Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Technische Aspekte Multimodaler Systeme
Seminar Informatikanwendungen in Nanotechnologien
Betreuer: Bernd Schütz
Sommersemester 2014

Shadingalgorithmen zur Visualisierung nanostrukturierter Oberflächen

Laszlo Korte
Ottersbekallee 9
20255 Hamburg
1lkorte@informatik.uni-hamburg.de
23. Juli 2014

Inhaltsverzeichnis

1 Einleitung

Im Folgenden wird erklärt, wie sich nanostrukturierte Oberflächen, deren geometrische Strukturen beispielsweise mit Hilfe eines Rasterkraftmikroskops erfasst wurden, visuell auf einem Bildschirm abbilden lassen. Der Schwerpunktliegt auf den Ideen der Algorithmen, die benutzt werden, um die Messdaten in Farbwerte, die von Pixeln dargestellt werden können, umzuwandeln. Die verwendeten Techniken sind allgemein in der Computergrafik etabliert, sodass sich das hier Erklärte auch zu großen Teilen auf Themengebiete außerhalb der Nanotechnologie übertragen lässt.
Allgemein ist Shading die Simulation der Oberflächeneigenschaften von Objekten. Im Folgenden ist damit die visuelle Darstellung der Oberflächenstruktur auf einem Bildschirm mit Hilfe eines Beleuchtungmodells gemeint.

2 Vektorrechnung

Um den dreidimensionalen Raum, der dargestellt werden soll, zu modellieren, wird sich der Vektorrechnung bedient. Anfangs wird von einem euklidischen Vektorraum mit drei Dimensionen ausgegangen, später wird dieser aus pragmatischen Gründen um eine Dimension erweitert. In diesem Raum stellt ein Vektor (x, y, z) einen Messpunkt dar, der von dem Mikroskop bestimmt wurde. Die Zuordnung der Koordinatenachsen ist beliebig. Vorstellen kann man sich, dass das Mikroskop die xy-Ebene abfährt und für jede Messstelle, den zugehörigen z-Wert ermittelt.
Mit den Methoden der Vektorrechnung stehen verschiedene Möglichkeiten zur Verfügung, mit diesen Vektoren zu rechnen. Zwei Vektoren können addiert oder subtrahiert werden um eine Parallelverschiebung/Translation in eine Richtung zu erreichen. Über das Skalaprodukt kann der Winkel zwischen zwei Vektoren bestimmt werden und per Multiplikation mit einer 3x3 Matrix können Vektoren linear transformiert werden. Zur Transformation zählen die Skalierung in Richtung der Koordinatenachsen, die Rotation um eine Achse und die Scherung. Letztere wird hier aber nicht weiter betrachtet.
Diese Techniken können verwendet werden, um die vom Mikroskop gelieferten Vektoren in unserem Koordinatensystem wie gewünscht zu positionieren, mit dem Ziel, beliebige Blickwinkel auf die Oberfläche zu simulieren.

3 Homogene Koordinaten

Da die Parallelverschiebung keine lineare Operation ist, kann sie für einen gegebenen Vektor nicht durch Matrixmultiplikation, sondern nur über Vektoraddition mit einem

anderen Vektor erzeugt werden. Um die Implementation aber zu vereinheitlichen, wird in der Computergrafik darauf zurückgegriffen, den Raum um eine Dimension zu erweitern. Diese Erweiterung ist unter dem Namen Homogene Koordinaten bekannt und erlaubt es, über eine lineare Scherung in der vierten Dimension eine Parallelverschiebung in den anderen drei Dimensionen zu erlangen. Das in den Vektoren hinzukommende vierte Feld ist ersteinmal nicht wichtig, wird aber mit dem Wert 1 initialisiert und kann später benutzt werden, um eine perspektivische Projektion auf den Bildschirm zu erzeugen. In den Matrizen kommen durch die homogene Erweiterung eine Zeile und eine Spalte hinzu. Die ersten drei Felder der Zeile werden auf 0 gesetzt, da sie nicht benötigt werden. Die drei Felder in der neuen Spalte können nun für die Parallelverschiebung in Richtung der drei Koordinatenachsen benutzt werden. Das Feld unten rechts an dem Schnittpunkt der neuen Zeile und neuen Spalte wird wie das neue Feld der Vektoren auf 1 gesetzt.
Mehrere Transformations-Matrizen können auch direkt miteinander multipliziert werden, um eine neue Transformations-Matrix zu erzeugen, die die beiden ursprünglichen Transformationen beinhaltet. Wichtig ist es, hierbei die Reihenfolge zu beachten, weil die Matrixmultiplikation nicht kommutativ ist.

