

Glossy Reflections

Seminar-Ausarbeitung von

Kai Westerkamp

An der Fakultät für Informatik
Institut für Visualisierung und Datenanalyse,
Lehrstuhl für Computergrafik

29. Mai 2016

Inhaltsverzeichnis

1	Einleitung	1
2	Ähnliche Arbeiten	3
2.1	Virtual Point Lights	3
2.2	Photon Mapping	3
3	BRDF und Spherical Gaussians	5
3.1	Spherical Gaussians	5
4	Basis Algorithmus und One-bounce Interreflection	7
4.1	One-Bounce Interreflection	7
4.1.1	Reflektiertes Licht	7
4.1.2	Beleuchtung mit dem reflektiertem Licht	8
4.1.3	Integration über ein sphärisches Dreieck	8
5	Baumstruktur	11
5.1	Abschätzung der Reflexion eines Knotens	11
5.2	Abschätzung des Fehlers	12
6	Implementierung	13
6.1	Sichtbarkeit	13
6.2	Umsetzung des Algorithmus	13
7	Ergebnisse und Vergleiche	15
	Literaturverzeichnis	17

1. Einleitung

Indirekte Beleuchtung: Einheitlicher Algorithmus für BRDF's mit allen Frequenzen.

2. Ähnliche Arbeiten

Ich bin mir hier noch nicht ganz sicher welche ich nehmen soll. Ich werde nicht die Zeit haben mich in alle aus dem Paper genügend tief einzulesen

2.1 Virtual Point Lights

Auf jeden Fall, die kommen nacher noch bei Sichtbarkeit

2.2 Photon Mapping

+ ...

3. BRDF und Spherical Gaussians

Die korrekte Beleuchtungsberechnung ist ein zentraler Bestandteil der Computergrafik. Besonders bei Szenen mit vielen unterschiedlichen Materialien stellt dies eine große Herausforderung dar.

Zur physikalischen Beleuchtungsberechnung muss hierzu die Rendergleichung berechnet werden.

$$L(x, o) = L_e(x, o) + L_r(x, o) = L_e(x, o) + \int_{\Omega^+} f_r(i, x, o) L_i(x, i) \max(0, i \cdot n) di \quad (3.1)$$

Hierbei ist $L(x, o)$ die Radiance, die an einem Oberflächenpunkt x in die Richtung o (outgoing) abgegeben wird. $L_e(x, o)$ ist das von dem Oberflächenpunkt emittierte Licht, und das Integral das reflektierte Licht in die Richtung o . Das reflektierte Licht wird bestimmt durch das Integral über die positive Hemisphäre, wobei $L_i(x, i)$ das einfallende Licht aus der Richtung i (incoming) und $\max(0, i \cdot n)$ der Kosinus zwischen der Oberflächennormalen n und i ist. $f_r(i, x, o)$ (kurz $f_r(i, o)$) ist die *Bidirectional Reflectance Distribution Function* (BRDF). Die BRDF ist vom Material abhängig und gibt an einem Oberflächenpunkt x an, wieviel Licht vom Einfallswinkel i in die Ausfallrichtung o reflektiert wird.

In der Rendergleichung 3.1 kann $L_i(x, i) = L(y, -i)$ mit $y = ray(x, i)$ geschrieben werden. Somit erhält man eine rekursive Darstellung. Diese zu lösen ist eine komplexe Aufgabe und wird häufig durch Approximationen angenähert und vorberechnet, um Beleuchtung in Echtzeit zu berechnen. Z.B. ignoriert man alle indirekte Beleuchtung und betrachtet nur alle Oberflächen, bei denen $L_e > 0$ ist. In dieser Ausarbeitung wird ein Algorithmus vorgestellt der die Gleichung mit einer indirekten Reflektion auswertet.

