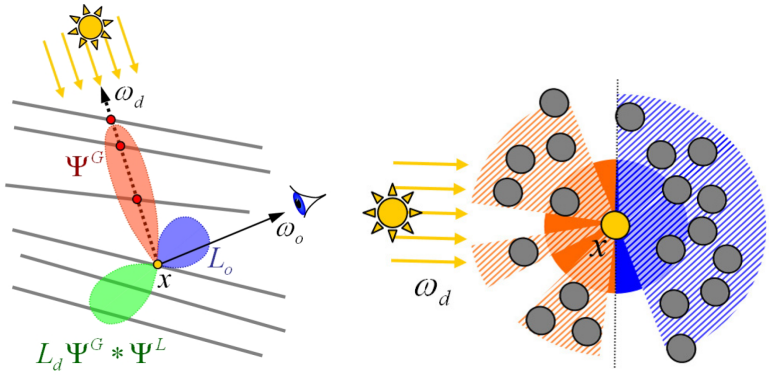
Figure 43: Double cylinder model of Yan et al. for the longitudinal direction (left) and the azimuthal direction (right). Image from [59].

The work of Zinke et al. about a dual scattering approximation in hair is a very sophisticated but valuable paper [65]. The basic idea here is to introduce a *multiple scattering function* $\Omega(x, \omega_d, \omega_i)$ which measures how much light arrives at a certain point x coming from direction ω_i. Thereby, the light enters the hair volume from a direction ω_d and scatters multiple times within the hair volume. Furthermore, Zinke et al. separate Ω in two components : *global multiple scattering* Ω^G and *local multiple scattering* Ω^L. The global component Ω^G computes the irradiance that arrives at a point x (cf. figure 44a) whereby the local component Ω^L computes the multiple scattering of this irradiance within the local neighborhood of x. According to Zinke et al., the multiple scattering function is then

$$\Omega(x, \omega_d, \omega_i) = \Omega^G(x, \omega_d, \omega_i) * (1 + \Omega^L(x, \omega_d, \omega_i)). \tag{70}$$

Figure 44b shows the cross section of a hair cluster which shows forward and backward scattering. Here, the strand x receives illumination from forward scattered light, which comes from the orange region, and it receives

illumination from light that scatters back from the blue region.
Implementing the dual scattering approximation is challenging, especially for real-time rendering. Zinke et al. proposed an algorithm that would fit within a real-time system by pre-computing a lot of things and by doing further approximations. One idea is to use deep opacity mapping to approximate the forward scattering transmittance function T_f and the variance of forward scattering σ_f^2 in order to compute the global scattering function Ω^G. When deep opacity mapping is not a valid option, one could apply the ideas of Sadeghi and Tamstorf to efficiently approximate T_f and σ_f^2 [50]. They assume that all hairs in front of x have the same orientation as x. Nevertheless, this is only true for flat hair styles.
In the work about an artist friendly hair shading system by Sadeghi et al. (see 3.4), the dual scattering approximation is also discussed and implemented. Therefore, the dual scattering approximation also fits within an artist friendly system. The implementation is highly related to the original but involves some modifications in order to provide artist friendly controls.



(a) *Dual Scattering* **(b)** *Cross section*

Figure 44: a) Global and local multiple scattering within the hair volume and b) forward and backward scattering in the cross section of a hair cluster. Images from [65].

7.2 Summary

Interactive physically-based rendering of hair is a challenging and expensive task. This thesis provides a broad overview of existing real-time-capable techniques for rendering hair that are commonly used in the industry. Additionally, a comprehensive evaluation of the techniques in terms of quality, performance, and memory requirements is provided. The main contribution of this work is the *DBK-Buffer*, which was developed in order to solve the limitations of existing transparency approaches. Here, the goal was to reduce the amount of memory that is used on the GPU. However, our considerations also result in better performance compared to other buffer

