

Rechnerarchitektur: 3D Graphikhardware

Eduard Eipert

Februar 18, 2003

Zusammenfassung

Graphikchips sind heute komplizierter als Prozessoren. Mit mehr als 100 Millionen Transistoren die komplizierteste Hardwarekomponente in einem PC.

Inhaltsverzeichnis

1	Richtung und Ziel	3
1.1	Richtung	3
1.2	Ziel	3
2	Aufbau und Funktionsweise	4
2.1	Der Graphikprozessor	7
2.1.1	Textureinheit	7
2.1.2	Pixel-Pipeline	10
3	Beispiele	12
3.1	Ati Radeon 9700 PRO	12
3.2	Nvidia GeForceFX	13
4	Abschluss	16

Abbildungsverzeichnis

2.1	Die Funktionseinheiten des Graphikprozessors	6
2.2	Multitexturierung	8
2.3	Der Pixel Shader	9
2.4	Der Vertex Shader	11
3.1	ATI Radeon 9700 Blockdiagramm	13
4.1	Ergebnis der Kombination mehrerer Algorithmen (Quelle: c't) . .	16

Kapitel 1

Richtung und Ziel

Die ganze Anstrengung vieler Graphikchipentwickler dient einzig der 3D-Spiele-Industrie und deren Anhängern. Denn die wenigen 2D-Funktionen, benötigt für Textverarbeitung und Tabellenkalkulationen, lassen sich im Mainboardchipsatz oder im Prozessor unterbringen. Übrig bleiben noch die Spezialkarten für professionelle Anwender.

1.1 Richtung

Die Entwicklung der Graphikprozessoren verläuft in zwei Richtungen. Auf der einen Seite wird an die Zusammensetzung von Landschaften, Figuren und Objekten aus immer feineren Elementen gearbeitet; dabei müssen Verfahren dafür sorgen, daß die zentralen Komponenten des Rechners möglichst wenig mit den Rohdatenmengen in Berührung kommen. Die durch die CPU und Arbeitsspeicher begrenzte Systemleistung bremst die weiterentwickelte Graphikhardware aus. Auf der anderen Seite wird die Texturierung weiterentwickelt; die Berechnung der Pixelfarben dient der möglichst realistischen Darstellung der natürlichen und künstlichen Lichtquellen auf verschiedenen Materialien mit allen Reflexions-, Schatten- und Transparenzeffekten.

1.2 Ziel

Ziel dieser Anstrengungen ist die Darstellung fotorealistischer Szenarien, 3D-Welten. Dieser versucht man durch immer mehr Details zu verhelfen, so daß man möglichst den künstlichen Ursprung nicht mehr ansieht.

Kapitel 2

Aufbau und Funktionsweise

Was macht der Graphikchip?

Er malt die als Polygonnetz angelieferten Objekte mit Texturen (Oberflächenbezügen) aus und bestimmt die Sichtbarkeit jedes einzelnen Bildpunktes (Pixel) – ob es von vor ihm liegenden Objekten verdeckt wird oder nicht – mittels der Werte in seinem Z-Buffer. [Bert00a]

Abbildung 2.1 auf Seite 6 zeigt den Aufbau und die Funktionsweise eines Graphikchips.

Nicht im Graphikprozessor, jedoch unverzichtbar ist der auf der Graphikkarte **lokale Speicher** (aktuell zwischen 32 und 128 MByte groß). Der lokale Speicher ist unterteilt in vier Bereiche:

- **Front-Buffer**

Der Front-Buffer enthält genau ein Bild und zwar das gerade sichtbare Bild, das vom CRTIC (Cathode Ray Tube Controller) für die Erzeugung des Monitorsignals ausgelesen wird. Er ist ungefähr drei MByte groß. Pro Pixel sind – entsprechend der 16, 24 oder 32 Bit Farbtiefe – zwei, drei oder vier Byte reserviert.

- **Back-Buffer**

Der Back-Buffer ist identisch mit dem Front-Buffer. Im Back-Buffer wird die nächste Szene aufgebaut. Ist diese fertig, wechseln beide Speicherbereiche ihre Funktion: Der Back-Buffer wird zum Front-Buffer und umgekehrt (Buffer-Flipping). Der Umschaltvorgang wird vom Graphikkartentreiber versteckt, indem er das Buffer-Flipping mit dem Bildwechsel des Monitors synchronisiert (VSync on). Damit der Graphikchip mit höchstmöglicher Geschwindigkeit arbeitet, muß die Synchronisierung abgeschaltet werden (zur Leistungsmessung im Benchmark).

- **Z-Buffer**

Im Z-Buffer werden Tiefeninformationen (Z-Koordinaten) als Ganzzahlwerte abgelegt. Bevor ein Bildpunkt im Back-Buffer abgelegt wird, wird

seine Z-Koordinate mit dem Wert im Z-Buffer verglichen. Ist der Wert im Z-Buffer kleiner als der neu errechnete Bildpunkt, wird der Letztere verworfen.

