Peter Zöller-Greer

Multi Media Systeme
Grundlagen
und
Anwendungen

Multi Media Systeme

Von der Film- und Musikindustrie bis zur Computer-Grafik und Computer-Games reicht die Palette der multimedialen Computeranwendungen. Vorliegender Kompaktkurs gibt einen Überblick über die wichtigsten Technologien.

Nachkolorierung von s/w-Bildern

Prof. Dr. rer. nat. Peter Zöller-Greer
ist Mathematiker und unterrichtet die Fächer Künstliche Intelligenz, Software-Engineering und Multi Media Systeme an der FH Frankfurt am Main-University of Applied Sciences.

9 783981 163926
Preis: 22,- Euro

ISBN 978-3-9811639-2-6

Die Reihe *Wissen & Praxis >kompakt<* nimmt sich komplexer Themen an und versucht diese so einfach wie möglich, beschränkt auf das Wesentliche, für Studium und Praxis gleichermaßen geeignet darzustellen.

Composia
Verlag
www.composia.de

Multi Media Systeme
Zöller-Greer
Composia Verlag

Peter Zöller-Greer

Multi Media Systeme

Grundlagen und Anwendungen

Mit einer Einführung in die Projektplanung, Datenkompression und Datenformate, digitale Bildverarbeitung, Audio- und Video-Verarbeitung, eLearning, Internet, Computer Games und Animationen

Reihe *Wissen & Praxis >kompakt<*

Composia
Verlag

Peter Zöller-Greer

Multi Media Systeme

Grundlagen und Anwendungen

Mit einer Einführung in die Projektplanung, Datenkompression und Datenformate, digitale Bildverarbeitung, Audio- und Video-Verarbeitung, eLearning, Internet, Computer Games und Animationen

Prof. Dr. Peter Zöller-Greer
studierte nach Abschluss einer Berufsausbildung als Physiklaborant (BASF-AG, Ludwigshafen/Rh.) von 1975-1981 Mathematik (Diplom) und Theoretische Physik an den Universitäten Siegen und Heidelberg. Er promovierte an der Universität Mannheim über eine approximationstheoretische Lösung eines Problems aus der Quantenmechanik und arbeitete zunächst als Systemanalytiker bei Brown Boveri Reaktor GmbH und danach als DV-Referent bei ABB Mannheim, bevor er 1989 die Geschäftsführung der Firma Composia GmbH in Mannheim übernahm. Bereits während dieser Tätigkeiten arbeitete er als freier Dozent an verschiedenen Hochschulen. Seit 1993 ist er Professor am Fachbereich Informatik und Ingenieurwissenschaften der FH Frankfurt am Main – University of Applied Sciences. Seine Lehr- und Arbeitsgebiete sind Künstliche Intelligenz, Software-Engineering und Multimedia-Systeme.

Bibliographische Information der Deutschen Bibliothek:
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Daten sind im Internet über http://dnb.ddb.de abrufbar.

2. Auflage

ISBN 978-3-9811639-2-6

Inhalt:

Vorwort

Eigentlich gibt es Multi Media Systeme schon immer. Bereits in der Antike wurden Theaterstücke aufgeführt, ein Beispiel für Audio-Visuelle Präsentationen. In unserer Zeit sind lediglich die Präsentationstools anders: Bild und Ton müssen nicht mehr live auf einer Bühne für einen begrenzten Zuschauerkreis erfolgen, sondern sind via TV, DVD-Player, Computer und Internet der ganzen Welt zugänglich, und das oft „on Demand". Doch damals wie heute werden immer die zwei gleichen Ziele dabei verfolgt: Unterhaltung (d.h. auch gleichzeitig Geld verdienen für irgendjemanden) und Lernen (womit man auch Geld verdienen können soll). Da der Mensch mehrere Sinne hat, mit denen er seine Umwelt wahrnimmt, liegt es natürlich nahe, möglichst all diese Sinne gleichzeitig anzusprechen. Gerade im Bereich des eLearning verspricht man sich viel davon, denn je mehr Sinne angesprochen werden -so jedenfalls die Theorie- je besser kann man sich etwas merken. Dieses Buch stellt einen Kompaktkurs darüber dar, wie man solche multimedialen Systeme entwickelt. Nach Klärung einiger Definitionen (Kapteil 1) werden Möglichkeiten zur Projektplanung aufgezeigt (Kapitel 2). Um Multi-Media-Systeme konkret entwickeln zu können, ist die Kenntnis der wichtigsten Kompressionsverfahren und Datenformate unerlässlich (Kapitel 3). In der Praxis gibt es glücklicher Weise gute Hilfsmittel zur Erstellung und Bearbeitung der wichtigsten Multimedia-Komponenten, nämlich digitale Bilder (Kapitel 4) sowie Audio- und Video-Anwendungen (Kapitel 5), wobei auch etwas Theorie hierbei von Nöten ist. Kapitel 6 beschäftigt sich mit Anwendungen wie Computerspielen und Animationen allgemein, während Kapitel 7 über eLearning- und Internetapplikationen informiert. Kapitel 8 schließlich zeigt ein Beispiel für viele andere Spezialanwendungen - das Nachkolorieren von Schwarzweißfilmen.
Manche Beschreibungen finden sich gleich in mehreren Kapiteln, aber unter verschiedenen Blickwinkeln. Dies würde gemacht, dass man nicht laufend hin und her blättern muss, wenn man ein zusammenhängendes Sachgebiet nachschlagen will.
Zu jedem dieser Bereiche gibt es eigene, z.T. sehr umfangreiche Lehrbücher, daher können die hier präsentierten Kapitel das jeweilige Gebiet natürlich nur einführend behandeln. Auch wird sicher jemand monieren, dass dies und das fehle, wie z.B. UMTS und vieles andere ach so „wichtige". Es sei daher schon jetzt eine pauschale Entschuldigung dafür abgegeben, denn ein so globales Gebiet wie Multi Media kann in einem einzigen Buch nicht erschöpfend abgehandelt werden, schon gar nicht in einem „Crash-Kurs" wie dem vorliegenden.
Besonderer Dank geht an den Mathematiker und Dipl.-Informatiker Matthias Wetzer, Dozent für Informatik am Berufsförderungswerk der Stiftung Rehabilitation in Heidelberg, für das Erstellen von Grafiken und Beschreibungen über Datenformate und Komprimierungstechniken in Kapitel 3.
Ich möchte außerdem an dieser Stelle meiner lieben Frau Diana für ihre unermüdliche Geduld mit mir danken. Dank auch unseren „Cratchits", die mir einen neuen Blick auf das Wesentliche gaben.

Im Januar 2010

Peter Zöller-Greer

1. Einführung

Der menschliche Körper verfügt über viele Sinne (es sind nicht nur 5). Ziel und Zweck von Multi Media Systemen ist es, soviel Sinne wie möglich gleichzeitig über eine computergesteuerte Anwendung anzusprechen. Dazu zählt eine Laser-show an Silvester über dem Brandenburger Tor in Berlin genauso wie eine eLearning-Anwendung im Internet. Dies zeigt, dass für uns in diesem Buch natürlich gewisse Einschränkungen gemacht werden müssen. So sind Laser-Shows nicht Gegenstand dieses Werkes. Überhaupt ist die Steuerung solcher multimedialer Hardware-Komponenten wie Laserkanonen, Disco-Kugeln oder Leinwände für Rockkonzert-Aufführungen etc. in diesem Kompaktkurs weitgehend ausgeklammert. Wir kümmern uns mehr um Multimedia-Anwendungen, die auf dem heimischen PC ablauffähig sind. Doch auch diese Einschränkung reicht noch nicht aus, denn z.B. allein über Computergame-Animationen kann man ein dickes Buch schreiben. Genauso über Audio- und Videoproduktionen. Daher können wir naturgemäß diese Dinge nur anschneiden, aber eben doch so, dass der Leser zumindest in die Lage versetzt wird, die theoretischen Grundprinzipien zu verstehen um praktisch wirklich etwas produzieren zu können. Manche sind der Meinung, dass diese Theorien z.B. für das Schneiden eines Videoclips nicht besonders wichtig ist. Spätestens wenn man aber eine Videoaufnahme nach dem Schneiden als MPEG2-File neu rendert, kann es sein, dass man sich über plötzlich auftretende „Klötzchen“ (auch *Artefakte* genannt) oder über plötzliches „Ruckeln“ wundert. Ein Ziel dieses Buches ist es, dass man *versteht* woran so was liegt, um dann daraus eine nutzbringende Lösung dieses Problems zu finden. Dies ist fast immer möglich, doch dazu muss man eben ein gewisses Verständnis der Funktionsweise solcher Renderprozesse haben.
Ein Wort zur Hardware des PCs. Früher waren Multimedia-Anwendungen den MAC-PCs vorbehalten, und viele schwören noch heute darauf. Und es ist wahr, MACs sind i.d.R. sehr stabil. Doch auch Windows ist heutzutage gut geeignet, aber man muss eine stabile Konfiguration schaffen, was nicht immer einfach ist. Alle hier vorgeführten Multimedia-Werkzeuge sind auf einem Windows-Rechner gelaufen, und es werden hierzu einige Tips gegeben. Auf jeden Fall sollte man über einen flotten Rechner mit viel Festplattenspeicher und mit soviel RAM wie möglich verfügen, sowie einer Grafikkarte, die selbst über möglichst viel eigenen Grafik-RAM verfügt. Letzteres ist gerade für das Schneiden von Videos wichtig, denn wenn es bei der Wiedergabe „ruckelt“, so kann das einfach an der Grafikkarte liegen und nicht unbedingt am Video selbst (was man herausfinden kann, in dem man das Video mal am TV oder einem anderem, schnelleren PC betrachtet).

Bevor wir allerdings in die Praxis gehen, sind gewisse Grundlagen unerlässlich, einerseits, um eine gemeinsame „Sprache“ zu sprechen und andererseits um ein gewisses Verständnis und „Gefühl“ für die ganze Problematik zu haben.
Wir beginnen daher zuerst mit einer Definition des Begriffs „Multi Media System“. Das Wort „Multi“ steht bekanntlich für „viele“ und das Wort „Media“ für „Vermittler“. Man könnten den Begriff „Multi Media“ auch landläufig als „vielfältiges Massenkommunikationsmittel“ übersetzen. Im Sinne der Informatik definieren wir daher:

Definition 1.1 (**Multi Media System**)
Ein *Multimedia-System* ermöglicht das Zusammenwirken verschiedner Medientypen (wie Texte, Bilder, Grafiken, Audio, Video und Animationen), wobei multimediale Information erzeugt, bearbeitet, gespeichert, transportiert und präsentiert werden kann.

Eines der schwierigsten Probleme und damit Gegenstand andauernder Forschung ist die Präsentation, die Speicherung und der Transport multimedialer Information. Diese Information ist nämlich sehr umfangreich, und es zielt alles darauf ab, die anfallenden Datenmengen zu reduzieren, sei es für das Internet oder die DVD. So kommt es, dass das Thema Datenkompression einen äußerst wichtigen Stellenwert einnimmt. Daher wurde diesem Thema auch ein eigenes Kapitel in diesem Buch gewidmet. Fast alle Probleme z.B. beim Abspielen eines Videos, seien es Artefakte, Ruckler, eingefrorene Bildsequenzen etc. rühren von Problemen mit den Kompressionsalgorithmen her.
Den Preis, den man für effektive Kompressionsalgorithmen zahlt, ist die Rechengeschwindigkeit. Und diese wiederum hängt u.a. von schneller Hardware ab, und diese wiederum ist teuer. Gerade Audio- und Video-Dateien sollen mindestens so schnell übertragen oder von einem Datenträger ausgelesen (bzw. auf ihm gespeichert) werden können, dass sie in „Echtzeit“ angesehen/angehört (oder abgespeichert) werden können. In diesem Zusammenhang ist der Begriff des *Streamings* sehr wichtig:

Definition 1.2 (**Streaming Media**)
Unter *Streaming Media* verstehen wir aus einem Computernetzwerk empfangene und gleichzeitig wiedergegebene Audio- und Videodaten. Den Vorgang der Übertragung selbst nennt man *Streaming* und gestreamte Programme werden als Livestream bezeichnet.

Wir werden im Kapitel 7, wo es u.a. um Internetanwendungen geht, darauf noch genauer eingehen. Doch der Grund, warum wir diese Definition hier schon bringen, ist der, dass Streaming eine wichtige Eigenschaft ist und die in Kapitel 3

betrachteten Datenkompressionsverfahren zielen letztlich alle darauf ab, neben platzsparender Speicherung einen ebenso platzsparenden Transport und die gleichzeitige „Live-Präsentation“ multimedialer Information zu ermöglichen.
Zum besseren Verständnis der Repräsentation multimedialer Information wird an dieser Stelle noch kurz ein Überblick über die Technologie der verschiedenen Hardwarekomponenten gegeben.

Multimedia-Technologien
Das zweifellos wichtigste Medium zum Betrachten multimedialer Information ist der PC-Monitor. Dieser wird von einer Grafikkarte versorgt, welche die an die Ausgabeeinheit gesendete Information in vom Monitor ansehbare Information umwandelt. Für Multimedia-Systeme ist es sehr ratsam, dass die Grafikkarte möglichst viel eigenen RAM besitzt. Eine Karte mit z.B. 256 MB RAM „erleichtert“ die RAM der PC-eigenen CPU eben um diesen Speicherbereich.
Bezüglich des Monitors werde ich, da Röhrenmonitore am Aussterben sind, hier nur auf LCD-Monitore eingehen. Und auch da möchte ich nur einige mehr praktische Dinge ansprechen, die für das Verständnis evtl. auftretender Problemfelder notwendig sind.

Pixel und Dots
Bekanntlich sind digitale Bilder aus Bildpunkten (Pixel) zusammengesetzt. Die übliche Schreibweise für deren Anzahl im Monitor ist *Spalten×Zeilen*, z.B. *1024×768*. Dies ist eine *absolute* Angabe und sagt nichts über die Größe des Monitors aus. Dies gilt auch für Digitalkameras. Wie groß letztlich der Chip ist, auf den das optische Bild fällt, ist eigentlich egal (wenn auch nicht ganz), wichtig ist die Pixel-Anzahl des Sensors. Das Wort Pixel ist ein Kunstwort, entstanden aus Picture element. Wir werden uns in Kapitel 4 noch genauer mit Rastergrafiken (zu diesem Thema gehören Pixel) auseinandersetzen. Ein Pixel ist dimensionslos (also weder rund noch quadratisch) und enthält außerdem Grau- bzw. Farbwerte (z.B. rot/blau/gelb). Physikalische Dimension bekommt ein Pixel erst bei der Darstellung. Z.B. bei einem 15-Zoll-Bildschirm mit einer Auflösung von 1024×768 misst ein Pixel etwa 0,3 Millimeter. Die maximal mögliche Pixel-Auflösung eines Bildschirms wird in *pixel per inch* (ppi) angegeben, das wäre dann eine *relative* Auflösung (hier also bezogen auf Inch). ppi ist in der Regel zu unterscheiden von *dots per inch* (dpi). Während bei Monitoren und Scannern oder Digitalkameras beides synonym verwendet wird, ist bei Ausgabegeräten normalerweise nur von dpi die Rede. Im Gegensatz zu Pixel haben Dots nämlich ein reale Ausdehnung, d.h. bei Dots handelt es sich immer um eine relative Angabe in Bezug auf z.B. Inch. Bei Ein- und Ausgabegeräten (z. B. Scanner, Drucker, Bildschirme, Belichter usw.) gibt die relative Auflösung die Dichte der Bildpunkte an. Bei Bilddateien gibt die relative Auflösung an, mit

welcher Dichte die Bildpunkte auf einem Ausgabegerät wiedergegeben werden sollen. Statt der Dichte lässt sich auch die Größe eines einzelnen Bildpunktes bzw. die Dicke einer Linie oder Zeile angeben.
Eine Auflösung von 1200 dpi horizontal und 600 dpi vertikal könnte z.B. einer Punktgröße von $21\frac{1}{6} \times 42\frac{1}{3}$ µm entsprechen. 1200 dpi horizontal bedeuten, dass sich 1200 Punkte in der Horizontalen auf 2,54 cm verteilen. Demnach hat ein Punkt in der Horizontalen eine Kantenlänge von

$$2{,}54 \text{ cm} / 1200 = 0{,}00211(6) \text{ cm} = 21\tfrac{1}{6} \text{ µm}.$$

Da die Auflösung in der Vertikalen nur 600 dpi beträgt, ist hier ein Punkt deutlich „länger“, nämlich

$$2{,}54 \text{ cm} / 600 = 0{,}0042(3) \text{ cm} = 42\tfrac{1}{3} \text{ µm}.$$

Daraus ergibt sich eine Gesamtfläche eines einzigen Punktes von

$$21\tfrac{1}{6} \text{ µm} \times 42\tfrac{1}{3} \text{ µm} = 896\tfrac{1}{18} \text{ µm}^2 .$$

Der Unterschied zwischen Pixel und Dots besteht also darin, dass Pixel in Bilddateien genau genommen nicht darstellbar sind, sondern nur durch Zahlenwerte (Helligkeit, Farbe) definiert werden. Dots hingegen können sowohl eingabeseitig (Scanner) als auch ausgabeseitig (Monitor, Drucker) über ihre Größe definiert werden, wobei der Dot bei einem Scanner unterschiedliche Helligkeiten analog erfasst und dann im A/D-Wandler in digitale Werte (also Pixel) umwandelt. Bei vielen Druckverfahren hingegen kann ein Dot nur entweder schwarz oder weiß sein (bzw. eine Farbe komplett oder gar nicht darstellen). Durch sog. Halbtonverfahren werden dann Halbtöne nur simuliert, d.h. durch ein Raster mal dicke oder dünnere Punkte erzeugt. Deswegen braucht ein Drucker häufig wesentlich mehr dpi um die gleiche Bildqualität zu erzeugen wie eine vergleichsweise geringe dpi-Auflösung am Monitor.

LCD-Monitore
Ein *Flüssigkristallbildschirm* oder eine *Flüssigkristallanzeige* (englisch *liquid crystal display*, kurz **LCD**), ist ein Bildschirm bzw. ein Display (Anzeige), bei denen spezielle Flüssigkristalle, die die Polarisationsrichtung von Licht beeinflussen können, zur Darstellung von Zeichen, Symbolen und Bildern genutzt werden. Die Bildpunkte werden über eine Matrix angesteuert, d.h. Zeilen und Spalten sind getrennt ansprechbar, so dass an deren „Schnittpunkt“ der gewünschte Bildpunkt aktiviert werden kann. Man unterscheidet dabei *aktive* und *passive* LCD. Aktive Displays erzeugen selbst Licht, fungieren also als Licht-

quelle, während passive LCD lediglich ihren Reflexionsgrad, d.h. ihre Lichtdurchlässigkeit ändern. Am populärsten sind zur Zeit sog. TFT-Displays. TFT steht für *Thin-Film-Transistor*. Sie zählen zu den aktiven LCD und jeder Bildpunkt wird dabei durch einen dahinter liegenden Transistor zum Leuchten gebracht. Gute Geräte übertreffen dabei meistens die Helligkeit herkömmlicher Monitore deutlich.
Ein wichtiger Maßstab ist hierbei noch das Kontrastverhältnis. Es gibt an, wie viel mal heller ein „eingeschalteter" Bildpunkt gegenüber dem „ausgeschalteten" ist. Passive LDC kommen meistens nicht über ein Kontrastverhältnis von 15:1 hinaus (d.h. ein angeregter Bildpunkt leuchtet 15 mal heller als ein nicht angeregter), während aktive LCD derzeit ein Kontrastverhältnis bis zu einigen Tausend zu Eins leisten können.
Das Digital Visual Interface[1] (DVI, zu unterscheiden vom Digital Video Interleave-Format, siehe Kapitel 3) ist eine Schnittstelle zur digitalen Übertragung von Videodaten. Im Computer-Bereich entwickelte sich DVI zu einem Standard für den Anschluss von hochwertigen TFT-Bildschirmen an die Grafikkarte eines PCs. Im Bereich der Unterhaltungselektronik gibt es Fernseher, die über einen DVI-Eingang Signale von digitalen Quellen, etwa PC oder DVD-Player, verarbeiten.
Durch die vollständig digitale Übertragung ergeben sich Qualitätsvorteile gegenüber der Verbindung mit einem VGA- oder SCART-Kabel. Da VGA- bzw. SCART-Kabel ein analoges Signal führen, fallen hier zwei unnötige Signalkonvertierungen an: Von digital nach analog am Videoausgang und zurück von analog nach digital im Monitor.
Für die eigentliche Datenübertragung nutzt DVI den Standard TMDS (Transition-Minimized Differential Signaling). Dabei hängt es von der zu übertragenden Datenmenge (Videobandbreite) ab, ob eine einfache (Single-Link) oder doppelte (Dual-Link) TMDS-Verbindung erforderlich ist. Die maximale Auflösung bei Single-Link-Kabeln ist 1600 × 1200 Pixel (UXGA) bzw. 1920 × 1200 (WUXGA) wenn Grafikkarte und Monitor reduced blanking unterstützen. Bei Dual-Link-Kabeln berechnet sich die Auflösung über die Formel

X-Auflösung × Y-Auflösung × Refreshrate × (1 + Totzeitverluste in %) = Bandbreite

Bei höherer Auflösung sinkt die Wiederholfrequenz. Dies ist bei TFT-Monitoren weniger kritisch, solange es sich nicht um Bewegtbilder handelt.
Es gibt TFT-Displays mit Auflösungen bis 3840 × 2400, welche entweder über einen oder zwei Dual Link-Anschlüssen (je Dual Link 330 Megapixel/s) betrieben werden können.

[1] vgl. auch http://de.wikipedia.org/wiki/Digital_Visual_Interface

Soundkarten
Nach dem Monitor (und natürlich der dazugehörigen Grafikkarte) stellt die Soundkarte das zweitwichtigste Medium im Computer dar. In Kapitel 5 wird auf Audioverarbeitung explizit eingegangen, doch diese kann natürlich nur mit einem geeigneten I/O-Gerät erfolgen. Heutzutage sind Grafik- und Soundkarten i.d.R. „On Board", also auf dem Motherboard, können aber immer auch zusätzlich installiert werden. Wer keinen Wert auf Dolby-Surround o.ä. legt, kommt mit der On-Board-Soundkarte meistens gut hin. Wer jedoch mehrere ein- oder Ausgabekanäle und/oder spezielle MIDI-Anforderungen hat, wie das im professionellen Tonstudiobereich der Fall ist, der sollte sich eine hochwertige Soundkarte zulegen, welche den benötigten Anforderungen entspricht. Näheres hierzu dann in Kapitel 5.

Scanner/Digitalkameras
Scanner und Digitalkameras verfügen über optische Sensoren, die analoge Signale mit Hilfe von Analog-Digitalwandlern (D/A-Wandler) in geeignete digitale Formate umwandeln. Solche Formatspezifikationen werden in Kapitel 3 und digitale Bildverarbeitung allgemein in Kapitel 4 näher betrachtet. Grundsätzlich gilt hier: Ein Gerät ist umso besser, je größer die *optische* Auflösung in dpi ist. Interpolierte Auflösungen sind irrelevant, da diese per Software erzeugt werden und auch nachträglich mittels geeigneter Bildbearbeitungssoftware vorgenommen werden können.

Beamer
Beamer, auch Videoprojektoren genannt, gibt es in mehreren Ausführungen: Beispiele sind *Röhrenprojektoren* und *LCD-Projektoren*. Während Röhrenprojektoren (ähnlich wie Röhrenmonitore) kaum noch im Einsatz sind, sind LCD-Projektoren sehr weit verbreitet. Letztere funktionieren im Prinzip wie Diaprojektoren, wobei anstatt des Dias ein kleines LCD-Display zwecks „Durchleuchten" zum Einsatz kommt. Die Lampen sind jedoch von recht kurzer Lebensdauer und relativ teuer, so dass sich ein Dauerbetrieb hier nur bedingt durchführen lässt.
Danben gibt es noch weitere Technologien wie z.B. *DLP-Projektoren* (DLP = Digital Light Processing), wobei hier die Bildpunkte von kleinen Spiegeln gesteuert werden, die durch integrierte Schaltkreise angesprochen werden und dadurch „kippen", so dass das Licht abgelenkt oder ausgegeben wird. DLP-Spiegel schalten bis zu 5000 Mal in der Sekunde. DLP-Projektoren werden häufig im Profi-Bereich (wie z.B. Kinos) eingesetzt.
In neuerer Zeit sind auch *LED-Projektoren* im Einsatz (LED = Light Emitting Diode), welche viele Nachteile z.B. von LCD-Projektoren aufheben. Sie arbeiten

ähnlich wie DLP-Beamer, wobei LED als bildgebende Lichtquelle eingesetzt werden.

