Physically based Rendering

Fabian Meister

Matrikelnummer: 298 5828
E-Mail: meisterfa68426@th-nuernberg.de
Technische Hochschule Nürnberg Georg Simon Ohm

1 Einleitung

Der Begriff „*Physically Based Rendering/Shading*“ (kurz PBR) beschreibt einen Oberbegriff, der verschiedene Rendering Methoden und Techniken umfasst, welche auf physikalischen Theorien und Prinzipen basieren. Die physikalischen Aspekte sind darauf ausgerichtet, die Wechselwirkung zwischen Licht und Materie so korrekt wie möglich zu modellieren (vgl. S.133 [23] u. [9]).

Dennoch stellt PBR keine physikalisch korrekte Simulation des Lichtes dar, da es Approximations-funktionen verwendet, um den Arbeitsaufwand und somit die gesamt Berechnungsdauer zu verringern. Aus diesem Grund wird es Physically Based (zu dt. physikalisch basiertes) Rendering und nicht Physically Rendering genannt [9].

Über die letzten Jahre wurde das Modell bereits von vielen großen und international bekannten Entwickler- und Animationsstudios umgesetzt, darunter Walt Disney [1], Activision Blizzard[2] und Electronic Arts[3]. Der Grund hierfür ist, dass das PBR-Modell gegenüber älteren Modellen, wie beispielsweise dem Blinn-Phong-Modell, eine Vielzahl an Vorteilen besitzt. In der Literatur werden dabei vor allem Folgende genannt [4, 13, 18, 9]:

- Ermöglicht ein konsistenteres Aussehen unter verschiedenen Beleuchtungsbedingungen
- Bietet ein realistischeres Aussehen
- Steigert die Produktivität der Künstler durch den Einsatz von vereinfachten Bedienelementen

In der folgenden Arbeit wird ein Überblick über die physikalischen Grundlagen, welche für das Verständnis nötig sind, gegeben. Außerdem werden die Methoden und Techniken des Physically Based Rendering vorgestellt und erläutert. Zuletzt wird auf das Projekt, welches in der Präsentation im Fach „*Programmierung von Grafik-Shadern*“ vorgestellt wurde, eingegangen.

2 Physikalische Grundlagen der Radiometrie

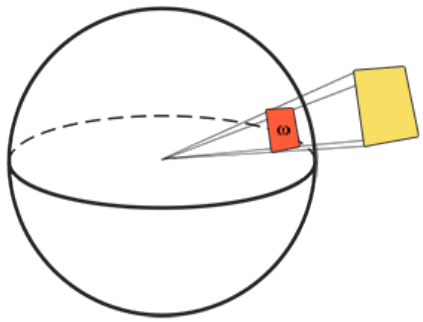
Die Radiometrie ist ein Teilgebiet der Strahlungsphysik, welche sich mit der Messung von elektromagnetischer Strahlung unabhängig von dem menschlichen Auge befasst (vgl. S.179f. [7]). Da im weiteren Verlauf des Papers Fachbegriffe bzw. physikalische Größen der Radiometrie referenziert werden, werden diese im Folgenden näher erläutert.

[1] `https://disney.de/`
[2] `https://www.activisionblizzard.com/`
[3] `https://www.ea.com/`

Der **Strahlungsfluss** (physikalisches Symbol ϕ) repräsentiert die Leistung einer Lichtquelle. Definiert ist die Leistung als die Strahlungsenergie, welche von der Quelle abgestrahlt wird pro gemessene Zeiteinheit (vgl. S.179f. [7]).

$$\Phi = \frac{\Delta Q}{\Delta t} \tag{1}$$

Der **Raumwinkel** (physikalisches Symbol ω) funktioniert analog zum Bogenmaß im 3-dimensionalen Raum. Er ist definiert als das Verhältnis der Kugelfläche zum quadratischen Radius der Kugel. Die Fläche, welche im Bild (1) zu sehen ist, wird gebildet, in dem man die Form auf die Hemisphäre (Einheitskugel) projiziert. Durch den Raumwinkel wird dabei nicht nur die einnehmende Fläche auf der Hemisphäre, sondern auch die Richtung des Objektes bestimmt (vgl. [9] u. S.2 [10]).

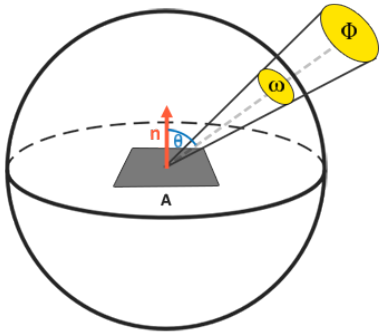


$$\omega = \frac{A}{r^2} \tag{2}$$

Abbildung 1: *Raumwinkel ω*
Quelle: *DeFries[9]*

Die **Strahldichte** (Symbol L) ist ein Maß für die Lichtabstrahlung einer Fläche. Sie definiert die Leistung einer Lichtquelle pro Raumwinkel ω und pro Flächeninhalt der emittierende Fläche $A \cdot \cos\varepsilon$. Der zusätzliche Gewichtungsfaktor $\cos\varepsilon$ im Nenner beschreibt dabei das Phänomen, dass eine Fläche unter einem Betrachtungswinkel kleiner wirkt als bei einer senkrechten Betrachtung der Fläche (vgl. S.8f.[16], S.36f [20] u. [3, 9]).

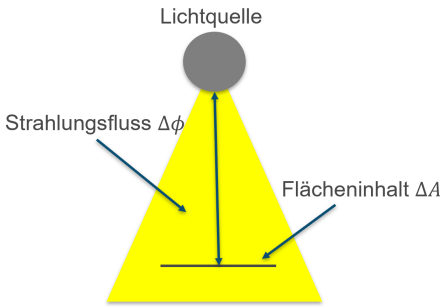
Zudem gehört die Strahldichte zu den sogenannten raumwinkelabhängigen Größen. Diese besitzen keine Abhängigkeiten bezüglich der Entfernung zu der Lichtquelle. Unter der Annahme eines Vakuums verändern sich somit die Werte der Strahldichte bis zum Auftreffen auf der Oberfläche nicht. Wodurch es möglich wäre mit der Formel der Strahldichte die gesamte Verteilung des Lichtflusses zu beschreiben (vgl. [19]).



$$L = \frac{\Delta\Phi \cdot}{\Delta\Omega \cdot \Delta A \cdot \cos\varepsilon} \tag{3}$$

Abbildung 2: *Strahldichte L unter einem Winkel θ* ***Quelle:*** *DeFries[9]*

Die Bestrahlungsstärke (Symbol E) ist ein Maß für die Stärke der Lichteinstrahlung auf einer Oberfläche. Sie ist definiert als die Relation zwischen der Strahlungsfluss $\Delta\phi$, die auf eine gegebene Fläche mit dem Flächeninhalt ΔA einfällt (vgl. S.10 [16]):



$$E = \frac{\Delta\phi}{\Delta A} \tag{4}$$

Abbildung 3: *Visualisierung der Bestrahlungsstärke*

Die oben gezeigt Formel in ihrer einfachsten Form besitzt eine Vielzahl von Abhängigkeiten (vgl. S.10 [16]):

