Meine Multimediageschichte (I)

vorgespielt von Magnus

Vorgeschichte

Zu meinem Linuxstart Anfang 2005 bewegte ich mich auf einer etwas schmalbrüstigen, zusammen-gesuchten Hardware (PII 350 MHZ, 256 MB, GeForce MX 200 64 MB, 10 GB), so dass das Thema Multimedia schon aus diesem Grunde eher am Rande meines Fokus stand. Nach einigem Hin und Her (bis Mitte 2006) und etwas erweiterten Linuxgrundkenntnissen funktionierte mein System (mehr oder weniger) so, wie ich es benötigte (OpenOffice, Internet), inkl. ein bisschen Musik, dem ruckelfreien Abspielen von Film-DVDs via Xine und dem örtlichen TV-Kabelprogramm via Kde TV mittels einer alten analogen Pinnacle TV-Karte (Bt878 Chip), deren nachträgliche Einbindung unter Mandriva 2006 problemlos klappte.

Der Jahreswechsel brachte dann endlich Mandriva 2007 auf verbesserter Hardware (P4 1,8 GHz, 256 MB, Ge-Force TI 4200 128 MB, 120 GB). Nach einigem Geschraube und Installationsanläufen stand mein System wieder. Die „normalen“ Funktionen waren nun da und mein Einstieg in die bewegte Multimediawelt konnte beginnen.

Prolog 1

Ausgehend von der TV-Karte und der Beschäftigung mit dem Thema Video ist ein ganzer „Rattenschwanz“ von Ideen entstanden. Mein blauäugiger Ansatz, mal einen kurzen Abriss über den Einsatz einer Handvoll Programme aus dem Bereich Video zu verfassen, hatte sich bei realistischer Betrachtung, d. h. nach der Installation der ersten Programme, deren Tests und den ersten Beschreibungen, in Luft aufgelöst.

Es gab einfach zu viele Programmoptionen und gleichzeitig zu wenig Zeit. Um das Ganze nicht ausufern zu lassen und seitenweise Programmbeschreibungen zu erstellen, habe ich die ganze Arbeit unter der Prämisse gestellt, den Weg vom Ausgangsmaterial (TV, Video-Kamera) hin zum fertigen Video bzw. zur DVD mit möglichst einfachen Mittel zu erreichen. Ich habe mich jeweils auf die aus meiner Sicht wichtigsten Funktionen und Optionen beschränkt, um einen praktikablen Minimalweg zu finden.

Ich hoffe, dass ich den Lesern so einen Weg beschreibe und die entsprechenden Hintergründe liefere, um dies dann vielleicht auf seinem Rechner relativ „schnell“ nachvollziehen zu können. Derjenige, der sich mit dem Thema Video intensiver befassen möchte, findet in allen Programmen eine Vielzahl von Möglichkeiten zu Gestaltung und Optimierung, ganz zu schweigen, dass es auch noch eine Reihe alternativer Programme gibt, die vielleicht den persönlichen Vorstellungen besser entsprechen.

Ich denke auch, dass sich das eine oder andere der folgende Geschichte über die Konsole schneller lösen ließe, zumal die „KDE-Programme“ oftmals nur graphische Oberflächen für konsolenbasierte Programme sind, die bestimmte, festgelegte Optionen nutzten. Meine Auswahl ist daher auch nicht als „best-of“ zu sehen. Mir ging es vielmehr darum, die verschiedenen Multimediaansätze einfach nur zum Laufen zu bekommen, um bei Bedarf eine praktikable Lösung zur Hand zu haben.

Somit erwartet Euch nun ein roter Faden durch den Dschungel der Videowelt, mit kurzen Programmanleitungen, Tipps und Punkten, die ich persönlich als wichtig empfinde bzw. über die ich gestolpert bin, also kein detailliertes Handbuch für die Nutzung der einzelnen Programme bis in ihre Tiefe. Zumal die Mehrzahl der Programme unter KDE laufen, sind viele Programmfunktionen auch selbsterklärend.

Einiges, was im Kern nichts mit Multimedia zu tun hat, war für mich Neuland (z. B. Shell-Scripte), daher habe ich es einfach auch aufgeschrieben. Zudem sollte jedem Leser klar sein, dass bis zum endgültigen Funktionieren in der Regel mehrere Anläufe notwendig waren und dass so auch dieser Text mit meine Erfahrungen „gewachsen“ ist.

Die folgenden Ausschweifungen bilden eine Mischung aus der chronologischen und thematischen Reihenfolge. Die einzelnen Punkte des Artikels lassen sich wie folgt logisch strukturieren:

- **Theorie**
  - Video-/Audio-Codecs
  - Interlacing / Deinterlacing
  - Organisation der Dateiablage
- **Eingabeschnittstellen**
  - TV-Karte (analog)
  - Video-Kamera
  - VHS-Videorecorder
  - DVD rippen
  - die „gute, alte“ Schallplatte
- **Verarbeitung (Schnitt und Aufbereitung)**
  - Kino, für die Video-Kamera
  - Avidemux
- **Videorecorder**
  - „zu Fuß“
  - TVBrowser
  - MythTV
- **DVD-Erstellung**
  - ManDVD
  - DVDStyler

Mein Startschuss war Mitte Januar 2007 und das hier beschriebene, vorläufige Ende ist April 2007.

Prolog 2

Im Laufe der Entwicklung des Artikels zeigte sich frühzeitig, dass der Umfang eine MagDriva-Ausgabe sprengen würde. Also entstand vom Konzept her eine kleine Artikelserie.

In diesem ersten Teil beschreibe ich nun schwerpunktmäßig, wie die laufenden Bilder (TV und Video-Kamera) auf meinen Rechner kommen und im zweiten Schritt aufbereitet werden. Weitere Verfeinerungen und die Erstellung von DVDs erscheinen dann in folgenden Ausgaben der MagDriva.

Video-Codecs

Nachdem ich erstmal mit dem Prinzip 'Try-and-error' losgelegt habe, wurde mir schnell klar, dass ich mir ein paar Gedanken zum Thema Videocodecs machen musste, um nicht jedesmal bei den Import und Exportaktionen darüber nachdenken zu müssen, mit welchen Formaten ich gerade kämpfe und was für ein Format eigentlich benötigt wird. So vielfältig die Computerwelt nun mal ist, gibt es auch diverse Video-Formate in den unterschiedlichen Ausprägungen. Weiterhin ist anzumerken, dass man sich durch die rechtliche Situation in Deutschland und der restriktiven Lizenzbedingungen bestimmter Video- und Sound-Formate (daher fehlen bei verschiedenen Linux-Distributionen teilweise auch einige Programmpakete) beim Thema Video und Sound in einer rechtlichen Grauzone bewegt.

So versuche ich an dieser Stelle eine kleine Einführung bzw. einen kleinen Überblick, allerdings ohne den Anspruch auf Vollständigkeit.

Eine digitale Video-Datei besteht wie der klassische Zelluloidfilm aus vielen Einzelbildern (Frames), die in regelmäßigen Abständen aufgenommen werden. Dies bedeutet, dass sich bei einer Auflösung von 720 x 576, 25 RGB-Bildern pro Sekunde und einer Stunde Spieldauer ca. 102 GB (= 1,7 GB/Min) ergeben. Ein bisschen viel für das normale Speicherleben, daher sind digitale Videos in der Praxis stets komprimiert.

Um z. B. zwei Stunden Film auf eine normale DVD (4,3 GB) zu brennen, müssen die Daten um den Faktor 50 vermindert werden. Also werden im Verleich vom angezeigten DVD-Film zum aufgenomenen Kamerabild Pixel „geklaut“. Dies muss nun so geschehen, dass dem geneigten Betrachter nichts bzw. fast nichts auffällt.

Die Programmbibliotheken, die Videos komprimieren (Codecs, Kunstwort vom englischen compress und decompress) nutzen die Tatsache, dass es sich bei Videos eben nicht um unzusammenhängende Einzelbilder handelt. Da sich von Bild zu Bild oft nicht viel ändert, werden nur die Veränderungen von Bild zu Bild gespeichert, wodurch natürlich eine Menge an Platz gespart wird. Somit könnte prinzipiell eine Wiedergabe nur am Anfang einer Datei starten. Daher fügen Videocodecs regelmäßig ein vollständiges Bild (= Keyframe) ein um dieses Problem zu beseitigen.

Grundsätzlich muss man zwei verschiedene Arten von Codecs unterscheiden:

- *Native Codecs* sind normale Linux-Binärdateien (meistens shared objects, libCODECNAME.so).
- *Windows-Codecs* sind die originalen oder leicht modifizierten Windows dynamic libraries (DLLs, AX u. ä.) Diese Codecs funktionieren nicht von Haus aus unter Linux. Es wird eine spezielle Library (*avifile*) benötigt, um diese Codecs verwenden zu können.