based approaches. Thus it can be stated that the *DBK-Buffer* is a very useful contribution. However, the *DBK-Buffer* also has a drawback: The user has to specify a certain depth shift so that neither too few nor too many fragments are stored in the buffer. To capture the first case, the hair beyond that depth shift is rendered as opaque geometry.
For hair shading models, the model of Kajiya and Kay was deduced from the well-known Phong shading model. Furthermore, the physically-based hair shading model of Marschner et al. was derived, which is to the present day the foundation of most research in that area. In addition to that, the artist friendly hair shading system of Sadeghi et al., which is used in production rendering at the Walt Disney Animation Studios, was shown. All these shading models were evaluated in a comprehensive analysis in chapter 6.1. It was shown that the model of Kajiya and Kay is suitable for providing a useful diffuse component. Their specular component however can only provide a harsh approximation of the primary highlight of hair. In contrast to this, the model of Marschner et al. results in much more realistic renderings. In addition to that, the model of Sadeghi et al. is much easier to control while providing a very good approximation of Marschner's model. Nonetheless, their model is not physically plausible as it is not energy conserving. If this is an issue, one can easily fix that by applying the Fresnel equations to the model and by normalizing the pseudo scattering function as $f_s'^{norm} = f_s'/(\int_\Omega f_s')$. Besides these three shading models, chapter 7.1 provides ideas for future work. So, an outlook on the prominent paper *Dual Scattering Approximation for Fast Multiple Scattering in Hair* by Zinke et al. is given. Furthermore, the ideas of Yan et al. to approximate fur fibers with two cylinders are mentioned briefly.
Moreover, this thesis also covers appropriate techniques for computing shadows in hair in real-time. Here, the evaluation has shown that Approximated Deep Shadow Mapping (ADSM) and Deep Opacity Mapping (DOM) provide the best results. For time critical applications such as video games, ADSM might be the better way to go as it is much faster than DOM. For applications that place the most importance on quality, DOM should be used instead.
In conclusion it can be stated that hair rendering at interactive, or even at real-time frame rates is practically possible. Using either the novel *DBK-Buffer* or the *Random Subsets* approximation allows very fast renderings with all the shadowing techniques presented using a mid-range GPU.

