UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Fotorealistisches Rendering von Fell

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Lucas Hilbig

Erstgutachter: Prof. Dr. Stefan Müller
Institut für Computervisualistik/Computergrafik

Zweitgutachter: M.Sc. Kevin Keul
Institut für Computervisualistik/Computergrafik

Koblenz, im November 2018

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.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Zusammenfassung

Das fotorealistische Rendering von Fell ist ein oft gesehenes Problem in der Computergrafik und wird besonders bei Animationsfilmen häufig gebraucht. In dieser Arbeit werden zwei Beleuchtungsmodelle, ursprünglich zum Rendern von menschlichen Haaren, vorgestellt. Dies ist zum einen das Modell von Marschner et al. aus dem Jahr 2003, welches als Grundlage für viele neuere Modelle gilt, sowie das Modell von d'Eon et al. aus dem Jahr 2011. Beide Modelle werden innerhalb eines Pathtracers, welcher globale Beleuchtung simuliert, implementiert. Es werden die Besonderheiten von Haar-Fasern aus Fell im Gegensatz zu menschlichen Haar-Fasern aufgezeigt und folglich erläutert, warum die präsentierten Modelle auch für viele Fellarten genutzt werden können. Dabei liegt der Fokus auf einer realistischen visuellen Darstellung. Zusätzlich wird die Performance beider Modelle verglichen und Verbesserungsvorschläge durch die Nutzung von zylinder-förmigen Schnittpunktobjekten für den Pathtracer gegeben und anhand der Implementation evaluiert.

Abstract

Photo realistic rendering of fur is a common problem in computer graphics and is often needed in animation films. This work presents two illumination models, originally presented for human hair rendering. The first model is from Marschner et al. presented in 2003, which is the basis of many other models. The second model is from d'Eon et al., which was presented in 2011. Both models are implemented into a path tracer, which simulates global illumination. The special features of fur fibers in contrast to human hair fibers will be shown and an explanation, to why both models can also be used for fur rendering, will be given. The main point of interest is a realistic visualization of fur. In addition to that the performance of both models will be compared and a suggestion to improve the performance will be given and evaluated in form of the use of a cylindrical intersection object for path tracing.

Inhaltsverzeichnis

1 Einleitung

Fotorealistisches Rendering von dreidimensionalen Szenen ist in der Computergrafik ein häufig betrachtetes Problem. Das Anwendungsgebiet für solche Renderer reicht von der Erstellung einzelner Bilder, beispielsweise für Werbungen, bis hin zu ganzen Animationsfilmen. Damit fotorealistische Bilder erzeugt werden können, muss die gesamte Lichtstreuung zwischen Lichtquellen und Szenen-Objekten, sowie zwischen den Szenen-Objekten untereinander, simuliert werden. Das in dieser Arbeit genutzte *Pathtracing*-Verfahren ist ein solches globales Beleuchtungsverfahren. Damit besondere Materialien realistisch dargestellt werden können, müssen die speziellen Eigenschaften dieser bei der Beleuchtung mit einbezogen werden. Genauer gesagt nutzt der Pathtracer sogenannte BSDF's um die Lichtwechselwirkung solcher Materialien innerhalb der Szene zu berechnen.

Eines dieser Materialien ist Fell, beziehungsweise Haare, aus welchen Fell besteht. Ob als Tierfell, wie in Disney's Animationsfilm „Zoomania", oder menschlichem Haar, wie im Film „Frozen", die Darstellung realistischer Haare ist eine Herausforderung. Um die Lichtstreuung zwischen verschiedenen Haaren und besonders innerhalb eines einzelnen Haares physikalisch korrekt zu berechnen, werden Beleuchtungsmodelle gebraucht, die dieses ermöglichen. Zwei solcher Modelle, welche BSDF's für menschliches Haar vorstellen, werden in dieser Arbeit implementiert und gegeneinander verglichen. Der Hauptaugenmerk liegt dabei auf der realistischen, visuellen Darstellung des Felles im Hinblick auf die durch Lichtstreuung entstehenden Highlights, sowie den theoretischen Besonderheiten, die von den Autoren der Verfahren vorgestellt wurden. Auf die Simulation von Bewegungen der Haare wird nicht eingegangen.

Das erste Modell ist das 2003 von Marschner et al. [MJC+03] vorgestellte *Light Scattering from Human Hair Fibers*, welches die Grundlage für viele folgende Haar-Beleuchtungsmodelle ist. Eines dieser auf Marschner aufbauenden Modelle, ist das zweite in dieser Arbeit implementierte Beleuchtungsmodell *An Energy-Conserving Hair Reflectance Model* von d'Eon et al. aus dem Jahr 2011. Der Autor beschreibt dieses Modell als „energie-erhaltend", wodurch es in der Theorie sämtliche Streuung des Lichtes simulieren kann. Marschner's Modell hingegen ist nicht energie-erhaltend, denn es beschränkt die Anzahl an Reflexionen, welche innerhalb eines Haares stattfinden und verliert somit Informationen über die Beleuchtung des Haares. Anhand der Ergebnisse dieser Arbeit wird dieses Verhalten überprüft und dessen Nutzen in der Praxis eingeordnet. Genannte Modelle sind ursprünglich für menschliches Haar entwickelt worden. Aus diesem Grund werden zunächst die Unterschiede zwischen menschlichen Haar-Fasern und Haar-Fasern aus Tierfell untersucht, damit aus den gewonnen Erkenntnissen erläutert werden kann, warum die Modelle auch für Fell anwendbar sind.

Bevor Haare allerdings in einer 3D-Szene gerendert werden können, muss für diese zunächst eine Geometrie erstellt werden. Nach analysieren der physikali-

schen Form eines Haares, wird in dieser Arbeit eine Geometrie für Haare vorgestellt, die innerhalb des Pathtracers mit verschiedenen Schnittpunktobjekt-Typen genutzt werden kann. Diese Typen werden gegeneinander anhand der visuellen Unterschiede, sowie der Performance im Bereich Renderzeit und Speicherplatzbedarf gegeneinander verglichen.

2 Grundlagen

In den nachfolgenden Abschnitten werden grundlegende Konzepte vorgestellt, die für das tiefere Verständnis dieser Arbeit und ihrer Implementation notwendig sind. Zu diesen zählen die Globale Beleuchtung, mithilfe derer Licht-Wechselwirkungen in 3D-Szenen nahezu fotorealistisch simuliert werden können, sowie die Rendering-Verfahren Pathtracing und dessen Grundlage, das Raytracing. Außerdem werden in 2.2 Datenstrukturen erklärt, welche von Rendering-Verfahren für einen effizienteren und schnelleren Bildaufbau genutzt werden.

2.1 Globale Beleuchtung

In vielen Anwendungsgebieten der Computergrafik spielt die Beleuchtung eine essentielle Rolle um 3D-Szenen plastisch und realistisch wirken zu lassen. Sie beschäftigt sich damit, die Interaktion von Licht mit Materie möglichst realitätsnah und schnell zu simulieren. Dazu wird versucht die Energie, welche von einer Lichtquelle ausgeht und Materie so „beleuchtet“, mithilfe von mathematischen Formeln zu berechnen. Ein Beleuchtungsmodell um diese „direkte“ Beleuchtung von Lichtquellen zu simulieren wird beispielsweise von Phong in [Pho75] vorgestellt. In Gleichung 1 ist dieses Modell dargestellt, welches eine Farbe L für einen Oberflächenpunkt berechnet. Dazu muss von diesem Punkt die Normale, die Blickrichtung aus Sicht der Kamera, sowie die Richtungen der einfallenden Lichtstrahlen aller Lichtquellen bekannt sein. Das Modell fügt eine diffuse Komponente, sowie eine spiegelnde Komponente mit einem parametrisierten Highlight zusammen und ermöglicht so eine lokale Beleuchtung.

$$L = M_d \cdot L_a + \sum_{i=1}^{\#Lq} (M_d \cdot \cos\theta_i + M_s \cdot \cos^n \psi_i) \cdot L_i \tag{1}$$

M_d	diffuse Materialfarbe
L_a	ambiente Lichtfarbe
$\sum_{i=1}^{\#Lq}$	Summe über alle Lichtquellen der Szene
θ_i	Winkel zwischen Lichtstrahl und Oberflächennormale
M_s	spiegelnde Materialfarbe. Kann unterschiedlich zu M_d sein
n	Glanzzahl. Geringe Werte sorgen für ein großes Highlight
ψ_i	Winkel zwischen reflektiertem Lichtstrahl und Blickrichtung
L_i	Lichtfarbe der i-ten Lichtquelle

In der realen Welt wird Materie allerdings nicht nur „direkt“ durch Lichtquellen, sondern auch „indirekt“ durch andere Materie beleuchtet. Dieses Phänomen nennt sich „Lichtstreuung“ (im englischen *Light scattering*) und wird in [HvdH81] erklärt. Wenn nun diese Licht-Wechselwirkung von Materie ebenfalls simuliert wird, dann nennt man das in der Computergrafik „globale Beleuchtung“. Dadurch werden zusätzlich Effekte, wie *weiche Schatten* oder *Kaustiken*, ermöglicht. Rendering-Verfahren der globalen Beleuchtung versuchen also ein physikalisch korrektes Bild zu berechnen, bei dem durch Zusammenfügen der direkten und indirekten Beleuchtung der Energieerhaltungssatz[1] vollständig erfüllt sein soll. Der Großteil der im Verlaufe dieses Kapitels beschriebenen Verfahren und Gesetze ist im Buch [PJH16] ausführlich erläutert.

2.1.1 Rendering-Gleichung

Grundlage für viele globale Beleuchtungsverfahren, wie das in Kapitel 2.1.5 vorgestellte Verfahren *Pathtracing* oder auch *Photon mapping* (siehe [Jen01]), ist die 1986 von Jim Kajiya in [Kaj86] vorgestellte Rendering-Gleichung. Kajiya beschreibt dabei die Ausbreitung von Licht als Energiestrom. Mit dem oben genannten Energieerhaltungssatz als Basis ist es ihm so gelungen, die Lichtverhältnisse von sämtlichen Lichtquellen und Objekten in einer 3D-Szene als Integralgleichung zu approximieren. Die konkrete Gleichung (in vereinfachter Form für einen Punkt x statt für die Gesamtheit aller Flächen in der Szene) ist in 2 dargestellt.

$$L_r(x, \vec{\omega_r}) = L_e(x, \vec{\omega_r}) + \int_{2\pi} f_r(x, \vec{\omega_i}, \vec{\omega_r}) \cdot L_i(x, \vec{\omega_i}) \cdot \cos\theta_i \cdot d\omega_i \qquad (2)$$

$\vec{\omega_i}$ Einfallender Lichtstrahl, bei dem es sich um einen normierten Vektor der Länge 1 handelt.

$\vec{\omega_r}$ Ausgehender/Reflektierter Lichtstrahl, bei dem es sich um einen normierten Vektor der Länge 1 handelt.

$L_r(x, \vec{\omega_r})$ Die Menge an Energie (in Form von Licht), die vom Punkt x auf der Oberfläche in Richtung $\vec{\omega_r}$ abgegeben wird. Auch als Strahldichte (engl. *radiance* oder Leuchtdichte in der Photometrie) bezeichnet.

$L_e(x, \vec{\omega_r})$ Die Emission des Punktes x in Richtung $\vec{\omega_r}$. Respektive die selbstleuchtende Menge Licht, welche die Oberfläche im Punkt x produziert und Richtung $\vec{\omega_r}$ abgibt.

$\int_{2\pi} ...d\omega_i$ Das Integral, welches sich über die Gesamtheit aller Lichteinfalls- und Reflexionswinkel der Hemisphäre von Punkt x spannt.

[1] Im Jahr 1847 unter anderem von Hermann von Helmholtz vorgestellt.

$f_r(x, \vec{\omega_i}, \vec{\omega_r})$ Der *Streuungsterm*. Er gibt an, wie viel des Lichtes, welches x aus Richtung $\vec{\omega_i}$ erhält, über Reflexion wieder in Richtung $\vec{\omega_r}$ abgegeben wird.

$L_i(x, \vec{\omega_i})$ Die Menge des Lichtes, welche x aus Richtung $\vec{\omega_i}$ erreicht. Wird auch als Strahldichte (engl. *radiance* oder Leuchtdichte in der Photometrie) des einfallenden Lichtes bezeichnet.

θ_i Winkels zwischen der Oberflächennormalen $\vec{n}$ und dem eingehenden Lichtstrahl $\vec{\omega_i}$.

Der genannte Streuungsterm $f_r(x, \vec{\omega_i}, \vec{\omega_r})$ kann durch eine *bidirektionale Reflexionsverteilungsfunktion* (Bidirectional Reflectance Distribution Function, BRDF), welche in Abschnitt 2.1.2 genauer erläutert wird, dargestellt werden.

Mathematisch gesehen zeichnet die Rendering-Gleichung zwei Eigenschaften aus, die in [Kaj86] bewiesen wurden und welche sie in der praktischen Anwendung beliebt macht. Zum einen ist die Linearität der Gleichung gegeben, denn sie besteht ausschließlich aus Addition und Multiplikation beziehungsweise Skalar-Multiplikation (vergleiche [TT13]). Dies ermöglicht ein modulares Berechnen der einzelnen Komponenten in verschiedenen Funktionen. Zum anderen ist die Rendering-Gleichung in ihrer Anwendung unabhängig von Position und Orientierung der Komponenten, wie Oberfläche und Vektoren. Dies wird auch räumliche Homogenität genannt und die zuvor genannten Verfahren nutzen diese Eigenschaften um die Rendering-Gleichung flexibel an spezifische Problemstellungen anzupassen.
So kann der Term $f_r(x, \vec{\omega_i}, \vec{\omega_r})$ erweitert werden, um spezifische Materialien, wie Haare oder Fell, zu simulieren. Der originale Streuungsterm ist auch der Grund für die Beschränkungen der ursprünglichen Gleichung. Dieser ist dazu ausgelegt, die Energie zu berechnen, welche durch Reflexionen an der Oberfläche entstehen. Es gibt allerdings viele Materialien, wie Wasser oder Glas, die zusätzlich einen Teil des Lichtes refraktieren und dieser zusätzliche Aspekt der Lichtstreuung wird nicht beachtet in der ursprünglichen Rendering-Gleichung. Wie das *Light Scattering* im Detail funktioniert und welche Anpassungen am Streuungsterm vorgenommen werden müssen um alle Aspekte der Lichtstreuung einzufangen wird im folgenden Kapitel 2.1.2 erläutert.

2.1.2 Light Scattering

Wenn Energie in Form eines Lichtstrahls auf ein lichtdurchlässiges Objekt, wie beispielsweise Glas oder Haare, trifft, dann wird diese Energie in verschiedene Richtungen gestreut. Dies wird als Lichtstreuung (englisch *Light Scattering*) bezeichnet. Zum einen wird ein Teil des Lichtes auf der Oberfläche *reflektiert*, also zurückgeworfen, wie in Abbildung 1 zu sehen ist. Dabei gilt für glatte Oberflächen das *Reflexionsgesetz*, wonach der Ausfallswinkel θ_r des reflektierten Strahls gleich

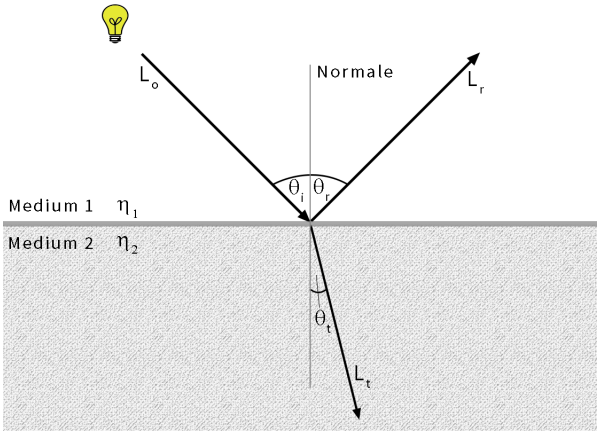


Abbildung 1: Streuung des Lichtes beim Auftreffen auf eine lichtdurchlässige Oberfläche.
Mit eingehendem Lichtstrahl (auf Oberfläche auftretende Energie) L_o, reflektiertem Lichtstrahl L_r, refraktiertem Lichtstrahl L_t, sowie dem Einfallswinkel θ_i, dem Reflexionswinkel θ_r, Refraktionswinkel θ_t und den Brechungsindizes der beiden Medien η_1 und η_2.

dem Einfallswinkel θ_i (Winkel zwischen Strahl und Normale) des eingehenden Lichtstrahls ist:

$$\theta_i = \theta_r \tag{3}$$

Der andere Teil des Lichtes wird *refraktiert*, das heißt der Lichtstrahl wird an der Oberfläche gebrochen und tritt vom ersten Medium in das zweite Medium ein (zu sehen in Abbildung 1). Hierbei verändert sich die Richtung des Strahls abhängig von den Brechungsindices η_1 des ersten Mediums und η_2 des zweiten Mediums (siehe [Hay14]). Es handelt sich bei diesem Brechungsindex η (*Index Of Refraction*, IOR) um eine optische Materialeigenschaft, welche in etwa den Wert 1 für Luft oder den Wert 1,33 für Wasser besitzt (vergl. [Hay14]). Um die Richtung des refraktierten Strahls zu berechnen wird das Brechungsgesetz (Gleichung 4) von Snell genutzt:

$$\eta_1 \sin(\theta_i) = \eta_2 \sin(\theta_t) \tag{4}$$

In dieser Gleichung sind η_1, η_2 die Brechungsindices, θ_i der Einfallswinkel und θ_t der Ausfallswinkel des gebrochenen Strahls (in Abbildung 1 zu sehen).
Wichtig ist außerdem, dass bei einer Refraktion nicht die gesamte Energie, die beim Eintreten in die Materie anliegt, beim Austreten aus dieser im Lichtstrahl vorhanden ist. Der Grund dafür ist, dass je nach physischer Beschaffenheit des Objektes ein unterschiedlich großer Teil der Energie absorbiert wird (siehe [HvdH81]). Über den Lichtanteil , welcher das Objekt wieder verlässt, gibt die *Transmission* Aufschluss.
Nach dem in 2.1 bereits genannten Energieerhaltungssatz teilt sich die auf der Oberfläche auftretende Energie L_O beim *Light Scattering* in die drei bereits er-

wähnten Komponenten auf, welche in Gleichung 5 festgehalten sind: Die *reflektierte Energie* L_r, die *transmittierte Energie* L_t und zuletzt die *absorbierte Energie* L_a.

$$L_O = L_r + L_t + L_a \tag{5}$$

2.1.3 BSDF

Eine Möglichkeit die Lichtstreuung einer Oberfläche zu beschreiben sind die *bidirektionalen Streuungsverteilungsfunktionen* (Bidirectional Scattering Distribution Function, BSDF). Bei diesen handelt es sich um eine mathematische Funktion, welche das Verhalten von Licht beim Auftreffen auf eine Oberfläche mit einem bestimmten Material beschreibt. *BSDF* ist dabei der Oberbegriff für viele verschiedene Funktionen, die jeweils unterschiedliche Aspekte der Lichtstreuung in unterschiedlichen Dimensionen betrachten. Einen Teil dieser Funktionen machen die in 2.1.1 genannten ***BRDF's*** aus, welche das Verhalten des Lichtes bei der ***Reflexion*** an einer Oberfläche darstellen[2]. Eine solche Funktion ist nach [PJH16] in Gleichung 6, mit gleichen Notationen wie die der Rendering-Gleichung, definiert als:

$$f_r(\vec{\omega_i}, \vec{\omega_r}) = \frac{dL_r(\vec{\omega_r})}{dE_i(\vec{\omega_i})} \tag{6}$$

Bei $f_r(\vec{\omega_i}, \vec{\omega_r})$ handelt es sich um einen möglichen Streuungsterm für die Rendering-Gleichung 2, wobei das fehlen von x darauf zurückzuführen ist, dass f_r hier nicht für einen konkreten Punkt x sondern für die Gesamtheit aller Flächen notiert ist. $L_r(\vec{\omega_r})$ ist die Strahldichte (auch spezifische Intensität, engl. *radiance*), welche von der Oberfläche in Richtung $\vec{\omega_r}$ ausgesandt wird. Der Term $E_i(\vec{\omega_i})$ ist die Bestrahlungsstärke (auch Strahlungsstromdichte, engl. *irradiance*) und gibt die gesamte Lichtenergie an, welche aus Richtung $\vec{\omega_i}$ auf die Oberfläche trifft. Das Differenzial $dE_i(\vec{\omega_i})$ steht proportional zur eingehenden Strahldichte :

$$dE_i(\vec{\omega_i}) = L_i(\vec{\omega_i}) \cdot \cos\theta_i \cdot d\omega_i \tag{7}$$

Nach Einsetzen von Gleichung 7 in Gleichung 6 und Umformen nach $L_r(\vec{\omega_r})$ ergibt sich das Integral aus der Rendering-Gleichung 2.

Als Gegenstück dazu befassen sich die *Bidirectional Transmittance Distribution Functions* (BTDF's) mit den Refraktionen eines lichtdurchlässigen Materials. Häufig ist mit BSDF eine Kombination aus BRDF und BTDF gemeint, denn so kann die Lichtstreuung eines Materials in seiner Gesamtheit approximiert werden. Dabei sind in der Literatur BSDF's für verschiedene Materialien vorgestellt worden, wie die Lambertsche BRDF für diffus reflektierende Materialien[3].
Es gehen sowohl BRDF als auch BTDF davon aus, dass das Licht am selben Punkt, an dem es auf das Objekt trifft, auch wieder Gebrochen und Reflektiert wird. Dies

[2] Deshalb *Bidirectional* ***Reflectance*** *Distribution Function*

[3] Nach Johann Heinrich Lambert, welcher das Konzept der perfekten diffusen Oberflächen in seinem Buch *Photometria* im Jahr 1760 vorstellte

ist aber bei Materialien, wie Haaren oder Fell, nicht der Fall (siehe [MJC+03]). Bei einem Haar wird das Licht intern Reflektiert und Refraktiert, weshalb es an verschiedenen anderen Punkten der Haarfaser austritt. In solch einem Fall hängt die von der Oberfläche ausgesandte Strahldichte nicht nur vom Lichteinfalls- und Lichtaustrittswinkel ab, sondern auch von dem Punkt relativ zur gesamten Oberfläche an dem das Licht auftritt. Funktionen, die auch diese Eigenschaft mit einbeziehen nennt man *Bidirectional Surface Scattering Reflectance Distribution Function* (BSSRDF).

2.1.4 Raytracing

Beim *Raytracing* (Strahlungsverfolgung) handelt es sich um ein Rendering -Verfahren, dass die Verdeckungsberechnung in 3D-Szenen durch Aussenden und Verfolgen von Strahlen durchführt. Es können mit diesem Algorithmus nahezu fotorealistische Bilder aus einer 3D-Szene erzeugt werden. Dazu wird Licht als Vielzahl einzelner Strahlen interpretiert, welche verfolgt werden, um ihre Schnittpunkte mit Objekten der Szene zu berechnen und diese je nach getroffenem Material zu beleuchten. Ein Strahl besteht dabei aus seiner Ursprungsposition und der Richtung in die er zeigt, jeweils durch Vektoren im Weltkoordinatensystem angegeben.

