Instant Radiosity

Alexander Keller
Universität Kaiserslautern

Seminarvortrag von Oliver Knoll
Sommersemester 1998

Inhaltsverzeichnis

1 Einleitung

Um fotorealistische Bilder auf dem Computer zu generieren, muss die Lichtverteilung im Raum berechnet werden. Das fundamentale Mass bei physikalisch basierten Renderalgorithmen ist Radiance. Dieses Mass $L(y, \omega_r)$ drückt aus, wieviel an Strahlung ein Oberflächenelement im Punkt y in eine bestimmte Richtung ω_r pro Einheitsoberfläche, welche senkrecht zu dieser Richtung liegt, und pro Solid Angle (siehe Situation in *Abbildung 1*) abgibt. Die Einheit der Radiance ist [W m^{-2} sr^{-1}].

Die Verteilung des Lichtes im Raum wird komplett durch die *Radiance Equation* beschrieben. Dabei geht man davon aus, dass die Strahlung entlang ihrer Verteilungsrichtung konstant ist, solange es keine Absorption, Emission oder Streueffekte durch Medien wie Rauch oder Nebel gibt. Deshalb beschreibt die *Radiance Equation* die Lichtverteilung im Vakuum. Zudem ist das Licht, welches von der Retina des menschlichen Auges aufgefangen wird, proportional zu der Strahlung, welche von der betrachteten Fläche emittiert wird. Diese Eigenschaften machen die Radiance zu einem geeigneten Mass für die Bildgenerierung durch Rendersysteme.

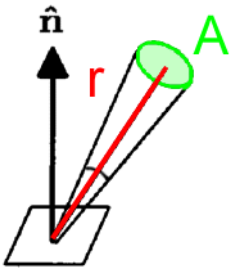

Abbildung 1: Solid Angle

Wenn man sich auf diffuse Reflexionen beschränkt, dann kommt man zu den sogenannten Radiosity-Algorithmen. Auf der einen Seite findet man die klassischen Methoden, welche die Lösung der Radiance Equation auf eine Menge von endlichen Basisfunktionen projizieren und eine Form Faktor Matrix berechnen. Dazu zerlegen sie die Szene in Patches. Die Formfaktoren hängen von der Geometrie der Szene ab und geben an, wieviel Licht zwischen zwei Oberflächenelementen ausgetauscht wird. Die Form Faktor Matrix ist von quadratischer Ordnung, gemessen an der Anzahl Szenenelemente, und sehr kostspielig zu berechnen. Zudem setzt sie voraus, dass man die Szene als ganzes kennt.

Um diesem enormen Speicherbedarf ein wenig entgegenzuwirken, wurden hierarchische Methoden entworfen. Diese Algorithmen können automatisch den Detaillevel bei der Patchgenerierung steuern. So können die Algorithmen adaptiv die Rechengenauigkeit dort steigern, wo dies notwendig ist, wie zum Beispiel an Schattengrenzen.

Diese *Galerkinverfahren*, wie diese Methoden auch genannt werden, müssen die Lösungen der *Radiance Equation* zwischenspeichern, was zu einem enormen Speicherbedarf führen kann. Zusammen mit der komplexen Zerlegung der Szene in kleinere Oberflächen führt die Projektion der *Radiance Equation* auf endliche Basen zu Diskretisationsfehlern.

Auf der anderen Seite sind die *Monte Carlo Methoden* anzusiedeln. Diese approximieren die Lösung von Integralen einer Funktion *f*, indem sie die Funktion an zufälligen Punkten auswerten, die Ergebnisse aufsummieren und durch die Anzahl Samples dividieren. Die Punkte werden dabei gemäss einer uniformen Wahrscheinlichkeitsverteilung zufällig gewählt. Nach dem Gesetz der grossen Zahlen konvergiert die Varianz der Lösung gegen null, wenn man eine grosse Menge von Samples in Betracht zieht. Die Konvergenzrate ist dabei gegeben durch $O(N^{-1/2})$. Da die Anzahl N von Samples in konkreten Berechnungen beschränkt ist, wird die Restvarianz in Bildern sichtbar als Rauschen.

Monte Carlo Methoden zeichnen sich dadurch aus, dass sie wenig bis gar keinen Speicher für Zwischenresultate benötigen, und sie sind einfach zu implementieren. Allerdings ist auch ein sehr hoher Rechenaufwand nötig, da die Funktion an vielen Samplepunkten ausgewertet werden muss. Um den Approximationsfehler zu halbieren, muss die Anzahl von Samples quadriert werden.

