Simulation und Photorealismus

Dissertation

der Fakultät für Informatik
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von

Dipl. Math. Jens-Uwe Hahn

aus Ofterdingen

Tübingen
2000

Tag der mündlichen Qualifikation: 19. Juli 2000
Dekan: Prof. Dr. Klaus-Jörn Lange
1. Berichterstatter: Prof. Dr.-Ing. Wolfgang Strasser
2. Berichterstatter: Prof. Dr. Wolfgang Küchlin

Kurzfassung

Durch die ständig steigende Rechenleistung haben Computer-generierte Bilder in vielen Anwendungen Einzug gehalten. Diese reichen von Planung und Design in der Architektur über synthetisch erzeugte Filme und Werbeclips bis hin zu Virtual-Reality-Anwendungen.

Dabei werden immer realistischere Darstellungen immer aufwendigerer Szenen gefordert. Dazu ist es notwendig, Bewegungsabläufe physikalisch korrekt darzustellen und die Beleuchtungssituation in den Szenen der Realität entsprechend wiederzugeben.

Der erste Teil der Arbeit befaßt sich mit der physikalischen Modellierung. Da es in vielen Situationen sehr schwierig und aufwendig wäre, Bewegungsabläufe und Verformungen von Objekten für ganze Animationssequenzen von Hand zu modellieren, muß auf Simulationen zurückgegriffen werden, die die physikalischen Vorgänge korrekt wiedergeben. So ergeben sich zum Beispiel die Bewegungen und Faltenbildung von Kleidungsstücken aus der Position und den Bewegungen der Personen, die diese tragen.

Diese Bewegungen und Verformungen werden durch Differentialgleichungen beschrieben, die sich aus den auf die simulierten Objekte einwirkenden Kräften ergeben. Diese Differentialgleichungen müssen anhand physikalischer Modelle bestimmt und dann numerisch gelöst werden. Eine wichtige Aufgabe dabei ist die Erkennung und Behandlung von Kollisionen mit Objekten der Umgebung, und bei deformierbaren Objekten auch die Eigenkollisionen. Dafür wurden im Rahmen dieser Arbeit effiziente Algorithmen entwickelt.

Der zweite Teil der Arbeit befaßt sich mit physikalischer Beleuchtungssimulation mit Hilfe des Radiosity-Verfahrens, das ein Strahlungsgleichgewicht des zwischen den einzelnen Oberflächen der Szene ausgetauschten Lichtes berechnet. Dieses Verfahren erlaubt die Berechnung photorealistischer Darstellungen und wird in vielen Anwendungen erfolgreich eingesetzt. Dies ist aber aufgrund des hohen Rechenaufwandes für dynamische Szenen meist auf Anwendungen in statischen Szenen beschränkt. Durch Ausnutzung von Kohärenzen kann aber ein großer Teil dieses Aufwandes eingespart werden.

Zunächst wird in die zugrundeliegenden physikalischen Vorgänge eingeführt und ein Überblick über bekannte Verfahren zur Berechnung von Radiosity-Lösungen gegeben. Im letzten Teil der Arbeit geht es schließlich um die Berechnung von

Radiosity-Lösungen für bewegte Szenen, insbesondere um die Berechnung von Filmsequenzen, wofür im Rahmen dieser Arbeit ein effizientes Verfahren entwickelt wurde.

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Graphisch-Interaktive Systeme des Wilhelm-Schickard-Instituts für Informatik an der Universität Tübingen.

Mein besonderer Dank gilt Herrn Professor Dr.-Ing. W. Straßer für die Betreuung und Unterstützung meiner Arbeit.

Main Dank gilt auch den Kollegen, insbesondere Ralf Sonntag und Bernd Eberhardt, für die gute Zusammenarbeit und die vielen wertvollen Anregungen.

Tübingen, Juni 2000 Jens-Uwe Hahn

Inhaltsverzeichnis

Kapitel 1

Einführung

Die Computer-Graphik hat sich mit der ständigen Steigerung der Rechenleistung zu einem wichtigen Teilgebiet der Informatik entwickelt. Durch Computer erzeugte Bilder sind aus vielen Anwendungen nicht mehr wegzudenken. Die möglichen Anwendungen reichen von Planung und Design in der Architektur über synthetisch erzeugte Filme und Werbeclips bis hin zu Virtual-Reality-Anwendungen.

Abbildung 1.1: Der Realitätseindruck und die räumliche Wirkung sind bei der Darstellung mit physikalisch korrekter Beleuchtung wesentlich besser.

Photorealistische Bilder und Echtzeitbewegungen durch dreidimensionale Modelle erlauben in der Architektur eine optische Kontrolle schon während der Planungsphase und ohne die Notwendigkeit, aufwendige Modelle zu bauen. So kann der Kunde, der sich das fertige Objekt aufgrund der Baupläne in der Regel nur unzureichend vorstellen kann, sein geplantes Objekt schon vor Baubeginn betrachten. Durch eine physikalisch korrekte Berechnung der Beleuchtungsverhältnisse kann eine wesentlich realistischere Darstellung und eine bessere räumliche Wirkung erzielt werden, als dies mit empirischen Verfahren möglich wäre (siehe Abbildung 1.1). Abweichungen von realen Beleuchtungsverhältnissen können von einem Betrachter aufgrund der Erfahrungen des täglichen Lebens leicht erkannt werden. Da solche Anwendungen heute bereits auf Standard-PC's in kurzer Zeit berechnet werden können, eröffnet sich damit ein sehr breites Anwendungsgebiet.

In öffentlichen Gebäuden, in Büroräumen oder Produktionshallen ist es außerdem wichtig, Beleuchtungsverhältnisse zu schaffen, die angenehme und den Vor-

schriften entsprechende Arbeitsbedingungen garantieren. Eine physikalisch korrekte Beleuchtungssimulation in der Planungsphase kann helfen, Planungsfehler zu vermeiden und Kosten einzusparen.

Auch in der Film- und Werbeindustrie erlangen synthetisch erzeugte Bilder immer mehr an Bedeutung. Durch die steigende Rechenleistung wird es zunehmend möglich, teure Kulissenaufbauten zu vermeiden und stattdessen auf den Einsatz von Computern zurückzugreifen. Um einen ausreichenden realistischen Eindruck zu erreichen, sind hier sowohl physikalisch korrekte Beleuchtungsverhältnisse, als auch physikalisch korrekte Bewegungsabläufe notwendig.

Dabei ist es in der Regel viel zu schwierig und viel zu aufwendig, die Bewegungsabläufe für die gesamte Animation manuell zu modellieren. Diese müssen vielmehr anhand weniger vorgegebener Randbedingungen und Einflüsse nach physikalischen Gesetzen berechnet werden. So ist es zum Beispiel viel zu aufwendig, die Bewegungen und die Faltenbildung von Kleidungsstücken für die gesamte Animation von Hand zu modellieren, zumal kleine Abweichungen von der Realität jedem Betrachter sofort ins Auge stechen. Diese müssen aus den Bewegungen der Personen nach physikalischen Gesetzen berechnet werden.

Virtual-Reality-Anwendungen erlauben es, komplizierte, teure oder gefährliche Vorgänge, wie zum Beispiel das Führen eines Flug- oder Fahrzeugs, Crashtests oder auch medizinische Eingriffe unter realistischen Bedingungen zu simulieren. Auch dabei ist es notwendig, die physikalischen Vorgänge korrekt zu simulieren.

Während es bei Anwendungen in der Beleuchtungsplanung auf physikalisch exakte Beleuchtungsergebnisse ankommt, ist bei der Architekturplanung und bei der Erzeugung von synthetischen Filmen und Werbeclips hauptsächlich der photorealistische Eindruck ausschlaggebend, bei Filmen und Werbeclips allerdings mit sehr hohen Anforderungen an die Qualität. Diese müssen frei von Artefakten, konsistent und flackerfrei sein.

Bei Virtual-Reality-Anwendungen sind die Anforderungen an die Qualität der Beleuchtungsrechnung nicht ganz so hoch, solange der realistische Eindruck dadurch nicht verloren geht. Dafür werden aber interaktive Reaktionszeiten auf Aktionen der Benutzer gefordert, so daß hier nur echtzeittaugliche Verfahren verwendet werden können. Auch die Berechnung der Bewegungsabläufe muß in hinreichender Genauigkeit in Echtzeit erfolgen.

Im Hinblick auf die für die Beleuchtungssimulation verwendeten Verfahren lassen sich die Anwendungen noch unterscheiden in solche, für die die Beleuchtung in statischen Szenen berechnet werden kann, das heißt Szenen, deren Objekte feste sich nicht verändernde Ausdehnungen, Positionen und Oberflächeneigenschaften besitzen, und solche, denen dynamische, sich verändernde Szenen zugrundeliegen (analog werden unbewegte, sich nicht verändernde Objekte einer Szene als statische Objekte bezeichnet, bewegte, sich verändernde Objekte als dynamische Objekte). Statische Szenen sind in der Regel bei Architektur- und Beleuchtungsplanungen zu finden, während es sich bei Filmen und Werbeclips in der Regel um dynamische Szenen handelt,

deren Veränderungen aber – im Gegensatz zu Virtual-Reality-Anwendungen – vor Beginn der Beleuchtungssimulation bekannt sind oder berechnet werden können.

Bei der physikalischen Simulation von Bewegungen und Verformungen von Objekten müssen zum einen die Differentialgleichungen aufgestellt und gelöst werden, die diese Bewegungen und Verformungen beschreiben, zum anderen müssen Kollisionen detektiert und behandelt werden. Bei der Detektion von Kollisionen zwischen Objekten sind ähnliche Probleme zu lösen, wie bei der Sichtbarkeitsdetektion in der Beleuchtungssimulation. Zur Detektion von Kollisionen zwischen Objekten kann deshalb auf Datenstrukturen und Algorithmen aus der Beleuchtungssimulation zurückgegriffen werden. Im Rahmen dieser Arbeit wurden zwei neue Algorithmen zur Kollisionsdetektion entwickelt [1, 2], die in den Abschnitten 2.1.2.2.1 und 2.1.2.2.2 beschrieben werden.

Das erste Verfahren zur Beleuchtungsberechnung, das auch globale Beleuchtungseffekte berücksichtigte, war das Strahlverfolgungs-Verfahren (im folgenden wird dafür der auch im deutschen Sprachraum geläufige englische Begriff *Raytracing* verwendet). Unter globalen Beleuchtungseffekten versteht man Effekte, die sich aus Interaktionen des Lichts mit mehreren Oberflächen der Szene ergeben, wie zum Beispiel Schatten oder Reflektionen. Die Idee des Raytracing-Verfahrens besteht darin, Strahlen auf ihrem Weg von den Lichtquellen durch die Szene zu verfolgen. Schnittpunktsberechnungen der Strahlen mit den Objekten der Szene liefern die Positionen, an denen die Strahlen auf Objekte treffen, um von dort wieder reflektiert oder transmittiert zu werden. Dabei kann aber nur das Verhalten an ideal spiegelnden oder transparenten Oberflächen korrekt wiedergegeben werden.

Das Raytracing-Verfahren liefert bereits sehr realistische Bilder, die einen guten räumlichen Eindruck vermitteln, liefert aber keine physikalischen Beleuchtungsergebnisse. Da in natürlichen Umgebungen, insbesondere in Innenraumszenen, in der Regel die diffuse Reflektion vorherrschend ist, können hier die Beleuchtungsverhältnisse mit Hilfe des Raytracing-Verfahrens nur unzureichend wiedergegeben werden.

Das Radiosity-Verfahren [3, 4, 5] ist ein effizientes Verfahren zur Simulation diffuser Beleuchtung. Es liefert sehr realistische Bilder und für diffuse Szenen physikalisch korrekte Ergebnisse. Das Verfahren beruht darauf, ein Gleichgewicht der von den Oberflächen der Szene ausgetauschten Strahlungsleistung zu berechnen.

Das Radiosity-Verfahren hat in den letzten Jahren in vielen Anwendungen Einzug gehalten. Dies ist aber meist auf Anwendungen in statischen Szenen beschränkt, da sich das Strahlungsgleichgewicht durch Veränderungen der Szene ändert und daher der Rechenaufwand für dynamische Szenen sehr hoch ist. Durch Anwendung von Verfahren, die zeitliche und räumliche Kohärenzen ausnutzen, kann jedoch ein großer Teil dieses Rechnenaufwandes eingespart werden, so daß das Radiosity-Verfahren auch für dynamische Szenen einsetzbar wird.

Im Rahmen dieser Arbeit wurde ein Verfahren zur effizienten Berechnung von Radiosity-Lösungen für Video-Animationen entwickelt [6, 7], das konsistente Beleuchtungsverhältnisse während der gesamten Animation garantiert und dadurch

flackerfreie Animationen erzeugt. Dieses Verfahren ist in Kapitel 4 ausführlich beschrieben. Die meisten anderen Verfahren leiden bei deren Anwendung zur Berechnung von Video-Animationen unter flackernden Ergebnissen. In Kapitel 5 werden einige mit Hilfe dieses Verfahrens erhaltene Ergebnisse präsentiert.

Kapitel 2

Physikalische Modellierung

Die physikalische Modellierung beschäftigt sich damit, Bewegungen und Verformungen von Objekten nach physikalischen Gesetzen zu berechnen. Die Anwendungsgebiete sind vielfältig. In vielen Situationen, wie zum Beispiel virtuellen Crash-Tests, ist man am Simulationsergebnis als solchem interessiert. Bei anderen Anwendungen, wie zum Beispiel der Erzeugung synthetischer Filme, benötigt man die physikalische Simulation, um natürliche visuelle Ergebnisse zu erzielen. So ergibt sich die Bewegung und Faltenbildung von Kleidungsstücken aus der Bewegung der Person, die Kleidung trägt (siehe Abbildung 2.1). Selbst bei unbewegten Objekten, wie einem über einem Tisch hängenden Tischtuch oder einem Vorhang, ergeben sich die Falten und damit die Form der Objekte aus dem Zusammenspiel der Schwerkraft und der aus dem Material rührenden inneren Kräfte.

Abbildung 2.1: Form und Lage des Umhangs ergeben sich aus dem Einfluß der Person.

Solche Formen und Bewegungen für die gesamte Animation von Hand zu modellieren wäre sehr arbeitsaufwendig und auch schwierig. Kleine Abweichungen vom

Verhalten realer Objekte sind für jeden Beobachter sofort sichtbar, und die gesamte Animation wird unrealistisch und künstlich empfunden. Deshalb ist es sinnvoll, das Verhalten der virtuellen Objekte unter Berücksichtigung weniger vorgegebener Randbedingungen und Einflüsse nach physikalischen Gesetzen zu berechnen.

Ausgehend von einer initialen Situation mit bekannter Form, Position, Bewegungsrichtung und Geschwindigkeit ergibt sich die weitere Bewegung eines Objektes aus Beschleunigungskräften, die auf dieses Objekt einwirken. Dies können innere Kräfte, wie Dehn-, Biege- oder Scherkräfte, oder auch äußere Kräfte wie die Schwerkraft, Reibungskräfte oder Einflüsse anderer Objekte sein. Diese Kräfte führen auf Differentialgleichungen, die die Bewegungen und Verformungen der Objekte beschreiben. Diese Differentialgleichungen müssen aus den physikalischen Gegebenheiten und Gesetzen hergeleitet und numerisch gelöst werden.

Bei der Simulation starrer Objekte muß die Bewegung der Massezentren und die Rotation der Objekte in Betracht gezogen werden. Eine Kraft, die auf ein starres Objekt einwirkt, kann aufgespalten werden in eine Komponente, die das Massezentrum in eine Richtung beschleunigt, und eine Komponente, die ein Drehmoment auf das Objekt ausübt und in einer Drehbeschleunigung resultiert.

Sollen deformierbare Objekte simuliert werden, muß eine Diskretisierung dieser Objekte vorgenommen werden, da eine analytische Simulation aufgrund der Komplexität der zu lösenden Probleme nicht möglich ist. Dafür muß man sich zwischen zwei unterschiedlichen Ansätzen entscheiden, dem *Finite-Elemente*-Ansatz [8, 9, 10], bei dem die Oberflächen in kleine kontinuierliche Teilflächen (oder die Volumina in kleine kontinuierliche Teilvolumina) unterteilt werden, oder dem *Partikel*-Ansatz [11, 12], bei dem die Objekte durch endlich viele ausdehnungslose zwei- oder dreidimensional gekoppelte Massepunkte approximiert werden.

Im folgenden wird die physikalische Simulation mit Hilfe von Partikelsystemen behandelt. Es wird eine kurze Einführung in die Lösungungsmethoden zur numerischen Lösung der zugrundeliegenden Differentialgleichungen gegeben. Dann wird die Behandlung von Kollisionen und deren Detektion mit Hilfe zweier im Rahmen dieser Arbeit entwickelter Methoden beschrieben. Und schließlich folgt die Anwendung auf die Simulation von Textilien.

2.1 Partikelsysteme

Partikelsysteme können zur Simulation von deformierbaren Objekten verwendet werden. Die Objekte werden durch eine endliche Anzahl ausdehnungsloser gekoppelter Massepunkte, den sogenannten Partikeln, approximiert (siehe Abbildung 2.2). Da diese Partikel ausdehnungslos sind, müssen keine Drehbewegungen und Drehbeschleunigungen berücksichtigt werden. Die Drehbewegungen der Objekte ergeben sich implizit aus den relativen Bewegungen der einzelnen Partikel.

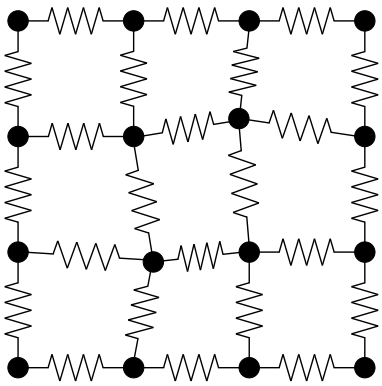


Abbildung 2.2: Die Objekte werden durch gekoppelte Massepunkte approximiert.

Jedes Partikel besitzt zu jedem Zeitpunkt t eine bestimmte Position $x(t)$ und Geschwindigkeit $v(t) = \dot{x}(t)$ und reagiert auf Kräfte, die von anderen Partikeln und von der Umgebung ausgeübt werden.

Dabei gehorchen die Partikel dem Newtonschen Gesetz der Mechanik:

$$F = m\ a.$$

F ist die aus allen inneren und äußeren Einflüssen resultierende Kraft, die auf ein Partikel wirkt, m die Masse des Partikels und a die aus dieser Kraft resultierende Beschleunigung. Mit $\ddot{x}(t) = a(t)$ erhält man eine gewöhnliche Differentialgleichung zweiter Ordnung

$$\ddot{x}(t) \ = \ \frac{1}{m}\ F(t, x(t), \dot{x}(t)),$$

die sich zu dem System

$$\frac{d}{dt}\begin{pmatrix} x \\ v \end{pmatrix} \ = \ \begin{pmatrix} v \\ \frac{1}{m}\ F(t, x, v) \end{pmatrix}$$

erster Ordnung umformen läßt. Dabei sind x und v jeweils Vektoren in drei Koordinaten. Ein aus n Partikeln bestehendes Partikelsystem führt also auf ein Differentialgleichungssystem der Dimension $6n$. Kennt man den Zustand des Systems, das heißt die Positionen und Geschwindigkeiten der einzelnen Partikel, zu einem Zeitpunkt t_0, so ergibt sich der Zustand zu späteren Zeiten durch Lösung des resultierenden Anfangswertproblems.

2.1.1 Lösung der Differentialgleichungen

Wie im vorigen Abschnitt beschrieben, wird das Verhalten eines Partikelsystems durch ein Anfangswertproblem der allgemeinen Form

$$\dot{x}(t) = f(t, x(t)), \qquad x(t_0) = x_0 \tag{2.1}$$

beschrieben. Dieses System ist für ein aus n Partikeln bestehendes Partikelsystem von der Dimension $6n$. Dabei ist $x(t)$ der Zustand des Systems zum Zeitpunkt t. Die Funktion f definiert für jedes t ein $6n$-dimensionales Vektorfeld. Ein Vektor $f(t, x)$ ist die zeitliche Ableitung des Systems, wenn es sich zum Zeitpunkt t im Zustand x befindet, und gibt damit an, mit welcher Geschwindigkeit und in welche Richtung sich das System von diesem Zustand aus weiter bewegt (siehe Abbildung 2.3). Die Trajektorie, auf der sich die Lösung ausgehend von einem Startzustand x_0 durch den Einfluß der Funktion f bewegt, ist eine Integralkurve des Vektorfeldes.

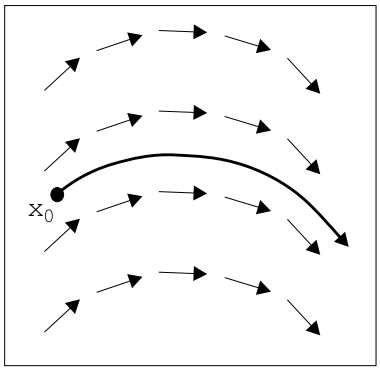


Abbildung 2.3: Die Funktion $f(t, x)$ definiert ein Vektorfeld, entlang dem sich die Lösungen bewegen.

Numerische Verfahren zur Lösung von Differentialgleichungen berechnen in jedem Berechnungsschritt ausgehend von einem Wert X_i, der die Lösung $x(t_i)$ zu einem Zeitpunkt t_i approximiert, in einem diskreten Zeitschritt Δt eine Approximation X_{i+1} der Lösung $x(t_{i+1})$ zum Zeitpunkt $t_{i+1} = t_i + \Delta t$.

Die Genauigkeit der Approximation hängt dabei entscheidend von der Wahl der Schrittweite Δt ab. Diese hat aber auch einen enormen Einfluß auf den Berechnungsaufwand einer Simulation. Es ist deshalb sinnvoll, die Schrittweite nicht für den gesamten Simulationsprozeß fest zu wählen, sondern Verfahren zu verwenden, die die Größe der Schrittweite in jedem Iterationsschritt abhängig vom lokalen Verhalten der Funktion f automatisch bestimmen. Dies geschieht meist durch einen Vergleich mit einem Verfahren anderer Ordnung.

Da die Differentialgleichungen, die bei der Berechnung von Partikelsystemen auftreten, steif sind (siehe [13]), müssen die verwendeten numerischen Methoden sorgfältig ausgewählt werden, um stabile Lösungen und effiziente Schrittweiten zu erhalten (siehe dazu auch [14]).

2.1.1.1 Das Euler-Verfahren

Das einfachste Verfahren zur numerischen Lösung von Differentialgleichungen ist das Euler-Verfahren, das sich leicht aus dem Satz von Taylor herleiten läßt. Die Taylorentwicklung der gesuchten Funktion $x(t)$ um t_i liefert

$$x(t_i + \Delta t) = x(t_i) + \Delta t \ \dot{x}(t_i) + O(\Delta t^2) \ .$$

Unter Vernachlässigung des quadratischen Restgliedes erhält man das Euler-Verfahren

$$\begin{aligned} X_0 &= x_0 \\ X_{i+1} &= X_i + \Delta t \ f(t_i, X_i) \ . \end{aligned}$$

Das heißt, ausgehend von einer gegebenen Näherungslösung X_i bewegt man sich einen Schritt in die Richtung $f(t_i, X_i)$.

Diese Methode ist zwar sehr einfach, dafür aber unglücklicherweise nicht sehr genau und in vielen Fällen instabil. Abbildung 2.4 zeigt zwei Beispiele, die diese Probleme veranschaulichen. In Bild a) bilden die Integralkurven einer zeitunabhängigen, zweidimensionalen Funktion f konzentrische Kreise. Die approximierten Lösungen bewegen sich aber nicht auf diesen Kreisen, sondern entfernen sich spiralförmig vom Zentrum. In Bild b) ist die eindimensionale Funktion $f(t, x) = -kx$ veranschaulicht. Hier gehen die exakten Lösungen für $t \to \infty$ gegen 0, während die approximierte Lösung explodiert.

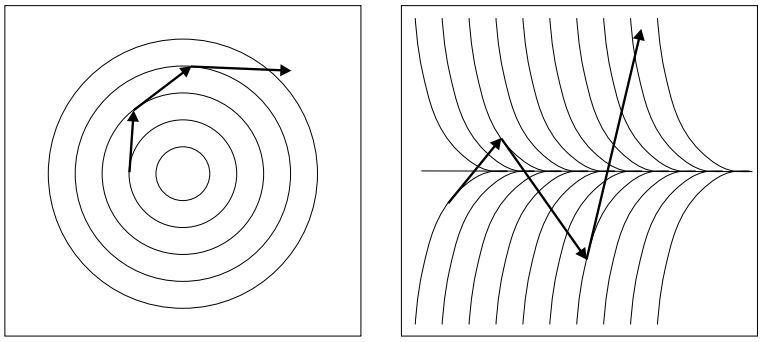


Abbildung 2.4: Ungenauigkeit und Instabilität der Euler-Methode.

Die Euler-Methode ist für die Anwendung auf steife Differentialgleichungen, wie sie bei der Simulation von Partikelmodellen auftreten, nicht geeignet. Hier müssen modernere Methoden angewendet werden, um mit größeren Zeitschritten stabile Lösungen zu erhalten.

2.1.1.2 Andere Lösungsverfahren

Weitere Lösungsverfahren zur numerischen Lösung von Differentialgleichungen erhält man zum Beispiel, wenn man die Differentialgleichung in (2.1) von t_0 bis t integriert und auf das dadurch entstandene Integral eine geeignete Quadraturformel anwendet. Durch Integration der Differentialgleichung erhält man

$$x(t_{i+1}) \;=\; x(t_i) \;+\; \int_{t_i}^{t_{i+1}} f(s, x(s))\; ds \;.$$

Approximiert man das Integral auf der rechten Seite der Gleichung durch $f(t_i, x(t_i))$, erhält man das im vorigen Abschnitt beschriebene explizite Euler-Verfahren. Approximiert man es stattdessen durch $f(t_{i+1}, x(t_{i+1}))$, erhält man das implizite Eulerverfahren

$$\begin{aligned} X_0 &= x_0 \\ X_{i+1} &= X_i + \Delta t\; f(t_{i+1}, X_{i+1}) \;, \end{aligned} \tag{2.2}$$

das im Gegensatz zum expliziten Eulerverfahren zur Lösung steifer Differentialgleichungen geeignet ist.

Mit diesem Verfahren läßt sich aber eine Approximation X_{i+1} der Lösung zum Zeitpunkt t_{i+1} nicht mehr direkt aus der Approximation des vorigen Berechnungsschrittes X_i bestimmen, sondern ergibt sich aus der Lösung der im Allgemeinen nichtlinearen Gleichung (2.2).

Lösungsverfahren, die die Approximation X_{i+1} nicht explizit sondern in Form einer Gleichung liefern, nennt man implizite Verfahren. Generell sind solche impliziten Verfahren zur Lösung steifer Differentialgleichungen besser geeignet als explizite.

Bei der Verwendung impliziter Verfahren wird man mit der zusätzlichen Schwierigkeit konfrontiert, zur Bestimmung einer approximierten Lösung X_{i+1} ein nichtlineares Gleichungssystem lösen zu müssen. Dies wird mit Hilfe der Newton-Methode durchgeführt, die auf ein System linearer Gleichungen führt. Dieses lineare Gleichungssystem kann mit einem geeigneten Eliminationsverfahren oder im Fall schwacher Besetzung durch ein iteratives Krylow-Verfahren gelöst werden [15].

Moderne implizite Verfahren, wie zum Beispiel implizite Runge-Kutta-Verfahren, Adams-Verfahren, Rückwärts-Differentiations-Formeln (englisch: backward differentiation formulas BDF), oder auch Exponential-Integratoren liefern gute Ergebnisse in der Partikel-Simulation (siehe dazu z.B. [16, 14]).

2.1.2 Kollisionen

Ein wichtiges Problem bei der physikalischen Modellierung ist die Erkennung und Behandlung von Kollisionen der simulierten Objekte untereinander und mit ihrer Umgebung.

Dazu muß überprüft werden, ob die Trajektorie eines Partikels auf die Oberfläche eines anderen Objektes oder auch auf eine andere Stelle des eigenen Objektes trifft. Wird solch eine Kollision festgestellt, muß entsprechend reagiert werden, um das Kollisionsverhalten korrekt wiederzugeben. Diese Reaktion hängt von der Orientierung der getroffenen Fläche und auch von Materialeigenschaften der kollidierenden Objekte ab.

In vielen Fällen kann der Einfluß eines simulierten Objektes auf das Objekt, mit dem es kollidiert, vernachlässigt werden, da die Masse des simulierten Objektes viel zu klein ist, um einen wahrnehmbaren Einfluß auszuüben. Ein Tischtuch, das auf einen Tisch fällt, hat keine Auswirkung auf die Position des Tisches. Auch Kleidungsstücke, die von einer Person getragen werden, beeinflussen die Bewegungen der Person nicht wesentlich und können deshalb vernachlässigt werden. Hier genügt es, die Auswirkungen von Kollisionen auf die simulierten Objekte zu berücksichtigen. In anderen Fällen, zum Beispiel, wenn zwei Falten eines Kleidungsstückes aneinander stoßen, beeinflussen sich die kollidierten Partikel gegenseitig.

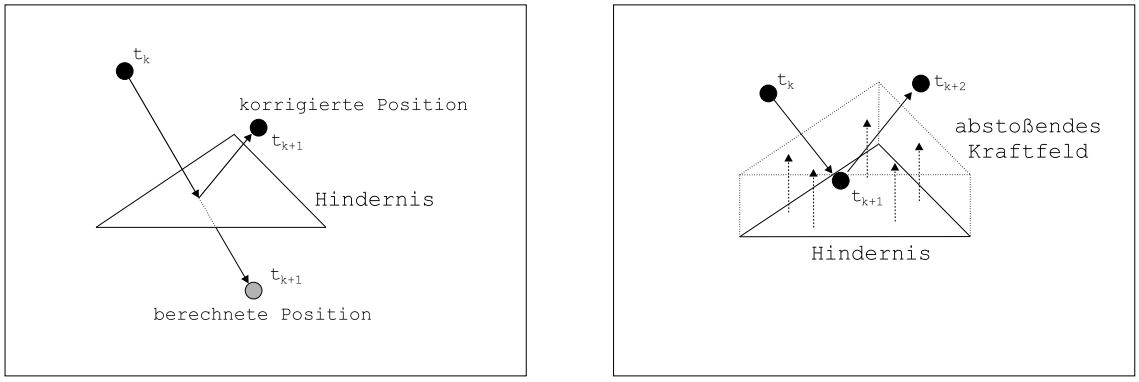
Die numerische Lösung des Differentialgleichungssystems liefert in jedem Iterationsschritt, ausgehend von einem Zustand des Systems zu einem Zeitpunkt t_k, den neuen Zustand des Systems zu einem späteren Zeitpunkt t_{k+1}, nicht jedoch die Trajektorien, auf denen sich die Partikel zwischen den Zeitpunkten t_k und t_{k+1} bewegen. Da die auftretenden Zeitschritte in der Regel sehr klein sind (andernfalls können sie durch Vorgabe einer oberen Schranke begrenzt werden), kann für die Kollisionsdetektion angenommen werden, daß die Trajektorien innerhalb eines Zeitschrittes linear sind.