Im allgemeinen Sprachgebrauch mischen sich allerdings die Begrifflichkeiten Videocodecs und Container, die ein definiertes Dateiformat besitzen, das den Inhalt verschiedener Dateiformate zulässt. Hierzu eine kleine Übersicht der verschiedenen, gängigsten Begrifflichkeiten:

- MPEG-4 ist ein offizieller Video-Kompressions-Standard. Es gibt keinen Codec, dessen Name einfach MPEG-4 lautet. Wenn über MPEG-4 gesprochen wird, ist nicht ein spezifischen Codec gemeint, sondern eine Sammlung von Techniken, Videos zu komprimieren. MPEG-4-kompatibel bedeutet, dass ein Codec Dateien erzeugt, die mit anderen MPEG-4-kompatiblen abgespielt werden können.
- DivX ist der "original" gehackte Microsoft MPEG-4-Codec, mit dem die ganze "Ripperei" anfing. Er wird unter Linux seltener benutzt, da es mittlerweile Codecs gibt, die eine bessere Qualität zur Verfügung stellen. Der Codec kommt als Windows dynamic library daher (divx.dll und einige andere) und benötigt unter Linux *avifile*.
- DivX 4 und 5 sind die offiziellen Nachfolger, wobei DivX 4 durch DivX 5 ersetzt wird. Dies sind native Linux shared objects - genannt libdivxdecore.*so* und libdivxencore.so. Die Sourcen dazu liegen nicht offen.
- XviD (das ist DivX rückwärts gelesen) ist eine Open-Source MPEG-4-Implementierung, die im Hinblick auf Kompression und Bildqualität wirklich gut ist.
- libavcodec oder kurz *lavc* ist ein weiterer Open Source MPEG-4-kompatibler Video-Codec, der in Performance und Qualität DivX 5 und auch XviD überlegen ist. Er ist Teil des ffmpeg-Projekts.
- MPEG-1-kompatible Codecs die für VCDs
- MPEG-2-kompatible Codecs für SVCDs oder DVDs.
- Die meisten anderen Codecs sind entweder veraltet oder andere gecrackte Version von Microsofts MPEG-4-Codec.
- AVI bezeichnet eine Container-Format, dass verschiedenste Video-Formate enthalten kann (schlechte bzw keine Unterstützung von Untertiteln, Menüs und Kapiteln, daher Weiterentwicklungen hinzu DivX.)

- x.264 bezeichnet die Apple-Variante eines MPEG-4-Codecs (auch MPEG-4 AVC oder H.264)

Aus meiner Sicht sind die wesentlichen zwei Codecs XviD und lavc. Die Gründe dafür sind, dass beide eine exzellente Qualität bieten, schnell sind und man ggf. nur einen MPEG4-kompatiblen Decoder (wie DivX 5 oder XviD) auf einem Win-System zum Abspielen braucht. Die Wiedergabe unter Linux ist überhaupt kein Problem - *MPlayer* oder *Xine* spielen fröhlich DivX 4/5, XviD und lavc-kodierte Filme. MPEG-2-codierte Videos lassen sich natürlich auch auf der Festplatte abspielen.

Im Vergleich zur MPEG-4-Kodierung wird hier aber für die gleiche Qualität mehr Platz benötigt, bzw. bei gleicher Größe liefert MPEG-4 die bessere Qualität. Allerdings ist dies das Eingangsformat zur Gestaltung von Video-DVDs.

Vollständigkeitshalber gibt es noch eine Liste für Audio-Codecs:

- MP3 ist die Kurzform für *MPEG1 layer 3* und ist ein offizieller Kompressions-Standard. Wenn über MP3 gesprochen wird, ist im Prinzip die Kompressions-Technik gemeint, nicht aber ein speziellen Codec. Es gibt mehr Codecs für MP3 als man an einem Tag aufzählen kann.
- *lame* ist eine Abkürzung für "Lame Ain't an MP3 Encoder" ("Lame ist kein MP3-Encoder" - auch wenn es einer ist ;-)) *lame* bietet einen Encoder der der MP3-Dateien in sehr guter Qualität produziert. Sowohl *transcode* als auch *mencoder* nutzen *lame* zur Audio-Kompression.
- AC3 ist wieder ein offizieller Audio-Kompressions-Standard. Nahezu alle DVDs enthalten AC3-kodierte Audiospuren. Es gibt heute Dekoder sowohl für Windows als auch für Linux, die mit AC3-Ton innerhalb von AVIs arbeiten. Der Vorteil ist, dass keine erneute Komprimierung nötig ist (eine erneute Komprimierung ist immer mit einem Verlust an Qualität verbunden) und das Mehrkanal-Ton (Dolby Surround und ähnliches) erhalten bleibt. Der Nachteil ist, dass *AC3*-Ton mehr Speicherplatz als *MP3*-kodierter Ton benötigt.
- OggVorbis als der Open Source Audio-Kompressions-Codec.
- FLAC die Apple-Variante.

Die Ausstattung der einzelnen Distributionen mit Video-Codecs ist unterschiedlich. Vom Gefühl her scheint aber Mandriva schon einiges mit zuliefern. Ein weitere Teil kommt auch im Zuge der die Installation von weiteren Multimedia-Programmen wie z. B. *Ffmpeg, Xine, MEncoder usw.*. Genug des Vorgeplänkels und der Theorie, jetzt geht es mit dem Praktischen los!

TV-**Karte**

Das Einbindung meiner TV-Karte funktionierte nicht direkt bei der Installation von Mandriva 2007 (Free, als Download), die Karte wurde nicht erkannt. Aber nach dem Wechsel des Steckplatzes und einem kurzen Aufenthalt im MCC ('Hardware/Betrachten und konfigurieren der Hardware') war die Hardware bereit.

Beim ersten Start von *KdeTV* brachte der initialisierte Suchlauf alle Sender meines Kabelnetzes, die ich dann nach meinen Bedürfnissen beschriftet und sortiert habe. Ohne weiteren Aufwand konnte ich nun in die Glotze schauen. Die Qualität einiger Sender ist aber etwas schlechter als in der normalen TV-Kiste. Zu beachten ist, dass meine TV-Karte über den Soundausgang mit dem Line-In der Sound-Karte verbunden werden muss, da man sich sonst nur in der Stummfilmwelt bewegt.

TV-Aufnahme (ffmpeg)

Um es vorweg zu nehmen, mit *Ffmpeg* ist es relativ einfach und auch schnell bewerkstelligt, eine laufende TV-Sendung im passenden Format aufzuzeichnen. Aber mal langsam von vorne.

Mit der funktionierenden TV-Karte waren auch sofort meine Aufnahmegelüste da, zusätzlich angestachelt durch einen Artikel in der EasyLinux 12/2006 [3].

Meine ersten Versuche auf der alten Hardware brachten die bewegten Bilder auch schnell auf die Platte, allerdings hinkte der Ton zeitversetzt hinterher. Die Experimente mit den vielfältigen Parametern von *Ffmpeg* führten auch zu keinem vernünftigen Ergebnis. Also habe ich es auf die Hardware geschoben und auf meinen verbesserten Rechner gewartet.

Hier lieferte dann der Befehl

```
ffmpeg -y -t 120 -vd /dev/v4l/video0 -ad /dev/dsp
-target pal-dvd /daten/film.mpeg
```

das passende Ergebnis. Zur Erläuterung der Parameter:

-y die Ausgabedatei wird automatisch überschrieben, wenn der Parameter fehlt, bei bestehender Datei erscheint eine Abfrage

-t Aufnahmedauer in Sekunden, z. B. 120 = 2 Minuten

-vd Videoquelle

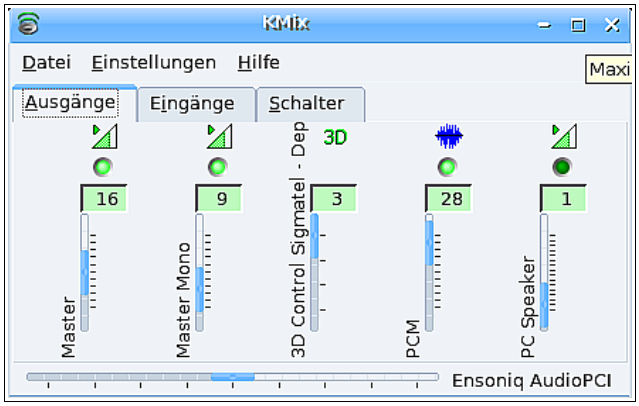
-ad Audioquelle

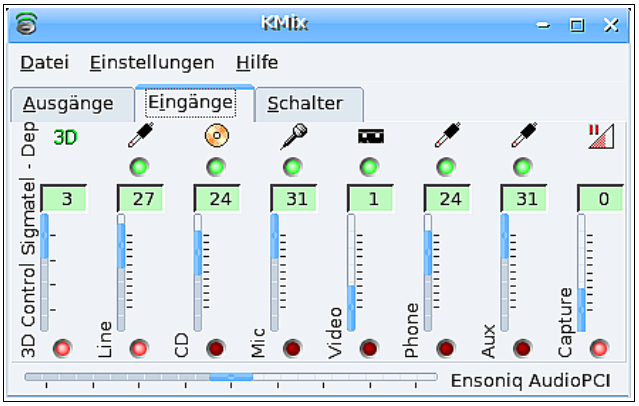
-target Zielformat, fasst die Einzeloptionen zusammen.

