

Friedrich-Schiller-Universität Jena
Physikalisch-Astronomische Fakultät
Theoretisch-Physikalisches Institut

Generierung von Irradiance Maps

Bachelorarbeit zur Erlangung des akademischen Grades Bachelor of Science (B.Sc.)
betreut durch Dr. Carsten Lojewski^a und Prof. Dr. Reinhard Meinel^b

vorgelegt von Markus Pawellek^c

geboren am 7.Mai 1995 in Meiningen
Matrikelnummer: 144645

Erstgutachter: Reinhard Meinel
Zweitgutachter: Carsten Lojewski

Jena, 16.Juni 2017

^aFraunhofer ITWM Kaiserslautern

^bFriedrich-Schiller-Universität Jena

^cE-Mail: markuspawellek@gmail.com

Zusammenfassung

Der Gegenstand dieser Arbeit ist es, eine Datenstruktur zu entwickeln, die die Lichtverteilung eines Modells speichert, um unnötige Berechnungen, die bei der Simulation globaler Beleuchtungseffekte im Regelfall ausgeführt werden müssen, zu eliminieren. Dies erlaubt, das Modell von beliebigen Punkten im Raum aus zu beobachten, ohne dessen gesamte Beleuchtung ständig neu berechnen zu müssen. Dafür nehme ich eine genauere Betrachtung der Irradianzbestimmung vor und konstruiere einen Algorithmus, der die erhaltenen Werte auf der Oberfläche des Modells speichert. Das Verfahren ermöglicht damit unter Verwendung zusätzlichen Speichers und einer gewissen Vorberechnungszeit die simultane Darstellung der Beleuchtungsverteilung ohne störende Rauscheffekte, wie sie für andere Verfahren typisch sind.

Inhaltsverzeichnis

1 Einleitung	1
2 Grundlagen und Methoden	3
2.1 Szenengeometrie	3
2.2 Streuung von Licht an Oberflächen	5
2.3 Beleuchtung und Szene	7
2.4 Raytracing	8
2.5 Radiometrie	10
2.6 Rendergleichung	11
3 Bestimmung der Irradianz	13
3.1 Konstruktion komplexer Materialien	13
3.2 Äquivalenz Lambertscher Strahler und der Irradianz	14
3.3 Schätzung der Irradianz	16
3.4 Fehler der Irradianzschatzung	20
3.5 Adaptive Schätzung der Irradianz	20
4 Aufbau und Generierung der Irradiance Map	27
4.1 Ein Rückblick auf Vertex Lighting	27
4.2 Datenstruktur der Irradiance Map	28
4.3 Irradiance Map Generator	32
5 Fazit und Aussicht	39
Literatur	41
A Verwendete Szenen	i
B Verwendete HDR Maps	iii
C Weitere Ergebnisse der adaptiven Irradianzschatzung	v
D Weitere Irradiance Map Ergebnisse	ix

Abbildungsverzeichnis

1	Einfache Irradiance Maps der »Shaderball«-Szene mit »Uffizi Gallery«-HDR	1
2	Facetten-Muster der »Dragon«-Szene ohne Shading-Normale	4
3	Beispiel des Verlaufs einer Vertex-Shading-Normalen	5
4	Schema einer BSDF	6
5	Wirkungsweise einer Umgebungsbeleuchtung	7
6	Skizze des Sichtbarkeitsproblems und des Raytracing-Verfahrens	8
7	Raytracing als Render-Verfahren	9
8	Skizzenhaftes Beispiel der Strahldichte	10
9	Relation zwischen der ein- und ausfallenden Strahldichte	11
10	Path Tracing am Beispiel der »Audi R8«-Szene mit »Basketball Court«-HDR	12
11	Unterschied direkter und indirekter Beleuchtung anhand der »Shaderball«-Szene . . .	15
12	Wirkung zusammengesetzter Materialien anhand der »Shaderball«-Szene	16
13	Rauschen durch Path Tracing anhand der »Fairy«-Szene	17
14	Vertex Lighting anhand der »Dragon«-Szene	18
15	Vertex Lighting anhand der »Shaderball«-Szene	19
16	Relativer Fehler der »Dragon«-Szene	21
17	Relativer Fehler der »Shaderball«-Szene	21
18	Vertex Lighting anhand der »Dragon«-Szene mit »Ennis-Brown House«-HDR	23
19	Vertex Lighting anhand der »Shaderball«-Szene mit »Ennis-Brown House«-HDR . .	23
20	Vertex Lighting anhand der »Shaderball«-Szene mit »Ennis-Brown House«-HDR . .	23
21	Vertex Lighting anhand der »Audi R8«-Szene mit »Sky 20«-HDR	24
22	Fehlerursachen des Vertex Lighting	27
23	Irradiance Map der »Cornell Box«-Szene mit »Sky 10«-HDR	28
24	Schema der Irradiance Map Datenstruktur	28
25	Schema der Speichenindex-Funktion der Irradiance Map	29
26	Schema der linear approximierenden Irradiance Map Fortsetzung	31
27	Irradiance Map der »Shaderball«-Szene mit der »Ennis-Brown House«-HDR	36
28	Irradiance Map der »Shaderball«-Szene mit der »Ennis-Brown House«-HDR	36
29	Irradiance Map der »Shaderball«-Szene mit der »Ennis-Brown House«-HDR	37
30	Irradiance Map der »Shaderball«-Szene mit der »Ennis-Brown House«-HDR	38
31	Vertex Lighting anhand der »Dragon«-Szene	v
32	Vertex Lighting anhand der »Shaderball«-Szene mit »Sky 20«-HDR	v
33	Vertex Lighting anhand der »Shaderball«-Szene mit »Sky 10«-HDR	vi
34	Vertex Lighting anhand der »Audi R8«-Szene mit »Sky 20«-HDR	vii
35	Vertex Lighting anhand der »Audi R8«-Szene mit »Sky 20«-HDR	viii
36	Irradiance-Map der »Shaderball«-Szene mit direktonaler Lichtquelle	ix
37	Irradiance Map Fehler der »Shaderball«-Szene mit direktonaler Lichtquelle	ix

Symboltabelle

Symbol	Definition
$\exists! \dots : \dots$	Es existiert genau ein \dots , sodass \dots gilt.
$x \in A$	x ist ein Element der Menge A .
$A \subset B$	A ist eine Teilmenge von B .
$A \cap B$	$\{x \in A \mid x \in B\}$ für Mengen A, B — Mengenschnitt
$A \cup B$	$\{x \mid x \in A \text{ oder } x \in B\}$ für Mengen A, B — Mengenvereinigung
$A \setminus B$	$\{x \in A \mid x \notin B\}$ für Mengen A, B — Differenzmenge
$A \times B$	$\{(x, y) \mid x \in A, y \in B\}$ für Mengen A und B — kartesisches Produkt
\emptyset	leere Menge
$f: X \rightarrow Y$	Abbildung f mit Definitionsmenge X und Wertebereich Y
$f(\cdot, y)$	Für Abbildung $f: X \times Y \rightarrow Z$ mit $y \in Y$ gilt $f(\cdot, y)(x) = f(x, y)$ für alle $x \in X$
$\text{im } f$	$\{f(x) \mid x \in X\} \subset Y$ mit der Abbildung $f: X \rightarrow Y$ und Mengen X, Y
$f(A)$	$\{f(x) \mid x \in A\}$ mit Abbildung $f: X \rightarrow Y$ und Menge $A \subset X$
$f^{-1}(A)$	$\{x \in X \mid f(x) \in A\}$ mit Abbildung $f: X \rightarrow Y$ und Menge $A \subset Y$
f^{-1}	Inverse einer bijektiven Abbildung $f: X \rightarrow Y$
$f _A$	Einschränkung der Abbildung $f: X \rightarrow Y$ auf die Menge $A \subset X$
\mathbb{N}	$\{1, 2, 3, \dots\}$ — Menge der natürlichen Zahlen
\mathbb{N}_0	$\{0, 1, 2, \dots\}$ — Menge der natürlichen Zahlen mit der Null
\mathbb{R}	Menge der reellen Zahlen
∞	$-\infty < x < \infty$ für alle $x \in \mathbb{R}$ — Unendlich
$\overline{\mathbb{R}}$	$\mathbb{R} \cup \{-\infty, \infty\}$
(a, b)	$\{x \in \mathbb{R} \mid a < x < b\}$ mit $a, b \in \overline{\mathbb{R}}$ — offenes Intervall
$[a, b]$	$\{x \in \mathbb{R} \mid a \leq x \leq b\}$ mit $a, b \in \overline{\mathbb{R}}$ — geschlossenes Intervall
$[a, b)$	$\{x \in \mathbb{R} \mid a \leq x < b\}$ mit $a, b \in \overline{\mathbb{R}}$
$(a, b]$	$\{x \in \mathbb{R} \mid a < x \leq b\}$ mit $a, b \in \overline{\mathbb{R}}$
$\sup A$	Supremum der Menge $A \subset \mathbb{R}$
$\min A$	Minimum der Menge $A \subset \mathbb{R}$
$\text{sgn } x$	Vorzeichen von $x \in \mathbb{R}$
$ x $	Betrag von $x \in \mathbb{R}$
$\mathbb{1}_A(x)$	charakteristische Funktion mit Menge $A \subset X$ und Wert $x \in X$ für Menge X
$\lfloor x \rfloor$	Abrundungsfunktion für ein $x \in \mathbb{R}$
$\lceil x \rceil$	Aufrundungsfunktion für ein $x \in \mathbb{R}$
$\langle x, y \rangle$	$\sum_{i=1}^n x_i y_i$ mit $x, y \in \mathbb{R}^n$ für $n \in \mathbb{N}$ — Skalarprodukt
$\ x\ $	$\sqrt{\langle x, x \rangle}$ mit $x \in \mathbb{R}^n$ für $n \in \mathbb{N}$ — Norm
$x \times y$	Kreuzprodukt der Vektoren $x, y \in \mathbb{R}^3$
$x_n \xrightarrow{n \rightarrow \infty} x$	Die Folge $(x_n)_{n \in \mathbb{N}}$ in \mathbb{R} konvergiert gegen $x \in \mathbb{R}$.
\mathcal{S}^2	$\{x \in \mathbb{R}^3 \mid \ x\ = 1\}$ — Sphäre der Richtungen
\mathcal{H}_μ^2	$\{x \in \mathcal{S}^2 \mid \langle \mu, x \rangle \geq 0\}$ mit $\mu \in \mathcal{S}^2$ — Hemisphäre der Richtungen
λ	Lebesgue-Maß
σ	Oberflächen-Maß einer Untermannigfaltigkeit in \mathbb{R}^n für $n \in \mathbb{N}$

ABBILDUNGSVERZEICHNIS

Symbol	Definition
$\int_X f \, d\mu$	Integral über Funktion f bezüglich Maßraum (X, \mathcal{A}, μ)
$p \otimes q$	Produktmaß der Maße p, q
$\mathcal{X} \otimes \mathcal{Y}$	Produkt- σ -Algebra der σ -Algebren \mathcal{X}, \mathcal{Y}
δ_x	Diracsche Delta-Distribution am Punkt $x \in X$ der Menge X
$\mathbb{E} X$	$\int_{\Omega} X \, dP$ — Erwartungswert der Zufallsvariable X auf (Ω, \mathcal{A}, P)
$\text{var } X$	$\mathbb{E} (X - \mathbb{E} X)^2$ — Varianz der Zufallsvariable X auf (Ω, \mathcal{A}, P)
$\Theta(f), \Theta_n(f(n))$	Landau-Symbol der asymptotisch scharfen Schranke für Abbildung $f: \mathbb{N} \rightarrow \mathbb{R}$
π	3.141592 ... — Kreiszahl
1 MiB	$2^{10} \text{ kiB} = 2^{20} \text{ B} = 2^{20} \text{ Byte}$ — Speichereinheit
1 s	Eine Sekunde
1 h	3600 s — Eine Stunde
1 %	0.01 — Ein Prozent

1 Einleitung

In der 3D-Computergrafik ist für die Erzeugung von realistischen Bildern die Simulation globaler Beleuchtungseffekte notwendig [34]. Diese Lichteffekte ergeben sich formal als Lösung der »Rendergleichung« [14]. Seit der Einführung dieser Gleichung im Jahre 1986 wurden verschiedene Algorithmen entwickelt, welche diese für beliebige Szenen numerisch lösen. Herauskristallisiert hat sich vor allem das »Path Tracing« [14, 34]. Dieses Verfahren kann die reale Beleuchtung beliebig genau und erwartungstreu schätzen. Aus diesem Grund werden die durch Path Tracing generierten Bilder meist als Referenzbild für die Bilder anderer Algorithmen verwendet, um deren Qualität zu untersuchen.

Um nun verschiedene Materialien von Objekten simulieren zu können, werden häufig verschiedene Arten von Lichtstreuung an Oberflächen betrachtet. Eine der wichtigsten Arten ist gerade die ideale diffuse Streuung, welche der Szene ein grundlegendes plastisches Aussehen gibt. Sie ist unabhängig von der Richtung des einfallenden Lichtstrahls und damit auch invariant unter Änderung des Beobachtungspunktes [44]. Für Path Tracing bedeutet dies, dass für jede Änderung der Kamera die eigentlich konstante Lichtverteilung neu berechnet werden muss. Diese Evaluierung nimmt aber auch den größten Rechenaufwand in Anspruch, da für jeden dieser Punkte Licht aus dessen gesamten Hemisphäre eingesammelt werden muss. Um also das Verfahren des Path Tracings zu optimieren, müssten die diffusen Lichtverhältnisse vorberechnet und auf der Oberfläche der Szene gespeichert werden [43].

Besonders in der Computerspielindustrie wird dieses Problem mithilfe von sogenannten »Lightmaps« gelöst, welche für viele Punkte der Szene deren »Irradianz« in einer Textur speichern [19]. Während des Renderings werden diese Werte dann zusammen mit der Farbe der Materialtextur ausgelernt, miteinander multipliziert und dargestellt. Die Generierung einer solchen Lightmap ist jedoch mit diversen Tücken verbunden, welche in vielen Fällen nur durch manuelle Optimierung beseitigt werden können.

Aus diesem Grund führe ich im Laufe dieser Arbeit die sogenannten »Irradiance Maps« ein, die die Irradianz gegebener Punkte auf der Oberfläche der zugehörigen Szene speichern. Die benötigte Datenstruktur soll dabei speziell für die Verwendung von »Raytracing« und Path Tracing ausgelegt sein. Ich werde eine genauere Betrachtung der Irradianzbestimmung vornehmen, um so deren Prozess zu vereinfachen. Des Weiteren präsentiere ich einen Algorithmus zur adaptiven und automatischen Generierung einer solchen Irradiance Map.

Abbildung 1 (Quelle aller Abbildungen: Markus Pawellek 2017) zeigt ein Beispiel, in welchem deutlich wird, dass Irradiance Maps in der Lage sind die diffusen Lichtverhältnisse sehr gut zu approximieren, jedoch bei schlechter Generierung viel Speicher benötigen (siehe 1b). Außerdem ist in 1c erkennbar, dass für einen Großteil der Szene eine vergleichsweise kleine Auflösung der Irradiance Map ausreicht. Es wird also auch darum gehen, diesen Sachverhalt bei der Konstruktion auszunutzen.

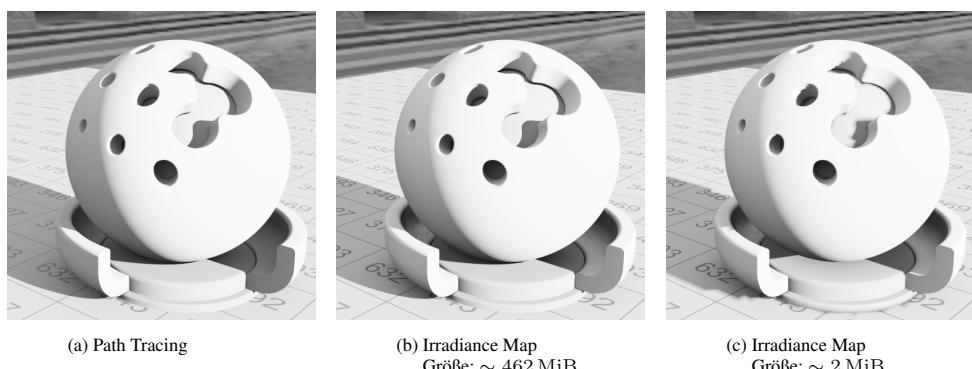


Abbildung 1: Die Bilder zeigen alle denselben Ausschnitt der »Shaderball«-Szene mit unterschiedlichen Shading-Verfahren. Die Lichtquellen bestehen aus einer direktionalen Lichtquelle und der »Uffizi Gallery«-HDR.

2 Grundlagen und Methoden

In den folgenden Kapiteln wird eine grundlegende Einführung und Definition der Verfahren gegeben, die für den weiteren Verlauf dieser Arbeit von Belang sind. Viele dieser Themen können hier nur angezissen werden, da ihre komplette Behandlung den Rahmen und das Ziel des Themas sprengen würden. Für eine genauere Einführung in die einzelnen Themengebiete, wird dem Leser geraten, sich mit den genannten Quellen auseinander zu setzen.

2.1 Szenengeometrie

Um in einem Computer ein zweidimensionales Bild einer dreidimensionalen Umwelt oder auch »Szenen« (engl.: *scene*) zu generieren, benötigen wir ein mathematisches Modell, welches diese hinreichend gut beschreibt. Wir wollen uns einen Beobachter vorstellen, der die Umwelt und die Objekte in ihr von einem bestimmten Punkt im Raum aus betrachtet. Für viele Algorithmen, die diese Aufgabe lösen, ist dabei vor allem das Verhalten von Licht auf den Oberflächen dieser Objekte wichtig [14, 34, 38]. Der Einfachheit halber wollen wir davon ausgehen, dass Licht nur in den oberen Schichten eines Körpers mit dessen Material wechselwirkt. Diese Annahme reduziert die Komplexität des mathematischen Modells, die dann noch aus der Charakterisierung der Oberflächen besteht.

Die Oberflächen realer physikalischer Körper sind im Allgemeinen beliebig geformt und können nicht in geschlossener Form durch eine Gleichung beschrieben werden. Dennoch lassen sie sich im analytischen Sinne durch einfache Hyperflächen (engl.: *shape* [34, S. 123 ff]) im Raum approximieren [16]. Für die Bildgenerierung wählt man für solch eine Fläche meist ein Dreieck. Es ist einfach zu beschreiben und flexibel genug, um die meisten Objekte beliebig genau zu approximieren [3, 25]. Dabei wollen wir entartete Dreiecke, die nur aus einem Punkt oder einer Strecke bestehen, ausschließen, da sie für die betrachteten Render-Verfahren nicht darstellbar sind.

DEFINITION 2.1: (Dreieck)

Ein Dreieck Δ wird durch eine Sequenz (A, B, C) von Eckpunkten in \mathbb{R}^3 , für die die Menge $\{B - A, C - A\}$ linear unabhängig ist, charakterisiert. Seien weiterhin

$$M := \{(u, v) \in [0, 1]^2 \mid u + v \leq 1\}$$

$$\varphi: M \rightarrow \mathbb{R}^3, \quad \varphi(u, v) := (1 - u - v)A + uB + vC$$

Dann ist die Menge der Punkte S des Dreiecks gegeben durch im φ und (M, φ) stellt deren Standardparametrisierung dar. Der Notation wegen, identifizieren wir Δ mit S und definieren für alle $(u, v) \in M$

$$\Delta(u, v) := \varphi(u, v)$$

Die baryzentrischen Koordinaten (u, v, w) eines Punktes $x \in \Delta$ sind weiterhin durch die folgenden Eigenschaften gegeben.

$$(u, v) \in M, \quad w = 1 - u - v, \quad \Delta(u, v) = x$$

Die analytische äußere Normale oder auch Normale ist definiert durch

$$\mu := \frac{(B - A) \times (C - A)}{\|(B - A) \times (C - A)\|}$$

Jedes Dreieck besitzt auf seiner gesamten Fläche eine eindeutige konstante äußere analytische Normale. Für die Simulation von globalen Beleuchtungseffekten ist diese Eigenschaft jedoch ein Nachteil,

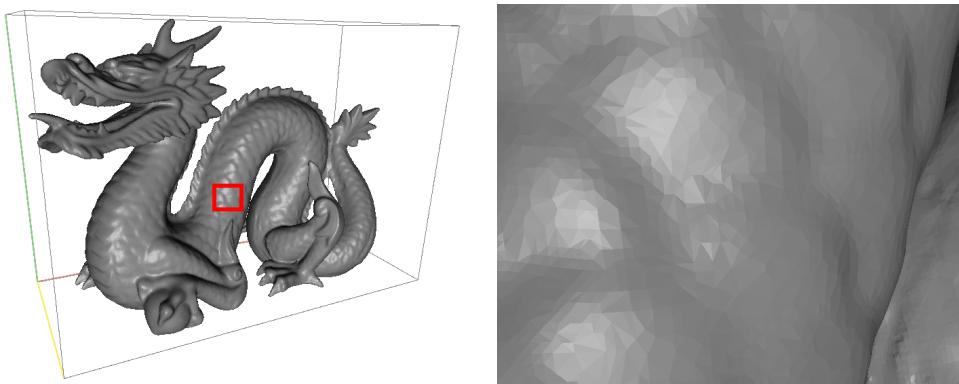


Abbildung 2: Die Bilder zeigen die gerenderte »Dragon«-Szene. Das rechte Bild entspricht dem roten Bereich des Linken. In dieser Szene werden die Normalen des Objektmodells durch die analytischen Normalen der Dreiecke angenähert. Da das Modell sehr fein trianguliert ist, fällt dies im ersten Bild nicht auf. Zoomt man jedoch mit der Kamera heran, werden die Fehler durch die Approximation deutlich und die einzelnen Dreiecke sind mit dem menschlichen Auge auszumachen.

weil die Beleuchtung eines Objektes stark vom Verlauf seiner Normalen abhängt. Nähern wir ein Objekt durch $n \in \mathbb{N}$ Dreiecke an, so nähern wir auch den Normalenverlauf des Objektes durch die stückweise konstanten Normalen der Dreiecke an. Der Fehler dieser Approximation tritt in Form eines facettenartigen Musters auf, welches für das menschliche Auge gut erkennbar ist [34, S. 166]. In Abbildung 2 wird dieser Effekt genauer an einem Beispiel demonstriert. Folglich muss darauf geachtet werden, dass bei stetig differenzierbaren Oberflächen die Stetigkeit der Normalen erhalten bleibt [16, S. 39 ff].

DEFINITION 2.2: (Normalen-Funktion / Shading-Normale)

Sei \triangle ein Dreieck mit der Normalen μ . Dann wird $\nu: \triangle \rightarrow \mathcal{H}_\mu^2$ als Normalen-Funktion oder auch als Shading-Normale von \triangle bezeichnet.