- **Textur-Cache**

Der Texture-Cache enthält lokal gespeicherte Texturen, 3D-Daten (Vertex-Cache) und anderes (Mauszeiger).

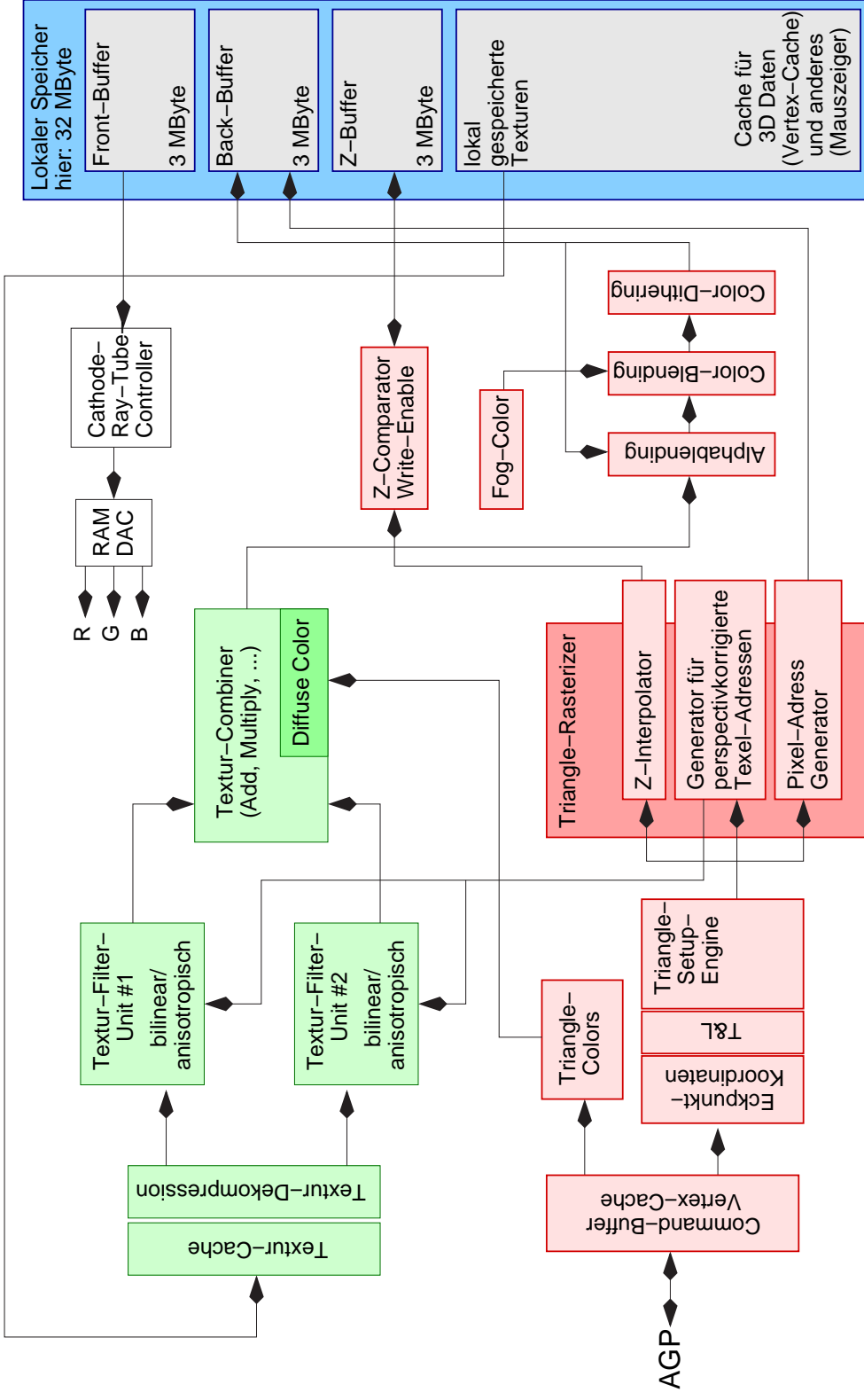


Abbildung 2.1: Die Funktionseinheiten des Graphikprozessors

2.1 Der Graphikprozessor

Der Graphikchip lässt sich grob in zwei Einheiten unterteilen:

- **Textureinheit**

Die Textureinheit ist der erste Ablauf im Graphikchip wenn ein Farbwert einer Textur an einer bestimmten Bildschirm-Koordinate benötigt wird.

- **Pixel-Pipeline**

Die in der Textureinheit gefilterten Texturen werden in den hier gelagerten Recheneinheiten mit andere Parameter zu den endgültigen Farbwerten gerechnet.

