Kapitel 22

Radiosity

22.1 Globale Beleuchtung

Ein Beleuchtungsmodell berücksichtigt bei der Berechnung der Farbe für einen Punkt das von den Lichtquellen direkt abgegebene Licht und das Licht, das den Punkt nach Reflexion und Transmission durch seine eigene und andere Flächen erreicht. Dieses indirekt reflektierte und hindurchgelassene Licht heißt *globale Beleuchtung*. Als *lokale Beleuchtung* bezeichnet man das Licht, das direkt von den Lichtquellen auf den schattierten Punkt fällt. Bisher wurde die globale Beleuchtung durch einen Term für ambiente Beleuchtung modelliert, der für alle Punkte auf allen Objekten konstant war. Dieser Term berücksichtigte weder die Positionen von Objekt und Betrachter noch Objekte, die das Umgebungslicht blockieren könnten.

In realen Szenen kommt nur ein geringer Teil des Lichts aus direkten Lichtquellen. Es gibt zwei Klassen von Algorithmen zur Erzeugung von Bildern, die die Bedeutung globaler Beleuchtung hervorheben. Das nächste Kapitel behandelt den Ray Tracing-Algorithmus, der neben der Ermittlung sichtbarer Flächen und deren Schattierung gleichzeitig Schatten, Reflexion und Brechung berechnet. Globale spiegelnde Reflexion und Transmission ergänzen dabei die für eine Fläche berechnete lokale spiegelnde, diffuse und ambiente Beleuchtung. Im Gegensatz dazu trennen die in diesem Kapitel behandelten Radiosity-Verfahren die Schattierung völlig von der Ermittlung sichtbarer Flächen. Sie berechnen erst in einem vom Blickpunkt unabhängigen Schritt alle Interaktionen einer Szene mit den Lichtquellen. Dann berechnen sie mit konventionellen Algorithmen zur Ermittlung sichtbarer Flächen und Schattierung durch Interpolation ein oder mehrere Bilder für die gewünschten Standpunkte des Betrachters.

Vom Blickpunkt abhängige Algorithmen (z.B. Ray Tracing) diskretisieren die Bildebene, um die Punkte zu ermitteln, an denen die Beleuchtungsgleichung für eine bestimmte Blickrichtung ausgewertet wird. Beleuchtungsalgorithmen, die vom Blickpunkt unabhängig sind (z.B. Radiosity), diskretisieren dagegen die Szene und verarbeiten sie weiter, um genügend Informationen zur Auswertung der Beleuchtungsgleichung an jedem beliebigen Punkt und für jede Blickrichtung zu erhalten. Die Algorithmen, die vom Blickpunkt abhängig sind, eignen sich gut zur Behandlung von Spiegelungen, die stark vom Standpunkt des Betrachters abhängen. Sie erfordern jedoch bei der Behandlung diffuser Phänomene zusätzlichen Aufwand, da sich diese über große Bildbereiche oder zwischen Bildern aus verschiedenen Blickpunkten wenig ändern. Die Algorithmen, die nicht vom Blickpunkt abhängen, modellieren diffuse Phänomene dagegen effizient, haben aber bei der Behandlung von Spiegelungen

einen enorm hohen Speicherbedarf.

22.2 Physikalische Ausgangslage

Jeder von einer Fläche abgestrahlten oder reflektierten Energie entspricht die Reflexion oder Absorption durch andere Flächen. Die gesamte von einer Fläche abgegebene Energie heißt *Strahlung* oder *Radiosity*. Sie besteht aus der Summe der abgestrahlten Energie, der reflektierten Energie und der Energie, die durch die Fläche hindurchtritt. Ansätze, die die Strahlung der Flächen einer Szene berechnen, heißen daher *Radiosity-Verfahren*. Im Gegensatz zu konventionellen Rendering-Algorithmen berechnen Radiosity-Verfahren erst unabhängig vom Blickpunkt alle Lichtinteraktionen einer Szene. Die Terme für ambiente und diffuse Beleuchtung müssen dann beim Einfärben nicht mehr berechnet werden, da sie wesentlich realistischer bereits in den Radiosity-Werten enthalten sind. Dann werden eine oder mehrere (von unterschiedlichen Augenpunkten betrachtete) Darstellungen gerastert. Dabei fällt nur noch der Aufwand für die Ermittlung der sichtbaren Flächen und für die Schattierung durch Interpolation an.