Nach den Quellen [34, S. 166, 584 ff] und [1, S. 38 ff, 183 ff] sind die typischen Verfahren für eine genauere Interpolation die »Vertex-Shading-Normale« und das »Bump Mapping« (engl.: *bump mapping*). Formal gesehen, werden bei beiden Verfahren die Normalen der vorhandenen Geometrie durch eine Normalen-Funktion ersetzt.

DEFINITION 2.3: (Vertex-Shading-Normale)

Seien \triangle ein Dreieck mit der Normalen μ und $\mu_A, \mu_B, \mu_C \in \mathcal{H}_\mu^2$ Normalen an den Eckpunkten des Dreiecks. Sei weiterhin ν eine Shading-Normale auf \triangle , sodass für alle $x \in \triangle$ mit den baryzentrischen Koordinaten (u, v, w) gilt

$$\nu(x) := \frac{w\mu_A + u\mu_B + v\mu_C}{\|w\mu_A + u\mu_B + v\mu_C\|}$$

Dann ist ν stetig und man nennt es eine Vertex-Shading-Normale von \triangle .

Durch das Setzen der Normalen an den Eckpunkten (engl.: *vertex*) eines Dreiecks können wir sicher gehen, dass der Verlauf der Normalen stetig von einem Dreieck zu einem anderen übergeht. Ein beispielhafter Verlauf einer Vertex-Shading-Normalen wird in Abbildung 3 gezeigt. Zu beachten ist, dass die Vektoren der Shading-Normale im Allgemeinen nicht mehr senkrecht auf der Geometrie stehen. Dadurch kann es, wie in [34, S. 574 ff] und [38, S. 150 ff] gezeigt, zu Artefakten bei der Generierung des Bildes kommen.

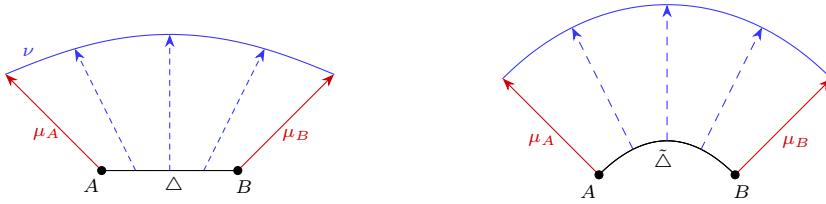


Abbildung 3: Die erste Skizze auf der linken Seite zeigt den Verlauf einer Vertex-Shading-Normalen ν anhand eines Beispiels. A und B sind dabei zwei Eckpunkte eines Dreiecks Δ . μ_A und μ_B sind die jeweilig gegebenen Vertex-Normalen an den Eckpunkten. Im rechten Bereich der Abbildung ist die durch ν approximierte gekrümmte Fläche $\tilde{\Delta}$, auf der die Shading-Normale ν senkrecht steht, eingezeichnet.

Der letzte Schritt zur vollständigen Beschreibung der Szenengeometrie besteht darin, eine Menge von Dreiecken zu bilden, sodass diese ein physikalisches Objekt gut genug approximieren. In [34] und [3] werden solche Mengen auch »Mesh« (engl.: *triangle mesh*) genannt.

DEFINITION 2.4: (Mesh)

Seien $n \in \mathbb{N}$ und Dreiecke Δ_i mit Shading-Normalen ν_i für alle $i \in \mathbb{N}, i \leq n$. Weiterhin definieren wir

$$\mathcal{T} := \bigcup_{i=1}^n \Delta_i \quad \mathcal{A} := \{x \in \mathcal{T} \mid \exists! i \in \mathbb{N}, i \leq n: x \in \Delta_i\}$$

$$\nu: \mathcal{T} \rightarrow \mathcal{S}^2 \cup \{0\}, \quad \nu(x) := \sum_{i=1}^n \nu_i(x) \mathbb{1}_{\mathcal{A} \cap \Delta_i}(x)$$

Dann nennen wir \mathcal{T} eine Mesh mit der Shading-Normalen ν , wenn für σ -fast-alles $x \in \mathcal{T}$ auch $x \in \mathcal{A}$ gilt.

Die Definition der Mesh verbietet, dass die Schnittpunkte der Dreiecke eine eigene Fläche im Raum bilden. So können wir sicher gehen, dass die Integration über die Punkte der Mesh eine eindeutige Lösung ergibt. Es ist jedoch erlaubt, dass sich Dreiecke in einem Punkt oder einer Strecke schneiden dürfen. Dies ist auch nötig, um komplexere Objekte formen zu können. In der Praxis ist die genannte Bedingung im Allgemeinen erfüllt und folglich auch keine fundamentale Einschränkung [3, 25, 34, 38].

2.2 Streuung von Licht an Oberflächen

Im letzten Abschnitt wurden die Grundbausteine einer Szene eingeführt, die deren Geometrie beschreiben. Wie bereits erwähnt, benötigen wir jedoch zusätzlich die Informationen, auf welche Art und Weise Licht mit den Oberflächen der Objekte interagiert. Dieses Verhalten wird durch die sogenannte »Bidirektionale Streuungsverteilungsfunktion« (engl.: *bidirectional scattering distribution function*, BSDF), die sich aus einer »Bidirektionalen Reflektanzverteilungsfunktion« (engl.: *bidirectional reflectance distribution function*, BRDF) und einer »Bidirektionalen Transmissionsverteilungsfunktion« (engl.: *bidirectional transmittance distribution function*, BTDF) zusammensetzt, erklärt. Strukturierte Einführungen zu diesem Thema erhält man in den Quellen [1, 5, 28, 34, 38].

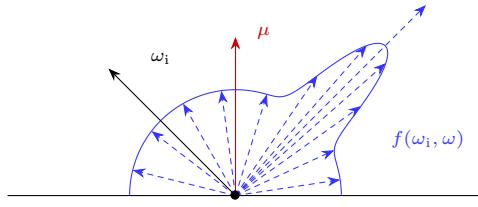


Abbildung 4: Die Abbildung zeigt die Verteilung des reflektierten Lichtes für alle $\omega \in \mathcal{H}_\mu^2$ bezüglich einer konstanten Einfallsrichtung $\omega_i \in \mathcal{H}_\mu^2$ einer BSDF f bezüglich der Oberflächennormalen $\mu \in \mathbb{S}^2$.

DEFINITION 2.5:

Sei $\mu \in \mathbb{S}^2$ die Normale am Punkt einer Oberfläche. Dann ist eine BSDF bezüglich μ gegeben durch eine integrierbare Abbildung $f: \mathbb{S}^2 \times \mathbb{S}^2 \rightarrow [0, \infty)$ mit den folgenden Eigenschaften.

(i) Für σ^2 -fast-alle (ω_i, ω_o) mit $\omega_i \in \mathbb{S}^2$, $\omega_o \in \mathcal{H}_\nu^2$ und $\nu := \text{sgn}(\langle \mu, \omega_i \rangle) \cdot \mu$ gilt

$$f(\omega_i, \omega_o) = f(\omega_o, \omega_i) \quad (\text{Helmholtz-Reziprozität})$$

(ii) Für σ -fast-alle $\omega_i \in \mathbb{S}^2$ gilt

$$\int_{\mathbb{S}^2} f(\omega_i, \omega) |\langle \mu, \omega \rangle| d\sigma(\omega) \leq 1 \quad (\text{Energieerhaltung})$$

Eine BSDF f gibt an, welcher Anteil des einfallenden Lichtes aus der Richtung $\omega_i \in \mathbb{S}^2$ in die Richtung $\omega_o \in \mathbb{S}^2$ gestreut wird. Gilt dabei $\omega_o \in \mathcal{H}_\nu^2$, so befindet sich ω_o in der gleichen Hemisphäre bezüglich μ wie ω_i und es handelt sich um eine Reflexion des Lichtes. Der verbleibende Fall beschreibt die Transmission. Die BSDF stellt damit eine Verallgemeinerung der idealen Reflexion und Brechung an Oberflächen dar. Die Quellen [34, S. 571 ff] und [38, S. 135 ff] weisen zudem nach, dass die Helmholtz-Reziprozität nur für den reflektierten Anteil des Lichtes gilt. In Abbildung 4 ist das Beispiel einer typischen BSDF gezeigt, die keine Transmission des Lichtes zulässt [34, S. 509 ff].

Die hier eingeführten Funktionen für die Charakterisierung von Materialien können auf verschiedene Weisen verallgemeinert werden. Zu beachten ist vor allem die fehlende Abhängigkeit von der Wellenlänge des einfallenden und des gestreuten Lichtes, durch welche Effekte wie Dispersion, Irisieren und Lumineszenz verhindert werden. Ein weiteres physikalisches Phänomen stellt die Volumenstreuung dar. Bei der Betrachtung dieser wird ein Material durch die sogenannte »BSSRDF« beschrieben [34, S. 671 ff].

BSDFs sind im Allgemeinen nicht in geschlossener Form formulierbar [34, S. 507 ff]. Aus diesem Grund wollen wir für die Konstruktion realistischer Materialien, wie bei der Approximation von Oberflächen, verschiedene einfache BSDFs zu Grunde legen. In den beiden folgenden Beispielen handelt es sich um die BSDFs f bezüglich der Normalen $\mu \in \mathbb{S}^2$ für die lambertsche diffuse Reflexion (engl.: *lambertian reflection*, [34, S. 531 ff]) und für die ideale Reflexion (engl.: *specular reflection*, [38, S. 144 ff]). Weitere BSDF-Modelle sind in [5, 34, 38] zu finden. Es seien $\omega_i, \omega_o \in \mathbb{S}^2$ mit $\nu := \text{sgn}(\langle \mu, \omega_i \rangle) \cdot \mu$ gegeben.

$$f(\omega_i, \omega_o) = \frac{1}{\pi} \mathbf{1}_{\mathcal{H}_\nu^2}(\omega_o) \quad (\text{lambertsch diffuse Reflexion})$$

$$f(\omega_i, \omega_o) = \frac{\delta_{\omega_o}(2 \langle \mu, \omega_i \rangle \mu - \omega_i)}{|\langle \mu, \omega_i \rangle|} \quad (\text{ideale Reflexion})$$

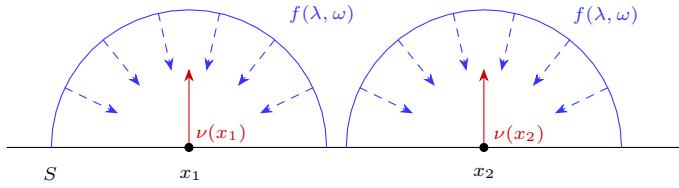


Abbildung 5: Die Darstellung zeigt, dass die Beleuchtung zweier Punkte x_1 und x_2 einer Oberfläche S mit den Normalen $\nu(x_1)$ und $\nu(x_2)$ durch eine Umgebungsbeleuchtung f nur abhängig von der Verdeckung der Punkte und nicht von deren Position ist. Bei x_1 und x_2 sind in diesem Falle die gleichen Lichtintensitäten zu messen. Es ist $\lambda \in (0, \infty)$ die Wellenlänge des Lichtes und $\omega \in \mathbb{S}^2$ der Raumwinkel.

2.3 Beleuchtung und Szene

Die BSDF an einem Punkt beschreibt lediglich die Streuung des einfallenden Lichtes. Um den Render-Verfahren einen Sinn zu geben, benötigen wir demnach Lichtquellen. In den Quellen [34, S. 707 ff] und [13] werden verschiedene Arten von Lichtquellen definiert, implementiert und gesamplet. Der einfachste Weg Lichtquellen einzuführen, besteht in einer abstrakten Formulierung, die unabhängig von der Szenengeometrie agiert. Wir wollen eine sogenannte »Umgebungsbeleuchtung« (engl.: *HDR environment map* oder *infinite area light*, [34, S. 737 ff]) einführen.

DEFINITION 2.6: (Umgebungsbeleuchtung)

Sei $f: (0, \infty) \times \mathbb{S}^2 \rightarrow [0, \infty)$ eine integrierbare Abbildung. Dann wird f eine Umgebungsbeleuchtung genannt.

Diese Funktion beschreibt das Licht verschiedener Wellenlängen, welches von einer Kugeloberfläche mit quasi unendlich großem Radius in die Szene ausgesandt wird. Für einen Punkt der Szene ist dessen Beleuchtung durch diese Funktion also unabhängig von dessen Position. Diese Tatsache wird in Abbildung 5 wieder an einem Beispiel gezeigt.

Aber auch die Materialien der Szene sollten in der Lage sein Licht auszusenden. Wir führen dementsprechend eine Abbildung ein, die die benötigten Eigenschaften erfüllt. Häufig werden diese Lichtquellen auch »Feldlichtquellen« (engl.: *area light*) genannt [34, S. 733 ff].

DEFINITION 2.7: (Emission)

Sei \mathcal{T} eine Mesh mit einer Shading-Normalen ν . Dann ist eine Emission von \mathcal{T} durch eine integrierbare Abbildung $E: \mathcal{T} \times (0, \infty) \times \mathbb{S}^2 \rightarrow [0, \infty)$ gegeben.

Wie bei der vorherigen Definition beschreibt diese Funktion das ausgesandte Licht in Abhängigkeit der Wellenlänge und des Raumwinkels. Der Unterschied besteht darin, dass sie auf der Oberfläche einer Mesh variieren kann und sich damit auch die Beleuchtung eines Punktes je nach Position verändert.

Durch die Einführung der Lichtquellen erhalten wir nun die vollständige Beschreibung einer Szene durch Zusammenführung der bereits definierten Strukturen.

DEFINITION 2.8: (Szene)

Eine Szene ist ein Tupel $(\mathcal{T}, \nu, f, E, U)$, bestehend aus einer Mesh \mathcal{T} mit einer Shading-Normalen ν , einer integrierbaren Abbildung $f: \mathcal{T} \times \mathbb{S}^2 \times \mathbb{S}^2 \rightarrow [0, \infty)$, wobei $f(x, \cdot, \cdot)$ für σ -fast-alle $x \in \mathcal{T}$ ein BSDF bezüglich $\nu(x)$ darstellt, einer Emission E von \mathcal{T} und einer Umgebungsbeleuchtung U .

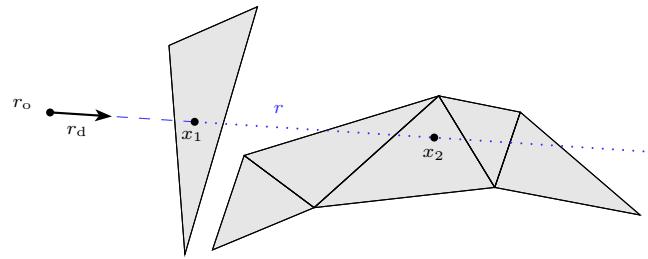


Abbildung 6: Die Abbildung zeigt eine Skizze, welche das Sichtbarkeitsproblem und den Raytracing-Algorithmus verdeutlicht. Die grau melierten Dreiecke sollen die gegebene Mesh \mathcal{T} darstellen. Der ausgesendete Strahl r , gegeben durch (r_o, r_d) , trifft in der Mesh genau zwei Punkte x_1 und x_2 . Dabei wird x_2 durch x_1 verdeckt. Es gilt also $V_{\mathcal{T}}(r_o, x_1) = 1$ und $V_{\mathcal{T}}(r_o, x_2) = 0$.

2.4 Raytracing

Im einfachsten Falle bezeichnet das Wort »Raytracing« (engl.: *ray tracing*) einen Algorithmus zur Ermittlung der Sichtbarkeit von dreidimensionalen Objekten bezüglich eines Ursprungspunktes (engl.: *origin*) im Raum [32, 34]. Häufig versteht man darunter jedoch auch eine Render-Technik für die Generierung eines gesamten Bildes aus einer gegebenen Szene, die auf dem eben genannten Raytracing-Algorithmus basiert [29, 32, 34].

Grundsätzlich gibt es viele Verfahren, um ein Szene auf ein Bild zu rendern [1, 6]. Die Erfahrung zeigt aber, dass vor allem in Bereichen, in denen die globalen Beleuchtungseffekte realistisch simuliert werden sollen, Raytracing eine wichtige Grundlage darstellt. Der Grund dafür besteht in der Tatsache, dass Raytracing das »Sichtbarkeitsproblem« (engl.: *visibility problem*, [6, 8]) löst und durch die Verwendung von Strahlen eine Basis für die Lichtberechnung im Sinne der geometrischen Optik bereitstellt [32, 34, 38].

DEFINITION 2.9: (Sichtbarkeitsproblem)

Sei \mathcal{T} eine Mesh. Dann ist die Sichtbarkeitsfunktion von \mathcal{T} die folgende Abbildung.

$$V_{\mathcal{T}}: \mathbb{R}^3 \times \mathcal{T} \rightarrow \{0, 1\}$$

$$V_{\mathcal{T}}(o, x) = \begin{cases} 1 & : \mathcal{T} \cap \{(1 - \gamma)o + \gamma x \mid \gamma \in (0, 1)\} = \emptyset \\ 0 & : \text{sonst} \end{cases}$$

Das Sichtbarkeitsproblem beschreibt die Aufgabe diese Funktion für gegebene Parameter zu evaluieren.

Die Sichtbarkeitsfunktion gibt an, ob der Oberflächenpunkt x vom Beobachtungspunkt o aus in gerader Linie gesehen werden kann oder ob zwischen diesen Punkten ein weiterer Punkt der Mesh \mathcal{T} den Punkt x verdeckt [8, S. 30]. Daran anknüpfend besteht die Basis des Raytracing-Verfahrens auf dem Aussenden von »Strahlen« (engl.: *ray*) bezüglich eines Ursprungspunktes [29, 32, 34]. Für diese Strahlen kann der Schnittpunkt mit \mathcal{T} ermittelt werden. Liegt der Schnittpunkt zwischen o und x , so beträgt der Wert der Sichtbarkeitsfunktion 0. Im noch bleibenden Fall ergibt sich der Wert zu 1. Die Sichtbarkeitsfunktion kann somit für alle gegebenen Parameter durch Raytracing berechnet werden. Abbildung 6 zeigt diese Methode anhand einer Skizze.

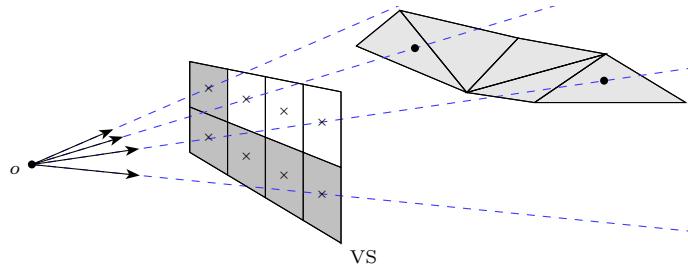


Abbildung 7: In der Skizze ist ein typisches Render-Verfahren auf der Basis des Raytracing-Algorithmus dargestellt. Bezuglich eines Beobachtungspunktes $o \in \mathbb{R}^3$ wird durch jeden Pixel eines virtuellen Bildschirms VS ein Strahl geschossen und der Wert der Raytracing-Funktion evaluiert. Die grau melierten Dreiecke bilden die Mesh \mathcal{T} . Je nachdem, ob ein Strahl einen Schnittpunkt mit \mathcal{T} aufweist, wird ein entsprechendes Shading-Verfahren ausgeführt. Der Übersicht wegen sind im Bild nur vier der eigentlich acht Strahlen sichtbar.

DEFINITION 2.10: (Strahl)

Seien ein Ursprung $o \in \mathbb{R}^3$, eine Richtung $d \in \mathcal{S}^2$ und die folgende Abbildung gegeben.

$$\varphi: [0, \infty) \rightarrow \mathbb{R}^3, \quad \varphi(\gamma) := o + \gamma d$$

Dann wird das Bild im φ ein Strahl r genannt. Dabei ist $([0, \infty), \varphi)$ die Standardparametrisierung von r . Der Notation wegen, definieren wir für alle $\gamma \in [0, \infty)$

$$r(\gamma) := \varphi(\gamma)$$

Durch das Tupel (o, d) charakterisieren und identifizieren wir den Strahl r . Die Menge aller Strahlen definieren wir durch \mathcal{R} .

Die vollständige Evaluierung von $V_{\mathcal{T}}$ ist jedoch häufig nicht notwendig. In Abbildung 6 ist klar, dass alle Punkte von r , die hinter x_1 liegen, vom Beobachtungspunkt r_o aus nicht sichtbar sind. Beim eigentlichen Raytracing-Algorithmus wirkt sich diese Eigenschaft positiv aus. Es reicht, für eine gegebene Richtung und einen gegebenen Beobachtungspunkt den nächsten Schnittpunkt des resultierenden Strahles zu ermitteln. Für alle weiteren Schnittpunkte ergibt sich, sofern diese existieren, die Sichtbarkeitsfunktion zu Null, wodurch diese Punkte im weiteren Vorgehen ignoriert werden können [34, S. 4 ff, 866].

DEFINITION 2.11: (Raytracing-Funktion)

Sei \mathcal{T} eine Mesh. Dann ist die Raytracing-Funktion von \mathcal{T} durch die folgende Abbildung gegeben.

$$rt_{\mathcal{T}}: \mathcal{R} \rightarrow (0, \infty]$$

$$rt_{\mathcal{T}}(r) := \begin{cases} \min \{\gamma \in (0, \infty) \mid r(\gamma) \in \mathcal{T}\} & : \mathcal{T} \cap (r \setminus \{r(0)\}) \neq \emptyset \\ \infty & : \text{sonst} \end{cases}$$

Wie bereits erwähnt, erweitert man diesen Algorithmus häufig mit der Berechnung eines gesamten Bildes, indem man für jeden Pixel des Bildes einen oder mehrere Strahlen durch einen analogen virtuellen Pixel im Szenenraum schießt und die Raytracing-Funktion für diese evaluiert [29, 32, 34]. Abbildung 7 zeigt dieses Verfahren anhand einer Skizze. Um gleichzeitig das sogenannte »Shading« zu ermöglichen, ermittelt man nicht nur den Schnittpunkt, sondern auch in welchem Dreieck

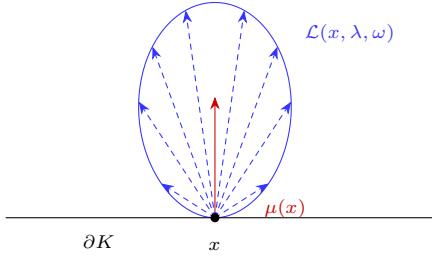


Abbildung 8: Die Skizze stellt das qualitative Beispiel einer spektralen Strahldichte \mathcal{L} dar, die an einem Punkt x auf der Oberfläche ∂K mit der äußereren Normalen $\mu(x)$ und für eine Wellenlänge λ eine Verteilung in Abhängigkeit von $\omega \in S^2$ besitzt.

sich dieser befindet und welche baryzentrischen Koordinaten er besitzt [24, 34]. Die Implementierung des Raytracing-Verfahrens soll hier nicht gezeigt werden, da eine einfache Implementierung einen zu großen Rechenaufwand darstellt und die hier verwendete optimierte Variante weit über das Thema dieser Arbeit hinausgeht. Für eine strukturierte Einführung von Beschleunigungsstrukturen sei auf [34, S. 247 ff] und [29, 32] verwiesen.