Die Funktionsweise eines einfachen Raytracers, welcher eine direkte Beleuchtung kalkuliert, kann in folgende Schritte aufgeteilt werden:

1) Primärstrahlen erstellen (*create Rays*)

2) Strahlen verfolgen und jeweils vordersten Schnittpunkt mit Szene finden (*Trace*)

3) Farbe pro Schnittpunkt berechnen (*Shade*)

Zu Beginn werden die *Primärstrahlen* erstellt. Von diesen wird aus der Sicht der Kamera, welche als Augpunkt dient, jeweils ein Strahl für jedes Pixel im Bild erstellt. Die Strahlen besitzen dabei die Kamera als Ursprung und zeigen in Richtung ihres Pixels. In Abbildung 2 ist dieser Ablauf für einen einzelnen Primärstrahl dargestellt. Der Augpunkt in der Abbildung ist in der Praxis die Kamera der Szene. Für ein $1920 \cdot 1080$ Pixel großes Bild würden somit 2073600 Primärstrahlen erstellt werden.
Jeder dieser Strahlen wird separat entlang seiner Richtung verfolgt. Es wird dabei versucht, den vordersten Schnittpunkt mit einem Primitiv der Szene zu finden. Dazu müssen alle Objekte in der Szene auf einen Schnittpunkt getestet werden, damit der vorderste gefunden und beleuchtet werden kann. Dieses Vorgehen wird in der Praxis durch *Datenstrukturen*, welche in Kapitel 2.2 erklärt werden, beschleunigt. Für den Fall, dass kein Schnittpunkt mit einem Objekt der Szene gefunden wird, nutzt man eine Hintergrundfarbe oder Textur für die Farbberechnung. Wenn ein Schnittpunkt, wie in Abbildung 2 zu sehen, gefunden wurde, dann wird dieser beleuchtet.

Mithilfe eines *Shaders*[4] wird die Farbe des Schnittpunktes ermittelt. Dazu werden für einfache Raytracer lokale Beleuchtungsmodelle, wie das von Phong (Gleichung 1, siehe [Pho75]), genutzt. Bei direkter Beleuchtung wird an einem Schnittpunkt Farbe gesehen, wenn dieser auf direktem Wege von einer Lichtquelle angestrahlt wird. Anderenfalls liegt der Schnittpunkt im Schatten. Um zu prüfen ob dies der Fall ist, wird vom Schnittpunkt aus für jede Lichtquelle der Szene ein neuer Strahl erstellt. Diese *Schattenfühler* haben den Schnittpunkt als Ursprung und zeigen in Richtung ihrer jeweiligen Lichtquelle. Sie werden verfolgt und wenn sich kein Objekt zwischen Schnittpunkt und Lichtquelle befindet, wird die errechnete Farbe des Beleuchtungsmodells für den Pixel des Primärstrahls eingetragen. Wenn jede Lichtquelle vom Schnittpunkt aus verdeckt ist, dann liegt dieser im Schatten und es wird Schwarz als Farbe genutzt.

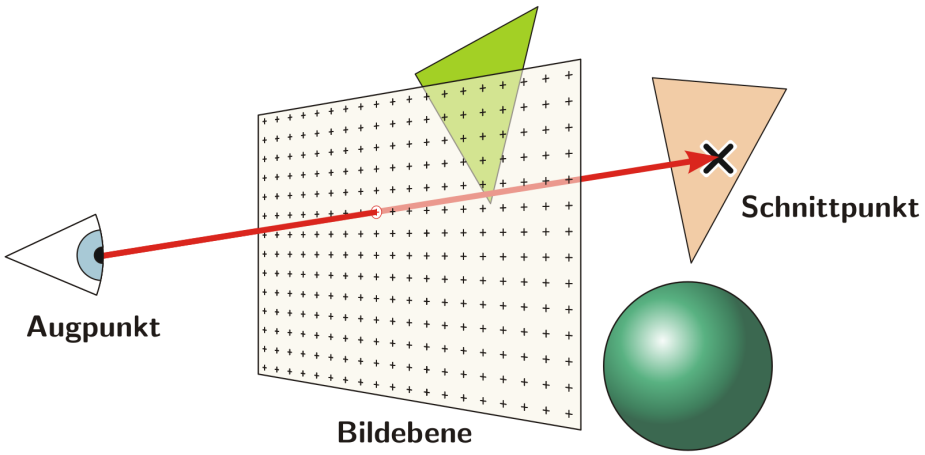


Abbildung 2: Raytracing-Verfahren. Strahl wird vom Augpunkt (Kamera) aus in Richtung eines Pixels der Bildebene verschossen.
Bildquelle: [Phr]

Zusätzliche Effekte. Mit dem zuvor beschriebenen Vorgehen wird eine direkte Beleuchtung mit *harten Schatten* erreicht. Um Teile einer globalen Beleuchtung zu simulieren, müssen zusätzliche Effekte zum Raytracer hinzugefügt werden. Zu diesen Zählen *weiche Schatten*, sowie indirekte Beleuchtung mithilfe von *Reflexionen* und *Refraktionen*. Außerdem ist anzumerken, dass Strahlen, die zusätzlich zu den Primärstrahlen verschossen werden (wie die bereits genannten Schattenfühler), auch als *Sekundärstrahlen* bezeichnet werden.
Weiche Schatten, auch Halbschatten genannt, werden simuliert, indem die Lichtquellen als Fläche dargestellt werden und pro Lichtquelle mehrere Schattenfühler

[4]Funktion/Teil des Renderers, welcher für die Farbberechnung zuständig ist.

an unterschiedliche Positionen dieser Fläche verschossen werden. Die Ergebnisse der einzelnen Schattenfühler werden dann pro Lichtquelle gemittelt. Eine höhere Anzahl an Schattenfühlern pro Lichtquelle sorgt dafür, dass die Schatten weicher und weniger verrauscht werden.
Reflexionen, wodurch andere Objekte indirekt beleuchtet werden, erhält man, indem für spiegelnde Oberflächen ein Sekundärstrahl erstellt wird. Dieser besitzt den Schnittpunkt als Ursprung und zeigt in die reflektierte Richtung, welche wie in 2.1.2 beschrieben mithilfe des Reflexionsgesetzes 3 berechnet wird. Der neue Strahl wird nun rekursiv verfolgt, dies bedeutet er wird als Primärstrahl interpretiert und die zuvor genannten Schritte 2) sowie 3) werden für den neuen Strahl durchgeführt. Um eine Endlosrekursion zu vermeiden, legt man eine maximale Rekursionstiefe fest, nach der keine neuen Strahlen mehr verfolgt werden.
Refraktionen erzeugt man nach dem gleichen Prinzip wie Reflexionen, mit dem Unterschied, dass diese nur an lichtdurchlässigen Objekten auftreten und die Richtung des neuen Strahls nach dem Brechungsgesetz 4 kalkuliert wird. Durch das rekursive vorgehen nennt man diese mit Reflexion und Refraktion erweiterte Variante auch *rekursives Raytracing*.

Trotz indirekter Beleuchtung können mithilfe des Raytracing Teile der globalen Beleuchtung, wie Kaustiken, emissive Oberflächen oder diffuse Interreflexion, nicht simuliert werden. Der Grund dafür ist unter anderem, dass Sekundärstrahlen für Reflexionen lediglich bei spiegelnden Oberflächen entsandt werden. In der Wirklichkeit reflektieren auch diffuse Oberflächen einen Teil des Lichtes (siehe [Hay14]).

2.1.5 Pathtracing

Um die zuvor genannten Phänomene simulieren zu können wurde das auf Raytracing basierende Verfahren des *Pathtracing* als Lösung der Rendering-Gleichung in [Kaj86] vorgestellt. Dieses ermöglicht eine vollständige globale Beleuchtung, indem es für jeden vom Auge aus sichtbaren Punkt über sämtliches einfallendes Licht integriert. Denn im Gegensatz zum Raytracing wird beim Pathtracing die abgestrahlte Licht-Energie von jeglichen Oberflächen in Betracht gezogen. Dazu zählen sowohl selbstleuchtende Flächen als auch diffus reflektierende Oberflächen.

Die Funktionsweise beginnt genau so, wie in Raytracing erklärt. Es werden ebenfalls Primärstrahlen für jeden Pixel in der Bildebene erstellt und verfolgt. Dies geschieht allerdings nicht nur einmal, sondern wird solange wiederholt, bis eine maximale Anzahl an *Samplen* erreicht ist. Diese Wiederholungen werden für das *Monte-Carlo-Sampling* (siehe [Ham60]) gebraucht, welches genutzt wird um das Integral aus 2 näherungsweise zu lösen. Die Vorgehensweise des Monte-Carlo-Sampling wird im Abschnitt zur „Richtungsbestimmung eines neuen Bounce“ weiter erläutert. Im Folgenden wird zunächst beschrieben, wie die Farbe eines einzelnen Pixels für eine Iteration (ein Sample) berechnet wird:
Nachdem ein Schnittpunkt eines Strahles mit einem Primitiv gefunden wird, ver-

folgt man den Pfad, welcher der Strahl nimmt, weiter bis zu einer maximalen Tiefe. Dabei wird von jedem Schnittpunkt (auch *Bounce* genannt) auf dem Pfad, die dort anliegende Lichtmenge mittels einer BSDF (siehe 2.1.3) bestimmt. Mithilfe dieser Lichtmenge wird die Farbe des Schnittpunktes nach der Rendering-Gleichung 2 berechnet. In Abbildung 3 ist ein solcher Pfad in rot zu sehen. Es würden in diesem Beispiel für jeden der 3 Bounces eine Farbe berechnet werden, also 3 einzelne Farben. Jede Farbe des Pfades wird anschließend aufsummiert und jeweils so gewichtet, dass spätere Schnittpunkte auf dem Pfad weniger Einfluss auf die Endfarbe des Pixels haben. Die genannte Gewichtung wird durch Division mit einer *Wahrscheinlichkeitsdichtefunktion* (Probability Density Function, PDF) erreicht. Für den Fall, dass die PDF den Wert 0 erreicht, also dieser Schnittpunkt keinen Einfluss mehr auf die Endfarbe hat, wird die Verfolgung des Pfades abgebrochen (auch dann, wenn die maximale Tiefe noch nicht erreicht ist) und die bis dahin berechnete Endfarbe für den Pixel genutzt. In der Theorie wird ein Pfad nur dann beleuchtet, wenn er am Ende auf eine Lichtquelle trifft. In der Praxis würde dies dazu führen, dass eine große Menge an Pfaden verfolgt werden, die am Ende keinen Einfluss auf die Beleuchtung haben. Im folgenden Abschnitt wird ersichtlich, weshalb viele Pfade nicht auf eine Lichtquelle treffen. Aus diesem Grund wird oft ein zusätzlicher Schattenfühler von einem Schnittpunkt in Richtung einer Lichtquelle verschossen, mithilfe dessen die direkte Beleuchtung an diesem Schnittpunkt berechnet wird.
In diesem Abschnitt wird die Richtungsbestimmung eines neuen Bounces erläutert. Wie zuvor bereits erwähnt wird ein *Monte-Carlo-Algorithmus* (siehe [Ham60]) verwendet, um das Integral über die Hemisphäre mathematisch durch Sampling anzunähern. Aus diesem Grund wird das Pathtracing oft als *Monte-Carlo-Pathtracing* bezeichnet. Dabei wird für jeden Bounce die neue, zu verfolgende Richtung zufällig und uniform in der Hemisphäre verteilt ausgewählt. Dieses Auswählen der neuen Richtung wird *Sampling* genannt und wird in vielen Implementationen von der BSDF übernommen. Indem für jedes neue Sample eine andere Richtung verfolgt wird, decken die Strahlen die komplette Hemisphäre ab, wenn man die Gesamtheit aller Samples zusammen nimmt. In Abbildung 3 ist dieses Vorgehen verdeutlicht. Dort ist in Rot der beim aktuellen Sample verfolgte Pfad zu sehen, während die gelben Pfeile die Richtungen anzeigen, welche der Pfad in zukünftigen Samplen nehmen könnte.
Das erläuterte Vorgehen, bei dem die Strahlen von der Kamera verschossen werden und versuchen eine Lichtquelle zu treffen, nennt man *Backwards-Pathtracing* (Rückwärts-Pfaderstellung). Es ist ebenfalls möglich mit den Strahlen von der Lichtquelle aus zu starten und zu versuchen die Kamera zu erreichen. Da dies den physikalischen Grundlagen (Lichtstrahlen werden von Lichtquelle versandt) entspricht, nennt man dies *Forward-Pathtracing* (Vorwärts-Pfaderstellung).
Die Gesamtfarbe eines Pixels wird bestimmt, indem die Farben aller einzelnen Samples zusammenaddiert werden. Anschließend wird die Farbe mithilfe der maximalen Anzahl an Samplen gemittelt. Da es sich bei der Pfadbestimmung um einen Monte-Carlo-Algorithmus handelt, nähert sich das Ergebnisbild an das „physikalisch korrekte“ Bild an, je mehr Samples berechnet werden. Durch eine größe-

re Gesamtzahl an Samples, haben einzelne Samples, welche möglicherweise eine falsche Farbe berechnen, einen geringeren Einfluss auf das Endergebnis. Dies ist ebenfalls der Grund dafür, dass das Bild weniger verrauscht ist, je höher die Anzahl an Samples ist.

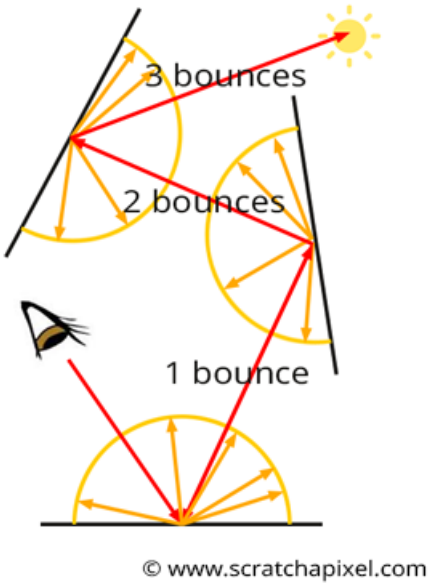


Abbildung 3: *Backwards*-Pathtracing-Verfahren. Pfad den ein einzelner Strahl eines Samples vom Augpunkt (Kamera) aus nimmt. Die neue Richtung eines jeden Bounce wird mithilfe einer BRDF berechnet, welche dem Material des jeweiligen Schnittpunktes zugeordnet ist. Bildquelle: [Scr]

2.2 Datenstrukturen

Wie schon in den Kapiteln 2.1.4 und 2.1.5 beschrieben, muss sowohl beim Raytracing als auch beim Pathtracing jeder verschossene Strahl Schnittpunkttests mit jedem einzelnen Primitiv der Szene durchführen. Besonders bei größeren Szenen muss so eine hohe Anzahl an Schnittpunkten berechnet werden und je nach Objekttyp sind diese Schnittpunkttests sehr rechenintensiv. Im Normalfall ist es so, dass ein Strahl nur auf einen Teil der Objekte trifft. Aus diesem Grund ist es sinnvoll die Liste mit den auf Schnittpunkte zu testenden Objekten effizient einzuschränken. Um dies umzusetzen gibt es Beschleunigungsdatenstrukturen. In der Literatur sind verschiedene Verfahren bekannt, wie das *Uniform-Grid* (siehe [FNK+89]) oder die auf Bäumen basierenden Datenstrukturen *Octree* (siehe [SVNB99]), *KD-Tree* (siehe [BF79]) und *Bounding-Volume-Hierarchie* (siehe [KHM+98]) (kurz: BVH). Letztere ist die in der Praxis am weitesten verbreitete Datenstruktur und ist auch in der Implementation dieser Arbeit genutzt worden. Dies liegt unter anderem daran, dass BVH's einfach zu implementieren sind und der mittlere Traversierungs-Aufwand bei einer Szene mit n Objekten nun bei $O(\log n)$ statt bei $O(n)$, wie ohne Datenstruktur, liegt (siehe [SS14]). Im folgenden Abschnitt wird auf die genaue Funktionsweise, sowie auf unterschiedliche Implementationsmöglichkeiten und weitere Gründe, weshalb die BVH weit verbreitet ist, näher eingegangen.

2.2.1 Bounding-Volume-Hierarchie

Bei dieser Datenstruktur handelt es sich um einen hierarchisch aufgebauten Baum. Die einzelnen Knoten bestehen dabei aus sogenannten *Bounding-Volumes* (kurz: BV), woraus sich auch der Name dieser Datenstruktur ableiten lässt.

Bounding-Volumes sind eine Art Hülle oder Hüllkörper, welche entweder einzelne Primitive, Gruppen von Primitiven oder, in der hierarchischen Baumstruktur, andere Bounding-Volumes enthalten. Wichtig ist, dass das BV folgende Kriterien erfüllt:

- Die zu umgebenden Objekte müssen vollständig innerhalb der Hülle liegen
- Das Bounding-Volume muss „minimal“ sein, das heißt es muss die Objekte so kompakt wie möglich umschließen
- Der Schnittpunkttest mit einem Strahl muss schnell und effizient zu berechnen sein

Der Baum einer BVH ist so aufgebaut, dass einzig die Blättern Primitive enthalten. Die restlichen BV's umfassen lediglich andere *Bounding-Volumes*. In Abbildung 4 ist auf der rechten Seite die hierarchische Baumstruktur dargestellt, welche aus den BV's der linken Seite erstellt wurde. Dort ist zu sehen, dass das BV eines Vaterknotens alle BV's seiner Kindknoten, also seines restlichen Teilbaumes,

umschließt. Die Wurzel des Baumes (BV 8 in der Abbildung 4) enthält somit die gesamte Szene, während die Blattknoten die Primitive 1 bis 7 enthalten.
Um den Baum aus der Szene zu konstruieren gibt es drei verschiedene Typen von Ansätzen: *Insertion-Methoden* [BHH15], *Bottom-Up-Methoden* [GHFB13] und *Top-Down-Methoden* [Wal07].
Die unter die Kategorie *Insertion* fallenden Methoden, beginnen mit einem leeren Baum und fügen die Objekte zur Laufzeit ein (*insertion*). Es wird dabei an der Stelle in den Baum eingefügt, welche ihn am wenigsten wachsen lässt. Im Gegensatz zu den anderen beiden Typen wird beim *Insertion-Typ* nicht die gesamte Szene benötigt um mit dem Erstellen des Baumes zu beginnen, weshalb diese Methoden gut geeignet sind um den Baum zur Laufzeit zu aktualisieren. Besonders bei dynamischen Szenen mit sich bewegenden Objekten ist dies von Vorteil, denn es muss nicht bei jedem Frame die gesamte Hierarchie neu aufgebaut werden.
Bottom-Up-Methoden beginnen mit der untersten Schicht des Baumes, also den Blättern. Es werden dann zwei Blätter unter einem neuen Knoten zusammengefügt und dies wird rekursiv wiederholt, bis nur noch ein Knoten (der Wurzel-Knoten) übrig ist. Das Implementieren dieser Methode ist schwieriger als die *Top-Down-Methode*, denn man muss die Objekte, welche zusammengefügt werden sollen, sinnvoll auswählen. Mit diesem Auswählen befasst sich das *Clustering*. Ein Beispiel dafür ist das *Approximate-Agglomerative-Clustering*-Verfahren (siehe [GHFB13]), welches eine BVH erstellt, die zu geringeren Kosten beim Raytracing führt, als mit *Top-Down* erstellte BVH's.
Die letzte Kategorie enthält die *Top-Down-Methoden*. Sie fangen mit der Wurzel an und teilen dann die gesamte Szene mithilfe von „Splitting-Planes" in zwei *Bounding-Volumes*. Dies wird solange rekursiv wiederholt bis jeder Teilbaum bei einem Blattknoten angekommen ist, welche nur noch die Primitive enthalten. Der Vorteil dieser Methoden ist, dass sie simpel zu implementieren sind und das Konstruieren schnell geht. Die meisten dieser Konstruktions-Verfahren nutzen die *SAH* (siehe [Wal07]), welche auch in der Implementation dieser Arbeit verwendet wird.

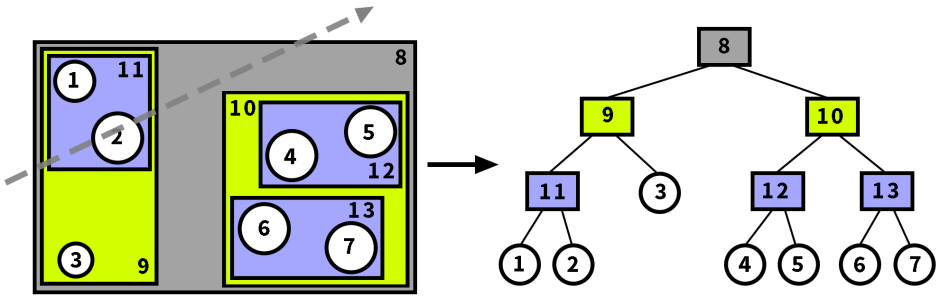


Abbildung 4: Aus *Bounding-Volumes* entstandene Baumstruktur.

Die Beschleunigung durch eine BVH beim Raytracing kommt dadurch zustande, dass nicht mehr die Primitive selbst, sondern die BV's auf einen Schnittpunkt getestet werden. Dieser Test ist nach obiger Definition effizient und schnell zu berechnen. Lediglich die Primitive, welche nach Traversieren des Baumes als Blatt übrig bleiben, werden selbst auf einen Schnittpunkt mit dem aktuellen Strahl getestet. Darüber hinaus wird durch die hierarchische Struktur des Baumes nicht jeder einzelne Knoten und somit nicht jedes BV getestet. Wenn der Schnittpunkttest mit einem BV fehlschlägt, kann der gesamte Teilbaum, welcher an diesem BV hängt, ignoriert werden. In Abbildung 4 wird der gesamte rechte Teilbaum übersprungen, nachdem der Schnittpunkttest mit 10 fehlgeschlagen ist. Konkret für das Beispiel aus Abbildung 4 würden folgende Tests gemacht werden müssen:

- Schnitt mit 8 erfolgreich, also
  - Schnitt mit 9 erfolgreich
    - Schnitt mit 11 erfolgreich
      - Schnitt mit 1 fehlgeschlagen
      - Schnitt mit 2 erfolgreich, da Primitiv Abbruch
    - Schnitt mit 3 fehlgeschlagen
  - Schnitt mit 10 fehlgeschlagen

Somit kommt man insgesamt auf 4 schnelle Tests mit *Bounding-Volumes* und 3 langsame Tests mit Primitiven. Im Vergleich dazu würden ohne Datenstruktur 7 langsame Tests mit Primitiven gemacht werden müssen.

3 Rendering von Fell

Fell, also die Körperbehaarung von Tieren, besteht, wie bereits in der Einleitung erwähnt, aus einzelnen Haar-Fasern. Laut der Definition von [DS86] spricht man von Fell, sobald die Anzahl an Haaren pro Quadratzentimeter Haut in einem Bereich von 50 bis 400 Haaren liegt. Dies ist bei Tieren wie Wölfen oder Hasen der Fall. Wenn die Haardichte geringer ist, wie bei den Menschen, dann wird von haararmer Haut gesprochen. Dichter behaarte Haut, wie die eines Fuchses, wird Pelz genannt. Um mithilfe des in Kapitel 2.1.5 vorgestellten Pathtracing-Verfahrens realistisch aussehende Haare und somit Fell rendern zu können, muss zunächst der grundlegende Aufbau und die wichtigsten Eigenschaften einer einzelnen Haar-Faser untersucht werden. Dies wird im folgenden Kapitel Eigenschaften von Haaren getan. Basierend auf diesen Ergebnissen haben Marschner et. al. in [MJC+03] die Lichtstreuung, welche stattfindet, wenn Licht auf eine Haar-Faser trifft, untersucht. Des weiteren präsentieren Marschner et. al. ein Beleuchtungsmodell, welches versucht die von ihnen gewonnenen Erkenntnisse zu simulieren, um so Haare in einem Renderer zu visualisieren. Dieses Verfahren wird auch in der Implementation dieser Arbeit umgesetzt und in Kaptiel 3.2 beschrieben.