Eine einfache BRDF ist durch das Phong Beleuchtungsmodell gegeben. Die Beleuchtung wird in eine diffuse und spekulare Komponente getrennt. Der diffuse Anteil wird berechnet durch $k_d * I_L * (N \cdot L)$ und der spekulare Anteil durch $k_s * I_L * (R \cdot V)^n$ (siehe 3). Hierbei ist I_L die einfallende Lichtintensität, k_d und k_s materialabhängige Konstanten und n der Phong Exponent, der die Größe der spekularen Glanzlichter beeinflusst. **(siehe Bild Nummer 3.1, da nur dort die Größen eingeführt werden)** **ToDo**

3.1 Spherical Gaussians

Eine Möglichkeit BRDF's zu approximieren, bieten Spherical Gaussians (SG). Spherical Gaussians sind definiert als

$$G(v; p, \lambda, c) = c * e^{\lambda(v \cdot p - 1)}$$

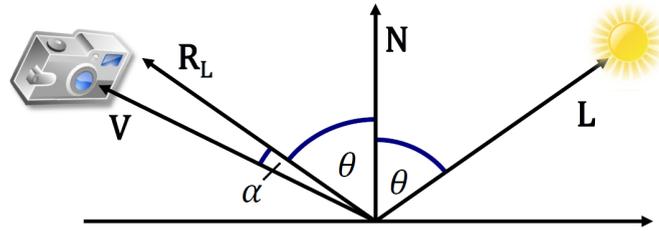


Abbildung 3.1: Richtungen im Phong Beleuchtungsmodell. N ist die Oberflächennormale, V der Viewvektor, L der Lichtvektor und R die Reflexion von L an N (Vorlesung Fotorealistiche Bildsynthese 02_ BRDF)

mit p als Mittelachse, λ als sharpness und c als Skalar. Als kurze Schreibweise ist $G_l(v) = G(v; p_l, \lambda_l) = G(v; p_l, \lambda_l, 1)$ BRDFs können in eine Summe aus einem diffusen und einem spekularen Anteil zerlegt werden:

$$f_r(i, o) = k_d + k_s f_s(i, o)$$

In Wang et al. [WRG⁺09] wird beschrieben, wie die spekulare Komponente als Summe von SGs dargestellt werden kann:

$$K_s f_r(i, o) \approx \sum_j = 1^n G(i, o^j, \lambda^j, c^j)$$

mit o^j , λ^j , c^j als Zentrum, sharpness und Koeffizient der j SG. Die diffuse Komponente kann als SG mit 0 sharpness dargestellt werden.

$$k_d = G(i; 2(o * n)n - o, 0, k_d)$$

Es kann somit die BRDF als Summe von SGs dargestellt werden. In [WRG⁺09] lassen sich hierfür Beispiele finden, z.B. für Bling-Phong und Cook-Torrance. Des Weiteren können Lichtquellen auch als SGs ($G_l(i)$) dargestellt werden und somit kann eine einheitliche Darstellung als Summe von SGs erzielt werden.

4. Basis Algorithmus und One-bounce Interreflection

In dem Paper von Xu et al. [XCM⁺14] wird ein Algorithmus vorgestellt, der die Renderinggleichung für die Rekursionstiefe 1 löst. Zur Beleuchtung eines Dreiecks (Empfänger) wird zusätzlich das indirekte Licht von anderen Dreiecken (Reflektor) berechnet. Dabei wird die SG-Darstellung von BRDF's verwendet und daraus eine stückweise definierte Approximation hergeleitet, um die Beleuchtung effizienter zu berechnen. Anschließend wird eine Baumstruktur 5 eingeführt, um diesen Algorithmus auch bei großen Szenen effizient anwenden zu können. Da nicht jedes Dreieck für alle anderen sichtbar ist und somit indirekt beleuchtet wird, wird zuletzt eine Lösung für das Sichtbarkeitsproblem 6 beschrieben.

4.1 One-Bounce Interreflection

In diesem Abschnitt leiten wir die Reflexionsgleichung für einen Oberflächenpunkt X (receiver) mit Normale n_x in Richtung o her. Die SG Lichtquelle l wird hierbei an einem Dreieck T (reflector) mit Normale n_r reflektiert. Für die Herleitung der Formel nehmen wir an, dass nichts zwischen x und T ist (Siehe Abb. 4.1).