Nachdem wir nun über einige Technologien gesprochen haben, können wir uns der Planung eines Multimedia-Projektes zuwenden. Wir werden sehen, dass hier hauptsächlich die Multimedia-Komponenten Text, Sound, Bild und Video in einer Software-Applikation geeignet zusammen spielen sollen, und dafür soll ein einfaches Modell zur groben Aufwandsabschätzung angegeben werden.

2. Multi Media Projektplanung

Während normalerweise im Software-Engineering die Projektplanung z.B. nach dem Wasserfall-Modell und dem Spiralmodell[2] vorgenommen wird, so stellt sich bei Multimedia-Projekten zusätzlich das Problem, dass Komponenten benutzt werden, die mit den Methoden des Software-Engineerings normalerweise nicht abgedeckt sind. Zum Beispiel wird dort nicht erfasst, welcher Aufwand (finanziell, personell und ressourcenmäßig) getrieben werden muss, um Grafik digital zu bearbeiten oder einen Videoclip nebst Vertonung mit Musik zu drehen. Ich möchte hier einen Ansatz vorstellen[3], welcher wenigstens einen Trend zum vermutlichen Aufwand angibt. Die Formeln hierfür sind zum größten Teil rein empirischer Natur und durch eigene Erfahrung in zahlreichen Projekten dieser Art entstanden. Das Ziel ist eine Erweiterung bestehender Planungsmodelle um gerade diese im Multimediabereich hinzukommenden Komponenten. Es wird nachfolgend eine Referenzliste erstellt, welche dem Planer helfen soll, an alles zu denken und die Aufwände und Ressourcen hierfür zu planen.

Planungs- und Entwicklungsphasen
Um die Möglichkeit zum Ausdruck zu bringen, dass einerseits einige der Multimediakomponenten eines zu entwickelnden Programmsystems gleichzeitig und relativ unabhängig voneinander entwickelt werden können und andererseits aber zu Beginn und zum Abschluss eines solches Projektes i.d.R. nur eine oder wenige Personen beteiligt sind, wurde für das nachfolgende Phasenkonzept eine 3-dimensionale Diamantform gewählt. Dieser „Diamant“ ist im Raum mit den Achsen „Zeit“, „Personal“ und „technische Resourcen“ platziert.

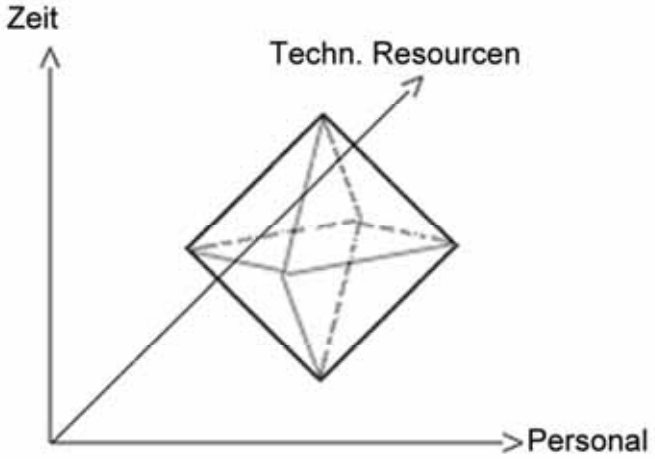


Abb. 2.1 Diamant-Modell

[2] Vgl. z.B. Zöller-Greer, P.: *Software Analyse und Design*, Composia Verlag 2007
[3] siehe auch Zöller-Greer, P., *Softwareengineering für Ingenieure und Informatiker*, Vieweg Verlag 2002, S. 16ff.

Die Personal-Zeit-Ebene

Nachfolgend werden Projektionen auf zwei Ebenen betrachtet. Es sei dabei ausdrücklich darauf hingewiesen, dass die Relationen der Komponenten- und Phasengrößen nachfolgend nicht in der richtigen Proportion wiedergegeben sind. Die tatsächlichen Größenverhältnisse sind natürlich von Art und Umfang des Gesamtprojekts bzw. der einzelnen Komponenten und Phasen abhängig und variieren daher von Projekt zu Projekt. Die Darstellung ist also rein qualitativ, nicht quantitativ zu interpretieren.

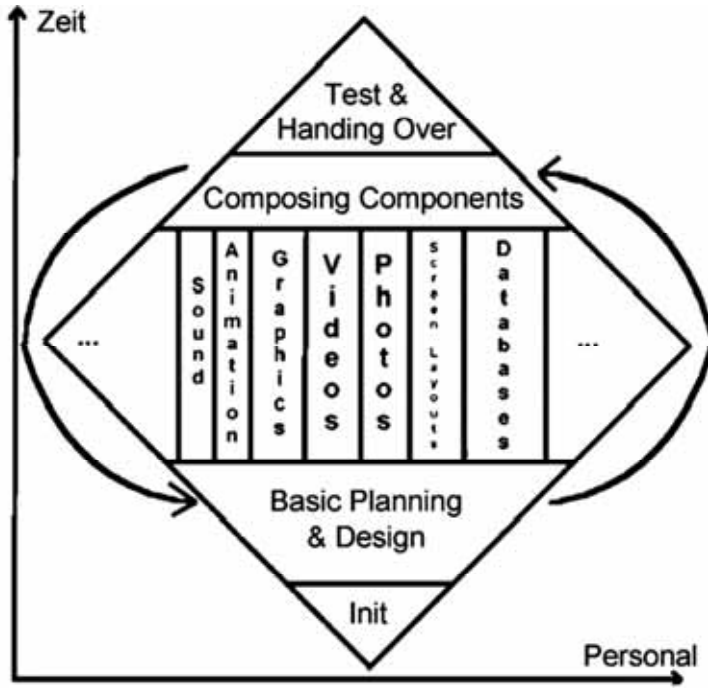


Abb.2.2 Personal-Zeit-Ebene

Die Zeitachse repräsentiert die *Projektphasen*, die Personalachse die von den beteiligten Entwicklern zu erstellenden *Projektkomponenten*. Dabei ist bezüglich der Personalachse die Breite des Diamant im Allgemeinen ein Maß für die Menge an Personaleinsatz (z.B. in Personen-Tagen). Grundsätzlich können natürlich auch alle Komponenten und Phasen von einer einzigen Person durchgeführt werden, was die Projektdauer entsprechend verlängert. Der Umstand, dass der Diamant in „die Breite“ geht, soll andeuten, dass in einer breiteren Phase mehr Entwicklungsaufwand nötig ist, welcher größtenteils unabhängig (und daher von mehreren Personen gleichzeitig) durchgeführt werden kann.

Die Init-Phase

Die Init-Phase repräsentiert die im Software-Engineering allgemein übliche Initialisierung eines DV-Entwicklungsprojekts. Wie üblich sind hier unter anderem zu finden:

- Ausgangslage
- Definitionen
- Zielvorstellung
- Vorgehensweise
- Angebot (incl. grobe kostenmäßige Aufwands- und Ressourcenabschätzung)

Die Phase Basic Planning & Design und ihre Einzelkomponenten
Auch diese Phase ist in den üblichen Phasenmodellen des Software-Engineerings zu finden. Allerdings unterscheidet sie sich bei Multi-Media-Projekten in ihrer praktischen Durchführung doch erheblich von üblichen DV-Entwurfsphasen. Auch in Hinblick auf die im Software-Engineering zunehmend eingesetzte Prototyping-Methode und der damit verbundenen phasenzyklischen statt einer linearen Vorgehensweise ist keine starre Trennung der drei Phasen „Basic Planning & Desgin", „Create Components" und „Composing Components" geben, sondern es handelt sich hier um einen zirkulären Prozess, bei dem die einzelnen Phasen sukzessive durch mehrfache Iteration sich der in der Init-Phase beschriebenen Zielvorstellung nähern. Nach wie vor sollte jedoch in der Entwurfsphase möglichst viel Detail-Planungsarbeit stecken, so dass in den darauf folgenden Iterationsschritten nur noch Korrekturen, aber nicht gänzlich neue Systemteile geplant und entwickelt werden müssen.
Neben den „klassischen" Inhalten des DV-Fachkonzepts und des DV-Entwurfs sollte ein besonderes planerisches Augenmerk in der Systemspezifikation gelenkt werden auf:

<u>Sound-Komponenten</u>
Festlegung von gesprochenen Texten
Festlegung der verwendeten Musik
Notwendige Ressourcen für Texte/Musik (Tonstudio, Sprecher, Musiker)
Kostenabschätzung dieser Ressourcen
Urheberrechtliche Kostenabschätzung (GEMA etc.)
Festlegung, wo welche Texte/Musik in der Applikation vorkommen

<u>Animations-Komponenten</u>
Festlegen, welche Animationen wohin kommen
Festlegen der Entwicklungswerkzeuge für die Animationen
Aufwandsplanung zur Erzeugung der Animationen

<u>Grafik-Komponenten</u>
Festlegen, welche Grafiken wohin kommen
Festlegen der Entwicklungswerkzeuge für die Grafiken

Aufwandsplanung zur Erzeugung der Grafiken

Video-Komponenten
Festlegung, welche Video-Clips wohin kommen
Notwendige Ressourcen zur Erzeugung der Clips (Video-/Filmstudio, Darsteller)
Kostenabschätzung dieser Ressourcen
Urheberrechtliche Kostenabschätzung (GEMA etc.)

Fotografische Komponenten
Festlegung, welche Fotos wohin kommen
Notwendige Ressourcen zur Erzeugung der Fotos (Fotostudio, Fotografen)
Kostenabschätzung dieser Ressourcen
Urheberrechtliche Kostenabschätzung

Die Komponenten *Screen-Layouts* und *Databases* sind zu planen wie bei allen betrieblichen Informationssystemen üblich.

Die Phase Composing Components
Eine besondere Bedeutung kommt der Planung der Zusammenführung aller zuvor genannten Komponenten für die Phase „Composing Components“ zu. Diese Phase sollte von möglichst wenigen Personen (am Besten von nur 1 Person, falls die Projektgröße dies zulässt) durchgeführt werden. Auf jeden Fall sollte ein entsprechendes Werkzeug, wie z.B. der Macromedia Director®, Flash® oder Toolbook® verwendet werden. Damit lassen sich die entwickelten Einzelkomponenten (Video, Animation, Grafik, Fotos, Schaltflächen, Screens etc.) häufig per Maus im Drag- und Drop-Verfahren auf eine sog. Bühne (Stage) ziehen. Im Macromeida Director Studio beispielsweise eröffnet sich dem Entwickler der Composing-Components-Phase eine solche Bühne, auf der der Entwickler die Rolle eines Regisseurs spielt. Er arrangiert die Einzelkomponenten in einem zeitlichen Ablauf, welcher durch eine Zeitachse die genaue Positionierung einer oder mehrere Multi-Media-Komponenten ermöglicht. Über Schaltflächen können dabei auf andere Komponenten verzweigt und so eine Interaktionen des späteren Anwenders mit der Applikation ermöglicht werden (was bisher in einem Bühnenstück natürlich nicht möglich ist).

Die Phase Test & Handing Over
Diese Phase ist wieder mit den traditionellen Methoden des Software-Engineerings realisiert.[4]

[4] siehe z.B. Zöller-Greer, P.: *Software Analyse und Design*, Composia Wächtersbach 2007

Die Ressourcen-Zeit-Ebene

Für die Entwicklung der jeweiligen Multimedia-Komponenten sind gewisse technische Ressourcen erforderlich, welche nachfolgend näher beschrieben werden (vgl. Abb. 2.3). Der Einsatz dieser Ressourcen sei so verstanden, dass als *Ergebnis* der Verwendung der Ressourcen-Komponenten immer eine *Datei* herauskommt, also ein Datenfile, der in dem vom Multimedia-Composing-Tool entsprechend verarbeitbaren Datenformat vorliegt.

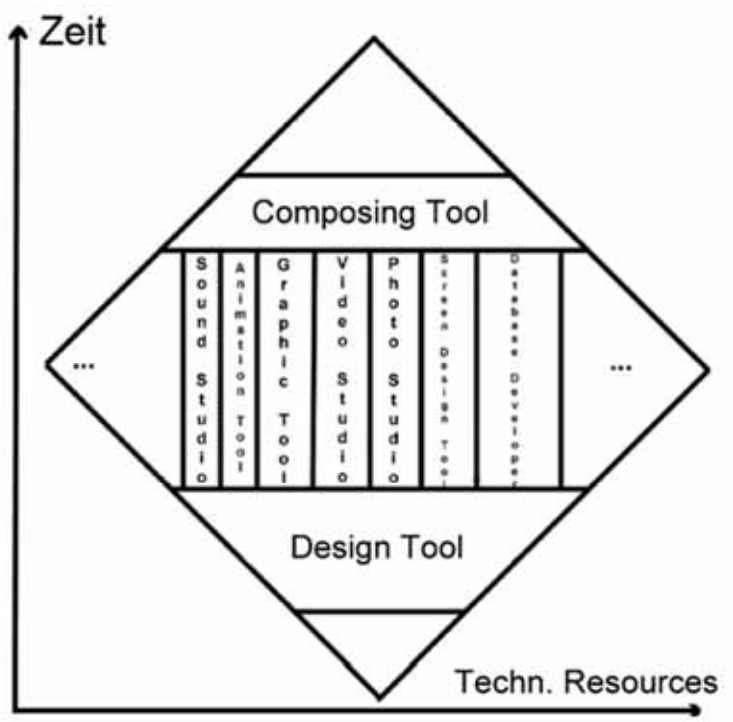


Abb. 2.3 Ressourcen-Zeit-Ebene

Nachfolgend werden die jeweiligen Ressourcen näher erläutert.

Resource Design-Tool & Composing-Tool

Diese beiden Ressourcen werden deshalb zusammengefasst, weil sie in der Regel mit dem selben Werkzeug durchgeführt werden (das ist jedenfalls dringend empfehlenswert). Dieses Werkzeug muss die Anforderungen erfüllen, die in den Personal-Zeit-Phasen *Basic Planning & Design* sowie *Compose Components* näher beschrieben sind.

Resource Sound Studio

Zur Erzeugung von Sprach- und/oder Musik-Dateien gibt es verschiedene Möglichkeiten. Sie hängen von dem Grad der Professionalität der Anwendung ab. Es wird hier davon ausgegangen, dass diese Dateien erzeugt werden müssen (alternativ könnte man z.B. für Musik auch bereits vorhandene Files benutzen; in

diesem Fall sollte unbedingt die urheberrechtliche Situation, z.B. bei der GEMA, geklärt werden)
Sollen Sprache und/oder Musik synchron zu bestimmten Bildern oder Videos laufen, so sind letztere zuerst fertig zu stellen. Unter rein technischen Aspekten sind die Kosten für die Miete eines Tonstudios und Toningenieurs zu berücksichtigen. Eine genaue Aufwandsschätzung ist hier dringend erforderlich und am Besten mit dem Toningenieur des Studios abzusprechen. So kann es z.B. durchaus sein, dass die Erzeugung eines Musikstückes von ca. 4 Min. Länge u.U. mehrere Studiotage in Anspruch nimmt. Kosten für evtl. Musiker/Sprecher sind ebenfalls einzukalkulieren.
Bei weniger professionellen Anwendungen und mit etwas Begabung kann auch ein PC als „Heimstudio“ herhalten. Ausgerüstet mit entsprechender Hard- und Software lassen sich auch hier mit wesentlich geringerem finanziellen Aufwand recht gute Ergebnisse erzielen.

Resource Animation Studio
Je nach Professionalität schwanken die Kosten für die Hardware zwischen 1.000,- EUR und mehreren 100.000,- EUR. Software aus diesem Bereich ist zwischen 500,- EUR bis mehreren 10.000,- EUR zu haben. Auch mit der unteren Preisklasse lassen sich bereits recht gute Ergebnisse erzielen, jedenfalls solange man im Bereich z.B. einer PC-Anwendung bleibt. Sehr aufwendig gestaltet sich dagegen die Entwicklungszeit solcher Animationen, speziell wenn es sich um 3-D-Animationen handelt. Bereits für wenige Sekunden sind häufig mehrere Mann-Wochen nötig (je nach Komplexität der gewünschten Animation).
Im Profi-Bereich ist –ähnlich wie bei den Tonstudios- die Anmietung von externen Animations-Studios zu empfehlen, deren Tagessätze mit denen eines Tonstudios vergleichbar sind bzw. darüber liegen können.

Resource Graphic Tool
Für gewöhnliche, nicht-animierte Bürografiken reicht i.d.R. ein einfaches Grafikprogramm aus, welches Daten aus einer Datenbank in Echtzeit als Grafik darstellt. Solche Tools sind für einige 100,- Euro erwerbbar. Bei MS-Office z.B. ist MS-Graph Bestandteil des Office-Paketes. Dieses Tool liefert mehrdimensionale Grafiken basierend z.B. auf MS-EXCEL®- oder MS-ACCESS®-Daten.

Resource Photo Studio
Hier sind 2 Phasen zu unterscheiden: Das eigentliche Fotografieren der Bilder und die anschließende digitale Bildbearbeitung. Für den ersten Fall sind die Kosten mit einem Fotografen abzusprechen. Die digitale Bildbearbeitung kann mit herkömmlichen Bildbearbeitungssoftwaren (z.B. Adobe Photoshop®) auf

einem schnellen PC erfolgen. Solche Nachbearbeitungen können auch mehrere Personen-Tage pro Bild in Anspruch nehmen, je nach Nachbearbeitungserfordernissen. Es ist darauf zu achten, dass die Bilder, sind sie konventionell auf Negativ oder Dia vorhanden, noch digitalisiert werden müssen.

Resource Screen Design Tools & Database Developer
Hier kommen „konventionelle" Entwicklungswerkzeuge zum Einsatz, wie sie seit langem bei der Entwicklung z.B. relationaler Datenbanken in Gebrauch sind. Diese Komponenten entsprechen denen bei der Entwicklung konventioneller Projekte. Es sind hier objektorientierte Prototypingmethoden vorzuziehen.

Aufwandsplanung einer Multimedia-Entwicklung
Aufgrund der vielen Komponenten sowie der Komplexitätsmöglichkeiten jeder Komponente ist eine generelle Aufwandsplanung sehr schwierig. Es wird daher versucht, nachfolgend eine sehr grobe Orientierungshilfe zu geben, welche keinesfalls als verbindliche Aufwandsschätzung z.B. für Festpreise o.ä. benutzt werden darf. Sie soll vielmehr eine unverbindliche Tendenz für den Entwickler sein, um so seinen Aufwand einigermaßen grob abzuschätzen („Daumengröße"). Naturgemäß ergibt sich der Gesamtaufwand als die Summe der Einzelaufwendungen. Damit gilt dann:

$$S_g = \sum_{\nu=0}^{n+1} \alpha_\nu s_\nu$$

wobei
n = Anzahl der Mutlimedia-Komponenten
S_g = Gesamtaufwand (Personen-Tage)
Für ν=1 bis n (=*Phase* „Einzelkomponenten-Entwicklung") bezeichnet
α_ν = Gewichtsfaktor für die ν-te Entwicklungskomponente
s_ν = Einzelaufwand für die ν-te Entwicklungskomponente (Personen-Tage)
Für $\nu=0$ sind die Aufwendungen der *Phase* „Basic Planning & Design" und für $\nu=n+1$ sind die Aufwendungen der *Phase* Composing Components einzusetzen.

Die Praxis hat gezeigt, dass es betreffs der Gewichtsfaktoren sinnvoll sein kann, zunächst errechnete Einzelaufwendungen mit einem Faktor zu multiplizieren, welcher eine realistische Korrektur ermöglicht. Im Allgemeinen ist $\alpha_\nu=1$ zu setzen (also keine Korrektur). Wenn aber ein mit der Entwicklung der ν-ten Entwicklungskomponente beauftragter Mitarbeiter z.B. aus Krankheitsgründen oder wegen anderer Arbeiten ausfällt, so kann es sinnvoll sein, hier bei Berechnung des Gesamtaufwandes den Gewichtsfaktor zu vergrößern. Der genaue

Wert kann von Mitarbeiter zu Mitarbeiter schwanken und muss aus der Erfahrung in dem jeweiligen Unternehmen ermittelt werden.

Phasenbezogene Aufwendungen

Initphase
Dieser Aufwand kann nicht formalisiert werden, er spielt im Gesamtprojekt in der Regel auch keine Rolle

Phase Basic Planning & Design
In dieser Phase erfolgt eine Planung der Einzelkomponenten, welche die jeweiligen Komponenten selbst lediglich als Black Boxes und ihrer „kompositorischen" Anordnung innerhalb des zu entwickelnden Gesamtsystems entwirft. Das detaillierte Design der Einzelkomponenten wird aufgrund der verfügbaren Werkzeuge direkt in der eigentlichen Entwicklungsphase dieser Komponente durchgeführt, da diese Werkzeuge i.d.R. während des Entwurfs bereits das entsprechende Modul selbst physikalisch erzeugen.
Erfahrungsgemäß liefert nachfolgende Abschätzung für diese Phase eine gute Annäherung an die Realität:

$$s_0 = \sum_{i=1}^{n} k_i$$

wobei

s_0 = Aufwand für die Phase „Basic Planning & Design"
n = Anzahl der Multimedia-Komponenten
k_i = Anzahl der Querverbindungen, welche eine Komponente i mit den restlichen Komponenten hat (soll z.B. die Soundkomponente in den drei Komponenten Grafik, Video und Databases vorkommen, so ist dieser Wert = 3). Jede Querverbindung wird dabei nur einmal gezählt

Phase Einzelkomponenten-Entwicklung
Nachfolgend werden erfahrungsmäßige Durchschnittswerte zugrunde gelegt. Es kann je nach Spezialfall vorkommen, dass einzelne Aufwendungen wesentlich höher liegen als hier angegeben. Dies ist entsprechend zu berücksichtigen. Hardware- oder externe Studiokosten (z.B. für Toningenieure, Musiker, Photografen etc.) sind nicht enthalten.

Komponente Sound

$$s_1 = \gamma \cdot \frac{t}{\omega}$$

wobei

s_1 = Aufwand für die Entwicklung der Sound-Komponente (in Personen-Tagen)

t = Zeitspanne für die Länge des Sounds in Minuten

γ = 1 Mann-Tag/Minute

ω = 3, falls Sound selbst erstellt, =30, falls Sound bereits fertig

Komponente Animation:

$$s_2 = ß \cdot t \cdot \gamma \cdot (\sum_{i=1}^{p} \psi_i)^2$$

wobei

s_2 = Aufwand für die Entwicklung der Animations-Komponente (in Personen-Tagen)

t = Zeitspanne für die Länge des Animation in Minuten

γ = 1 Mann-Tag/Minute Sound

$ß$ = Skalierungsfaktor, welcher je nach Realitätsgehalt der zu animierenden Objekte einen Wert zwischen 1 und 10 annehmen kann (z.B. „1“ für ein Strichmännchen und „10“ für ein real wirkendes 3-D-Objekt)

p = Anzahl der Objekte in der Animation

ψ_i = Anzahl der Extremitäten je Objekt („Sub-Objekte“) des i-ten Objekts (>0)

Komponente Graphics:

Es wird davon ausgegangen, dass mit Hilfe eines Grafik-Tools lediglich eine aus einer vorgegebenen Menge von Grafik-Vorlagen (z.B. Histogramme oder Kuchengrafik etc.) ausgewählt werden muss und die Daten für diese Grafik in Tabellenform vorliegen.

$$s_3 = \frac{g}{2}$$

wobei

s_3 = Aufwand für die Entwicklung der Grafik-Komponente (in Personen-Tagen)

g = Anzahl der Grafiken

Komponente Videos:
Die Entwicklung eines Video-Clips ist –je nach Komplexität- einer der aufwendigsten Tätigkeiten. Neben hohen Kosten für die Hardware-Ressourcen (incl. Kameramänner, Regisseur, Maskenbildner, Beleuchter, Schauspieler etc.) fällt dadurch auch der Zeitaufwand wesentlich ins Gewicht. In der Regel fallen die reinen Man-Power-Kosten dagegen i.d.R. gar nicht ins Gewicht. Oft ist mit Produktionskosten von mehreren –zig-Tausend bis Millionen Euro pro Clip-Minute zu rechnen. Nachfolgend wird davon ausgegangen, dass der Videoclip bereits in digitalem Format (z.B. als *.avi-Datei) vorliegt. Näherungsweise wird für die reine „Nachbearbeitung“ am System abgeschätzt:

$$s_4 = \gamma \cdot \frac{t}{\omega}$$

wobei
s_4 = Aufwand für die Nachbearbeitung der Video-Komponente (in Personen-Tagen)
t = Zeitspanne für die Länge des Videoclips in Minuten
γ = 1 Mann-Tag/Minute Videoclip
ω = 10, falls Video einfach (z.B. nur eine Person mit statischem Kamerabild), bis zu =1, falls aufwändige Videoproduktion mit häufigen Szenenwechsel und daher viel Editierarbeit

Komponente Photos:
Auch hier wird angenommen, dass die Photos bereits in digitalem Format vorliegen (z.B. als *.bmp-Datei). Nachfolgende Angaben beziehen sich demgemäss auf die reine Nachbearbeitung am System:

$$s_5 = \frac{m}{2}$$

wobei
s_5 = Aufwand für die Nachbearbeitung der Photos (in Personen-Tagen)
m = Anzahl der Photos

Komponente Screen-Layouts und Databases
Hier können gängige Abschätzungen wie z.B. die beim Projekt-Management für betriebliche Informationssysteme benutzt werden. Es werden hierfür daher keine eigenen Formeln aufgestellt.

