Hybride Beleuchtungssimulation für Flächen und Volumen mittels Photon Maps und Radiosity

Diplomarbeit im Fach Informatik

vorgelegt von

Matthias Ott

geb. am 23. Juli 1972 in Ansbach

angefertigt am

Institut für Mathematische Maschinen und Datenverarbeitung
Lehrstuhl für Graphische Datenverarbeitung
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: Hartmut Schirmacher
Betreuender Hochschullehrer: Prof. Dr. Hans-Peter Seidel

Beginn der Arbeit: 15. Juli 1998

Abgabe der Arbeit: 13. Januar 1999

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Globale Beleuchtungsberechnung beschäftigt sich mit der Beleuchtung einzelner Objekte unter Berücksichtigung der Interaktion mit anderen Objekten im Raum. Dabei kann es zu Effekten wie der Abschattung einzelner Objekte durch andere oder auch der indirekten Beleuchtung von Objekten durch Reflexion kommen. Diese indirekte Beleuchtung kann über eine

- perfekt spekulare Reflexion,
- schwach spekulare Reflexion,
- oder perfekt diffuse Reflexion

erfolgen. Bei einer perfekt spekularen Reflexion wird das Licht von einem Objekt reflektiert, ohne daß es gestreut wird, bei einer schwach spekularen Reflexion gibt es eine Streuung in einem kleinen Bereich des Raumes und bei einer diffusen Reflexion wird das Licht gleichmäßig in den gesamten sichtbaren Raum gestreut.

Für diese verschieden Möglichkeiten der Interaktion von Objekten bei der globalen Beleuchtungssimulation wurde schon eine große Anzahl von Algorithmen entwickelt, die diese durch verschiedene Ansätze zu lösen versuchen:

- Die direkte Beleuchtung kann durch Verschicken von Strahlen mit den Ray Tracing Ansatz [15] berechnet werden, zusätzlich kann auch die indirekte Beleuchtung durch verschiedene strahlenbasierte Monte Carlo Ansätze, wie zu Beispiel von Lafortune und Willems [17], bestimmt werden.
- Es existieren für die direkte und indirekte diffuse Beleuchtung mehrere Radiosity Algorithmen [8], die diese Aufgabe durch Zerlegung der Szene in finite Elemente erledigen.
- Eine andere Möglichkeit zur Bestimmung der indirekten diffusen als auch der indirekten spekularen Beleuchtung ist der Photon Maps Ansatz von Henrik Wann Jensen [11, 12, 13, 14], der unter anderem durch Verschießen von Photonen von den Lichtquellen aus diesen Beleuchtungsteil berechnet.
- Auch Gregory J. Ward hat mit seinem Irradiance Gradients Algorithmus [34, 33], welcher sich unter anderem auf das Speichern von einmal berechneten Irradiance-Werten stützt, ein Verfahren vorgestellt, daß den indirekten diffusen Beleuchtungsanteil

berechnen kann.

Bei dieser Vielfalt ergibt sich, daß jeder dieser Algorithmen in bestimmten Bereichen seine Stärken, in anderen aber durchaus auch unübersehbare Schwächen hat. So sind zum Beispiel einige dieser Algorithmen auf Flächen beschränkt, andere hingegen bei Volumen sehr effizient. Aus dieser Erkenntnis heraus entstand die Idee, ein Hybridverfahren zu entwickeln, das versucht, von jedem Algorithmus primär dessen Stärken zu erkennen und daraus ein Gesamtsystem zu formen.
Hier muß nun natürlich auch berücksichtigt werden, daß die hier verwendeten Algorithmen teilweise verschiedene Repräsentationen der Szene verwenden, und deshalb versucht werden muß, unter anderem die Verwendung von Photonen mit der Zerlegung der Szene in finite Elemente und auch dem strahlenbasierten Ray Tracing Verfahren sinnvoll zu kombinieren. Hierzu wird der für diese Verfahren gut geeignete Ansatz der Lighting Networks [27] verwendet, bei dem durch verschiedene wohldefinierte Schnittstellen und Kombinationsobjekte die Interaktion von Algorithmen mit unterschiedlichen Beleuchtungsrepräsentationen möglich ist. So wird im Folgenden in dieser Arbeit mit Hilfe dieser Lighting Networks eine Kombination erarbeitet, die für jeden Teilbereich der globalen Beleuchtungssimulation jeweils den effizientesten der hier vorgestellten Algorithmen wählt und dadurch ein Gesamtverfahren erzeugt, das der Summe seiner Einzelobjekte überlegen ist.

Die weitere Arbeit ist wie folgt gegliedert. In Kapitel 2 werden die physikalischen und geometrischen Grundlagen von Flächen und Volumen im Kontext von globaler Beleuchtungssimulation kurz besprochen. Ferner werden hier die verwendeten Algorithmen auf theoretischer Basis vorgestellt.
In Kapitel 3 wird dann aufgezeigt, welche der benötigten Algorithmen schon vorhanden waren, welche davon erweitert werden mußten, und welche Teile des Hybridverfahrens vollständig neu entwickelt wurden.
In Kapitel 4 wird nun die tatsächliche Implementierung im Programmpaket Vision [26] vorgestellt, und es wird näher auf programmiertechnische Details eingegangen, wie z.B. die Aufteilung in einzelne Klassen.
In Kapitel 5 nun werden die einzelnen Verfahren anhand von Beispielsdatensätzen verglichen, ihre Stärken und Schwächen erörtert, und die verschiedenen möglichen Kombinationsmethoden bewertet.
In Kapitel 6 erfolgt schließlich eine Bewertung des gesamten Ansatzes, es wird hier aufgezeigt, inwieweit das hier vorgestellte Hybridverfahren die Erwartungen erfüllt hat und welche zusätzlichen Erweiterungen noch denkbar sind.
In Anhang A werden einige interne Parameter näher erklärt, die benötigt werden, um die implementierten Algorithmen von Vision aus verwenden zu können.
Im Anhang B sind abschließend Bildbeispiele zu sehen, die mit Vision und den verschiedenen möglichen Lighting Networks berechnet wurden.

Kapitel 2

Grundlagen und verwandte Arbeiten

Ziel dieser Arbeit ist die Kombination und der Vergleich der beiden Verfahren *Photon Maps* und *Volumen-Radiosity* zur effizienten und universellen Beleuchtungssimulation für Flächen und Volumenelemente. Hierzu wurde ein Hybridverfahren entworfen, das eine Kombination beider Ansätze darstellt und in jeder Beleuchtungssituation jeweils den Algorithmus wählt, der jeweils am besten geeignet ist.

Im folgenden werden in dieser Arbeit nur die Aspekte der *geometrischen* Optik berücksichtigt, bei den hier beschriebenen Algorithmen wird deshalb auf die Unterteilung der Lichtes in verschiedene Wellenlängen verzichtet, also auch auf Effekte, die durch Brechung des Lichtes entstehen. Da hierdurch auf spezielle Eigenschaften der *physikalischen* Optik nicht eingegangen wird, kann die Interaktion des Lichtes mit Flächen und Volumen vollständig mit Hilfe der linearen Transporttheorie [2, 3, 7] erklärt werden. Bei der geometrischen Optik wird das Licht vollständig durch die Menge der abgestrahlten Energie definiert, die sich innerhalb eines Intervalles in eine bestimmte Richtung bewegt.

2.1 Grundbegriffe der globalen Beleuchtungsberechnung

Hier werden als Einführung die Grundgrößen, die bei der globalen Beleuchtungsberechnung existieren, vorgestellt und definiert.

Strahlungsenergie: Ein Maß für die Menge an Energie δE, die während der Zeit dt von einem Flächenelement da in einen Raumwinkel $d\Omega$ in Richtung $\vec{n}$ gestrahlt wird, ist definiert durch

$$\delta E = I(x, \vec{n}) \cos\vartheta \, da \, d\Omega \, dt,$$

mit dem Winkel ϑ zwischen $\vec{n}$ und der Normalen auf da.

Ausgehende Flußdichte: Diese ist als der Strahlungsfluß definiert, der von einem Oberflächenelement da insgesamt verstrahlt wird. Bei opaquen Flächenelementen kann man diesen durch die Integration der spezifischen Intensität über die obere Halbkugel berechnen. Die ausgehende Flußdichte wird auch mit Radiosity B bezeichnet und ergibt sich aus

$$B = \frac{d\phi^{out}}{da} = \int_{\Omega+} I \cos\vartheta \, d\Omega,$$

wobei ϑ der Winkel zwischen der Normale von da und der ausgehenden Richtung ist. Der Cosinus-Term berücksichtigt hier die Flächenverkürzung, die das Verhältnis der tatsächlichen Größe von da zu der auf die jeweils ausgehenden Richtung projizierten Fläche darstellt.

Eingehende Flußdichte: Diese wird auch mit Irradiance D bezeichnet und wird analog zur Radiosity durch

$$D = \frac{d\phi^{in}}{da} = \int_{\Omega-} I \cos\vartheta \, d\Omega$$

berechnet.

Dem Vernachlässigen einzelner Frequenzbereiche des Lichtes durch die geometrische Optik kann auch durch die Integration über einige relevante Frequenzbereiche Rechnung getragen werden. Im Gegensatz zur physikalischen Optik kann man deshalb bei der geometrischen Optik auch von integrierter Intensität oder integrierter Radiance sprechen. Im Laufe dieser Arbeit wird aber hierauf verzichtet, die Begriffe Intensität, Radiance, usw. stehen jeweils für die Integration dieser. Für eine ausführliche Beschreibung der Eigenschaften der geometrischen Optik sei hier auch auf [9] verwiesen, im folgenden wird hier nur auf die für diese Arbeit relevanten Bereiche eingegangen.

Typ	*Definition*	*Einheit*
Strahlungsenergie	E	J
Strahlungsfluß	$\phi = \frac{dE}{dt}$	$W = \frac{J}{s}$
Strahlungsintensität	$J = \frac{d\phi}{d\Omega}$	$\frac{W}{sr}$
Ausgehende Flußdichte, Radiosity	$B = \frac{d\phi^{out}}{da}$	$\frac{W}{m^2}$
Eingehende Flußdichte, Irradiance	$D = \frac{d\phi^{in}}{da}$	$\frac{W}{m^2}$
Spezifische Intensität, Radiance	$I = \frac{dJ}{\cos\vartheta \, da}$	$\frac{W}{m^2 sr}$

Tabelle 2.1: Radiometrische Größen.

2.2 Interaktion von Licht mit Flächen und Volumen

2.2.1 Flächen, Reflexion und Transparenz

Die in für diese Arbeit vor allem interessanten Eigenschaften sind die Reflexion von Licht und die teilweise Transmission von Licht durch diese Flächen.

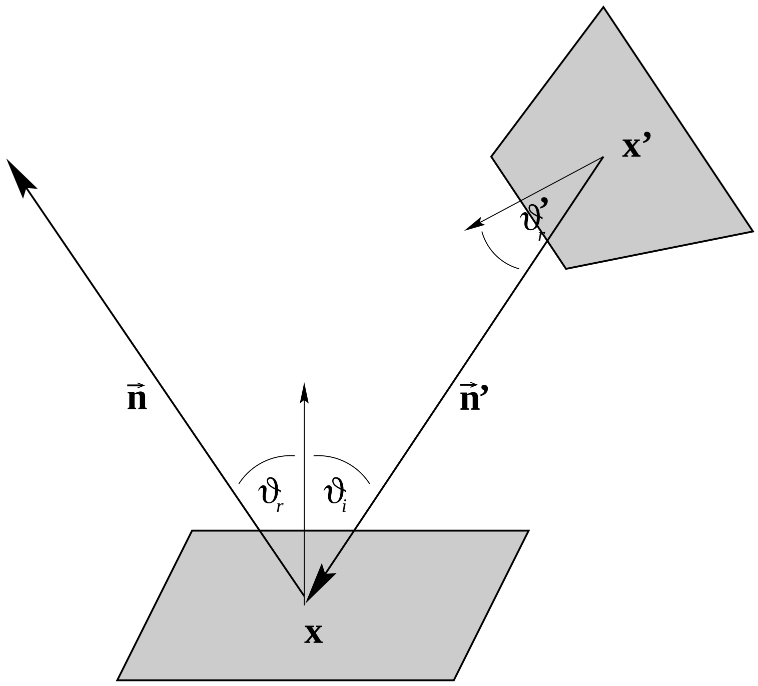

Abbildung 2.1: Reflexion von Licht auf Flächen.

Um die Reflexion von Licht an einer Fläche zu erörtern, ist es hilfreich, sich die bekannte Rendering-Gleichung [15] für Flächen zu vergegenwärtigen, in der die Radiance I durch

$$I(x,\vec{n}) = E(x,\vec{n}) + \int f_r(x,\vec{n}',\vec{n}) \frac{\cos\vartheta_r' \cos\vartheta_i}{|x-x'|^2} I(x',\vec{n}')da', x \in S \tag{2.1}$$

berechnet wird. $E(x,\vec{n})$ bezeichnet hier die Eigenemission an der Position x. Der Term $f_r(x,\vec{n}',\vec{n})$, der die *bidirectional reflectance distribution function* oder auch BRDF bezeichnet, ist durch das Verhältnis des Teiles der spezifischen Intensität dI definiert, die von einer Oberfläche in die Richtung $\vec{n}$ reflektiert wird aufgrund eines aus der Richtung $\vec{n}'$ einfallenden Teiles der Irradiance dD:

$$f_r(x,\vec{n}',\vec{n}) = \frac{dI}{dD} = \frac{dI(x,\vec{n})}{I^{in}(x,\vec{n}')\cos\vartheta_i d\Omega'}$$

Diese BRDF ist ein heute häufig verwendetes Modell in der Computergraphik, sie ist oberflächenspezifisch und somit von den Materialeigenschaften einer Fläche abhängig.
Es muß aber beachtet werden, daß dieses Modell die tatsächlichen physikalischen Eigenschaften nur annähert, das heißt daß dieses Modell z.B. nur einen exakten Reflexionspunkt kennt, die Interaktion des Lichtes mit Mikrofacetten an dieser Stelle also vollständig außer Acht läßt.

Das Verhältnis zwischen dem reflektierten und dem einfallenden Fluß wird mit Reflektanz ρ

$$\rho(\gamma_i, \gamma_r) = \frac{d\Phi^r(\gamma_r)}{d\Phi^i(\gamma_i)}$$

bezeichnet [7], wobei γ_i und γ_r jeweils der einfallende und reflektierte Raumwinkel sind. ρ kann Werte zwischen 0 und 1 annehmen, wobei die Extremwerte folgende Bewandtnis haben:

- **Reflektanz 0:** Das gesamte einfallende Licht wird verschluckt, dieser Fall ist uninteressant für die globale Beleuchtungsberechnung, da hierdurch ein Objekt nicht sichtbar ist.

- **Reflektanz 1:** Dies definiert einen idealen Reflektor, wie er aber in der Natur nicht auftritt. Auch dies ist für die globale Beleuchtungsberechnung ungeeignet, da hierdurch bei der Reflexion der Energiegehalt des Strahles nicht verringert wird und es deshalb zu einer unendlichen Rekursion kommen kann. Theoretisch lassen sich auch Werte größer 1 konstruieren, dies würde dann aber bedeuten, daß der reflektierte Fluß eines Objektes größer wäre als der einfallende Fluß.

Ein interessanter Spezialfall dieser Rendering-Gleichung ergibt sich bei sogenannten *Lambertschen* Flächen. Diese Flächen erscheinen aus allen Richtungen gleich hell. Deshalb ist hier $I(x)$ isotrop, also richtungsunabhängig. Hieraus folgt, daß das Licht in alle Richtungen gleich stark reflektiert wird, die Reflexion nicht von der einfallenden Richtung abhängt[1], was auch bedeutet, daß die zugehörige BRDF selbst isotrop auf solchen Flächen ist.
Solche *Lambertschen* Flächen kann man deshalb sehr gut durch eine Reflektanz $\rho(x) = B/D$ beschreiben, welche das Verhältnis der gesamt einfallenden Radiosity B zur ausfallenden Irradiance D bestimmt. Die Radiosity B ist hier definiert als

$$B = \int_{\Omega+} I(x) \cos\vartheta \, d\Omega = \pi I(x) = \pi f_r(x) \int_{\Omega-} I_{ein}(x, n') \cos\vartheta_i \, d\Omega',$$

die Irradiance D als

$$D = \int_{\Omega-} I_{ein}(x, \vec{n}') \cos\vartheta_i \, d\Omega'. \tag{2.2}$$

Hieraus ergibt sich, daß sich $\rho(x)$ zur BRDF f_r wie folgt verhält:

$$\rho = \frac{B}{D} = \pi f_r. \tag{2.3}$$

Wenn es in einer Szene nun nur *Lambertsche* Flächen gibt, kann man hier die Rendering-Gleichung 2.1 mit Hilfe von 2.3 zur Radiosity-Gleichung vereinfachen,

$$I(x) = E(x) + \frac{1}{\pi}\rho(x) \int \frac{\cos\vartheta'_r \cos\vartheta_i}{|x - x'|^2} I(x') da', x \in S. \tag{2.4}$$

Im Gegensatz zu Volumen (siehe Abschnitt 2.2.2) ist die Transparenz bei Flächen ziemlich einfach zu bestimmen. Man kann sich transparente Flächen als Volumen mit einer unendlich

[1]Der Term $\cos\vartheta_i$ ist hier nur für die Flächenverkürzung zuständig, $I(x)$ ist hier nicht vom Winkel zwischen Einfalls- und Ausfallsrichtung abhängig.

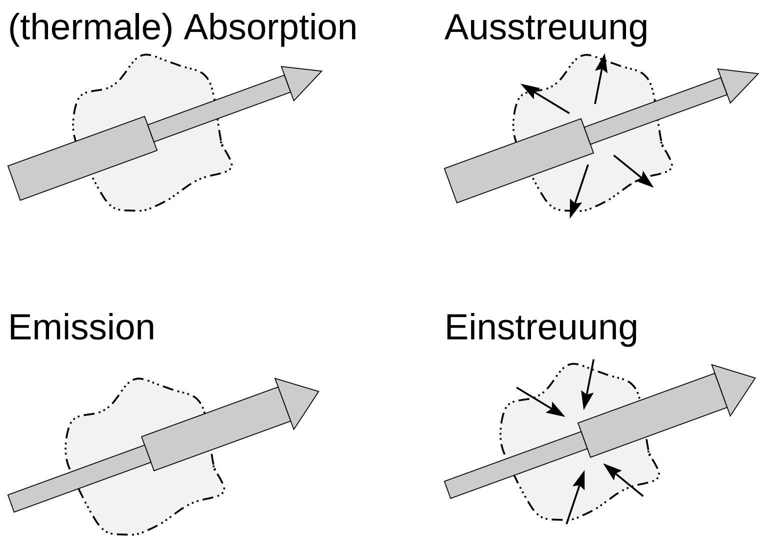

Abbildung 2.2: Verschiedene Möglichkeiten der Radiance Änderung entlang eines Strahles.

dünnen Ausdehnung vorstellen, deshalb wird die Transparenz hier einfach durch eine Konstante definiert.
Besondere Konstanten sind hier:

- **Transparenz 0:** Die Fläche ist opaque, also vollständig undurchsichtig.
- **Transparenz 1:** Die Fläche ist nicht sichtbar, das Licht kann diese ungehindert durchdringen.

2.2.2 Volumen, Streuung und Absorption

Mit dem Begriff Volumen ist hier die Approximation von Partikelwolken gemeint, die mit ihren winzigen Teilchen mit der Umgebung und sich selbst interagieren. Beispiele hierfür sind etwa Rauch, Nebel oder auch Feuer.
Wenn eine Strahlung solch ein Medium durchdringt, wird häufig der Energiegehalt dieser Strahlung verändert. Hier gibt es nun die in Abbildung 2.2 dargestellten Möglichkeiten [9]:

1. Verringerung der Radiance an Position x wird *Absorption* genannt, wobei hier zwischen *thermaler* Absorption und Ausstreuung unterschieden wird. Bei der thermalen Absorption wird die Energie in unsichtbare Wärme umgewandelt, dies wird mit dem *Absorptionskoeffizienten* $\kappa(x, \vec{n})$, der die Einheit $1/m$ hat, spezifiziert. Bei der *Ausstreuung* der Radiance an einer Position x reagiert ein Photon an einer bestimmten Stelle mit dem Volumen und wird von dort in eine andere Richtung umgelenkt. Der zugehörige Koeffizient ist der *Streuungskoeffizient* $\sigma(x, \vec{n})$.
   Das Verhältnis des gesamten Energieverlustes wird mit dem *Auslöschungskoeffizienten*

$$\chi(x, \vec{n}) = \kappa(x, \vec{n}) + \sigma(x, \vec{n}) \tag{2.5}$$

(auch *totaler* Absorptionskoeffizient) bezeichnet, das Verhältnis σ/χ zwischen der Ausstreuung und der totalen Absorption als *Albedo*.
Die Energiemenge, die ein Strahl mit einer Intensität $I(x, \vec{n})$ entlang der Richtung $\vec{n}$ in einem zylindrischen Volumen der Länge ds und des Querschnittes da verliert, ist durch

$$\delta E_a = \chi(x, \vec{n}) I(x, \vec{n}) ds\, da\, d\Omega\, dt$$

beschrieben.

2. Die Radiance kann auch durch *Eigenemission* erhöht werden, dies wird durch den *Emissionskoeffizienten* $\eta(x, \vec{n})$ dargestellt. Die während der Zeit dt durch ein zylindrisches Volumen der Länge ds und des Querschnittes da in einen Raumwinkel $d\Omega$ in eine Richtung $\vec{n}$ abgestrahlte Energiemenge ist gegeben durch

$$\delta E_e = \eta(x, \vec{n}) I(x, \vec{n})\, ds\, da\, d\Omega\, dt.$$

3. *Einstreuung* liegt vor, wenn eine aus einer anderen Richtung $\vec{n}'$ kommende Radiance $I(x, \vec{n}')$ an dieser Position in die Strahlrichtung umgeleitet wird:

$$I_{ein}(x, \vec{n}) = \int_\Omega p(x, \vec{n}, \vec{n}') I(x, \vec{n}') dn'. \tag{2.6}$$

Da die Streuung winkelabhängig ist, wird diese durch eine *Phasenfunktion* $p(x, \vec{n}, \vec{n}')$ beschrieben. Die Menge der ausgestrahlten Energie aus der Richtung $\vec{n}$ in die Richtung $\vec{n}'$ folgt aus

$$\delta E_{aus} = \sigma I\, ds\, da\, d\Omega\, dt \times \frac{1}{4\pi} p(x, \vec{n}, \vec{n}') d\Omega'.$$

Die normalisierte Phasenfunktion ergibt sich hier aus

$$\frac{1}{4\pi} \int p(x, \vec{n}, \vec{n}') d\Omega' \;= 1,$$

wobei durch den Faktor $\frac{1}{4\pi}$ die Normalisierungsbedingung der Einheitsphasenfunktion über eine Vollkugel auf 1 sichergestellt wird.
Hieraus ergibt sich, daß die einfachste Phasenfunktion für jede Richtung konstant 1 liefert, was bedeutet, daß die Streuung *isotrop* oder auch *diffus* ist. Alle anderen Fälle der Streuung werden demhingegen *anisotrop* oder auch *richtungsabhängig* genannt. Für Beispiele solcher spezifischeren Phasenfunktionen wird auf die Literatur verwiesen, so etwa die Rayleigh Phasenfunktion [3] oder die Henvey und Greenstein Phasenfunktionen [10].