2.5 Radiometrie

Ein physikalischer Körper K mit der Oberfläche ∂K und äußerer Normalen-Funktion μ sendet aufgrund verschiedener physikalischer Effekte, wie zum Beispiel Reflexion oder Emission, elektromagnetische Wellen aus [30]. Für die Simulation von Beleuchtungseffekten ist es nötig, die Abhängigkeit dieser Abstrahlung für verschiedene Wellenlängen $\lambda \in (0, \infty)$, verschiedene Oberflächenpunkte $x \in \partial K$ und verschiedene Richtungen $\omega \in S^2$ zu betrachten. Dafür führen wir die sogenannte »spektrale Strahldichte« (engl.: *spectral radiance*) ein. Sie gibt an, wie viel Energie pro Zeit, pro Wellenlängenänderung, pro Flächenelement und pro Raumwinkel abgestrahlt beziehungsweise empfangen wird [22, 31, 44]. Sind also messbare Teilmengen $\Lambda \subset (0, \infty)$, $U \subset \partial K$ und $S \subset S^2$ gegeben, so kann man die spektrale Strahldichte über die abgegebene Strahlungsleistung $\Phi(U, \Lambda, S)$ der Oberfläche U in den Raumwinkelbereich S definieren [44]. Die zugehörige Abbildung ist dann wie folgt gegeben.

$$\mathcal{L}: \partial K \times (0, \infty) \times S^2 \rightarrow [0, \infty)$$

$$\Phi(U, \Lambda, S) := \int_U \int_{\Lambda} \int_S \mathcal{L}(x, \lambda, \omega) |\langle \mu(x), \omega \rangle| d\sigma(\omega) d\lambda(\lambda) d\sigma(x)$$

Abbildung 8 zeigt eine Skizze, welche die Struktur dieser Funktion verdeutlicht. Der Erfahrung nach stellt die spektrale Strahldichte für das menschliche Auge die sinnvollste messbare Größe für die empfundene Helligkeit einer Oberfläche bezüglich eines Beobachtungspunktes dar [22, 31].

Eine weitere Größe, die ich hier einführen möchte, ist die sogenannte spektrale »Bestrahlungsstärke« oder auch spektrale Irradianz (engl.: *spectral irradiance*). Sie wird vor allem bei der noch folgenden Konstruktion der Irradiance Maps gebraucht werden. Wir definieren sie für eine messbare Teilmenge $S \subset S^2$ [12, S. 9 f].

$$\mathcal{E}: \partial K \times (0, \infty) \rightarrow [0, \infty), \quad \mathcal{E}(x, \lambda) := \int_S \mathcal{L}(x, \lambda, \omega) |\langle \mu(x), \omega \rangle| d\sigma(\omega)$$

Die spektrale Bestrahlungsstärke quantifiziert die Energie pro Zeit, pro Wellenlängenänderung und pro Flächenelement, die an einem Punkt auf der Oberfläche des Objektes aus dem Raumwinkelbereich S empfangen wird [12, 44].

Für die weiteren Algorithmen und Implementierungen reicht es, entkoppelte Abbildungen zu betrachten, welche für jeden Punkt einer Szene und jede gegebene Richtung der Strahldichte beziehungsweise der Irradianz entsprechen. Später werden dies die Funktionen sein, die wir durch »Path Tracing« simulieren und durch eine Irradiance Map auf der Oberfläche speichern wollen.

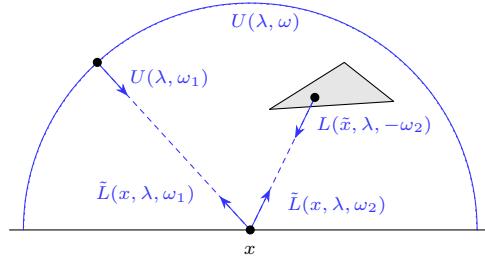


Abbildung 9: Die Abbildung skizziert an einem Beispiel die Relation der Funktionen L und \tilde{L} aus der Definition der Strahldichte. Dabei stellen das Dreieck einen Teil der Mesh, U die Umgebungsbeleuchtung für $\omega \in S^2$, ω_1, ω_2 Raumrichtungen, λ die betrachtete Wellenlänge und x einen Punkt auf der Mesh dar. Für \tilde{x} gilt nach Definition $\tilde{x} = r(rt_{\mathcal{T}}(r))$, wobei der Strahl r durch (x, ω_2) gegeben ist. Der Strahl in Richtung ω_1 besitzt keinen Schnittpunkt mit der Szene, wodurch sich die einfallende Strahldichte aus der Umgebungsbeleuchtung ergibt.

DEFINITION 2.12: (Strahldichte und Irradianz)

Sei $\Sigma := (\mathcal{T}, \nu, f, E, U)$ eine Szene. Dann ist eine Strahldichte von Σ gegeben durch eine integrierbare Abbildung

$$L: \mathcal{T} \times (0, \infty) \times S^2 \rightarrow [0, \infty)$$

Die einfallende Strahldichte \tilde{L} der Szene Σ bezüglich L wird für alle $x \in \mathcal{T}$, $\lambda \in (0, \infty)$ und $\omega \in S^2$ wie folgt formuliert.

$$\tilde{L}(x, \lambda, \omega) := \begin{cases} L(r(rt_{\mathcal{T}}(r)), \lambda, -\omega) & : r \in \mathcal{R}, r \equiv (x, \omega), rt_{\mathcal{T}}(r) < \infty \\ U(\lambda, \omega) & : \text{sonst} \end{cases}$$

Die zu L gehörige Irradianz definieren wir durch die folgende Funktion.

$$R: \mathcal{T} \times (0, \infty) \rightarrow [0, \infty), \quad R(x, \lambda) := \int_{\mathcal{H}_{\nu(x)}^2} \tilde{L}(x, \lambda, \omega) \langle \mu(x), \omega \rangle d\sigma(\omega)$$

In der Definition entspricht $\tilde{L}(x, \lambda, \omega)$ der aus ω einfallenden Strahldichte am Punkt x der Wellenlänge λ . Die Funktionen \tilde{L} und L sind über die Raytracing-Funktion miteinander verbunden. Existiert jedoch für den gegebenen Strahl r kein Schnittpunkt mit der Szene, so entspricht die einfallende Strahldichte gerade der Umgebungsbeleuchtung. In Abbildung 9 wird dieser Zusammenhang an einem Beispiel demonstriert.

2.6 Rendergleichung

Die »Rendergleichung« (engl.: *rendering equation* oder *light transport equation*, [34, S. 861]) ist eine Integralgleichung, die 1986 von James T. Kajiya entwickelt wurde, um die bis zu diesem Zeitpunkt verbreiteten Rendertechniken für die Simulation globaler Beleuchtungseffekte auf eine gemeinsame Basis zu stellen [14]. In den Quellen [14], [34, S. 349 ff, 862 ff] und [38] wird sie mithilfe der Prinzipien der geometrischen Optik und dem Energieerhaltungssatz hergeleitet. Quelle [38] stellt dabei die resultierende Gleichung zusätzlich in einer Operator-Formulierung dar. Insbesondere handelt es sich bei der Rendergleichung um eine Fredholm-Integralgleichung 2. Art, die damit unter gewissen Voraussetzungen eine eindeutige Lösung besitzt (siehe hierzu [38, S. 103 ff] und [35]).



Abbildung 10: Die Abbildung stellt eine durch Path Tracing simulierte Lösung der Rendergleichung für die »Audi R8«-Szene mit »Basketball Court«-HDR dar. Für die BSDFs ist immer eine Mischung aus idealer Reflexion, lambertsch diffuser Reflexion und idealer Brechung gewählt worden.

DEFINITION 2.13: (Rendergleichung)

Seien $\Sigma := (\mathcal{T}, \nu, f, E, U)$ eine Szene und L eine Strahldichte von Σ . Dann gehorcht L der Rendergleichung, wenn für alle $x \in \mathcal{T}$, $\lambda \in (0, \infty)$ und $\omega_o \in \mathbb{S}^2$ das Folgende gilt.

$$L(x, \lambda, \omega_o) = E(x, \lambda, \omega_o) + \int_{\mathbb{S}^2} f(x, \omega, \omega_o) \tilde{L}(x, \lambda, \omega) |\langle \nu(x), \omega \rangle| d\sigma(\omega)$$

Die Strahldichte in Richtung ω entspricht damit der Summe des emittierten und des gestreuten Lichtes am Punkt x . Gehorcht die Strahldichte einer Szene der Rendergleichung, so simuliert sie die globalen Beleuchtungseffekte dieser Szene auf einer physikalischen Basis, wodurch die entstehenden Lichtverhältnisse für das menschliche Auge real wirken [14, 34, 38]. Die Rendergleichung lässt sich aber im Allgemeinen nicht analytisch lösen [14, 34, 38]. Aus diesem Grund sind verschiedene numerische Lösungsmethoden für die Berechnung einer Strahldichte, die der Rendergleichung gehorcht, entwickelt worden. Die berühmtesten Beispiele stellen das »Path Tracing« [14], das »Bidirectional Path Tracing« [18], der »Metropolis Light Transport« [39] und das »Photon Mapping« [13] dar. Eine strukturierte Einführung zu diesen Verfahren gibt es auch hier in [34]. Das Beispiel für eine durch Path Tracing erzeugte Lösung ist in Abbildung 10 zu sehen.

Alle genannten Verfahren arbeiten auf der Basis von Zufallszahlen. Da die Rendergleichung eine mehrdimensionale Integralgleichung ist, bietet sich die Verwendung verschiedener »Monte-Carlo-Methoden« an, weil diese eine effiziente Schätzung mehrdimensionaler Integrale erlauben [26]. Für uns bedeutet diese Tatsache, dass es sich bei der Abfrage der Strahldichte für diese numerischen Verfahren um die Realisierung einer Zufallsvariable handelt, deren Erwartungswert im besten Falle der der wahren Strahldichte entspricht. Eine detailliertere Betrachtung dieser Eigenschaft wird in den weiteren Kapiteln folgen. Für die Konstruktion der Irradiance Maps ist es im Grunde genommen unwichtig, welcher Algorithmus verwendet wird, sofern er in der Lage ist, die Strahldichte für gegebene Parameter zu evaluieren. Wir verwenden hier einen einfachen Path Tracing Algorithmus.

3 Bestimmung der Irradianz

Nach der Einführung der Rendergleichung und dem Nennen verschiedener Methoden zur Simulation dieser, wollen wir nun die bereits in Kapitel 1 angesprochenen Probleme aufgreifen und Lösungsmethoden entwerfen. In diesem Kapitel wird es vor allem um die Eigenschaften spezieller Materialien, die auch Lambertsche Strahler genannt werden, und die Beschreibung einer Messmethode der Irradianz gehen. Im noch folgenden Kapitel werden die gewonnenen Informationen implizit bei der Konstruktion der Irradiance Map angewendet.

3.1 Konstruktion komplexer Materialien

Wie bereits in Abschnitt 2.2 erwähnt, ist es im Allgemeinen nicht möglich, die BSDF eines Materials in geschlossener Form anzugeben. Es gibt jedoch verschiedene Verfahren um einfache BSDF-Modelle zu konstruieren, die in der Simulation der Strahldichte Anwendung finden. In Quelle [34, S. 507 f] wird hierfür die Verwendung von Messdaten, phenomenologischen Beobachtungen, Simulationsergebnissen und physikalischen Gesetzen als Beispiel genannt. Durch die Kombination solcher Modelle ist man in der Lage auch diverse komplexe Materialien zu simulieren. Das folgende Theorem formuliert dieses Vorgehen genauer.

THEOREM: (BSDF-Reihe)

Seien $\mu \in \mathcal{S}^2$, $(f_n)_{n \in \mathbb{N}}$ eine Folge von BSDFs bezüglich μ und $(\alpha_n)_{n \in \mathbb{N}}$ eine Folge von Werten in $[0, 1]$, sodass die beiden folgenden Eigenschaften für σ^2 -fast-alle $(\omega_i, \omega_o) \in \mathcal{S}^2 \times \mathcal{S}^2$ gelten.

$$\sum_{n \in \mathbb{N}} \alpha_n f_n(\omega_i, \omega_o) < \infty, \quad \sum_{n \in \mathbb{N}} \alpha_n \leq 1$$

Dann ist auch die Abbildung f mit der folgenden Definition eine BSDF bezüglich μ .

$$f: \mathcal{S}^2 \times \mathcal{S}^2 \rightarrow [0, \infty), \quad f(\omega_i, \omega_o) := \begin{cases} \sum_{n \in \mathbb{N}} \alpha_n f_n(\omega_i, \omega_o) & : \sum_{n \in \mathbb{N}} \alpha_n f_n(\omega_i, \omega_o) < \infty \\ 0 & : \text{sonst} \end{cases}$$

Beweis:

Für die Wohldefiniertheit von f betrachten wir dessen Definitions- und Wertebereich. Der Definitionsbereich von f entspricht dem der f_n für alle $n \in \mathbb{N}$. Weiterhin folgt aus der Definition von f , dass $f < \infty$ gilt. Für $n \in \mathbb{N}$ betrachtet man dann

$$f_n \geq 0 \implies \alpha_n f_n \geq 0 \implies f = \sum_{n \in \mathbb{N}} \alpha_n f_n \geq 0$$

Insbesondere ist also $f(\omega_i, \omega_o) \in [0, \infty)$ für alle $\omega_i, \omega_o \in \mathcal{S}^2$. Damit ist f eine Abbildung der Form $f: \mathcal{S}^2 \times \mathcal{S}^2 \rightarrow [0, \infty)$ und wohldefiniert.

Kommen wir nun zur Integrierbarkeit. Für alle $n \in \mathbb{N}$ sind die Abbildungen f_n und infolgedessen auch $\alpha_n f_n$ integrierbar. Wir definieren die Folge $(g_n)_{n \in \mathbb{N}}$ von Funktionen durch

$$g_n := \sum_{i=1}^n \alpha_i f_i$$

Dann ist g_n aufgrund der Linearität des Integrals integrierbar und es gilt $g_n \leq g_{n+1}$ für alle $n \in \mathbb{N}$. Weiterhin erhält man durch die Anwendung der Definition die folgende Aussage für σ^2 -fast-alle $(\omega_i, \omega_o) \in \mathcal{S}^2 \times \mathcal{S}^2$.

$$g_n(\omega_i, \omega_o) \xrightarrow{n \rightarrow \infty} f(\omega_i, \omega_o)$$

Nach dem Satz über die Monotone Konvergenz [9, S. 125] ist damit f eine integrierbare Funktion, für die das Folgende aufgrund der Linearität des Integrals gilt.

$$\begin{aligned}\int_{\mathbb{S}^2 \times \mathbb{S}^2} f \, d\sigma^2 &= \lim_{n \rightarrow \infty} \int_{\mathbb{S}^2 \times \mathbb{S}^2} g_n \, d\sigma^2 = \lim_{n \rightarrow \infty} \int_{\mathbb{S}^2 \times \mathbb{S}^2} \sum_{i=1}^n \alpha_i f_i \, d\sigma^2 \\ &= \lim_{n \rightarrow \infty} \sum_{i=1}^n \alpha_i \int_{\mathbb{S}^2 \times \mathbb{S}^2} f_i \, d\sigma^2 = \sum_{n \in \mathbb{N}} \alpha_n \int_{\mathbb{S}^2 \times \mathbb{S}^2} f_n \, d\sigma^2\end{aligned}$$

Die Helmholtz-Reziprozität von f wird direkt durch die Verwendung der Helmholtz-Reziprozität der $f_{n,n} \in \mathbb{N}$ klar. Für σ^2 -fast-alle (ω_i, ω_o) mit $\omega_i \in \mathbb{S}^2, \omega_o \in \mathcal{H}_\nu^2$ und $\nu := \text{sgn}(\langle \mu, \omega_i \rangle) \cdot \mu$ gilt demnach

$$f(\omega_i, \omega_o) = \sum_{n \in \mathbb{N}} \alpha_n f_n(\omega_i, \omega_o) = \sum_{n \in \mathbb{N}} \alpha_n f_n(\omega_o, \omega_i) = f(\omega_o, \omega_i)$$

Für die Energieerhaltung erhalten wir eine entsprechende Aussage durch die Anwendung des Satzes von Fubini [9, S. 175 f]. Es ist damit $f(\omega_i, \cdot)$ σ -integrierbar für σ -fast-alle $\omega_i \in \mathbb{S}^2$. Die Funktion $|\langle \mu, \cdot \rangle|$ ist stetig und durch 1 beschränkt. Mithin ist auch $f(\omega_i, \cdot) |\langle \mu, \cdot \rangle|$ messbar und es gilt für σ -fast-alle $\omega_i \in \mathbb{S}^2$

$$f(\omega_i, \cdot) |\langle \mu, \cdot \rangle| \leq f(\omega_i, \cdot) \implies \int_{\mathbb{S}^2} f(\omega_i, \cdot) |\langle \mu, \cdot \rangle| \, d\sigma \leq \int_{\mathbb{S}^2} f(\omega_i, \cdot) \, d\sigma < \infty$$

Analog zum Beweis der Integrierbarkeit von f lässt sich dann mithilfe der Energieerhaltung der $f_{n,n} \in \mathbb{N}$ die Energieerhaltung von f formulieren.

$$\int_{\mathbb{S}^2} f(\omega_i, \cdot) |\langle \mu, \cdot \rangle| \, d\sigma = \sum_{n \in \mathbb{N}} \alpha_n \int_{\mathbb{S}^2} f_n(\omega_i, \cdot) |\langle \mu, \cdot \rangle| \, d\sigma \leq \sum_{n \in \mathbb{N}} \alpha_n \leq 1$$

□

Nach dieser Aussage ist man in der Lage, abzählbar viele BSDFs mithilfe entsprechender Koeffizienten miteinander zu kombinieren. Eine direkte Folgerung bildet die Anwendung des Satzes auf nur endlich viele BSDFs.

KOROLLAR: (BSDF-Summe)

Seien $\mu \in \mathbb{S}^2$ und $n \in \mathbb{N}$. Weiterhin seien f_k eine BSDF bezüglich μ und $\alpha_k \in [0, 1]$ für alle $k \in \mathbb{N}$ mit $k \leq n$, sodass $\sum_{k=1}^n \alpha_k \leq 1$ gilt. Dann ist die folgende Abbildung eine BSDF bezüglich μ .

$$f := \sum_{k=1}^n \alpha_k f_k$$

3.2 Äquivalenz Lambertscher Strahler und der Irradianz

Eines der einfachsten BSDF-Modelle ist das der Lambertschen diffusen Reflexion, welches in Abschnitt 2.2 als Beispiel eingeführt wurde. Die einfallende Strahldichte wird bei diesem Modell gleichmäßig in alle Richtungen der Halbkugel gestreut, was auch der idealen diffusen Reflexion entspricht. Eine Oberfläche mit dieser Eigenschaft wird auch Lambertscher Strahler (engl.: *Lambertian radiator*, [44, S. 17]) genannt. Zu beachten ist, dass diese BSDF keinerlei Transmission von Licht beschreibt und auch im Allgemeinen keiner physikalischen Basis entspricht. Dennoch stellt sie eine gute Approximation vieler realer Oberflächen, die ein mattes Aussehen besitzen, dar [34, S. 532]. Das Modell eignet sich aufgrund seiner Simplizität für die numerische Simulation der globalen Beleuchtung. In Abbildung 11 ist erkennbar, dass der Szene durch die Verwendung des Modells ein plastisches und damit auch realistischeres Aussehen zu Teil wird.

Wir wollen nun eine Szene $\Sigma := (\mathcal{T}, \nu, f, E, U)$ und einen Punkt $x \in \mathcal{T}$ betrachten, sodass $f(x, \cdot, \cdot)$ einen Lambertschen Strahler bezüglich $\nu(x)$ darstellt. Die Rendergleichung für die Strahldichte L am

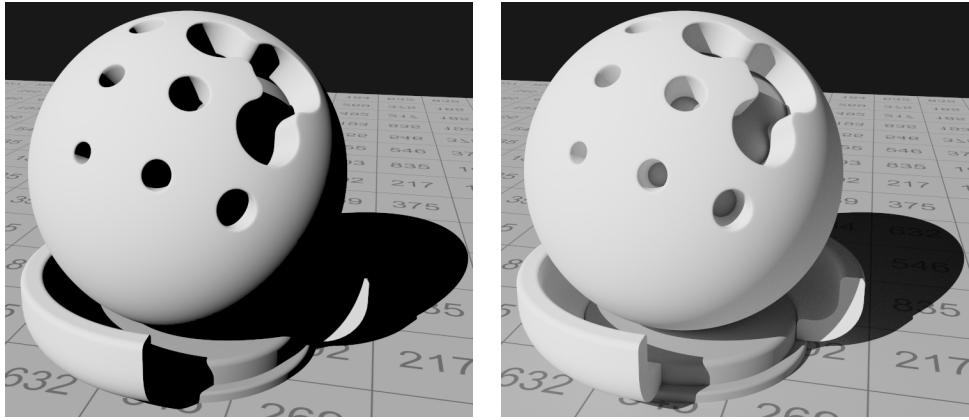


Abbildung 11: Die Bilder zeigen die gerenderte »Shaderball«-Szene, beleuchtet durch eine direktionale Lichtquelle. Im linken Bild wurde die Strahldichte der Szene nur durch direkte Beleuchtung ohne die Verwendung einer entsprechenden BSDF berechnet. Im rechten Bild wurde für jedes Material eine lambertsch diffuse Reflexion angenommen. Die Szene wirkt hier wesentlich plastischer und realistischer als im linken Bild.

Punkt x in Richtung $\omega_o \in \mathcal{H}_{\nu(x)}^2$ für eine Wellenlänge $\lambda \in (0, \infty)$ lautet dann wie folgt.

$$L(x, \lambda, \omega_o) = E(x, \lambda, \omega_o) + \underbrace{\frac{1}{\pi} \int_{S^2} \mathbb{1}_{\mathcal{H}_{\nu(x)}^2}(\omega) \tilde{L}(x, \lambda, \omega) |\langle \nu(x), \omega \rangle| d\sigma(\omega)}_{=: L_r(x, \lambda, \omega_o)}$$

$L_r(x, \lambda, \omega_o)$ beschreibt den reflektierten Anteil der Strahldichte. Die charakteristische Funktion im Integranden ändert die Integrationsmenge. Bezeichnen wir die Irradianz von L durch die Variable R , so folgt

$$L_r(x, \lambda, \omega_o) = \frac{1}{\pi} \int_{\mathcal{H}_{\nu(x)}^2} \tilde{L}(x, \lambda, \omega) \langle \nu(x), \omega \rangle d\sigma(\omega) = \frac{R(x, \lambda)}{\pi}$$

Der reflektierte Anteil $L_r(x, \lambda, \omega_o)$ ergibt sich damit aus der mit $\frac{1}{\pi}$ skalierten Irradianz $R(x, \lambda)$ [33, S. 785 f]. Insbesondere ist er damit unabhängig von der Richtung ω_o . Das Einsetzen in den übrigen Teil der Gleichung ergibt dann direkt

$$L(x, \lambda, \omega_o) = E(x, \lambda, \omega_o) + \frac{1}{\pi} R(x, \lambda)$$

In den meisten Fällen ist E selbst unabhängig von ω_o und an vielen Punkten im Raum sogar identisch mit Null. Für matte Oberflächen, deren Reflexionsverhalten durch einen Lambertschen Strahler charakterisiert werden kann, reicht es dementsprechend die Irradianz zu betrachten [42, 43].