3.1 Eigenschaften von Haaren

Ein einzelnes Haar ist in seiner Gesamtheit geometrisch ähnlich wie ein Zylinder geformt, welcher sich zur Spitze hin verjüngt. Diese zylindrische Form ist in Abbildung 5 zu erkennen. Die Haar-Faser besteht überwiegend aus Keratin und ist in drei Schichten aufgebaut: Die äußerste Schuppenschicht, auch *Cuticula* genannt. Auf diese folgt die Faserschicht, auch *Cortex* genannt, und zuletzt im inneren Zentrum die *Medulla*. Der in diesem Kapitel beschriebene Aufbau ist in [OA79] erklärt.

Die Cuticula besteht aus mehreren flachen, sich überlagernden Schichten abgestorbener, verhornter Zellen. Diese Schichten sind durch den überlagernden Aufbau nicht ganz eben und neigen sich in Richtung der Haarwurzel, vergleichbar mit den Schuppen eines Tannenzapfens. In Abbildung 6 ist dies schematisch darge-

Abbildung 5: Der Aufbau einer einzelnen menschlichen Haar-Faser. Bildquelle: [bF]

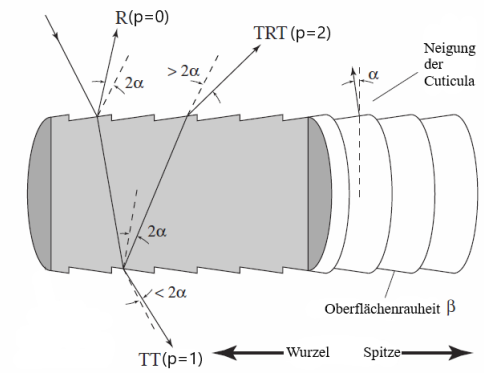
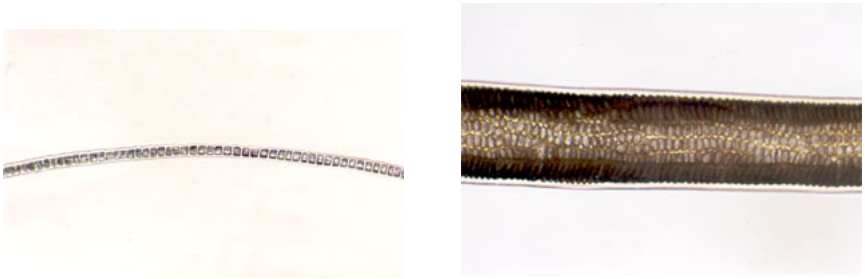


Abbildung 6: Neigung der Oberfläche einer Haar-Faser, sowie die Lichtpfade nach dem Modell von Marschner et al.. Die gestrichelten Linien stellen die Winkel der Lichtstreuung für einen glatten und ebenen Zylinder dar. Bildquelle: [MJC+03]

stellt mit einer longitudinalen Verschiebung α (engl. *longitudinal shift*) der Haaroberfläche. Diese Neigung α in Richtung Haarwurzel beträgt laut [BS91] etwa 3 Grad. Zusätzlich führt der Aufbau der Cuticula zu einer Rauheit der Oberfläche, notiert mit β (engl. *longitudinal width*). Aufgrund dieser Schuppenschicht wird Licht, welches auf ein Haar trifft, nicht perfekt nach dem Reflexionsgesetz 3 reflektiert (siehe [BS91]). Die leichte Neigung des Haares muss in einem Renderer beachtet werden. Des weiteren ist die Cuticula durchscheinend, was es dem Licht ermöglicht sich an dem Aufprallpunkt zu brechen und in das Haar einzudringen. Die Lichtbrechung eines Haares kann somit, wie in Kapitel 2.1.2 beschrieben, mit Hilfe des Brechungsgesetzes 4 berechnet werden. Dabei hat der Brechungsindex η eines Haares laut Stamm et. al. [SGF77] einen Wert von 1,55. Luft hingegen hat einen Wert von ungefähr 1 (siehe [SGF77]), woraus folgt, dass sich das Licht innerhalb eines Haares langsamer ausbreitet als in Luft. Wie in Abbildung 5 zu sehen, macht den Hauptteil der Haar-Faser (etwa 88%, vergl. [BS91]) der Cortex aus. Dieser enthält Melaninpigmente, welche für die Farbe, in der wir das Haar sehen, verantwortlich sind. Das an der Oberfläche gebrochene Licht durchdringt den Cortex und trifft dabei auf die Farbpigmente, welche dann einen Teil des Lichtes selektiv absorbieren und so die Farbe des Lichtes verändern. Bei den Pigmenten handelt es sich um *Eumelanin* und *Phäomelanin*, die in unterschiedlichen Mengenverhältnissen vorhanden sein können und so zu verschiedenen Haar-/Fellfarben führen (siehe [Zah89]). Wenn keines der Pigmente mehr vom Körper produziert wird, wie es mit hohem Alter eines Menschen der Fall ist, dann wird kein Licht im Cortex absorbiert und somit erscheint das Haar weiß. Im inneren Zentrum des Haares befindet sich die Medulla. Diese ist je nach Haartyp unterschiedlich in Größe und Zusammensetzung. Im menschlichen Haar ist die Medulla entweder nicht sehr ausgeprägt und wird teilweise vom Cortex durchbrochen oder sie ist nicht vorhanden(siehe [DK04a]). Aus diesem Grund hat sie bei einer menschlichen Haar-Faser



(a) Uniserial-Ladder Medulla **(b)** Multiserial-Ladder Medulla

Abbildung 7: Mikrophotographie der Medulla von zwei unterschiedlichen Typen von Hasenhaaren. Bildquelle: [DK04b]

auch keinen sichtbaren Einfluss auf die Lichtstreuung eines Haares. Dies gilt für alle Haartypen mit einer kleinen oder nicht vorhandenen Medulla. Bei Haartypen, welche eine stark ausgeprägte Medulla besitzen, ändert diese die Lichtstreuung. Denn im Gegensatz zum Cortex ist die Medulla nicht immer lichtdurchlässig und kann Lichtstrahlen reflektieren, welche auf sie treffen. Somit ist die Reflexionsfläche des Haares größer und Fell mit einem solchen Haartypen erscheint heller als viele Felle von Tieren mit einer schwach ausgeprägten Medulla. Da dies nicht der einzige Grund für das Zustandekommen der Fellfarbe ist gibt es auch Ausnahmen. So hat ein Reh eine ausgeprägtere Medulla als ein Meerschweinchen (siehe [DK04b]), dennoch gibt es viele Meerschweinchen mit weißem oder orangem Fell, welches heller ist als das Fell eines braun farbigen Rehs. Dass die Zusammensetzung eines Haares stark variiert ist am Fell eines Hasen zu erkennen. In [DK04b] sind Haar-Fasern im Ganzen, sowie deren Medulla separat untersucht worden. Die Medulla von zwei verschiedenen Typen von Hasenhaaren ist in Abbildung 7 zu sehen. Die linke Medulla ist kleiner als die rechte und an ihrer durchsichtigen Farbe ist zu erkenne, dass sie Lichtdurchlässig ist und wenig Licht reflektiert. Die rechte Medulla hingegen ist dicker und besitzt mehr Schichten. Die dunkle Farbe deutet darauf hin, dass sie das meiste Licht reflektiert.

Das Licht streut beim Auftreffen auf ein Haar in verschiedene Richtungen (siehe [BS91]) und es entstehen somit verschiedene Pfade, welche vom Licht genommen werden. Wie zuvor erwähnt, wird ein Teil des Lichtes an der Cuticula reflektiert und ein weiterer Teil wird refraktiert und dringt in den Cortex ein. Dieser refraktierte Teil durchdringt nun das Haar, bis er auf die Cuticula an der anderen Seite des Haares trifft. Dort angekommen kann das Licht sich weiter aufteilen, indem es sich an der Cuticula erneut bricht und das Haar verlässt oder von dieser reflektiert wird und im inneren des Haares verbleibt. Der Teil, welcher im Haar bleibt, fließt weiter bis er wieder auf die äußere Schicht des Haares trifft. An dieser Stelle kann er sich, wie schon zuvor, wieder aufteilen. Dadurch entstehen viele verschiedene Pfade durch innere Reflexion und das Licht kann verschiedenen Stellen aus dem Haar austreten. In Abbildung 6 sind die ersten 3 Pfade, die das Licht nehmen kann

bis es das Haar wieder verlässt, dargestellt. Diese sind eine einfache Reflexion (In Abbildung 6 R (p = 0)), eine doppelte Refraktion (In Abbildung 6 TT (p=1)) und eine doppelte Refraktion mit einer inneren Reflexion dazwischen (In Abbildung 6 TRT (p=2)). Ebenfalls in der Abbildung 6 zu sehen ist die Veränderung der Austrittswinkel des Lichtes, welche durch die bereits genannte Neigung der Cuticula α hervorgerufen wird. Nach dem Energieerhaltungssatz 5 wird die Menge an Licht, die ein Strahl beim Austreten aus dem Haar noch besitzt geringer, je mehr Reflexionen oder Refraktion der Strahl im Haar genommen hat. Für den Fall, dass das Haar eine Medulla besitzt, kann diese das Licht zusätzlich ablenken, bevor es auf die gegenüberliegende Seite der Cuticula trifft. Weiteren Einfluss auf die aus Entfernung wahrgenommene Helligkeit eines Haares hat der Durchmesser der einzelnen Fasern. Ist dieser größer wirkt das Haar heller, was mehrere Gründe hat. Zum einen besitzt die Cuticula eine größere Fläche um Licht zu reflektieren. Zum anderen ist der Weg, den das Licht durch das Haar wandert, größer und somit trifft es auf mehr Farbpigmente. Das führt dazu, dass mehr Licht absorbiert wird und die Farbe des Haares intensiver erscheint (siehe [BS91]).

Die dargestellten Unterschiede bei verschiedenen Fell- und somit Haartypen machen es schwierig ein verallgemeinertes und physikalisch korrektes Beleuchtungsmodell zu erstellen, mit dem jegliche Arten von Haaren, ob Hasenfell oder menschliches Haar, visualisiert werden kann. Wenn im Modell die Medulla in die Geometrie der Haare eingebaut wird, um physikalisch korrekt reflektiertes Licht zu simulieren, dann kann dieses Modell nicht ohne weiteres für ein menschliches Haar ohne Medulla genutzt werden. Zusätzlich hat sie keinen Einfluss auf die diffuse Farbe des Haares, sondern erhöht den spekularen Anteil des reflektierten Lichtes. Und wie zuvor erklärt, haben öfter abgelenkte Strahlen eine geringeren Menge an Energie und somit weniger Einfluss auf die Farbe. Aus diesen Gründen wird in dieser Arbeit eine Verallgemeinerung für Fell beziehungsweise Haare getroffen, nach der die Medulla im Beleuchtungsmodell, sowie in der Geometrie der Haare nicht berücksichtigt wird. Somit können ein Großteil der Haare/Felle realistisch und der restliche Teil optisch beinahe, aber nach dieser Verallgemeinerung ebenfalls realistisch dargestellt werden. Das in folgendem Kapitel 3.2 erläuterte Modell [MJC$^+$03] bezieht sich auf Haare ohne Medulla und stellt für diese ein Beleuchtungsmodell vor, welches in dieser Arbeit implementiert und Aufgrund der Verallgemeinerung für Fell genutzt wird.

3.2 Marschner et al.: Lichtstreuung von Haar-Fasern

Um die Lichtstreuung von menschlichem Haar mithilfe eines auf Raytracing basierenden Rendering-Verfahrens (siehe Kapitel 2.1.4 und 2.1.5) simulieren zu können und so fotorealistische Bilder von Haaren aus einer 3D-Szene erstellen zu können haben Marschner et. al. in [MJC$^+$03] im Jahr 2003 ein Beleuchtungsmodell vorgestellt. Wichtigster Aspekt dieses Modelles ist die vorgestellte *scattering function*, also eine BSDF welche unter die Kategorie der BSSRDF's fällt (siehe Kapitel

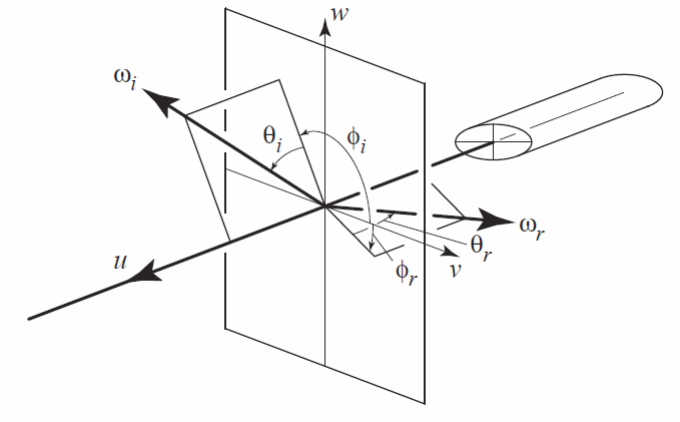


Abbildung 8: Notationen im lokalen Haar-Koordinatensystem nach Marschner et al. Bildquelle: [MJC+03]

2.1.3). Dazu haben Marschner et. al. die Lichtstreuung echter Haare mit einer festen Lichtquelle und sich änderndem Einfalls- und Ausfallwinkels gemessen. Nach Analyse dieser Ergebnisse konnte festgestellt werden, wo und unter welchen Umständen primäre sowie sekundäre Highlights bei Haaren auftreten. Die entwickelte BSDF versucht auf Basis analytischer Vorgehensweise diese Highlights in einem virtuellen Haar zu simulieren. Marschner's Modell hat sich zur Grundlage für die meisten der heute genutzten Verfahren entwickelt, wie die Modelle von D'Eon et. al. [dFH+11] aus dem Jahr 2011 oder Chiang et. al. [CBTB16] aus 2016. Des weiteren ist Marschner's Modell von bekannten Firmen, wie Pixar (siehe [PHVL15]) oder ATI (siehe [Sch04]), implementiert und in ihre Rendering-System integriert worden. Somit ist die Wahl zum Implementieren eines Modelles auf Marschner et. al. gefallen, welches für Haare ohne Medulla genutzt wird, wie es die in 3.1 getroffene Verallgemeinerung auch für Fell vorsieht.

3.2.1 Lichtstreuung

Zunächst werden die im folgenden genutzten Notation und Variablen festgelegt. Dabei wird sich an die von Marschner genutzten Notationen gehalten, welche für die Geometrie einer einzelnen Haar-Faser in Abbildung 8 zu sehen sind. Das Haar bildet ein lokales Koordinatensystem (KS) bestehend aus den *uvw*-Achsen in Abbildung 8. Dieses wird auch als *BSDF*-Koordinatensystem bezeichnet. Die Tangente des Haares, welche Zentral im Haar von der Wurzel in Richtung Spitze verläuft, ist der Vektor $\vec{u}$. Dieser bildet die x-Achse des lokalen KS, welche fortan als *u*-Achse bezeichnet wird. Betrachtet man das Haar als elliptisches Objekt, dann ist Vektor $\vec{v}$ die *Hauptachse* und Vektor $\vec{w}$ die *Nebenachse*. Beide stehen jeweils orthogonal zu $\vec{u}$ und komplettieren eine *Orthonormalbasis* bestehend aus diesen 3 Vektoren. Bei dem resultierenden KS handelt es sich um ein *Rechtssystem* (rechtshändiges KS) mit $\vec{v}$ als y-Achse, fortan *v*-Achse genannt, und $\vec{w}$ als z-Achse, fortan

w-Achse. Die $\vec{v} - \vec{w}$-Ebene wird als *Normalenebene* bezeichnet. Geht man von einem Querschnitt des Haares aus, dann befinden sich in dieser Normalenebene alle Vektoren, welche eine Normale für einen Schnittpunkt in diesem Punkt darstellen können. Die Richtung des Lichteinfalls, also ein Vektor zu einer Lichtquelle, ist mit $\vec{\omega_i}$ notiert. Gegenüberstehend ist der Vektor $\vec{\omega_r}$ die Richtung, in welche das Licht von der Oberfläche aus gestreut wird, berechnet durch die BSDF. Beide Richtungen zeigen vom Zentrum, dem Schnittpunkt, weg. Sowohl $\vec{\omega_i}$ als auch $\vec{\omega_r}$ werden in *Polar-Koordinaten* (auch Kugel-Koordinaten genannt) angegeben, da die Winkel aus dieser Darstellung θ (Theta) und ϕ (Phi) für die meisten Berechnungen benötigt werden. Die Neigung zwischen $\vec{\omega_i}$ bzw. $\vec{\omega_r}$ und Normalenebene ist der Winkel θ_i bzw. θ_r. Diese sind jeweils so gemessen, dass ein Winkel von $0°$ bedeutet, dass der Vektor senkrecht zum Haar steht und sich somit in der Normalenebene befindet. Bei einem θ Winkel von $90°$ ist der Vektor auf Höhe der positiven *u*-Achse ($\vec{u}$) und bei $-90°$ auf Höhe der negativen *u*-Achse ($-\vec{u}$). Daraus resultiert für θ_i und θ_r ein Wertebereich von $[-90°, 90°]$. Die Azimut-Winkel, also der Winkel zwischen *v*-Achse und orthogonal auf Normalenebene projiziertem Vektor $\vec{\omega_i}$ bzw. $\vec{\omega_r}$, werden mit ϕ_i und ϕ_r notiert. Gemessen sind diese so, dass $0°$ der positiven *v*-Achse ($\vec{v}$) entspricht, $90°$ der positiven *w*-Achse ($\vec{w}$) und $180°$ der negativen *v*-Achse ($-\vec{v}$) . Dies ergibt einen Wertebereich für ϕ_i und ϕ_r von $[-180°, 180°]$. Mit der Abbildung 8 können die genannten Winkel, sowie deren Wertebereiche nachvollzogen werden.
Aus diesen Winkeln werden weitere abgeleitet, welche in folgender Auflistung erläutert sind:

$\theta_d = (\theta_r - \theta_i)/2$ Theta Differenzwinkel (*difference angle*)

$\theta_h = (\theta_i + \theta_r)/2$ Theta Halbwinkel (*half angle*)

$\phi_h = (\phi_i + \phi_r)/2$ Phi Halbwinkel (*half angle*)

$\phi = \phi_r - \phi_i$ Relativer Azimut-Winkel (*relativ azimuth*)

Mit diesen Winkeln kann die bidirektionale Streuungsfunktion S berechnet werden. Diese teilt sich die physikalischen Einheiten mit der BRDF in Gleichung 6, allerdings sind die Strahldichte und Bestrahlungsstärke über die gesamte Kurve des Haares angegeben und nicht nur über dessen Oberfläche. Dies ist der Grund, weshalb S, wie zuvor bereits erwähnt, zu den BSSRDF's zählt. Um diesen Unterschied zu verdeutlichen, wird die Strahldichte mit $\bar{L}$ und die Bestrahlungsstärke mit $\bar{E}$ angegeben, was zu folgender Gleichung 8 für Marschner's Streuungsfunktion führt:

$$S(\vec{\omega_i}, \vec{\omega_r}) = \frac{d\bar{L}_r(\vec{\omega_r})}{d\bar{E}_i(\vec{\omega_i})} \tag{8}$$

Die in Kapitel 2.1.3 erläuterte Proportionalität zwischen Bestrahlungsstärke und eingehender Strahldichte ist auch bei $\bar{E}_i$ gegeben, mit dem Unterschied, dass der Durchmesser D der Haar-Faser mit einbezogen wird:

$$d\bar{E}_i(\vec{\omega_i}) = D \cdot L_i(\vec{\omega_i}) \cdot \cos\theta_i \cdot d\omega_i \tag{9}$$

Aus Gleichungen 8 und 9 folgt die speziell für Haare geltende, abgewandelte Rendering-Gleichung 10 :

$$\bar{L_r}(\vec{\omega_r}) = D \int S(\vec{\omega_i}, \vec{\omega_r}) \cdot L_i(x, \vec{\omega_i}) \cdot \cos\theta_i \cdot d\omega_i \tag{10}$$

Da die Streuungsfunktion S die gesamte Haarkurve inklusive dessen Inneres, also auch refraktierte Strahlen, einbezieht, erstreckt sich das Integral aus Gleichung 10 über die gesamte Hemisphäre (positive und negative) eines Schnittpunktes x. Das Einbeziehen des Haardurchmessers D führt zu der Eigenschaft, dass dicke Haar-Fasern aus größerer Entfernung heller erscheinen als dünne Haar-Fasern, wie bereits in 3.1 erklärt.

3.2.2 Streuungsfunktion S

Marschner et. al. teilt die 4-dimensionale Streuungsfunktion S in ein Produkt von zwei 2-dimensionalen Termen auf. Diese sind die longitudinale Streuungsfunktion M, welche abhängig von θ ist und die azimutale Streuungsfunktion N, abhängig von ϕ. Für Objekte mit einem kreisförmigen Querschnitt, wie ein Zylinder, welcher in dieser Arbeit ein Haar repräsentiert, werden zudem Funktionen für M (folgt in Kapitel 3.2.4) und N (folgt in Kapitel 3.2.5) vorgestellt. Nach dieser Aufteilung sieht die gesamte Streuungsfunktion für glatte Zylinder mit einem kreisförmigen Querschnitt wie folgt aus:

$$S(\phi_i, \theta_i; \phi_r, \theta_r) = M(\theta_i, \theta_r) \cdot N(\eta^{'}(\theta_d); \phi_i, \phi_r) / \cos^2\theta_d \tag{11}$$

Der erste Faktor dieser Gleichung 11 deutet an, dass Licht nur in einem spekularen Kegel gestreut wird. Um den projizierten Raumwinkel dieses spekularen Kegels auszugleichen ist der Divisor $\cos^2\theta_d$ der Gleichung hinzugefügt worden. Bei $\eta^{'}$ handelt es sich um einen virtuellen Brechungsindex. Dieser sogenannte *Bravais-Index* (siehe [Tri71]) wird genutzt, da beim Modellieren der physikalisch korrekten Lichstreuung eines Zylinder auf eine 2D-Umgebung der Brechungsindex dieses Zylinders vom Lichteinfallswinkel abhängig ist. Die Gleichungen um den Bravais-Index zu berechnen werden im weiteren Verlauf in Kapitel 3.2.3 vorgestellt.
Wie bereits bei den Eigenschaften von Haaren beschrieben, kann Licht beim Auftreffen auf ein Haar verschiedene Pfade verfolgen. Marschner et. al. bezeichnen mit der Variable p den aktuell betrachteten Lichtpfad, wobei $p + 1$ die Anzahl an Reflexionen und Refraktionen ist, die das Licht auf dem aktuellen Pfad nimmt. Da die Energie eines Lichtpfades und somit auch der Einfluss auf den *Scattering*-Wert kleiner wird, je größer p wird, haben Marschner et. al. entschieden alle Pfade mit $p > 2$ zu ignorieren. In Abbildung 9 sind alle von diesem Modell genutzten Pfade zu sehen, welche in folgender Aufzählung erläutert sind:

R-Pfad, $p = 0$ Einfache Reflexion an der Oberfläche. Stellt ein primäres, spekulares Highlight in Richtung Haarwurzel dar. Tritt ungefähr bei $\theta_r = -\theta_i$ auf. Aufgrund der in Kapitel 3.1 beschriebenen Neigung der Haaroberfläche verschiebt sich dieses Highlight bei realen Haaren um 6 bis 10 Grad in Richtung Haarwurzel.

TT-Pfad, $p = 1$ Refraktive Transmission. Tritt stärker bei hellen, farbigen Haaren auf und bringt einen Farbanteil mit sich. Führt dazu, dass farbiges Haar sehr hell und durchscheinend aussieht, wenn es von hinten beleuchtet wird.