Hieraus ergibt sich, daß die Änderung der Radiance I an Position x in Richtung $\vec{n}$

$$\frac{\delta I(x, \vec{n})}{\delta x} = \kappa(x) I_e(x, \vec{n}) + \sigma(x) I_{ein}(x, \vec{n}) - \kappa(x) I(x, \vec{n}) - \sigma(x) I(x, \vec{n})$$

ist, wobei I_e für die emittierte Radiance und I_{ein} für die eingestreute Radiance steht. Dies läßt sich mit Hilfe von Formeln 2.5 und 2.6 in

$$\frac{\delta I(x, \vec{n})}{\delta x} = \kappa(x) I_e(x, \vec{n}) + \sigma(x) \int_\Omega p(x, \vec{n}, \vec{n}') I(x, \vec{n}') dn' - \chi(x) I(x, \vec{n}) \tag{2.7}$$

umformen.
Hieraus ergibt sich, wenn man beide Seiten der Gleichung 2.7 entlang eines Strahles von x_0 bis x in Richtung $\vec{n}$ aufintegriert, folgende Integralgleichung für die Intensität I:

$$\begin{aligned} I(x,\vec{n}) &= \int_{x_0}^{x} \tau(x',x)\kappa(x')I_e(x',\vec{n})dx' && (2.8)\\ &+ \int_{x_0}^{x} [\tau(x',x)\sigma(x')\int_{\Omega} p(x',\vec{n},\vec{n}')I(x',\vec{n}')dn']dx' && (2.9)\\ &+ \tau(x_0,x)I(x_0,\vec{n}). && (2.10)\end{aligned}$$

$\tau(x_0,x)$ ist dabei die Transmission entlang des Strahles zwischen x_0 und x und über

$$\tau(x_0,x) = e^{-\int_{x_0}^{x}\chi(\xi)d\xi}$$

definiert.

2.3 Beleuchtungssimulation

In diesem Abschnitt werden die Grundlagen der für diese Arbeit wichtigen Basisalgorithmen, die es bei der globalen Beleuchtungssimulation gibt, auf theoretischer Basis vorgestellt.

2.3.1 Verschiedene Beleuchtungs- und Reflexionsarten

Einleitend werden hier die verschiedenen möglichen Arten der Beleuchtung von Objekten und der Reflexion von Licht an Objekten näher erklärt.

direkt: Direkte Beleuchtung ist der Lichtterm, der von einer Lichtquelle durch eine einmalige Reflexion über ein Objekt direkt beim Auge ankommt (siehe auch Abbildung 2.3, Pfade (1) und (2)).

indirekt: Indirekte Beleuchtung ist der Teil der Gesamtbeleuchtung, der über eine mindestens zweimalige Reflexion durch Objekte der Szene den Augpunkt erreicht (siehe auch Abbildung 2.3, Pfade (3) und (4)).

diffus: Diffuse Reflexion liegt dann vor, wenn das einfallende Licht in praktische alle möglichen Ausfallsrichtungen gleich stark reflektiert wird.

spekular: Spekulare Reflexion liegt dann vor, wenn das einfallende Licht nur in einem bestimmten, von der Einfallsrichtung abhängigen Bereich reflektiert wird, wobei die Größe dieses Bereiches häufig durch einen Exponenten definiert wird, der in das zumeist benutzte Phong-Modell eingesetzt wird. Die ideale spekulare Reflexion ist die *spiegelnde* Reflexion, hier wird das einfallende Licht in genau eine Ausfallsrichtung reflektiert.

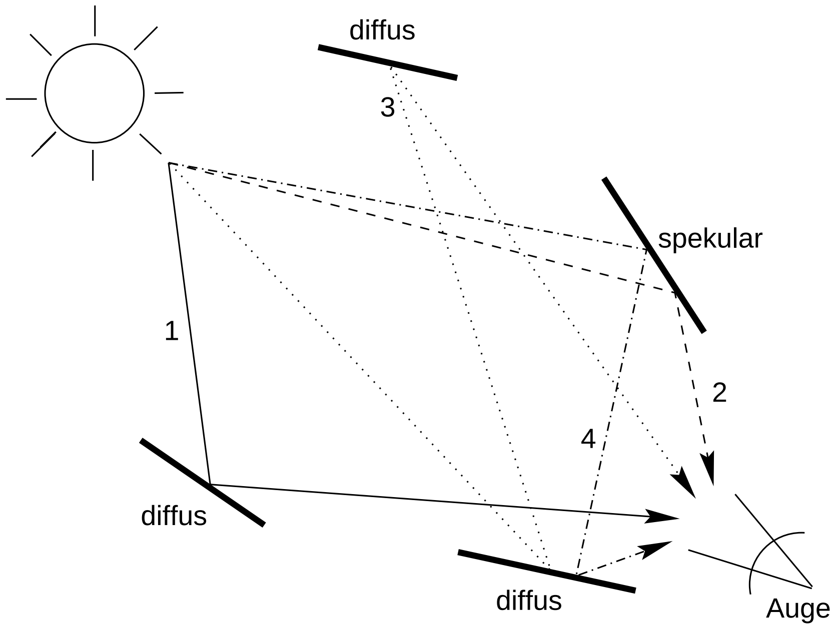

Abbildung 2.3: Verschiedene Lichtpfade, (1) und (2) sind direkte Beleuchtungspfade, (3) ist indirekte diffuse Beleuchtung und (4) ergibt eine Kaustik.

2.3.2 Raytracing

Bei Ray Tracing wird für jeden Pixel der Bildebene ein Strahl von Augpunkt durch diesen Bildpunkt in die Szene verschossen. Wenn ein Objekt getroffen wird, wird die Farbe durch die direkte Beleuchtung bestimmt, die durch Sichtstrahlen zu dem Lichtquellen ermittelt wird. Wenn auch die indirekte Beleuchtung bestimmt werden soll und das getroffene Objekt teilweise transparent oder reflektierend ist (siehe 2.2.1), dann müssen von dem Treffpunkt erneut Strahlen in die Szene verschossen werden. Ein Verfahren, das solche Strahlen verschickt, nennt man einen *rekursiven* Ray Tracer.
Hiermit kann unter anderem die *indirekte diffuse* Beleuchtung, die durch die diffuse Reflexion von Licht auf Flächen in den gesamten von dort sichtbaren Halbraum entsteht, durch diese Rekursion berechnet werden. Ein Problem hierbei ist aber, daß die so angestrahlten Objekte dieses Licht meist erneut reflektieren, was zu einer potentiell unendlichen Rekursion und einem exponentiellen Anwachsen der Anzahl der Strahlen führt.
Ein einfacher Algorithmus, der die Irradiance auf diese Art berechnen will, muß nun im konkreten Berechnungsfall, wenn die maximale Rekursionstiefe mit der Zahl R begrenzt wird und in jedem Rekursionsschritt S Strahlen in den Halbraum verschickt werden, insgesamt $(X \cdot Y) \cdot S^R$ Strahlen verschiessen, wobei $X \cdot Y$ die Auflösung der Bildebene darstellt, was zu einem exponentiellen Wachstum über die Rekursionstiefe R führt.
So werden zum Beispiel bei $R = 5$, $S = 512$ und 1024 x 768 Punkten in der Bildebene $2,8 \cdot 10^{19}$ Strahlen benötigt.

2.3.3 Radiosity

Ein Lösungsansatz, diffuse Interaktion bei Objekten zu berechnen, ist der Radiosity Ansatz, der 1984 an der Cornell Universität [8] entwickelt wurde.

2.3.3.1 Radiosity auf Flächen

Die Idee des Radiosity Ansatzes ist es, die Objekte in einzelne Teile (auch *Patches* genannt) zu zerlegen, auf denen die Beleuchtung dann als konstant angesehen wird. Die Interaktion erfolgt dann nur zwischen diesen einzelnen Patches und nicht pixel- oder strahlbasiert. Deshalb ist Radiosity ein Algorithmus über den Objektraum, die Lösung ist blickpunktunabhängig, das heißt eine einmal berechnete Szene kann in relativ kurzer Zeit aus vielen verschiedenen Ansichten betrachtet werden. Hierbei ist aber die Grundannahme wichtig, daß alle Flächen ideale Lambertsche Strahler (siehe auch Abschnitt 2.2.1) sind. Mit diesem Ansatz können also keine spekularen Reflexionen berechnet werden.
Die Radiosity selbst ist wie folgt definiert:

> Radiosity B ist die Energie pro Einheitsfläche, die einen Patch pro Zeiteinheit verläßt, also die Summe der emittierten und reflektierten Energie:

$$\text{Radiosity} \times \text{Fläche} = \text{emittierte Energie} + \text{reflektierte Energie},$$

$$B_i dA_i = E_i dA_i + R_i \int\limits_j B_j F_{ji} dA_j.$$

E_i ist hier die Energie, die vom Patch emittiert wird und R_i ist die Reflektivität dieses Patches. Die einfallende Energie ist die Energie, die am Patch i von allen anderen Patches der Szene einfällt. Dazu wird über die Patches der gesamten Umgebung der Term $B_j F_{ji} dA_j$ für jedes $j (j \neq i)$ integriert. F_{ji} ist hier eine Konstante, der sogenannte *Formfaktor*, der die Beziehung zwischen den Patches j und i angibt, welche von der geometrischen Lage dieser beiden Patches abhängt.
Unter der Annahme der Reziprozitätsbeziehung [24]

$$F_{ij} A_i = F_{ji} A_j$$

ergibt sich, wenn man durch dA_i teilt, die Radiosity-Formel 2.4 aus Abschnitt 2.2.1.

2.3.3.2 Volumen-Radiosity

Im folgenden wird nun ein Weg aufgezeigt, diesen Radiosity-Ansatz auf Volumen zu erweitern [23], ohne auf allgemeingültige Formeln zu verzichten. Dazu wird intern auf den Strahlungsfluß P umgestellt, um eine Interaktion zwischen Flächen und Volumen einheitlich darstellen zu können.
Die Emission in einem isotropen Volumen ergibt sich für einen Punkt $\vec{x}$ wie folgt:

$$\eta(\vec{x}) = q(\vec{x}) + \frac{1}{4\pi} \sigma(\vec{x}) \int\limits_\Omega I(\vec{x}, \vec{n})\, d\omega,$$

und die Transportgleichung für isotrope Emission und Streuung lautet:

$$I(\vec{x},\vec{n}) = I^{(\mathrm{out})}(\vec{x}_A,\vec{n})e^{-\tau(\vec{x}_A,\vec{x})} + \int\limits_{\vec{x}_A}^{\vec{x}} \eta(\vec{x}')e^{-\tau(\vec{x}',\vec{x})}ds'.$$

Die Volumenradiosity-Gleichung ergibt sich hieraus als:

$$\begin{aligned}\eta(\vec{x}) = q(\vec{x}) \quad &+ \quad \tfrac{1}{4\pi}\,\sigma(\vec{x}) \int\limits_{\Omega} I^{(\mathrm{out})}(\vec{x}_A,\vec{n})\, e^{-\tau(\vec{x}_A,\vec{x})} d\omega \\ &+ \quad \tfrac{1}{4\pi}\,\sigma(\vec{x}) \int\limits_{\Omega} \int\limits_{\vec{x}_A}^{\vec{x}} \eta(\vec{x}')\, e^{-\tau(\vec{x}',\vec{x})}\, ds' d\omega,\end{aligned} \tag{2.11}$$

wobei die Grenzbedingungen (bei Flächen) für Gleichung 2.11 folgende sind:

$$\begin{aligned}I^{(\mathrm{out})}(\vec{x}_A) = I^{(q)}(\vec{x}_A) \quad &+ \quad \tfrac{1}{\pi} p(\vec{x}_A) \int \cos\vartheta_i\, I^{(\mathrm{out})}(\vec{x}'_A,\vec{n})\, e^{-\tau(\vec{x}'_A,\vec{x}_A)} d\omega \\ &+ \quad \tfrac{1}{\pi} p(\vec{x}_A) \int \int\limits_{\vec{x}_A}^{\vec{x}'_A} \cos\vartheta_i\, \eta(\vec{x}')e^{-\tau(\vec{x}',\vec{x}_A)} ds'\, d\omega.\end{aligned} \tag{2.12}$$

Das Volumen V und die Grenzflächen A werden hier in eine finite Anzahl von Patches zerlegt:

$$\begin{aligned}V &= V_0 \oplus V_1 \oplus V_2 \oplus \cdots \oplus V_{N_V-1},\\ A &= A_0 \oplus A_1 \oplus A_2 \oplus \cdots \oplus A_{N_A-1}.\end{aligned}$$

Die Integrationsvariablen in Gleichungen 2.11 und 2.12 kann man durch differentielle Patches ersetzen:

$$d\omega = \frac{\cos\vartheta'_r}{|\vec{x}-\vec{x}'_A|^2}\, da',\; ds'\, d\omega = \frac{1}{|\vec{x}-\vec{x}'_A|^2}\, dV'$$

Nun kann man die Integrale über die Kugeln durch finite Summen über alle Patches ersetzen. Hier muß jetzt aber ein Sichtbarkeitsfaktor eingeführt werden, der die Verdeckung von Punkten durch Flächenpatches oder bei teilweise transparenten Patches die Abschwächung des Strahles berücksichtigt. Dieser Sichtbarkeitsfaktor hat einen Wertebereich von $[0:1]$ und die Einheit 1.

Aus den Gleichungen 2.11 und 2.12 ergeben sich, wenn über die Empfänger-Domäne integriert wird und dann das Mittel über die Division durch die Größe der Empfänger-Domäne berechnet wird, folgende Formeln:

$$\begin{aligned}\eta(V_i) = q_{V_i} \quad &+ \quad \sum_{j'=0}^{N_A} \left[\tfrac{1}{4\pi V_i} \sigma_{V_i} \int\limits_{V_i} \int\limits_{A_{j'}} \tfrac{\cos\vartheta'_r}{|\vec{x}'_A-\vec{x}|^2} H(,\vec{x},\vec{x}'_A)\, e^{-\tau(\vec{x},\vec{x}'_A)}\, da' dV \right] I_{A_{j'}} \\ &+ \quad \sum_{i'=0}^{N_V} \left[\tfrac{1}{4\pi V_i} \sigma_{V_i} \int\limits_{V_i} \int\limits_{V_{i'}} \tfrac{1}{|\vec{x}'-\vec{x}|^2} H(\vec{x},\vec{x}')\, e^{-\tau(\vec{x},\vec{x}')}\, dV' dV \right] \eta_{V_{i'}}\end{aligned} \quad , \tag{2.13}$$

$$\begin{aligned}I(A_j) = I_j^{(q)} \quad &+ \quad \sum_{j'=0}^{N_A} \left[\tfrac{1}{\pi A_j} p_j \int\limits_{A_j} \int\limits_{A_{j'}} \tfrac{\cos\vartheta_i\, \cos\vartheta'_r}{|\vec{x}'_A-\vec{x}_A|^2} H(\vec{x}'_A,\vec{x}_A)\, e^{-\tau(\vec{x}'_A,\vec{x}_A)}\, da' da \right] I_{A_{j'}} \\ &+ \quad \sum_{i'=0}^{N_V} \left[\tfrac{1}{\pi A_j} p_j \int\limits_{A_j} \int\limits_{V_{i'}} \tfrac{\cos\vartheta'_r}{|\vec{x}'-\vec{x}_A|^2} H(\vec{x}',\vec{x}_A)\, e^{-\tau(\vec{x}',\vec{x}_A)}\, dV' da \right] \eta_{V_{i'}}\end{aligned} \quad , \tag{2.14}$$

wobei

$$\tau(\vec{x}_1, \vec{x}_2) = \sum_i \chi_{V_i} \cdot s_{V_i}, \ i \in \mathrm{Trav}(\vec{x}_1, \vec{x}_2).$$

ist.

Für Radiosity-Berechnungen zwischen nur Flächen oder nur Volumen sind die Gleichungen 2.13 und 2.14 durchaus hilfreich. Wenn man aber die Radiosity in gemischten Umgebungen mit sowohl Flächen als auch Volumen berechnen will, stellt man fest, daß sich die zwei Größen Volumenemission η und Flächenradiance I nicht so einfach miteinander mischen lassen, da die verschiedenen physikalischen Größen von η (W/m^3) und I (W/m^2) jeweils eine Konvertierung zwischen den einzelnen Formfaktoren erfordern würden.
Wenn man aber die Gleichungen 2.13 und 2.14 jeweils mit ihren Empfänger-Domänen V_i und A_j multipliziert, bekommt man eine einheitliche Darstellung über den Strahlungfluß P beider Formeln. Wenn man jetzt noch die Terme $\eta_{V_{i'}}$ und $I_{A_{j'}}$ auf den rechten Seiten durch $P_{V_{i'}}$ und $P_{A_{j'}}$ ersetzt, führt dies zu folgendem einfachen linearen System:

$$P_p = P_p^{(\mathrm{src})} + \sum_{q=0}^{N} F_{p \leftarrow q} P_q, \tag{2.15}$$

wobei $P^{(\mathrm{src})}$ der emittierte Quellfluß eines Elementes und N die Gesamtanzahl von Flächen- und Volumenelementen ist. Die Formfaktoren $F_{p \leftarrow q}$ werden in Abhängigkeit der Art des Senders q und des Empfängers p wie folgt berechnet, wobei hier schon die Reflexions- und Streuungskoeffizienten enthalten sind, sich deshalb alle Einheiten zu 1 normieren:

$$\begin{aligned}
F_{A_j \leftarrow A_{j'}} &= \frac{p_{A_j}}{\pi A_{j'}} \int\limits_{A_j} \int\limits_{A_{j'}} \frac{\cos\vartheta'_r \cos\vartheta_i}{|\vec{x}_A - \vec{x}'_A|^2} H(\vec{x}'_A, \vec{x}_A)\, e^{-\tau(\vec{x}'_A, \vec{x}_A)}\, da'\, da \\
F_{A_j \leftarrow V_{i'}} &= \frac{p_{A_j}}{\pi V_{i'}} \int\limits_{A_j} \int\limits_{V_{i'}} \frac{cos\vartheta_i}{|\vec{x}_A - \vec{x}'|^2} H(\vec{x}', \vec{x}_A)\, e^{-\tau(\vec{x}', \vec{x}_A)}\, dV'\, da \\
F_{V_i \leftarrow A_{j'}} &= \frac{\sigma_{V_i}}{4\pi A_{j'}} \int\limits_{V_i} \int\limits_{A_{j'}} \frac{\cos\vartheta'_r}{|\vec{x} - \vec{x}'_A|^2} H(\vec{x}'_A, \vec{x})\, e^{-\tau(\vec{x}'_A, \vec{x})}\, da'\, dV \\
F_{V_i \leftarrow V_{i'}} &= \frac{\sigma_{V_i}}{4\pi V_{i'}} \int\limits_{V_i} \int\limits_{V_{i'}} \frac{1}{|\vec{x} - \vec{x}'|^2} H(\vec{x}', \vec{x})\, e^{-\tau(\vec{x}', \vec{x})}\, dV'\, dV
\end{aligned}$$

2.3.4 Photon Maps

Die Grundalgorithmen des hier beschriebenen Photon Maps Verfahrens wurden von Henrik Wann Jensen [11, 12, 13, 14] entwickelt. Der Gedanke hinter diesem Verfahren ist es, Photonen von den Lichtquellen in eine Szene zu verschicken. Wenn diese Photonen nun auf eine Fläche treffen, werden sie dort abgelegt und dann entsprechend der zugehörigen BRDF mit Hilfe von Russian Roulette [1] reflektiert. Hierdurch entsteht eine grobe Repräsentation der Szene, mit deren Hilfe man nun mit verschiedenen Algorithmen die tatsächliche Beleuchtung bestimmen kann.

2.3.4.1 Erzeugen einer Photon Map

Dies ist der erste Teil des Photon Maps Algorithmus, in dem die grobe Repräsentation der Lichtverteilung in der Szene durch Verschicken von Photonen von den Lichtquellen erzeugt wird.
Die maximal N zu verschiessenden Photonen werden auf die M Lichtquellen im Verhältnis zu deren jeweiligen Helligkeit aufgeteilt und dann von dort in die Szene verschickt. Wenn ein Photon eine Fläche trifft, wird es in einem zugehörigen Kd-Tree an dieser Position abgelegt, und es wird mit Hilfe von Russian Roulette [1] entschieden, ob dieses hier absorbiert oder reflektiert wird. Bei einer erfolgten Reflexion wird die neue Richtung durch die BRDF dieses Objektes bestimmt. Falls das Photon sich durch ein Volumen bewegt, kann dieses entweder ungehindert durchdringen, oder es interagiert an irgendeiner Position x mit diesem Medium. Dies wird über die kumulative Wahrscheinlichkeits-Dichtefunktion

$$F(x) = 1 - \tau(x_s, s) = 1 - e^{-\int_{x_s}^{x} \chi(\xi) d\xi}$$

entschieden, wobei x_s der Eintrittspunkt in das Volumen ist.
Im Falle einer Interaktion mit diesem Medium wird das Photon, falls es nicht direkt von einer Lichtquelle kommt, an dieser Position im zugehörigen Kd-Tree abgelegt. Auch hier wird durch Russian Roulette entschieden, ob das Photon absorbiert oder gestreut wird, wobei die Wahrscheinlichkeit für eine Streuung des Photons durch das Albedo σ/χ (siehe 2.2.2) berechnet wird. Falls sich eine Streuung des Photons ergibt, wird die neue Richtung über die Phasenfunktion p an dieser Stelle ermittelt.
Um die nachfolgende Rekonstruktion zu erleichtern, werden zwei verschiedene Photon Maps angelegt, eine zur indirekten diffusen Beleuchtungsberechnung (siehe 2.3.4.4) und eine zur Berechnung von Kaustiken (siehe 2.3.4.3). Die Anzahl der zu durchsuchenden Photonen kann auf diese Weise bei beiden Teilalgorithmen verringert werden. In der Kaustiken Photon Map werden im Gegensatz zu der indirekt diffusen Photon Map nur Photonen abgelegt, die über n spekulare Objekte auf ein diffuses Objekt treffen. Dabei wird hier eine höhere Anzahl von Photonen als für die diffuse Photon Map verschickt, da die Rekonstruktion der Kaustiken direkt aus dieser Map erfolgt, während bei der diffusen Photon Map noch ein Zwischenschritt erfolgen muß.

2.3.4.2 Rekonstruktion der Beleuchtung

In diesem zweiten Schritt gibt es nun die folgenden zwei unterschiedlichen Teilalgorithmen.

2.3.4.3 Berechnung von Kaustiken

Kaustiken entstehen, wenn Licht über n spekulare Flächen auf eine diffuse Fläche reflektiert wird, welche dann direkt für den Betrachter sichtbar ist (siehe auch Abbildung 2.3, Pfad 4). Diese Kaustiken kann man schlecht mit herkömmlichen Ray Tracing Verfahren visualisieren, da man eine sehr große Anzahl von Strahlen für jede spekulare Fläche benötigen würde. In Gegensatz zum reinen Ray Tracing ist das Photon Maps Verfahren für die Visualisierung dieses Effekts sehr gut geeignet, da die verschossenen Photonen die Kaustiken sehr gut annähern

[12]. Wenn an einer Position P die Helligkeit berechnet werden soll, werden an dieser Position die n nächsten Photonen aus der Kaustiken Photon Map gesucht, wobei n über einen maximalen Suchradius begrenzt wird, um die Lokalität dieser Berechnung zu gewährleisten. Die Berechnung der Kaustiken erfolgt nun durch die Ermittlung der Dichte dieser Photonen, wobei sich die Formeln für Flächen und Volumen an dieser Stelle etwas unterscheiden.