- Position der Fläche
- Orientierung der Fläche
- Distanz von Lichtquelle zur Fläche

Um diese Abhängigkeiten mit einzubeziehen, verwendet man für Physically Based Rendering folgende Formel der Bestrahlungsstärke:

$$E = \int_{\Omega} L \cdot \cos\varepsilon \; d\omega \tag{5}$$

Sie beschreibt die Bestrahlungsstärke über die Strahlungsdichte multipliziert mit einem Kosinus Faktor, welcher sich aus dem Winkel zwischen der Flächennormalen (n) und der Richtung des einfallenden Lichtstrahls (ω_i) ergibt. Dieser Faktor repräsentiert das Lambertsches Kosinusgesetz, welches die jeweilige Geometrie der Situation (die Positionierung von Lichtquelle zur Oberfläche) mit einbezieht. Je direkter das Licht auf die Oberfläche auftrifft, umso größer ist schlussendlich die Intensität des Lichtstrahls, welche wahrgenommen werden kann (vgl. S.21f [11] u. [21, 2]).

Um letzten Endes die Gesamtmenge des auf die Fläche fallenden Lichts zu errechnen, wird die Summe der Strahldichten aller Lichtquelle gemessen. Physikalisch korrekt wird hierzu die Summe als Integral über die Hemisphäre Ω gebildet.

3 Reflectance Equation

Wie bereits in der Einleitung erwähnt, wird beim Physically Based Rendering die Interaktion zwischen dem Licht und den Materialen eines Objektes modelliert. Um dies zu berechnen, verwendet das PBR eines der derzeit besten Modelle zur Simulation der visuellen Erscheinungen des Lichtes, die sogenannte Reflectance Equation (zu dt. Reflexions-Gleichung) (vgl. [9]).

$$L_o\left(p, \omega_o\right) = \int_{\Omega} f_r\left(p, w_i, W_o\right) L_i\left(p, w_i\right) n \cdot w_i \; dw_i \tag{6}$$

Diese Formel ist eine Vereinfachung bzw. eine Spezialisierung der Render Equation (zu dt. Render Gleichung), welche 1986 von David Immel et al. und James Kajiyain, in ihrem Artikel „*The Rendering Equation*“ (vgl. [15]) vorgestellt wurde. Mit ihr lässt sich die Strahldichte (L_o) berechnen, welche im Kapitel (2) bereits detailliert beschrieben wurde. Die Strahldichte im speziellen Fall der Reflectance Equation liefert eine Aussage über die Gesamtmenge an Licht, welche von einem Punkt (p) ausgehend entlang einer bestimmten Blickrichtung (ω_o) reflektiert wird. Dabei ist die Intensität der reflektierenden Strahlung abhängig von der Menge an Licht, welche in dem jeweiligen Punkt auftrifft und der Wechselwirkung des Lichts mit der Oberfläche (vgl. [9]).

Ersteres wird durch die sogenannte Bestrahlungsstärke aus dem Kapitel (2) erfasst. Letzteres wird durch die Bidirectional Reflective Distribution Function (kurz BRDF) ausgedrückt. Auf die BRDF und deren einzelnen Funktionen, wird in einem späteren Kapitel im Detail eingegangen. Das Ergebnis aus BRDF und Bestrahlungsstärke wird anschließend mit einem Kosinus Faktor gewichtet, welcher das Lambertsche Kosinusgesetz aus Kapitel (2) repräsentiert (vgl. S.133f. [23] u. [9]).

3.1 Bidirectional Reflective Distribution Function

Das Reflexionsverhalten eines Objektes ist im Allgemeinen bestimmt durch den Ein-/Ausfallswinkel des Lichtes und das Material aus, welchem es besteht. So lässt sich beispielsweise in der Natur unter gleichbleibenden Beleuchtungsverhältnissen bei verschiedenartigen Materialien ein unterschiedlich starker Glanz beobachten. Dieses Verhalten wird durch die BRDF ausgedrückt, welche 1977 amtlich vom National Bureau of Standards (USA) definiert wurde, um Reflexionsdarstellungen und -berechnungen zu vereinheitlichen (vgl. S.1 [10] u. S.2 [22]). In ihrer Grundform beschreibt die BRDF den Zusammenhang zwischen der differentiellen Strahldichte in der Betrachtungsrichtung und der differentiellen Bestrahlungsstärke, welche aus der Beleuchtungsrichtung auf die Oberfläche einwirkt (vgl. S.3 [10]). Vereinfacht ausgerückt beschreibt die BRDF den Anteil eines Lichtstrahles, welcher beim Betrachter ankommt.

$$f_r\left(p, w_i, w_o\right) = \frac{\Delta L_o\left(p, \omega_o\right)}{\Delta E_i\left(p, w_i\right)} \tag{7}$$

Damit eine BRDF auch ein physikalisches Modell darstellen kann, müssen folgende zusätzliche physikalische Gesetze gelten (vgl. S.312f. [5], S.2 [22], S.3 [10] u. [9]):

1. Die Helmholz-Reziprozität, welche besagt, dass beim Vertauschen des Einfalls- und Ausfallswinkels des Lichtes der Funktionswert der BRDF sich nicht verändert. In der Praxis verletzen BRDFs, die beim Rendering verwendet werden, häufig die Helmholtz-Reziprozität ohne das erkennbare Artefakte entstehen. Eine Ausnahme bilden hierbei Offline-Rendering-Algorithmen, die speziell Reziprozität erfordern.

2. Der Energieerhaltungssatz gibt an, dass die Energie, die ein System verlässt, niemals größer sein kann als die Energie, welche einem System hinzugefügt wurde. Dies bedeutet bezogen auf die BRDF, dass die Summe des Lichtes, welches in alle Richtungen reflektiert wird, nicht mehr sein kann als die Menge des auftretenden Lichtes. Verstößt eine BRDF signifikanter Weise gegen die Energieerhaltung, so werden Oberflächen deutlich zu hell dargestellt. Wodurch schlussendlich der Realismus beeinträchtigt wird.

3. Die Superposition ist eine weitere häufig genannte Eigenschaft von BRDFs. Sie besagt, dass Lichtstrahlen, welche auf den gleichen Punkt einer Oberfläche auftreffen, keinen Einfluss aufeinander auswirken.

3.2 Metalle und Nicht Metalle

Um die Theorien bzw. Funktionsweisen der BRDFs in den folgenden Kapiteln besser verstehen zu können, muss zunächst das unterschiedliche Verhalten von metallischen (Leiter) und nichtmetallischen (Dielektrikum) Oberflächen bei Lichteinwirkung verdeutlicht werden.

