69 Cyberpunk 2077: Die Mathematik hinter der Computergrafik

Lineare Algebra liefert die Grundlage für Beleuchtungsmodelle und die dreidimensionale Bewegung in der Computergrafik.

Wir wollen in diesem Kapitel untersuchen, inwieweit Mathematik in Computerspielen eine Rolle spielt. Dabei werden wir kennenlernen, wie uns die Vektorrechnung (lineare Algebra) hilft, Objekte (wie z.B. Wände in einem Labyrinth) im dreidimensionalen Raum zu charakterisieren und hilft diese Objekte zu bewegen, zu rotieren. Mathematik ist auch nötig, um zu berechnen, welche Teile eines Objektes überhaupt sichtbar sind und wie sich die Beleuchtung auf die Darstellung der Oberfläche auswirkt.

Die Grundidee der 3D-Grafik besteht darin, eine mathematische Beschreibung einer Welt in ein Bild davon zu verwandeln, wie diese Welt für jemanden im Inneren der Welt aussehen würde. Die mathematische Beschreibung könnte so sein: es gibt ein Objekt Box mit dem Mittelpunkt (1,2,3), Seitenlänge 2 und Farbe Gelb. Der Betrachter (genaugenommen seine Augen) befindet sich bei (5,14,9) und schaut direkt auf den Mittelpunkt der Box. Damit kann man berechnen, wie die Welt für diesen Betrachter aussehen würde.

Abbildung 97: Titelbild Cyberpunk 2077 (Quelle: CD Projekt Red, Press Center)

Nehmen wir nun einen Betrachter mit Augen im Punkt A; zwischen ihm und der Box ist eine Glasplatte (Ebene), auf die er die gesehene Box malen will. Eine der Boxecken befindet sich am Punkt B, und um herauszufinden, wo diese Ecke auf der Glasplatte sein sollte, zieht man eine Linie L von seinen Augen A zur Boxecke B und sieht dann, wo

diese Linie durch die Glasplatte G, verläuft. So einen Schnittpunkt Gerade mit Ebene kann man leicht bestimmen. Verfolgt man diese Prozedur für jede Stelle der Box, erhält man ein Abbild der Box auf der Glasplatte, also dem Bildschirm. Und genau das passiert sehr grob im Computer, wenn man z.B. im Spiel wie Cyberpunk 2077 herumläuft, obwohl die Details doch etwas komplizierter sind.

Das Raytracing-Verfahren

Was gerade oben beschrieben wurde, ähnelt dem, was der Computer jedes Mal tut, wenn Sie im Spiel wie Cyberpunk 2077 herumlaufen, obwohl die Details etwas anders sind. Die neuesten Computerspiele verwenden kompliziertere Beschreibungen der Welt, indem sie gekrümmte Oberflächen, NURBS und andere seltsam klingende Dinge verwenden, aber am Ende reduziert sich das Ganze immer auf Dreiecke.

Nehmen wir einmal an, dass wir unser Objekt in Dreiecke trianguliert haben. Der nächste Schritt ist das Beleuchtungsmodell: die Menge des Lichts oder Schattens, die auf jede Dreiecksfläche fällt, hängt von der Ausrichtung dieses Dreiecks in Bezug auf die Lichtquelle ab. Wenn ein Dreieck direkt der Lichtquelle zugewandt ist, wird es hell beleuchtet, aber wenn es von der Lichtquelle abgewandt ist, liegt es im starken Schatten. Die Ausrichtung jeder einzelnen Dreiecksfläche wird durch einen Normalenvektor beschrieben, der senkrecht aus der jeweiligen ebenen Fläche heraus zeigt. In ähnlicher Weise können die Strahlen der Lichtquelle durch einen Vektor beschrieben werden, und das Skalarprodukt dieser beiden Vektoren ergibt den Cosinus des Winkels zwischen diesen beiden Vektoren.

An dieser Stelle wollen wir kurz das *Skalarprodukt* vorstellen. Um das Skalarprodukt zweier Vektoren $u = (x_1, y_1, z_1)$ und $v = (x_2, y_2, z_2)$ zu berechnen, multipliziert man deren jeweilige Koordinaten und addiert sie:

$$\langle u, v \rangle = x_1 x_2 + y_1 y_2 + z_1 z_2.$$

Das Skalarprodukt kann auch als Produkt aus der Länge der beiden Vektoren und dem Winkel θ zwischen ihnen geschrieben werden:

$$\langle u, v \rangle = (\text{Länge von } u) \cdot (\text{Länge von } v) \cdot \cos\theta.$$

Wenn ein Dreieck direkt der Lichtquelle zugewandt ist, beträgt der Winkel zwischen der Normalen und der Richtung der Lichstrahlen $0°$, und das Skalarprodukt dieser beiden Vektoren ergibt den Wert 1, d.h. es fällt die maximale Lichtmenge auf diese Fläche und sie sollte daher weiß schattiert werden. Wenn eine dreieckige Fläche fast seitlich zu den Lichtstrahlen liegt, so dass der Winkel zwischen der Flächennormalen und der Lichtrichtung nahe bei $90°$ liegt, ist der Wert des Skalarprodukts nahe bei 0, also sollte dass Dreieck sehr dunkel schattiert werden, weil es im Schatten liegt. Ist ein Dreieck vom Licht abgewandt, also der Winkel größer als $90°$ ist, dann wird das Skalarprodukt negativ und dieser Wert wird normalerweise vom Computer auf Null gesetzt. Nun kommt hinzu, dass sich alle 3D-Objekte bewegen. Somit muss der Computer diese

(relativ einfachen) Berechnungen ständig für alle Dreiecke wiederholen (das sog. Rendering). Hierzu gibt es inzwischen spezialisierte Hardware, die diese Berechnungen erledigt. Häufig wird auch die Triangulierung vergröbert, damit die Berechnungen für schnell bewegte Objekte effektiver wird; das menschliche Auge kann ohnehin nicht so fein auflösen.

Abbildung 98: Screenshot 2077 (Quelle: CD Projekt Red, Press Center)

Als nächsten Schritt möchte man die Oberflächen realistischer erscheinen lassen, ohne die Anzahl der Dreiecke weiter zu erhöhen. Das geschieht mit dem Konzept der Textur. Für Anwendungen, die eine schnellere Rendering-Zeit erfordern, wie z.B. das Echtzeit-Rendering in Computerspielen, verwendet man das *Normal Mapping*. Es entspricht dem zuvor beschriebenen Verfahren, bei dem die Textur mit ihren Unebenheiten durch lokal hochaufgelöste Modelle erzielt wird.

Für Anwendungen, wie z.B. in Filmen, die lange Zeiten für das Rendern einzelner Bilder zulassen, gibt es eine andere Methode. Man versucht mit dem *Displacement Mapping* mit positiven und negativen Werten genau zu kodieren, wie weit sich die Punkte auf der Oberfläche des vereinfachten Gitters von den Punkten auf dem Original-Gitters unterscheiden.