Phase Composing Components:
Hier werden die Einzelkomponenten gemäß der Phase „Basic Planning & Design“ zusammengesetzt. Es zeigt sich, dass dieser Aufwand vergleichbar mit dem der Phase „Basic Planning & Design“ ist. Man kann hier also wieder ansetzen:

$$s_{n+1} = \sum_{i=1}^{n} k_i$$

wobei

s_{n+1} = Aufwand für die Phase „Basic Planning & Design“
n = Anzahl der Multimedia-Komponenten
k_i = Anzahl der Querverbindungen, welche eine Komponente i mit den restlichen Komponenten hat. Jede Querverbindung wird dabei nur einmal gezählt

Kommen alle 7 Multimedia-Komponenten zum Einsatz, so ergibt sich zusammenfassend die Formel

$$S_g = \alpha_0 \sum_{i=1}^{7} k_i + \alpha_1 \gamma \cdot \frac{t_1}{\omega_1} + \alpha_2 ß \cdot t_2 \cdot \gamma \cdot (\sum_{i=1}^{p} \psi_i)^2 + \alpha_3 \frac{g}{2} + \alpha_4 \gamma \cdot \frac{t_4}{\omega_4} + \alpha_5 \frac{m}{2} + \alpha_6 s_6 + \alpha_7 s_7 + \alpha_8 \sum_{i=1}^{7} k_i$$

wobei also zusammenfassend gilt:

S_g = Gesamtaufwand (Personen-Tage)
$\alpha_0 ... \alpha_8$ = Gewichtsfaktoren
γ = 1 Personen-Tage/Minute
t_1 = Zeitdauer für Sound (in Minuten)
ω_1 = 3, falls Sound selbst erstellt, =30, falls Sound bereits fertig
t_2 = Zeitdauer für Animation (in Minuten)
p = Anzahl der Animationsobjekte
$ß$ = Skalierungsfaktor, welcher je nach Realitätsgehalt der zu animierenden Objekte einen Wert zwischen 1 und 10 annehmen kann (z.B. „1“ für ein Strichmännchen und „10“ für ein real wirkendes Objekt)
ψ_i = Anzahl der Extremitäten („Sub-Objekte“) des i-ten Objekts (muss >0 sein)
g = Anzahl der Grafiken
t_4 = Zeitdauer für Video-Clips (in Minuten)
ω_4 = 10, falls Video einfach (z.B. nur eine Person mit statischem Kamerabild), bis zu =1, falls aufwändige Videoproduktion mit häufigen Szenenwechsel und daher viel Editierarbeit
m = Anzahl Photos

s_6 = Aufwand für Screen-Layouts (Personen-Tage)
s_7 = Aufwand für Database-Objekte (Personen-Tage)
k_i = Anzahl der Querverbindungen, welche eine Komponente i mit den restlichen Komponenten hat. Jede Querverbindung wird dabei nur einmal gezählt

Phase Test & Handing Over:
Hier wird wieder auf die gängigen Aufwände aus dem Projekt-Management gewöhnlicher Informationssysteme verwiesen. Es sollten allerdings etwas größere Testzeiten als üblich angenommen werden.
Die Vielfältigkeit multimedialer Anwendungen macht es recht schwierig, eine Detailplanung wie z.B. bei betrieblichen Informationssystemen zu machen. Ein wesentlicher Grund dafür liegt bei einem relativ hohen Anteil an künstlerischer Tätigkeit, wie z.B. die Erzeugung von Animationen, Grafiken, Video-Clips etc.; die zeitlichen Aufwendungen solcher Prozesse kann man kaum abschätzen. Dennoch sind natürlich die Auftraggeber interessiert, vor der Entwicklung eines solchen Systems einen Anhaltspunkt über den Aufwand zu haben. Das beschriebene Diamant-Modell kann dafür eine erste Planungshilfe darstellen.

Modellierung und Anwendungsentwurf
Es ist nicht einfach, hier ein allgemeines „Kochrezept" zu finden, denn je nach Zielgruppe und Zweck der Anwendung können hier stark von einander abweichende Konzepte sinnvoll sein. Wir beschränken und hier beispielhaft auf Präsentationen, die einen informativen Charakter haben, wie z.B. eLearning-Veranstaltungen oder Fernkurse. Immer hat man die gleichen Komponenten, welche in bestimmten Abschnitten synchron laufen, wie z.B. Videos mit Bild- oder Texteinblendungen zu vorgegebenen Zeitpunkten.

Drehbücher
Bei der Entwicklung multimedialer Software ist es häufig sinnvoll, den detaillierten Entwurf der Komponenten und vor allem deren Inhalte zunächst in Form eines zeitgesteuerter Ablaufentwurfs zur erstellen. Manchmal wird bei Multimedia-Entwicklungen dies ein „Drehbuch" genannt. Während im Filmgeschäft ein Drehbuch in erster Linie eine cartoonartige Darstellung mit Schlüsselszenen enthält, welche durch ergänzende Texte beschrieben werden, ist dies bei multimedialen Anwendung allenfalls für die Komponenten „Videoclip" und/oder „Animation" zu empfehlen.
Z.B. im Rahmen des eLearning sind Videoclips und Animationen dagegen oft nicht sehr szenenintensiv. Beispielsweise kann ein Videoclip eines Dozenten eingeblendet werden. Da hier kaum Szenenwechsel vorhanden sind, erübrigt sich hier ein detailliertes Drehbuch für diesen Clip. Auch Animationen sind

heutzutage mit Hilfsmitteln wie „Macromedia Flash“ so einfach zu erzeugen, dass für einfache Animationen auch hier eine Idee direkt am Computer zumindest zu Beginn schon grob umgesetzt werden kann und quasi als Prototyp dient.
Auf die „klassische“ Form der Filmdrehbücher wird daher hier nicht näher eingegangen und auf gängige Fachliteratur aus der Filmbranche verwiesen.
Allerdings, wenn man unter einem Drehbuch im Sinne einer Multimedia-Session eine „erweiterte“ Version eines Filmdrehbuchs versteht, so sind diese Art von Drehbücher hier näher zu betrachten. Im folgenden sei daher unter einem Drehbuch so eine erweiterte Multimediale Version verstanden. Es handelt sich dabei um ein höheres Abstraktionsniveau als bei Filmdrehbüchern.
Allgemein seien in einer Multimedia-Anwendung *n* multimediale Komponenten (auch Multi-Media-Objekte genant) im Sinne von Abb. 2.3 vorhanden. Diese Komponenten können nun theoretisch alle jeweils in *m* verschiedenen Bildschirmansichten, welche durch Schaltflächen erreicht und auch untereinander verknüpft sein können, vorkommen.
Eine Drehbuchplanung beginnt normalerweise zuerst mit der Planung der *m* Ansichten. Diese entsprechen zunächst den Masken einer gewöhnlichen Applikation. Zuerst wird dabei die Maskenhierarchie geplant, die grundsätzlich festlegt, welche Bildschirm-Masken in welcher Hierarchieebene (durch Mausklick auf eine Schaltfläche) zugänglich sind. Danach erfolgt die Planung der einzelnen Inhalte der Bildschirmmasken. Schließlich werden die multimedialen Komponenten jeder dieser Masken mit ihren Interaktionsmöglichkeiten entworfen. Unser Fokus richtet sich jetzt speziell auf die Komponentenplanung einer multimedialen Bildschirmmaske.

Theoretische Entwurfsmodellierung
Eine konkrete Maske beinhalte *n* multimediale Objekte (das können Videos, Sounds, Grafiken, Bilder, Animationen, Menüleisten oder Schaltflächen etc. sein). Im Gegensatz zu gewöhnlichen Applikationen haben wir hier zeitdynamische Elemente, welche ggf. eine endliche Lebenszeit innerhalb der Maske haben können. Deswegen ist eine Zeitabhängigkeit zu berücksichtigen. Wir definieren zunächst einen Objektvektor:

$$\vec{O}^m(\vec{o}) = \begin{pmatrix} O_1^m(\vec{o}) \\ O_2^m(\vec{o}) \\ \cdots \\ O_n^m(\vec{o}) \end{pmatrix}$$

Jedes $O_i^m(\vec{o})$, *i=1...n*, stellt ein konkretes multimediales Objekt des *m*-ten Bildschirms dar, wobei der Bildschirm *m* aus *n* solchen Objekten besteht. Solche Objekte können z.B. Videoclips oder Animationen sein.
Der Vektor $\vec{o}$ bezeichnet die Instanzen des *i*-ten Objekts der *m*-ten Bildschirmmaske. Die Dimension von $\vec{o}$ hängt von der Maximalzahl der möglichen Instanzen-Parameter der benutzten Objekte ab. Insbesondere ist es möglich, dass $\vec{o}$ selbst wiederum aus Bildschirmmasken, also aus (anderen) $\vec{O}^{\tilde{m}}(\vec{\tilde{o}})$ besteht. Rekursikonen sind ausdrücklich erlaubt. Dieser Spezialfall sei jedoch zunächst nicht weiter verfolgt.
Was aber häufig vorkommt, ist, dass $\vec{o}$ *komplex* in dem Sinne ist, dass es selbst aus mehreren multimedialen Objekten, die selbst nicht Bildschirmmasken sind, besteht. Dabei handelt es sich z.B. um die typischen multimedialen Komponenten wie Video (bzw. Animation), Text, Sound und/oder Grafiken sowie Fotos. Im einfachen Fall kann $\vec{o}$ aber auch schlicht eine Schaltfläche sein.

Beispiele für $\vec{o}$:

Schaltfläche: $$\vec{o} = \begin{pmatrix} x_{lo} \\ x_{ru} \\ Farbwert \\ Beschriftung \\ click() \end{pmatrix},$$

(*x* bezeichnet die „links-unten“ und „Rechts-oben“ Koordinaten im Bildschirm). Es können natürlich noch weitere Parameter wie „Style“, „Farbe der Schrift“ etc. hinzukommen.

Video-Clip: $$\vec{o} = \begin{pmatrix} x_{lo} \\ x_{ru} \\ t_a \\ t_e \\ \vec{n} \end{pmatrix}$$

Auch hier stellt *x* die „links-unten“ und „Rechts-oben“ Position im Bildschirm dar, *t* regelt die Start- und Ende-Zeit des Clips, und $\vec{n}$ ist eine Navigationsleiste speziell zum Navigieren des Video-Clips (ist Vektor, da die Leiste wieder aus Schaltflächen etc. besteht)
Es kann auch ein „Universalobjekt“ aus der Vereinigung aller möglicher Parameter aller vorkommenden Einzelobjekte gebildet werden, von denen die Einzelobjekte die jeweiligen Attribute ererben kann. So ein Superobjekt könnte dann wie folgt aussehen:

$$\vec{o} = \begin{pmatrix} x_{lo} \\ x_{ru} \\ t_a \\ t_e \\ Farbwert \\ Beschriftung \\ Click() \\ \vec{n} \\ Objekttyp \\ \dots \end{pmatrix}$$

Für einen Entwurf multimedialer Anwendungen sind also die Objekte $\vec{O}^m(\vec{o})$ vollständig zu bestimmen.

Wenn dies geschehen ist, empfiehlt sich allerdings eine etwas übersichtlichere Darstellung. Der sich aus den Objekten ergebende Tensor wird am besten in seine 2-dimensionalen Matrix-Projektionen zerlegt, und zwar so, dass der Zeitverlauf auf eine x-Achse projiziert wird und auf der y-Achse einzelnen Objekte sichtbar werden. So kann die Interaktion der einzelnen Multimedia-Komponenten übersichtlich dargestellt werden.

Praktische Entwurfsmodellierung
Ich empfehle für die zeitdarstellende Zerlegung des Tensors eine sog. „Schwimmbahnen-Darstellung“. Sie könnte z.B. so aussehen:

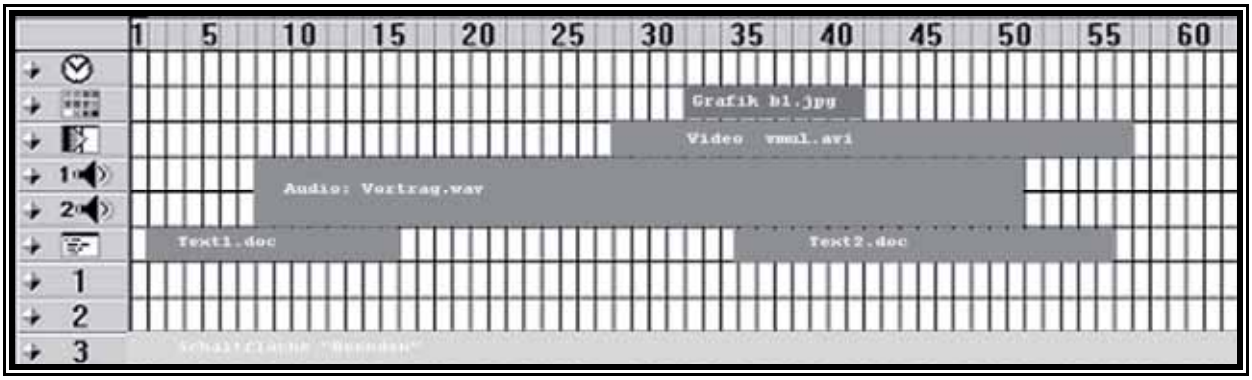


Abb. 2.4 Schwimmbahnen-Darstellung

Jede Spalte steht für einen Zeitabschnitt (z.B. Minuten/Sekunden/Frames) und jede Zeile repräsentiert eine multimediale Komponente eines Objekts (die i.d.R. selbst wieder Objekte sind). Die erste Spalte repräsentiert durch Text (oder Nummern) oder Symbole diese Komponenten. Der Balken aus der 3. Zeile könnte z.B. einen Videoclip repräsentieren, der nach 27 Sekunden ein-, und nach 57 wieder ausgeblendet ist. In der 6. Spalte sieht man Texteinblendungen, ganz unten eine Schaltfläche und so weiter.
Mit der Schwimmbahnendarstellung erhält man eine effektive Möglichkeit, zeitabhängige Bildschirmelemente und ihre Synchronisierung zu entwerfen.

Szenarien

Bei Multimedia-Anwendungen sind Szenarien ein äußerst wichtiges Hilfsmittel, denn diese stellen repräsentativ die Planung des gesamten Projekts dar. Es sei die skizzierte Methode an einem Implementierungsbeispiel als ein solches Szenario demonstriert. Dabei sollen aber auch allgemeine Implementierungsprinzipien vorgeführt werden.
Hier das Szenario: Es soll eine virtuelle Vorlesung „on demand“ für das Fach „Software Engineering“ entwickelt werden. Diese Vorlesung soll online über das Internet (und auch offline auf CD/DVD) verfügbar gemacht werden. Ein moderner Standard-Browser (z.B. Windows Explorer) soll die Veranstaltung aufrufbar machen.
Aus pädagogisch-didaktischen Gründen soll die Internet-Veranstaltung aus allen derzeit verfügbaren multimedialen Komponenten bestehen: Der Dozent ist sichtbar in einem Videoclip, ggf. ergänzt durch Einblendung sinnvoller Animationen; ein „Tafelanschrieb“ soll an geeigneten Stellen zusätzlich eingeblendet werden können, gleiches gilt für „Overheadfolien“ mit Grafiken und Bildern. Der/die Studierende soll darüber hinaus ein Skript in Textform downloaden und ausdrucken können, welches parallel mit der virtuellen Veranstaltung gelesen werden kann und ergänzende Hinweise enthält. Aber auch auf der Internetpage der Veranstaltung sollten sich Text-Seiten befinden, die abwechselnd vor oder nach einem Clip bearbeitet werden müssen, damit nicht nur ein Medium den

Zuhörer ermüdet. Bei den Clips sollen die Teilnehmer jederzeit durch eine Navigationsleiste den Clip anhalten oder Teile davon wiederholt ablaufen lassen können.
Der Dozent muss zunächst sein Drehbuch liefern. Es muss also ein zeitlicher und inhaltlicher Ablauf empfohlen werden. Die Inhalte plant der Dozent gemäß den Anforderungen seines Fachs. Für unser Implementierungsbeispiel für das Fach „Software-Engineering“ sieht die Entwurfsplanung der einzelnen zu bietenden Kapitel („Inhalt“) z.B. so aus:

Abb. 2.5 Lektionen

Die schwarzen Einträge stellen reine Textseiten („n.T.“ = „nur Text“) dar. Der Rest soll Video-Clips (des vortragenden Dozenten) sowie geeignete Text- und Grafik-Einblendungen beinhalten.
Auf die weitere Planung der reinen HTML-Text-Seiten soll hier noch nicht weiter eingegangen werden, sondern nur auf den uns interessierenden Teil der multimedialen Seiten.
Wie schon erläutert ist die Idee hier, einen möglichst realitätsnahen Unterricht „on demand“ zu simulieren. Es soll also der Dozent sichtbar sein (Video) und sein „Tafelanschrieb“ an geeigneter Stelle ein- und ausgeblendet werden, gleiches gilt für Overheadfolien oder Dias in Form von Bildern. Grundsätzlich

könnte man das alles in einem einzigen Videoclip unterbringen. Dem steht aber entgegen, dass dafür sowohl die Größe des Video-Fensters als auch die Auflösung so hoch sein müsste, dass ein reines Video-Internetstreaming hier nicht sinnvoll ist. Eine Alternative wäre, Videos von Texten und Bildern zu trennen und synchron ablaufen zu lassen. Dazu eignet sich z.B. die HTML-ähnliche Sprache „SMIL“ (Synchronized Multimedia Integration Language) der Firma Real Inc., welche von einem Real-Player problemlos ausgeführt wird. Denkbar wäre auch eine Flash-Animation.
Zunächst muss der Ablauf einer Lektion z.B. mittels einer Schwimmbahndarstellung geplant werden. Wenn dies geschehen ist, kann mit der Planung des Anwendungsdesigns begonnen werden.
Beispielsweise kann man sich folgende Bildschirmaufteilung vorstellen:

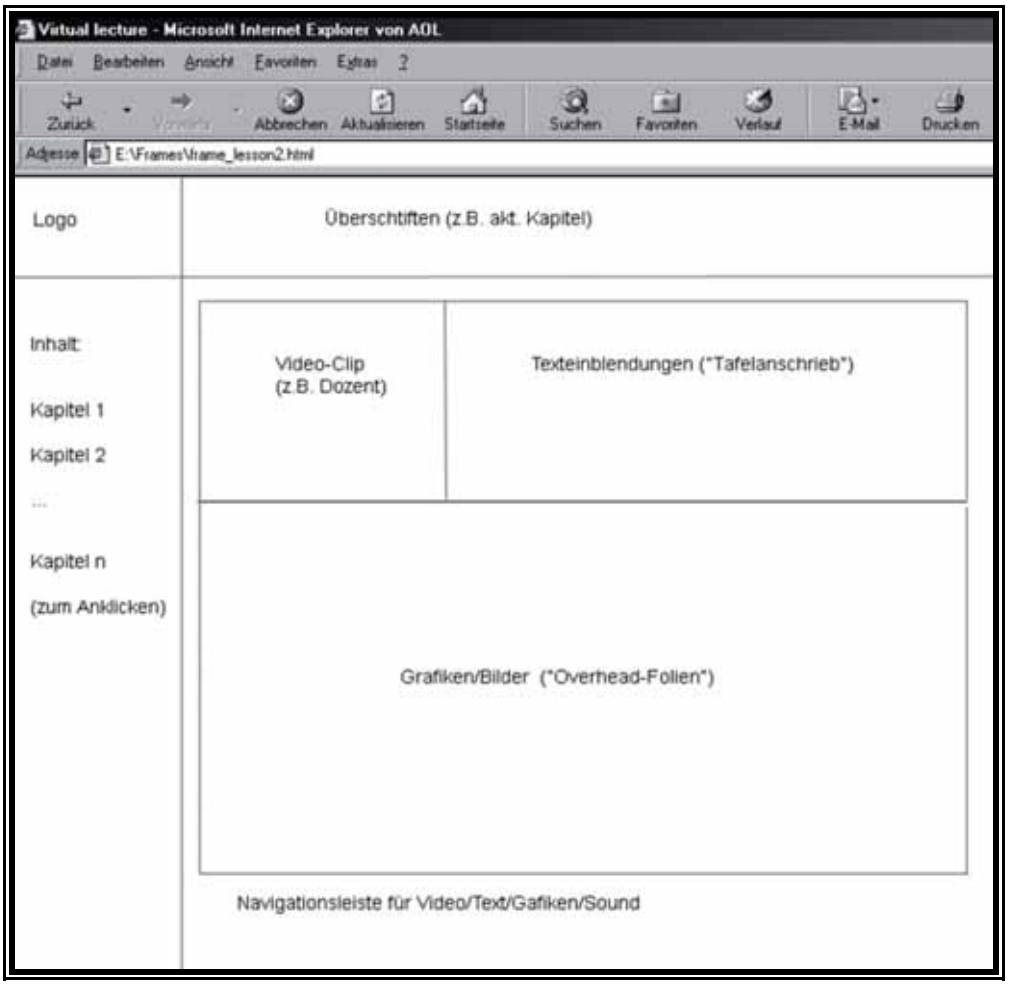


Abb. 2.6 Positionierung der multimedialen Komponenten

Die Lektionen (hier Kapitel genannt) sollen als Navigation links permanent zu sehen und anklickbar sein. Im Fenster „Video-Clip“ ist der Dozent zu sehen während er redet. Hier können natürlich auch sonstige Filme ablaufen. Rechts

daneben werden von Zeit zu Zeit Texte eingeblendet und im unteren Bereich ggf. Grafiken oder Bilder.
Hier ein Beispiel, wie sich ein Kunde das ganze vorstellt (Ausschnitt aus der virtuellen Vorlesung „Software Engineering“, mit SMIL erstellt, vgl. Kapitel 7):

Abb. 2.7 Positionierung der multimedialen Komponenten in der HTML-Page

Hier sieht man also einen Screen-Shot aus dem Kapitel „Methodische Ansätze“ des Faches Software-Engineering. Während der Dozent redet, wird die Definition eines „Systems mit geplanten Reaktion“ textuell eingeblendet und eine Gra-

fik zur Erläuterung angezeigt. Im unteren Bereich sieht man eine Video-Navigationsleiste, mit deren Hilfe man jede beliebigen Stelle der Lektion anfahren und wiederholen kann. Text- und Grafikeinblendungen werden dazu synchron angezeigt. Wichtig ist hier noch mal anzumerken, dass die Texte und Grafiken nicht Teil des Video-Clips sind, sondern die Texte z.B. als ASCII-Datei und das Bild als jpg-File vorliegen können.
Im Drehbuch, welches der Auftraggeber gewöhnlich gemeinsam mit dem Entwickler erstellt, wird jedes der Lern-Kapitel (linke Navigationsleiste in Abb. 2.7) entsprechend beschrieben, z.B. mit Hilfe von Skizzen für den gewünschten Bildschirmaufbau und mittels der „Schwimmbahnen“ um genau festzulegen, an welchen Stellen (zeitlich) Text- und/oder Grafikeinblendungen kommen sollen.
Wie dann die technische Realisierung erfolgen kann, wird im Kapitel 7 genauer beschrieben. Hier, in der Planungsphase, geht es lediglich darum, was ein Kunde gerne *hätte*. Zur tatsächlichen Realisierung bedarf es noch einiger technischer Kenntnisse, die in den nächsten Kapiteln erarbeitet werden.

Übungen zum Selbsttest

Es soll eine Teachware entwickelt werden, welche aus 10 Lektionen besteht, wobei jede Lektion jeweils aus einem ca. fünfminütigen (einfachen) Video-Clip besteht. Es wird für jeden Clip eine eigene Hintergrundmusik (gekauft) dafür eingespielt. Bei 4 der Clips ist eine ca. 10-sekündige Animation vorhanden, bestehend aus einem kleinen Männchen (1 Kopf, zwei Arme, zwei Beine, alle beweglich), das irgend wie agiert. Bei 3 Clips werden digitale Fotos und bei 1 Clip eine Grafik eingeblendet.
Welcher Aufwand zur Erstellung der Einzelkomponenten errechnet sich dafür nach Diamant-Modell?