- Bei Flächen erfolgt die Rekonstruktion über

$$\begin{aligned} I_r(x,\vec{n}) &= \int\limits_{\Omega} f_r(x,\vec{n}',\vec{n}) \frac{d^2\Phi(x,\vec{n}')}{dA\,dn'} dn' \\ &\approx \sum_{p=1}^{n} f_r(x,\vec{n}'_p,\vec{n}) \frac{\Delta\Phi_p(x,\vec{n}'_p)}{\pi r^2}, \end{aligned}$$

  wobei f_r die BRDF, r die Entfernung des n-ten Photons, x der Punkt auf der Fläche, $\Delta\Phi_p$ der Fluß des Photon p in Richtung $\vec{n}'_p$ und $\vec{n}$ die Betrachtungsrichtung ist. Die n Photonen mit den Betrachtungsrichtungen $\vec{n}'_p$ und den Strahlenflüßen $\Delta\Phi_p$ sind hier die Photonen mit der kürzesten Entfernung zu x. Man kann sich dies auch so vorstellen, daß man eine Kugel mit Mittelpunkt x genau so weit ausdehnt, bis sie n Photonen enthält.

- Bei Volumen hingegen ergibt sich die Formel zur Berechnung der Radiance aus

$$\begin{aligned} I_r(x,\vec{n}) &= \int\limits_{\Omega} f_r(x,\vec{n}',\vec{n}) \frac{d^2\Phi(x,\vec{n}')}{\sigma(x) dn'\,dV} dn' \\ &= \frac{1}{\sigma(x)} \int\limits_{\Omega} f_r(x,\vec{n}',\vec{n}) \frac{d^2\Phi(x,\vec{n}')}{dV} \\ &\approx \frac{1}{\sigma(x)} \sum_{p=1}^{n} f_r(x,\vec{n}'_p,\vec{n}) \frac{\Delta\Phi_p(x,\vec{n}'_p)}{\frac{4}{3}\pi r^3}, \end{aligned}$$

  wobei $\sigma(x)$ der zugehörige Streuungskoeffizient ist. Da die Einfallsrichtung $\vec{n}'_p$ bekannt ist, kann man über diesen Ansatz auch anisotrope Phasenfunktionen für Volumen verwenden.

Wenn man diese beiden Formeln vergleicht, fällt als Hauptunterschied auf, daß bei Flächen die Dichte über die projizierte Fläche berechnet wird, während bei Volumen über den gesamten Kugelinhalt integriert wird.
Wenn durch einen adaptiven Ray-Marching Algorithmus [5] ein Volumen traversiert wird, dann wird die Radiance iterativ in jedem Schritt entlang des Strahles ermittelt und so die Gesamtradiance berechnet. Die Radiance $I(x,\vec{n})$ aus Gleichung 2.8 wird iterativ an den Punkten x_k durch folgende Gleichung approximiert:

$$\begin{aligned} I(x_k,\vec{n}) &\approx \kappa(x_k) I_e(x_k,\vec{n}) \Delta x_k \\ &+ \sigma(x_k) I_i(x_k,\vec{n}) \Delta x_k \\ &+ e^{-\chi(x_k)\Delta x_k} I(x_{k-1},\vec{n}), \end{aligned}$$

wobei $\Delta x_k = |x_k - x_{k-1}|$ die Schrittweite und x_0 entweder die Rückseite des Volumens oder der näheste Schnittpunkt mit einer Fläche ist.

2.3.4.4 Berechnung von indirekter diffuser Beleuchtung

Ein anderer Bereich der globalen Beleuchtung, der mittels Photon Maps abgedeckt werden kann, ist die *indirekte diffuse* Beleuchtung (siehe auch Abbildung 2.3, Pfad 3).
Problematisch bei der Berechnung der indirekten Beleuchtung ist, daß man eine sehr große Zahl von Strahlen benötigt, wenn diese mit einem herkömmlichen Verfahren berechnet werden soll (siehe 2.3.2). Auch pure Monte Carlo Methoden, wie z.B. das von Lafortune und Willems [17] entwickelte bidirektionale *Path Tracing for Paticipating Media* benötigen sehr viel Rechenzeit, um Bilder für Szenen, die Volumen enthalten, ohne sichtbares Rauschen zu berechnen.
Das Photon Map Verfahren für indirekte diffuse Beleuchtung versucht nun, durch geschicktes Ausnutzen der Informationen aus der zugehörigen Photon Map die Anzahl der Strahlen zu verringern.

Wenn man die Irradiance D an einer Position x berechnen will, muß man über die gesamte einfallende Radiance I_{in} integrieren. Dieses Integral wird in der Praxis durch eine diskrete Anzahl von Strahlen von der Position P weg in den sichtbaren Halbraum approximiert. Mit diesem Ansatz kann Formel 2.2 durch

$$D = \int_{\Omega-} I_{in}(x, \vec{n}') \cos\vartheta_i \, d\Omega' \approx \sum_{p=1}^{n} I_{in}(x, \vec{n}'_p) \cos\vartheta_i$$

angenähert werden, wobei $\vec{n}'_p$ die verschiedenen Richtungen in den Halbraum darstellen. Je geschickter diese Richtungen gewählt werden, desto besser ist dann auch die Approximation. Anstelle einer uniformen Verteilung dieser Strahlen werden diese Richtungen häufig über ein Importance Sampling bestimmt. Dazu wird die zugehörige BRDF ausgewertet, da man aus ihr die Richtungen bestimmen kann, die am meisten zum reflektierten Licht beitragen. Dies funktioniert für spekulare Flächen auch sehr gut, auf diffusen Oberflächen kann diese Unterscheidung aber nicht getroffen werden, da hier Licht aus allen einfallenden Richtungen zur ausfallenden Beleuchtung beitragen kann. Hier müßte der Halbraum wieder uniform abgetastet werden, was vor allem dann sehr ineffizient ist, wenn die eintreffende Radiance stark ungleichförmig, also auf kleine Bereiche konzentriert ist. Um solche Fälle abdecken zu können, braucht man ein approximatives Wissen über die Lichtverteilung in der Szene, das nun über eine Photon Map erlangt werden kann.

Es wird hierzu eine $m \cdot n$ große Umgebungskarte (siehe Abbildung 2.4) erzeugt, die den von dem Betrachtungspunkt x sichtbaren Halbraum auf ein Einheitsquadrat abbildet. In dieses Quadrat wird der jeweilige Fluß $\Delta\Phi_{i,p}$ der N nähesten Photonen eingetragen, wobei die Position in diesem Einheitsquadrat über die Einfallsrichtung des jeweiligen Photons p_i ermittelt wird. Wenn nun angenommen wird, daß die N Photonen das Objekt jeweils an der Position x treffen, ergibt sich der Beitrag jedes Photons p zum reflektierten Fluß $\Delta\Phi_r$ aus

$$\Delta\Phi_r = f_r(\vartheta_p, \phi_p)\Delta\Phi_{i,p}(\vartheta_p, \phi_p),$$

wobei ϑ_p und ϕ_p die Position des Photons p im Einheitsquadrat ist.
Uber diese Umgebungskarte werden die Richtungen der Strahlen, die vom Punkt x aus erneut

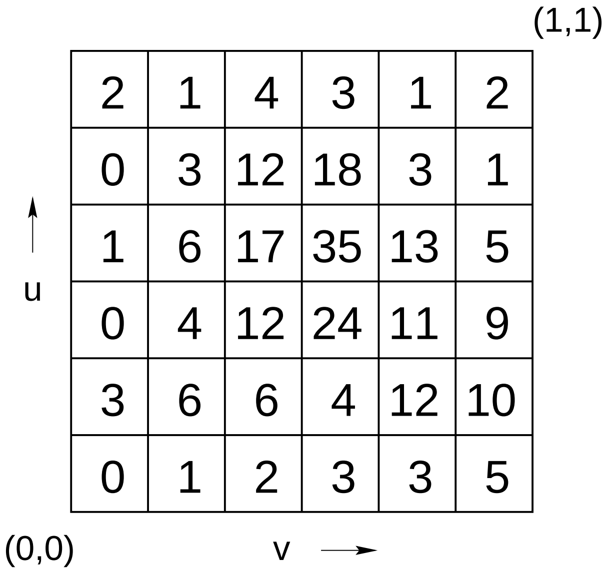

Abbildung 2.4: Eine Umgebungskarte, die durch Photonen erzeugt wurde.

in den Halbraum verschickt werden, ermittelt. Hierbei werden proportional zu den Energiemengen, die in den jeweiligen Kartenzellen der Umgebungskarte angehäuft wurden, Werte berechnet, die die Wahrscheinlichkeit angeben, mit der ein Strahl in die zugehörige Richtung verschickt wird. In dem so über die Wahrscheinlichkeitsfunktion ausgewählten Bereich des Halbraumes wird zufällig eine Richtung ermittelt und ein Strahl dorthin verschossen.

2.3.5 Irradiance Gradients

Ein anderer Ansatz, die Anzahl der Strahlen bei der Berechnung der *indirekten diffusen* Beleuchtung zu verringern, ist der Irradiance Gradients Algorithmus, der ursprünglich von Gregory J. Ward [34, 33] entwickelt und in seinem Raytracing-Paket Radiance [22] implementiert wurde. Der Ansatz stützt sich auf folgende Prämissen:

- Da reflektierende Flächen in einer normalen Szene weit verteilt sind, benötigt man viele Strahlen, um die indirekte Beleuchtung zu berechnen.

- Ein berechneter indirekter diffuser Beleuchtungswert ist unabhängig vom Beobachtungpunkt nach der Lambertschen Annahme [16].

- Die indirekte diffuse Beleuchtung ändert sich auf einer Fläche meist nur langsam, da der stärker variierende direkte Lichtanteil und die dazugehörigen Schatten schon durch die direkte Beleuchtungsberechnung (z.B. durch Ray Tracing) berechnet wurden.

Als direkte Schlußfolgerung ergibt sich, daß es sinnvoll ist, einen einmal berechneten Irradiance-Wert durch Caching zu sichern und dann durch geschickte Kohärenzausnutzung in der näheren Nachbarschaft wiederzuverwerten. Die vorhandenen Änderungen der Irradiance

in einer Szene werden durch Gradienten erkannt, die die 1. Ableitung der Irradiance darstellen. Zur besseren Verwendbarkeit werden zwei getrennte Gradienten entwickelt, einerseits ein Translationsgradient, der die Änderung bei einer Bewegung des Betrachtungspunktes angibt und andererseits ein Rotationsgradient, der bei einer Drehung der Normalen der Betrachtungsposition die Änderung wiedergibt. Die Interpolation eines neuen Wertes erfolgt nun über eine gewichtete Mittelung der in Frage kommenden gecachten Irradiance Werte mit Hilfe dieser Gradienten. Für die genauere Beschreibung der Berechnung und Anwendung dieser Gradienten und auch der Interpolation der Irradiance-Werte wird hier auf [19] verwiesen.
Aus diesen Überlegungen ergibt sich, daß der gesamte Algorithmus sich in zwei Teile gliedert, einer primären und einer sekundären Methode:

```
Wenn
     gültige Wert(e) in der Nähe der Berechnungspunkte existieren
dann
     wende sekundäre Methode an
sonst
     wende primäre Methode an
```

Vereinfacht ausgedrückt werden bei der primären Methode neue Werte durch Verschicken eines Strahles erzeugt, bei der sekundären Methode neue Werte durch Interpolation aus bereits vorhandenen gewonnen.

2.3.6 Lighting Networks und Lichtpfade

Um die verschiedenen oben beschriebenen Algorithmen, die häufig nur einen Teilbereich der in einer Szene vorkommenden gesamtem Beleuchtung berechnen, zu einem einzigen Hybridverfahren vereinigen zu können, das möglichst viele Teile der Gesamtbeleuchtung berücksichtigt, müssen diese Verfahren gekoppelt werden. Natürlich möchte man keinen Lichtanteil doppelt berechnen, deshalb ist es wichtig, die Lichtpfade, die die einzelnen Algorithmen berechnen, zu analysieren und zu vergleichen. Die für diese Arbeit wichtigsten Lichtpfade sind in Tabelle 2.2 mit der von Heckbert entwickelte Notation aufgezeigt, Beispiele für zugehörige Strahlen in Abbildung 2.3, wobei L hier die Lichtquelle, E den Augpunkt, D eine diffuse Fläche, S eine spekulare Fläche, $|$ den *oder* Operator $^+$ das mindestens einmalige Auftreten eines Elementes und * das beliebig häufige Auftreten eines Elementes darstellt.
Bei den in dieser Arbeit verwendeten Algorithmen ergeben sich folgende Auswahlmöglichkeiten:

direkt diffuse Lichtpfade: Diese können entweder durch Ray Tracing oder Radiosity berechnet werden.

direkt spekulare Lichtpfade: Für diese bietet sich Ray Tracing an, Radiosity kann nur mit diffusen Objekten verwendet werden.

Strahl	*Lichtpfad*	*Art der Beleuchtung*
1	LDE	direkt, diffus
2	LSE	direkt, spekular
3	$LS^*D(S\|D)^*DE$	indirekt, diffus
4	LS^+DE	indirekt, Kaustik

Tabelle 2.2: die Lichtpfade der Abbildung 2.3

indirekt diffuse Lichpfade: Hierfür ist die Auswahl groß, es kann entweder pures Ray Tracing (in der Regel aber nicht praktikabel, siehe 2.3.2), Radiosity, Photon Maps, oder Irradiance Gradients gewählt werden.

Kaustiken: Photon Maps ist das einzige der hier vorgestellten Verfahren, das diese berechnen kann.

Da nun geklärt ist, welche Algorithmen miteinander kombiniert werden können, werden diese zu einem Netzwerk vereinigt. Hierfür wird hier der Ansatz der Lighting Networks [27] verwendet, mit dem auch Algorithmen, die die Beleuchtung auf verschiedene Arten berechnen (z.B. Ray Tracing mit Strahlen und Photon Maps mit Photonen), über spezielle Konvertermodule verbunden werden können.
In einem Lighting Network gibt es nun die Möglichkeit, diese verschiedenen Verfahren entweder sequentiell oder parallel zu verbinden (siehe Abbildung 2.5). Bei einer sequentiellen Verknüpfung liefert der eine Algorithmus Daten, die ein anderer als Eingabewerte verarbeiten kann. Bei einer parallelen Schaltung der Algorithmen hingegen berechnen beide unabhängige Lichtanteile und deren Ergebnisse werden dann über ein Kombinationsmodul vereinigt.

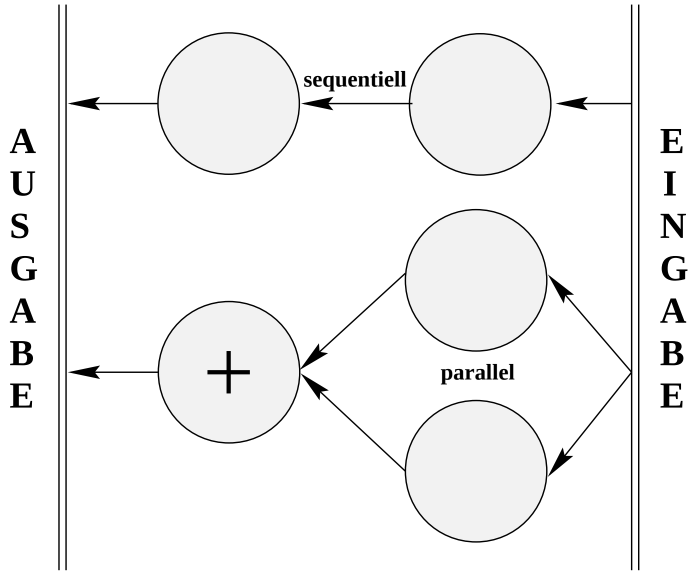

Abbildung 2.5: Parallele und sequentielle Algorithmen zur Beleuchtungsberechnung.

Kapitel 3

Umsetzung der Verfahren

3.1 Integration in Vision

Das Programmpaket Vision [26] wurde ursprünglich von Philipp Slusallek im Rahmen seiner Doktorarbeit entwickelt. Vision selbst kann in drei große Bereiche eingeteilt werden, in Rendering (Strahlverfolger), Global Illumination (globale Beleuchtung) und Scene Description (Szenenbeschreibung), wobei sich der Bereich Global Illumination noch einmal in Lighting (Beleuchtung), Shader, LightSource Shader und Volume Lighting (Volumenbeleuchtung) unterteilt.
Ein Lighting-Objekt wiederum kann aus mehreren LightOps bestehen, die dann mit Hilfe der Skriptsprache Tcl/Tk zu größeren Lighting Networks verbunden werden können. Die hier vorgestellten Argorithmen zur Beleuchtungsberechnung werden dementsprechend als eigene LightOps in dieses Programmpaket eingefügt, die benötigte Interaktion zwischen diesen einzelnen Verfahren mit Tcl/Tk über das Lighting Networks Konzept realisiert.

3.1.1 Lighting Networks

Lighting Networks sind notwendig, wenn man mehrere LightOps, die jeweils nur einen Teilbereich der globalen Beleuchtung abdecken, zu einem größeren Gesamtverbund vereinigen möchte [27]. Da man nicht erwarten kann, daß ein einzelnes Datenformat der Beleuchtung als Schnittstelle für alle Arten von Beleuchtungsinformationen der verschiedenen Algorithmen ausreichend ist, existiert das Konzept der Beleuchtungsbasen. Mit Hilfe dieses Konzeptes können die an verschiedenen LightOps Anfragen über die verschiedenen Beleuchtungsarten, die diese von weiteren LightOps empfangen können, gestellt werden. Es wird hier zwischen möglicher Eingabe und möglicher Ausgabe unterschieden, wobei folgende für diese Arbeit wichtigen Typen zur Verfügung stehen:

- **Punktsampling Basis:** Hierüber wird ein Punktsampling Interface definiert, das heißt es können Anfragen über einzelne Strahlen erfolgen.

- **Diffuse Punktsampling Basis:** Auch dies ist ein Punktsampling Interface, es wird hierüber aber nur indirekte diffuse Beleuchtung ausgetauscht.

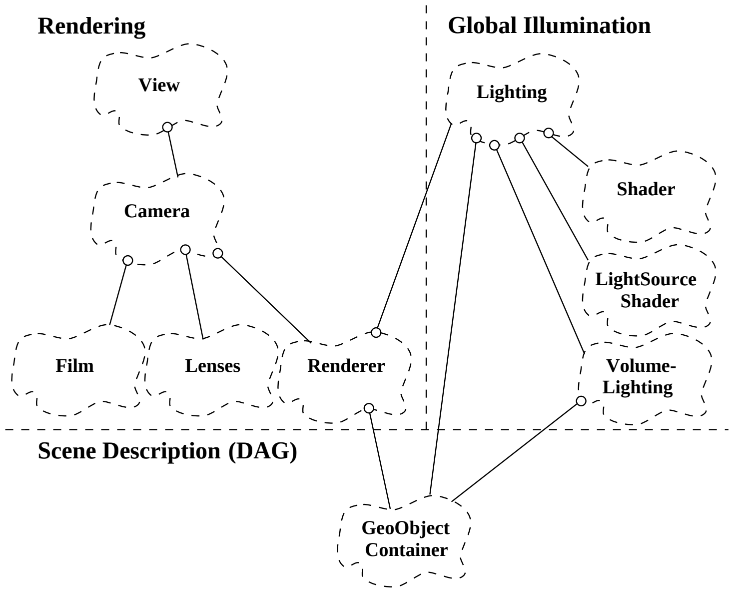

Abbildung 3.1: Die Struktur von Vision

- **Kd-Tree Basis:** Dies dient zum Austausch eines Kd-Trees, der theoretisch beliebige Informationen enthalten kann, hier aber meist zum Speichern von Photonen verwendet wird.

- **Photonen Basis:** Auch hiermit werden bereits verschossene Photonen ausgetauscht, wobei dies aber hier über das Zwischenspeichern in einer Datei geschieht. Diese Basis wird im Augenblick nur für den Austausch von Photonen mit einem **Inventor LightOp** verwendet, mit dem diese Photonen visualisiert werden können (siehe Bild B.11).

3.1.1.1 Beispiel eines Lighting Networks

Ein mögliches Lighting Network, mit dem nun sowohl direkte Beleuchtung, Kaustiken als auch indirekte diffuse Beleuchtung berechnet werden kann, ist in Abbildung 3.2 aufgeführt.

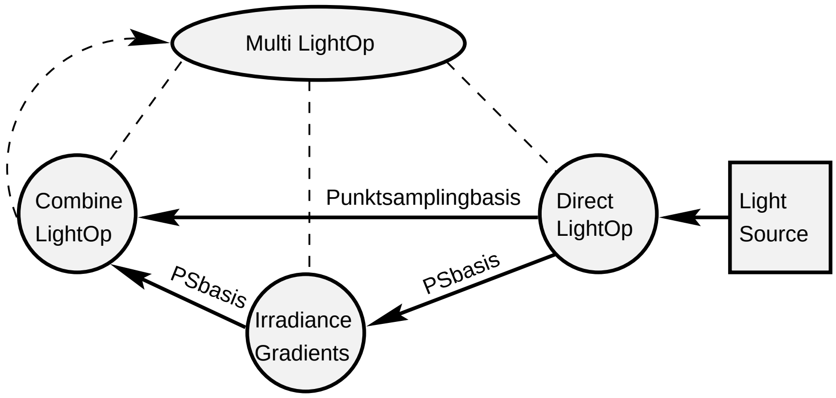

Abbildung 3.2: Ein mögliches Lighting Network.

Über die Tcl/Tk Schnittstelle ist dieses Netzwerk wie folgt definiert:

```
proc gradients {lighting filebase} {
  set dirOp [crt_dir]
  set gradOp [crt_grad]
  set combOp [crt_comb]

  $lighting addLightOp $gradOp
  $lighting addLightOp $dirOp
  $lighting addLightOp $combOp

  $lighting connectLightOp $combOp $gradOp $psbasis
  $lighting connectLightOp $combOp $dirOp $psbasis
  $lighting connectLightOp $gradOp $dirOp $psbasis
  $lighting setMasterLightOp $combOp

  return "[code output $lighting $filebase rt]"
}
```

3.2 Photon Maps

3.2.1 Photon LightOp

Wie alle anderen Algorithmen auch, die in dieser Arbeit implementiert bzw. erweitert werden, soll auch das Photon LightOp mit Volumen zurechtkommen. Die hier implementierte Rekonstruktion der Photonen in einem Volumen verwendet den gleichen Algorithmus wie beim **Caustic LightOp** (siehe Abschnitt 3.2.2) und wird deshalb hier auch nicht mehr näher beschrieben.

