>> Anleitung für Hardware 3D-Beschleunigung

[Bitte Kapitel auswählen]

1. Einführung

1.1 Was ist 3D Hardware-Beschleunigung und wofür brauche ich es?

Mit 3D Hardware-Beschleunigung wird dreidimensionales Rendern vom Prozessor der Grafikkarte übernommen, anstatt wertvolle Rechenleistung der CPU für das Darstellen von 3D Bildern abzuzweigen. Dies wird auch als "Hardware-Beschleunigung" bezeichnet, im Gegensatz zur "Software Beschleunigung", wo die CPU mit Hilfe der Mesa Software Bibliotheken das Zeichnen übernimmt. Während XFree in der Regel 2D Hardware-Beschleunigung unterstützt, gibt es bei der 3D-Beschleunigung einige Lücken. Spiele, 3D-CAD und Mdoellierungen kommen in der Regel nicht ohne 3D Hardware-Beschleunigung aus.

1.2 Wie erhalte ich 3D Hardware-Beschleunigung?

In den meisten Fällen existieren sowohl binäre als auch Open-Source Treiber. Letzteresind zu bevorzugen, da wir nun einmal Linux verwenden und Open Source eines unserer Hauptprinzipien ist. In machen Fällen sind binäre Treiber oft die einzige Möglichkeit, wie zum Beispiel mit den nVidia Karten. Weitere binäre Treiber sind media-video/mgavideo für Matrox und media-video/ati-drivers für ATI Karten. Weitere Open-Source Projekte sind media-video/kyro-kernel für KyroII Karten und media-video/ati-gatos mit speziellen Augenmerk auf die Video-Möglichkeiten der ATI Grafikkarten.

1.3 Was ist DRI?

Die Direct Rendering Infrastructure (dri.sourceforge.net), besser bekannt unter dem Kürzel DRI ist eine effiziente Schnittstelle, die direkten und dennoch sicheren Zugang zur Grafikhardware ermöglicht. Sie beinhaltet Änderungen am X Server, verschiedenen Client-Bibliotheken und dem Kernel selbst. Hauptverwendeungszweck für DRI sind schnelle OpenGL Implementierungen.

1.4 Was ist XFree-DRM und was verbindet es mit dem normalen XFree86?

XFree-DRM ist eine **Erweiterung** zu XFree86, die 3D Hardware-Beschleunigung bereitstellt, die von XFree86 nicht unterstützt werden.

1.5 Zweck dieser Dokumentation?

Diese Dokumentation richtet sich an alle, die Direct Rendering in XFree zum Laufen bekommen wollen. XFree-DRM funktioniert für 3dfx, gamma, i8x0, matrox, rage128, radeon und sis Treiber. Die laufende Entwicklung für mach64 ist bereits im CVS Tree vorhanden -- weitere Infos hier und im HOWTO. Benutzer einer 2.4. Kernel Reihe benötigen das xfree-drm Paket, da der Direct Rendering Manager XFree 4.3 nicht unterstützt. Dagegen können 2.6. Kernel Nutzer dieses Dokument überspringen, da XFree 4.3 bereits unterstützt wird. Weitere Informationen erhält man auf der DRI Homepage. Weitere Vorschläge, Fragen oder Mails können an Donnie Berkholz gerichtet werden.

2. Installation von XFree86 und Kernel Konfiguration

2.1 Installation von XFree86

Befehlsauflistung 1: Installation von XFree86

```
# emerge x11-base/xfree
```

2.2 Kernel Konfiguration

Zunächst sollte der Chipsatz sondiert werden und die Unterstützung für diesen aktiviert werden

Befehlsauflistung 2: Prüfen des AGP Chipsatzes

```
[# emerge pciutils; lspci | grep AGP
# 00:01.0 PCI bridge: Intel Corp. 440BX/ZX/DX - 82443BX/ZX/DX AGP bridge (rev 03)
// Je nach entsprechender Hardware kann dies anders aussehen.
```

Dies sollte mit allen Kerneln funktionieren. In diesem Fall erfolgte die Konfiguration unter gentoo-sources-3.4.20-r5.