Aus der Renderinggleichung 3.1 folgt für den Receiver:

$$L(x, o) = \int_{\Omega_T} f_r(-r, x, o) L_i(x, -r) \max(0, -r \cdot n_x) dr \quad (4.1)$$

Wobei r die Richtung von einem Punkt y auf dem Reflektor ist und Ω_T die sphärische Projektion des Dreiecks T. Für das einfallende Licht ist $L_i(x, -r) = L(y, r)$:

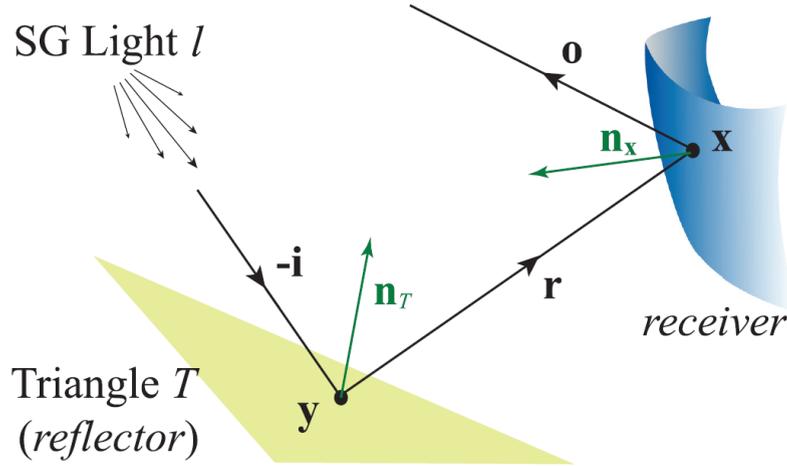
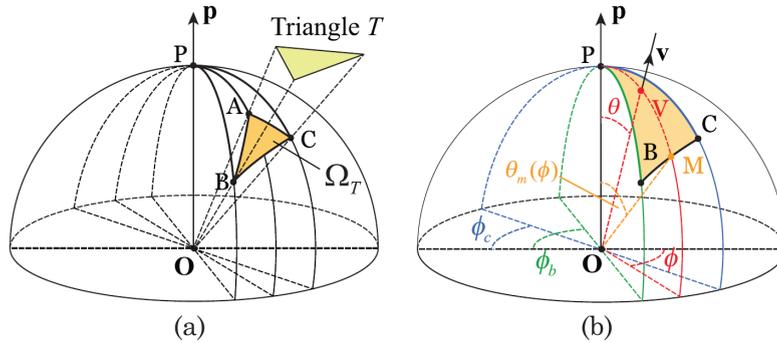
$$L(y, r) = \int_{\Omega_T} f_r(i, y, r) G_l(i) \max(0, i \cdot n_r) di \quad (4.2)$$

Beim Receiver wird hier die SG der Lichtquelle eingesetzt.

4.1.1 Reflektiertes Licht

Im Folgenden approximieren wir die Gleichung 4.2, um eine besser berechenbare Funktion zu erhalten. Der Kosinus-Term $\max(0, i \cdot n_r)$ durch die mittlere Richtung des Dreiecks T annähern. (**verstehe ich nicht - liegt vielleicht am fehlenden Verb**) Damit erhält man die Konstante $\max(0, i_T(r) \cdot n_r)$, die man aus dem Integral ziehen kann. Die BRDF $f_r(i, y, r)$ ist wie in Kapitel 3.1 beschreiben als Summe von SGs darstellbar $f_r(i, y, r) \approx$

ToDo

Abbildung 4.1: Lichtpfad für die Reflexion an einem Dreieck [WRG⁺09]Abbildung 4.2: Integration eines sphärischen Dreiecks ω_T [WRG⁺09]

$G_T(i)$. Das Integral über 2 SGs kann durch ein SG angenähert werden (siehe Anhang von [WRG⁺09]). Daraus ergibt sich eine Gleichung mit einer glatten Funktion und einer SG (die genauen Parameter sind nachzulesen im paper von Wu et al. 3.1).