Trifft ein Lichtstrahl auf eine Oberfläche, dann wird das Licht in einen Brechungsanteil und einen Reflexionsanteil aufgeteilt. Der Reflexionsteil ist das Licht, welches direkt von der Oberfläche reflektiert wird und dabei nicht in das Material eindringt. Diese Art der Reflexion wird auch Spiegelnde Reflexion (dt. für Specular reflection) genannt. Sie folgt dem physikalischen Reflexionsgesetz, welches besagt, dass auf einer vollkommen ebenen Oberfläche der Reflexionswinkel gleich dem Einfallswinkel ist. Jedoch sind die meisten Oberflächen nicht perfekt plan, weswegen die Richtung der Reflexion von der Oberflächenrauheit abhängt. Auf raueren Oberflächen erscheinen die Reflexionen der Lichter insgesamt zerstreuter und gleichzeitig dunkler. Bei glatteren Oberflächen bleiben die Spiegelreflexionen fokussiert, und erscheinen dabei intensiver. Der Brechungsanteil ist der verbleibende Anteil des Lichtes, welcher in das Material gebrochen wird. Im inneren trifft das Licht auf mikroskopisch kleine Unterschiede in der Materialdichte. Hierbei wird es an den Grenzen zwischen den verschiedenen Dichten abermals gestreut, gebrochen bzw. reflektiert und zum Teil wieder als diffuse Reflexion zufällig in den Raum zurück reflektiert. Im Laufe dieses Prozesses absorbiert das Material teilweise die Energie des Lichtes. Bewegt es sich zu lange in einem solchen Material, kann es vollständig absorbiert werden. Infolgedessen hat das Licht, welches dieses Material tatsächlich verlässt, wahrscheinlich nur eine sehr geringe Entfernung vom Eintrittspunkt zurückgelegt. Daher kann der Abstand zwischen dem Eintritts- und Austrittspunkt als vernachlässigbar gering angesehen werden (vgl. S.5 [10], S.305f./316f. [5], S.6/11 [14] u. [9, 18, 12]).

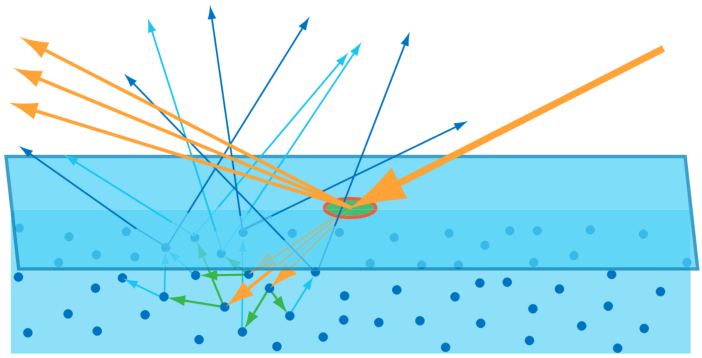


Abbildung 4: *Reflexion und Brechung beim auftreffen eines Lichtstrahles* ***Quelle:*** *Hoffman, S.9 [14]*

Metalle besitzen einen sehr hohen Absorptionskoeffizienten für die Strahlung im sichtbaren Spektrum, da das gebrochene Licht sofort von freien Elektronen absorbiert wird. Infolgedessen verlässt das gebrochene Licht die Oberfläche des Metalls nicht. Daher muss bei der diffusen Reflexion zwischen Leitern und Dielektrika unterschieden werden. Rein

metallischen Materialien besitzen somit keine diffuse Reflexion. Eine Streuung des Lichtes unter der Oberfläche findet nur bei den Dielektrika statt, d.h. sie allein besitzen sowohl spiegelnde als auch diffuse Komponenten. Zuletzt könnte man noch zwischen Halbleiter (Semiconductor) unterscheiden. Diese werden aber in der Regel aus Einfachheit zu den nicht Metallen gezählt (vgl. S8 [14] u. [18, 9, 12]).

Möchte man im Folgenden die gesamte Reflexion einer Oberfläche berechnen, so errechnet man einmal die Diffuse und die Spiegelnde separat voneinander und addiert diese anschließend (vgl. [9]) (wie im Bild 5 dargestellt).

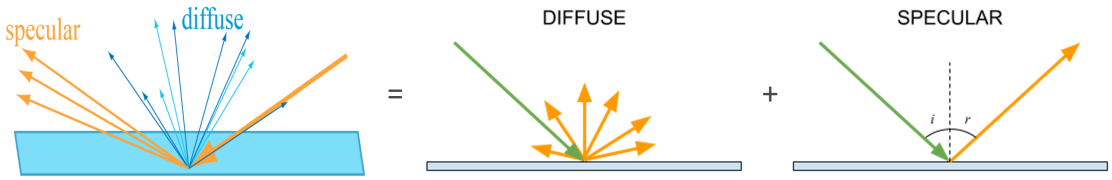


Abbildung 5: *Berechnung der Gesamtreflexion aus diffuser und spiegelnder Reflexion* ***Quelle:*** *astrobites. org [17]*

3.3 Lambert Diffuse BRDF

Die Lambert Diffuse BRDF ist das einfachste BRDF-Modell, um diffuse Reflexion zu modellieren. Es basiert auf der Idee, dass alle Oberflächen perfekt diffuse und somit sogenannte Lambertsche Flächen bzw. Lambertschen Reflektoren sind. Ein Lambertscher Reflektor beschreibt eine Fläche, bei der die Strahldichte der Reflexion in jedem Punkt über die gesamte Fläche und in alle Richtungen gleich groß ist. Vereinfacht gesagt beschreibt es eine Oberfläche, welche das eintreffende Licht in alle Richtungen gleichmäßig verstreut. Da in jedem Punkt die Lichtstreuung identisch ist, muss das Verhältnis von eingehender zu ausgehender Beleuchtung konstant sein und somit auch die BRDF (vgl. S.313f. [5], S.5f. [10] u. [9]).

$$f_{Lambert} = \frac{c}{\pi} \tag{8}$$

Der konstante Reflexionswert einer Lambertschen Diffuse BRDF wird allgemein als die diffuse Farbe oder Albedo bezeichnet und besitzt Werte, die zwischen 0 und 1 liegen können. Der Faktor $1/\pi$ ergibt sich dabei aus der Integration eines Kosinus Faktors über die Halbkugel (vgl. S.314 [5])

Obwohl dieses Reflexionsmodell physikalisch nicht plausibel ist, stellt es eine solide Annäherung an viele reale Oberflächen dar. Außerdem ist die Lambertsche Diffuse BRDF äußerst effizient, da nur mit konstanten Werten gerechnet werden muss. Dennoch muss zuletzt erwähnt werden, dass es verschiedene Gleichungen für den diffusen Teil der BRDF gibt, die tendenziell realistischer aussehen, allerdings auch einen größeren Rechenaufwand benötigen (vgl. [9]).