22.3 Die Radiosity-Gleichung (Beleuchtungsgleichung)

Die bisher betrachteten Schattierungsalgorithmen behandeln die Lichtquellen immer unabhängig von den beleuchteten Flächen. Bei den Radiosity-Verfahren kann dagegen jede Fläche Licht abstrahlen. Daher werden alle Lichtquellen mit einem inhärenten Flächeninhalt modelliert. Die Szene wird in eine endliche Anzahl n diskreter Flächenelemente (*Patches*) zerlegt. Jedes dieser Elemente hat endliche Größe, strahlt über die gesamte Fläche gleichmäßig Licht ab und reflektiert Licht. Wenn man jedes der n Flächenelemente als opaken, diffusen Lambertschen Strahler und Reflektierer betrachtet, gilt für Fläche i

$$B_i \cdot A_i = E_i \cdot A_i + \rho_i \sum_{j=1}^{n} B_j \cdot F_{ji} \cdot A_j, 1 \leq i \leq n \; .$$

mit

E_i	=	von der Fläche i abgegebene Eigenstrahlung pro Flächeneinheit
ρ_i	=	Reflexionsvermögen der Fläche i
n	=	Anzahl der Flächen
A_i	=	Größe der Fläche i
F_{ji}	=	Anteil an der von Fläche j abgegebenen Energie, die auf Fläche i auftrifft, pro Flächeneinheit, genannt *Formfaktor*
B_i	=	von Fläche i abgestrahlte Energie (Summe aus Eigenstrahlung und Reflexion als Energie pro Zeit und pro Flächeneinheit), genannt *Radiosity* der Fläche i

Die Gleichung besagt, daß die Energie, die eine Flächeneinheit verläßt, aus der Summe des abgestrahlten und des reflektierten Lichts besteht. Das reflektierte Licht berechnet sich aus der Summe des einfallenden Lichts, multipliziert mit dem Reflexionsvermögen. Das einfallende Licht besteht wiederum aus der Summe des Lichts, das alle Flächen der Szene verläßt, multipliziert mit dem Lichtanteil, der eine Flächeneinheit des empfangenden Flächenelements erreicht. $B_j \cdot F_{ji} \cdot A_j$ ist der Betrag des Lichts, das die ganze Fläche j verläßt und die Fläche i erreicht.

Zwischen den Formfaktoren in diffusen Szenen besteht eine nützliche Beziehung:

$$A_i \cdot F_{ij} = A_j \cdot F_{ji} \ ,$$

wobei A_i und A_j die Flächeninhalte sind. Daraus ergibt sich

$$F_{ij} = \frac{A_j}{A_i} \cdot F_{ji}$$

Umordnen der Ausdrücke liefert

$$B_i - \rho_i \sum_{1 \le j \le n} B_j \cdot F_{ij} = E_i \ .$$

Also kann der Austausch von Licht innerhalb der Flächenelemente der Szene durch ein Gleichungssystem ausgedrückt werden:

$$\begin{pmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \dots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \dots & -\rho_2 F_{2n} \\ . & . & . & \\ . & . & . & \\ . & . & . & \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \dots & 1-\rho_n F_{nn} \end{pmatrix} \cdot \begin{pmatrix} B_1 \\ B_2 \\ . \\ . \\ . \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ . \\ . \\ . \\ E_n \end{pmatrix}$$

Man beachte, daß der Beitrag eines Flächenelements zu seiner eigenen reflektierten Energie berücksichtigt werden muß (es könnte zum Beispiel konkav sein). Daher haben im allgemeinen nicht alle Terme auf der Diagonalen den Wert Eins.

