Universität Leipzig
Institut für Informatik
Abteilung Bild- und Signalverarbeitung

Computergrafik-Praktikum
Sommersemester 2013
Betreuer: Steven Schlegel (schlegel@informatik.uni-leipzig.de)

Aufgabenblatt 3 – Abgabe in der Woche vom 17. Juni

Spiegelung und Transparenz

Aufgabe 1: Spiegelnde Flächen

Um spiegelnde Flächen zu rendern, benötigen wir einen Wert, der uns sagt, wie sehr eine Fläche/Material einfallendes Licht zurückwirft.

Hinweis: Im OBJ-Format kann der Wert *sharpness* herangezogen werden. Laut MTL-Spezifikation kann er einen Wert von 0 bis 1000 annehmen. *Blender* beherrscht zwar spiegelnde Flächen, scheint sie jedoch nicht korrekt in das OBJ-Format zu exportieren, wenn keine separate Reflectionmap angegeben ist. Der Sharpness-Wert sollte also von Hand nachgetragen werden. Wird ein spiegelndes Objekt in den *CGViewer* eingeladen, so teilt er den Sharpness-Wert automatisch durch 1000, um einen Wert zwischen 0 und 1 zu erhalten.

1 steht für einen perfekten Spiegel, wohingegen bei 0 das Material kein Licht zurückwirft.
Ein Wert A zwischen 0 und 1 bedeutet also, dass das Material partiell spiegelnd ist. In diesem Fall wird die Farbe nur zu einem Anteil von A durch I_λ aus dem Beleuchtungsmodell bestimmt. Der verbleibende Anteil von (1-A) wird durch rekursive Verfolgung des reflektierten Strahls ermittelt. Die Formel für den reflektierten Strahl ist dem letzten Aufgabenblatt zu entnehmen.

Im Quellcode sind folgende Member wichtig:
Struktur *Material:* *sharpness* – der Spiegelungsfaktor

Aufgabe 2: Transparente Flächen

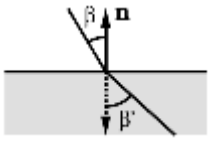

Abbildung 1: Brechung eines Strahls am Materialübergang.

In der Natur stellen Flächen Übergänge unterschiedlicher Materialien dar. In diesem Fall verringert oder erhöht sich der Winkel eines Strahls zur Flächennormalen beim Austreten auf der anderen Flächenseite, siehe Abbildung 1. Je nach Richtung des Strahls, bezogen auf die ursprüngliche Orientierung der Flächennormale, wird der Winkel mit einem Brechungsfaktor multipliziert oder durch selbigen geteilt. Der Brechungsfaktor ist durch den *density* – Wert bestimmt. Ein Wert von 1.0 bedeutet, dass das Licht beim Materialdurchgang nicht gebrochen wird (Glass z.B. hat einen Faktor um 1.5). Werte unter 1.0 sollten nicht genommen werden, da sie merkwürdige Artefakte produzieren können.
Erhöht sich der Winkel auf über 90°, so wird der Strahl reflektiert statt gebrochen. (Für ein korrektes Rendering müsste die Brechung auch bei der Auswertung des Beleuchtungsmodells berücksichtigt werden, was nicht zu realisieren ist, da die Strahlen immer in einer Lichtquelle enden müssen. Hier helfen andere Verfahren, wie z.B. Radiosity.)

Ob ein Material überhaupt transparent ist oder nicht bestimmt der *alpha* - Wert. Bei 1.0 ist das Objekt undurchsichtig, bei 0.0 völlig durchsichtig (unsichtbar). Im CGViewer-Format liegt der Wert im Bereich [0, 1]. Wie im spiegelnden Fall wird der gebrochene Strahl fortgesetzt und rekursiv weiter verfolgt. Die zurückgelieferte Farbe des rekursiven Strahls bestimmt zu einem Anteil von (1-A) die aktuelle Pixelfarbe.

Die Richtung des gebrochenen Strahls lautet:

$$r = (dir * n) + normal * (n * (-c1) - c2),$$

mit

dir	- Richtung des Sichtstrahles
n	- n1/n2 (n1 ist der Brechungsfaktor des ersten Materials, n2 der des 2. Materials)
normal	- Oberflächennormale
c1	- scalarProduct(normal, dir)
c2	- sqrt(1.0 - n*n * (1.0 - c1*c1))

Hinweis: für den Wert n1 nimmt man immer den Wert 1 an, falls der Sichtstrahl aus der „Luft“ kommt.

Im Quellcode sind folgende Member wichtig:

Struktur *Material:* *alpha* – der Transparenzfaktor
density – der Brechungsfaktor

Freiwillige Zusatzaufgabe: Weiche Schatten

Das im vorherigen Aufgabenblatt beschriebene Verfahren zur Schattierung ist nur in der Lage Schatten mit festen Grenzen zu generieren. In Wirklichkeit aber haben Lichtquellen eine Gewisse Größe, die dafür Sorgen, dass Schatten an Übergängen weich und verschwommen wirken.

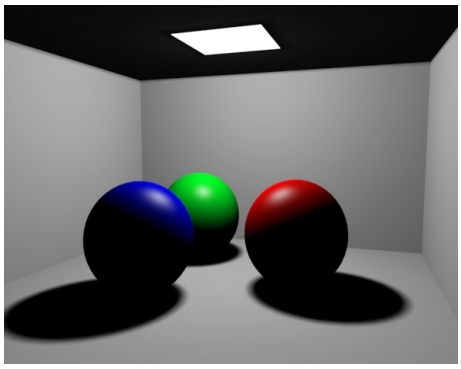

Abbildung 2: Soft Shadows

Um weiche Schatten zu erzeugen, wird nicht nur ein Strahl zur Lichtquelle geschickt, sondern mehrere Strahlen (meist zufällig verteilt), die alle in einer kleinen Umgebung des auszuwertenden Punktes starten. Das zu beleuchtende Pixel erhält dann den Mittelwert aller dieser Farbauswertungen. Welche Größe diese Umgebung hat und wie viel Strahlen zusätzlich ausgewertet werden, bestimmen maßgeblich, wie gut die Schatten am Ende aussehen (wahrscheinlich muss man ein wenig mit den Werten herum experimentieren, bevor man die optimale Lösung findet). Dieses Verfahren erhöht zwar massiv die Darstellungsqualität der Schatten, jedoch kosten die zusätzlichen Strahlauswertungen auch eine Menge Rechenzeit.