Abgabemodalitäten

Testatstermin: Dienstag, 28.11.2017 (während der Übung)

Das lauffähige Programm, mit allen Unteraufgaben, wird in der Übungsstunde testiert. Zusätzlich bitte vorher den Code im SVN-Gruppenordner hochladen (Die Zugangsdaten für die SVN-Ordner werden noch vergeben und den einzelnen Gruppen zugeschickt). 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.

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

1. Programmieraufgabe (10 Punkte)

In dieser Übung sollen 3D-Modelle eingelesen und visualisiert werden. Damit die 3D-Modelle korrekt beleuchtet werden, müssen zudem Eckpunkt-Normalen berechnet werden.

Für C++ stellen wir ein Framework zur Verfügung, welches angepasst werden kann. Es steht Ihnen frei, ob Sie das Framework als Vorlage verwenden oder nicht. Alle zu berarbeitenden Stellen sind mit „`// TODO:`“ gekennzeichnet.

a) Zunächst sollen 3D-Modelle aus `.off`-Dateien (siehe [1]) eingelesen werden. Geeignete Modelle finden Sie im Ordner „Modelle“ des bereitgestellten Frameworks oder beispielsweise im Princeton Shape Benchmark [2] oder im Aim@Shape Repository [3].

   Im Framework finden Sie bereits die Funktion `TriangleMesh::loadOFF`, welche den Header korrekt einliest.

   2 Punkte

b) Weiter sollen 3D-Modelle aus `.lsa`-Dateien eingelesen werden, welche ebenfalls im Ordner „Modelle“ des bereitgestellten Frameworks zu finden sind. Das Dateiformat ist nahezu identisch zum .off-Format aufgebaut. Die Unterschiede sind:

   - Die Datei beginnt mit dem Wort „LSA“ anstelle von „OFF“.
   - Nebst der Anzahl der Eckpunkte, Dreiecke und Kanten wird noch der Abstand zwischen Laser und Kamera als Fließkommazahl angegeben. Diese Zahl wird im Framework bereits eingelesen.
   - Anstelle der Eckpunktkoordinaten befinden sich drei Winkelangaben α, β und γ (in Grad), wobei α und β die Winkel aus dem Laserscan-Aufbau der ersten Theorieübung sind. Hinzu kommt der Winkel γ, der sich zwischen dem Laserstrahl und der XZ-Ebene befindet. Damit lässt sich auch der Höhenwert y des Eckpunktes berechnen.

     Hinweis: γ nicht mit dem Spiegelwinkel aus der Theorieaufgabe verwechseln!

   Nach den Winkelangaben folgt, wie in einer OFF-Datei, eine Liste mit Eckpunktindizes, welche zusammen ein Dreieck bilden.

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

Wie in der Theorieübung befindet sich die Kamera im Ursprung, blickt aber nach $-z$ (openGL Konvention)! Der Laser befindet sich in x-Richtung verschoben mit Abstand *baseline* von der Kamera.

2 Punkte

c) Schreiben Sie eine Prozedur, um das Dreiecksnetz im *immediate mode* (siehe auch Kapitel 2 im Red Book [4]) mittels **glBegin**(GL_TRIANGLES) und **glEnd**() zu zeichnen.

1 Punkt

d) Um die Netze zu beleuchten (z.B. *smooth shading* von OpenGL) benötigen wir für jeden Eckpunkt eine Oberflächennormale.

Berechnen Sie hierzu für jedes Dreieck einen Normalenvektor durch das Kreuzprodukt der Kantenvektoren (verwenden Sie die Klasse `Vec3f`).

Anschließend wird die Normale auf die Normalen der Eckpunkte des Dreiecks aufsummiert. Beachten Sie, dass dies mit einem Aufwand von $\mathcal{O}(\#\text{Dreiecke})$ direkt beim Einlesen der Netze erledigt werden kann!

Für die Beleuchtung müssen zum Schluss die Eckpunkt-Normalen normiert werden.

2 Punkte

e) Die Eckpunkt-Normalen aus b) sind nach Dreiecksfläche gewichtet (warum?). Alternativ sollen die Normalen in dieser Aufgabe mit dem Winkel des Dreiecks am betrachteten Eckpunkt gewichtet werden (`acos` des Skalarprodukts der normierten Kantenvektoren).

2 Punkte

f) Erstellen Sie eine kleine Szene mit mehreren Dreiecksnetzen, die Sie nach belieben platzieren. Versuchen Sie die Hardware mit dem *immediate mode* auszureizen. Wie viele Dreiecke können Sie noch flüssig darstellen?

1 Punkt

Literatur

[1] `.off` Dateiformat `http://shape.cs.princeton.edu/benchmark/documentation/off_format.html`

[2] Princeton Shape Benchmark `http://shape.cs.princeton.edu/benchmark/`

[3] Aim@Shape Repository `http://visionair.ge.imati.cnr.it/ontologies/shapes/`

[4] Red Book: OpenGL Programming Guide. `http://www.glprogramming.com/red/`