Befehlsauflistung 3

```
[# ls -l /usr/src/linux
lrwxrwxrwx    1 root     root           22 May 29 18:20 /usr/src/linux -> linux-2.4.20-gentoo-r5
// Stellen Sie sicher, dass /usr/src/linux auf Ihren aktuellen Kernel zeigt!
# cd /usr/src/linux
# make menuconfig
```

Befehlsauflistung 4: Optionen in make menuconfig

```
Processor type and features --->
  >*< MTRR (Memory Type Range Register) support
Character devices --->
  >*< /dev/agpgart (AGP Support)
  [*] Intel 440LX/BX/GX and I815/I820/I830M/I830MP/I840/I845/I850/I860 support
// Natürlich sollten Sie die Auswahl an Ihren Chipsatz anpassen!
  [ ] Direct Rendering Manager (XFree86 DRI support)
```

Stellen Sie sicher, dass die Unterstützung für DRM **abgeschaltet**ist. Das XFree-DRM Paket stellt seine eigene Kernel-Unterstützung bereit. Die Kernel-Version ist für XFree 4.2 gedacht.

2.3 Den Kernel kompilieren und installieren

Befehlsauflistung 5

```
# make dep && make clean bzImage modules modules_install
# mount /boot
# cp arch/i386/boot/bzImage /boot
```

Wenn der Kernel einen anderen Namen erhalten soll als bzImage, sollten Sie stattdessen nach /boot/neuer_name kopieren. Nicht vergessen grub.conf oder lilo.conf entsprechend anzupassen, und als LILO-Benutzer /sbin/lilo ausführen.

3. Installation von XFree-DRM und Konfiguration von Direct Rendering

3.1 Installation von XFree-DRM

Befehlsauflistung 6

```
# ACCEPT_KEYWORDS="~x86" emerge xfree-drm
```

3.2 Konfiguration von XF86Config

Öffnen Sie /etc/X11/XF86Config mit Ihrem Lieblingseditor und fügen Sie die Funktionen für DRI und GLX hinzu.

Befehlsauflistung 7: XF86Config

```
...
Section "Module"
     Load "dri"
     Load "glx"
     ...
EndSection
...
Section "Device"
     Driver "radeon"
     ...
EndSection
...
Section "DRI"
     Mode 0666
EndSection
```

Sollten Sie einen anderen Treiber verwenden, müssen Sie radeon durch den entsprechenden Treiber ersetzen

4. 3D Beschleunigung testen

4.1 Neustart mit dem neuen Kernel

Nachdem Sie ihren Computer neu gestartet haben, wollen wir schauen, ob Direct Rendering aktiviert wurde und wie gut es funktioniert.

Befehlsauflistung 8: Rendering testen

```
# startx
// Es ist nicht notwendig den Treiber oder APGART zu laden, auch wenn sie als Modul kompiliert w
//Sie werden automatisch geladen
# glxinfo | grep rendering
direct rendering: Yes
// Kommt hier "No", dann ist die Installation von Direct Rendering fehlgeschlagen
# glxgears
// Prüft die Framerate pro Sekunde (FPS) im Standardfenster. Dies sollte dann durchgeführt werde
```

5. Verwenden der CVS Sourcen

5.1 Wird CVS benötigt?

Probieren Sie zuerst das xfree-drm Paket aus. Sollte es nicht funktionieren, und Sie haben eine sehr neue Grafikkarte, dann sind die CVS Sourcen ein Versuch wert. Das xfree-drm Paket unterstützt auch dir Radeon 9000 Serie.