Die Implementierung der Berechnung der indirekten diffusen Beleuchtung mit Hilfe von Photon Maps auf Flächen war bereits vorhanden und konnte nach einigen kleinen Änderungen

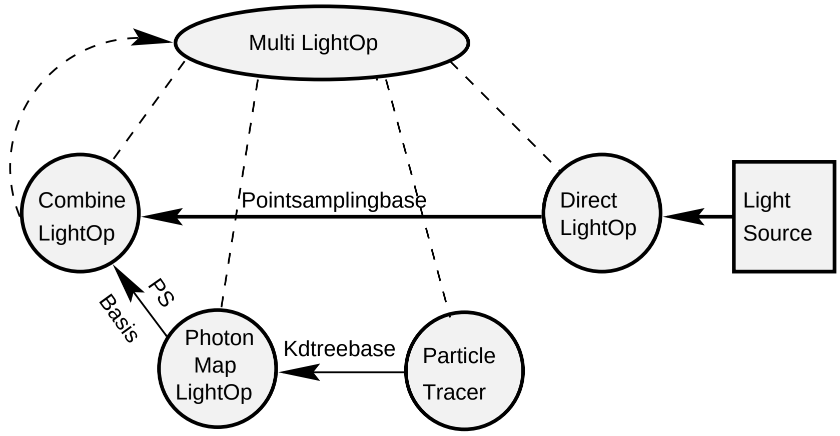

Abbildung 3.3: Das Lighting Network zur Berechnung von indirekter diffuser Beleuchtung durch Photon Maps.

übernommen werden. Nur das Verschicken der Photonen im ersten Schritt des Verfahrens (siehe 2.3.4) und die bisher vorhandene Berechnung der direkten Beleuchtung in diesem LightOp wurden wieder analog zum **Caustic LightOp** in die zuständigen anderen LightOps ausgelagert. Hierzu wurde das in Abbildung 3.3 skizzierte Lighting Network konstruiert, das die einzelnen benötigten LightOps verbindet.

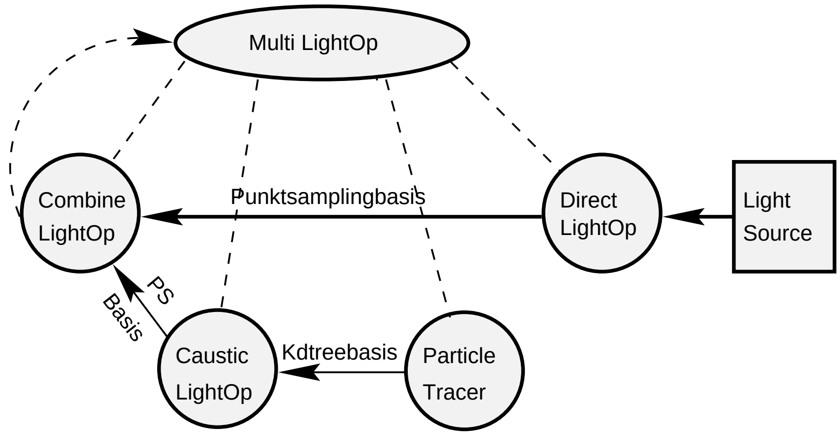

Abbildung 3.4: Das Lighting Network zur Kaustiken Berechnung.

3.2.2 Caustic LightOp

Das **Caustic LightOp** wurde ähnlich wie das **Photon LightOp** auch um die Fähigkeit, Volumen zu berücksichtigen, erweitert. So können nun hiermit sowohl Kaustiken auf Flächen als auch in Volumen berechnet werden. Damit dies möglich ist, muß das in Abbildung 3.4 dargestellte Lighting Network aufgebaut werden, das zusätzlich zum dem **Caustic LightOp** noch ein **Particle LightOp** (siehe Abschnitt 4.5) enthält, welches die benötigten Photonen von den Lichtquellen verschießt.

3.2.2.1 Gewichtung der einzelnen Photonen

Um Artefakte bei der Rekonstruktion zu verringern, wurde eine Gewichtung der einzelnen Photonen eingeführt. Dadurch wird bewirkt, daß die Bedeutung eines einzelnen Photons stärker lokal bleibt, d.h. ein etwaiger Ausreißer (ein Photon mit stark unterschiedlicher Helligkeit von den Benachbarten) dann viel weniger sichtbaren Schaden verursachen kann. Den Exponenten E der Gewichtung kann der Benutzer frei wählen, wobei sich als sinnvolle Werte 1 oder 2 ergeben haben. Setzt man $E = 0$ bedeutet dies, daß jedem Photon die gleiche Bedeutung zukommt. Die Gewichtung W ergibt sich aus

$$\begin{aligned} W_n &= \frac{R - \|P_b - P_p\|^E}{R}, \\ W_t &= \frac{\sum_{n=1}^{N} W_n}{N}, \\ W &= \frac{W_n}{W_t}, \end{aligned}$$

wobei R der Radius des Kreises, der das äußerste Photon gerade noch enthält, P_b die Position des Betrachtungspunktes, P_p die Position des betreffenden Photons und N die Gesamtanzahl der gefundenen Photonen ist.

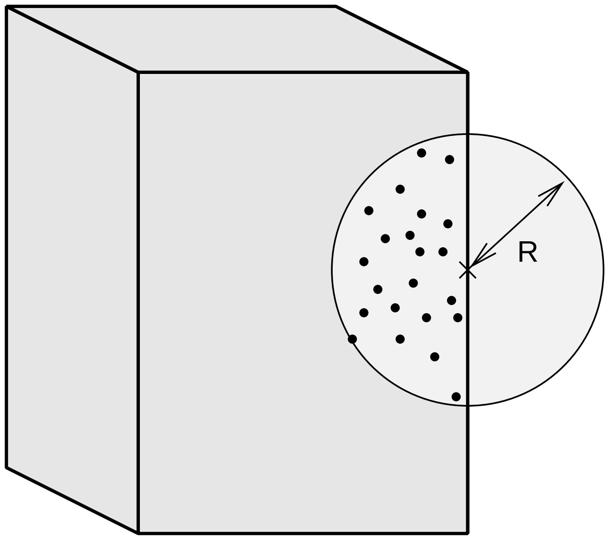

Abbildung 3.5: Photonen an Objektkanten.

3.2.2.2 Aufhellung an den Kanten

Die direkte Rekonstruktion der Kaustiken stellt an Objektkanten ein gewisses Problem dar. Die Helligkeit wird über die Dichte berechnet, aber die Photonen befinden sich an Kanten nur in einer Hälfte des Kreises (siehe Abbildung 3.5), der den Betrachtungspunkt als Mittelpunkt und die Entfernung zum äußersten Photon als Radius hat. Daher reduziert sich die Helligkeit in Kantennähe bis auf die Hälfte, was nicht der gewollten Verteilung entspricht.
Um dies zu verbessern, wurde ein Algorithmus implementiert, der die konvexe Hülle über alle enthaltenen Photonen bildet und diese Fläche mit dem Flächeninhalt des Betrachtungskreises ins Verhältnis setzt. Mit diesem Faktor wird dann die Helligkeit gewichtet.
Dazu wird unter anderem der Graham-Algorithmus [18] zur Bestimmung der m Photonen verwendet, die eine konvexe Hülle um alle n Photonen bilden. Dabei werden zuerst alle n aus der Photon Map ausgelesenen Photonen durch eine Projektion auf die zum Suchpunkt zugehörige Objektebene abgebildet. Der Algorithmus bewältigt dann die Auswahl der m Photonen, die die konvexe Hülle bilden, in einer Zeitkomplexität von $O(n \cdot \log n)$. Die Berechnung des Flächeninhaltes A dieses nun m-seitigen Polygons erfolgt durch folgende Formel:

$$A = \sum_{i=1}^{n} (P_i \times P_{i-1}) \frac{1}{2\vec{n}},$$

wobei $\vec{n}$ die Normale der projizierten Ebene ist, und P_i die Position der einzelnen Photonen. Hieraus läßt sich nun das Verhältnis für die Aufhellung gewinnen, und die Beleuchtung wird auf diese Weise an den Positionen aufgehellt, bei denen die Photonen sich nur in einem Bereich des betrachteten Kreises befinden. Dadurch ergeben sich bei einer Kaustik keine dunklen Ränder in der Nähe von Kanten.

3.2.2.3 Entfernen der Extrema

Es hat sich gezeigt, daß einige 'Ausreiser', d.h. Photonen, die sich von den benachbarten Photonen stark in ihrer Helligkeit unterscheiden, visuelle Artefakte verursachen können. Um diese Photonen zu erkennen, ermittelt man den mittleren Helligkeitswert der in Frage kommenden Photonen und anschließend die am stärksten abweichenden Photonen. Diese werden dann nicht zur Berechnung des neuen Beleuchtungswertes an dieser Stelle verwendet. Hierdurch können Artefakte, die durch das stochastische Sampling (siehe 2.3.4) entstehen, verringert werden.

3.3 Smart Radiosity

Smart Radiosity [28, 29] ist ein Radiosity Verfahren, daß versucht, den Basis Radiosity Ansatz durch die Verwendung von Schranken einzelner Werte zu beschleunigen und verbessern.
Hierdurch kann dieser Radiosity Ansatz nicht nur mit planaren Patches verwendet werden, sondern zusammen mit jeglicher Geometrie, für die sich eine Begrenzungsbox und ein Beschränkungsbereich der Normalen angeben läßt. Deshalb ist dieses Verfahren unter anderem sehr gut für gebogene Objekte geeignet, die nicht mehr, wie sonst bei der Radiosity üblich, in einem Vorverarbeitungsschritt in planare Flächenstücke zerlegt werden müßen.
Durch diese Begrenzungsboxen und dem angegebenen Bewegungsbereich der Normalen, der hier über *Cones of Normals* (siehe auch Abschnitt 4.9) definiert ist, wird eine Fehlerschranke berechnet, die für das Unterteilungsorakel und die Berechnung des Energietransfers verwendet wird. Bei diesen *Cones of Normals* gibt die Achse der Cones die 'Haupt'-Richtung aller Objektnormalen an und der Öffnungswinkel eine Begrenzung aller vorkommenden Objektnormalen.
Mit diesem einfachen Interface zwischen den Objekten kann dieser Algorithmus zusätzlich leicht auf Cluster erweitert werden, was bedeutet, daß eine Grobunterteilung der Szene durch solche Cluster die Berechnung stark beschleunigt.
Aus der Differenz der oberen und unteren Schranke der Radiosity des Empfängers ergibt sich eine Oberschranke des Fehler, der für das Unterteilungsorakel verwendet wird. Diese Radiosity eines Elementes in einer Interaktion hängt von drei Faktoren ab, von der Reflektanz des Empfängers, dem Form Faktor Kernel zwischen den beiden Interaktionsobjekten und der Radiosity des Senders. Die Extremwerte der Radiosity ergeben sich aus den zugehörigen oberen und unteren Schranken für diese drei Werte. Da diese zusätzlichen Informationen über die Schranken in das Unterteilungskriterium einfließen, kann die Präzision des Orakels verbessert werden, jetzt wird nicht mehr nur die Menge des transportierten Lichtes berücksichtigt, sonders auch dieses zusätzlich eingeführte Fehlermaß.

Dieser Smart Radiosity Ansatz, der schon in Vision durch das **SRLightOp** vorhanden war, wurde analog zu allen anderen in dieser Arbeit verwendeten LightOps zusätzlich um die Fähigkeit erweitert, mit Volumen-Objekten umzugehen.
Wie schon in Abschnitt 2.3.3.2 aufgezeigt, mußte hierfür intern die Berechnung von der Größe *Radiosity* auf die Größe *Power* umgestellt werden. Jetzt kann die Behandlung der einzelnen

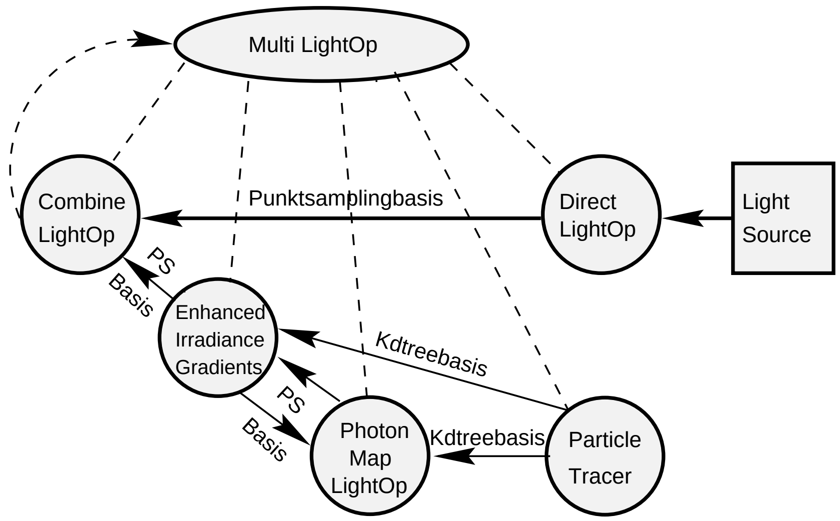

Abbildung 3.6: Das Lighting Network zur Berechnung der indirekten diffusen Beleuchtung durch Enhanced Irradiance Gradients.

Objekttypen weiterhin transparent geschehen, es ist keine getrennte Berechnung der Formfaktoren für Flächen und Volumen notwendig.

3.4 Enhanced Irradiance Gradients

Der Standard Irradiance Gradients Ansatz von Ward [33] wurde erweitert, damit auch die Informationen aus den Photon Maps für das Supersampling und der Bestimmung der bei einem Rekursionsabbruch zurückgelieferten Farbe verwertet werden können. Außerdem kann die indirekte diffuse Beleuchtung mit diesem erweiterten Ansatz auch in Szenen mit Volumen berechnet werden. Hierzu wurde über die Lighting Networks Schnittstelle das **Irradiance Gradients LightOp** mit einem **ParticleTracer LightOp** und einem **PhotonMap LightOp** verbunden. Ein Anfrage nach der indirekten diffusen Beleuchtung bei Volumen wird dann an das **PhotonMap LightOp** weitergereicht und von diesem beantwortet. Das zugehörige Lighting Network ist in Abbildung 3.6 skizziert.

Bei einem Abbruch der Rekursion beim Irradiance Gradients Ansatz wurde bisher immer eine vorher definierte konstante Farbe als Approximation für hier nicht mehr weiter berechnete diffuse Beleuchtung verwendet. Die Erweiterung dieses Ansatzes besteht jetzt darin, daß die Farbe, die bei einem Rekursionsabbruch zurückgeliefert wird, mit der Information aus einer Photon Map sehr viel exakter bestimmt werden kann. Dazu wird diese Farbe aus der Dichte über die n nächsten Photonen bestimmt, die aus einen Kd-Tree gelesen werden. Dieser Kd-Tree wird in einem Vorverarbeitungsschritt vom **ParticleTracer LightOp** berechnet. Mit Hilfe dieser Erweiterung kann der Enhanced Irradiance Gradients Ansatz auch bei einer geringeren Rekursionstiefe der Berechnung schon gute Ergebnisse erzielen.

Ferner erfolgt auch das Supersampling kritischer Bereiche der Halbkugel mit Hilfe von Photon Maps, das heißt die Richtungen, in die Strahlen verschossen werden, werden unter Zuhilfenahme der Photonenverteilung bestimmt. Hierzu wird eine Umgebungskarte durch die Richtungsinformationen der n nächsten Photonen berechnet und in die so ermittelten hellsten Bereiche, die den größten Beitrag zur Irradiance an dieser Stelle liefern, erneut Strahlen verschickt. Das Verfahren zur Erzeugung dieser Umgebungskarte wurde schon in Abschnitt 2.3.4 beschrieben.

Kapitel 4

Implementierung

4.1 Verschiedene Volumentypen

In dem Programmpaket Vision wurden Volumen implementiert, da diese benötigt werden, um die hier verwendeten Beleuchtungsalgorithmen dort zu integrieren.

4.1.1 Die Basisklasse

Es wurde bei der Implementierung der Volumen viel Wert auf eine allgemeine Schnittstellendefinition gelegt, um so eine spätere Erweiterung um weitere Volumentypen zu erleichtern. So sieht die Basisklasse *VisVolume* u.a. folgende Methoden vor:

```
class VisVolume : public VisGeoObject
{
...
public:
  VisVolumeEvent getNextEvent();
  getTransmission(VisPoint P1, VisPoint P2);
  getScattering(VisPoint P);
  getEmission(VisPoint P);
  getBoundScattering(VisPoint Ll, VisPoint Ur);
  getBoundEmission(VisPoint Ll, VisPoint Ur);
  getBoundAbsorption(VisPoint Ll, VisPoint Ur);
  getPhaseFunction(VisPoint P, VisPoint D);
  getIntegrationStepSize(VisPoint P);
  getMinSampleSize();
...
};
```

Mit *getNextEvent()* kann man sich von einem Volumen zu einer gegebenen Position und Richtung die nächste Stelle geben lassen, an der sich entlang dieses Strahles innerhalb des betreffenden Volumens etwas ereignet. Diese Methode wird hauptsächlich von dem **Particle LightOp** verwendet (siehe Abschnitt 4.5). Als Ereignisse *VisVolumeEvent* kommen hierbei in Frage:

1. **ABSORB:** Hier wird das betreffende Photon innerhalb des Volumens an der zusätzlich zurückgelieferten Position P absorbiert.

2. **SCATTER:** Das eintreffende Photon wird an der Position S in Richtung D gestreut. Die Richtung D wird durch Auswerten der Phasenfunktion bestimmt, die auch einzeln für beliebige S und D über die Methode *getPhaseFunction()* angefordert werden kann. Es werden sowohl S als auch D zurückgegeben.

3. **PASS:** Das eintreffende Photon durchdringt das Volumen ungehindert, die Austrittsposition P aus dem Volumen wird zurückgeliefert.

Mit *getTransmission()* wird der Transmissionskoeffizient zwischen den beiden Punkten *P1* und *P2* bestimmt.
Die Methoden *getScattering()* und *getEmission()* liefern nun jeweils den Streuungs- bzw. Emissionswert, der sich an der Position P im Volumen ergibt. Ist der Wertebereich wichtig, in dem sich zum Beispiel die Emissionswerte in einem bestimmten Bereich des Volumens bewegen, was u.a. für die **Smart Radiosity** (siehe Abschnitt 4.8) notwendig ist, dann können die Methoden *getBound{Scattering|Emission|Absorption}()* aufgerufen werden. Die Punkte Ll und Ur definieren jeweils die linke untere bzw. obere rechte Ecke des zu betrachtenden Bereiches. Diese Methoden berechnen den Mittelwert der in diesem Bereich vorkommenden Werte wie auch ein Epsilon für die maximale Abweichung.
Die zwei letzten Methoden liefern Hinweise über die Beschaffenheit des Volumens, die für die verschiedenen Traversionsalgorithmen von Nutzen sind. *getIntegrationStepSize()* liefert einen Hinweis auf eine sinnvolle Schrittlänge, mit der das Volumen entlang eines Strahles traversiert werden kann, ohne zu viele Details zu mißachten. Entsprechend liefert *getMinSampleSize()* einen Wert für eine sinnvolle Seitenlänge einer minimalen Unterteilungseinheit, wie sie z.B. für **Smart Radiosity** gebraucht wird (siehe 4.8).

4.1.2 Parametrische Volumen

Bei diesem Volumentyp werden die Werte innerhalb des Volumens über eine dreidimensionale Funktion berechnet.

```
class VisParametricVolume : public VisVolume
{
...
public:
  getTransmission(VisPoint P1, VisPoint P2);

private:
  VisDouble getValue(const VisPoint P);
...
};
```

Die Methode *getValue()* liefert hier für jeden Punkt P einen zugehörigen Skalar zurück, der dann durch die Transferfunktionen (siehe 4.1.4) auf einen Ausgabewert für Emission, Absorption und Streuung abgebildet wird.
Die Transmissionsberechnung für zwei Punkte *P1* und *P2* durch *getTransmission()* erfolgt hier durch Strahltraversierung durch das Volumen. Einen Wert für eine geeignete Schrittweite sollte abhängig von der jeweiligen parametrischen Funktion gewählt werden.

```
getTransmission() {
  ...
  fertig = false;
  Richtung = (P2 - P1).Normalisiere();
  Position = P1;
  do {
    totaleAbsorption = getValue(Position);
    totaleAbsorption = TransferFunktionAbsorption(totaleAbsorption) +
                       TransferFunktionStreuung(totaleAbsorption);

    Position = P1;
    P1 += Richtung * Schrittweite;

    for (i = 0; i < 3; i++) {
      if (Richtung[i] > 0) {
        if (Position[i] > P2[i]) {
          Fertig = true;;
          P1 = P2;
        }
      } else {
        if (Position[i] < P2[i]) {
          Fertig = true;
          P1 = P2;
    } } }

    Transmission *= exp(-(P1 - Position).Länge() * totaleAbsorption);

  } while(Fertig == false);
  ...
}
```

4.1.3 Uniforme Volumen

Das Uniforme Volumen wird über mehrere uniforme Unterzellen definiert, daher auch der Name.

```
class VisUniformVolume : public VisVolume
{
...
public:
  getTransmission(VisPoint P1, VisPoint P2);

private:
  bool openHeader(char *HEADER_NAME, char *DATA_NAME);

  bool openData(char *NAME);

  VisDouble triLinearFilter(VisPoint P);
...
};
```

Die Datenwerte dieser blockförmigen Unterzellen werden durch die Methode *openData()* aus einer Datei gelesen, wobei die Auflösung in alle drei Dimensionen beliebig gewählt werden kann. Die Größe der Auflösung und andere wichtige Werte über diese Datendatei werden durch eine separate Steuerungdatei definiert, welche mit *openHeader()* eingelesen wird. Dieses Datenformat wurde einem umfangreichen Paket zur Visualisierung von medizinischen Datensätzen entnommen, wobei hier aber nur einige wenige der möglichen Konfigurationswerte verwertet werden:

```
ObjectFileName: NAME DER WERTEDATEI
TaggedFileName: ---
Resolution:     X Y Z
SliceThickness: dX dY dZ
Format:         UCHAR | USHORT
NbrTags:        N (nicht unterstützt)
ObjectType:     TEXTURE_VOLUME_OBJECT
ObjectModel:    RGBA
GridType:       EQUIDISTANT
```

Hierdurch werden alle wichtigen Parameter der Daten definiert. Das Feld **Resolution** gibt die Auflösung in alle drei Dimensionen und **SliceThickness** die Größe einer einzelnen Zelle an. Mit **Format** wird die Bitauflösung der enthaltenen Werte festgelegt, die anderen Werte sind für dieses Modul nicht variierbar.

Die so eingelesenen Daten bilden jeweils die acht Ecken der hiermit definierten Unterzellen. Für die Interpoaltion eines Ausgabewertes für eine beliebige Position P im Volumen wird aus den zu P jeweils nähesten acht Eckwerten durch trilineares Filtern (*triLinearFilter()*) der Wert an dieser Stelle P berechnet.