$$L(y, r) \approx F(r)G(r; i_r, \lambda_r) \quad (4.3)$$

4.1.2 Beleuchtung mit dem reflektiertem Licht

Die Gleichung 4.3 wird nun in 4.1 eingesetzt und die BRDF als Summe von SGs ersetzt.

$$L(x, o) = \int_{\Omega_T} F(r) \max(0, -r \cdot n_x) G(r; i_r, \lambda_r) G(r; -o_x, \lambda_x, c_x) dr \quad (4.4)$$

Das Produkt zweier SGs lässt sich durch eine SG ersetzen und die Funktion $F(r) \max(0, -r \cdot n_x)$ ist "wirklich glatt" [WRG⁺09]. Man kann man es aus dem Integral ziehen. Somit erhält man eine vereinfachte Version der Rendergleichung:

$$L(x, o) \approx H(r'_h) \int_{\Omega_T} G(r; r_h, \lambda_h, c_h) dr \quad (4.5)$$

4.1.3 Integration über ein sphärisches Dreieck

ToDo (hier etwas herleitung vlt?) Um die Gleichung 4.5 auszuwerten, muss $\int_{\Omega_T} G(v; p, \lambda) dv$

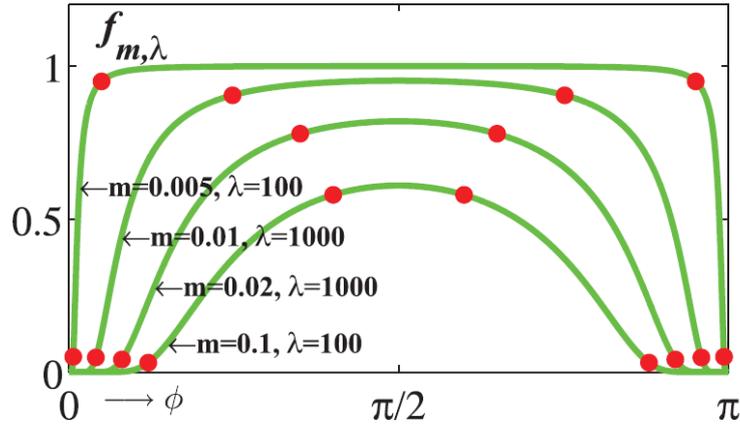


Abbildung 4.3: Plot der Funktion $f_{m,\lambda}$. Diese kann durch lineare Funktionen durch die roten Punkte angenähert werden. [WRG⁺09]

berechnet werden. Mit der Mittelachse P des SG ergibt sich aus dem Bild 4.1.3, dass

$$\Omega_t = \Omega_{\triangle ABC} = \Omega_{\triangle PBC} - \Omega_{\triangle PAB} - \Omega_{\triangle PCA}$$

ist. Durch Integration über die Winkel und einigen Umformungen ergibt sich ein Integral über eine eindimensionale reelle Funktion 4.6, die durch eine stückweise definierte lineare Funktion angenähert werden kann.

$$f_{m,\lambda}(\phi) = \exp \left[\lambda \left(\frac{\sin \phi}{\sqrt{m^2 + \sin^2 \phi}} - 1 \right) \right] \quad (4.6)$$

5. Baumstruktur

Um die indirekte Beleuchtung eines Oberflächenpunktes auszuwerten, muss die Formel aus dem vorherigen Kapitel für alle Dreiecke berechnet werden. Bei Szenen mit wenigen Dreiecken ist das möglich. Jedoch steigen die Kosten linear mit der Anzahl der Dreiecke. In diesem Kapitel wird eine binäre Baumstruktur eingeführt, die die Effizienz bei vielen Dreiecken steigert. Die Blätter des Baumes sind die Dreiecke der Szene und die einzelnen Knoten die Referenzen und Mittelwerte der Kinder. Der Baum wird von der Wurzel aufgebaut, und in jedem Schritt in 2 Kinder gesplittet. Für eine Split wird eine Hauptkomponentenanalyse durchgeführt. Im 6-dimensionalen Raum aus Dreiecksmittelpunkt und gewichteter Normale wird die Hauptachse berechnet und am Median gesplittet. Dieses Vorgehen garantiert das Aufteilen eines Knotens entlang der größten Varianz in 2 gleich große Kinder.

