CG1 Übungsblatt 3

10. Dezember 2007

Abgabe: Donnerstag, 20. Dezember 2007

AUFGABE 1: **Binary Space Partitioning**

In dieser Übung soll das Sichtbarkeitsproblem mittels eines BSP Baumes gelöst werden, für die gesamte Aufgabe muss also der OpenGL Z-Buffer ausgeschaltet sein (`glDisable(GL_DEPTH_TEST)`). Unter http://www.cg.tu-berlin.de/cg1_07.html steht ein C Programmskelett zur Verfügung, das zu ergänzen ist.

1. Modelle in Form von Dreiecksnetzen sollen geladen und angezeigt werden können. Als Modellformat soll OFF verwendet werden. Dieses ist wie folgt aufgebaut:

Zeile	Inhalt	Beschreibung
1	OFF	Datei-Kennung
2	V F E	V: Anzahl der Dreiecksknoten, F: Anzahl der Polygone, E: Anzahl der Kanten
3	x y z	Koordinate des Knotens mit Index 0
4	x y z	Koordinate des Knotens mit Index 1
...	x y z	Koordinate des Knotens mit Index ...
V+2	x y z	Koordinate des Knotens mit Index V-1
V+3	n i_1 i_2 … i_n	Polygon mit n Knoten in den Positionen der durch i_k referenzierten Einträge
V+4	n i_1 i_2 … i_n	Polygon mit n Knoten in den Positionen der durch i_k referenzierten Einträge
...	n i_1 i_2 … i_n	Polygon mit n Knoten in den Positionen der durch i_k referenzierten Einträge
V+F+2	n i_1 i_2 … i_n	Polygon mit n Knoten in den Positionen der durch i_k referenzierten Einträge

   Implementieren Sie die Funktion `readOff(...)`. Gehen sie davon aus, dass n konstant drei ist, also alle Flächen Dreiecke sind. Legen sie die Modellinformation in der Datenstruktur TriangleList ab (siehe bsp.h). Generieren Sie aus den Knoten Normalen für jede Facette, implementieren sie dafür die Funktion `computeNormals(...)`. (1 Punkt)
2. Implementieren sie die Funktion `computePartitioningPlane(...)`, welche eine „gute“ Partitionierungsebene aus einer TriangleList berechnet. Sie sollen dabei aus der Menge der Ebenen wählen, die durch die Dreiecke in der TriangeList aufgespannt werden (sog. autopartition). Beachten sie, dass bei der Wahl typischerweise ein Kompromiss zwischen Qualität der Partitionierung und Geschwindigkeit eingegangen werden muss, eine optimale Lösung ist schwierig zu finden. Sie müssen jedoch in der Abnahme der Übung die Wahl ihres Algorithmus begründen und dessen Eigenschaften erklären können. (1 Punkt)
3. Erzeugen sie aus einer TriangleList rekursiv einen BSP Baum, implementieren sie dafür die Funktion `constructBsp(...)`. Diese Funktion soll die übergebene TriangleList mittels der Funktion `computePartitioningPlane(...)` in zwei Teilräume partitionieren, aus welchen dann rekursiv jeweils wieder ein Unterbaum erzeugt wird. (1 Punkt)
4. Implementieren sie die Funktion `draw(...)`, welche eine TriangleList mittels OpenGL zeichnet. Zeichnen sie die Dreiecke als Gittermodel, wobei verdeckte Kanten anders gezeichnet werden (z.B. andere Farbe) als die sichtbaren Kanten. (1 Punkt)
5. Implementieren sie die Funktion `traverse(...)`, welche einen BSP Baum traversiert und für die in den Knoten enthaltenen TriangleLists die Zeichenfunktion `draw(...)` aufruft. (1 Punkt)

Technische Universität Berlin
Computer Graphics

Prof. Dr. Marc Alexa
Mathias Eitz

AUFGABE 2: **Theoriefragen**

1. Ein Dreieck wird mittels Gouraud shading gezeichnet. Welches Bild ist zu erwarten, wenn eine Normale in Richtung Lichtquelle zeigt und die anderen Normalen davon abweichen. (1 Punkt)
2. Auf welche der Beleuchtungskomponenten {ambient, diffus, spekulär} hat die Position des Betrachters bei einer ansonsten statischen Szene Einfluss? Begründen Sie Ihre Antwort. (1 Punkt)
3. Bei der Transformation grafischer Primitive ist es notwendig auch ihre Normalen zu transformieren, um die Beleuchtungsrechnung durchzuführen. Im Allgemeinen können Knoten und Normalen jedoch nicht gleich behandelt werden. Sei $\boldsymbol{n}$ die Normale der von $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ aufgespannten Ebene und $\boldsymbol{M}$ die Transformation, welche auf die Knoten angewandt wird:
    a) Geben sie ein Beispiel einer Transformation an, bei der $\boldsymbol{Mn}$ nicht die Normale der Ebene ist, die durch $\boldsymbol{Mv}_1$, $\boldsymbol{Mv}_2$ gegeben ist. (0,5 Punkte)
    b) Um Normalen korrekt zu transformieren, wird die Transponierte der Inversen von $\boldsymbol{M}$ gebildet. Zeigen sie, warum $\boldsymbol{n'} = (\boldsymbol{M}^{-1})^T\boldsymbol{n}$ die transformierte Normale ist. (1 Punkt)
    c) Unter welchen Umständen kann von der Berechnung von $(\boldsymbol{M}^{-1})^T$ abgesehen werden? (0,5 Punkte)
4. Sei n die Anzahl der Dreiecke in einem Dreicksnetz. Ein aus diesem Dreiecksnetz erzeugter, balancierter BSP Baum hat typischerweise eine Tiefe von O(log n). Bei welcher Art von Dreiecksnetzen ist jedoch eine Tiefe von O(n) des BSP Baumes unvermeidbar? Was hat dies für Auswirkungen auf die Rendering-Performance? (1 Punkt)
5. Aus welchem Grund werden beim Aufbau eines BSP Baumes aus einem Dreicksnetz oftmals die enthaltenen Dreiecke selbst zum Spannen der Partitionierungsebene verwendet? Nennen sie mindestens eine weitere Möglichkeit, Partitionierungsebenen zu bestimmen. (1 Punkt)