Additional Figures

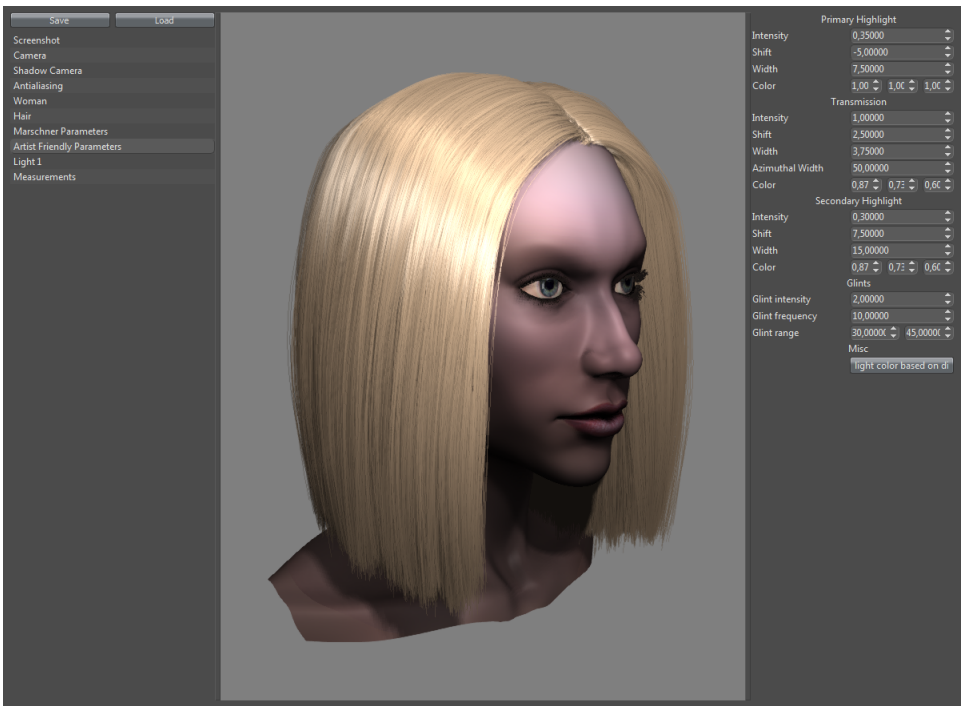


Figure 45: The user interface using the Qt library [45]. The left side lists all objects in the scene as cameras, meshes, lights, and certain controllable parameters. Objects can be selected by the user by clicking on them. Here, the object *Artist Friendly Parameters* is currently selected. The right side of the user interface provides a simple parameter editor to adjust exposed values during the run-time. As *Artist Friendly Parameters* are selected in the object list, the parameter editor shows all controllable parameters of the artist friendly hair shading system (cf. chapter 3.4). The OpenGL viewport lies in between both interfaces, currently rendering the *Straight* hair model (cf. figure 28).

Kajiya and Kay Marschner et al. Sadeghi et al.

Figure 46: *Curly* hair model rendered using either the model of Kajiya and Kay (left), the model of Marschner et al. (middle), or the model of Sadeghi et al. (right).

