Sprechertext Shader

Inhalt

1001 (A) Einleitung 2

1002 (A) Flat-Shading 2

1003 (A) Gouraud-Shading 3

1004 (A) Phong-Shading 3

1005 (I) Vergleich zwischen Flat-, Gouraud - und Phong-Shading 3

1006 (A) Vertex Shader 4

1007 (I) Vertex Shader 4

1008 (A) Geometry-Shader 4

1009 (A) Pixel- / Fragment-Shader 5

1010 (A) Toon-/Cel-Shading 5

1011 (A) Programmierung eines Shaders 6

1001 (A) Einleitung

Shader sind spezielle Programme, welche in Computerspielen, in der Postproduktion von Videoinhalten und bei Computer Generated Imagery, kurz CGI, zum Einsatz kommen. Shader waren ursprünglich, wie der Name schon sagt, für das Schattieren bzw. Erzeugen von verschieden Stufen von Licht, Dunkelheit und Farben in der Computergrafik zuständig. Beispiele hierfür sind Flat-, Gouraud - und Phong-Shading, welche fest im Grafikchip verbaut sind. Heutzutage können Shader frei programmiert werden und übernehmen auch Aufgaben, die nichts mit dem ursprünglichen Schattieren zu tun haben, wie z.B. das Erzeugen neuer Geometrien.

Die Verarbeitung von Shadern findet in den sogenannten Shadereinheiten statt. Diese befinden sich in Grafikchips von Grafikkarten, welche ein wichtiger Bestandteil in Computern, Spielekonsolen, Smartphones und anderen vergleichbaren Geräten sind. In Grafikkarten gibt es eine sogenannte Grafikpipeline, welche die Reihenfolge der auszuführenden Shader festlegt. Zur Nutzung von Shadern ist noch eine programmierbare Schnittstelle nötig, wie z.B. DirectX oder OpenGL.

1002 (A) Flat-Shading

Flat-Shading, manchmal auch Constant-Shading genannt, ist ein sehr einfaches Schattierungsverfahren, da pro Polygon nur eine Farbe möglich ist. Wenn man z.B. ein Dreieck als Polygon nimmt, wird der Farbwert unteranderem aus dem Flächen-Normalenvektor, der Flächenfarbe und der Lichtintensität berechnet.

Danach werden alle Pixel des Polygons auf diese Farbe gesetzt. Als Ergebnis erhält man dann, besonders bei gekrümmten Oberflächen, eine Facettenartige Darstellung. Das ist auch der größte Nachteil des Flat-Shading, weil es dadurch zum sogenannten Mach-Band-Effekt kommt, wodurch die entstandenen Kanten besonders stark vom menschlichen Auge wahrgenommen werden.

Um die facettenartige Darstellung zu vermindern, muss die Anzahl der Polygone erhöht werden, wodurch der Rechenaufwand aber steigt. Aufgrund der genannten Nachteile kommt das Flat-Shading meistens bei Objekten mit ebenen Flächen wie z.B. Quader, Würfel oder Pyramiden zum Einsatz.

1003 (A) Gouraud-Shading

Das Gouraud-Shading wurde nach seinem Entwickler Henri Gouraud benannt, der es erstmals 1971 vorstellte.

Das Besondere am Gouraud-Shading ist, dass im Gegensatz zum Flat-Shading, Farbverläufe dargestellt werden können. Dafür werden die Normalenvektoren an den Vertices berechnet. Diese erhält man durch den Mittelwert der Normalen aller angrenzenden Polygone. Danach werden durch Interpolation die Farbwerte an den Vertices berechnet.

Durch dieses Verfahren erscheinen die Kanten der Polygone weniger hart, wodurch Objekte besser rund oder gekrümmt dargestellt werden können. Ein Nachteil ist die bei manchen Objekten fehlerhafte Darstellung von Glanzlichtern und das Vorkommen von Sprüngen im Farbverlauf.

1004 (A) Phong-Shading

Das Phong-Shading, benannt nach seinem Entwickler Bùi Tường Phong, wurde erstmals 1975 vorgestellt.

Bei diesem Verfahren werden zu Beginn, wie beim Gouraud-Shading auch, die Normalen an den Vertices eines Polygons berechnet. Daraufhin wird beim Einfärben der Pixel eine Normale zwischen den Eckpunktnormalen interpoliert, mit der die Farbe entsprechend der Beleuchtung ausgewertet wird. Dadurch erhält jedes Pixel die entsprechende Beleuchtung und somit auch die korrekte Farbe.

Das Phong Shading liefert aufgrund von glänzenden Oberflächen und Specular highlights, welche hier für den hellen Fleck sorgen, ein realistischeres Ergebnis als das Gouraud Shading, ist aber auch deutlich rechenintensiver.

1005 (I) Vergleich zwischen Flat-, Gouraud - und Phong-Shading

Hier kannst du die verschiedenen Shader anhand eines Objekts vergleichen, indem du unten einen Shader auswählst.