Die Ermittelung eines Transmissionswertes zwischen zwei Punkten *P1* und *P2* durch *getTransmission()* erfolgt anders als bei den parametrischen Volumen, da man hier schon genaue Informationen über die Verteilung der Werte innerhalb des Volumens hat, und diese Zusatzinformationen für eine höhere Genauigkeit der Berechnung nutzen möchte. Die Position auf dem zu traversierenden Strahl wird in jedem Schritt genau bis zum Mittelpunkt der nächsten Unterzelle verschoben und dann dort ein neuer Wert berechnet. Die so aus einer Datei gelesenen Werte können z.B. Werte eines medizinischen Datensätzes sein, wie z.B. ein Kopf in Bild B.1.

```
getTransmission() {
  ...
  fertig = false;
  Richtung = (P2 - P1).Normalisiere();
  do {
    Wert = triLinearFilter(Position1);
    Wert = TransferFunktionAbsorption(Wert) +
           TransferFuntkionStreuung(Wert);

    Max = I = 0;
    // Suche nach der Hauptbewegungsrichtung
    for (i = 0; i < 3; i++) {
      if (Richtung[i] > 0) {
        d[i] = (floor(P1[i]) + 1.0) - P1[i];
      } else {
        d[i] = (ceil(P1[i] - 1.0)) - P1[i];
      }
      if (Max < Richtung[i] / d[i]) {
        Max = Richtung[i] / d[i];
        I = i;
      }
    }

    Position = P1;
    for (i = 0; i < 3; i++) {
      // Hauptbewegungsrichtung
      if (I == i) {
        P1[i] += d[i];
      } else { // Nebenrichtungen
        P1[i] += d[I] * Richtung[i] / Richtung[I];
      }

    }
    if (P1 > P2) {
      P1 = P2;
      Fertig = true;
    }

    Transmission *= exp(-(P1 - Position).Länge() * Wert));
    }

    if (outPos == true) {
      break;
    }
  } while (Fertig == false);
  ...
}
```

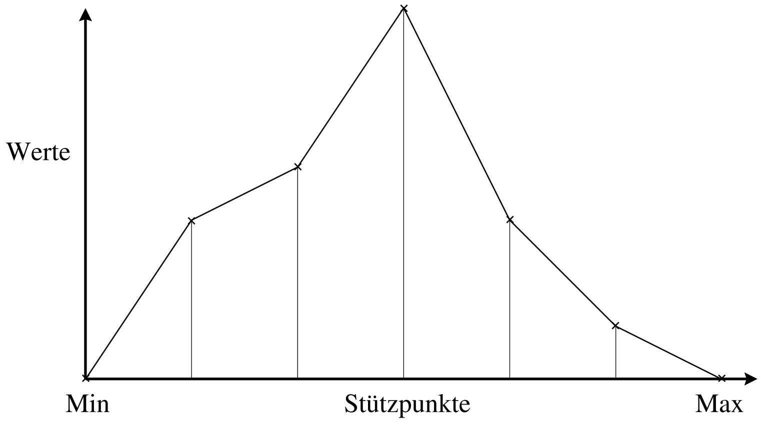

Abbildung 4.1: Eine lineare Transferfunktion, durch Stützpunkte definiert.

4.1.4 Transferfunktionen

Transferfunktionen werden bei den hier implementierten Volumen benötigt, um die durch eine Funktion bestimmten beziehungsweise aus einer Datei gelesenen Skalare auf die einzelnen Werte für Emission, Absorption und Streuung abzubilden. Hierbei können die drei Farbkanäle einzeln angesteuert werden.

```
class VisTransferFunction
{
...
public:
  VisTransferFunction(int NUMMER, double *ROT, double *GRÜN,
                      double *BLAU);

  setMinMax(double MIN, double MAX);

  getValues(double IN_WERT, double ROT, double GRÜN, double BLAU) = 0;
...
};
```

Der Transferfunktionen werden bei der Instantiierung während des Parsens der RIB-Datei [31] jeweils *NUMMER* Stützpunkte für *ROT*, *GRÜN* und *BLAU* mitgegeben. Die minimal und maximal vorkommenden Werte innerhalb eines Volumen werden mit *setMinMax()* von diesem gesetzt.

Ferner wird über die RIB-Schnittstelle die Methode der Interpolation zwischen den einzelnen Stützstellen gewählt, wobei für diese Arbeit nur eine *lineare* Interpolation durch die Klasse **VisLinearTransferFunction** implementiert wurde.

```
class VisLinearTransferFunction : public VisTransferFunction
{
...
public:
  getValues(double IN_WERT, double ROT, double GRÜN, double BLAU);
...
};
```

In dieser speziellen Transferfunktion ist jetzt die Methode *getValues()* implementiert, die hier die Abbildung der Eingabewerte auf die Ausgabewerte berechnet. Eine Definition einer linearen Transferfunktion durch eine RIB-Datei sieht beispielsweise wie folgt aus:

```
Declare "RedAbsorption" "float[5]"
Declare "GreenAbsorption" "float[5]"
Declare "BlueAbsorption" "float[5]"
Declare "RedScattering" "float[3]"
Declare "GreenScattering" "float[3]"
Declare "BlueScattering" "float[3]"
Declare "RedEmission" "float[2]"
Declare "GreenEmission" "float[2]"
Declare "BlueEmission" "float[2]"

Geometry UniformVolume" "VolFile" "volume.dat"
  "AbsorptionPoints" 5
  "RedAbsorption" [0 0.2 0.4 0.5 1]
  "GreenAbsorption" [0 0.3 0.4 0.6 1]
  "BlueAbsorption" [0.2 0.4 0.6 0.8]
  "ScatteringPoints" 3
  "RedScattering"   [0 0.3 1]
  "GreenScattering" [0 0.3 1]
  "BlueScattering"  [0 0.3 1]
  "EmissionPoints" 2
  "RedEmission"   [0 0.5]
  "GreenEmission" [0 0.5]
  "BlueEmission"  [0 0.5]
```

4.2 Direct LightOp

Das schon bestehende **Direct LightOp** wurde um die Fähigkeit erweitert, die direkte Beleuchtung auch in Szenen zu berechnen, die Volumen enthalten. So können jetzt beliebige Szenen, die Volumen enthalten, mit diesem LightOp berechnet werden, wie z.B. das mit einem Nebel gefüllte Zimmer in Bild B.2. Da während der Belechtungsberechung in einem Volumen im Gegensatz zur Berechnung bei Flächen auch die Informationen über den Transparenzwert und dem Austrittspunkt aus dem Volumen als Nebenprodukt abfallen, wurde die Schnittstelle von *VisLighting* erweitert.

```
class VisLighting
{
...
public:
  virtual void getLightInfo(VisLight LICHT, VisColor TRANSMISSION,
                VisTransRay *STRAHL, VisPoint OUT_POSITION);

  virtual VisIllumination* getIllumination();
...
};
```

Der Ray Tracer von Vision wurde so erweitert, daß er die Methode *getIllumination()* aufruft, wenn es sich um eine Fläche handelt und *getLightInfo()*, wenn ein Volumen getroffen wurde. Auf diese Weise können durch diese zusätzliche Methode die Informationen, die durch das Traversieren des Volumens sowieso berechnet werden müssen, gleich mit zurückgeliefert werden.

Die Integration des Volumens selbst erfolgt bei der direkten Beleuchtung in zwei Schritten von vorne nach hinten. Dies hat den Vorteil, daß diese vorzeitig abgebrochen werden kann, wenn die Transmission einen Wert unterhalb einer Schranke (hier $0,05$) erreicht.

- Zuerst werden die Werte des Volumen iterativ in einer vorgegebenen Schrittweite entlang des zu betrachtenden Strahles aufintegriert. Ein sinnvoller Wert für diese Schrittweite wird mit *getIntegrationStepSize()* vom zugehörigen Volumen erfragt.
- Wenn die Differenz der Helligkeit an zwei benachbarten Stützstellen im Laufe der Integration über einer bestimmten Schranke liegt, wird dieses Segment in *testDifference()* rekursiv unterteilt und erneut getestet [14]. Hierdurch wird in Bereichen, in denen sich große Unterschiede der Helligkeit ergeben, die Unterteilung verfeinert, um keine Details zu übersehen.

```
getLightInfo(LICHT, TRANSMISSION, *STRAHL, OUT_POSITION) {
  ...
  Schrittweite = Strecke_durch_Volumen /
                 Volumen->getIntegrationStepSize();
  t = 0;
  fertig = false;
  do {
    Position1 = Startpunkt + Richtung * t;
    Licht1 = calcLight(Position1);
    Schrittweite1 = jitter(Schrittweite);
    t += Schrittweite1 / TRANSMISSION;
    if (t > Strecke_durch_Volumen) {
      t = Strecke_durch_Volumen;
      Fertig = true;
    }
    Position2 = Startpunkt + Richtung * t;
    Licht2 = calcLight(Position2);
    LICHT += testDifference(TRANSMISSION,Position1,Licht1,
                            Position2,Licht2);
    if (TRANSMISSION < 0.05) {
      Fertig = true;
    }
  } while (Fertig != true);
  OUT_POSITION = Startpunkt + Richtung * Strecke_durch_Volumen;
  ...
}
```

```
VisLight testDifference(TRANSMISSION,Position1,Licht1,
                        Position2,Licht2) {
  ...
  if (Licht1 - Licht2 > Schranke) {
    Position3 = (Position1 + Position2) / 2;
    Licht3 = calcLight(Position3);
    Licht  = testDifference(TRANSMISSION,Position1,Licht1,
                                         Position3,Licht3);
    Licht += testDifference(TRANSMISSION,Position3,Licht3,
                                         Position2,Licht2);
  }
  ...
}
```

4.3 Kd-Tree

Um später die n nähesten verschossenen Photonen bezüglich einer beliebigen Position bei dem Photon Map Verfahren effizient suchen zu können, werden die Photonen in einen Kd-Tree abgelegt. Da der Baum nach dem Einfügen aller Photonen ausbalanciert wird, kann man als Komplexität der Suche nach M Photonen in einem Baum mit N Photonen $O(M \cdot log_2(N))$ ansetzen. Nach außen sind von der Klasse *VisKdTree* im wesentlichen drei Methoden sichtbar.

```
class VisKdTree
{
...
public:
  insertNode();
  computeTree();
  SearchResults* computeNearestPts(int MaxPhotonen, double MaxDistance,
                                   VisPoint SuchPunkt,
                                   VisPoint SuchNormale);
...
};
```

Es können mit *insertNode()* neue Photonen eingefügt und mit *computeTree()* der fertig gefüllte Baum ausbalanciert werden. Wenn nun Photonen in diesem Baum gesucht werden sollen, steht die Methode *computeNearestPts()* zur Verfügung, die dann folgende Tests auf den Kd-Tree durchführt:

- Die Maximalanzahl der gefundenen Photonen darf **MaxPhotonen** nicht übertreffen.
- Das am weitesten vom Betrachtungspunkt **SuchPunkt** entfernte Photon darf eine Entfernung von maximal **MaxDistance** haben.
- Das Skalarprodukt der Normale der zu diesem Photon zugehörigen Fläche mit der Normale **SuchNormale** des Objektes an der Suchposition muß größer einer Schranke (hier $0,3$) sein. Als Sonderfall für Volumen kann als Normale auch $(0,0,0)$ übergeben werden, was dann bedeutet, daß kein Normalenvergleich stattfinden soll.

Die Informationen eines jeden eingefügten Photons werden in der Klasse *VisKdTreeNode* gespeichert. Da die Normale nur für einen Ausschlußvergleich bei der Suche im Kd-Tree (siehe oben) benötigt wird, kann diese in drei **char**-Variablen gespeichert werden. Auch die Einfallsrichtung des Photons wird auf drei Bytes reduziert, da dieser Wert ausschließlich für die Berechnung der Importance Map (siehe 2.3.4.4) verwendet wird, und hier die Umgebung sowieso in n diskrete Bereiche unterteilt wird, die verringerte Genauigkeit also kein Problem darstellt. Durch diese Vereinfachungen wird für jedes Photon nur 32 Bytes an Platz benötigt.

```
class VisKdTreeNode
{
...
private:
  float Farbe[3];
  float Position[3];
  signed char EinfallendeRichtung[3];
  signed char Normale[3];
  char Kd-TreeKey;
  char Type;
...
};
```

4.4 Lighting Networks

Die Implementierung der Schnittstelle zwischen zwei LightOps erfolgt über Beleuchtungsbasen, die durch die Basisklasse *VisIllumBasis* realisiert sind. Hier gibt es für jeden möglichen Typ der Verbindung zweier LightOps eine abgeleitete Klasse dieser. Wenn beide zu verbindenden LightOps diese Verbindungsbasis beherrschen, ist ein Datenaustausch hierüber möglich. An jedem LightOp kann durch die Methoden *getInputBasis() und getOutputBasis()* erfragt werden, welche verschiedenen Austauschmöglichkeiten für Ein- und Ausgabe es beherrscht.

```
VisIllumBasis *getInputBasis(VisCardinal i)
{
  VisPSIllumBasis ps;
  switch(i)
  {
    case 0:
      return &ps;
    ...
    default:
      return NULL;
  }
}
```

```
VisIllumBasis *getOutputBasis(VisCardinal i)
{
  VisPSIllumBasis ps;
  switch(i)
  {
    case 0:
      return &ps;
    ...
    default:
      return NULL;
  }
}
```

Die folgenden Basen werden in dieser Arbeit benötigt (siehe auch Abschnitt 3.1.1):

- **VisPSIllumBasis:** Dies ist ein Punktsampling Interface.
- **VisPSDiffuseIllumBasis:** Dies steht für ein diffuses Punktsampling Interface.
- **VisKdTreeIllumBasis:** Mit Hilfe dieser Basis werden Daten über einen Kd-Tree ausgetauscht.
- **VisPhotonsIllumBasis:** Photonen werden hier über ein Zwischenspeichern in eine Datei ausgetauscht, dies wird vor allem für die Inventorausgabe über ein **Inventor LightOp** benötigt (siehe Bild B.11).

4.5 Particle Tracer

Der Particle Tracer verschickt Photon von den Lichtquellen aus in die Szene. Bei diesem Path Tracing werden die Photonen auf den getroffenen Oberflächen abgelegt und dann entsprechend der zugehörigen BRDF mit Hilfe von Russian Roulette reflektiert. Hierdurch wird die Lichtverteilung in der Szene angenähert.
Um eine saubere Gliederung der einzelnen LightOp zu erreichen, und doppelten Programmcode zu vermeiden, wurde die Implementierung des Verschickens von Photonen aus dem **Photon LightOp** und dem **Caustic LightOp** ausgegliedert, und vollständig in den bereits bestehenden **Particle LightOp** integriert.

Es können über diesen Particle Tracer auch Photonen in Szenen mit Volumen verschossen werden Wenn ein Photon mit einem Volumen interagiert, wird an diesem über die Methode *getNextEvent()* die Art der Interaktion und die zugehörige Position erfragt (siehe auch Abschnitt 4.1.1). Ferner wird geprüft, ob sich innerhalb des Volumens nicht noch andere Objekte befinden, die räumlich gesehen vor dem Volumen mit diesem Photon interagieren. Volumen und Flächen können mit Hilfe dieses Ansatzes beliebig verschachtelt werden. Wenn nun eine entgültige Interaktionsposition bestimmt wurde, wird das Photon anhand dieser

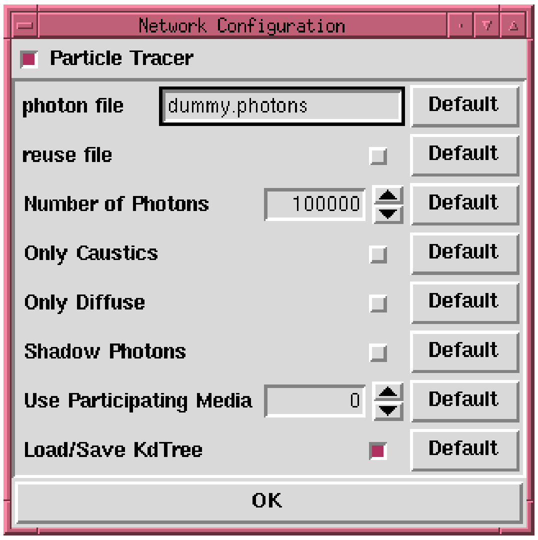

Abbildung 4.2: Werte, die beim Particle Tracer über die Tcl/Tk - Schnittstelle verändert werden können.

Information entsprechend der Interaktionsart an dieser Position abgelegt und/oder von hier weiterverschossen.

Da der **Particle Tracer** für die vielen Algorithmen, für die er benötigt wird, immer etwas unterschiedliche Photon Maps erstellen muß, gibt es einige Parameter, mit denen das Verhalten dieses LightOps gesteuert werden kann (siehe Abbildung 4.2).

- **photon file:** Name der Datei, über die eine Inventordarstellung der Photonen erfolgen kann.
- **Number of Photons:** Anzahl der primären Photonen, die verschossen werden.
- **Only Caustics:** Wenn dies aktiviert ist, werden Photonen für eine Photon Map berechnet, die von dem **Caustic LightOp** benötigt wird. Dafür werde nur Photonen in einem Kd-Tree abgelegt, wenn diese über n spekulare Flächen auf eine diffuse treffen.
- **Only Diffuse:** Hiermit kann man steuern, ob nur der diffuse Lichtanteil von den Shadern angefordert werden soll. Dies wird zum Beispiel benötigt, wenn in einem Lighting Network sowohl ein **Caustic LightOp** als auch ein **Photon LightOp** verwendet wird, damit hier keine Lichtpfade doppelt berechnet werden.
- **Shadow Photons:** Es könnten hier auch sogenannte Schatten Photonen berechnet werden, diese werden aber im Augenblick von keinem LightOp unterstützt.
- **Use Participating Media:** Diese Variable kann folgende Werte annehmen:
  - **0:** Es wird nur ein Kd-Tree für Flächen berechnet.

- **1:** Es werden zwei Kd-Trees berechnet, einer für Flächen und einer für Volumen.
- **2:** Es wird nur ein Kd-Tree für Volumen erzeugt.
- **3:** Es werden zwei Kd-Trees berechnet, einer für Flächen und einer für Volumen, die aber nur Photonen enthalten, die mindestens einmal reflektiert wurden.
- **4:** Es wird nur ein Kd-Tree für Volumen erzeugt, der auch nur mindestens einmal reflektierte Photonen enthält.

- **Load/Save KdTree:** Wenn dieser Schalter aktiviert ist, werden die verschickten Photonen auch in einer Datei gesichert und können dann später bei der gleichen Szene wiederverwertet werden, ohne daß der **Particle Tracer** diese erneut verschiessen muß.

4.6 Photon Maps

4.6.1 Photon LightOp

Das **Photon LightOp** wurde im Zuge der Neugliederung der Algorithmen stark verändert. Der Teil des Algorithmus, der im ersten Schritt die Photonen von den Lichtquellen weg in die Szene verschießt, wurde in das **Particle LightOp** ausgegliedert. Da nun jedes an dem Hybridverfahren beteiligte Verfahren auch Szenen mit Volumen berechnen können muß, wurde auch hier, wie in allen anderen beteiligten LightOps, die Methode *getLightInfo()* implementiert.

```
class VisPhotonLightOp : public VisDirectLightOp
{
...
public:
  getLightInfo();
...
};
```

Im Gegensatz zum Photon LightOp Algorithmus für Flächen, der in Abschnitt 2.3.4 erklärt wurde, erfolgt bei Volumen eine direkte Rekonstruktion über die aus dem Kd-Tree gefundenen Photonen. Dies ist dasselbe Verfahren, das auch beim **Caustic LightOp** Anwendung findet und in 4.6.2 beschrieben wird.
Die Implementierung des Algorithmus für Flächen war schon weitgehend vorhanden und kann mit allen Details in [21] nachgelesen werden.

Als konfigurierbare Parameter sind in diesem Modul vorhanden (siehe Abbildung 4.3):

- **points in sphere:** Dies ist die maximale Anzahl der Photonen, die für die Erzeugung der Umgebungsmap verwendet werden.

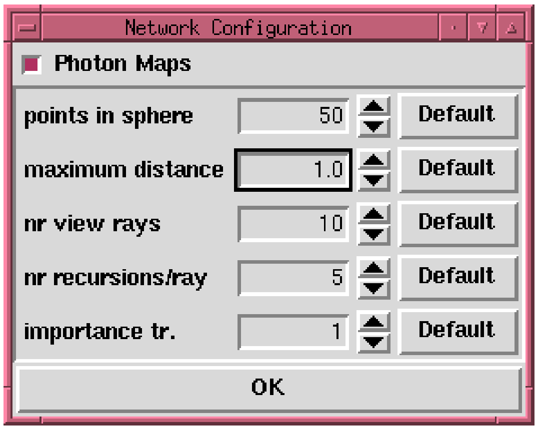

Abbildung 4.3: Werte, die beim Photon LightOp über die Tcl/Tk - Schnittstelle verändert werden können.

- **maximum distance:** Hiermit kann die maximale Entfernung des vom Betrachtungspunkt am weitesten entfernten noch benutzten Photons begrenzt werden.
- **nr view rays:** Die Anzahl der Strahlen, die in jeder Rekursion verschossen werden sollen, wird hiermit angegeben.
- **nr recursions/ray:** Diese Zahl begrenzt die Tiefe, bis zu der maximal rekursiert wird.
- **importance tr.:** Durch diesen Schalter kann die Verwendung des Importance Tracing, das heißt die Erzeugung einer Umgebungskarte und deren Auswertung, ein- und ausgeschalten werden.

4.6.2 Caustic LightOp

Mit dem **Caustic LightOp** können jetzt Kaustiken sowohl auf Flächen als auch in Volumen berechnet werden. Für Volumen steht hier wieder die Methode *getLightInfo()* zur Verfügung.

```
class VisCausticLightOp : public VisLightOp
{
...
public:
  getLightInfo();

private:
  correctLight();
...
};
```

Bei der Methode *getLightInfo()* erfolgt, genauso wie beim in Abschnitt 4.2 beschriebenen **Direct LightOp**, die Integration des Volumens von vorne nach hinten. Es werden die

n nächsten Photonen an einer Position P_0 entlang des zu betrachtenden Strahles aus dem Kd-Tree ausgelesen und aus diesen die Dichte gebildet. Der Radius der Kugel, die das am weitesten entfernteste Photon gerade noch enthält wird als nächste Schrittweite genommen. An dieser Stelle P_1 wird dann erneut die Dichte bestimmt. Die diskrete Aufiteration der Gesamthelligkeit ergibt sich für jeden Teilschritt jeweils aus:

$$L = L + ((L_0 + L_1 \cdot \tau(P_0, P_1)) \cdot \frac{|P_0 - P_1|}{2} \cdot \tau(P_{start}, P_0))$$

```
VisColor berechne_Dichte(Photonen) {
  Radius= Photonen->getRadius();
  Area= Radius * Radius * Radius * M_PI * 4/3;
  for(i = Photonen->photonCount()-1; i >= 0; i--) {
    ...
    Photonen->getPhoton(i, ..., Licht);
      Color += Licht;
  }
  Color /= Area;
  return Color;
}
```

```
getLightInfo() {
  ...
  Photonen = Photon_Map->computeNearestPts(..., StartPunkt, Normale);
  dt = Photonen->getRadius();
  Licht0 = berechne_Dichte(Photonen);
  Laenge = (EndPunkt - StartPunkt).Laenge();
  P0 = sPoint;
  do {
    if (t + dt <= Laenge) {
      // add offset
      t += dt;
    } else {
      // we are through
      t = Laenge;
      fertig = true;
    }
    P1 = StartPunkt + Strahl_Richtung * t;
    Photonen = Photon_Map->computeNearestPts(..., P1, Normale);
    dt = Photonen->getRadius();
    Licht1 = berechne_Dichte(Photonen);
    Transmission = Volumen->getTransmission(P0,P1);
    gesamt_Licht += (Licht0 + Licht1 * Transmission)
      * (P0 - P1).Laenge() / 2.0 * gesamt_Transmission;
    gesamt_Transmission *= Transmission;
    Licht0 = Licht1;
    P0 = P1;
  } while(fertig != true);
  ...
}
```

Als konfigurierbare Parameter sind bei dieser Klasse vorhanden (siehe Abbildung 4.4):

- **points in sphere:** Dies ist die maximale Anzahl der Photonen, die für die Erzeugung der Umgebungsmap verwendet werden.