2.1.1 Textureinheit

Die Textureinheit ist in den folgenden Modulen unterteilt:

- **Textur-Filter-Unit**

Die **Textur-Filter-Unit** errechnet den Farbwert eines Bildpunktes aus der an dieser Stelle vorgesehene Textur. Texturen können nicht unverändert übernommen werden, denn ein **Texel** (Textur-Element) aus der Nähe betrachtet schachbrettartig aussieht weil er mehrere Pixel überdeckt und andererseits aus großer Distanz nur ein Farbchaos zu erkennen wäre, weil ein Pixel nur ein zufälliges Texel wiedergibt.

Mit **bilineares Filtern** faßt man vier zusammenhängende Texel durch Interpolation zu einem Texel zusammen. Jedoch um das Ergebnis bei flachem Betrachtungswinkel zu verbessern errechnet man aus einer höheren Anzahl von Texeln mittels **anisotropisches Filtern** ein Wert aus. Zusätzlich je nach Betrachtungsabstand kommt eine höher aufgelöste oder eine niedrig aufgelöste Fassung der Textur. Wenn Texturen in verschiedene Auflösungen bereitstehen, nennt man das **MIP-Mapping**. Und darüber hinaus erzeugt die Texture-Filter-Unit **trilinear gefilterte Texturen**. Hierbei wird zuerst noch zwischen den Texturstufen gemittelt, wenn die Entfernung des Objekts nicht genau einer der MIP-Mapping-Stufen entspricht.

- **Textur-Combiner** ab DirectX 7

Der **Textur-Combiner** ist eine komplexe Arithmetikeinheit in der verschiedene Operationen implementiert sind. Verschiedene Texturen werden arithmetisch oder logisch verknüpft; zwei bilinear gefilterte MIP-Mapping-Stufen werden interpoliert für trilinear gefiltertes MIP-Mapping; die Farbwerte einer der beiden Texturen werden als Adress-Offset für den Zugriff auf eine weitere Textur für das Environment-mapped Bump-Mapping interpretiert. Das Environment-Mapped-Bump-Mapping setzt sich zusammen aus dem Bump-Mapping Verfahren, das Prägestrukturen aus mehrlagigen Texturen simuliert, und dem Environment-Mapping Verfahren, wel-

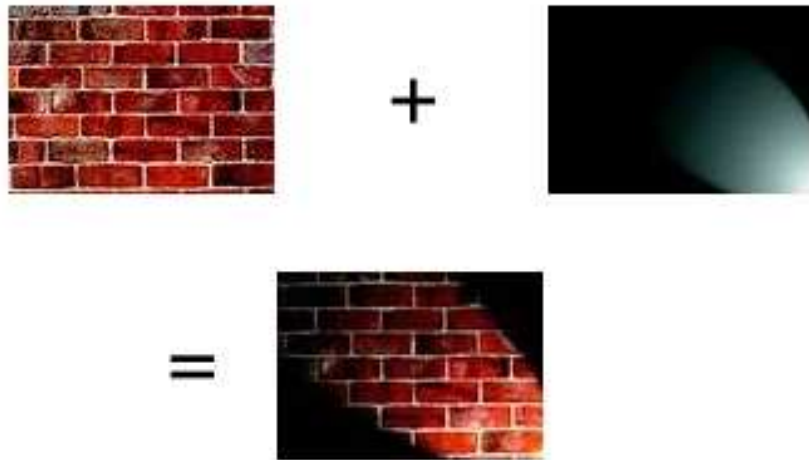


Abbildung 2.2: Multitexturierung

ches die Reflexion der Umgebung auf glatten und metallischen Oberflächen nachahmt.

Eine Kombination aus einer Textur und einer Lighting-Map kann schon ein gutes Ergebnis liefern (siehe Abbildung 2.2).

Das Verfahren **DotProduct3-Bump-Mapping** benötigt eine Textur, deren Texel nicht als Farbwerte, sondern als Richtungsinformation (Normale) zu interpretieren sind und so die Orientierung der Oberfläche in diesem Punkt angibt (Normal Map) [DP3BM].

- **Pixel Shader** ab DirectX 8

Der Pixel Shader berechnet die endgültigen Farbwerte für die Bildpunkte aus den einzelnen Texturpunkten, wobei eine ALU mit Programmsteuerung unterschiedliche Effekte erzeugen kann.

Die Abbildung 2.3 auf Seite 9 zeigt eine schematische Darstellung des Pixel-Shaders.