Die Konvergenzrate kann aber mit deterministischen Punktsequenzen verbessert werden. Der Algorithmus *Instant Radiosity* nutzt solche Punktsequenzen zusammen mit dem Konzept des *Quasi-Random Walk* aus, um eine Lichtpartikelsimulation zu implementieren. Zudem kann er direkt die Fähigkeiten von Hardwarebeschleuniger

---

Solid Angle: Ein Solid Angle ist analog definiert zu einem Winkel im Kreis, welcher in Radians gemessen wird. Ein Winkel im Kreis, aufgespannt durch die Länge l auf dem Kreisbogen, ist gegeben durch die Grösse l/r, wobei r der Radius des Kreises ist. Der Kreis selbst spannt einen Winkel von 2π auf, da der Kreisumfang $2\pi r$ ist. Ein Solid Angle ist nun gegeben durch die Grösse A/r^2, wobei A die sphärische Fläche ist, die den Winkel aufspannt, und r der Radius der Kugel. Dieses Winkel wird in *Steradians* [sr] angegeben. Da die Oberfläche einer Kugel $4\pi r^2$ gross ist, gibt es 4π Steradians in einer Kugel.

ausnutzen und braucht keine Zwischenlösungen. Dies wird uns zu einem schnellen und einfach zu implementierenden Algorithmus führen.

2 Globale Beleuchtung

2.1 Die Radiance Equation

Die Verteilung von Licht in einem luftleeren Raum kann vollständig durch die *Radiance Equation* beschrieben werden:

$$L(y, \vec{\omega}_r) = L_e(y, \vec{\omega}_r) + \int_\Omega f_r(\vec{\omega}_i, y, \omega_r) L(h(y, \vec{\omega}_i), -\vec{\omega}_i) \cos\theta_i d\omega_i \qquad (1)$$

Die Radianz $L(y, \omega_r)$ beschreibt, wieviel Strahlung von einem Punkt *y* aus in eine gegebene Richtung ω_r abgegeben wird (siehe *Abbildung 2*). Sie ist die Summe der in Punkt *y* emittierten und reflektierten Strahlung. Die reflektierte Strahlung setzt sich dabei aus allen einfallenden Strahlen zusammen, welche in die ausgehende Richtung ω_r an der Oberfläche in Punkt *y* gestreut werden. Dabei wird über alle Richtungen $\omega = (\theta, \phi)$ der Hemisphäre Ω integriert, welche normal zur Oberfläche im Punkt y liegt. Die einfallende Strahlung selbst kann auch schon an anderen Oberflächen reflektiert worden sein. Diese Kopplung aller sichtbaren Objekte macht es so schwierig, die Radiance Equation zu lösen.

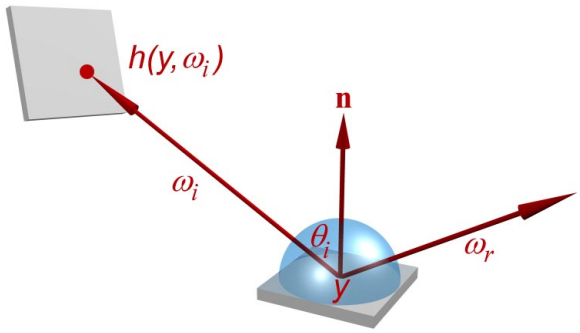

Abbildung 2: Die Radiance Equation

Die Funktion *h* berechnet den ersten Punkt, welcher getroffen wird, wenn man einen Strahl von *y* aus in Richtung ω_i schiesst. Der Term $\cos\theta_i$ projiziert die einfallende Strahlung auf die Flächennormale, wobei θ_i der Winkel zwischen der einfallenden Strahlung und der Flächennormalen ist.

Diese Gleichung heisst auch *Fredholm'sche Integralgleichung 2. Art* und lässt sich im Allgemeinen nicht analytisch lösen. Man stellt fest, das die gesuchte Lösung *L* auf beiden Seiten vorkommt. Diese Gleichung ist für jede Wellenlänge des sichtbaren Spektrums zu lösen, wobei man sich in der Computergrafik auf die Wellenlängen für Rot, Grün und Blau beschränkt. In Operatornotation lässt sich die Gleichung wie folgt aufschreiben:

$$L = L_e + T_{f_r} L$$

T_{fr} ist dabei der Transportoperator. Er berechnet das reflektierte Licht, welches auf die Oberfläche im Punkt *y* einfällt.

2.2 Radiosity

Die *Bidirectional Reflectance Distribution Function* (BRDF) f_r beschreibt, wieviel der einfallenden Strahlung in welche Richtung reflektiert werden soll und berücksichtigt somit Oberflächeneigenschaften wie Glanz, Oberflächenbeschaffenheit oder Farbe (*Abbildung 3*). Sie hängt von der Richtung der einfallenden und reflektierten Strahlen ab. Wenn wir uns aber auf diffuse Strahlung beschränken, dann wird die BRDF unabhängig von der Richtung der Strahlen und wir können die *Radiosity Equation* aufschreiben:

$$L(y) = L_e(y) + \frac{\rho_d(y)}{\pi}\int_\Omega L(h(y, \vec{\omega}_i))\cos\theta_i d\omega_i$$

Hier haben wir $f_r = f_d := \rho_d(y)/\pi$ gesetzt. Dabei ist $\rho_d(y)$ die Reflexionsstärke der diffusen Oberfläche im Punkt y. Somit ist die Radianz $L(y)$ unabhängig von der Richtung der Strahlen.

