Abgabemodalitäten

Testatstermin: Dienstag, 06.02.2018 (während der Übung)

Das lauffähige Programm, mit allen Unteraufgaben, wird in der Übungsstunde testiert. Zusätzlich bitte vorher den Code im SVN-Gruppenordner hochladen. Während des Testats sind Verständnisfragen und ein Blick über den Code möglich. Alle Fragen sollten von allen Gruppenmitgliedern beantwortet werden können. Wenn eine Aufgabe nicht vollständig funktioniert gibt es Teilpunkte, falls der Versuch in die richtige Richtung geht.

Sie können das Programm gerne selbständig weiter ausbauen. Schöne Ergebnisse werden mit Bonuspunkten belohnt. Weiter werden wir die Bilder gerne auf die GDV1-Moodleseite stellen.

Hinweis: Bitte das Programm auf **release** (optimiert) erstellen. Das kann bei OpenGL einen riesigen Performance-Sprung ausmachen.

4. Programmieraufgabe - 10 Punkte

In dieser Übung soll das Raytracing-Verfahren mit Dreiecksnetzen implementiert werden. Auf der Webseite finden Sie ein Framework für diese Aufgabe, das Sie verwenden *können*. Die Reihenfolge der Bearbeitung der Teilaufgaben ist nicht relevant.

Das Framework stellt ein oder mehrere Dreiecksnetze in einem OpenGL-Fenster dar. Optional können Texturen geladen und auf das Modell aufgebracht werden. Mit 'x' starten Sie den Raycast-Prozess, und mit 'y' löschen Sie das Bild und kehren zur OpenGL-Anzeige zurück.

Die Prozedur `raytrace` erzeugt für jedes Pixel einen Strahl und schneidet diesen gegen das Modell. Für den Schnittest Strahl-Dreiecksnetz gibt es die Prozedur `intersectRayObjects`.

Weiterhin enthält das Framework eine Klasse `Ray`, die auch den Strahl-Schnittest mit einem Dreieck implementiert sowie die Prozedur `rayAABBIntersect`, die einen Strahl gegen eine *axis aligned bounding box* (AABB) schneidet.

a) Führen Sie für jeden Punkt eine Beleuchtungsrechnung nach dem Phong-Modell aus. Legen Sie hierzu eine Lichtquelle und Materialparameter (k_a, k_d, k_s) für ambienten, diffusen und spekulären Anteil fest. Spätestens jetzt dürfen Sie nicht mehr nur irgendeinen Schnittpunkt nehmen, sondern müssen den Schnittpunkt mit dem *kleinsten Strahlparameter t* finden. Passen sie hierzu die `intersectRayObjects`-Methode an. Die Beleuchtungsrechnung erfolgt zudem komponentenweise, d.h. für $\lambda = \{R, G, B\}$ und n Lichtquellen berechnen wir die Intensität im betrachteten Punkt zu

$$I_\lambda = I_{\lambda,a} k_a + \sum_i^n I_{\lambda,i} \left[k_d f_d + k_s f_s \right],$$

wobei f_d und f_s den Reflektionsmodellen des idealen diffusen und spekulären Anteils entsprechen, siehe auch Folie 27 in *Vorlesung 08-light* oder [1], und $I_{\lambda,i}$ ist die Intensität der Lichtquelle. Wenn Sie in Fließkomma-Arithmetik rechnen, müssen die Farbwerte wieder auf 8 Bit normiert und die fertigen Farbwerte zu einem `unsigned int` kombiniert werden.

Um echtes Phong shading zu erhalten, muss zudem die Normale im betrachteten Punkt aus einer linearen Interpolation (mittels der u, v-Koordinaten) der Eckpunktnormalen ermittelt werden.

Prof. Dr. Dieter W. Fellner
M.Sc. Roman Getto
Julian Zobel

Erweitern Sie auch die OpenGL-Ansicht durch Beleuchtung (einfaches OpenGL Shading), wobei Sie die gleichen Parameter wie für das Raycasting verwenden sollten. Vergleichen Sie zwischen dem OpenGL Shading und dem Ergebnis des Raytracing.

2.5 Punkte

b) Führen Sie nun ein *rekursives Raytracing* durch. In jedem Schnittpunkt wird zunächst ein Strahl in Richtung der Lichtquelle(n) geschickt (Schattenstrahlen). Sobald der Strahl ein Objekt bzw. Dreieck trifft, kann die Suche abgebrochen werden. Der Punkt liegt dann bezüglich der Lichtquelle im Schatten (nur ambienter Term). Weiterhin wird ein Reflektionsstrahl ausgesandt, um indirektes Licht einzufangen. Geben Sie hierfür eine maximale, aber variable Rekursionstiefe vor.

Wir erweitern also das Beleuchtungsmodell aus a) zu

$$I_\lambda = I_{\lambda,a} k_a + \sum_i^n S_i I_{\lambda,i} \left[k_d f_d + k_s f_s \right] + k_r I_{\lambda,r},$$

wobei $S_i = 0$, wenn der Punkt im Schatten liegt, und 1 sonst. $I_{\lambda,r}$ ist die Intensität der rekursiv ausgesandten Strahlen und k_r ist der Reflektionsparamter (1 bei einem perfekten Spiegel, nahe 0 bei matten Oberflächen).

Beachten Sie, dass aufgrund numerischer Schwierigkeiten der Schnittberechnung die rekursiven Strahlen als nächsten Schnittpunkt den Ausgangspunkt finden könnten. Um dies zu verhindern, können Sie entweder überprüfen, ob der Strahl das selbe Objekt wiederholt schneidet, oder Sie verschieben den Strahl-Aufpunkt um ein kleines $\epsilon > 0$ entlang der Strahl-Richtung.

2.5 Punkte

c) Erweitern Sie die Strahlverfolgung um Transparenz und gebrochene Strahlen, d.h. wir addieren noch den Term $k_t I_{\lambda,t}$ auf die Beleuchtungsgleichung aus b) auf, wobei der Parameter k_t die Opazität des Objektes angibt. Wie berechnet sich der gebrochene Strahl?

2.5 Punkte

d) Erhöhen Sie die Komplexität der Szene durch Hinzunahme weiterer Dreiecksnetze. Platzieren Sie farbige, spiegelnde und transparente Objekte im Raum. Fertigen Sie ein paar Screenshots Ihrer Ergebnisse an.

2.5 Punkte

Optionale Aufgabe

Anregungen, um das Programm schöner zu gestalten und weitere Bonuspunkte zu bekommen:

a) Integrieren Sie die Landschaft und die Skybox aus Aufgabenblatt 3, um eine malerisch schöne Kulisse zu erzeugen.

b) Bisher wird das Raytracing nur mittels Test gegen eine Boundingbox beschleunigt. Beschleunigen Sie den Strahl-Schnitt-Test weiter mit einer räumlichen Datenstruktur.

c) Fügen Sie Anti-Aliasing-Methoden hinzu. Beispielsweise können mehrere Primärstrahlen für jedes Pixel oder mehrere Sekundärstrahlen an jedem Schnittpunkt verfolgt werden.

Prof. Dr. Dieter W. Fellner
M.Sc. Roman Getto
Julian Zobel

Literatur

[1] Lighting Tutorial http://www.falloutsoftware.com/tutorials/gl/gl8.htm