Gem. dem obigen Artikel bietet *ffmpeg* mit diesem Zielformat ein gutes Verhältnis zwischen Qualität und Speicherplatz (ca. 50 MB/Minute). Als Format ergibt sich ein MPEG-2-Container (MPEG-2 Video, AC3 Audio). Im reichhaltigen Optionsvorrat von *Ffmpeg* finden sich weitere Ausgabeoptionen bzw. -formate, für das Ziel der DVD-Erstellung ist das ausgewählte aus meiner Sicht ok, da so nicht wieder umformatiert werden muss.

Grundsätzlich lässt sich die Aufnahmefunktionalität auch mit den Konsolenprogrammen *Mencoder, Transcode* realisieren, allerdings ist dabei die Parametrisierung etwas komplexer.

Auf Grund fehlerhafter Einstellungen im *KMix* gab es erst noch ein paar Stummfilmsequenzen. So musste ich feststellen, dass auch ein scheinbar einfaches Programm so seine Tücken hat. Letztendlich sieht mein aktuelles *KMix* nun wie folgt aus:





An Speicher benötigt *Ffmpeg* in meiner Konstellation für die Aufnahme ca. 47 - 50 MB pro Minute (MPEG-2/AC3).

Anzumerken ist noch, dass *KdeTV* mit dem richtigen TV-Programm aktiv sein muss. Auch gibt es während der Aufnahme nur den Live-Ton, das Bild friert ein. Ich vermute, dies liegt aber an meiner Hardware (CPU, Arbeitsspeicher, Grafikkarte).

Als fauler, vergesslicher Mensch habe ich mir dann ein kleines, einfaches, interaktives Script geschrieben, dass den obigen Befehl enthält und die Aufnahmedauer und den Dateinamen abfragt.

Listing *tvauf01.sh*

```
#!/bin/csh
#
#TV-Aufnahme mit Eingabe Dauer + Dateiname
#
echo “Dauer der Aufnahme in Minuten : “
@ zeit = $<
@ zeit = $zeit * 60
#
echo “Dateiname : “
@ film = $<
#
ffmpeg -y -t $zeit -vd /dev/video0
-ad /dev/dsp -target pal-
dvd /daten/filme/tv/datei$film.mpeg
#
```

Sowohl *zeit* als auch film sind numerische Variablen, so dass die Aufnahmedatei z. B. datei4711.mpeg lautet. Leider ist es mir bisher nicht gelungen, den kompletten Dateinamen als Texteingabe zu realisieren.

Anmerkung:

Nach der Erstellung des Scripts habe ich es mit den folgenden Befehlen ausführbar gemacht, als Link im zentrale Scriptverzeichnis (als su) abgelegt und die entsprechenden Rechte so gesetzt, dass es durch jeden über die Eingabe des Scriptnamens ausführbar ist.

```
chmod +x tvauf01.sh
ln -s ~/Video/scripte/tvauf01.sh /usr/local/bin/
chmod u=rwx,go=rx /usr/local/bin/tvauf01.sh
```

Damit war dann auch die erste TV-Aufnahmerunde abgeschlossen.

Video

Auf vielfachen Wunsch meiner Kinder konnte ich das Christkind überreden, ihnen eine Videokamera auf den Gabentisch zu legen. Natürlich habe ich auch den Auftrag bekommen, ein entsprechendes Gerät auszuwählen. Nach einigen Informationsstunden im WWW und einer guten Beratung (bei SATURN!) fiel die Wahl auf eine Sony DCR-HC23. Ausschlaggebend waren hierfür

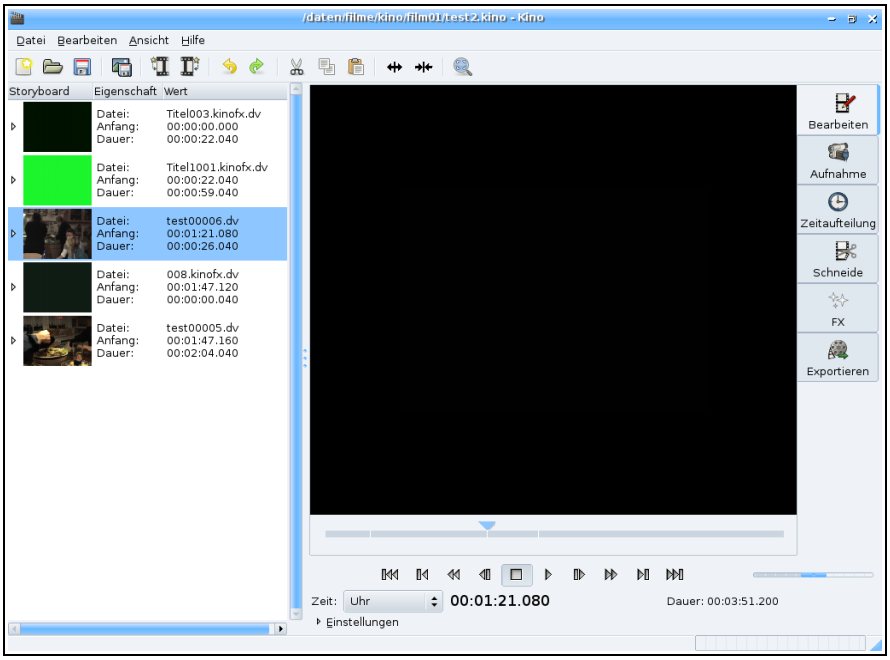
- der Preis (269,- €),
- das Speichermedium Kassette (da eine nicht so hohe Kompression, Speicherbedarf ca. 200 MB/Min.),
- der Preis der Kassetten (60 Minuten je Qualität von 2,50 € bis 12,00 €).

Für uns Gelegenheitsfilmer reicht dies. Bei Kameras mit DVD oder Harddisk verdoppelt sich der Preis ganz schnell und für die weitere Kompression gibt es nun eine nicht zu sehr komprimierte Vorlage.

Ergänzend kam noch für ca. 50,- € das Firewire-Equipment (PCI-Karte, Cardbus-Karte, zwei Kabel) vom Versender meines Vertrauens [4] hinzu.

Kino [a]

Passend lieferte Easy-Linux [5] den Artikel zum notwendigen Programm: *Kino*. Damit lassen sich Videodaten von der Kamera einlesen und bearbeiten. Via *urpmi* hatte ich ratz fatz das Programm inkl. dem zugehörigen Menüeintrag auf meiner Maschine (Version 0.9.5). U. U. muss noch *mpegtab* nachinstalliert werden.



Und nun ging es in die erste Runde, Kamera an Firewire und *Kino* gestartet. Beim ersten Aufruf müssen wie üblich die Grundeinstellungen über den entsprechenden Menüpunkt (Datei/Einstellungen) konfiguriert werden. Die meisten Einstellungen können unverändert übernommen werden. Folgende Punkte sollten überprüft bzw. angepasst werden:

- Reiter: Standardwerte
  - PAL, 48khz Stereo, 4:3
- Reiter: Aufnahme
  - Eingabe einer Ausgabedatei
  - Dateityp: RAW-DV,
  - Automatisches Aufteilen der Dateien in Filmsequenzen gem. der Aufnahme,
  - Rahmen pro Datei und Max. Dateigröße gleich 0 (für Systeme mit Größenbegrenzungen relevant)

Die weiteren Optionen für die Grundeinstellung können so bleiben, wie sie standardmäßig eingestellt sind, da sie für die hier beschriebene Funktionalität nicht relevant sind. Von den Funktionen her kann man mit Kino Kameradaten übernehmen, diese Daten schneiden, passend zusammenfügen und anschließend das fertige Ergebnis im benötigten Format speichern (= exportieren).