3.4 Cook-Torrance specular BRDF

Die Cook-Torrance BRDF, auch Blinn-Cook-Torrance BRDF genannt, ist ein Modell, welches die Glanzlichtreflektion des Lichtes auf einer Oberfläche beschreibt. Der Term der BRDF besitzt folgende Form:

$$f_{CookTorrance} = \frac{D \cdot G \cdot F}{4 \cdot (w_0 \cdot n)\,(w_i \cdot n)} \tag{9}$$

Im Zähler besitzt die Cook-Torrance BRDF die Verteilungsfunktion der Normalen (D), die Fresnel Gleichung (F) und die Geometrie Funktion (G), welche in folgenden Kapiteln näher erläutert werden. Der Nenner mit dem Term $4 \cdot (w_0 \cdot n)(w_i \cdot n)$ ist ein Normalisierungsfaktor der Funktionen, welcher zu berücksichtigen gilt (vgl. [9]).

Jede der eben genannten Funktionen ist dabei nur eine Approximation ihrer physikalischen Äquivalente. Da unterschiedliche Art und Weisen bestehen, um sich der zugrunde liegenden Physik anzunähern, existiert eine Vielzahl an Approximationsfunktionen. Diese unterscheiden sich dabei in Effizienz und Realismus. Im weiteren Verlauf des Papers wird sich an der Trowbridge-Reitz GGX für D, die Fresnel-Schlick-Approximation für F und die Smith's Schlick-GGX für G orientiert (vgl. [9]).

3.4.1 Mikrofacetten-Modell

Wie bereits in einem vorherigen Kapitel angesprochen sind Oberflächen nie perfekt plan, weswegen das Reflexionsverhalten eines Objektes von der Oberflächenrauheit bzw. Beschaffenheit abhängt. Um Oberflächenstrukturen nahezu physikalische korrekt abzubilden, verwendet die Cook-Torrance BRDF wie viele BRDFs ihrer Art ein Mikrofacetten-Model/Theorie. Die Grundannahme des Modells ist, dass eine Oberfläche aus vielen Mikrofacetten besteht, die zu klein sind, um einzeln wahrgenommen zu werden. Jede dieser Mikrofacetten ist optisch flach, gleichgroß und ein perfekter Spiegel. In Abhängigkeit von der Rauheit einer Oberfläche kann die Orientierung zwischen den Facetten zueinander sehr unterschiedlich ausfallen (vgl. S.12 [14], S.331f. [5] u. [9]).

Abbildung 6: *Visualisierung einer rauen Oberfläche im Microfacet-Model* ***Quelle:*** *DeFries [9]*

Abbildung 7: *Visualisierung einer glatten Oberfläche im Microfacet-Model* ***Quelle:*** *DeFries [9]*

Die Bilder 6 und 7 zeigen exemplarisch zwei unterschiedlich raue Oberflächen, welche durch ein Mikrofacetten-Modell dargestellt werden. Bild 6 veranschaulicht eine sehr raue Oberfläche, bei welcher die einzelnen Facetten sehr unregelmäßig angeordnet sind. Dies führt dazu, dass die eintreffenden Lichtstrahlen auf rauen Oberflächen mit größerer Wahrscheinlichkeit in völlig unterschiedliche Richtungen gestreut werden, was zu einer diffuseren Spiegelreflexion führt. Bild 7 zeigt im Gegensatz dazu eine nahezu ebene Oberfläche, welche die Lichtstrahlen in etwa die gleiche Richtung reflektiert. Die Reflexion scheint hierdurch gebündelter und schärfer. Folglich existiert eine Abhängigkeit zwischen der Rauheit und dem Reflexionsverhalten eine Oberfläche (vgl. [9]).

3.4.2 Verteilungsfunktion der Normalen

Wie bereits im vorherigen Kapitel erwähnt wird in der Mikrofacetten-Theorie angenommen, dass eine Facette immer ein perfekter Spiegel ist und somit auch dessen Reflexionseigenschaften besitzt. Für einen ebenen Spiegel entspricht bei der Reflexion der Ausfallswinkel des Lichtstrahles gleich dem Einfallswinkel. In der Cook-Torrance BRDF muss

diese Eigenschaft berücksichtigt werden. Nur die Mikrofacetten, welche zufälligerweise genau im richtigen Winkel zum eintreffenden Licht orientiert sind, leisten schlussendlich einen Beitrag zum gesamten BRDF-Wert. Die Ausrichtung einer Facette wird dabei durch ihre Flächennormale (m) bestimmt. Eine Fläche reflektiert dann das Licht zum Betrachter, wenn dessen Normale in der Mitte zwischen dem Vektor des einfallenden Lichtes (l) und dem Vektor in Richtung des Betrachters (v) ist. Folglich ist die Flächennormale der Reflektierenden Flächen gleich der Winkelhalbierenden (h) (dt. für Halfway vector) zwischen l und v. Solche Facetten werden aktive Mikrofacetten genannt (vgl. S.15f. [14], S.332f./337f. [5] u. [9]).

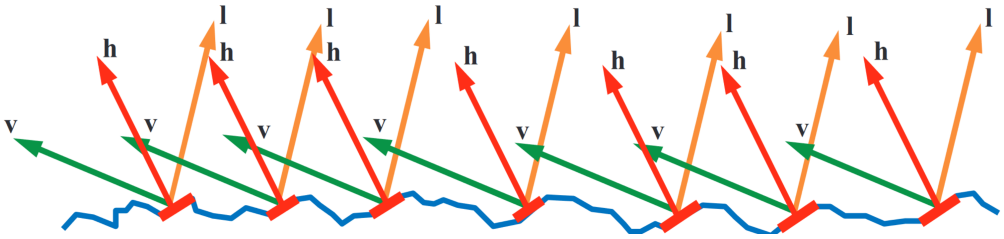


Abbildung 8: *Orientierung der aktiven Mikrofacetten für die $m = h$ gilt* ***Quelle:*** *Real-Time Rendering: 4th edition, S.337 [5]*

Die Verteilungsfunktion der Normalen D (dt. für normal distribution function kurz NDF) approximiert statistisch den relativen Flächeninhalt von Mikrofacetten, die exakt so orientiert sind, dass $m = h$ ist. Es gibt eine Vielzahl von statistischen Verteilungsfunktionen, die eine allgemeine Ausrichtung der Mikrofacetten bei gegebenem Rauheitsparameter approximieren. In der Arbeit wurde hierfür die Trowbridge-Reitz GGX Approximationsfunktion verwendet (vgl. [9]):

$$D_{GGXTR}(n, h, \alpha) = \frac{\alpha^2}{\pi \cdot \Big((n \cdot h)^2 \cdot (\alpha^2 - 1) + 1\Big)^2} \tag{10}$$

Die Funktion besitzt neben der Oberflächennormalen (n) und dem Halfway Vektor (v) noch einen weiteren Parameter α. Dieser bestimmt die Oberflächenrauheit eines Materials und kann grundsätzlich frei gewählt werden. Dennoch empfiehlt es sich hier an bereits etablierte Formeln zur Berechnung von α zu orientieren. Die weiter oben dargestellte Formel zur Berechnung von α wird in einem Paper von Walt Disney Animation Studios und Brent Burley ([8]) vorgestellt.