Verwendet man jetzt das Korollar des vorigen Abschnittes, so lässt sich das einfache Modell der Lambertschen diffusen Reflexion zum Beispiel mit einer idealen Reflexion verbinden, wie es in Abbildung 12 gezeigt ist. Das resultierende Material ist komplexer aufgebaut und beschreibt eine andere Klasse von BSDFs. Die Verwendung eines Lambertschen Strahlers kann somit durch Kombination mit anderen BSDF-Modellen auf einen größeren Bereich von Materialarten ausgeweitet werden. Mithin ist es in der Praxis in fast allen Szenen nötig, diffuse Reflexionen an den Oberflächen der Objekte auszuwerten.

Wir nehmen jetzt an, dass sich die BSDF am Punkt x aus der Kombination eines Lambertschen Strahlers mit Koeffizient α und einer beliebigen BSDF $\tilde{f}(x, \cdot, \cdot)$ mit Koeffizient β ergibt. Dabei fordern wir, dass $\alpha, \beta \in [0, 1]$ und $\alpha + \beta \leq 1$ gilt. Das Korollar über die Summe von BSDFs tätigt dann eine klare Aussage über die Form des reflektierten Anteils.

$$L(x, \lambda, \omega_o) = E(x, \lambda, \omega_o) + \frac{\alpha}{\pi} R(x, \lambda) + \beta \int_{S^2} \tilde{f}(x, \omega, \omega_o) \tilde{L}(x, \lambda, \omega) |\langle \nu(x), \omega \rangle| d\sigma(\omega)$$

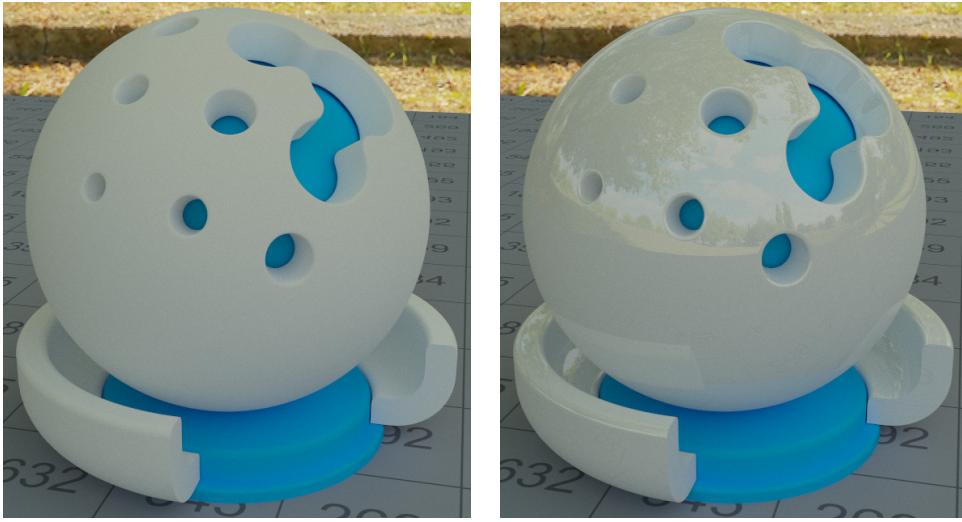


Abbildung 12: Die Bilder zeigen die gerenderte »Shaderball«-Szene, beleuchtet durch eine Umgebungsbeleuchtung. Im linken Bild bestehen die verwendeten BSDFs nur aus der lambertsch diffusen Reflexion. Im rechten Bild wurde das Material der äußeren Körpers zusätzlich mit der BSDF einer idealen Reflexion kombiniert. Das entstehende Material ist kein idealer Spiegel, aber auch kein idealer diffuser Strahler.

Diese Form der Rendergleichung für Lambertsche Strahler besitzt für die Verwendung in der Praxis einen Vorteil. Die BSDF $\tilde{f}(x, \cdot, \cdot)$ ergibt sich meistens aus stark richtungsabhängigen Anteilen, wie zum Beispiel der idealen Reflexion, der idealen Brechung oder ähnlich glänzenden Materialien. Diese Anteile können vergleichsweise gut durch Raytracing beziehungsweise auch Path Tracing gerendert werden. Die Berechnung von $R(x, \lambda)$ erfordert jedoch immer eine Integration über die gesamte Hemisphäre, bei der jeder erhaltene Wert gleichermaßen gewichtet werden muss. Diese Tatsache erschwert die Evaluierung und führt zu entsprechend langen Zeiträumen des Renderings.

Ein Vorteil der Irradianz besteht jedoch darin, dass sie unabhängig von der betrachteten Richtung ist. Die empfangene Strahldichte von x ist konstant, egal von welchem Punkt im Raum aus man x betrachtet. Diese Eigenschaft ermöglicht es, $R(x, \lambda)$ vorzuberechnen und am Punkt x zu speichern. Jedes Mal, wenn man nun den Punkt x aus einer Richtung $\tilde{\omega} \in \mathcal{H}_{\nu(x)}^2$ betrachtet, wird das Integral des rechten Teils der Gleichung effizient durch Path Tracing geschätzt. Der Wert $R(x, \lambda)$ kann mit $E(x, \lambda, \tilde{\omega})$ ausgelesen und zum resultierenden Wert hinzugefügt werden. Infolgedessen verkürzen sich die benötigten Renderzeiten. Da es sich bei R um die Irradianz handelt, wollen wir die zugehörige Datenstruktur, die ihre Werte speichert, »Irradiance Map« nennen.

Wir möchten hier bemerken, dass die genannten Eigenschaften nur für die Translation des Beobachtungspunktes gelten. Die Änderung der Beleuchtung oder Geometrie führt zu einer neuen Szene $\tilde{\Sigma}$, deren Irradianz \tilde{R} im Allgemeinen nicht R entspricht. Die Irradiance Map muss demnach für jede Modifizierung der Szene neu berechnet werden, um die korrekten Werte der Irradianz zu beinhalten.

3.3 Schätzung der Irradianz

In Abschnitt 2.6 wurde erwähnt, dass die meisten Renderverfahren, wie zum Beispiel das hier verwendete Path Tracing, auf der Basis von Zufallsvariablen arbeiten. Das Integral in der Rendergleichung wird dabei mithilfe geeigneter Monte-Carlo-Methoden berechnet. Aus diesem Grund bildet sich bei den Lösungen der Rendergleichung immer ein gewisses »Rauschen« (engl.: *noise*) aus. In Abbildung 13 wird dies an einem Beispiel demonstriert.

Möchte man die Irradianz für die Irradiance Map ermitteln, so muss darauf geachtet werden, dass das Rauschen des Endresultates und damit dessen Fehler klein bleibt. Erst durch die Beschränkung des Fehlers können gemessene Irradianzen an verschiedenen Punkten interpoliert werden, ohne das



Abbildung 13: Die Bilder zeigen die gerenderte »Fairy«-Szene. Für das linke Bild wurde an jedem sichtbaren Punkt der Szene die Irradianz durch 16 Messungen ermittelt. Das Resultat beinhaltet noch einen großen Anteil des Rauschens. Im rechten Bild wurden jeweils 1024 Messungen durchgeführt, wodurch es wesentlich rauschärmer als das Linke ist.

Artefakte wie in der späteren Abbildung 15b entstehen.

Für die folgende Betrachtung nehmen wir wieder an, dass eine Szene $\Sigma := (\mathcal{T}, \nu, f, E, U)$ und eine Strahldichte L von Σ , die der Rendergleichung genügt, gegeben seien. Wir bezeichnen R als die Irradianz von L und wählen eine festen Punkt $x \in \mathcal{T}$ und eine feste Wellenlänge $\lambda \in (0, \infty)$. Um die Zufälligkeit der Simulation mathematisch zu fassen, verwenden wir einen Wahrscheinlichkeitsraum (X, \mathcal{X}, p) und für jedes $\omega \in \mathcal{S}^2$ die folgende Zufallsvariable.

$$\mathcal{L}_\omega: X \rightarrow [0, \infty), \quad \mathbb{E} \mathcal{L}_\omega = \tilde{L}(x, \lambda, \omega)$$

Die Simulation von $\tilde{L}(x, \lambda, \omega)$ durch Path Tracing oder einem ähnlichen Algorithmus entspricht dann der Realisierung $\mathcal{L}_\omega(\xi)$ mit $\xi \in X$. Für die Irradianz $R(x, \lambda)$ führen wir einen weiteren Wahrscheinlichkeitsraum (Y, \mathcal{Y}, q) ein. Seien $n \in \mathbb{N}$ und unabhängige, identische Zufallsvariablen $w_i: Y \rightarrow \mathcal{H}_{\nu(x)}^2$ für alle $i \in \mathbb{N}, i \leq n$ mit der folgenden zugehörigen Wahrscheinlichkeitsdichte bezüglich σ gegeben.

$$\varrho: \mathcal{H}_{\nu(x)}^2 \rightarrow [0, \infty), \quad \varrho(\omega) := \frac{1}{2\pi}$$

Die w_i sind damit für alle $i \in \mathbb{N}, i \leq n$ auf der Hemisphere $\mathcal{H}_{\nu(x)}^2$ gleichverteilt. Wir sind nun in der Lage weitere Zufallsvariablen \mathcal{R}_i für alle $i \in \mathbb{N}, i \leq n$ auf dem Produkt-Wahrscheinlichkeitsraum $(X \times Y, \mathcal{X} \otimes \mathcal{Y}, p \otimes q)$ zu definieren, deren Erwartungswert eine Aussage über die Irradianz trifft.

$$\mathcal{R}_i: X \times Y \rightarrow [0, \infty), \quad \mathcal{R}_i(\xi, \zeta) := \mathcal{L}_{w_i(\zeta)}(\xi) \langle \nu(x), w_i(\zeta) \rangle$$

Die folgende Rechnung zeigt den Zusammenhang des Erwartungswertes von \mathcal{R}_i mit $R(x, \lambda)$ für alle $i \in \mathbb{N}, i \leq n$.

$$\begin{aligned} \mathbb{E} \mathcal{R}_i &= \int_{X \times Y} \mathcal{R}_i \, d(p \otimes q) \\ (\text{Satz von Fubini [9, S. 175 f]}) \quad &= \int_Y \int_X \mathcal{R}_i(\xi, \zeta) \, dp(\xi) \, dq(\zeta) \\ (\text{Definition } \mathcal{R}_i) \quad &= \int_Y \int_X \mathcal{L}_{w_i(\zeta)}(\xi) \langle \nu(x), w_i(\zeta) \rangle \, dp(\xi) \, dq(\zeta) \\ (\text{Linearität Integral}) \quad &= \int_Y \langle \nu(x), w_i(\zeta) \rangle \int_X \mathcal{L}_{w_i(\zeta)}(\xi) \, dp(\xi) \, dq(\zeta) \end{aligned}$$

$$\begin{aligned}
(\text{Definition Erwartungswert}) &= \int_Y \langle \nu(x), w_i(\zeta) \rangle \mathbb{E} \mathcal{L}_{w_i(\zeta)} dq(\zeta) \\
(\text{Definition } \mathcal{L}_{w_i(\zeta)}) &= \int_Y \tilde{L}(x, \lambda, w_i(\zeta)) \langle \nu(x), w_i(\zeta) \rangle dq(\zeta) \\
(\text{Transformation [9, S. 191 f]}) &= \int_{w_i(Y)} \tilde{L}(x, \lambda, \zeta) \langle \nu(x), \zeta \rangle dq_{w_i}(\zeta) \\
(\text{Maße mit Dichten [9, S. 127 f]}) &= \int_{\mathcal{H}_{\nu(x)}^2} \tilde{L}(x, \lambda, \omega) \langle \nu(x), \omega \rangle \varrho(\omega) d\sigma(\omega) \\
(\text{Definition } \varrho) &= \frac{1}{2\pi} \int_{\mathcal{H}_{\nu(x)}^2} \tilde{L}(x, \lambda, \omega) \langle \nu(x), \omega \rangle d\sigma(\omega) \\
(\text{Definition Irradianz}) &= \frac{R(x, \lambda)}{2\pi}
\end{aligned}$$

Durch diese Äquivalenz erlangen wir die Möglichkeit einen erwartungstreuen Schätzer $\bar{\mathcal{R}}$ der Irradianz $R(x, \lambda)$ zu definieren.

$$\bar{\mathcal{R}} := \frac{2\pi}{n} \sum_{i=1}^n \mathcal{R}_i$$

Nach Quelle [15, S. 249 ff] ergibt sich dann mit einem $i \in \mathbb{N}, i \leq n$ für den Erwartungswert und die Varianz

$$\mathbb{E} \bar{\mathcal{R}} = R(x, \lambda), \quad \text{var } \bar{\mathcal{R}} = \frac{4\pi^2}{n} \text{ var } \mathcal{R}_i$$

Wir können also zufällige Richtungen der Hemisphere auswählen und für diese die Strahldichte ermitteln. Das arithmetische Mittel sollte dann nach dem »schwachen Gesetz der großen Zahlen« [15, S. 254] für eine steigende Anzahl von Messwerten gegen den Wert der Irradianz konvergieren. Es sei hier auch auf die Quellen [14, 34, 38] verwiesen, die diese Herleitung mithilfe der »Pfadintegral-Formulierung« der Rendergleichung vollführen. Das eben beschriebene Verfahren ist ein typisches Beispiel einer Monte-Carlo-Methode [26]. Die Abbildungen 14 und 15 zeigen es anhand zweier Beispielezenen für verschiedene Sampleanzahlen.

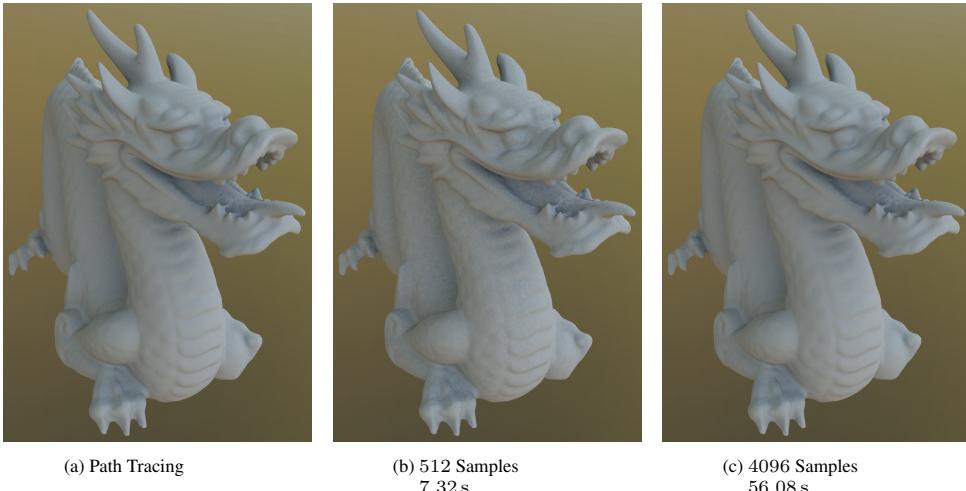


Abbildung 14: Die Bilder zeigen die gerenderte »Dragon«-Szene mit einer schwach variierenden Umgebungsbeleuchtung. 14a ist ein durch Path Tracing aufgenommenes Referenzbild. Für 14b und 14c wurden die Irradianzen an den Eckpunkten der Mesh geschätzt und gespeichert. Zwischen den Eckpunkten fand eine lineare Interpolation statt. Die Anzahl der Samples gibt an, wie viele Stichprobenwerte verwendet wurden, um die Irradianz an jedem Punkt zu schätzen. Die Zeit entspricht dabei der Berechnungsdauer. 14c weist zum Referenzbild keine Unterschiede auf. In 14b sind jedoch einige schwache Artefakte in Form von Lichtflecken zu beobachten.

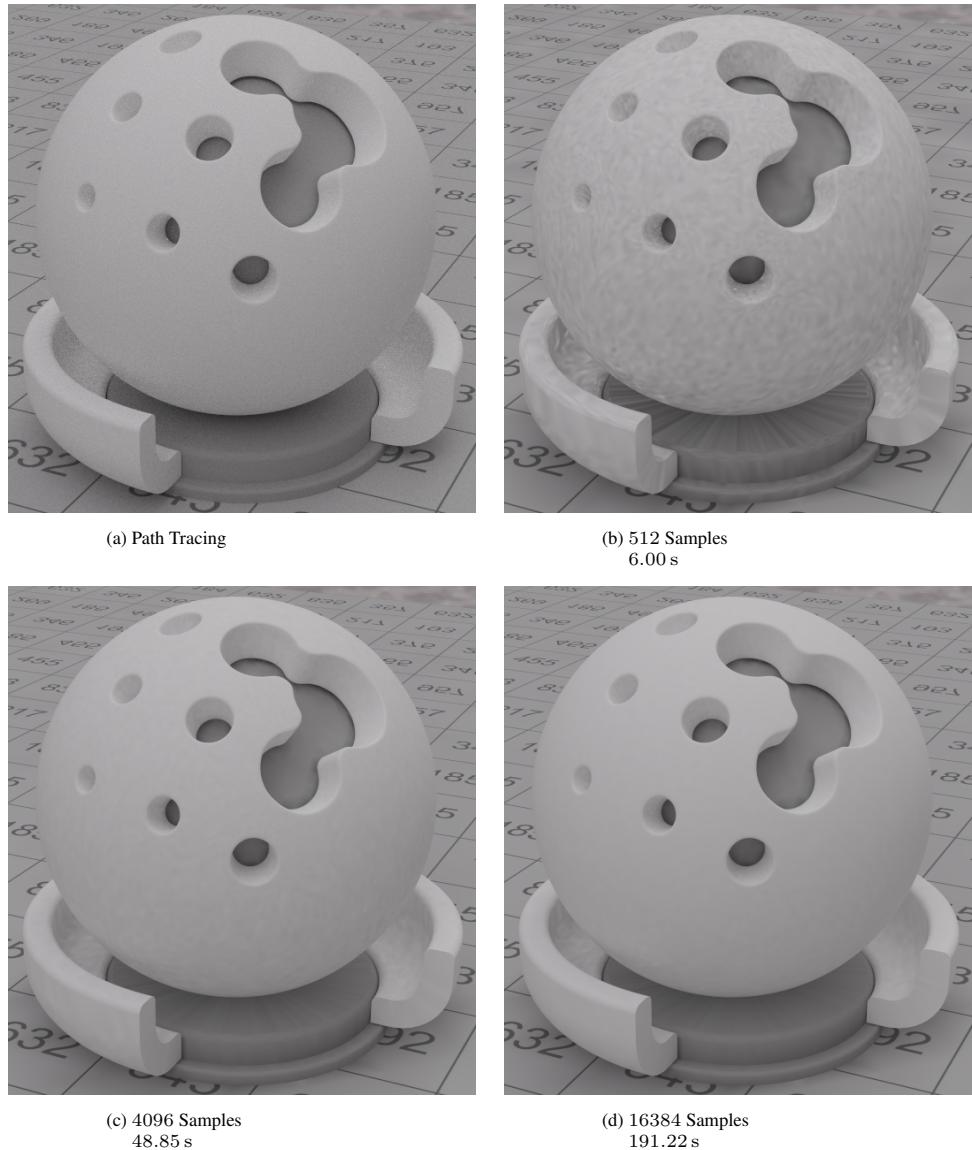


Abbildung 15: Die Bilder zeigen die gerenderte »Shaderball«-Szene mit der »Uffzi Gallery«-HDR entsprechend der Benennung aus Abbildung 14. Die Monte-Carlo-Integration erzeugt in 15b mit nur 2^9 Samples ein fleckiges Muster, welches durch die Verteilung der Zufallsvariable um den Erwartungswert hervorgerufen wird. In 15c und 15d ist erkennbar, dass die Entstehung von Flecken durch eine höhere Sampleanzahl verringert werden kann. Dennoch sind in 15d auch für 2^{14} Samples im unteren Stand der Szene noch Flecken sichtbar.

Die Ergebnisse des Algorithmus variieren stark mit der Szenengeometrie und der gegebenen Umgebungsbeleuchtung. Während im Bild 14c in der »Dragon«-Szene mit einer einfachen Umgebungsbeleuchtung bereits nach 4096 Messungen an einem Punkt kein Unterschied mehr zum Referenzbild 31a festgestellt werden kann, sind in der »Shaderball«-Szene mit der »Uffzi Gallery«-HDR in Bild 15d mit 16384 Messungen pro Punkt immer noch fleckige Artefakte zu sehen.

Eine einfache Erklärung für dieses Phänomen kann durch eine Analyse der Szene gefunden werden. Für komplexe Umgebungsbeleuchtungen wie in Anhang B muss im Allgemeinen eine sehr viel höhere Sampleanzahl verwendet werden, um in der Lage zu sein Details, wie zum Beispiel kleine Lichtquellen, aufzulösen und in die Schätzung mit einzubeziehen. Weiterhin spielt auch die Konstellation der Dreiecke zum Rest der Szene eine Große Rolle. In der »Dragon«-Szene werden vom Beobachter ausgesendete Strahlen meistens nur ein einziges Mal an der Oberfläche der Mesh reflektiert. Nach der Reflexion

treffen sie keine weiteren Dreiecke, wodurch im letzten Schritt nur noch die Umgebungsbeleuchtung evaluiert wird. Dies verringert die Varianz der gemessenen Irradianz, weil die einfallende Strahldichte für fast alle Richtungen durch die feste Umgebungsbeleuchtung und nicht durch eine Zufallsvariable gegeben ist. Im unteren Stand der »Shaderball«-Szene ist dies nicht mehr der Fall. Hier kann es durchaus passieren, dass ein Strahl mehrere Male reflektiert werden muss, um entweder zur Umgebungsbeleuchtung zu gelangen oder durch ein Material absorbiert zu werden. Dieser Effekt erhöht die Varianz eines Messpunktes.

Aus dieser Betrachtung lässt sich schließen, dass die Verringerung der Varianz aller Punkte für beliebige Szenen nur durch extrem hohe Sampleanzahlen möglich ist. Eine solche Schätzung für alle Eckpunkte der Szene durchzuführen, wäre zeitraubend und ineffizient.