5.2 Wird Ihre Karte von den CVS Sourcen unterstützt?

Schauen Sie, ob Ihre Karte auf der Liste der unsterstützten Karten steht. Sollte Ihre Karte nicht dabei sein, aber eine ähnliche Karte darunter sein, ist es ein Versuch wert.

Warnung
Linux 2.4. unterstützt derrzeit kein AGP 8x. Versuchen Sie im BIOS auf AGP 4x herunterzuschalten. Sollte das nicht möglich sein, müssen Sie einen Patch auf den Vanilla Kernel anwenden.

5.3 Installation der CVS Sourcen

Folgen Sie der Anleitung unter "Kompilieren und Installation des Kernels". Dann springen Sie zu Punkt 6 dieser Anleitung und folgen Sie bis zu Punkt 8.3.

5.4 Herunterladen der CVS Sourcen

Legen Sie ein Verzeichnis für die CVS Dateien an:

Befehlsauflistung 9

```
# cd ~
# mkdir DRI-CVS
```

Nun werden die CVS Sourcen ausgecheckt:

Befehlsauflistung 10

```
# cd ~/DRI-CVS
# cvs -d:pserver:anonymous@cvs.dri.sourceforge.net:/cvsroot/dri login
// ( ENTER drücken, wenn ein Passwort verlangt wird)
# cvs -z3 -d:pserver:anonymous@cvs.dri.sourceforge.net:/cvsroot/dri co xc
// Mit der -z3 werden die Daten komprimiert, wass eine kürzere Downloadzeit zur Folge hat.
```

5.5 Aktualisierung der CVS Sourcen

Um in Zukunft ab und zu den DRI Quellcode mit den neuesten Änderungen zu erhalten, muss ein Update ausgeführt werden:

Befehlsauflistung 11

```
# cd ~/DRI-CVS
# cvs -z3 update -dA xc
```

5.6 Erstellen eines build Verzeichnisbaumes

Anstatt Objektdateien und Bibliotheken direkt in den Quellcode-Verzeichnisbaum zu legen, können diese in einen parallelen **build** Verzeichnisbaum abgelegt werden. Dieser wird mit dem **lndir** Befehl erzeugt.

Befehlsauflistung 12

```
# cd ~/DRI-CVS
# ln -s xc XFree40
# mkdir build; cd build
# lndir -silent -ignorelinks ../XFree40
```

Dadurch werden im build Verzeichnisbaum symbolische Links angelegt, welche auf den CVS Verzeichnisbaum verweisen. Fortgeschrittene Benutzer haben oft verschiedene build Verzeichnisbäume zum kompilieren und testen der verschiedenen Optionen.

5.7 Anpassung der host.def Datei

Die Datei ~/DRI-CVS/build/xc/config/cf/host.def dient der Steuerung der XFree86 Kompilierung. Diese kann natürlich den eigenen Wünschen oder Systemkonfigurationen angepasst werden. Standardmässig sieht die Datei etwa so aus:

Befehlsauflistung 13: ~/DRI-CVS/build/xc/config/cf/host.def

```
            #define DefaultCCOptions -Wall
// Für i386:
            #define DefaultGcc2i386Opt -O2
// Für Alpha:
            #define DefaultGcc2AxpOpt -O2 -mcpu=ev6 (or similar)
// Für alle anderen Architekturen
            #define LibraryCDebugFlags -O2
            #define BuildServersOnly YES
            #define XF86CardDrivers vga tdfx mga ati i810
            #define LinuxDistribution LinuxRedHat
            #define DefaultCCOptions -ansi GccWarningOptions -pipe
            #define BuildXF86DRI YES
            /* Optionally turn these on for debugging */
            /* #define GlxBuiltInTdfx YES */
            /* #define GlxBuiltInMga YES */
            /* #define GlxBuiltInR128 YES */
            /* #define GlxBuiltInRadeon YES */
            /* #define DoLoadableServer NO */
            #define SharedLibFont NO
```

Mit der Variable ProjectRoot kann die Installation der XFree86 Dateien angepasst werden. Wir empehlen die Installation der DRI Dateien in die bestehende XFree86 Installation - dies ist in der regel sicherer und weniger störanfällig, auch wenn wir in anderen Richtlinien nicht so verfahren.