In den Knoten wird der durchschnittliche Mittelpunkt, die Durchschnittsnormale, die Bounding Box, der Normalenkegel, und die absolute Dreiecksfläche gespeichert. Bei texturtierten Dreiecken wird zusätzlich der Durchschnittsfarbwert und die Maximal- und Minimalfarbe gespeichert. Aus diesen Daten lässt sich das reflektierte Licht zu einem Empfänger approximieren 5.1, und eine Abschätzung des Fehlers errechnen 5.1.

Beim Rendern wird zunächst der Schnitt durch den Baum bestimmt. Angefangen wird mit dem Wurzelknoten und es wird immer der Knoten mit dem größten Fehler durch seine beiden Kinder ersetzt. Dieser Vorgang wird solange wiederholt bis 1.000 Knoten im Schnitt enthalten sind oder der Fehler kleiner ist als 1% des reflektierten Lichts. Wenn Blätter im Schnitt enthalten sind, kann das reflektierte Licht wie in Kapitel 4 berechnet werden. Jedoch funktioniert der Algorithmus nur bei Dreiecken mit einheitlicher BRDF. Bei texturtierten Dreiecken wird deshalb der Mittelwert der Farbe bestimmt und wir nehmen ihn als uniforme BRDF. Sollte der Fehler durch diese Annäherung zu groß werden, wird das Dreieck in kleinere unterteilt und dann ausgewertet. Für die Knoten im Baum wird im folgenden Abschnitt beschrieben, wie die Reflexion abgeschätzt werden kann.

5.1 Abschätzung der Reflexion eines Knotens

Um das reflektierte Licht auszurechnen, ändern wir Gleichung 4.5 geringfügig ab. Für einen Knoten N sei I_N der Mittelpunkt, n_N die durchschnittliche Normale und t_N die Durchschnittsfarbe. Wir ändern den Integrationsbereich von einem einzelnen Dreieck Ω_T zu der sphärischen Projektion aller Dreiecke des Knotens Ω_N .

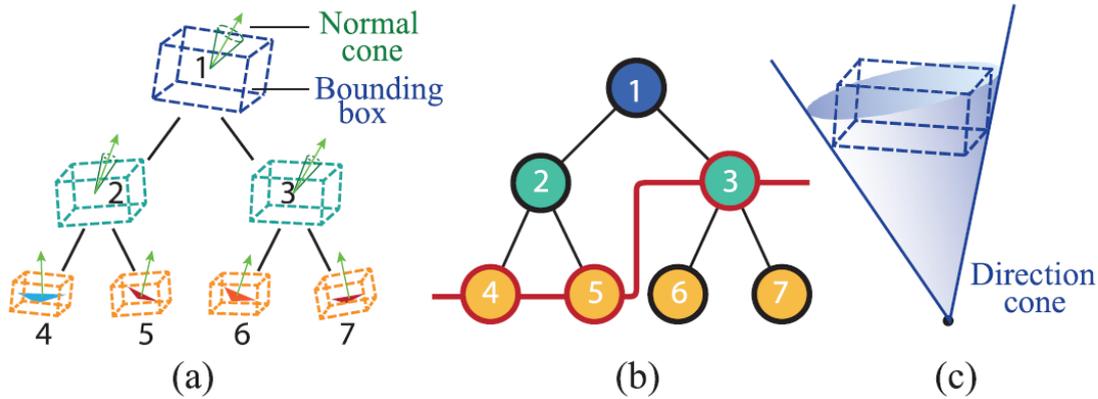


Abbildung 5.1: a) Der Binärbaum der Dreiecke mit Bounding box und Normal cone. b) Ein Beispiel Reflectorcut, c) Direction cone [WRG⁺09]

$$L(x, o) \approx t_N \cdot H(r'_h) \int_{\Omega_N} G(r; r_h, \lambda_h, c_h) dr \quad (5.1)$$