3.4 Fehler der Irradianzschätzung

Um die Schätzung der Irradianz effizienter zu gestalten, benötigen wir ein Maß für den Fehler dieser Schätzung. Im letzten Abschnitt verwendeten wir hierfür die Varianz der Zufallsvariable. Für eine feste vorgegebene Anzahl von Samples zeigte sich diese als ausreichend. Möchte man jedoch die Sampleanzahl an verschiedenen Punkten der Oberfläche variieren, so benötigen wir ein Fehlermaß, welches in den Bereichen von Artefakten einen hohen und sonst einen niedrigen Wert besitzt. Die Varianz erfüllt diese Eigenschaft nicht, da die Strahldichte und damit auch die Irradianz nach unten aber nicht nach oben beschränkte Funktionen sind. Eine Strahldichte mit großem Erwartungswert weist im Allgemeinen einen höheren Varianz als eine Strahldichte mit niedrigem Erwartungswert auf. Die weitere Verwendung der Varianz als Fehlermaß würde damit zu extrem genauen Messungen in hellen Bereichen und zu extrem ungenauen Messungen in dunklen Bereichen der Szene führen. Die Abbildungen 14 und 15 zeigen jedoch, dass die Entstehung von Artefakten nicht durch die Helligkeit charakterisiert wird.

Die hier verwendete Alternative ist der sogenannte »Variationskoeffizient« (engl.: *coefficient of variation* oder *relative standard deviation*, [10]), der im Folgenden für die Zufallsvariable $\bar{\mathcal{R}}$ des vorigen Abschnittes definiert wird. Mithilfe der Zufallsvariablen $\mathcal{R}_i, i \in \mathbb{N}, i \leq n$ führen wir einen Schätzer \mathcal{C} dieses Koeffizienten ein.

$$\text{rsd } \bar{\mathcal{R}} := \frac{\sqrt{\text{var } \bar{\mathcal{R}}}}{\mathbb{E} \bar{\mathcal{R}}}, \quad \mathcal{C} := \frac{1}{\bar{\mathcal{R}}} \sqrt{\frac{1}{n(n-1)} \sum_{i=1}^n (2\pi\mathcal{R}_i - \bar{\mathcal{R}})^2}$$

\mathcal{C} ist im Allgemeinen nicht erwartungstreu, kann aber für spezielle Verteilungen durch eine Skalierung korrigiert werden [10]. Für hinreichend große $n \in \mathbb{N}$ ist der eingeführte Bias jedoch vernachlässigbar. Um bei der Schätzung des Variationskoeffizienten keine Division durch Null zu erhalten, verwenden wir im Computer eine leicht modifizierte Variante $\tilde{\mathcal{C}}_\varepsilon$ für ein $\varepsilon \in (0, \infty)$, die wir auch als »relativen Fehler« bezeichnen.

$$\tilde{\mathcal{C}}_\varepsilon := \frac{1}{\bar{\mathcal{R}} + \varepsilon} \sqrt{\frac{1}{n(n-1)} \sum_{i=1}^n (2\pi\mathcal{R}_i - \bar{\mathcal{R}})^2}$$

Es stellte sich heraus, dass $\varepsilon = 0.0001$ eine gute Wahl ist. Die Abbildungen 16 und 17 zeigen die Szenen der Abbildung 14 beziehungsweise 15. Bei ihnen wird der modifizierte Variationskoeffizient über die Eckpunkte der Meshes interpoliert und dargestellt. Ein Vergleich mit den zugehörigen Irradianzbildern zeigt, dass das so erhaltene Fehlermaß gerade in den Bereichen der Artefakte einen hohen Wert besitzt und damit den gestellten Anforderungen Genüge leistet.

3.5 Adaptive Schätzung der Irradianz

Das im letzten Abschnitt eingeführte Fehlermaß soll nun verwendet werden, um einen Algorithmus zu formulieren, der die Irradianz eines Punktes effizient durch eine variable Anzahl von Samples schätzen

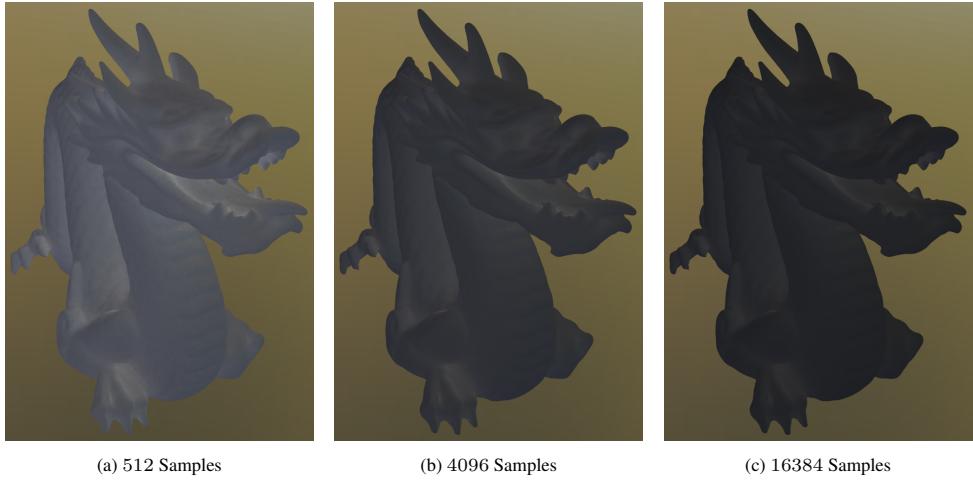


Abbildung 16: Die Bilder zeigen die »Dragon«-Szene aus Abbildung 14 mit den an den Eckpunkten geschätzten modifizierten Variationskoeffizienten. Auch hier findet eine lineare Interpolation zwischen ihnen statt. Eine hellere Farbe bedeutet dabei ein höherer Wert. Deutlich erkennbar ist die Abnahme des Fehlers für größere Sampleanzahlen. Dennoch bleiben auch für 2^{14} Samples noch Bereiche, die einen vergleichsweise hohen Fehler aufweisen.

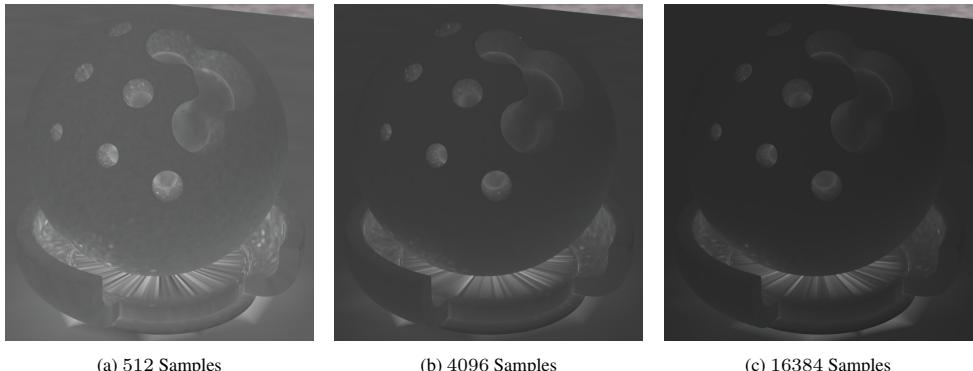


Abbildung 17: Die Bilder zeigen die »Shaderball«-Szene aus Abbildung 15 mit den an den Eckpunkten geschätzten modifizierten Variationskoeffizienten entsprechend der Abbildung 16. Vor allem Bereiche, die schwach von außen beleuchtet werden oder in Nischen liegen, weisen einen erhöhten Fehler auf. Die regelmäßigen Artefakte im unteren Teil entstehen durch die geringe Auflösung der Mesh.

kann. Die Idee besteht darin, solange Samples über der Hemisphere aufzunehmen, bis der geschätzte modifizierte Variationskoeffizient kleiner als eine gegebenen Schranke ist. Um die Rechenzeit zu beschränken, verwendet man eine maximale Sampleanzahl, nach derer der Algorithmus den geschätzten Wert speichert und terminiert. Es gibt aber auch eine vorgeschriebene Mindestanzahl von Samples, die der Algorithmus braucht, um überhaupt den Variationskoeffizienten zu schätzen. Der folgende Quelltext implementiert dieses Verfahren durch die Sprache »C++«. Zu beachten ist, dass in der Implementierung selbst die Realisierungen nicht mit 2π multipliziert werden. Diese Vereinfachung ändert jedoch nicht die Funktionsweise.

Quelltext: Adaptive Irradianzschätzung

```
vecf4 est_irr_adap(const vecf4& pos, const vecf4& n, uint& sample_count){
    const uint max_count = max_sample_count;
    const uint min_count = 256;
    const float rel_err_eps = 0.0001f;

    // initialisiere temporäre Variablen
```

```

    vecf4 sum(0,0,0);
    vecf4 sum_sq(0,0,0);
    uint count = 0;
    float rel_err = 0.0f;

    // generiere weitere Samples, bis Abbruchbedingung erreicht
    while ((count < max_count) && ((count < min_count) || (rel_err >
        rel_err_bound))){  

        // konstruiere Strahl fuer weitere Messung
        ray r;
        r.dir = rnd_dir();
        r.ori = pos;  

        float tmp_dot = vecf4::dot(n,r.dir);
        // Strahl muss nach aussen zeigen
        if (tmp_dot < 0.0f){
            tmp_dot = -tmp_dot;
            r.dir = -r.dir;
        }
        // verhindere Schnittpunkt mit sich selbst
        r.ori += ray_eps * r.dir;  

        // berechne skalierte einfallende Strahldichte durch Path Tracing
        const vecf4 tmp = tmp_dot * path_trace(r);  

        // schaetze fuer Sampleanzahl Mittelwert und Varianz
        sum += tmp;
        sum_sq += tmp * tmp;
        count++;
        const float inv_count = 1.0f / float(count);
        const float inv_count1 = 1.0f / float(count-1);
        const vecf4 mean = sum * inv_count;
        const vecf4 var = ((sum_sq * inv_count1) - (float(count)*inv_count1 *
            mean * mean)) * inv_count;
        // berechne den relativen Fehler
        rel_err = max(sqrt(var) / (mean + rel_err_eps));
    }  

    // setze zurueckgegebene Sampleanzahl
    sample_count = count;  

    // gebe geschaetzte Irradianz zurueck
    return sum * 2.0f * M_PI / float(count);
}

```

Die Ergebnisse des Algorithmus sind in den Abbildungen 18, 19, 20 und 21 für verschiedene Szenen und Umgebungsbeleuchtungen gezeigt. Weitere Beispiele sind in Anhang C aufgelistet. Lässt man Artefakte, die einer zu geringen Auflösung der Szenengeometrie entspringen, beiseite, so führt die Näherung der Irradianz in allen Ergebnissen zu einer sehr guten Übereinstimmung mit den zugehörigen Referenzbildern. Auch die Berechnungszeit betrug in den meisten Fällen nicht einmal die Hälfte der Zeit des ursprünglichen Verfahrens. Zu allen Ergebnissen wurde zudem die Verteilung der Samples angegeben. Diese bestätigt die in Abschnitt 3.3 aufgestellten Theorien zur Erklärung der Entstehung von Flecken und ähnlichen Artefakten. Die Verwendung einer komplizierten Umgebungsbeleuchtung, wie der »Ennis-Brown House«-HDR, für die »Dragon«-Szene in Abbildung 18 resultierte in einer wesentlich höheren durchschnittlichen Sampleanzahl als die Verwendung einer schwach variierender Umgebungsbeleuchtung in Abbildung 31 Anhang C. Auch im unteren Stand oder den oberen Löchern der »Shaderball«-Szene in den Abbildungen 19 und 20, in denen Strahlen mehrmals reflektieren, wurden deutlich mehr Samples benötigt als im Rest der gesamten Bilder. Dies führt jedoch dazu, dass jegliche vorher sichtbaren zufälligen Artefakte verschwinden.

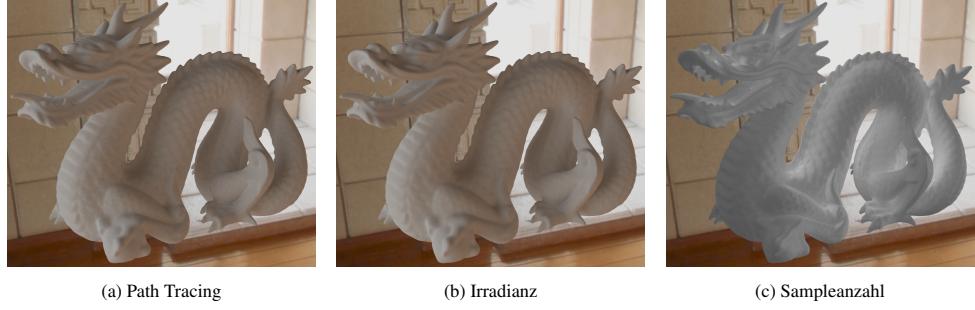


Abbildung 18: Die Bilder zeigen die »Dragon«-Szene mit der »Ennis-Brown House«-HDR. 18a ist das durch Path Tracing erzeugte Referenzbild. 18b zeigt die an den Eckpunkten adaptiv aufgenommenen Irradianzen mit linearer Interpolation. 18c stellt dabei die zugehörigen Sampleanzahlen dar. Hier bedeutet eine hellere Farbe eine höhere Sampleanzahl. Die Berechnungszeit betrug 559 s mit der maximalen Sampleanzahl 2^{16} und einem maximalen relativen Fehler von 3 %.

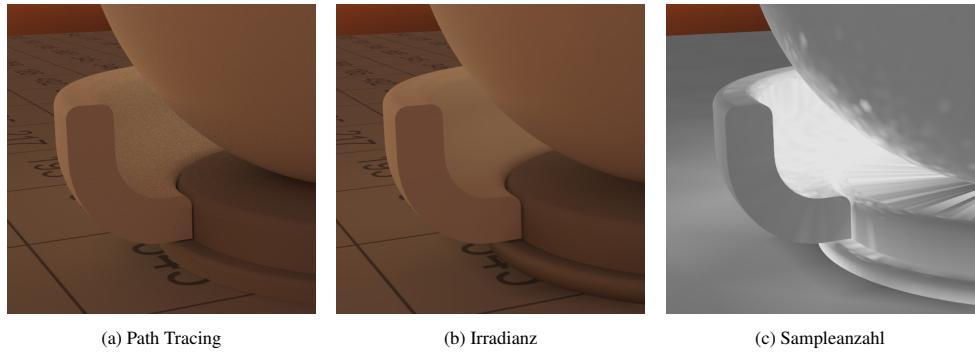


Abbildung 19: Die Bilder zeigen die »Shaderball«-Szene der Abbildung 32 Anhang C aus einem anderen Blickwinkel unter Verwendung der »Sky 20«-HDR. Die Benennung ist analog zu Abbildung 18. Die aufgenommenen Irradianzen beinhalten im unteren Stand teilweise weniger Rauschen als das Referenzbild. Die Berechnungszeit betrug 1170 s mit einer maximalen Sampleanzahl 2^{18} und einem maximalen relativen Fehler von 1 %.

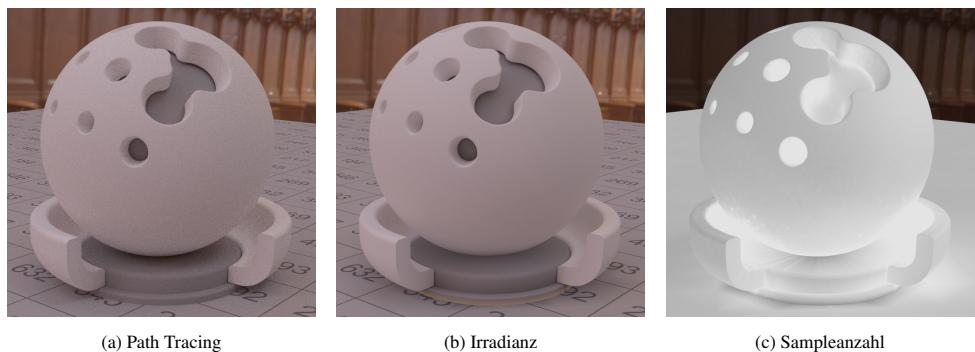
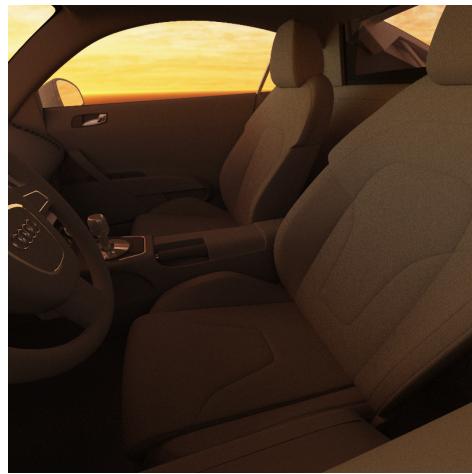


Abbildung 20: Die Bilder zeigen die »Shaderball«-Szene mit der »Grace Cathedral«-HDR. Die Benennung ist analog zu Abbildung 18. Die Berechnungszeit betrug 1470 s mit einer maximalen Sampleanzahl von 2^{18} und einem maximalen relativen Fehler von 1 %.



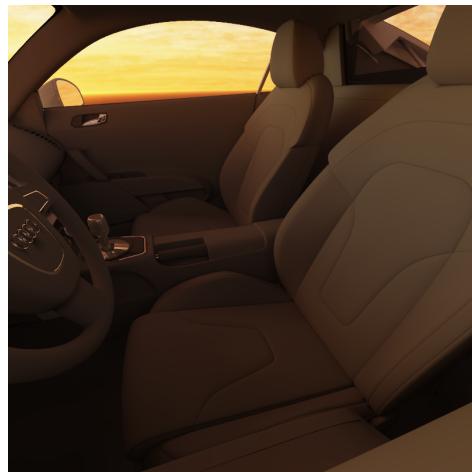
(a) Path Tracing



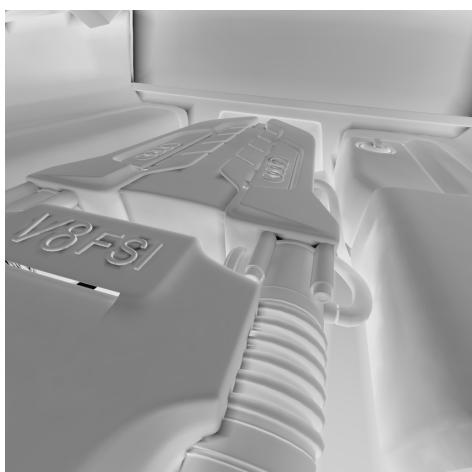
(b) Path Tracing



(c) Irradianz



(d) Irradianz



(e) Sampleanzahl



(f) Sampleanzahl

Abbildung 21: Die Bilder zeigen die »Audi R8«-Szene mit der »Sky 20«-HDR aus zwei unterschiedlichen Blickwinkeln und einer analogen Benennung zu Abbildung 18. Die Berechnungszeit betrug 17.8 h mit einer maximalen Sampleanzahl 2^{18} und einem maximalen relativen Fehler von 1 %. Zu beachten ist, dass die adaptive Irradianzschätzung im Inneren der Szene zu einem rauschfreien Resultat führt.

Ein komplexeres Beispiel anhand der »Audi R8«-Szene mit der »Sky 20«-HDR ist in den Abbildungen 34 und 35 aus Anhang C und der Abbildung 21 zu sehen. In der Szene wurden diverse zusammengesetzte BSDFs verwendet um einen realistischeren Eindruck zu vermitteln. Für die Scheiben und Scheinwerfer wurde ein Glas-Material verwendet, welches die ideale Brechung von Licht für einen gegebenen Brechungsindex zulässt. Vor allem im Inneren der Szene in Abbildung 21 werden extrem viele Strahlen benötigt. In diesen Bereichen kann Licht, welches durch die Umgebungsbeleuchtung ausgesendet wird, nur durch die Transmission von Strahlen an den Glasscheiben eingesammelt werden. Da aber auch die Transmission als zufälliger Prozess betrachtet wird, ist die Wahrscheinlichkeit, Licht einzusammeln, äußerst gering. Die Folge ist, dass sowohl bei der Irradianzschätzung als auch beim Path Tracing mit sehr hohen relativen Fehlern beziehungsweise starkem Rauschen zu rechnen ist. Außerhalb der Szene werden eigentlich nur in den Radkästen und unter dem Auto viele Samples benötigt. Ein Großteil der Strahlen trifft nach einer Reflexion direkt auf die Umgebungsbeleuchtung, was wieder einen sehr kleinen relativen Fehler zur Folge hat.

In allen Messungen konnten durch die Verwendung einer oberen relativen Fehlerschranke von 1 % und einer maximalen Sampleanzahl von 2^{18} die zufälligen Fehler der Schätzung stark verringert werden, sodass das menschliche Auge nicht mehr in der Lage war, diese zu erkennen. Auch in sehr dunklen Bereichen war die Verwendung von 256 Samples als Mindestanzahl vollkommen ausreichend. Dennoch sind die Berechnungszeiten in allen Ergebnissen viel größer als die Darstellungszeit des Path Tracing Bildes. Eine der Ursachen besteht in der Konvergenzordnung des adaptiven Irradianzschätzers. Aus Abschnitt 3.3 ist bekannt, dass sich die Varianz dieses Schätzers aus der folgenden Gleichung mit den zuvor eingeführten Variablen für ein $i \in \mathbb{N}, i \leq n$ ergibt.

$$\text{var } \bar{\mathcal{R}} = \frac{4\pi^2}{n} \text{ var } \mathcal{R}_i$$

Die Standardabweichung und damit auch der relative Fehler des Schätzers konvergieren also nur mit der Laufzeitkomplexität $\Theta_n \left(\frac{1}{\sqrt{n}} \right)$ gegen Null. Die Halbierung des Fehlers fordert dementsprechend viermal so viele Samples wie zuvor [38, S. 39 f]. Um diese Eigenschaft zu verbessern, wurde eine Vielzahl von Varianzreduktionsmethoden entwickelt, wie zum Beispiel »Stratified Sampling« [34, S. 432 ff], »Importance Sampling« [34, S. 794 ff] und »Quasi-Monte-Carlo Integration« [20]. Viele dieser Verfahren sind zwar auf eine konstante Sampleanzahl ausgelegt, könnten aber durch Modifikationen für eine adaptive Irradianzschätzung verwendet werden. Zum Beispiel würde es sich bei Stratified Sampling als Vorteil erweisen eine konstante Anzahl von »Stratas« zu verwenden und durch eine unabhängige Stichprobe deren Gewichtung zu ermitteln. Die Schätzung der Irradianz mithilfe dieser Gewichte würde zu einer geringeren benötigten Gesamt-sampleanzahl führen und so das Verfahren beschleunigen. Eine weitere Herangehensweise ist die Abänderung der Interpolation zwischen Punkten. Anstatt für jeden Punkt ein lineares Verfahren zu verwenden, könnte man einen komplexeren Filter oder auch Kern (engl.: *kernel*) durch Berechnung der Faltung mit den geschätzten Irradianzwerten benutzen [34, S. 402 ff]. Dies könnte dazu führen, dass sich zufällige Fehler nicht durch sichtbare kleine Flecken auf der Oberfläche auswirken, sondern durch nicht-wahrnehmbare Farbverläufe. Damit wäre man nicht gezwungen eine obere Fehlerschranke von 1 % zu verwenden und demnach würde auch dieser Ansatz das Verfahren beschleunigen. Eine Änderung der Interpolation würde den Algorithmus der adaptiven Schätzung jedoch nicht abändern.