TRT-Pfad, $p = 2$ Einmalige interne Reflexion. Stellt ein sekundäres Highlight über dem primären dar und bringt den Haupt-Farbanteil des Haares. Enthält meist 2 Höhepunkte (*glints*), deren Position abhängig von θ sind.

Diese 3 Pfade zusammengenommen machen den visuell größten Anteil für das Aussehen eines Haares aus. Dennoch gehen durch das Beschränken auf diese Pfade Informationen über das Aussehen des Haares verloren und somit ist das Modell nicht energieerhaltend. Visuell handelt es sich bei diesen um sekundäre Brennpunkte, durch die nebensächliche Highlights entstehen. Aufgrund der Aufteilung in diese 3 Pfade ergibt sich die Gleichung 12 für die gesamte Streuungsfunktion S, als Summe der einzelnen Lichtpfade:

$$\begin{aligned} S(\phi_i, \theta_i; \phi_r, \theta_r) = {} & M_R(\theta_h) N_R(\eta'(\eta, \theta_d); \phi) / \cos^2 \theta_d + \\ & M_{TT}(\theta_h) N_{TT}(\eta'(\eta, \theta_d); \phi) / \cos^2 \theta_d + \\ & M_{TRT}(\theta_h) N_{TRT}(\eta'(\eta, \theta_d); \phi) / \cos^2 \theta_d \end{aligned} \tag{12}$$

Die vorgestellten Funktionen M und N sind abhängig von p und variieren für jeden dieser 3 Lichtpfade. Zusätzlich ist N abhängig von dem Austrittswinkel $\phi(h)$. Um diesen zu bestimmen wird analytisch vorgegangen. Dazu muss zunächst der Abstand h zwischen eingehendem Lichtstrahl $\vec{\omega_i}$ und Mittelpunkt des Querschnittes, in Abbildung 9 zu sehen, berechnet werden. Da für den Querschnitt von einem Einheitskreis ausgegangen wird, liegt der Wertebereich für diesen Abstand bei $h = [-1, 1]$. Mithilfe von h können nun der Eintrittwinkel γ_i, sowie der Austrittswinkel des refraktierten Strahles γ_t mit Bezug auf den Einheitskreis des Querschnittes berechnet werden. Für den Eintrittswinkel γ_i gilt Gleichung 13:

$$\sin \gamma_i = h \tag{13}$$

Für den Austrittswinkel γ_t wird in Gleichung 14 zusätzlich der Brechungsindex η', nach *Snell's Law* (Kapitel 2.1.2, Gleichung 4), mit einbezogen:

$$\eta' \sin \gamma_t = h \tag{14}$$

Des weiteren ist aus der Abbildung 9 abzuleiten, wie stark der eingehende Strahl bei jeder Reflexion oder Refraktion von seiner ursprünglichen Richtung abweicht.

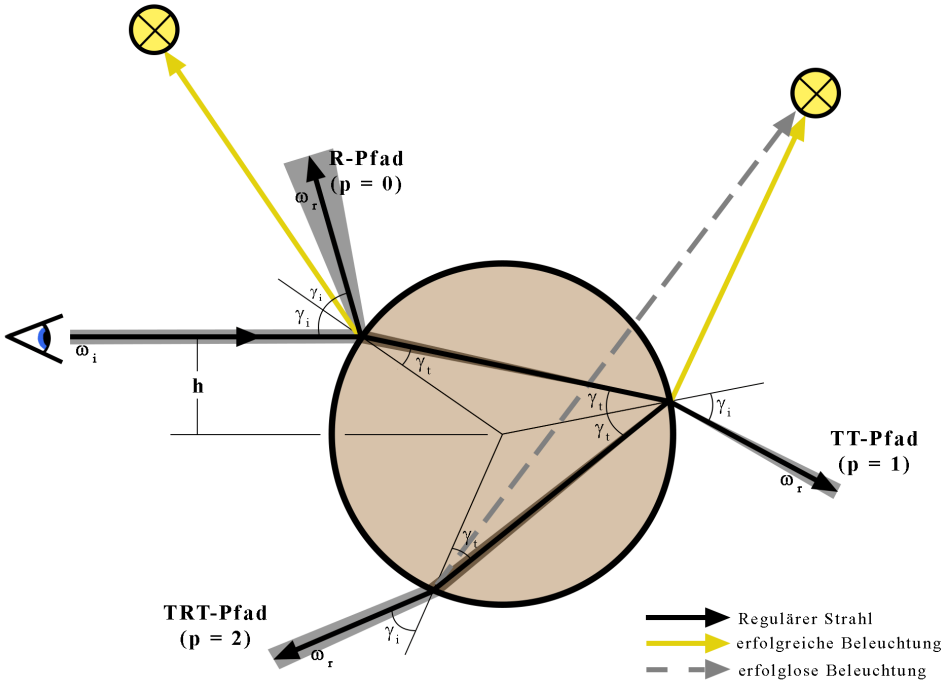


Abbildung 9: Die verschiedenen Pfade, nach Marschner et. al., die Licht beim Auftreffen auf ein Haar nehmen kann.

Beim Eintreten in den Kreis weicht er um $\gamma_t - \gamma_i$ ab und bei jeder haar-internen Reflexion um weitere $\pi + 2\gamma_t$, bis er beim Austreten aus dem Haar um finale $\gamma_t - \gamma_i$ abweicht. Aus diesen einzelnen Teilen ergibt sich Gleichung 15, mithilfe derer der Austrittswinkel $\phi(h)$ aus dem Kreissegment für einen bestimmten, einzelnen Lichtpfad p berechnet werden kann:

$$\phi(p, h) = 2p\gamma_t - 2\gamma_i + p\phi \tag{15}$$

Für die Berechnung des *Scattering*-Wertes wird nicht nur der Wert eines einzelnen Pfades h benötigt, sondern der Einfluss von allen Pfaden in eine Streuungsrichtung ϕ. Diese Werte für h entsprechen den Nullstellen der Gleichung 16, welche als $h(p, r, \phi)$ notiert wird, wobei verschiedene Nullstellen mit verschiedenen r-Werten ausgedrückt werden. Für die Pfade R ($p = 0$) und TT ($p = 1$) ergeben sich dabei immer eine Nullstelle, während der TRT-Pfad ($p = 2$) 2 mögliche Nullstellen besitzt.

$$\phi(p, h) - \phi = 0 \tag{16}$$

Durch Gleichungen 15 und 16 können Werte für ϕ und h hergeleitet werden, was die Berechnung der Strahldichte $\bar{L}(\phi(h))$ unter Berücksichtigung aller eingehenden Strahlen für den Ausfallwinkel $\phi(h)$ ermöglicht. Für ein Intervall dh der einfallenden Lichtstrahlen, welche in ein Intervall von Austrittswinkeln $d\phi$ gestreut

werden, was als graue Hinterlegung der Strahlen in Abbildung 9 zu sehen ist, berechnet man die Strahldichte mit der folgenden Gleichung 17:

$$\bar{L}(\phi(h)) = A(p,h) \left| 2\frac{d\phi}{dh} \right|^{-1} \bar{E} \tag{17}$$

Da ein Einheitskreis mit Durchmesser 2 betrachtet wird, für welchen die Bestrahlungsstärke $\bar{E}$ gilt, enthält die Gleichung den Faktor 2. Bei $A(p,h)$ handelt es sich um einen Lichtdämpfungs-Faktor (engl. *attenuation factor*), der im nächsten Kapitel 3.2.3 näher erläutert wird.

3.2.3 Lichtdämpfung durch Reflexion und Absorption

Die Dämpfung (engl. *attenuation*) des Lichtes beim Durchqueren des Haares hängt von zwei Faktoren ab. Zum einen wird ein Teil des Lichtes bei jeder Reflexion gedämpft und zum anderen wird durch das Durchqueren des Haares ein Teil absorbiert. Aus diesen 2 Teilen ergibt sich der Lichtdämpfungs-Faktor $A(p,h)$, welcher abhängig von h und dem genommenen Pfad p ist, da unterschiedliche Pfade eine unterschiedliche Menge an Reflexionen und somit auch durch das Haar zurückgelegte Wegstrecke haben. Die Dämpfung durch Reflexion wird nach den fresnelschen Formeln (siehe [PJH16]) berechnet und mit dem Term $F(\eta,\gamma)$ für ein Brechungsindex η und einen Reflexionswinkel γ angegeben. Marschner et. al. nutzten für ihr Modell mehrere Generalisierungen für den virtuellen 3D-Raum. Eine von diesen ist der in Kapitel 3.2.2 bereits erwähnte virtuelle Brechungsindex $\eta^{'}$ und $\eta^{''}$, welche zur Berechnung der Fresnel-Terme genutzt werden. Diese können in Abhängigkeit eines Reflexions- oder Brechungswinkels γ (abhängig vom genommenen Pfad) und dem originalen Brechungsindex η mit folgenden Gleichungen 18 für $\eta^{'}$ und 19 für $\eta^{''}$ kalkuliert werden:

$$\eta^{'}(\gamma) = \frac{\sqrt{\eta^2 - \sin^2\gamma}}{\cos\gamma} \tag{18}$$

$$\eta^{''}(\gamma) = \frac{\eta^2 \cos\gamma}{\sqrt{\eta^2 - \sin^2\gamma}} \tag{19}$$

Den zweiten Teil der Lichtdämpfung wird durch die Volumenabsorption des Haares hervorgerufen, die nach dem Energieerhaltungssatz 5 stattfinden beim Durchqueren des Haares. Diese ist abhängig von der Länge der Strecke, welche das Licht durch das Haar genommen hat. Diese Länge wird für jedes Pfad-Segment durch $2 + 2\cos(2\gamma_t)$ berechnet und ist somit indirekt abhängig von h und zeigt außerdem, dass mehr Licht absorbiert wird, je mehr Strecke im Haar zurückgelegt wurde. Die Menge an Licht, die pro Einheitslänge (engl. *unit length*) absorbiert wird ist mit σ_a notiert. Bei einem Haar entspricht eine Einheitslänge dem Radius des Haares. Genannter Absorptionsfaktor σ_a ist der Grund für die bunte Färbung der Haare und ist im Marschner Modell der einzige Parameter über den die Farbe des

3D-Haares gesteuert werden kann. In einem realen Haar setzt sich dieser Absorptionsfaktor aus Melaninen zusammen, wie in Kapitel 3.1 beschrieben. Für σ_a wird eine weitere Generalisierung zum 3D-Modell getroffen. Aufgrund der innerhalb der Haar-Segmente gleichbleibenden Einfallswinkel θ_t zur Kreisachse wird σ_a ersetzt durch $\sigma_a^{'}$ mit Gleichung 20

$$\sigma_a^{'} = \frac{\sigma_a}{\cos\theta_t} \tag{20}$$

Insgesamt wird die Absorption $T(\sigma_a^{'}, h)$ mithilfe der folgenden Gleichung 21 berechnet:

$$T(\sigma_a^{'}, h) = e^{-2\sigma_a^{'}(1+\cos(2\gamma_t))} \tag{21}$$

Fügt man die Dämpfung durch Reflexion und die Absorption innerhalb des Haares zusammen, erhält man mit Gleichung 23 den finalen Lichtdämpfungs-Faktor $A(p, h)$. Einen Sonderfall stellt dabei der R-Pfad ($p = 0$) mit Gleichung 22 dar, bei welchem nicht in das Haar eingedrungen wird und somit lediglich die Fresnel-Terme eine Auswirkung haben.

$$A(0, h) = F(\eta^{'}, \eta^{''}, \gamma_i) \tag{22}$$

$$A(p, h) = (1 - F(\eta^{'}, \eta^{''}, \gamma_i))^2 F(\frac{1}{\eta^{'}}, \frac{1}{\eta^{''}}, \gamma_t)^{p-1} T(\sigma_a^{'}, h)^p \tag{23}$$

3.2.4 Longitudinale Streuungsfunktion M

Bei einem glatten Zylinder reflektieren Lichtstrahlen stets in den Bereich eines spiegelnden Kegels. Da die Oberfläche eines Haares allerdings, wie in Kapitel 3.1 beschrieben, eine Rauheit β und eine Neigung α besitzt, weicht das Verhalten beim Reflektieren und Refraktieren an der Haaroberfläche von dem Standard eines Zylinders ab. Diese Abweichungen werden in Marschner's Modell durch die longitudinalen Streuungsfunktion M modelliert, welche für jeden Lichtpfad separat berechnet wird. Konkret bedeutet dies, dass die Position der durch Reflexionen hervorgerufenen Highlights verschoben wird. In Abbildung 6 sind die Positionsverschiebungen durch Neigung α für jeden der Marschner Lichtpfade zu sehen, welche wie folgt aussehen:

R-Pfad, $p = 0$ Verschiebung um 2α in Richtung Haarwurzel

TT-Pfad, $p = 1$ Verschiebung um *weniger* als 2α in Richtung Haarspitze

TRT-Pfad, $p = 2$ Verschiebung um *mehr* als 2α weit in Richtung Haarspitze

Um diese Effekte darzustellen nutzten Marschner et al. eine normalisierte Gauß-Funktion $g(x, \mu, \sigma)$ mit Mittelwert μ und Standartabweichung σ. Die Gleichung für diese Normalverteilung ist in 24 abgebildet.

$$g(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{24}$$

Parameter	Purpose	Typical values
Fiber properties		
η	index of refraction	1.55
σ_a	absorption coefficient (R, G, B)	0.2 to ∞
a	eccentricity	0.85 to 1
Surface properties		
α_R	longitudinal shift: R lobe	-10° to -5°
α_{TT}	longitudinal shift: TT lobe	$-\alpha_R/2$
α_{TRT}	longitudinal shift: TRT lobe	$-3\alpha_R/2$
β_R	longitudinal width (stdev.): R lobe	5° to 10°
β_{TT}	longitudinal width (stdev.): TT lobe	$\beta_R/2$
β_{TRT}	longitudinal width (stdev.): TRT lobe	$2\beta_R$
Glints		
k_G	glint scale factor	0.5 to 5
w_c	azimuthal width of caustic	10° to 25°
$\Delta\eta'$	fade range for caustic merge	0.2 to 0.4
Δh_M	caustic intensity limit	0.5

Abbildung 10: Alle genutzten Parameter aus Marschner's Beleuchtungsmodell. Bildquelle: [MJC+03]

Approximiert wird die Verschiebung der Highlights, indem für jeden Pfad der Streuungsfunktion M unterschiedliche, den Eigenschaften des Pfades angepasste Werte für die Gaus-Funktion genutzt werden. Der R-Pfad ist in Gleichung 25 zu sehen, der TT-Pfad in Gleichung 26 und der TRT-Pfad in Gleichung 27. Außerdem sind in der Tabelle aus Abbildung 10 von Marschner et. al. vorgeschlagene Werte für alle Parameter einsehbar, die durch eine Gleichung des Modell's genutzt werden.

$$M_R(\theta_h) = g(\theta_h - \alpha_R, 0, \beta_R) \tag{25}$$

$$M_{TT}(\theta_h) = g(\theta_h - \alpha_{TT}, 0, \beta_{TT}) \tag{26}$$

$$M_{TRT}(\theta_h) = g(\theta_h - \alpha_{TRT}, 0, \beta_{TRT}) \tag{27}$$

3.2.5 Azimutale Streuungsfunktion N

Die azimutale Streuungsfunktion N wird ebenfalls separat für die einzelnen Lichtpfade berechnet und folgt aus der Gleichung 17. Sie simuliert die Lichtstreuung auf der Normalen-Ebene und ist somit abhängig vom Austrittswinkel ϕ. Außerdem werden statt des einzelnen h, die Werte aus allen Licht-Einfallsrichtungen, berechnet durch $h(p, r, \phi)$, betrachtet:

$$N_p(p, \phi) = \sum_r A(p, h(p, r, \phi)) \left| 2\frac{d\phi}{dh}(p, h(p, r, \phi)) \right|^{-1} \tag{28}$$

Die Summe erstreckt sich dabei über alle Nullstellen (r), was notwendig ist, da der TRT-Pfad mehrere Nullstellen besitzen kann.

Damit der *Scattering*-Wert für Gleichung 28 berechnet werden kann, muss zunächst der Austrittswinkel ϕ durch lösen der Gleichung 15 ermittelt werden. Marschner et. al. stellen fest, dass das Lösen dieser Gleichung, um h exakt zu berechnen, kostspielig ist, da unter anderem das Brechungsgesetz von Snell (Gleichung 4) genutzt werden muss. Um dies zu umgehen wird eine Annäherung für $\phi(p, h)$ präsentiert, welche einen maximalen Approximationsfehler von weniger als $0,75\,\mathrm{deg}$ für $\eta > 1,5$ besitzt. Dabei werden die Abhängigkeiten der Winkeln vereinfacht, indem γ_t mithilfe von γ_i und $c = \sin^{-1}(1/\eta)$ angenähert wird. Dies führt zu einem kubischen Polynom, Gleichung 29 mit Winkeln in Radianten angegeben, welches die gleichen Werte wie Snell's Gesetz bei $+ - 90\,\mathrm{deg}$ besitzt.

$$\gamma_t = \frac{3c}{\pi}\gamma_i - \frac{4c}{\pi^3}\gamma_i^3 \tag{29}$$

Mithilfe von Gleichung 29 ist die Approximation von ϕ eine kubische Funktion, welche abhängig vom Winkel γ_i ist. Sie wird notiert als $\hat{\phi}(p, \gamma_i)$ und ist in folgender Gleichung 30 dargestellt:

$$\hat{\phi}(p, \gamma_i) = \left(\frac{6pc}{\pi} - 2\right)\gamma_i - \frac{8pc}{\pi^3}\gamma_i^3 + p\pi \tag{30}$$

Die Nullstellen dieser Gleichung enthalten die Werte für γ_i, mit dessen Hilfe h durch Gleichung 13 gefunden wird. Für den R- und TT-Pfad besitzt die Gleichung eine einzelne Nullstelle und für den TRT-Pfad kann sie eine oder drei Nullstellen besitzen. Mithilfe von $\hat{\phi}(p, \gamma_i)$ kann der Ausdruck $\frac{d\hat{\phi}}{dh}$ aus Gleichung 28 gelöst werden, indem er abgeleitet wird. Dies wurde von Martin Ruenz in [Rue12]durchgeführt, was zu folgender Gleichung 31 führt:

$$\frac{d\hat{\phi}}{dh} = \frac{1}{\sqrt{1-h^2}}\left(-\frac{24pc}{\pi^3}\gamma_i^2 + \frac{6pc}{\pi} - 2\right) \tag{31}$$

Durch Gleichung 31 und Gleichung 23 bzw. 22, für den Lichtdämpfungsfaktor $A(p, h)$, kann die azimutale Streuungsfunktion N mit Gleichung 28 berechnet werden. Damit sind, zusammen mit Gleichung 25 ff. für die longitudinale Streuungsfunktion M, alle Komponenten zur Berechnung der finalen Streuungsfunktion S mit Gleichung 12 gegeben.

Die Berechnung von N_{TRT} des TRT-Pfades unterscheidet sich zusätzlich von den anderen Pfaden. Mithilfe der Untersuchungen an realen Haaren, haben Marschner et. al. festgestellt, dass die Kaustiken im TRT-Anteil eine Singularität in der Streuungsfunktion S mit unendlicher Intensität erzeugen. Aufgrund der Oberflächenrauheit des Haares ist dies allerdings unrealistisch, weshalb die Kaustiken aus dem TRT-Anteil des Modells ersetzt werden. An den Positionen der Kaustiken werden stattdessen *smooth lobes* dargestellt, deren Breite die Unschärfe der Kaustiken, hervorgerufen durch Oberflächenrauheit, simuliert. Die Prozedur der erklärten Approximation von N_{TRT} ist als Pseudocode in Figur 11 zu sehen, deren Parameter in der Tabelle aus Abbildung 10 einsehbar sind.

```
1  Funktion N_TRT(θ, φ; w_c, k_G, Δη', Δh_M)
2      if (η'(θ) < 2)
3          Berechne h_c mithilfe von h_c^2 = (4 − (η'(θ))^2)/3
4          Berechne φ_c mithilfe von h_c und Gleichung 15
5          Δh = min(Δh_M, 2√(2w_c/|d^2φ/dh^2 (h_c)|))
6          t = 1
7      else
8          φ_c = 0
9          Δh = Δh_M
10         t = smoothstep(2, 2 + Δη', η'(θ))
11     L = N_p(2, φ)
12     L = L · (1 − tg(w_c, 0, φ − φ_c)/g(w_c, 0, 0))
13     L = L · (1 − tg(w_c, 0, φ + φ_c)/g(w_c, 0, 0))
14     L = L + tk_G A(2, θ, φ)Δh(g(w_c, 0, φ − φ_c) + g(w_c, 0, φ + φ_c))
15     return L
```

Abbildung 11: Funktion um N_{TRT} zu berechnen. Approximiert die Kaustiken des TRT-Lichtpfades. Die Funktion smoothstep(a, b, x) ist 1 für $x < a$, 0 für $x > b$ und interpoliert stufenlos für Werte zwischen a und b. Quelle: [MJC+03]

3.3 d'Eon et al.: Ein energieerhaltendes Reflexionsmodell für Haare

Eugene d'Eon et. al. präsentieren in [dFH+11] ein Beleuchtungsmodell zum Simulieren der Lichtstreuung in Haaren und Fell, welches das Ziel hat energieerhaltend zu sein. Dabei baut d'Eon's Modell in der grundlegenden Struktur auf Marschne's Modell (Kapitel 3.2) auf. Das Modell ist ebenfalls für eine Zylinder-Geometrie mit rauer Oberfläche ausgelegt, wie es bei Haar-Fasern der Fall ist (siehe Kapitel 3.1). Außerdem ist das Vorgehen, verschiedenen Lichtpfade durch den Zylinder zu verfolgen und separat zu berechnen, einzusehen in Abbildung 9, gleich. Im Gegensatz zu Marschner's Modell versucht d'Eon nicht die Austrittswinkel der Pfade analytisch zu berechnen, wodurch keine kubische Funktion (Gleichung 30) berechnet werden muss und somit mehr als die 3 in Marschner's Modell betrachteten Lichtpfade in den finalen *Scattering*-Wert einfließen können. Dies ist ein Grund, warum das Modell energieerhaltend ist. Die dem Modell zu Grunde liegende Rendering-Gleichung ist ebenfalls Gleichung 10. Im folgenden werden die wichtigsten Gleichungen und Eigenschaften von d'Eon's Modell erläutert. Dabei wird sich an die in Kapitel 3.2 beschriebenen Notationen der Variablen gehalten.

3.3.1 Streuungsfunktion S

Die Streuungsfunktion S, bei der es sich um eine BSSRDF handelt, baut auf der Basis von Marschner (Gleichung 12) auf. Dies bedeutet, dass sie ebenfalls auf eine longitudinale Streuungsfunktion M (folgt in Kapitel 3.3.3) und eine azimutale Streuungsfunktion N (folgt in Kapitel 3.3.4) aufgeteilt wird. Diese einzelnen Streuungsfunktionen sind weiterhin aufgeteilt auf einzelne Lichtpfade p durch das Haar, wodurch sich die gesamte Streuungsfunktion S in folgender Gleichung 32

als Summe aus den Streuungsfunktionen der einzelnen Pfade S_p ergibt:

$$S(\theta_i, \theta_r, \phi) = \sum_{p=0}^{\infty} S_p(\theta_i, \theta_r, \phi) \tag{32}$$

Mit Gleichung 33 für den *Scattering*-Wert S_p eines einzelnen Lichtpfades p:

$$S_p(\theta_i, \theta_r, \phi) = M_p(\theta_i, \theta_r) N_p(\theta_i, \theta_r, \phi) \tag{33}$$

In diesem Modell kann die Anzahl der Lichtpfade, welche für die konkrete Kalkulation von S betrachtet werden, in der Theorie bis ins unendliche gehen. Wie aber in Kapitel 3.1 erläutert, wird die Energie und somit der Beitrag zum *Scattering*-Wert kleiner bis hin zu keinem Beitrag mehr, je mehr interne Reflexion der Pfad nimmt, also je größer p wird. Gleichzeitig wird die Performance schlechter, denn es müssen mehr Reflexionen für den Austrittswinkel θ_r berechnet werden. Daraus folgt, dass es in der praktischen Anwendung nicht sinnvoll ist zu viele Pfade für die Berechnung zu nutzen, denn ab einem gewissen Punkt ist das Verhältnis zwischen Mehrkosten an Rechenleistung für ein S_p und dessen Einfluss auf den finalen Wert von S nicht mehr Effizient genug.

