Praktikum WP Computergrafik
WS 2016/2017, Hochschule für Angewandte Wissenschaften (HAW), Hamburg
Prof. Dr. Philipp Jenke

Aufgabenblatt 4: Marching-Cubes Algorithmus

In diesem Aufgabenblatt implementieren Sie den Marching Cubes (MC) Algorithmus zur Tesselierung von impliziten Funktionen in 3D.

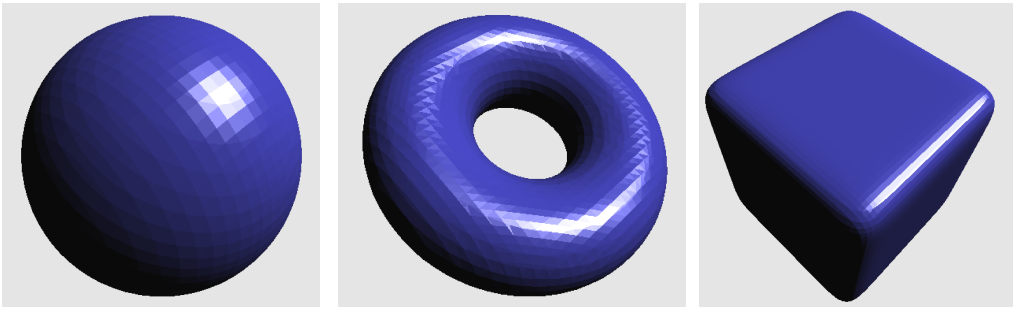

Abbildung 1: Darstellung von Oberflächen-Dreiecksnetzes (Kugel, Torus, Superquadrik).

Aufgabe 4.1: Tesselierung eines Würfels

Schwerpunkte: Implementierung der MC-Algorithmus für eine Zelle
Aufgabe: Beginnen Sie die Implementierung mit der Tesselierung eines einzelnen Würfels. Schreiben Sie dazu eine Methode

```
private void createTriangles( List<Vector3> points, List<Double> values );
```

die für die acht Eckpunkte des Würfels und die dazugehörigen Funktionswerte die notwendigen Dreiecke erzeugt. Die Dreiecke werden in einem Dreiecksnetz-Objekt abgelegt.

Berechnen Sie dazu für jeden Eckpunkt den zugehörigen Fall:

$b_i = (v_i > \tau)? 1 : 0$

und

caseIndex = b1*1 + b2*2 + b3*4 + b4*8 + b5*16 + b6*32 + b7*64 + b8*128.

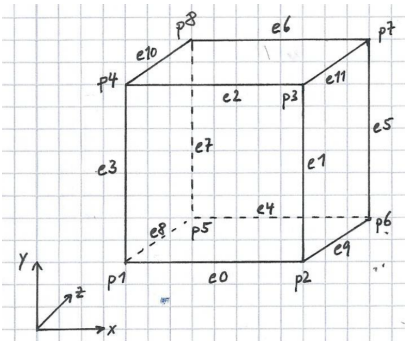

Abbildung 2: Würfel mit Bezeichnungen der Eckpunkte (p1-p8) und der Kanten (e0-e11).

Es gibt demnach 256 verschiedene Fälle. In jedem der Fälle werden zwischen 0 und 5 Dreiecke erzeugt. Die Eckpunkte jedes Dreiecks liegen auf je einer der 12 Kanten des Würfels.

Um festzustellen, wie viele und welche Dreiecke benötigt werden, gibt es eine Lookup-Tabelle in der Datei *casesLookupTable.txt*, die einfach in Ihre Java-Klasse integriert werden kann. In der Tabelle ist jeder Fall also mit 5 (Anzahl Dreiecke) * 3 (Knoten pro Dreieck) = 15 Indizes dargestellt. Der Index -1 gibt an, dass kein Dreieck benötigt wird, alle anderen Indizes geben die zugehörige Kante an. Die Position eines Dreieck-Eckpunktes auf einer Kante berechnen Sie mit der Formel aus den Vorlesungsfolien:

$p = (1 - t) * p_{i,j,k} + t * p_{i+1,j,k}$ (für Kanten in x-Richtung)

$t = (\tau - v_{i,j,k}) / (v_{i+1,j,k} - v_{i,j,k})$

$v_{i,j,k}$ ist dabei der Funktionswert an der Stelle $p_{i,j,k}$ und τ ist der Isowert. Zum Testen können Sie zunächst auch die Mittelpunkte der Kanten verwenden (t = 0.5).

Beispiel

- v2 ist größer als 0, alle anderen vi sind kleiner als 0 → caseIndex = 2
- In der Tabelle sind also die Einträge mit den Indizes caseIndex*15 bis (caseIndex+1)*15-1 relevant, also 30 bis 44: 0, 1, 9, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1.
- Nur die ersten drei Einträge sind ungleich -1, daher muss genau ein Dreieck erzeugt werden.
- Die Eckpunkte des Dreiecks liegen auf den Kanten 0, 1 und 9.
- Bei der Kante e_0 bedeutet das: Dreiecks-Eckpunkt = $0.5*(p_1+p_2)$ (einfach Approximation des Eckpunktes, keine Interpolation)

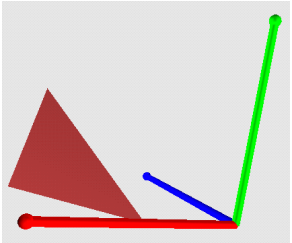

Abbildung 2: Erzeugtes Dreieck für das Beispiel.

Aufgabe 4.2: Tesselierung eines Volumens

Schwerpunkte: Erweiterung des MC-Algorithmus auf gesamtes Gitter
Aufgabe: Erweitern Sie jetzt im zweiten Schritte das Verfahren auf ganze Volumen. Hier gehen wir von einem konstanten Volumen im Bereich $[-2;2]^3$ aus. Um das Volumen in Würfel zu unterteilen, verwenden Sie eine angemessene Auflösung, z.B. 25x25x25. Nun müssen Sie die in der ersten Aufgabe entwickelte Logik auf jeden Würfel einzeln anwenden. Die Werte an den Eckpunkten jedes Würfels werten Sie für eine implizite Funktion aus. Testen Sie Ihre Implementierung mit mindestens drei verschiedenen impliziten Funktionen.

Anhang: Implizite Funktionen

Hier finden Sie einige implizite Funktionen zum Testen. Am besten kapseln Sie die Auswertung einer impliziten Funktion in eine extra Datenstruktur/Klasse:

Kugel:

$f(x,y,z) = x^2 + y^2 + z^2 - r^2$

Radius r, z.B. r = 1

Torus:

$f(x,y,z) = (x^2 + y^2 + z^2 + R^2 - r^2)^2 - 4R^2(x^2 + y^2)$

äußerer Radius R, innerer Radius r, z.B. R = 1 und r = 0.5

Weitere implizite Funktionen (insbesondere die Superquadriken in Kapitel 3.2) finden Sie unter [1]. Die Arbeit liegt als PDF auf der EMIL-Seite.

Hinweis: Generell sind Ihnen die Details der Implementierung freigestellt (auch die Modularisierung in Methoden). Die hier beschriebenen Vorgaben dienen dem Zweck einer möglichst guten Testbarkeit. Fall das Gesamtergebnis nicht korrekt aussieht, müssen Sie in der Lage sein, einzelne kleine Module zu testen.

Quellen

[1] Eldar Sultanow (Hasso-Plattner-Institut an der Universität Potsdam): Implizite Flächen Darstellung (z.B. als Linien)