Da Ω_N nicht bekannt ist, integrieren wir über die komplette Hemisphäre und ersetzen Ω_N durch eine binäre Maske $V_{\Omega_N}(r)$.

$$\int_{\Omega_N} G(r; r_h, \lambda_h, c_h) dr = \int_{\Omega} G(r; r_h, \lambda_h, c_h) V_{\Omega_N}(r) dr \quad (5.2)$$

Diese Binärmaske ist wiederum durch eine SG approximierbar $V_{\Omega_N}(r) \approx G(r; r_N, \lambda_N, c_N) = G_N(r)$. Hierbei ist r_N die Richtung von I_N zu X und die Sharpness errechnet aus der Varianz und Dreiecksfläche des Knotens. Damit erhalten wir ein Integral über zwei SGs, das, wie oben beschrieben, eine analytische Lösung hat.

5.2 Abschätzung des Fehlers

Bei der Abschätzung der Funktion $V_{\Omega_N}(r)$ kann es zu großen Fehlern kommen. Ist der Fehler eines Knotens zu groß, so wird er in die beiden Kinderknoten aufgesplittet. Um die Obergrenze des Fehlers in Gleichung 5.2 abzuschätzen, berechnen wir die größten und kleinsten Werte der SG g_{min} & g_{max} , den Winkel Ω_N , $||\Omega_N||_{min}$, $||\Omega_N||_{max}$ und der Texturwerte t_{min} , t_{max} . **(willst du wirklich das Kaufmannsund? Ich würde hier das Wort "und" schreiben.)** Der Fehler lässt sich dann mit $H(r'_h) * (t_{max} * g_{max} * ||\Omega_N||_{max} - t_{min} * g_{min} * ||\Omega_N||_{min})$ berechnen. Die Texturwerte werden in dem jeweiligen Knoten gespeichert und die restlichen Werte können aus der Bounding Box und der Normaleinverteilung des Knotens errechnet werden. **(Normal"ein"verteilung ?) (sind hier die Fehlerwerte gemeint, die auch im Knoten gespeichert werden?)**

ToDo

ToDo

ToDo

6. Implementierung

6.1 Sichtbarkeit

Bisher haben wir die Reflexion von Licht aus einer Lichtquelle über ein Dreieck zu einem Oberflächenpunkt berechnet. Dabei haben wir nicht betrachtet, ob der Lichtpfad durch Objekte blockiert ist. Um die Sichtbarkeit zwischen Lichtquelle und dem Reflektor zu evaluieren, wird Variance Shadow Map (VSM, [DL06]) verwendet. Hier wird die Shadowmap der Lichtquelle für 16 gleichmäßig verteilte Samples auf dem Reflektordreieck ausgewertet. Aus den Werten wird der Durchschnitt berechnet und zu dem Dreieck gespeichert.

Die Sichtbarkeit zwischen Reflektor und Retriever ist schwieriger zu bestimmen. Zunächst werden während der Initialisierung 200 virtuelle Lichter in der Szene verteilt um zur Laufzeit Imperfect Shadow Maps [RGK⁺08] zu berechnen. **(warum 200? Es können auch andere Werte sinnvoll sein)** Imperfect Shadow Maps sind Shadow Maps mit geringer Auflösung für jede virtuelle Lichtquelle. In den Knoten der Baumstruktur werden die drei nächsten Lichter gespeichert. Bei der Berechnung der Sichtbarkeit wird der Mittelpunkt der Knoten, bzw. des Dreiecks in die Ebene der drei Punktlichtquellen projiziert. Die Werte aus den zugehörigen ISM werden ausgelesen und mit Hilfe von baryzentrische Koordinaten linear gemittelt. **ToDo**

6.2 Umsetzung des Algorithmus

Zur Initialisierung muss die Szene geladen und die Baumstruktur errechnet werden. Hier wird für jedes Mesh, das mehr als 200 Dreiecke enthält, der Binärbaum errechnet. Für kleinere Dreiecksnetze ist es effizienter die Reflexion von allen Dreiecken auszurechnen.

