# DDS-Texturen

## Inhaltsverzeichnis


- 1 DDS-Unterformate
  - 1.1 DXT1
  - 1.2 DXT3
  - 1.3 DXT5
  - 1.4 3Dc (für Normal Maps)
- 2 Mipmaps
- 3 Optimierungen für DDS
  - 3.1 Die 4x4-Texel-Blöcke
  - 3.2 Wahl des Formates
- 4 Kompressions-Tools
- 5 Weiterführende Quellen


# 1 DDS-Unterformate

Die wichtigsten Kompressionsvarianten:

| | | | |
|---|---|---|---|
| DXT1 | RGB | 5-6-5 | kein Alpha-Kanal |
| DXT1 (a) | RGBA | 5-6-5 | 1-Bit Alpha |
| DXT3 | RGBA | 5-6-5 | 4-Bit Alpha |
| DXT5 | RGBA | 5-6-5 | interpoliertes Alpha |
| DXT5_NM | XY | 0-6-0-A | X wird nach G und Y nach Alpha verschoben |
| 3Dc | XY | | Für Normal Maps |

## 1.1 DXT1

Die Textur wird in 4x4 Pixel große Blöcke eingeteilt; sogenannte Texel. Normalerweise würden diese 16x32 (=512) Bit belegen. Bei 24bit Texturen werden intern i.d.R. 8 leere Bits aufgefüllt. Ein Texelblock besteht aus folgenden Informationen: 2x16 Bit für 2 Farben (RGB565: Rot 5 Bit, Grün 6 Bit, Blau 5 Bit) und dann aus einem Abschnitt mit je 2 Bit für jedes Pixel: Also 16x2 Bit. Diese 2 Bit repräsentieren eine Farbmatrix aus den 2 16bit-Farben. (Diese werden intern wieder auf 24 Bit hochgerechnet. Die Zwischenfarben sind also nicht 16 Bit).

Wird kein Alpha-Kanal gespeichert gilt:

| **Bitmuster** | **Bedeutung** |
|---|---|
| 00 | Farbe1 |
| 01 | 2/3 Farbe1 + 1/3 Farbe2 |
| 10 | 1/3 Farbe1 + 2/3 Farbe2 |

| | |
|---|---|
| 11 | Farbe2 |

Bei DXT1 mit Alpha spielt zusätzlich der Wert von Farbe 1 und 2 eine Rolle: Hat Farbe2 einen höheren Wert gilt folgende Tabelle:

| Bitmuster | Bedeutung |
|---|---|
| 00 | Farbe1 |
| 01 | 1/2 Farbe1 + 1/2 Farbe2 |
| 10 | **Transparent/Schwarz** |
| 11 | Farbe2 |

## 1.2 DXT3

Hier stehen 128 Bit für jedes 4x4 Pixel große Texel zur Verfügung. Zunächst wird in den ersten 64 Bit der Alpha-Kanal kodiert (4 Bit pro Pixel, d.h. in 16 Abstufungen pro Pixel). Der 2. Block mit 64 Bit entspricht DXT1 ohne Alpha-Kanal.
DXT3 ist sinnvoll bei Alpha-Kanälen mit scharfen Abgrenzungen. Weiche Übergänge sind eher schlecht darstellbar.

## 1.3 DXT5

Stellt ebenfalls 128 Bit zur Verfügung. Nur der Alpha-Kanal wird hier interpoliert. Die ersten 64 Bit stehen wieder für den Alpha-Kanal zur Verfügung. Es werden zwei 8 Bit Werte für Farben gespeichert gefolgt von einem Block mit 16x3 Bit für Werte ähnlich zusammengesetzt wie die Tabelle für DXT1:

Ist Farbe 1 größer im Wert als Farbe 2 gibt es folgende Abstufungen:

| Bitmuster | Bedeutung |
|---|---|
| 000 | Farbe1 |
| 001 | 6/7 Farbe1 + 1/7 Farbe2 |
| 010 | 5/7 Farbe1 + 2/7 Farbe2 |
| 011 | 4/7 Farbe1 + 3/7 Farbe2 |
| 100 | 3/7 Farbe1 + 4/7 Farbe2 |
| 101 | 2/7 Farbe1 + 5/7 Farbe2 |
| 110 | 1/7 Farbe1 + 6/7 Farbe2 |
| 111 | Farbe2 |

Ist Farbe 2 größer, gilt folgendes:

| Bitmuster | Bedeutung |
|---|---|
| 000 | Farbe1 |
| 001 | 2/5 Farbe1 + 3/5 Farbe2 |
| 010 | 3/5 Farbe1 + 2/5 Farbe2 |
| 011 | 4/5 Farbe1 + 1/5 Farbe2 |
| 101 | **0** |
| 110 | **255** |
| 111 | Farbe2 |

DXT5 ist also für feinere Abstufungen, wie Rundungen etc sehr gut. Ist aber im Vergleich zu DXT3 manchmal unschärfer bzw. kann Blockartefakte liefern. Hier gilt: einfach ausprobieren, was nach der Komprimierung besser aussieht.