3.4.3 Geometrie-Funktion

Nicht alle Mikroflächen, für die $m = h$ gilt, tragen schlussendlich zur Reflexion bei. Es existieren weitere Gegebenheit, welche durch das Mikrofacetten-Modell entstanden sind und in der Cook-Torrance-BRDF berücksichtigt werden müssen. Zu diesen gehören die Selbstbeschattung (dt. für self shadowing) und die Abblendung (dt. für masking), welche durch die V-Förmigen Hohlräume (V-cavities) zustande kommen (vgl. S.12f. [14] u. S.9 [10]). Die folgenden Abbildungen veranschaulichen die genannten Gegebenheiten:

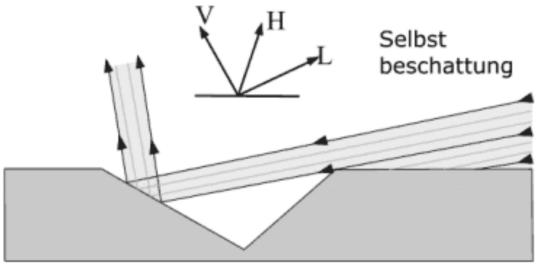


Abbildung 9: *Selbstbeschattung einer Mikrofacette* ***Quelle:*** *Gebhardt, S.9 [10]*

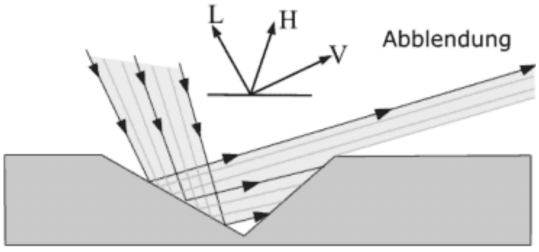


Abbildung 10: *Abblendung einer Mikrofacette* ***Quelle:*** *Gebhardt, S.9 [10]*

Es kommt zu einer Selbstbeschattung, wenn ein Teil des Lichtes bevor es an einer aktive Mikrofacette ankommt von einer anderen Facette abgelenkt wird. Eine Abblendung entsteht dadurch, dass ein Anteil des reflektierten Lichts, bevor es beim Betrachter ankommt, an einer Mikrofacette umgelenkt wird. Folglich läuft es in beiden Fällen darauf hinaus, dass nur ein Bruchteil des Lichtes, welches durch eine aktive Mikrofacette in Richtung des Betrachters reflektiert wurde, auch bei ihm ankommt.

Der Abblendungs- und Selbstbeschattungs-Term wird in der Literatur auch oft als Geometrie G bezeichnet. Die Funktion $G(l, v, m)$ stellt die Wahrscheinlichkeit dar, dass Mikrofacetten mit einer gegebenen Normalen m sowohl aus der Lichtrichtung als auch aus der Blickrichtung v sichtbar (somit nicht verdeckt oder verschattet) sind. Da G eine Wahrscheinlichkeit darstellt, sind ihre Werte skalar und müssen deswegen zwischen 0 und 1 liegen (vgl. S.16 [14]).

Zur Berechnung der Geometrie-Funktion wurde eine Kombination aus der GGX- und der Schlick-Beckmann-Approximation, bekannt als die Schlick-GGX Approximation, verwendet, welche folgende Form besitzt:

$$G_{SchlickGGX}\left(n, v, k\right) = \frac{n \cdot v}{(n \cdot v) \cdot (1 - k) + k} \tag{11}$$

$$k_{direct} = \frac{(\beta + 1)^2}{8} \tag{12}$$

Neben der Oberflächennormale und dem Licht-/Betrachtungsvektor benötigt die Schlick-GGX-Funktion einen weiteren Parameter k. Dieser ist eine Abbildung der Rauheit β, welche variiert je nachdem, ob die Geometrie-Funktion für direkte Beleuchtung oder IBL-Beleuchtung verwendet wird. In der Literatur wird in der Regel für die Rauheit der Geometrie-Funktion der griechische Buchstabe α, wie bei der Verteilungsfunktion der Normalen aus Kapitel 3.4.2 verwendet. Diese Doppeltbelegung des Terms könnte suggerieren, dass die Rauheit der Geometrie-Funktion und die der Verteilungsfunktion der Normalen in Zusammenhang stehen. Dies ist allerdings nicht der Fall, da die Rauheit der Geometrie frei gewählt werden darf. Dennoch empfiehlt es sich wie bei der D-Funktion den Rauheitswert anhand etablierter Methoden zu errechnen (vgl. [9]).

Um die Geometrie effektiv approximieren zu können, muss sowohl die Blickrichtung (Geometrie Abblendung) als auch der Lichtrichtungsvektor (Geometrie Selbstbeschattung) berücksichtigt werden. Hierfür wird die Methode von Smith verwendet, welche beide geometrischen Besonderheiten mit einbezieht (vgl. [9]):

$$G_{Smith}\left(n, v, l, k\right) = G_{sub}\left(n, v, k\right) \cdot G_{sub}\left(n, l, k\right) \tag{13}$$

3.4.4 Fresnel Gleichung

Wie bereits aus einem Vorherigen Kapitel bekannt ist, wird Licht, wenn es auf eine Oberfläche auftrifft in einen Reflektierenden und einen Gebrochenen Anteil aufgeteilt. Die Teilmenge des Reflektieren Lichtes wird durch die sogenannte Fresnel Gleichung (dt. für Fresnel Equation) F beschrieben, welche von Augustin-Jean Fresnel erfunden wurde. Die Ergebnisse der Funktion variieren in Abhängigkeit von zwei Faktoren: dem Einfallswinkel (Winkel zwischen Lichtvektor und Oberflächennormale) und dem Brechungsindex des Materials. Die Werte befinden sich dabei immer zwischen 0 und 1. Der Grund hierfür ist, dass eine Fläche nicht weniger als 0 % oder mehr als 100 % des einfallenden Lichtes reflektieren kann. Außerdem ist das Endergebnis als ein RGB-Vektor definiert. Hierdurch wird der Fresnel-Effekt für alle drei Farbkanäle modelliert (vgl. S.316ff.[5], S.13f. [14]). Die Folgende Grafik veranschaulicht für verschiedenen Materialien den Zusammenhang des Reflexionsanteiles in Abhängigkeit zum Eintrittswinkel:

