Karlsruher Institut für Technologie

Angewandte Geometrie (CAGD)
Prof. Dr. Hartmut Prautzsch
Dipl.-Ing. Yijun Xu

MARS-Basispraktikum **WS 2022**

Versuch 6

MARCHING CUBES

Inhaltsverzeichnis

1 Isoflächen

In diesem Versuch soll die Visualisierung von Isoflächen implementiert werden. Isoflächen sind die zweidimensionale Verallgemeinerung von Isolinien. Diese werden zum Beispiel in der Meteorologie als Isobaren oder in der Kartografie als Höhenlinien verwendet.

Eine Isobare ist die Menge der Punkte im zweidimensionalen, an welchen der gleiche Druck herrscht. Analog ist eine Isofläche die Menge der Punkte im dreidimensionalen, an welchen ein Merkmal einen konstanten Wert I annimmt. Dieser Wert wird als *Isowert* bezeichnet.

In der Medizin wird beispielsweise mit einem CT-Scan die Röntgenopazität gemessen. Die Röntgenopazität vieler Stoffe im Körper wird durch die *Hounsfieldskala*[1] klassifiziert. Zeichnet man zum Beispiel eine Isofläche mit dem Isowert $I = 350HU$, erhält man die Oberfläche der Knochen. Mit einem anderen Wert wie $I = 40HU$ kann die graue Substanz im Gehirn visualisiert werden.

[1] `https://en.wikipedia.org/wiki/Hounsfield_scale`

Der bekannteste Algorithmus, der zum Zeichnen von Isoflächen verwendet werden kann, ist der Marching-Cubes-Algorithmus. Dafür werden Messwerte auf einem dreidimensionalen Gitter benötigt. Der Einfachheit halber verwenden wir in diesem Versuch anstelle von Messdaten mathematische Funktionen, die wir an Gitterpunkten abtasten.

2 Der Marching-Cubes-Algorithmus

Der Marching-Cubes-Algorithmus soll eine Isofläche in einem Würfel mit der Seitenlänge $2L$, der um den Ursprung zentriert ist, darstellen. Die Isofläche ist durch die Funktion F und den Isowert I mit folgender Gleichung definiert.

$$F(x, y, z) = I \iff F(x, y, z) - I = 0$$

Dazu wird dieser in n^3 kleine Würfel mit Seitenlänge $\frac{2L}{n}$ unterteilt und über alle kleine Würfel iteriert. Im Folgenden wird beschrieben, wie die Fläche innerhalb eines kleinen Würfels durch Dreiecke approximiert wird.

2.1 Triangulierung eines Würfels

Zuerst wird die gegebene Funktion an den acht Eckpunkten des Würfels abgetastet. Wenn die Funktion an einem Ende einer Kante größer als der Isowert ist, und am anderen Ende kleiner als der Isowert, dann muss die Isofläche diese Kante schneiden. Das Ziel ist diese Schnittpunkte an allen relevanten Kanten zu finden und anschließend zu bis zu fünf Dreiecken zusammenzusetzen.

In Abbildung 1 sind zwei Würfel dargestellt, deren Ecken mit + beschriftet sind, wenn $F(x, y, z) - I$ an diesem Eckpunkt positiv ist. Ein $-$ bedeutet dementsprechend, dass die Funktion einen kleineren Wert als I an dem Eckpunkt annimmt. Es gibt $2^8 = 256$ mögliche Kombinationen $+/-$ den Ecken zuzuordnen. Diese sind in einer Look-up Tabelle kodiert, auf die mit einem 8-bit Flag zugegriffen wird.

Das Flag wird folgendermaßen konstruiert: Für jeden Eckpunkt mit dem Index i, ist das 2^i-wertige Bit 1, wenn der Punkt mit $-$ markiert ist, und ansonsten 0. Ist dieses Flag 0 oder 255, ist dieser Würfel leer und kann übersprungen werden.

Als Nächstes wird an jeder Kante der Punkt approximiert, an dem die Isofläche die Kante schneidet. Dazu geht man davon aus, dass die Funktion auf der Kante linear ist.

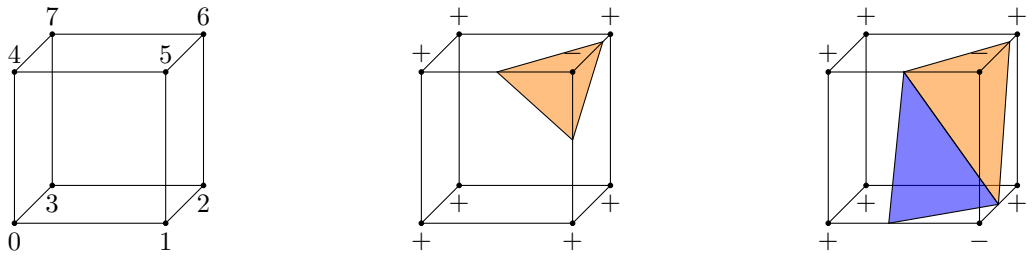
Sind p_1 und p_2 benachbarte Eckpunkte und $v_1 = F(p_1) - I$ und $v_2 = F(p_2) - I$, dann nimmt F den Isowert bei

$$q = \frac{v_1 p_2 - v_2 p_1}{v_1 - v_2}$$

näherungsweise an. Der Punkt q ist ein Eckpunkt eines Dreiecks.