Optisch stellt sich das Programm mit drei Hauptteilen dar. Auf der linken Seite befindet sich das Storyboard, in dem sich die einzelnen Filme /Filmsequenzen eines Projektes befinden. In der Mitte ist das Hauptbild gem. der aktivierten Hauptfunktion.

Am rechten Rand liegt die Leiste mit den Hauptfunktionen:

- Bearbeiten, Filme schneiden, zusammenfügen usw.,
- Aufnahme, zur Übernahme der Kameradaten
- Zeitaufteilung,
- Schneide,
- FX, Erstellen der Übergänge zwischen zwei Filmen,
- Exportieren.

Über allem befindet sich eine Symbolleiste mit den wesentlichen, üblichen Bearbeitungsfunktionen sowie den Funktionen, um weitere Filme /Filmsequenzen hinzuzufügen und einzelne Filme/ Filmsequenzen zusammen zu fügen (s. a. unter Bearbeiten).

Philosophie

Grundsätzlich werden die eingelesen Kamera-Dateien nicht verändert. Alle Bearbeitungsaktionen (Schnitte, Trennungen, Zusammenfügen) bzw. erstellte Übergänge zwischen zwei Filmsequenzen werden innerhalb eines Projektes logisch gespeichert. Dabei werden für die Übergänge jeweils noch eigene Dateien erstellt und abgelegt. Die Dateien mit den Film-Rohdaten werden nur als Verknüpfungen dem Projekt zugeordnet. Erst über die Funktion „Exportieren“ werden dann die Ergebnisse der einzelnen Bearbeitungsschritte in einer neuen Video-Datei zusammengefasst.

Ein weiterer Aspekt ist die Darstellung der „Dauer“ einer Filmsequenz. Aus den verschiedenen Anzeigeformaten (Millisekunden, Sekunden, Minuten, Stunden jeweils mit den Bruchteilen der nächst kleineren Einheit, Uhr - analog zur Uhrzeit mit Millisekunden, SMPTE - Uhrzeit mit Frames, Frames) kann sich im Hauptbild je nach Geschmack eine Variante ausgewählt werden.

Die Anzeige (auch im Storyboard) beinhaltet dann jeweils den Startpunkt bezogen auf den Gesamtfilm und die Dauer einer Sequenz. Persönlich nutze ich entweder das Format „Uhr“ oder „SMPTE“.

Aufnahme

Nachdem die Kamera via Firewire angeschlossen ist (im Wiedergabe-Modus, andernfalls wird das aktuelle Bild der Kamera geliefert) und *Kino* gestartet wurde, erscheint im Hauptbild das erste Bild des Films. Unterhalb dieses Bildes befindet sich eine Steuerleiste:

- Aufnahme,
- Pause,
- Stop
- A/C, zur erneuten Verbindungsherstellung zur Kamera, falls die Verbindung nicht automatisch hergestellt wurde.

Nach der Eingabe eines Dateinamens und dem Betätigen des Aufnahme-Buttons wird die bespielte Kassette in Echtzeit (Aufnahmezeit = Spielzeit) übertragen. Die übertragenen Filmsequenzen (gekennzeichnet durch jede Stop-Aktivität bei der Videoaufnahme) werden in einzelne Dateien (Dateiname + lfd. Nr.) abgelegt. Während der Übernahme sollte die Maschine möglichst wenig zusätzlich belastet werden, um eine vollständige, fehlerfreie Übernahme zu gewährleisten. Nach dem Abschluss befindet sich der Film bzw. die einzelnen Filmsequenzen im Storyboard und sollten nun als Projekt gespeichert werden.

Bearbeiten

Nach dem Öffnen eines Kino-Projektes bzw. nach der Übernahme kann nun mit der Bearbeitung der Rohdaten begonnen werden.

Über die „einfachen“ Funktionen aus der Symbolleiste (Cut & Paste, Drop & Drag) lassen sich im Storyboard einzelne Filmsequenzen verschieben, kopieren und löschen. Des Weiteren lassen sich über entsprechende Funktionen (in der Symbolleiste) einzelne Filmsequenzen (= Dateien) zum Projekt (d. h. ins Storyboard) hinzufügen und Sequenzen zusammenfügen oder auch trennen.

Im Hauptbildschirm wird die Filmsequenz angezeigt, die im Storyboard ausgewählt ist. Über entsprechende Button kann nun die Sequenz abgespielt oder gespult werden. Außerdem befindet sich ein Schieberegister unterhalb der Filmanzeige, mit der auch durch die Sequenz navigiert werden kann. Über die Trennfunktion in der Symbolleiste kann an der Position des Schieberegister der Film in zwei Teile geteilt. Über dies Verfahren können die nicht benötigten Filmteile separiert und aus dem Storyboard gelöscht werden.

Nachdem einzelne Filmsequenzen herausgeschnitten wurden, sind sie im Projekt nicht mehr verfügbar. Daher handelt es sich bei dieser Technik um „harte“ Schnitte.

Schneiden

Mit dieser Funktion bietet sich die Möglichkeit von „weichen“ Schnitten, D. h. mit dieser Schnittoption verbleiben die herausgeschnittenen Teile im Projekt, so dass die jeweiligen Aktionen später zurückgenommen werden können. Allerdings wird hier der umgekehrte Ansatz verfolgt, da die nicht markierten Teile einer Filmsequenz logisch herausgeschnitten werden.

Unterhalb des Hauptbildschirms liegt wieder das Schieberegister und folgende Funktionsbuttons bzw Felder:

- Anfang
  Eingabefeld für Bildnummer
  Button, um die Position des Schieberegisters ins Eingabefeld zu übernehmen
  Zurücksetzen Anfang?
- Anfang und Ausgang verketten,damit lässt sich die definierte Bildlänge frei durch die ausgewählte Sequenz verschieben
- Ausgang
  Eingabefeld für Bildnummer, Button, um die Position des Schieberegisters ins Eingabefeld zu übernehmen
  Zurücksetzen Ausgang?
- Methode: Überschreiben oder Einfügen
- Dateieingabefeld (für Methode Einfügen)
- Anwenden: Button um ausgewählten Schnitt zu realisieren

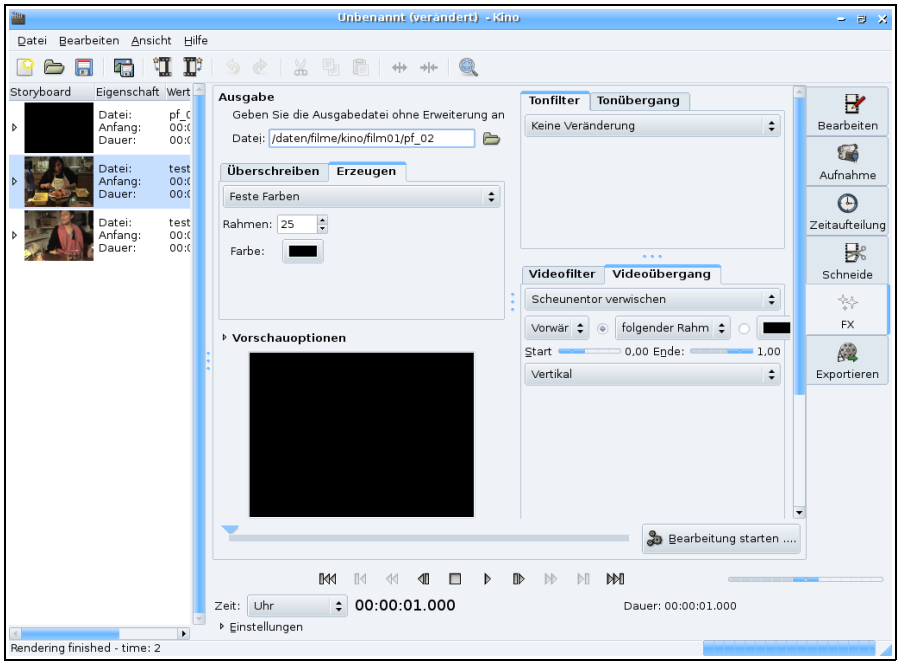
Für einen Schnitt fährt man mit dem Schieberegister zur angestrebten Schnittstelle. Zur Festlegung des Anfangspunktes klickt man nun auf das linke der beiden schwarzen Dreiecke, um die Zahl (= Bildnummer) in das links davon liegende Eingabefeld zu übernehmen. Alternativ kann hier auch die Bildnummer eingetragen werden. Für den Endpunkt gilt das gleiche Verfahren nur mit dem rechten schwarzen Dreieck.