3.3.2 Lichtdämpfung durch Reflexion und Absorption

Die Dämpfung innerhalb der Haar-Faser durch Absorption und fresnelsche Reflexion wird in diesem Modell ähnlich zu dem in Kapitel 3.2.3 vorgestellten Verfahren und dessen Gleichungen berechnet. Die Unterschiede sind die genutzten Parameter für die Fresnel-Terme und die Absorptionsfunktion T (Gleichung 21), welche dieselben Gleichungen in beiden Modellen nutzen. Da die in Marschner's Modell analytisch berechneten Winkel durch die Kreissektion γ_i und γ_t in diesem Modell nicht genutzt werden, werden die Parameter für die Fresnel-Terme aus der Einfalls- und Austrittsrichtung beziehungsweise den aus diesen resultierenden θ Winkeln berechnet. Außerdem ist die Nutzung des Bravais-Index η' nicht mehr nötig, da der echte Brechungsindex η des Haares genügt. Es ergeben sich somit folgende Gleichungen für den Lichtdämpfungsfaktor $A(p, h)$ eines einzelnen Pfades p durch das Haar:

$$A(0, h) = F(\eta, \frac{1}{2} \arccos(\omega_i \cdot \omega_r)) \tag{34}$$

$$A(p, h) = (1 - f)^2 f^{p-1} T(\mu_a^{'}, h)^p \tag{35}$$

Mit fresnelschem Reflexionsterm:

$$f = F(\eta, \arccos(\cos(\theta_d) \cos(\arcsin(h)))) \tag{36}$$

Der reduzierte Absorptionskoeffizient $\mu_a^{'}$ berechnet sich durch:

$$\mu_a^{'} = \frac{\mu_a}{\cos(\theta_t)} \tag{37}$$

und nutzt den neu vorgestellten Absorptionskoeffizient μ_a, welcher sich direkt aus der Konzentration der Haar-Farbpigmente Eumelanin ρ_e und Pheomelanin ρ_p mit folgender Gleichung 38 berechnen lässt:

$$\mu_a = \rho_e \begin{pmatrix} 0.419 \\ 0.697 \\ 1.37 \end{pmatrix} + \rho_p \begin{pmatrix} 0.187 \\ 0.4 \\ 1.05 \end{pmatrix} \tag{38}$$

Die Vektoren sind notwendig, um die entstehenden Farben, welche in der Realität sich in einem kontinuierlichen Spektral-band befinden, auf RGB zu *mappen*. Die konkreten Werte sind von d'Eon et al. empirisch ermittelt worden. Der Vorteil dieser Berechnung mithilfe der Melanin-Werte ist der, dass dadurch reale Konzentrationen der Pigmente genutzt werden können, welche dann in der 3D-Szene zu einer ähnlichen Haarfarbe wie die Haare aus der realen Welt führen. Dies macht d'Eon's Modell benutzerfreundlicher, als das Modell von Marschner, welches den Absorptionskoeffizient σ_a nutzt, der allerdings in keinem direkt vorstellbaren Verhältnis zu der finalen Haarfarbe steht.

3.3.3 Longitudinale Streuungsfunktion M

Um eine longitudinale Streuungsfunktion M herzuleiten, welche die Strahldichte gleichmäßig in dem spekularen Kegel, welcher abhängig von dem Einfallswinkel θ_i ist, verteilt und gleichzeitig die Oberflächenrauheit des Haares berücksichtigt, haben d'Eon et al. eine normalisierte sphärische Gaußfunktion untersucht. Nach Integration über den spekularen Kegel und einsetzen von Kugelkoordinaten (vergl. *Appendix A* in [dFH$^+$11]) ergibt sich für die Streuungsfunktion eines einzelnen Pfades M_p folgende Gleichung 39:

$$M_p(v, \theta_i, \theta_r) = \frac{csch(1/v)}{2v} e^{\frac{\sin(-\theta_i)\sin(\theta_r)}{v}} I_0 \left[\frac{\cos(-\theta_i)\cos(\theta_r)}{v} \right] \tag{39}$$

Die Varianz v der Oberflächenrauheit β führt zur Verschiebung der spekularen Highlights, wie es durch die geneigte und raue Oberfläche bei einem realen Haar der Fall ist und wird wie folgt berechnet:

$$v = \beta^2 \tag{40}$$

Bei $csch$ handelt es sich um die *Kosekans-hyperbolicus*-Funktion, welche eine Hyperbelfunktion (siehe [NEBES13]) ist, die berechnet wird durch:

$$csch(x) = \frac{2}{e^x - e^{-x}} = \frac{1}{sinhx} \tag{41}$$

Die letzte unbekannte Funktion aus Gleichung 39 ist die modifizierte Besselfunktion der erste Art und nullter Ordnung $I_0(x)$ (siehe [NEBES13]) mit folgender Gleichung:

$$I_0(x) = \sum_{k=0}^{\infty} \frac{(\frac{1}{4}x^2)^k}{(k!)^2} \tag{42}$$

3.3.4 Azimutale Streuungsfunktion N

Die azimutale Streuungsfunktion N simuliert, wie auch in Marschner's Modell, die Lichtstreuung in der Normalen-Ebene und ist somit abhängig von ϕ. Dabei decken die Einfallsrichtungen ω_i mit Offsets h, in Abbildung 9 für eine einzelne Einfallsrichtung zu sehen, über die gesamte Anzahl an Samples die gesamte Haar-Faser ab. In der Gleichung 43 drückt sich dies durch das Integral aus, welches sich über den gesamten Einheitskreis des Haar-Querschnittes spannt. Jeder einzelne Lichtstrahl mit Offset h führt zu einer Verteilung des azimutalen *Scattering*-Beitrags $D_p(\phi - \phi(p,h))$, welcher sich über den spekularen Kegel des spezifischen Pfades p erstreckt. Bei dieser Verteilung handelt es sich um eine Gauß-Verteilung in ϕ, welche Aufgrund der Oberflächenrauheit des Haares um $\phi(p,h)$ (siehe Gleichung 15) verschoben ist. Nach hinzufügen der Lichtdämpfung $A(p,h)$ aus Kapitel 3.3.2 ergibt sich folgende Gleichung 43 für die azimutale Streuungsfunktion N_p eines einzelnen Pfades p durch das Haar:

$$N_p(\phi) = \frac{1}{2}\int_{-1}^{1} A(p,h) D_p(\phi - \phi(p,h)) dh \tag{43}$$

Für $D_p(\phi)$ wird eine neue Verteilungsfunktion vorgestellt, welche *Gaussian detector* genannt wird. Da mit einer einfachen Gauß-Funktion der Winkel ϕ nicht auf den gesamten zuvor genannte spekulare Kegel eines Pfades verteilt werden kann, wird in der *Gaussian-detector*-Verteilungsfunktion durch Summieren einzelner normalisierter Gauß-Funktionen (Gleichung 24) sich an den gesamten Kegel, welcher von ϕ abgedeckt werden soll, angenähert. Die konkrete Gleichung ist in 44 dargestellt.

$$D_p(\phi) = \sum_{k=-\infty}^{\infty} g(\phi - 2\pi k, 0, \beta_p) \tag{44}$$

Nach empirischer Untersuchung ist festgestellt worden, dass für die Implementation dieser Arbeit ein *Gaussian detector* mit 20 Iterationen zufriedenstellend ist.

4 Konzeption

In diesem Kapitel werden die Vorgehensweisen und Methoden, welche in der Implementation dieser Arbeit genutzt werden, sowie deren Aufbau auf einer konzeptionellen Ebene beleuchtet und erklärt. Die Grundlage der Implementation ist das *Koblenz-Interactive-Raytracing-Framework* (KIRK, siehe [AD17]), aus welchem die nicht genutzten Klassen entfernt wurden. Dieses stellt einen funktionsfähigen Pathtracer inklusive BVH-Datenstruktur, sowie Klassen zum Verwalten und Laden einer 3D-Szene aus Modellierungssoftware bereit. Auf dieser Basis aufbauend wurden dann die benötigten Klassen und Verfahren zum Rendern von Fell hinzugefügt. Dabei handelt es sich unter anderem um *Shader* und *BSDF's*, welche die in Kapitel 3.2 und 3.3 vorgestellten Beleuchtungsmodelle von Marscher et al. und d'Eon et al. implementieren. Damit diese für Fell genutzt werden können, ist es zunächst notwendig, die Geometrie eines einzelnen Haares und folgend die Geometrie von Fell zu modellieren. In folgenden Kapiteln wird zunächst das Konzept der implementierten Geometrie eines Haares erklärt. Anschließend wird die grundlegende Rendering-Pipeline des Pathtracers beschrieben, worauf aufbauend dann der Shader inklusive BSDF's der beiden implementierten Beleuchtungsmodelle erläutert werden.

4.1 Geometrie eines Haares

Die Szenen-Geometrie ist bei KIRK in zwei verschiedenen Strukturen gespeichert. Zum einen gibt es einen Szenengraphen, welcher die gesamte Szene hierarchisch verwaltet. Dieser bietet die Möglichkeit aus verschiedenen Dateiquellen externe Szenen zu laden. Zum anderen gibt es die Schnittpunkt-Geometrie, welche vom Pathtracer zum Rendern verwendet werden. Diese enthalten, im Gegensatz zu den Objekten des Szenengraphen, einen Schnittpunkttest, um die Strahlen des Pathtracers auf einen solchen testen zu können. Die in dieser Implementation verwendete Datenstruktur BVH (siehe Kapitel 2.2.1) konstruiert aus den Objekten und der Hierarchie des Szenengraphen eine für das Pathtracing geeignete Hierarchie, welche statt den normalen Objekten die Schnittpunkt-Objekte enthält. Die genutzte Basis des KIRK verwendet sowohl für den Szenengraphen, als auch für die Pathtracing-Objekte, eine Geometrie bestehend aus Dreiecken, wobei im Szenengraphen mehrere zusammenhängende Dreiecke (*faces*) als ein *Mesh* repräsentiert werden, damit die Eckpunkte der Dreiecke nicht doppelt gespeichert werden müssen. Um Haare in KIRK zu integrieren muss deren Geometrie sowohl im Szenengraphen, als auch in Form von Schnittpunkt-Objekten, repräsentiert und abgespeichert werden können. Da es für beide Fälle unterschiedliche Anforderungsprofile gibt, die in den folgenden Kapiteln 4.1.1 und 4.1.2 genannt werden, werden diese auch separat voneinander betrachtet.

4.1.1 Haar im KIRK-Szenengraphen

Da es sich bei dem Szenengraphen um eine Ansammlung von Daten handelt, bei denen die einzelnen Objekte keine mathematischen Operationen oder Funktionen ausführen müssen, sieht die Anforderung für die Repräsentation einer Haar-Faser wie folgt aus: Das Objekt muss möglichst wenig Speicherplatz benötigen, aber gleichzeitig alle Informationen enthalten, die notwendig sind, um die wichtigsten Aspekte der Geometrie, mithilfe derer die gesamte Geometrie rekonstruiert werden kann, korrekt zu beschreiben. Betrachtet man die Geometrie einer realen Haar-Faser, zu sehen in Abbildung 5, liegt es nahe, das virtuelle Haar ebenfalls mithilfe einer Zylinder-Geometrie darzustellen. Um die oben genannte Anforderung zu erfüllen, betrachtet man zunächst den einfachsten Fall einen Zylinder darzustellen. Dies setzt einen geraden Zylinder mit gleichbleibendem Radius voraus, welcher mithilfe seiner Höhe h und seines Radius r beschrieben werden kann. Ein Haar hat allerdings nicht an jeder Stelle den gleichen Radius, denn es verjüngt sich, je nach Haar-Typ unterschiedlich stark, in Richtung Haar-Spitze (siehe [OA79]). Des weiteren ist ein Haar nicht gerade, sondern kurvig. Aus diesen zwei Gründen reicht die zuvor genannte Darstellung für einen Zylinder nicht für eine Haar-Faser aus. Nimmt man statt der Höhe die Position des Start- und Endpunktes und statt eines festen Radius einen eigenen Radius für die Start- und Endposition, kann mittels Interpolation jede Position inklusive Radius zwischen beiden Position berechnet werden (siehe [PJH16]). Daraus resultiert eine Darstellung für einen geraden Zylinder mit unterschiedlichen Radien, der im Extremfall (einer von beiden Radien 0) zu einem Kegel wird. Fügt man dazu weitere Positionen inklusive Radien hinzu, dann können auch kurvige Haare beliebiger Länge dargestellt werden, deren Auflösung von der Distanz der einzelnen Positionen zueinander abhängt. Daraus folgt als Darstellung für eine einzelne Haar- bzw. Fell-Faser (engl. *fur fiber*) im Szenengraphen folgende Struktur 12:
Es gibt zwei Listen gleicher Länge. Die erste Liste enthält die einzelnen Positio-

```
1   struct furFiber
2   {
3       List<3D−Vector> positions;
4   List<float> radius;
5   }
```

Abbildung 12: Struktur zur Darstellung einer einzelnen Haar-Faser im KIRK-Szenengraph.

nen als 3D-Vektoren, geordnet von der Wurzel der Faser bei Index 0, bis hin zur Spitze beim letzten Index. Die zweite Liste enthält den Radius als Gleitkommazahl zur entsprechenden Position mit gleichem Index. Da es sich um eine reine Ansammlung von Daten handelt, ist es ausreichend ein *struct* statt einer *Class* zu nutzen.

Mithilfe der Struktur eines einzelnen Haares kann das gesamte Fell dargestellt

werden. Dieses besteht aus vielen einzelnen Haaren, weshalb es innerhalb des zuvor genannten *Mesh* als Liste repräsentiert ist, welche die *furFiber-Structs* enthält. Daraus folgt, dass Haare nur in einem zugehörigen *Mesh* gespeichert werden können, welches als Haut oder Fläche interpretiert werden kann, aus dem die Haare wachsen. Um eine Beispielszene für Fell erstellen zu können, bei der nicht jede einzelne Position für alle gewünschten Haare manuell modelliert werden muss, gibt es eine Funktion, welche ein *Mesh* automatisch mit Haaren bestückt. Dabei werden Haare für Fell erstellt, welche, relativ zu menschlichen Haaren, kurz und stärker gekrümmt sind[5]. Die Vorgehensweise dieser Funktion ist in folgender Abbildung 13 als Pseudocode zu sehen:

```
1  Funktion fuegeFellZuMeshHinzu (int haare_pro_dreieck, int pos_pro_haar, float base_radius)
2      foreach Dreieck tri in Mesh this
3          foreach x in haare_pro_dreieck
4              furFiber ← new furFiber()
5              pos ← randDist3dPos(tri)
6              pos.y ← pos.y − 0.004
7              furFiber.positions ← add(pos)
8              radius ← base_radius
9              furFiber.radius ← add(radius)
10             foreach i in (pos_pro_haar − 1)
11                 newPos ← pos + Vector3D(0, log(i)/90, 0.06)
12                 radius ← radius − (radius /(i + 5))
13                 furFiber.positions ← add(newPos)
14                 furFiber.radius ← add(radius)
15                 pos ← newPos
16             end
17             this.furFibers ← add(furFiber)
18         end
19     end
20 end
```

Abbildung 13: Pseudocode der Funktion um Fell auf einem *Mesh* zu erstellen. Hier stellt *randDist3dPos()* eine Funktion dar, die eine zufällige, uniform verteilte Position auf einem Objekt als 3D-Vektor zurück gibt.

4.1.2 Haar als Schnittpunkt-Objekt

Damit die Informationen (Positionen und Radien) der einzelnen Haare zum Pathtracing genutzt werden können, müssen aus diesen Objekte erstellt werden, welche einen Schnittpunkttest implementieren. Um dies zu erreichen, gibt es zwei verschiedene Möglichkeiten. Zum einen das Konstruieren einer zylindrischen Form aus Dreieck-Schnittpunktobjekten und zum anderen das Nutzen von Zylinder -Schnittpunktobjekten. Beide Varianten erstellen dabei gerade Zylinder, weshalb nicht mit einem einzelnen Zylinder eine gesamte Haar-Faser dargestellt werden kann. Es wird deshalb, um eine Haar-Faser zu rendern, zwischen jeweils zwei

[5]Die Form, Dicke und Länge von Fell-Fasern variiert stark für verschiedene Tierarten (siehe Kapitel 3.1)

benachbarten Positionen einer Haar-Faser ein Zylinder erstellt. Für eine einzelne Haar-Faser mit n Positionen werden somit $n - 1$ Zylinder erstellt. Dabei ist darauf zu achten, dass der Abstand zwischen zwei Positionen, welche eine Neigung im Haar erzeugen, klein genug ist, um weiche Haare ohne Kanten im finalen Bild zu erzeugen. Sowohl der Schnittpunkttest für Zylinder, als auch das Konstruieren der Zylinder-Geometrie aus Dreiecken, ist von Kevin Keul geschrieben, aus dem CVK (siehe [Keu18]) entnommen und anschließend an die Haar-Struktur des Szenengraphen angepasst worden. Im Allgemeinen werden Dreiecke als Schnittpunktobjekte verwendet, da diese einen schnellen Schnittpunkttest mit Strahlen erlauben (siehe [PJH16]) und außerdem fast alle Geometrien darstellen können, wenn sie klein genug aufgelöst sind. Ein Zylinder-Schnittpunktobjekt braucht für die Schnittpunktkalkulation länger als ein Dreieck (siehe [PJH16]), hat aber für ein Haar den Vorteil, dass es sich um die richtige Form handelt und somit ein einzelnes Zylinder-Objekt den gleichen Teil eines Haares, wie mehrere Dreieck-Objekte darstellen kann. Die Anzahl der Dreiecke $n_{Dreiecke}$, die gebraucht wird, um einen einzelnen Zylinder zu simulieren, kann mit folgender Gleichung 45 berechnet werden:

$$n_{Dreiecke} = resolution^2 \cdot 2 \tag{45}$$

Die Implementation dieser Arbeit nutzt eine Auflösung (engl. *resolution*) von 8, woraus sich mit Gleichung 45 eine Anzahl von 128 Dreiecken für einen Zylinder ergibt. Weitere Vergleiche zwischen Dreieck- und Zylinder-Schnittpunktobjekten für ein Haar finden sich im Ergebniskapitel 6, welche sich auf den Speicherplatzverbrauch, die Rendering-Zeit, sowie die visuellen Unterschiede fokussieren.

4.1.3 Rendering von Haaren

Der Pathtracer im KIRK nutzt die in Kapitel 2.1.5 beschriebene Vorgehensweise, um ein Bild aus der 3D-Szene zu erstellen. Bevor das Integrieren der Beleuchtungsmodelle von Marschner und d'Eon in KIRK erklärt wird, wird zunächst der konzeptionelle Aufbau der Rendering-Pipeline im KIRK beschrieben, damit verstanden werden kann, an welcher Stelle die Modelle in diese eingefügt werden müssen.

4.1.4 Rendering-Pipeline des Pathtracers

Das Erzeugen des Bildes wird durch Aufrufen der *render()*-Methode des Pathtracers begonnen. Dies führt dazu, dass die Rendering-Pipeline ein einzelnes mal durchlaufen wird, folglich ein Sample berechnet wird. Durch das Aufrufen der *render()*-Methode innerhalb der Rendering-Schleife, wird sobald ein Sample fertig gerendert ist, das nächste Sample gerendert und das solange, bis die maximale Anzahl an Samples erreicht ist. Eine einzelne Iteration der Rendering-Pipeline, also die Farbberechnung für ein Sample, ist im Diagramm in Abbildung 14 dargestellt. Der Pathtracer nutzt *Buffer* um die wichtigen Objekte und Daten zu speichern. Diese müssen beim allerersten Aufruf der Rendering-Pipeline initialisiert werden.

Einer dieser Buffer ist der *Bounce-Buffer*, welcher die Pfade (*Bounces*, siehe Kapitel 2.1.5) speichert, die das Licht vom Primärstrahl, also von einem Pixel der Bildebene, beginnend durch die Szene nimmt. Dabei enthält jeder Bounce die aktuelle Farbe des Pfades, sowie die momentan anliegende Strahldichte (engl. *radiance*). Diese beiden Werte werden beim zurücksetzen des Buffers auf 0 (Farbe) und 1 (Strahldichte) gesetzt, dann für jeden neuen Schnittpunkt auf dem Pfad weiter angepasst und mithilfe der Komponente *Schnittpunkte schattieren* aus dem Diagramm 14 berechnet. Der Pathtracer ist normalisiert auf Werte zwischen 0 und 1, womit er für die Beleuchtung keine Lichtstärken auf physikalischer Basis nutzt. Die *Schnittpunkte schattieren* Komponente besteht aus zwei Teilen: Zum einen ein *Shader*, welcher für die Farbberechnung des Schnittpunktes zuständig ist und zum anderen einer *BSDF*, welche innerhalb des *Shaders* aufgerufen wird, um die Strahldichte und die neue zu verfolgende Richtung des Pfades zu berechnen. Diese beiden Komponenten sind es, die für die Beleuchtungsmodelle implementiert werden müssen. Da sowohl der zu nutzende Shader, als auch die zu nutzende BSDF im Material eines Objektes gespeichert sind, weiß der Pathtracer welchen Shader er aufrufen muss, nachdem ein Schnittpunkt durch Strahlenverfolgung gefunden wurde. Respektive weiß der Shader welche BSDF er aufrufen muss, da an ihn ein Schnittpunkt-Objekt (engl. *Intersection-Object*) übergeben wird, welches neben der Position des Schnittpunktes und des eingehend Strahls auch eine Referenz auf das getroffene Objekt inklusive Material enthält.