Nachdem die vom Vertex Shader angelieferten 3D-Punkte wieder zu Dreiecken zusammen gesetzt worden sind und der Rasterizer diese in Pixel zerlegt hat, berechnet der Pixel Shader die Einfärbung der Pixel aus der zugehörigen Textur und der Beleuchtungssituation. Zuerst bestimmt die Sampler Stage mit Hilfe der Texturkoordinaten eine Gruppe von Texeln für die Einfärbung. Dabei interpoliert der Sampler zwischen vier oder

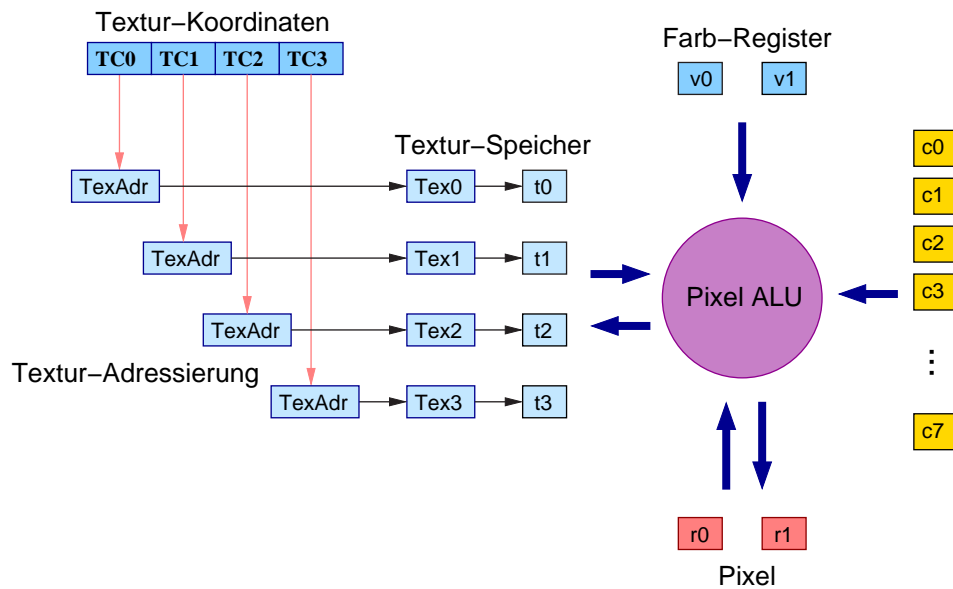


Abbildung 2.3: Der Pixel Shader

mehr Texeln (bilineare oder anisotrope Textur-Filterung). So steht das gefilterte Texturwert für die Weiterverarbeitung in der Pixel-ALU bereit. Es gibt mehrere Arten von Texturadressierung. Texturen können in der ALU überlagert werden (Color Blending). Um verschiedene Arten von Bump-Mapping und Environment-Mapping zu realisieren, werden Texturen einbezogen die Informationen über die Orientierung der Oberfläche (Normal-Map) oder die Richtung des Lichteinfalls enthalten.

Zwei weitere Größen (v_0 , v_1) können zusätzlich zu den Texturwerten von der ALU verarbeitet werden. In den Farb-Register v_0 , v_1 legt der Vertex Shader – der Color-Interpolator im Rasterizer – die Grundfarbe des zugehörigen Bildpunktes ab. Die ALU kann aus acht Konstantenregistern (c_0 bis c_7) feste Parameter abrufen.

Der Befehlsatz der Pixel-ALU umfaßt acht Operationen wie Addition, Multiplikation und Interpolation für Farbmischung sowie Skalarprodukt für die Beleuchtungsberechnungen (Per-Pixel-Lighting).

Am Ende steht ein gefilterter RGBA-Wert in der Pixel-ALU.

2.1.2 Pixel-Pipeline

Die einfachsten Ausführung des Graphikchips zerlegt die Dreiecke in einzelne Punkte (Rasterization), ordnet den Punkten Bildschirmkoordinaten zu (Viewport Mapping) und berechnet die Farbwerte für die Pixel. Aus den Texturen und

der Beleuchtungssituation wird die Farbe und Helligkeit abgeleitet. Vor dem Ablegen der Pixeldaten im Bildspeicher berechnet er bei Bedarf noch Nebel (Fog)- sowie Transparenz-Effekte (Alpha-Blending) und überprüft mit Hilfe des Z-Buffers, ob weiter vorn liegende Objekte den Bildpunkt verdecken und dieser eventuell gar nicht in den Bildspeicher gelangen darf.

- **AGP**

Über den Accelerated Graphics Port (**AGP**) werden Dreiecke bzw. Dreiecksnetze (Triangle Mesh) an den Graphikchip übertragen. Jeder Knotenpunkt (Vertex) wird als XYZ-Koordinaten übertragen, mit der Oberflächenfarbe (RGB-Triple) an dieser Stelle und Texturkoordinaten, die festlegen, welcher Teil der zugehörigen Textur in welche Lage auf das Dreieck gehört.

Der AGP-Slot ist ein spezieller Steckplatz auf Motherboards mit einem Datentransfer zwischen Graphikprozessor und Chipsatz des Mainboards von maximal 266 MByte/s, 533 MByte/s, 1.1 GByte/s oder 2.1 GByte/s (je nach Betriebsmodus 1x,2x,4x oder 8x). Über den Chipsatz hat die Graphikchip direkten Zugriff auf den Arbeitsspeicher und die CPU kann direkt mit der Graphikkarte im AGP-Slot kommunizieren, ohne daß andere Busaktivitäten stören.