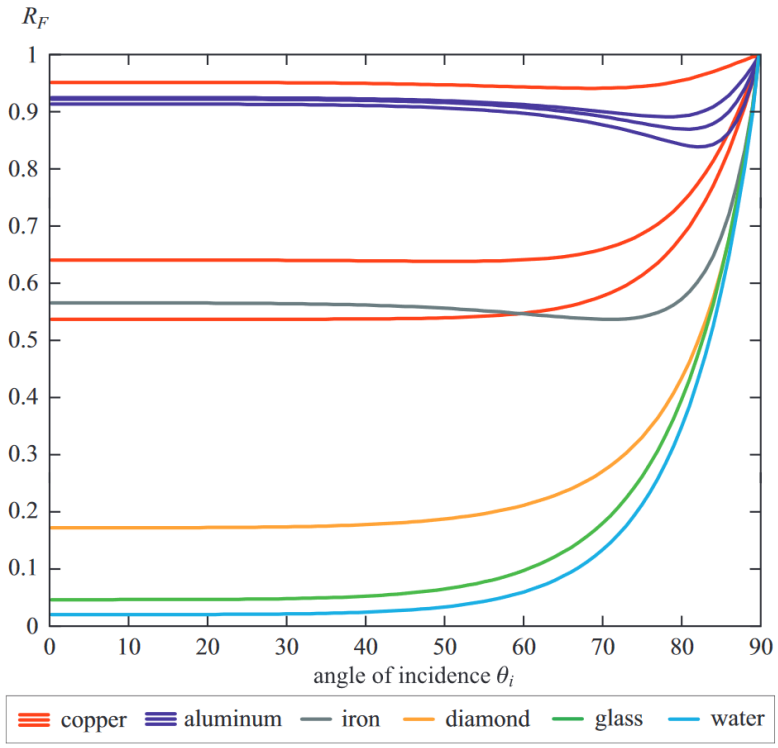


Abbildung 11: *Fresnel-Reflexion von verschiedenen Substanzen in Abhängigkeit vom Einfallswinkel* ***Quelle:*** *Real-Time Rendering: 3rd edition, S.233 [6]*

In dem Diagramm fällt auf, dass der Reflexionsgrad bei einem Einfallswinkel von 0◦ bis etwa 45◦ nahezu konstant und am Niedrigsten ist. Zwischen 45◦und ca. 75◦ verändert sich der Reflexionsgrad deutlicher. Im weiteren Verlauf des Reflexionsgrads zwischen 75◦und 90◦ strebt dieser fast exponentiell immer gegen 1. Des Weiteren zeigt die Grafik das Metalle, wie beispielweise Kupfer, unterschiedliche Werte für die drei Farbkanäle Rot, Grün und Blau besitzen. Aufgrund dessen haben Metalle einen farblichen Glanz, welcher für Kuper z.B. eher rötlich ist und je nach Auftrittswinkel des Lichtes stärker oder weniger stark durchkommt.

Zu erkennen ist außerdem, dass Metalle grundsätzlich einen deutlich höheren Refle-

xionsgrad als Dielektrika besitzen. Tatsächlich haben die Dielektrika bei 0 Grad Einfallswinkel ($F(0)$) einen Reflexionsanteil von 1% bis maximal 17%, während Metalle einen Anteil größer gleich 50% besitzen. Da der Fresnel-Reflexionsgrad im Bereich von 0 Grad bis 45 Grad nahezu identisch ist, wird bei 0 Grad der Wert der Fresnel-Gleichung als das Basis-Reflexionsvermögen des Materials bezeichnet. Dieser stellt eine wichtige Konstant bei der Berechnung der Fresnel-Gleichung dar und wird im Folgenden als F_0 bezeichnet. F_0 lässt sich dabei mit Hilfe des sogenannten Brechungsindexes (dt. für indices of refraction) berechnen (vgl. S.321 [5], S.13 [14] u. [9]).

Die vollständigen Fresnel-Gleichungen ist sehr komplex und benötigt einige Materialparameter, wodurch diese Funktion nicht besonders einfach für Künstler und Entwickler zu benutzen ist. Wie bei den anderen Funktionen auch wird bei der Fresnel-Gleichung eine Approximationsfunktion genutzt, die Sogenannte Fresnel-Schlick Approximation, welche Folgende Form besitzt (vgl. [9]):

$$F_{Schlick}\left(h, v, F_0\right) = F_0 + (1 - F_0) \cdot (1 - (h \cdot v))^5 \tag{14}$$

Es fällt auf, dass die Fresnel-Schlick Funktion nicht die Oberflächennormale n sondern den Halfway-Verktor h zur Berechnung des Eintrittes Winkels verwendet. Dies hängt damit zusammen, dass bei der Fresnel-Gleichung nur die aktiven Mikrofacetten betrachtet werden. Für die Oberflächennormale m einer Mikrofacette gilt m = h. Somit wird auch bei der Fresnel-Gleichung die Orientierung der Mikrofacetten mit einbezogen.

Des Weiteren ist es möglich bei der Berechnung der Fresnel-Gleichung noch zusätzliche Annährung zu machen. Für die dielektrischen Oberflächen wird beispielweise ein fester Basisreflexionsgrad gesetzt ($F_0 = 0{,}04$), welcher dennoch zu physikalisch plausiblen Ergebnissen führt (vgl. [9]).

Zuletzt lässt sich mit der Fresnel-Gleichung auch der Anteil des diffusen Lichtes ausrechnen. Hierzu wird eine Aussage über die Energieerhaltung getroffen. Die gesamte Energie des Lichtstrahls, welcher auf eine Oberfläche auftritt, wird zwischen dem gebrochenen und reflektierten Licht aufgeteilt. Durch die Fresnel-Gleichung wird der Reflexionsanteil ausgerechnet. Dieser Anteil kann anschließend von der Gesamtmenge an Energie abgezogen werden, um damit den Anteil des gebrochenen Lichts zu errechnen (vgl. [9]).

4 Physically Based Rendering Programm

In dem folgenden Kapitel wird auf das Programm, welches in der Präsentation im Fach „*Programmierung von Grafik-Shadern*“ vorgestellt wurde, eingegangen. Dabei sollen vor allem die Unterschiede und Anpassungen an dem Programm von David Wolf, welches als Grundlage für den Programmcode galt, aufgezeigt werden. Die Grundidee des Programmes war eine Sandbox zu erstellen, in welcher die Studierenden die Lichtinteraktion auf verschiedenen Materialien und Objekten beobachten können. Ein Nutzer sollte dabei die Möglichkeit haben sich frei in der Szenerie der Sandbox bewegen und die Parameter der Umgebung verändern zu können.

Für ersteres wurde die Kamera-Klasse (*camera.h*), welche Joey DeFries in seinem Buch „*Learn OpenGL: Learn modern OpenGL graphics programming in a step-by-step fashion*“ [9] beschreibt in das Programm von David Wolf eingebunden. Diese Klasse erlaubt die Navigation in einem Raum mit Maus- und Tastensteuerung. Obwohl die Klasse richtig funktioniert wurden dennoch kleinere Anpassungen gemacht. Zum einen wurde die Zoom Funktion entfernt, da sie nicht unbedingt gebraucht wurde.