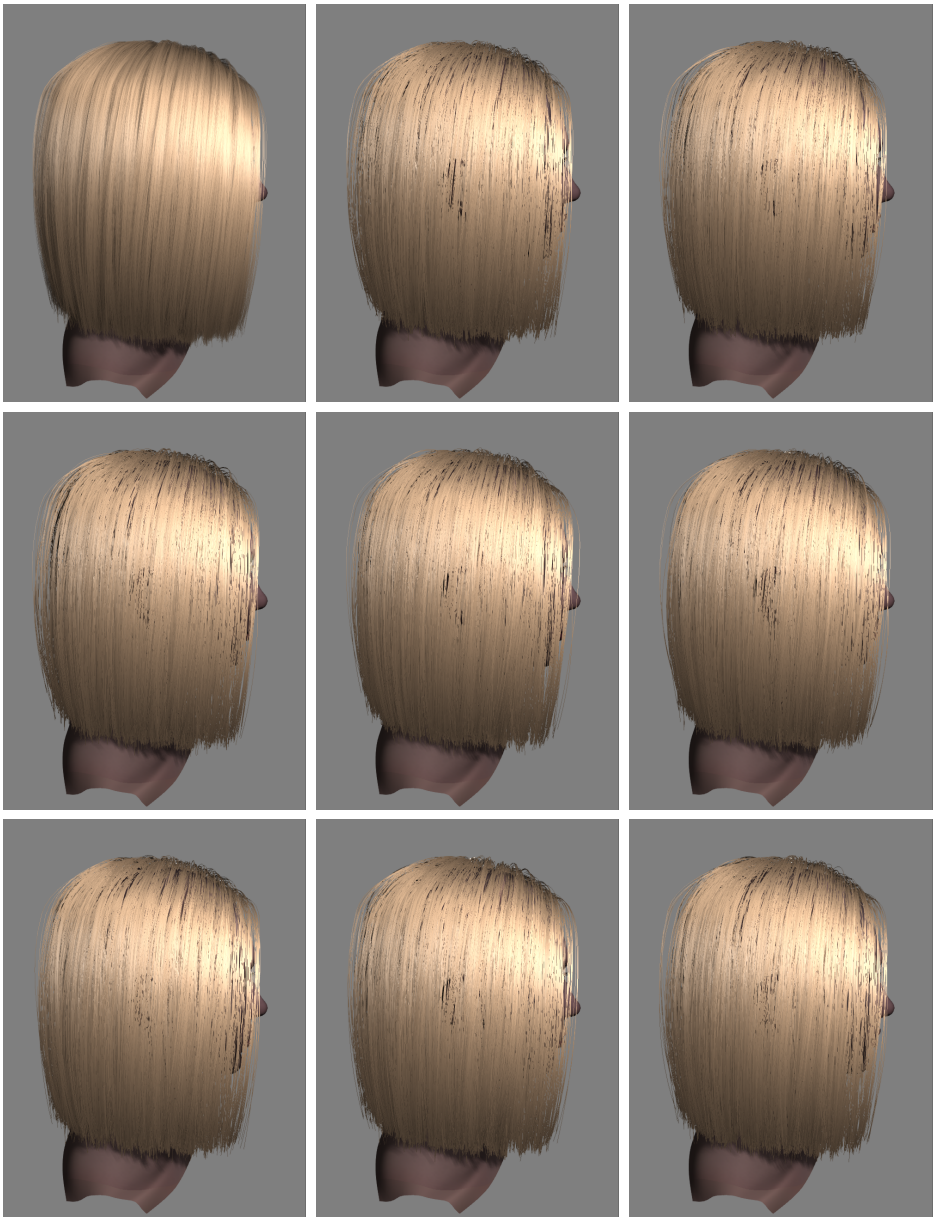


Figure 47: Top left: Hair rendered transparent using 8 random subsets. Rest: The 8 random subsets rendered deferred. A fragment shader sorts each fragment per depth, shades, and blends them from back to front.

List of Symbols

Symbol	Description	Definition
ν	frequency	eq. 2
λ	wavelength	eq. 3
Q	radiant energy	eq. 4
Φ	radiant flux	eq. 5
I	radiant intensity	eq. 6
E	irradiance	eq. 7
B	radiosity	eq. 8
L	radiance	eq. 9
$f(d\omega_i, d\omega_o)$	bidirectional reflectance distribution function (BRDF)	eq. 10
$f_s(d\omega_i, d\omega_o)$	single scattering function	eq. 32
$f'_s(d\omega_i, d\omega_o)$	pseudo scattering function	eq. 60
ρ	directional reflectance	eq. 12
$\rho(\theta_i)$	directional-hemispherical reflectance	eq. 13
η	index of refraction	1.55 (for hair)
η'	bravais index perpendicular	equation 38
η''	bravais index parallel	equation 53
η^*	Corrected index of refraction for elliptical fibers	equation 59
σ_a	absortion coefficient (R,G,B)	$[0.2, \infty]$
a	eccentricity	$[0.85, 1]$
u	hair tangent pointing from root toward the tip	(u_x, u_y, u_z)
v	major axis	(v_x, v_y, v_z)
w	minor axis	(w_x, w_y, w_z)
l / l'	incoming light direction / projected	equation 26
e / e'	outgoing view direction / projected	-
ω_i / ω_o	incoming / outgoing light direction in spherical coordinates	(θ, ϕ)
θ_i / θ_o	incoming / outgoing longitudinal angle	$[-\frac{\pi}{2}, \frac{\pi}{2}]$
θ_d	difference longitudinal angle	eq. 21
θ_h	half longitudinal angle	eq. 23
ϕ_i / ϕ_o	incoming / outgoing azimuth	$[0, 2\pi]$
ϕ	relative azimuth	eq. 22
ϕ_h	half azimuth	eq. 24