Wenn man die Gleichung löst, erhält man für jedes Flächenelement einen Strahlungswert. Die Elemente können dann für jeden gewünschten Blickpunkt mit einem gewöhnlichen Algorithmus zur Ermittlung sichtbarer Flächen gerastert werden. Die Strahlungswerte sind die Intensitäten dieses Elements.

Gauß-Seidel-Iterations-Verfahren zur Lösung von linearen Gleichungssystemen

Die i-te Gleichung eines linearen Gleichungssystems $Ax = b$ lautet:

$$\sum_{j=1}^{n} A[i,j]x[j] = b[i] \ .$$

Wenn alle Diagonalelemente von A ungleich Null sind, gilt:

$$x[i] = \frac{1}{A[i,i]} \left(b[i] - \sum_{j \ne i} A[i,j]x[j] \right) \ .$$

Das Iterations-Verfahren startet mit einer Schätzung x_1, die auf der rechten Seite eingesetzt wird zur Berechnung von $x_1[1]$. Im nächsten Schritt wird zur Berechnung von $x_1[2]$ bereits $x_1[1]$ benutzt. Ein Iterationsschritt lautet also:

$$x_k[i] = \frac{1}{A[i,i]} \left(b[i] - \sum_{j=1}^{i-1} x_k[j]A[i,j] - \sum_{j=i+1}^{n} x_{k-1}[j]A[i,j] \right) \ .$$

Bei Konvergenz wird das Verfahren nach k Iterationsschritten abgebrochen, wenn $b - Ax_k$ genügend klein geworden ist.

22.4 Berechnung der Formfaktoren

Die Formfaktoren lassen sich definieren über die Gleichung

$$F_{ij} = \frac{1}{A_i} \int_{F_i} \int_{F_j} \frac{\cos(\phi_i)\cos(\phi_j)}{\pi r_{ij}^2} b_{ij} dF_j dF_i$$

mit

ϕ_i	=	Winkel zwischen Normale auf Fläche i und Verbindungslinie zwischen Fläche i und Fläche j
ϕ_j	=	Winkel zwischen Normale auf Fläche j und Verbindungslinie zwischen Fläche i und Fläche j
r_{ij}	=	Entfernung zwischen Fläche dF_i und Fläche dF_j
b_{ij}	=	Blockierungsfunktion, falls Teile von Fläche j aus der Sicht von Fläche i verdeckt sind.
A_i	=	Flächeninhalt von Fläche F_i