- **Transform & Lighting** ab DirectX 7

Die Transform & Lighting - Einheit entlastet die CPU des Rechners indem sie Objekte in allen möglichen Größen und Lagen transformiert. Zusätzlich berechnet sie für jeden Dreieckspunkt die Helligkeit in Abhängigkeit von von der vorhandenen Lichtquellen (**Vertex Lighting**) und wandelt die 3D in 2D-Koordinaten um (perspektivische Projektion)

- **Vertex Shader** ab DirectX 8

Der Vertex Shader besteht aus einer programmgesteuerten Recheneinheit (ALU – Arithmetic/Logic Unit), die – wie die Transform & Lighting Einheit, die dadurch ersetzt wird – 3D-Punkte und Koordinaten verarbeitet. Dazu gehört ein Programmspeicher nebst diversen Registern für Konstanten und Zwischenergebnisse und ein bestimmter Befehlssatz (reicht über die T&L-Funktionen hinaus). So lassen sich die 3D-Punkte leicht im Raum manipulieren, um etwa Wellen über Oberflächen zu schicken oder Fahnen flattern zu lassen (**Procedural Deformation**), um das Falten und Ausbeulen von Stoff oder Haut an den Gelenkstellen einer Spielfigur (**Matrix Skinning**) oder im Gesicht eines entsprechenden Kopfes (**Facial Animation durch Keyframe Interpolation**).

Die Abbildung 2.4 auf Seite 11 zeigt die Funktionsweise dessen.

Neben den Raum-Koordinaten führt jeder der 3D-Punkte mit sich noch mindestens eine Farbe, die Koordinaten einer oder mehrerer Texturen sowie die Normale für die Beleuchtungsberechnung. Aus den Input Registern

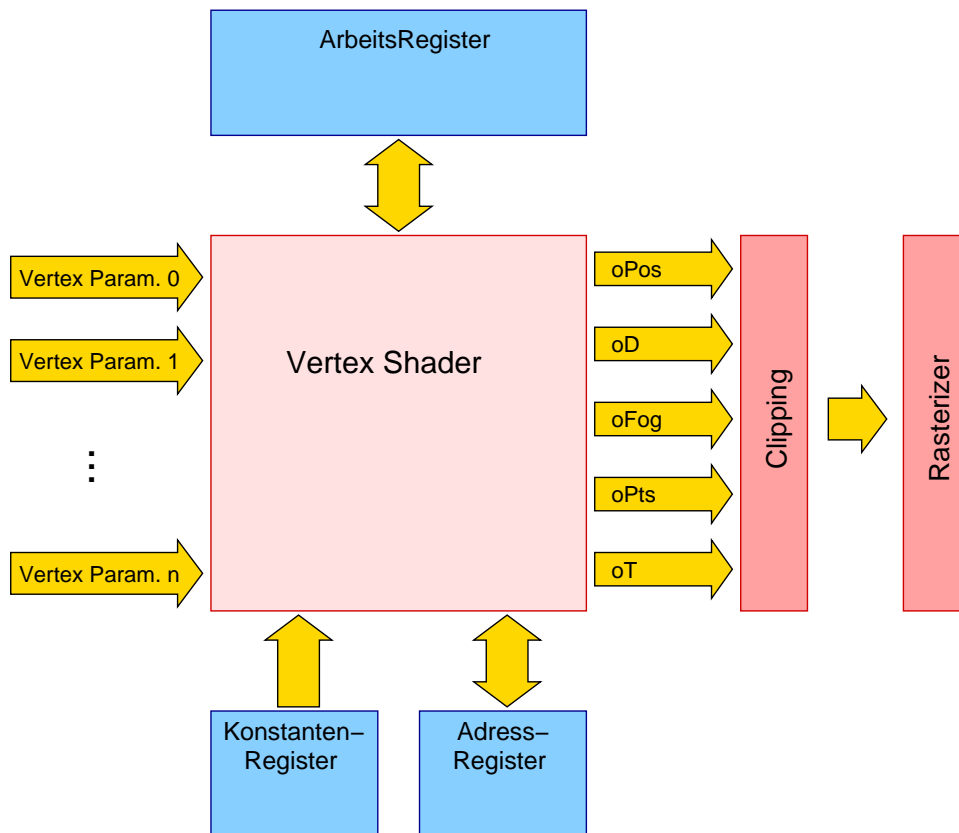


Abbildung 2.4: Der Vertex Shader

v0 bis v15 können insgesamt 16 Komponente verarbeitet werden. Die Befehle eines Vertex-Shader-Programms können auf jedes dieser Register zugreifen. Vor dem Start eines Programms können mindestens 96 Konstanten in Register geladen werden (wie die Skalierungsfaktoren für eine Ellipsoidenverformung). Für Zwischenergebnisse steht ein weiterer Satz Register zur Verfügung, die sich auch über Adressregistern indirekt adressieren lassen. Die Manipulation geht über die räumliche Änderung eines 3D-Punktes hinaus; die Farbe, die Texturkoordinaten eines Punktes können auch manipuliert werden. Am Schluß gelangen die Ergebnisse in spezielle Register für Farbe (oDn), Position (oPn), Texturkoordinaten (oTn), Nebel (oFog) und Partikelgröße – Point Sprites (oPts).