3. Datenkompression, Formate und Codecs

Dieser Abschnitt beschäftigt sich mit etwas „Theorie“, welche zum effektiven Arbeiten und Entwickeln von Multimediakomponenten unerlässlich ist. Ich versuche dabei, so wenig Theorie wie möglich, aber doch soviel wie nötig, darzustellen. Viele praktische Probleme beim Bearbeiten und Erzeugen von multimedialen Anwendungen können durch ein wenigstens grobes Verständnis der wichtigsten Datenformate und deren Eigenschaften gelöst werden. Und gerade im Bereich Multimedia sind viele dieser Formate dadurch charakterisiert, dass dabei verlustfreie oder verlustbehaftete Datenkompression zum Einsatz kommt. Die Algorithmen, welche die so formatierten Daten auf einem Computer darstellbar und bearbeitbar machen, nennt man *Codecs*. Dieses Wort kommt von **Co**dieren/**Dec**odieren. Codecs stellen also eine Software-Schnittstelle zwischen einem Datenformat und i.d.R. dem Betriebssystem dar. Das Betriebssystem (z.B. Windows) kennt die Hardware und kann dann aufgrund des Codecs den entsprechend „aufbereiteten“ Datenstrom z.B. zur Grafikkarte oder Soundkarte schicken.

Doch was sind die wesentlichen Kennzeichen dieser Algorithmen? Dies soll nachfolgend für die wichtigsten Datenformate und Kompressionsverfahren kurz erläutert werden. Es wird natürlich kein Anspruch auf Vollständigkeit erhoben, und auch veralten solche Spezifikationen recht schnell. Daher sind nachfolgende Ausführungen ggf. auf Aktualität zu überprüfen. Auch sind die Verfahren hier teilweise nur im Überblick, d.h. qualitativ, beschrieben, wer sich da weiter einarbeiten möchte, dem sei entsprechende Fachliteratur empfohlen.

Standards zur Komprimierung für Einzelbilder, Bildfolgen und Videokonferenzsysteme wurden insbesondere entwickelt von der ISO (International Standards Organisation), der ITU (International Telecomunications Union) und der IEC (International Electrotechnical Commission). Daneben existiert eine Reihe von Techniken zur Komprimierung. Diese gehen in der Regel Hand in Hand mit den bestimmten Formaten. So ist z.B. ein jpeg-Bild einerseits das Ergebnis eines bestimmten, verlustbehafteten Kompressionsverfahrens und gleichzeitig besitzt es einen wohldefinierten „inneren“ Aufbau betreffs Header, Tags etc.; die vorgestellten Standards und Techniken im Überblick:

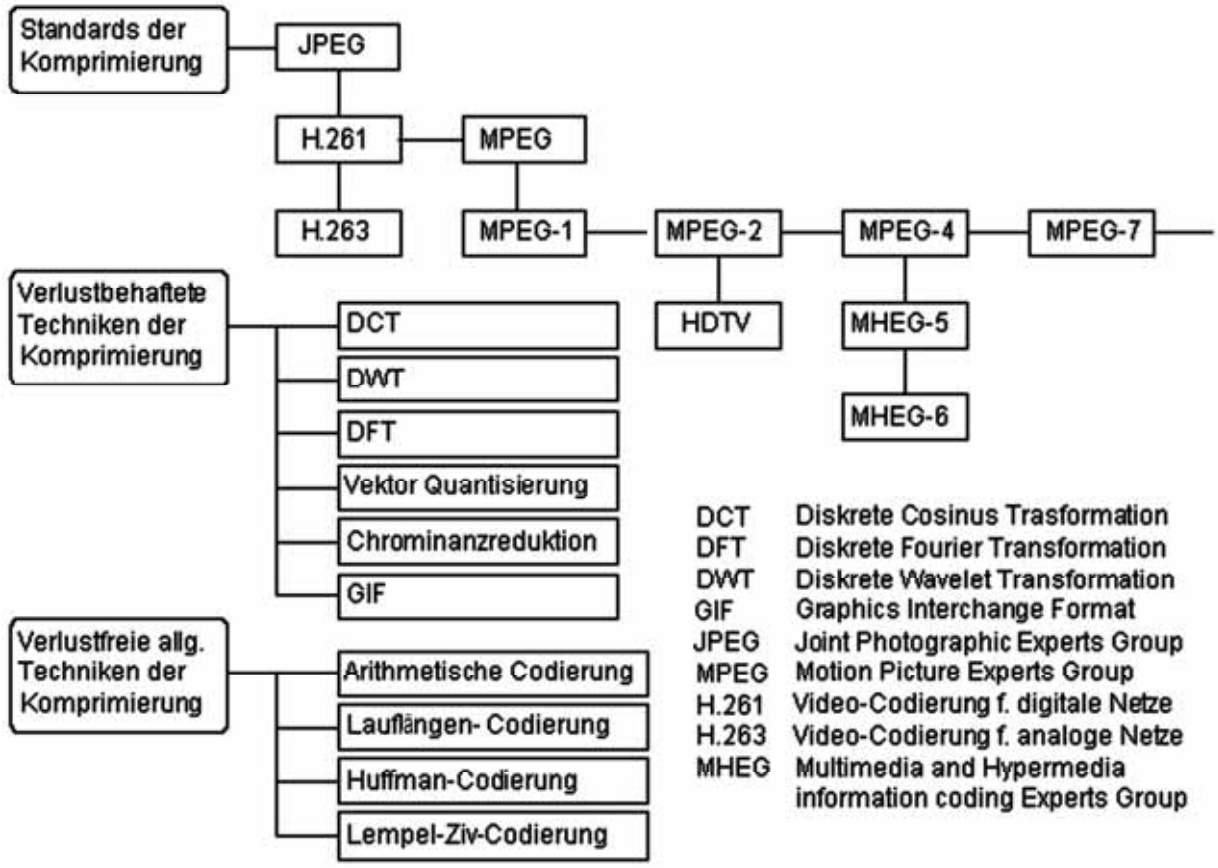


Abb. 3.1 Überblick über Standards zu Komprimierungsverfahren

Komprimierungsverfahren kann man folgendermaßen charakterisieren:

- **verlustfrei:** Originaldaten vollständig rekonstruierbar
- **verlustbehaftet:** Originaldaten nicht vollständig rekonstruierbar
- **intraframe:** zeitliche Redundanzen zwischen Bildern bleiben unberücksichtigt
- **interframe:** zeitliche Redundanzen zwischen Bildern werden berücksichtigt
- **symmetrisch:** Codierung und Decodierung erfolgt ungefähr gleich schnell
- **asymmetrisch:** Codierung dauert viel länger als Decodierung
- **Skalierung:** Bilder liegen codiert in mehreren Auflösungen vor
- **Echtzeit:** Verzögerung durch Codierung und Decodierung unter 150 ms

Entropie-Codierungen sind verlustfreie Techniken, die einzelne Symbole codieren. Source-Codierungen sind verlustbehaftete Techniken, die zwischen wichtigen und unwichtigen Informationen trennen. Die eigentliche Komprimierung erfolgt durch die Quantisierung, bei der als unwichtig erachtete Informationen weggelassen werden. Die meisten Verfahren bedienen sich der Source-Codierung und der meist danach durchgeführten Entropie-Codierung und stellen

eine Hybrid-Codierung dar.

Verlustfreie Entropie-Codierungen

Die mittlere Länge eines Codeworts eines Quellalphabets $A = \{a1,...,an\}$ ist definiert mit

$$L(A) = \sum_{i=1}^{n} p_i i \cdot L_i$$

mit den Wahrscheinlichkeiten p_i und den Codewortlängen L_i. Die Entropie wird für Codealphabete *m*-ter Ordnung definiert mit

$$L(A) = -\sum_{i=1}^{n} p_i i \cdot \log_m(p_i) = \sum_{i=1}^{n} p_i i \cdot \log_m(\tfrac{1}{p_i}),$$

Die Entropie gibt bei festen Auftrittswahrscheinlichkeiten eine größte untere Schranke für die Codewortlänge an und ist damit ein Effizienzmaß.

Lauflängen-Codierung

Bei der Lauflängen-Codierung werden aufeinanderfolgende gleiche Symbole zusammengefasst. Symbol und Lauflänge werden abwechselnd codiert. Es können mehrere Codierer verwendet werden, etwa um Symbole und Lauflängen einer separaten Entropiecodierung zu unterziehen.

Huffman Codierung

Für bekannte Häufigkeiten und damit Wahrscheinlichkeiten des Auftretens von Symbolen legt die Huffman-Codierung eine Codetabelle an, die jedem Symbol eine Bitfolge zuordnet. Die Bitfolgen werden als Präfixcode generiert, womit das Ende jeder Bitfolge ohne Trennzeichen erkannt wird. Die Huffman-Codierung verwendet Bitfolgen variabler Länge, häufige Symbole erhalten einen kurzen Code, seltene einen langen. Die Codierung hängt dabei von den zu codierenden Daten ab, die Codetabelle selbst muss jedes Mal aufgebaut oder adaptiert werden. Für die Decodierung günstig sind Codetabellen in Form von Binärbäumen, deren Blätter die Symbole und die von den Knoten ausgehenden Äste die Werte 0 und 1 repräsentieren. Liegt eine codierte Bitfolge vor, findet man das Zeichen, indem man ausgehend von der Wurzel für jede 0 zum linken Nachfolger wechselt und für jede 1 zum rechten, bis man zum Symbol gelangt. Bei der rekursiven Konstruktion erstellt man anhand der Wahrscheinlichkeiten einen binären Codierungsbaum, bei dem häufige Zeichen nahe der Wurzel stehen (kurze Bitfolgen), seltene weiter entfernt (lange Bitfolgen). Zur Konstruktion des Codierbaums generiert man für die beiden kleinsten Häufigkeiten jeweils einen Teilbaum, in dessen Verzweigungsknoten die Summe der beiden Wahrscheinlichkeiten eingetragen wird. Der Teilbaum gilt danach als ein Symbol mit der im Verzweigungsknoten abgelegten Wahrscheinlichkeit. Der Algorithmus

endet, wenn alle Symbole im Baum eingetragen sind. In Endknoten sind die Symbole hinterlegt, in Verzweigungsknoten die Wahrscheinlichkeiten.
Adaptive Varianten der Huffman-Codierung berechnen nicht den ganzen Baum neu, sondern tauschen lediglich einzelne Knoten aus. Für den erzeugten Huffman-Code gilt $H(A) \le L(A) \le H(A) + 1$. Die Entropie kann wegen der ganzzahligen Codewortlänge meist nicht erreicht werden. Für die Klasse der ganzzahligen Entropiecodierungen mit festen Auftrittswahrscheinlichkeiten ist die Codierung optimal und wegen der Komplexität $O(n)$ auch schnell. Das Vorgehen sei an nachfolgendem Beispiel illustriert (die Buchstaben A kommen mit 40% Häufigkeit vor, B mit 30%, C mit 10% und D mit 20% in einem Text vor):

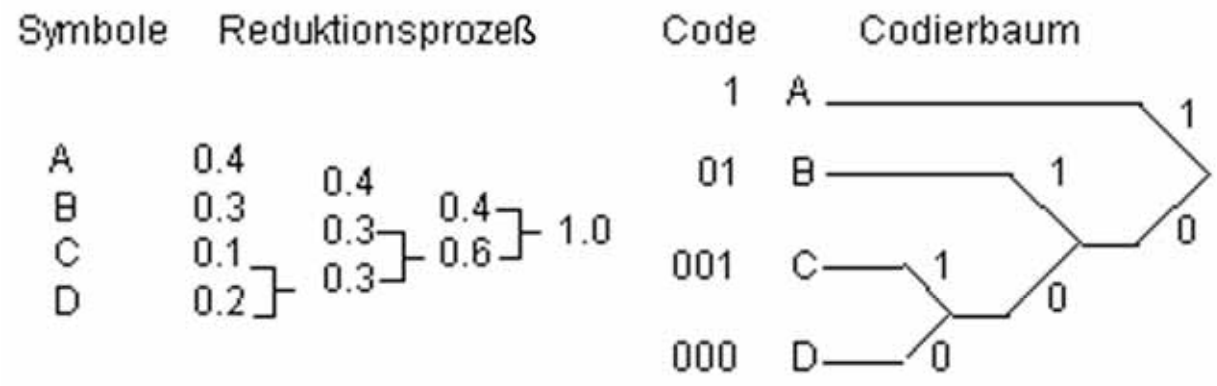


Abb. 3.2 Beispiel zur Huffman-Codierung

Arithmetische Codierung
Die arithmetische Codierung bildet Folgen von Eingabesymbolen mit gegebenen Auftrittswahrscheinlichkeiten auf Gleitkommazahlen aus [0,1] ab. Mit Kenntnis dieser Zahl und den Auftrittswahrscheinlichkeiten kann die Symbolfolge rekonstruiert werden. Da die arithmetische Codierung nicht mit ganzzahligen Codewortlängen arbeitet wie die Huffman-Codierung, kann die beste Entropie erreicht werden. Als Terminierungskriterium wird entweder die Anzahl zu decodierender Symbole mitgegeben oder ein Terminatorsymbol verwendet.

Lempel-Ziv-Codierung
Das Prinzip der von Lempel und Ziv entwickelten Codierung besteht darin, Symbolfolgen in einem Puffer für bereits übertragene Symbole zu suchen und geeignete Kanalsymbole zu übertragen.
Die Codierung beruht auf der Annahme, dass bereits erkannt Symbolfolgen wiederholt auftreten. Die Symbolfolgen sind für jede Quelle typisch und müssen jeweils adaptiv aufgebaut werden. Die längstmögliche Teilfolge wird gesucht und deren Anfangsposition und Länge übertragen. Da der Puffer für bereits übertragene Symbolfolgen beschränkt ist, arbeitet der Algorithmus zwangsläufig

adaptiv. Um die Suche nach Symbolfolgen zu minimieren, werden z.B. Suchbäume verwendet.
Neben der Codierung mit einem Puffer für beliebige Symbolfolgen entwickelt der Algorithmus die Codierung mit in einem Wörterbuch abgelegten Symbolfolgen. Es wird jeweils das längste passende Wort im Wörterbuch gesucht und seine Nummer übertragen. Mit jedem neuen Wort wird das Wörterbuch erweitert. Ist die meist beschränkte Größe des Wörterbuchs erreicht, werden keine neuen Wörter mehr aufgenommen. Das Wörterbuch kann dann übertragen und danach neu aufgebaut werden.

Differential Pulse Code Modulation
Die Differential Pulse Code Modulation (DPCM) versucht den Wertebereich der Eingabesymbole zu verkleinern, indem statt der Eingabewerte die Differenzen zu einem Referenzwert codiert werden. Dient bei der Folge 4,6,8,10,12 jeweils der vorhergehende Wert als Referenzwert, erhält man die Folge 4,2,2,2,2 und kann diese anschließend erfolgreich einer Lauflängen-Codierung unterziehen.
Die Übertragung auf Bildfolgen können zeitliche Redundanzen zwischen aufeinanderfolgenden Bildern zur Komprimierung nutzen. Diese Redundanzen entstehen beispielsweise durch Bewegungen von Objekten vor einem festen Hintergrund. Die Bewegungskompensation vergleicht zwei rechteckige eventuell verschobene Bereiche. Bei ausreichender Übereinstimmung werden diese DPCM-codiert und eventuell mit einem Translationsvektor versehen. Da hierbei entstehenden Nullwerte gestatten anschließend eine erfolgreiche Entropie-Codierung.

Verlustbehaftete Komprimierungstechniken
Hier sind also komprimierte Daten nicht vollständig zu rekonstruieren. Gerade im Bereich der Bildverarbeitung sind diese Techniken sehr effektiv, da häufig ausgenutzt wird, dass z.B. das menschliche Auge nicht in jedem Farbbereich alles gleich gut wahrnimmt. Aber auch in der Audio-Kompression, z.B. bei mp3-Dateien, werden solche Techniken eingesetzt. Hier wird ausgenutzt, dass das menschliche Ohr bestimmte Frequenzbereiche ohnehin schlechter wahrnimmt als andere, so dass solche „kaum hörbaren“ Frequenzen „wegkomprimiert“ werden.

Luminanz und Chrominanz
Luminanz bezeichnet die Grauwerte (also den s/w-Anteil eines Bildes) und Chrominanz (manchmal auch einfach nur Chromanz genannt) den Farbanteil. Jede Farbebene kann als Graustufenbild betrachtet und getrennt behandelt werden. Die RGB-Farbebenen können in die Luminanzebene Y und zwei Farbebe-

nen U und V konvertiert werden mit

$$Y = 0.299R + 0.587G + 0.114B$$
$$U = -0.169R - 0.331G + 0.500B$$
$$V = 0.500R - 0.419G - 0.081B$$

Für U und V sind auch die Bezeichner Cb und Cr gebräuchlich. Die Luminanz dominiert die menschliche Wahrnehmung gegenüber der eigentlichen Farbenwahrnehmung. Dies nutzt man für eine Reduktion der Bildinformation im Verhältnis 3:2 oder 2:1 und codiert im 4:2:2 oder 4:2:0 Format statt im 4:4:4 Format.

Diskrete Cosinus Transformation (DCT)
Grundlage der sich bisher durchgesetzten verlustbehafteten Komprimierungsverfahren ist die diskrete Cosinus-Transformation (DCT), eine besondere Form der diskreten Fourier-Transformation (DFT) bzw. der Fast-Fourier-Transformation (FFT). Grundidee ist die Konvertierung der Bilder aus der Zeit-Domäne in die Frequenz-Domäne, Quantisierung der Frequenzen und damit Unterdrückung der für den Bildeindruck unwichtigen hochfrequenten Anteile mit kleinen Amplituden. Eine Ausnahme bilden scharfe Kanten im Bild, die hohe Frequenzen erzeugen. Fallen diese bei der Quantisierung weg, entsteht der Eindruck eines verschwommenen Bildes. Durch die DCT selbst geht keine Bildinformation verloren, erst die Quantisierung führt zu einem Verlust. Je nach Grad der Quantisierung ergeben sich verschiedene Komprimierungsfaktoren, ohne Quantisierung liegen die Komprimierungsraten bei 15:1 bis 20:1.

Diskrete Wavelet Transformation (DWT)
Die Komprimierung mit Wavelets basiert auf der diskreten Wavelet-Transformation (DWT). Wahrend die DCT auf Bildausschnitte angewendet wird, transformiert die DWT das ganze Bild. Die Wavelet-Transformation besteht aus einer fortgesetzten Tiefpass- und Hochpassfilterung. Dazu wird zunächst in horizontaler Richtung tiefpass- (TP) und hochpass- (HP) gefiltert. Als Ergebnis erhält man 2 Teilbilder. Diese werden danach vertikal tiefpass- bzw. hochpassgefiltert. Man erhält 4 Teilbilder (TP/TP, TP/HP, HP/TP und HP/HP). Das TP/TP-Teilbild enthält die für die Wahrnehmung wesentliche Information. Dieses Teilbild kann fortgesetzt der DWT unterzogen werden. In DWT-basierten Komprimierungsverfahren werden die Teilbilder anschließend wieder verlustbehaftet quantisiert und mit verlustfreien Techniken weiter komprimiert. Die DWT liefert deutlich schärfere Kanten als die DCT.

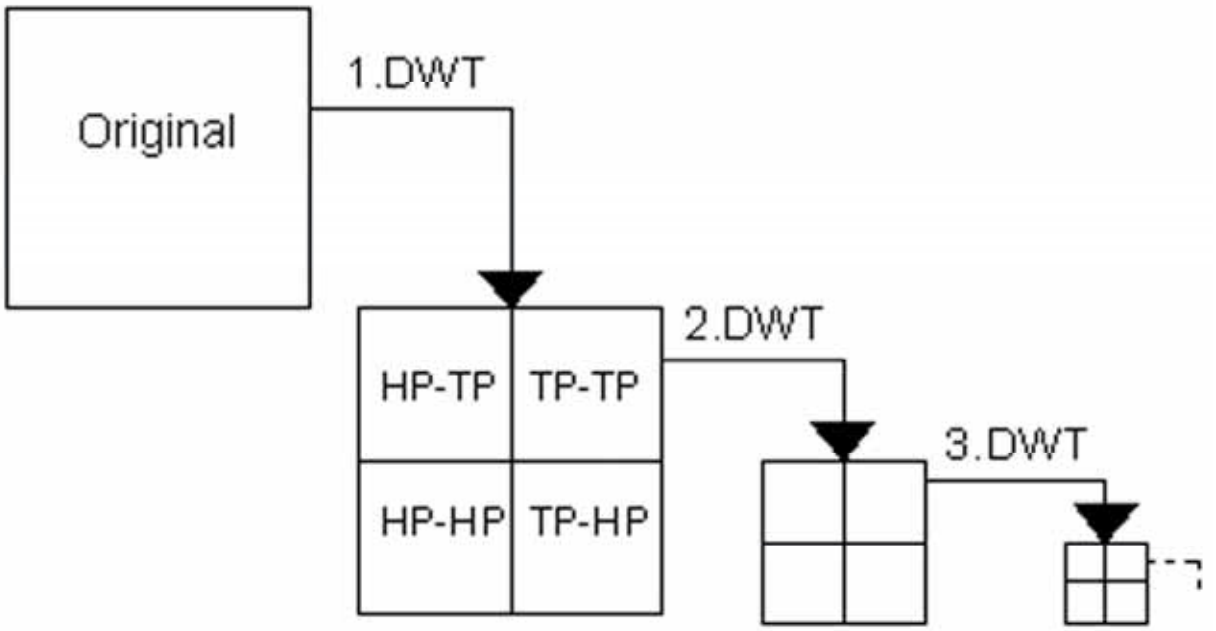


Abb. 3.3 Diskrete Wavelet-Transformation

Vektor-Quantisierung

Die Vektor-Quantisierung wird in vielen Produkten eingesetzt wie Indeo, QuickTime oder Video for Windows. Informationen können in Echtzeit mit Software in akzeptabler Qualität komprimiert werden. Ein Vektor ist eine Symbolfolge der Quelle. In einem Codebuch mit Symbolfolgen wird die dazu ähnlichste gesucht und der entsprechende Index als Kanalsymbol verwendet.
Bei der Decodierung wird der Index verwendet, um die entsprechende Symbolfolge aus einem identischen Codebuch zu erhalten. Das Codebuch muss möglichst klein gehalten werden, um kleine Indices zu erhalten. Andererseits muss das Codebuch so aufgebaut werden, das die Decodierung akzeptable Ergebnisse ohne allzu großen Zerfall bzw. Entstellung liefert. Um sowohl das Codebuch klein als auch die Ergebnisse akzeptabel zu halten, werden adaptive Methoden verwendet, wie sie Lempel und Ziv vorgestellt haben.

Diskrete Fraktal Transformation (DFT)

Bilder besitzen häufig selbstähnliche Teile, etwa Konturen, die durch geringe Veränderungen in Helligkeit, Kontrast oder Größe einander ähnlich gemacht werden können. Die Codierung erfolgt durch eine geeignete funktionale Darstellung selbstähnlicher Bildteile. Bei der fraktalen Transformation mit integrierten Funktionensystemen (IFS) wird ein Bild in Segmente zerlegt, die durch affine Transformationen approximiert werden. Diese affinen Approximationen benötigen wenig Speicherplatz und sind häufig fraktaler Natur. Fraktal transformierte Bilder können bei jeder Auflösung beliebig gezoomt werden und zeigen hier Ähnlichkeit mit Vektorgrafiken, störende Blockstrukturen wie bei DCT-basierten Verfahren treten nicht auf. Bei DFT basierten Verfahren beschränkt

sich die Quantisierung auf Kontrast und Helligkeit. Die auf fraktaler Transformation beruhenden Verfahren sind asymmetrisch, die Codierung dauert signifikant länger als die Decodierung.

Graphics Interchange Format (GIF89a)
Das Graphics Interchange Format (GIF) definiert ein Protokoll für die Online-Übertragung rastergrafischer Daten unabhängig von der Hardware der Erzeugung oder Darstellung mit bis zu 256 Farben. Die erste Version wurde 1987 als GIF87a definiert, 1989 die erweiterte Version GIF89a. Die verwendete variable Lauflängen-Codierung nach Lempel-Ziv-Welch (LZW) verwendet Codes variabler Länge zwischen 3 und 12 Bits, um Bitmuster der Originaldaten zu ersetzen. Bei der Codierung wird dynamisch eine Übersetzungstabelle aus Bitmustern erzeugt, die in den Originaldaten bisher erkannt wurden. Jedes neue Muster wird in diese Tabelle aufgenommen.
Anstelle des Bitmusters wird der gefundene Index in den Datenstrom aufgenommen. GIF stellt einen effizienten one-pass Codierer und Decodierer bereit, Decodierung und Darstellung kann gleichzeitig erfolgen. GIF89a erlaubt mehrere Bilder in einer GIF-Datei. GIF erhält scharfe Bildkanten und besitzt eine hohe Komprimierungsrate für Bilder mit wenigen Farben. Bei photorealistischen Bildern mit weichen Farbübergängen ohne allzu scharfe Kanten ist JPEG vorteilhafter.