2.1.2.1 Kollisionsreaktion

Wurde während der Partikelsimulation eine Kollision festgestellt und Kollisionsposition und -zeitpunkt berechnet, muß entsprechend reagiert werden, um ein Durchdringen der Objekte zu verhindern und das Kollisionsverhalten korrekt wiederzugeben.

Dafür sind zwei verschiedene Vorgehensweisen möglich, die in Abbildung 2.5 veranschaulicht werden. Zum einen kann der Zustand, das heißt Position und Geschwindigkeit, eines kollidierten Partikels in dem Zeitschritt, in dem das Partikel eine Oberfläche durchdringen würde, entsprechend der eingetretenen Kollision neu berechnet und abgeändert werden (Repositionierung).

Zum andern können die Objekte durch ein Kraftfeld umgeben werden, das die Partikel bei Annäherung wieder abstößt. Tritt ein Partikel in einem Zeitschritt in ein solches eine Oberfläche umgebendes Kraftfeld ein, bewirkt die daraus resultierende Kraft, daß sich das Partikel im darauffolgenden Zeitschritt der Oberfläche nicht



a) b)

Abbildung 2.5: Kollisionsreaktion a) durch Repositionierung und b) durch ein abstoßendes Kraftfeld.

weiter nähern kann oder sich wieder von dieser entfernt. Die Zeitschritte müssen dabei so beschränkt werden, daß sich ein Partikel nicht in einem einzigen Zeitschritt durch das gesamte Kraftfeld hindurch bewegen kann.

Bei der Kollisionsreaktion durch Repositionierung lassen sich die Kollisionsbedingungen exakt modellieren. Physikalische Vorgänge wie elastische oder unelastische Stöße können exakt berechnet werden. Diese Vorgehensweise verursacht aber Diskontinuitäten, die zu Stabilitätsproblemen in den nachfolgenden Schritten bei der Lösung der Differentialgleichungen führen und deshalb in einer erheblichen Reduzierung der Schrittweiten und somit auch Verlängerung der Rechenzeiten resultieren.

Solche Stabilitätsprobleme sind bei der Anwendung von Kraftfeldern wesentlich geringer. Hier lassen sich aber die Kollisionsbedingungen nur ungenau modellieren.

2.1.2.2 Kollisionsdetektion

Das Problem der Kollisionsdetektion hat in den letzten Jahren einige Forschungsaktivitäten auf sich gezogen. Eine Übersicht über die verschiedenen Methoden ist in [17] zu finden. Für konvexe Polyeder wurden Algorithmen entwickelt, die in linearer Zeit arbeiten [18, 19]. Andere Methoden ohne die Beschränkung auf konvexe Polyeder arbeiten mit verschiedenen Ansätzen auf der Basis hierarchischer Boundingvolumes: mit orientierten Bounding-Boxen, die in OBB-Bäumen benutzt werden [20], mit kugelförmigen Bewegungsvolumen [21] oder mit diskret orientierten Polytopen [22].

Mit diesen Ansätzen sind mit aktueller Hardware interaktive Antwortzeiten für Modelle mit mehreren tausend Flächen möglich. Zur effizienten Berechnung von Simulationen, die mit Hilfe numerischer Lösungen von Differentialgleichungen durchgeführt werden, sind aber noch schnellere Algorithmen notwendig.

In den folgenden Abschnitten werden effiziente Verfahren zur Kollisionsdetektion beschrieben, die im Rahmen dieser Arbeit entwickelt wurden.

Das erste Verfahren [1] verwendet Raytracing und kann sowohl zur Detektion von Kollisionen mit statischen und bewegten Objekten der Umgebung als auch zur Detektion von Eigenkollisionen verwendet werden.

Das zweite Verfahren [2] berechnet eine implizite Darstellung aller statischen und bewegten Objekte der Umgebung, die dann zur schnellen Abstandsberechnung beliebiger Punkte im Raum von den Objektoberflächen herangezogen werden kann.

2.1.2.2.1 Kollisionsdetektion durch Raytracing

Bei der Kollisionsdetektion muß festgestellt werden, ob ein Partikel auf seinem Weg von der Position $x(t_k)$ zum Zeitpunkt t_k zur neu berechneten Position $x(t_{k+1})$ zum Zeitpunkt t_{k+1} auf ein anderes Objekt trifft. Raytracing ist eine wohlbekannte Technik, um Oberflächen zu bestimmen, die von der Verbindungsstrecke zweier Punkte geschnitten werden, und um diese Schnittpunkte zu berechnen.

Dazu wird ein Strahl von der Position $x(t_k)$ bis zur neu berechneten Position $x(t_{k+1})$ verfolgt und auf Schnittpunkte mit den Objekten der Szene getestet (siehe Abbildung 2.6).

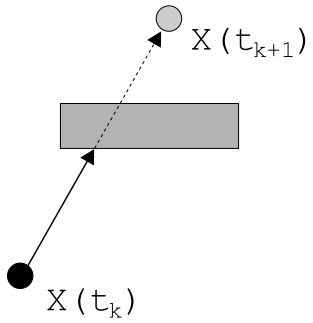


Abbildung 2.6: Kollisionsdetektion mit Raytracing.

Für Raytracing sind effiziente Algorithmen und Beschleunigungstechniken verfügbar, die dann gleichzeitig auch zur Beleuchtungssimulation in der Szene verwendet werden können.

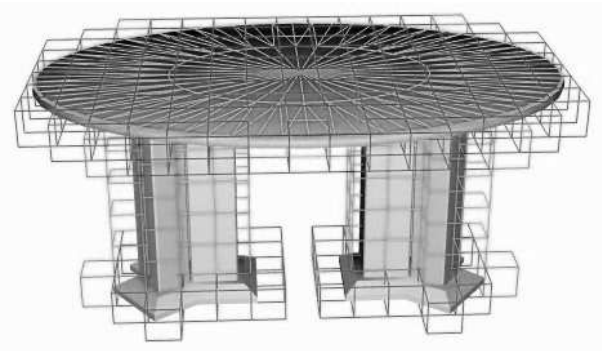


Abbildung 2.7: Tisch mit regulärem Raumgitter.

Zur Beschleunigung der Raytracing-Berechnung werden raumunterteilende Strukturen, wie hierarchische Boundingvolumes, reguläre Raumgitter, Oktrees oder BSP-Trees (binary space partitioning) verwendet (siehe Abbildung 2.7), in die die Objekte der Szene einsortiert werden und die dann bei der Schnittpunktsberechnung von den Strahlen traversiert werden. Diese Strukturen sind für die Kollisionsdetektion sehr effizient, da hier nur sehr kurze Strahlen verwendet werden, die aus der Bewegung eines Partikels während eines kleinen Zeitschrittes resultieren (Bewegungsstrahlen).

So liegen zum Beispiel Anfangs- und Endpunkte von Strahlen bei der Verwendung von regulären Raumgittern in den meisten Fällen in der selben Gitterzelle oder zumindest in benachbarten Zellen. Wird die Gitterauflösung abhängig von der Komplexität der Szene gewählt, ist der Aufwand zur Verfolgung eines Bewegungsstrahls konstant (das heißt unabhängig von der Komplexität der Szene). Der Speicherbedarf zur Speicherung des Gitters und der Aufwand zur Einsortierung der Szenenobjekte, was für Umgebungsobjekte nicht während der Simulation sondern in einem Vorbearbeitungsschritt geschieht, ist proportional zur Komplexität der Szene.

Kollisionen mit unbewegten Objekten

Zur Detektion von Kollisionen mit festen, unbewegten Objekten kann einfach konventionelles Raytracing verwendet werden. Die Kollisionspunkte sind schon durch die Schnittpunkte der Bewegungsstrahlen mit den Objekten gegeben. Der Kollisionszeitpunkt läßt sich aufgrund der Annahme einer linearen Bewegung der Partikel innerhalb eines Zeitschrittes leicht aus den Anfangs-, End- und Kollisionspositionen der Partikel berechnen.

Abbildung 2.8 zeigt die Kollision eines mit Hilfe eines Partikelmodells simulierten Tischtuches mit einem feststehenden Tisch.

Abbildung 2.8: Kollision mit einem unbewegten Objekt.

Kollisionen mit bewegten Objekten

In praktischen Situationen ist es oft nicht ausreichend, nur auf Kollisionen mit festen Objekten der Umgebung zu reagieren. Es müssen auch Kollisionen mit bewegten Objekten sowie Eigenkollisionen behandelt werden, wie sie zum Beispiel in dem in Abbildung 2.9 dargestellten Beispiel auftreten. Für den Kollisionstest eines Bewegungsstrahls mit bewegten Objekten ist konventionelles Raytracing nicht mehr ausreichend. Die Berechnungen müssen unter Berücksichtigung der Zeit in 4D durchgeführt werden. Damit lassen sich dann auch Schnittpunkte mit bewegten Objekten schnell und effizient berechnen.

Beim Eintrag in raumunterteilende Strukturen muß ein bewegtes Objekt nun in alle Zellen eingetragen werden, in denen sich das Objekt irgendwann zwischen den Zeitpunkten t_k und t_{k+1} befindet. Die Schnittpunktsberechnungen müssen nun vierdimensional unter Berücksichtigung der Zeit durchgeführt werden und liefern dann als Ergebnis sowohl die Kollisionsposition als auch den Kollisionszeitpunkt.

Zur Berechnung von Eigenkollisionen wird für das eigene Objekt ein separates Gitter verwendet, das nach jedem Berechnungsschritt neu aufgebaut werden muß. Der Aufwand, der dafür nötig ist, ist proportional zur Komplexität des simulierten Objektes.

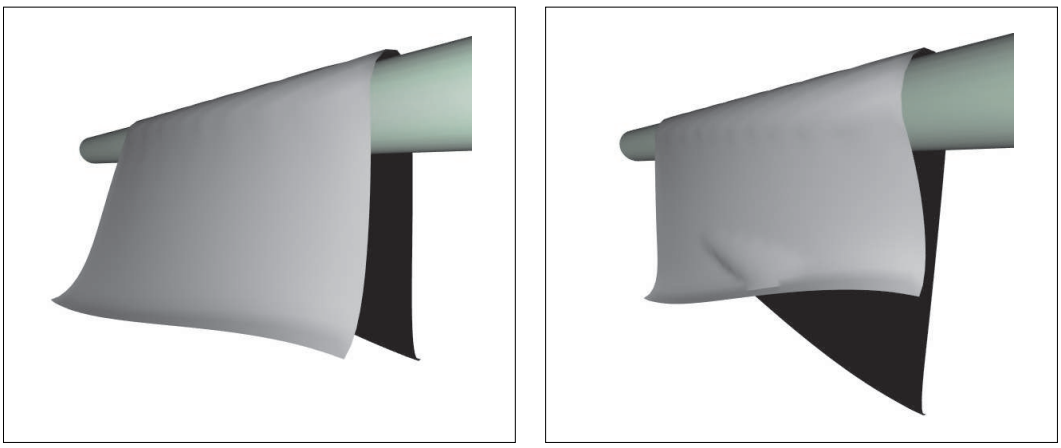


Abbildung 2.9: Eigenkollision eines über eine Stange fallenden Tuches.

2.1.2.2.2 Kollisionsdetektion mit impliziten Flächen

Zur Detektion von Kollisionen eines Partikels mit anderen Objekten ist es nicht zwingend erforderlich, explizit auf Schnittpunkte mit diesen Objekten zu testen. Zur Erkennung einer Kollision genügt es, festzustellen, ob sich die Punkte $x(t_k)$ und $x(t_{k+1})$ innerhalb oder außerhalb von Objekten der Szene befinden. Lassen sich im Fall einer Kollision zusätzlich noch die Abstände der beiden Punkte von der Objektoberfläche bestimmen, kann daraus auch leicht die Kollisionsposition errechnet werden. Zur Kollisionsreaktion wird dann noch die Normale im Kollisionspunkt benötigt. Entscheidet man sich zur Kollisionsreaktion durch ein Kraftfeld, ist es

sogar ausreichend, den Abstand eines Punktes $x(t_k)$ von der nächstliegenden Objektoberfläche und die Normale der Oberfläche an dieser Stelle zu kennen, um die resultierende Kraft berechnen zu können.

Dies kann mit Hilfe von impliziten Darstellungen der Objekte erreicht werden, mit denen auf Kollision getestet werden soll. Es ist sogar möglich, alle Objekte einer beliebig komplexen Umgebung durch eine einzige implizite Funktion darzustellen, die durch eine einzige Funktionsauswertung mit einer geringen Anzahl an Rechenoperationen den Abstand eines beliebigen Punktes im Raum zur nächstgelegenen Oberfläche eines Umgebungsobjektes liefert. Als implizite Funktionen zur Repräsentation der Umgebung verwenden wir skalare trivariate B-Spline Funktionen.

Damit können sowohl für statische als auch für bewegte Objekte der Umgebung Kollisionstests und -reaktionen extrem schnell und in konstanter Zeit, das heißt unabhängig von der Komplexität der Umgebung, durchgeführt werden, die zudem aufgrund der Stetigkeit der Kraftfelder eine schnelle und stabile Lösung der Differentialgleichungen mit großen Zeitschritten erlauben. Die Flächennormale in einem Oberflächenpunkt ist durch den Gradienten der impliziten Funktion gegeben und kann mit dem selben Aufwand errechnet werden.

Die vorgestellte Methode ist eine Abwandlung der Arbeit von Raviv und Elber [23], die solche trivariate B-Spline Funktionen zur Modellierung von Freiformflächen einsetzen.

Implizite B-Spline Tensorprodukt-Flächen

Eine implizite Fläche im dreidimensioalen Raum ist die Isofläche einer Funktion $\mathcal{F} : \mathbb{R}^3 \to \mathbb{R}$ zu einem gegebenen Isowert w, das heißt die Menge aller Punkte (x, y, z) mit $\mathcal{F}(x, y, z) = w$. Um die Oberflächen der Objekte der Umgebung in einer Partikelsimulation als implizite Fläche darzustellen, muß eine solche Funktion $\mathcal{F}$ bestimmt werden, so daß die Oberflächen durch die Menge $\{(x, y, z) : \mathcal{F}(x, y, z) = w\}$ für ein geeignetes w hinreichend genau approximiert werden. Dazu verwenden wir trivariate B-Spline Tensorprodukte eines geeigneten Grades d:

$$\mathcal{F}(x, y, z) = \sum_{i=0}^{l-1} \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} b_{ijk} B_i(x) B_j(y) B_k(z),$$

wobei $B_i(x), B_j(y)$ und $B_k(z)$ die normalisierten B-Spline Basisfunktionen und die b_{ijk} die zugehörigen skalarwertigen deBoor-Punkte sind. Der Grad d der B-Spline Funktionen kann beliebig gewählt werden, $d = 1, 2$ oder 3 ist für diese Anwendung aber völlig ausreichend und erlaubt eine schnelle Auswertung der Funktionswerte und Gradienten.

Eine solche trivariate B-Spline Funktion ist auf einem quaderförmigen Parametergebiet

$$D = [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$$

definiert, das alle zu behandelnden Objekte enthält. Die Idee für die Anwendung in der Kollisionsdetektion ist nun, die Knotenpunkte innerhalb des Parametergebietes durch eine gleichmäßige Unterteilung

$$\begin{aligned} x_{min} = x_0 < x1 < \ldots < x_{l-1} = x_{max} \\ y_{min} = y_0 < y1 < \ldots < y_{m-1} = y_{max} \\ z_{min} = z_0 < z1 < \ldots < z_{n-1} = z_{max} \end{aligned}$$

mit

$$\begin{aligned} x_{i+1} - x_i = \Delta x \quad \text{konstant für alle} \quad i = 0, \ldots, l-2 \\ y_{j+1} - y_j = \Delta y \quad \text{konstant für alle} \quad j = 0, \ldots, m-2 \\ z_{k+1} - z_k = \Delta z \quad \text{konstant für alle} \quad k = 0, \ldots, n-2 \end{aligned}$$

zu wählen. Das heißt, das Parametergebiet D, das alle Umgebungsobjekte enthält, wird mit einem gleichmäßigen Gitter von Knotenpunkten überzogen. Dadurch kann bei der Funktionsauswertung an einem beliebigen Punkt (x, y, z) direkt auf die benötigten deBoor-Punkte zugegriffen werden, was eine extrem schnelle Auswertung erlaubt.

Nun müssen die deBoor-Punkte b_{ijk} für $i = 0, \ldots l-1, j = 0, \ldots m-1$ und $k = 0, \ldots n-1$ bestimmt werden, so daß die Oberflächen der Umgebung durch die Menge $\{(x, y, z) \ : \ \mathcal{F}(x, y, z) = w\}$ für ein geeignetes w repräsentiert werden. Im Fall linearer B-Spline Funktionen ist dies gegeben, wenn $w = 0$ gewählt wird und die deBoor-Punkte b_{ijk} auf die kürzeste Distanz des Gitterpunktes (x_i, y_j, z_k) zu einem Umgebungsobjekt gesetzt werden. Dabei werden an Gitterpunkten, die sich außerhalb der Objekte befinden, positive Entfernungswerte gesetzt, und an Gitterpunkten, die sich innerhalb von Objekten befinden, negative Entfernungswerte (siehe Abbildung 2.10). Die Auswertung der Funktion $\mathcal{F}$ an einem Punkt (x, y, z) liefert dann den vorzeichenbehafteten kürzesten Abstand dieses Punktes zu einem Oberflächenpunkt der Umgebung.

Auf diese Weise lassen sich also extrem schnell die Abstände beliebiger Punkte zu den Umgebungsobjekten, sowie die Oberflächennormalen an den nächstliegenden Oberflächenpunkten bestimmen, was während der Simulationsberechnung sehr häufig benötigt wird. Die Bestimmung der deBoor-Punkte b_{ijk} geschieht einmalig in einem Vorbearbeitungsschritt, in dem die Umgebungsobjekte in das Gitter eingetragen werden. Dazu werden für jedes Umgebungsobjekt die kürzesten Entfernungen zu den Gitterpunkten berechnet, die sich in einer näheren Umgebung des Objektes befinden, und die deBoor-Werte an den Gitterpunkten auf das Minimum aller zu dem jeweiligen Punkt berechneten Entfernungen gesetzt.

Die Verwendung von B-Spline Funktionen höherer Ordnung führt zu glatteren Repräsentationen der Oberflächen. Bei der Bestimmung der deBoor-Punkte müssen hier Pseudo-Entfernungen verwendet werden.

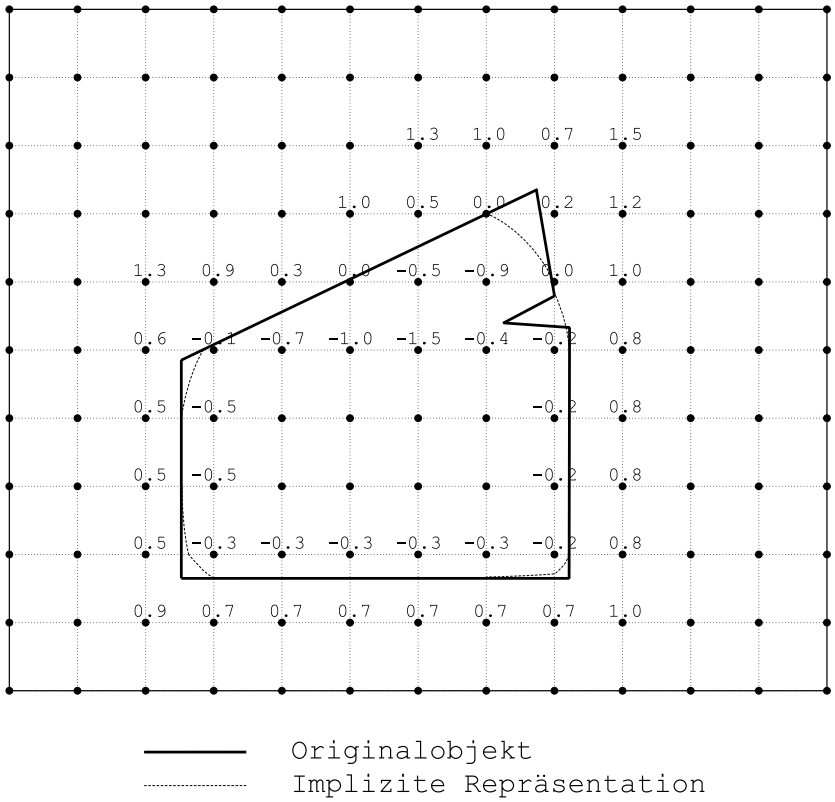


Abbildung 2.10: Repräsentation des Objektes durch eine implizite B-Spline-Tensorprodukt-Fläche vom Grad eins mit gleichmäßiger Unterteilung des Parametergebiets. Die Zahlen sind die deBoor-Werte an den Gitterpunkten und geben den Abstand zur Objektoberfläche an.

Abbildung 2.11 zeigt ein Objekt und dessen Repräsentation durch eine implizite B-Spline-Tensorproduktfläche.

Auswertung der impliziten Funktion

Um festzustellen, ob sich ein Punkt (x, y, z) innerhalb oder außerhalb von Objekten der Umgebung befindet oder um den Abstand eines Punktes zu den Umgebungsobjekten zu bestimmen, muß lediglich der Funktionswert $\mathcal{F}(x, y, z)$ der impliziten B-Spline-Tensorprodukt-Funktion berechnet werden. Dazu werden die deBoor-Punkte an den Gitterpunkten des Trägers der Funktion benötigt. Bei linearen B-Spline-Funktionen sind dies 8, bei quadratischen 27 und bei kubischen 64 Punkte. Die Indizes der beteiligten deBoor-Punkte können aufgrund der gleichmäßigen Unterteilung sehr schnell und in konstanter Zeit bestimmt werden. Die Berechnung des Funktionswertes kann mit Hilfe des deBoor-Algorithmus mit wenigen Rechenoperationen aus den deBoor-Punkten durchgeführt werden.

Die Normale der Objektoberfläche an einem Punkt (x, y, z) ist durch den Gradienten $\nabla\mathcal{F}(x, y, z)$ der impliziten Funktion an der Stelle (x, y, z) gegeben und kann auch mit Hilfe des deBoor-Algorithmus mit dem selben Aufwand bestimmt werden.

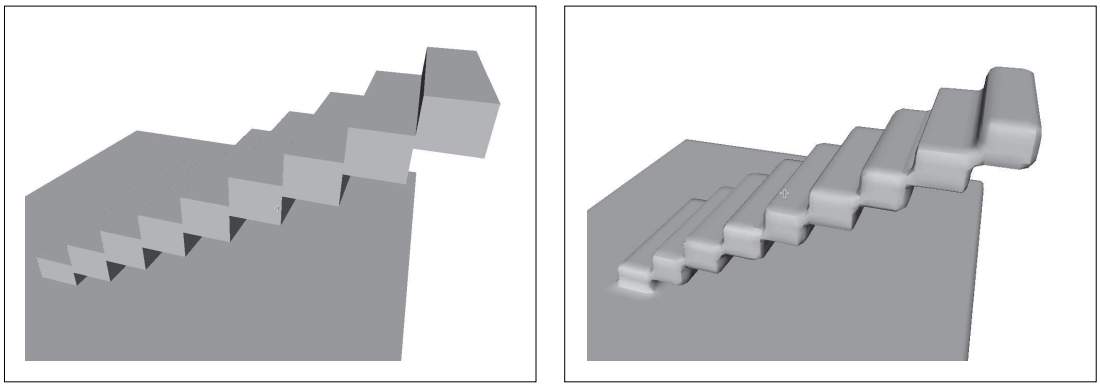


Abbildung 2.11: Originalobjekt (links) und dessen Repräsentation durch eine implizite lineare B-Spline-Tensorproduktfläche (rechts).

Führt man Kollisionsreaktionen mit Hilfe eines Kraftfeldes aus, ist es nicht notwendig, die exakte Kollisionsposition zu kennen. Es genügt, den Abstand eines Partikels von den Oberflächen der Umgebungsobjekte und bei festgestelltem Eintritt in das Kraftfeld die Normale am nächstliegenden Oberflächenpunkt zu bestimmen. Ist $P_1(x, y, z)$ ein nahe bei einer Oberfläche liegender Punkt und $P_2(x', y', z')$ der zu P_1 nächstliegende Oberflächenpunkt, so ist $\nabla\mathcal{F}(P_1) \approx \nabla\mathcal{F}(P_2)$. Die Oberflächennormale kann also – ohne den nächstliegenden Oberflächenpunkt P_2 explizit zu berechnen – direkt durch Auswertung des Gradienten an der Partikelposition P_1 bestimmt werden, wozu die selben deBoor-Punkte benötigt werden, die zuvor zur Abstandsberechnung benutzt wurden. Ist man trotzdem an der Position P_2 des nächstliegenden Oberflächenpunktes interessiert, läßt sich dieser durch $P_2 \approx P_1 - \mathcal{F}(P_1)\ \nabla\mathcal{F}(P_1)\ /\ \|\nabla\mathcal{F}(P_1)\|$ bestimmen.

Führt man Kollisionsreaktionen durch Repositionierung durch, läßt sich die Kollision eines Partikels mit einem Umgebungsobjekt im k-ten Zeitschritt dadurch erkennen, daß der Funktionswert $\mathcal{F}(x(t_k))$ an der neu berechneten Partikelposition $x(t_k)$ negativ und der Funktionswert $\mathcal{F}(x(t_{k-1}))$ an der vorherigen Position $x(t_{k-1})$ positiv ist. Der Kollisionspunkt kann dann mit Hilfe der Abstände $\mathcal{F}(x(t_{k-1}))$ und $\mathcal{F}(x(t_k))$ der beiden Positionen von der Objektoberfläche auf der Verbindungsstrecke zwischen $x(t_{k-1})$ und $x(t_k)$ interpoliert werden. Die Oberflächennormale ergibt sich dann wieder durch Auswertung von $\nabla\mathcal{F}$ an diesem Punkt.

Stabilität der Differentialgleichungen

Die Repräsentation der Umgebungsobjekte durch B-Spline Tensorprodukt-Flächen mit gleichmäßiger Knotenverteilung im Parametergebiet bewirkt, daß die so erhaltene Repräsentation der Umgebung mit einem konstanten Tiefpass-Filter (Boxfilter) geglättet wird und hochfrequente Details unterdrückt werden. Dies ist für die Kollisionsdetektion von Vorteil, da Kollisionen mit hochfrequenten Objekten aufgrund der endlichen Partikel-Auflösung sowieso nicht korrekt behandelt werden können. In den meisten praktischen Situationen wird die reale Situation damit aber trotzdem korrekt wiedergegeben. Bedeckt man zum Beispiel einen Tisch, auf dem sich

kleine Gegenstände befinden mit einem Tischtuch, so paßt sich dieses nicht allen Unebenheiten an, sondern legt sich relativ glatt über die Gegenstände.

Die glatte Repräsentation der Gegenstände erhöht die Stabilität der Differentialgleichungen und beschleunigt deren Lösung enorm, da dadurch mit großen Zeitschritten gerechnet werden kann.

Speichereffizienz durch Hash-Tabellen

Wie bereits weiter oben erwähnt, bewegen sich die Partikel während der Simulation eines Zeitschrittes nur um sehr kurze Distanzen weiter. Deshalb genügt es für die Kollisionsdetektion, wenn Abstände zu den Oberflächen der Objekte nur in einer nahen Umgebung der Objekte bestimmt werden können. An weiter entfernt liegenden Punkten genügt es, zu wissen, daß die Entfernung für eine Kollision noch zu groß ist.

Deshalb müssen nur die deBoor Punkte wirklich gesetzt werden, die für Funktionsauswertungen in einer nahen Umgebung der Objekt-Oberflächen benötigt werden. Dies führt zu einem schwach besetzten Gitter, das mit geeigneten Strukturen, wie zum Beispiel Hash-Tabellen, platzsparend gespeichert werden kann. Da der Zugriff auf die deBoor-Punkte nun über die Hash-Tabellen erfolgt, erhöht sich dadurch der Aufwand zur Bestimmung von Funktionswerten und Gradienten um einen kleinen konstanten Faktor.

Verwendung mehrerer Detail-Stufen

Wie schon weiter oben erwähnt, bewirkt die Approximation durch B-Spline Tensorprodukt-Flächen mit gleichmäßiger Knotenverteilung im Parametergebiet eine Glättung mit konstantem Tiefpassfilter. Um die für die verwendete Partikel-Auflösung des simulierten Objektes notwendige Genauigkeit gewährleisten zu können, muß eine entsprechend feine Gitterauflösung der Knotenpunkte gewählt werden, ansonsten würden eventuell benötigte Details durch die Tiefpassfilterung einfach verschwinden. Oberflächen mit schwacher Krümmung können aber auch mit grober Gitterauflösung gut approximiert werden.

Der zur Auswertung von Funktionswerten oder Gradienten der impliziten Funktion $\mathcal{F}$ benötigte Aufwand ist konstant und wird von der verwendeten Gitterauflösung nicht beeinflußt.

Der zur Speicherung des Gitters der deBoor-Punkte b_{ijk} benötigte Speicherplatz hängt aber von der Gitterauflösung ab. (Auch der zur Bestimmung der deBoor-Punkte benötigte Aufwand hängt von der Gitterauflösung ab. Da dies aber in einem Vorbearbeitungsschritt und nicht während der Simulationsberechnung geschieht, ist eine Optimierung dieses Aufwandes von geringerem Interesse) Da die Gitter nur schwach besetzt sind, ist der Speicherbedarf bei Verwendung geeigneter Speicherstrukturen, wie zum Beispiel Hash-Tabellen, bei fester Gitterauflösung proportional zur Größe der Oberflächen der repräsentierten Objekte.

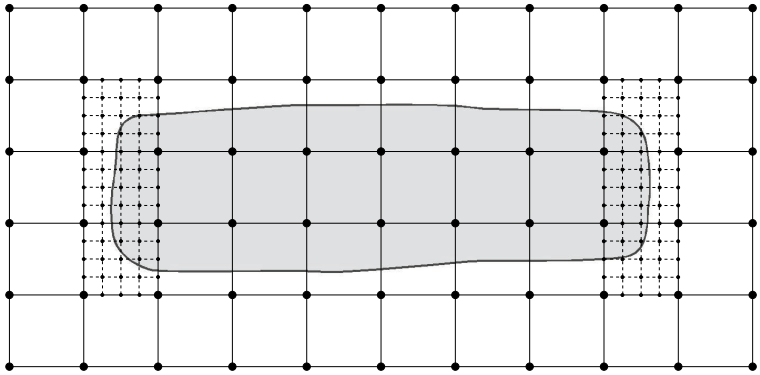


Abbildung 2.12: Verwendung verschiedener Detail-Stufen abhängig von der Oberflächenkrümmung.