Um die interpolierten Punkte den Dreiecken richtig zuordnen zu können, geht man folgendermaßen vor. Zunächst wird das 8-bit Flag von oben in ein 12-bit Flag umgewandelt. Dabei ist das 2^k-wertige Bit genau dann 1, wenn die k-te Kante geschnitten wird. Wird die k-te Kante geschnitten, wird der interpolierte Punkt auf der Kante berechnet und an die k-te Stelle in ein Array A geschrieben.

Anschließend wird mit dem 8-bit Flag eine Liste aus Indizes geladen, die die Punkte auf drei Kanten zu einem Dreieck gruppiert. Für den Fall aus Abbildung 1c ist die Liste $(0, 4, 5, 1, 0, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1)$. Das bedeutet, dass die Punkte $A[0], A[4], A[5]$ sowie die Punkte $A[1], A[0], A[5]$ jeweils ein Dreieck bilden. Der Wert -1 bedeutet, dass für diesen Würfel keine weiteren Dreicke entstehen.



(a) Indizierung der Ecken (b) Fall 00100000 (c) Fall 00100010

Abbildung 1: Zwei beispielhafte Triangulierungen

2.2 Mehrdeutige Fälle

Es gibt Konfigurationen für die die Triangulierung nicht eindeutig ist. Überlegt euch, unter welchen Bedingungen es mehrere Möglichkeiten gibt, einen Würfel zu triangulieren! Es gibt Methoden, die dieses Problem lösen – wir befassen uns damit aber nicht.

2.3 Look-up Tabellen

Die meisten beschriebenen Schritte lassen sich am einfachsten mit Look-up Tabellen lösen. Das gilt auch für einfache Zusammenhänge, wie das bestimmen der Eckpunkte einer Kante oder die Berechnung aller Eckpunkte aus nur einem. Wenn die Tabellen konsistent verwendet werden, macht man am wenigsten Fehler.

3 Implementierung

In diesem Versuch soll der Marching-Cubes Algorithmus wie oben beschrieben werden. Dabei wird die bisher verwendete Bibliothek nicht mehr benötigt. Stattdessen soll ein eigenständiges Programm entwickelt werden. Die Look-up Tabellen müssen aber nicht selbst berechnet werden.

3.1 Programmiersprache

Ihr könnt die Programmiersprache frei wählen, allerdings sollten Bitoperationen unterstützt werden um die Berechnung der Flags zu vereinfachen. Es bietet sich an eine Bibliothek zu verwenden, die mit dreidimensionalen Vektoren rechnen kann. Allerdings werden nur einfache Addition, Subtraktion und Multiplikation mit Skalaren benötigt – ihr könnt also auch leicht selbst eine Klasse implementieren.

Die Look-up Tabellen sind in Python geschrieben, können allerdings schnell in Syntax aus C oder ähnlichen Sprachen umgewandelt werden.

3.2 Visualisierung des Dreiecksnetzes

Ihr könnt die Dreiecke direkt durch euer Programm zeichnen lassen, wenn ihr beispielsweise OpenGL verwendet. Alternativ kann das Dreiecksnetz in einer Datei gespeichert werden, und durch ein anderes Programm angezeigt werden. Hier bietet sich das OFF-Format[2] an, da es sehr einfach ist und direkt mit geomview[3] angezeigt werden kann.

[2] `https://en.wikipedia.org/wiki/OFF_(file_format)`
[3] `http://www.geomview.org`

3.3 Parameter

Euer Programm soll vier Parameter akzeptieren. Die Werte L, n und I aus der Beschreibung des Algorithmus, sowie einen Parameter, der bestimmt, welche Funktion visualisiert werden soll.

3.4 Funktionen

Es sollen mindestens die folgenden vier Funktionen unterstützt werden.

Kugel

$$F(x, y, z) = \sqrt{x^2 + y^2 + z^2}$$

Oktaeder (Kugel der Manhattan-Metrik)

$$F(x, y, z) = |x| + |y| + |z|$$

Würfel (Kugel der Maximum-Metrik)

$$F(x, y, z) = \max(|x|, |y|, |z|)$$

Torus Die folgende Funktion entspricht nur einem Torus, wenn gleichzeitig der Isowert $I = 0$ gewählt wird.

$$F(x, y, z) = (x^2 + y^2 + z^2 + 0.375)^2 - 2(x^2 + y^2)$$