4.1.5 Shader und BSDF für Haare

Sowohl der Shader, als auch die BSDF's sind eigene Klassen. Da Marschner's und d'Eon's Modell im Kern ähnlich aufgebaut sind und beide die Rendering-Gleichung 10 für die Berechnung der Strahldichte nutzen, verwenden beide denselben Shader. Zusätzlich zu dem zuvor genannten *Intersection*-Objekt, wird dem Shader das *Bounce*-Objekt des aktuellen Pfades übergeben. Mithilfe dieser Information wird zunächst die Sampling-Funktion der BSDF aufgerufen, welche den Scattering-Wert S, die ausgehende Richtung des Lichtstrahls ω_r, sowie die PDF berechnet. Wie in Kapitel 3.2 beschrieben, gibt es bei Marschner's Modell 3 Lichtpfade, die zu berechnen sind. Damit für d'Eon's Modell derselbe Shader genutzt werden kann und ein besserer Vergleich zwischen beiden möglich ist, werden auch für d'Eon 3 Lichtpfade berechnet. Die 3 Pfade haben 3 Schnittpunkte und somit auch 3 ausgehende Richtungen und PDF's. Der Shader kann aber nur eine Richtung weiterverfolgen, weshalb aus den 3 von der BSDF berechneten Richtungen eine zufällig ausgewählt wird. Betrachtet man die Gesamtzahl an Samples, dann werden alle möglichen ausgehenden Richtungen gleichmäßig verteilt abgegangen. Die PDF's werden gemittelt und für die Gewichtung der Strahldichte genutzt, wie in Kapitel 2.1.5 beschrieben. Die Strahldichte wird daraufhin mit Rendering-Gleichung 10 berechnet und im Bounce-Objekt gespeichert. Als letztes wird die Farbe dieses Schnittpunktes berechnet und auf die Gesamtfarbe des Pfades (im Bounce-Objekt gespeichert) aufaddiert. Dazu wird die direkte Beleuchtung am Schnitt-

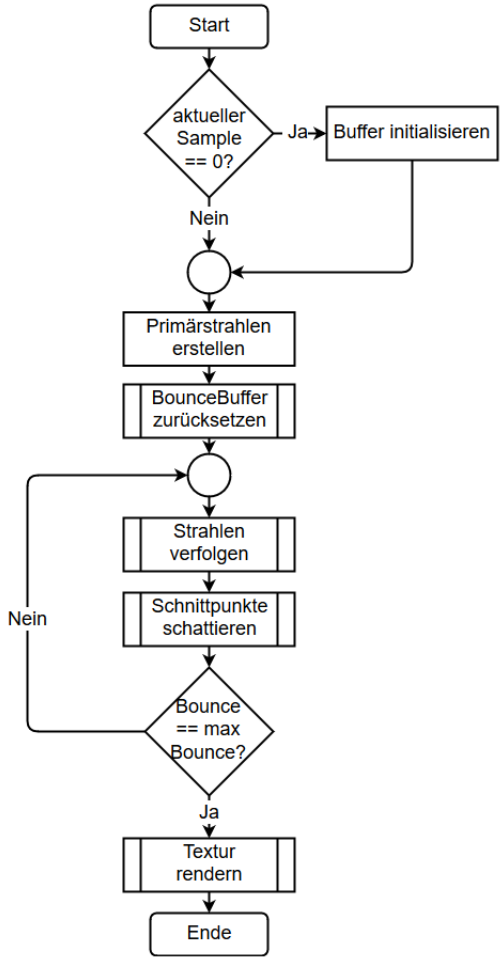


Abbildung 14: Einzelne Iteration der Rendering-Pipeline des KIRK Pathtracers. Berechnet die Farbe für ein Sample.

punkt berechnet, wie in Abbildung 9 als gelbe bzw. grau gestrichelte Pfeile zu sehen. Kalkuliert wird die direkte Beleuchtung ähnlich wie Phong's Gleichung 1 mit einer Glanzzahl von 5. Allerdings wird keine Materialfarbe mit eingerechnet, denn Phong's Gleichung wird genutzt, um die richtige Gewichtung für die Lichtfarbe im Hinblick auf den Betrachter und der Lichtposition zu errechnen. Außerdem wird eine zufällige Lichtquelle genutzt, statt der Summe über alle Lichtquellen. Wie auch bei den ausgehenden Strahlrichtungen, wird über alle Samples verteilt jede einzelne Lichtquelle gleichmäßig abgetastet. Die resultierende Lichtfarbe der direkten Beleuchtung wird mit der Strahldichte, also den Scattering-Werten aller bisherigen Schnittpunkte, multipliziert und ist somit der finale Farbwert des Schnittpunktes. Im Falle des TT- und TRT-Pfades enthält der Scattering-Wert die Farbinformationen des Haares. Die direkte Beleuchtung ist notwendig, da ohne diese nur dann Farbe dargestellt wird, wenn der Lichtpfad auf eine Lichtquelle trifft. Dies würde

zu vielen Pfaden führen, welche kalkuliert werden müssen, aber keinen Einfluss auf das finale Bild besitzen.

Bei den BSDF's gibt es jeweils eine eigene Klasse für das Marschner- und das d'Eon-Beleuchtungsmodell. Diese sind konzeptionell gleich aufgebaut, nutzten aber zur Berechnung ihrer Scattering- und PDF-Werte die in Kapitel 3.2 (Marschner) und 3.3 (d'Eon) vorgestellten Formeln. Da beide Modelle nicht für einen normalisierten Pathtracer mit Farbwerten zwischen 0 und 1 gemacht sind, müssen die Scattering-Werte noch mit einem Skalierungsfaktor an den Wertebereich des Pathtracers angepasst werden. Die ausgehende Lichtrichtung ω_r wird hingegen nicht, wie in Marschner's Modell beschrieben, mit dem analytisch berechneten Austrittswinkel γ_t kalkuliert. Stattdessen wird ausgenutzt, dass für das Haar eine Zylinder-Geometrie als Schnittpunktobjekt vorhanden ist. Dies erlaubt eine genauere Berechnung mithilfe des Reflexionsgesetzes 3 und des Brechungsgesetzes 4 im Gegensatz zur Annäherung des Austrittswinkel durch Marschner's Methode. Durch das konkrete Berechnen der Pfade, statt zufällig die Hemisphäre zu samplen (siehe Kapitel 2.1.5), wird außerdem nicht die gesamte Hemisphäre abgetastet, sondern die spekularen Kegel in denen das Licht nach 3 Lichtpfaden austreten kann. Zu sehen sind diese Kegel als graue Hinterlegung bei den austretenden Strahlen in Abbildung 9. Um zusätzliche eine größere Reichweite des Samplings in den Kegeln zu erlangen und gleichzeitig die in Kapitel 3.1 beschriebene Neigung der Haaroberfläche einzubeziehen, werden die austretenden Strahlen mit einem Winkel α um die u-Achse des Haares rotiert. Dabei hat α einen zufälligen, aber für ein einzelnes Haar konstanten Wert zwischen -5 und -10 Grad. Dieser α-Wert wird für die verschiedenen Pfade durch das Haar abgewandelt, was in der Tabelle der Abbildung 10 zu sehen ist.

5 Implementation

In diesem Kapitel werden Schwierigkeiten und deren Lösungen bei der spezifischen Implementation dieser Arbeit vorgestellt. Dies betrifft den Shader und die zugehörigen BSDF's von Marschner's und d'Eon's Beleuchtungsmodell.

5.1 Schwierigkeiten im Haar-Shader

Das größte Problem für die Implementation des Haar-Shader stellt die vorhandene Rendering-Pipeline des KIRK dar. Denn die *shade()*-Methode des Shaders aus dem Modul *Schnittpunkte schattieren* aus Abbildung 14 ist darauf ausgelegt, dass auf einen einzelnen konkreten Schnittpunkt ein einziger einfallender Strahl trifft und daraufhin ein einzelner austretender Strahl zurückgegeben wird. Dieser ausgehende Strahl wird dann von einer anderen Komponente der Rendering-Pipeline weiter verfolgt, bis bei dem nächsten Schnittpunkt wieder die *shade()*-Methode des im dortigen Material gespeicherten Shaders aufgerufen wird. Das Problem ist nun, dass die *shade()*-Methode beim zweiten Schnittpunkt nichts von den intern berechneten Werten des ersten shade()-Aufrufes weiß, selbst wenn sie von demselben Shader-Objekt aufgerufen wurde. Für ein Haar bedeutet dies, dass bei der Berechnung des Scattering-Wertes S_1 für den Zylinder-Pfad TT (siehe Abbildung 9 p=1) der Wert von S_0 des ersten Pfades wieder vergessen ist. Es geht die Zusammengehörigkeit der 3 Pfade des Haares verloren, welche aber zum Berechnen des gesamten Scattering-Wertes S mithilfe der Gleichung 12 benötigt wird, da dort die Werte der drei einzelnen Pfade addiert werden.

Folglich müssen die 3 Scattering-Werte S_0, S_1 und S_2 in einem einzelnen *shade()*-Aufruf des Haar-Shader berechnet werden. Dazu müssen zunächst die zwei fehlenden Schnittpunkte für den Austrittspunkt aus dem Haar des TT- und TRT-Pfades berechnet werden. Durch das an die *shade()*-Methode übergebene *Intersection*-Objekt ist das Geometrie-Objekt des Haares bekannt, mit welchem die einzelnen Strahlen der Pfade aus Abbildung 9 mithilfe von Reflexion und Refraktion berechnet werden können. Diese Strahlen werden dann für die Schnittpunkt-Berechnung des TT- und TRT-Pfades genutzt. Anschließend wird die *sample()*-Methode der Marschner-Haar-BSDF oder der d'Eon-Haar-BSDF, je nachdem welche im Material gespeichert ist, aufgerufen. Dies geschieht 3 mal, jeweils für den R-, TT- und TRT-Pfad, mit ihren entsprechenden Schnittpunkten und einer *material flag*, damit die BSDF weiß, für welchen Pfad der Scattering- und PDF-Wert, sowie die ausgehende Strahlrichtung, berechnet werden müssen. Das restliche Vorgehen verläuft wie in Kapitel 4.1.5 beschrieben.

Des weiteren sind die Modelle von Marschner und d'Eon nicht auf den Wertebereich zwischen 0 und 1 normalisiert, in welchem der Pathtracer agiert. Trotz des in Kapitel 4.1.5 angesprochenen Skalierungsfaktor kann es in Ausnahmefällen vorkommen, dass der finale Scattering-Wert, also die Summe der einzelnen Pfad-Anteile, den Wert 1 in verschiedenen Farbkanälen überschreitet. Wenn dies der Fall ist, wird der Vektor auf Länge 1 normalisiert, was dazu führt, dass Hellig-

keit verloren geht, aber der Farbwert bleibt erhalten. Über die gesamte Anzahl an Samplen gleicht sich der so entstandene Helligkeitsverlust wieder aus. Eine weitere Möglichkeit das genannte Problem zu lösen, wäre ein *clampen* der Werte auf 0 und 1, dies würde allerdings unter Umständen dazu führen, dass nur ein einzelner Farbkanal geclampt wird und somit der Farbwert ungewollt verändert wird.

6 Ergebnisse

In diesem Kapitel werden die Ergebnisse der Implementation dieser Arbeit präsentiert. Dazu werden die in der Einleitung 1 gestellten Fragen aufgegriffen und beantwortet. Insbesondere meint dies ein präsentieren der Ergebnisbilder der Beleuchtungsmodelle von Marschner und d'Eon. Diese werden zunächst einzeln visuell analysiert, wobei auf den Beitrag der einzelnen Pfad-Segmente zum Gesamtbild eingegangen wird und die Highlights mit den theoretisch vorkommenden, in Kapitel 3.2 und 3.3 beschrieben, verglichen werden. Daraufhin werden beide Beleuchtungsmodelle gegeneinander verglichen in Hinsicht auf das Visuelle und die Performance im Renderer. Um dies machen zu können, müssen zunächst die beiden Optionen für die Haar-Geometrie, die Zylinder- und Dreieck-Schnittpunktobjekte, gegeneinander verglichen werden, um zu entscheiden mit welcher Geometrie-Art die restlichen Ergebnisbilder gerendert werden sollen. Auch die Geometrie-Arten werden im Hinblick auf das Visuelle und die Performance (benötigte Zeit zum Rendern und Speicherbedarf) verglichen. Der Fokus liegt, wie auch bei den Beleuchtungsmodellen, auf dem visuellen Aussehen beider. Nachdem sich für eine Geometrie-Art entschieden wurde, wird die Performance des Pathtracers im Hinblick auf die maximal benötigte Sample-Anzahl untersucht, damit die restlichen Ergebnisbilder mit einer zufriedenstellenden, gleichbleibenden Sample-Anzahl gerendert werden können, welche ein gutes Ergebnis im Vergleich zu einem Bild mit sehr hoher Sample-Anzahl haben, aber gleichzeitig nicht zu viel Renderzeit benötigen.

Alle Ergebnisbilder sind mit dem implementierten Pathtracer, welcher alle Berechnungen auf der CPU durchführt und einer eigens erstellten Testszene mit einer *Plane*, welche ein Hautabschnitt simulieren soll, die mit der in Abbildung 13 beschriebenen Methode mit Fell bestückt worden ist. Die Auflösung aller Bilder beträgt 1280x720 Pixel und der Pathtracer verfolgt Pfade bis zu einer maximalen Tiefe von 5 *Bounces*. Zum rendern ist folgendes System genutzt worden:

OS Windows 10 (64-Bit)

CPU Intel i7-8700k mit max 4,5 GHz Turboboost

RAM 32GB DDR4

GPU Geforce GTX 1080TI

6.1 Vergleich Schnittpunkt-Objekte

Damit entschieden werden kann, ob Zylinder- oder Dreieck-Schnittpunktobjekte für die Ergebnisbilder genutzt werden sollen, müssen diese zunächst gegeneinander verglichen werden. Das Ziel eines Schnittpunktobjektes ist es, einen schnellen Schnittpunkttest zu liefern und gleichzeitig möglichst geringen Speicherplatzbedarf vorzuweisen. Das Wichtigste ist allerdings, dass die Schnittpunkttests einen

korrekten Schnittpunkt für variable Input-Werte (Strahlen aus verschiedenen Richtungen) liefern und somit die Objekte durch den Pathtracer mit der richtigen Geometrie angezeigt werden.

Zunächst wird der Speicherplatzbedarf gegeneinander verglichen. Dazu wird die tatsächlich benötigte Anzahl an Bytes für das Speichern eines einzelnen Objekts berechnet. Dabei werden Attribute, welche nicht direkt für die Schnittpunktberechnung des Objekts benötigt werden, wie einen Zeiger auf das Material oder der Bounding-Box des Objekts für die Datenstruktur, nicht für die Berechnung berücksichtigt. Da für das Zylinder-Objekt keine Textur-Koordinaten unterstützt werden, fließen diese auch nicht in die Berechnung ein. In folgender Tabelle sind die verwendeten C++ Typen mit benötigtem Speicherplatz zu sehen (siehe [SS14]):

Typ	**Bytes**
float	4
vec3<float>	12

Bei *vec3<float>* handelt es sich um einen Vektor mit 3 *float*-Werten, also $3 \cdot 4$ Byte $= 12$ Byte. Mit diesen Werten kann der Speicherplatzbedarf zwischen einem einzelnen Dreieck und einem einzelnen Zylinder verglichen werden. In folgender Tabelle sind die benötigten Attribute beider Objekte zu sehen, aus deren Anzahl dann der gesamte Bedarf in Bytes berechnet wird:

Objekt	**Typ**	**Anzahl**	**Bytes**	**Attribute**
Dreieck	float	1	$1 \cdot 4 = 4$	1xEpsilon-Konstante
	vec3<float>	10	$10 \cdot 12 = 120$	3xPosition,3xEckNormale, 3xKanten,1xNormale
	Gesamt		124	
Zylinder	float	7	$7 \cdot 4 = 28$	2xRadius,1xHöhe, 1xSchräge, 3xDurchmesser
	vec3<float>	2	$2 \cdot 12 = 24$	2xPosition
	Gesamt		52	

Damit ergibt sich ein Speicherplatzbedarf von 124 Byte für ein einzelnes Dreieck und 52 Byte für einen einzelnen Zylinder. Als nächstes wird die gesamte Testszene betrachtet, wobei nur die Schnittpunkt-Objekte für Haare und nicht die der Bodenplatte gezählt werden. In der Testszene gibt es 1480 Haare[6], wobei jedes Haar aus $n = 10$ Positionen besteht. Wie in Kapitel 4.1.2 erläutert, werden aus einem Haar $n - 1$ Zylinder erstellt, während für einen Zylinder 128 Dreiecke erstellt werden müssen, um dieselbe Geometrie in der Szene abzudecken. Dies ergibt für die Schnittpunktobjekte pro Haar $10 - 1 = 9$ Zylinder und $128 \cdot 9 = 1152$ Dreiecke. Zusammen mit der Anzahl der Haare und dem Speicherplatz pro Objekt folgt der insgesamt benötigte Speicherplatz für Haar-Schnittpunktobjekte in der Testszene:

[6]Haare sind als furFiber-Struct im Szenengraphen gespeichert. Anzahl ist dementsprechend die Länge der Liste, die diese speichert.

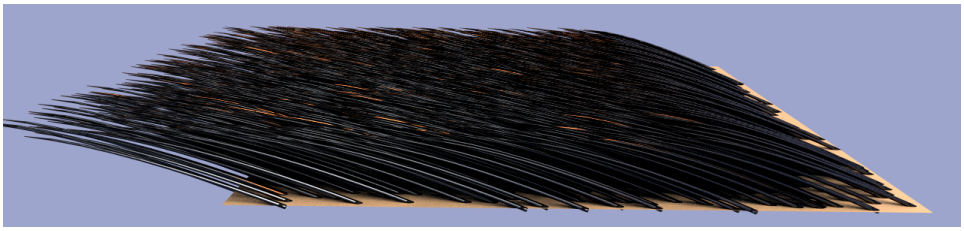
	Speicherplatzbedarf
Zylinder	1480 [Haare] * 9 [Zylinder pro Haar] * 52 Byte = 692 640 Bytes = 0,69264 MByte
Dreiecke	1480 [Haare] * 1152 [Dreiecke pro Haar] * 124 Byte = 211 415 040 Bytes = 211,41504 MByte

Folgend werden die Renderzeiten beider Schnittpunktobjekt-Typen verglichen. Dazu sind die Renderzeit pro Sample, sowie die gesamte Renderzeit für 250 Samples der Bilder in Abbildung 15 mithilfe der *chrono-CPU-Clock* aus der C++ Standardbibliothek gemessen worden:

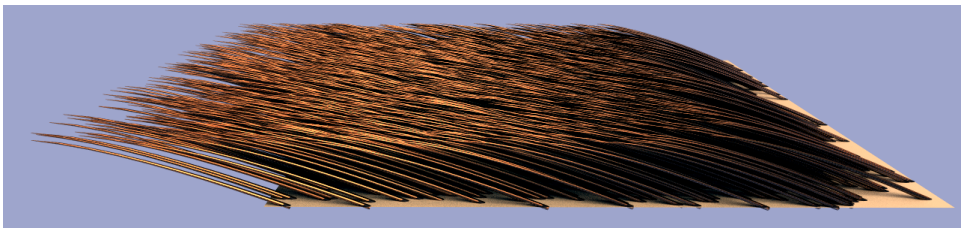
	Zeit 1 Sample	**Zeit 250 Samples**
Zylinder	870 ms	220 026 ms = 3:40 min
Dreiecke	1210 ms	301 860 ms = 5:02 min

Die Ergebnisse der Renderzeit-Messung zeigt, dass die Szene schneller mit Zylinder-Schnittpunktobjekten rendert. Der Grund dafür liegt in der Anzahl an Objekten, welche pro Strahl auf einen Schnittpunkt getestet werden müssen. Die einzelnen Haare liegen im Fall von Fell sehr eng beieinander und wie bereits erwähnt, sind für ein Haar 128 mal mehr Dreiecke als Zylinder vorhanden. Dies bedeutet, dass um den Schnittpunkt eines Strahls mit der Szene zu finden, deutlich mehr Objekte auf einen Schnittpunkt getestet werden müssen, wenn Dreiecke als Objekt-Typen verwendet werden. Aufgrund des hierarchischen Aufbaus der BVH-Datenstruktur erhöht sich der Rechenaufwand nicht linear, aber die BVH kann die einzelnen *Bounding-Volumes* nicht klein genug Auflösen, damit pro Strahl nur noch wenige Dreiecke auf einen Schnittpunkt getestet werden müssten. Die resultierende erhöhte Anzahl an durchzuführenden Schnittpunkttests für einen Strahl mit der Szene spiegelt sich in der erhöhten Renderzeit pro Sample und damit auch in einer erhöhten gesamten Renderzeit wieder.

Trotz des weit niedrigeren Speicherplatzbedarfes und der geringeren Renderzeit der Zylinder-Objekte, werden für die restlichen Ergebnisbilder in dieser Arbeit die Dreieck-Schnittpunktobjekte verwendet. Der Grund ist der in Abbildung 15 zu sehende visuelle Unterschied zwischen beiden Varianten. Während bei den Dreieck-Objekten die Farbe des Haares, welche durch den TRT-Pfade von Marschner's Modell zustande kommt, deutlich sichtbar ist, kann man sie bei den Zylinder-Objekten nur an einzelnen Haaren erkennen. Der Grund dafür liegt in den Oberflächennormalen, welche von dem Zylinder-Objekt berechnet werden. Da ein Haar einen geringen Durchmesser hat (siehe Kapitel 3.1), wird das Zylinder-Objekt mit einem Radius von 0.005 erstellt, was aber dazu führt, dass das Objekt zu klein aufgelöst ist um die Normalen korrekt zu berechnen. Dies ist in Abbildung 16 zu erkennen, denn die roten Normalen des Zylinder-Schnittpunktobjektes auf dem linken Bild stehen zum einen in einem rechten Winkel zueinander und variieren zum anderen in der Horizontalen, trotz konstanter eingehender Strahlen in grün. Die Dreiecks-Schnittpunktobjekte auf der rechten Seite hingegen sind gleichmäßig verteilt und bilden keinen rechten Winkel zueinander. Wie in den Bildern 15 zu



(a) Fell mit Zylinder-Schnittpunktobjekten



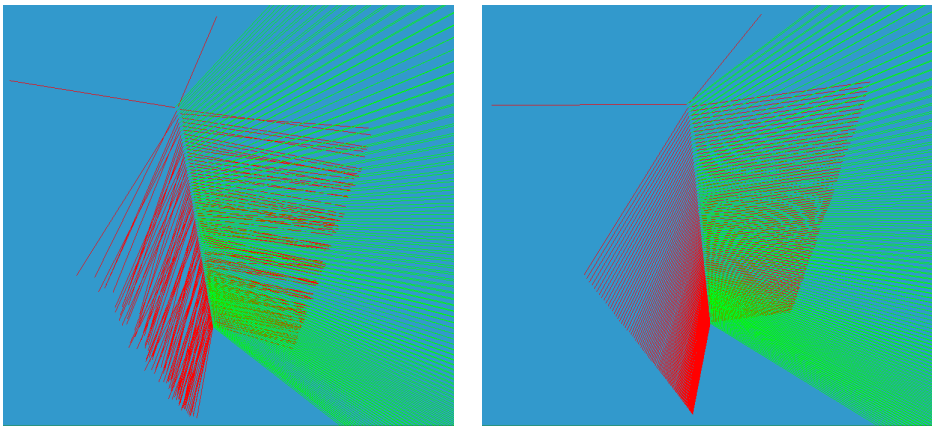
(b) Fell mit Dreieck-Schnittpunktobjekten

Abbildung 15: Vergleich zwischen Zylinder- (a) und Dreieck-Schnittpunktgeometrie (b) für Fell bei Marschner's Beleuchtungsmodell.
Beide Bilder mit 250 Samplen bei einer Auflösung von 1280x720 Pixeln gerendert.

sehen, wirkt sich dies deutlich auf das visuelle Ergebnis aus. Die Normalen werden gebraucht um sowohl die ausgehende Richtung des Strahls und somit auch die Winkel θ und ϕ, als auch die direkte Beleuchtung zu berechnen.