Der Speicherbedarf kann deshalb durch die gleichzeitige Verwendung mehrerer Gitter in unterschiedlichen Auflösungsstufen reduziert werden. Oberflächenteile mit starker Krümmung werden dann in ein feineres, Oberflächenteile mit schwacher Krümmung in ein gröberes Gitter eingetragen (siehe Abbildung 2.12).

Dabei muß aber darauf geachtet werden, daß bei der Bestimmung von deBoor-Punkten, die aufgrund eines Oberflächenstückes in einer Auflösungsstufe gesetzt werden, die gesamte Umgebung berücksichtigt wird und nicht nur die Objekte, die in dieser Stufe eingetragen werden. Das heißt, aufgrund der Krümmung der Oberflächenstücke wird bestimmt, welche Punkte in welcher Stufe gesetzt werden müssen, die Werte dieser Punkte werden dann aber unter Berücksichtigung der gesamten Szene bestimmt.

Bei der Auswertung von Funktionswerten oder Gradienten der Funktion $\mathcal{F}$ wird nun die feinste verfügbare Stufe verwendet.

Bewegte Umgebungsobjekte

Der Ansatz der Kollisionsdetektion mit Hilfe impliziter B-Spline-Tensorprodukt-Flächen läßt sich leicht auch auf bewegte Umgebungsobjekte erweitern.

Eine Möglichkeit ist, den dreidimensionalen Raum und die eindimensionale Zeit zu einem vierdimensionalen Raum zusammenzufassen. Statt trivariaten B-Spline-Funktionen werden nun 4-variate B-Spline-Funktionen verwendet. Die Distanzen oder Pseudo-Distanzen werden für vorgegebene Keyframes der Animation in einem vierdimensionalen Gitter eingetragen. Bei der Verwendung linearer B-Spline-Funktionen entspricht dies der Verwendung linearer Interpolation der Objekte zwischen den Keyframes. Werden B-Splines höherer Ordnung verwendet, wird auch die Bewegung der Objekte glatter interpoliert.

In der Praxis liegen animierte Objekte oft in vorgegebenen Keyframes mit linearer Interpolation vor, während die Oberflächen der Objekte zu einem festen Zeitpunkt aber durch B-Spline-Flächen höherer Ordnung repräsentiert werden sollen,

zumal dies mit höherer Stabilität bei der Lösung der Differentialgleichungen verbunden ist.

Aus diesem Grunde verwenden wir auch bei Vorhandensein bewegter Umgebungsobjekte nur trivariate B-Spline-Funktionen und damit ein dreidimensionales Gitter an Knotenpunkten. Diese B-Spline-Funktionen sind nun aber nicht mehr skalarwertig, sondern haben zeitabhängige Funktionen als Werte. Dazu werden zeitabhängige deBoor-Punkte $b_{ijk}(t)$ verwendet.

Oft bewegt sich aber auch in dynamischen Szenen nur ein kleiner Teil der Objekte, während ein Großteil der Objekte statisch ist. Oft sind die Bewegungen einfacher Natur, wie zum Beispiel lineare Bewegungen. In Zeitintervallen, in denen ein statisches Objekt das zu einem Gitterpunkt (x_i, y_j, z_k) nächstliegende Objekt ist, ist die Funktion $b_{ijk}(t)$ konstant. Bei linearen Bewegungen von Objekten treten lineare Funktionen b_{ijk} auf. Deshalb werden die Funktionen $b_{ijk}(t)$ durch stückweise lineare Funktionen approximiert, die somit effizient und platzsparend im Speicher gehalten werden können.

2.1.3 Kleidermodellierung

In diesem Abschnitt wird eine Anwendung der Partikelsimulation auf die Modellierung von Textilien beschrieben. Die in diesem Kapitel dargestellten Ergebnisbilder (Abbildungen 2.1, 2.8 und 2.9) wurden mit der hier vorgestellten Methode berechnet.

In unserem Modell betrachten wir Textilien als anisotrope Materialien, die zwei Hauptrichtungen mit verschiedenen Eigenschaften besitzen. Dies ist durch den Herstellungsprozeß bedingt. Ein gewobener Stoff verhält sich unterschiedlich in Kett- und Schußrichtung, auch Strickwaren verhalten sich in Stäbchenrichtung anders als in Reihenrichtung. Wir approximieren die Textilien deshalb durch ein Rechtecksnetz gekoppelter Partikel. Das heißt, jedes Partikel ist mit maximal vier vordefinierten Nachbarn gekoppelt, mit denen es Kräfte austauschen kann (siehe Abbildung 2.13).

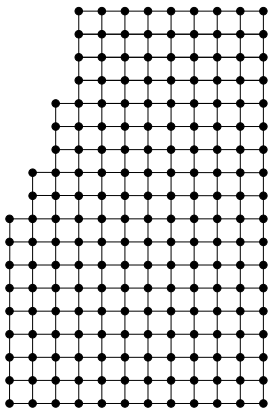


Abbildung 2.13: Textilien werden durch ein Rechtecksnetz gekoppelter Partikel approximiert.

Nun müssen die auf die einzelnen Partikel wirkenden Kräfte bestimmt werden. Dies sind zum einen innere Kräfte, die von den Partikeln auf ihre Nachbarn ausgeübt werden, und zum anderen äußere Kräfte, die durch äußere Einflüsse der Umgebung auf die Partikel wirken.

Die inneren Kräfte der Textilien sind Zugkräfte, die durch Dehnung des Stoffes entstehen, sowie Biege- und Scherkräfte (siehe Abbildung 2.14). Diese Kräfte können je nach Material und Herstellungsart des Stoffes in den beiden Hauptrichtungen unterschiedlich groß sein.

Die Zugkräfte wirken entlang der Verbindungslinie zweier Partikel, die Scherkräfte innerhalb der Tangentialebene senkrecht zur Zugkraft und die Biegekräfte parallel zur Normale der Tangentialebene.

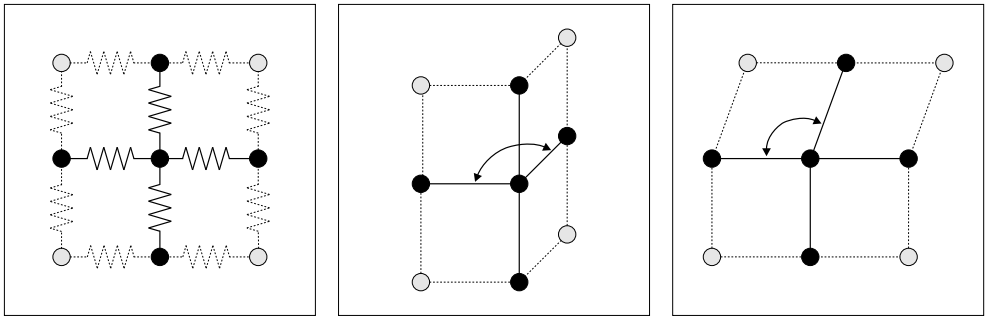


Abbildung 2.14: Innere Kräfte von Textilien: a) Zugkräfte b) Biegekräfte c) Scherkräfte.

Die äußeren Kräfte ergeben sich durch die Gravitation, die Luftreibung, Interaktionen mit der Umgebung, wie zum Beispiel Kollisionen oder von außen angesetzte Zugkräfte, und Interaktionen des Stoffes mit sich selbst (zum Beispiel wenn zwei sich bildende Falten aneinanderstoßen).

2.1.3.1 Kawabata-Experimente

Kawabata [24] führte in enger Zusammenarbeit mit der Textilindustrie umfangreiche Experimente zur Bestimmung der inneren Kräfte in Textilien durch. Dabei wurden auch Hysterese-Effekte berücksichtigt: Stoffe nehmen nach einer Auslenkung nicht mehr ihre ursprüngliche Ruhelage ein, auch die wirkenden Kräfte sind auf dem Rückweg nach erfolgter Auslenkung schwächer als zu Beginn. Die Kräfte hängen also sowohl von der momentanen als auch von der bisherigen maximalen Auslenkung ab.

Die experimentell bestimmten Kräfte werden in Diagrammen festgehalten und für die physikalische Simulation verwendet (siehe Abbildung 2.15).

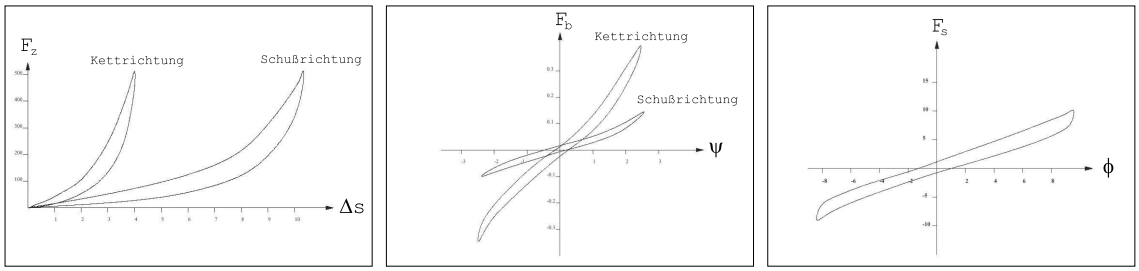


Abbildung 2.15: Kräftediagramme aus den Kawabata-Experimenten: a) Zugkräfte b) Biegekräfte c) Scherkräfte. Die oberen Kurven geben jeweils die Kräfte bei der Auslenkung an, die unteren Kurven die auf dem Rückweg nach erfolgter Auslenkung.

2.1.3.2 Der Lagrange-Ansatz

Im allgemeinen ist es unpraktisch, alle Kräfte, die auf ein Partikel wirken, direkt zu berechnen und aufzuaddieren, da es oft aufwendig ist, deren Richtungen zu bestimmen. Dies gilt insbesondere für die Biege- und Scherkräfte. Diese Kräfte können aber durch Vektorfelder beschrieben werden, die einer Integrabilitätsbedingung genügen, die es erlaubt, die Kräfte durch Potentiale oder Energiefunktionen zu beschreiben. Statt vektorwertiger Kräfte können damit skalarwertige Energiefunktionen betrachtet werden. Die Kräftefunktion kann dann durch den Lagrange-Ansatz beschrieben werden, der zum Beispiel ausführlich in [25] beschrieben wird.

Die Lagrange-Funktion $\mathcal{L}$ für ein Partikel ist gegeben durch

$$\mathcal{L} \;=\; E_{kin} \;-\; V \;,$$

wobei E_{kin} die kinetische Eneregie des Partikels und V die Summe aller durch Potentiale gegebenen Energien ist. Dann beschreibt die Lagrange-Gleichung

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial v_i} \;=\; \frac{\partial \mathcal{L}}{\partial x_i} \;, \quad i = 1, \ldots, 3 \tag{2.3}$$

als Differentialgleichung zweiter Ordnung die Trajektorie, auf der sich das Partikel aufgrund der durch die Potentiale wirkenden Kräfte bewegt. Sind die Potentiale von der Geschwindigkeit unabhängig, ergibt die Lagrange-Gleichung das Newtonsche Gesetz der Mechanik, denn in diesem Fall erhält man mit $E_{kin} = 1/2\ m\|v\|^2$ die Gleichung

$$ma \;=\; \frac{d}{dt}mv \;=\; \left(\frac{\partial V}{\partial x_1}, \frac{\partial V}{\partial x_2}, \frac{\partial V}{\partial x_3}\right)^T \;=\; \nabla V \;. \tag{2.4}$$

Der Gradient ∇V des Energiefeldes V ist aber gerade die auf das Partikel wirkende Kraft.

Mit diesem Ansatz lassen sich die Gravitationsenergie, sowie die inneren Zug-, Biege- und Scherenergien leicht modellieren, deren Potentiale von der Geschwindigkeit unabhängig sind. Die Kräfte, die sich aus der Luftreibung und den Interaktionen des Partikelsystems mit der Umgebung und mit sich selbst ergeben, werden dann nach der Differentiation der Lagrange-Funktion als Kraftvektoren dazuaddiert.

Die Lagrange-Funktion für ein gesamtes aus n Partikeln bestehendes Partikelsystem ist dann gegeben durch

$$\mathcal{L} = \sum_{i=0}^{n-1} E_{kin_i} - \left(\sum_{i=0}^{n-1} \left(E_{grav_i} + E_{z_i} + E_{b_i} + E_{s_i} \right) \right) .$$

Dabei sind E_{kin_i}, E_{grav_i}, E_{z_i}, E_{b_i} und E_{s_i} die kinetische, Gravitations-, Zug-, Biege- und Scherenergien des i-ten Partikels.

2.1.3.2.1 Energien

Die kinetische Energie E_{kin_i} und die Gravitationsenergie E_{grav_i} sind gegeben durch

$$\begin{aligned} E_{kin_i} &= \frac{1}{2} m_i \, \|v_i\|^2 , \\ E_{grav_i} &= \int_0^{z_i} g \, m_i \, dz = m_i \, g \, z_i \end{aligned}$$

wobei m_i die Masse, v_i die Geschwindigkeit und z_i die z-Koordinate des i-ten Partikels und g die Gravitationskonstante sind.

Die inneren Energien ergeben sich aus den Wechselwirkungen der Partikel mit ihren Nachbarn. Dabei müssen die von den einzelnen Nachbarn resultierenden Energien addiert werden. Die Energien, die sich aus der Wechselwirkung mit einem Nachbarn ergeben, sind gegeben durch

$$\begin{aligned} E_{z_i} &= \int_0^{l_i} F_z(l) \, dl , \\ E_{b_i} &= \int_0^{\psi_i} F_b(\alpha) \, d\alpha , \\ E_{s_i} &= \int_0^{\phi_i} F_b(\alpha) \, d\alpha , \end{aligned}$$

wobei l_i die Auslenkung des Partikels relativ zur Ruhelage ist und ψ_i bzw. ϕ_i die Winkelauslenkungen relativ zum Ruhewinkel. Die Funktionen $F_z(\cdot)$, $F_b(\cdot)$ und $F_s(\cdot)$ müssen so implementiert werden, daß bei größeren Auslenkungen als der bisher maximalen der selben Partikel-Verbindung die obere Kurven der Kawabata-Diagramme benutzt werden und andernfalls die untere.

2.1.3.2.2 Äußere Kräfte

Die Kräfte, die nach der Differentiation der Lagrange-Funktion dazuaddiert werden müssen, sind die Kräfte, die aus der Interaktion der Umgebung resultieren (siehe Abschnitt 2.1.2.1) und die Kraft der Luftreibung F_{luft_i}, die durch

$$E_{luft_i} = \frac{1}{2}\rho c_w A v^2$$

gegeben ist. Dabei ist ρ das spezifische Gewicht der Luft, c_w der Luftreibungskoeffizient und A die projizierte Querschnittsfläche des durch das Partikel repräsentierten Stoffteils.

2.1.3.2.3 Aufstellen der Differentialgleichung

Bevor nun mit der numerischen Lösung des Anfangswertproblems begonnen werden kann, muß noch die rechte Seite von Gleichung (2.4) berechnet werden. Dazu muß die im vorigen Abschnitt bestimmte Lagrange-Funktion partiell differenziert werden. Dies kann mit Hilfe eines Computeralgebra-Programms, wie zum Beispiel *Maple*, erfolgen, mit dem die Differentiation symbolisch durchgeführt und optimierter C-Code erzeugt werden kann.

Kapitel 3

Globale Beleuchtungssimulation

Ziel der globalen Beleuchtungssimulation ist, die Lichtausbreitung in einer Szene physikalisch korrekt zu simulieren, um möglichst photorealistische Darstellungen zu erreichen. Dabei sollen alle physikalischen Vorgänge wie die Interaktionen des Lichts zwischen den einzelnen Oberflächen in der Szene berücksichtigt und möglichst genau modelliert werden.

3.1 Die Beleuchtungsgleichung

Um physikalisch korrekte Beleuchtungsverhältnisse in einer Szene berechnen zu können, ist es notwendig, den Vorgang der Reflektionen und Lichtstreuungen an den Oberflächen in der Szene zu modellieren.

Ähnliche Vorgänge wurden seit vielen Jahren in der Literatur über Wärmeleitung studiert. Eine Übertragung der Wärmeleitungsgleichung auf das Problem der Lichtausbreitung wurde 1986 von Kajiya veröffentlicht, der in [26] die Beleuchtungsgleichung vorstellte. Diese Gleichung ist gegeben durch

$$L(x, \omega_r) = L_\epsilon(x, \omega_r) + L_r(x, \omega_r) \tag{3.1}$$

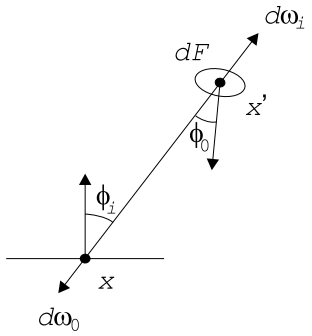
mit

$$L_r(x, \omega_r) = \int_F \tilde{\rho}(x, \omega_r, \omega_i)\ L(x', \omega_0)\ H(x, x') \frac{\cos\phi_i \cos\phi_0}{|x - x'|^2}\ dx' \tag{3.2}$$

für alle Oberflächenpunkte x in der Szene und alle Richtungen ω_r. Dabei ist F die Menge aller Oberflächenpunkte der Szene und

$L(x,\omega)$ = die Strahldichte von Punkt x in Richtung ω
$L_\epsilon(x,\omega)$ = die emittierte Strahldichte von Punkt x in Richtung ω
$L_r(x,\omega)$ = die reflektierte Strahldichte von Punkt x in Richtung ω
$\tilde{\rho}(x,\omega_r,\omega_i)$ = die bidirektionale Reflektionsfunktion
$H(x,x')$ = $\begin{cases} 0, & \text{falls } x \text{ von } x' \text{ aus unsichtbar} \\ 1, & \text{falls } x \text{ von } x' \text{ aus sichtbar} \end{cases}$
die Sichtbarkeitsfunktion zwischen x und x'.

ϕ_i und ϕ_0 sind die in der Skizze dargestellten von x und x' abhängigen Winkel.



Die Strahldichte, die einen Punkt x in Richtung ω verläßt, setzt sich also zusammen aus der emittierten Strahldichte, die dieser Punkt in Richtung ω ausstrahlt, und dem in Richtung ω reflektierten Anteil der von allen anderen Punkten der Szene auf diesen Punkt einfallenden Strahldichte. Die emittierte Strahldichte ist an allen Punkten, die nicht zu einer Lichtquelle gehören, gleich Null. Die bidirektionale Reflektionsfunktion $\tilde{\rho}(x,\omega_r,\omega_i)$ (in der englischen Literatur als "bidirectional reflection distribution function" (BRDF) bezeichnet) gibt an, welcher Anteil der aus Richtung ω_i am Punkt x einfallenden Strahldichte in Richtung ω_r reflektiert wird.

Die Gleichungen wurden hier für monochromatisches Licht formuliert. Alle beteiligten Größen sind in Wirklichkeit wellenlängenabhängig. Um farbige Szenen zu berechnen, werden die Gleichungen in der Regel für eine Anzahl fester Wellenlängen berechnet. Auch die Berechnung für die drei Grundfarben rot, grün und blau liefert gute Ergebnisse.

3.2 Das Radiosity-Verfahren

Das Radiosity-Verfahren, das 1983 von Goral, Torrance, Greenberg und Battaile vorgestellt wurde, ist ein globales Beleuchtungsverfahren, das auf einer Einschränkung der Beleuchtungsgleichung (3.2) auf diffus reflektierende Oberflächen beruht, die in natürlichen Umgebungen und insbesondere in Innenraumszenen vorherrschend sind. Damit lassen sich Ergebnisse mit sehr hohem Realitätsgrad erreichen.

Bei diesem Verfahren wird vereinfachend angenommen, daß alle Flächen der Szene rein diffuse lambertsche Reflektoren sind (Radiosity-Annahme), das heißt,

daß die reflektierte Leuchtdichte

$$L_r(x, \omega_r) = L_r(x)$$

unabhängig von der Abstrahlrichtung ist. Das auf einen Punkt einfallende Licht wird also mit gleicher Leuchtdichte in alle Richtungen des Halbraumes über der Tangentialebene in diesem Punkt reflektiert. Als relevante Größe kann deshalb die spezifische Strahlungsleistung (englisch: Radiosity) verwendet werden, die mit dem Buchstaben B bezeichnet wird. Die zusätzlichen Annahmen, daß die BRDF auch unabhängig von der Einstrahlrichtung und die emittierte Leuchtdichte auch unabhängig von der Abstrahlrichtung sind, liefern

$$\tilde{\rho}(x, \omega_r, \omega_i) = \tilde{\rho}(x).$$

und

$$L_\epsilon(x, \omega_r) = L_\epsilon(x).$$

Die spezifische emittierte Strahlungsleistung $E(x)$ und die gesamte spezifische Strahlungsleistung $B(x)$ im Punkt x sind dann gegeben durch

$$\begin{aligned} E(x) &= \int_\Omega L_\epsilon(x, \omega_r) \cos\phi_r d\omega_r \\ &= \pi\ L_\epsilon(x) \end{aligned}$$

und

$$\begin{aligned} B(x) &= \int_\Omega (L_\epsilon(x, \omega_r) + L_r(x, \omega_r)) \cos\phi_r d\omega_r \\ &= \int_\Omega L_\epsilon(x, \omega_r) \cos\phi_r d\omega_r + \int_\Omega L_r(x, \omega_r) \cos\phi_r d\omega_r \\ &= \pi\ L_\epsilon(x) + \pi\ L_r(x) \ =\ \pi\ L(x) \\ &= E(x) + \pi\ L_r(x) \end{aligned}$$

Ersetzt man $\pi\ \tilde{\rho}(x)$ durch den Reflektionsfaktor $\rho(x)$ mit $0 \leq \rho(x) < 1$, so ergibt sich damit unter Berücksichtigung von Gleichung (3.2) die Radiositygleichung

$$\begin{aligned} B(x) &= E(x) + \pi\ \tilde{\rho}(x) \int_F L(x')\ H(x, x') \frac{\cos\phi_i \cos\phi_0}{|x - x'|^2}\ dx' \\ &= E(x) + \rho(x) \int_F B(x')\ H(x, x') \frac{\cos\phi_i \cos\phi_0}{\pi\ |x - x'|^2}\ dx'. \end{aligned} \tag{3.3}$$

Die gesuchte Funktion B ist dabei eine Funktion endlicher Energie. Das heißt, gesucht ist eine Funktion B im unendlichdimensionalen Hilbertraum $L^2(F)$ (im folgenden nur noch mit L^2 bezeichnet), die diese Gleichung erfüllt.

Gleichung (3.3) kann auch mit Hilfe linearer Operatoren formuliert werden:

$$B = E + \mathcal{K}B \ , \tag{3.4}$$

mit

$$(\mathcal{K}f)(x) := \int_F K(x,x')f(x') \ dx'$$

und

$$K(x,x') := \rho(x) \ H(x,x')\frac{\cos\phi_i \cos\phi_0}{\pi \ |x-x'|^2}. \tag{3.5}$$

Dabei ist $\mathcal{K}$ ein linearer Operator auf L^2. Die Funktion K wird *Radiosity-Kern* genannt.

Für realistische Szenen, die aus tausenden oder gar hunderttausenden von Flächen bestehen können, ist diese Gleichung auf analytische Weise nicht zu lösen. Deshalb wählt man einen endlichdimensionalen Teilraum $V \subset L^2$ und löst die auf diesen Teilraum projizierte korrespondierende Gleichung. Das heißt, man wählt endlich viele Basisfunktionen, mit deren Hilfe man hofft, die Radiosity-Lösung hinreichend genau approximieren zu können, und bestimmt dann eine Linearkombination dieser Basisfunktionen, die die auf den von diesen Basisfunktionen aufgespannten Teilraum projizierte korrespondierende Gleichung löst.

Sei nun $V \subset L^2$ ein solcher Teilraum mit $dim(V) = n$, $\{N_i : i = 1,\ldots,n)$ eine Basis von V und $\{\tilde{N}_i : i = 1,\ldots,n)$ die zugehörige duale Basis, das heißt

$$\langle N_i, \tilde{N}_j \rangle = \delta_{ij} \ .$$

Dabei ist δ_{ij} das Kronecker-Delta und

$$\langle f_1, f_2 \rangle := \int_F \ f_1(x)f_2(x) \ dx \qquad \text{für } f_1, f_2 \in L^2$$

das innere Produkt von L^2. Dann ist die orthogonale Projektion $P_V : L^2 \to V$ gegeben durch

$$P_V(f) := \sum_{i=1}^{n} b_i \; N_i$$

$$\text{mit} \qquad b_i := \langle f, \tilde{N}_i \rangle \qquad \text{für } f \in L^2$$

Durch Einschränkung des linearen Integraloperators $\mathcal{K}$ auf den Teilraum V erhält man die zu (3.4) korrespondierende endlichdimensionale Gleichung

$$\hat{B} = \hat{E} + P_V \mathcal{K} \hat{B} \; . \tag{3.6}$$

Dabei ist $\hat{E} = P_V E$ die orthogonale Projektion von E auf V. (Es ist zu bemerken, daß die Lösung $\hat{B}$ dieser Gleichung nicht notwendigerweise die orthogonale Projektion $P_V B$ der Lösung der ursprünglichen Gleichung (3.4) ist)

Diese endlichdimensionale lineare Gleichung läßt sich nun in Matrixform darstellen:

$$\hat{B} = \hat{E} + \hat{\mathcal{K}} \hat{B} \; . \tag{3.7}$$

Die Koeffizienten $(k_{ij})_{i,j=1,\ldots,n}$ der Matrix $\hat{\mathcal{K}}$ sind dabei gegeben durch

$$k_{ij} = \langle \mathcal{K} N_j, \tilde{N}_i \rangle = \int_F \int_F K(x, x') N_j(x') \tilde{N}_i(x) \; dx' \; dx \; . \tag{3.8}$$

3.2.1 Klassisches Radiosity

Beim klassischen Radiosity zerlegt man F in n einfache Teilgebiete F_i (Flächenstücke oder auch Patches genannt) und wählt als endlichdimensionalen Teilraum V den Raum der auf diesen Flächenstücken F_i konstanten Funktionen. Als Basisfunktionen wählt man dann die charakteristischen Funktionen der Flächenstücke:

$$N_i(x) = \begin{cases} 1, & \text{falls } x \text{ auf Flächenstück } F_i \text{ liegt} \\ 0, & \text{sonst,} \end{cases} \qquad i = 1, \ldots, n.$$

Die duale Basis ist in diesem Fall gegeben durch

$$\tilde{N}_i(x) = \frac{1}{A_i} N_i(x) \qquad i = 1, \ldots, n \; ,$$

wobei A_i der Flächeninhalt von Flächenstück F_i ist.

Die Lösung der endlichdimensionalen Gleichung (3.7) liefert dann eine Approximation der Radiosity-Lösung der Form

$$B(x) \approx \sum_{i=1}^{n} B_i \; N_i(x) \; ,$$

die auf den einzelnen Flächenstücken F_i konstante Werte B_i annimmt. Die projizierte Emissionsfunktion $\hat{E}$ ist gegeben durch

$$\hat{E}(x) = \sum_{i=1}^{n} E_i \; N_i(x)$$

und nimmt auf den einzelnen Flächenstücken F_i ebenso konstante Werte an.
Für die Koeffizienten k_{ij} der Matrix $\hat{\mathcal{K}}$ ergibt sich damit

$$\begin{aligned} k_{ij} &= \langle \mathcal{K} N_j, \tilde{N}_i \rangle \\ &= \int_F \int_F K(x,x') N_j(x') \frac{1}{A_i} N_i(x) \; dx' \; dx \\ &= \frac{1}{A_i} \int_{F_i} \int_{F_j} K(x,x') \; dx' \; dx \\ &= \frac{1}{A_i} \int_{F_i} \int_{F_j} \rho(x) H(x,x') \frac{\cos\phi_i \cos\phi_j}{\pi \; |x-x'|^2} \; dx' \; dx \end{aligned}$$

Nimmt man ferner an, daß auf jedem Flächenstück F_i der Reflektionsfaktor $\rho(x) = \rho_i$ konstant ist, erhält man die klassische Radiositygleichung

$$\begin{aligned} B_i &= E_i + \rho_i \sum_{j=1}^{n} B_j \; \frac{1}{A_i} \int_{F_i} \int_{F_j} H(x,x') \frac{\cos\phi_i \cos\phi_j}{\pi \; |x-x'|^2} \; dx' \; dx \\ &= E_i + \rho_i \sum_{j=1}^{n} B_j \; F_{ij} \qquad \text{für alle } i = 1, \ldots, n. \end{aligned} \tag{3.9}$$

Dabei ist

$$F_{ij} = \frac{1}{A_i} \int_{F_i} \int_{F_j} H(x,x') \frac{\cos\phi_i \cos\phi_j}{\pi \; |x-x'|^2} \; dx' \; dx \tag{3.10}$$

der sogenannte Formfaktor [27] von Flächenstück F_i zu Flächenstück F_j. Dieser Formfaktor ist eine geometrischen Größe und gibt an, welcher Anteil der Strahlungsleistung, die von Flächenstück F_i abgestrahlt wird, auf Flächenstück F_j ankommt.

Da die Integrale in Gleichung (3.10) unter Vertauschung invariant sind, ergibt sich zwischen den Formfaktoren F_{ij} und F_{ji} die Reziprozitätsbeziehung

$$A_i \, F_{ij} \;=\; A_j \, F_{ji} \qquad \text{für alle } i,j = 1,\ldots,n. \tag{3.11}$$

Aus Gründen der Energieerhaltung gilt außerdem

$$\sum_{j=1}^{n} F_{ij} \leq 1 \qquad \text{für alle } i = 1,\ldots,n \tag{3.12}$$

und, da ein im Grenzfall ebenes Flächenstück kein Licht auf sich selber abstrahlt, gilt

$$F_{ii} \;= 0 \qquad \text{für alle } i = 1,\ldots,n.$$

Die Radiosity-Gleichung (3.9) lautet damit in Matrixform:

$$\begin{pmatrix} 1 & -\rho_1 F_{12} & \ldots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1 & \ldots & -\rho_2 F_{2n} \\ \vdots & & & \vdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \ldots & 1 \end{pmatrix} \cdot \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{pmatrix}. \tag{3.13}$$

Da die Reflektionsfaktoren ρ_i alle kleiner als 1 sind und wegen (3.12) ist die Koeffizientenmatrix strikt zeilendiagonaldominant und damit auch regulär.

Um die Radiosity-Werte B_i zu berechnen, müssen nun zum einen die Formfaktoren berechnet und zum anderen das Gleichungssystem gelöst werden. Dabei stellt für realistische Szenen die Größe der Matrix ein besonderes Problem dar, da diese oft aus hunterttausenden oder gar Millionen Flächenstücken bestehen.