**Die Kompressionsqualität ist also in erster Linie vom Kompressionstool abhängig: Von der Bestimmung der 2 Farben sowie deren Anordnung.**

## 1.4 3Dc (für Normal Maps)

Normal Maps stellen bei der Kompression eine besondere Herausforderung dar. Der klassische DXT-Kompressionsalgorithmus kann bei Normal Maps deutliche Kompressionsartefakte erzeugen. Dadurch können die gespeicherten Normalen verfälscht werden, was sich im Spiel durch diffuse oder sichtbar zerstörte Oberflächen und eine fehlerhafte Beleuchtung bemerkbar macht.

Um dieses Problem zu reduzieren, wurde von ATI eine speziell für solche Daten geeignete Kompression entwickelt:

Eine Normal Map enthält für jeden Pixel einen Normalenvektor mit den drei Komponenten **X, Y und Z**. Dieser Vektor beschreibt die Ausrichtung der Oberfläche und ermöglicht es der Engine, die Beleuchtung so zu berechnen, als wären wesentlich mehr geometrische Details vorhanden. Dadurch können beispielsweise auf einer ansonsten glatten Oberfläche kleine Unebenheiten und Strukturen simuliert werden.

Beim **3Dc-Format** werden nur X und Y des Vektors gespeichert. Z kann rekonstruiert werden, da bei Normal Maps die Länge des Vektors immer 1 beträgt. Dadurch muss nur ein Teil des Vektors gespeichert werden. Gleichzeitig kann die verfügbare Datenmenge wesentlich gezielter für die beiden relevanten Komponenten verwendet werden, wodurch 3Dc für Normal Maps deutlich besser geeignet ist als eine gewöhnliche DXT-Kompression.

Leider gibt es zur Verwirrung aller zwei verschiedene Varianten von 3Dc. Der einzige Unterschied liegt in der Reihenfolge der beiden Orientierungen X und Y. Die ursprüngliche normale Variante hat die FourCc `ATI2` (FourCc ist eine 4 Zeichen lange Kennung). Dieser kann beispielsweise mit einem Hex-Editor in einer entsprechenden DDS-Datei eingesehen werden.

Bei der anderen Variante sind die X- und Y-Kanäle vertauscht. Hier findet sich zusätzlich die Kennung `A2XY`. Diese Kennung gehört allerdings nicht zum eigentlichen FourCC. Dadurch können Train Fever und viele ältere Tools die beiden Varianten nicht zuverlässig voneinander unterscheiden.

Wird eine Normal Map mit vertauschten Kanälen als normale ATI2-Normal Map interpretiert, sind die Normalen im Spiel entsprechend falsch und die Beleuchtung der Oberfläche wirkt fehlerhaft.

Leider unterstützen auch nur wenige Tools diese Format. (Z.B. die Standalone Version des NVIDIA DDS-Kompressors sowie Intels Texture Works Plugin für Photoshop. NVIDIAs Photoshop Plugin unterstützt nur `ATI2`)

3Dc `ATI2A2XY` wird von TransportFever als Standard für die Normal Maps genutzt. **Um diese Kompression nutzen zu können muß eines der neuen `PHYSICAL_` Materialien verwendet werden.**

# 2 Mipmaps

**Transport Fever benötigt in DDS-Dateien Mipmaps für Texturen**. Mipmaps sind verkleinerte Versionen einer Textur, die automatisch erzeugt und zusammen mit der eigentlichen Textur gespeichert werden. Je weiter ein Objekt von der Kamera entfernt ist, desto kleinere Versionen der Textur können verwendet werden. Dadurch muss die Grafikkarte bei entfernten Objekten nicht mehr die vollständige, hochauflösende Textur verarbeiten. Das reduziert den Speicher- und Rechenaufwand und sorgt gleichzeitig für ein ruhigeres und saubereres Bild, da stark verkleinerte Texturen weniger flimmern und Aliasing erzeugen.