Abbildung 3: die Bidirectional Reflectance Distribution Function (BRDF)

2.3 Die Detektorfunktion

Um nun das globale Beleuchtungsproblem zu lösen, müssen Funktionen der Form

$$\langle L, \Psi\rangle := \int_S\int_\Omega L(y, \vec{\omega})\Psi(y, \vec{\omega})\cos\theta d\omega dy$$

entweder unter Berücksichtigung der vollständigen BRDF oder in der *Radiosity*-Einstellung gelöst werden. S ist dabei die Oberfläche der Szene. Für die Detektorfunktion Ψ kann zum Beispiel die Summe von orthonormalen Basisvektoren gewählt werden, wie in den klassischen oder hierarchischen *Radiosity*-Methoden. Anstatt die Lösung auf diese Weise zu diskretisieren und dann in einem zweiten Schritt die Szene zu rendern, wählen wir für unseren Algorithmus die Funktion Ψ_{mn}, welche direkt die durchschnittliche Radianz durch den Pixel P_{mn} der Bildmatrix berechnet, die von einer Lochkamera im Fokuspunkt y_f aufgefangen wird (*Abbildung 4*). Dies wird im nächsten Abschnitt näher erläutert.

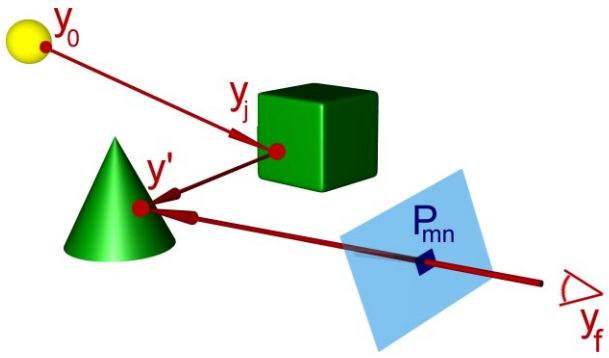

Abbildung 4: Der Algorithmus

3 Der Algorithmus

Die Idee hinter dem Algorithmus ist eine Lichtpartikelsimulation, welche durch den *Quasi-Random Walk* implementiert wird. Lichtpartikel werden von der Lichtquelle aus in die Szene geschossen, und dort, wo sie auftreffen, wird eine Lichtquelle erzeugt. Für jede so erzeugte Lichtquelle rendert die Grafikhardware die Szene mit Schatten und Texturen und legt sie in einem Akkumulationsbuffer ab. Dies entspricht der *Monte Carlo Integration*. Die globale Beleuchtung lässt sich zum Schluss aus dem Akkumulationsbuffer auslesen und auf dem Bildschirm anzeigen.

Dies kann man auch formal darstellen. Der Algorithmus berechnet die durchschnittliche Radianz

$$\begin{aligned} \bar{L}_{mn} := \langle L, \Psi_{mn} \rangle &= \langle L_e, \Psi_{mn} \rangle + \langle T_{f_r} L, \Psi_{mn} \rangle \\ &= \langle L_e, \Psi_{mn} \rangle + T_{mn} L \end{aligned} \tag{2}$$

welche durch den Pixel P_{mn} der Bildmatrix hindurchfliesst. Die Abkürzung $T_{mn}L$ definiert den Render-Operator, welcher die Radianz durch P_{mn} berechnet, welche auf ihrem Weg mindestens einmal reflektiert wurde.

Wir approximieren nun die Radianz L durch eine diskrete Dichte von M Punktlichtquellen:

$$L(y) \approx \sum_{i=0}^{M-1} L_i \delta(y - P_i)$$

Dabei ist L_i die Radianz und P_i die Position der i-ten Lichtquelle, δ ist das Kronecker-Delta. Diese Approximation in (2) eingesetzt ergibt die formale Beschreibung des Algorithmus:

$$L_{mn} \approx \langle L_e, \Psi_{mn} \rangle + \sum_{i=0}^{M-1} T_{mn} L_i \delta(y - P_i) \tag{3}$$

Der Render-Operator T_{mn} wird nun durch Grafikhardware berechnet, wie wir später in der Implementation sehen werden. Die sichtbaren Lichtquellen im Term $<L_e, \Psi_{mn}>$ können dabei direkt berechnet werden, indem den entsprechenden Oberflächenelementen Emission zugewiesen wird.

3.1 Die Quasi-Monte Carlo Integration

Die *Fredholm'sche Integralgleichung 2. Art* kann formal in eine Neumann Reihe überführt werden, welche aus einer Summe von Integralen besteht. Wegen der hohen Dimension der Integrale und den Diskontinuitäten der Integranden, welche typisch für Computergrafiken sind, versagen herkömmliche Quadraturformeln. So bleibt nur die Klasse der *Monte Carlo Methoden* geeignet. Diese benutzen uniforme (Pseudo-)Zufallszahlen, um ein Integral einer Funktion f auf dem s-dimensionalen Einheitscubus $I^s = [0, 1)^s$ zu approximieren, indem sie die Funktion f an Zufallspositionen $x_i \in I^s$, $i = 0, ..., N\text{-}1$ auswerten und den Durchschnitt bilden:

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} f(x_i)$$

Um die Konvergenzrate $O(N^{-1/2})$ zu verbessern, verwendet man speziell für die Integration entwickelte Zahlensequenzen. Die Uniformität ist dabei wichtiger als Zufälligkeit, so dass Sequenzen wie die Halton- oder Hammersley-Sequenz durch jegliche Zufallszahlentests fallen, da sie deterministisch gebildet werden. Daher kommt der Name „Quasi-Random".