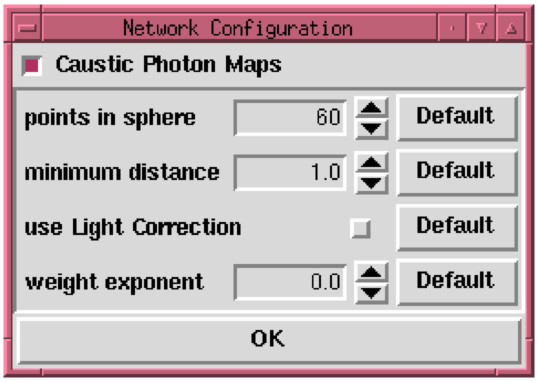

Abbildung 4.4: Werte, die beim Caustic LightOp über die Tcl/Tk - Schnittstelle verändert werden können.

- **maximum distance:** Mit diesem Parameter kann die maximale Entfernung des vom Betrachtungspunkt am weitesten entfernten Photons begrenzt werden.
- **use Light Correction:** Mit diesem Schalter kann man die Kantenkorrektur (siehe Abschnitt 4.6.2.3) ein- und ausschalten.
- **weight exponent:** Über diesen Parameter kann der Benutzer einen Exponent für die erweiterte Rekonstruktionsmethode (siehe 4.6.2.1) setzen.

4.6.2.1 Verschiedene Rekonstruktionsmethoden

Um eine gewichtete Rekonstruktion durchzuführen, müssen die betreffenden Photonen zweimal ausgewertet werden, da im ersten Schritt die Summe der gesamten Gewichte berechnet und im zweiten Schritt dann jede Einzelsumme mit dieser Gesamtsumme normiert wird. Durch diesen Ansatz ist an Schluß die Gesamthelligkeit wieder korrekt.

```
radius = Entfernung_des_äussersten_Photons;
exponent = BENUTZEREINGABE;
foreach (Photon) {
  Gewicht = (Radius - Photon.Position()).Länge() / Radius;
  Gewicht = pow(Gewicht, Exponent);
  Gesamt_Gewicht += weight;
}
Gesamt_Gewicht /= Anzahl_der_Photonen;
foreach (Photon) {
 Gewicht = (Radius - photon.Position()).Länge() / Radius;
 Gewicht = pow(Gewicht, Exponent);
 Photon.Licht *= Gewicht / Gesamt_Gewicht;
}
```

Diese Gewichtung erfolgt aber nur bei Flächen. Bei Volumen ist die Korrektur nicht notwendig, da dort die Integration über die Tiefe diese Artefakte verhindert.

4.6.2.2 Entfernen der Maxima

Wenn die Maxima der Photonen bei der Rekonstruktion nicht berücksichtigt werden sollen, wird ein doppelter Durchlauf wie bei der Gewichtung der Photonen benötigt. Zur Beschleunigung der Rekonstruktion kann man den ersten Schritt dieses Verfahrens aber sehr gut mit dem ersten Schritt des Gewichtungsverfahrens verbinden. Auf diese Weise sind dann insgesamt nur zwei Phasen notwendig.

```
foreach(Photon) {
  finde_N_Maxima(Photon);
}
foreach(Photon) {
  if (istMaximum(Photon) == true) {
    continue;
  } else {
  ...
  }
}
```

4.6.2.3 Aufhellung an den Kanten

Um die Helligkeit an den Kanten verstärken zu können, müssen solche kritischen Fälle erst einmal erkannt werden. Da sich die Photonen an den Kanten nur in einer Hälfte des zu betrachteten Kreises befinden, ist die konvexe Hülle, die diese bilden, entsprechend klein. Als erstes werden alle Punkte in die Objektebene projiziert (*Projiziere_Punkte_auf_Objekt_Ebene()*)und dann mit Hilfe des Graham-Algorithmus [18] (*Graham()*) die n_c Photonen ermittelt, die die konvexe Hülle bilden. Mit *Area_Polygon()* wird nun der Flächeninhalt des durch diese n_c Photonen gebildeten Polygons berechnet und die Helligkeit an der Betrachtungsstelle mit dem Verhältnis des Kreisinhaltes zu diesem Polygoninhalt gewichtet.

```
Punkte = Projiziere_Punkte_auf_Objekt_Ebene(Punkte);
Punkte = Graham(Punkte);
Inhalt_Polygon = Area_Polygon(Punkte);
Inhalt_Kreis = Radius * Radius * PI;
Gewicht = Inhalt_Kreis / Inhalt_Polygon;
Licht *= Gewicht;
```

4.7 Enhanced Irradiance Gradients

Damit der **Enhanced Irradiance Gradients** Algorithmus einerseits für Volumen verwendet werden kann und andererseits die zusätzlichen Informationen der Photon Maps verarbeiten kann, mußten auch hier einige Änderungen vorgenommen werden.

```
class VisIRPMLightOp : public VisIRLightOp {
...
public:
  getLightInfo();

  VisColor ambColor();
...
};
```

```
class VisIRPMSampler : public VisIRHemisphereRADSampler
{
...
public:
  trySuperSampling();
...
};
```

Die Methode *ambColor()* des Basis **Irradiance Gradients** Ansatzes wurde überladen, um hier die Berechnung der Dichte über die Photon Maps vornehmen zu können. Der Algorithmus ermittelt die n nähesten Photonen bezüglich der Position P und der Normalen N aus einer zugehörigen Photon Map, und berechnet über diese dann die Dichte an dieser Position (siehe auch direkte Rekonstruktion, Abschnitt 4.6.2). Dieser Wert wird jetzt als ambienter Helligkeitswert an dieser Stelle zurückgeliefert.

```
VisColor ambColor(...) {
  Photonen = Photon_Map->computeNearestPts(n, ..., P,N);

  Radius= Photonen->getRadius();
  Area= Radius * Radius * M_PI;

  for(i = Photonen->photonCount()-1; i >= 0; i--) {
    ...

    Photonen->getPhoton(i, ..., Licht);
      Color += Licht;
  }
  Color /= Area;

  return Color;
}
```

In der Klasse *VisIRPMSampler* wurde die Supersampling Routine überladen. Sie bestimmt die die Richtungen der erneut zu schickenden Strahlen über eine Umgebungskarte, die aus Informationen aus einer Photon Map erzeugt wurde. Da auch dieser Algorithmus schon beim **Photon LightOp** erklärt wurde, wird an dieser Stelle nur auf Abschnitt 4.6.1 verwiesen.

Damit **Enhanced Irradiance Gradients** mit Volumen zurechtkommt, wurde auch in dieser Klasse die Methode *getLightInfo()* implementiert. Diese macht nichts anderes, als die Anfrage direkt an das angeschlossene **Photon LightOp** weiterzuleiten.
Für die genauere Beschreibung der Parameter aus Abbildung 4.5 wird hier auf [19] verwiesen. Hier seien nur kurz die Parameter **Recursion**, mit welchem man die maximal zu verwendente Rekursionstiefe begrenzen kann, **Division**, mit dem man die anfängliche Anzahl der Strahlen über den Halbraum festlegen kann, und **Number of SSamples**, womit man die Anzahl der Supersamples pro Irradiancewert-Berechung, die jetzt über den neuen Algorithmus erfolgt, festlegen kann, erwähnt.

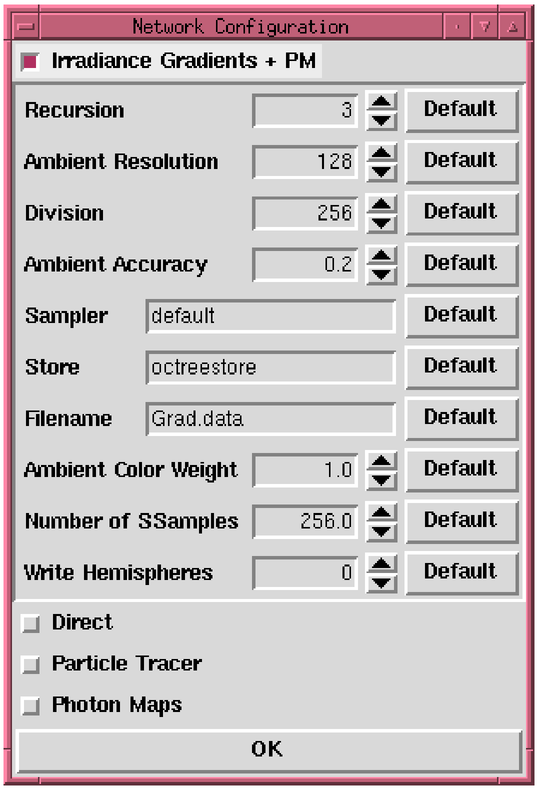

Abbildung 4.5: Werte, die beim Enhanced Irradiance Gradients LightOp über die Tcl/Tk - Schnittstelle verändert werden können.

4.8 Smart Radiosity

4.8.1 Erweiterung um Volumen

Wie auch alle anderen LightOps mußte der **Smart Radiosity** Algorithmus um die Fähigkeit, Szenen mit Volumen berechnen zu können, erweitert werden, um ihn als Alternative für den Photon Maps Algorithmus bei indirekter diffuser Beleuchtung verwenden zu können.

```
class VisSRLightOp : public VisLightOp, public VisSampleGenFactory
{
...
public:
  getLightInfo();
...
};
```

Für jedes Volumen im Szenenbaum wird ein zugehöriges *VisSRVolume* Objekt erzeugt.

```
class VisSRVolume : public VisSRObject, public VisAttribute
{
...
public:
  subdivide();

  VisVolumeSub *getVolumeSubdiv();

  doInitialDivision();

protected:
  VisVolumeSub *volume;
...
};
```

Dieses Objekt stellt die Schnittstelle des Volumens zum Radiosity Algorithmus dar. Alle Anfragen von Smart Radiosity an das Volumen werden an dieses Objekt gerichtet und dann gegebenenfalls an das tatsächliche Volumen über ein zugehöriges *VisVolumeSub* Objekt weitergereicht. Jedes *VisSRVolume* Objekt besitzt ein *VisVolumeSub* Objekt, das unter anderem die Information enthält, welchen Teilbereich des Gesamtvolumen dieses SRVolume darstellt. Dieses *VisVolumeSub* Objekt ist es auch, daß durch die *subdivide()* Methode sich selbst wieder in acht neue Kinder unterteilen kann. In jedem SRVolumeobjekt, das ein Blatt des zugehörigen Volumen-Unterteilungsbaumes darstellt, ist dann die Beleuchtung analog zu Patches bei Flächen konstant.

```
class VisVolumeSub
{
...
public:
  VisCardinal subdivide();

  // get number of children
  VisCardinal getSubdivisions();

 // get child number i
  VisVolumeSub *getSubdivision(VisCardinal i);
...
};
```

Das *VisSRVolume* Objekt cached einmal berechnete Werte des Volumens um das Verfahren zu beschleunigen. Aus diesem Grund wird auch eine initiale Unterteilung des Volumens über die Methode *doInitialDivision()* vorgenommen. Die benötigten Unterteilungen im Laufe des Radiosity Algorithmus können dadurch effizienter durchgeführt werden. Der Vorteil dieses Bottom-Up Verfahrens ist, daß durch diese anfängliche gleichförmige Unterteilung in den früheren Unterteilungsschritten noch nichts berechnet werden muß, sondern die Werte dort

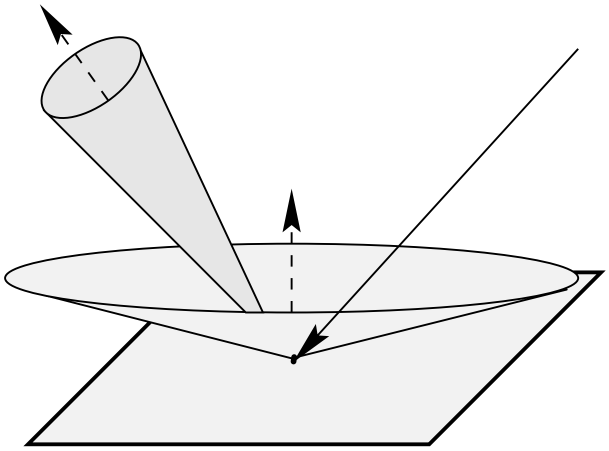

Abbildung 4.6: Reflexionsbereiche, die über Cones of Normals definiert sind.

durch einen Mittelungsschritt über alle Kinder berechnet werden können. Das heißt, daß nur in den Blattknoten Anfragen an das Volumen aus dem Szenenbaum gestellt werden müssen. Die anfängliche Unterteilungstiefe wird von dem zugehörigen Volumentyp über die Methode *getMinSampleSize()* erfragt. Dieser Wert hängt zum Beispiel bei einem Uniformen Volumen direkt von der Stützpunktverteilung ab.
Falls die Differenz der Werte der erzeugten Kinder unter eine gewisse Schwelle fällt, wird die Unterteilung des Volumens an dieser Stelle wieder rückgängig gemacht. Hiermit wird verhindert, daß eine künstliche Detailfülle vorgetäuscht wird. Diese würde zu einer unnötigen Unterteilung der anderen Objekte der Szene führen. Wenn das Volumen im Laufe des Radiosity Algorithmus noch tiefer unterteilt werden muß, erfolgt dies mit einem gewöhnlichen Top-Down Verfahren über die Methode *subdivide()*.

4.9 Lichtpfade

Damit bei einer Kombination von mehreren Verfahren keine Beleuchtung doppelt berechnet wird, muß man die Unabhängigkeit der einzelnen Verfahren gewährleisten. Dies kann man erreichen, indem die Lichtpfade, die die einzelnen Teilalgorithmen berechnen, bestimmt und miteinander verglichen werden. Dabei soll sichergestellt werden, daß kein Lichtpfad doppelt vorhanden ist.
Wenn nun zwei verschiedene Algorithmen die gleiche Art der Beleuchtung (zum Beispiel direkt) aber verschiedene Teilbereiche davon (zum Beispiel diffus $\leftrightarrow$ spekular) berechnen, dürfen diese auch nur den jeweiligen Lichtanteil verwenden.
In Vision liefert jeder Shader sogenannte *Cone Of Normals* (siehe Abbildung 4.6), die angeben, in welchen Bereich die zugehörigen Objekte Licht reflektieren beziehungsweise emittieren. Diese Bereiche werden über Koeffizienten für den Öffnungswinkel und den zugehörigen Exponenten definiert. Hierdurch kann man bei einem Shader den diffusen und den spekularen Anteil trennen, wodurch man die Beleuchtungsberechnung gut auf einzelne

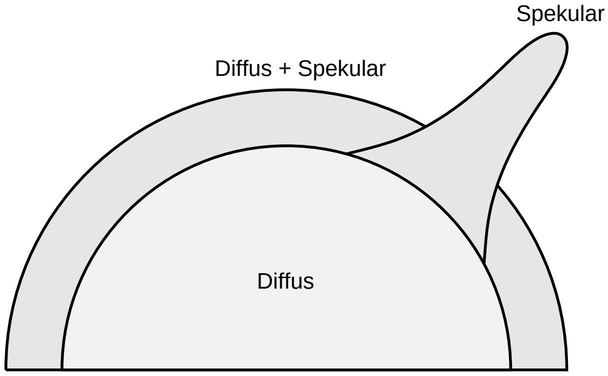

Abbildung 4.7: Trennung zwischen diffusem und spekularem Beleuchtungsanteil.

LightOps aufteilen kann. So kann zum Beispiel das **Enhanced Irradiance Gradients LightOp** die indirekte diffuse Beleuchtung berechnen und das **Caustics LightOp** die Kaustiken, ohne daß doppelte Lichtpfade verfolgt werden.
Um die Unterscheidung über die *Cones Of Normals* transparent zu machen, wurde die Klasse **VisShader** erweitert. Hier kann jetzt explizit auch diffuse Radiance über die Methode *getDiffuseRadiance()* berechnet werden. Die Methode *getRadiance()* liefert den gemittelten diffusen und spekularen Anteil und die Differenz *getRadiance() - getDiffuseRadiance()* den spekularen Anteil für jede benötigte Position (siehe auch Abbildung 4.7).

```
class VisShader : public VisAttribute
{
...
public:
  VisLight getRadiance();

  VisLight getDiffuseRadiance();
...
```

Die Methode *getDiffuseRadiance()* überprüft bei jedem Cone den Öffnungswinkel und Exponenten, und nur wenn einer von diesen beiden ein vorgegebenes Kriterium erfüllt (hier: der Winkel muß $> \pi/4$ sein oder der Exponent $< 10)$[1], wird die Beleuchtung für diesen berechnet.

[1]Dies ist äquivalent der Methode *isSpecular()* aus der Klasse **VisLighting**.

```
VisColor getDiffuseRadiance() {

  VisIllumReceiver* receiver = Shader->getReflectanceApprox(Richtung);

  VisColor Color;
  double weight = 0.0;


  for (VisImportanceConeIt i = rec->getCones(); !i.eol(); ++i) {
    // bin ich diffus?
    if ((*i).angle > M_PI/4 || (*i).exp < 10) {
      weight += (*i).getWeight();
    }
  }

  for (VisImportanceConeIt i = rec->getCones(); !i.eol(); ++i) {
    // bin ich diffus?
    if ((*i).angle > M_PI/4 || (*i).exp < 10) {

      ...

      Color *= ((*i).getWeight() / weight);
    }
  }

  ...

  return Color;
}
```

Kapitel 5

Vergleich der Verfahren

Die hier vorgestellten und auch implementierten Algorithmen erlauben nun vielfältige Möglichkeiten der Kombination von zugehörigen LightOps zu einem großen Gesamtsystem. Dabei sind natürlich nicht alle sinnvoll. Wie schon im Abschnitt 2.3.6 ausgeführt wurde, muß auf jeden Fall sichergestellt werden, daß keine Lichtpfade doppelt berechnet werden. So ergeben sich die verschiedenen praktikablen Kombinationen durch die Auswahl von jeweils einem Algorithmus aus den drei Bereichen direkte, indirekte diffuse und indirekte spekulare Beleuchtungsberechnung. In diesem Kapitel wird nun ein Vergleich zwischen den in diesen einzelnen Bereichen konkurrierenden Verfahren durchgeführt, um dann das insgesamt beste Hybridverfahren über diese drei Bereiche zu ermitteln. Hierzu wurden Tests mit verschiedenen Testszenen durchgeführt, vor allen mit einer Szene, die eine Variation der Cornellbox darstellt und eine geschlossene Kiste mit zwei darin eingeschlossenen verschieden großen Quadern enthält. Die Reflexionseigenschaften der Objekte, zum Beispiel ob und wie sehr sie spekular reflektieren, werden jeweils den Bedürfnissen der Vergleiche angepaßt.

5.1 Direkte Beleuchtung

Bei der direkten Beleuchtung bieten sich das Ray Tracing Verfahren durch das **Direct LightOp** (siehe Abschnitt 4.2) und auch der Smart Radiosity Ansatz durch das **SRLightOp** (siehe 4.8) an. Beide Verfahren können die direkte Beleuchtung berechnen, aber auf stark unterschiedlichen Arten. Der Radiosity Ansatz zerlegt die Objekte der Szene in finite Elemente, für diese wird dann jeweils an einem Punkt die Beleuchtung berechnet und anschließend durch einen Ray Tracing Nachbearbeitungsschritt visualisiert, optional wird hier die Patch-basierte Beleuchtungsinformation durch ein Gouraud - Shading gefiltert (siehe Bild B.3). Hier ergeben sich aber die Prämissen, daß die bei Radiosity verwendeten Flächen perfekt diffus, also Lambertsche Strahler, sein müssen, und alle enthaltenen Objekte in Sub-Objekte zerlegbar sein müssen. Beim Ray Tracing hingegen werden Strahlen vom Augpunkt in die Szene verschossen, deshalb können hiermit problemlos spezielle Reflexionsarten wie Spiegelungen oder spekulare Highlights auf Objekten berechnet werden (siehe auch Bild B.4). Ferner hat dieses Verfahrens an Geometrieobjekte nur die Anforderung, bei diesem Algorithmus eine Schnittpunktberechnung mit einem Strahl zulassen zu müssen. Aus diesem Grund ist die Auswahl an verwendbaren Objekttypen größer als bei der Radiosity.

Der Radiosity Ansatz kann deshalb nur LDE Pfade (siehe auch 2.3.6) berechnen, mit Ray Tracing kann hingegen die vollständige direkte Beleuchtung, also L(D|S)E, ermittelt werden.

5.2 Indirekte diffuse Beleuchtung

Für die indirekte diffuse Beleuchtung kommen drei der hier verwendeten Algorithmen in Frage, die Strahl-basierten Ansätze Photon Maps und Enhanced Irradiance Gradients, sowie das Patch-basierte Smart Radiosity Verfahren. Im Folgenden werden zuerst die beiden Strahl-basierten Verfahren verglichen, und dann der sich hier als geeigneter erweisende Ansatz mit Smart Radiosity. Bevor auf den Vergleich eingegangen wird, werden die Vorzüge des erweiterten Enhanced Irradiance Gradients Ansatzes gegenüber dem ursprünglichen Irradiance Gradients Ansatz aufgezeigt.

5.2.1 Irradiance Gradients versus Enhanced Irradiance Gradients

Bereits das ursprüngliche Irradiance Gradients Verfahren kann indirekte diffuse Beleuchtung berechnen, wenn auch nicht bei Szenen, die Volumen enthalten. Abgesehen von dieser Erweiterung auf Volumen hat die Einbeziehung der Photon Maps in das Irradiance Gradients Verfahren auch noch andere Vorteile. Durch die vorgenommenen Änderungen ist es jetzt möglich, auch schon durch geringere Rekursionstiefen die indirekte Beleuchtung gut anzunähern, da die bei einem Rekursionsabbruch zurückgelieferte Farbe über die Dichte der Photonen aus der zugehörigen Photon Map der tatsächlichen Beleuchtung in einer Szene gut angenähert werden kann. Dies fällt vor allem in Szenen auf, die eine lokal stark unterschiedliche indirekte Beleuchtung aufweisen, da hier der Wert, der bei einem Rekursionsabbruch gewählt wird, dieser unterschiedlichen Lichtverteilung durch verschiedenen Dichten der Photonen angepaßt werden kann.

5.2.2 Photon Maps versus Enhanced Irradiance Gradients

Das in Bild B.5 aufgezeigt Problem des ursprünglichen Photon Maps Algorithmus ist es, daß das berechnete Bild durch das Monte Carlo basierte Importance Tracing stark verrauscht ist. Dieses Rauschen ist zwar laut Henrik Wann Jensen [11] um bis zu 70% gegenüber einem puren Monte Carlo Sampling reduziert, aber leider immer noch stark sichtbar. Dies kann, wie in 2.3.4 beschrieben, durch Erhöhen der Strahlen pro Rekursion zwar verringert werden, die Anzahl der Strahlen, die hierzu benötigt werden, wird so aber schnell sehr groß.
Im Gegensatz hierzu treten diese Artefakte beim Enhanced Irradiance Gradients Verfahren nicht auf. Das Wiederverwerten von benachbarten Werten auf Flächen, auf denen sich die indirekte Beleuchtung nur wenig und langsam ändert gewährleistet einen glatten Helligkeitsverlauf (siehe Bild B.6). Außerdem sorgt das Caching dafür, daß die Strahlenanzahl nicht zu groß wird.