4 Aufbau und Generierung der Irradiance Map

Dieses Kapitel wird die eigentliche Datenstruktur und den Generator der Irradiance Map einführen. Durch die bisher genannten Definitionen und Verfahren ist man in der Lage, die Irradianz einer Szene für beliebige Oberflächenpunkte und Wellenlängen bis zu einem gewissen Fehler zu ermitteln. Es wird darum gehen, eine endliche Anzahl von Punkten auf der Oberfläche auszuwählen und deren Irradianz zu bestimmen. Aus den erhaltenen Werten soll dann eine einfach zu bestimmende approximierte Irradianz erstellt werden. Wenn nicht anders behauptet, verwenden wir in diesem Kapitel wieder die bereits eingeführten Variablen des vorigen Kapitels.

4.1 Ein Rückblick auf Vertex Lighting

Im letzten Kapitel wurde die Irradianz R der Szene Σ nur an den Eckpunkten der Mesh evaluiert und gespeichert. Für Punkte im Inneren von Dreiecken führen wir eine lineare Interpolation durch. Die Approximation $\tilde{R}(x, \lambda)$ des Wertes $R(x, \lambda)$ für einen Punkt $x \in \mathcal{T}$, der in einem Dreieck (A, B, C) mit den baryzentrischen Koordinaten (u, v, w) lag, und einer Wellenlänge $\lambda \in (0, \infty)$ beschrieben wir durch die folgende Definition.

$$\tilde{R}(x, \lambda) := wR(A, \lambda) + uR(B, \lambda) + vR(C, \lambda)$$

Dieses Verfahren wird auch »Vertex Lighting« genannt und findet häufige Anwendungen in der Computerspieleindustrie [19]. Vor allem bei der »Dragon«-Szene reichte diese Form der Interpolation aus, weil alle Dreiecke der Mesh klein genug waren, um den Verlauf der Irradianz gut zu beschreiben (siehe Abbildungen 31 und 18). Allerdings kam es bereits in der »Shaderball«-Szene zu ersten Interpolationsartefakten, wie in Abbildung 19 beim Übergang des Fußes in den Boden zu sehen ist. Solche Fehler treten immer dann auf, wenn zwischen zwei Eckpunkten eine unstetige oder extrem schnelle Änderung der Beleuchtung stattfindet. Ein typisches Beispiel dafür ist eine als Wand dienende Ebene, die die Szene in zwei Halbräume teilt. Abbildung 22 beschreibt das Phänomen anhand zweier Skizzen. Abbildung 23 zeigt es zudem in der Praxis für die »Cornell Box«-Szene. 23b weist im Inneren der Box im Gegensatz zu 23a fast keinen Schattenverlauf auf. 23c zeigt, dass die später eingeführte Irradiance Map in der Lage ist, diese Fehler zu kompensieren.

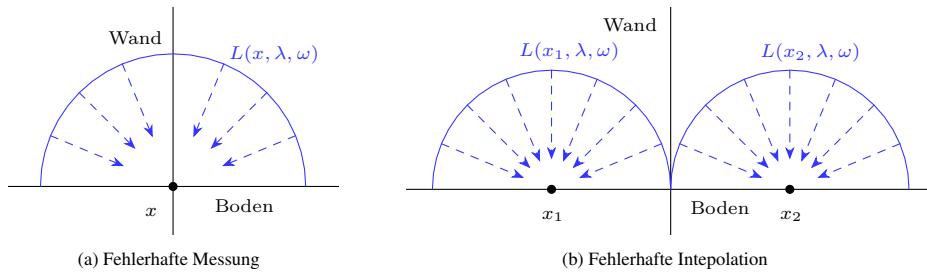


Abbildung 22: Die Skizzen erklären die Ursachen der beim Vertex Lighting auftretenden Flecken und Artefakte am Beispiel einer Szene, die durch eine Ebene zweigeteilt wird. In 22a befindet sich der Messpunkt x auf der Ebene. Er kann somit Licht von beiden Seiten der Ebene empfangen und wird in der späteren Interpolation keinen Schattenverlauf aufweisen. In 22b befinden sich die Messpunkte x_1 und x_2 auf jeweils einer Seite der Ebene. Beide empfangen damit das Licht nur von einer Seite. Die Interpolation zwischen den Punkten findet aber über die Wand hinweg statt, was zu sogenannten »Light Leaks« führt.

Die genannten Probleme des Vertex Lighting lassen sich nicht komplett durch eine approximierte Irradianz beheben. Jede Form der Interpolation benötigt eine endliche Anzahl von aufgenommenen Messpunkten. Für jede solcher Mengen von Messpunkten treten aber dieselben beiden Phänomene auf. Das Ziel besteht also darin den Fehler durch die Positionierung der Messpunkte sehr klein zu halten, da eine Beseitigung unmöglich ist. Im Falle die teilenden Wand aus Abbildung 22 würde dies bedeuten, die Punkte x_1 und x_2 so dicht wie möglich an die Wand, aber nicht auf die Wand zu setzen.

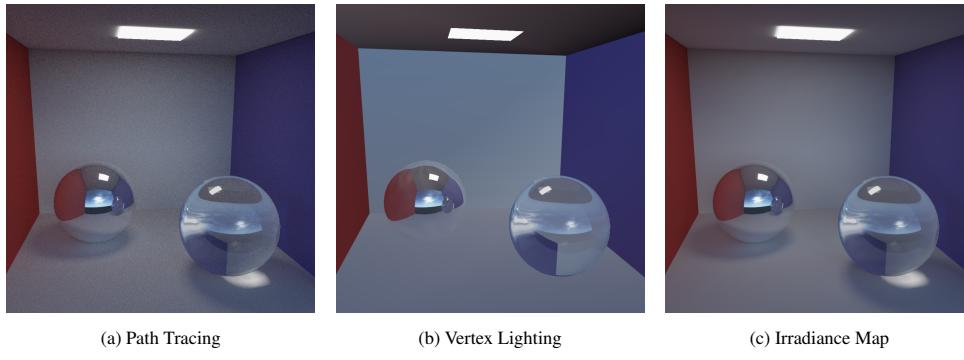


Abbildung 23: Die Bilder zeigen die »Cornell Box«-Szene mit der »Sky 10«-HDR unter Verwendung von Path Tracing als Referenzbild, von Vertex Lighting und der Irradiance Map. Aufgrund der geringen Auflösung der Mesh ist Vertex Lighting nicht in der Lage die Irradianz der Szene ausreichend zu approximieren. Für die adaptive Irradianzschätzung wurden eine obere Fehlerschranke von 1 % und maximal 2^{18} Samples verwendet. Die Irradiance Map wurde durch circa 278000 Messpunkte bestimmt, die insgesamt 1.06 MiB einnehmen. Die Berechnungszeit betrug 5120 s.

4.2 Datenstruktur der Irradiance Map

Um genauere Approximationen zu ermöglichen, benötigen wir nicht nur Messpunkte an den Eckpunkten der Dreiecke, sondern auch im Inneren dieser. Eine Variante die Anforderung zu erfüllen, besteht in der Verwendung von »Mesh Colors« [45]. Bei diesem Verfahren speichert man die aufgenommenen Werte an regelmäßigen Punkten des Dreiecks. Für jedes Dreieck gibt es dabei eine gewisse Freiheit in der Wahl der Anzahl dieser Punkte, die wir »Ordnung« nennen. Das führt dazu, dass nur dann viele Messpunkte gespeichert werden, wenn in einem Dreieck eine starke Änderung des Irradianzverlaufs auftritt. Insbesondere erlaubt die regelmäßige Verteilung der Messpunkte einen schnellen und einfachen Zugriff auf diese durch Verwendung der baryzentrischen Koordinaten eines Punktes. Beim Raytracing ist es üblich und meistens auch nötig die baryzentrischen Koordinaten des Schnittpunktes eines Strahls mit einem Dreieck zu berechnen [24]. Die Datenstruktur eignet sich damit vor allem für Algorithmen, die, wie Path Tracing in unserem Falle, auf Raytracing basieren. In Abbildung 24 werden die Verteilungen der Messpunkte für verschiedene Ordnungen gezeigt.

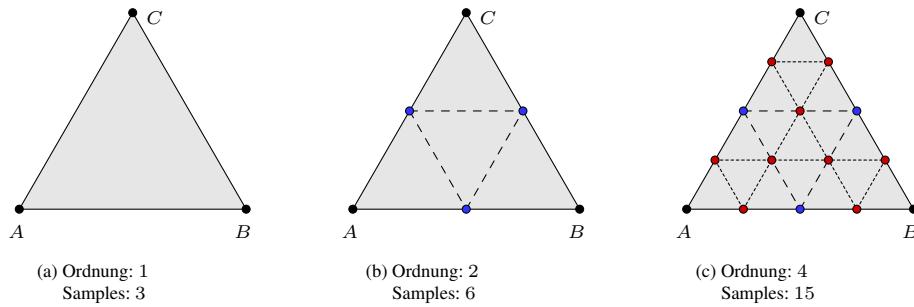


Abbildung 24: Die Skizzen zeigen die Datenstruktur der Irradiance Map auf einem Dreieck (A, B, C) für verschiedene Ordnungen. 24a speichert Messpunkte nur an den Eckpunkten des Dreiecks. 24b speichert zusätzlich noch Messpunkte auf dessen Kante. Erst für Ordnungen größer gleich 3 werden Messpunkte auch im Inneren des Dreiecks aufgenommen, wie in 24c zu sehen ist.

In Abbildung 24 wird deutlich, dass einige Messpunkte direkt auf den Kanten und Ecken der Dreiecke gespeichert werden. Um aber die Fehler des Vertex Lighting aus Abbildung 22 zu minimieren, sollte eine Evaluierung der Messwerte nicht an Kanten und Ecken erfolgen. Um dies zu gewährleisten, werden wir später die Messwerte im Inneren des Dreiecks schätzen und dann am Rand oder den Ecken speichern. Dies wird einen systematischen Fehler der Approximation zur Folge haben. Die Verschiebung muss demnach im Vergleich zur Größe des Dreiecks klein sein. Die folgende Definition führt das Schema der Abbildung 24 genauer ein und beschreibt zusätzlich eine Abbildung in den Speicher des

Computers.

DEFINITION 4.1: (Irradiance Map auf Dreieck)

Es seien ein Dreieck Δ gegeben durch die Sequenz (A, B, C) in \mathbb{R}^3 und $n \in \mathbb{N}$. Dann definiert man die Ordnungsfunktion s , die Grundindexmenge U_n , die Speicherindexmenge I_n , die Speicherindex-Funktion m_n und die Positionsfunction p_n wie folgt.

$$s: \mathbb{N} \rightarrow \mathbb{N}, \quad s(k) := \frac{(k+1)(k+2)}{2}$$

$$U_n := \{(u, v) \in \mathbb{N}_0^2 \mid u + v \leq n\}, \quad I_n := \{i \in \mathbb{N}_0 \mid i < s(n)\}$$

$$m_n: U_n \rightarrow I_n, \quad m_n(u, v) := \frac{(u+v)(u+v+1)}{2} + v$$

$$p_n: U_n \rightarrow \Delta, \quad p_n(u, v) := \Delta \left(\frac{u}{n}, \frac{v}{n} \right)$$

Eine Irradiance Map der Ordnung n auf Δ ist gegeben durch eine Abbildung $f: \text{im } p_n \rightarrow \Delta$. Wir nennen $s(n)$ die Größe von f .

Der Definitionsbereich einer Irradiance Map besteht genau aus den markierten Punkten des Dreiecks aus Abbildung 24. U_n, I_n und m_n bieten eine Möglichkeit diese Menge durch Index-uv-Koordinaten zu indizieren und in den Speicher abzubilden. Um eine explizite Vorstellung dieser drei Konstrukte zu bekommen, zeigt Abbildung 25 sie anhand der Ordnungen 2 und 4. p_n transformiert die jeweiligen Index-uv-Koordinaten in eine Position auf dem Dreieck.

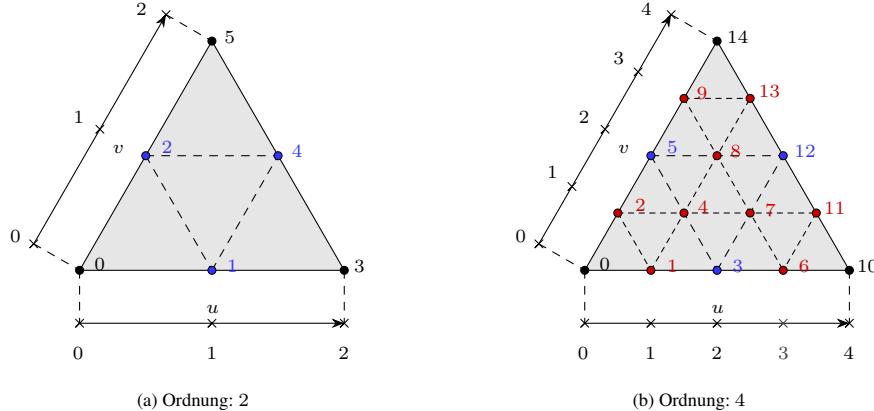


Abbildung 25: Die Skizzen zeigen die Datenstruktur der Irradiance Map auf einem Dreieck für verschiedene Ordnungen $n \in \mathbb{N}$. Neben jedem Messpunkt $x \in \text{im } p_n$ befindet sich der zugehörige Speicherindex $m_n(p_n^{-1}(x))$. Für die Bestimmung der Index-uv-Koordinaten wurden deren Koordinatenachsen eingezeichnet.

Im noch folgenden Irradiance Map Generator wollen wir nach der Bestimmung einer Irradiance Map der Ordnung $n \in \mathbb{N}$ die Möglichkeit haben, diese durch weitere Messpunkte zu verfeinern, sodass die vorherigen Messpunkte nicht umsonst aufgenommen wurden. Solch eine Verfeinerung oder auch Fortsetzung ist nicht für beliebige zwei Ordnungen möglich. In Abbildung 25 ist jedoch zu sehen, dass die Verdopplung der Ordnung eine derartige Möglichkeit bietet. Es ist auch zu sehen, dass zum Beispiel der Messpunkt mit Index 4 der Ordnung 2 auf den Index 12 der Ordnung 4 abbgebildet werden muss. Auch die Verfeinerung und die Verschiebung des Speicherindex wollen wir durch die folgende Definition spezifizieren.

DEFINITION 4.2: (Irradiance Map Verfeinerung)

Sei f eine Irradiance Map der Ordnung $n \in \mathbb{N}$ auf einem Dreieck Δ . Sei weiterhin $r \in \mathbb{N}$. Dann ist eine Irradiance Map g der Ordnung $n \cdot 2^r$ auf Δ eine Verfeinerung von f , wenn

$$g|_{\text{im } p_n} = f$$

Die verfeinerte Speicherindex-Funktion ist durch die folgende Definition gegeben.

$$m_{n,r}: U_n \rightarrow I_{n \cdot 2^r}, \quad m_{n,r}(u, v) := m_{n \cdot 2^r}(u \cdot 2^r, v \cdot 2^r)$$

Der Verfeinerungsgrad $r \in \mathbb{N}$ in Abbildung 25 beträgt gerade 1, da $2 \cdot 2 = 4$. Ab einem bestimmten Verfeinerungsgrad wird man genügend Messpunkte aufgenommen haben, sodass die Fehler der Irradianzapproximation auf dem Dreieck gering sein sollten. Die Konstruktion der Approximation kann, wie im letzten Kapitel erwähnt, durch verschiedene Interpolationsmethoden und Filter stattfinden. Eine allgemeine Formulierung gibt folgende Definition.

DEFINITION 4.3: (Approximierende Irradiance Map Fortsetzung)

Sei f eine Irradiance Map der Ordnung $n \in \mathbb{N}$ auf einem Dreieck Δ . Dann nennen wir eine Fortsetzung $g: \Delta \rightarrow [0, \infty)$ von f auf Δ eine approximierende Fortsetzung von f .

Man stellt die Interpolation als eine Fortsetzung der Irradiance Map dar. So ist die Datenstruktur selbst unabhängig von der genauen Interpolationsmethode. In unserem Falle wird es sich allerdings nur um eine lineare Interpolation in einer Dreieckszelle handeln. Man kann sich diese Form der Approximation als eine Verallgemeinerung des Vertex Lighting vorstellen. Die gegebene Definition wird durch Abbildung 26 veranschaulicht.

DEFINITION 4.4: (Lineare approximierende Irradiance Map Fortsetzung)

Seien f eine Irradiance Map der Ordnung $n \in \mathbb{N}$ auf einem Dreieck Δ und g eine approximierende Fortsetzung von f , sodass für alle $x \in \Delta$ mit den baryzentrischen Koordinaten (u, v, w) das Folgende gilt.

$$g(x) := \begin{cases} (1 - u_w)f(x_3) + (1 - v_w)f(x_1) + (u_w + v_w - 1)f(x_2) & : u_w + v_w > 1 \\ u_w f(x_1) + v_w f(x_3) + (1 - u_w - v_w)f(x_0) & : \text{sonst} \end{cases}$$

$$\bar{u} := \lfloor nu \rfloor, \quad \bar{v} := \lfloor nv \rfloor, \quad u_w := nu - \bar{u}, \quad v_w := nv - \bar{v}$$

$$x_0 := p_n(\bar{u}, \bar{v}), \quad x_1 := p_n(\bar{u}+1, \bar{v}), \quad x_2 := p_n(\bar{u}+1, \bar{v}+1), \quad x_3 := p_n(\bar{u}, \bar{v}+1)$$

Zu beachten ist, dass die Interpolation nur in einem Dreieck stattfindet und nicht auch über mehrere Dreiecke hinweg. Für den Fall der linearen Interpolation ist dies auch ausreichend, da wir davon ausgehen, dass Dreiecke, die eine Oberfläche approximieren, an Kanten und Eckpunkten dieselben Messwerte aufweisen oder im sonstigen Fall keine Interpolation stattfinden darf. Möchte man aber kompliziertere Interpolationsmethoden verwenden, so ist die Information über Messpunkte in benachbarten Dreiecken unabdingbar. Hierfür empfiehlt es sich eine temporäre Datenstruktur zu erzeugen, die die Nachbarschaftsinformationen zwischenspeichert. Aus der mit einem anderem Filter erzeugten Irradiance Map lässt sich durch die nochmalige Generierung einer weiteren Irradiance Map mit linearem Filter

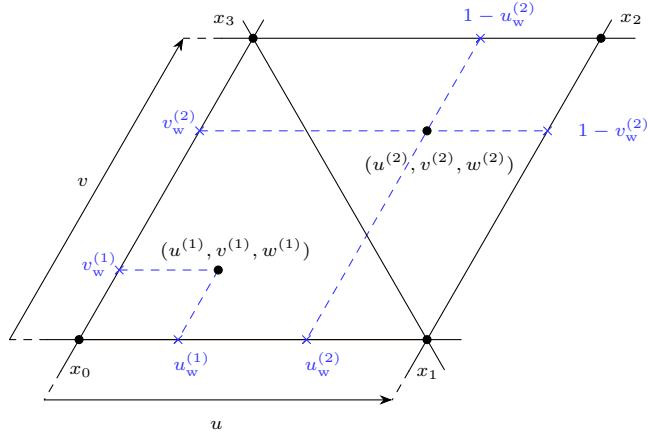


Abbildung 26: Die Skizze zeigt die in der Definition der Linearen approximierenden Irradiance Map eingeführten Variablen, um deren Bedeutung zu erläutern. Die Punkte mit den baryzentrischen Koordinaten $(u^{(i)}, v^{(i)}, w^{(i)})$, $i \in \{1, 2\}$ befinden sich im Inneren eines Dreiecks mit Ordnung $n \in \mathbb{N}$. Ein Punkt liegt immer in einer Dreieckszelle. Hier sind das entweder (x_0, x_1, x_3) oder (x_2, x_3, x_1) . Die Interpolation wird nur durch die Eckpunkte dieser Zelle und den erhaltenen Gewichten $u_w^{(1)}, v_w^{(1)}, u_w^{(2)}, v_w^{(2)}$ berechnet.

eine ähnliche Approximation erzielen, die die Nachbarschaftsinformationen nicht mehr benötigt. Dieses Verfahren eignet sich vor allem dann, wenn die Generierung von Messpunkten, wie in unserem Falle bei der Irradianzmessung, aufwendig ist und viel Zeit in Anspruch nimmt. Der folgende »C++«-Quelltext implementiert die Irradiance Map Datenstruktur für ein einzelnes Dreieck unter Verwendung aller bisher eingeführten Definitionen.

Quelltext: Dreieck Irradiance Map Datenstruktur

```

namespace irr_map{

    // Irradiance Map Datenstruktur
    struct triangle_uvmap{
        uint order; // Ordnung
        vecf4 *data; // Messwerte

        vecf4 operator()(float u, float v) const; // lineare Interpolation
    };

    // Ordnungsfunktion
    inline uint size(uint order){
        return ((order+1)*(order+2))>>1;
    }

    // Speicherindex-Funktion
    inline uint memidx(uint uidx, uint vidx){
        const uint sum = uidx + vidx;
        return ((sum * (sum+1)) >> 1) + vidx;
    }

    // verfeinerte Speicherindex-Funktion
    inline uint memidx(uint uidx, uint vidx, uint shift){
        const uint suidx = uidx << shift;
        const uint svidx = vidx << shift;
        return memidx(suidx, svidx);
    }

    // lineare Interpolation
    inline vec4f triangle_uvmap::operator()(float u, float v) const{
}
}

```

```

const float nu = order * u;
const float nv = order * v;
const float uidx = floorf(nu);
const float vidx = floorf(nv);
const float wu = nu - uidx;
const float vv = nv - vidx;
const float sum_w = wu + vv;

if (sum_w > 1.0f) {
    const uint idx = memidx(uidx+1, vidx);
    const uint sum = uidx + vidx + 1;

    return (1.0f-wu) * data[idx + 1] + // memidx(u, v+1)
           (1.0f-vv) * data[idx] + // memidx(u+1, v)
           (sum_w-1.0f) * data[idx + sum + 2]; // memidx(u+1, v+1)
} else{
    const uint idx = memidx(uidx, vidx);
    const uint sum = uidx + vidx;

    return wu * data[idx + sum + 1] + // memidx(u+1, v)
           vv * data[idx + sum + 2] + // memidx(u, v+1)
           (1.0f-sum_w) * data[idx]; // memidx(u, v)
}
}

} // irr_map

```

4.3 Irradiance Map Generator

Die Generierung der Irradiance Map erfolgt durch die Erzeugung der Irradiance Map für jedes Dreieck. Da die Szenen im Allgemeinen sehr viele Dreiecke besitzen, ist diese Operation leicht parallelisierbar.

Die Grundidee bei der Erzeugung besteht in der Verwendung der Verfeinerung einer Irradiance Map. Damit man Punkte nicht mehrmals messen muss, behält man die bereits berechneten Messwerte und verdoppelt die Ordnung. Für die neu hinzukommenden Messpunkte führt man wieder die Irradianzschätzung durch und wiederholt das Verfahren. Auch hier gibt es wieder eine maximale Ordnung, um die Berechnungszeit zu beschränken. Um das Verfahren früher abzubrechen, benötigen wir ein Fehlermaß, welches klein bei ausreichenden Approximationen ist.