Hinweis:

Die Steuerung der Schnitte erfolgt indirekt über das ausgewählte Zeitformat. Optimalerweise sollte hier das Format ‚frame’ ausgewählt werden, um framegenau zu schneiden. Da Kino aber keine Keyframes unterstützt, sind die Schnitte sorgfältig zu testen, um im Endprodukt Verzerrungen an den Schnittstellen zu vermeiden (Stichwort Kammartefakte). Die durch Kino automatisch durch-geführten Schnitte bei Szenenwechsel sind natürlich von dieser Problematik nicht betroffen.

FX

Nachdem nun die einzelnen Filmsequenzen passend geschnitten wurden kann man sie nun wie oben beschrieben einfach aneinander fügen oder sie mit Übergängen versehen. Hierzu bietet *Kino* unter dieser Hauptfunktion die Möglichkeit, unterstützt durch einige Vorlagen. Dies bedeutet eine Menge Spielraum für eigenen Kreativität. Ich beschränke mich im Rahmen des Artikels zur Verdeutlichung des Grundprinzip auf eine einfache Variante für die Szenenübergänge.



Im Hauptbildschirm befinden sich ein Vorschaubild zum erstellten Übergang, ein Eingabefeld für den Dateinamen des Übergangs und folgende drei mal zwei Reiter:

- Ausgabe: Angabe eines Dateinamens für den erzeugten Übergang
- Überschreiben/Erzeugen mit den zugehörigen Optionen
- Tonfilter/Tonübergang: jeweils mit einem Auswahlfeld für definierte Vorlagen
- Videofilter/Videoübergang: jeweils mit einem Auswahlfeld für definierte Vorlagen und verschiedenen Optionen (abhängig von der ausgewählten Vorlage)
- Bearbeitung starten: Erzeugung des Überganges

Damit ist ein breites Spektrum von Optionen geschaffenen, um seine Kreativität in Bildübergänge zu „packen“ (inkl. eines rudimentären Titelgenerators). Im Folgenden möchte ich anhand eines einfachen Beispiels die Grundstrukturen dieser Funktion aufzeigen. Hierbei geht es um das einfache Überblenden zwischen zwei Filmsequenzen.

Dies ist ein so genannter Scheunentoreffekt, bei dem zwei schwarze Balken von rechts und links (alternativ von oben und unten) die Szene abschließen und ein Überblenden, da sich die beiden Varianten sehr ähneln.

Für mein Beispiel habe ich zwei Filmausschnitte (Szene1 und Szene2) mit Eingangs-, Übergangs- und Abschlussblende versehen. Dabei habe ich jeweils eine neue Datei erzeugt. Hierzu muss im Ausgabefeld ein entsprechendes Verzeichnis und ein Grunddateiname eingestellt werden. Die vier erzeugten Sequenzen haben dann als Name diese Grundbezeichnung plus eine laufende Nummer

Einblenden

- Auswählen Szene1,
- Erzeugen:
  Einstellungen beibehalten
  (Rahmen (25) = 1 Sekunde,
- Videoübergang:
  Scheunentor verwischen,
  Vorwärts,
  folgender Rahmen,
  vertikal (von den Seiten),
- Bearbeiten starten
  erzeugt die neue Datei

Überblenden

- Auswählen Szene2,
- Erzeugen:
  Einstellungen beibehalten
- Videoübergang:
  Scheunentor verwischen,
  Rückwärts,
  vorheriger Rahmen,
  vertikal,
- Bearbeiten starten

- Auswählen Szene2,
- Erzeugen:
  Einstellungen beibehalten
- Videoübergang:
  Scheunentor verwischen,
  Vorwärts,
  folgender Rahmen,
  vertikal,
- Bearbeiten starten

Abschlussblende

- letze Szene nochmals kopieren und auswählen
- Erzeugen:
  Einstellungen beibehalten
- Videoübergang:
  Scheunentor verwischen,
  Rückwärts,
  vorheriger Rahmen,
  vertikal,
- Bearbeiten starten
- kopierte Szene wieder löschen

Das Vorgehen bei der Abschlussblende ist zwar nicht elegant aber so geht es einfach und schnell. Als Alternative zum 'Scheunentor' lässt sich der Übergang 'Blenden' verwenden. Hierzu sehen die Optionen genauso aus wie oben beschrieben. Das Ergebnis wirkt allerdings weicher und gefällt mir besser.

Das ist nur ein kleiner Ausschnitt der Möglichkeiten, Übergänge zu erzeugen. Für alle Varianten könnte ein eigenes Buch geschrieben werden.

Zeiteinteilung

Über diese Funktion wird die im Storyboard markierte Filmsequenz in 20 – 25 gleich große Teile (gem. der gewählten Zeitanzeige) optisch aufgeteilt und im Hauptbildschirm als Miniaturbild angezeigt. Bei einem Doppelklick auf einem dieser Teile landet man in der Bearbeitungsfunktion an der ausgewählten Stelle der Filmsequenz. Aus meiner Sicht dient diese Funktion nur zur besseren Orientierung in einer Filmsequenz. Wie sich die Anzahl der Aufteilung ergibt, hat sich mir beim Testen nicht erschlossen.

Exportieren

Nachdem nun das neue Video zu seiner endgültigen Form aufbereitet ist, kommt der letzte Schritt, der die einzelnen logischen physischen Teile jetzt zu einer einzelnen neuen, Datei zusammenfasst. Abhängig von der weiteren Verwendung bieten sich auch entsprechende Optionen, die über die Reiter des Hauptbildschirms ausgewählt werden. Wichtig ist, dass vor dem Export klar ist, was mit dem fertigen Film weiter geschehen soll, denn daraus ergibt sich das richtige Exportformat.

Folgende Exportoptionen (Reiter) werden angeboten:

- IEEE 1394 - zurück zur Kamera via Firewire,
- DV Datei - Speicherung im DV-Format (RAW DV), kein Verlust durch Komprimierung,
- MPEG - mehre Option um einen MPEG-2-Container zu erzeugen,
- Andere - Auswahl von mehreren Formaten und Qualitätsstufen (= Auflösungen),
- Standbild - Standbild einer ausgewählten Szene als Bild,
- Ton - Speicherung der Tonspur als *wav* oder *mp3* (nur bei installiertem *lame*).

Unter jedem Reiter muss eine Ausgabedatei ohne Dateierweiterung angegeben werden. Den zugehörigen Datei-Suffix legt *Kino* automatisch an.

Nachdem ich alle Varianten durchgetestet habe, nutze ich nun drei Formatvarianten aus dem Punkt „Andere', abhängig vom Nutzungsziel:

- Win - MPEG4 AVI (DivX/mp3, 5:22 Min., 26,9 MB, 640x480),
- Linux - XVID MPEG4 AVI (Xvid/mp3, 5:11 Min., 34,2 MB640x480),
- CD/DVD - DVD Video Standard VOB (mpeg2/AC3, 3:05 Min., 48,2 MB, fix).

Die Werte in den Klammern beziehen sich auf das neue Format, die Dauer, die Größe und die gewählte Auf-lösung. Die Basis für die Testwerte war jeweils ein DV-Format mit der Laufzeit 1:04 Minuten und der Größe 221 MB. Bei Halbierung der Auflösung reduzierte sich die Exportzeit auf ca. die Hälfte und die Größe um ca. 70 %. Bei einigen Ausgabeformaten lässt sich die Auflösung je nach Bedarf bis um den Faktor vier reduzieren.

Für den Export-Prozess eines kompletten Films müssen je nach der Größe der Ausgangsdatei(en) und der Auflösung schon mehrere Stunden in Kauf genommen werden. Also eine klassische Nachtschicht für den Rechner.

Fazit

Aus meiner Sicht ist *Kino* ein einfaches, schnell zu beherrschendes Programm, dass innerhalb relativ kurzer Zeit erlaubt, Videofilme von der Kamera auf den Rechner zu transportieren, die Rohdaten aufzubereiten und im benötigten Format abzuspeichern. Die Bearbeitungsmöglichkeit im Bereich von Effekten ist sicherlich eingeschränkt, doch für den „Hausgebrauch" ausreichend.

Die 'Quick-and-Dirty' Lösung ist ein AVI-Container mit Formaten je nach Zielsystem (die Win-Familie will ja auch bedient werden). Meine Zielrichtung ist aber die Erstellung einer entsprechende DVD, so dass das Format hier ein MPEG-2-Container ist (mpeg2, AC3).

Hinweis:

Parallel zu dem Artikel habe ich mit einer ausführlicheren Kurzanleitung begonnen, die sich auf die nun verfügbare „finale" Version 1.0.0 beziehen wird. Wenn sie fertig ist, kann sie bei Bedarf bei mir angefordert werden und wird auch irgendwo auf MandrivaUser.de auftauchen. Außerdem möchte ich noch explizit auf die ausführliche (englischsprachige) Dokumentation auf der Projektseite hinweisen.