5.2.3 Enhanced Irradiance Gradients versus Smart Radiosity

Die Erweiterung der Smart Radiosity auf Volumen konnte bis zum jetzigen Zeitpunkt noch nicht fertiggestellt werden. Die theoretischen Vorgaben sind zwar schon vollendet, und die Berechnung der direkten Beleuchtung funktioniert ebenfalls korrekt (siehe Bild B.8), indirekte Beleuchtung kann aber noch nicht berechnet werden. In Bild B.8 wurde für die abschließende Rekonstruktion Flat-Shading gewählt wurde, um die Unterteilung der Szene durch den Radiosity Ansatz besser zeigen zu können.
Aus diesem Grund erfolgt hier der Vergleich nur mit dem bereits vorhandenen und auch vollständig implementierten Smart Radiosity Algorithmus für Flächen, die Übertragung der Aussagen auf Szenen mit Volumen sollte aber keine Veränderung der Ergebnisse und der Schlußfolgerungen hieraus bewirken. Die Vor- und Nachteile der beiden Strahl-basierten Verfahren Photon Maps und Enhanced Irradiance Gradients (EIG) untereinander wurden schon im Abschnitt 5.2.2 erörtert, für den Vergleich wird deshalb das beste Strahl-basierte Verfahren EIG mit dem Patch-basierten Verfahren Smart Radiosity (SR) verglichen.
Es hat sich gezeigt, daß der visuelle Eindruck der Bilder meist ähnlich ist. Das mit EIG berechnete Bild B.6 benötigte auf einem R10000 Prozessor eine Berechnungszeit von 16 Minuten, das mit Smart Radiosity berechnete Bild B.7 eine Zeit von 14 Minuten. Diese Zeiten sind jedoch stark abhängig von den jeweils gewählten Parametern, hier wurde Bild B.6 mit einer maximalen Rekursionstiefe von 4 und einer anfänglichen Strahlanzahl pro Halbraum von 256 erzeugt, Bild B.7 ebenso mit einer Rekursionstiefe von 4.
Bezüglich des Speicherverbrauches ist festzustellen, daß dieser bei Radiosity erwartungsgemäß höher liegt, beide Ansätze reizen die verwendeten Rechner aber nicht vollständig aus. Auch die zu erwartenden Effekte wie z.B. Color-Bleeding sind bei beiden berechneten Bildern zu beobachten. Was letztendlich den Ausschlag für den Radiosity Ansatz gibt, ist dessen Stabilität bei vielen etwas komplexeren Szenen, der EIG Ansatz hat hier durchaus einige Probleme (siehe auch [19]), wie zum Beispiel an deutlich sichtbaren indirekten Schattenkanten, die stark hiermit verwaschen werden.

5.3 Indirekte spekulare Beleuchtung (Kaustiken)

Bei der indirekten spekularen Beleuchtung ergab sich im Laufe der Arbeit keine Alternative zu dem durch das **Caustic LightOp** implementierte Photon Maps Verfahren, welches die Kaustiken sowohl auf Flächen als auch bei Volumen berechnen kann. Ein hierzu passendes Lighting Network wurde unter anderem in 3.4 vorgestellt.

5.3.1 Erweiterungen vom Caustic LightOp

Bei Testen des Photon Maps Verfahrens für Kaustiken haben sich einige Probleme ergeben. Deshalb wurden mehrere Verbesserungen dieses Algorithmus entwickelt, die diese Probleme teilweise beheben. Die Kantenbereiche bei Kaustiken sind bei Berechnung mit dem Standardalgorithmus auf spekular angestrahlten diffusen Flächen dunkler als der Rest der Kaustik (siehe Bild B.9). Der hierfür entwickelte Korrekturalgorithmus (siehe 3.2.2.2) für Kaustiken an Objektkanten (siehe Bild B.10) erzieht demhingegen eine saubere Darstellung der Kaustiken auch auf solchen Stellen. Zusätzlich kann mit dem Gewichtungsalgorithmus (siehe 3.2.2.1) die Wirkung von gestreuten Photonen lokal stärker eingrenzen werden, wie Bilder B.14, B.15 und B.16 zeigen, als Exponent wurden hier die Werte 0, 1 und 2 gewählt. Trotz all dieser Bemühungen konnten die visuellen Artefakte nicht vollständig beseitigt werden, deshalb wurde nach dem Teilalgorithmus gesucht, der diese größtenteils verursacht.

5.3.2 Particle Tracer versus Bounded Radiance

Es hat sich herausgestellt, daß das Verschießen der Photonen mit dem hier implementierten **Particle Tracer** im ersten Schritt des Algorithmus das größte Problem darstellt, da dieses durch einen auf Monte Carlo Sampling basierenden Algorithmus erfolgt. Hier läßt es sich nicht vermeiden, daß das Verfahren auch einzelne 'Ausreißer' produziert, das heißt Photonen in von der eigentlichen Kaustik entfernten Bereichen, die sich von der Helligkeit ihrer Nachbarn stark unterscheiden. Dies führt zu den in Bild B.12 sichtbaren Artefakten. Auch das Verschicken einer größeren Anzahl von Photonen kann diesen Effekt nicht verhindern, da die Rekonstruktion auf einer bestimmten Maximalanzahl von Photonen basiert und deshalb auch der maximale Suchradius kleiner werden würde. Das Extremverfahren, einfach den Suchradius der Photonen entsprechend groß zu machen, bis das Gewicht der problematischen Photonen bei der Rekonstruktion sehr klein wird, ist auch keine Lösung. Die 'Ausreiser' verschwinden zwar, ein größer Suchradius bedeutet aber auch, daß das gesamte Bild mit einem starken Tiefpassoperator gefiltert wird, deshalb gehen so alle Details der Kaustik verloren.
Eine Änderung, die diese Artefakte tatsächlich beheben könnte, müßte dementsprechend schon auf der **Particle Tracer** Seite greifen, da die Verteilung der Photonen diese Artefakte bewirkt, wie unter anderem Bild B.13 zeigt, bei dem der Suchradius bei der Rekonstruktion stark verkleinert wurde, damit hier die Lage der einzelnen Photonen erkennbar ist. Um zu zeigen, daß eine gleichmäßigere Verteilung der Photonen diese Artefakte behebt, wurde zu Testzwecken der **Particle Tracer** durch das Verfahren **Bounded Radiance** [30] ausgetauscht. **Bounded Radiance** ist ein auf Patches basierendes Verfahren, das auch solche Effekte wie Kaustiken berechnen kann (siehe Bild B.18). Bei diesem Verfahren lassen sich die Patch-basierenden Ergebnisse am Schluß in eine Kd-Tree Darstellung umwandeln, die dann die zu der berechneten Beleuchtung äquivalente Photonendarstellung enthält.
Das **Bounded Radiance** Verfahren kann leider nicht mit allen Testszenen für Kaustiken verwendet werden, aber wie zum Beispiel die Bilder B.17 und B.18 zeigen, die eine Kaustik darstellen, die von einem kleinem spekularem Flächenstück auf eine größere Fläche geworfen wird, ergeben sich mit diesem Verfahren keine visuellen Artefakte mehr. Dies bedeutet, daß die Rekonstruktionsmethode des **Caustic LightOp** korrekt funktioniert, auf der Seite des **Particle Tracers** aber durchaus noch Verbesserungen möglich sind.

Kapitel 6

Bewertung

Wie im letzten Kapitel gezeigt wurde, können große Unterschiede zwischen den einzelnen Verfahren festgestellt werden. Wenn nun diese Erkenntnisse alle zusammengefaßt werden, läßt sich jetzt ein einziges, für die meisten Szenen geeignetes Hybridverfahren zusammenstellen:

- Bei direkter Beleuchtung stehen entweder das Ray Tracing Verfahren oder der Radiosity Ansatz zur Auswahl, wobei sich mit letzteren nur diffus abstrahlende Objekte verwenden lassen. Beim Ray Tracing hingegen kann man problemlos auch Spiegeleffekte und Highlights (schwach spekulare Flächen) berechnen. Deshalb wird hier für die direkte Beleuchtungsberechnung dieses Ray Tracing Verfahren vorgezogen, da sich hier bei allen Tests auch keine besonderen Schwächen ergeben haben.

- Bei der indirekten diffusen Beleuchtung gibt es einige verschiedene Möglichkeiten zur Berechnung dieses Beleuchtungsbereiches. Es hat sich ergeben, daß das nur auf Photon Maps und Importance Tracing basierende Verfahren von Henrik Wann Jensen keine optimale Lösung darstellt. Das Laufzeitverhalten ist inakzeptabel, da viel zu viele Strahlen verschossen werden. Dies kann entweder durch die Einführung eines Cachings von Irradiancewerten (wie bei Enhanced Irradiance Gradients) geschehen, oder aber durch das Patch basiertes Verfahren Smart Radiosity, wobei sich im Laufe dieser Arbeit gezeigt hat, daß ein der Radiosity Algorithmus hier unter gewissen Voraussetzungen die beste Lösung darstellt. Die einzigen benötigten Prämissen sind, daß sich die einzelnen Objekte der Szene in finite Elemente zerlegen lassen, und daß der verwendete Rechner mit genügend Hauptspeicher ausgestatten ist (siehe auch Abschnitt 5.2.3).

- Bei Kaustiken wurde im Laufe der Arbeit keine echte Alternative zu dem Photon Maps basierten Monte Carlo Verfahren gefunden, wobei dieses Verfahren aber durchaus noch seine Schwächen hat Hier ist wohl noch am meisten Freiraum für Verbesserungen.

Hieraus ergibt sich als letztendlich bestes Hybridverfahren das in Abbildung 6.1 dargestellte Lighting Network, das nun eine sehr gute Verbindung der einzelnen Algorithmen darstellt.

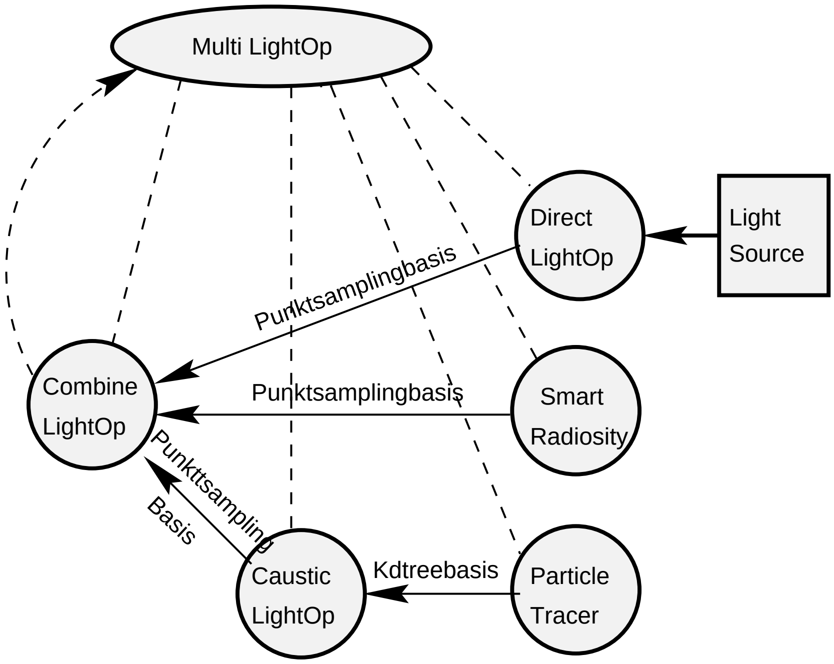

Abbildung 6.1: Ein Lighting Network für sowohl direkte, indirekte Beleuchtung als auch zur Kaustikenberechnung

Das in der dieser Diplomarbeit entwickelte Hybridverfahren stellt natürlich noch nicht das Ende der möglichen Entwicklung dar. Es hat sich aber gezeigt, daß es durchaus sinnvoll ist, die Beleuchtung einer Szene, die möglichst viele Lichtpfade berücksichtigen soll, nicht nur von einem einzigen Algorithmus berechnen zu lassen. Es läßt sich nicht ein 'ultimativer' Beleuchtungsalgorithmus bestimmen, der als Lösung für alle Probleme angesehen werden kann. Vielmehr ist die geschickte Kombination von mehreren Verfahren aus unterschiedlichen Bereichen ein vielversprechender Weg, dem Ziel, ein effizientes Verfahren zu entwickeln, welches die gesamte Beleuchtung einer Szene berechnet, näher zu kommen.
Bei den in dieser Arbeit verglichen Algorithmen hat sich dementsprechend gezeigt, daß jedes Verfahren einen bestimmten Bereich aufweist, in dem es den anderen Verfahren überlegen ist. Es hat sich aber auch ergeben, daß die meisten dieser Verfahren in vielen Bereichen noch verbessert werden können.
So ist vor allem das Berechnen der Kaustiken durch den Photon Maps Ansatz noch nicht zufriedenstellend. Auch bei der indirekten diffusen Beleuchtung hat sich kein eindeutiger Favorit ergeben, dem Radiosity Ansatz in seiner jetzigen Form ist zwar in Augenblick der Vorzug zu geben, aber auch das durch die Photon Maps erweiterte Enhanced Irradiance Gradients Verfahren berechnet diesen Beleuchtungsanteil in vielen Szenen ähnlich gut. Hier läßt sich erst durch weitere Forschungen in beiden Bereichen abschließend klären, welches dieser beiden Verfahren, wenn überhaupt, sich als das endgültig bessere erweist.

Anhang A

Benutzung von Vision und den einzelnen LightOps

Wenn man mit Vision eine Szene mit einem hier vorgestellten Algorithmus berechnen lassen will, kann man diesen über die Kommandozeilenoption **-lnw** auswählen, wobei zu beachten ist, daß bei den folgenden Netzwerken die Kombination mit der direkten Beleuchtung herausgelassen wurde, um die Ergebnisse der einzelnen Algorithmen besser bewerten zu können:

```
vision -lnw <network>:<outputmode> (<rib file>|<ribm4 file>|<mgf file>)
```

Als Parameter für **-lnw** stehen folgende Möglichkeiten zur Wahl:

- **gradients:rt**: Mit dieser Option kann das Irradiance Gradients Netzwerk (siehe Abschnitt 2.3.5) gewählt werden, bei dem die indirekte diffuse Beleuchtung mit dem **Irradiance Gradients** LightOp berechnet wird.

- **irphoton:rt**: Dies aktiviert das **Enhanced Irradiance Gradients** Netzwerk (siehe Abschnitt 3.4). Es wird hier die indirekte diffuse Beleuchtung auf Flächen mit dem **Irradiance Gradients** LightOp und bei Volumen mit dem **Photon Map** LightOp bestimmt, wobei die Photonen für beide Algorithmen durch den **Particle Tracer** verschossen werden.

- **radiosity:rt**: Durch diesen Schalter kann man eine Berechnung durch **Smart Radiosity** starten (siehe auch Abschnitt 4.8.1). Hiermit kann sowohl direkte als auch indirekte diffuse Beleuchtung bestimmt werden.

- **br:rt**: Hiermit kann man den **Bounded Radiance** (siehe 5.3.2) Algorithmus verwenden, der unter anderem auch Kaustiken berechnen kann.

- **photon:rt**: Das **Photon Map** Netzwerk berechnet die indirekte diffuse Beleuchtung mit dem mit dem **Photon Map** LightOp, die Photonen werden durch den **Particle Tracer** verschossen (siehe auch Abschnitt 3.2.1).

- **caustics:rt**: Mit diesem Schalter wird das **Caustic** LightOp aktiviert, das Kaustiken auf Flächen und in Volumen berechnen kann (siehe 3.2.2). Das Photonenverschicken erfolgt wiederum durch den **Particle Tracer**.

- **photonall:rt**: Wenn eine Berechnung mit dem in Kapitel 6 vorgestellten Hybrid-Netzwerk erfolgen soll, ist dies die richtige Option. Hier wird die direkte Beleuchtung mit dem **Direct** LightOp, die indirekte diffuse mit dem **Smart Radiosity** LightOp und die Kaustiken mit dem **Caustic** LightOp berechnet.

Anhang B

Bildteil

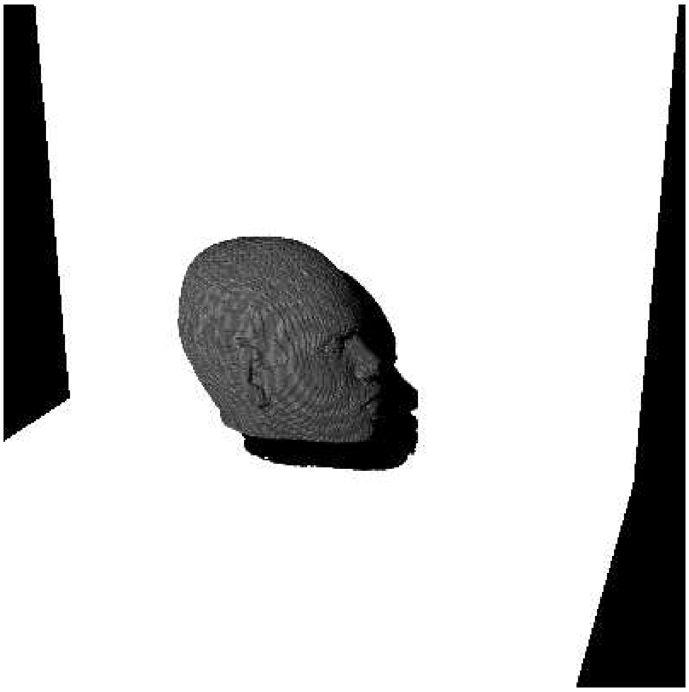

Abbildung B.1: Ein Kopf aus einem medizinischen Datensatz, mit Ray Tracing visualisiert.

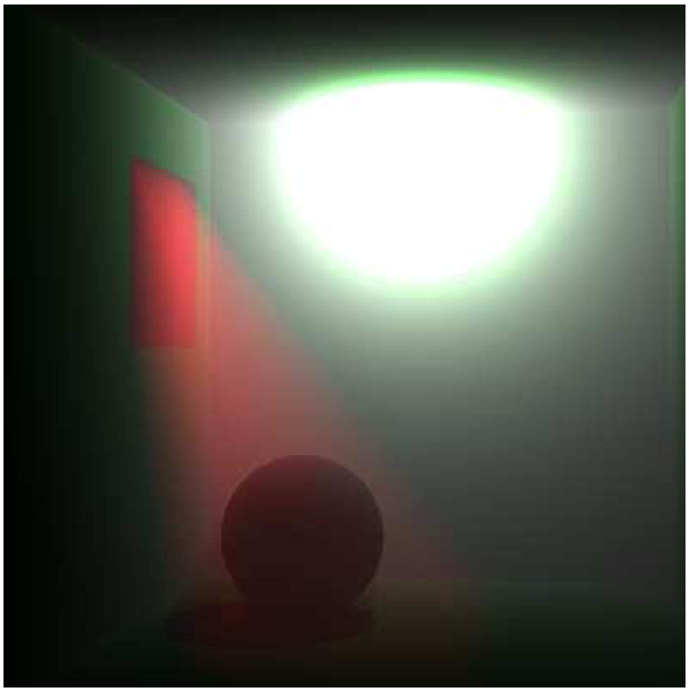

Abbildung B.2: Nebel in einem Zimmer mit einem semitransparenten roten Fenster.

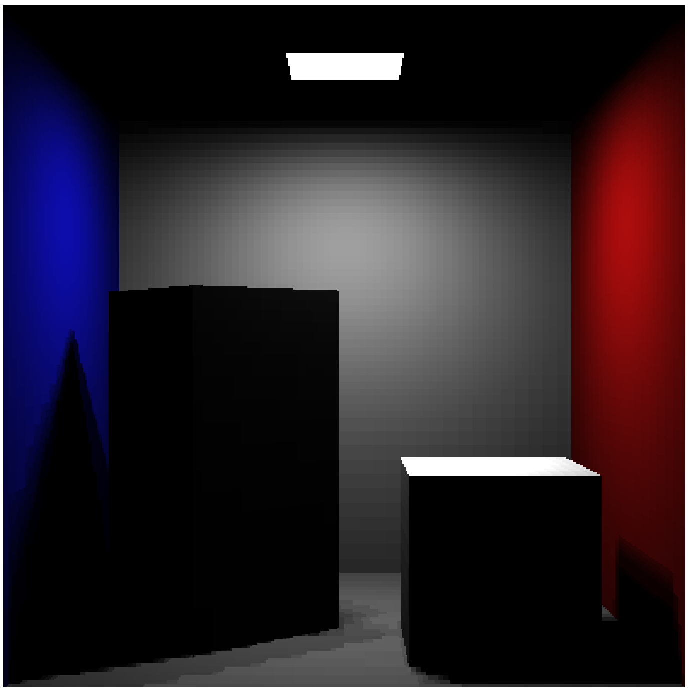

Abbildung B.3: Direkte Beleuchtung mit Radiosity.

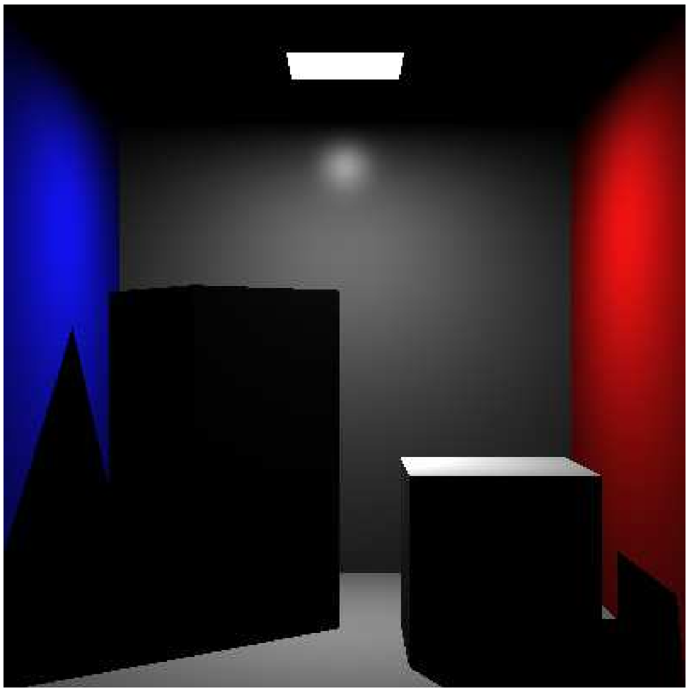

Abbildung B.4: Direkte Beleuchtung mit Ray Tracing.

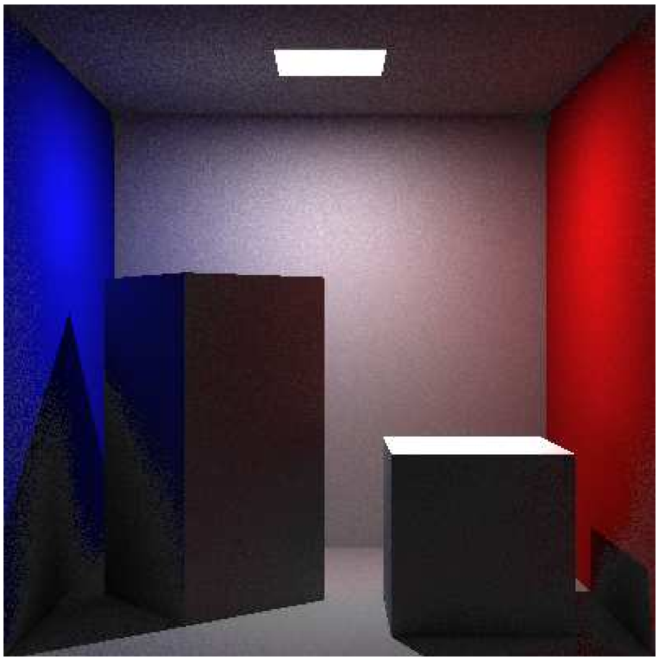

Abbildung B.5: Direkte und indirekte Beleuchtung mit dem Verfahren Photon Maps berechnet.