DPCM und ADPCM
In der Audiotechnik wird die DPCM (Differential Pulse Code Modulation) auf Folgen PCM-codierter Abtastwerte (samples) angewendet. Nicht jeder Abtastwert wird mit voller Bitanzahl abgelegt, sondern nur die Differenz zum vorhergehenden Abtastwert. Dies reduziert den Wertebereich und damit die notwendigen Bits je Abtastwert.
Bei der ADPCM (Adaptiven Differential Pulse Code Modulation) wird der Abtastwert durch eine Konstante dividiert, mit der der Decodierer wieder multipliziert. Die Konstante passt der Codierer dem DPCM-codierten Signal an. Probleme bereiten dabei stark ansteigende Signalflanken (Slope Overload) oder ein großer Anteil hoher Frequenzen, was zu einer starken Quantisierung führt.

JPEG
JPEG, entwickelt von der ISO/IEC JTC1/SC29 WG10, ist der Standard zur Komprimierung einzelner Farb- oder Graustufenbilder, beschrieben in der ISO/IEC 10918-1. Grundidee ist die blockweise Konvertierung der Bilder aus der Zeit-Domäne in die Frequenz-Domäne, Quantisierung der Frequenzen und Unterdrückung hochfrequenter Anteile mit kleinen Amplituden. Die menschliche Wahrnehmung ist unempfindlich gegenüber hochfrequenten Bildteilen und

einer gröberen Quantisierung der Frequenzen. Zudem sind die Amplituden der hochfrequenten Anteile deutlich niedriger als der niederfrequenten Anteile. Dies zusammen erlaubt eine verlustbehaftete Reduktion der Bildinformation mit noch akzeptabler Reproduktion des Originalbildes. Neben der Standardversion (baseline system) existieren darüber hinausgehende Versionen (extended system).

Baseline System

Im Baseline-System werden die Bildgrenzen auf ein Vielfaches von 8 angepasst und die Pixel aus dem Intervall [0,2n-1] in den vorzeichenbehaftete Integerbereich [-2n-1,2n-1] transformiert. Bei n=8 Bit Farbtiefe entspricht das einer Subtraktion von 128. Das Bild wird in Blöcke von 8x8 Pixeln unterteilt, die mit der FDCT (Forward Discrete Cosinus Transformation) in die Frequenzdomäne transformiert werden.

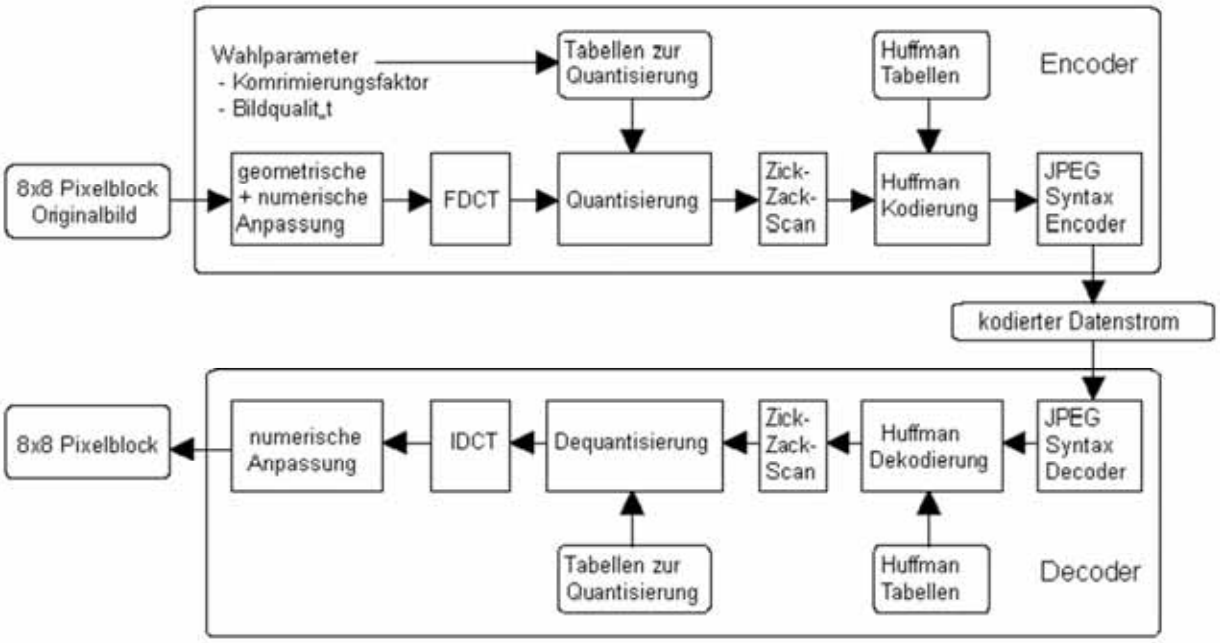


Abb. 3.4 Baseline-System

Hierbei nimmt die Datenmenge durch den Übergang von Integer nach Real zu. Die Koeffizienten der zweidimensionalen DCT-Matrix werden mit speziellen Werten aus der Quantisierungsmatrix ganzzahlig dividiert. Hochfrequente Anteile können so gezielt einer gröberen Quantisierung unterzogen werden. Dieser Prozess ist verlustbehaftet und in der resultierenden DCT-Koeffizientenmatrix werden signifikant viele Werte zu 0 oder 1. Die DCT-Koeffizienten werden dann in einem Zick-Zack-Scan in einen eindimensionalen Vektor konvertiert, der dann den verlustfreien Verfahren der Lauflängen-Codierung (RLE), VLI-Codierung (variable length integers) und der Huffman-Codierung unterzogen wird. In einem letzten Schritt wird die JPEG-Syntax generiert.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Abb. 3.5 DCT-Koeffizientenmatrix

Die Dekomprimierung erfolgt in umgekehrter Reihenfolge, in der letztlich die IDCT (Inverse DCT) die räumlichen Pixel liefert. Die Linearisierung durch den Zick-Zack-Scan liefert größtmögliche Folgen von Nullen, die eine bestmögliche Entropie-Codierung durch die verlustfreie Lauflängen-Codierung zulassen. Als Beispiel die von der JPEG vorgeschlagene Quantisierungsmatrix für den Standard CCIR-601:

-6	0	0	0	0	0	0	0
3	-1	1	-1	0	0	0	0
2	1	0	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Abb. 3.6 Quantisierungsmatrix

Extended System
JPEG unterstützt 4 Modi der Codierung. Die Standardversion beschreibt den sequentiellen Modus. Für die Übertragung über Verbindungen geringer Bandbreite ist der progressive Modus interessant, für die Unterstützung von Devices unterschiedlicher Auflösung der hierarchische Modus:

- **Sequentieller Modus (baseline system):** Das Bild wird von links nach rechts und von oben nach unten Block für Block codiert und in umgekehrter Reihenfole wieder decodiert und angezeigt.
- **Progressiver Modus (extended system):** Das Bild wird in mehreren Scans codiert beginnend mit einem verschwommenen Bild aus niederfrequenten Anteilen, das nach und nach Schärfe erhält durch die hochfrequenten Anteile (spectral selection). Dies ist bei Übertragung über Leitungen geringer Bandbreite interessant, bei dem der Darstellungsprozess bei ausreichender Darstellung beendet werden kann. Neben der progressiven Bildgenerierung unterstützen diese Systeme eine Auflösung bis 12 Bit und nutzen die arithmetische Codierung, die bis zu 10% stärker komprimiert als die Huffman-Codierung.
- **Hierarchischer Modus:** Das Bild wird zunächst in geringer räumlicher Auflösung als Basisbild codiert. Danach wird ein Bild höherer Auflösung codiert, indem lediglich die Differenz zwischen der höheren Auflösung und der interpolierten Basis codiert wird. Unterschiedlicher Auflösungen des gleichen Bilds können so hierarchisch codiert unterschiedliche Devices unterstützen.
- **Verlustfreier Modus:** Zur verlustfreien Codierung muss auf die DCT verzichtet werden, lediglich die verlustfreie Codierung kommen zum Einsatz.

MPEG
Der MPEG Standard wurde von der Motion Picture Experts Group der ISO/IEC entwickelt mit dem Ziel, eine Übertragung von digitalen Bewegtbildern mit digitalem Audio über Medien geringer Bandbreite bis 1,5 Mbit/s zu ermöglichen. MPEG-1 und folgende sind als generische Standards gedacht im Sinne eines anwendungsunabhängigen Toolkits. Spezifiziert wurde nicht die Implementierung, sondern der Bitstrom. Eine Implementierung muss nicht alle Fähigkeiten bereitstellen. MPEG stellt z.B. Funktionen für schnelle und langsame Anzeige oder Suche vorwärts und rückwärts bereit sowie die Synchronisierung von Video- und Audiosignalen.

MPEG-1 und MPEG-2
MPEG-1 wurde 1993 verabschiedet und ist in der ISO/IEC 11172 beschrieben. Um sowohl wahlfreien Zugriff auf Einzelbilder zu gestatten als auch eine aus-

reichende Komprimierungsrate zu erzielen, verwendet MPEG insgesamt 4 Bildtypen:

- **I-Frame:** Intraframe Bilder lassen zeitliche Redundanzen zu aufeinanderfolgenden Bildern unberücksichtigt. Sie werden analog zu JPEG codiert und dienen als Referenz für die übrigen Bildtypen. Auf sie kann wahlfrei zugegriffen werden. I-Bilder besitzen die geringste maximale Komprimierungsrate.
- **P-Frame:** Prädikative Bilder benötigen zur Codierung und Decodierung das vorangegangene I-Bild oder P-Bild. P-Bilder nutzen zeitliche und räumliche Redundanzen zur Komprimierung. Auch P-Bilder können Referenzbilder sein. Der Zugriff auf ein P-Bild ist nur nach Decodierung des vorangehenden I-Bildes und der dazwischen liegenden P-Bilder möglich.
- **B-Frame:** Bidirektionale Bilder benötigen Informationen des vorangehenden und/oder des nachfolgenden I-Bilds und/oder P-Bilds. Sie können nicht als Referenzbild verwendet werden, bieten dafür maximale Komprimierungsraten. B-Frames haben ihren Namen von der Tatsache, dass Referenzframes in beiden Richtungen gefunden werden können.
- **D-Frame:** D-Bilder sind wie I-Bilder intraframe codiert, dienen der Anzeige beim schnellen Vor- und Rücklauf und dürfen nicht mit den anderen Bildtypen im selben Datenstrom vorkommen. D-Bilder enthalten nur niedrige Frequenzanteile eines Bildes.

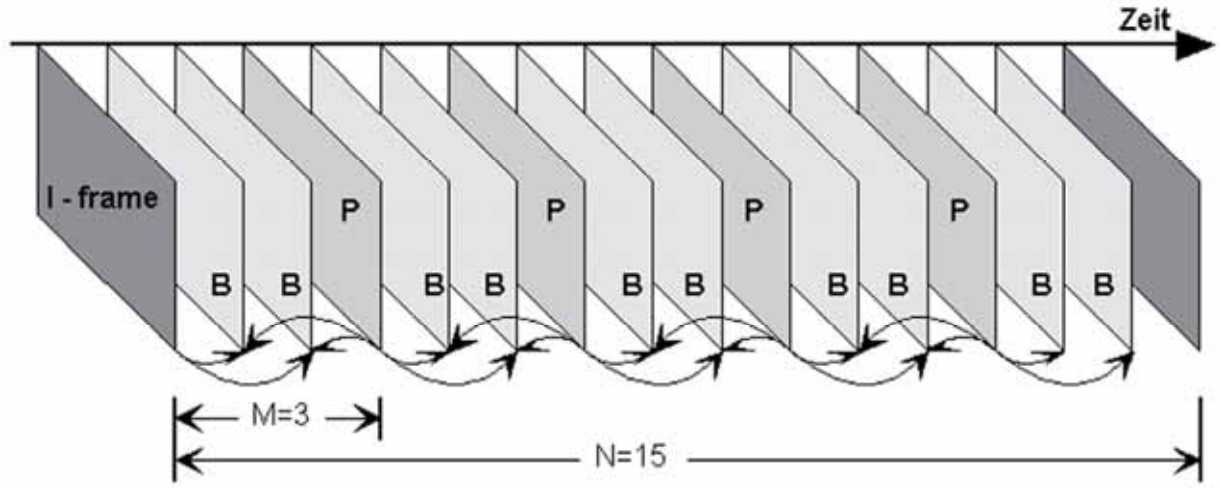


Abb. 3.7 MPEG1 und MPEG2

MPEG definiert eine GOP-Struktur (GOP=Group Of Pictures), die insgesamt N Frames enthält, mit einem I-Frame beginnt und eine Reihe von P-Frames enthält. Zwischen den I- und P-Frames existieren M-1 B-Frames. Normalerweise wird nur das erste Bild einer GOP als I-Frame gespeichert. Enthält die GOP viele Bilder, liegen die I-Frames zeitlich weit auseinander und der wahlfreie Zugriff ist erschwert. Zu kleine GOPs reduzieren die Komprimierungsrate. Übli-

cherweise gilt N=15 und M=3 bei einer Bildfrequenz von 25Hz. Die übliche Auflösung von MPEG-1 ist 320x240 Pixel. Jeder Frame wird in 15 Streifen unterteilt und diese wiederum in 22 Makro Blöcke (MBs). MPEG-1 unterstützt das 4:2:0 Format mit 4 Blöcken für die Luminanz und jeweils ein Chrominanzblock für U und V (U=Cr und V=Cb). MPEG-1 unterstützt nur den noninterlaced-Modus und ist abwärtskompatibel zu JPEG und H.261.

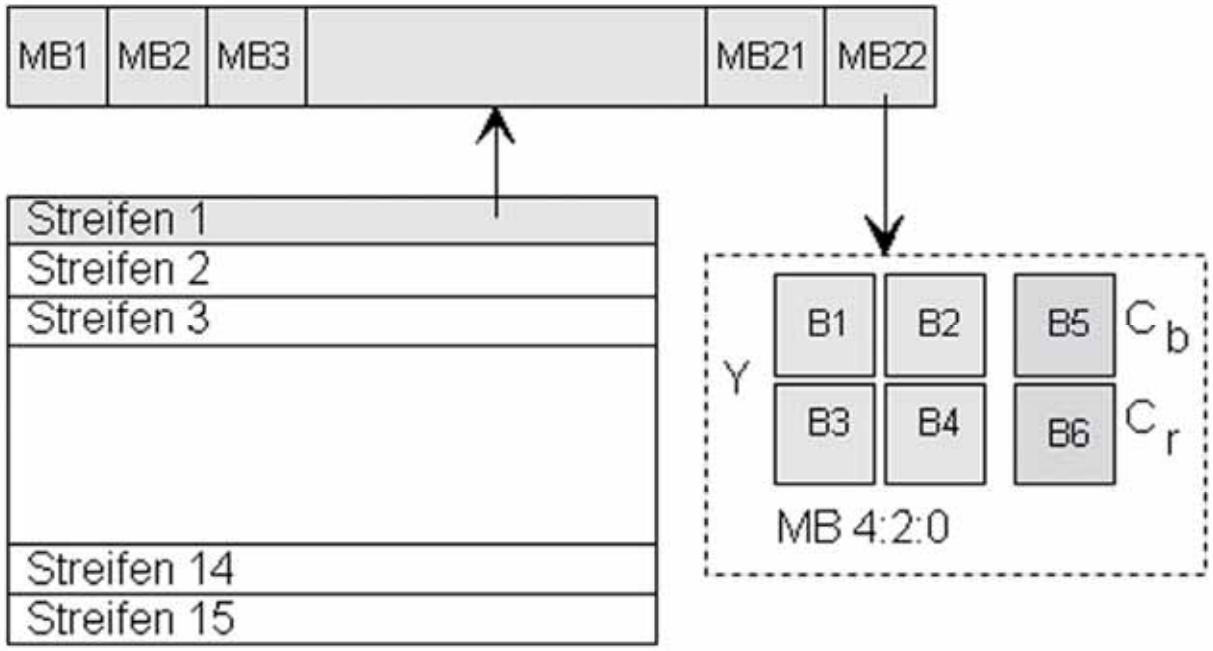


Abb. 3.8 MPEG1-Blockstruktur

Erst MPEG-2, beschrieben in der ISO/IEC DIS 13818-2, unterstützt die auch für TV übliche Auflösung (PAL) mit 720x480 Pixel. Ein Frame wird dabei in 30 Streifen mit 44 MBs unterteilt. MPEG-2 unterstützt zudem die Formate 4:2:2 und 4:4:4.

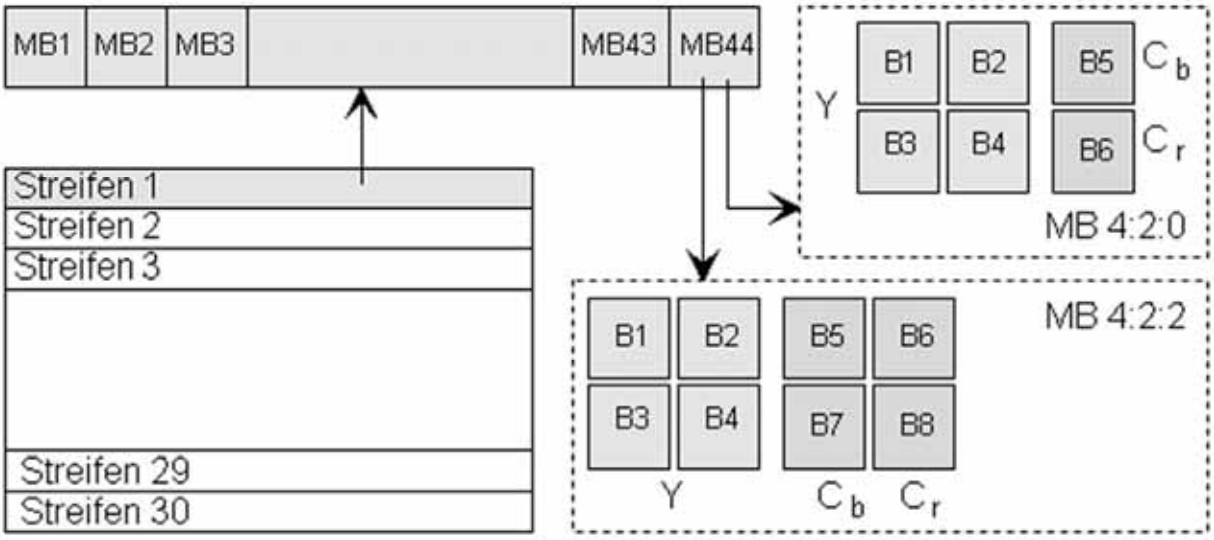


Abb. 3.9 MPEG2-Blockstruktur mit erweiterte FarbCodierung

Zur Codierung werden I-Frames mit DCT, Quantisierung, Zick-Zack-Scan, Lauflängen- und Huffman-Codierung komprimiert und in den MPEG-Bitstrom geschrieben. Die quantisierten DCT-Koeffizienten werden invers quantisiert und die IDCT wird verwendet, um Referenzframes zu generieren, die im Speicher gehalten und zur Bewegungsabschätzung für P- und B-Frames verwendet werden. Die Bewegungsvektoren der MBs in den P-Frames werden auf Grundlage der Referenzframes des nächsten P- oder I-Frames gebildet. Die B-Frames werden auf Grundlage der beiden nächststehenden P- oder I-Frames gebildet.

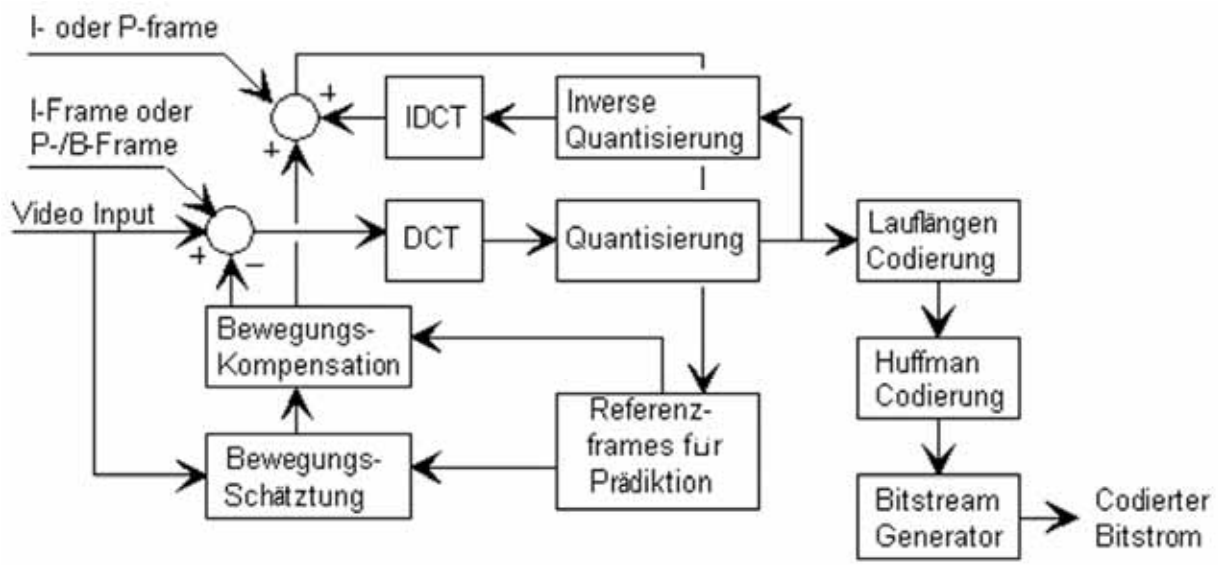


Abb. 3.10 MPEG2-Codierung

Die rechenintensivste Stufe ist die Bewegungsschätzung, wofür aber gute Algorithmen existieren. Grundidee ist, für jeden Frame einen geeigneten Referenzframe zu finden und dessen Bewegungsvektor zu übernehmen, bevor ein neuer Bewegungsvektor gebildet werden muss. Um auch bei stark wechselnden Bildern mit der gegebenen Bandbreite zurechtzukommen, wird der Quantisierungsprozess skaliert, bei starken Bewegungen wird eine verminderte Qualität in Kauf genommen.

Die Decodierung verläuft in umgekehrter Richtung. Zunächst wird ein I-Frame decodiert und als Referenzframe zur Bewegungsschätzung im Speicher gehalten und dient zusammen mit dem entsprechenden Bewegungsvektor zur Generierung bewegungskompensierter P-Frames.

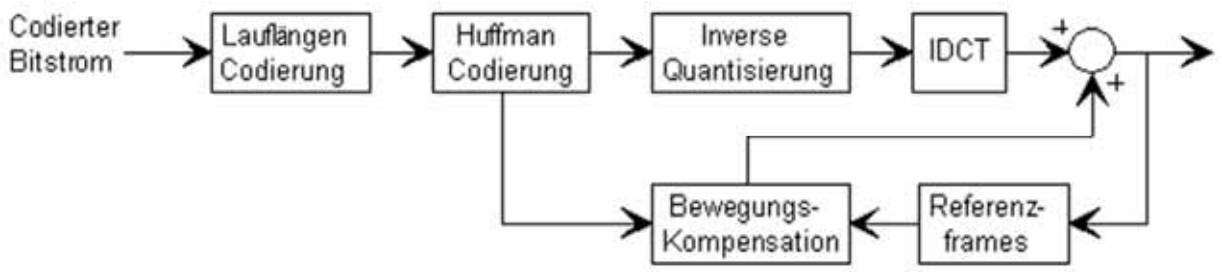


Abb. 3.11 MPEG2-Decodierung

MPEG-2 gestattet verschiedene Skalierungen. Dies ist notwendig, wenn z.B. in Netzwerken unterschiedliche Belastungen in den Übertragungsstrecken auftreten oder verschiedene Geräte mit unterschiedlichen zeitlichen und räumlichen Auflösungen beteiligt sind wie Standard TV, HDTV, Videotelephon, Videokonferenzsysteme, Monitore etc. Um den unterschiedlichen Anforderungen Rechnung zu tragen, werden in MPEG-2 verschiedene Qualitätsstufen (Layer) unterstützt. Die unteren Layer besitzen eine niedrigere Auflösung oder Bildrate. Kommen bei geringerer Bildrate weniger Bilder/s an, sind lediglich die unteren zeitlichen Layer beteiligt, die Bilder der höheren Raten werden von den höheren Layer übernommen. Beide zeitlichen Layer besitzen dieselbe räumliche Auflösung. Bei räumlicher Skalierbarkeit unterstützen die Layer unterschiedliche Auflösungen. Die qualitative Skalierung bei gleicher räumlicher Auflösung ist durch eine entsprechende Quantisierung und der Farbreduktion möglich. Hybride Skalierbarkeit basiert auf 3 Layer, von denen immer 2 genutzt werden.