Während eines eigenen Renderdurchgangs wird zunächst die direkte Beleuchtung der Szene bestimmt. Anschließend muss für jeden Pixel ein Schnitt durch den Baum bestimmt werden. Dafür wird zunächst für jeden Vertex ein Schnitt bestimmt, die dann für die Pixel gemittelt werden. **(ist das Relativpronomen “die” richtig? Ich denke, es muss “der” (der Schnitt) heißen.)** Im finalen Rendervorgang wird dann der Schnitt ausgewertet, für jeden Pixel die Reflexion berechnet und mit den Werten aus den Shadow Maps gewichtet. **ToDo**

Um die Performance zu verbessern, kann die Schnitt-Selektion in CUDA implementiert werden. Jedoch gibt es noch keine effizienten Prioritätenlisten für CUDA. Die Schnitt-Selektion wird in 5 normale Warteschlangen mit unterschiedlichen Fehlern aufgeteilt.

ToDo

(Stimmt das nach meiner Änderung noch?) Die erste Schlange speichert z.B alle Knoten mit Fehlern größer als 16% und nach dem Splitten werden die Kinder zu einer Liste mit Fehler größer als 8% hinzugefügt. **(immer beide Kinder? Ein Kind könnte doch genauer sein)** So erhält man eine nicht ganz akkurate Prioritätenliste, aber eine ausreichend genaue für diesen Zweck.

ToDo

7. Ergebnisse und Vergleiche

Soll ich sowas überhaupt behandeln? (**unbedingt ;-)**)

ToDo

(meinst du in der letzten Bildunterschrift: “ Würfel auf Ebenen mit verschiedenen specularen Materialien”)

ToDo

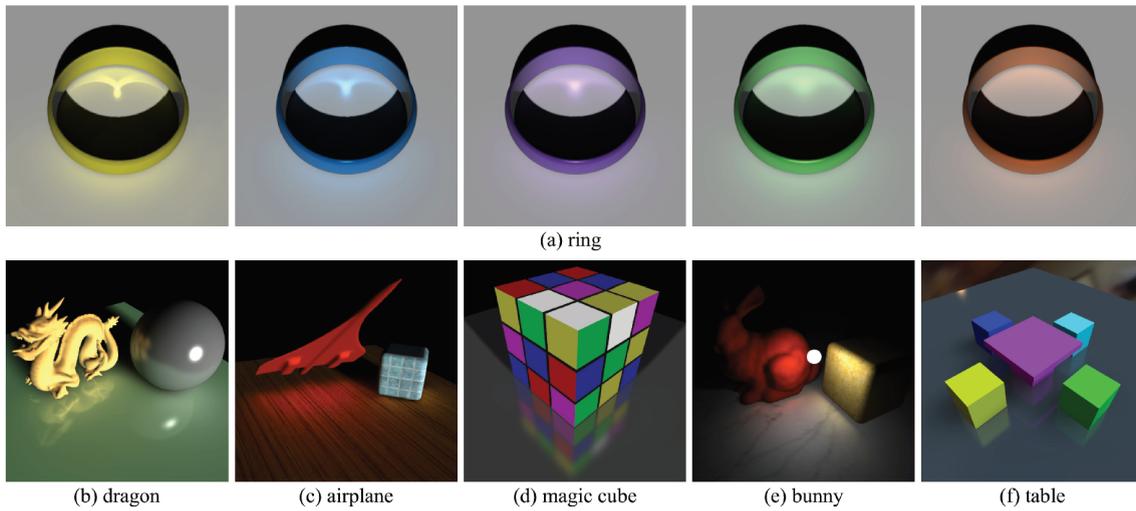


Abbildung 7.1: Einige Beispielbilder: a) Kaustiken mit unterschiedlichen Ring-BRDFs, b) indirekte Highlights c) diffuse Reflexion d) glossy Reflexionen e) eine Punktlichtquelle f) Umgebungslicht [WRG⁺09]