Abbildung B.6: Direkte und indirekte Beleuchtung mit dem Verfahren Enhanced Irradiance Gradients berechnet.

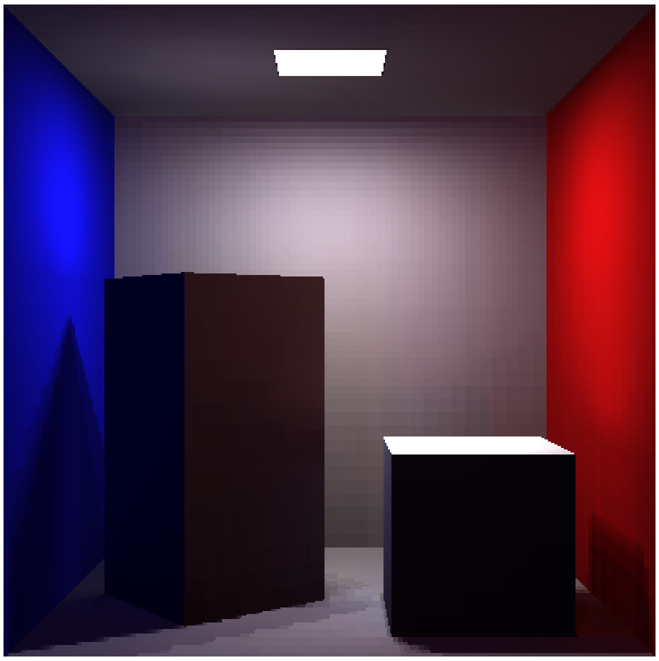

Abbildung B.7: Direkte und indirekte Beleuchtung mit dem Verfahren Smart Radiosity berechnet.

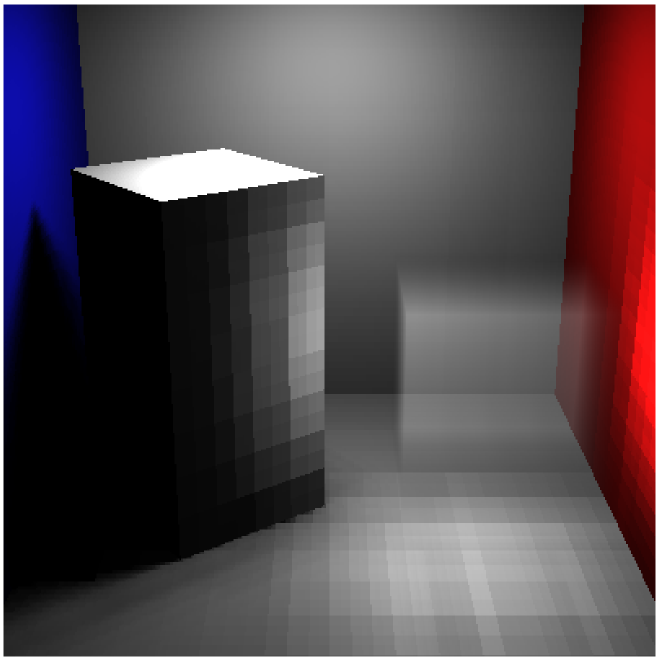

Abbildung B.8: Direkte Beleuchtung einer Szene mit Flächen und Volumen durch Radiosity und anschliessendem Flat-Shading.

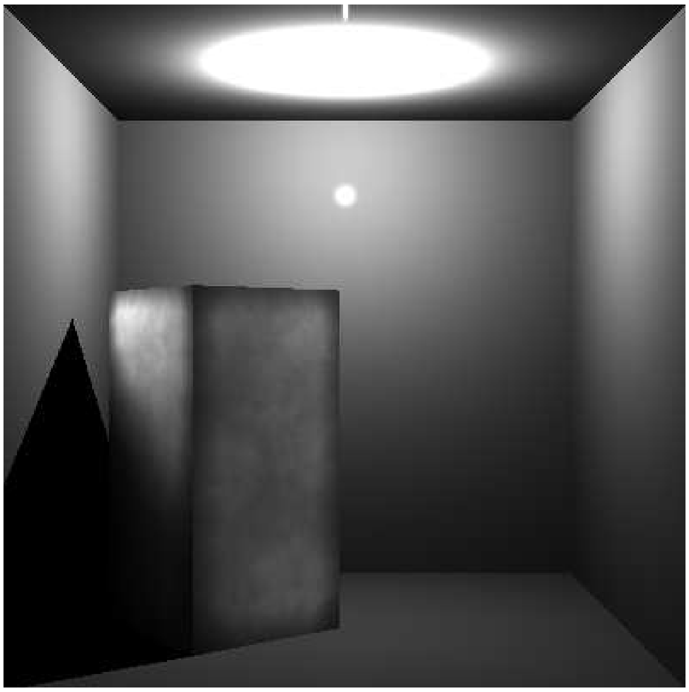

Abbildung B.9: Eine Kaustik, von spekularen Wänden auf einer diffusen Box erzeugt.

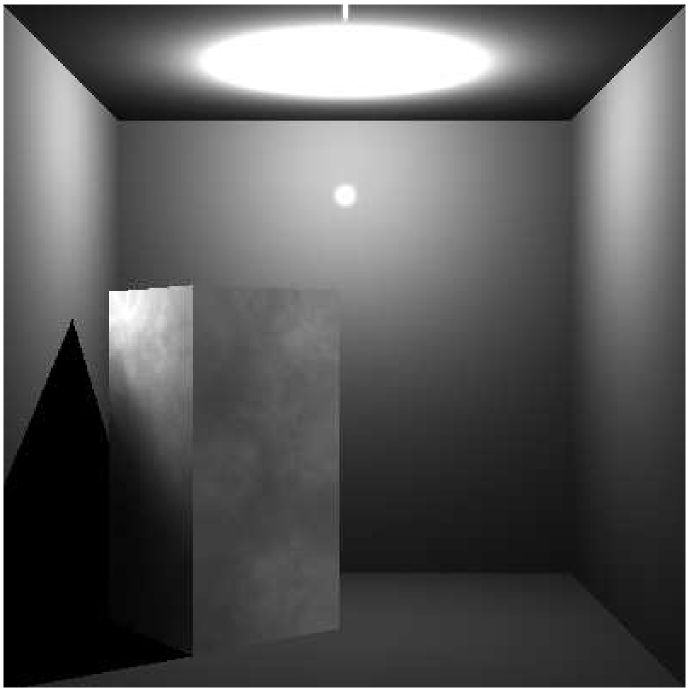

Abbildung B.10: Eine Kaustik, von spekularen Wänden auf einer diffusen Box erzeugt, mit Korrektur der Kanten.

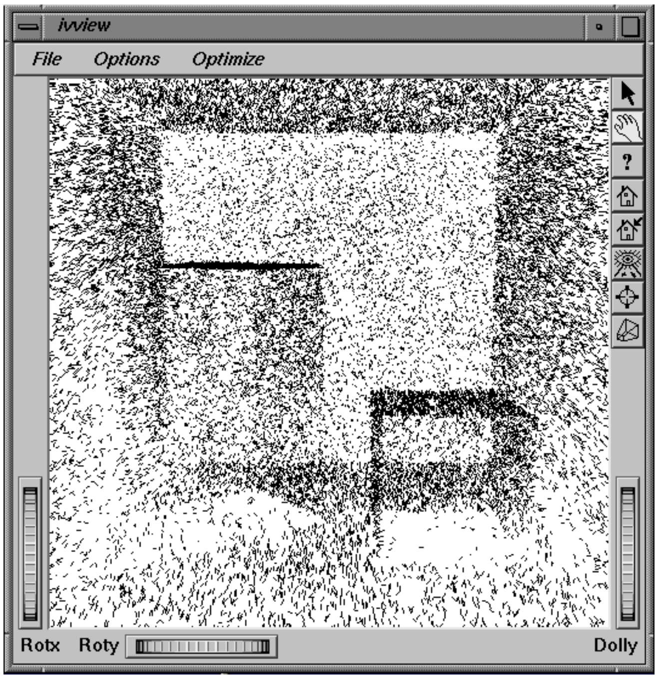

Abbildung B.11: Die Verteilung der Photonen in einer Szene zusammen mit der Einfallsrichtung.

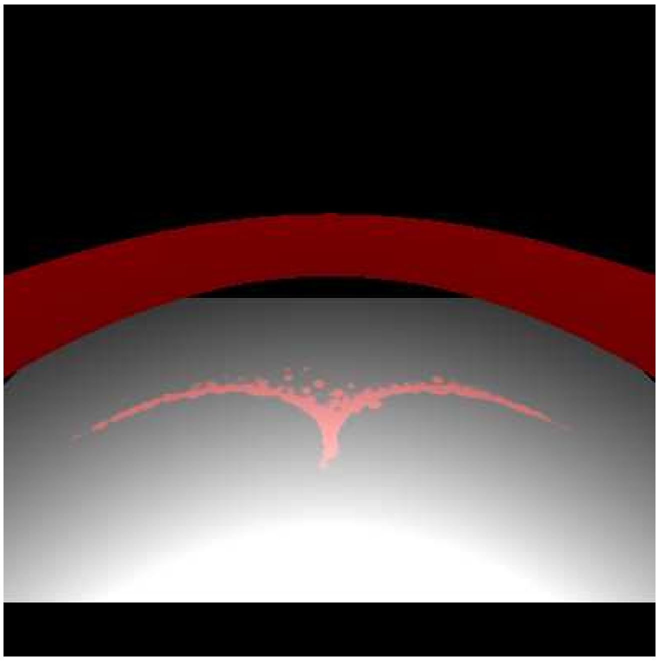

Abbildung B.12: Eine Kaustik, von einem Zylinder auf eine Ebene geworfen.

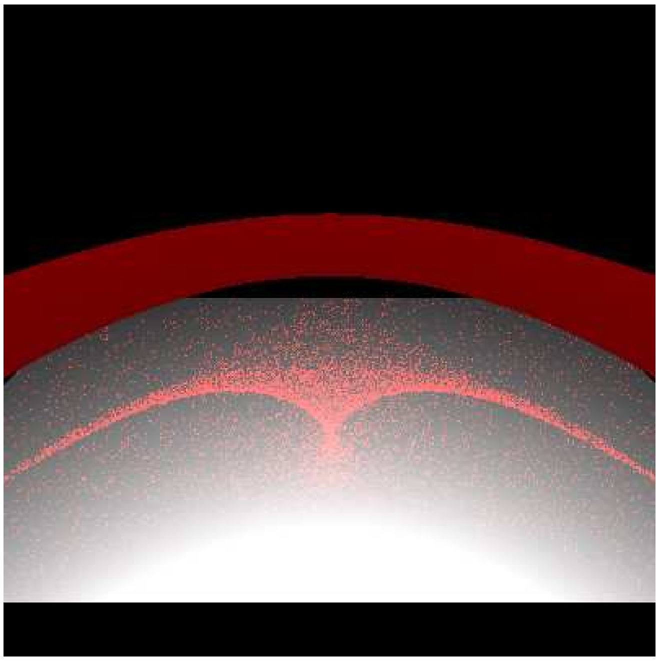

Abbildung B.13: 100000 Photonen einer Kaustik, mit kleinem Gültigkeitsradius, um die Verteilung zu visualisieren.

Abbildung B.14: Eine Kaustik mit keiner Entfernungsgewichtung.

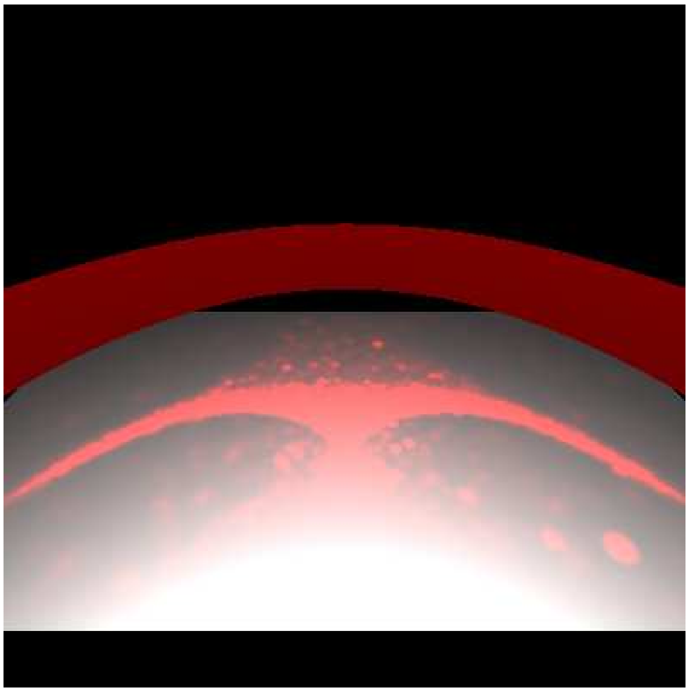

Abbildung B.15: Eine Kaustik mit einer linearen Gewichtung.

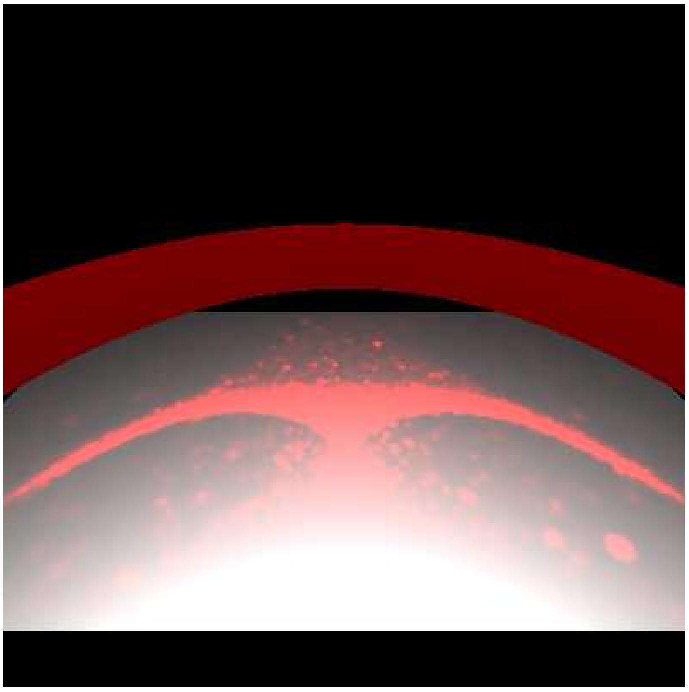

Abbildung B.16: Eine Kaustik mit einer quadratischen Gewichtung.

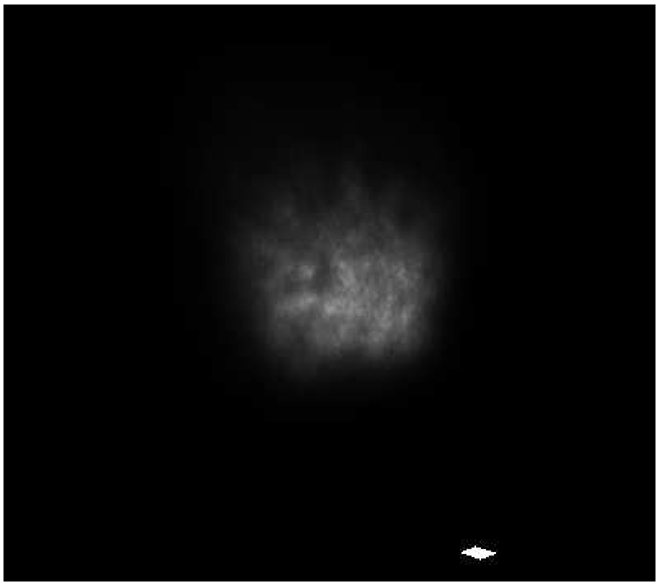

Abbildung B.17: Eine Kaustik, bei der die Photonen mit dem Particle LightOp verschossen wurden.

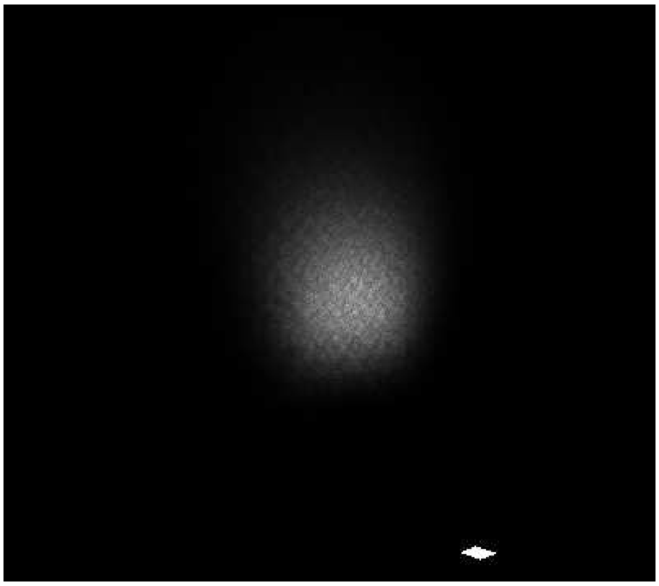

Abbildung B.18: Eine Kaustik, bei der die Photonen mit dem Bounded Radiance Algorithmus verschossen wurden.

Literaturverzeichnis

[1] James Arvo and David Kirk. Particle transport and image synthesis. *Computer Graphics*, 24(4):53–66, 1990.

[2] K.M. Case and P.F. Zweifel. *Linear Transport Theory*. Addison-Wesley, Reading, Massachusetts, 1967.

[3] S. Chandrasekhar. *Radiative Transfer*. Oxford University Press, 1950.

[4] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 137–45, July 1984.

[5] David S. Ebert, F.Kenton Musgrave, Darwyn Peachey Ken Perlin, and Steven Worley. *Texturing and Modeling: A Procedural Approach*. AP Professional, 1994.

[6] James D. Foley et al. *Computer graphics: principles and practice*. Addison-Wesley, Reading, Massachusetts, 1991.

[7] Andrew S. Glassner. *Principles Of Digital Image Synthesis*. Morgan Kaufman Publisher, San Francisco, 1995.

[8] C. Goral, K.E. Torrance, D.P. Greenberg, and B. Battaile. Modelling the interaction of light between diffuse surfaces. *Computer Graphics*, 18(3):212–222, 1984.

[9] H. C. Hege, T. Höllerer, and D. Stalling. Volume rendering. Technical Report TR 93-7, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1993.

[10] L.G. Henvey and J.L. Greenstein. Diffuse reflection in the galaxy. In *Astrophysics*, page 70, 1941.

[11] Henrik Wann Jensen. Importance-driven path tracing using the photon map. In *Rendering Techniques '95 (Proceedings Sixth Eurographics Workshop on Rendering)*, pages 326–335. Springer, jun 1995.

[12] Henrik Wann Jensen. Global illumination using photon maps. In Xavier Pueyo and Peter Schröder, editors, *Rendering Techniques '96 (Proceedings Seventh Eurographics Workshop on Rendering)*, pages 21–30. Springer, June 1996.

[13] Henrik Wann Jensen. Globel illumination using photon maps. In *Rendering Techniques ´96 (7th Eurographics Workshop on Rendering)*, pages 21–30, 1996.

[14] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 311–320, 1998.

[15] James T. Kajiya. The Rendering Equation. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150, August 1986.

[16] John Kaufmann. *IES Lighting Handbook, Reference Volume.* IESNA, New York, 1981.

[17] Eric P. Lafortune and Yves D. Willems. Rendering participating media with bidirectional path tracing. In *Rendering Techniques ´96 (7th Eurographics Workshop on Rendering)*, pages 92–101, 1996.

[18] Joseph O'Rourke. *Computational Geometry in C.* Cambridge University Press, New York, 1993.

[19] Matthias Ott. Ausnutzung der Beleuchtungskohärenz mit Irradiance Gradients, November 1997. Studienarbeit Universität Erlangen.

[20] Pixar. *The RenderMan Interface, Version 3.1*, September 1989.

[21] Jan Christian Popp. Integration von Multi-Pass-Verfahren in Vision. Master thesis, University of Erlangen, IMMD 9, Computer Graphics Group, March 1997.

[22] Radiance. http://radsite.lbl.gov/radiance/home.html.

[23] H. Schirmacher, M. Stamminger, and H.-P. Seidel. Unified Radiosity Using Flux. Technical report, Universiät Erlangen-Nürnberg, Erlangen, Germany, 1998.

[24] R. Siegel and J.R. Howell. *Thermal Radiation Heat Transfer.* Hemisphere Publishing, Washington DC, 1984.

[25] P. Slusallek and H.-P. Siedel. Vision - an architecture for global illumination calculations. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):77–96, March 1995. ISSN 1077-2626.

[26] Philipp Slusallek. Vision: An Architecture for Physically-Based Rendering. Technical report, Universiät Erlangen-Nürnberg, Erlangen, Germany, 1995.

[27] Phillip Slusallek, Marc Stamminger, Wolfgang Heidrich Jan-Christian Popp, and Hans-Peter Seidel. Composite lighting simulations with lighting networks. *Computer Graphics*, 18(2):22–31, 1998.

[28] Marc Stamminger, Philipp Slusallek, and Hans-Peter Seidel. Bounded radiosity – illumination on general surfaces and clusters. *Computer Graphics Forum (EUROGRAPHICS '97 Proceedings)*, 16(3), September 1997.

[29] Marc Stamminger, Philipp Slusallek, and Hans-Peter Seidel. Bounded clustering – finding good bounds on clustered light transport. In *Proc. Pacific Graphics '98*, 1998.

[30] Marc Stamminger, Philipp Slusallek, and Hans-Peter Seidel. Three point clustering for radiance computations. In *Rendering Techniques (Proc. Eurographics Workshop on Rendering '98)*, 1998.

[31] Steve Upstill. *The RenderMan Companion.* Addison-Wesley, Reading, Massachusetts, 1990.

[32] G. Ward, F. Rubinstein, and R. Clear. A ray tracing solution for diffuse interreflection. *Computer Graphics*, 22(4), August 1988.

[33] Gregory J. Ward and Paul Heckbert. Irradiance Gradients. *Third Eurographics Workshop on Rendering*, pages 85–98, May 1992.

[34] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 85–92, August 1988.

Erklärung

Ich versichere, daß ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und daß die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Ich bin damit einverstanden, daß die Arbeit veröffentlicht wird und daß in wissenschaftlichen Veröffentlichungen auf sie Bezug genommen wird.

Der Universität Erlangen-Nürnberg, vertreten durch den Lehrstuhl für Graphische Datenverarbeitung, wird ein (nicht ausschließliches) Nutzungsrecht an dieser Arbeit sowie an den im Zusammenhang mit ihr erstellten Programmen eingeräumt.

Erlangen, den 13. Januar 1999

(Matthias Ott)