Geht man davon aus, dass die Irradianz bezüglich der Wellenlänge $\lambda \in (0, \infty)$ auf dem Dreieck eine gleichmäßig stetige Funktion ist, so konvergiert eine Folge von linear approximierenden Irradiance Map Fortsetzungen $(g_n)_{n \in \mathbb{N}}$ der Irradiance Maps $f_n, n \in \mathbb{N}$, sodass f_{n+1} eine Verfeinerung von f_n für alle $n \in \mathbb{N}$ darstellt, punktweise gegen diese. $(g_n(x))_{n \in \mathbb{N}}$ ist damit für jeden Punkt $x \in \Delta$ auch eine Cauchy-Folge. Insbesondere gelten dann die folgenden Aussagen für ein $\varepsilon \in (0, \infty)$.

$$\begin{aligned}
g_n &\xrightarrow{n \rightarrow \infty} R(\cdot, \lambda)|_{\Delta}, & |g_n(x) - g_m(x)| &\xrightarrow{n, m \rightarrow \infty} 0 \\
\sup_{x \in \Delta} |g_n(x) - g_m(x)| &\xrightarrow{n, m \rightarrow \infty} 0, & \sup_{x \in \Delta} \frac{|g_n(x) - g_m(x)|}{R(x, \lambda) + \varepsilon} &\xrightarrow{n, m \rightarrow \infty} 0
\end{aligned}$$

In der letzten Aussage tritt wieder ein relativer Fehler auf, der in seiner Funktionsweise dem modifizierten Variationskoeffizienten aus Abschnitt 3.4 sehr ähnelt. Auch hier wollen wir die Approximation, aufgrund der in diesem Abschnitt genannten Tatsachen, durch einen relativen Fehler bewerten. Da man $R(x, \lambda)$ nicht für alle $x \in \Delta$ kennt, betrachten wir den folgenden relativen Fehler e_{n+1} einer Funktion g_{n+1} für $n \in \mathbb{N}$.

$$e_{n+1} := \sup_{x \in \text{im } p_{n+1}} \frac{|g_{n+1}(x) - g_n(x)|}{R(x, \lambda) + \varepsilon} \xrightarrow{n \rightarrow \infty} 0$$

e_{n+1} lässt sich vergleichsweise einfach bestimmen und stellt ein gutes Fehlermaß für die Bewertung der Approximation dar. Im Algorithmus werden wir eine Fehlerschranke definieren, die angibt, wann eine berechnete Verfeinerung ausreicht, um so die Rechenzeit weiter zu verkürzen. Zu beachten ist, dass mindestens g_2 konstruiert werden muss, da e_1 nicht definiert ist und keinen Sinn ergeben würde.

Ein wichtiges Problem besteht allerdings darin, dass sich $R(\cdot, \lambda)$ nur an wenigen Punkten ermitteln lässt, ohne die Berechnungszeit stark zu erhöhen. Für hochfrequente Änderungen von $R(\cdot, \lambda)|_{\Delta}$ kann es demnach passieren, dass der relative Fehler eine Approximation gut bewertet, während dies aber nicht der Fall ist. Um das Problem zu beheben, ist es entweder notwendig, extrem viele Punkte in Δ aufzunehmen, um ein besseres Fehlermaß zu erhalten, oder durch weitere einfache Fehlermaße, wie zum Beispiel $d_{n+1} := \frac{e_{n+1}}{e_n}$, $n \in \mathbb{N}$, einige solcher auftretenden Fehler zu beseitigen. Die Aufnahme einer sehr viel größeren Zahl von Messpunkten, wird zu einem sehr genauen Fehlermaß führen, durch welches eine klare und unverfälschte Bewertung der Approximation möglich ist. Auf der anderen Seite ist das Messen der Irradianz ein aufwendiger Prozess, sodass die Berechnung des Fehlers einen immensen Anteil der Rechenzeit einnehmen würde. Die Anwendung weiterer Fehlermaße ist meistens einfach zu implementieren und behebt diverse entstehende Fehler. Dennoch neigen mehrere Fehlermaße dazu, eine Approximation zu schlecht zu bewerten und resultieren damit auch wieder in einer höheren Berechnungszeit.

Das beschriebene Verfahren wurde im folgenden »C++«-Quelltext implementiert.

Quelltext: Irradiance Map Generator

```

namespace irr_map{

vecf4 irr(triangle* prim, float u, float v){
    const float w = 1.0f - u - v;

    pos =
        prim->v0.pos * w +
        prim->v1.pos * u +
        prim->v2.pos * v;

    n = prim->v0.n * w +
        prim->v1.n * u +
        prim->v2.n * v;

    vecf4::normalize(n);

    uint sample_count;
    return est_irr_adap(pos, n, sample_count);
}

void gen_irr_map(){
    const float rel_err_eps = 0.0001f;
    const float border_eps = 0.00001f;
    const float rel_err_bound = ctrlWnd->irrmap_dsb->value();

    const float samples_on_edge = ctrlWnd->sample_diag_dsb->value() * prim->edge
        / bvh->edge_mean;
    const float cur_max_order = samples_on_edge+1.0f;
    const uint cur_max_exp = max(ceilf(log2f(cur_max_order)), 1.0f);

    // const uint max_exp = 7;
    const uint read_max_exp = ctrlWnd->irrmap_max_order_sb->value();
    const uint max_exp = (cur_max_exp < read_max_exp)?(cur_max_exp):(read_max_exp
    );
    uint max_order = 1 << max_exp;
    const uint half_max_order = max_order >> 1;
    const uint max_size = size(max_order);
}

```

```

const uint half_max_size = size(half_max_order);

// memory indices of vertices
const uint a = 0;
const uint b = max_size-half_max_order-1;
const uint c = max_size-1;

// memory indices of half edge samples
const uint eab = half_max_size-half_max_order-1;
const uint ebc = max_size-half_max_order-1;
const uint eca = half_max_size-1;

vecf4 data[max_size];
triangle* prim;

while(prim = next_triangle()){
    uint cur_exp = 1;
    uint cur_order = 1 << cur_exp;

    // vertices
    data[a] = irr(prim, border_eps, border_eps);
    data[c] = irr(prim, border_eps, 1.0f - 2.0f * border_eps);
    data[b] = irr(prim, 1.0f - 2.0f*border_eps, border_eps);

    // edges
    data[eca] = irr(prim, border_eps, 0.5f);
    data[eab] = irr(prim, 0.5f, border_eps);
    data[ebc] = irr(prim, 0.5f - border_eps, 0.5f - border_eps);

    float rel_err = max(fabsf( 0.5f*(data[a] + data[b]) - data[eab] ) / (
        rel_err_eps + data[eab]));
    rel_err = max(rel_err, max(fabsf( 0.5f*(data[b] + data[c]) - data[ebc] ) /
        (rel_err_eps + data[ebc])));
    rel_err = max(rel_err, max(fabsf( 0.5f*(data[c] + data[a]) - data[eca] ) /
        (rel_err_eps + data[eca])));

    while (rel_err > rel_err_bound && cur_exp < max_exp){
        cur_exp++;
        cur_order = 1 << cur_exp;
        float inv_cur_order = 1.0f / float(cur_order);

        // compute new samples and residual norm
        const uint shift = max_exp - cur_exp;

        // edges
        for (uint i = 1; i < cur_order; i+=2){
            const float tmp = float(i) * inv_cur_order;

            uint idx = memidx(i,0,shift);
            data[idx] = irr(prim, tmp, border_eps);

            idx = memidx(0,i,shift);
            data[idx] = irr(prim, border_eps, tmp);

            idx = memidx(i,cur_order-i,shift);
            data[idx] = irr(prim, tmp-border_eps, 1.0f-tmp-border_eps);

            rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(i-1,0,shift)
                ] + data[memidx(i+1,0,shift)]) - data[memidx(i,0,shift)] ) /
                (rel_err_eps + data[memidx(i,0,shift)))));

            rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(0,i-1,shift)
                ] + data[memidx(0,i+1,shift)]) - data[memidx(0,i,shift)] ) /
                (rel_err_eps + data[memidx(0,i,shift)))));
        }
    }
}

```

```

        rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(i-1,
            cur_order-i+1,shift)] + data[memidx(i+1,cur_order-i-1,shift)
        ]) - data[memidx(i,cur_order-i,shift)] ) / (rel_err_eps +
            data[memidx(i,cur_order-i,shift)])));
    }

    // inner
    // odd i
    for (uint i = 1; i < cur_order; i+=2){
        // odd j
        for (uint j = 1; i+j < cur_order; j+=2){
            const float u = float(i) * inv_cur_order;
            const float v = float(j) * inv_cur_order;
            data[memidx(i,j,shift)] = irr(prim, u, v);

            rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(i-1,j
                +1,shift)] + data[memidx(i+1,j-1,shift)]) - data[memidx(
                    i,j,shift)] ) / (rel_err_eps + data[memidx(i,j,shift)]))
            );
        }
    }

    // even j
    for (uint j = 2; i+j < cur_order; j+=2){
        const float u = float(i) * inv_cur_order;
        const float v = float(j) * inv_cur_order;
        data[memidx(i,j,shift)] = irr(prim, u, v);

        rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(i,j-1,
            shift)] + data[memidx(i,j+1,shift)]) - data[memidx(i,j,
            shift)] ) / (rel_err_eps + data[memidx(i,j,shift)])));
    }
}

// even i
for (uint i = 2; i < cur_order; i+=2){
    // odd j
    for (uint j = 1; i+j < cur_order; j+=2){
        const float u = float(i) * inv_cur_order;
        const float v = float(j) * inv_cur_order;
        data[memidx(i,j,shift)] = irr(prim, u, v);

        rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(i-1,j,
            shift)] + data[memidx(i+1,j,shift)]) - data[memidx(i,j,
            shift)] ) / (rel_err_eps + data[memidx(i,j,shift)])));
    }
}

const uint shift = max_exp - cur_exp;
prim->irr_map.order = cur_order;
prim->irr_map.data = (vecf4*)mem_alloc(sizeof(cur_order) * sizeof(vecf4));

uint idx = 0;
for (uint r = 0; r <= cur_order; r++){
    for (uint j = 0; j <= r; j++){
        const uint i = r - j;
        prim->irr_map.data[idx] = data[memidx(i,j,shift)];
        idx++;
    }
}
}
}

```

```
    } // irr_map
```

Die Abbildungen 23, 36, 37, 27, 30, 28 und 29 zeigen die Ergebnisse des Algorithmus für ausgewählte Szenen. In allen Fällen übertrifft die Approximation durch die Irradiance Map das Vertex Lighting Verfahren. Deutlich wird dies in Abbildung 36. Eine direktionale Lichtquelle lässt sich aufgrund der Unstetigkeit der entstehenden Irradianz im Allgemeinen nur schlecht durch Messpunkte interpolieren. Bild 36b ist durch eine zu geringe Auflösung der Mesh nicht in der Lage, den Kernschatten des Objektes akzeptabel zu approximieren. Die Irradiance Map in Bild 36c entspricht aber dem Anschein nach dem Referenzbild 36a. Erst durch Heranzoomen in Abbildung 37 erkennt man kleine Aliasing-Artefakte am Rand des harten Schattens. Die Darstellung der Ordungen der einzelnen Dreieck Irradiance Maps zeigt zudem, dass der Algorithmus nur dann eine höhere Ordung verwendet, wenn eine Änderung der Irradianz, die größer als die gegebene Fehlerschranke ist, gemessen wird. Auch für einen vergleichsweise glatten Schattenverlauf in den Abbildungen 27, 30, 28 und 29 erzielt die Irradiance Map bessere Ergebnisse, ohne für jedes Dreieck extrem viele Samples zu verwenden. Die durch Vertex Lighting entstehenden Artefakte sind zwar schwach ausgeprägt, können aber dennoch im Vergleich mit den Referenzbildern beobachtet werden. In allen Ergebnissen besaß das Vertex Lighting Verfahren jedoch einen großen Vorteil gegenüber der Irradiance Map. Die Berechnungszeit von Vertex Lighting betrug immer weniger als 5 % der Irradiance Map Generierungszeit.

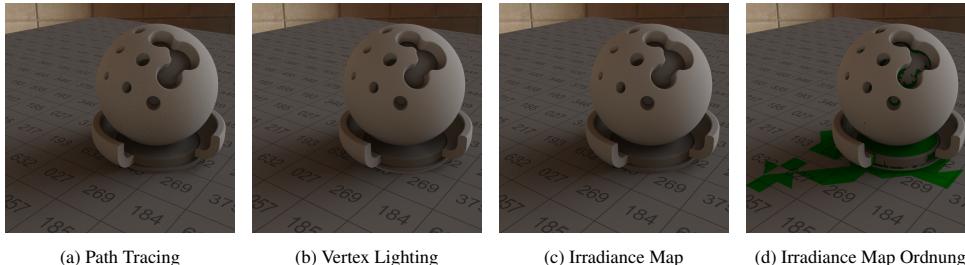


Abbildung 27: Die Bilder zeigen die »Shaderball«-Szene mit der »Ennis-Brown House«-HDR unter Verwendung von Path Tracing als Referenzbild, von Vertex Lighting und der Irradiance Map. Für die adaptive Irradianzschatzung wurden eine obere Fehlerschranke von 1 % und maximal 2^{16} Samples verwendet. Die Irradiance Map wurde durch circa $9.27 \cdot 10^6$ Messpunkte bestimmt, die insgesamt 35.4 MiB einnehmen. Die obere Fehlerschranke betrug 10 % und die maximale Ordnung 2^7 . Die Generierung benötigte 3430 s. Die grün markierten Bereich in 27d stellen Dreiecke mit einer Ordnung größer 2 dar.

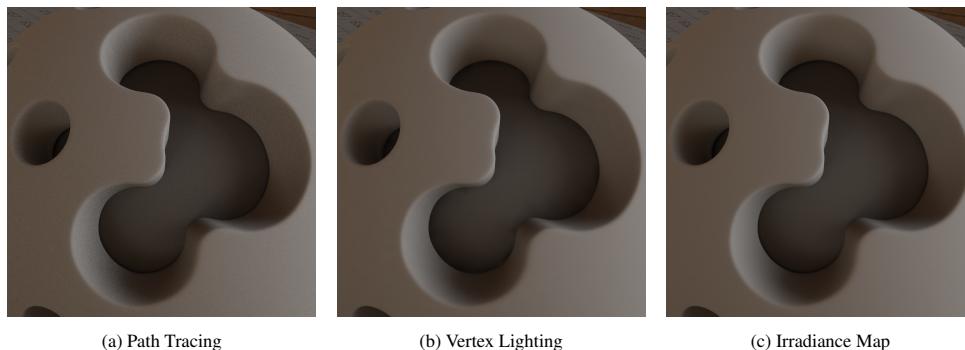


Abbildung 28: Die Bilder zeigen die oberen Löcher der »Shaderball«-Szene aus Abbildung 27. Die Schattenverläufe in 28a und 28c können nicht unterschieden werden. Für Vertex Lighting in 28b sind am Rand der Löcher Fehler in den Schatten zu sehen.



Abbildung 29: Die Bilder zeigen den Kernschatten der »Shaderball«-Szene aus Abbildung 27. Die Schattenverläufe in 29a und 29c können nicht unterschieden werden. Für Vertex Lighting in 29b sind vor allem am Rand des Schattens kleine Artefakte zu entdecken.

Die Probleme des verwendeten Fehlermaßes wurden bereits beschrieben. Abbildung 37 zeigt das Resultat einer Fehlbewertung. Eine der Dreieck Irradiance Maps auf der unteren Platte approximiert den harten Schattenverlauf auf eine extrem schlechte Weise. Höchstwahrscheinlich konnten in diesem Dreieck aufgrund der schnellen Änderung der Irradianz nur Messpunkte im Kernschatten aufgenommen werden. Die Bewertung durch das Fehlermaß musste demnach positiv ausfallen und die Berechnung beenden. In der Praxis ist es allerdings vergleichsweise einfach, die Beleuchtung direktonaler oder anderer singulärer Lichtquellen durch Raytracing und Path Tracing zu simulieren. Die Verwendung einer Irradiance Map für singuläre Lichtquellen wie in den Abbildungen 36 und 37 ist damit ein unwahrscheinlicher Fall. Dennoch zeigt die Szene, dass es auch in schwierigeren Situationen möglich ist eine Irradiance Map einzusetzen.

Im derzeitigen Algorithmus wird für jedes Dreieck die Irradianz an Ecken und Kanten mit einer kleinen Verschiebung in das Innere geschätzt. So werden Fehler der Art aus Abbildung 22 verhindert, wie man in Abbildung 23 anhand der »Cornell Box«-Szene sehen konnte. Die Ergebnisse des Vertex Lighting aus Abschnitt 3.5 und Anhang C zeigen aber, dass diese Fehler nicht bei allen Eckpunkten und Kanten auftreten. Ein Großteil der Generierungszeit der Irradiance Map besteht also darin, bereits evaluierte Werte noch einmal zu schätzen. Durch eine Analyse der Mesh ist es aber möglich, jedem Eckpunkt und jeder Kante eine Markierung zu geben, die eine Aussage darüber trifft, ob die Irradianz auf dem Eckpunkt beziehungsweise der Kante geschätzte werden darf oder ob sie im Inneren der Dreieck bestimmt werden muss. Die Berechnungszeit könnte somit um ein Vielfaches beschleunigt werden.



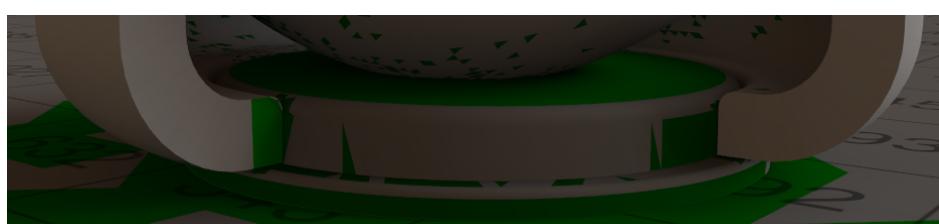
(a) Path Tracing



(b) Vertex Lighting



(c) Irradiance Map



(d) Irradiance Map Ordnung

Abbildung 30: Die Bilder zeigen den unteren Stand der »Shaderball«-Szene aus Abbildung 27. Die grün markierten Bereiche in 30d stellen Dreiecke mit einer Ordnung größer 2 dar. Die Schattenverläufe in 30a und 30c können nicht unterschieden werden. Für Vertex Lighting in 30b ist dies nicht der Fall.

5 Fazit und Aussicht

Das in der Arbeit angewendete Verfahren zur Schätzung der Irradianz an der Oberfläche der Mesh wurde zuerst nur mithilfe von Vertex Lighting implementiert. Für hochauflöste Meshes ist dieses Verfahren eine gute Approximation und brilliert durch seine Einfachheit. Da die meisten Szenen in dieser Arbeit diese Eigenschaft jedoch nicht erfüllen, musste eine Alternative, wie die Irradiance Map, gefunden werden.

Die adaptive Schätzung der Irradianz führt für eine vorgegebene relative Fehlerschranke und einer maximalen Sampleanzahl zu akzeptablen Ergebnissen. In den meisten Fällen konnten keine Unterschiede zum Referenzbild ausgemacht werden. Insbesondere wurde im Laufe der Untersuchung festgestellt, dass die besten Ergebnisse für eine relative Fehlerschranke von 1 % und einer maximalen Sampleanzahl von 2^{16} bis 2^{18} entstehen. Auftretende Fehler in diesem Bereich sind für das menschliche Auge kaum noch wahrnehmbar. Für sehr dunkle Bereiche der Darstellung sollte allerdings noch eine weitere Fehlermetrik verwendet werden, die die Ausbildung schwarzer Flecken in schwach beleuchteten Bereichen verhindert.

Die Datenstruktur der Irradiance Map ist für einen optimierten Raytracing-Algorithmus sehr gut geeignet, da die Szenen durch Dreiecke gegeben sind und damit eine natürliche Speichermethode für ein schnelles Auslesen der Irradianz-Werte Anwendung findet [24]. Light Leaks werden aufgrund der Speicherung auf der Oberfläche verhindert. Im Gegensatz zum Irradiance Caching lassen sich auch direkte Irradianzen annähern. Zu berücksichtigen bleibt jedoch die Entstehung neuer Artefakte aufgrund möglicher fehlerhafter Geometrien und Aliasing-Effekten.

Die Generierung der Irradiance Map funktioniert sowohl für direkte als auch indirekte Irradianzen. Allerdings ist hierbei eine Anpassung der Parameter notwendig. Bessere Ergebnisse werden im Allgemeinen für indirekte Irradianzen erzielt. In der Mehrzahl der Fälle wird das Rauschen vollständig eliminiert, sodass die entstehenden Bilder im Vergleich zu den Referenzbildern realistischer wirken.

Aufgrund der Tatsache, dass die Geometrie einer Szene im Allgemeinen nicht konvex ist, ist man bei der Berechnung der Irradiance Map dazu gezwungen die Messwerte innerhalb eines Dreiecks aufzunehmen. Diese Einschränkung verhindert die Benutzung von Vertex Lighting als Basis und führt zu einem höheren Rechenzeit. Die geringe Sampleanzahl, die man pro Dreieck verwendet, kann die Irradianzfunktion auf dem Dreieck nicht ausreichend interpolieren, sodass bei einzelnen Dreiecken der Schattenverlauf gar nicht oder falsch reproduziert wird.

Schlussfolgernd bleibt festzuhalten, dass das Verfahren zur Generierung der Irradiance Map zwar eine Möglichkeit der simultanen Berechnung der Strahldichte darstellt, aber aufgrund der großen Berechnungszeit, des hohen Speicheraufwandes, der unzureichenden Interpolation und den daraus resultierenden Beleuchtungsartefakten nur begrenzt effizient in der Praxis anwendbar ist. Dennoch sind wir durch geeignete Algorithmen in der Lage die Funktionsweise des Irradiance Map Generators zu verbessern. In den Quellen ... sind diverse Varianzreduktionsmethoden erläutert, die das Messen der Irradianz unter Anwendung der hier eingeführten Fehlermetrik durch eine geringere Anzahl von Samples ermöglichen. Dies würde zu einer erheblichen Beschleunigung des Verfahrens beitragen. In Quelle [33] wird das Verfahren des sogenannten »Irradiance Caching« beschrieben. Hierbei wird jedem Samplepunkt durch eine Abstandsmetrik eine maximale Interpolationsdistanz zugeordnet. Die Verwendung einer solchen Größe in der Irradiance Map kann zur Verringerung des Speicherverbrauches führen. Fortführend ist in [?] die Methode der »Irradiance Gradients« eingeführt worden, die ein alternatives Interpolationsschema zwischen aufgenommenen Samplepunkten der Irradianz auf Basis der restlichen Szenengeometrie erklären. Eine Abwandlung des Verfahrens für die Irradiance Map könnte in der Lage sein, die Interpolationsartefakte ohne größeren Rechenaufwand zu minimieren und damit auch zu einer Beschleunigung des gesamten Verfahrens beitragen.

