

Avengina: Etwas Technisches

Erörterung der häufig gestellten Frage: “Warum will die nicht mit meiner Grafikkarte?!”

An Auskenner: Das vorliegende Paper ist an Laien gerichtet. Dem Autor ist bekannt, dass es einer wissenschaftlichen Kritik nicht standhalten würde.

Herzlich willkommen, ich werde versuchen meine Ausführungen knapp zu halten. Zunächst sind dennoch einige Grundlagen zu behandeln:

Beschäftigen wir uns zunächst mit der Frage: Wie funktioniert so eine Echtzeit 3d Grafikingine heutzutage eigentlich?

Stellen wir uns die Engine als eine Art Betriebssystem für eine ganz bestimmte Sorte von Daten vor. Diese Daten bestehen zum großen Teil aus Zahlen. Diese Zahlen beschreiben das, was wir schließlich zu sehen bekommen, jedoch nicht so, wie wir es sehen:

- Die Zahlen definieren die Objekte der Szene in ihren geometrischen Eigenschaften.
- Die Zahlen definieren die Objekte bezüglich ihrer Materialeigenschaften. Vereinfacht ausgedrückt wird hierbei festgelegt, welche Lichtfarbe wie stark reflektiert wird.
- Die Zahlen positionieren Lichtquellen in der Szene und bestimmen ihre Eigenschaften.
- Die Zahlen bewirken noch manches mehr, was uns hier aber nicht interessieren muss.

Nun, wir haben also Objekte, die haben Geometrie, bilden Gebilde, die sich aus mehreren bis vielen Dreiecken zusammensetzen. Diese Art von Geometrie nennt man Polyeder. Dass man die Dreiecke beim betrachten der Szene nicht oder selten sieht muss uns nicht verwirren. Es genügt zu glauben, dass sie da sind.

Diese Dreiecke werden von virtuellen Lichtquellen beleuchtet, die eine relative Position zum Dreieck im virtuellen Raum einnehmen. Position und Eigenschaften der Lichtquellen (z.B. die Lichtfarbe) und die Materialeigenschaften der Dreiecke bestimmen schließlich unter anderem darüber, wie jedes Dreieck auf dem Bildschirm aussieht. Das bedeutet, dass dieses Aussehen in jedem Frame neu berechnet wird. Ändern sich Eigenschaften der Lichtquelle oder ändert sich ihre Position relativ zum Objekt (zum Dreieck) ändert sich entsprechend auch die Ausgabe auf dem Bildschirm.

Das war sehr knapp, sollte aber als Einleitung genügen.

Kommen wir nun zu der Frage, die es hier zu klären gilt: Warum geht's bei mir nicht?

Dazu müssen wir wissen, dass das oben umrissene „Rendering“ in den Zuständigkeitsbereich der Grafikkarte fällt. Diese können wir uns als einen hochspezialisierten Computer vorstellen, der im 3d Betrieb folgendes extrem schnell lösen muss: Aus 3d Daten (den Zahlen) mache Pixel auf dem

Bildschirm.

Dazu sind viele komplizierte Vorgänge nötig, uns interessiert hier nur einer: Wie berechnet die Grafikkarte die Farbe eines Pixels? Denn, zur Wiederholung: Es geht um nichts anderes als um Farben von Pixeln.

Die GPU (der Rechenkern der Grafikkarte) nimmt sich ein Dreieck vor (1), betrachtet seine Eckpunkte (Profiausdruck: Vertices (Singular: Vertex)), die Position der Lichtquelle, allerlei Eigenschaften und berechnet zu den Eckpunkten passende Ergebnisse (das ist jetzt sehr abstrakt). Dies ist ein wichtiger Schritt. Die gewonnenen Ergebnisse werden nun weitergegeben, im nächsten Schritt (2) wird aus den Ergebnissen ein Ergebnis berechnet, welches auf die Position eines Pixels auf dem Bildschirm passt. Aus diesem Ergebnis soll später die endgültige Pixelfarbe berechnet werden.

Bis vor einigen Jahren war es nun so, dass die Grafikkarten den oben beschriebenen Vorgang nur fest verdrahtet vornehmen konnten. Man spricht auch von einer Fixed-Function-Pipeline. Damit war es dem Programmierer lediglich möglich, das Verhalten dieser Pipeline mit Parametern zu beeinflussen, sprich, hier und da ein paar Schalter umzulegen. Für eine aufwändige Beleuchtungsberechnung reichte das nicht aus. Vor einigen Jahren kamen darum die ersten Karten mit einer programmierbaren Pipeline auf den Markt. Die Karten waren jetzt in der Lage, kleine Programme auszuführen, mit denen der Programmierer mehr Einfluss auf das Ergebnis nehmen konnte. Zunächst konnten solche Programme nur pro Vertex ausgeführt werden (sog. Vertexshader, richtiger: Vertexprogramme) (siehe (1)), kurze Zeit später erschienen Grafikkarten mit der Fähigkeit, frei programmierbaren Code pro Pixel auszuführen (Pixelshader, richtiger: Fragmentshader, noch richtiger: Fragmentprogramme) (siehe (2)). Damit war das ShaderModel2.0 geboren (heute sind wir übrigens bei 4.0 (DirectX10)).

Damit nun das Beleuchtungskonzept von Avengina aufgeht, muss die Grafikkarte Vertex- wie auch Fragmentprogramme unterstützen. Nur so kann das Wirken von Licht und Schatten angemessen und im Sinne des Schöpfers berechnet werden.

Die ältesten Grafikkarten, die dies unterstützen sind die GeforceFX (das ist die 5. Generation Geforce, also 5200, 5600, 5700...) und die Radeon9500 Karten.

Sir Karman