Kapitel 3

Beispiele

Die neue Version von Microsofts DirectX API - DirectX 9 - erlaubt durch die neue Floating Point Genauigkeit präzisere Effekte. Die Recheneinheiten in denen die Adress- und Farboperationen auf Gleitkomma-Hardware laufen, erfordern viele Transistoren, was ein Überschreiten der 100 Millionen Transistoren Grenze erzwingen. Im Vergleich zu DirectX 8 wurde die maximale Länge der Vertex Shader Programme auf 256 Befehle verdoppelt. Loops und Schleifen innerhalb der Shaderprogramme sind nun möglich. In dem Pixel Shader ist der Zugriff auf maximal 16 Texturen vorgesehen und der Befehlsatz auf das vierfache ausgebaut.

Eine Konkretisierung erfolgt mit den neuen Graphikchips von Ati und Nvidia. Die Tabelle 3.1 auf der Seite 15 beinhaltet die wichtigsten technischen Details der ATI Radeon 9700 (R300) und Nvidia GeForceFX (NV30) Graphikchips.

3.1 Ati Radeon 9700 PRO

Die Radeon 9700, mit dem R300-Chip, erfüllte als erste Karte am Markt die neuen DirectX 9 Spezifikationen von Microsoft. Der in 0.15 Micron Prozess erstellte R300-Chip besteht aus ca. 100 Millionen Transistoren. Er besitzt 4 Vertex Shader und 8 Pixel Pipelines. Pro Pipeline kann der R300 1 Textur pro Takt berechnen. Die Breite des DDR-Busses beträgt 256 Bit. Der Chip ist auch für den schnelleren DDR2 Speicher vorbereitet.

Zur Schonung der Speicherressourcen hat ATI in den Chip Hyper Z III integriert. Es setzt sich aus den Komponenten "Hierarchical-Z", "Z-Compression" und "Fast-Z-Clear" zusammen. Hierarchical Z unterteilt den Z-Buffer-Speicher, in dem die Tiefeninformationen der einzelnen Pixel vorliegen, in Blöcke und untersucht deren Sichtbarkeit im Bild. Durch das integrierte Multi-Sampling Anti-Aliasing bekommt Z Kompression einen höheren Stellenwert. Nach der Fertigstellung eines jeden Frame muß der dieser Szene zugeordnete Tiefenpuffer gelöscht werden, bevor er für Daten des nächsten Frame zur Verfügung steht.