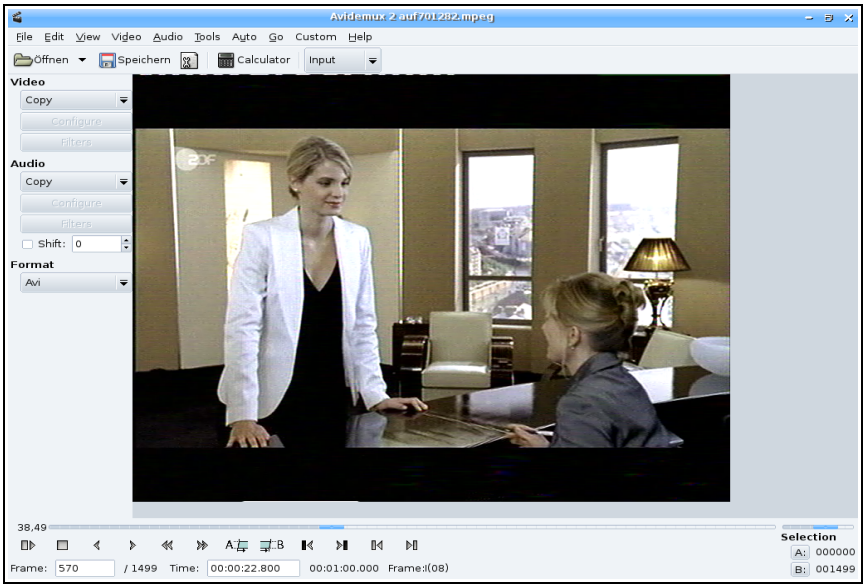
Der Filmschnitt (Avidemux [b])

Wie schon oben beschrieben nutze ich zum Schneiden und Aufbereiten meiner (Kamera-)Videos *Kino*. Um allerdings meine TV-Aufnahmen passend zu machen (korrekte Anfänge, Werbung raus usw.) ist dies Programm nicht geeignet, so dass ich mich nach etwas Recherche für *Avidemux* und damit für die grafische Oberfläche entschieden habe.

Und damit begann eine kleine Geschichte, die für mich im positiven Sinne die Linux-Welt ausmacht. Ich dachte mir so einmal *urpmi* und schon bin ich fertig. Pustekuchen! Das Ding verlangte eine Abhängigkeit zu Firefox 1.5 und das bei meinem Firefox 2!?. Nach einer Suchanfrage im Forum (natürlich MandrivaUser.de) zeigte sich, dass dies Problem auch schon unter Mandriva 2006 aufgetreten war. Meine Rückfrage an unsere Paketbauer brachte eine schnelle Antwort bzw. ein korrektes Paket (Oliver Burger sei hier nochmals gedankt). Ein schnelles

'urpmi avidemux'

und schon ging es los. Nach meinen ersten Tests und auch brauchbaren Ergebnissen lief plötzlich und unerwartet der Prozess zur DVD-Erstellung nicht mehr! Und das nach erwiesenermaßen korrekten Testergebnissen! Da ich nun schon eine Weile mit Linux lebe, gab es natürlich auch nur den Linux-Weg, d.h. Recherche im WWW. Und so bin ich im Avidemux-Forum gelandet. Dort auf englisch mein Problem beschrieben, da es dort keine passende Lösungsidee gab, und abgewartet.



Anderntags gab es einen Lösungsansatz: Austausch meiner Version 2.2 gegen die 2.4 svn (unstable), da hier der Output bessere Informationen liefert.. Ans Kompilieren habe ich mich nicht herangewagt. Damit gab es nochmals die Bitte an unsere Paketbauer.

Meinen Steilpass nahm doktor5000 auf und verwandelte ihn ein schöner RPM (nochmals ein dickes Dankeschön). Die neue Version produzierte den Abbruch nicht mehr und es ging weiter.

Nach dem Start des Programms (aus dem K-Menü, alternativ über den KDE-Schnellstarter [Alt-F2 und Eingabe von 'a*videmux*') öffnet man die zu bearbeitende Video-Datei auf dem üblichen Weg. *Avidemux* stellt fest, dass es sich um ein Videoformat handelt (in meinem Fall ein MPEG-2-Container) und fragt ab, ob ein Index zum Video erstellt werden soll. Dies wird bestätigt und für die nächsten ca. 10 Minuten (bei einer Dateigröße von ca. 5 GB = ca. 100 Minuten TV-Aufnahme) ist die Maschine beschäftigt. Dieser Index wird für den eigentlichen Filmschnitt benötigt.

Hinweis:

Unter 2007.1 gibt es das entsprechende Paket aus der MandrivaUser.de Quelle.

Einstellungen

Wie gehabt, gibt es wieder ein Reihe von Optionen unter '*Edit/Preferences*' einzustellen. Ich habe hier nur die wichtigsten und ihre Anpassungen dargestellt.

User Interface:

Save preferences on exit (ja)

Input:
Automatically index MPEG files (ja)
Use libavcodec MPEG decoder (ja)

Output:
Allow non standard audio frequency for DVD output (ja)
Audio:Audio output (= ALSA)
Video: Xvideo accel (best)

Für denjenigen, die tiefer eintauchen möchte, wieder eine schöne Spielwiese.

Optik

Im zweigeteilten Bildschirm finden sich alle für den Videoschnitt und die Videokompression relevanten Funktionen. Der überwiegende Teil wird durch das Wiedergabefenster eingenommen. In der Leiste links daneben befinden sich die Optionen für die Speicherung der geschnittenen Datei.

Unterhalb des Hauptbildschirms liegt ein Schieberegister (zur schnellen Navigation im Video) und eine Button-Leiste, über die man sich framegenau innerhalb eines Filmes bewegt.

Hier gibt es folgte zwölf Tasten:

- Start und Stop der Wiedergabe,
- Filmanfang, Filmende,
- nächster Frame, vorangehender Frame,
- nächster Keyframe, vorangehender Keyframe,
- nächster 'Black Frame', vorangehender 'Black Frame',
- Auswahlanfang und Auswahlende.?

Schnitt

Das realisierte Verfahren ermöglicht ein einfaches Arbeiten ohne Qualitätsverluste. Um z. B. einen Werbeblock zu entfernen, navigiert man über den Schieberegister in die Nähe der gesuchten Stelle.

Mit den Tasten

'Next/Previous frame'

nähert man sich der genaue Position langsam. Wie weiter oben schon ausgeführt, sollten Schnitte an den Keyframes angesetzt werden. Hierzu besitzt *Avidemux* entsprechende Tasten, um sich von Keyframe zu Keyframe zu bewegen. Um die besagten Werbeblöcke zu entfernen wird auch das passende Hilfsmittel geliefert.

Wiederum kann entsprechend einem

'Black frame'

zum nächsten navigiert werden. Um harte Übergänge zu vermeiden, wird .i. d. R. der Film abgeblendet und danach die Werbung eingeblendet. Dazwischen befindet sich dann mindestens ein schwarzer Frame, den man genau durch diese Funktionalität erreicht. Wenn allerdings, was manchmal auch vorkommt, mit Überblendungen gearbeitet wird, muss de genaue Schnittstelle per Hand ermittelt werden.

Wenn nun der entsprechende Stelle lokalisiert ist, wird sie über die entsprechende Taste

( )

als Beginn markiert. Analog wird für das Ende verfahren. Nachdem nun der der unerwünschte Werbeblock markiert ist, kann er über die Entf-Taste (oder Menüpunkt Edit/del) ins Datennirwana geschickt werden. Gleichermaßen kann man den Anfang und das Ende passend schneiden.

Speichern = Kompression

Nachdem der exakte Beginn, das Ende und ggf. weitere überflüssigen Teile wie Werbung usw, entfernt sind, geht es an die Sicherung der Arbeit. Vorher sollte sollte über die entsprechenden Tasten Anfang und Ende nochmals markiert werden, um zu vermeiden, dass nur ein Teil des Films gespeichert wird. Wie auch schon beim Export in *Kino* wird dies über den weiteren Verwendungszweck vorgegeben.

Aus meiner Sicht ergeben sich hierzu fünf Varianten:

1. im bestehenden Format ablegen,
2. für die Weiterverarbeitung in Richtung DVD,
3. Wiedergabe auf Linux-Rechner,
4. Wiedergabe auf Win-Rechner,
5. Wiedergabe auf iPod (= Apple-Format).

Die notwendigen Optionen werden in den drei Bereichen Video, Audio und Ausgabeformat eingestellt. Diese Eingabefelder befinden sich links neben dem Hauptbildschirm.