6.2 Sample-Anzahl des Pathtracers

Die Gesamtzahl an Samplen, die für das Rendern eines Bildes mithilfe des Pathtracers genutzt wird, entscheidet über die Qualität des Bildes. Je mehr Sample genutzt werden, desto weniger verrauscht ist das Bild, aber die Renderzeit erhöht sich mit jedem zusätzlichen Sample. Für den Vergleich verschiedener Kamerawinkel ist es nicht praktikabel mehrere Stunden für eine Einstellung zu rendern, wenn die wichtigen Eigenschaften schon mit einer geringeren Renderzeit sichtbar sind. Um einen geeigneten Kompromiss zwischen Qualität und Renderzeit zu finden, werden im folgenden Bilder der Testszene mit Marschner's Beleuchtungsmodell, gleichbleibendem Kamerawinkel und unterschiedlicher Sample-Anzahl verglichen. In Abbildung 17 ist ein Ergebnisbild mit 250 Samplen und einer Renderzeit von 5:12 Minuten zu sehen, sowie ein Ergebnisbild mit 10 000 Samplen, welches als fertiges Bild angesehen werden kann und eine Renderzeit von 3:22:34 Stunden benötigt hat. Die Unterschiede in beiden Bildern sind nicht sehr deutlich zu erkennen, denn es handelt sich vor allem um Rauschen, welches weniger wird mit erhöhter Anzahl der Samples. Betrachtet man den weichen Schatten der Haare auf der Bodenplatte, ist bei 250 Samplen ein Rauschen zu erkennen, während bei 10 000 Samplen kein Rauschen mehr auszumachen ist.



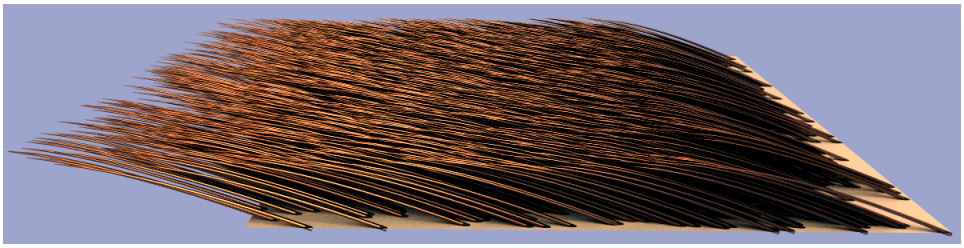
(a) Zylinder-Schnittpunktobjekt **(b)** Dreieck-Schnittpunktobjekte

Abbildung 16: Berechnete Normalen (rot) eines Zylinders mit Radius 0.005 nach Auftreffen der eingehenden Strahlen (grün). In a) ist ein Zylinder-Schnittpunktobjekt und in b) Dreiecks-Schnittpunktobjekte genutzt worden.

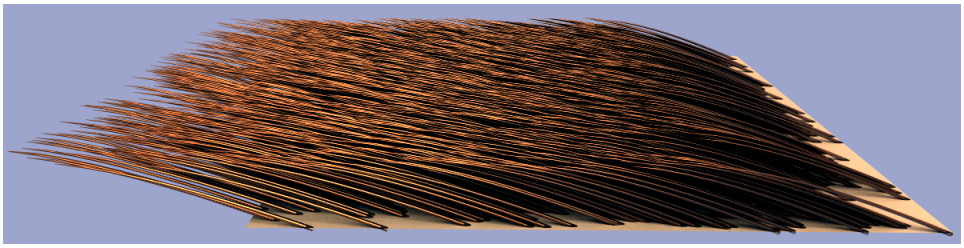
Um die verschiedenen Sample-Anzahlen aussagekräftig gegeneinander zu vergleichen, reicht es nicht das rein Visuelle mit dem menschlichen Auge zu betrachten. Die Farbwerte der Bilder müssen dazu statistisch gegeneinander verglichen werden. Dazu wird die Differenz eines Pixel zwischen zwei Bildern I mit gleicher Größe mit folgender Formel berechnet:

$$I_{diff}(x, y) = \sqrt{(|I_1(x, y) - I_2(x, y)|)^2} \tag{46}$$

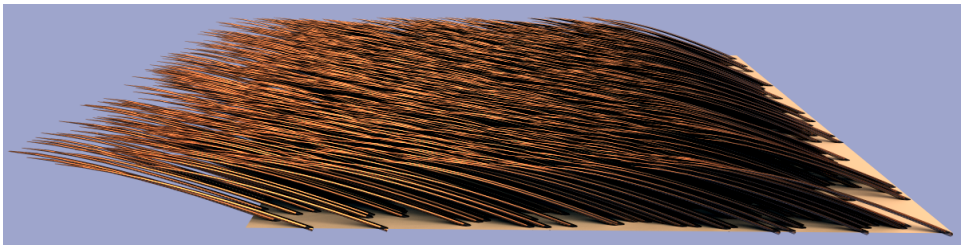
Berechnet man diesen maximalen Fehler für jeden Pixel im Bild, können die Unterschiede zwischen beiden Eingabebildern als *Heatmap* dargestellt werden. Dazu werden die Differenzen der Pixel als Farbwerte dargestellt, in der Form, dass blau keine Differenz bedeutet und je stärker es ins rötliche geht, desto größer ist die Differenz zwischen den Pixeln. In Abbildung 18 sind die Differenzen der Pixelwerte als Heatmaps zwischen einer jeweils größer werdenden Sample-Anzahl und dem finalen Bild mit 10 000 Samplen zu sehen. Als Vergleichswert kann in diesen Bildern der blaue Hintergrund betrachtet werden, was bedeutet, dass dort keine Differenz in den Pixel-Werten vorhanden ist. In Bild a) (100 Samples) der Abbildung 18 ist eine relativ große Differenz an den türkis- bis grün-farbigen Stellen auszumachen, während bereits in Bild b) (250 Samples) die türkis-farbigen Stellen nur noch als Ausnahme vorkommen. Ab etwa 2000 Samplen verändert sich das Bild fast nicht mehr, was an der nahezu komplett blauen Heatmap d) zu sehen ist. Da für 2000 Samples die Renderzeit mit 41:35 Minuten aber zu hoch ist, um viele Ergebnisbilder zu rendern, werden für die restlichen Bilder dieser Arbeit 500 Samples verwendet. Diese Anzahl weist fast keine Extremwerte mehr in der Heatmap c) auf und hat mit einer Renderzeit von 10:25 Minuten einen akzeptablen Zeitaufwand.



(a) Marschner 250 Samples mit 0:05:12 h Renderzeit



(b) Marschner 500 Samples mit 0:10:25 h Renderzeit

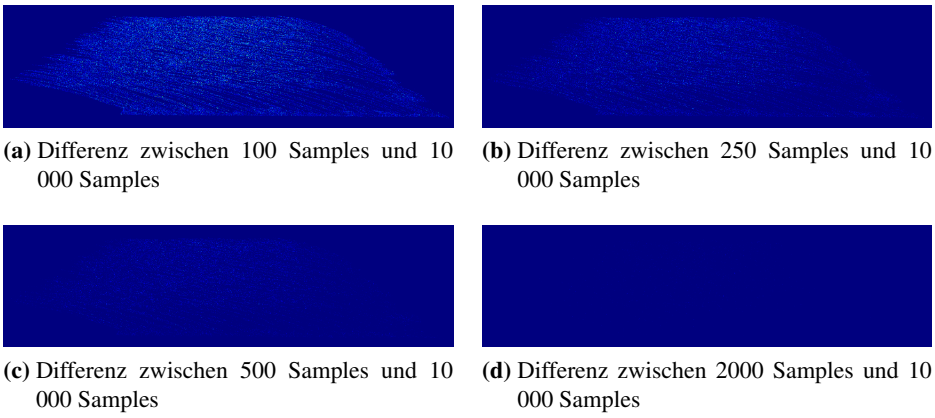


(c) Marschner 10000 Samples mit 3:22:34 h Renderzeit

Abbildung 17: Vergleich Ergebnisbilder der Fell-Testszene mit verschiedener Sample-Anzahl und Marschner's Beleuchtungsmodell.

6.3 Haare mit Marschner et al.

Verschiedene Ergebnisbilder der Fell-Testszene, gerendert mit Marschner's Beleuchtungsmodell und 500 Samples, sind in Abbildung 22 und 23 im Anhang zu sehen. Die durch die Lichtabsorption des TRT-Pfades entstandene Farbe ist gut erkennbar und entsteht überall dort, wo die Strahlen aus der Kamera einen Weg zur Lichtquelle finde können. Abbildung 23 c) ist ein Beispiel dafür, dass aus bestimmten Winkeln die Strahlen nicht zur Lichtquelle reflektiert werden können und somit das Haar schwarz bleibt. Aus der Ferne betrachtet sind die auftretenden Highlights der Haare plausibel, nach den theoretischen Vorkommnissen, die in Kapitel 3.2.2 erklärt wurden. Der Grund für die schwarze Färbung der Haare am Übergang zur Bodenplatte ist das Verschieben der Haare in die Bodenplatte, welches gemacht wurde, da die Zylinder der Haare geneigt sind und ansonsten aus der Bodenplatte herausragen würden. Betrachtet man die Haare aus der Nähe, wie in Bild 23 a), dann fallen zum einen die primären Highlights der Reflexion auf, zum anderen aber



(a) Differenz zwischen 100 Samples und 10 000 Samples

(b) Differenz zwischen 250 Samples und 10 000 Samples

(c) Differenz zwischen 500 Samples und 10 000 Samples

(d) Differenz zwischen 2000 Samples und 10 000 Samples

Abbildung 18: Differenzen der Pixelwerte zwischen größer werdender Sample-Anzahl und finalem Bild mit 10 000 Samplen als *Heatmaps*.

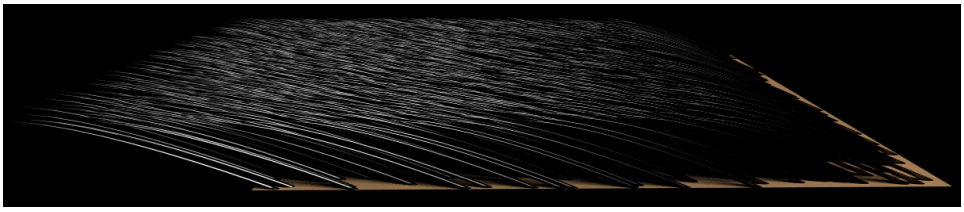
auch die Artefakte, welche durch die einzelnen Dreiecke entstehen, wenn diese von der Lichtquelle abgewandt sind.

Um die Effekte der einzelnen Licht-Pfade durch das Haar[7] ausmachen zu können, sind in Abbildung 19 die Beiträge der Pfade jeweils einzeln dargestellt. In Bild a) ist das primäre Highlight des R-Pfades, hervorgerufen durch Reflexion, zu sehen. Aufgrund der Licht-Position (hinter und über der Kamera) und der direkten Beleuchtung ist dieses primäre Highlight stark ausgeprägt, aber wie im farblich visualisierten Bild c)[8] zu sehen, fokussiert sich der Hauptteil des Highlights auf die Mitte des Haares. Der TRT-Pfad in Bild b) ist das sekundäre Highlight, welches etwas verschoben vom primären Highlight auftritt und sich wie in Kapitel 3.2.2 beschrieben verhält. Die Farbe des TRT-Pfades stimmt nicht mit der aus dem fertigen Bild überein, da sie ohne den Anteil des R- und TT-Pfades nicht mehr kräftig genug ist und somit künstlich verstärkt werden musste um nicht schwarz zu sein.

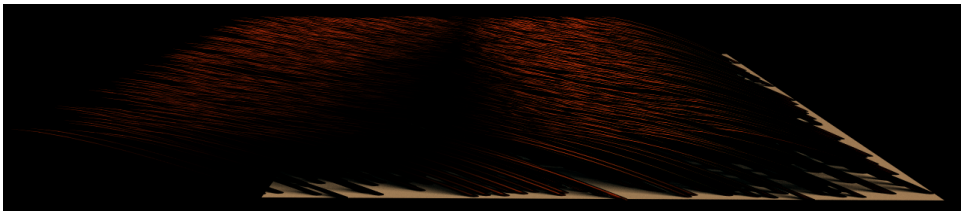
Vergleicht man die Pfade dieser Implementation mit den Bildern des originalen Marschner Paper in Abbildung 25 des Anhangs, fällt zum einen auf, dass das primäre Highlight dieser Implementation eine größere Fläche abdeckt als das von Marschner. Die relativ zentrale Position im Haar ist dennoch gleich. Zum anderen fällt auf, dass das sekundäre Highlight stärker verschoben ist als bei Marschner, diese Verschiebung aber der Theorie entspricht. Des weiteren ist der zweite Ausbreitungspunkt des TRT-Pfad größer als bei Marschner. Der Grund für beide Abweichungen ist zum einen eine andere Kamera und Lichtposition und zum anderen eine unterschiedliche Geometrie der Haare, denn Marschner nutzt lange, menschliche Haare, während diese Implementation kurz-haariges Tierfell simuliert. Im Anhang in Abbildung 26 ist zum Vergleich das Fell eines echten Rehs zu sehen.

[7] R- und TRT-Pfad. Der TT-Pfad wird in diesem Vergleich nicht berücksichtigt, da dieser transmissive Pfad nur Farbe liefert, wenn sich die Kamera hinter den Haaren befindet.

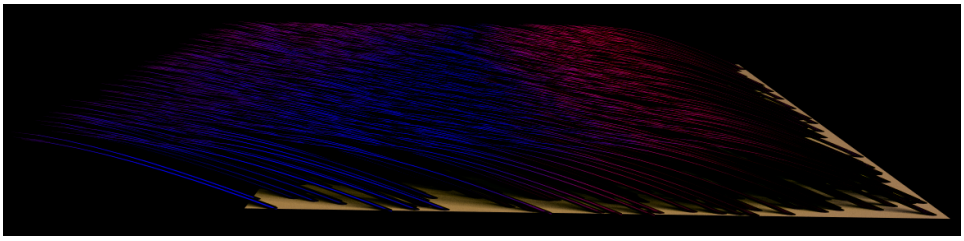
[8] Anteile des R-Pfades im B-Kanal, Anteile des TT-Pfades im G-Kanal und Anteile des TRT-Pfades im R-Kanal des RGB-Bildes.



(a) Marschner's R-Pfad



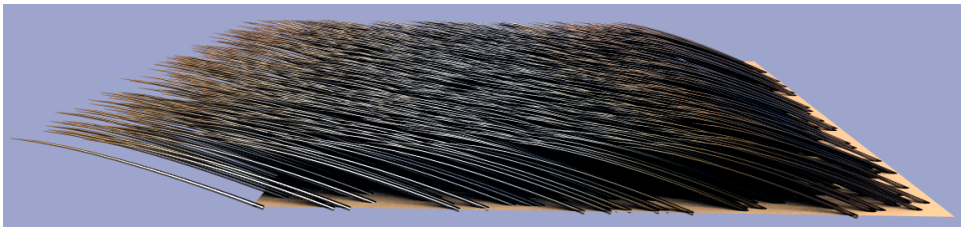
(b) Marschner's TRT-Pfad



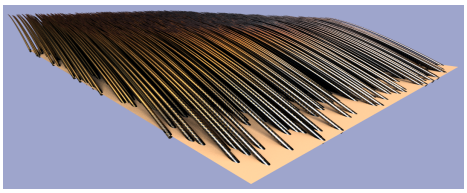
(c) Marschner's R-Pfad in blau, TT-Pfad in grün und TRT-Pfad in rot

Abbildung 19: Die Lichtpfade durch das Haar aus Marscher's Modell jeweils einzeln betrachtet. Außerdem in c) die Anteile der einzelnen Pfade jeweils in einem eigenen Farbkanal dargestellt.

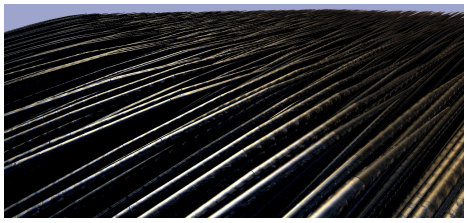
Dieses weißt eine ähnliche Struktur, wie das gerenderte Fell auf, wenn man die verschiedenfarbigen Haare eines Rehs und unterschiedlichen Ausrichtungen der Haare außen vor lässt.



(a) Renderzeit 8:07 Minuten



(b) Renderzeit 13:16 Minuten



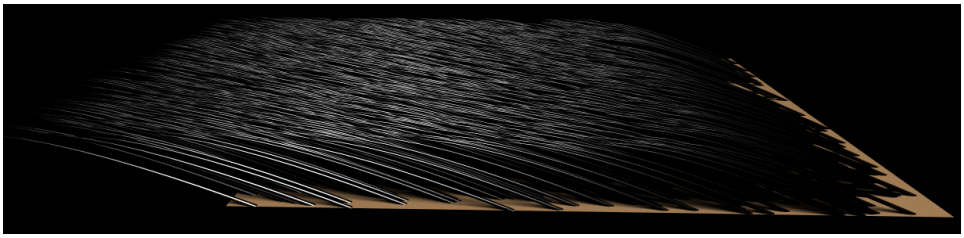
(c) Renderzeit 36:37 Minuten

Abbildung 20: Fell-Testszene mit d'Eon's Beleuchtungsmodell (500 Samples) aus verschiedenen Kamerawinkeln. Die Quad-Lichtquelle befindet sich von Bild a) gesehen über der Kamera und zeigt in die Mitte der Bodenplatte.

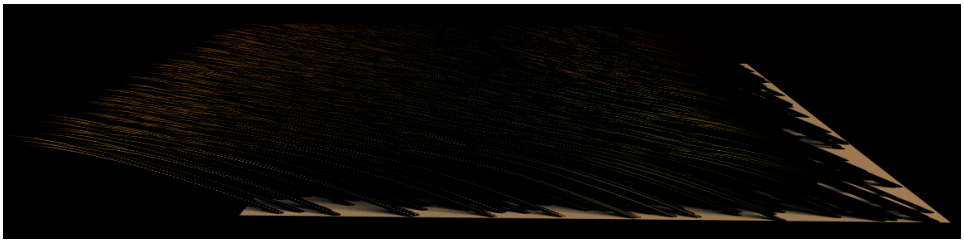
6.4 Haare mit d'Eon et al.

Die Ergebnisbilder von d'Eon's Beleuchtungsmodell sind mit gleicher Testszene, wie bei Marschner's Modell, in Abbildung 20, sowie mit weiteren Kameraeinstellungen im Anhang in Abbildung 24 zu sehen. In diesen Bildern ist zu erkennen, dass das primäre Highlight von den sekundären Highlights deutlicher abgegrenzt ist, als bei Marschner's Modell. Die Problematik der Artefakte durch die Dreiecks-Schnittpunktgeometrie ist in Abbildung 20 c) ebenfalls zu sehen, dafür hat d'Eon's Modell keine Artefakte beim Übergang von Bodenplatte zu Haar, wie in Bild b) einsehbar.

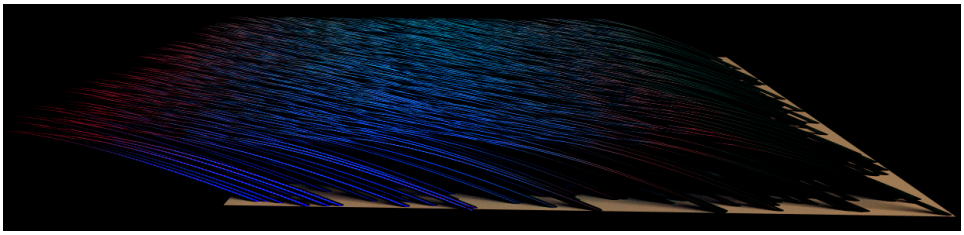
Die Anteile der einzelnen Lichtpfade aus d'Eon's Modell sind in Abbildung 21 mit der gleichen Kameraeinstellung, wie die entsprechenden Gegenstücke von Marschner's Modell dargestellt. Der R-Pfad (a) mit dem primären Highlight ist auch in diesem Modell deutlich zu sehen und breitet sich von der Mitte des Fells nach außen hin aus. Der TRT-Pfad (b) mit sekundären Highlights ist etwas von der Mitte verschoben, tritt in dieser allerdings noch mit kleinen Werten auf. In Bild c) sind zudem Anteile des TT-Pfades in grün zu sehen, besonders am hinteren Rand des Fells, also an den Stellen, an denen die Lichtstrahlen durch das Haar und auf der Rückseite in Richtung Hintergrund austreten und nicht direkt auf die Bodenplatte treffen.



(a) d'Eon's R-Pfad



(b) d'Eon's TRT-Pfad



(c) d'Eon's R-Pfad in blau, TT-Pfad in grün und TRT-Pfad in rot

Abbildung 21: Die Lichtpfade durch das Haar aus d'Eon's Modell jeweils einzeln betrachtet. Außerdem in c) die Anteile der einzelnen Pfade jeweils in einem eigenen Farbkanal dargestellt.

6.5 Vergleich Marschner und d'Eon

Der Hauptaugenmerk dieser Arbeit ist die visuelle Qualität des Fells, gefolgt von der Performance der Modelle. Vergleicht man die Highlights von Marschner's (Abbildung 19) und d'Eon's (Abbildung 21) Modell, kann gesagt werden, dass sowohl das primäre als auch die sekundären Highlights an den gleichen Stellen im Fell auftauchen. Das primäre Highlight aus d'Eon's Modell ist zentrierter als das von Marschner, dafür ist das sekundäre Highlight bei Marschner zur Wurzel des Haares hin stärker ausgeprägt.

Bei der Nutzerfreundlichkeit ist das Modell von d'Eon besser einstellbar. Da in diesem der Absorptionskoeffizient direkt über die Melanin-Konzentrationen des Haares berechnet wird, ist es einfacher die Farbe des Haares anhand der Melanin-Konzentrationen realer Haare einzustellen. Der Absorptionskoeffizient σ_a, welcher in Marschner's Modell genutzt wird, kann nicht mithilfe realer Haare bestimmt werden und muss dementsprechend über Evaluation an die gewünschte Haarfarbe angenähert werden. Dennoch sei gesagt, dass keines der Modelle Parameter zum Einstellen der Haarfarbe über direkt kontrollierbare RGB-Werte mit sich bringt, welche in der Computergrafik genutzt werden um Farben darzustellen.

Für einen Performance-Vergleich sind in folgender Tabelle die Renderzeiten verschiedener Ergebnisbilder nach 500 Samplen angegeben, wobei die Bilder jeweils aus demselben Kamerawinkel für beide Modelle gerendert sind:

Marschner Abbildung	**22 a)**	**22 b)**	**22 c)**
d'Eon Abbildung	**20 a)**	**20 b)**	**24 a)**
Marschner Zeit	7:35 min	12:39 min	11:15 min
d'Eon Zeit	8:07 min	13:16 min	11:37 min

Marschner Abbildung	**23 a)**	**23 b)**	**23 c)**
d'Eon Abbildung	**20 c)**	**24 b)**	**24 c)**
Marschner Zeit	32:23 min	9:33 min	8:54 min
d'Eon Zeit	36:37 min	10:02 min	8:57 min

Es ist zu erkennen, dass d'Eon's Modell für jede Kameraeinstellung eine längere Renderzeit als Marschner's Modell hat. Der Grund dafür ist zum einen der *gaussian detector* (Gleichung 44), der bei d'Eon für die Streuungsfunktion N genutzt wird. Dieser durchläuft 20 Iterationen und berechnet somit 20 Gauß-Funktionen, was rechenintensiver als die Kosinus- und Sinus-Berechnungen der einmalig durchgeführten Gleichungen 28 aus Marschner's Modell ist. Zum anderen wird für die Streuungsfunktion M bei Marschner (Gleichungen 25,26 und 27) eine einfache Gauß-Funktion für jeden Pfad berechnet, während bei d'Eon durch Gleichung 39, zusätzlich zu der e-Funktion und dem Logarithmus, eine Hyperbel-Funktion sowie Bessel-Funktion berechnet werden müssen. Die geforderte erhöhte Rechenleistung spiegelt sich in höheren Renderzeiten bei d'Eon's Modell wieder.