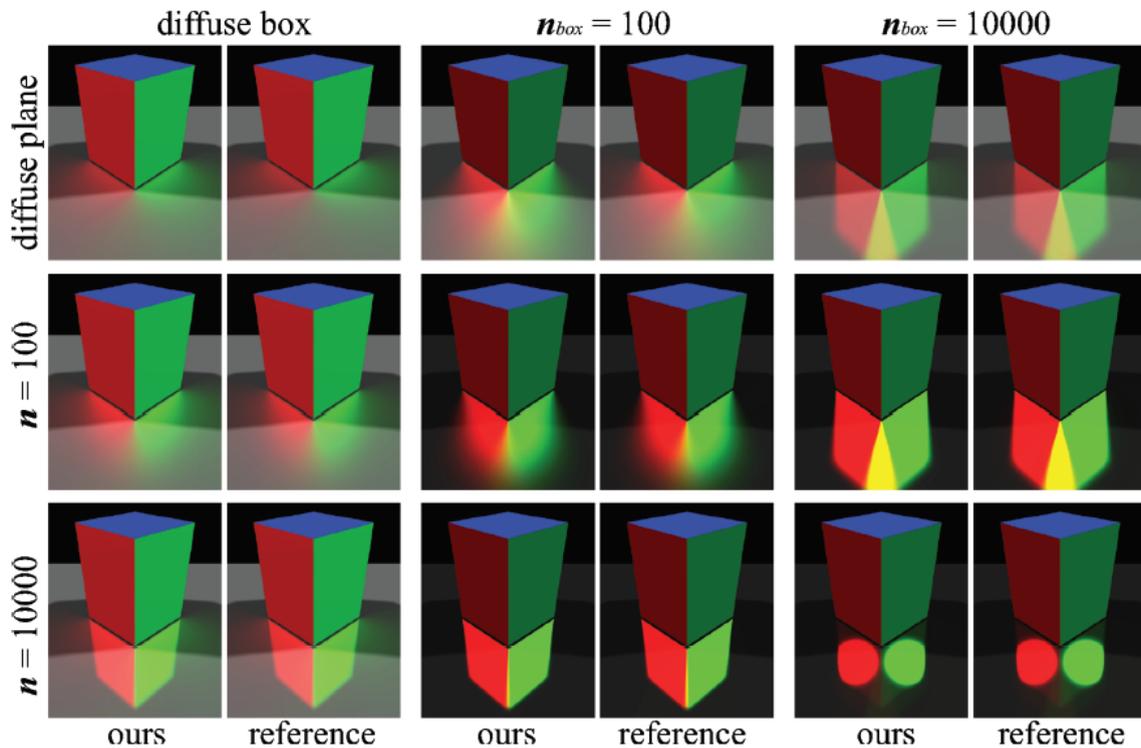


Abbildung 7.2: Verschiedene spekulare Materialien auf der Ebene und dem Würfel. [WRG⁺09]

Literaturverzeichnis

- [DL06] William Donnelly und Andrew Lauritzen: *Variance Shadow Maps*. In: *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, I3D '06*, Seiten 161–165, New York, NY, USA, 2006. ACM, ISBN 1-59593-295-X. <http://doi.acm.org/10.1145/1111411.1111440>.
- [RGK⁺08] Tobias Ritschel, Thorsten Grosch, Min H. Kim, Hans Peter Seidel, Carsten Dachsbacher und Jan Kautz: *Imperfect Shadow Maps for Efficient Computation of Indirect Illumination*. ACM Trans. Graph. (Proc. of SIGGRAPH ASIA 2008), 27(5), 2008.
- [WRG⁺09] Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder und Baining Guo: *All-Frequency Rendering of Dynamic, Spatially-Varying Reflectance*. ACM Trans. Graph., 28(5):1–10, 2009.
- [XCM⁺14] Kun Xu, Yan Pei Cao, Li Qian Ma, Zhao Dong, Rui Wang und Shi Min Hu: *A Practical Algorithm for Rendering Interreflections with All-frequency BRDFs*. ACM Trans. Graph., 33(1):1–10:16, 2014.

Erklärung

Ich versichere, dass ich die Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe. Die Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt und von dieser als Teil einer Prüfungsleistung angenommen.

Karlsruhe, den 29. Mai 2016

(Kai Westerkamp)