4 Projektion auf die Bildschirmebene

Mit den bisher genannten Mitteln ist es also möglich, mit einfacher Matrixmultiplikation Vektoren beliebig im Raum zu positionieren. Um aus diesem virtuellen Raum nun auf die Pixelkoordinaten zu kommen, ist eine Projektion der vierdimensionalen Vektoren auf die zweidimensionale Bildschirmebene notwendig. Hierfür gibt es zwei Möglichkeiten. Zum einen die orthogonale Projektion, bei der einfach nur zwei der Vektorwerte benutzt und die anderen beiden verworfen werden, um eine x- und eine y- Koordinate zu erzeugen. Diese Methode führt aber zu einem seltsamen Seheindruck, da der gewohnte perspektivische Effekt, dass entfernte Objekte kleiner wirken, nicht simuliert wird. Zum anderen gibt es die perspektivische Projektion, bei der ebenfalls zwei der Werte eines Vektors benutzt werden, allerdings jeweils noch mit dem vierten Wert, der durch die Erweiterungen zu homogenen Koordinaten hinzugekommen ist, multipliziert wird. Für vom Ursprung entfernte Vektoren (x,y,z,w) ergibt sich durch die Parallelverschiebung, die, wie weiter oben beschrieben, über eine Scherung in der vierten Dimension erzeugt wurde, ein kleinerer Wert für w, sodass diese dann in der Projektion näher am Ursprung, der als Fluchtpunkt benutzt wird, liegen. Um auch die Bereiche der Oberfläche, die zwischen zwei Messpunkten liegt auf dem Bildschirm abbilden zu können, wird davon ausgegangen, dass jeweils drei benachbarte Messpunkte ein Dreieck bilden. Aus den Dreiecken ergibt sich eine lückenlose Fläche. Die Messpunkte sind somit auch Eckpunkte der Dreiecke.

5 Modellierung der Beleuchtung

Nun konnten zwar die Messpunkte, bzw die daraus zusammengesetzten Dreiecke, über Transformation und Projektion den Pixelkoordinaten zugeordnet werden und somit ist klar, welche Pixel zur Darstellung der Oberfläche angesprochen werden müssen, allerdings fehlt noch jegliche Information darüber, in welcher Farbe und Helligkeit die Pixel leuchten sollen. Hier kommt das Shading, also die Schattierung, ins Spiel. Um die Struktur einer Oberfläche visuell zu erkennen, sind die Helligkeitswerte zentral, die Farbwerte spielen nur eine untergeordnete Rolle.
In der Realität entsteht der Helligkeitseindruck im Auge darüber, wie hoch die Photonenkonzentration auf der Netzhaut ist, also wieviele Photonen von der Oberfläche in das Auge reflektiert werden. Da es aber nicht möglich und auch nicht sinnvoll ist, das Verhalten einzelner Photonen an einer Oberfläche zu berechnen, wurden in der Computergrafik Beleuchtungsmodelle entwickelt, die die Lichtreflektion an einer Oberfläche vereinfacht, teilweise gar nur empirisch, modellieren. Je nach Modell sind die Ergebnisse unterschiedlich physikalisch korrekt. Besonders für den hier zugrundeliegenden Zweck, eine nanostrukturierte Oberfläche darzustellen, ist eine physikalisch exakte Berechnung überhaupt nicht von Nöten, da es sich um Strukturen handelt, die der Mensch in dem dargestellten Maßstab sowieso nicht sehen kann, also auch keinen Realitätsvergleich hat.
Um hier zwei Alternativen zu nennen seien das Torrance-Sparrow-Beleuchtungsmodell und das Phong-Beleuchtungsmodell aufgeführt. Ersteres wurde von K. E. Torrance und E. M. Sparrow entwickelt. Es ist physikalisch akkurater, aber komplexer zu berechnen. Letzteres wurde von B. T. Phong empirisch entwickelt, ist einfacher zu berechnen und sehr weit in der Computergrafik verbreitet.