Profil / Level	einfach 4:2:0 NS	normal 4:2:0 NS	qualitativ 4:2:0 S 2 Layer	räumlich 4:2:0 S 3 Layer	hybrid 4:2:2 NS 4:2:2/4:2:0 S 3 Layer
niedrig		352x288 30 Bilder/s 3.04 MS/s 4 Mbit/s	352x288 30 Bilder/s 3.04 MS/s 4 Mbit/s		
normal	720x576 30 Bilder/s 10.4 MS/s 15 Mbit/s	720x576 30 Bilder/s 10.4 MS/s 15 Mbit/s	720x576 30 Bilder/s 10.4 MS/s 15 Mbit/s		720x576 30 Bilder/s 11.06/14.75 MS/s 20 Mbit/s
hoch		1440x1152 60 Bilder/s 47 MS/s 60 Mbit/s		1440x1152 60 Bilder/s 47 MS/s 60Mbit/s	1440x1152 60 Bilder/s 47/62.7 MS/s 60 Mbit/s
sehr hoch		1920x1152 60 Bilder/s 62.7 MS/s 80 Mbit/s			1920x1152 60 Bilder/s 62.7/83.5 MS/s 100 Mbit/s

Abb. 3.12 MPEG2-Profile und Levels

Ein MPEG-2 System besitzt 2 Systemebenen, um Video- und Audioströme in Pakete aufzuteilen. Die erste Ebene erzeugt einen Strom von Paketen aus den elementaren Datenströmen für Video, Audio, Daten und Kontrollinformationen und fügt Zeitmarken hinzu, damit die verschiedenen Datenströme beim Empfänger wieder synchronisiert werden können. Die zweite Ebene generiert abhängig vom Übertragungsmedium den Programmstrom für fehlerfreie Medien wie lokale Speicher und den Transportstrom für fehlerbehaftete Übertragungsmedien.

MPEG-4

Die Entwicklung von MPEG-4 wurde 1993 begonnen und 1997 erstmals beschrieben. Dieses Format erfreut sich mannigfaltigen Einsatzes in fast allen aktuellen Internet-Video-Anwendungen wie REAL-Player-Format und DivX. MPEG-4 unterscheidet sich erheblich von MPEG-2, das lediglich einen Komprimierungsstandard spezifiziert. Die mit MPEG-4 verbundenen Eigenschaften sind

- flexible und skalierbare Bitstreams z.B. für Systeme unterschiedlicher Auflösung,
- einfache Erweiterbarkeit für neue Algorithmen bei Empfänger- und Transportdiensten,
- inhaltsbasierte Interaktivität für Multimedia-Anwendungen und
- Netzwerkunabhängigkeit, um bei jedem Verbreitungsmedium eingesetzt werden zu können.

Dadurch ergeben sich Anwendungen wie

- Echtzeit-Kommunikation wie Videokonferenzen,
- mobiles Computing auf Laptops über drahtlose Verbindung zum Host/Router/Modem,
- inhaltsbasiertes Speichern von Videos,
- Videoübertragung über Internet, Videokonferenzen über Internet,
- Digitaler Rundfunk in High Qualtity,
- Studio- und Post-TV-Produktionen und
- interaktive Filme.

Die Möglichkeiten von MPEG-4 unterscheiden sich also beträchtlich gegenüber den bisherigen reinen Komprimierungsstandards. Teil des Standards sind Tools der SNCH-Group (Synthetic and Natural Hybrid Coding) zur effektiven Repräsentation synthetisch generierter audio-visueller Informationen (wie z.B. Flash-Animationen). Sie integrieren Texte und Grafiken, unterstützen die Codierung synthetischer Körper und Gesichter, deren Animation, Texturen und synthetische Sprach- wie Klanggenerierung. Die Interaktion des Benutzers umfasst

Aktionen wie Navigation durch eine Szene, Objektpositionen verändern oder Ereignisauslösung über die Maus wie Start und Stop eines Videos.

MPEG-4 enthält sog. AVOs (audio-visuelle Objekte), die audio-visuelle Szenen beschreiben. Basiskonzept sind primitive AVOs wie fester 2D-Hintergrund, Bild einer Person, gesprochener Text der Person etc. Zusammengehörende visuelle und audio-Komponenten bilden ein hierarchisch geordnetes, zusammengesetztes AVO. MPEG-4 beschreibt eigentlich die Konfiguration, Kommunikation und Instantiierung von Klassen von Objekten. Zu Beginn einer Operation werden zunächst in einer Konfigurationsphase die Klassen der Objekten bestimmt, die benötigt werden. In der Kommunikationsphase werden fehlende Klassen in den Encoder geladen und schließlich werden die Klassen instantiiert durch Laden der Klassendescriptoren mit den Datenstrukturen und Verarbeitungsmethoden. Danach kann der Decoder die Sequenz generieren.

Neben dem AVO ist die VOP (video object plane) ein zentrales Konzept. Ein VOP repräsentiert ein Video Objekt über den Zeitraum seiner Existenz. Jedes VOP wird getrennt codiert in Bezug auf Form, Bewegung, Textur, Sound etc. Die Codierung der Form geschieht mit Schwarzweiß oder Graustufen-Bitmaps. Bewegung und Textur werden wie bei MPEG-2 codiert. Diese und Informationen über den Bildaufbau wie Position und Orientierung werden an einen Multiplexer übergeben. Dieser multiplext und synchronisiert die Daten und generiert den Bitstrom. Der Bitstrom kann über Netzwerkverbindungen übertragen oder auf externen Medien gespeichert werden. Die Spezifikation der Transportschnittstelle ist Teil von MPEG-4. Beim Empfänger stellt ein Demultiplexer aus dem ankommenden Bitstrom die spezifischen Bitströme für Form, Bewegung, Textur, Sound etc. wieder her. Danach werden alle AVOs decodiert und mit Hilfe der Informationen über Bildaufbau etc. die Audiodaten und Videodaten ausgegeben.

MPEG-4 erlaubt sowohl räumliche als auch zeitliche Skalierung, um auch eine Übertragung bei geringen Bandbreiten zu ermöglichen oder gestattet dem Empfänger, nur Teile des Bitstroms zu decodieren und Bilder oder ganze Bildsequenzen zu rekonstruieren. Es ist eine geringere räumliche oder zeitliche Auflösung bei gleichbleibender Bildqualität oder eben schlechtere Bildqualität bei unveränderter Auflösung möglich. MPEG-4 erfüllt Forderungen nach Fehlertoleranz und Robustheit und umfasst die Resynchronisation unterbrochener Bitströme, Wiederherstellung beschädigter Bitströme und verhindert die Fehlerfortpflanzung beim Decodieren des Bitstroms.

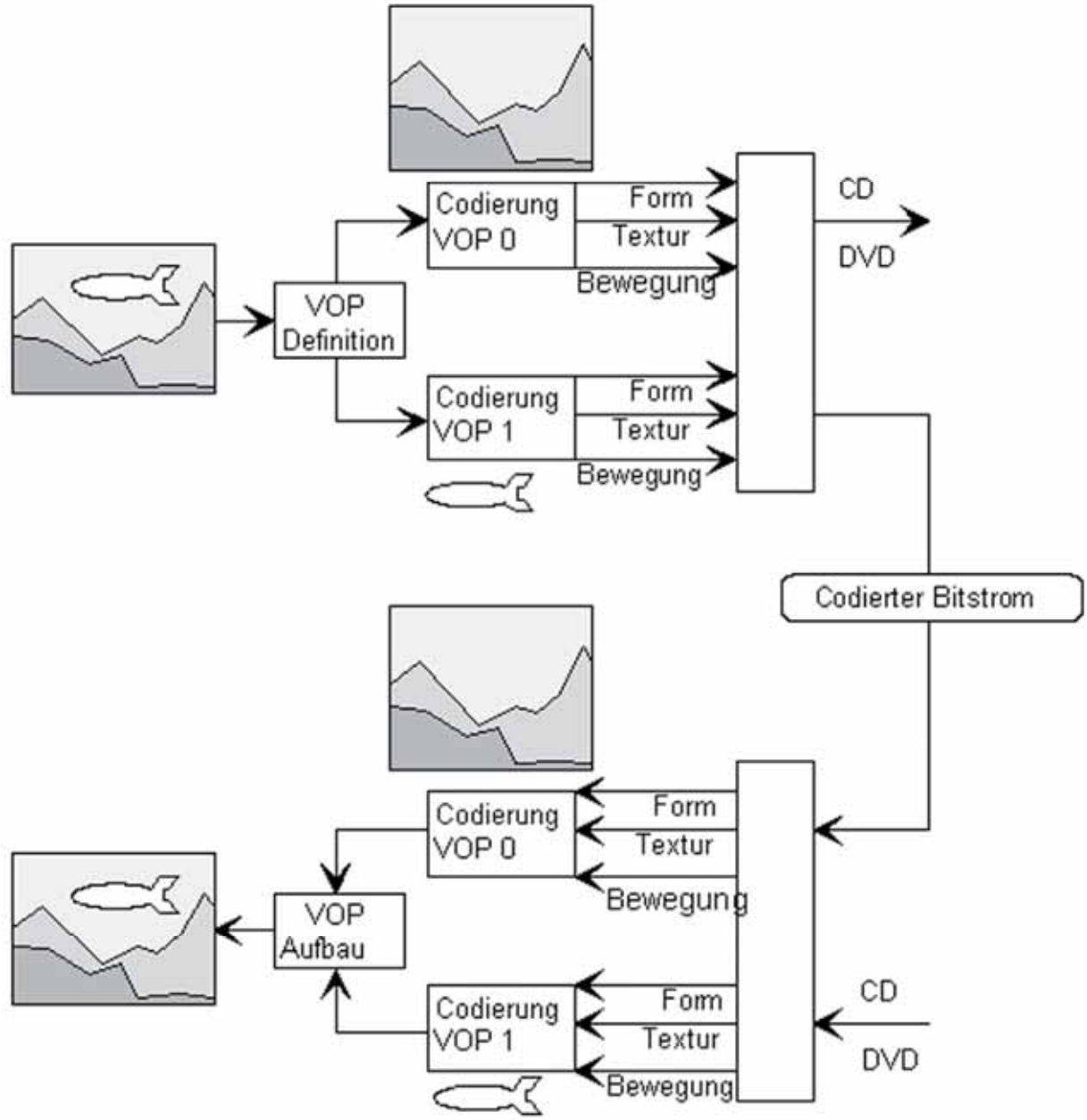


Abb. 3.13 MPEG4-Codierung

MPEG4-Varianten: WMV, DivX, RealVideo (rm)

Dateien, die WMV-codierte Video-Streams enthalten, sind meist in Microsofts sogenanntes „Container-Format Advanced Streaming Format“ (ASF) eingebettet. Diese Dateien haben normalerweise die Dateiendung .asf, im Falle einer Video-Datei können sie aber auch die Endung .wmv tragen (siehe auch weiter hinten: Video for Windows -Format).

Eine neue Variante von WMV ist das Format Windows Media Video High Definition (WMVHD).

Das Windows Media Format unterstützt auch die Einbindung von Digital Rights Management (DRM), die dem Urheber von 'geistigem Eigentum' die Regelung der Zugriffsrechte auf Ton- sowie Videomaterial ermöglichen soll. In der Praxis wird diese Technik häufig als Kopierschutzmaßnahme eingesetzt.
Es existieren zur Zeit drei Versionen von WMV, die allesamt vom Aufbau her ähnlich zu MPEG-4 sind. Diese Versionen heißen WMV1 bis WMV3, werden oft aber als WMV7 bis WMV9 bezeichnet, da sie gemeinsam mit Version 7.0 bis 9.0 des Windows Media Players erschienen sind. Eine erweiterte Fassung von WMV3 wurde als VC-1 bei SMPTE eingereicht und 2006 als offener (aber dennoch proprietärer) Standard verabschiedet. VC-1 kommt unter anderem auf Blu-Ray Disc und HD DVD als einer von drei obligatorischen Videocodecs zum Einsatz.
DivX ist ein MPEG-4-kompatibler Video-Codec, der von DivX Inc. (früher DivXNetworks Inc.) entwickelt wurde. Der Codec ist für seine Fähigkeit bekannt, große Videodateien bei guter Qualität vergleichsweise stark komprimieren zu können.
Xvid ist ein Open-Source-MPEG-4-Video-Codec, der ursprünglich auf dem OpenDivX-Quelltext basierte. Der zugrunde liegende Quellcode von OpenDivX stammte wiederum aus der MPEG-4-Referenzimplementierung des EU-Projekts MoMuSys. Das Xvid-Projekt wurde von mehreren freien Programmierern gestartet, nachdem der Quellcode von OpenDivX geschlossen wurde. Auch der Name des Projekts ist eine Anspielung darauf ('XviD' ist 'DivX' rückwärts). Durch den unverschlüsselt veröffentlichten Quelltext von OpenDivX bekamen die Programmierer nun die Möglichkeit, den Codec in den grundlegenden Eigenschaften zu verändern und zu optimieren. Zusammen mit DivX und Nero Digital ist dieser Codec der bekannteste MPEG-4-Encoder.
Real Video, von Real Inc. in USA entwickelt, stellt Videos mit der Endung .rm oder .ram zur Verfügung. Diese sind analog MPEG4 aufgebaut mit einigen Produktspezifischen Header.
Eine Formatwandlung zwischen alle den MPEG-Formaten ist mit geeigneten (und oft kostenlosen) Konvertern heute problemlos möglich.

MPEG-7
Der MPEG-7 Standard soll die Suche nach audio-visuellen Informationen im Web in entsprechenden Datenbanken ermöglichen. Die meisten Suchmaschinen sind bisher textbasiert. MPEG-7 enthält Bilder, Grafiken, Tondokumente, Video und die Informationen, wie diese Objekte zusammenhängen (siehe MPEG-4). Die Fertigstellung des MPEG-7 Standards wurde im Jahr 2002 veröffentlicht. Es handelt sich hier also um kein Komprimierungsverfahren, sondern um eine standardisierte Beschreibung multimedialer Objekte.

MPEG-Audio

Die Bandbreite für MPEG Audiodaten beträgt 20 KHz, die Abtastraten (auch Sampling Rate oder Sampling Frequenz) betragen üblicherweise 32, 44,1 oder 48 Ksamples/s bei 16 Bit/sample. Die Frequenzdomäne bietet deutlich mehr Möglichkeiten zur Komprimierung als die Zeitdomäne. Die Komprimierung beruht auf psycho-akustischen Algorithmen, die Charakteristika der menschlichen Wahrnehmung ausnutzen. Wenn in einem bestimmten Frequenzbereich starke Signale auftreten, werden schwächere Signale aus dem benachbarten Frequenzspektrum nicht mehr wahrgenommen und lassen eine starke Komprimierung zu.

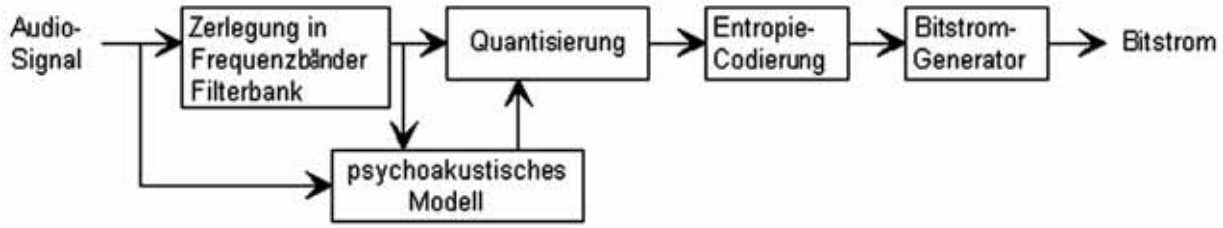


Abb. 3.14 MPEG-Audio-Codierung

Bei der Codierung werden eingehenden Audiosignale in Frequenzbändern zerlegt und in einer Filterbank gehalten. Das psycho-akustische Modell zusammen mit den Frequenzinformationen aus der Filterbank selektieren signifikante Signale und bestimmen den Rauschpegel. Die Quantisierung kann für die Frequenzbänder unterschiedlich sein. In Layer 1 und 2 erfolgt danach lediglich eine PCM-Codierung (Pulse Code Modulation), in Layer 3 (mp3) eine Huffman-Codierung. Danach wird der in Frames, Audio-Access-Units und Slots strukturierte Audio-Bitstrom generiert, der dann mit dem Video-Bitstrom gemischt wird, wobei Zeitmarken zur Synchronisation beider Bitströme beim Decoder hinzugefügt werden. MPEG-1 besitzt 2 Audio-Kanäle, MPEG-2 besitzt fünf und erlaubt verschiedene Formen des Surround-Sound. Die Kanäle 1 und 2 sind identisch zu den MPEG-1 Kanälen. MPEG-2 verfügt über einen Loudness-Kanal für niedere Frequenzen zwischen 20 und 120 Hz und unterstützt zusätzlich die niederen Abtastraten mit 16, 22,05 und 24 Ksamples/s.

Die Audio-Fähigkeiten von MPEG-4, beschrieben in ISO/IEC 14496-3 MPEG-4 Audio, ermöglichen Skalierbarkeit und Wiedergabe bei verschiedenen Geschwindigkeiten. MPEG-4 standardisiert die Audio-Codierung von 2 bis 64 Ksamples/s. Parametrisierte Codierung wird für niedrige Raten bis 6 Ksamples/s verwendet, von 6 Ksamples/s bis 24 Ksamples/s wird die Code Excited Linear Predictive Codierung (CELP) angewandt und darüber die Time-to-Frequency (T/F) Codierung.

WMA wird zur Komprimierung von digitalen Audioinhalten verwendet, und in der Regel mit verlustbehafteter Kompression eingesetzt, vergleichbar dem MP3-Verfahren. Der Codec unterstützt bis zu 24 bit/96 KHz bei einer variablen Bitrate von bis zu 768 kb/s und Surround-Ton mit bis zu 7.1 Kanälen. Daneben gibt es eine Version, die explizit auf Quellmaterial mit Stimmen ausgelegt ist (Windows Media Audio Voice, ACELP), sowie den verlustfreien Codec Windows Media Audio Lossless.
Das standardmäßige verlustbehaftete Kompressionsverfahren von WMA basiert auf demselben Prinzip wie die MP3-Kompression: Nach einer Umwandlung in eine Frequenz-Amplituden-Domäne werden Maskierungseffekte (nahe Frequenzen, die nicht unterscheidbar sind) oder Töne, die generell nicht hörbar sind (Hörgrenze) gelöscht. Dadurch wird Speicherplatz gespart.
WMA-codierte Audio-Streams sind in der Regel in ASF-Container eingebettet. Im Falle einer reinen Audio-Datei tragen die Dateinamen zumeist die Endung .wma

Dolby Digital (AC3)
Dolby Digital (auch ATSC A/52 und AC-3) ist ein Mehrkanal-Tonsystem der Firma Dolby, das in der Filmtechnik, auf Laserdiscs, DVDs und in der Fernsehtechnik zum Einsatz kommt. Im Bereich Kino und DVD sind die direkten Konkurrenten DTS (Digital Theater Systems) und SDDS (Sony Dynamic Digital Sound speziell für das Kino entwickelt). Dolby Digital unterstützt bis zu sechs diskrete Kanäle und verwendet ein psychoakustisches, verlustbehaftetes Verfahren zur Datenkompression. Das Format wurde vom Advanced Television Systems Committee international standardisiert und trägt offiziell den Namen ATSC A/52. Dolby Digital (DD) ist der Marketingname. AC-3 schließlich bezeichnet das zugehörige Bitstream-Format (Adaptive Transform Codec 3) und hat sich ebenfalls als Bezeichnung eingebürgert. Daher kommt auch die typische Dateiendung *.ac3. Auch leicht abgewandelte Bezeichnungen wie Dolby Stereo Digital oder Dolby SR-Digital und einige andere werden verwendet.

MHEG
MHEG (Multimedia and Hypermedia information coding Expert Group) definiert keinen neuen Standard zur Komprimierung, sondern vereint verschiedene Standards und spezifiziert, wie verschiedene Medien einer Multimedia-Anwendungen zusammengeführt, wie Video- und Audiodaten wiedergegeben werden und wie Benutzer mit Multimedia-Anwendungen interagieren können. MHEG wurde von der ISO/IEC JTC1/SC29 WG12 spezifiziert und enthält plattformunabhängige Beschreibungen von Objektklassen z.B. zur Steuerung der Wiedergabe.
MHEG besitzt einen anderen Ansatz wie HTML (Hypertext Markup Language)

aber mit gleicher Zielrichtung. MHEG besitzt gegenüber HTML weitergehende Möglichkeiten zur Synchronisation der Bitströme oder der Kontrolle der Wiedergabegeschwindigkeit. So verwendet MHEG z.B. ein dreidimensionales Koordinatensystem und Zeitmarken zur Synchronisation bei der Wiedergabe.
MHEG definiert eine abstrakte Syntax zur Beschreibung logischer Einheiten wie Audiodaten, Videodaten, Konvertierungen, Informationsaustausch, Methoden zur Registrierung, Identifikation, Kommunikation, Bildverarbeitung usw. Benutzt wird die von der ISO entwickelte Abstract Syntax Notation (ASN.1) zur Darstellung von Datenstrukturen für verschiedene Plattformen, Betriebssystemen und Programmiersprachen (ISO/IEC 13522-1). Die Portierung auf verschiedene Plattformen soll durch Tools generiert werden können. MHEG-5 wurde für interaktive Multimedia-Anwendungen in Client-Server-Umgebungen entwickelt und enthält Klassen wie

- **Inhaltsklassen:** Video, Audio, Bilder, Grafiken oder Text
- **Verhaltensklassen:** Kontrolle der Wiedergabe der Inhaltsklassen
- **Interaktionsklassen:** Benutzerinteraktion mit der Multimedia-Anwendung

Objekte dieser Klassen können instantiiert werden. Anwendungen bestehen aus Szenen und Objekten. Szenen bestehen aus mehreren Objekten. Die interaktiven Objekte erlauben die Benutzersteuerung der Anwendung. Damit MHEG-basierte Anwendungen bei einem Client ausgeführt werden können, benötigt das Client-System eine MHEG-Engine, die eine vom Server geladene Anwendung zur Ausführung bringt. Der objektorientierte Ansatz reduziert die Auslastung bei Server, Netzwerk und Client-Computer.
MHEG-6 ist in Arbeit und enthält eine Java Virtual Machine (VM) und weitere Klassen, um Multimedia-Anwendungen über Internet stärker zu unterstützen.

HDTV
Ein Bild in Kino-Qualität (HDTV) benötigt 24 Megabyte Speicherplatz. Bei 30 Bilder/s wird eine Übertragungsrate von 720 Megabyte/s oder 5.8 Gigabit/s benötigt. High Definition Television (HDTV) nutzt zur Komprimierung die Erkenntnis, dass bei schneller Bildfolge Unschärfen nicht wahrgenommen werden und weniger Bildinformationen übertragen werden müssen. Bei langsamen Bildfolgen ist eine langsame Bildübertragung (< 30 Bilder/s) nicht problematisch. Beim HDTV-Verfahren werden grundsätzlich 50 Viertelbilder/s übertagen. Dabei gilt:

- bei wenig Bildänderungen werden diese zu einem Bild zusammengesetzt. Damit ergibt sich eine Bildrate von 12.5 Bilder/s.
- bei mittlerer Bildänderung werden nur 2 Viertelbilder verwendet, der Rest

wird interpoliert. Daraus resultiert eine Bildrate von 25 Bilder/s.
- bei hoher Bildänderung wird nur 1 Viertelbild verwendet, der Rest wird interpoliert. Daraus resultiert eine Bildrate von 50 Bilder/s.

HDTV besitzt ein Bildformat mit Seitenverhältnis 16:9, arbeitet mit einer Bildwiederholfrequenz von 100 Hz interleaved bzw. 50 Hz non interleaved und einer Zeilenfrequenz von 1250 Zeilen. Bilder werden im 4:2:0 Format übertragen. Die dargestellte Zeilenanzahl beträgt 1152, die Luminanzspaltenzahl 1920 Punkte/Zeile und Chrominanzspaltenzahl 960 Punkte/Zeile. Die Übertragungsrate für Videodaten allein beträgt 132 MBit/s, für Video- und Audiodaten 140 Mbits/s bei einer Komprimierungsrate von 1:6,7.