3.2.2 Berechnung der Formfaktoren

Die Berechnung des Formfaktors F_{ji} von einer Fläche F_j zu einer Fläche F_i erfordert die Lösung eines doppelten Flächenintegrals (siehe Gleichung (3.10)). Die Flächen sind beliebig im dreidimensionalen Raum orientiert und können komplizierte Randkurven besitzen. Erschwerend kommt noch dazu, daß der Integrand aufgrund der Sichtbarkeitsfunktion $H(x,x')$ Unstetigkeiten besitzt, die nicht achsenparallel liegen.

Diese Integrale lassen sich nur in einfachen Anordnungen analytisch lösen, wie zum Beispiel im Fall polygonaler Flächen und ohne Berücksichtigung der Sichtbarkeitsfunktion $H(x,x')$ (siehe [28]). Auch gewöhnliche Quadraturformeln zur numerischen Lösung der Integrale lassen sich wegen der Unstetigkeiten der Integranden nicht effizient anwenden. Deshalb müssen entweder vereinfachende Annahmen gemacht werden, oder die Integrale müssen mit Hilfe einer Monte-Carlo Methode gelöst werden.

3.2.2.1 Näherungsweise Berechnung

Eine wesentliche Vereinfachung für die Berechnung des Formfaktors F_{ji} ergibt sich durch die Annahme, daß die Ausdehnungen der beiden Flächen F_j und F_i im Vergleich zu ihrem Abstand klein sind (Fernnäherung). Die Fläche F_j wird dann als infinitesimal kleine Fläche betrachtet. Damit kann angenommen werden, daß die Sichtbarkeit eines festen Punktes x auf der Fläche F_i für alle Punkte x' auf der infinitesimal kleinen Fläche F_j konstant bleibt. Die Integration über die Fläche F_j wird damit zur Multiplikation mit ihrem Flächeninhalt A_j. Da auch die Ausdehnung der Fläche A_i im Vergleich zum Abstand der beiden Flächen klein ist, ändert sich das Ergebnis nicht wesentlich, wenn die Abstände $|x-x'|$ durch eine Konstante r ersetzt werden. Der Formfaktor F_{ji} ist dann gegeben durch

$$F_{ji} = \int_{F_i} H(x) \frac{\cos\phi_i \cos\phi_j}{\pi \; r^2} \; dx \; .$$

Dieses Integral läßt sich für spezielle Flächen F_i, wie zum Beispiel polygonale Flächen (siehe [28]) oder Kreisscheiben (siehe unten), ohne Berücksichtigung der Sichtbarkeitsfunktion $H(x)$ analytisch lösen.

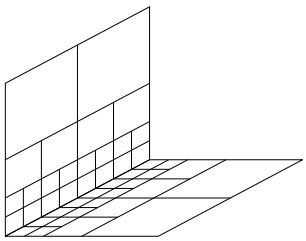


Abbildung 3.1: Die Annahme der Fernnäherung läßt sich für Flächen, die sich berühren, auch durch Unterteilung nicht erfüllen.

Die zugrundeliegende Annahme, daß die Ausdehnung der Flächen F_j und F_i klein sind im Vergleich zu deren Abstand, läßt sich in vielen Fällen durch Unterteilung der Flächen erreichen. Es gibt aber auch Fälle, wo sich diese Annahme auch durch Unterteilung nicht erfüllen läßt, zum Beispiel bei sich berührenden nicht in einer Ebene liegenden Flächen (siehe Abbildung 3.1). Diese Formfaktoren sind dann fehlerbehaftet.

3.2.2.1.1 Das Hemi-Cube-Verfahren

Das Hemi-Cube-Verfahren war das erste effiziente Verfahren zur Formfaktorberechnung, das gleichzeitig auch das Verdeckungsproblem löste, und wurde 1985 von Cohen und Greenberg vorgestellt [29]. Dieses Verfahren berechnet in einem Rechenschritt die Formfaktoren F_{ji} von einem als infinitesimal klein angenommenen Flächenstück F_j zu allen anderen Flächenstücken F_i der Szene.

Da der Formfaktor F_{ji} nur vom Raumwinkel abhängt, den das Flächenstück F_i vom Flächenstück F_j aus einnimmt, kann statt des Flächenstücks F_i eine beliebige andere Fläche zur Berechnung herangezogen werden, die den selben Raumwinkel einnimmt. Die Idee ist nun, alle Flächen F_i der Szene unter Berücksichtigung der gegenseitigen Verdeckung auf einen das infinitesimal kleine Flächenstück F_j umgebenden Halb-Würfel (englisch: Hemi-Cube) zu projizieren und dann die Formfaktoren von F_j zu diesen projizierten Flächen zu berechnen (siehe Abbildung 3.2).

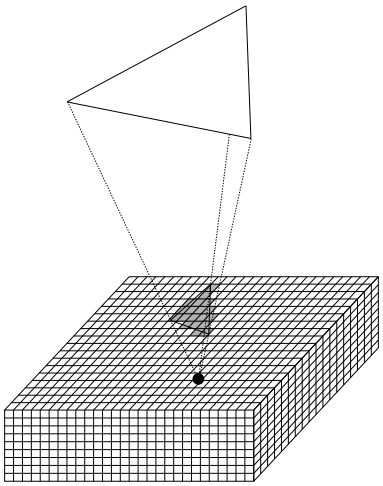


Abbildung 3.2: Formfaktorberechnung mit dem Hemi-Cube-Verfahren

Dazu wird jede Seite des Halb-Würfels mit einem Rechtecksraster sogenannter Hemi-Cube-Pixel überzogen. Die Projektion der Szene auf diese Hemi-Cube-Pixel kann dann durch Rasterisierung der Szene mit Hilfe von Standard-Rasteralgorithmen (oder Raster-Hardware) durchgeführt werden. Die Verwendung eines z-Buffers erlaubt dabei auch die Berücksichtigung der gegenseitigen Verdeckung der einzelnen Flächen. Anstatt Farbwerten werden dabei aber Objekt-IDs in den Hemi-Cube-Pixeln gespeichert.

Der Formfaktor von F_j zu einer Fläche F_i wird dann durch die Summe der sogenannten Delta-Formfaktoren von F_j zu den von der Fläche F_i belegten Hemi-Cube-Pixeln approximiert. Auf diese Weise läßt sich eine komplette Zeile der Formfaktormatrix durch Rasterisierung der Szene auf die fünf Flächen des Halb-Würfels berechnen. Die Delta-Formfaktoren können einmalig vorberechnet und dann für alle Formfaktorberechnungen verwendet werden.

Aufgrund der endlichen und festen Anzahl der Hemi-Cube-Pixel treten bei diesem Verfahren aber Alias-Fehler auf, die zu auffälligen Artefakten führen.

3.2.2.1.2 Formfaktorberechnung mit Raytracing

Bei der Formfaktorberechnung mit dem Verfahren von Wallace [30] wird der Formfaktor F_{ji} einer als infinitesimal klein angenommenen Fläche F_j zu einer Fläche F_i näherungsweise bestimmt, indem die Fläche F_i durch eine Anzahl Kreisscheiben des selben Flächeninhalts approximiert wird.

Die Sichtbarkeit zwischen F_j und den Kreisscheiben wird durch Raytracing bestimmt. Dazu werden Strahlen von F_j zu den Kreisscheibenmittelpunkten durch die Szene verfolgt und auf Schnittpunkte mit anderen Objekten getestet. Der Formfaktor F_{ji} wird dann durch die Summe der Formfaktoren von F_j zu den sichtbaren Kreisscheiben approximiert (siehe Abbildung 3.3).

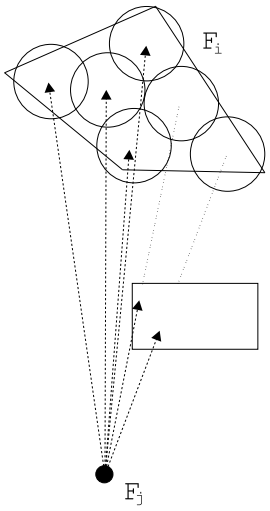
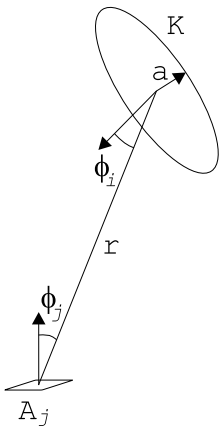


Abbildung 3.3: Approximation der Fläche F_i durch Kreisscheiben und Sichtbarkeitsberechnung mit Hilfe von Raytracing.

Ist K eine Kreisscheibe mit Radius a, so erhält man den unverdeckten Formfaktor von F_j zu K durch

$$F_{jK} = \frac{a^2 \cos\phi_i \cos\phi_j}{r^2 + a^2} .$$

Dabei ist r der Abstand von F_j zu K und ϕ_i bzw. ϕ_j die aus der Skizze ersichtlichen Winkel zwischen den Flächennormalen und der Verbindungslinie von F_j zu K.



3.2.2.2 Monte-Carlo-Formfaktorberechnung

Die Idee der Monte-Carlo-Methode (siehe [31, 32]) ist, die Lösung eines mathematischen Problems mit Hilfe geeignet gewählter Zufallsexperimente abzuschätzen. Diese Idee kann auch zur Berechnung von Formfaktoren auf verschiedene Weise angewendet werden. So kann das doppelte Flächenintegral, das zur Berechnung des Formfaktors F_{ji} zwischen zwei Flächenstücken F_j und F_i gelöst werden muß, durch Monte-Carlo-Integration bestimmt werden.

Oder es können ganze Serien von Formfaktoren F_{ji}, $(i = 1, \ldots, n)$ bestimmt werden, indem zufällige vom Flächenstück F_j ausgehende Strahlen durch die Szene verfolgt werden. Eine große Anzahl globaler, unabhängig von allen Flächenstücken ausgewählter Linien erlauben die Bestimmung der gesamten Formfaktoren F_{ji}, $(i = 1, \ldots, n;\ j = 1, \ldots, n)$ (siehe dazu auch [33]).

3.2.2.2.1 Monte-Carlo Integration

Bei der Monte-Carlo Integration wird das Integral einer Funktion $g(x)$ über ein Parametergebiet P approximiert durch

$$\int_P g(x)dx \;\approx\; \mu(P)\;\frac{1}{N}\sum_{i=0}^{N-1} g(x_i)\;,$$

wobei $\{x_0, \ldots, x_{N-1}\}$ eine gleichverteilte Folge zufälliger Sample-Punkte im Parametergebiet P und $\mu(P)$ das Maß des Parametergebiets ist (ist P ein reelles Intervall, so ist $\mu(P)$ die Intervallänge; ist P eine Fläche, so ist $\mu(P)$ deren Flächeninhalt). Für große Zahlen von Sample-Punkten konvergiert diese Näherung gegen die richtige Lösung des Integrals. Der Erwartungswert des Fehlers ist beschränkt durch

$$\left|\int_P g(x)dx \;-\; \mu(P)\;\frac{1}{N}\sum_{i=0}^{N-1} g(x_i)\right| \;\leq\; \sigma(g)\frac{1}{\sqrt{N}}\;.$$

Dabei ist $\sigma(g)$ die Varianz der Funktion g.

Die Approximation des Formfaktors F_{ji} von einer Fläche F_j zu einer Fläche F_i ist dann gegeben durch

$$F_{ji} \;\approx\; \frac{A_i}{N\;M}\;\sum_{i=0}^{N-1}\;\sum_{j=0}^{M-1} H(x_i, x'_j)\;\frac{\cos\phi_i \cos\phi_j}{\pi\;|x_i - x'_j|^2}$$

wobei $\{x_0, \ldots, x_{N-1}\}$ eine gleichverteilte Folge zufälliger Sample-Punkte auf der Fläche A_i und $\{x'_0, \ldots, x'_{M-1}\}$ eine gleichverteilte Folge zufälliger Sample-Punkte auf der Fläche A_j sind. Die Sichtbarkeit $H(x_i, x'_j)$ zwischen den Punkten x_i und x'_j wird durch Raytracing ermittelt.

Die Konvergenzrate $\mathcal{O}(1/\sqrt{N})$, die man bei Verwendung unabhängiger Sample-Punkte erhält, ist sehr langsam. Erhöht man die Genauigkeitsanforderung um eine Dezimalstelle, muß der 100-fache Aufwand getrieben werden. Durch andere Sampling-Strategien läßt sich die Konvergenzrate aber erheblich verbessern.

Mit Hilfe der Quasi-Monte-Carlo Methode [34], bei der die Sample-Punkte gleichmäßiger verteilt aber nicht mehr unabhängig gewählt werden, wird eine Konvergenzrate der Ordnung $\mathcal{O}(1/N)$ erreicht. Sequentielle Monte-Carlo Methoden [35], bei denen die Berechnung in mehreren Stufen mit adaptiver Anpassung der Sampling-Strategie abläuft, erreichen sogar exponentielle Konvergenzraten.

3.2.2.2.2 Richtungsabtastung mit Cosinus-Verteilung

Tastet man den Halbraum über einem Flächenstück F_j durch Strahlen ab, deren Ursprung auf dem Flächenstück F_j zufällig und gleichverteilt und deren Richtungen zufällig und Cosinus-verteilt gewählt werden (siehe Abbildung 3.4), so entspricht die Wahrscheinlichkeit, daß der erste Schnittpunkt eines solchen Strahls auf einem Flächenstück F_i liegt, genau dem Formfaktor F_{ji}. Solche Linien nennt man lokale gleichverteilte Linien.

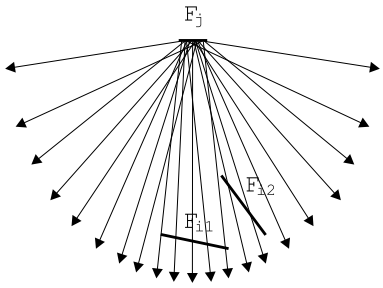


Abbildung 3.4: Richtungen mit Cosinus-Verteilung.

Werden in einem Experiment N solcher lokalen Linien von einem Flächenstück F_j ausgehend durch die Szene verfolgt und ist N_i die Anzahl der Strahlen, die auf Flächenstück F_i treffen, so ist der Formfaktor F_{ji} gegeben durch

$$F_{ji} \approx \frac{N_i}{N} .$$

Auf diese Weise läßt sich in einem Experiment eine komplette Zeile der Formfaktormatrix bestimmen, wozu aber im allgemeinen eine sehr große Anzahl an Strahlen nötig ist.

3.2.2.2.3 Gleichverteilte Globale Linien

Auch gleichverteilte globale Linien, die ihren Ursprung nicht, wie im vorigen Abschnitt, gezielt in einem vorgegebenen Flächenstück haben, lassen sich zur Formfaktorberechnung heranziehen (siehe Abbildung 3.5). Globale gleichverteilte Linien

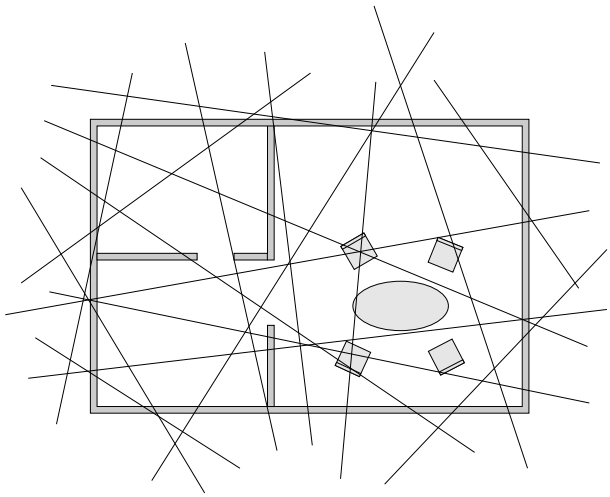


Abbildung 3.5: Globale Linien.

lassen sich zum Beispiel erzeugen, indem je zwei gleichverteilte zufällig gewählte Punkte auf einer die Szene umgebenden Kugeloberfläche miteinander verbunden werden.

Es kann gezeigt werden, daß die Wahrscheinlichkeit, daß eine solche Linie eine Oberfläche schneidet, proportional zu deren Flächeninhalt ist. Ist in einem Experiment, in dem N solche gleichverteilte globale Linien erzeugt werden, N_j die Anzahl der Linien, die das Flächenstück F_j schneiden und N_{ji} die Anzahl der Linien, die die Flächenstücke F_j und F_i ohne weitere dazwischenliegende Schnittpunke schneiden, so ist der Formfaktor F_{ji} gegeben durch

$$F_{ji} \approx \frac{N_{ji}}{N_j} .$$

Dies liefert eine Abschätzung aller Formfaktoren der gesamten Formfaktormatrix. In [36] wird gezeigt, daß solche globalen Monte-Carlo-Methoden nur effizient arbeiten können, wenn die von den primären Lichtquellen ausgehenden Formfaktoren durch lokale Linien bestimmt werden. Dies ist darin begründet, daß der Einfluß der primären Lichtquellen auf das Gesamtergebnis im allgemeinen viel größer ist als der indirekter Reflektionen.

3.2.3 Full-Matrix-Verfahren

Zur Lösung von Gleichung (3.13) könnte im Prinzip ein Standard-Lösungsverfahren für lineare Gleichungssysteme verwendet werden, wie zum Beispiel das Eliminationsverfahren von Gauß . In der Praxis sind solche Verfahren aber nicht praktikabel, da diese eine Komplexität von $O(n^3)$ besitzen und realistische Szenen häufig aus tausenden oder gar hunderttausenden von Flächenelementen bestehen.

Auch iterative Verfahren wie die Jacobi- oder die Gauß-Seidel-Iteration, die eine geringere Komplexität besitzen, stoßen sehr schnell an ihre Grenzen, da es in der Regel aufgrund ihrer Größe nicht einmal möglich ist, die gesamte Koeffizienten-Matrix zu speichern, geschweige denn alle ihre Koeffizienten zu berechnen, in die die gesamte geometrische Information der Szene fließt, wie zum Beispiel die gegenseitige Verdeckung.

Es ist deshalb notwendig, Verfahren zu finden, die mit einem kleinen Teil der Koeffizienten der Matrix auskommen. Es gibt zwei Ansatzpunkte, wie dies erreicht werden kann. Zum einen trägt der Strahlungsaustausch zwischen manchen Flächen-Paaren nur unwesentlich zum Ergebnis bei, so daß die entsprechenden Koeffizienten vernachlässigt werden können. Diese Tatsache wird zum Beispiel durch das Progressive-Refinement-Verfahren (siehe Abschnitt 3.2.4) ausgenutzt. Zum andern kann räumliche Kohärenz ausgenutzt werden, da sich die Koeffizienten von benachbarten Flächenstücken oft nur unwesentlich unterscheiden. So können ganze Blöcke der Matrix durch einen einzigen Wert approximiert werden. Dies wird zum Beispiel beim hierarchischen Radiosity (siehe Abschnitt 3.2.5) oder beim Wavelet-Radiosity (siehe Abschnitt 3.2.6) ausgenutzt.

3.2.4 Das Progressive-Refinement-Verfahren

Löst man das Radiosity-Gleichungssystem mit Hilfe des Gauß-Seidel-Verfahrens, ist dies physikalisch äquivalent mit dem sukzessiven Einsammeln von Strahlungsleistung. In jedem Iterationsschritt sammelt eine ausgewählte Fläche die von allen anderen Flächen einfallende Strahlungsleistung auf.

Cohen et. al. stellten 1988 den Progressive-Refinement Algorithmus [37] vor, der diesen physikalischen Vorgang umkehrt. Hier wird in jedem Iterationsschritt die Strahlungsleistung einer ausgewählten Fläche auf alle anderen Flächen verteilt. Bei dieser Vorgehensweise ist es nun möglich, in jedem Iterationsschritt eine Fläche mit maximaler noch zu verteilender Strahlungsleistung (Fläche größter Strahlungsleistung) auszuwählen, was eine sehr schnelle Konvergenz zu Beginn des Berechnungsprozesses zur Folge hat und schon nach wenigen Iterationsschritten sehr gute visuelle Ergebnisse liefert, auch wenn die absolute Abweichung von der exakten Lösung im Allgemeinen noch relativ groß ist.

Bei der Berechnung mit dem Progressive-Refinement Verfahren muß für jedes Flächenstück F_i zusätzlich zur gesuchten Radiosity B_i noch über die unverteilte Strahlungsleistung ΔB_i buchgeführt werden, die auf dem Flächenstück angekommen ist aber noch nicht in die Szene reflektiert wurde. Der Algorithmus kann wie in Abbildung 3.7 dargestellt formuliert werden.

Der Progressive-Refinement Algorithmus ist eine Variante der Southwell Relaxation [38]. Die Konvergenz ist aufgrund der Diagonaldominanz der Formfaktormatrix garantiert.

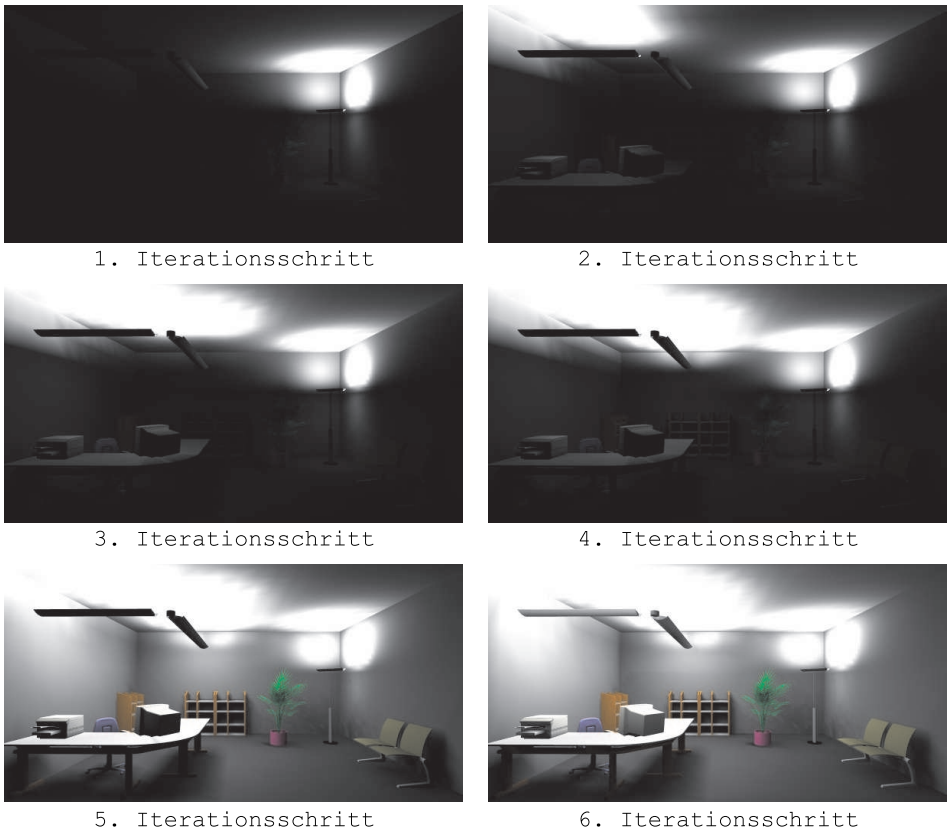


Abbildung 3.6: Berechnungsfolge des Progressive-Refinement-Verfahrens

Zur Durchführung eines Iterationsschrittes werden die Formfaktoren einer Spalte (oder äquivalent aufgrund der Reziprozitätsbeziehung (3.11) eine Zeile) der Formfaktormatrix benötigt , die zu dem Zeitpunkt berechnet werden können, zu dem sie benötigt werden, und nicht abgespeichert werden müssen. Da bereits nach wenigen Iterationsschritten gute Ergebnisse erzielt werden, muß ein großer Teil der Formfaktoren überhaupt nicht berechnet werden. Es kommt allerdings auch vor, daß Formfaktoren mehrfach berechnet werden (zum Beispiel, wenn eine Fläche mehrmals als verteilende Fläche ausgewählt wird). Die Zeit- und Speicherplatzersparnis macht diesen Nachteil aber mehr als wett.

In Abbildung 3.6 ist ein Beispiel dargestellt, das mit Hilfe des Progressive-Refinement-Algorithmus berechnet wurde. Die Szene besteht aus etwas mehr als 30 000 Flächenstücken und besitzt vier Lichtquellen, die den größten Anteil ihres Lichtes gegen die Decke strahlen und den Raum damit durch indirektes Licht erhellen. Insgesamt wurden nur sechs Iterationsschritte durchgeführt, allerdings wurde in den letzten beiden Iterationsschritten, in denen indirekte Strahlungsleistung in die Szene verteilt wurde, nicht ein einzelnes Flächenelement als abstrahlende Fläche ausgewählt, sondern eine komplette Fläche, die während der Berechnung abhängig von der Empfängerposition hierarchisch unterteilt wurde (ähnlich der in Abschnitt 3.2.5 beschriebenen Technik). Das Ergebnis sieht trotz der geringen Anzahl der durchgeführten Iterationsschritte bereits sehr realistisch aus.

```
for (alle Patches F_i) {
  B_i = E_i
  ΔB_i = E_i
}
while (weitere Iterationen notwendig) {
  wähle eine Fläche F_j, so daß ΔB_j * A_j maximal ist
  for (alle Patches F_i) {
    Δrad = ΔB_j * ρ_i F_ij
    ΔB_i = ΔB_i + Δrad
    B_i = B_i + Δrad
  }
  ΔB_j = 0
}
```

Abbildung 3.7: Der Progressive-Refinement Algorithmus.

3.2.4.1 Progressive-Refinement nach Wallace

Beim Progressive-Refinement nach Wallace [30] wird die Sichtbarkeitsberechnung mit Hilfe von Raytracing durchgeführt (siehe Abbschnitt 3.2.2.1.2). Dabei wird in jedem Iterationsschritt die abstrahlende Fläche durch kreisförmige Samples approximiert. Die empfangenden Flächen werden mit einem Punktenetz überzogen. Die Radiosity-Werte werden an den Punkten (Vertices) dieser Netze berechnet.

Zur Bestimmung der Sichtbarkeiten werden Strahlen von jedem Vertex der Empfängerflächen zu jedem Sample der Lichtquelle durch die Szene verfolgt (siehe Abbildung 3.8). An jedem Vertex werden nur die Beiträge der unverdeckten Samples berücksichtigt. Bei dieser Vorgehensweise werden Formfaktoren von infinitesimal kleinen Flächen (Vertices) zu Kreisscheiben benötigt, die sich, anders als im Fall beliebiger Flächen, analytisch berechnen lassen (siehe Abschnitt 3.2.2.1).

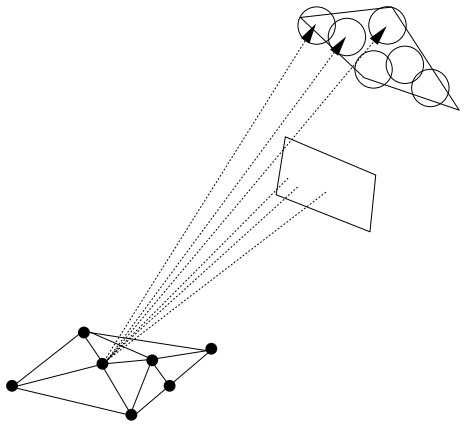


Abbildung 3.8: Sichtbarkeitsberechnung mit Raytracing

3.2.5 Hierarchisches Radiosity und Clustering

Hanrahan, Salzmann und Aupperle stellten 1991 in ihrem Artikel [39] einen Algorithmus vor, der es ermöglicht, die Formfaktormatrix, die für eine aus n Flächenstücken (sogenannten Patches) bestehende Szene n^2 Formfaktoren enthält, zu einer hierarchischen Repräsentation in Blockmatrixform aus $O(n)$ Blöcken zu reduzieren. Die Matrixeinträge werden dabei durch sogenannte *Links* zwischen den zugehörigen Flächenstücken repräsentiert.

Der Algorithmus arbeitet in zwei Phasen. In der ersten "Initial Linking"-Phase wird die Blockmatrix erzeugt, in der zweiten Phase eine Lösung des Gleichungssystems bestimmt.

Dabei startet man nicht schon mit einer in kleine Flächenstücke unterteilten Szene, sondern beginnt mit möglichst großen rechteckigen Flächenstücken, die dann während der "Initial-Linking"-Phase unterteilt werden. Als Unterteilungskriterium dient die Größe der Formfaktoren. Dabei geht man davon aus, daß ein zwischen zwei Flächenelementen ohne weitere Unterteilung abgeschätzter Formfaktor, der kleiner als eine vom Benutzer vorgegebene Schranke F_ϵ ist, den exakten Formfaktor genügend genau approximiert. Ist ein Formfaktor F_{ij} größer als F_ϵ, so wird die Empfängerfläche F_j in vier Teilflächen unterteilt. Die Unterteilung wird abgebrochen, wenn die Flächenstücke einen Flächeninhalt kleiner als eine vorgegebene Schranke A_ϵ erreicht haben.

Die Idee ist nun, Interaktionen in verschiedenen Ebenen der Unterteilungshierarchie zuzulassen.

In C-ähnlichem Pseudocode kann die "Initial-Linking"-Phase durch eine Prozedur *Entwickle* wie in Abbildung 3.9 dargestellt formuliert werden.

Bei den hier verwendeten Formfaktoren wurde noch keine Verdeckung berücksichtigt. Die Prozedur *Link* ermittelt durch eine Anzahl von Strahlen, die durch die Szene verfolgt werden, die prozentuale Verdeckung der beiden Flächenstücke und speichert einen Link, der die um die Verdeckung korrigierten Formfaktoren enthält und die Interaktion der beiden Flächenstücke beschreibt. Da alle von einem Flächenstück ausgehenden Formfaktoren in ihrer Summe höchstens 1 ergeben (siehe Gleichung 3.12) und die Unterteilung abgebrochen wird, sobald ein Formfaktor kleiner als F_ϵ wird, ist die Anzahl der Blöcke in einer Zeile der mit diesem Verfahren bestimmten Formfaktor-Blockmatrix durch eine Konstante beschränkt. Die Matrix enthält also $O(n)$ Blöcke. Dies ist allerdings nur richtig, wenn die Formfaktoren, die sich aus der Ausgangsunterteilung ergeben, nicht schon kleiner als F_ϵ sind. Ist dies der Fall, kann eine Beschränkung auf $O(n)$ Blöcke nur durch Clustering-Techniken (siehe [40]) erreicht werden. Dabei werden mehrere Ausgangsflächen oder auch ganze Bereiche der Szene zu Clustern zusammengefaßt, die als ganzes wie einzelne Flächen Radiosity senden oder empfangen können. Solche Cluster können aber nicht wie lambertsche Reflektoren behandelt werden. Die abgegebene Strahlungsleistung hängt in der Regel von der Abstrahlrichtung ab. Auch bei der Behandlung

```
Entwickle(Patch *p, Patch *q, float F_ε, float A_ε)
{
    berechne die Formfaktoren F_pq und F_qp
    if (F_pq < F_ε  &&  F_qp < F_ε)
      Link( p, q );
    else {
      if (F_pq > F_qp) {
        if (A_q > A_ε) {
          unterteile q in 4 Teile;
          for (alle i von 0 bis 3)
            Entwickle(p, q → Teil[i], F_ε, A_ε);
        } else
          Link( p, q );
      } else {      // (F_pq ≤ F_qp)
        if (A_p > A_ε) {
          unterteile p in 4 Teile;
          for (alle i von 0 bis 3)
            Entwickle(p → Teil[i], q, F_ε, A_ε);
        } else
          Link( p, q );
      }
    }
}
```

Abbildung 3.9: die "Initial-Linking"-Phase.

als mögliche Verdecker anderer Flächen muß beachtet werden, daß der Grad der Verdeckung in verschiedenen Richtungen unterschiedlich groß sein kann.

