



Technical Brief

NVIDIA Unified Driver Architecture

Der elegante Ansatz für
niedrigere TCO

NVIDIA Unified Driver Architecture: Das effiziente Bindeglied zwischen Hardware and Software

NVIDIAs vereinheitlichte Treiberarchitektur, die Unified Driver Architecture (UDA), dient als Grundlage für den preisgekrönten NVIDIA®-Grafiktreiber für NVIDIA-Grafikprozessoren (GPUs) sowie die Treiber für NVIDIA nForce-Plattformprozessoren. Ein einzigartiges Konzept und die ständige Konzentration auf stabile, leistungsfähige Software haben UDA zur einzigen Treiberlösung mit gleichzeitiger Abwärts- und Aufwärtskompatibilität gemacht. Ein einziger Treiber unterstützt alle aktuellen und vergangenen Versionen der jeweiligen Hardware. Kommt eine neue Version eines NVIDIA-Prozessors auf den Markt, so vereinfacht der vorhandene NVIDIA-Treiber die Integration der neuen Hardware bzw. des neuen Systems in heterogene Umgebungen aus vielen verschiedenen Desktop- und Betriebssystemkonfigurationen. Dabei geht die NVIDIA UDA keinerlei Leistungskompromisse für alte oder neue Grafik- und Plattformhardware ein – vielmehr reduziert sie überflüssigen Software-Aufwand für die Treiberfunktionalität.

Auf der ganzen Welt nennen OEMs und Systemhersteller immer wieder den Erfolg der NVIDIA UDA als entscheidenden Faktor, wenn es um die Auswahl von Grafik- und Plattformlösungen für neue Produkte geht. UDA ermöglicht allen Endanwendern flexible, einfache Upgrades bei der Verfügbarkeit neuer Treiber. Die hervorragende Abwärts- und Aufwärtskompatibilität bedeutet gleichzeitig beträchtliche Einsparungen bei der TCO (Total Cost of Ownership), die der Hersteller an seine Kunden weitergeben kann – ganz gleich, ob es sich um Großunternehmen oder kleine und mittelständische Betriebe handelt. Zur Senkung der TCO tragen mehrere Merkmale der UDA-Technologie bei:

- ❑ **Niedrigerer Zeit- und Kostenaufwand für die Administration:** Es wird lediglich eine einzige NVIDIA-Treiberversion benötigt – also muss selbst beim Rollout großer Systemumgebungen nur ein einziges Software-Image verwaltet, konfiguriert und installiert werden.
- ❑ **Investitionssicherheit:** NVIDIA erweitert seine Treiber fortlaufend um neue Funktionen und Leistungsmerkmale. So kommen auch ältere Prozessoren in den Genuss zusätzlicher Funktionalität und höherer Leistung – ganz ohne Zusatzkosten und völlig transparent für den Endanwender und das EDV-Team.
- ❑ **Weniger Hardware-/Treiberkonflikte:** Jeder Treiber wird ausgiebig auf allen vorhergehenden Produkten getestet – darüber hinaus werden jedoch selbst neue Produkte mit alten Treibern getestet. Diese Tests führen zu stabilerem Systembetrieb über die gesamte Lebensdauer der Lösung hinweg.
- ❑ **Bessere Skalierbarkeit:** Produktupgrades und Hardware-Erweiterungen werden einfacher. IT-Manager können ihre Konfigurationen um Produkte aus dem gesamten NVIDIA-Angebot erweitern, ohne dass andere Treiber nötig wären.

- **Überlegene Mehrplattform-Unterstützung:** 90% des NVIDIA-Treibercodes ist betriebssystemunabhängig. So kann NVIDIA für alle Zentralprozessoren (CPUs) und Betriebssysteme leistungsfähige, grundsolide Treiber mit voller Funktionalität anbieten.

Dieser Artikel bietet einen Überblick über traditionelle Treiberarchitekturen und erläutert deren Probleme in technischer und wirtschaftlicher Hinsicht. Als Ansatz zur Überwindung dieser Hindernisse wird schließlich die NVIDIA UDA-Technologie vorgestellt.