S-dimensionale Zahlensequenzen, welche eine Diskrepanz von $D^* O((\log^S N)/N)$ aufweisen, gehören zu der Klasse der *Low Discrepancy Points*. Wie die Halton-Sequenz entsteht auch die Hammersley-Sequenz aus der *Radical Inverse Function*, welche eine natürliche Zahl i in ihre Repräsentation zur Basis b umwandelt und sie am Dezimalpunkt spiegelt. Die Hammersley-Sequenz hat sogar bloss die Diskrepanz $D^* O((\log^{s-1} N)/N)$. Sie ist jedoch nicht inkrementell, d.h. zuvor berechnete Werte müssen verworfen und neu berechnet werden, will man die Anzahl der Samples erhöhen. Dieser Umstand macht sie ungeeignet für adaptives Sampling.
Deshalb wird hier die Halton-Sequenz für den *Quasi-Random Walk* verwendet. Die *Radical Inverse Function* ist wie folgt definiert:

$$\Phi_b(i) := \sum_{j=0}^{\infty} a_j(i) b^{-j-1} \in [0,1) \Leftrightarrow i = \sum_{j=0}^{\infty} a_j(i) b^j$$

Mit diesen Funktionswerten bilden wir die Halton-Sequenz, welche eine Diskrepanz $D^* O((\log^S N)/N)$ besitzt:

$$x_i = (\Phi_{b_1}(i), \ldots, \Phi_{b_s}(i)), i \in \mathrm{N}_0$$

Die Basis b_j ist die j-te Primzahl. Jedes Segment $P_{N'}$ von einem grösseren Segment P_N, $N' < N$, ist ebenfalls eine Quadraturregel. Die Halton-Punkte können genauso schnell berechnet werden wie herkömmliche Pseudo-Zufallszahlen.

Der Vollständigkeit halber ist hier die Hammersley-Sequenz angegeben, welche analog gebildet wird:

$$x_i = \left(\frac{i}{N}, \Phi_{b_1}(i) \ldots, \Phi_{b_{s-1}}(i)\right), i \in \mathrm{N}_0$$

Auch hier bestehen die Zahlen x_i aus s-dimensionalen Vektoren.

In [HK94a] , [Kel95] und [Kel96b] finden sich numerische Untersuchungen für verschiedene Punktsequenzen. *Abbildung 5* zeigt die Konvergenz einiger Zahlensequenzen. Es wird hierbei von einer Masterkalkulation mit 640 Samples pro Pixel ausgegangen und beobachtet, wie schnell sich die Lösungen mit den verschiedenen Zahlensequenzen mit zunehmender Samplezahl N an diese annähern.

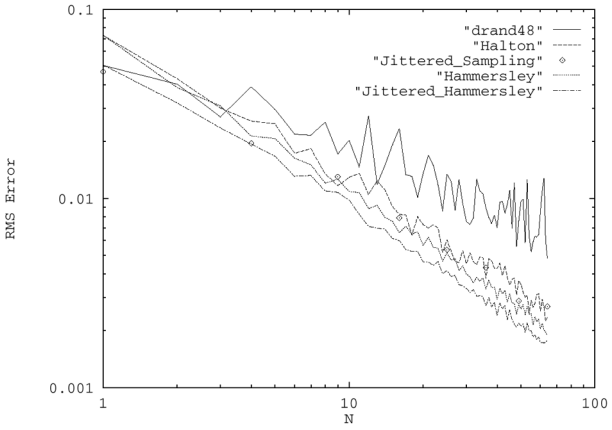

Abbildung 5: Konvergenzraten einiger Sequenzen

Diskrepanz D*: Die Diskrepanz $D^*(P_N)$ ist ein Mass für die Abweichung von einem Punktset $P_N = \{x_0, \ldots, x_{N-1}\}$ von der uniformen Verteilung. Etwas formaler ist $D^*(P_N)$ definiert als der grösste Integrationsfehler beim Integrieren der charakteristischen Funktionen aller Subcubi J von I^s, inklusive dem Ursprung:

$$D^*(P_N) := \sup_{J = \prod_{j=1}^{s} [0, a_j) \subset I^s} \left| \int_{I^s} \chi_J(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_J(x_i) \right|$$

3.2 Der Quasi-Random Walk

Wir können die Lösung der *Radiance Equation* (1) formal durch die Neumann-Reihe beschreiben:

$$L = (I - T_{f_r})^{-1} L_e = \sum_{j=0}^{\infty} T_{f_r}^{j} L_e$$

Da in realistischen Szenen die Norm des Transportoperators $|T_{fr}| < 1$ ist, konvergiert die Neumann-Reihe. Physikalisch bedeutet dies, dass weniger als 100% der einfallenden Lichtstrahlen reflektiert werden. Diese Summe kann auf diese Weise interpretiert werden: Mit jedem Glied kommt ein zusätzliches Reflexionsereignis hinzu, d.h. für $i = 1$ Einfachreflexion, für $i = 2$ Mehrfachreflexion, u.s.w.