6 Diffuses Licht im Phong-Beleuchtungsmodell

Da keine besonders hohe Realitätstreue benötigt wird, sollen hier die Grundlagen des Phong-Beleuchtungsmodells beschrieben werden. In diesem Modell wird von drei verschiedenen Arten der Lichteffekte ausgegangen: dem ambienten Licht, dem diffusen Licht und dem reflektierenden Licht. Das ambiente Licht verleiht dem zu schattierenden Objekt seine Farbe, das diffuse Licht sorgt für die Schattierung und das reflektierende Licht fügt einem Objekt Glanzreflexe, wie sie von sehr glatten Oberflächen bekannt sind, hinzu.
Um einen Tiefeneindruck der Oberfläche zu bekommen, ist das diffuse Licht am wichtigsten. Darum soll das Prinzip der Berechnung an diesem gezeigt werden. Das reflektierende Licht, lässt sich dann nach einem sehr ähnlichen Prinzip berechnen.
Die Helligkeit soll für jeden Messpunkt berechnet werden. Es wird vorrausgesetzt, dass es eine unendlich weit entfernte Lichtquelle gibt, deren Lichtstrahlen parallel zueinander auf

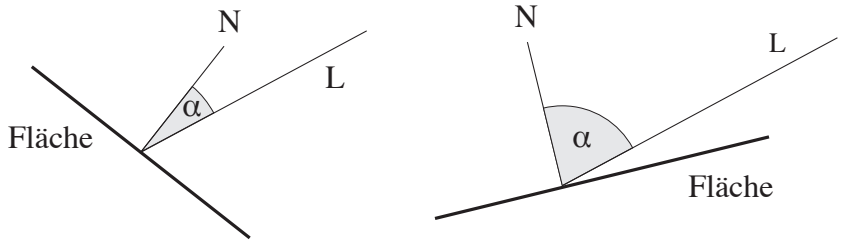

Abbildung 1: Eine steil vom Licht beschienene Fläche (links) wird heller erleuchtet als eine flach beschienene (rechts).

alle Messpunkte fallen. Kurz gesagt: Sei L ein beliebiger aber fester Vektor, der die Richtung des Lichts angibt. Außerdem wird für jeden Messpunkt einen Vektor benötigt, der an diesem Messpunkt senkrecht auf der Oberfläche steht, der so genannte Normalenvektor oder Eckpunktnormale (Vertexnormal). Die Normalen müssen anhand der Eckpunktpositionen nachberechnet werden. Dazu werden zunächst die Flächennormalen der Dreiecke, die sich aus je drei benachbarten Eckpunkten ergeben, berechnet. Dannach wird für jeden Eckpunkt der Durchschnitt der Flächennormalen aller ihm benachbarten Dreiecke gebildet. Um die Flächennormale eines solchen Dreiecks zu berechnen kann das Kreuzprodukt aus zwei der Seiten des Dreiecks gebildet werden. Die Berechnung der Eckpunktnormalen muss für jeden Eckpunkt ausgeführt werden. Phong hat sein Beleuchtungsmodell nach der empirischen Beobachtung entwickelt, dass eine Fläche, die senkrecht von Lichtstrahlen getroffen wird, sehr hell erleuchtet wird, wohingegen eine Fläche, auf die das Licht sehr flach einfällt, dunkler wirkt (Abb. 1). Umformuliert auf unsere Eckpunktnormalen bedeutet dies, dass ein Eckpunkt, dessen Normale in die gleiche Richtung wie der Lichtvektor L zeigt, sehr hell sein muss, aber ein Eckpunkt, dessen Normale senkrecht auf dem Lichtvektor steht, sehr dunkel sein muss. Genau so ein Zusammenhang stellt auch die Kosinus-Funktion eines Winkels dar. Die Helligkeit eines Eckpunktes kann also berechnet werden, indem der Kosinus des Winkels zwischen Eckpunktnormale und Lichtvektor berechnet wird. Hierfür kann die folgende Formel benutzt werden, wobei $\circ$ für das Vektorskalarprodukt steht:

$$cos(\alpha) = \frac{\vec{u} \circ \vec{v}}{|\vec{u}| \cdot |\vec{v}|} \tag{1}$$