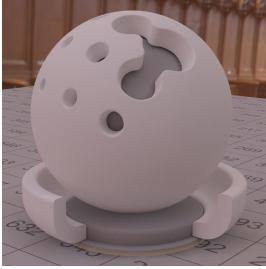
Literatur

- [1] Akenine-Möller, Tomas, Eric Haines, and Naty Hoffman: *Real-Time Rendering*. A K Peters, Ltd., third edition, 2008.
- [2] Blochi: *Basketball court, sibl archive*, Aug 2009. <http://hdrlabs.com/sibl/archive.html>.
- [3] Botsch, Mario, Mark Pauly, and C. Ross: *Geometric modeling based on triangle meshes*. Acm Siggraph 2006 ..., pages 1–148, 2006. <http://dl.acm.org/citation.cfm?id=1185839>.
- [4] Bronstein, Semendjajew, Musiol und Mühlig: *Taschenbuch der Mathematik*. Europa-Lehrmittel, zehnte Auflage, 2016, ISBN 978-3-8085-5789-1.
- [5] Cohen, Michael F. and John R. Wallace: *Radiosity and Realistic Image Synthesis*. Academic Press Professional, 1995, ISBN 0-12-059756-X.
- [6] Cohen-Or, Daniel, Yiorgos L. Chrysanthou, Cláudio T. Silva, and Frédo Durand: *A survey of visibility for walkthrough applications*. IEEE Transactions on Visualization and Computer Graphics, 9(3):412–431, 2003, ISSN 10772626.
- [7] Creative Technologies, USC University of Southern California Institute for: *High-resolution light probe image gallery*, Jun 2017. <http://vgl.ict.usc.edu/Data/HighResProbes/>.
- [8] Durand, F: *3D Visibility: analytical study and applications*. Université Joseph Fourier, Grenoble, France, 1999.
- [9] Elstrodt, Jürgen: *Maß- und Integrationstheorie*. Springer, siebte korrigierte und aktualisierte Auflage, 2011, ISBN 978-3-642-17904-4.
- [10] Forkman, Johannes: *Estimator and Tests for Common Coefficients of Variation in Normal Distributions*. 38(2):233–251, 2009, ISSN 0361-0926. <http://dx.doi.org/10.1080/03610920802187448>.
- [11] Gautron, Pascal, Jaroslav Kriva, Sumanta Pattanaik, and Kadi Bouatouch: *Radiance Caching for Efficient Global Illumination Computation*. 11(5):1–11, 2005.
- [12] Grant, Barbara G.: *Field Guide to Radiometry*. SPIE, 2011, ISBN 978-0-8194-8827-5.
- [13] Jensen, Henrik Wann: *A practical guide to global illumination using photon maps*. SIGGRAPH 2000 Course Notes CD-, 38:20–es, 2000.
- [14] Kajiya, James T: *The rendering equation*. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.
- [15] Kusolitsch, Norbert: *Maß- und Wahrscheinlichkeitstheorie: Eine Einführung*. SpringerWienNewYork, 2011, ISBN 978-3-7091-0684-6.
- [16] Kühnel, Wolfgang: *Differentialgeometrie: Kurven - Flächen - Mannigfaltigkeiten*. Springer Spektrum, sechste Auflage, 2013, ISBN 978-3-658-00614-3.
- [17] Laboratory, Stanford University Computer Graphics: *The stanford 3d scanning repository*, Jun 2017. <https://graphics.stanford.edu/data/3Dscanrep/>.
- [18] Lafortune, Eric P and Yves D Willems: *Bi-Directional Path Tracing*.

- [19] LaMothe, André: *Tricks of the 3D Game Programming Gurus: Advanced 3D Graphics and Rasterization*. Sams Publishing, 2003.
- [20] Leobacher, Gunther and Friedrich Pillichshammer: *Introduction to Quasi-Monte Carlo Integration and Applications*. Birkhäuser, 2014, ISBN 978-3-319-03424-9.
- [21] Levoy, Marc and Pat Hanrahan: *Light field rendering*. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96, pages 31–42, 1996, ISSN 00978930. <http://portal.acm.org/citation.cfm?doid=237170.237199>.
- [22] Malacara, Daniel: *Color Vision and Colorimetry: Theory and Applications*. SPIE, 2011, ISBN 978-0-8194-8397-3.
- [23] McGuire, Morgan: *Computer graphics archive*, August 2011. <http://graphics.cs.williams.edu/data>.
- [24] Mm Oller, Tomas and Ben Trumbore: *Fast, Minimum Storage Ray/Triangle Intersection*. ACM SIGGRAPH 2005 Courses, (1):1–7, 2005, ISSN 1086-7651.
- [25] Morvan, Jean Marie and Boris Thibert: *Smooth surface and triangular mesh: comparison of the area, the normals and the unfolding*. SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications, pages 147–158, 2002.
- [26] Müller-Gronbach, Thomas, Erich Novak und Klaus Ritter: *Monte Carlo-Algorithmen*. Springer, 2012, ISBN 978-3-540-89140-6.
- [27] Nate just: *20 hdri images*, deviantart, Jun 2017. <http://just-nate.deviantart.com/art/20-HDRI-Images-109988135>.
- [28] Nicodemus, Fe, Jc Richmond, and Jj Hsia: *Geometrical considerations and nomenclature for reflectance*. Science And Technology, 60(October):1–52, 1977, ISSN 10411135. <http://graphics.stanford.edu/courses/cs448-05-winter/papers/nicodemus-brdf-nist.pdf>.
- [29] Nikodym, Tomas: *Ray Tracing Algorithm For Interactive Applications Tomas Nikodym*. page 76, 2010.
- [30] Nolting, Wolfgang: *Grundkurs Theoretische Physik 3: Elektrodynamik*. Springer, achte Auflage, 2007, ISBN 978-3-540-71251-0.
- [31] Ohta, Noboru and Alan Robertson: *Colorimetry: Fundamentals and Applications*. Wiley, 2005, ISBN 978-0-470-09472-3.
- [32] Parker, Steven, William Martin, Peter Pike J Sloan, Peter Shirley, Brian Smits, and Charles Hansen: *Interactive ray tracing*. Proceedings of the 1999 symposium on Interactive 3D graphics SI3D 99, pages:119–126, 1999. <http://portal.acm.org/citation.cfm?doid=300523.300537>.
- [33] Pharr, M. and G. Humphreys: *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, second edition, 2010.
- [34] Pharr, M., W. Jakob, and G. Humphreys: *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, third edition, 2017.
- [35] Pipkin, A. C.: *A Course on Integral Equations*. Springer Science+Business Media, LLC, 2013, ISBN 978-1-4612-8773-5.

- [36] Scherzer, Daniel, Chuong H. Nguyen, Tobias Ritschel, and Hans Peter Seidel: *Pre-convolved Radiance Caching*. Computer Graphics Forum, 31(4):1391–1397, 2012, ISSN 01677055.
- [37] Shirley, P., B. Wade, P. M. Hubbard, D. Zareski, B. Walter, and D. P. Greenberg: *Global Illumination via Density-Estimation*. Proceedings of 6th Workshop on Rendering, pages 219–230, 1995. http://link.springer.com/chapter/10.1007/978-3-7091-9430-0{_\}21{\%}5Cnhttp://www.cs.utah.edu/{~}shirley/papers/de95.pdf.
- [38] Veach, Eric: *Robust Monte Carlo Methods for Light Transport Simulation*. Dissertation at the Department of Computer Science of Stanford University, 134(December):759–764, 1997, ISSN 13509462. http://graphics.stanford.edu/papers/veach{_\}thesis/.
- [39] Veach, Eric and Leonidas J. Guibas: *Metropolis light transport*. Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97, pages 65–76, 1997, ISSN 00978930. <http://portal.acm.org/citation.cfm?doid=258734.258775>.
- [40] Wald, Ingo: *Fairy forest, the utah 3d animation repository*, Jun 2017. <http://www.sci.utah.edu/~wald/animrep/>.
- [41] Walter, Bruce, Philip M Hubbard und Peter Shirley: *Global Illumination Using Local Linear Density Estimation*.
- [42] Ward, Gregory J. und Paul S Heckbert: *Irradiance gradients*. ACM SIGGRAPH 2008 classes on - SIGGRAPH '08, Seite 1, 2008. <http://portal.acm.org/citation.cfm?doid=1401132.1401225>.
- [43] Ward, Gregory J., Francis M. Rubinstein, and Robert D. Clear: *A ray tracing solution for diffuse interreflection*. ACM SIGGRAPH Computer Graphics, 22(4):85–92, 1988, ISSN 00978930. <http://portal.acm.org/citation.cfm?doid=378456.378490>.
- [44] Wolfe, William L.: *Introduction to Radiometry*. SPIE, 1998, ISBN 0-8194-2758-6.
- [45] Yuksel, Cem, John Keyser und Donald H. House: *Mesh colors*. ACM Transactions on Graphics, 29(2):1–11, 2010, ISSN 07300301.

A Verwendete Szenen

Bild	Daten
	<p>»Dragon«-Szene:</p> <p>871306 Dreiecke 439704 Eckpunkte</p> <p>Quelle: [17, 23]</p>
	<p>»Shaderball«-Szene:</p> <p>119250 Dreiecke 61258 Eckpunkte</p> <p>Quelle: vom <i>Fraunhofer ITWM</i> zur Verfügung gestellt</p>
	<p>»Audi R8«-Szene:</p> <p>1749705 Dreiecke 1601106 Eckpunkte</p> <p>Quelle: vom <i>Fraunhofer ITWM</i> zur Verfügung gestellt</p>
	<p>»Cornell Box«-Szene:</p> <p>2188 Dreiecke 1281 Eckpunkte</p> <p>Quelle: [23]</p>
	<p>»Fairy«-Szene:</p> <p>174117 Dreiecke 103523 Eckpunkte</p> <p>Quelle: [40]</p>

B Verwendete HDR Maps

Bild	Daten
	<p>»Uffizi Gallery«: Uffizi Gallery, Italy Quelle: [7]</p>
	<p>»Grace Cathedral«: Grace Cathedral, San Francisco, California Quelle: [7]</p>
	<p>»Ennis-Brown House«: Ennis-Brown House, Los Angeles, California Quelle: [7]</p>
	<p>»Basketball Court«: Quelle: [2]</p>
	<p>»Sky 10«: Quelle: [27]</p>
	<p>»Sky 20«: Quelle: [27]</p>

C Weitere Ergebnisse der adaptiven Irradianzschätzung



(a) Path Tracing

(b) Irradianz

(c) Sampleanzahl

Abbildung 31: Die Bilder zeigen die »Dragon«-Szene mit einer schwach variierenden Umgebungsbeleuchtung. 31a ist das durch Path Tracing erzeugte Referenzbild. 31b zeigt die an den Eckpunkten adaptiv aufgenommenen Irradianzen mit linearer Interpolation. 31c stellt dabei die zugehörigen Sampleanzahlen dar. Hier bedeutet eine hellere Farbe eine höhere Sampleanzahl. Die Berechnungszeit betrug 253 s mit der maximalen Sampleanzahl 2^{16} und einem maximalen relativen Fehler von 3 %.
Die ersten beiden dienen dem Vergleich der Korrektheit des Algorithmus. Das dritte Bild



(a) Path Tracing

(b) Irradianz

(c) Sampleanzahl

Abbildung 32: Die Bilder zeigen die »Shaderball«-Szene mit »Sky 20«-HDR analog zu Abbildung 31. Die Berechnungszeit betrug 1170 s mit der maximalen Sampleanzahl 2^{18} und einem maximalen relativen Fehler von 1 %.

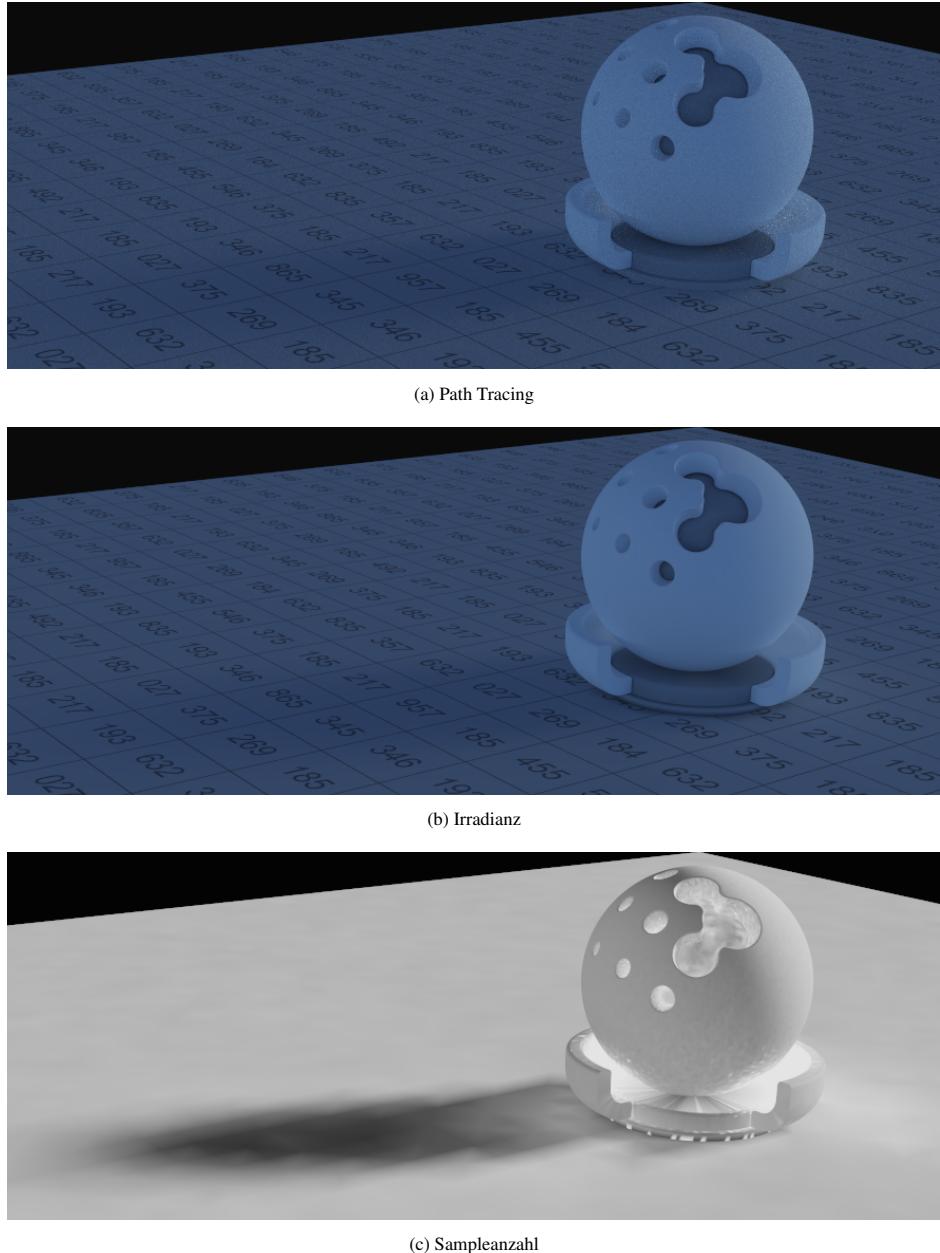


Abbildung 33: Die Bilder zeigen die »Shaderball«-Szene mit »Sky 10«-HDR analog zu Abbildung 31. Die Berechnungszeit betrug 1090 s mit der maximalen Sampleanzahl 2^{18} und einem maximalen relativen Fehler von 1 %.



(a) Path Tracing



(b) Irradianz



(c) Sampleanzahl

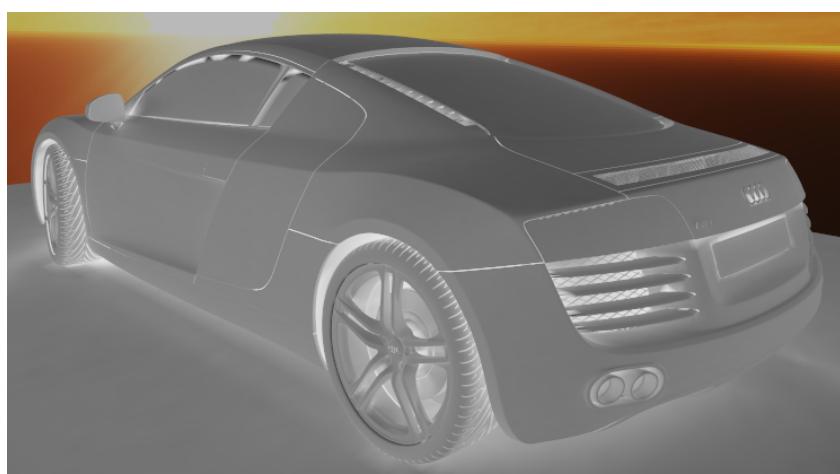
Abbildung 34: Die Bilder zeigen die »Audi R8«-Szene mit der »Sky 20«-Umgebungsbeleuchtung analog zu Abbildung 31. Die Berechnungszeit betrug 17.8 h mit der maximalen Sampleanzahl 2^{18} und einem maximalen relativen Fehler von 1 %.



(a) Path Tracing



(b) Irradianz



(c) Sampleanzahl

Abbildung 35: Die Bilder zeigen die »Audi R8«-Szene mit der »Sky 20«-Umgebungsbeleuchtung analog zu Abbildung 31. Die Berechnungszeit betrug 17.8 h mit der maximalen Sampleanzahl 2^{18} und einem maximalen relativen Fehler von 1 %.

D Weitere Irradiance Map Ergebnisse

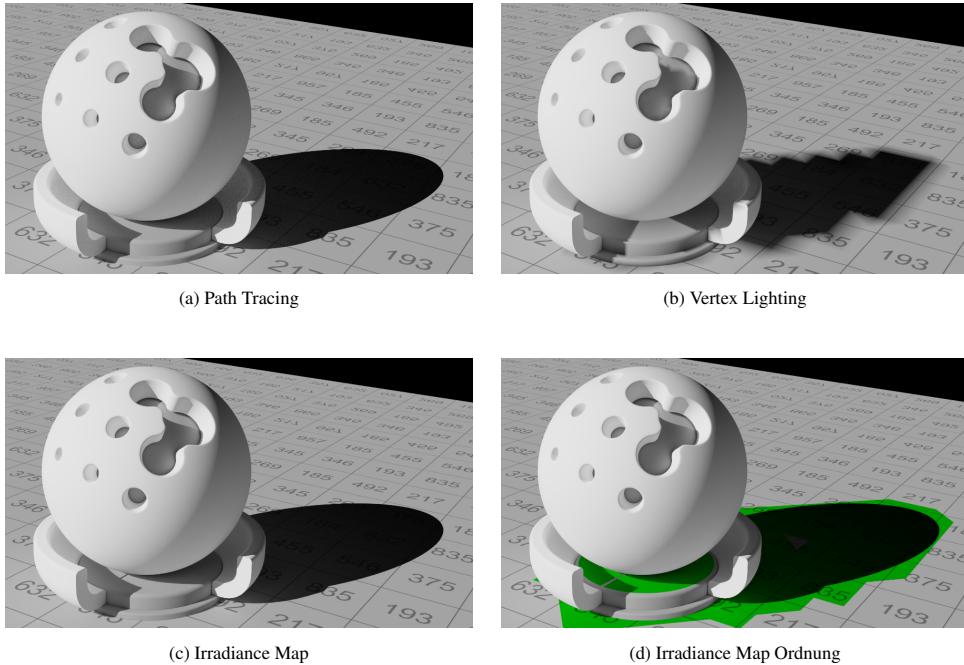


Abbildung 36: Die Bilder zeigen die »Shaderball«-Szene mit einer direktonalen Lichtquelle unter Verwendung von Path Tracing als Referenzbild, von Vertex Lighting und der Irradiance Map. Für die adaptive Irradianzschätzung wurden eine obere Fehlerschranke von 1 % und maximal 2^{14} Samples verwendet. Die Irradiance Map wurde durch circa $15.6 \cdot 10^6$ Messpunkte bestimmt, die insgesamt 59.7 MiB einnehmen. Die obere Fehlerschranke betrug 2.5 % und die maximale Ordnung 2⁷. Die Generierung benötigte 11500 s. Die grün markierten Bereiche in 36d stellen Dreiecke mit einer Ordnung größer 63 dar.

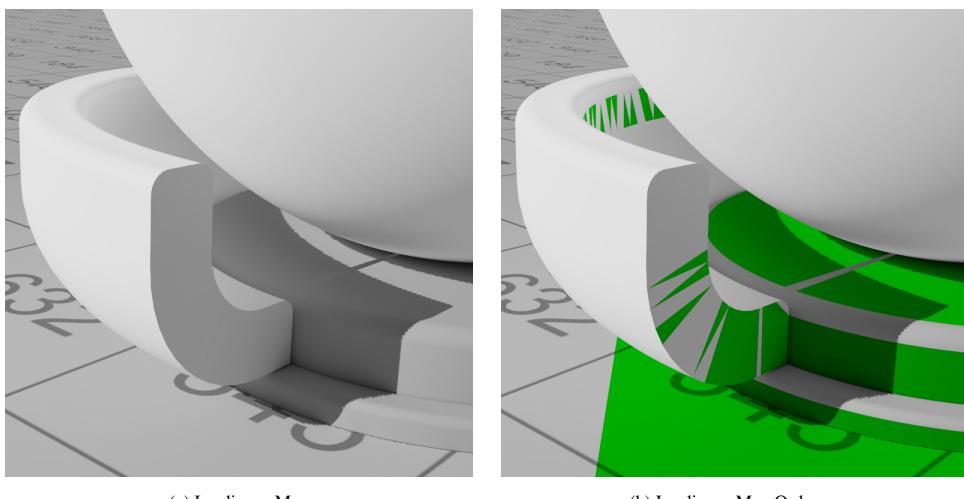


Abbildung 37: Die Bilder zeigen die »Shaderball«-Szene aus Abbildung 36 aus einem anderen Blickwinkel. Die grün markierten Bereiche in 37b stellen Dreiecke mit einer Ordnung größer 8 dar. Es ist erkennbar, dass am Rand des harten Schattens Aliasing-Artefakte entstehen. Weiterhin gibt es ein Dreieck auf der unteren Platte, welches einen fehlerhaften Schattenverlauf zeigt. Hier bewertete das verwendete Fehlermaß das Dreieck falsch, da die Irradianz eine hohe Variation besitzt.

Eigenständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Hilfsmittel und Quellen angefertigt habe. Die eingereichte Arbeit ist nicht anderweitig als Prüfungsleistung verwendet worden oder in deutscher oder einer anderen Sprache als Veröffentlichung erschienen.

Seitens des Verfassers bestehen keine Einwände, die vorliegende Bachelorarbeit für die öffentliche Benutzung zur Verfügung zu stellen.

Jena, 16.Juni 2017

Markus Pawellek