Der Aufwand für die "Initial-Linking"-Phase ist von der Ordnung $O(n)$.

Das Gleichungssystem in Blockmatrix-Form kann nun mit Hilfe von Standard-Lösungsverfahren, oder aber auch mit der Progressive-Refinement Methode gelöst werden. In diesem Fall kann die hierarchische Unterteilung der Flächenstücke und die Bestimmung der benötigten Links auch während der Progressive-Refinement-Berechnung durchgeführt werden. Andererseits ist es aber auch möglich, in den oben in Abschnitt 3.2.4 beschriebenen Progressive-Refinement Algorithmus hierarchische Unterteilungen der abstrahlenden und empfangenden Flächen zu integrieren.

3.2.6 Wavelet-Radiosity

Beim klassischen Radiosity wird eine Approximation der Radiositygleichung (3.4) berechnet, indem diese auf den Teilraum stückweise linearer Funktionen projiziert und dann dort gelöst wird (siehe Abschnitt 3.2.1).

Generell kann dafür ein beliebiger endlichdimensionaler Teilraum verwendet werden, der möglichst so gewählt wird, daß die gesuchte Radiosity-Funktion mit einer möglichst geringen Anzahl von Koeffizienten gespeichert werden kann.

Beim Wavelet-Radiosity [41, 42] werden dazu Wavelet-Konstruktionen hierarchischer Basisfunktionen verwendet. Diese Basisfunktionen können, anders als beim hierarchischen Radiosity, aus Funktionen höherer Ordnung bestehen. Die hierarchische Radiosity-Methode ist also ein Spezialfall der Wavelet-Radiosity-Methode mit stückweise konstanten Basisfunktionen.

So werden beim Wavelet-Radiosity die Vorteile einer hierarchischen Darstellung und die von Basisfunktionen höherer Ordnung miteinander vereint. Hierarchische Methoden nutzen Kohärenzen in den Formfaktoren (und damit Kohärenzen des Radiosity-Kerns (3.5) aus und vernachläßigen Koeffizienten, die fast Null sind. Basisfunktionen höherer Ordnung erlauben eine bessere Approximation mit einer geringeren Anzahl von Funktionen. Man kommt deshalb mit einer etwas gröberen Unterteilung der Flächenstücke aus.

Allerdings steigt dabei auch der Aufwand für die einzelnen Links. Die Anzahl der Koeffizienten, die pro Link berechnet und gespeichert werden müssen, steigt mit der 4. Potenz der Ordnung der Basisfunktionen. Während dies beim hierarchischen Radiosity mit konstanten Basisfunktionen nur ein Koeffizient pro Link ist, sind es bei linearen Basisfunktionen schon 16 und bei quadratischen 81.

Beim hierarchischen Radiosity wird der Radiosity-Kern durch eine über Intervalle variierender Größe konstante Funktion approximiert. An Stellen, an denen sich der Kern nur wenig ändert, werden größere Intervalle, an Stellen an denen er sich stark ändert, kleinere Intervalle verwendet.

Beylkin, Coifman und Rokhlin zeigen in [43], daß Integraloperatoren, die sehr allgemeine Stetigkeitsbedingungen erfüllen, mit $O(n)$ Koeffizienten beliebig genau approximiert werden können, wenn sie auf eine geeignete Wavelet-Basis projiziert werden. Dies läßt sich auch auf den Radiosity-Kern anwenden. Das heißt, Wavelet-Basen können dazu verwendet werden, Radiosity-Algorithmen der Komplexität $O(n)$ zu entwickeln.

Nach der Wahl einer geeigneten Wavelet-Basis müssen zur Berechnung einer Radiosity-Lösung die Koeffizienten k_{ij} (siehe Gleichung (3.8)) der Radiosity-Matrix berechnet werden. Dazu wird zunächst der Radiosity-Kern auf die Wavelet-Basis transformiert, was in [43] ausführlich beschrieben wird. Aufgrund von Glattheitseigenschaften des Radiosity-Kerns entstehen dabei viele sehr kleine Koeffizienten, die vernachlässigt werden können.

Die Integrale werden dann mit Hilfe geeigneter Quadraturformeln, wie zum Beispiel der Gauss-Legendre-Quadratur, numerisch gelöst (siehe zum Beispiel [44]).

Als Kriterium für die hierarchischen Unterteilungen wurden beim hierarchischen Radiosity die geometrischen Formfaktoren herangezogen. Beim Wavelet-Radiosity stehen hierzu nur die abstrakten Koeffizienten der Projektion des Radiosity-Kerns zur Verfügung. Es zeigt sich aber, daß hier ähnliche Argumente gefunden werden können [45]. Statt geometrischer Überlegungen zur Fehlerabschätzung werden hier polynomiale Schätzer verwendet, die den Fehler der Approximation des Radiosity-Kerns abschätzen.

3.2.7 Monte-Carlo Radiosity

Die Monte-Carlo-Methode [31, 32] kann auf verschiedene Arten bei der Lösung der Radiosity-Gleichung eingesetzt werden (siehe [33]). Sie kann zum einen zur Berechnung der Formfaktoren verwendet werden, was bereits in Abschnitt 3.2.2.2 beschrieben wurde, zum anderen zur stochastischen Lösung des linearen Gleichungssystems durch Anwendung stochastischer Relaxationsmethoden [46, 47] und schließlich zur direkten Berechnung des Energieaustausches.

Dies kann entweder durch Partikel-Transport-Simulation geschehen, wo die Ausbreitung von Photonen in der Szene simuliert wird [48], oder durch die Simulation des Energieaustausches entlang globaler Linien [36].

Stochastische Verfahren zur Berechnung von Radiosity-Lösungen führen in der Regel zu Bildern, die mit Rauschen behaftet sind. Dieses Rauschen kann durch eine entsprechend hohe Anzahl an verwendeten Strahlen und durch anschließende Filterung des Ergebnisses reduziert werden.

3.2.7.1 Partikel-Transport-Simulation

Bei der Berechnung einer Lösung der Beleuchtungsgleichung (3.1) mit Hilfe von Partikel-Transport-Simulationen [48] werden die Photonen auf Ihrem Weg von den Lichtquellen durch die Szene verfolgt. Dabei können beliebige Emissionsfunktionen der Lichtquellen und beliebige bidirektionale Reflektionsfunktionen (siehe Abschnitt 3.1) verwendet werden.

Ausgehend von den Lichtquellen werden zufällig gewählte Strahlen, die den Transport von Partikeln simulieren und deren Verteilung der Emissionsfunktion der Lichtquelle entspricht, bis zu ihrem ersten Schnittpunkt x mit einer Objektoberfläche durch die Szene verfolgt.

Integriert man die bidirektionale Reflektionsfunktion $\tilde{\rho}(x, \omega_r, \omega_i)$ an diesem Punkt x und der Einfallsrichtung des Strahls ω_i über alle möglichen Reflektionsrichtungen ω_r des über x liegenden Halbraumes, erhält man die Wahrscheinlichkeit ρ, mit der ein Partikel, das aus Richtung ω_i am Punkt x eintrifft, reflektiert wird. $1 - \rho$ ist dann die Wahrscheinlichkeit, daß ein solches Partikel absorbiert wird.

Mit dieser Wahrscheinlichkeit ρ wird nun ein neuer, mit der Verteilung der bidirektionalen Reflektionsfunktion zufällig gewählter, von x ausgehender Strahl erzeugt, der auf die gleiche Weise weiterverfolgt wird. So werden die von der Lichtquelle ausgehenden Partikel auf ihrem Weg durch die Szene verfolgt, bis sie entweder an einer Oberfläche absorbiert werden, oder die Szene verlassen. Über jedes Auftreffen von Partikeln auf einer Oberfläche wird buchgeführt. Daraus ergibt sich dann das Beleuchtungsergebnis.

3.2.7.2 Globales Monte-Carlo Radiosity

Beim globalen Monte-Carlo-Radiosity [36] wird der Austausch der Strahlungsleistung in der Szene entlang globaler Linien (siehe 3.2.2.2) simuliert. Hier muß wieder angenommen werden, daß alle Oberflächen der Szene rein diffuse lambertsche Reflektoren sind.

Das Verfahren wird in zwei Stufen durchgeführt. In der ersten Stufe wird die direkte von den Lichtquellen emittierte Strahlungsleistung entlang lokaler gleichverteilter Linien auf die Oberflächen der Szene verstrahlt. In einer zweiten Stufe wird diese Strahlungsleistung dann von den in der ersten Stufe getroffenen Flächenstücken entlang globaler Linien in die Szene emittiert.

Da die Wahrscheinlichkeit eines Flächenstückes, von einer globalen Linie getroffen zu werden, zu dessen Flächeninhalt proportional ist, kann anhand der Gesamtzahl der verwendeten globalen Linien abgeschätzt werden, welche Anzahl von Treffern eines Flächenstückes durch globale Linien zu erwarten ist. Damit kann für jedes Flächenstück die Strahlungsleistung bestimmt werden, die entlang jeder durch diese Fläche gehenden globalen Linie transferiert werden muß.

An jedem Flächenstück werden nun drei Werte gespeichert: die gesamte angesammelte Strahlungsleistung (wird später in Radiosity umgerechnet), die unverteilte Strahlungsleistung und die aus der ersten Stufe rührende Strahlungsleistung, die pro globaler Linie emittiert werden muß.

Der Algorithmus läßt sich dann wie in Abbildung 3.10 dargestellt formulieren:

```
// Erste Stufe: Verteilung ausgehend von den Lichtquellen
while (weitere lokale Linien notwendig) {
    wähle eine Lichtquelle
    verfolge einen zufälligen Strahl von der Lichtquelle durch die Szene
    summiere die Strahlungsleistung am getroffenen Flächenstück auf
}

berechne für alle getroffenen Flächenstücke die zu emittierende Strahlungsleistung pro Strahl

// Zweite Stufe: Verfolgung globaler Linien
while (weitere globale Linien notwendig) {
    erzeuge einen zufälligen globalen Strahl (globale Linie)
    berechne alle seine Schnittpunkte mit der Szene
    sortiere diese Schnittpunkte nach deren Abstand vom Strahlursprung
    for (je zwei aufeinanderfolgende geschnittene Flächenstücke F_j und F_i) {
        transferiere die unverteilte Strahlungsleistung von F_j nach F_i
        transferiere die zu emittierende Strahlungsleistung von F_j nach F_i
    }
}
```

Abbildung 3.10: Der globale Monte-Carlo-Algorithmus.

Kapitel 4

Radiosity in dynamischen Szenen

Die Anwendungen, in denen das Radiosity-Verfahren für statische Szenen erfolgreich eingesetzt wird, haben in den letzten Jahren ständig zugenommen. Der Wunsch, dieses Verfahren auch zur Berechnung dynamischer Szenen einzusetzen, ist deshalb naheliegend.

Dabei handelt es sich zum einen um interaktive Anwendungen, in denen die Beleuchtungsergebnisse nach an der Szene vorgenommenen Änderungen korrigiert und dem neuen Zustand angepaßt werden müssen . Zum andern handelt es sich um Video-Sequenzen, zum Beispiel für synthetisch erzeugte Filme oder Werbeclips, in denen die Beleuchtung für ein ganzes Zeitintervall simuliert werden muß. Die Simulation für das gesamte Zeitkontinuum durchzuführen wäre aber viel zu aufwendig für einen sinnvollen Einsatz in der Praxis. Deshalb wird die Berechnung nur für diskrete Zeitpunkte, den sogenannten Frames, der Animationssequenz durchgeführt. Dies ist keine weitere Einschränkung, da Animationen ohnehin in schnell aufeinanderfolgenden Einzelbildern abgespielt werden (Beim Fernsehen oder Video sind dies 25 Bilder pro Sekunde).

Mit Hilfe der Radiosity-Methode wird ein Strahlungsgleichgewicht zwischen den einzelnen Oberflächen der Szene berechnet. Dieses Gleichgewicht hängt unter anderem von den Sichtbarkeitsverhältnissen und den Formfaktoren zwischen den einzelnen Flächenelementen ab.

Durch jede Veränderung der Szenengeometrie können sich prinzipiell beliebige Formfaktoren verändern. Die Sichtbarkeit zwischen zwei Flächenstücken kann durch ein bewegtes Objekt blockiert oder auch wieder freigegeben werden. Die dadurch verursachten Veränderungen der Formfaktor-Matrix beeinflussen die gesamte Radiosity-Lösung.

In der Praxis kann jedoch die räumliche und photometrische Kohärenz ausgenutzt werden, die zwischen aufeinanderfolgenden Frames der Animation besteht. Position, Form und auch die Beleuchtungswerte der einzelnen Objekte ändern sich in den meisten Situationen nur unwesentlich von einem Frame der Animation zum

nächsten. An manchen Stellen in der Szene mag es gravierende Änderungen zwischen aufeinanderfolgenden Frames geben, zum Beispiel bedingt durch eine Änderung der Sichtbarkeit zu primären Lichtquellen. Aber selbst dort treten längere zusammenhängende Zeitintervalle auf, in denen sich die Beleuchtung nur wenig ändert und die Kohärenz ausgenutzt werden kann.

Einzelne Veränderungen in der Szene resultieren in entfernten Regionen oft nur in vernachlässigbaren Änderungen des Beleuchtungsergebnisses. So verändert sich in einem Büroraum die Beleuchtung des Bodens, der Decke und des größten Teils der Wände nicht wahrnehmbar, wenn ein auf dem Schreibtisch liegendes Buch verschoben wird. Davon sind nur wenige Formfaktoren und die Sichtbarkeit zwischen wenigen Flächenstücken betroffen, und auch die indirekten Auswirkungen sind gering.

Deshalb kann durch die Verwendung geeigneter Algorithmen, die Kohärenzen ausnutzen und die Neuberechnung solcher Formfaktoren und Sichtbarkeiten vermeiden, die sich in einzelnen Zeitintervallen nicht ändern, sehr viel Rechenaufwand eingespart werden.

Außerdem kann die photometrische Kohärenz dazu genutzt werden, die Radiosityfunktionen auf den einzelnen Flächenstücken effizient und platzsparend zu speichern.

Die Algorithmen zur Berechnung einer Radiosity-Lösung in dynamischen Szenen können in zwei Gruppen eingeteilt werden. Solche, bei denen alle Veränderungen der Szene bereits zu Beginn der Berechnung bekannt sein müssen, und solche, die mit einer initialen Lösung starten und eine Lösung der nachfolgenden Frames dadurch erhalten, daß das Ergebnis des vorhergehenden Frames entsprechend den Veränderungen in der Szene korrigiert wird.

Die Algorithmen der ersten Gruppe sind zur Berechnung von Filmsequenzen geeignet, während interaktive Anwendungen Verfahren verlangen, die in der Lage sind die Lösung nach einer Veränderung der Szene schnell zu korrigieren. Generell können letztere auch zur Berechnung von Filmsequenzen verwendet werden, dabei treten aber aufgrund des in den einzelnen Frames der Animation unterschiedlich verteilten Restfehlers grundsätzliche Schwierigkeiten auf, die zu flackernden Animationen führen. Dies wird in Abschnitt 4.2 ausführlich beschrieben werden.

Das in Abschnitt 4.3 beschriebene Verfahren, das im Rahmen dieser Arbeit entwickelt wurde, garantiert hingegen die effiziente Berechnung konsistenter und flackerfreier Lösungen.

4.1 Bekannte Lösungsverfahren

In diesem Abschnitt werden einige Lösungsverfahren beschrieben, die in den letzten Jahren zur Berechnung von Radiosity-Lösungen in dynamischen Szenen entwickelt wurden.

4.1.1 Full-Matrix-Verfahren

Das erste Verfahren zur Berechnung einer Radiosity-Lösung in dynamischen Szenen wurde von Baum, Wallace, Cohen und Greenberg 1986 vorgestellt [49]. Dieses Verfahren setzt voraus, daß die Bewegungen aller dynamischen Objekte zu Beginn der Berechnung bekannt sind. Es arbeitet in zwei Stufen, einer Vorbearbeitungs-Stufe, die einmal zu Beginn durchgeführt wird, und einer Update-Stufe, die für jeden Frame der Animation durchgeführt werden muß.

In der Vorbearbeitungs-Stufe werden die Bewegungsvolumen aller bewegten Objekte der Szene bestimmt (oder deren Approximation durch ein einfaches umgebendes Volumen); das ist der Bereich der Szene, der von dem entsprechenden bewegten Objekt während der Animation durchlaufen wird.

Die Sichtbarkeiten von statischen Flächen werden bestimmt, indem die Szene von der Position eines im Zentrum der statischen Fläche positionierten gedachten Beobachters unter Verwendung eines z-Buffers gerendert wird. Dabei werden aber nicht Farbwerte, sondern Objekt-Ids in den Bildspeicher geschrieben. Zunächst werden nur statische Objekte gerendert. Daraus werden, noch ohne Berücksichtigung der Verdeckung durch dynamische Objekte, Basis-Formfaktoren zwischen den statischen Objekten berechnet.

Die Bewegungsvolumen werden dann in einem separaten Schritt gerendert. Daraus wird bestimmt, welche Sichtbarkeiten zwischen den statischen Objekten durch welche dynamischen Objekte blockiert werden könnten. Aus den z-Buffern der Bilder wird für jedes dynamische Objekt eine Projektionsmaske bestimmt, in der markiert ist, in welchem Pixel sich das dynamische Objekt vor oder hinter den statischen Objekten befindet.

Die Basisformfaktoren sowie die auf diese Weise von der Position aller statischen Flächen aus gerenderten Bilder und die zugehörigen Projektionsmasken der dynamischen Objekte werden unter Verwendung von RLE-Codierung (Lauflängencodierung, englisch: run length encoding) gespeichert.

Im Update-Schritt, der für jeden einzelnen Frame der Animation durchgeführt werden muß, werden dann noch die dynamischen Objekte gerendert und mit Hilfe der Projektionsmasken in die im Vorbearbeitungsschritt abgespeicherten Bilder mit den statischen Objekten eingefügt. Damit können die Formfaktoren zwischen den statischen Flächen, deren Sichtbarkeit wegen der Verdeckung durch ein dynamisches Objekt blockiert wurde, entsprechend korrigiert werden, und die Formfaktoren von den statischen zu den dynamischen Flächen können bestimmt werden.

Die Formfaktoren zwischen Paaren dynamischer Flächen werden mit Hilfe des Hemi-Cube-Verfahrens [50] bestimmt.

Dann kann die Lösung der Radiosity-Gleichung durchgeführt werden, wofür ein Full-Matrix-Verfahren verwendet wird.

Die Nachteile dieses Verfahrens sind zum einen die grundsätzlichen Probleme von Full-Matrix-Verfahren (siehe Abschnitt 3.2.3) und zum anderen der enorme

Speicherbedarf, der zur Speicherung der gerenderten Bilder und Projektionsmasken benötigt wird. Das Verfahren ist deshalb nicht zur Berechnung komplexer Szenen geeignet.

4.1.2 Progressive-Refinement-Verfahren

Von George, Sillion und Greenberg [51] und Chen [52] wurden 1990 Verfahren zur Radiosity-Redistribution vorgestellt, bei dem das Radiosity-Ergebnis einer Szene nach deren Veränderung korrigiert wird, indem die direkten Auswirkungen der Lichtquellen entsprechend den Änderungen der Szene korrigiert werden und die indirekten Korrekturen durch weitere Iterationsschritte in die Szene propagiert werden.

Ein weiteres auf Progressive-Refinement beruhendes Verfahren wurde 1994 von Schöller und Müller vorgestellt [53], das nach Änderungen der Szene eine Korrektur der bisher ausgeführten Iterationsschritte vornimmt.

4.1.2.1 Radiosity-Redistribution

Bei den von George, Sillion, Greenberg [51] und Chen [52] vorgeschlagenen Methoden wird die Radiosity-Lösung einer Szene nach deren Änderung korrigiert, indem die Änderungen in nachfolgenden Iterationsschritten durch die Szene verteilt werden.

Bei Änderungen der Szenengeometrie werden die Änderungen, die sich aufgrund der Auswirkungen der direkten Lichtquellen ergeben, berechnet und diese Änderungen (in Form von positiven oder negativen Radiosity-Werten) zur unverteilten Strahlungsleistung der betroffenen Flächen addiert. Auch Änderungen von Reflektionskoeffizienten oder Lichtemissionen können zur unverteilten Radiosity der entsprechenden Flächenstücke addiert werden. Läßt man nun bei der Berechnung auch negative Strahlungsleistung zu, verteilen sich die durchgeführten Änderungen bei Ausführung weiterer Iterationsschritte in die Szene.

Änderungen, die sich aufgrund von Sichtbarkeitsänderungen zwischen Empfängerflächen und indirekt abstrahlenden Flächenstücken ergeben (das sind Flächenstücke, die in einem Iterationsschritt als abstrahlende Fläche ausgewählt werden und dabei von anderen Flächen oder Lichtquellen eingestrahltes Licht in die Szene reflektieren), werden mit dieser Methode aber nicht korrekt durchgeführt.

4.1.2.2 Radiosity-Repropagation mit SFFL

In dem 1994 von Schöller und Müller vorgestellten Verfahren [53] werden bei der Berechnung der Progressive-Refinement-Lösung der Ausgangsszene die Informationen gespeichert, die später bei einer Änderung der Szene notwendig sind, um die durchgeführten Iterationsschritte entsprechend der Änderung zu korrigieren.

Dazu wird für jeden Iterationsschritt eine *Schatten-Formfaktor-Liste* (SFFL) angelegt, in der zu jedem Flächenstück F_i der Szene in einem Flag gespeichert wird,

ob Sichtbarkeit zum abstrahlenden Flächenstück F_j vorliegt oder nicht. Im Fall der Sichtbarkeit wird außerdem der berechnete Formfaktor gespeichert und andernfalls die ID des verdeckenden Objektes.

In einer globalen *Shooting Liste* (SL) wird für jeden Iterationsschritt der Berechnung die ID j des abstrahlenden Flächenstücks F_j und die in diesem Iterationsschritt verwendete unverteilte Radiosity ΔB_j gespeichert, außerdem ein Zeiger auf den ersten und auf den nächsten Listeneintrag, bei dem das selbe Flächenstück F_j als abstrahlendes Flächenstück benutzt wurde, sowie ein Zeiger auf die jeweilige Schatten-Formfaktor-Liste. Desweitern gibt es für jeden Iterationsschritt ein Feld, in dem für spätere Korrekturdurchgänge die Änderungen der unverteilten Radiosity ΔB_j abgelegt werden können. (siehe Abbildung 4.1).

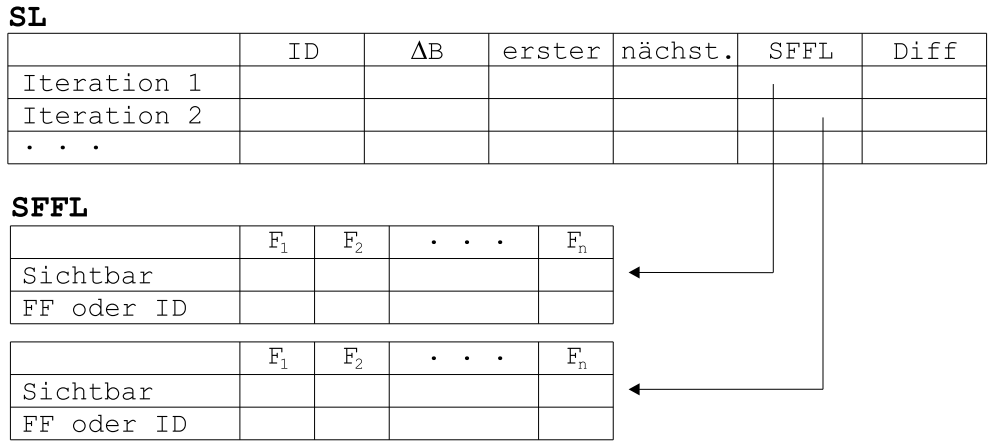


Abbildung 4.1: Shooting-Liste und Schatten-Formfaktor-Listen.

Änderungen der Lichtemissionen oder Reflektionskoeffizienten können mit Hilfe dieser Liste direkt korrigiert werden. Bei Änderung der Szenengeometrie werden zunächst die primären Auswirkungen dieser Änderung korrigiert. Dies sind die Änderung der Radiosity-Werte der bewegten Objekte, die Entfernung deren Schatten an der ursprünglichen Position und das Einfügen der Schatten an der neuen Position.

Zur Änderung der Radiosity-Werte der bewegten Flächenstücke müssen die entsprechenden Formfaktoren und Sichtbarkeiten neu berechnet und in den Schatten-Formfaktor-Listen geändert werden. Die Radiosity und die unverteilte Radiosity kann dann anhand der Shooting-Liste und der Schatten-Formfaktor-Listen neu bestimmt werden.

Zur Entfernung der Schatten des bewegten Objektes an seiner ursprünglichen Position kann anhand der Schatten-Formfaktor-Listen identifiziert werden, welche Verdeckungen durch das bewegte Objekt verursacht wurden. Die Sichtbarkeiten der betroffenen Flächenstücke muß dann neu getestet werden. Tritt Verdeckung aufgrund eines anderen Objektes auf, muß die ID des neuen Verdeckers in die Schatten-Formfaktor-Liste eingetragen werden, andernfalls muß der Formfaktor berechnet und eingetragen und die Radiosity korrigiert werden.

Die Flächenstücke, deren Sichtbarkeit zu den abstrahlenden Flächenstücken der einzelnen Iterationsschritte möglicherweise durch das Objekt an seiner neuen Position verdeckt sein könnten, werden mit Hilfe von Schattenvolumen ermittelt, die sich aus der Position des abstrahlenden Flächenstückes und der neuen Position des Objektes ergeben. Die Sichtbarkeit zu diesen so ermittelten Flächenstücken wird dann neu getestet und die Einträge in den Schatten-Formfaktor-Listen sowie die betroffenen Radiosity-Werte werden entsprechend korrigiert.

Falls es sich bei dem veränderten Objekt um eine Lichtquelle handelt oder um ein Flächenstück, das bereits in einem Iterationsschritt als abstrahlendes Flächenstück ausgewählt wurde, müssen die entsprechenden Schatten-Formfaktor-Listen und die daraus resultierenden Radiosity-Werte neu berechnet werden.

Nun müssen auch die indirekten Auswirkungen der bisherigen Korrekturen korrigiert werden, die sich in nachfolgenden Iterationsschritten ergeben, in denen ein Flächenstück als abstrahlendes Flächenstück gewählt wurde, dessen Radiosity-Wert geändert wurde. Zu diesem Zweck wird bei jeder Änderung eines Radiosity-Wertes, die sich während der oben beschriebenen Korrektur der direkten Auswirkungen ergibt, anhand der Shooting-Liste geprüft, ob das geänderte Flächenstück in einem späteren Iterationsschritt als abstrahlendes Flächenstück gewählt wurde. Ist dies der Fall, wird die Änderung der unverteilten Strahlungsleistung dieses Flächenstücks in das Feld *Diff* der Shooting-Liste eingetragen. Nach Abschluß aller direkten Änderungen werden die im Feld *Diff* eingetragenen Änderungen der in den einzelnen Iterationsschritten verwendeten Emissionen anhand der Schatten-Formfaktor-Listen korrigiert.

Mit dieser Methode können alle bisher durchgeführten Iterationsschritte einer Progressive-Refinement-Berechnung bei Änderungen der Szene korrigiert werden. Die Folge der durchgeführten Progressive-Refinement-Schritte bleibt dabei unverändert. In manchen Situationen ist es aber notwendig, daß nach einer Änderung der Szene das Licht, das von einem Flächenstück reflektiert wird, das bisher nicht als abstrahlendes Flächenstück ausgewählt worden war, für entscheidende Beleuchtungseffekte verantwortlich ist (Bei der in Abbildung 4.4 dargestellten Szene wäre dies zum Beispiel der Fall). In solchen Fällen müssen nach der durchgeführten Korrektur weitere Iterationsschritte durchgeführt werden.

4.1.3 Hierarchisches Radiosity

Auch zur Berechnung von hierarchischen Radiosity-Lösungen für dynamische Szenen wurden in den letzten Jahren Verfahren entwickelt. Forsyth, Yang und Teo stellen in ihrer Arbeit [54] eine Methode vor, die es erlaubt, Links zu bewegten Objekten in der Hierarchie abhängig von deren neuer Position nach oben oder unten zu verschieben.

Ein erster Ansatz für das Problem der Sichtbarkeitsänderungen durch bewegte Objekte wurde von Shaw [55] präsentiert. Er führte sogenannte Schattenlinks ein, in denen Informationen über Verdecker gespeichert werden. Bei der Bewegung eines Objektes wird mit Hilfe der Begrenzungsebenen der beiden Positionen des Objektes

ein Bewegungsvolumen bestimmt, mit dem alle Links auf Schnittpunkte getestet werden, um festzustellen, ob sie sich durch die Bewegung geändert haben.

In [56] werden Algorithmen vorgestellt, die nach der Bewegung eines Objektes Links zwischen den Flächen korrigieren und sich um die Auflösung der Netze kümmern.

In [57] präsentieren die Autoren einen Algorithmus für hierarchisches Radiosity und Clustering, mit dem Radiosity-Lösungen nach Veränderungen in der Szene korrigiert werden können. Sie führen eine Linien-Raum-Hierarchie ein, anhand derer die Links zwischen hierarchischen Elementen (das heißt Flächenstücken oder Cluster) schnell identifiziert werden können, die von Änderungen der Szene betroffen sind. Außerdem stellt dieser Algorithmus einen Mechanismus zur Kontrolle der Framerate zur Verfügung, der es erlaubt, vorgegebene Frameraten unter Reduzierung der Qualität zu erreichen.

Beim hierarchischen Radiosity (siehe 3.2.5) werden die Flächenelemente abhängig von der Größe der Formfaktoren hierarchisch unterteilt. Anschließend werden zwischen den so erhaltenen Flächenstücken Links erzeugt, entlang derer die Formfaktoren berechnet und somit der Energieaustausch simuliert wird.