Des Weiteren wurde ein Teil der Logik angepasst. Um den Richtungsvektor der Kamera in die Standardausrichtung ($x = 0$, $y = 0$, $z = -1$) zu bringen, verwendet DeFries einen Negativen Offset in der Horizontalen Achse, welche durch den Parameter *yaw* ausgedrückt wird. Dies ist an sich Mathematisch korrekt, erscheint aber bei erster Betrachtung nicht intuitiv, da man ausgehen würde, dass die Ausgangsposition für *yaw* gleich 0 (und nicht -90) sein sollte. Aus diesem Grund wurden die Cosinus- und Sinusterme in der *updateCameraVectors* Funktion angepasst. Die angepasste Gleichung zur Berechnung des Kamera Richtungsvektor sieht nun Folgendermaßen aus.

Listing 1: *Ausschnitt der angepassten updateCameraVectors-Funktion von Joey DeFries*

```
front.x = sin(glm::radians(Yaw)) * cos(glm::radians(Pitch));
front.y = sin(glm::radians(Pitch));
front.z = ((-1) * cos(glm::radians(Yaw))) * cos(glm::radians(Pitch))
    ;
```

Als nächstes wurde die Tastensteuerung der Anwendung implementiert. Diese überprüft per Polling auf vorher Festgelegte Tasten, ob diese derzeit gedrückt sind. Diese Funktion mit dem Namen *processKeypress* wird sukzessiv in While-Schleife der Method *mainLoop* in der Klasse *scenerunner* Aufgerufen.

Listing 2: *Auszug der While-Schleife aus der Klasse Scenerunner*

```
while( ! glfwWindowShouldClose(window) && !glfwGetKey(window,
    GLFW_KEY_ESCAPE) ) {
  keypress = "";
  [...]
  processKeypress(window, keypress);
  scene->update(float(glfwGetTime()), keypress);
  [...]
}
```

Kommt es zu einem Tastendruck dann wird der Variablen *keypress* ein String zugewiesen. Dieser String steht als Repräsentant der jeweiligen Taste oder der Funktion, welche ausgeführt werden soll.

Listing 3: *Methode processKeypress der Scenerunner-Klasse zur Zuweisung von Strings*

```
void processKeypress(GLFWwindow* window, std::string& keypress)
{
    if (glfwGetKey(window, GLFW_KEY_1) == GLFW_PRESS)
        keypress = "gold";
    else if (glfwGetKey(window, GLFW_KEY_2) == GLFW_PRESS)
        keypress = "copper";
    [...]
    else if (glfwGetKey(window, GLFW_KEY_C) == GLFW_PRESS)
        animate = true;
}
```

(Eine gesamte Auflistung aller Funktionen und Eingabemöglichkeiten befindet sich unter Folgendem Link: `https://github.com/MeisterFa/openGLPbrExample`.)

Der Inhalt der Variable wird anschließend über eine weitere Updatemethode an die Klasse *scene* übergeben. Diese besitzt eine Methode *processKeyboardInput*, welche in

Abhängigkeit des Übergebenen Strings anschließend eine Funktion aufruft oder einen Parameter der Szenerie ändert.

Listing 4: *Ausschnitt der processKeyboardInput-Methode der Klasse ScenePbr*

```
void ScenePbr::processKeyboardInput(std::string& keypress, float
    deltaT)
{
  if (keypress == "forward")
    camera.ProcessKeyboard(FORWARD, deltaT);
  else if (keypress == "backward")
    camera.ProcessKeyboard(BACKWARD, deltaT);
  [...]
}
```

Auf eine nahezu homogene Art und Weise funktioniert die Maussteuerung des Programmes. Über Polling wird in der Methode *mainLoop* die derzeitige Mausposition abgefragt. Diese Position, beschrieben durch eine x und y Koordinate, wird anschließend an die *updateMouseMovement*-Methode der Klasse *scene* übergeben.

Listing 5: *Funktionsaufrufe zum empfangen und setzen der Mausposition*

```
glfwGetCursorPos(window, &xpos, &ypos);
scene->updateMouseMovement(xpos, ypos);
```

Beide Koordinaten werden anschließend an die Kameraklasse weitergereicht, welche die Neuberechnung der Kameraorientierung übernimmt. Damit die Kamera Klasse korrekt funktioniert (und der Nutzer sich 360 Grad umsehen kann), muss der Mauszeiger deaktiviert werden. In Folge dessen wurden zwei Tasten mit der Funktion der Deaktivierung und Aktivierung des Mauszeigers belegt. In der Zeit, in welcher der Mauszeiger aktiviert ist, ist das Aktualisieren der Mauszeigerposition nicht notwendig und wird durch folgende Codezeile verhindert:

Listing 6: *If-Abfrage ob derzeitiger Cursor deaktiviert ist*

```
if (glfwGetInputMode(window, GLFW_CURSOR) == GLFW_CURSOR_DISABLED)
```

Durch das Aktivieren und Deaktivieren des Mauszeiger kann es zu kurzzeitigen Sprüngen der Kamera kommen. Dieses Problem und deren Lösung wird von Joey DeFries bereits in seinem Buch „*Learn OpenGL: Learn modern OpenGL graphics programming in a step-by-step fashion*“ [9] vollumfassend erklärt. Die Ursache des Problems liegt dabei, dass die neue x und y Position des Mauszeigers zu stark von der vorhergehenden Mausposition abweicht. In Folge dessen ist der berechnete Offset zu hoch, was schlussendlich in Kamerasprüngen resultiert. DeFries hat zur Lösung des Problems bereits eine Logik implementiert, welche nach jedem Deaktivieren des Mauszeigers aufgerufen werden muss (vgl. [1]).

Wie bereits weiter oben erwähnt wurden auch Anpassungen an der Szenerie von David Wolf gemacht. Dabei wurden zuerst alle Objekte bis auf eins aus der Szene entfernt. Anschließend wurde das eine Objekt in der Ebene zentriert. Durch die weiter oben angesprochene Tastensteuerung können die Eigenschaften, bezüglich der Materialart und des Modelltyps des Objektes verändert werden. Bei den verschiedenen Materialarten stehen Gold, Kupfer, Aluminium, Titan, Silber und ein generisches Nicht-Metall zur Auswahl.

Das generische Material hat zudem die Eigenschaft, dass dessen Rauheit verändert werden kann. Wodurch sich die Wirkung der Mikrofacetten besser betrachten lässt.

Listing 7: *If-Abfrage zur Entscheidung des Objekt-Materials*

```
if (objMaterial == "gold")
{
  // Gold
  drawSpot(glm::vec3(0.0f, 0.0f, 1.5f), metalRough, 1, glm::vec3(1,
      0.71f, 0.29f));
}
[...]
else if (objMaterial == "copper")
{
  // Copper
  drawSpot(glm::vec3(0.0f, 0.0f, 1.5f), metalRough, 1, glm::vec3
      (0.95f, 0.64f, 0.54f));
}
```

Des Weiteren ist es möglich zwischen drei verschiedenen Modellen zu wählen, welche bereits zu Beginn des Programmes geladen werden. Zwei dieser Modelle stammen aus den Projekten von David Wolf. Das dritte Modell wurde auf der Internet Seite: `https://free3d.com/de` heruntergeladen.