Die Herausforderung

Zur Integration von Hardware mit dem Betriebssystem (BS) werden Treiber traditionell anhand eines Schichtenmodells entwickelt. Die Hardware-Abstraktionsschicht (Hardware Abstraction Layer, HAL) umfasst hierbei den Code, der zur direkten Ansteuerung der Hardware dient. Das “Rückgrat” des Treibers besteht aus Betriebssystem-Abhängigkeiten. Dieser Teil wird auch als “gemeinsamer” Abschnitt bezeichnet, da die Treiber für die verschiedenen Hardwareprodukte diesen Code jeweils ganz oder teilweise benötigen. (Siehe Abb. 1.)

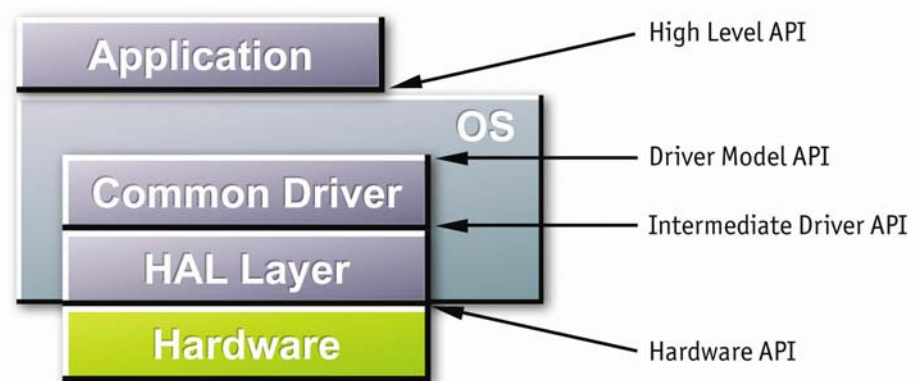


Abb. 1: Traditionelles Schichtenmodell für Treiber

Technische Probleme

Das traditionelle Treiber-Schichtenmodell ist zwar relativ unkompliziert – mit wachsender Komplexität der Systeme und Rechnerumgebungen hat es jedoch auch zu einigen Problemen geführt. Systemtreiber werden von drei externen Faktoren der Rechnerumgebung beeinflusst:

- ❑ **Betriebssystemversionen** wie zum Beispiel die verschiedenen Microsoft® Windows®-Betriebsumgebungen (Windows NT®, Windows 2000, Windows XP, Windows 9x), Linux und Apple® Mac OS X
- ❑ **Anwendungsprogrammierschnittstellen (APIs)** wie zum Beispiel GDI, VPE, WDM, OpenGL® und Microsoft DirectX®
- ❑ **Hardwaregeräte**, die von den vielen verschiedenen Grafik- und Plattformlösungen von NVIDIA unterstützt werden und wurden – gestern, heute und morgen

Wenn ein einziger, allumfassender Treiber alle möglichen Kombinationen in allen drei Bereichen unterstützen müsste, würde er sehr schnell sehr unhandlich. Zudem wäre seine Leistung äußerst unterdurchschnittlich. Infolgedessen gibt es in der traditionellen Treiberarchitektur immer eine Vielzahl verschiedener Treiber, die jeweils eine „Untermenge“ der Betriebssysteme, APIs und Hardwaregeräte abdecken.

Der Einsatz traditioneller Treiberarchitekturen und Programmiermethoden zieht eine Vielzahl von technischen Problemen mit sich:

- ❑ **Längere Entwicklungsdauer**
 Wenn neue oder verbesserte Betriebssysteme, APIs oder Hardwarelösungen unterstützt werden sollen, müssen immer alle Treiberversionen geändert werden. Die Vielzahl der möglichen Kombinationen geht einher mit langwierigen Test- und Zertifizierungszyklen.
- ❑ **Höhere Supportkosten**
 Wenn für unterschiedliche Betriebssysteme, APIs und Hardwarelösungen verschiedene Treiber benötigt werden, bedeutet dies für eine EDV-Abteilung konkret, dass es länger dauert, bis eine stabile Hardware-Software-Kombination gefunden ist.
- ❑ **Leistungseinbußen**
 Beim Einsatz mehrerer Treiber wird es praktisch unmöglich, für jeden denkbaren Ausführungspfad die optimale Leistung zu erreichen. Wird ein einziger „Riesentreiber“ eingesetzt, sinkt die Leistung sogar noch weiter.
- ❑ **Schlechte Skalierbarkeit**
 Konventionelle Programmieransätze bieten nur schlechte Skalierbarkeit, wenn Hardware und Software komplexer werden. Bei „Riesentreibern“ steigt die Komplexität exponentiell – es wird unmöglich, gleichzeitig hohe Qualität und hohe Leistung zu erreichen.