Wenn wir die Neumann-Reihe in (2) einsetzen, so erhalten wir die Gleichung:

$$T_{mn}L = \frac{1}{|P_{mn}|} \sum_{j=0}^{\infty} \int_{P_{mn}} \int_{\Omega^j} \int_{S_e} p_j(y_0, \vec{\omega}_0, \ldots, \vec{\omega}_j) V(y_j, y') f_d(y') \frac{\cos\theta_j \cos\theta'}{|y_j - y'|^2} dy_0 d\omega_0 \ldots \omega_j dP$$

Als erstes wird über den Support $S_e := \text{supp } L_e \in S$ der Lichtquellen integriert, als nächstes über die einzelnen Hemisphären Ω_j der Aufprallpunkte der einzelnen Lichtstrahlen und dann über alle Pixel P_{mn} der Bildmatrix. Die äusserste Integration dient dem Oversampling, wenn man mehrere Strahlen durch einen Pixel P_{mn} der Bildmatrix schiesst. Der Punkt $y' = h(y_f, P - y_f)$ ist der erste Punkt, der getroffen wird, wenn wir einen Strahl von der Kamera im Fokuspunkt y_f aus durch den Pixel $P \in P_{mn}$ in die Szene schiessen. Die Funktion $V(y_j, y')$ überprüft die gegenseitige Sichtbarkeit der Punkte y_j und y'. Im Falle von Sichtbarkeit ergibt sie 1, ansonsten 0. *Abbildung 4* veranschaulicht die Situation. Hier taucht jetzt auch der Form Faktor auf, welcher die Positionen den Flächen zueinander berücksichtigt. Im Divisor steckt die bekannte Regel, das Licht mit dem Quadrat der Distanz abnimmt.

Die Radianz von einer Lichtquelle nach *j* Reflexionen wird durch den folgendem Term dargestellt:

$$p_j(y_0, \vec{\omega}_0 \ldots, \vec{\omega}_j) := L_e(y_0) \prod_{l=1}^{j} (\cos\theta_{l-1} f_d(y_l))$$

Der Punkt $y_0 \in S_e$ ist ein Punkt auf der Lichtquelle, die nachfolgenden Punkte y_l bilden den *Quasi-Random Walk*.

Die Transportoperator-Norm kann durch die Durchschnittsreflexivität $\bar{\rho}$ der Szene abgeschätzt werden:

$$\bar{\rho} := \frac{\sum_{k=1}^{K} \rho_{d,k} |A_k|}{\sum_{k=1}^{K} |A_k|} \approx \|T_{f_d}\|$$

Die Szene besteht hier aus K Oberflächenelementen A_k, welche eine Reflexivität von $\rho_{d,k}$ aufweisen. Der *Quasi-Random Walk* besteht jetzt darin, dass man N Lichtpartikel von einer Lichtquelle aus starten lässt. Von diesen N Lichtpartikeln überleben $\lfloor \bar{\rho} N \rfloor$ Partikel die erste Reflexion, die anderen werden verschluckt. Die zweite Reflexion überleben noch $\lfloor \bar{\rho}^2 N \rfloor$ Partikel, bis alle verschluckt worden sind. Auf den Aufprallorten werden Punktlichter generiert, und für jedes Punktlicht wird die Szene mit Schatten und Texturen durch die Grafikhardware berechnet. So werden in (3) mit N Punktlichtern der Term $T_{mn}L_e$ ausgewertet, mit $\lfloor \bar{\rho} N \rfloor$ Punktlichtern der Term $T_{mn}T_{fd}L_e$, mit $\lfloor \bar{\rho}^2 N \rfloor$ Punktlichtern der Term $T_{mn}T_{fd}^2L_e$ und so weiter. *Abbildung 6* illustriert den *Quasi-Random Walk*.

Die Anzahl *M* der so generierten Punktlichtquellen ist begrenzt durch:

$$M < \sum_{j=0}^{\infty} \bar{\rho}^{j} N = \frac{1}{1-\bar{\rho}^{j}} N$$

M ist somit linear in *N* und hängt von der durchschnittlichen Szenenreflexivität ab. Die Partikel werden mit Hilfe der Halton-Sequenz gebildet. Da die meisten Partikel für die tiefen Potenzen des Transportoperators verwendet werden, erhalten die Lichtstrahlen mit grossem Einfluss auch entsprechend Gewicht. Der *Quasi-Random Walk* nützt auch die Tatsache aus, das Teilsegmente der Halton Sequenz ebenfalls eine Quadraturregel bilden.

