WP Einführung in die Computergrafik

WS 2015/2016, Hochschule für Angewandte Wissenschaften (HAW), Hamburg
Prof. Dr. Philipp Jenke

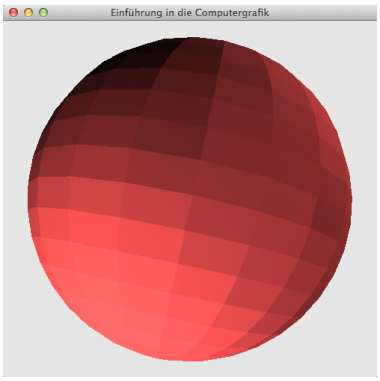
Für alle praktischen Aufgaben gelten folgende Regeln:

- Der Code muss die Code-Konventionen einhalten (finden Sie auf der EMIL-Seite).
- Die Funktionalität muss ausreichend kommentiert sein.
- Testbare Funktionalität muss getestet sein (automatisierte Tests, z.B. Unit-Test)

Aufgabenblatt 2 - Praxis: Dreiecksnetz

In diesem Aufgabenblatt entwickeln Sie eine Halbkanten-Datenstruktur zur Repräsentierung und Darstellung von Dreiecksnetzen. Es müssen Tests vorhanden sein, die sicherstellen, dass die Halbkanten-Datenstruktur korrekt aufgebaut wurde.

Lernziele: Mit Dreiecksnetzen arbeiten, Algorithmen in der Halbkanten-Datenstruktur umsetzen, OpenGL Display-Listen verwenden.



Darstellung eines Oberflächen-Dreiecksnetzes.

Aufgabe 1: Datenstruktur

Vorgegeben ist das Interface `ITriangleMesh`. Schreiben Sie eine Klasse `HalfEdgeTriangleMesh`, die das Interface implementiert. Setzen Sie die Datenstruktur mit dem Halbkanten-Ansatz um. Dazu finden Sie im Package `datastructures` bereits die notwendigen Klassen `HalfEdge`, `Vertex` und `TriangleFacet`. In der Klasse `HalfEdgeTriangleMesh` müssen Halbkanten, Facetten und Vertices jeweils als Listen abgelegt werden.
Die Halbkantenstruktur wird durch die im Interface vorgegebenen Methoden aufgebaut. `addVertex()` fügt einen neuen Vertex hinzu. `addTriangle()` fügt ein Dreieck zwischen drei Vertices hinzu. Dazu sind die Indices der drei Vertices als Parameter übergeben. Sie müssen damit folgendes machen: Erzeugen einer neuen Facette und dreier Halbkanten. Setzen aller Eigenschaften der neuen Objekte (außer der Referenzen auf gegenüberliegende Halbkanten, das geschieht im folgenden Schritt).
Nachdem alle Vertices und alle Facetten eingefügt wurden, müssen noch zwei weitere Informationen gesetzt werden:

- Jedem Vertex muss eine ausgehende Halbkanten zugewiesen werden.
- Zu jeder Halbkante muss die gegenüberliegende Halbkante gesetzt werden. Finden Sie dazu für jede Halbkante vom Vertex `v1` zum Vertex `v2` die passende gegenüberliegende Halbkante von `v2` nach `v1`. Den Zielvertex einer Halbkante erhalten Sie durch `halbkanten.getNext().getStartVertex()`.

Setzen Sie dies in einer Hilfsmethode um. Diese Methode müssen Sie dann manuell aufrufen, nachdem Vertices und Facetten eingefügt wurden. Im Interface sind auch Methoden zur Verarbeitung von Texturkoordinaten vorgegeben. Diese können Sie zunächst in Ihrer Implementierung mit leeren Dummy-Methoden umsetzen.

Aufgabe 2: Dreiecksnormalen

Bei der Beleuchtungsrechnung spielt die Oberflächennormale eine zentrale Rolle. Sie müssen Funktionalität schreiben, die für jedes Dreieck des Netzes die Normale berechnet. Diese muss senkrecht auf der Dreiecksfläche stehen und die Länge 1 haben. Implementieren Sie dazu die Methode `computeTriangleNormals()`, die alle Dreiecksnormalen berechnet und abspeichert.
Tip: Kreuzprodukt.

Aufgabe 3: Darstellung

Um Dreiecksnetze mit OpenGL darstellen zu können, müssen Sie in einem geeigneten Knoten in den Szenengraph eingehängt werden. Erweitern Sie den Szenengraph um einen Knoten, der ein `HalfEdgeTriangleMesh` beinhaltet und dieses in OpenGL darstellt.

Aufgabe 4: Displaylisten

Displaylisten sind eine Möglichkeit, komplexere Objekte effizienter zu zeichnen - anstelle von `glBegin(...) ... glEnd()`. Verändern Sie die Klasse zur Darstellung von Dreiecksnetzen so, dass jedes Dreiecksnetz mit Hilfe einer Displayliste gezeichnet wird. Insbesondere soll die Displayliste nur einmal erzeugt werden und zur Laufzeit bei jedem Zeichnen lediglich aufgerufen werden. Wie Displaylisten arbeiten finden Sie in vielen Quellen, unter anderem hier: [1]

Hinweis: In dem Basisframework finden Sie eine Klasse `ObjIO` mit deren Hilfe Sie ein Dreicksnetz aus einer Text-Datei einlesen können. Einige Beispieldateien finden Sie ebenfalls bei EMIL. Damit können Sie die Funktionalität einfacher testen, als eigene Dreiecksnetze zu entwerfen.

[1] Displaylisten mit JOGL: *http://www.java-tips.org/other-api-tips/jogl/how-to-create-a-display-list-in-jogl.html*, abgerufen am 24.02.2015