In dem in [57] präsentierten Algorithmus werden auch die Links in den höheren Hierarchiestufen gehalten und als passiv markiert. So wird zusätzlich zur Hierarchie der Flächen und Cluster eine Hierarchie von Links, die Linien-Raum-Hierarchie, gespeichert. An jedem Link wird außerdem der zugehörige Korridor gespeichert. Dies ist das Volumen, das zwischen den durch diesen Link verbundenen Flächenstücken liegt (siehe Abbildung 4.2). Diese Korridore erlauben schnelle Schnittpunkttests mit Quadern, was eine zentrale Operation in diesem Algorithmus ist.

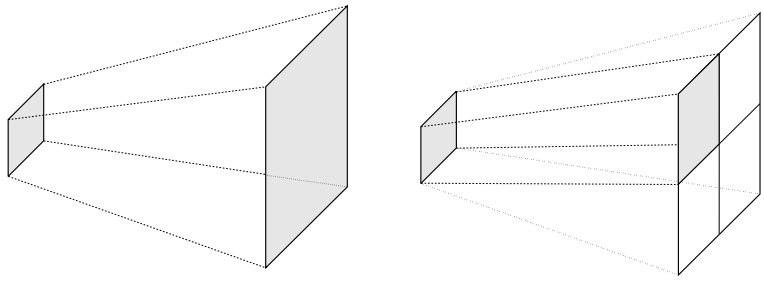


Abbildung 4.2: Korridor: Volumen zwischen zwei durch einen Link verbundenen Flächenstücken.

Wird nun ein Objekt der Szene bewegt, werden zunächst die Links bestimmt, die von dieser Bewegung betroffen sein könnten. Dies geschieht durch Traversieren der Linien-Raum-Hierarchie.

Zuerst werden dabei alle passiven Links besucht. Bei der Überprüfung wird zunächst der zugehörige Korridor mit dem umschließenden Quader des bewegten Objektes in seiner alten und in seiner neuen Position auf Schnittpunkte getestet. Wird ein Schnitt festgestellt, ist dieser und die in der Hierarchie unterhalb liegenden Links möglicherweise betroffen. In diesem Fall wird zunächst überprüft, ob das

Unterteilungskriterium, das zu diesem Link geführt hat, in der geänderten Situation noch greift. Ist dies nicht der Fall, wird der Link deinstalliert und seine Kinder gelöscht. Andernfalls werden rekursiv die Kinder überprüft.

Anschließend werden alle aktiven Links überprüft und diejenigen, die von der Änderung der Szene betroffen sind, in einer Liste gespeichert.

Nun werden die in der Liste gespeicherten Links korrigiert. Dazu muß die Verdeckung neu bestimmt werden und eventuell müssen die Links weiter unterteilt werden. Zum Schluß wird dann die Radiosity-Lösung korrigiert. Möglicherweise müssen auch noch weitere Iterationsschritte der Berechnung durchgeführt werden.

Eine Kontrolle der erreichten Framerate kann dadurch erreicht werden, daß Unterteilungen nur durchgeführt werden, wenn dadurch die zur Verfügung stehende Rechenzeit nicht überschritten wird. Die Unterteilung der in der Liste gespeicherten Links wird dabei in der Reihenfolge der entlang dieser Links ausgetauschten Strahlungsleistung durchgeführt. Dadurch wird erreicht, daß die wichtigen Unterteilungen in jedem Fall durchgeführt werden.

4.1.4 Monte-Carlo Radiosity

Besuievsky und Sbert stellten 1996 einen Algorithmus zur Berechnung einer globalen Monte-Carlo-Lösung (siehe Abschnitt 3.2.7.2) in dynamischen Szenen vor [58]. Der Algorithmus setzt voraus, daß alle Änderungen der Szene schon vor Beginn der Berechnung bekannt sind.

Zu Beginn wird die gesamte Geometrie- und Lichtquellen-Information aller Frames der Animationssequenz zu einer einzigen Szene zusammengefaßt. In dieser Szene besitzt jedes dynamische Objekt für jeden Frame der Animationssequenz eine Instanz, die mit dem zugehörigen Frame markiert ist. Werden Schnittpunktsberechnungen einer lokalen oder globalen Linie mit dieser Szene durchgeführt, werden Schnittpunkte mit bewegten Objekten für jeden Frame berechnet und entlang der Linie mit der Markierung des zugehörigen Frames gespeichert.

Wie schon im statischen Fall wird zunächst in einer ersten Stufe die direkte von den Lichtquellen emittierte Strahlungsleistung entlang lokaler Linien auf die Oberflächen der Szene verstrahlt, die dann in der zweiten Stufe von den Empfängerflächen in die Szene emittiert wird. Aus diesem Grund müssen die Lichtquellen während der gesamten Animation unverändert bleiben.

Nach der Durchführung der ersten Stufe wird für jedes Flächenstück die Strahlungsleistung bestimmt, die entlang jeder globalen Linie in die Szene emittiert werden muß.

In der zweiten Stufe werden globale Linien erzeugt und deren Schnittpunkte mit den Oberflächen der Szenenobjekte bestimmt. Die so erhaltene Liste von Schnittpunkten wird nun in einzelne unabhängige Listen für jeden Frame der Animationssequenz aufgespalten. Damit wird nun für jeden Frame getrennt in einer eigenen Szene der Austausch der Strahlungsleistung vollzogen.

Der Algorithmus läßt sich wie in Abbildung 4.3 dargestellt formulieren

```
// Vorbearbeitungsschritt
erzeuge eine Szene, die alle Objekte aller Frames enthält.

// Erste Stufe: Verteilung ausgehend von den Lichtquellen
while (weitere lokale Linien notwendig) {
    wähle eine Lichtquelle
    verfolge einen zufälligen Strahl von der Lichtquelle durch die Szene
    for (alle Frames)
        finde das Flächenstück, das zuerst getroffen wird
        summiere die Strahlungsleistung am getroffenen Flächenstück auf
    }
}

berechne für alle getroffenen Flächenstücke die zu emittierende Strahlungsleistung pro Strahl

// Zweite Stufe: Verfolgung globaler Linien
while (weitere globale Linien notwendig) {
    erzeuge einen zufälligen globalen Strahl (globale Linie)
    berechne alle seine Schnittpunkte mit der Szene
    for (alle Frames)
        bestimme Liste der Schnittpunkte in diesem Frame
        sortiere diese Schnittpunkte nach deren Abstand vom Strahlursprung
        for (je zwei aufeinanderfolgende geschnittenen Flächenstücke F_j und F_i) {
            transferiere die unverteilte Strahlungsleistung von F_j nach F_i
            transferiere die zu emittierende Strahlungsleistung von F_j nach F_i
        }
    }
}
```

Abbildung 4.3: Der globale Monte-Carlo-Algorithmus für dynamische Szenen.

4.2 Progressive-Refinement in Animationen

Bei der Verwendung des Progressive-Refinement Algorithmus (siehe Abschnitt 3.2.4) wird in jedem Iterationsschritt eine Fläche größter Strahlungsleistung als verteilende Fläche gewählt. Deshalb ist die Konvergenzrate am Anfang des Berechnungsprozesses sehr hoch. Später, wenn sich die Lösung der exakten Lösung nähert, nimmt die Konvergenzrate stark ab. In praktischen Situationen genügt es aber fast immer, sich auf wenige Iterationsschritte zu beschränken, um gute visuelle Ergebnisse zu erhalten, auch wenn der absolute Fehler dann noch relativ groß ist.

Aus diesem Grund eignet sich das Progressive-Refinement-Verfahren auch zur Berechnung komplexer Szenen. Nur einige Flächen, die für den optischen Eindruck

wichtige Beiträge liefern, müssen ihre Strahlungsleistung in die Szene verteilen. Das heißt, nur einige Spalten der Formfaktor-Matrix müssen berechnet werden. Man erhält gute, photorealistische Ergebnisse, aber der absolute Fehler ist relativ groß.

Wird eine bereits berechnete Szene verändert, ist es nicht immer ausreichend, die bereits durchgeführten Iterationsschritte entsprechend der Änderungen der Szene zu korrigieren, um eine gute Lösung für die veränderte Szene zu erhalten. Flächen, die für die ursprüngliche Lösung nicht relevant waren und deshalb ihre Strahlungsleistung nicht in die Szene verteilt haben, können nach der Veränderung plötzlich stark beleuchtet sein, so daß das indirekte Licht, das eine solche Fläche verläßt, für wichtige Beleuchtungseffekte ausschlaggebend ist. Deshalb ist es oft notwendig, noch weitere Iterationsschritte durchzuführen, um diese Effekte sichtbar zu machen.

Andererseits erhalten wir aber unterschiedliche absolute Fehler in den einzelnen Frames einer Animation, wenn die Folge der Iterationsschritte nicht in allen Frames die selbe ist. Dies führt zu einer flackernden Animation, was für interaktive Anwendungen sicher kein Problem darstellt, für Video-Animationen aber nicht akzeptabel ist. Das selbe Problem tritt auch auf, wenn alle Frames der Animation unabhängig voneinander berechnet werden, da die Folge der Iterationsschritte auch hier in den einzelnen Frames unterschiedlich sein kann. Dieses Problem tritt bei allen in den Abschnitten 4.1.2 und 4.1.3 beschriebenen Algorithmen auf.

Wenn jedoch alle Veränderungen der Szene zu Beginn der Berechnung bekannt sind, kann eine Folge von Iterationsschritten gefunden werden, die für alle Frames der Animation gute Ergebnisse liefert. Werden alle Frames mit dieser Folge berechnet (siehe Abschnitt 4.3), erhält man eine glatte, flackerfreie Animation.

Dies wird in Abbildung 4.4 illustriert. Der rote Quader befindet sich zu Beginn der Animation nahe der rechten Wand und bewegt sich dann während einer aus 71 Frames bestehenden Animation nach links, bis unter die Lichtquelle, die sich in der Mitte des Raumes in der Nähe der Decke befindet. Die obere Fläche des Quaders ist in den ersten Frames der Animation fast unbeleuchtet und wird deshalb bei der Berechnung nicht als indirekte Lichtquelle ausgewählt. In den letzten Frames ist diese Fläche aber sehr stark beleuchtet und bewirkt durch Reflektion des einfallenden Lichtes, daß an der Decke ein roter Fleck sichtbar wird. In diesen Frames ist es also wichtig, daß die Fläche als indirekte Lichtquelle ausgewählt wird.

In der ersten Zeile ist das Ergebnis dargestellt, das man erhält, wenn alle Frames unabhängig voneinander berechnet werden. In jedem Frame wurden 20 indirekte Iterationsschritte durchgeführt, bei denen jeweils eine komplette Fläche (zum Beispiel der gesamte Boden) mit adaptiver Abtastung ihre Strahlungsleistung verteilte.

In den Frames 1 bis 42 wird die obere Fläche des roten Quaders nicht als indirekte Lichtquelle ausgewählt. Deshalb fehlt hier die rote Reflektion an der Decke. In Frame 43 taucht diese Reflektion dann plötzlich auf, da hier die obere Fläche des roten Quaders im letzten Iterationsschritt als indirekte Lichtquelle ausgewählt wird. In Frame 44 sind alle Flächen der Szene leicht rötlich. Die obere Fläche des Quaders wurde hier schon in einem früheren Iterationsschritt als indirekte Lichtquelle

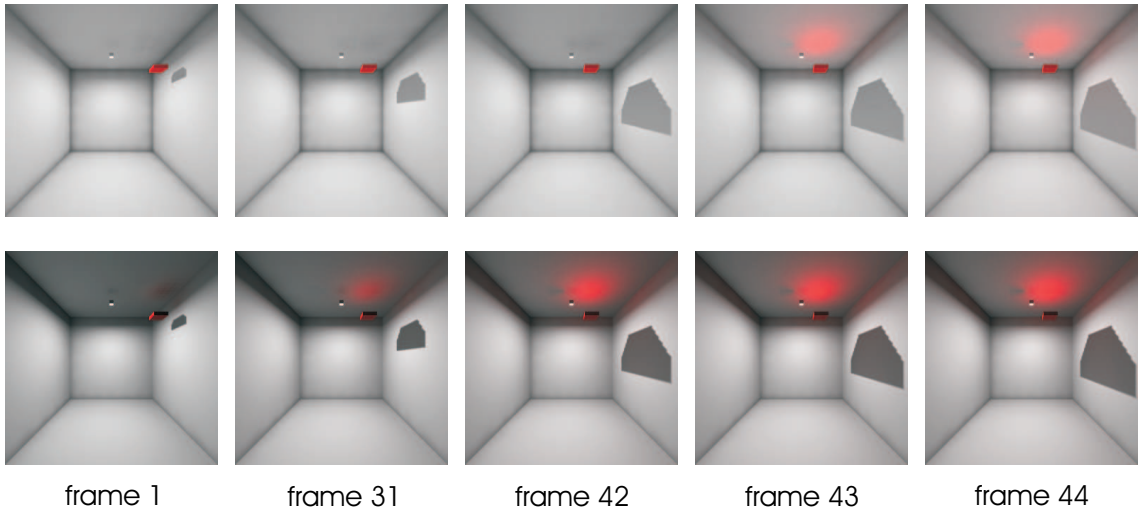


Abbildung 4.4: Erste Zeile: die unabhängige Berechnung führt trotz der Durchführung von 20 indirekten Iterationsschritten zu einem flackernden Ergebnis. Zweite Zeile: die simultane Berechnung aller Frames liefert eine glatte, flackerfreie Animation. Obwohl der Iterationsprozeß bereits nach 5 indirekten Schritten noch bei großem absolutem Fehler abgebrochen wurde, ist der visuelle Eindruck bereits ganz gut.

ausgewählt und der rote Fleck an der Decke in einem nachfolgenden Schritt weiter in die Szene reflektiert.

Die Bildfolge in der zweiten Zeile kennt kein solch plötzliches Auftreten von Beleuchtungseffekten, die im vorhergehenden Frame noch nicht vorhanden waren. Diese Bildfolge wurde mit dem simultanen Progressive-Refinement Algorithmus berechnet. Obwohl in diesem Beispiel nicht wie oben 20, sondern nur 5 indirekte Iterationsschritte durchgeführt wurden, ist das Ergebnis wesentlich besser als bei unabhängiger Berechnung. Die rote Reflektion wird schon vom ersten Frame an völlig weich eingeblendet. Da die Iteration in der 2. Zeile schon sehr früh abgebrochen wurde, sind die berechneten Beleuchtungswerte hier dunkler als bei den Berechnungen in der ersten Zeile. Deshalb wurden die Bilder in der zweiten Zeile mit dem Faktor 1.7 skaliert.

Ähnliche, nicht ganz so auffällige aber trotzdem störende Effekte treten zum Beispiel aufgrund des Bodens auf, der im ersten Frame bereits im ersten Iterationsschritt als indirekte Lichtquelle ausgewählt wird, im letzten Frame aber erst in einem späteren Schritt. Irgendwo dazwischen springt die Auswahl von einem Frame zum darauffolgenden vom ersten Iterationsschritt zu einem späteren, was eine Unstetigkeit des Beleuchtungsergebnisses bewirkt.

4.3 Simultanes Progressive-Refinement

In diesem Kapitel wird ein neues Verfahren zur simultanen Berechnung einer Radiosity-Lösung für Animations-Sequenzen vorgestellt [6, 7]. Alle Änderungen der Szene müssen bereits zu Beginn der Berechnung bekannt sein. Deshalb ist das Verfahren nicht für interaktive Anwendungen geeignet. Es garantiert aber eine konsistente und flackerfreie Lösung, was für Filmsequenzen essentiell wichtig ist. Das Verfahren basiert auf der Progressive-Refinement Methode von Wallace [30] und ist in der Lage, Szenen mit sich ändernder Geometrie, Materialien und Lichtemissionen zu berechnen. Das Berechnungsergebnis ist noch unabhängig von Kamerapositionen oder -pfaden, die für die spätere Erzeugung einer Filmsequenz verwendet werden sollen.

4.3.1 Der Algorithmus

Der Algorithmus geht von einer Szenenbeschreibung aus, in der jedes Objekt der Szene seine Bewegungen und Verformungen während der gesamten Animations-Sequenz kennt. Die Progressive-Refinement Berechnung wird dann für alle Frames der Animation gleichzeitig durchgeführt, um ein konsistentes und flackerfreies Ergebnis zu erreichen.

Sei nun n die Anzahl der Flächenstücke, aus denen die Szene besteht, und m die Anzahl der Frames der Animation. Dann bezeichne $F_i,\ i = 1, \ldots, n$ das i-te Flächenstück und $P_{i,f},\ f = 1, \ldots, m$ die unverteilte Strahlungsleistung von F_i im f-ten Frame.

Die Fläche F_i heißt eine *Fläche größter Strahlungsleistung*, falls

$$\max_{f=1,\ldots,m} P_{i,f} \geq \max_{\substack{f=1,\ldots,m \\ j=1,\ldots,n}} P_{j,f}. \tag{4.1}$$

Für jeden Progressive-Refinement Schritt wird nun eine Fläche L größter Strahlungsleistung ausgewählt, die dann gleichzeitig für alle Frames der Animations-Sequenz ihre unverteilte Strahlungsleistung in die Szene verteilt.

Handelt es sich bei dieser Fläche L um eine dynamische Fläche, so müssen die Sichtbarkeits- und Formfaktorberechnungen für jeden Frame einzeln durchgeführt werden. Handelt es sich dagegen um eine statische Fläche, bleiben die unverdeckten Formfaktoren zu statischen Empfängerflächen während der gesamten Animations-Sequenz konstant. Auch die Sichtbarkeitsberechnung zu statischen Empfängerflächen kann für alle Frames gleichzeitig durchgeführt werden, indem Strahlen von den Empfänger-Vertices zu den Lichtquellensamples geschickt werden, deren Start- und Endpunkte für die gesamte Animations-Sequenz konstant bleiben. Mit Hilfe dieser Strahlen kann dann festgestellt werden, in welchen Frames der Animation Verdeckung und in welchen Sichtbarkeit vorliegt.

Der Algorithmus ist in Abbildung 4.5 in C-ähnlichem Pseudocode dargestellt:

```
while (weitere Iterationen notwendig) {
  wähle eine Fläche L größter Strahlungsleistung
  if (L ist dynamisch)
    verteile die unverteilte Radiosity von L für alle Frames einzeln
  else          // L ist statisch
    verteile die unverteilte Radiosity von L für alle Frames gleichzeitig
}
```

Abbildung 4.5: Simultanes Progressive-Refinement.

Im ersten Fall, wenn L eine dynamische Fläche ist, kann die Verteilung der unverteilten Radiosity wie in Abbildung 4.6 beschrieben, formuliert werden:

```
for (alle Frames f, Flächen F, Vertices v von F und Samples s von L)
{
  verfolge einen Strahl v → s durch die Szene
  if (Sichtbarkeit) {
    berechne den Formfaktor
    übertrage die Radiosity vom s nach v
  }
}
```

Abbildung 4.6: Radiosity-Verteilung für eine dynamische Lichtquelle.

Im zweiten Fall, wenn L eine statische Fläche ist, wird die Radiosity-Verteilung wie in Abbildung 4.7 beschrieben durchgeführt.

In Falle einer statischen ausstrahlenden Fläche L und einer statischen Empfängerfläche F wird die Sichtbarkeit mit Hilfe eines einzigen Strahls $R : v \rightarrow s$ für alle Frames der Animation gleichzeitig bestimmt.

Für diesen Zweck wird jeder Strahl $R : v \rightarrow s$ mit einer Sichtbarkeitsmaske versehen, die für jeden Frame der Animation ein Flag zur Markierung von Verdeckung oder Sichtbarkeit enthält, d.h. ein Flag wird auf Eins gesetzt, wenn in dem zugehörigen Frame Sichtbarkeit vorliegt, und andernfalls auf Null. Alle Flags werden mit Eins initialisiert und auf Null gesetzt, sobald ein verdeckendes Objekt gefunden wird.

Bei der Verfolgung eines Strahls R durch die Szene sind Schnittpunkt-Tests mit statischen Objekten für alle Frames der Animation gültig, während Schnittpunkt-Tests mit dynamischen Objekten für jeden Frame einzeln durchgeführt werden müssen, sofern der Strahl in dem entsprechenden Frame nicht schon vorher aufgrund eines anderen verdeckenden Objektes als verdeckt markiert wurde. Sind alle Frames als verdeckt markiert, kann der Prozeß der Strahlverfolgung abgebrochen werden. Mit Hilfe von raumunterteilenden Verfahren kann vermieden werden, daß Schnittpunktsberechnungen mit dynamischen Objekten immer für alle Frames der Animation durchgeführt werden müssen. Sie können damit auf solche Frames beschränkt

```
for (alle Flächen F) {
  if (F ist dynamisch) {
    for (alle Frames f, alle Vertices v von F und alle Samples s von L) {
      verfolge einen Strahl v → s durch die Szene
      if (Sichtbarkeit) {
        berechne den Formfaktor
        übertrage die Radiosity von s nach v
      }
    }
  } else { // F ist statisch
    for (alle Vertices v von F und alle Samples s von L) {
      verfolge einen Strahl v → s durch die Szene
      if (Sichtbarkeit in mindestens einem Frame) {
        berechne den Formfaktor
        for (alle Frames, in denen Sichtbarkeit vorliegt)
          übertrage die Radiosity von s nach v
      }
    }
  }
}
```

Abbildung 4.7: Radiosity-Verteilung für eine statische Lichtquelle.

werden, in denen sich das dynamische Objekt in einer kleinen Umgebung um den Strahl befindet (siehe Abschnitt 4.3.3).

In dem in Abbildung 4.8 dargestellten Beispiel wird ein Strahl von einem Empfänger-Vertex v zu einem Sample s der abstrahlenden Fläche durch eine dynamische Szene verfolgt. Die Animations-Sequenz besteht aus 5 Frames. Die Sichtbarkeitsmaske des Strahls wird für alle Frames mit 1 initialisiert, d.h zunächst haben wir Sichtbarkeit in allen Frames (a). Der Schnittpunkttest mit dem statischen Objekt O_1 muß nur einmal durchgeführt werden und liefert als Ergebnis, daß dieses Objekt den Strahl in keinem der Frames verdeckt (b). Der Schnittpunkttest mit dem rechteckigen dynamischen Objekt O_2 liefert Verdeckung in den Frames zwei und drei, was in der Sichtbarkeitsmaske des Strahls markiert wird, indem die Flags zwei und drei auf Null gesetzt werden (c). Beim Schnittpunkttest mit dem ovalen dynamischen Objekt O_3 können die Frames zwei und drei ignoriert werden, da diese schon wegen eines anderes Objekts als verdeckt markiert wurden. Frame vier wird nun zusätzlich als verdeckt markiert (d). Die Sichtbarkeitsmaske am Ende des Raytracing-Prozesses (e) zeigt an, daß im ersten und fünften Frame Sichtbarkeit vorliegt und in den Frames zwei, drei und vier verdeckende Objekte gefunden wurden.

Anschließend wird der Formfaktor berechnet und die Radiosity wird in den Frames eins und fünf von s nach v übertragen.

Mit Hilfe dieses Algorithmus muß der Aufwand der Sichtbarkeitstests und der Formfaktorberechnung zwischen statischen Flächen nur einmal für die gesamte Animations-Sequenz betrieben werden. Deshalb kann für Szenen mit einem hohen Anteil

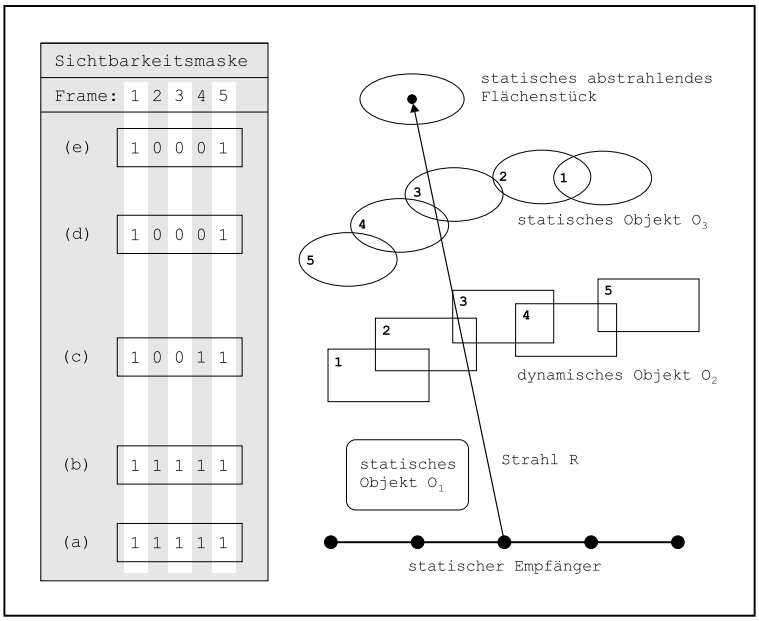


Abbildung 4.8: Simultanes Ray Tracing mit Sichtbarkeitsmasken. (a)–(e): Sichtbarkeits-Status des Strahls R

an statischen Objekten gute Performance erwartet werden.

Ein Nachteil dieses Verfahrens ist die Tatsache, daß die Radiosity-Werte der Flächen gleichzeitig in allen Frames der Animations-Sequenz zur Berechnung benötigt werden, was einen sehr großen Speicherbedarf für lange Animations-Sequenzen bedeutet. Aber glücklicherweise können zur Speicherung der Radiosity-Werte effiziente Darstellungen herangezogen werden, die den Speicherbedarf durch Ausnutzung von photometrischer Kohärenz drastisch reduzieren, so daß das Verfahren auch für lange und komplexe Animations-Sequenzen praktikabel ist (siehe Abschnitt 4.3.6).

4.3.2 Konvergenz

In diesem Abschnitt wird zunächst das Konzept der Relaxation dargestellt, das dann im folgenden zum Beweis der Konvergenz des Verfahrens herangezogen werden wird. Der Konvergenzbeweis, der hier gegeben wird, ist ähnlich zu dem, der in [38] für das Progressive-Refinement-Verfahren in statischen Szenen gegeben wird. Weitere Details über Relaxation finden sich auch in [59] und [60].

4.3.2.1 Relaxation

Betrachten wir das lineare Gleichungssystem

$$Mx = b$$

wo M eine $n \times n$ Matrix ist und $x, b \in \mathbb{R}^n$. Um dieses Gleichungssystem nach x zu lösen, soll eine iterative Methode verwendet werden, die mit einem Startwert $x^{(0)}$ startet.

Für eine approximative Lösung $x^{(k)} = (x_1^{(k)}, \ldots, x_n^{(k)})$, die man nach dem k-ten Iterationsschritt erhält, sei der k-te Fehler $e^{(k)}$ and das k-te Residuum $r^{(k)}$ definiert als

$$e^{(k)} := x - x^{(k)} \quad und \quad r^{(k)} := b - Mx^{(k)} = Me^{(k)}.$$

Wenn die Residuen $r^{(k)} = (r_1^{(k)}, \ldots, r_n^{(k)})$ für $k \to \infty$ gegen Null konvergieren, dann konvergiert $x^{(k)}$ gegen die Lösung des Gleichungssystems.

Relaxation ist eine Methode, die dies zu erreichen versucht. Die Idee ist, in jedem Iterationsschritt eine Variable $x_i^{(k)}$ zu relaxen, das heißt so zu verändern, daß das Residuum $r_i^{(k+1)} = 0$ ist. Dabei werden andere $r_j^{(k)}$ möglicherweise vergrößert. Wenn jedoch die Variablen, die relaxt werden sollen, geschickt gewählt werden, kann Konvergenz des Verfahrens erreicht werden (siehe unten und 4.3.2.2).

Wenn die Variable $x_i^{(k)}$ relaxt werden soll, muß

$$x_i^{(k+1)} := \frac{b_i - \sum_{j \neq i} M_{ij} x_j^{(k)}}{M_{ii}} \tag{4.2}$$

gesetzt werden, denn nach der Definition von $r^{(k)}$ gilt

$$r_i^{(k)} = b_i - \sum_j M_{ij} x_j^{(k)}, \tag{4.3}$$

und damit

$$\begin{aligned} r_i^{(k+1)} &= b_i - \sum_j M_{ij} x_j^{(k+1)} && (4.4)\\ &= b_i - \sum_{j \neq i} M_{ij} x_j^{(k)} \; - \; M_{ii}\; x_i^{(k+1)} && (4.5)\\ &= 0\ . && (4.6) \end{aligned}$$

Aus (4.2) und (4.3) ergibt sich

$$x_i^{(k+1)} = x_i^{(k)} + \frac{r_i^{(k)}}{M_{ii}}.$$

Durch Anwendung der Gleichung (4.3) auf $r_j^{(k)}$ und $r_j^{(k+1)}$, erhalten wir für das neue Residuum die Gleichung

$$r_j^{(k+1)} = \begin{cases} r_j^{(k)} - \frac{M_{ji}}{M_{ii}} r_i^{(k)} & \text{für } j \neq i, \\ 0 & \text{für } j = i. \end{cases} \tag{4.7}$$

Wenn die $x_i^{(k)}$ in der Reihenfolge der Indizes relaxed werden, erhält man das wohlbekannte Iterationsverfahren von Gauss-Seidel, das für diagonal dominante Matrizen gegen die richtige Lösung konvergiert. Die Methode, bei der in jedem Iterationsschritt die Variable $x_i^{(k)}$ mit dem betragsgrößten Residuum $r_i^{(k)}$ relaxed wird, heißt *Southwell Relaxation*

In [38] wird gezeigt, daß diese Methode für strikt spalten-diagonal-dominante Matrizen konvergiert, indem gezeigt wird, daß das totale Residuum $r^{(k)}$ in jedem Iterationsschritt um einen konstanten Faktor abnimmt.

4.3.2.2 Konvergenzbeweis

In diesem Abschnitt wird ein formaler Konvergenzbeweis des oben beschriebenen Verfahrens gegeben.

Die Radiosity-Lösung in einer statischen Szene erhält man nach Gleichung (3.9) durch Lösung des linearen Gleichungssystems

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ji} \frac{A_j}{A_i}, \quad i = 1, \ldots, n$$

wo n die Anzahl der Flächenstücke in der Szene ist und

B_i = die Radiosity des Flächenstücks F_i
E_i = die Emission des Flächenstücks F_i
A_i = der Flächeninhalt des Flächenstücks F_i
ρ_i = der Reflektionskoeffizient des Flächenstücks F_i
F_{ij} = der Formfaktor von Flächenstück F_i zu Flächenstück F_j.