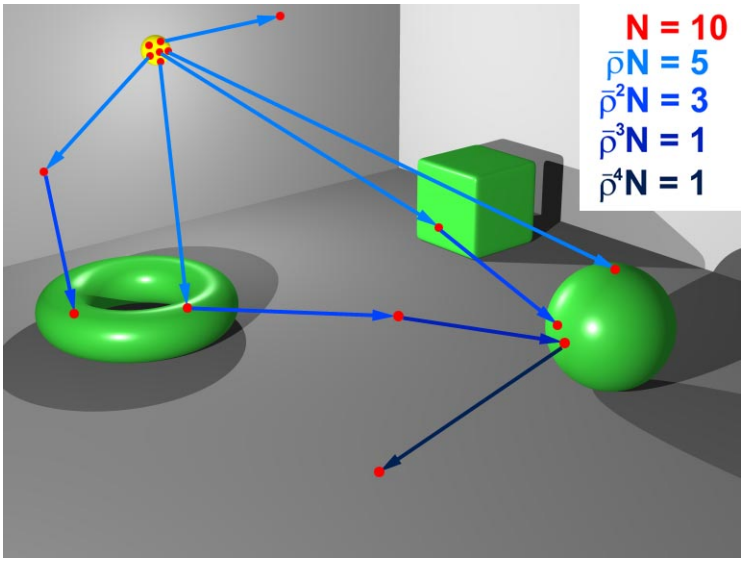

Abbildung 6: Der Quasi-Random Walk für N = 10

4 Implementation

```
Void InstantRadiosity(int N, double ρ̄)
{
          double ω, Start; int End, Reflections = 0;
          Color L; Point y; Vektor ω̄;

          Start = End = N;

          While(End > 0)
          {

          Start *= ρ̄;

          for(int i = (int) Start; i < End; i++)
          {
                    // select starting point on light source
                    y = y0(Φ2(i)Φ3(i));
                    L = Le(y) * supp Le;
                    ω = N;

                    // trace reflections
                    for(int j = 0; j <= Reflections; j++)
                    {
                              glRenderShadowedScene(N/⌊ω⌋ * L, y);
                              glAccum(GL_ACCUM, 1/N);
                              // diffuse scattering
                              ω̄ = (arcsin(Φ2j+2(i))^1/2, 2*π*Φ2j+3(i));
                              // trace ray from y into direction ω̄
                              y = h(y, ω̄);
                              // Attenuate and compensate
                              L *= fd(y);
                              ω *= ρ̄;
                    }
          }

          Reflections++;
          End = (int) Start;
          }

          glAccum(GL_RETURN, 1.0);
}
```

Um die diskrete Dichtefunktionsapproximation p_j zu berechnen, wird als erstes die Anzahl der Lichtpartikel, welche von der Lichtquelle aus starten, auf N gesetzt. Mit einer Isometrie y_0 werden die ersten zwei Komponenten der Halton-Sequenz vom Einheitsquadrat auf die Oberfläche der Lichtquelle gemappt, was den Startpunkt y ergibt. Die Lichtstärke L wird auf $L = L_e(y)$ supp L_e festgesetzt. Wenn mehrere Lichtquellen die Szene erhellen, wird eine ausgewählt und mit den anderen anschliessend analog verfahren.

Von diesen N Partikeln auf der Lichtquelle werden die ersten $\lfloor \bar{\rho} N \rfloor$ Partikel in die Szene hinein geschossen, d.h. sie überleben die erste Reflexion. Im Aufprallpunkt y werden sie mit der diffusen BRDF $f_d(y)$ attenuiert. Von diesen Partikeln werden die ersten $\lfloor \bar{\rho}^2 N \rfloor$ Partikel weiter verfolgt, bis keine Partikel mehr übrigbleiben.In jedem Punkt y wird eine Lichtquelle mit der Stärke $N/\lfloor \omega \rfloor L$ generiert und die Szene mit einem OpenGL-Aufruf mit Texturen und Schatten gerendert. Der Term $\omega = \bar{\rho}^j N$ sorgt dafür, dass jedes Bild eines Pfades gleiches Gewicht erhält.

Am Schluss wird die *Quasi-Monte Carlo Integration* durchgeführt, indem alle Bilder mit dem Gewicht $1/N$ akkumuliert werden.

5 Diskussion

5.1 Erweiterungen

Der Algorithmus kann durch folgende Punkte erweitert werden:

5.1.1 Jittered Low Discrepancy Sampling

Die Komponenten der Vektoren, welche durch die Haltonpunkte generiert werden, werden um einen gewissen Betrag verwackelt. Dadurch entsteht ein Antialiasing-Effekt, der entsprechende Hardware überflüssig macht. Der Betrag, um den die Punkte verschoben werden, nimmt dabei mit der Tiefe der *Quasi-Random Walk* Pfade ab. Nebenbei wird dadurch auch die Konvergenzrate positiv beeinflusst.

5.1.2 Spekulare Effekte

Um spekulare Effekte hinzuzufügen, lassen wir die Hardware mit der vollständigen BRDF f_r rechnen. Jede Oberfläche wird dabei zufällig getestet, ob sie gemäss ihrer BRDF spekular oder diffus ist. Im Falle einer spekularen Oberfläche wird der Ursprung der einfallenden Strahles an der Oberfläche gespiegelt und somit eine virtuelle Lichtquelle generiert. Diese beleuchtet nun die Szene innerhalb der Pyramide, welche von der virtuellen Lichtquelle und der Kontur des spekularen Oberflächenelements aufgespannt wird. Anschliessend wird das Lichtpartikel gemäss der BRDF weitergestreut. Die virtuellen Lichtquellen verlängern die Pfade des *Quasi-Random Walk* um eine zufällige Länge. Die sichtbaren spekularen Objekte müssen aber in einem zweiten Durchgang mit Ray Tracing separtat behandelt werden.