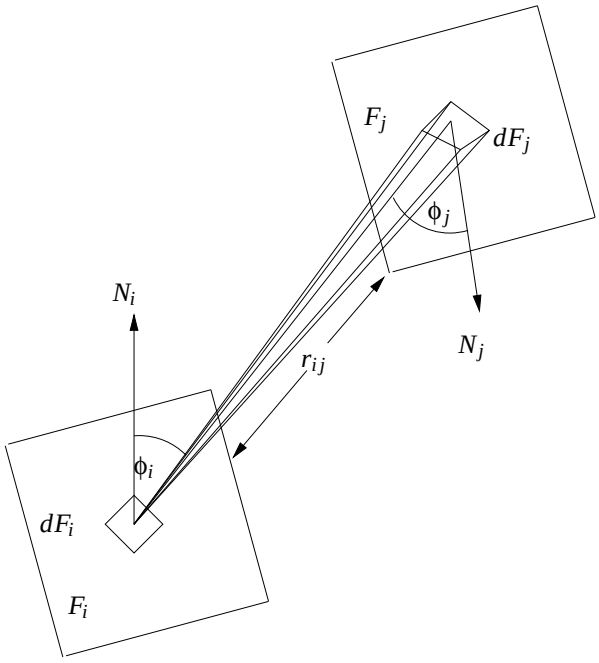


Abbildung 22.1: Der Berechnung von Formfaktoren zugrundeliegende Geometrie

Die geometrische Interpretation beschreibt den Formfaktor als das Verhältnis der Basisfläche einer Halbkugel zur Orthogonalprojektion der auf die Halbkugel projizierten Fläche: Erst projiziert man die von F_i aus sichtbaren Teile von F_j auf eine Halbkugel mit Radius 1 um dF_i, projiziert diese Projektion orthogonal auf die kreisförmige Grundfläche der Halbkugel und dividiert schließlich durch die Kreisfläche. Die Projektion auf die halbe Einheitskugel entspricht in der Gleichung dem Term $\cos\phi_j / r_{ij}^2$, die Projektion auf die Grundfläche entspricht der Multiplikation mit $\cos\phi_i$, und die Division durch

den Flächeninhalt des Einheitskreises liefert den Wert π im Nenner.

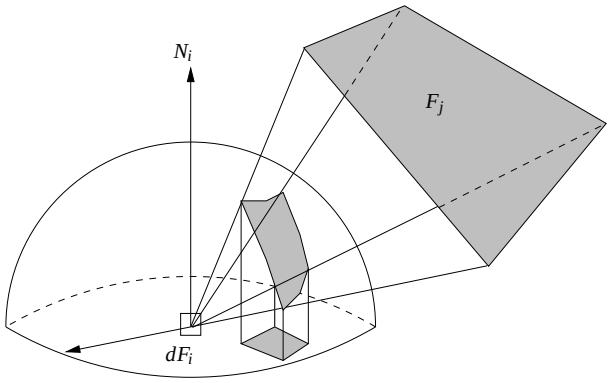


Abbildung 22.2: Geometrische Interpretation des Formfaktors

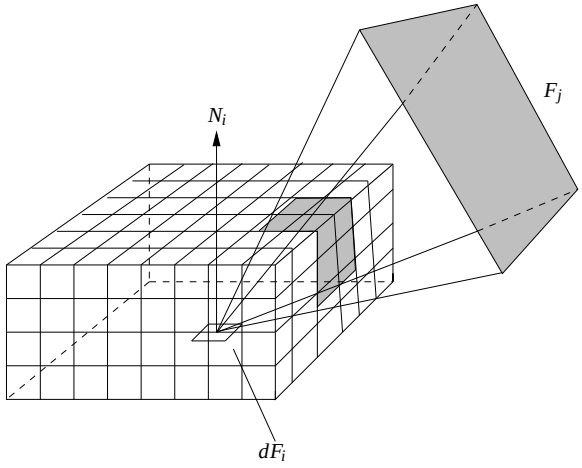


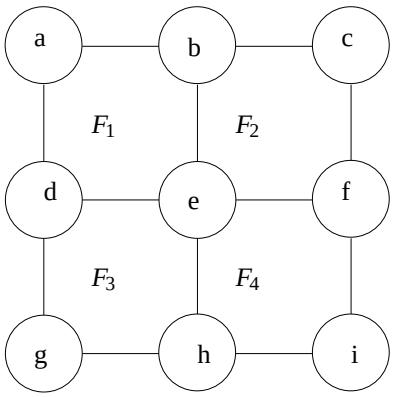
Abbildung 22.3: Simulation der Halbkugel durch Halbwürfel

Zur numerischen Berechnung wird die Halbkugel durch einen *Halbwürfel* (hemi-cube) mit dem Zentrum im Ursprung und dem Normalvektor in der z-Achse ersetzt. Die Oberseite des Würfels ist dabei parallel zur Fläche. Jede Seite des Halbwürfels wird in ein Raster gleich großer quadratischer Zellen aufgeteilt. (Die Auflösungen reichen von 50×50 bis zu mehreren Hundert pro Seite.) Dann wird jedes Flächenelement auf die Seiten des Halbwürfels projiziert. Jeder Zelle des Halbwürfels ist ein *Delta-Formfaktor* zugeordnet, der von der Position der Zelle abhängt und vorher berechnet wird. Für eine beliebig feine Rasterung der Quaderoberfläche ergibt sich der Formfaktor F_{ij} als Summe aller von der Fläche F_j überdeckten Rasterzellen. Wird eine Rasterzelle von mehreren Flächen überdeckt, wird sie der Fläche mit der geringsten Entfernung zugerechnet. Für die Summe aller Formfaktoren einer Fläche F_i gilt