Table 1: List of all symbols and terms used in this thesis

Symbol	Description	Definition
Ψ_d/Ψ_s	diffuse / specular component of Kajiya-Kay	-
k_d/k_s	diffuse / specular reflection coefficient	-
p	number of internal path segments	$\{R, TT, TRT\}$
h	offset from the center of a unit circle	$[-1, 1]$
γ_i	angle of incidence in a circular cross section	equation 41
γ_t	angle of the refracted ray in a circular cross section	equation 42
γ_c	angle at which caustic appears	equation 57
$\phi(p, h)$	exit angle in a ciruclar cross section	equation 43
$\hat{\phi}(p, \gamma_i)$	approximation of $\phi(p, \gamma_i)$	equation 46
α_R	logitudinal shift path R	$[-10°, -5°]$
α_{TT}	logitudinal shift path TT	$-\alpha_R/2$
α_{TRT}	logitudinal shift path TRT	$-3\alpha_R/2$
β_R	logitudinal width path R	$[5°, 10°]$
β_{TT}	logitudinal width path TT	$\beta_R/2$
β_{TRT}	logitudinal width path TRT	$2\beta_R$
δ^2_{TT}	azimuthal width path TT	-
δ^2_G	azimuthal width for glints	-
k_G	glint scale factor	$[0.5, 5]$
w_c	azimuthal width of caustic	$[10°, 25°]$
$\Delta\eta'$	fade range of caustic merge	$[0.2, 0.4]$
Δh_M	caustic intensity limit	0.5
$M_p(\theta_h)$	longitudinal scattering function path p	equation 40
$M'_p(\theta_h)$	longitudinal pseudo scattering function path p	equation 61
$N_p(\theta_d, \phi)$	azimuthal scattering function path p	equation 55
$N'_p(\theta_d, \phi)$	azimuthal pseudo scattering function path p	equation 62,63,64
$F(\eta_i, \eta_t, \theta_i)$	Fresel equation for dielectrics	equation 18
$g(\beta, x, \mu)$	normalized Gaussian function with mean μ and standard deviation β	equation 39
$T(\sigma_a, h)$	absorption factor	equation 52
$A(p, h)$	attenuation factor	equation 54
$\tau(x)$	transmittance function	equation 67
$O(x)$	opacity at point x	equation 66
κ	scaling constant $e^{-\kappa} = 2^{-d}$	$b = 8 \Rightarrow k \approx 5.56$

Table 1: List of all symbols and terms used in this thesis

List of Figures

References

[1] Charles L Adler, James A Lock, and Bradley R Stone. Rainbow scattering by a cylinder with a nearly elliptical cross section. *Applied Optics*, 37(9):1540–1550, 1998.

[2] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-time rendering*. CRC Press, 2008.

[3] Inc. Autodesk. Maya. http://www.autodesk.de/products/maya/overview (last access: 03/03/2016).

[4] Louis Bavoil, Steven P Callahan, Aaron Lefohn, João LD Comba, and Cláudio T Silva. Multi-fragment effects on the gpu using the k-buffer. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 97–104. ACM, 2007.

[5] Louis Bavoil and Kevin Myers. Order independent transparency with dual depth peeling. *NVIDIA OpenGL SDK*, pages 1–12, 2008.

[6] James F Blinn. Light reflection functions for simulation of clouds and dusty surfaces. In *ACM SIGGRAPH Computer Graphics*, volume 16, pages 21–29. ACM, 1982.

[7] Jeff Bolz and Mathias Heyer. Nv_fragment_shader_interlock. https://developer.nvidia.com/sites/default/files/akamai/opengl/specs/GL_NV_fragment_shader_interlock.txt (last access: 02/01/2016).

[8] Helen K Bustard and Robin W Smith. Investigation into the scattering of light by human hair. *Applied Optics*, 30(24):3485–3491, 1991.

[9] Steven P Callahan, Milan Ikits, Joa OLD Comba, and Claudio T Silva. Hardware-assisted visibility sorting for unstructured volume rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 11(3):285–295, 2005.

[10] Loren Carpenter. The a-buffer, an antialiased hidden surface method. *ACM Siggraph Computer Graphics*, 18(3):103–108, 1984.

[11] Nvidia Corporation. Nalu. http://www.nvidia.de/coolstuff/demos#!/ nalu (last access: 03/03/2016).

[12] Nvidia Corporation. Nvidia hairworks. https://developer.nvidia.com/hairworks (last access: 03/03/2016).

[13] Advanced Micro Devices. Tressfx - gpuopen. http://gpuopen.com/gaming-product/tressfx/ (last access: 03/03/2016).

[14] Elmar Eisemann, Michael Schwarz, Ulf Assarsson, and Michael Wimmer. *Real-time shadows*. CRC Press, 2011.

[15] Cass Everitt. Interactive order-independent transparency. *White paper, NVIDIA*, 2(6):7, 2001.

[16] Tim Foley, Brent Insko, Tomasz Janczak, Marco Salvi, Larry Seiler, and Tomasz Poniecki. Intel_fragment_shader_ordering. https://www.opengl.org/registry/specs/INTEL/fragment_shader_ordering.txt (last access: 02/01/2016).