7 Fazit

Sowohl das Beleuchtungsmodell von Marschner et al., als auch das von d'Eon et al., ist für die Testszene mit kurzen Haaren gut geeignet und visuell ansprechend. Die Highlights des Fells sind in beiden Fällen an den, nach der Lichtreflexion im allgemeinen und den in Kapitel 3.2 und 3.3 beschriebenen theoretischen Merkmalen, zu erwartenden Stellen. Während das primäre Highlight in d'Eon's Modell besser auszumachen ist, kann die Fellfarbe in Marschner's Modell auf größeren Flächen einheitlicher erkannt werden. Insgesamt sind in d'Eon's Modell weniger Artefakte auszumachen, aber bei beiden Modellen sind aus der Nah-Ansicht störende Artefakte in der Geometrie des Fells zu sehen. Das d'Eon's Modell energieerhaltend ist und somit besonders bei sehr hellen Fell-Farben ein realistischeres Erscheinungsbild gegenüber Marschner's Modell besitzen sollte (siehe [dFH+11]), ist anhand der Ergebnisbilder nicht festzustellen. Ein Grund dafür ist, dass d'Eon's Modell in dieser Implementation 3 Lichtpfade durch das Haar verfolgt, genau wie Marschner's Modell. Der in Kapitel 3.3 beschriebene Energieverlust von Marschner's Modell tritt aufgrund dieser Beschränkung auf. In d'Eon's Modell müssten zusätzliche Pfade mit einbezogen werden, um diese bei Marschner verlorene Energie mit zu berechnen. Da dies aber einen deutlich erhöhten Rechenaufwand pro zusätzlich betrachtetem Pfad darstellt und gleichzeitig der zusätzliche Anteil am Aussehen des Haares, im Gegensatz zu den vorherigen Pfaden, immer kleiner wird, wurde dies nicht implementiert.

Vergleicht man nur einen Fell-Ausschnitt, wie in Abbildung 22 a), der 3D-Szene mit einem Fell-Ausschnitt eines realen Rehs in Abbildung 26, ist das Nutzen der Modelle für Fell, obwohl diese für menschliches Haar entwickelt wurden, eine plausible Lösung. Es fällt dennoch auf, das besonders die Rauheit und Textur des realen Fells nicht optimal simuliert wird. Um die Frage der Einleitung, ob die Modelle auch für Fell gut genutzt werden können, final zu beantworten, müssen die Ergebnisse noch erweitert werden. Denn dafür müssen die Modelle auf einem vollständig modellierten Tier, in einer der Tierart entsprechenden Umgebung angewandt werden. Das Tier sollte dabei auch mit unterschiedlichen Fellfarben darstellbar sein, denn die meisten Tiere besitzen nicht nur einen Farbton in ihrem Fell. Zusätzlich ist es möglich, das implementierte Modell von d'Eon et al. mit weiteren Lichtpfaden durch das Haar zu erweitern, damit der Effekt dieser und somit der theoretische Vorteil gegenüber dem Modell von Marschner, welches auf 3 Pfade beschränkt ist, besser zu sehen ist.

Die Performance beider Modelle ist für einen fotorealistischen Pathtracer, welcher die Lichtstreuung exakt berechnet, angemessen. Eine Nutzung in Echtzeit ist nicht möglich, aber dies ist weder der Anspruch dieser Implementation noch ist es notwendig für das Anwendungsgebiet, welches beim Rendern von einzelnen Bildern oder Animationsfilmen liegt. Die Nutzung der Zylinder-Schnittpunktobjekte würde sowohl eine Performance- Verbesserung, als auch eine Verminderung der

Artefakte in den Nahaufnahmen mit sich bringen. Aufgrund der zu kleinen Auflösung der Zylinder und daraus resultierenden fehlerhaften Berechnung der Normalen ist das visuelle Ergebnis bei Nutzung der Zylinder-Schnittpunktgeometrie allerdings zu schlecht im Vergleich mit der Dreieck-Schnittpunktgeometrie, um diese nutzen zu können.

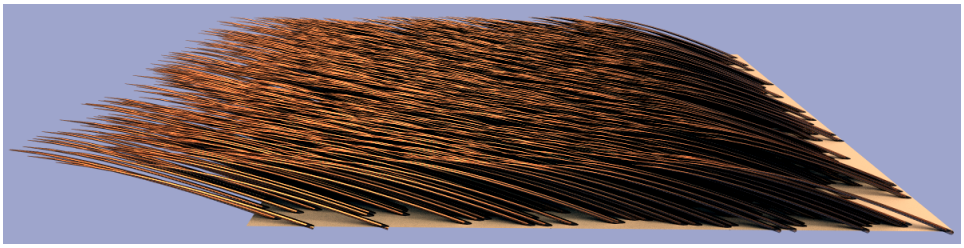
7.1 Zukunftsausblick

Die vorgestellten Modelle von Marschner et al. und d'Eon et al. weisen Schwächen im Bereich der Nutzerfreundlichkeit auf. Aufgrund der Steuerung der Haarfarbe über den Absorptionskoeffizienten, welcher in keiner direkten Relation zu RGB-Farben steht, die beim modellieren von Charakteren für deren Aussehen genutzt werden, ist es für Anwender schwierig, die Parameter der Modelle so einzustellen, dass sie ihr gewünschtes Ergebnis erhalten. Eine Weiterführung von Marschner's Modell, um ein anwendungsfreundliches Beleuchtungsmodell für Haare zu erhalten, wurde von Sadeghi et al. in [SPJT10] vorgestellt. Dieses setzt sich zum Ziel, kontrollierbare Parameter zum Einstellen des Beleuchtungsmodells bereitzustellen, damit Künstler in der Anwendung ihr gewünschtes Aussehen der Haare modellieren können.

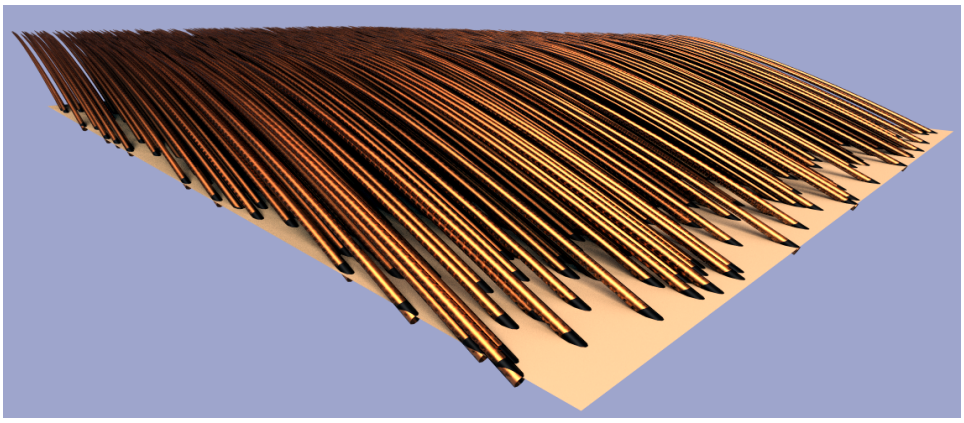
Die genannten Beleuchtungsmodelle sind für Haare im allgemeinen ausgelegt und betrachten die Besonderheiten von Fell, wie das in Kapitel 3.1 beschriebene Vorhandensein der Medulla, nicht. Ein Beleuchtungsmodell, welches sich speziell auf die Lichtreflexion in Fell fokussiert, ist von Yan et al. in [YTJR15] vorgestellt worden und in den folgenden Jahren speziell für globale Beleuchtung als BSSRDF in [YSJR17] erweitert worden. Dieses nimmt auch die potenzielle Reflexion und Refraktion an der Medulla in Betracht und kann so sowohl die Lichtstreuung innerhalb eines Fell-Haares, als auch die Streuung zwischen verschiedenen Fell-Haaren untereinander mithilfe der Volumenstreuung (engl. *subsurface scattering*) simulieren. Für die Zukunft wäre es interessant, ein solches speziell auf Fell bezogenes Modell anhand der in dieser Arbeit geschaffenen Grundlage zu implementieren.

Anhang

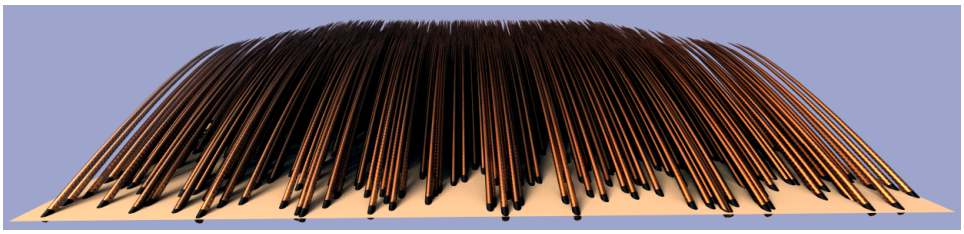
Marschner Haarmodell Bilder



(a) Renderzeit 7:35 Minuten

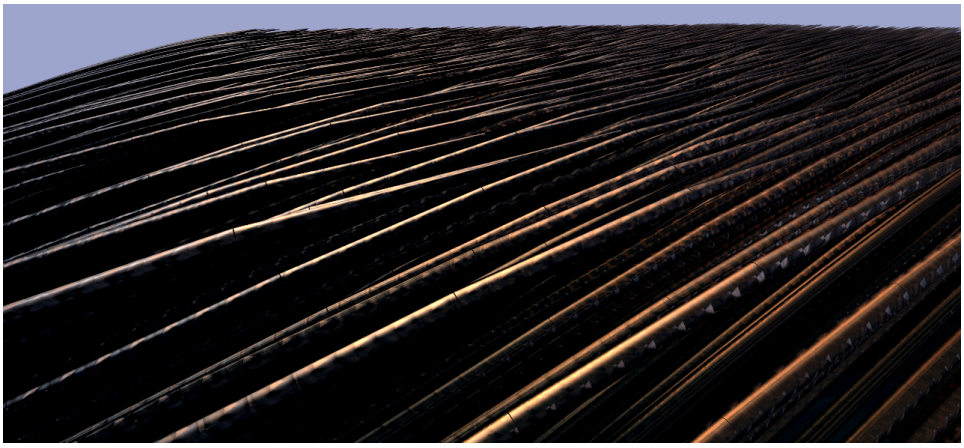


(b) Renderzeit 12:39 Minuten

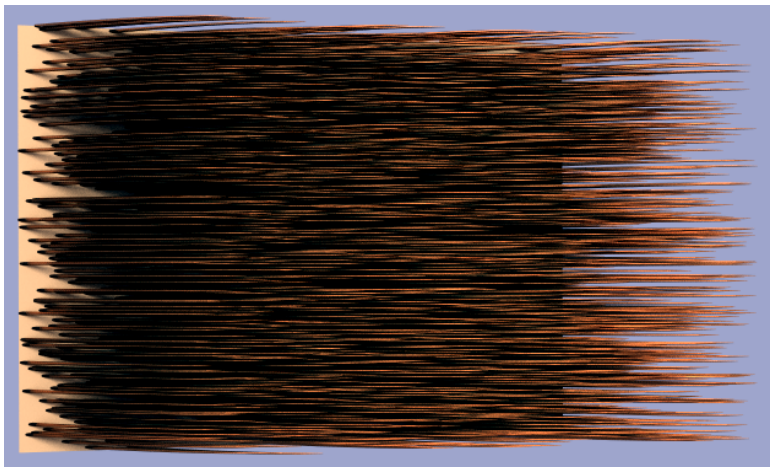


(c) Renderzeit 11:15 Minuten

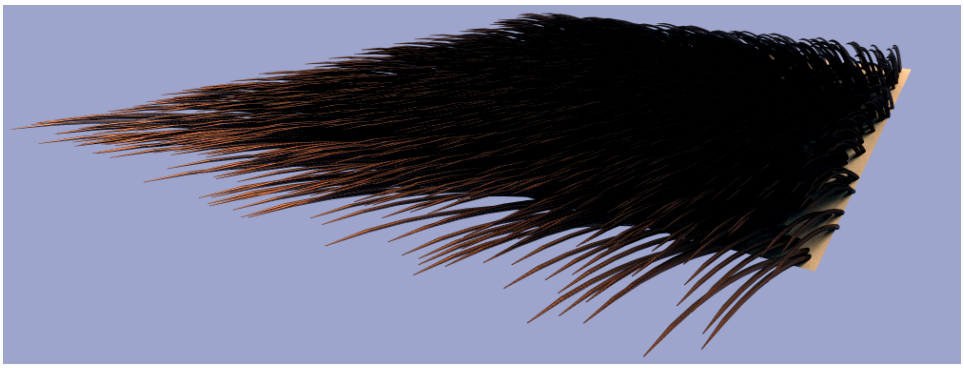
Abbildung 22: Fell-Testszene mit Marschner's Beleuchtungsmodell (500 Samples) aus verschiedenen Kamerawinkeln. Die Quad-Lichtquelle befindet sich von Bild a) gesehen über der Kamera und zeigt in die Mitte der Bodenplatte.



(a) Renderzeit 32:23 Minuten



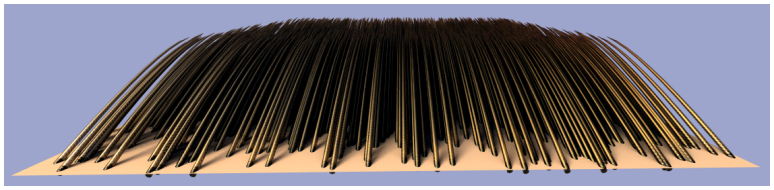
(b) Renderzeit 9:33 Minuten



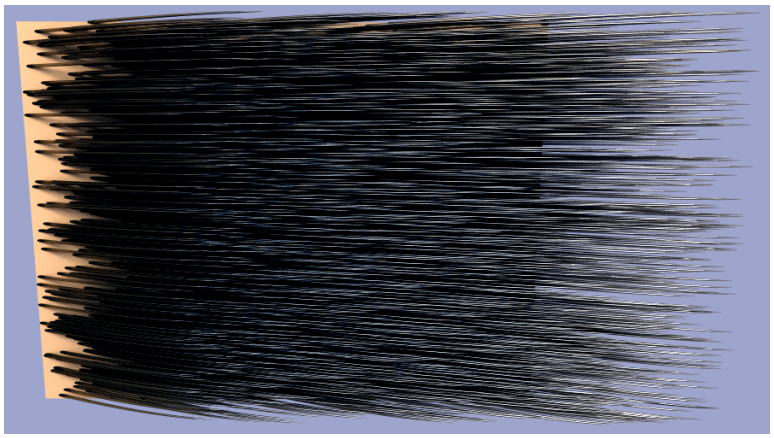
(c) Renderzeit 8:54 Minuten

Abbildung 23: Fell-Testszene mit Marschner's Beleuchtungsmodell (500 Samples) aus verschiedenen Kamerawinkeln. Die Quad-Lichtquelle befindet sich von Bild a) gesehen gegenüber und zeigt in die Mitte der Bodenplatte.

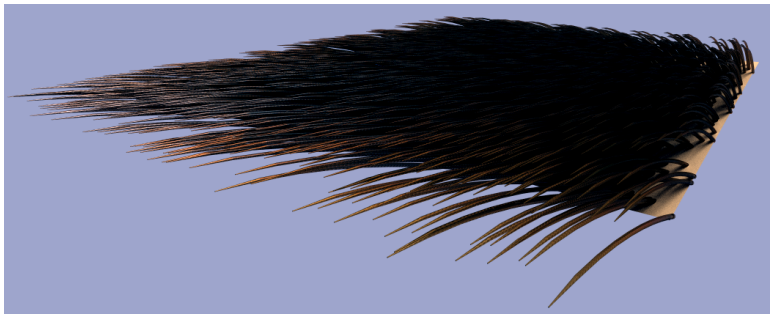
d'Eon Haarmodell Bilder



(a) Renderzeit 11:37 Minuten



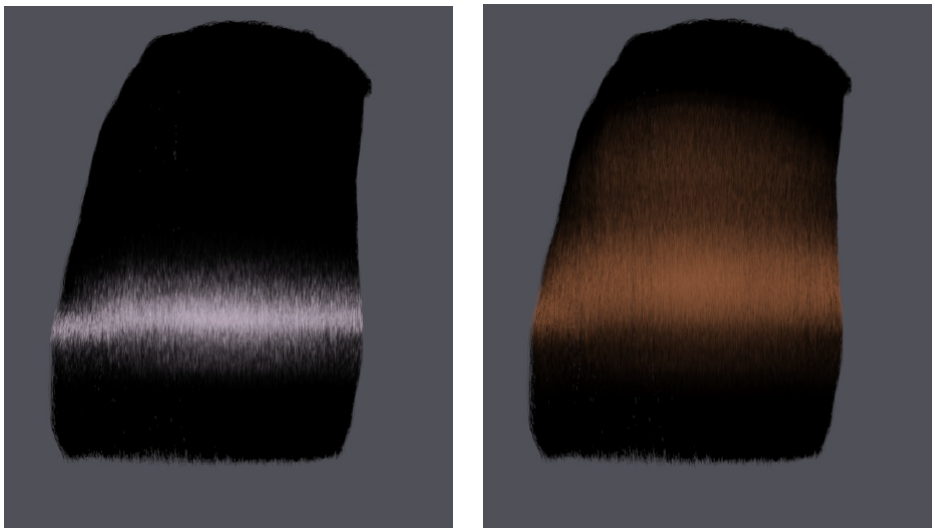
(b) Renderzeit 10:02 Minuten



(c) Renderzeit 8:57 Minuten

Abbildung 24: Fell-Testszene mit d'Eon's Beleuchtungsmodell (500 Samples) aus verschiedenen Kamerawinkeln. Die Quad-Lichtquelle befindet sich von Bild a) gesehen aus links und zeigt in die Mitte der Bodenplatte.

Sonstige



(a) **(b)**

Abbildung 25: Der originale R-Pfad (a) und TRT-Pfad (b) aus Marscher's Paper [MJC+03] (ist ebenfalls Bildquelle).

Abbildung 26: Fell eines echten Rehs. Bildquelle: [reh]

Literatur

[AD17] Arenz Jan Kraft Emma Braun Johannes Nilles Alexander Deeming Alexander Schmidt Niko Ell Nico Schwanekamp Hendrik Hilbig Lucas Zettler Nico Allan David, Korbach Christian. Kirk - koblenz interactive raytracing framework. `https://gitlab.uni-koblenz.de/KIRK/RT`, 2017.

[bF] bilderzwerg Fotolia.com. `https://medlexi.de/Datei:Haare.jpg`.

[BF79] Jon Louis Bentley and Jerome H Friedman. Data structures for range searching. *ACM Computing Surveys (CSUR)*, 11(4):397–409, 1979.

[BHH15] Jiří Bittner, Michal Hapala, and Vlastimil Havran. Incremental bvh construction for ray tracing. *Computers & Graphics*, 47:135–144, 2015.

[BS91] Helen K Bustard and Robin W Smith. Investigation into the scattering of light by human hair. *Applied Optics*, 30(24):3485–3491, 1991.

[CBTB16] Matt Jen-Yuan Chiang, Benedikt Bitterli, Chuck Tappan, and Brent Burley. A practical and controllable hair and fur model for production path tracing. In *Computer Graphics Forum*, volume 35, pages 275–283. Wiley Online Library, 2016.

[dFH+11] Eugene d'Eon, Guillaume Francois, Martin Hill, Joe Letteri, and Jean-Marie Aubry. An energy-conserving hair reflectance model. In *Computer Graphics Forum*, volume 30, pages 1181–1187. Wiley Online Library, 2011.

[DK04a] Douglas W Deedrick and Sandra L Koch. Microscopy of hair part 1: a practical guide and manual for human hairs. *Forensic Science Communications*, 6(1), 2004.

[DK04b] Douglas W Deedrick and Sandra L Koch. Microscopy of hair part ii: A practical guide and manual for animal hairs. *Forensic Science Communications*, 6(3), 2004.

[DS86] H Dathe and P Pelztieratlas Schöps. Veb gustav fischer verlag. *Jena*, 320:8, 1986.

[FNK+89] Wm Randolph Franklin, Chandrasekhar Narayanaswami, Mohan Kankanhalli, David Sun, Meng-Chu Zhou, and Peter YF Wu. Uniform grids: A technique for intersection detection on serial and parallel machines. In *Proceedings of Auto Carto*, volume 9, pages 100–109. Citeseer, 1989.

[GHFB13] Yan Gu, Yong He, Kayvon Fatahalian, and Guy Blelloch. Efficient bvh construction via approximate agglomerative clustering. In *Proceedings of the 5th High-Performance Graphics Conference*, pages 81–88. ACM, 2013.

[Ham60] John M Hammersley. Monte carlo methods for solving multivariable problems. *Annals of the New York Academy of Sciences*, 86(3):844–874, 1960.

[Hay14] William M Haynes. *CRC handbook of chemistry and physics*. CRC press, 2014.

[HvdH81] Hendrik Christoffel Hulst and Hendrik C van de Hulst. *Light scattering by small particles*. Courier Corporation, 1981.

[Jen01] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. AK Peters/CRC Press, 2001.

[Kaj86] James T Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.

[Keu18] Kevin Keul. The cvk framework. `https://gitlab.uni-koblenz.de/CVK`, 2018.

[KHM+98] James T Klosowski, Martin Held, Joseph SB Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization & Computer Graphics*, (1):21–36, 1998.

[MJC+03] 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.

[NEBES13] Helmut Neunzert, Winfried G Eschmann, Arndt Blickensdörfer-Ehlers, and Klaus Schelkes. *Analysis 1: Ein Lehr-und Arbeitsbuch für Studienanfänger*. Springer-Verlag, 2013.

[OA79] Constantin E Orfanos and JPL Astore. *Haar und Haarkrankheiten*. Fischer, 1979.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[Phr] Phrood. `https://commons.wikimedia.org/w/index.php?curid=23778642`.

[PHVL15] Leonid Pekelis, Christophe Hery, Ryusuke Villemin, and Junyi Ling. A data-driven light scattering model for hair. *Pixar Technical Memo 15-02*, 2015.

[PJH16] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[reh] `https://pixabay.com/de/fell-reh-rehwild-natur-s%C3%A4ugetier-2615013/`.

[Rue12] Martin Ruenz. Real-time hair simulation and rendering. 2012.

[Sch04] Thorsten Scheuermann. Practical real-time hair rendering and shading. In *ACM SIGGRAPH 2004 Sketches*, page 147. ACM, 2004.

[Scr] `http://www.scratchapixel.com/lessons/3d-basic-rendering/global-illumination-path-tracing`.

[SGF77] 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.

[SPJT10] 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.

[SS14] Gunter Saake and Kai-Uwe Sattler. *Algorithmen und Datenstrukturen: Eine Einführung mit Java*. dpunkt. verlag, 2014.

[SVNB99] Carlos Saona-Vazquez, Isabel Navazo, and Pere Brunet. The visibility octree: a data structure for 3d navigation. *Computers & Graphics*, 23(5):635–643, 1999.

[Tri71] Ronald Alfred Ranson Tricker. Introduction to meteorological optics. *Introduction to meteorological optics., by Tricker, RAR. London (UK): Mills and Boon, 286 p.*, 1971.

[TT13] Gerald Teschl and Susanne Teschl. *Mathematik für Informatiker: Band 1: Diskrete Mathematik und Lineare Algebra*. Springer-Verlag, 2013.

[Wal07] Ingo Wald. On fast construction of sah-based bounding volume hierarchies. In *Interactive Ray Tracing, 2007. RT'07. IEEE Symposium on*, pages 33–40. IEEE, 2007.

[YSJR17] Ling-Qi Yan, Weilun Sun, Henrik Wann Jensen, and Ravi Ramamoorthi. A bssrdf model for efficient rendering of fur with global illumination. *ACM Transactions on Graphics (TOG)*, 36(6):208, 2017.

[YTJR15] 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.

[Zah89] Helmut Zahn. Das haar aus der sicht des chemikers. *Chemie in unserer Zeit*, 23(5):141–150, 1989.