Universität Leipzig
Institut für Informatik
Abteilung Bild- und Signalverarbeitung

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

Aufagenblatt 1 – Abgabe in der Woche vom 27.Mai

Raytracing einer triangulierten Szene

Raytracing ist ein hochentwickeltes Renderingverfahren das zur Erzeugung photorealistischer Bilder eingesetzt wird. Im Gegensatz zu dem effizienteren z -Buffer Algorithmus, der meistens in Grafikhardware implementiert ist, ermöglicht Raytracing das Modellieren vieler Spezialeffekte, wie z.B. die Berücksichtigung von spiegelndem und gebrochenem Licht an transparenten Materialübergängen.

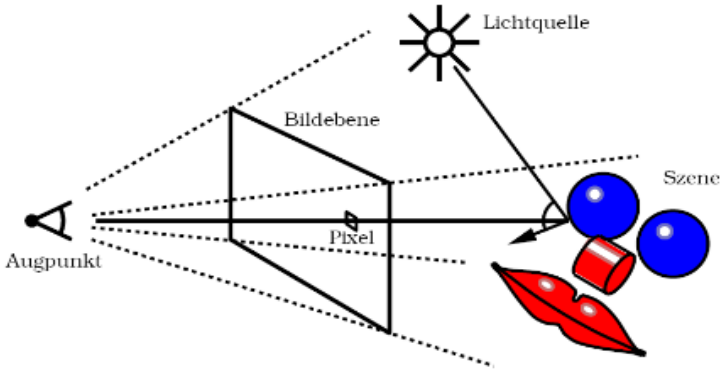

Abbildung 1: Raytracing einer Szene.

Raytracing verfolgt das Licht entgegen seiner Ausbreitungsrichtung vom Betrachter zurück zu den Lichtquellen. Für jeden Bildpunkt (Pixel) wird ein Strahl ausgehend vom Augpunkt mit der Szene geschnitten. Am ersten Schnittpunkt wird mit Hilfe eines Beleuchtungsmodells die Farbe des Pixels ermittelt. Dazu benötigen wir die Flächennormale und Materialeigenschaften am Schnittpunkt, sowie die Position und Sichtbarkeit (vom Schnittpunkt) aller Lichtquellen, siehe Abbildung 1. Im Falle einer partiell durchsichtigen oder spiegelnden Fläche wird der Strahl gebrochen bzw. reflektiert und rekursiv weiter verfolgt.

Aufgabe 1: Schnittpunktberechnung

Alle Primitive (Dreiecke) einer Szene werden zunächst mit dem Strahl

$$ray(t) = \mathbf{e} + t\mathbf{b}$$

geschnitten. Hierbei bezeichnet $\mathbf{e}$ den Augpunkt und $\mathbf{b}$ die Richtung des Strahls. Von allen Schnittpunkten wird derjenige ermittelt, der dem Augpunkt am nächsten ist, d.h. derjenige mit kleinstem (positivem) Parameter t. Wurde kein Schittpunkt gefunden, so erhält der zugehörige Pixel die Hintergrundfarbe.

Um den Schnittpunkt des Strahls mit einem Dreieck, gegeben durch die Punkte $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, zu ermitteln, wird zunächst die Ebene des Dreiecks in Normalenform aufgestellt:

$$(\mathbf{x} - \mathbf{p}_1) \cdot \mathbf{n} = 0$$

wobei $\mathbf{n}$ die Normale des Dreiecks ist. Durch Einsetzen des Strahls für $\mathbf{x}$ erhalten wir den Parameter t des Schnittpunktes mit der Ebene,

$$t = ((\mathbf{p}_1 - \mathbf{e}) \cdot \mathbf{n}) \; / \; (\mathbf{b} \cdot \mathbf{n})$$

sofern Strahl und Ebene nicht parallel sind ($\mathbf{b} \cdot \mathbf{n} \neq 0$).

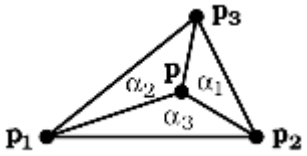

Abbildung 2: Baryzentrische Koordinaten.

Mit dem Parameter t ist auch der Schnittpunkt **p** ermittelt, der jedoch nur dann gültig ist, wenn er innerhalb des Dreiecks liegt und wenn t positiv ist. Ersteres kann mit Hilfe der baryzentrischen Koordinaten α_i festgestellt werden: der Punkt **p** besitzt die Darstellung

$$\mathbf{p} = \alpha_1 \mathbf{p}_1 + \alpha_2 \mathbf{p}_2 + \alpha_3 \mathbf{p}_3, \qquad \text{mit } \sum \alpha_i = 1.$$

Liegt **p** außerhalb des Dreiecks, so ist mindestens eine der baryzentrischen Koordinaten negativ. Die baryzentrischen Koordinaten entsprechen den Flächenanteilen der in Abbildung 2 dargestellten Teildreiecke und können wie folgt berechnet werden:

$$||\alpha_i|| = area(\mathbf{p},\mathbf{p}_{i+1},\mathbf{p}_{i+2}) \; / \; area(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3),$$

$$area\,(\mathbf{a}, \mathbf{b}, \mathbf{c}) = 0.5 * ||(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})||,$$

mit Indizes modulo 3. Die Vorzeichen der baryzentrischen Koordinaten können durch Vergleichen der Orientierungen der Vektoren n und $(b - a) \times (c - a)$ ermittelt werden.