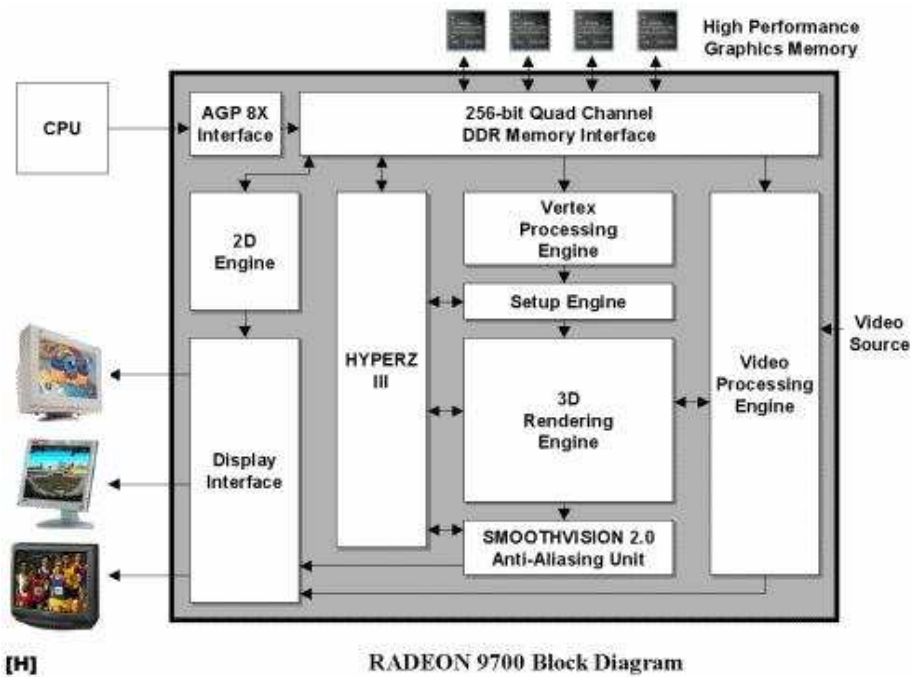


Abbildung 3.1: ATI Radeon 9700 Blockdiagramm

Der Chip besitzt 2 integrierte Ramdacs mit 400 MHz, die intern mit 10 Bit Genauigkeit arbeiten. Somit ist ein echtes Multimonitoring möglich. Es gibt keine fest implementierte Video-Optimierung. Die Pixel Shader erledigen die Filterung des Videostreams; somit sind auch Echtzeit-Bildeffekte und Optimierungen des Videostreams möglich. Praktische Anwendung findet sich bei Internet-Videostreams, bei welchen durch die hohe Kompression und niedrige Auflösung Artefakte in Form von Blöcken entstehen. Diese werden glatt gerechnet. Die Abbildung 3.1 auf der Seite 13 zeigt das Blockschaltbild des Chips.

3.2 Nvidia GeForceFX

Nvidia geht über die DirectX 9 Spezifikation hinaus und veröffentlicht den Cg Compiler (in Zusammenarbeit mit Microsoft entwickelt), denn der GeForceFX Chip kann längere und komplexere Shader Programme ausführen. Cg (C for graphics) ist an die Hochsprache C angelehnt und kompatibel zu der in DirectX 9 verfügbaren High Level Shading Language (HLSL). Die GeForceFX-GPU (NV30) ist der erste Consumer Graphikchip, der im 0.13 Micron Fertigungsprozess hergestellt wurde. Das erlaubt den im Flip-Chip Design gefertigten Chip mit bis zu 500MHz zu takten. Wegen dem großen Stromverbrauch ist - wie bei

ATI's R300 Chip (Radeon 9700) - eine zusätzliche Stromversorgung nötig. Der Chip erfordert wegen der großen Wärmeentwicklung eine aufwendige Kühlung. Er arbeitet auf einer aufwendigen 12-Layer Platine. Der GeForceFX Chip wurde für die neuen DDR2 Speichermodule ausgelegt. Bei DDR2 werden, wie bei DDR Speicherbausteinen, Daten auf beide Flanken des Signals übertragen. Der Unterschied liegt im Aufbau der Speicherzelle: statt in Burst von 2, überträgt DDR2 intern in Bursts von 4. Somit kann man den Speicher mit höheren Taktraten betreiben, denn die Taktrate innerhalb des Speicherbausteins gegenüber DDR halbiert wurde. Die Speicherbandbreite berechnet sich bei Nvidia's Chip genauso wie bei herkömmlichen DDR Speicher:

$$128 \text{ Bit} / 8 \text{ Bit/Byte} * 500 \text{ MHz} * 2 \text{ (2 Transfers/Takt)} = 16 \text{ GB/s}$$

Um den Nachteil bei der Speicherbandbreite auszugleichen, hat Nvidia den Chip zusätzlich zu "Z-Compression" mit "Color Compression" ausgestattet. Das erlaubt eine verlustfreie Kompression von Farbdaten in Echtzeit mit einem Faktor von bis zu 4:1.

	ATI Radeon 9700 PRO	NVIDIA GeForceFX
Chiptechnologie	256 Bit	256 Bit
Prozess	0.15 Micron	0.13 Micron
Transistoren	ca. 100 Mio.	125 Mio.
Speicherbus	256 Bit DDR	128 Bit DDR2
Speicherbandbreite	19.8 GB/s	16 GB/s
Pixel Füllrate	2.6 Gigapixel/s	4Gigapixel/s
AGP-Busbandbreite	1x/2x/4x/8x	1x/2x/4x/8x
Speicherbestückung	128/256 MB	128/256 MB
Chiptakt	325 MHz	500 MHz
Speichertakt	310 MHz (620 DDR)	250 MHz (1000 DDR)
Gehäuse / Zugriffszeit	BGA 2.9 ns	BGA 2.0 ns
Vertex Shader	4	FP Array
Vertex Shader Version	2	2.0+
Pixel Pipelines	8	8
Textureinheit pro Pipe	1	1
Texturen pro Textureinheit	8	16
Pixel Shader Version	2	2.0+
DirectX Generation	9	9.0+
FSAA-Modi	MultiSampling	MultiSampling
Max. FSAA Mode	6x	8x
Speicheroptimierung	Hyper Z III	LMA II Optimized Color Compression
Display Ausgänge	2	2
interne Chip-Ramdacs	2 x 400 MHz	2 x 400 MHz
Bits pro Color Channel	10	10
Special	TV Encoding On-Chip Extended Programmability Adaptive Filtering	TV Encoding On-Chip FullStream Adaptive Filtering

Tabelle 3.1: Technische Eigenschaften der Ati Radeon 9700 und Nvidia GeForceFX Chips

Kapitel 4

Abschluss

Ein optimales Ergebnis wird durch die Abstimmung des Zusammenspiels aller Einheiten erreicht. Dabei dürfen keine Datenstaus entstehen und mit jedem Takt sollte ein Pixel berechnet werden. Es bedarf jedoch bei weitem mehr Anstrengungen, um ein Graphikchip zu entwickeln, der Ergebnisse wie in Abbildung 4.1 auf der Seite 16 liefert.