Die erste Variante biete sich an, wenn die fertig geschnittene Datei kleiner als 4,3 GB ist und sie somit ohne weitere Kompression auf eine DVD passt. Hierbei wird bei Video- und Audiooption jeweils 'Copy' eingestellt, das Ausgabeformat wäre dann 'PS MPEG'. In dieser Konstellation benötigte Avidemux ca. 30 Minuten für die 3,8 GB Datei.

Für die obige Variante 2. sind nun folgende Schritte notwendig:

i. Videooption: DVD
ii. Audiooption: Copy, da sich hier nichts ändert
iii. Ausgabeformat: PS MPEG
iv. Aufruf Kalkulator und anschließend mit folgenden Optionen rechnen (Button ‚Anwenden'):
Format: mpeg
Ziel: DVD5
Die Rechnung liefert eine Größe von ca 4,3 GB (Video + Audio)
v. Kontrolle der Konfigurationseinstellungen der Videooption. Hier sollte die Größe von höchstens 4,3 GB eingestellt sein.

Ausgehend von einer Aufnahmedatei von 90 Minuten mit einer Größe von 4,8 GB ergibt sich eine Rechenzeit von ca. 12 Stunden bei zwei Durchläufen.

Nachdem die entsprechenden Optionen eingestellt sind, kann die neue Datei gespeichert werden. Sobald dieser Prozess beginnt erscheint ein Pop-Up-Fenster, das den Verlauf (prozentual und zeitlich) darstellt. Die angezeigte Dauer war bei meinen Speicheraktionen in der Regel immer realistisch.

Fazit

Ähnliche wie *Kino* ist *Avidemux* im ersten Schritt ein einfaches und schnell zu beherrschendes Schnittprogramm. Durch die Sprungunterstützung zu den Keyframes und Black Frames und der einfachen Schnitttechnik ist der zu bearbeitende Film schnell in die passende Form gebracht. Es eignet sich allerdings nicht dazu, um einzelne Filmsequenzen einfach umzusortieren. Das Speichern im zweiten Schritt ist dagegen schon deutlich komplexer, da sich eine Vielzahl von Optionen bieten und diverse Möglichkeiten das Videomaterial zu optimieren. Hierbei ist sicherlich ein tieferes Verständnis zum Thema Videobearbeitung und eine längere Einarbeitungs- und Testzeit nötig.

Dem Anfänger bieten aber die fertigen Profile eine einfache Möglichkeit, seine Arbeit in das passende Ergebnis abzuspeichern. Beim Austesten von Speicheroptionen und den verschiedenen Filtern sollte mit kurzen Filmchen gearbeitet werden, da hierbei die Zeiten schon mal ausufern können. Ich habe einmal knapp 30 Stunden auf das Ergebnis von 90 Minuten Film gewartet!

Grundsätzlich möchte ich noch auf die Projektseite verweisen, da hier auch ein sehr umfangreiches, englischsprachiges Dokumentations-Wiki und ein aktives Forum (englisch, französisch) angesiedelt sind.

Interlacing / Deinterlacing

Zum Ende hin gibt es noch einmal etwas Theorie. Sowohl bei *Kino* als auch bei *Avidemux* fielen die Begriffe Kammartefakte bzw. Deinterlacing. Speziell Sportaufnahmen mit ihren schnellen Bewegungen können Verzerrungen in Form eines Kamms (daher auch Kammartefakte) aufweisen. Der Grund hierfür liegt darin, dass bei dem in Deutschland üblichen PAL-Fernsehformat pro Frame nur ein Halbbild übertragen wird. Somit werden abwechselnd jeweils nur die geradzahligen oder ungradzahligen Bildzeilen pro Frame übertragen. Dies Verfahren wird als Zeilensprung oder Interlacing bezeichnet.

Wenn nun ein TV-Gerät diese Wechsel mit der originalen Bildwechselfrequenz von 25 Herz abbildet, fällt dies dem menschlichen Auge nicht auf, da dies der zeitlichen Auflösefähigkeit von ca. 25 Bildern pro Sekunde entspricht. Gleichzeitig nimmt ein Betrachter weit weniger Flimmern wahr, als würden nur 12,5 Bilder pro Sekunde übertragen. Beim PAL-Standard sind es nicht 25 Vollbilder sondern 50 Halbbilder pro Sekunde mit einer Zeitverzögerung von 0,02 Sekunden zwischen den einzelnen Bildern. Im Prinzip bildet diese Technik das älteste Kompressionsverfahren, da die zu übertragene Datenmenge halbiert wird.

Computermonitore (Röhre) nutzen heute kein Interlacing mehr, zumal die Bildwiederholungsfrequenzen deutlich höher sind. Genau aus diesen Gründen verschieben sich speziell bei schnellen Bewegungen die Halbbilder gegeneinander und es kommt zu den beschriebenen Unschärfen. Dies wirkt sich entsprechend störend aus und muss für eine akzeptable Qualität der Wiedergabe beseitigt werden.

Das Verfahren, mit dem nun die vorliegenden Halbbilder in Vollbilder konvertiert werden, wird als Zeilenentflechtung oder mit dem englischen Begriff Deinterlacing bezeichnet. Faktisch sind davon neben 100-Hz-Fernsehgeräten alle Nicht-Röhren-Fernseher, Rück- und Frontprojektionsbildschirme und alle Computermonitore betroffen. Das Entflechten für das TV-Bild geschieht entweder durch das TV-Gerät oder durch die Hardware, die das entsprechende Signal liefert (DVD-Spieler, DVB-Empfänger usw.). Für die Darstellung über den Computer wird das Deinterlacing entweder durch die Abspielsoftware oder über die Hardware (z. B. TV-Karte) realisiert.

Um nochmals auf die schon oben angesprochenen Sportaufnahmen zurückzukommen, muss festgehalten werden, dass bei schnellen Bewegungen die Grenzen dieser Software teilweise überschritten werden, so dass zumindestens bei der Neuerstellung von Videodateien die Daten entsprechend bearbeitet bzw. gefiltert werden müssen. Somit ist die Bildqualität entscheidend vom den eingesetzten Deinterlacing-Algorithmus abhängig.

Leider gibt es keine perfekte Methode und alle haben ihre Vor- und Nachteile. Die heute im Wesentlichen genutzten Verfahren unterscheiden sich teilweise erheblich im betriebenen Aufwand. Daher im Folgenden ein kurzer Abriss der wichtigsten Verfahren:

Weave (field insertion)

Hier werden die Halbbilder zu einem Vollbild zusammengefügt und dies dann doppelt angezeigt. Durch den Zeitversatz bei TV-Aufnahmen entstehen deutliche Kammartefakte, daher ungeeignet für TV-Material. Allerdings wird teilweise diese einfache Methode als Vorstufe für weitere Deinterlacing-Verfahren genutzt.

Unschärfe

Das Verfahren funktioniert analog zur Weave-Methode. Allerdings werden die Vollbilder noch einmal weich gezeichnet, um die Kammartefakte abzuschwächen. Im Ergebnis ist das Videomaterial deutlich unschärfer.

Skip Field

Hierbei wird jeweils ein Halbbild weggelassen (das neue Bild wäre somit halb so groß) und alternativ die fehlenden Zeilen durch die Verdoppelung ersetzt oder per Interpolation als besseres Verfahren errechnet. Anschließend wird das erzeugte Bild doppelt angezeigt. Diese Methode hat den Vorteil, dass keine Kammartefakte auftreten, allerdings auf Kosten der Qualität (weicher als das Original), da Bildinformationen entfallen sind.

Bobbing (line averaging)

Beim Bobbing entfallen keine Bilder, sonder es werden beide Halbbilder zu einen Vollbild hochgerechnet und nacheinander abgespielt. Auch treten hier keine Kammartefakte auf und das Bild ist weicher. Da keine Bildinformationen entfallen, wirken Bewegungen flüssiger. Da die ersten und letzten Zeilen durch die fehlende Zeile schlecht zu interpolieren sind, kommt es bei der Wiedergabe zu einem Wackeleffekt (auf und ab).

Blending

Arbeitet ähnlich wie das Bobbing, wobei allerdings die beiden erzeugten Vollbilder übereinander gelegt, ihr Mittelwert errechnet und zweifach wiedergegeben werden. Der Vorteil ist das das Zittern entfällt. Hinsichtlich der Qualität verwischen bewegte Strukturen durch das Mischen der Bilder. Hinzu kommt noch eine erkennbare Unschärfe.