Mit Hilfe der reziproken Beziehung $F_{ji}A_j = F_{ij}A_i$ erhalten wir

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij}, \quad i = 1, \ldots, n. \tag{4.8}$$

Anstatt des Gleichungssystems (4.8) kann, wiederum mit Hilfe der reziproken Beziehung, das äquivalente Gleichungssystem

$$B_i A_i = E_i A_i + \rho_i \sum_{j=1}^{n} B_j A_j F_{ji}, \quad i = 1, \ldots, n \tag{4.9}$$

gelöst werden, oder in Matrix-Form

$$M \; B = \tilde{E},$$

wobei

$$M = \begin{bmatrix} A_1 - \rho_1 A_1 F_{11} & \ldots & -\rho_1 A_n F_{n1} \\ -\rho_2 A_1 F_{12} & A_2 - \rho_2 A_2 F_{22} & \\ \vdots & & \vdots \\ -\rho_n A_1 F_{1n} & \ldots & A_n - \rho_n A_n F_{nn} \end{bmatrix},$$

$$B = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} \quad \text{und} \quad \tilde{E} = \begin{bmatrix} E_1 A_1 \\ E_2 A_2 \\ \vdots \\ E_n A_n \end{bmatrix}.$$

Dieses Gleichungssystem ist strikt spalten-diagonal-dominant, da $\sum_{j=1,\ldots,n} F_{ij} < 1$ und $\rho_i < 1$ für alle $i = 1, \ldots, n$.

In [38] wird gezeigt, daß ein Progressive-Refinement Schritt der Gleichung (4.8), in dem eine Fläche größter Strahlungsleistung ihre unverteilte Radiosity verteilt, äquivalent ist zu einem Southwell Relaxation Schritt der Gleichung (4.9), in dem die Variable mit dem größten Residuum relaxt wird, wobei das Ergebnis die Addition der Variablen und deren Residuen ist.

Die Konvergenz des Progressive-Refinement-Verfahrens für die Radiosity-Gleichung (4.8) ist deshalb äquivalent zur Konvergenz des Southwell Verfahrens für Gleichung (4.9).

Um eine Radiosity-Lösung für eine dynamische Szene zu erhalten, muß Gleichung (4.8) für jeden Frame der Animation gelöst werden. Für eine Animationssequenz aus m Frames muß daher das Gleichungssystem

$$B_i^f = E_i^f + \rho_i^f \sum_{j=1}^{n} B_j^f F_{ij}^f, \quad i = 1, \ldots, n, \; f = 1, \ldots, m \tag{4.10}$$

oder das äquivalente System in Matrix-Form

$$\begin{bmatrix} M^1 & 0 & \dots & 0 \\ 0 & M^2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & M^m \end{bmatrix} \begin{bmatrix} B^1 \\ B^2 \\ \vdots \\ B^m \end{bmatrix} = \begin{bmatrix} \tilde{E}^1 \\ \tilde{E}^2 \\ \vdots \\ \tilde{E}^m \end{bmatrix}, \tag{4.11}$$

das wieder ein strikt spalten-diagonal-dominantes System ist, gelöst werden. Die Matrix ist eine Blockmatrix mit einem Block für jeden Frame der Animation. Die Werte in den entsprechenden Frames sind durch den hochgestellten Index gekennzeichnet.

Die Durchführung eines simultanen Progressive-Refinement Schrittes, wie oben beschrieben, ist nun äquivalent zur Durchführung von m Relaxations-Schritten an Gleichung (4.11). Diese m Relaxations-Schritte relaxen genau eine Variable in jedem Block der Matrix und sind deshalb voneinander unabhängig in dem Sinne, daß in jedem Schritt nur die Residuen eines einzigen Blocks verändert werden und deshalb die Residuen von relaxten Variablen durch die Relaxation der anderen Variblen nicht wieder von Null verschieden werden. Das heißt, es werden in einem simultanen Progressive-Refinement Schritt gleichzeitig m Variablen relaxt.

Im folgenden wird gezeigt, daß das totale Residuum bei jedem simultanen Progressive-Refinement Schritt um einen konstanten Faktor abnimmt. Der Einfachheit halber benutzen wir die l_1 Norm:

$$\|r\| := \sum_i |r_i|.$$

Angenommen im $(k+1)$-ten Iterationsschritt verteile das Flächenstück F_i seine unverteilte Radiosity in die Szene. Da Gleichung (4.7) für jeden Block der Matrix gilt, haben wir

$$\begin{aligned} \|r^{(k+1)}\| &= \|r^{(k)}\| - \sum_{f=1}^{m} |r_i^{f\ (k)}| + \sum_{f=1}^{m} \sum_{j\neq i} \left| \frac{M_{ji}^f}{M_{ii}^f}\ r_i^{f\ (k)} \right| \\ &= 0\|r^{(k)}\| - \sum_{f=1}^{m} |r_i^{f\ (k)}| + \sum_{f=1}^{m} |r_i^{f\ (k)}| \sum_{j\neq i} \left| \frac{M_{ji}^f}{M_{ii}^f} \right|. \end{aligned}$$

Da die Matrix strikt spalten-diagonal-dominant ist, gilt

$$p := \max_{i,f} \sum_{j\neq i} \left| \frac{M_{ji}^f}{M_{ii}^f} \right| < 1$$

und deshalb

$$\|r^{(k+1)}\| \leq \|r^{(k)}\| - (1-p)\sum_{f=1}^{m} |r_i^{f\ (k)}|. \tag{4.12}$$

In dem Algorithmus für simultanes Progressive-Refinement wird in jedem Iterationsschritt stets eine Fläche größter Strahlungsleistung (siehe (4.1)) als verteilende Fläche gewählt. Mit $P_j^f = B_j^f A_j^f$ sieht man deshalb, daß die verteilende Fläche F_i immer so gewählt wird, daß

$$|r_i^{\tilde{f}\ (k)}| \geq \max_{j,f} |r_j^{f\ (k)}| \quad \text{für ein geeignetes } \tilde{f} \in \{1,\ldots,m\}$$

und damit

$$\sum_{f=1}^{m} |r_i^{f\ (k)}| \geq |r_i^{\tilde{f}\ (k)}| \geq \frac{\|r^{(k)}\|}{n\ m}.$$

Durch Anwendung von (4.12) erhält man

$$\|r^{(k+1)}\| \leq \|r^{(k)}\| - \frac{(1-p)}{n\ m}\|r^{(k)}\| = q\ \|r^{(k)}\|,$$

wobei

$$q = 1 - \frac{1-p}{n\ m} < 1.$$

Daraus folgt

$$\|r^{(k)}\| \leq q^k\ \|r^{(0)}\|,$$

was für $k \to \infty$ gegen Null konvergiert. □

4.3.2.3 Betrachtungen zur Konvergenzrate

Der in Abschnitt 4.3.2.2 gegebene Konvergenzbeweis liefert eine Abschätzung für die Konvergenzrate, die im ungünstigsten Fall zu erwarten ist. Dies ist die Konvergenzrate, die man mit Southwell Iteration für einen einzelnen Frame erhalten würde, multipliziert mit dem Faktor $1/m$.

Aber glücklicherweise kann in praktischen Situationen, aufgrund von Kohärenzen zwischen den Frames, eine viel schnellere Konvergenz erwartet werden. Der

ungünstigste Fall tritt ein, wenn in jedem Iterationsschritt die gewählte Fläche größter Strahlungsleistung nur in einem einzigen Frame die größte Strahlungsleistung besitzt und in allen übrigen Frames die Strahlungsleistung Null beträgt.

Im günstigsten Fall erhält man die Konvergenzrate der Southwell Iteration für einen einzelnen Frame. Dieser Fall tritt dann ein, wenn in jedem simultanen Iterationsschritt alle m gleichzeitig durchgeführten Relaxationsschritte Southwell-Relaxationsschritte im jeweiligen Block der Matrix sind, das heißt, wenn die ausgewählte Fläche größter Strahlungsleistung gleichzeitig in jedem einzelnen Frame eine Fläche größter Strahlungsleistung ist. Dies ist zum Beispiel dann der Fall, wenn die Szene keine dynamischen Objekte besitzt.

Für eine allgemeine Aussage über die Konvergenzrate für praktische Situationen wollen wir folgendes betrachten:

Für einen simultanen Iterationsschritt sei i_f der Index einer Fläche größter Strahlungsleistung im Frame f, das heißt i_f ist so gewählt, daß

$$P_{i,f} \geq P_{j,f} \quad \text{für alle} \quad j = 1, \ldots, n.$$

Sei nun

$$l = |\{i_f : f = 1, \ldots, m\}|$$

die Anzahl der verschiedenen Flächen, die in den verschiedenen Frames als Flächen größter Strahlungsleistung auftreten, und M sei die kleinste Zahl, so daß für alle Iterationsschritte $l \leq M$ ist. Dann ist für die simultane Progressive-Refinement Berechnung eine Konvergenzrate garantiert, die der der Southwell-Iteration für einen einzelnen Frame entspricht, multipliziert mit dem Faktor $1/M$. Für praktische Situationen ist die Zahl M, bedingt durch photometrische und räumliche Kohärenz sehr klein und weit entfernt von der Anzahl der Frames in der Animation.

Der Algorithmus arbeitet besonders effizient für Szenen mit einem hohen Anteil an statischen Objekten. Der Aufwand für die Sichtbarkeitsberechnungen, die für einen Iterationsschritt durchgeführt werden müssen, nimmt mit der Anzahl der bewegten Objekte zu. Dieser Aufwand läßt sich aber durch geeignete Techniken wesentlich reduzieren, die in den folgenden Abschnitten beschrieben werden. In der in Abbildung 4.4 dargestellten Szene ist ein Drittel der Objekte bewegt. Die Beschleunigungen, die mit den nachfolgend beschriebenen Techniken bei diesem Beispiel erzielt werden, sind in Tabelle 4.10 dargestellt.

4.3.3 Beschleunigung durch raumunterteilende Strukturen

Zur Sichtbarkeitsberechnung zwischen den einzelnen Flächenstücken werden Strahlen verwendet, die von einem Flächenstück bis zum andern durch die Szene verfolgt werden. Dabei müssen Schnittpunktsberechnungen mit den dazwischenliegenden Objekten durchgeführt werden. Um die Schnittpunktsberechnungen für einen Strahl

nicht mit allen Objekten in der Szene durchführen zu müssen, werden raumunterteilende Strukturen verwendet, die es erlauben, aus der Gesamtzahl von Objekten diejenigen zu selektieren, die sich möglicherweise zwischen den beiden Flächenstücken befinden und somit als mögliche Verdecker in Frage kommen.

Solche raumunterteilende Strukturen sind zum Beispiel eine Hierarchie von Boundingvolumes, reguläre eventuell rekursiv unterteilte Raumgitter, Oktrees oder BSP-Trees (binary space partitioning), in die in einem Vorbearbeitungsschritt alle Szenenobjekte einsortiert werden und die dann bei der Sichtbarkeitsberechnung von den Strahlen traversiert werden (siehe Abbildung 4.9). Schnittpunktsberechnungen müssen dann nur mit den Objekten durchgeführt werden, die in den vom Strahl durchlaufenen Teilbereichen der Szene eingetragen sind.

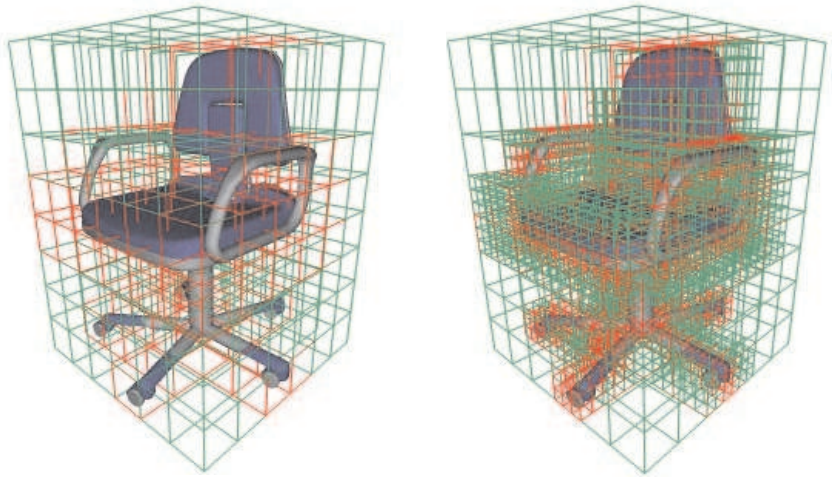


Abbildung 4.9: Rekursiv unterteiltes reguläres Gitter.

Bei bewegten Objekten wird in den einzelnen Zellen der raumunterteilenden Strukturen zusätzlich zu den Verweisen auf die enthaltenen Objekte noch die Information gespeichert, in welchen Zeitintervallen sich das entsprechende Objekt innerhalb der Zelle befindet.

Bei der Sichtbarkeitsberechnung zwischen zwei (in einem Zeitintervall) statischen Flächenstücken, wo die Sichtbarkeit der Strahlen für die gesamte Animationszeit (oder das Zeitintervall, in dem die Flächenstücke unbewegt sind) getestet werden muß, kann damit verhindert werden, daß Schnittpunkttests mit bewegten Objekten für alle Frames durchgeführt werden müssen. Es ist vielmehr ausreichend, sich auf die Frames zu beschränken, in denen die Objekte in den vom Strahl durchlaufenen Zellen der raumunterteilenden Struktur eingetragen sind und sich damit in der näheren Umgebung um den Strahl befinden.

4.3.4 Schnittpunktsberechnungen mit bewegten Objekten

Bei der Sichtbarkeitsberechnung zwischen statischen Flächenstücken müssen die Schnittpunkttests mit dynamischen Objekten für alle Frames der Animation durch-

geführt werden, oder, bei Verwendung von Beschleunigungsstrukturen wie in Abschnitt 4.3.3 beschrieben, mit einer Serie aufeinanderfolgender Frames. Im allgemeinen Fall bedeutet dies, daß für ein Objekt für jeden Frame ein einzelner Schnittpunkttest durchgeführt werden muß.

Im Fall von einfachen regelmäßigen Bewegungen eines Objektes, wie zum Beispiel einer linearen Bewegung, ist es jedoch möglich, mit einer einzigen Berechnung, die kaum aufwendiger ist als ein Schnittpunkttest für einen Frame, die kompletten Zeitintervalle der Verdeckung bzw. Sichtbarkeit zu bestimmen.

Bei komplizierteren Bewegungen, die eine solche direkte Berechnung nicht zulassen, kann die Anzahl der Schnittpunkttests reduziert werden, indem zunächst für ein geeignet gewähltes m nur jeder m-te Frame überprüft wird und die zwischen zwei solchen Stützstellen km und $(k+1)m$ liegenden Frames auch als sichtbar bzw. verdeckt angenommen werden, wenn die Frames km und $(k+1)m$ beide sichtbar bzw. verdeckt sind. Unterscheiden sich die Sichtbarkeiten in benachbarten Stützstellen, wird das dazwischenliegende Intervall rekursiv unterteilt. Je nach Art und Richtung der Bewegung, Form der Objekte und Abstand der Stützstellen können bei dieser Strategie aber Abtastfehler auftreten.

4.3.5 Abtastung bewegter Lichtquellen und Empfängerflächen

Bei bewegten Empfängerflächen muß die Sichtbarkeitsberechnung zwischen Lichtquelle und Empfängerfläche sowie die Formfaktorberechnung und die Bestimmung der übertragenen Strahlungsleistung für jeden Frame einzeln durchgeführt werden.

Noch aufwendiger wird es bei bewegten direkten oder indirekten Lichtquellen. Hier muß der gesamte Iterationsschritt, das heißt die Sichtbarkeits- und Formfaktorberechnung zu jeder einzelnen Empfängerfläche für alle Frames einzeln durchgeführt werden. Ein Iterationsschritt, bei dem eine bewegte Fläche als verteilende Fläche ausgewählt wird, ist also um ein Vielfaches aufwendiger als ein Iterationsschritt, bei dem eine statische Fläche ausgewählt wird. Da der weitaus größte Teil des Berechnungsaufwandes für die Sichtbarkeitsbestimmung verwendet werden muß, läßt sich dieser erheblich reduzieren, indem die Sichtbarkeitsberechnung wie im vorigen Abschnitt beschrieben, nur für einige Schlüsselframes durchgeführt wird und nur im Falle einer unterschiedlichen Verdeckung in benachbarten Schlüsselframes rekursiv unterteilt wird. Die Formfaktoren werden dann für alle sichtbaren Frames einzeln berechnet.

Wie im vorigen Abschnitt können dadurch auch hier Abtastfehler auftreten, die sich aber in ähnlicher Häufigkeit und Größenordnung ergeben, wie dies auch durch die räumliche Abtastung der Flächen und Lichtquellen der Fall ist. Durch geeignete Wahl der Abtastung werden in der Praxis aber ganz gute Ergebnisse erzielt. Die Abtastrate kann zum Beispiel so gewählt werden, daß die zwischen zwei Schlüsselframes zurückgelegte Strecke dem Abstand der Samples der räumlichen Abtastung entspricht.

Statische Szene	
ohne Beschleunigungstechniken	216 042
mit regulärem Raumgitter	39 748

Dynamische Szene mit 71 Frames	
ohne Beschleunigungstechniken	5 328 559
mit regulärem Raumgitter	897 740
Raumgitter und Abtastung bei den Schnittpunktsberechnungen	649 759
Raumgitter und Schnittpunktsberechnungen bei linearer Bewegung	618 164
zusätzlich Abtastung bewegter Empfängerflächen	560 221
zusätzlich Abtastung bewegter abstrahlender Flächen	165 450

Abbildung 4.10: Tabelle der für das in Abbildung 4.4 dargestellte Beispiel benötigten Schnittpunktsberechnungen unter Anwendung der in den vorigen Abschnitten beschriebenen Beschleunigungstechniken

4.3.6 Speicherung der Radiosity-Funktionen

Da der Algorithmus für simultanes Progressive-Refinement die gleichzeitige Verfügbarkeit der Radiosity-Werte aller Frames der Animationssequenz zur Berechnung voraussetzt, ist es notwendig, eine effiziente Repräsentation der Radiosity-Funktion heranzuziehen, um exzessiven Speicherplatzverbrauch zu vermeiden.

Mit Hilfe einer solchen Repräsentation muß es zum einen möglich sein, die Radiosity-Werte für Szenen, die in praktischen Situationen vorkommen, platzsparend zu speichern, zum anderen aber auch die notwendigen Berechnungen effizient und direkt in dieser Repräsentation durchführen zu können.

Bedingt durch photometrische Kohärenz kann in praktischen Situationen erwartet werden, daß sich die Radiosity an einem Vertex nicht von jedem Frame zum nächsten willkürlich ändert. Zwischen abrupten Änderungen, die sich zum Beispiel bei Änderungen der Sichtbarkeit zwischen dem Vertex und direkten Lichtquellen ergeben können, gibt es Zeitintervalle, in denen sich die Radiosity nicht oder nur mäßig und stetig ändert.

Für die meisten Vertices in der Szene bleibt die Sichtbarkeit zu den direkten Lichtquellen über lange Zeitintervalle unverändert.

In solchen Zeitintervallen, in denen sich die Sichtbarkeit zur abstrahlenden Fläche oder zur Lichtquelle nicht ändert, ist die zeitabhängige Funktion der einfallenden Radiosity $B(t)$ an statischen Vertices vom selben Typ als die Emissionsfunktion der abstrahlenden Fläche. Das heißt, $B(t)$ ist konstant, wenn die Lichtquelle eine konstante Emissionsfunktion besitzt. Analog ist $B(t)$ linear in Zeitintervallen, in denen sich die Sichtbarkeit zur Lichtquelle nicht ändert, wenn die Emissionsfunktion der Lichtquelle linear ist.

Wenn sich die Sichtbarkeit zur abstrahlenden Fläche oder Lichtquelle ändert,

hat $B(t)$ eine Unstetigkeit, wenn es sich um eine punktförmige Lichtquelle handelt, andernfalls hängt $B(t)$ von der Form der abstrahlenden Fläche und der Bewegung des verdeckenden Objektes ab.

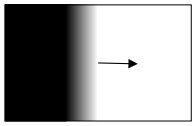


Abbildung 4.11: Flächenelement mit bewegtem Schatten.

Lineare Emissionsfunktionen treten zum Beispiel bei Flächen auf, über die sich bei konstanter Bestrahlung durch eine Lichtquelle mit konstanter Geschwindigkeit eine Schattengrenze hinwegbewegt, wie in Abbildung 4.11 dargestellt.

Nichtlineare Bewegungen, mehrfache Reflektionen, nichtkonstante Emissionsfunktionen direkter Lichtquellen oder nichtkonstante Reflektionskoeffizienten führen zu Funktionen höherer Ordnung.

Die Rechenoperationen, die mit den Radiosity-Funktionen durchgeführt werden müssen, sind Addition zweier Funktionen, Multiplikation von Funktionen mit Formfaktoren und Multiplikation von Funktionen mit Reflektionskoeffizienten. In Szenen mit einem hohen Anteil an statischen Objekten ist der größte Teil der Formfaktoren und der Reflektionskoeffizienten konstant. Das heißt, der Großteil der Operationen, die mit diesen Funktionen durchgeführt werden müssen, sind Addition zweier Funktionen und Multiplikation einer Funktion mit Konstanten.

In unserem Ansatz verwenden wir an jedem Vertex eine stückweise lineare Funktion in der Zeit, um die Radiosity-Funktion für den gesamten Lebenszyklus des Vertex zu repräsentieren, auch dann, wenn sich die Position des Vertex verändert. Funktionen höherer Ordnung werden durch stückweise lineare Funktionen approximiert.

Um eine solche stückweise lineare Funktion zu repräsentieren, wird ein Knotenvektor gespeichert, der die verwendeten Keyframe-Nummern enthält, und ein Vektor mit den zugehörigen Radiosity-Werten. Die Radiosity-Werte der übrigen Frames können dann durch Interpolation zwischen den beiden Nachbarn bestimmt werden. An Unstetigkeitsstellen von $B(t)$ werden zwei aufeinanderfolgende Frames als Keyframes gespeichert (siehe Abbildung 4.12).

Bei der Addition zweier Radiosityfunktionen in dieser Repräsentation muß die Vereinigung der Keyframes der beiden Funktionen gebildet und die zugehörigen

Frame	Radiosity
1	200
6	300
7	400
10	200

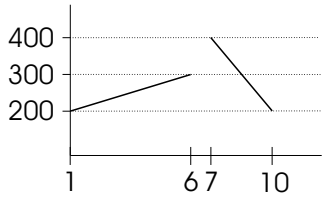


Abbildung 4.12: Repräsentation der Radiosityfunktion an einem Vertex

Radiosity-Werte gesetzt werden. Bei Multiplikationen mit Konstanten müssen nur alle Werte im Radiosity-Vektor mit der Konstanten multipliziert werden. Bei Multiplikationen mit stückweise konstanten Funktionen müssen zusätzliche Keyframes an den Unstetigkeitsstellen dieser Funktion eingefügt werden. Für andere Funktionen wird die Multiplikation in den nicht konstanten Intervallen der Funktion für jeden Frame einzeln durchgeführt und anschließend überprüft, welche Keyframes aufgrund eines Funktionswertes, der mit dem zwischen den Nachbarn interpolierten Wert übereinstimmt, eliminiert werden können.

Zusammenfassend können alle Additionen und Multiplikationen mit konstanten Formfaktoren und konstanten Reflektionskoeffizienten sehr effizient direkt in der Repräsentation durch stückweise lineare Funktionen durchgeführt werden, während dort mehr Aufwand nötig ist, wo sich Formfaktoren oder Reflektionskoeffizienten in der Zeit ändern. Dies ist für dynamische Flächen der Fall und ebenso für statische Flächen, über die sich weiche Schattengrenzen hinwegbewegen. In praktischen Situationen stellt sich aber heraus, daß auch dann ein großer Teil der Keyframes eliminiert werden kann, bedingt durch ruhige Bewegungen der dynamischen Objekte.

Für die Berechnung von Video-Sequenzen kann eine Approximation der Radiosity-Funktionen durch stückweise lineare Funktionen in Betracht gezogen werden, die die Farbwerte der einzelnen Pixel des entstehenden Filmes nicht beeinflußt. Die Verwendung noch gröberer Approximationen würde unerwünschte Artefakte hervorrufen.

Dieser Ansatz bewährte sich in unseren Beispielszenen als sehr effizient in Bezug auf die Speicherkosten, was im nächsten Kapitel ausführlich beschrieben wird.

Anstatt mit stückweise linearen Funktionen könnte man auch versuchen, die Radiosity-Funktion an einem Vertex durch Funktionen höherer Ordnung zu approximieren, in der Hoffnung, mit einer kleineren Anzahl an Knoten auszukommen. Aber betrachten wir einmal folgendes: An Stellen mit bewegten Schattengrenzen müssen wir in der Lage sein, Unstetigkeiten in der Radiosity-Funktion und auch in ihren Ableitungen darstellen zu können. Wir müssten also B-Splines mit Mehrfachknoten verwenden oder Einzelstücke von Funktionen höherer Ordnung verwalten, während Unstetigkeiten bei stückweise linearen Funktionen einfach durch Knoten in zwei aufeinanderfolgenden Frames repräsentiert werden können. Dies würde bei Funktionen höherer Ordnung zum Überschwingen führen. Die Verwendung von B-Splines oder das Verwalten von Einzelstücken würde aber sowohl den Speicherplatzbedarf als auch die Komplexität der notwendigen arithmetischen Operationen erhöhen. Bei einem großen Prozentsatz der Vertices können wir in praktischen Situationen ohnehin erwarten, daß die Radiosity-Funktion gut durch stückweise lineare Funktionen approximiert werden kann, da die Beiträge höherer Ordnung in den meisten Fällen durch indirekte Beleuchtung hervorgerufen werden, die im Vergleich zur Gesamtbeleuchtung des Vertex nur sehr klein ist.

4.3.7 Ausblick

Die zukünftige Forschungsarbeit wird sich auf eine weitere Beschleunigung des Berechnungsverfahrens und eine Verbesserung der Qualität konzentrieren.

Das wichtigste Gebiet wird dabei die Erzeugung geeigneter Netze zur Repräsentation der Radiosity-Funktion auf den Flächen sein. Es sollen Strategien entwickelt werden, die die Berechnung von konsistenten und flackerfreien Animationen unter Verwendung von sich im Laufe der Animation verändernden Netzen erlauben.

Dabei sollen, ausgehend von einem konstanten Ausgangsnetz, dynamische Unterteilungen eingearbeitet werden. An Stellen, an denen die Radiosity-Funktion eine starke Krümmung aufweist, sollen die Netze mit Hilfe hierarchischer Unterteilungen verfeinert werden. An Stellen, an denen die Radiosity-Funktion oder eine ihrer Ableitungen Unstetigkeiten aufweisen, zum Beispiel an Schattengrenzen oder im Innern von Halbschattenbereichen, sollen die Unstetigkeitslinien ins Netz eingearbeitet werden (siehe dazu zum Beispiel [61, 62, 63, 64]).

Kapitel 5

Ergebnisse

In diesem Kapitel werden anhand einiger Beispiele Ergebnisse vorgestellt, die mit dem simultanen Progressive-Refinement Algorithmus berechnet wurden und die Effizienz und Einsatzfähigkeit des Algorithmus demonstrieren.

Der Algorithmus wurde in C++ implementiert und in das am WSI/GRIS entwickelte objektorientierte Beleuchtungssystem RadioLab [65] integriert. Alle Zeiten wurden auf einem Windows NT4.0 PC mit einem 400 Mhz Intel Pentium II Prozessor gemessen.

5.1 Büroraum und Wohnzimmer

Abbildung 5.1: Erste Beispielszene.

Als erstes Beispiel dient die in Abbildung 5.1 dargestellte Szene, die aus zwei durch eine Schiebetür verbundenen Räumen besteht, einem Büro-Raum und einem Wohnzimmer. Der Büro-Raum wird durch zwei Deckenleuchten und eine Schreibtischlampe hell erleuchtet, während das Wohnzimmer nur durch einen Fernsehbildschirm schwaches Licht erhält. Während der Animation öffnet sich die Tür zwischen

den beiden Räumen, so daß das helle Licht aus dem Büro-Raum auch ins Wohnzimmer strahlen kann. Außerdem läuft auf dem Fernsehschirm im Wohnzimmer ein Film ab. Beleuchtungseffekte durch den Lichteinfall durch die sich öffnende Tür und das „Farbbluten“, das durch die wechselnden Bilder auf dem Fernsehschirm entsteht, sind in Bild 5.7 deutlich zu erkennen.

Die Szene wurde in 20 139 Flächenstücke unterteilt. Zur Beschleunigung der Sichtbarkeitsberechnung mit Raytracing wurde ein reguläres Raumgitter der Auflösung 50 mal 20 mal 13 Zellen verwendet, mit rekursiver Unterteilung dicht besetzter Zellen.

direktes Licht			
	1 Frame	40 Frames	1000 Frames
Rechenzeit (in Sekunden)	7	9	62
für zusätzlichen Frame		0.05=0.7%	0.05=0.7%
Anzahl Schnittpunktsberechnungen	1 063 126	1 168 481	3 797 506
für zusätzlichen Frame		2 701=0.3%	2 737=0.3%
Speicherbedarf in KB ohne Film	252	634	634
Speicherbedarf in KB mit Film	252	1 414	2 092

gesamte Berechnung			
	1 Frame	40 Frames	1000 Frames
Rechenzeit (in Sekunden)	30	91	1 305
für zusätzlichen Frame		1.56=5.2%	1.27=4.3%
Anzahl Schnittpunktsberechnungen	5 577 852	8 339 351	32 158 133
für zusätzlichen Frame		70 807=1.3%	26 607=0.5%
Speicherbedarf in KB ohne Film	256	700	753
Speicherbedarf in KB mit Film	256	1 758	2 412

Abbildung 5.2: Rechenzeiten und Anzahl der Schnittpunktsberechnungen, die für die Berechnung des direkten Lichtes und für die gesamte Berechnung der ersten Szene benötigt wurden, sowie Speicherbedarf für die Speicherung der Radiosity-Funktionen.

Die Rechenzeiten, die Anzahl der durchgeführten Schnittpunktsberechnungen, die für die Berechnung des direkten Lichtes und für die gesamte Berechnung des simultanen Progressive-Refinement Algorithmus benötigt wurden, und der Speicherbedarf, der in diesem Beispiel zur Speicherung der totalen und der unverteilten Radiosity benötigt wird, sind für eine Animation mit 40 Frames und eine Animation mit 1000 Frames im Vergleich zu einem einzelnen Frame in Abbildung 5.2 aufgeführt.

Die gesamte Berechnung umfaßt die Berechnung des direkten Lichtes, sowie die indirekte Abstrahlung des Bodens, der Decke, der vier Wände und der Tischplatte im Büroraum, sowie des Bodens, der Decke und der beiden hellsten Wände im Wohnzimmer. Dabei wurde für die abstrahlenden Flächen adaptives Sampling verwendet.

Die durchschnittliche Rechenzeit für einen zusätzlichen Frame beträgt in der aus 40 Frames bestehenden Animation 1.56 Sekunden und in der aus 1000 Frames bestehenden Animation 1.27 Sekunden, während zur Berechnung eines einzelnen Frames 30 Sekunden benötigt werden.