Wirtschaftliche Probleme

Unterzieht man die traditionelle Treiberarchitektur einer Wirtschaftlichkeitsanalyse, so treten weitere Probleme ans Tageslicht:

- ❑ **Software-Entwicklungskosten**
 Im traditionellen Treibermodell wird ein unverhältnismäßig großes Entwicklerteam benötigt, um die verschiedenen Treiber weiterzuentwickeln und auftretende Fehler zu beheben. Die damit entstehenden Kosten müssen an die Kunden weitergegeben werden.
- ❑ **Mühseliger Freigabeprozess**
 Die anfallenden QS- und QM-Aktivitäten sind komplex und zeitaufwändig. Verschlimmert wird die Situation noch dadurch, dass die Tests auf jeder Ebene der Lieferkette wiederholt werden müssen – beim Treiberentwickler, bei den OEMs und Systemherstellern und schließlich bei Großkunden, die alle Systeme vor der Bereitstellung zunächst zertifizieren.
- ❑ **Kostenintensive, langwierige Zertifizierungstests**
 Jede Kombination aus Treiber und Hardware muss einzeln zertifiziert werden – und jede Systemkonfiguration, in die dieser Treiber integriert wird, ebenso! Bei großen, „allumfassenden“ Treibern werden in der Regel mehrere Testdurchgänge und längere Zertifizierungszyklen fällig.

Die NVIDIA-Lösung

Die NVIDIA-Systemarchitekten wollten eine Treiberarchitektur entwickeln, die hohen Ansprüchen genügen sollte: Sie sollte die TCO für NVIDIA-Prozessoren möglichst niedrig halten, die maximal mögliche Leistung ausreizen und gleichzeitig unabhängig vom verwendeten Grafik- oder Plattformprozessor konkurrenzlos stabil und kompatibel sein. Ihre Lösung, die NVIDIA Unified Driver Architecture (siehe Abb. 2), erfüllt all diese Ansprüche und hat sich inzwischen als branchenführende Innovation etabliert.

Beim NVIDIA UDA-Ansatz konzentriert sich die NVIDIA-Treibersoftware nicht darauf, alle möglichen Hardwareunterschiede zu berücksichtigen; vielmehr implementiert sie die Funktionalität der Treiber-API. Die gesamte Hardware-Ansteuerung wird von einem klassenbasierten, objektorientierten Programmiermodell übernommen. Die leistungskritische Funktionalität (Funktionen zur Grafik- und Soundverarbeitung) nutzt eine hardwareseitig auf den NVIDIA-Chips implementierte HAL. Da diese Funktionalität hardwareseitig implementiert ist, reduziert sich der mit traditionellen Treibern einhergehende Mehraufwand auf ein Minimum. Diese Hardware-Software-Kombination bietet eine einzigartige Balance aus hoher Leistung und Qualität – ein Ergebnis, das eine rein softwarebasierte Lösung nicht bieten könnte.

Alle NVIDIA-Treiber bieten hervorragende Leistung bei gleichzeitiger Abwärts- und Aufwärtskompatibilität. Ein Treiber unterstützt alle Versionen der jeweiligen Hardware. Möglich wird dies durch einen stark abstrahierten, klassenbasierten Programmieransatz, der die Software von der Hardwareebene abkapselt.

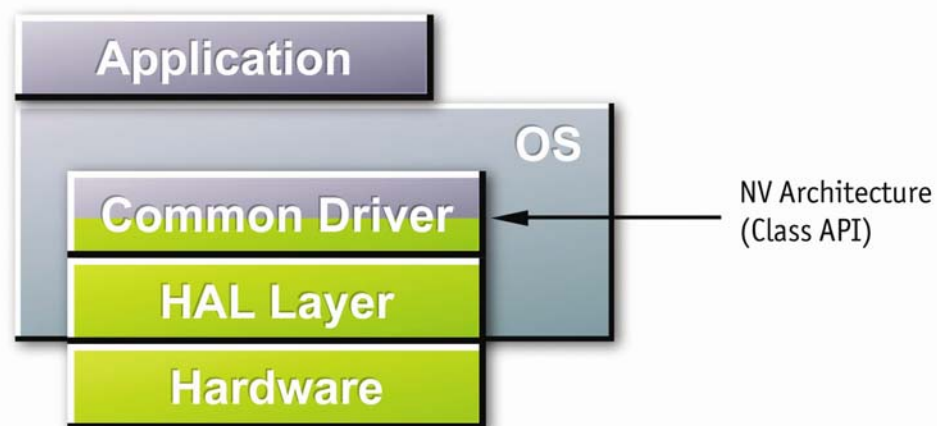


Abb. 2 NVIDIA-Treiber-Schichtenmodell für GPUs und leistungskritische Funktionalität