Sollte XFree86 4.x nicht in /usr/X11R6/ installiert sein, muss folgende Zeile zur host.def hinzugefügt werden:

Befehlsauflistung 14

```
#define ProjectRoot pathToYourXFree86installation
// Note the XF86CardDrivers line to be sure your card's driver is listed.
// Zur Verwendung von 3DNow! Optimierungen in Mesa und DRI Treibern:
#define MesaUse3DNow YES
// Für diese Option ist kein AMD Prozessor erforderlich.
// Der DRI Treiber wird beim Start nach 3DNow! schauen, und bei Bedarf anwenden.
```

Für die Verwendung von SSE Optimierungen in Mesa und DRI muss mindestens ein 2.4.x Kernel vorhanden sein. Mesa überprüft selbstständig, ob SSE sowohl vom Prozessor als auch vom Betriebssystem unterstützt wird. Um Mesa in den DRI Treibern kompilieren zu können, müssen die 2.4.x Kernel Header in /usr/src/linux befinden. Sollten Sie SSE für eine ältere Kernelversion in /usr/src/linux aktivieren, wird die Kompilation von Mesa fehlschlagen. Sagen Sie nicht, wir hätten Sie gewarnt! Als 2.4.x Kernel Benutzer können Sie folgendes hinzufügen:

Befehlsauflistung 15

```
#define MesaUseKatmai YES
```

5.8 Kompilieren des XFree86/DRI Verzeichnisbaumes

Zum Kompilieren genügt folgendes:

Befehlsauflistung 16

```
# cd ~/DRI-CVS/build/xc/
# make World >& world.log
```

Manchmal scheint es notwendig zu sein, zusätzlich folgendes auszuführen:

Befehlsauflistung 17

```
# cd ~/DRI-CVS/build/xc/programs/Xserver/hw/xfree86/os-support/linux/drm/kernel
# make -f Makefile.linux radeon.o
// Ersetzen Sie radeon.o durch Ihren eigenen Treiber
```

Es ist normal, dass während der Kompilierung Warnungen auftreten. Da dies eine Weile dauern wird, empfehlen wir solange das Lesen Ihrer Mails oder ein Besuch bei heise.

Warnung
Die make Option -j sollte nicht verwendet werden. Dies funktioniert nicht unter XFree86/DRI

Mit Ihrem Text Editor können Sie die Log-Datei nach Fehlern durchsuchen, die in der Regel mit * * * beginnen.

5.9 Installation des Treibers

Überprüfen Sie, ob die DRI Kernelmodule erfolgreich erstellt wurden:

Befehlsauflistung 18

```
# cd ~/DRI-CVS/build/xc/programs/Xserver/hw/xfree86/os-support/linux/drm/kernel; ls
```

Für einen 3dfx Voodoo Treiber sollten Sie tdfx.o sehen. Für Matrox G200/G400 ist dies mga.o, für ATI Rage128 r128.o, für Intel i810 ein i810.o, für ATI Radeon ist dies radeon.o. Sollte die Kompilation der DRI Module fehlgeschlagen sein, sollten Sie die Linux Kernel Version überprüfen. Auch werden nicht immer die allerneuesten Kernel unterstützt.

Nun kopieren wir alle Treiberdateien in die XFree86 Installation. Bei Bedarf können die zu überschreibenden Dateien vorher gesichert werden. In diesem Beispiel verwenden wir eine Radeon Karte. Für Ihre eigene Karte müssen Sie die entsprechenden Dateien kopieren. Ein Blick in das entsprechende Verzeichnis hilft hier oft weiter.

