3D-Computergrafik I

Sommersemester 2014 – Aufgabenblatt 4

Hasso-Plattner-Institut
Fachgebiet Computergrafische Systeme

Aufgabe 18: Normalenberechnung in Polygonnetzen (8 Punkte)

Ziel dieser Aufgabe ist es, ein polygonales Netz eines vorliegenden 3D-Modells (repräsentiert durch Dreiecke) so darzustellen, dass visuell zwischen scharfen und weichen Kanten unterschieden werden kann (siehe Abbildung 1). Die Unterscheidung zwischen scharfen und weichen Kanten wird anhand eines konfigurierbaren Schwellwert-Winkels zwischen angrenzenden Polygonen getroffen. Ist der Winkel, den die Flächennormalen zweier benachbarter Polygone bilden, kleiner, so soll die gemeinsame Kante als weiche Kante dargestellt werden, im anderen Fall als scharfe Kante.

Weiche Kanten werden mit OpenGL dadurch realisiert, dass ein Eckpunkt einer gemeinsamen Kante von zwei Polygonen mit derselben Normale attributiert wird (*vertex normals*). Visuell scharfe Kanten werden erzeugt, indem der Eckpunkt der beiden Polygone mit zwei unterschiedlichen Normalen attributiert wird (*face normals*). Der unterschiedliche visuelle Effekt resultiert aus der Beleuchtungsberechnung, die u. a. auf der Normaleninformation der Eckpunkte basiert.

Der Programmrahmen stellt mit den Klassen `ObjIO` und `PolygonalDrawable` Grundfunktionalität bereit, die ein dreieck-basiertes Modell in Form einer *.obj* Datei einliest und in einer indizierten Knotenliste sowie einer Halbkanten-Datenstruktur `HalfEdgeStructure` zur Verfügung stellt.

(1) Implementieren Sie die Methode `calculatePerFaceNormals` der Klasse `HalfEdgeStructure`. Hier soll über alle Dreiecke des Modells iteriert werden, die Flächennormale berechnet und diese jeweils in `m_faces[i].normal` gespeichert werden.

(2) Implementieren Sie die Methode `calculatePerVertexNormals` der selben Klasse. In dieser Funktion sollen mithilfe der HalfEdge-Datenstruktur die Normalen pro Halbkante durch die angrenzenden Dreiecke gemittelt und diese jeweils in `m_halfEdges[i].normal` gespeichert werden. Dabei ist ein Schwellwert von `threshold` zu berücksichtigen.

(3) Rendern Sie die die Flächennormalen und die Vertex-Normalen (jeweils unterschiedlich gefärbt). Ergänzen Sie dazu die Methoden `drawFaceNormals` und `drawVertexNormals` der Klasse `Exercise18`. Die Flächennormalen sollen als Linien (GL_LINES) jeweils ausgehend vom Mittelpunkt der Fläche, die Vertex-Normalen jeweils ausgehend vom entsprechenden Vertex gezeichnet werden. Wählen Sie eine angemessen Skalierung der Normalen.

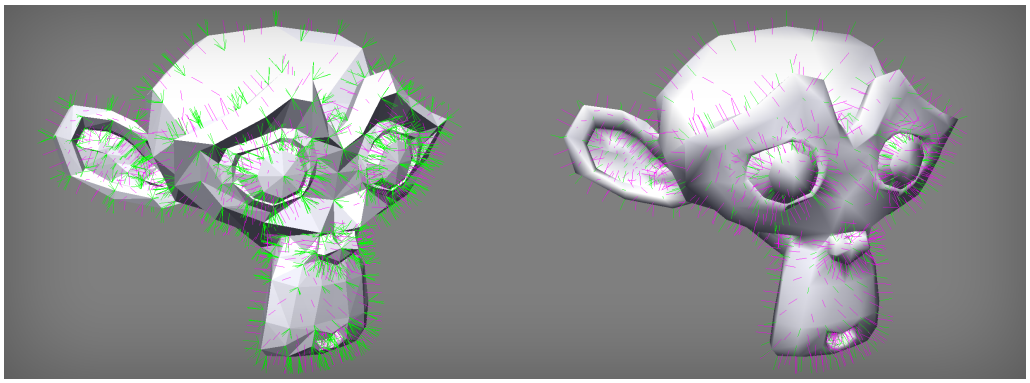

Abbildung 1: Beispiel: Visualisierung der zuvor berechneten Flächen- (rosa) und Vertex-Normalen (grün). Links mit einem Schwellwert von 0 deg, rechts mit 180 deg.

Aufgabe 19: Tessellation von beliebigen Polygonen (7 Punkte)

Ziel dieser Aufgabe ist es, mit Hilfe des `GLUtesselators` eine Tessellation von beliebigen Polygonen umzusetzen (siehe Abbildung 2). Im gegebenen Programmrahmen können Punkte durch einen Linksklick auf das Canvas hinzugefügt werden. Diese Punkte werden als Eckpunkte der Kontur eines Polygons betrachtet. Mit einem Rechtsklick wird die aktuelle Kontur abgeschlossen.

Jede abgeschlossene Kontur wird in der Struktur `Contour` abgelegt, welche aus einer Menge von Punkten besteht. Alle aktuell abgeschlossenen Konturen befinden sich in der Liste `ContourList`.

Abbildung 2: Beispiel einer Tessellation.

Zeichnen Sie alle Konturen aus dieser Liste, indem Sie in der Klasse `Exercise19` die Tesselation und das Rendering umsetzen. Implementieren Sie dazu die Funktion `tesselatePolygons`, welche die Konturen an den `GLUtesselator` übergibt. Hinweis: Der Tesselator darf nur einmal aufgerufen werden!

Der Tesselator gibt nun die verarbeiteten Konturen durch Callbacks zurück. Zeichnen Sie diese Konturen, indem Sie die folgenden Callbacks implementieren:

beginCallback wird aufgerufen, wenn ein neues Primitiv begonnen wird.

vertexCallback wird für jeden vom Tesselator bearbeiteten Punkt aufgerufen.

endCallback wird am Ende der aktuellen Kontur aufgerufen. Zeichnen Sie hier die aktuelle Kontur einerseits gefüllt und andererseits als Linienzug.

Weitere Informationen im Moodle und unter `http://www.glprogramming.com/red/chapter11.html`.

Allgemeine Hinweise:

- Die Aufgaben sollen maximal zu zweit bearbeitet werden; Ausnahmen müssen mit den Übungsleitern abgesprochen werden.
- Bitte reichen Sie Ihre Lösungen bis Dienstag, den **24.06.2014**, um **13.15** Uhr ein.
- Tragen Sie Ihre Matrikelnummern in die Quellcode-Dateien ein. Beachten Sie, dass nur die vollständigen Quelltexte und Projektdateien geschickt werden sollen. Senden Sie uns keinesfalls ausführbare Dateien oder bereits kompilierte Binär- oder Temporärdateien (*.obj, *.pdb, *.ilk, *.ncb etc.) zu! Testen Sie vor dem Verschicken, ob die Projekte aus den Zip-Dateien fehlerfrei kompiliert und ausgeführt werden können.
- Zippen Sie Ihre Lösungen in **eine** Zip-Datei. Geben Sie der Zip-Datei einen Namen nach folgendem Schema:

  cg1_blatt5_matrikelnummer1_matrikelnummer2.zip.
- Die Zip-Datei laden Sie dann bei moodle hoch.