H.261

Der Video Codec Standard H.261 wurde von der ITU-T ursprünglich empfohlen für Videokonferenzen und Bildtelefone, die ISDN-Leitungen nutzten. Um unterschiedlichen Bandbreiten Rechnung zu tragen, existieren die 2 Formate CIF (Common Intermadiate Formate) und QCIF (Quarter CIF) mit geringerer Auflösung. Die Komprimierung nutzt eine Bewegungsschätzung basierend auf 16x16 Pixel großen Makro Blöcken (MB). Die Auflösungen sind daher Vielfache dieser Blockgrößen:

	CIF		QCIF	
	Zeilen/Bild	Punkte/Zeile	Zeilen/Bild	Punkte/Zeile
Y	288	360	144	180
$U = C_b$	144	180	72	90
$V = C_r$	144	180	72	90

Abb. 3.15 H.261 Formate

Bei 15 Bilder/s werden ohne Kompression 18.3Mbit/s benötigt. Um eine Übertragung über ISDN-Leitungen zu ermöglichen, benötigt man Komprimierungsraten von 50:1 bis 300:1. Über ISDN können p x 64Kbit/s übertragen werden. CIF kann ab 384Kbit/s eingesetzt werden, bei geringerer Bandbreite wird QCIF verwendet. QCIF verwendet 10 Bilder/s und eine Komprimierungsrate von 47.5:1.

Der Codec verarbeitet Blöcke von 8x8 Pixel und MBs von 16x16 Pixel zur Erkennung von Bewegung. Jeder Makro-Block enthält 4 Blöcke mit

Luminanzdaten und 2 mit Chrominanzdaten (4:2:0-Format). 3x11 MBs bilden eine GOB (Group Of Blocks). QCIF enthält 3 GOBs, CIF 12 GOBs. Der Aufbau von CIF und QCIF zeigt die Abbildung 3.16:

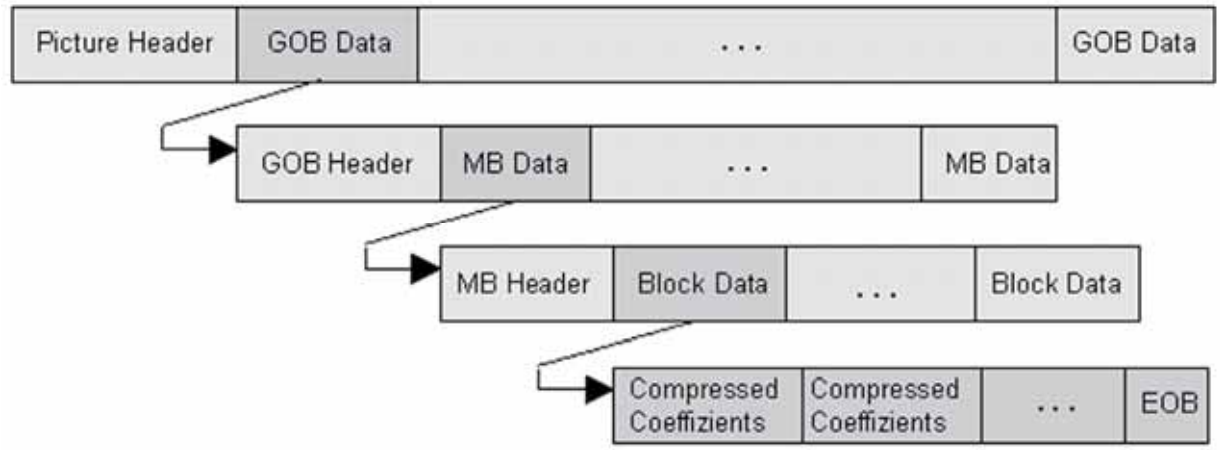


Abb. 3.16 Aufbau CIF/QCIF

MPEG ist zu H.261 abwärtskompatibel. Das H.261 System nutzt Intra-Frames und Inter-Frames. Intra-Frames werden nur beim ersten Bild gesendet oder wenn die Szene wechselt und dienen nicht der Erkennung von Bewegungen. Für jedes MB werden DCT, Quantisierung, Zick-Zack-Scan, Lauflängen- und Huffman-Codierung angewendet. Mit der inversen Quantisierung und der inversen DCT werden Referenzframes gebildet, die der Erkennung von Bewegungen für Inter-Frames dienen, in dem ein neues Frame mit dem Referenzframe des Nachbarbildes verglichen werden. Wenn die Differenz der beiden Frames unter einer gewissen Schwelle liegen, braucht der Block nicht übertragen zu werden. Ansonsten wird für die Differenz wieder DCT, Quantisierung, Zick-Zack-Scan, Lauflängen und Huffman-Codierung angewendet. Ein Loop-Filter unterdrückt störendes Rauschen.

H.263

H.263 optimiert H.261 für geringere Bandbreiten analoger Übertragungswege. Die auf Bewegung basierende Kompression wurde verstärkt, die Quantisierung wurde angepasst und arithmetische Codierung eingesetzt. H.263 wurde 1996 eingeführt und zusammen mit MPEG-4 weiterentwickelt.

H.264

H.264/MPEG-4 AVC ist ein weitere Standard zur Videokompression. Er wurde ebenfalls von der ITU (Study Group 16, Video Coding Experts Group) entwickelt. Im Jahre 2001 schloss sich die ITU-Gruppe mit MPEG-Visual zusammen und führte die Entwicklung gemeinschaftlich im Joint Video Team (JVT) fort.

Ziel des Projektes war es, ein Kompressionsverfahren zu entwerfen, das im Vergleich zu bisherigen Standards sowohl für mobile Anwendungen als auch im TV- und HD-Bereich die benötigte Datenrate bei gleicher Qualität mindestens um die Hälfte reduziert. Im Jahr 2003 wurde der Standard in beiden Organisationen im identischen Wortlaut verabschiedet. Die ITU-Bezeichnung lautet dabei H.264. Bei ISO/IEC MPEG läuft der Standard unter der Bezeichnung MPEG-4/AVC (Advanced Video Coding) und ist Teil des MPEG-4-Standards (MPEG-4/Part 10, ISO/IEC 14496-10).
MPEG-4/AVC unterscheidet sich deutlich von MPEG-4/ASP und seinen Derivaten DivX und Xvid. H.264 erreicht typischerweise eine etwa dreimal so hohe Codiereffizienz wie H.262 (MPEG-2) und ist auch für hoch aufgelöste Bilddaten (z. B. HDTV) ausgelegt. Das heißt, vergleichbare Qualität ist etwa bei einem Drittel der MPEG-2-Datenmenge zu erreichen. Allerdings ist der Rechenaufwand auch um den Faktor 2 bis 3 höher.

Wichtigen Anwendungen
Video for Windows
Video for Windows (VfW) von Microsoft dient der Wiedergabe, Komprimierung und Dekomprimierung von Videodaten. VfW enthält u.a. folgende Codecs:

- **Video 1:** Video Codec von Microsoft
- **Indeo:** Video Codec von Intel (R 3.1 und 3.2)
- **Cinepak:** Video Codec von Radius Inc.
- **RLE:** Lauflängen-Codierung von Microsoft

Diese Komponenten sind als Software in VfW enthalten und benötigen keine spezielle Hardware. Videos sind in Dateien mit der Endung .AVI gespeichert (Audio-Video-Interleave). Audio- und Videodaten eines Frames werden zusammen abgespeichert. Eine Anwendung kann sich an den ICM (installable Compression Manager) wenden, um den passenden Codec auszuwählen.

Für die Wiedergabe eines AVI-Files nutzt das MCI zur Decodierung VfW und übergibt die decodierten Audio- und Videoströme an die Peripherie. Umgekehrt können Eingabesignale von der Videokarte und der Soundkarte mit VfW codiert und in einem AVI-File abgelegt werden. VfW enthält einen Videoeditor, zur Bearbeitung von Videosequenzen. Dabei ergeben sich Probleme aus dem AVI-Format mit seinen 2 Arten von Frames. Auf ein Key-Frame folgen abhängig von der Codierung eine Reihe von Delay-Frames, die nicht unmittelbar editiert werden können, es muss immer mit einem Key-Frame begonnen werden.

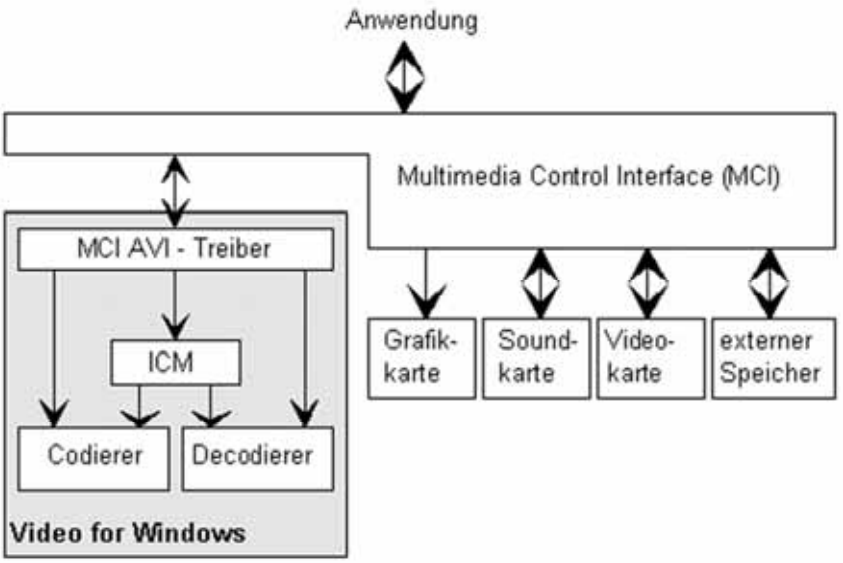


Abb. 3.17 Schnittstellen für Video for Windows

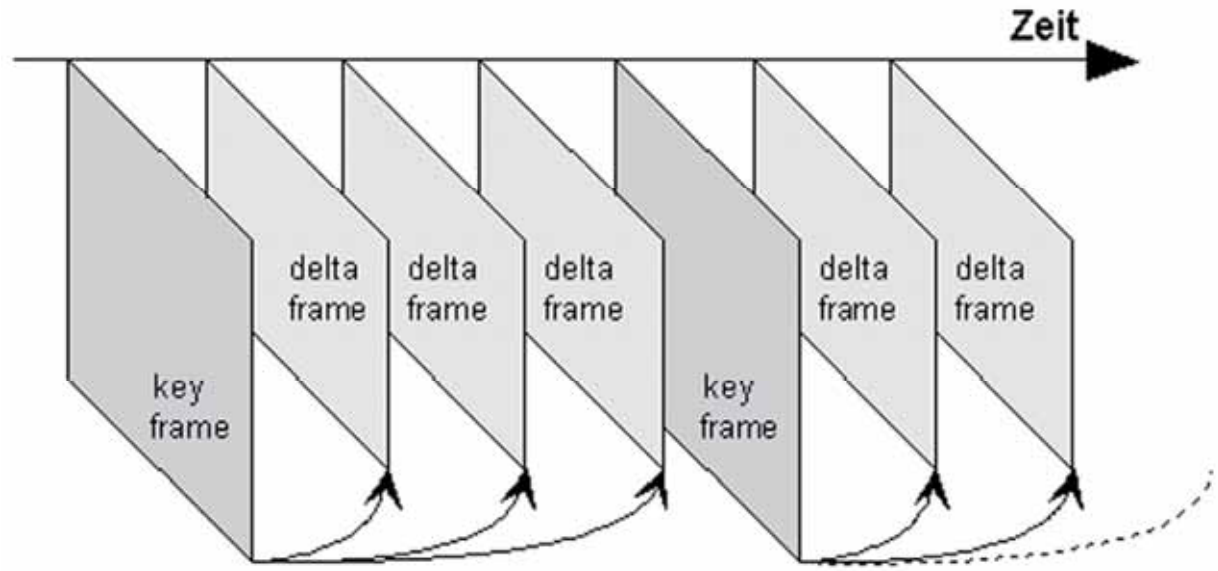


Abb. 3.18 Frames in Video for Windows

QuickTime

QuickTime ist die Multimedia Systemsoftware für Macintosh Computer und integriert Audio, Video, Grafik und Animation. QuickTime unterstützt im Standardmodus 30 Bilder/s bei 320x240 Pixel und 15 Bilder/s bei 640x480 Pixel. QuickTime unterstützt interaktives TV und somit auch Video on Demand. QuickTime enthält im Wesentlichen die 3 Teile

- **Movie Toolbox:** erlaubt Filme in Anwendungen einzubinden ohne dafür speziellen Code schreiben zu müssen
- **Image Compression Manager:** QuickTime verfügt über eine Reihe von Codecs. Der ICM stellt eine einheitliche Schnittstelle bereit, über die Filme in beliebigem Format abgespielt werden können.
- **Component Manager:** registriert die Funktionen externer Geräte (Capture-

Board, MIDI etc.). Daher genügt die genannte einheitliche Schnittstelle ohne spezielle externe Geräte ansteuern zu müssen.

QuickTime unterstützt verschiedene Codecs, z.B.:

- **Apple Compact Video Codec:** Software-Codec mit 30 Frames/s bei 320x240
- **Apple Animation Codec:** Software-Codec mit Lauflängen-Codierung für animierte Sequenzen
- **Apple Graphics Compressor:** Verlustfreier Software-Codec für 8-Bit Bilder oder 8-Bit-Filme, der genauer aber langsamer als der Compact Video Codec arbeitet.
- **Cinepak:** Software-Codec von Radius Inc. mit geringer Anforderung an CPU-Leistung bei der Wiedergabe, viel verwendeter Codec, wegen seinem plattformübergreifenden Format steigende Bedeutung für das Internet
- **Indeo:** Plattformunabhängiger Software-Codec (RTV) von Intel.
- **JPEG:** Von der ISO entwickelte Standard zur Codierung von Einzelbildern.
- **MPEG:** Von der ISO entwickelte Standard zur Codierung von Videos.
- **M-JPEG:** Codec für Bewegtbilder basierend auf JPEG. Es existiert kein einheitlicher Standard der Implementierung. Einige Anbieter haben sich zum QuickTime-Open-Forum zusammengeschlossen, um plattformunabhängige Codecs bereitzustellen. M-JPEG kann als Software implementiert werden.

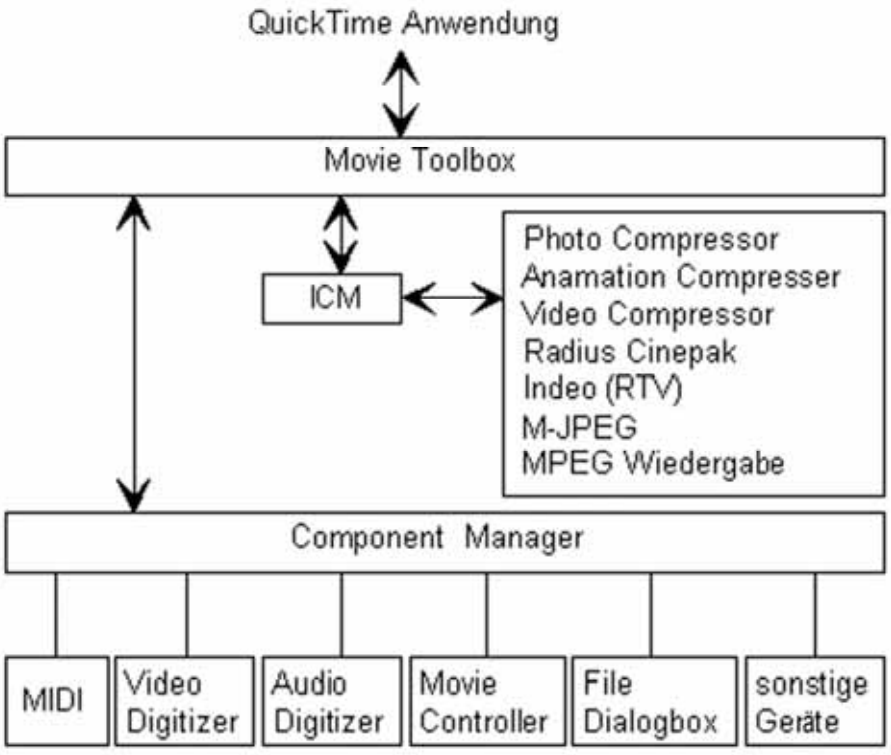


Abb. 3.19 Schnittstellen QuickTime

Neben diesen Codecs umfasst QuickTime Dateiformate, einen Standard der Bedienerführung, einen Standard Movie Controller, eine Standard File-Dialog-Box etc. Die Dateiformate beschreiben Movies und Apple Pictures (PICT/PICT2). Ein Movie enthält mehrere Tracks und Trackpointer zu jedem Track auf dem Speichermedium. Jeder Track, ob Audio, Video, MIDI, oder Text benutzt eine vorgegebene Zeitbasis (movie time) zur Synchronisation.
QuickTime VR (Virtual Reality) unterstützt weitere Features wie Videokonferenzen. Eine neuere Entwicklung ist VR, das 3D-Szenen darstellt, als ob man sich darin bewegen könnte. Es erlaubt Benutzern (auch im Internet) 3D-Operationen auszuführen wie zoomen und bewegen/drehen sowie Objekte anzufassen und mit Objekten zu interagieren
Mit QuickTime für Windows stellt Apple eine plattformunabhängige Software-Lösung bereit, die Multimedia auch im Internet ermöglicht. QuickTime für Windows nutzt MCI und OLE. Schwierigkeiten bietet das Fehlen eines Resource-Fork, der die Headers, Indizes, Bytegrößen usw. des aktuellen Movies enthält. Daher wird aus dem Zweig der Datenströme und dem Zweig der Resourcenverwaltung ein einziger Fork generiert, der nach den Daten die Resourcen-Informationen enthält. QuickTime nutzt auf einem Macintosh das MooV-Format, auf dem PC das QT- oder MOV-Format. Im Gegensatz dazu sind die AVI-Files in Video for Windows Framebasiert.

ActiveMovie
Microsoft entwickelte zunächst die 32-Bit-DirectX-Technologie. DirectX ist ein low-Level API mit den Komponenten

- **DirectDraw:** Zusammenfassung von 2D und 3D Video und Animation zu einer einheitlichen Schnittstelle, die hardwareunabhängigen Zugriff auf Grafikkarten erlaubt. DirectDraw bildet die Basis der DirectX-Technologie.
- **Direct3D:** 3D-Grafik-Renderer für geometrische Transformation, Beleuchtung, Schattierung, Texturen usw.
- **DirectSound:** gestattet hardwareunabhängigen Zugriff auf Soundkarten, Sound-Mixing, Playback, Sound-Effekte und über Internet downloadbare Musik.
- **DirectPlay:** erlaubt interaktive Anwendungen mit mehreren Benutzern über Internet.
- **DirectInput:** unterstützt Joysticks, VR-Handschuhe usw.

In der Multimedia-Weiterentwicklung ersetzt ActiveMovie das MCI (Media Control Interface) und VfW. ActiveMovie ist jedoch abwärts kompatibel. ActiveMovie wurde speziell für Internet-Applikationen entwickelt und stellt Operationen bereit für Video und Audio am Arbeitsplatz, über Internet oder Intranet,

Wiedergabe typischer Medien wie MPEG-Video und Audio, AVI-Files, Quicktime Video oder Windows Sound Files (WAV). ActiveMovie kann MPEG Videos mit 24 Frames/s und 11 KHz Stereo dekomprimieren und DVDs wiedergeben.

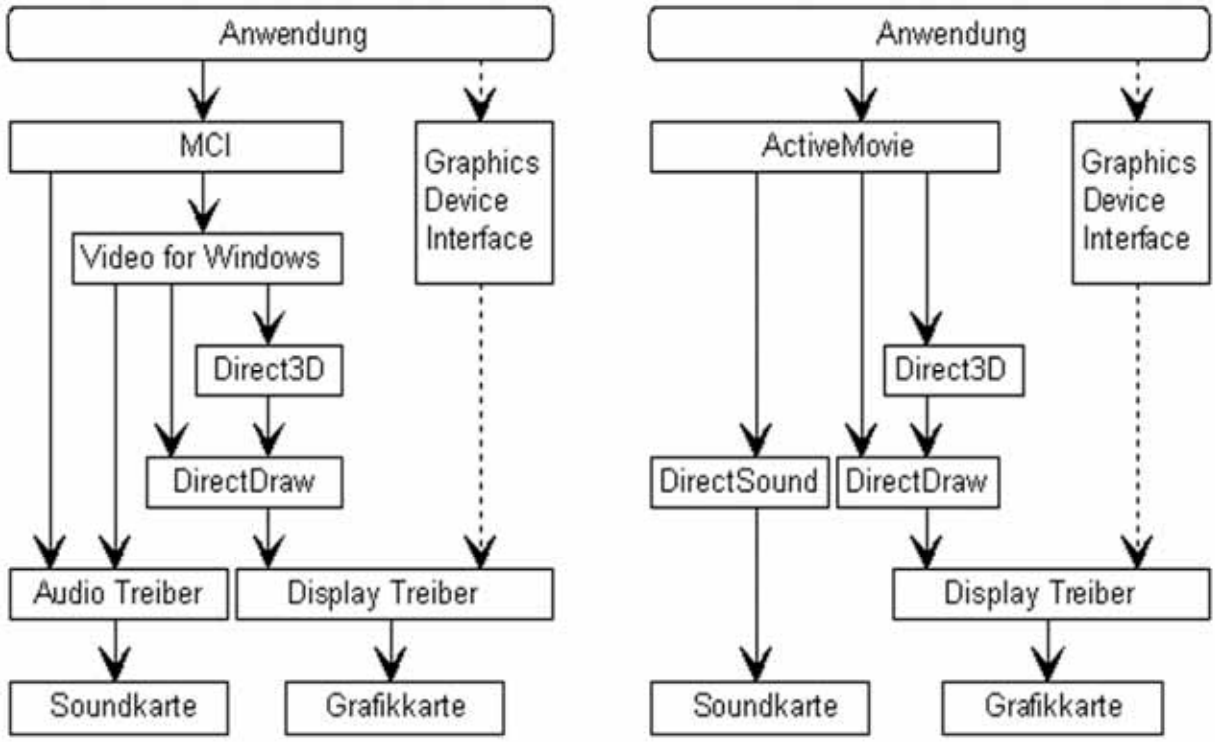


Abb. 3.20 Unterschied MCI/ActiveMovie

ActiveMovie decodiert das Video und hält es danach im Arbeitsspeicher, Dierect3D übernimmt den 3D-Grafikteil und DirectDraw erledigt den Rest. Wenn die Grafikkarte DirectDraw unterstützt, wird das decodierte Video an die Grafikkarte übergeben zur Konvertierung in den RGB-Farbraum und räumlichen Skalierung.

Für Internet-Anwendungen wurde ActiveMovie zur ActiveX-Technologie erweitert. Die Wiedergabe von Videos beginnt bereits, bevor das ganze Video über das Internet geladen ist. Surround Video erlaubt 360 Grad -Bilder als Hintergrund auf die mit Chroma-Keying Videos oder interaktive Objekte platziert werden können.