Objektorientierte Programmierung

Die NVIDIA UDA beinhaltet eine objektorientierte Abstraktion der Hardware-Funktionalität, die auch als klassenbasiertes Modell bezeichnet wird. *Objekte* und *Klassen* bieten Mechanismen, mit denen sich der Zugriff auf Funktionen und Kontextinformationen effizient steuern lässt. Indem die Engine-Funktionalität abstrahiert wird, bleibt die NVIDIA-Architektur von der Engine unabhängig, während gleichzeitig die Prozessor-, Speicher- und Busbelastung für Grafik- und Systemfunktionen sinkt.

Hinweis: Eine **Klasse** ist eine abstrakte Definition einer bestimmten Hardwarefunktion (die als Registerzuordnung implementiert wird). Ein **Objekt** ist eine einzelne Instanz einer Klasse (d. h. eine bestimmte Registerzuordnung) und enthält daher einen bestimmten Kontext. Eine **Methode** ist ein Objektregister, auf das nur schreibend zugegriffen werden kann.

Die NVIDIA UDA-Klassenstruktur und ihre *Methoden* bilden aktuelle API-Modelle und deren Verwendung nach. Beispielsweise stellt der NVIDIA-GPU-Treiber Klassen zur Verfügung, die so programmiert sind, dass sie exakt aktuellen Microsoft Windows-Programmierschnittstellen entsprechen.

Ressourcen-Manager

In der NVIDIA UDA-Hardwareschicht arbeitet der NVIDIA Ressourcen-Manager – eine intelligente Lösung zur Erkennung und Verwaltung von Klassen und Objekten (siehe Abb. 3). Die NVIDIA-Treibersoftware stellt auf Kernebene eine Funktionsbibliothek bereit, mit der Klassen und Objekte erkannt werden können. Diese NVIDIA-Innovation übernimmt die Verwaltung von Zuständen und Kontexten und teilt den verschiedenen Treiber-Clients je nach Bedarf Ressourcen der Architektur zu. Aufgrund dieses Kontext-Managements sind Clients, die den Chip gleichzeitig nutzen, füreinander unsichtbar.

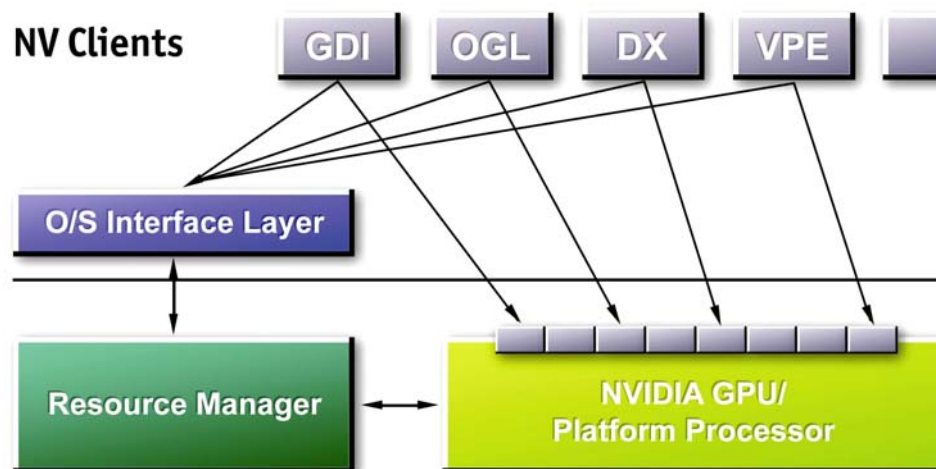


Abb. 3: Treiber-Architekturmodell mit Ressourcen-Manager

Die Zuteilung und Verwaltung von Architektur-*Objekten* (Hardwareressourcen) eröffnet verschiedene Möglichkeiten für das Kompatibilitäts-Management. Wie bereits beschrieben stellt der Ressourcen-Manager einen Kanal zwischen der Hardware und dem Treiber-Client her, der den treiberseitigen Zusatzaufwand für die Zusicherung von Hardwareressourcen auf ein Minimum reduziert.