$$\sum_{j=1}^{n} F_{ij} = 1 \; .$$

22.5 Interpolation der Pixelfarben

Für jede Fläche i liege nun ihre *Radiosity* B_i vor. Hieraus interpoliert man die Strahlung für die Eckpunkte:



$$
\begin{array}{lcllcl}
B(e) & = & (B_1+B_2+B_3+B_4)/4 & & & \\
B(a) & = & B_1+(B_1-B(e)) & B(c) & = & B_2+(B_2-B(e)) \\
B(g) & = & B_3+(B_3-B(e)) & B(i) & = & B_4+(B_4-B(e)) \\
 & & & & & \\
B(b) & = & (B(a)+(B(c))/2 & B(d) & = & (B(a)+B(g))/2 \\
B(f) & = & (B(c)+(B(i))/2 & B(h) & = & (B(g)+B(i))/2
\end{array}
$$

Nun wird jedes Patch in zwei Dreiecke zerteilt, so daß nach der Projektion die Strahlungswerte an den Dreiecksecken zur zweifachen Interpolation verwendet werden können.

22.6 Schrittweise Verfeinerung

In Anbetracht des hohen Aufwands beim bisher beschriebenen Radiosity-Algorithmus stellt sich die Frage, ob man die Ergebnisse der Beleuchtungsberechnung inkrementell approximieren kann. Die Auswertung der i-ten Zeile des Gleichungssystems liefert eine Schätzung für die Strahlung B_i des Flächenelements, die auf den Schätzungen für die Strahlungswerte der anderen Flächenelemente basiert. Jeder Term in der Summe der Gleichung beschreibt die Auswirkung des Elements j auf die Strahlung des Elements i:

$$B_i \text{ updaten durch } \sum_j^n \rho_i B_j F i j$$

Diese Methode “sammelt” also das Licht der restlichen Szene ein.

Der Ansatz zur schrittweisen Verfeinerung verteilt dagegen die Strahlung eines Flächenelements auf die Szene, wobei jede Fläche j mithilfe von Fläche i aktualisiert wird:

$$B_j \text{ updaten durch } \rho_j B_i F ji$$

Wenn man eine Schätzung für B_i hat, kann man den Beitrag des Flächenelements i zum Rest der Szene ermitteln, indem man vorstehende Gleichung für jedes Element j auswertet. Dazu braucht man leider F_{ji} für alle j. Jeder dieser Werte wird mit einem separaten Halbwürfel bestimmt. Dies erfordert ebensoviel Aufwand an Speicherplatz und Rechenzeit wie der ursprüngliche Ansatz. Man kann die Gleichung jedoch umschreiben, wenn man die Reziprozitätsbeziehung berücksichtigt.

$$\text{Beitrag von } B_i \text{ zu } B_j = \rho_j B_i F_{ij} \frac{A_i}{A_j} \text{ für alle } j \ .$$