1006 (A) Vertex Shader

Vertex-Shader sind Programme, welche im Verlauf der Grafikpipeline in den Shadereinheiten einer Grafikkarte ausgeführt werden. Diese verarbeiten die sogenannten Vertices, bei denen es sich um Eckpunkte eines 3D-Modells handelt.

Beim Vertex-Shader werden die Koordinaten der einzelnen Vertices im dreidimensionalen Raum für die zweidimensionale Darstellung transformiert. Dadurch lässt sich die Geometrie und somit die Form von Objekten beeinflussen, was sich wiederum auch auf die Beleuchtung auswirken kann. Da der Shader aber pro Vertex aufgerufen wird, kann er keine neuen Punkte zum 3D-Modell hinzufügen.

1007 (I) Vertex Shader

Hier ist ein Vertex Shader aktiv, welcher durch verschieben des Reglers Einfluss auf die Geometrie der Wellen nimmt.

1008 (A) Geometry-Shader

Der Geometry-Shader wird in der Grafikpipeline nach dem Vertex Shader aufgerufen, um neue primitive Geometrien aus bereits vorhandenen Punkten, Linien und Dreiecken zu erzeugen und diese erneut in die Grafikpipeline einzufügen.

Beispiele für die Anwendung des Geometry-Shader sind die Erzeugung von Schattenvolumen oder die Erzeugung von Fell- oder Haargeometrie.

Um Haare zu erzeugen, erhält der Geometry-Shader die benötigten Vertices vom Vertex-Shader als Input. Diese Vertices werden dann durch mehrere Kopien ersetzt, wodurch eine Haar-Struktur entsteht. Nach der Verarbeitung werden die fertigen Fragmente an den Pixel-Shader weitergegeben.

1009 (A) Pixel- / Fragment-Shader

Pixel-Shader, auch Fragment-Shader genannt, werden ebenfalls in den Shadereinheiten einer Grafikkarte ausgeführt. Der Shader ist für die Farbberechnung der einzelnen Pixel, auch Fragmente genannt, zuständig und folgt in der Grafikpipeline auf den Geometry-Shader.

Pixel-Shader werden genutzt, um eine realistische Darstellung von Oberflächen- und Materialeigenschaften zu erreichen oder eine Texturdarstellung zu bewerkstelligen bzw. zu verändern. Ein Pixel kann dabei aus mehreren Fragmenten bestehen, was zum Beispiel bei Transparenz der Fall ist, wenn ein Objekt hinter einer Scheibe steht. Die richtige Farbe wird über eine Beleuchtungsberechnung zugewiesen.

1010 (A) Toon-/Cel-Shading

Beim Toon-Shading, auch Cel Shading genannt, handelt es sich um eine Technik zum nicht-fotorealistischen Rendern von 3D-Computergrafiken. Als Ergebnis erhält man bei diesem Verfahren eine Optik, die der von gezeichneten Comics oder Zeichentrickfilmen entspricht.

Um diesen Effekt zu erhalten, wird auf weiche Verläufe verzichtet und es kommen nur drei oder vier Helligkeitsstufen zum Einsatz. In der Regel sind das weiß, hellgrau und dunkelgrau. Außerdem verzichtet man meistens auf eine Textur und verwendet nur einzelne Farbtöne.

Um die schwarzen Linien zu erhalten, welche die innere und äußere Kontur des Objekts darstellen, invertiert man Polygone die aufgrund der Perspektive nicht sichtbar wären. Dafür wird das Backface Culling, welches nicht sichtbare Polygone aufgrund der Performanceverbesserung entfernt, rückgängig gemacht. Dies wird teilweise mehrmals mit leichten Variationen durchgeführt, um eine bessere Kontur zu erhalten. Die neu hinzugefügten Polygone werden zum Schluss schwarz gefärbt.

1011 (A) Programmierung eines Shaders

Der Code eines Shaders erinnert stark an die Programmiersprache C, weshalb es viele Parallelen zu dieser und auch anderen Programmiersprachen gibt. Dieses einfache Beispiel wurde in der OpenGL Shading Language, kurz GLSL, geschrieben.

Shader haben in der Regel folgenden Aufbau. Als erstes wird die Version deklariert, gefolgt von den Input- und Output-Variablen. In der Main-Funktion werden die Inputs verarbeitet und das Ergebnis an die Output-Variablen weitergegeben.

In diesem Beispiel wird ein Vertex Shader verarbeitet. Als Input wird eine Vertex-Position benutzt, welche als Output eine Farbe erhalten soll. In der Main-Funktion wird die Position von Vektor 3 an Vektor 4 übergeben und danach die Farbe zugewiesen.

Der Output vom Vertex Shader wird dann als Input beim Fragment Shader genutzt. Dort wird die Farbe für ein Fragment übernommen.

Als Ergebnis erhält man ein Fragment mit der zuvor einprogrammierten Farbe.