Angenommen, dass der Lichtvektor auf die Normalenvektoren auf die Länge 1 normiert worden sind, kann die Formel sogar noch vereinfacht werden. Dann ergibt sich für die Helligkeit H(P) an einem Eckpunkt P für einen Lichteinfach in Richtung L, wobei N(P) die Eckpunktnormale von P sei:

$$H(\vec{P}) = \vec{L} \circ N(\vec{P}) \tag{2}$$

7 Rasterung der Dreiecke

Da nun also für jeden Messpunkt sowohl ein Helligkeitwert bekannt ist, als auch seine projizierte Position auf dem Bildschirm in Pixelkoordinaten, kann für jeden Messpunkt nun schon ein Pixel auf dem Bildschirm angesteuert und entsprechend beleuchtet werden. Was abschließend noch fehlt, ist die Beleuchtung der Pixel, die in einem Dreieck zwischen drei Messpunkten liegen und bisher keinen Helligkeitwert zugewiesen bekommen haben. Deren Helligkeit muss anhand der umliegenden drei Messpunkte interpoliert werden. Dafür werden die drei Helligkeiten der drei Messpunkte, gewichtet nach dem Abstand zu dem betroffenen Pixel, zusammenaddiert. Rechnerisch funktioniert dies per Baryzentrische Interpolation so, dass dieser Pixel das äußere Dreieck aus Messpunkten in drei kleinere Dreiecke teilt (Abb. 2). Jedes dieser kleineren Dreiecke liegt einem Messpunkt gegenüber. Aus dem Verhältnis eines kleineren Dreiecks zum Größeren ergibt sich die Gewichtung mit der die Helligkeit des gegenüberliegenden Messpunktes in die Helligkeit des Pixels mit einfließt.

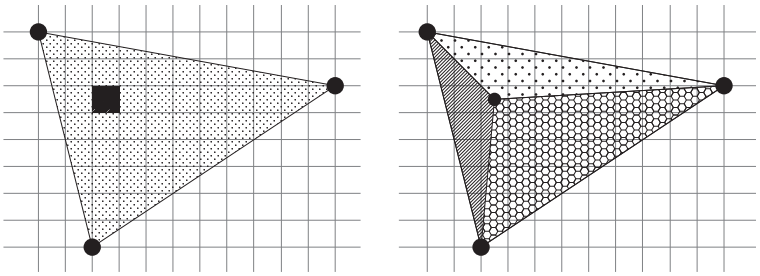

Abbildung 2: Gewichtung der umgebenden Eckpunkte zur Interpolation eines Pixels

8 Interpolation der Normalenvektoren

Anstatt, wie oben beschrieben, die Helligkeit für die Eckpunkte zu berechnen und dann über die Pixel linear zu interpolieren(Gouraud-Shading), gibt es auch die Möglichkeit, für die Eckpunkte statt der Helligkeit nur die Normalenvektoren zu berechnen, diese dann über die Pixel zu interpolieren, um somit für jeden Pixel, statt nur für jeden Messpunkt, einen Normalenvektor zu erhalten. Damit kann dann für jeden Pixel die Helligkeit in Abhängigkeit des Winkels seines Normalenvektors zum Licht berechnet werden. Die letztere Möglichkeit wird Phong-Shading genannt, ist aber nicht zu verwechseln mit dem Phong-Beleuchtungsmodell, welches oben beschrieben wurde. Phong-Shading erzielt noch glatter erscheinende Ergebnisse als das Gouraud-Shading, ist aber etwas komplexer zu berechnen, da die Winkelberechnung für jeden Pixel durchgeführt werden muss und die Pixelanzahl die Zahl der Messpunkte meist übersteigt.

9 Fazit

Somit wurde der Weg von Messdaten aus einem Mikroskop bis zur Abbildung auf die Helligkeit von Pixeln eines Bildschirms vollendet. Die beschriebenen Techniken lassen sich in vielen Bereichen der Computergraphik wiederfinden, wie in der Pipeline einer Grafikkarte und damit auch in Vertex- und Fragmentshadern von OpenGL.

10 Literaturverzeichnis

Literatur

[1] A. Beutelspacher, U. Rosenbaum. *Projektive Geometrie*. Wiesbaden, 2004

[2] S. R. Buss. *3D Computer Graphics: A Mathematical Introduction with OpenGL*. Cambridge University Press, 2003

[3] M. Gross. *Graphische Datenverarbeitung*. Eidgenössische TH Zürich, http://graphics.ethz.ch/teaching/viscomp11/downloads/GDV_Part1.pdf, 23.07.2014