Zur Auswertung dieser Gleichung für alle j sind nur die Formfaktoren nötig, die mit einem einzigen Halbwürfel um das Flächenelement i berechnet wurden. Kann man die Formfaktoren des Elements i schnell berechnen (z.B. mit z-Puffer-Hardware), kann man sie wieder löschen, sobald die Strahlungen vom Flächenelement i aus berechnet sind. Man muß also immer nur einen einzigen Halbwürfel und dessen Formfaktoren gleichzeitig berechnen und speichern.
Sobald die Strahlung eines Elements verteilt wurde, wählt man ein anderes Element aus. Ein Element kann wieder Strahlung verteilen, sobald es neues Licht von anderen Elementen erhält. Dabei wird nicht die gesamte geschätzte Radiosity des Elements i verteilt, sondern nur der Betrag ΔB_i, den das Element i seit dem letzten Verteilen empfing. Der Algorithmus läuft so lange weiter, bis die gewünschte Genauigkeit erreicht ist. Es ist sinnvoll, das Element mit der größten Differenz zu nehmen, statt die Elemente in zufälliger Reihenfolge auszuwählen. Man wählt also das Element, das noch am meisten Energie abzustrahlen hat. Da die Strahlung pro Flächeneinheit gemessen wird, wählt man ein Flächenelement, bei dem $\Delta B_i F_i$ maximal ist. Am Anfang gilt für alle Flächenelemente $B_i = \Delta B_i = E_i$. Dieser Wert ist nur bei Lichtquellen ungleich Null.

Im folgenden bezeichnet ΔB_i also die noch nicht verteilte Strahlung, d.h. die Differenz zwischen der *Radiosity* im letzten und im gegenwärtigen Iterationsschritt. Es wird ausgenutzt, daß gilt $F_{ji} = (F_{ij} \cdot A_i)/A_j$.

```
for i:= 1 to n do                          {initialisiere}
   if patch i ist Lichtquelle              {für jede Fläche}
     then Bi := ΔBi := Emissionswert       {Strahlung und Differenz}
     else Bi := ΔBi := 0
end;

repeat
   for {jede Fläche i, beginnend bei größter Ausstrahlung} do begin
     Plaziere Hemicube auf Fläche i
     Berechne Fij für alle 1 ≤ j ≤ n
     for j := 1 to n do begin              {für jede Fläche j tue}
        ΔR := ρj * ΔBi * Fij * Ai/Aj       {Strahlung von Fläche i}
        ΔBj := ΔBj + ΔR                    {Differenz erhöhen}
        Bj := Bj + ΔR                      {Strahlung erhöhen}
     end;
     ΔBi := 0;                             {Überschuß ist verteilt}
   end
until fertig                               {bis zur Konvergenz}
```

Bei jeder Ausführung der äußeren FOR-Schleife verteilt ein weiteres Flächenelement seine unver-

brauchte Strahlung auf die Szene. Daher werden nach der ersten Ausführung nur die Flächen beleuchtet, die selbst Lichtquellen sind sowie solche, die beim Verteilen der Strahlung des ersten Elements direkt beleuchtet werden. Rastert man am Ende jeder Ausführung des Codes ein neues Bild, so wird das erste Bild relativ dunkel und die nachfolgenden Bilder immer heller.

Abbildung 22.4 faßt den gesamten Ablauf zusammen.

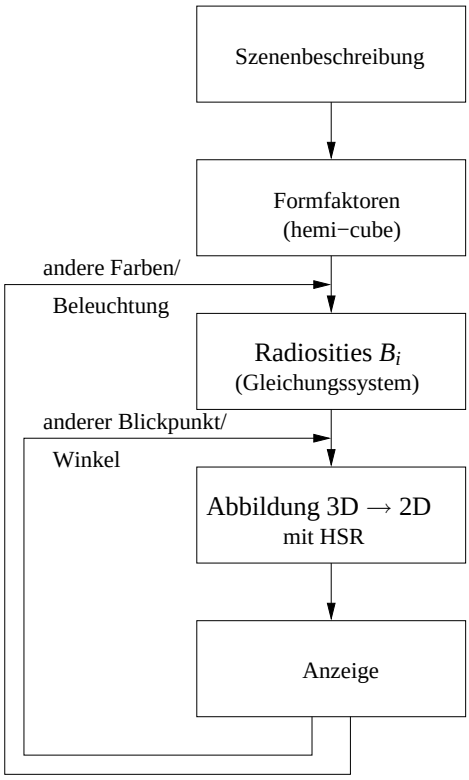
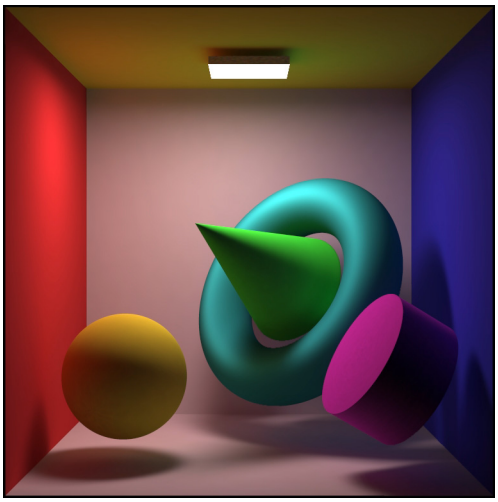