Anhand der Erkennung festgelegter Klassen kann der Ressourcen-Manager auch Ausnahmesituationen verarbeiten. Unterschiede in der Hardware-Implementation bleiben für den Anwender unsichtbar und werden so gehandhabt, dass die Treiberkompatibilität über alle Plattformen hinweg gewährleistet bleibt. Darüber hinaus lässt der Ressourcen-Manager die Nutzung von Legacy-Schnittstellen zu, um in zukünftigen Hardwareversionen Abwärtskompatibilität zu erreichen. Der Anwendungsentwickler kann alle Hardwarefunktionen über diese Standardkonventionen ansprechen – Legacy-Code kann also ohne weitere Änderungen neue Hardware unterstützen.

Darüber hinaus unterstützt der Ressourcen-Manager folgende Funktionen:

❑ **Fehlermanagement:**

Die Klassenerkennungsfunktionen des Ressourcen-Managers sind ein hervorragender Mechanismus für die Implementierung von Fehlerkorrekturen.

❑ **Betriebssystemabhängige Architekturkomponenten**

Der Ressourcen-Manager kann zudem zur Verwaltung versionsabhängiger Unterschiede bei BS-Speicherverwaltung, Prozess-/Thread-Steuerung, Plug-n-Play, Energiemanagement und anderen Funktionen verwendet werden.

Nutzen von UDA

Die NVIDIA UDA hat sich über mehrere NVIDIA-Technologiegenerationen und eine Vielzahl von Produktversionen bewährt, beginnend mit der Einführung der NVIDIA RIVA128™ im Jahre 1998. Kunden und OEM-Partnern von NVIDIA bringt UDA laut eigener Aussage vielfältigen Nutzen:

❑ **Besseres Design und mehr integrierte Leistungsmerkmale**

Da sich weniger Entwickler mit Versionskontrolle und Release-Management beschäftigen müssen, können mehr Ressourcen darauf verwendet werden, die Leistung zu optimieren und neue Funktionen zu entwickeln. Ebenso ermöglicht der NVIDIA UDA-Ansatz einen flexibleren Austausch von Low-Level-Hardware ohne Auswirkungen für OEMs und Endanwender.

❑ **Hervorragende Leistung**

Die hardwareseitig implementierte Treiberfunktionalität ermöglicht optimale Performance und Leistungssteigerungen in allen Anwendungen.

❑ **Stabilität und Zuverlässigkeit**

Die NVIDIA UDA hat sich über mehrere Jahre hinweg bewährt und ist ein wesentlicher Faktor für NVIDIAs konkurrenzlose Stellung als Lieferant stabiler, zuverlässiger Hard- und Software.

❑ **Extrem kurze Entwicklungszyklen**

NVIDIA kann neue Produkte in Rekordzeit vorstellen: Zwischen Tape-Out (Planfreigabe) und Produktion liegen weniger als 100 Tage, während der Rest der Branche in der Regel mehr als ein halbes Jahr braucht. So erhalten OEMs schnellen Zugriff auf neue Technologien. Ebenso vereinfacht UDA die Installation und den Support neuer Versionen, da ein einziger gemeinsamer Treiber alte und neue Lösungen gleichzeitig unterstützt.

□ **Niedrigere TCO**

IT-Verantwortliche können ein einziges Software-Image mit einer einzigen NVIDIA-Treiberversion erstellen und damit Tausende von Systemen bereitstellen. So kann für verschiedene Hardwarekonfigurationen eine gemeinsame Softwarearchitektur festgelegt, verwaltet und unterstützt werden.