5.1.3 Echtzeitanimationen

Die Pfadlänge wird auf einen Wert l_{max} festgesetzt. Dabei werden die Bilder eines Pfades wie beschrieben berechnet und im Akkumulationsbuffer abgelegt. Im Akkumulationsbuffer befinden sich jeweils N Bilder der letzten N Pfade. Sobald ein Pfad fertig berechnet wurde, wird das älteste Bild im Buffer durch das neue überschrieben. Die N Bilder werden dann akkumuliert und angezeigt. So entsteht ebenfalls eine Art Antialiasing. Die Bildrate der Grafikhardware wird dabei um den Wert 1/ l_{max} herabgesetzt. L_{max} ist der längste vorkommende Pfad in einem Quasi-Random Walk bei gegebenem N und lässt sich berechnen:

$$l_{max} := \left\lfloor -\frac{\log N}{\log \bar{\rho}} \right\rfloor$$

Für *Abbildung 8* ergibt dies für $\bar{\rho} = 0.5774$ und $N = 64$ den Wert $l_{max} = 7$, dass heisst die Bildrate der Grafikhardware wird um 1/7 abgebremst.

6 Diskussion

6.1 Eigenschaften

Die folgenden Punkte kennzeichnen den Algorithmus Instant Radiosity:

- Die Geschwindigkeit des Algorithmus hängt hauptsächlich von der Hardware ab, da die Partikelsimulation sofort (instantly) generiert werden kann.
- Der Algorithmus ist einfach zu implementieren.
- Der Algorithmus arbeitet direkt auf der Szenenbeschreibung – dies kommt ihm besonders auch bei animierten Szenen zugute, wo sich Objekte bewegen und somit die globale Beleuchtung ändert.
- Die Lösung der Radiance Equation wird direkt berechnet – somit wird kein zusätzlicher Speicher für Zwischenresultate benötigt.
- Der einzige zusätzliche Speicher wird für Strukturen für die Szene wie z.B. BSP-trees benötigt, um das Strahlenschiessen h(y, ?) zu beschleunigen.
- Die Zeitkomplexität ist beschränkt durch $O(NK)$, wobei N die Anzahl der Pfade ist und K die Anzahl Elemente in der Szene.

Zwei negative Effekte treten bei niedrigen Samplingraten N auf:

- Die schwache Singularität des Transportoperators T_{mn} macht sich störend bemerkbar, indem Flächen zu dunkel berechnet werden. Wenn die Distanz $|y_j - y'|^2$ zwischen der Lichtquelle und dem zu beleuchtenden Objekt gegen Null geht, wird der Wert, der in den Framebuffer geschrieben werden soll, übermoduliert und wird geklippt zum maximal darstellbaren Wert.
- Durch Texturen eingefärbte Lichtquellen haben aufgrund der zu kleinen Anzahl Samples einen zu geringen Einfluss auf die Szene. Eine rote Wand z.B. färbt dann zu wenig ab auf die Objekte in der Umgebung

Ein weiterer Nachteil ist die hohe Hardwareanforderung. Dies ist aber auch ein Vorteil, da vorhandene Hardware direkt unterstützt wird.

6.2 Zukünftige Arbeit

Die Aufmerksamkeit wird auf das effiziente Sampeln von Lichtquellen im Falle von mehreren Lichtquellen gelegt. Lichtquellen mit besonders grossem Einfluss auf die Szene sollen durch mehr Lichtpartikel approximiert werden als schwache Lichtquellen. Insbesondere bei grossen oder indirekt beleuchteten Szenen würde eine solche Reduktion der Anzahl *M* der generierten Punktlichtquellen einen spürbaren Geschwindigkeitsvorteil bringen. Ausserdem soll der Einfluss der schwachen Singularität des Transportoperators abgeschwächt werden. Der Algorithmus lässt sich auch einfach durch Streuung durch Volumina hindurch erweitern. Vorausgesetzt, die Grafikhardware beherrscht Fog Attenuation, lassen sich so auch partizipierende Medien wie Nebel oder Rauch einbinden.

Referenzen

[CW93] M. Cohen and J. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Cambridge 1993.

[HK94a] S.Heinrich, A.Keller. *Quasi-Monte Carlo methods in computer graphics, Part I: The QMC-Buffer*, Technical Report 242/94, University of Kaiserslautern, 1994.

[HK94b] S.Heinrich, A.Keller. *Quasi-Monte Carlo methods in computer graphics, Part II: The Radiance Equation*, Technical Report 243/94, University of Kaiserslautern, 1994.

[Kel95] A. Keller. *A Quasi-Monte Carlo Algorithm for the Global Illumination Problem in the Radiosity Setting*. In H. Niederreiter and P. Shiue, editors, Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing. Springer 1995.

[Kel96b] A. Keller. *Quasi-Monte Carlo Radiosity*. In X. Pueyo and P. Schröder, editors, Rendering Techniques ’96. Springer, 1996.

[Kel97] A. Keller. *Instant Radiosity*. Proceedings of ACM SIGGRAPH 97.