Befehlsauflistung 19

```
# cd ~/DRI-CVS/build/xc/exports/lib/modules/
# cp dri/r200_dri.so /usr/X11R6/lib/modules/dri/
# cp drivers/atimisc_drv.o /usr/X11R6/lib/modules/drivers/
# cp drivers/radeon_drv.o /usr/X11R6/lib/modules/drivers/
# cp extensions/libdri.a /usr/X11R6/lib/modules/extensions/
# cp linux/libdrm.a /usr/X11R6/lib/modules/linux/
```

Dann folgen Sie den Anweisungen in Kapitel 3, Abschnitt "Konfiguration von XF86Config".

um das entsprechende DRM Modul in den laufenden KErnel zu laden, sollte das KErnel modul nach /lib/modules/ kopiert werden und mit uname -r`/kernel/drivers/char/drm/ ausgeführt werden. Anschliessend **modules-update** und den X Server neu starten. Sollten Sie den Treiber nicht für den aktuell benutzten Kernel erstellt haben, muss uname -r an den zu verwendenden Kernel angepasst werden.

Warnung
Stellen Sie sicher, dass ältere DRI Module vorher entladen werden. Manche DRM Module erwarten das apgart vorher geladen wird.

6. Performance Tuning

6.1 Das meiste aus Direct Rendering herausholen

Einige Optionen können die Performance um mehr als 30% steigern. Diese können in /etc/X11/XF86Config gesetzt werden:

Befehlsauflistung 20: XF86Config

```
Section "Device"
     Option     "AGPMode" "4"
// Dies steigerte meine FPS von 609 zu 618.
     Option     "AGPFastWrite" "True"
// Dies hatte keinen messbaren Effekt, kann aber die Instabilität des Computers erhöhen.
// Eventuell muss es im BIOS zusätzlich aktiviert werden.
     Option     "EnablePageFlip" "True"
// Dies steigerte FPS von 618 zu 702. Angeblich soll es riskant sein, aber wenige Leute hatten P
     ...
EndSection
```

Für weitere Tuning Optionen sollte die DRI Infoseite konsultiert werden.

7. Problembehebung

7.1 Es funktiniert einfach nicht. Ich habe kein Rendering und ich weiss nicht warum.

Versuchen Sie ein *insmod radeon* vor dem Starten des X server. Versuchen Sie ausserdem apgart als Modul zu erstellen, statt direkt in den Kernel zu kompilieren.

7.2 Beim Ausführen von startx erhalte ich folgenden Fehler: "[drm] failed to load kernel module agpgart"

Dies erscheint immer, wenn apgart direkt in den Kernel einkompiliert wird. Sie können dies getrost ignorieren.

7.3 Direct rendering funktioniert nicht und in /var/log/XFree86.0.log erhalte ich eine Fehlermeldung, das die Treiberversion falsch ist.

Sie benutzen nicht den xfree-drm Treiber. SInd Sie sicher, dass DRM und der Treiber in den richtigen Kernel kompiliert wurde?

7.4 Ich besitze eine Radeon und möchte TV-Out nutzen

Dafür gibt es die ati-gatos Treiber. *emerge -s gatos*.

7.5 Es funktioniert immer noch nicht. Meine Karte ist so neu und megaobercool, dass sie nicht unterstützt wird.

Probieren Sie die binären Treiber aus. Sollte für Ihre Karte noch keine Treiber geben, probieren Sie VESA. Es ist langsam, funktioniert aber. Halten Sie nach neuen Treibern Ausschau.

7.6 I habe eine PCI Karte, die nicht arbeiten will. Hilfe!

Im Abschnitt "Device" muss ForcePCIMode eingetragen sein.

Befehlsauflistung 21

```
Option "ForcePCIMode" "True"
```

8. Referenzen

- http://forums.gentoo.org/viewtopic.php?t=46681
- http://forums.gentoo.org/viewtopic.php?t=29264

- http://dri.sourceforge.net/
- http://www.retinalburn.net/linux/dri_status.html