Im Fall eines konstanten Bildes auf dem Fernsehschirm wird sowohl für die aus 40 Frames als auch für die aus 1000 Frames bestehende Animation für die komplette Sequenz weniger als das Dreifache des Speicherplatzes für einen einzelnen Frame benötigt. Dies ist ein sehr gutes Ergebnis, wenn man bedenkt, daß eine stückweise lineare Funktion mindestens das Doppelte an Speicherplatz benötigt, wie ein einzelner Wert.

Auch in dem Fall, in dem ein Film auf dem Fernsehschirm abläuft, ist der Speicherbedarf gering, obwohl man hier vielleicht aufgrund der sich ändernden Lichtemission einer Primärlichtquelle schlechte Kompressionsraten erwarten würde.

5.2 Theater

Abbildung 5.3: Zweite Beispielszene.

Als zweite Beispielszene dient ein Theater, das aus 98 110 Flächenstücken und 59 primären Lichtquellen besteht (siehe Abbildung 5.3). In einer 6-sekündigen Animation mit 151 Frames wird während der ersten 2 Sekunden die Hauptbeleuchtung an der Decke verdunkelt und anschließend in der nächsten Sekunde der mittlere Strahler eingeschaltet, der gegen den Bühnenvorhang gerichtet ist. In den nächsten zwei Sekunden öffnet sich der Bühnenvorhang und in der letzten Sekunde schließlich werden die beiden anderen Strahler und die Stehleuchte auf der Bühne eingeschaltet. Eine Bildfolge der berechneten Animation ist in Abbildung 5.8 dargestellt.

Zur Beschleunigung des Raytracing wurde wieder ein regelmäßiges, rekursiv unterteiltes Raumgitter verwendet, diesmal mit einer Auflösung von 30 mal 20 mal 10 Zellen.

In Abbildung 5.4 sind die benötigten Rechenzeiten, die Anzahl der durchgeführten Schnittpunktsberechnungen sowie der benötigte Speicherbedarf zur Speicherung der Radiosity-Werte aufgeführt, wieder zunächst für die Berechnung des direkten Lichtes und anschließend für die gesamte Berechnung. Die gesamte Berechnung umfaßt die Berechnung des direkten Lichtes, sowie die indirekte Abstrahlung der Decke, der Wände und des Bühnenbodens.

Die durchschnittliche für einen zusätzlichen Frame benötigte Rechenzeit beträgt in diesem Beispiel 9.2 Sekunden, während zur Berechnung eines einzelnen Frames 1 666 Sekunden nötig sind.

direktes Licht		
	1 Frame	151 Frames
Rechenzeit (in Sekunden)	1 037	1 796
für zusätzlichen Frame		5.1=0.5%
Anzahl Schnittpunktsberechnungen	313 235 413	372 167 603
für zusätzlichen Frame		392 881=0.13%
Speicherbedarf in KB	2 850	7 306

gesamte Berechnung		
	1 Frame	151 Frames
Rechenzeit (in Sekunden)	1 666	3 051
für zusätzlichen Frame		9.2=0.6%
Anzahl Schnittpunktsberechnungen	515 375 637	583 883 381
für zusätzlichen Frame		456 718=0.09%
Speicherbedarf in KB	2 930	17 975

Abbildung 5.4: Rechenzeiten und Anzahl der Schnittpunktsberechnungen, die für die Berechnung des direkten Lichtes und für die gesamte Berechnung der zweiten Szene benötigt wurden, sowie Speicherbedarf für die Speicherung der Radiosity-Funktionen.

5.3 Hüpfende Bälle

Als drittes Beispiel dient eine Animation der in Abbildung 5.5 dargestellten Szene. In dieser Animation hüpfen sieben Bälle die Treppe hinunter. Die Bewegung dieser Bälle wurden mit Hilfe einer Partikelsimulation (siehe Abschnitt 2.1) berechnet. Zur Kollisionsdetektion wurde eine implizite B-Spline-Tensorprodukt-Fläche vom Grad eins verwendet (siehe Abschnitt 2.1.2.2.2) mit Kollisionsreaktion durch ein Kraftfeld. Für die Simulation der Bälle, die etwas mehr als zwei Sekunden dauert, wurde weniger als eine halbe Minute benötigt.

Für die Beleuchtungssimulation wurde die Szene in 15 662 Flächenstücke unterteilt. Sie wird durch vier von der Decke abgehängte Lichtquellen beleuchtet. Die

Abbildung 5.5: Dritte Beispielszene.

Daten über die Anzahl der benötigten Schnittpunktsberechnungen, die Rechenzeiten, und den Speicherbedarf für die aus 60 Frames bestehende Animation sind in Abbildung 5.6 und eine Bildfolge der berechneten Animation in Abbildung 5.9 dargestellt.

direktes Licht		
	1 Frame	60 Frames
Rechenzeit (in Sekunden)	11	35
für zusätzlichen Frame		0.41=3.7%
Anzahl Schnittpunktsberechnungen	1 571 217	3 862 239
für zusätzlichen Frame		38 831=2.5%
Speicherbedarf in KB	577	1 211

gesamte Berechnung		
	1 Frame	60 Frames
Rechenzeit (in Sekunden)	26	92
für zusätzlichen Frame		1.12=4.3%
Anzahl Schnittpunktsberechnungen	3 406 288	7 931 131
für zusätzlichen Frame		76 692=2.3%
Speicherbedarf in KB	577	1 836

Abbildung 5.6: Rechenzeiten und Anzahl der Schnittpunktsberechnungen, die für die Berechnung des direkten Lichtes und für die gesamte Berechnung der dritten Szene benötigt wurden, sowie Speicherbedarf für die Speicherung der Radiosity-Funktionen.

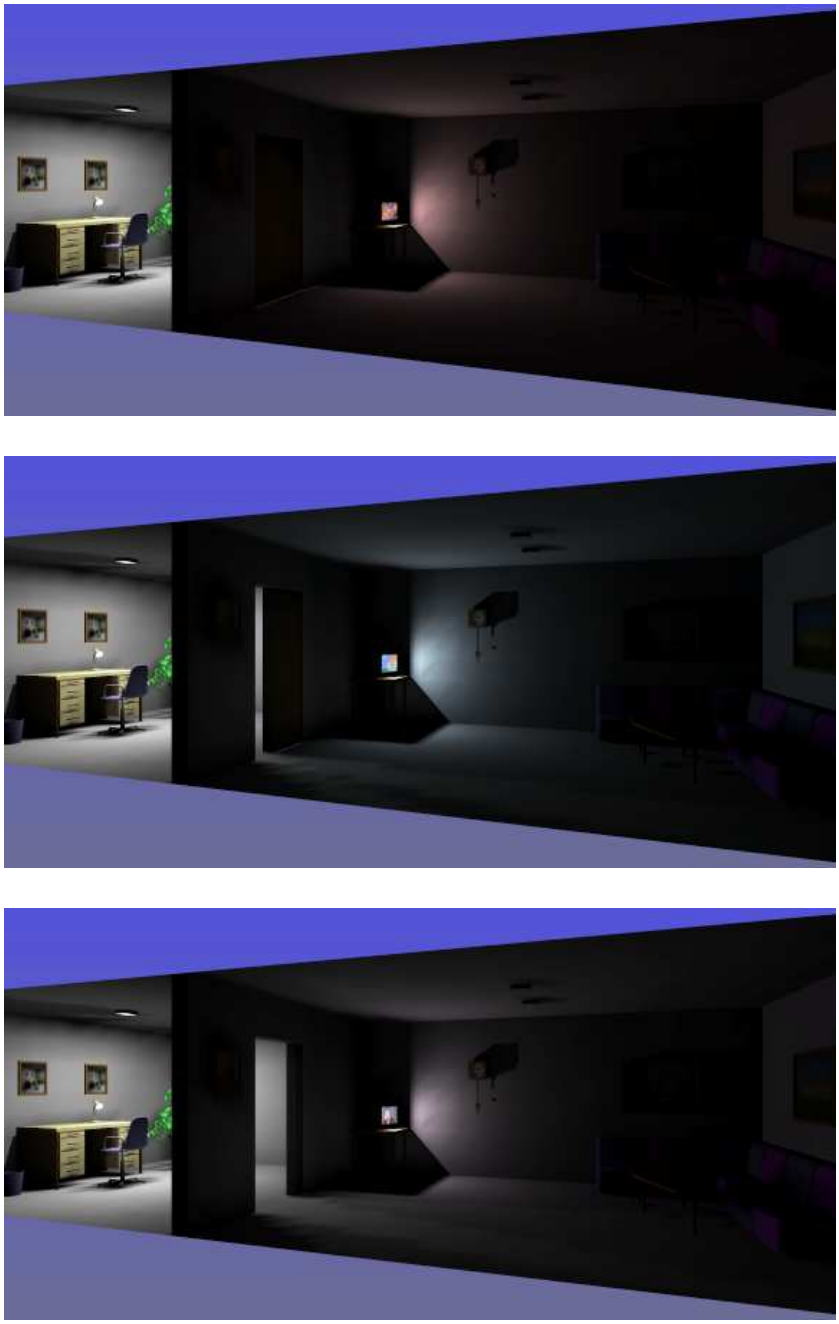


Abbildung 5.7: Erste Beispielszene.

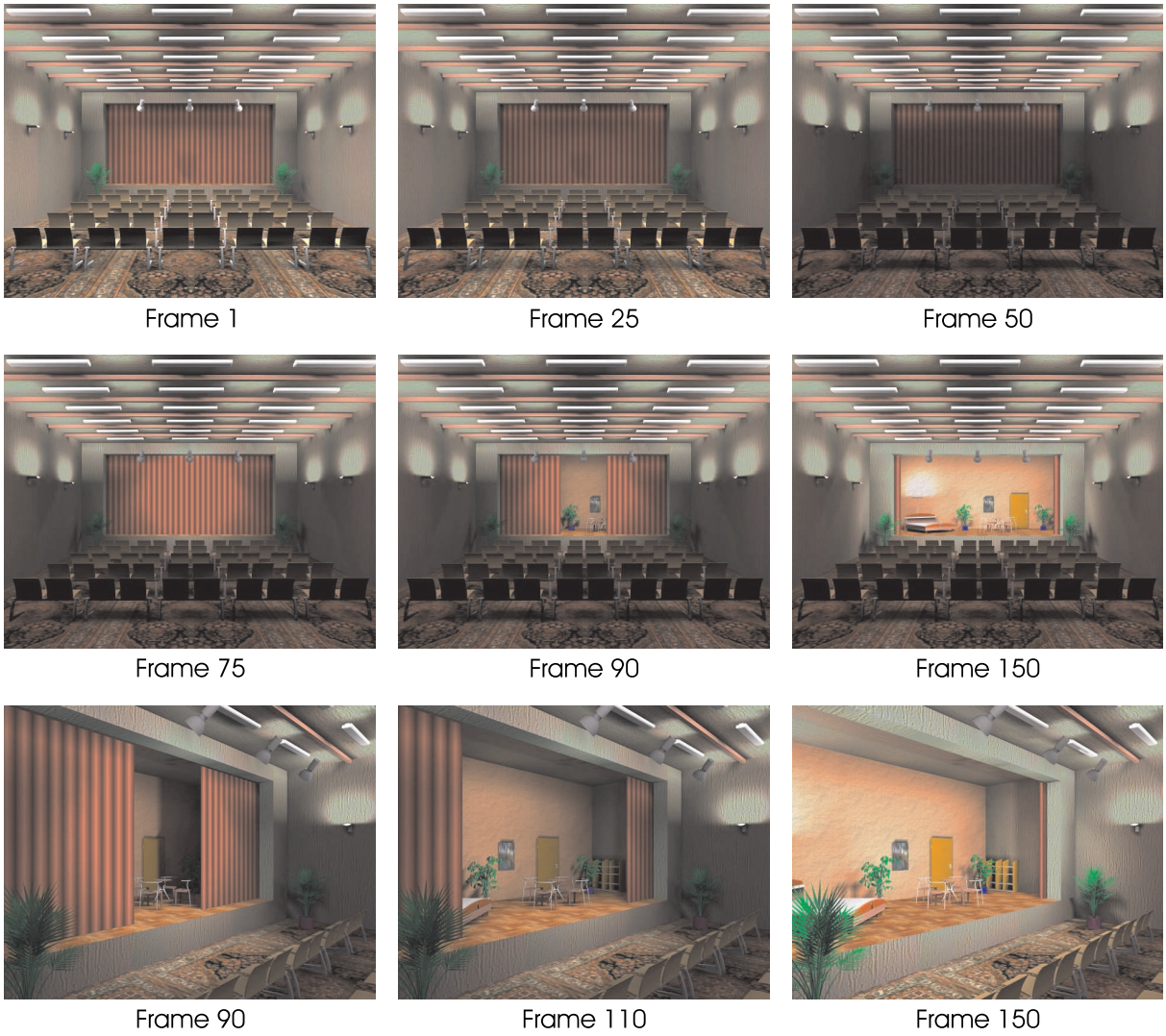


Abbildung 5.8: Zweite Beispielszene.

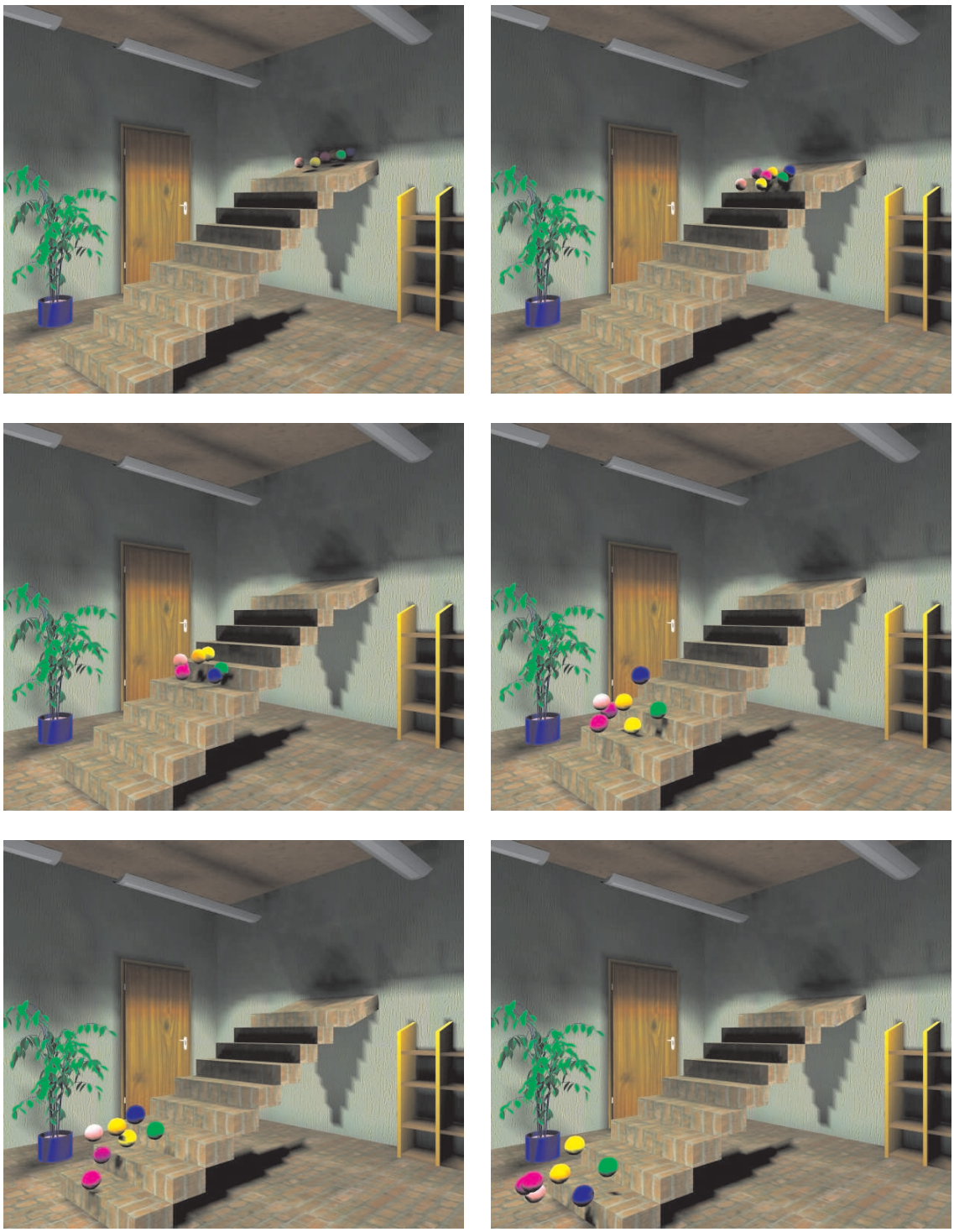


Abbildung 5.9: Dritte Beispielszene.

Literaturverzeichnis

[1] B. Eberhardt, J. Hahn, and T. Hüttner, "Raytracing and collision detection for cloth modeling," Tech. Rep. WSI–2000–10, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 72076 Tübingen, Germany, 2000.

[2] B. Eberhardt, J. Hahn, R. Klein, W. Strasser, and A. Weber, "Dynamic implicit surfaces for fast proximity queries in physically based modeling," Tech. Rep. WSI–2000–11, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 72076 Tübingen, Germany, 2000.

[3] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, "Modelling the Interaction of Light Between Diffuse Surfaces," in *Computer Graphics '84 Proceedings* (H. Christiansen, ed.), vol. 18, pp. 213–222, ACM SIGGRAPH, July 1984.

[4] M. F. Cohen and J. R. Wallace, *Radiosity and Realistic Image Synthesis*. Boston, MA: Academic Press Professional, 1993. ISBN 0-12-178270-0.

[5] F. Sillion and C. Puech, *Radiosity and Global Illumination*. San Francisco, CA: Morgan Kaufmann, 1994. ISBN 1-55860-277-1.

[6] J. Hahn and W. Strasser, "Simultaneous progressive refinement in dynamic environments," Tech. Rep. WSI–98–3, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 72076 Tübingen, Germany, 1998.

[7] J. Hahn and W. Strasser, "Simultaneous progressive refinement for consistant radiosity animations," Tech. Rep. WSI–2000–12, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 72076 Tübingen, Germany, 2000.

[8] J. Collier, B. Collier, G. O'Toole, and S. Sargand, "Drape prediction by means of finite element analysis," in *J. of Textile Inst.*, vol. 82, pp. 96–107, The Textile Institute, 1991.

[9] Gan, Ly, and Steven, "A study of fabric deformation using nonlinear finite elements," in *J. of Textile Inst.*, vol. 65, p. 660, The Textile Institute, 1995.

[10] Kang, Yu, and Chung, "Drape simulation of woven fabric by using the finite element method," in *J. of Textile Inst.*, vol. 86, p. 635, The Textile Institute, 1995.

[11] R. Hockney and J. Eastwood, *Computer Simulation using Particles*. Bristol and Philadelphia: Institute of Physics Publishing, 1994.

[12] B. Eberhardt, A. Weber, and W. Strasser, "A fast, flexible, particle-system model for cloth draping," *IEEE Computer Graphics and Applications*, vol. 16, pp. 52–60, Sept. 1996.

[13] E. Hairer and G. Wanner, Solving Ordinary Differential Equations II. Berlin: Springer-Verlag, 1996.

[14] M. Hauth, "Numerische Verfahren zur Simulation von Textilien," Diplomarbeit, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 72076 Tübingen, Germany, 1999.

[15] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Boston: PWS Publishing, 1996.

[16] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *SIGGRAPH 98 Conference Proceedings* (M. Cohen, ed.), Annual Conference Series, pp. 43–54, ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

[17] M. Lin and S. Gottschalk, "Collision detection between geometric models: A survey," in *Proceedings of IMA Conference on Mathematics of Surfaces*, 1998. ftp://ftp.cs.unc.edu/pub/users/manocha/PAPERS/COLLISION/cms.pdf.

[18] L. J. Guibas, D. Hsu, and L. Zhang, "H-walk: hierarchical distance computation for moving convex bodies," in *Proceedings of the fifteenth annual symposium on Computational Geometry*, (Miami, FL), pp. 265–273, Association for Computing Machinery, 1999.

[19] B. Mirtich, "V-Clip: fast and robust polyhedral collision detection," *ACM Transactions on Graphics*, vol. 17, no. 3, pp. 177–208, 1998.

[20] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: a hierarchical structure for rapid interference detection," in *Computer Graphics '96 Proceedings* (H. Rushmeier, ed.), pp. 171–180, ACM SIGGRAPH, August 1996.

[21] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," Tech. Rep. TR99-018, Department of Computer Science, University of North Carolina, Chapel Hill, 1999. http://http://www.cs.unc.edu/ geom/SSV/ssv.pdf.

[22] J. Klosowski, H. Held, J. Mitchell, K. Zikan, and H. Sowizral, "Efficient collision detection using bounding volume hierarchies of k-DOPs," *IEEE Trans. Visual. Comput. Graph.*, vol. 4, no. 1, 1998.

[23] A. Raviv and G. Elber, "Three dimensional freeform sculpting via zero sets of scalar trivariate functions," in *Proceedings of the fifth symposium on solid modeling and applications (SMA '99) Association for Computing Machinery*, (Ann Arbor, MI, USA), pp. 246–257, June 1999.

[24] S. Kawabata, *The Standardization and Analysis of Hand Evaluation.* Osaka: The Textile Machinery Society of Japan, 1980.

[25] T. Fließbach, *Mechanik, Lehrbuch zur Theoretischen Physik I.* Heidelberg, Berlin, Oxford: Spektrum, Akademischer Verlag, second ed., 1996.

[26] J. T. Kajiya, "The Rendering Equation," in *Computer Graphics '86 Proceedings* (D. Evans and R. Athay, eds.), vol. 20, pp. 143–150, ACM SIGGRAPH, August 1986.

[27] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics, Principles and Practice, Second Edition.* Reading, Massachusetts: Addison-Wesley, 1990.

[28] P. Schroder and P. Hanrahan, "On the Form Factor Between Two Polygons," in *Computer Graphics '93 Proceedings* (S. Cunningham, ed.), pp. 163–164, ACM SIGGRAPH, 1993.

[29] M. Cohen and D. P. Greenberg, "The Hemi-Cube: A Radiosity Solution for Complex Environments," in *Computer Graphics '85 Proceedings* (B. Barsky, ed.), ACM SIGGRAPH, August 1985.

[30] J. R. Wallace, K. A. Elmquist, and E. A. Haines, "A Ray Tracing Algorithm for Progressive Radiosity," in *Computer Graphics '89 Proceedings* (R. J. Beach, ed.), vol. 23, pp. 315–324, ACM SIGGRAPH, July 1989.

[31] N. P. Buslenko, D. I. Golenko, Y. A. Shreider, I. M. Sobol, and V. G. Sragovich, *The Monte Carlo Method – The Method of Statistical Trials.* New York, NY: Pergamon Press, 1962.

[32] J. M. Hammersley and D. C. Handscomb, *The Monte Carlo Method.* Bristol, UK: Cambridge University Press, 1964.

[33] P. Bekaert, *Hierarchical and Stochastic Algorithms for Radiosity.* Dissertation, Katholische Universität Leuven, Faculteit Wetenschappen, Departement Computerwetenschappen, Heverlee, Belgien, 1999.

[34] H. Niederreiter, "Random Number Generation and Quasi-Monte Carlo Methods," in *CBMS-NSF regional conference series in Appl. Math.*, vol. 63, (SIAM, Philadelphia), 1992.

[35] J. H. Halton, "Sequential Monte Carlo," *Proc. Cambridge Phil. Soc.*, vol. 58, pp. 57 – 78, 1962.

[36] M. Sbert, X. Pueyo, L. Neumann, and W. Purgathofer, "Global Multipath Monte Carlo Algorithms for Radiosity," *The Visual Computer*, vol. 12, no. 2, pp. 47–61, 1996.

[37] M. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg, "A Progressive Refinement Approach to Fast Radiosity Image Generation," in *Computer Graphics '88 Proceedings* (R. J. Beach, ed.), vol. 22, pp. 75–84, ACM SIGGRAPH, August 1988.

[38] S. J. Gortler, M. F. Cohen, and P. Slusallek, "Radiosity and Relaxation Methods," *IEEE Computer Graphics and Applications*, vol. 14, pp. 48–58, November 1994.

[39] P. Hanrahan, D. Salzman, and L. Aupperle, "A Rapid Hierarchical Radiosity Algorithm," in *Computer Graphics '91 Proceedings* (R. J. Beach, ed.), vol. 25, pp. 197–206, ACM SIGGRAPH, July 1991.

[40] F. Sillion, "Clustering and Volume Scattering for Hierarchical Radiosity Calculations," in *Fifth Eurographics Workshop on Rendering* (G. Sakas, P. Shirley, and S. Müller, eds.), (Darmstadt, Germany), pp. 105–117, Eurographics, June 1994.

[41] S. J. Gortler, P. Schroder, M. F. Cohen, and P. Hanrahan, "Wavelet Radiosity," in *Computer Graphics '93 Proceedings*, pp. 221–230, ACM SIGGRAPH, 1993.

[42] P. Schroder, S. J. Gortler, M. F. Cohen, and P. Hanrahan, "Wavelet Projections for Radiosity," in *Fourth Eurographics Workshop on Rendering* (M. Cohen, C. Puech, and F. Sillion, eds.), (Paris, France), pp. 105–114, Eurographics, June 1993.

[43] G. Beylkin, R. Coifman, and V. Rokhlin, "Fast Wavelet Transforms and Numerical Algorithm I.," in *Communications on Pure and Applied Mathematics*, no. 44, pp. 141–183, 1991.

[44] J. Stoer and R. Burlisch, *Introduction to Numerical Analysis*. New York, USA: Springer-Verlag, 1980.

[45] P. Schröder, *Wavelet Algorithms for Illumination Computations*. Ph.D. thesis, Department of Computer Science, Princeton University, Princeton, NJ, November 1994.

[46] P. Shirley, "A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes," in *Proceedings of Graphics Interface '90*, (San Francisco, CA), pp. 205–212, Morgan Kaufmann, May 1990.

[47] L. Neumann, M. Feda, M. Kopp, and W. Purgathofer, "A New Stochastic Radiosity Method for Highly Complex Scenes," in *Fifth Eurographics Workshop on Rendering*, (Darmstadt, Germany), pp. 195–206, Eurographics, June 1994.

[48] P. Shirley, "Radiosity via Ray Tracing," in *Graphics Gems II* (J. Arvo, ed.), pp. 306–310, Boston, MA: Academic Press Professional, 1991.

[49] D. R. Baum, J. R. Wallace, M. F. Cohen, and D. P. Greenberg, "The Back-Buffer Algorithm: An Extension of the Radiosity Method to Dynamic Environments," *The Visual Computer*, vol. 2, pp. 298–306, September 1986.

[50] M. Cohen and D. P. Greenberg, "The Hemi-Cube: A Radiosity Solution for Complex Environments," in *Computer Graphics '85 Proceedings* (B. Barsky, ed.), vol. 19, pp. 31–40, ACM SIGGRAPH, Addison Wesley, August 1985.

[51] D. W. George, F. X. Sillion, and D. P. Greenberg, "Radiosity Redistribution for Dynamic Environments," *IEEE Computer Graphics and Applications*, vol. 10, pp. 26–34, July 1990.

[52] S. E. Chen, "Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System," in *Computer Graphics '90 Proceedings* (R. J. Beach, ed.), vol. 24, pp. 135–144, ACM SIGGRAPH, August 1990.

[53] S. Muller and F. Schoffel, "Fast Radiosity Repropagation for Interactive Virtual Environments Using a Shadow-Form-Factor-List," in *Fifth Eurographics Workshop on Rendering*, (Darmstadt, Germany), pp. 325–342, June 1994.

[54] D. A. Forsyth, C. Yang, and K. Teo, "Efficient Radiosity in Dynamic Environments," in *Fifth Eurographics Workshop on Rendering*, (Darmstadt, Germany), pp. 313–323, June 1994.

[55] E. S. Shaw, "Hierarchical Radiosity for Dynamic Environments," M.Sc. thesis, Program of Computer Graphics, Cornell University, Ithaca, NY, August 1994.

[56] E. Shaw, "Hierarchical radiosity for dynamic environments," *Computer Graphics Forum*, vol. 16, no. 2, pp. 107–118, 1997.

[57] G. Drettakis and F. Sillion, "Interactive update of global illumination using a line-space hierarchy," in *Computer Graphics '97 Proceedings*, pp. 57–64, ACM SIGGRAPH, New York, Aug. 1997.

[58] G. Besuievsky and M. Sbert, "The Multi-Frame Lighting Method: A Monte Carlo Based Solution for Radiosity in Dynamic Environments," in *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, (New York, NY), pp. 185–194, Springer-Verlag/Wien, 1996. ISBN 3-211-82883-4.

[59] R. Burlisch and J. Stoer, *Introduction to Numerical Analysis.* Berlin, Germany: Springer-Verlag, 1972.

[60] N. Gastinel, *Linear Numerical Analysis.* Boston, MA: Academic Press, 1970.

[61] S. J. Teller, "Computing the antipenumbra of an area light source," in *Computer Graphics '92 Proceedings* (E. E. Catmull, ed.), vol. 26, pp. 139–148, ACM SIGGRAPH, July 1992.

[62] G. Drettakis and E. Fiume, "A Fast Shadow Algorithm for Area Light Sources Using Backprojection," in *Computer Graphics '94 Proceedings* (A. Glassner, ed.), pp. 223–230, ACM SIGGRAPH, ACM Press, 1994.

[63] N. Gatenby and W. T. Hewitt, "Optimizing Discontinuity Meshing Radiosity," in *Fifth Eurographics Workshop on Rendering*, (Darmstadt, Germany), pp. 249–258, Eurographics, June 1994.

[64] A. Worrall, C. Willis, and D. Paddon, "Dynamic Discontinuities for Radiosity," in *Edugraphics + Compugraphics Proceedings* (H. P. Santo, ed.), (P.O. Box 4076, Massama, 2745 Queluz, Portugal), pp. 367 – 375, GRASP- Graphic Science Promotions & Publications, December 12 1995. ISBN 972-8342-00-4.

[65] R. Sonntag, *RadioLab: An object oriented global illumination system.* Dissertation, Universität Tübingen, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 72076 Tübingen, Germany, 2001. Forthcoming.

Lebenslauf

25.08.1968	geboren in Tübingen
1975 – 1979	Grundschule in Ofterdingen
1979 – 1988	Quenstedt-Gymnasium in Mössingen Abschluß: Abitur
1988 – 1989	Wehrdienst
1989 – 1995	Mathematik-Studium mit Nebenfach Informatik an der Eberhard-Karls-Universität Tübingen (1993 – 1994: Auslandsaufenthalt an der Louisiana State University in Baton Rouge). Abschluß: Diplom
1995 – 2000	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Graphisch-Interaktive Systeme des Wilhelm-Schickard-Instituts für Informatik an der Universität Tübingen (Prof. Strasser).