Adaptiv

Die am weitesten entwickelte und aufwendigste Methode ist das adaptive Deinterlacing. Hierbei werden für die Bearbeitung eines Halbbildes auch die vorangehenden und nachfolgenden Halbbilder mit genutzt. Dann wird eine detaillierte Bewegungsanalyse durchgeführt, um auf dieser Basis Bildteile mit geringen Bewegungen über das einfache Weave-Verfahren zu ergänzen. In diesen Bereichen sind keine Kammartefakte zu erwarten. Für die anderen, bewegten Teile wird nun versucht, diese möglichst verlustfrei aus anderen Halbbildern zu rekonstruieren bzw. zu interpolieren. Dies Verfahren ist naturgemäß sehr rechenintensiv und außerdem benutzen zuverlässige adaptive Deinterlacer auch komplexe Algorithmen. Schwächen in diesem Rechenwerk schlagen sich sofort in der Bildqualität nieder. Gleichzeitig ergibt auch nur ein hochwertiges Eingangssignal ein gutes Ergebnis, da Bildstörungen wie z. B. ein „Rauschen“ selbst gute Deinterlacer scheitern lassen können. In diesen Fällen kommt man mit den oben skizzierten Verfahren oftmals weiter.

Motion Compensation

Die heutigen Deinterlacing-Verfahren beruhen auf Bewegungskompensation (Motion Compensation). Hierbei werden im ersten Schritt die bewegten Bildelemente identifiziert und anschließend versucht, das erste und zweite Halbbild zur Deckung zu bringen. Im Folgeschritt werden dann die Bilder mittels Weave kombiniert. Hierdurch werden die Kammartefakte des reinen Weave-Verfahrens vermieden. In der Regel wird diese Methode noch mit adaptiven Filtern kombiniert, so dass hierbei das beste Ergebnis aller Verfahren erreicht wird. Dies sehr aufwendige Verfahren ist zurzeit der Standard im TV-Bereich. Nur bei im Niedrigpreissegment liegenden LCD-Monitoren, die auf der PC-Monitor-Technik basieren, wird noch unkompensiertes Deinterlacing genutzt.

Ein bisschen Organisation

Bei meinen ersten Versuchen (Videokamera, TV-Aufnahmen) fielen die Ergebnisse etwas verstreut auf die Platte, so dass ich in kürzester Zeit die einzelnen Daten suchen musste. Shit happens. Also habe ich mir mal ein paar Minuten Zeit genommen und folgende Struktur entworfen:

```
~/Video/test/
      /tmp/
      /scripte/

/daten/filme/kamera/
          /kino/film01/
          /kino/film02/
          /kino/film01.dv
          /kino/film01.mpeg
          /mandvd/
          /sonstiges
          /test/
          /tmp/
          /tv
```

Zur Erläuterung:

Meine 120 GB sind im Wesentlichen in `/`, `/home` und `/daten` partitioniert, wobei /home aus Sicherungsgründen nur ca. 10 GB groß ist. Das Home-Verzeichnis (/home/Video) ist für mich das Arbeitsverzeichnis meiner Video-Spielereien, die Daten-Partition ist der Arbeitsort für *Kino*, TV-Aufnahmen und auch Lager- bzw. Zwischenlagerort für die fertigen Video, da die meisten dann für den Rest der Familie auf dem Server abgelegt oder als einfache Daten-CD/DVD gebrannt werden.

Jeder hat sicherlich seine eigenen Vorlieben, doch liefert eine vernünftige Struktur eine schnelle Übersicht und erleichtert das Wiederfinden. Außerdem lassen sich so wichtige Daten einfach und schnell sichern. Grundsätzlich ist zu beachten, dass das Thema Video, wie auch schon an einigen Beispielen aufgezeigt, mit großem Speicherbedarf gleichzusetzen ist. Ehe man sich versieht ist die Platte voll!

Erstes Resümee I

Ich für meinen Teil habe gelernt und versuche hier auch zu vermittelt, dass das Thema Video nicht mal so eben abzuarbeiten ist. Im Gegensatz zu „normalen“ Dateien (Daten, Bilder, Audio) liegt hier eine deutlich komplexere interne Struktur (plus diverse Formate) vor, die bei Bearbeitung und Wiedergabe zu beachten ist.

Zum zweiten hat man es mit den unterschiedlichsten Quellen (TV, Video-Kamera, Video-DVD, VHS-Video usw.), den verschiedensten „Zielen“ (Wiedergabedatei, DVD, CD, Internet usw.) und mindestens zwei Typen von Wiedergabegeräten (PC, DVD-Player) zu tun.

Der übliche Ansatz „Oh, ist das ein tolles Bild, Musikstück, Gedicht schicke mir bitte eine Kopie!“ bzw. „Gebe mir doch bitte die ersten zehn Seiten deines Aufsatzes als Datei!“ lässt sich in dieser Einfachheit auf Videos nicht so ohne weiteres übertragen. Vor diesem Hintergrund ist die Komplexität und Optionsvielfalt der Programme leicht nachzuvollziehen, man schaue nur einmal auf das Ergebnis von

```
ffmpeg -h,
```

oder das Doku-Wiki zu *Avidemux*.

Ich hoffe allerdings, dass dieser Artikel den Dschungel etwas lichtet und die erste, schmale Wegstrecke freimacht. Denn trotz allem habe ich für mich das landläufige Vorurteil, Linux und Multimedia passen nicht zu einander, widerlegt.

Schon bei meinen ersten Schritten ist zu sehen, dass Linux und Multimedia, speziell die bewegten Bilder, zwei Dinge sind, die gut zueinander passen. Selbst mir, dem relativen Anfänger und Gelegenheits-Tux, ist es gelungen, die wichtigsten Grundfunktionen aus meiner subjektiven Sicht praktikabel zum Laufen zu bringen.

Eine weitere Erkenntnis in technischer Sicht war, dass ein Wechsel des Videoformats gleichzeitig einen zeitintensiven Komprimierungsprozess bedeutet, der auch immer einen Qualitätsverlust mit sich bringt. Daher sollte man sich genau überlegen, für welchen Zweck die Videorohdaten aufbereitet werden sollen. Mein primäres Ziel ist die Video-DVD mit einem MPEG-2-Container (mpeg2, AC3) als Zielformat, so dass dies mein „Arbeitsformat“ wurde, zumal auch meine TV-Aufnahmen in diesem Format vorliegen.

Die (Schnaps-)Idee, meine Erlebnisse parallel für MagDriva festzuhalten, brachte den riesigen Vorteil mit sich, dass ich gezwungen war, alles zu dokumentieren.

So sind die vielen Kleinigkeiten, die man nur selten durchführt, festgehalten und ich habe es dann parat, wenn ich mal wieder ins multimediale Geschäft eintauche. Außerdem hat die ganze Geschichte mein Linux-Wissen vertieft, nicht nur bzgl. Multimedia (z.B. die ganze Codecs-Geschichte) sondern auch zu allgemeinen Themen (Scripte, MySQL usw.), die am Rande auftauchten.

Hier auch noch einmal vielen Dank an alle Forums-User, die mir mit Rat und Idee zur Seite gestanden haben und ohne deren Hilfe meine Realisierung und damit auch dieser Artikel nicht zustande gekommen wäre.

Mit mehr Einsatz, Wissen und Übung ist der Kreativität kaum eine Grenze gesetzt. Vielleicht ist der eine oder andere Leser auf den multimedialen Geschmack gekommen und überrascht die MagDriva-Gemeinde demnächst mit weiteren multimedialen Geschichten. Ich würde mich sehr freuen.

Epilog

Falls jemand zu den beschriebenen Sachverhalten, Aktivitäten und Abläufen Verbesserungsvorschläge und /oder Anmerkungen hat, würde ich mich über eine Rückmeldung **sehr** freuen und dies im nächsten Teil des Artikels nachtragen. Für Rückfragen stehe ich natürlich via Forum, PN und /oder Email zur Verfügung.

Meine urpmi-Quellen

Für meine Updates und Nachinstallationen nutze ich *urpmi*

(*urpmi –auto-update* bzw. *uprmi Pgm*)

und habe dafür standardmäßig folgende Quellen eingebunden:

main_release,
main_update
contrib_release,
contrib_update,
mud-free,
plf-free,
plf-nonfree

Hinzu kommt noch die Quelle „lokales“ (*~/rpms/*) in der ich Pakete ablege, die ich per CD, DVD oder Download finde.

Links

[1] LinuxUser

[2] EasyLinux

[3] LinuxUser 2006.09: Bedrohte Spezies, s. 36l Aufnahme TV

[4] Fa. Reichelt

[5] EasyLinux 01/2007: Videoschnitt mit Kino, S. 22

Homepage Programme

[a] http://www.kinodv.org

[b] www.avidemux.org