Eine Textur mit einer Auflösung von beispielsweise 1024×1024 Pixeln enthält dabei typischerweise zusätzlich Mipmap-Stufen von 512×512, 256×256, 128×128 usw., bis hinunter zu 1×1 Pixel.

Beim Erstellen von DDS-Texturen sollte daher darauf geachtet werden, dass die vollständige Mipmap-Kette vorhanden ist. Fehlen die Mipmaps, stürzt das Spiel einfach ab.

# 3 Optimierungen für DDS

Das Wissen über die Funktionsweise von DXT kann bei der Optimierung von Texturen direkt genutzt werden. Sind bestimmte Bereiche einer Textur besonders stark verschmiert oder ist beispielsweise eine Aufschrift nicht mehr sauber genug erkennbar, können einige Eigenschaften des Formats ausgenutzt werden:

## 3.1 Die 4x4-Texel-Blöcke

DXT komprimiert Texturen in **4×4-Texel-Blöcken**. Dadurch kann bereits eine geringfügige Verschiebung eines Details das Kompressionsergebnis verändern. Bei einer Aufschrift oder einer besonders scharfen Kante kann es sich daher lohnen, das Detail um **ein bis zwei Pixel** zu verschieben. Je nachdem, wie die Kante anschließend innerhalb der 4×4-Blöcke liegt, kann die Kompression deutlich besser oder schlechter ausfallen.

Auch die Bereiche außerhalb des eigentlichen UV-Bereichs spielen eine Rolle. Ein ausreichendes **Padding von mindestens 8 Pixeln** ist empfehlenswert. Dabei sollten die Randbereiche möglichst mit den angrenzenden Farben bzw. Strukturen aufgefüllt werden. Besonders an UV-Nähten sollte darauf geachtet werden, dass dort möglichst ähnliche Farben vorhanden sind.

Diese zusätzlichen Pixel sind nicht völlig irrelevant: Sie werden ebenfalls in 4×4-Blöcken komprimiert und beeinflussen damit unter Umständen die für einen Block verwendeten Referenzfarben.

## 3.2 Wahl des Formates

**DXT1** hat das beste Kompressionsverhältnis (für 24 Bit-Texturen 6:1, 32 Bit 8:1), kann aber nur ja und nein im Alpha-Kanal liefern. Für Texturen mit Alpha-Kanal also nur dann zu benutzen, wenn eine 100% Transparenz einiger Pixel erwünscht ist. Für Texturen ohne Alpha (Schmutz/Rost/CBlend, METAL/GLOSS/AO etc.) sicherlich die beste Wahl, da am effektivsten komprimiert wird.

Für Texturen mit Alpha-Kanälen ist abzuwägen, ob **DXT3 oder DXT5** besser passt. Beide komprimieren gleich stark: 4:1. Generell sollte man jetzt auch gründlich abwägen, ob es vllt sinnvoll ist, nur eine kleine Fenster/Glas-Textur (z.B. 16x16 Pixel) separat zu erstellen und diese in ein eigenes Material auszulagern, da dann der Rest mit DXT1 komprimiert werden kann. Dies erfordert dann natürlich wieder ein detaillierteres Mesh, was natürlich auch wieder auf die Performance drücken kann. Ihr habt es jetzt in der Hand.

Für Normal-Maps ist **3Dc** die beste Wahl. Verwendet man nicht die neuen `PHYSICAL_` Materialien, sollte man entweder DXT1 nutzen oder TGA verwenden. DXT5_NM wird von Transport Fever nicht unterstützt.

# 4 Kompressions-Tools

| Tool | Software | 3Dc - Variante | |
|---|---|---|---|
| NVIDIA Texture Tools | Photoshop Plugin<br>Paint Shop Pro Plugin | ATI2 | Download |
| Intel Texture Works | Photoshop Plugin | ATI2A2XY | Download |
| Gimp DDS | Gimp Plugin | ATI2 | Download |
| NVIDIA Texture Tools (NVTT) | Standalone | ATI2A2XY | Download |
| AMD Compress | Standalone / GUI | ATI2<br>ATI2A2X | Download |

Achtung: Tools nur mit dem Eintrag ATI2 und ohne ATI2A2XY komprimieren eine nicht passende Kanalbelegung! Dort müssten dann per Hand die R und G Kanäle getauscht werden.

# 5 Weiterführende Quellen

http://alt.3dcenter.org/artikel/3dc/index5.php
http://www.nvidia.com/object/r…-map-dxt-compression.html
https://en.wikipedia.org/wiki/S3_Texture_Compression