[17] NVIDIA GameWorks. Nvidia gameworks opengl samples. https://developer.nvidia.com/gameworks-opengl-samples (last access: 01/27/2016).

[18] William Jackson Humphreys. *Physics of the Air*. Franklin Institute of the state of Pennsylvania, 1920.

[19] David S Immel, Michael F Cohen, and Donald P Greenberg. A radiosity method for non-diffuse environments. In *ACM SIGGRAPH Computer Graphics*, volume 20, pages 133–142. ACM, 1986.

[20] Inc. Joseph Alter. Shave and a haircut. http://www.joealter.com (last access: 03/03/2016).

[21] James T Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.

[22] James T Kajiya and Timothy L Kay. Rendering fur with three dimensional textures. In *ACM Siggraph Computer Graphics*, volume 23, pages 271–280. ACM, 1989.

[23] Tae-Yong Kim. Modeling, rendering and animating human hair. *University of Southern California, Los Angeles, CA*, 2002.

[24] Tae-Yong Kim and Ulrich Neumann. *Opacity shadow maps*. Springer, 2001.

[25] JH Lambert. Photometria, sive de mensura et gradibus luminis, colorum et umbrae (augsberg: Eberhard klett). 1760.

[26] Dominik Lazarek. Real-time simulation and rendering of hair, fur and grass. Master's thesis, University of Koblenz-Landau, 2014.

[27] Jason Lcroix. Advanced visual effects with directx 11: Tomb raider on dx11. http://www.gdcvault.com/browse/gdc-13 (last access: 01/27/2016).

[28] Jason Lcroix. Tressfx 2.0 porting guide. http://developer.amd.com/tools-and-sdks/graphics-development/amd-radeon-sdk/ (last access: 01/27/2016).

[29] Tom Lokovic and Eric Veach. Deep shadow maps. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 385–392. ACM Press/Addison-Wesley Publishing Co., 2000.

[30] Abraham Mammen. Transparency and antialiasing algorithms implemented with the virtual pixel maps technique. *Computer Graphics and Applications, IEEE*, 9(4):43–55, 1989.

[31] D Marcuse. Light scattering from elliptical fibers. *Applied Optics*, 13(8):1903–1905, 1974.

[32] Stephen R Marschner, Henrik Wann Jensen, Mike Cammarano, Steve Worley, and Pat Hanrahan. Light scattering from human hair fibers. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 780–791. ACM, 2003.

[33] Timothy Martin, E Wolfgang, Nicolas Thibieroz, Jason Yang, and Jason Lacroix. Tressfx: Advanced real-time hair rendering. *GPU Pro*, 5:193–209, 2014.

[34] Morgan McGuire and Louis Bavoil. Weighted blended order-independent transparency. *Journal of Computer Graphics Techniques (JCGT)*, 2(2), 2013.

[35] Houman Meshkin. Sort-independent alpha blending. *GDC Talk*, 2007.

[36] Tadao Mihashi, Christina Tempelaar-Lietz, and George Borshukov. Generating realistic human hair for the matrix reloaded. In *ACM SIGGRAPH 2005 Courses*, page 17. ACM, 2005.

[37] Catherine M Mount, David B Thiessen, and Philip L Marston. Scattering observations for tilted transparent fibers: evolution of airy caustics with cylinder tilt and the caustic merging transition. *Applied Optics*, 37(9):1534–1539, 1998.

[38] Hubert Nguyen and William Donnelly. Hair animation and rendering in the nalu demo. *GPU Gems*, 2:361–380, 2005.

[39] FE Nicodemus, JC Richmond, JJ Hsia, IW Ginsberg, and T Limperis. Geometric considerations and nomenclature for reflectance, volume 161 of monograph. *National Bureau of Standards (US)*, 1977.

[40] Inc. Peregrine Labs. yeti. http://peregrinelabs.com/yeti (last access: 03/03/2016).

[41] Matt Pharr and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2004.