Eine andere Möglichkeit festzustellen, ob der Punkt p innerhalb des Dreiecks liegt, besteht darin, die (nicht orientierten) Flächen der Teildreiecke zu addieren und mit der Gesamtfläche des Dreiecks zu vergleichen. Ist die Summe der einzelnen Teilflächen größer, so liegt der Punkt außerhalb. Bei diesem Test sollte jedoch eine kleine Diskrepanz ε zugelassen sein, um numerische Fehler auszuschließen.

Implementieren Sie eine Ausgabe, indem Sie jedem Dreieck eine spezielle Farbe zuweisen. Diese Farbe soll der Rückgabewert der Funktion raytrace() sein. Da Sie wissen, wieviel Dreiecke in der Szene vorhanden sind („triangles.size()“), können Sie sich eine Zuweisung Dreiecksindex – Farbe überlegen. Falls Sie kein Dreieck getroffen haben, geben Sie die Hintergrundfarbe („backgroundcolor“) zurück.

Aufgabe 2: Beschleunigung

Der Zeitaufwand für das Rendering wird im Wesentlichen dadurch bestimmt, dass jeder Strahl mit allen Dreiecken einer Szene geschnitten werden muss. Da die meisten Strahlen jedoch nur geringe Teile der Szene durchdringen, ist eine räumliche Partitionierung der Szene und eine darauf basierende Vorauswahl der zu verschneidenden Dreiecke sinnvoll.

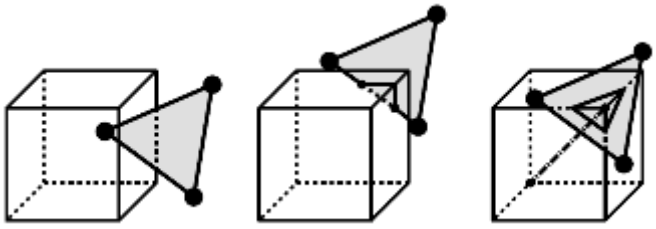

Abbildung 3: Drei Unterschiedliche Fälle des Hineinragens eines Dreiecks in einen Voxel.

Generell wird die Szene in mehrere kleinere Voxel unterteilt und für jedes Voxel existiert eine Liste von Dreiecken, die in diesem Voxel liegen (bzw. Zeiger oder Indizes auf die Dreiecke, da einzelne Dreiecke häufig in mehrere Voxel hineinragen).
Ein Dreieck - gegeben durch die Punkte $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ - ragt in einen Voxel hinein, wenn mindestens einer der folgenden drei Tests positiv ist (siehe auch Abbildung 3):

1. Einer der Eckpunkte p_i liegt innerhalb des Voxels.
2. Eine der drei Kanten $\mathbf{p}_1\mathbf{p}_2$, $\mathbf{p}_2\mathbf{p}_3$ und $\mathbf{p}_3\mathbf{p}_1$ schneidet die Oberfläche des Voxels.
3. Eine der vier Hauptdiagonalen des Voxels schneidet das Dreieck.

Im einfachsten Fall nehmen wir eine uniforme Partitionierung der Szene vor, indem wir zuerst die kleinste *Bounding Box* (umschließenden Quader) der Szene bestimmen. Diese wird in n^3 Voxel (in diesem Fall sind das Quader gleicher Größe) unterteilt. Die Konstante n ist dabei selbst festzulegen. Durch Strahlverfolgung wird ermittelt, welche Voxel ein bestimmter Strahl (Sichtstrahl) durchdringt. Dieser wird dann nur noch mit denjenigen Dreiecken geschnitten, welche sich in den zugehörigen Listen befinden.

Wer Lust und Laune hat, kann auch andere Verfahren zur Raumteilung anwenden. Dazu bieten sich entweder ein kd-Baum (http://de.wikipedia.org/wiki/K-d-Baum) oder ein BSP-Baum (Binary Space Partition http://de.wikipedia.org/wiki/Binary_Space_Partitioning) an. Der Programmieraufwand ist im Vergleich zu dem oben genannten Verfahren höher, jedoch wird im Allgemeinen auch die Effizienz des Raytracers erhöht. Es gibt im Internet natürlich bereits Implementierungen dieser Datenstrukturen (z.B. in Form von Bibliotheken), die Sie gerne benutzen dürfen.