Abbildung 22.4: Ablaufschema beim Radiosityverfahren

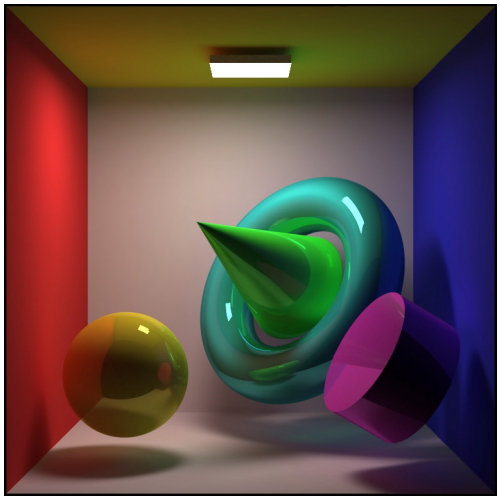
22.7 Screenshots

Auf der Seite `http://www.graphics.cornell.edu/online/research` gibt es einige Beispiele für Radiosity-Bilder.

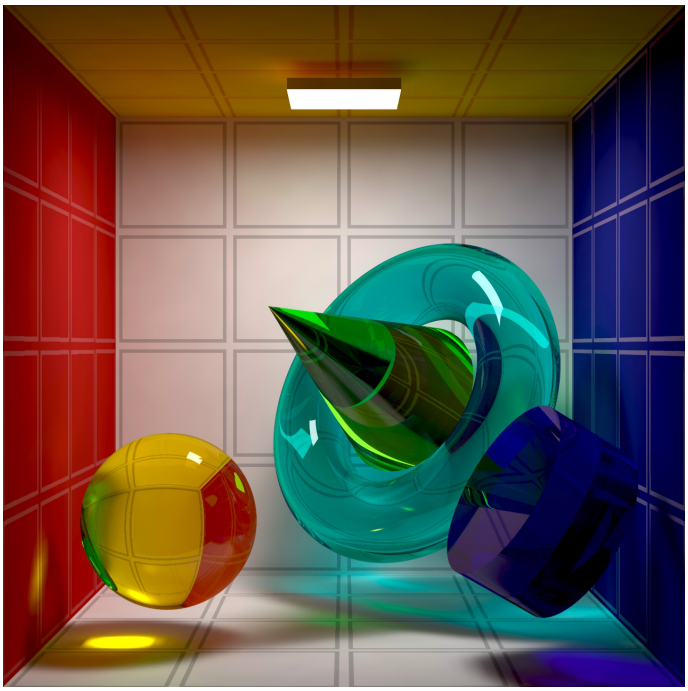
Die nächsten drei Bilder sind Screenshots vom Advanced Rendering Toolkit der Technischen Universität Wien (`http://www.cg.tuwien.ac.at/research/rendering/ART`). Hier wurde das klassische Radiosity-Verfahren um Ray-Tracing-Komponenten erweitert.



Diffuse Reflexion



Diffuse und spekulare Reflektion



Diffuse, spekulare und transparente Reflexion