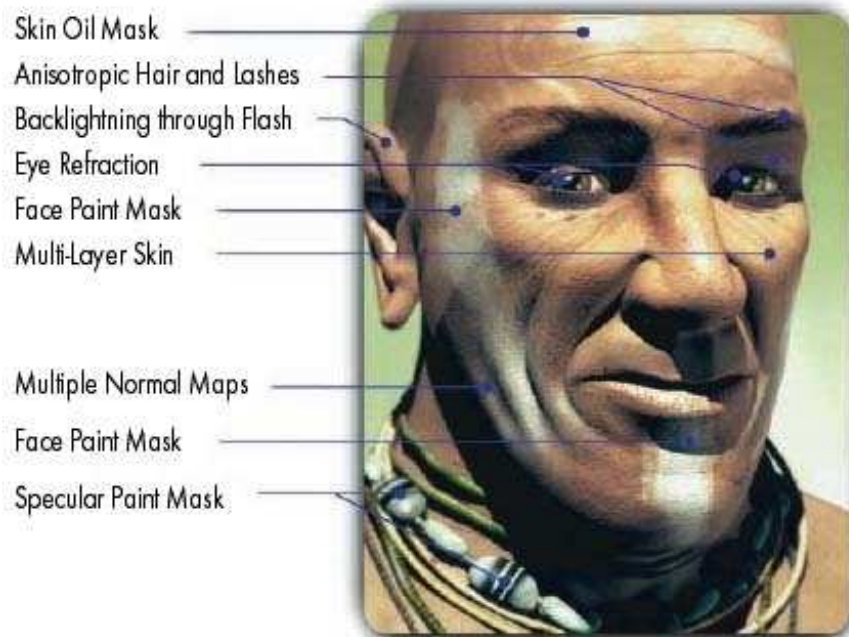


Abbildung 4.1: Ergebnis der Kombination mehrerer Algorithmen (Quelle: c't)

Dazu ist die Arbeit der Softwareentwickler noch einzubeziehen, die durch die Entwicklung der Algorithmen zur realitätsnahen Darstellung der 3D-Szenarien das Grundgerüst liefern.

Mit dem Schritt von DirectX 7 nach DirectX 8 wurde eine wichtige Entscheidung getroffen. Der Abschied von dem starren Aufbau der Transform & Lighting Einheit und das Verlagern der Programmierbarkeit auf die Spieleentwickler verlagerte auch die Gespräche und Kritiken in diese Richtung.

Die Pointierung der 3D Graphikhardware auf die DirectX API dient nur dem besseren Verständnis im Bezug auf die Einstufung und Sortierung der Eigenschaften. An dieser Stelle ist OpenGL gleichwertig.

Literaturverzeichnis

- [Bert00a] Manfred Bertuch. *c't 8/2000 — Polygonfabriken. Wie Grafikchips Bits und Bytes in plastische Bilder verwandeln.* Verlag Heinz Heise GmbH & Co KG, Hannover, 2000
- [Bert00b] Manfred Bertuch. *c't 24/2000 — Pixelzauber. Die Grafiktricks der 3D-Chips und ihr Einsatz in Spielen.* Verlag Heinz Heise GmbH & Co KG, Hannover, 2000
- [Bert02] Manfred Bertuch. *c't 15/2002 — Aufklärung in 3D. Was Sie schon immer über Grafikchips wissen wollten.* Verlag Heinz Heise GmbH & Co KG, Hannover, 2002
- [DP3BM] Whitepaper. *Dot Product3 Bump Mapping.* <http://www.nvidia.com/Marketing/Developer/DevRel.nsf/WhitepapersFrame?OpenPage>
- [Enca96] J.Encarnação/W.Straßer/R.Klein. *Graphische Datenverarbeitung I.* R.Oldenbourg Verlag GmbH , München, ISBN 3-486-23223-1, 1996
- [Enca97] J.Encarnação/W.Straßer/R.Klein. *Graphische Datenverarbeitung II.* R.Oldenbourg Verlag GmbH , München, ISBN 3-486-23469-2, 1997
- [nvidia] <http://www.nvidia.com>
- [ati] <http://www.ati.com>
- [matrox] <http://www.matrox.com>
- [matrox] <http://www.tomshardware.com>
- [matrox] <http://www.heise.de>