[Slu97] Philipp Slusallek. *Photo-Realistic Rendering – Recent Trends and Developments*. EUROGRAPHICS ’97

Demoprogramme zu diesem Algorithmus sind unter http://www.uni-kl.de/AG-Heinrich/Alex.html verfügbar.

7 Beispielbilder

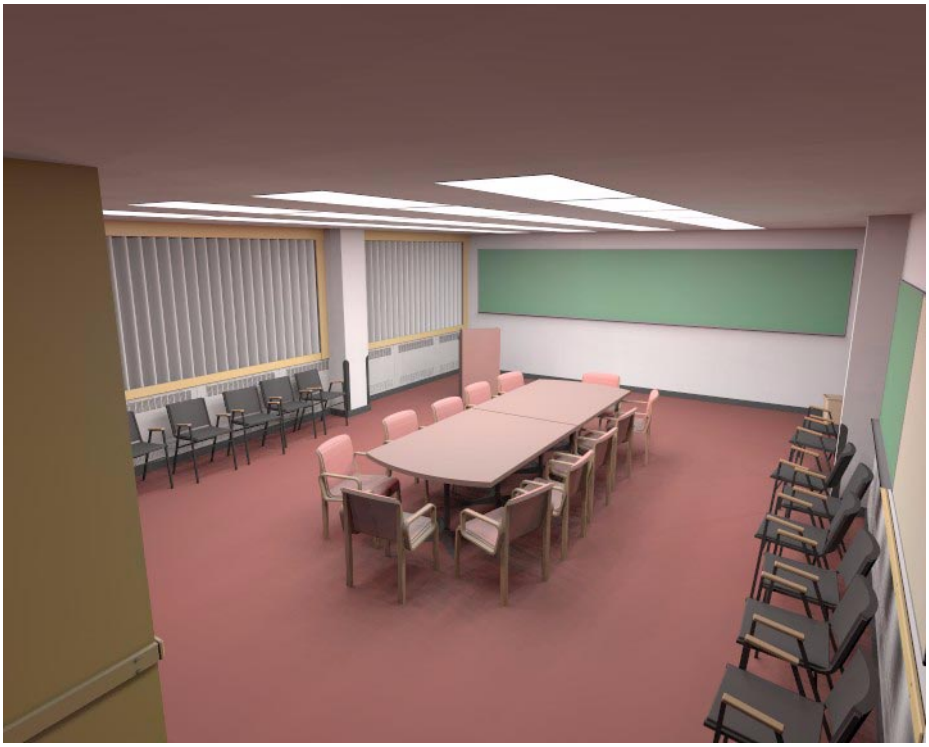

Abbildung 7: Konferenzraum – Bild für N = 128

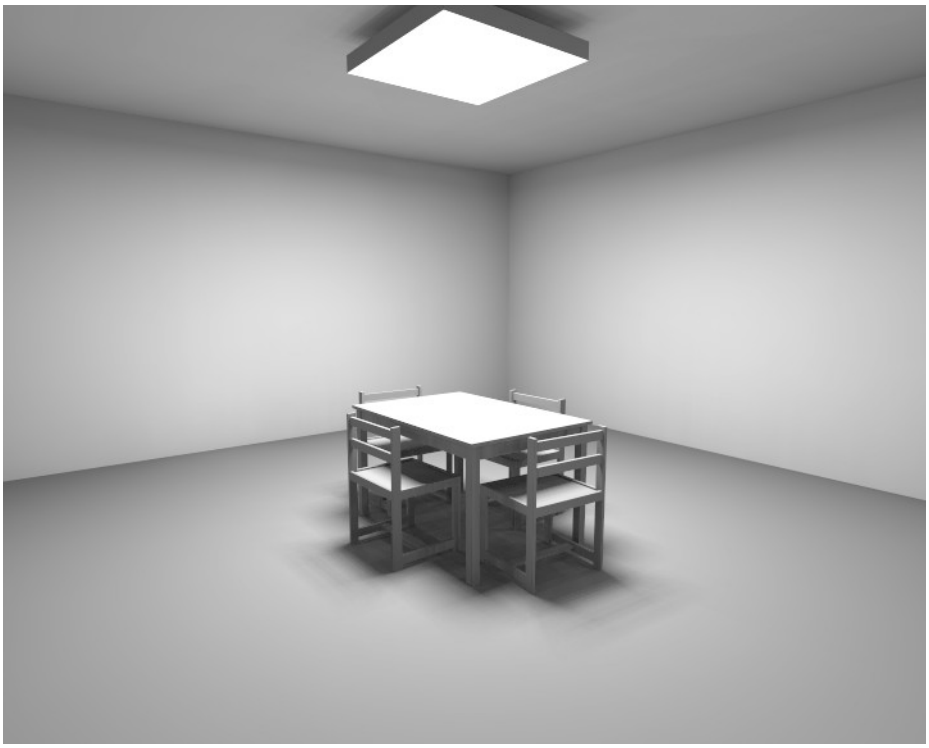

Abbildung 8: SGI Onyx mit Reality Engine2, 75 Mhz R8000: 24 Sekunden