Schlussbemerkungen

NVIDIA hat mit UDA einen neuen Standard im Bereich der Treiberarchitekturen entwickelt. UDA bietet optimale Leistung, konkurrenzlose Kompatibilität und deutlich niedrigere TCO. Allerdings ist UDA nicht nur technisch gesehen eine elegante Lösung – alle Nutzer- und Kundengruppen profitieren von ihren Vorteilen:

- **Unternehmen** können dank weniger Aufwand für Support und Upgrades den Zeit- und Kostenaufwand für ihre EDV reduzieren.
- **IT-Verantwortliche** können mit ein und demselben Treiber verschiedene Hardwareprodukte unterstützen. Ein Upgrade auf neue Plattformen ist ohne Änderungen an den Treibern möglich. Der Rollout neuer Systeme geht schneller vonstatten, und flexible Hardwarekonfigurationen werden möglich.
- **Endanwender** kommen schneller in den Genuss neuer Produkte und können ihre vorhandenen Spiele und Anwendungen auch auf den neuen Plattformen weiter nutzen.
- **Software-Entwickler** können eine einzige Anwendungsversion schreiben, die automatisch für alle NVIDIA-Lösungen gleichzeitig optimiert ist.
- **Systemhersteller** können ihren Kunden die Möglichkeit bieten, mit einem einzigen Software-Image eine ganze Hardware-Produktlinie zu unterstützen. So lassen sich neue Plattformen einfacher in die installierte Basis integrieren, Zertifizierungs- und Qualifikationszyklen werden rationalisiert und die TCO für den Kunden sinkt insgesamt.

NVIDIA-Lösungen werden weiterhin auf UDA aufbauen, da sie eine rasche, verlässliche Bereitstellung hochwertiger, leistungsfähiger Softwaretreiber ermöglicht. UDA überwindet die Probleme herkömmlicher, althergebrachter Treiberarchitekturen und ermöglicht die schnelle Einführung von Hardware-Innovationen ohne Einbußen bei der Kompatibilität mit bestehenden Systemen und Anwendungen. Kein anderer Lieferant von Grafik- und Plattformprozessoren kann ein einziges gemeinsames Software-Image für seine ganze Produktpalette anbieten.

Rechtliche Hinweise

ALLE NVIDIA-DESIGNSPEZIFIKATIONEN, REFERENZPLATINEN, DATEIEN, ZEICHNUNGEN, DIAGNOSEPROGRAMME, LISTEN UND SONSTIGEN DOKUMENTE (EINZELN ODER IM GANZEN ALS "MATERIALIEN" BEZEICHNET) WERDEN „AS IS“ ("WIE BESEHEN") BEREITGESTELLT. NVIDIA GIBT HINSICHTLICH DER MATERIALIEN KEINERLEI GARANTIE, UNABHÄNGIG DAVON, OB DIESE AUSDRÜCKLICH, KONKLUDENT, GESETZLICH ODER ANDERWEITIG BEGRÜNDET SIND. INSBESONDERE WERDEN AUSDRÜCKLICH KEINERLEI GARANTIE HINSICHTLICH DER NICHTVERLETZUNG VON URHEBERRECHTEN, DER MARKTGÄNGIGKEIT SOWIE DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK ÜBERNOMMEN.

Die in diesem Artikel genannten Informationen sind nach bestem Wissen und Gewissen zutreffend und zuverlässig. Die NVIDIA Corporation übernimmt jedoch keinerlei Verantwortung für Konsequenzen, die aus der Nutzung dieser Informationen entstehen, bzw. für Patentrechtsverletzungen oder andere Verstöße gegen die Rechte Dritter, die aus einer solchen Nutzung entstehen. Es wird weder konkludent noch anderweitig eine Lizenz im Rahmen eines Patents oder eines Patentanspruchs der NVIDIA Corporation gewährt. Die in diesem Artikel genannten Spezifikationen können sich jederzeit ohne weitere Ankündigung ändern. Dieser Artikel löst alle eventuell vorab bereitgestellten Informationen ab und ersetzt diese. Ohne die ausdrückliche vorherige schriftliche Genehmigung der NVIDIA Corporation dürfen Produkte der NVIDIA Corporation nicht als missionskritische Komponenten in lebenserhaltenden Geräten oder Systemen eingesetzt werden.

NVIDIA und das NVIDIA-Logo sind eingetragene Warenzeichen und RIVA128 ist ein Warenzeichen der NVIDIA Corporation.

Bei anderen Firmen- und Produktnamen kann es sich um Warenzeichen der jeweils damit verbundenen Unternehmen handeln.

Copyright NVIDIA Corporation 2003



NVIDIA.
NVIDIA Corporation

www.nvidia.de