Listing 8: *If-Abfrage zur Entscheidung des Objekt-Modells*

```
if (meshNumber == 1){
  model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f,
      1.0f, 0.0f));
  setMatrices();
  mesh1->render();

}
else if (meshNumber == 2){
  ...
}
else if (meshNumber == 3){
  ...
}
```

Zuletzt wurde der Fragment-Shader von David Wolf angepasst. Dieser ist eine Umsetzung der im Kapitel 3 bereits angesprochenen Reflactance Equation. Da die Umsetzung der Formel (15) fast eins zu eins ist, soll im Folgenden nur auf die wichtigen Unterschiede, zwischen dem Shader dieses Programms und dem von David Wolf, eingegangen werden.

$$L_0\left(p,\omega_0\right) = \int_\Omega \left(k_d \cdot \frac{c}{\pi} + k_s \frac{D \cdot G \cdot F}{4 \cdot \left(w_i \cdot n\right)\left(\omega_0 \cdot n\right)}\right) \cdot L_i\left(p, w_i\right) \cdot n \cdot \omega_i \; d\omega_i \tag{15}$$

Im Shader wurde zuerst der *MaterialInfo*-uniform struct, um einen float alpha, erweitert. Dieser repräsentiert die Rauheit α, welche in der Verteilungsfunktion der Normalen D benötigt wird. Es wäre möglich gewesen alpha erst im Shader selbst zu berechnen. Dennoch macht es rein didaktisch Sinn diesen Wert als eigenen Eingabeparameter zu betrachten, da die einzelnen Funktionen somit den realen Formeln noch mehr entsprechen.

Des Weiteren wurden syntaktische und semantische Korrekturen an David Wolfs Shader vorgenommen. Beispielsweise hatte David Wolf die Funktion für die Geometry-SchlickGGX-Approximation fälschlicherweise als GeometrieSmith benannt, siehe Folgenden Codeabschnitt:

Listing 9: *Geometrie-Funktion aus dem Fragment-Shader von David Wolf*

```
float geomSmith( float dotProd ) {
  float k = (Material.Rough + 1.0) * (Material.Rough + 1.0) / 8.0;
  float denom = dotProd * (1 - k) + k;
  return 1.0 / denom;
}
```

Außerdem wurde eine Verbesserung in Bezug auf die Energieerhaltung vorgenommen. Wie im Kapitel 3.4.4 angemerkt lässt sich der Anteil der diffusen Reflexion aus der Teilmenge der spiegelnden Reflexion (Fresnel Gleichung) errechnen. Der diffuse Anteil wurde bis dahin von David Wolf nicht mit in den Shader einbezogen, was zu einer teilweise Überbelichtung der Objekte geführt hat. Der folgende Codeabschnitt zeigt, wie der diffuse Anteil aus der Fresnel-Gleichung errechnet und anschließend in die finale Gleichung eingesetzt wird.

Listing 10: *Quellcode zur Berechnung des Anteiles der diffusen Strahlung*

```
vec3 F = schlickFresnel(hDotV);
vec3 kD = mix((vec3(1.0) - F), vec3(0.0), Material.Metal);
(kD * diffuseBrdf /  PI  + specBrdf) * lightIntensity * nDotL
```

Literatur

[1] Kamera klasse joey defries. `https://learnopengl.com/code_viewer_gh.php?code=includes/learnopengl/camera.h`. [Online; Accessed: 2020-12-03].

[2] Online-tutorium computergrafik. `http://www.bungenstock.de/studienarbeit/lambertapplet/lambert.html`. [Online; Accessed: 2020-12-03].

[3] Tros laserstrahlung teil - allgemeines. `http://regelwerke.vbg.de/vbg_trosla/tros_la0/tros_la0_0_.html`. [Online; Accessed: 2020-12-03].

[4] Adopting a physically based shading model. `https://seblagarde.wordpress.com/2011/08/17/hello-world/`, 2011. [Online; Accessed: 2020-12-03].

[5] Tomas Akenine-Mller, Eric Haines, and Naty Hoffman. *Real-Time Rendering, Fourth Edition.* A. K. Peters, Ltd., USA, 4th edition, 2018.

[6] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-Time Rendering 3rd Edition.* A. K. Peters, Ltd., Natick, MA, USA, 2008.

[7] Jürgen Beyerer, Fernando Puente León, and Christian Frese. *Radiometrie*, pages 179–201. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[8] Brent Burley. Physically-based shading at disney. `https://media.disneyanimation.com/uploads/production/publication_asset/48/asset/s2012_pbs_disney_brdf_notes_v3.pdf`, 2012. [Online; Accessed: 2020-12-03].

[9] Joey de Vries. *Learn OpenGL: Learn modern OpenGL graphics programming in a step-by-step fashion*. Kendall & Welling, June 2020.

[10] Nikolaus Gebhardt. Einige brdf modelle. [Online; Accessed: 2020-12-03].

[11] J.J. Gray. Johann heinrich lambert, mathematician and scientist, 1728 – 1777. *Historia Mathematica*, 5(1):13 – 41, 1978.

[12] Romain Guy and Mathias Agopian. Physically based rendering in filament. `https://google.github.io/filament/Filament.html#overview/physicallybasedrendering`. [Online; Accessed: 2020-12-03].

[13] Koray Hagen. Physically-based rendering: Theory and practice. `https://de.slideshare.net/korayhagen/physically-based-rendering`, April 2015. [Online; Accessed: 2020-12-03].

[14] Naty Hoffman. Background: Physically-based shading. SIGGRAPH '12, 2010.

[15] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, August 1986.

[16] Siegfried Kokoschka. Grundlagen der lichttechnik. `http://www.lti.kit.edu/rd_download/Grundlagen_der_Lichttechnik(1).pdf`. [Online; Accessed: 2020-12-03].

[17] Anthony Maue. Spectacular specular reflections on titan. `https://astrobites.org/2020/10/22/specular-reflections-titan/`, 2020. [Online; Accessed: 2020-12-03].

[18] Wes McDermott. The pbr guide - part 1. `https://academy.substance3d.com/courses/the-pbr-guide-part-1`, 2018. [Online; Accessed: 2020-12-03].

[19] Gordon Müller. Beschleunigung strahlbasierter rendering-algorithmen. Master's thesis, Rheinische Friedrich-Wilhelms-Universitat Bonn, 1997.

[20] Jürgen Nolting and Günter Dittmar. How does it work? – teil 13: Radiometrische grundbegriffe. `https://www.hs-aalen.de/uploads/publication/file/9371/1005_Nolting.pdf`. [Online; Accessed: 2020-12-03].

[21] Alex Ryer, Ultraviolet Light, and Visible Light. Light measurement handbook, 1997.

[22] Ulrich Weidenbacher. *Beleuchtungsmodelle*.

[23] David Wolff. *OpenGL 4 Shading Language Cookbook: Build High-Quality, Real-Time 3D Graphics with OpenGL 4.6, GLSL 4.6 and C++17, 3rd Edition*. Packt Publishing, 2018.