[42] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[43] Pixar. Renderman. https://renderman.pixar.com (last access: 03/03/2016).

[44] Thomas Porter and Tom Duff. Compositing digital images. In *ACM Siggraph Computer Graphics*, volume 18, pages 253–259. ACM, 1984.

[45] Qt. Qt web page. http://www.qt.io/ (last access: 17/02/2016).

[46] William T Reeves, David H Salesin, and Robert L Cook. Rendering antialiased shadows with depth maps. In *ACM Siggraph Computer Graphics*, volume 21, pages 283–291. ACM, 1987.

[47] Clarence R Robbins. *Chemical and physical behavior of human hair*, volume 4. Springer, 1994.

[48] Martin Ruenz. Real-time hair simulation and rendering. Master's thesis, University of Koblenz-Landau, 2012.

[49] Iman Sadeghi, Heather Pritchett, Henrik Wann Jensen, and Rasmus Tamstorf. An artist friendly hair shading system. In *ACM Transactions on Graphics (TOG)*, volume 29, page 56. ACM, 2010.

[50] Iman Sadeghi and Rasmus Tamstorf. Efficient implementation of the dual scattering model in renderman. Technical report, Tech. rep., Walt Disney Animation Studios, 2010.

[51] Thorsten Scheuermann. Practical real-time hair rendering and shading. In *ACM SIGGRAPH 2004 Sketches*, page 147. ACM, 2004.

[52] Nicolas Schulz. The rendering technology of ryse. 2014.

[53] Gerrit Sobottka and Andreas Weber. Geometrische und physikalische eigenschaften von human-haar. *Computer Graphics Technical Reports*, 2003.

[54] Robert F Stamm, Mario L Garcia, and Judith J Fuchs. The optical properties of human hair i. fundamental considerations and goniophotometer curves. *J. Soc. Cosmet. Chem*, 28(9):571, 1977.

[55] Andreas A Vasilakis and Ioannis Fudos. k+-buffer: Fragment synchronized k-buffer. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 143–150. ACM, 2014.

[56] Andreas-Alexandros Vasilakis, Georgios Papaioannou, and Ioannis Fudos. -buffer: An efficient, memory-friendly and dynamic-buffer framework. *Visualization and Computer Graphics, IEEE Transactions on*, 21(6):688–700, 2015.

[57] Hermann Von Helmholtz. *Handbuch der physiologischen Optik*, volume 9. Voss, 1867.

[58] Lance Williams. Casting curved shadows on curved surfaces. In *ACM Siggraph Computer Graphics*, volume 12, pages 270–274. ACM, 1978.

[59] Ling-Qi Yan, Chi-Wei Tseng, Henrik Wann Jensen, and Ravi Ramamoorthi. Physically-accurate fur reflectance: modeling, measurement and rendering. *ACM Transactions on Graphics (TOG)*, 34(6):185, 2015.

[60] Jason C Yang, Justin Hensley, Holger Grün, and Nicolas Thibieroz. Real-time concurrent linked list construction on the gpu. In *Computer Graphics Forum*, volume 29, pages 1297–1304. Wiley Online Library, 2010.

[61] Xuan Yu, Jason C Yang, Justin Hensley, Takahiro Harada, and Jingyi Yu. A framework for rendering complex scattering effects on hair. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 111–118. ACM, 2012.

[62] Cem Yuksel. Hair model files. http://www.cemyuksel.com/research/ hairmodels (last access: 17/02/2016).

[63] Cem Yuksel and John Keyser. Deep opacity maps. In *Computer Graphics Forum*, volume 27, pages 675–680. Wiley Online Library, 2008.

[64] Arno Zinke et al. Photo-realistic rendering of fiber assemblies. In *Ausgezeichnete Informatikdissertationen*, pages 331–337. Citeseer, 2008.

[65] Arno Zinke, Cem Yuksel, Andreas Weber, and John Keyser. Dual scattering approximation for fast multiple scattering in hair. In *ACM Transactions on Graphics (TOG)*, volume 27, page 32. ACM, 2008.