Graphische Datenverarbeitung I

Praxisübung 1

Prof. Dieter Fellner
Daniel Ströter, M.Sc.
David Spitzenberg, M.Sc.

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 2021/2022
Ausgabe: 2. November 2021, Abgabe: 23. November 2021, 09:30 Uhr

Abgabemodalitäten

Das *lauffähige* Programm mit allen Unteraufgaben wird im Rahmen eines Zoom-Meetings testiert. Vorab muss der gesamte Quelltext fristgerecht **als eine ZIP-Datei** auf Moodle abgegeben werden. Es genügt eine Abgabe je Gruppe.

Während des Testats werden allgemeine Verständnisfragen und Fragen zum Quelltext gestellt. Alle Fragen sollten von allen Gruppenmitgliedern beantwortet werden können. Für nicht vollständig gelöste Aufgaben werden Teilpunkte vergeben, falls die Abgabe in die richtige Richtung geht.

Sie können Ihre Abgabe über die geforderten Aufgaben hinaus gerne freiwillig und selbstständig erweitern. Schöne Ergebnisse werden mit Bonuspunkten belohnt.

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

Aufgabe 1: Einlesen von OFF-Dateien (2 Punkte)

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 und entsprechend der Aufgabenstellung erweitert werden muss.

Aufgabe 2: Einlesen von LSA-Dateien (2 Punkte)

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.

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.

Aufgabe 3: Rendering eines TriangleMesh (1 Punkt)

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.

Aufgabe 4: Berechnung der Vertexnormalen (4 Punkte)

4a) Gewichtung nach Dreiecksfläche (2 Punkte)

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.

Die Eckpunkt-Normalen sind durch diese Methode nach Dreiecksfläche gewichtet. Warum?

4b) Gewichtung nach Winkel (2 Punkte)

Implementieren Sie eine zweite Variante bei der die Normalen mit dem Winkel des Dreiecks am betrachteten Eckpunkt gewichtet werden (`acos` des Skalarprodukts der normierten Kantenvektoren).

Aufgabe 5: Mehrere Dreiecksnetze (1 Punkt)

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?

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/`