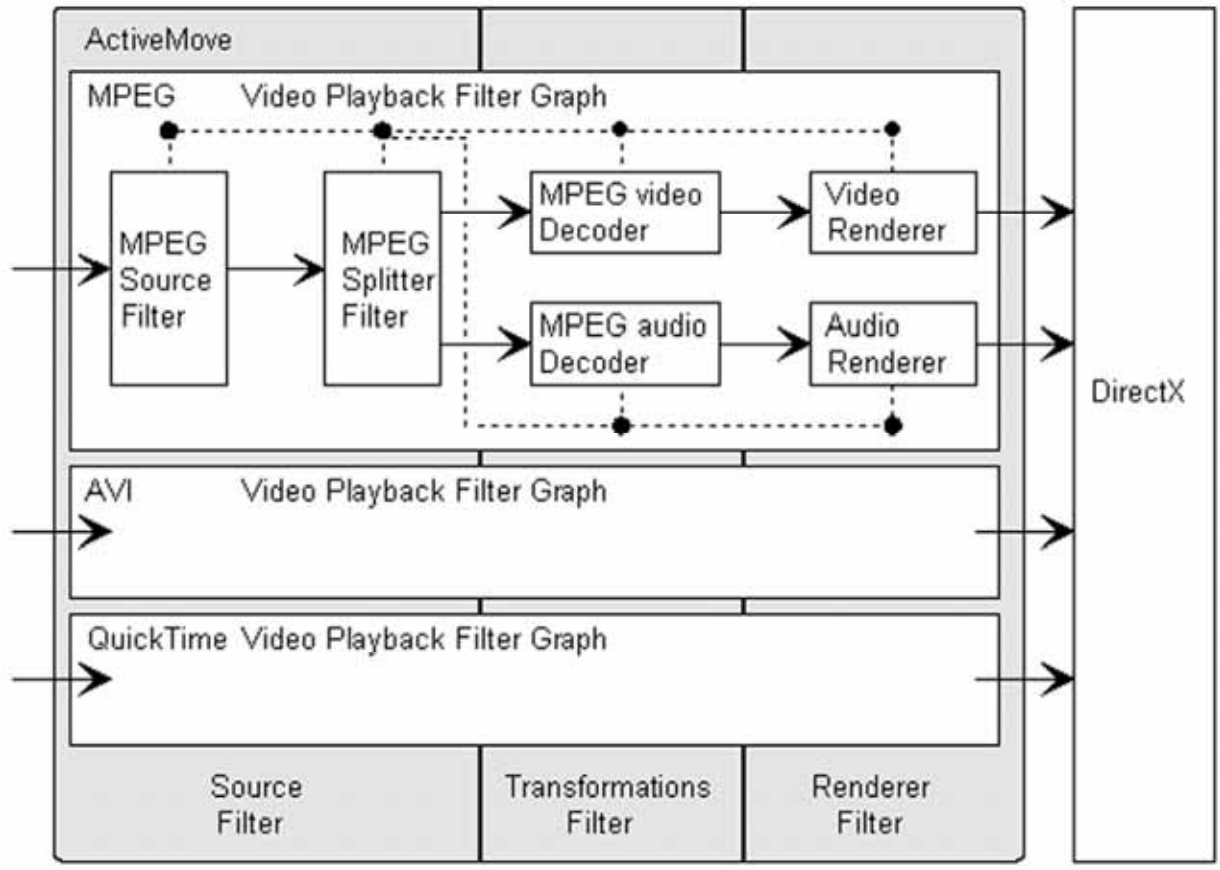


Abb. 3.21 Schnittstellen ActiveMovie

Die Source-Filter genannte Komponente stellt die Daten bereit von CD-ROM, Internet, Kabel-TV etc. Der Transform-Filter verarbeitet die Daten, dazu gehört die Dekomprimierung. Der Renderer-Filter liefert die weiter transformierten Daten an die Hardware-Geräte wie Videokarte, Soundkarte oder Speicher. Das Zusammenwirken verschiedener Filter nennt man einen Filter-Graph. ActiveMovie enthält Filter-Graphen für die verschiedenen Dekomprimierungsverfahren wie MPEG, AVI, Quicktime etc.; ein Filter-Graph-Manager entscheidet aufgrund ankommender Daten, welcher Filter-Graph zu verwenden ist. Es existieren vordefinierte Filter-Graphen, eigene können mit einem Filter-Graph Editor erstellt werden. ActiveMovie kann dynamisch aktuelle Kapazitäten ermitteln und den Filter-Graphen modifizieren oder die beteiligten Komponenten informieren/beeinflussen. Da alle Filter gemeinsame Resourcen nutzen, wie die CPU, bedeutet das: wenn z.B. der Source-Filter weniger Frames/s liefert, haben nachfolgende Filter weniger Arbeit, Resourcen werden weniger belastet und der Source-Filter kann wieder mehr Frames/s liefern. Diese Technologie wurde im Hinblick auf das Internet entwickelt.

Flash Video (flv)

Flash Video[5] (FLV) ist ein von Adobe Systems entwickeltes offenes Containerformat, das vornehmlich für Internetübertragungen von Videoinhalten genutzt wird. Es ist je nach verwendetem Codec kompatibel zu dem Adobe Flash Player in entsprechender Version. Derzeit unterstützt der Adobe Flash Player, und damit auch das Flash-Video-Format, die Video-Codecs Sorenson (eine MPEG-4 ASP H.263-Codec Variante), VP6 (von On2) und MPEG-4 nach dem H.264-Standard. Außerdem steht ein spezieller Screen-Capture-Codec zur Verfügung. Als Audio-Codecs sind MP3, Nellymoser Asao Codec, Speex, sowie der HE-AAC-Codec möglich.

Der Adobe Flash Player kann ab Version 6 FLV-Videos wiedergeben. Diese werden dazu von einem RTMP-Streaming-Server geladen. Ab Version 7 ist das Laden per HTTP-Protokoll möglich. Dadurch kann die FLV-Video-Datei von einem Webserver verbreitet werden.

Bis einschließlich Adobe Flash Player Version 7 kann in dem FLV-Video für das Bild nur der Sorenson-Codec und der Screen-Capture-Codec verwendet werden (Audio: MP3). FLV-Videos mit VP6-Codec können ab Version 8 abgespielt werden. Ab Unterversion 9.0.115 kann der Adobe Flash Player auch andere Videodateiformate als FLV abspielen. Es wird der H.264-Codec und HE-AAC Audio unterstützt, welches sich auch in anderen Videodatei-Containern befinden darf (mp4, flv, mov, 3gp, etc.).

Ein installiertes Wiedergabeprogramm für Adobe Flash-Dateien als Zusatzmodul (Plugin) im Webbrowser erreicht derzeit im Vergleich zu Plugins konkurrierender Formate wie Quicktime oder Real eine weitaus höhere Verbreitung. Flash-Video zeichnet sich infolgedessen dadurch aus, dass Videos auf Webseiten in diesem Format mit entsprechend mehr Rechnern angezeigt werden können. Auf diese Basis ist es zurückzuführen, dass bedeutende Videoportale wie YouTube, MyVideo, sevenload und clipfish auf Flash-Videos setzen.

Als Wiedergabesoftware existieren neben dem proprietären Adobe Flash Player mit Gnash aus dem GNU-Projekt und Swfdec Bemühungen, freie Alternativen zu schaffen.

Um Flash-Videos außerhalb des Webbrowsers abzuspielen, ist, unabhängig vom Flash-Player, die Installation eines Videoplayers (bzw. Codecs wie ffdshow), der das Flash-Video-Format unterstützt, erforderlich.

Dateien im Flash-Video-Format können unter anderem mit dem Realplayer, MPlayer, VLC media player, Winamp, FLV-Media Player oder dem Adobe Media Player wiedergegeben werden. Am Macintosh kann QuickTime mit Hilfe des Perian Codecpacks Flash-Video Dateien abspielen. Es existiert zudem auch ein Quick Look Plug-In.

[5] Vgl. http://de.wikipedia.org/wiki/Flash_Video

Videokonferenzen

Videokonferenzen erlauben neben der audio-visuellen Kommunikation auch die verteilte Bearbeitung von Dokumenten (document sharing). Vorraussetzung neben entsprechender Hardware (Video- und Audio-Komponenten zur Eingabe und Ausgabe) sind Kommunikationseinrichtungen und Software für Videokonferenzen, die Verbindungen aufbaut, für die Komprimierung und Verteilung der Daten sorgt, den Informationsaustausch regelt, Möglichkeiten der verteilen Bearbeitung von Dokumenten bietet und letztendlich eine bestmögliche Wiedergabe gewährleistet. Da die Kommunikation zwischen verschiedensten Systemen möglich sein muss, kommt der Interoperabilität größte Bedeutung zu. Um die Zusammenarbeit unterschiedlicher Geräte und verschiedener Hersteller zu ermöglichen, wurden von der ITU-T (International Telecommunications Union - Standardization Section) eine Reihe Empfehlungen (Recommendations) mit Standards für analoge Telefondienste, ISDN und Netzwerke herausgegeben. Neben der ITU ist u.a. das IMTC (International Multimedia Teleconferencing Consortium) und das NIST (National Institute for Standards and Technology) an der Etablierung und Validierung von Standards beteiligt.

ISDN mit Übertragungsraten mit 128 Kbits/s und analoge Telefondienste mit bis zu 56Kbits/s garantieren eine bestimmte Dienstgüte. LANs bieten zwar weitaus höhere Bandbreiten, aber da LANs wie das Internet verteilte Übertragungsmedien nutzen, können feste Bandbreiten und damit eine geforderte Dienstgüte (Quality of Service) über verteilte Leitungen nicht immer garantiert werden. Einen Überblick über die ursprünglichen Standards für Videokonferenzen gibt die Tabelle in Abb. 3.22.

Kommunikation via DSL ermöglicht natürlich mittlerweile auch Übertragungsraten bis zu 16000 Bit/s (z.T. auch darüber), so dass hier TV-Qualität erreicht werden kann.

Videokonferenzen existieren als Punkt-zu-Punkt oder als Multipunkt-Kommunikation. Sind mehr als zwei Teilnehmer an einer Videokonferenz beteiligt, verwaltet und kontrolliert eine MCU (Multipoint Conferencing Unit) die Verbindungen zu den Teilnehmern und sorgt für eine Priorisierung des sprechenden Teilnehmers. Die Tabelle zeigt bereits die wichtigsten Komponenten eine Videokonferenzsystems: Video- und Audiodaten werden wie die auszutauschenden Daten codiert und komprimiert und daraus ein Bitstrom generiert (Multiplexer, Bitstrom-Generator), in dem die einzelnen Teile in Frames gepackt und gemischt werden. Über die aufgebauten Verbindungen muss den Teilnehmenden Systemen ein gewünschter Dienst angezeigt werden, der Datenaustausch synchronisiert werden. Bei Konferenzen mit vielen Teilnehmern werden dafür auch höhere Dienste einer MCU benötigt. Übertragene Daten können verschlüsselt übertagen werden und bei einigen Systemen ist eine Steuerung der Kameras oder der Mikrophone möglich, die dem sprechenden Teilnehmer in

seiner Bewegung folgen.

Komponente / Standard / Netztyp	Video Codec	Audio Codec	Data Codec	Bitstrom Generator	Konferenz Aufbau Synchron.	Konferenz Kontrolle	Kamera Steuerung	Ver-schlüs-selung
H.320 ISDN	H.261	G.711 G.722 G.728	T.120	H.221	H.230 H.242	H.231 H.243	H.224 H.281	H.233 H.234
H.324 analoge Servicenetze	H.261 H.263	G.723.1	T.120 T.434 T.84	H.223	H.245		H.224 H.281	H.233 H.234
H.323 LAN	H.261 H.263	G.711 G.722 G.723.1 G.728 G.729	T.120	H.225.0	H.245	H.323	H.224 H.281	H.233 H.234
H.321 BISDN	H.261 H.263	G.711 G.722 G.728	T.120	H.221	Q.2931	H.231 H.243	H.224 H.281	H.233 H.234
H.322 Ethernet	H.261 H.263	G.711 G.722 G.728	T.120	H.221	H.242	H.231 H.243	H.224 H.281	H.233 H.234
H.310 BISDN	MPEG-2 H.261 H.263	MPEG-2 MPEG-1 G.711 G.722 G.728	T.120	H.221 H.222.1	H.245		H.224 H.281	

Abb. 3.22 Standards Videokonferenzen

H.320

Diese Gruppe von Standards wurde 1990 verabschiedet und definiert, wie Teilnehmer an Sprach- und Videokonferenzen über ISDN mit einer Bandbreite von 64Kbit/s bis 2Mbit/S kommunizieren. H.320 umfasst die einzelnen Standards

- **H.221** Spezifiziert, wie Video- und Audiodaten in Frames untergebracht werden bei einer Bandbreite von 64-1920 Kbit/s.
- **H.232** Spezifiziert die MCU (multipoint-control-unit) zur Kontrolle der Konferenz
- **H.261** Spezifiziert den Algorithmus zur Videokomprimierung mit den Auflösungen 352 × 288 (CIF) und 176 × 144 (QCIF)
- **H.242** Spezifiziert, wie die Verbindungen zwischen audio-visuellen Terminals aufgebaut werden über digitale Kanäle mit bis zu 1920Kbit/s
- **H.243** Spezifiziert, wie die Verbindungen zwischen 3 und mehr audio-

visuellen Terminals aufgebaut werden über digitale Kanäle mit bis zu 1920Kbit/s

- **H.230** Spezifiziert Signale und die Synchronisation der Frames
- G.xxx Spezifizieren Audio-Codecs
  - **G728** geringe Qualität für Nahbereich über einen Kanal mit bis zu 16Kbit/s.
  - **G.711** mittlere Qualität mit Bandbreite von 3.0KHz über Kanal mit bis zu 64Kbit/s
  - **G722** hohe Qualität mit Bandbreite von 7.5KHz über Kanal mit bis zu 64Kbit/s
- **H.233** Spezifiziert die Verschlüsselung der übertragenen Daten
- **H.234** Spezifiziert die Verschlüsselung der übertragenen Daten
- **H.224** Remote Video Camera Control

Niedere Bildraten bewirken ruckelnde Bewegungen, hohe Raten gestatten fließende Bewegungen. H.320 unterstützt die Auflösungen CIF (Common Intemediate Format) mit 352 x 288 Bildpunkten und QCIF (Quarter CIF) mit 176 x 144 Bildpunkten. Die Systeme werden gemäß der Qualität der Darstellung in drei Klassen eingeteilt:

	Klasse 1	Klasse 2	Klasse 3
Video Auflösung	QCIF	QCIF / CIF	CIF
Bilder/s	7.5	bis 15	bis 30
Bewegungscodierung	nein	teilweise	ja
Audio-Codec	G.711	G.722 / G.728	G.722 / G.728
Breitband	48-64 Kbit/s	48-64 Kbits/s	48-64 Kbits/s
Schmalband		16 Kbits/s	16 Kbits/s

Abb. 3.23 H.320-Klassen

Systeme unterschiedlicher Klassen können untereinander kommunizieren, allerdings mit der Qualität des schwächsten Terminals. Bewegungskompensation wird nur von Systemen der Klasse 3 voll unterstützt, von Systemen der Klasse 1 überhaupt nicht. Der Standard für Audio-Codierung ist G.711 und spezifiziert PCM (Pulse Code Modulation) mit einer Rate von 8000 samples/s. Jedes sample wird einem von 212 Werten zugeordnet und dann logarithmisch komprimiert. Das analoge 3.4 KHz Telefonsignal wird in 56-48 Kbit/s konvertiert. Der Standard G.722 komprimiert ein 7 KHz Audiosignal in 56 Kbits/s.

H.310, H.321, H.322

Diese 3 Standards wurden 1995 verabschiedet. H.321 beschreibt die technischen Spezifikationen für den Anschluß von Schmalband-ISDN-Bildtelefonen nach H.320 an Breitband-ISDN (BISDN) und ATM-Netzwerke. H.321-Terminals für BISDN können mit H-320-Terminals für ISDN kommunizieren. H.321 enthält die Video-Codecs H.261 und H.263. Der Standard H.310 enthält zudem den MPEG-2-Codec. H.322 basiert auf H.320 und spezifiziert die technischen Anforderungen an Bildtelefone im Schmalband-Bereich in paketorientierten Netzwerken. H.322 garantiert die Bandbreite auch dann, wenn die Übertragungswege über ein oder mehrere LANs reichten.

H.323

Der Standard wurde 1996 verabschiedet und erweitert H.320 für Netzwerke wie LANs oder Internet, die nicht die Quality of Service bieten können. H.323 unterstützt Punkt-zu-Punkt und Mehrpunkt-Verbindungen. H.323 erlaubt Netzadministratoren über einen Gatekeeper die Übertragungslast für Videodaten zu verwalten, um Quality of Service sicherzustellen und einen LAN/H.320-Gateway, damit H.323-Terminals mit H320- und H.324-Terminals zusammenarbeiten können.

Um Telephon und Netzwerke zu verbinden, definiert H.323 Gateways, Gatekeeper, MCUs und Terminals, die als Clients in einem Netzwerk agieren. Der Standard H.323 greift die Entwicklung des Internets, der Intranets und LANs ausgelöst durch den Standard H.320 von 1990 auf und integriert das Telephon. H.323 basiert auf dem RTP (RealTime Protocol) der IETF (Internet Engineers Task Force) und kann damit genügend Bandbreite im Internet für Videokonferenzen bereitstellen.

Die Gateways in H.323 definieren die Integration von H.323-Terminals in analoge PSTN-Terminals (Public Switching Telephone Network), H.320 kompatible Terminals über ISDN und H.324-Terminals über analoge Telekommunikationsdienste. Die Aufgaben der Gateways sind die Übersetzung der Übertragungsformate H.225.0 und H.221, Übersetzung zwischen den Kommunikationsprozeduren H.245 und H.242, Übersetzung zwischen den verschiedenen Audio- und Video-Codecs sowie automatischen Verbindungsauf- und Abbau im LAN und Telephonnetzwerk.

Videokonferenzen benötigen hohe Bandbreiten. Mit dem Gatekeeper wurde ein Management der verfügbaren Bandbreite eingeführt. Netzwerkadministratoren können die Anzahl Nutzer oder die Bandbreite für eine H.323 Verbindung begrenzen. Gatekeeper übersetzen die "Telephonnummern" oder symbolischen Namen der Terminals in IP- oder IPX-Adressen. Ein einzelner Gatekeeper kann für einen ganzen H.323-Bereich Terminals, Gateways und MCUs verwalten.

H.324
Der Standard wurde 1996 verabschiedet und gestattet Videokonferenzen über analoge Signalwege mit einem Leistungsspektrum wie bei H.320. Als Video-Codec wird der hier besser geeignete Standard H.263 verwendet. H.263 ist eine verbesserte Version des H.261 Standards erweitert um 128x96 SQCIF (sub-QCIF). Mit einem 28.8 oder 36.6 V.34-Modem können über analoge Signalwege dieselben Frame-Raten erreicht werden wie nach H.320 über ISDN. Der Konferenz-Standard H.324 umfasst:

- **H.263** Spezifiziert die Video-Codierung
- **H.223** Spezifiziert das Multiplex-Protokoll für einfache Multimedia-Terminals
- **H.245** Spezifiziert Controls für die Kommunikation von Multimedia-Terminals
- **G.723.1** Spezifiziert die Sprachcodierung für Multimedia Telekommunikations Terminals mit 5.3 bis 6.3 Kbit/s. V.34-Modes sind ein wichtiger Bestandteil der Verbindungen der Terminals
- **H.233, H.234** Spezifizieren optionale Verschlüsselung der Daten
- **H.224, H.281** Spezifizieren remote video camera control
- **T.84** Komprimierung und Codierung von Standbildern
- **T.434** Spezifiziert ein Dateiübertragungsformat für verteilte Dokumente

Der Standard H.324 unterstützt Whiteboard-Sessions und T.120 Applikationen. Daher besitzen Videodaten hier niedere Priorität. Eine Session wird durch das H.245 Multimedia System Protokoll verwaltet, das auch den Austausch mit LANs handhabt. Mit H.245 werden logische Kanäle definiert, was die Anwendungsentwicklung deutlich vereinfacht hat. Der H.263 Standard spezifiziert 5 Bildformate:

Format	Auflösung
sub-QCIF	128 x 96
QCIF	176 x 144
CIF	352 x 288
4CIF	702 x 576
16CIF	1408 x 1152

Abb. 3.24 H.324-Auflösungen

Speicherformate und -medien

Die Compact Disk (CD) hat eine Entwicklung hinter sich, die diverse Standards hervorgebracht hat, meist initiiert von einzelnen Unternehmen oder Unternehmensgruppen. Ein Überblick über gängige Standards gibt die Tabelle:

Standard	Spezifikation Hersteller	Beschreibung
CD-DA	Red Book Philips, Sony	Digital Audio, Zugriffsrate 150 Kbits/s, Abtastrate 44,1 Ksamples/s, Kapazität 682 Mbytes
CD-ROM	Yellow Book Sony	Mode 1: Text und Daten, Mode 2: Audio- und Videodaten Teile der CD können Mode 1besitzen, andere Mode 2. Die Zugriffsrate beträgt bis zu 32 xž 150 Kbits/s
CD-I	Green Book Philips, Sony	Die CD-Interaktive enthält Text, Grafik-, Audio- und Videodaten, gestattet Spiele und Wiedergabe von Multimedia in Echtzeit. CD-I unterstützt den CD-DA Standard, erfordert spezielle Hardware mit Betriebssyste OS9, gibt CD-I, CD-DA, Video CDs und PhotoCDs wiede
CD-XA	Extended Yellow Book Philips, Sony, Microsoft	Die CD-ROM Extended Architecture erhöht die Kapazität, enthält Video und Audio gemischt wie bei CD-I, Text oder Daten. CD-XA ist kompatibel zu CD-ROM, CD-I, PhotoCD und MPC Level 2, multisessionfähig.
Video CD	White Book Philips, JVC	74 Minuten Video, MPEG-1 komprimiert.
CD-R	Orange Book Philips, Sony	CD-Recordable, um Daten, Bilder, Video oder Audio zu speichern, entspricht dann einer CD-ROM.
PhotoCD	Kodak	Enthält bis zu 100 komprimierte Bilder im CD-XA Format
DVD		Digital Video Disk (auch Digital Versatile Disk), Kapazität von 4,7 bis 17 GBytes, für beliebige Daten

Abb. 3.25 Speichemedien

DVD gestattet Videos im 16:9 Format in HDTV-Qualität und Dolby AC-3, Interaktivität und besitzt hohe Zugriffsraten. DVD besitzen eine kleinere pit-Größe und daher eine größere Kapazität. DVD-Leser besitzen einen Laser mit 635 bis 650 Nanometer, CD-ROMs einen mit 780 Nanometer. Es gibt die Varianten

- single sided single layerd 4.7 GByte
- double sided single layerd 9.4 GByte
- single sided double layerd 8.5 GByte
- double sided double layerd 17.0 Gbyte
- Blu Ray 25 GB single und 50 GB double layerd

Die DVD-Formate unterscheiden sich zur Zeit in DVD-RAM, DVD-R, DVD+R, DVD-RW, DVD+RW, Blu Ray und HD DVD.

Es sind allerdings auch schon Prototypen von DVD-Nachfolgern wie die **Holographic Versatile Disc** (**HVD)**, welche bis zu 3,9 TerraByte umfassen kann.

Grafikformate

Man unterscheidet grundsätzlich zwischen Rastegrafik und Vektorgrafik. Es seien aus jeder Gruppe repräsentativ einige bekannte Formate kurz beschrieben.

Rastergrafik

SGI

SGI steht für Silicon Graphics Image. Dieses Format wurde von Silicon Graphics Computer Systems entwickelt zusammen mit einer Bibliothek von Operationen. Die Grundstruktur einer SGI-Datei besteht bei unkomprimierten Bildern aus Header und Bilddaten, bei Lauflängen-codierten Bildern aus Header, Offset Tabellen und Bilddaten. Der Header enthält die Informationen zur Interpretation der Bilddaten mit den Feldern:

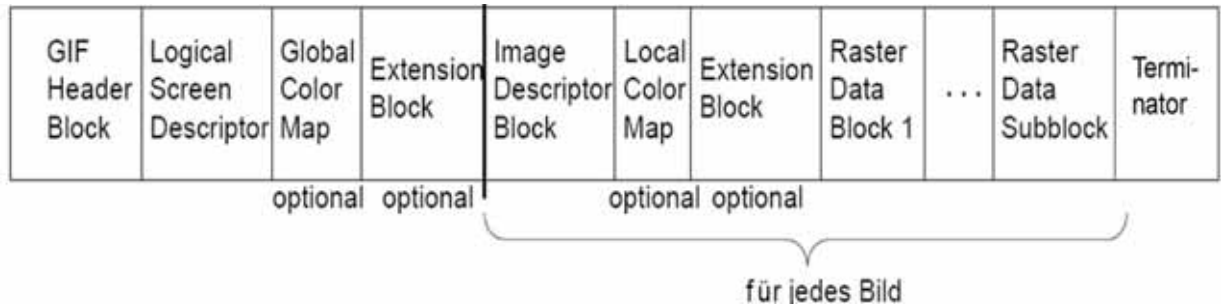
Feld	Bedeutung	Werte
MAGIC	Signatur	Wert für SGI-Datei immer 474
BPC	Farbtiefe	1 = 28 Farben, 2 = 216 Farben je Pixel und Kanal
DIMENSION	Dimensionen	1 = 1 Farbkanal mit 1 Zeile der Länge XSIZE 2 = 1 Farbkanal mit YSIZE Zeilen der Länge XSIZE 3 = ZSIZE Kanäle mit YSIZE Zeilen der Länge XSIZE
XSIZE	Anzahl Spalten	
YSIZE	Anzahl Zeilen	
ZSIZE	Anzahl Kanäle	Die Anzahl Farbkanäle ist prinzipiell nicht beschränkt, Zusätzlich zu den RGB-Kanälen führen Alpha-Kanäle, Filter-Kanäle usw. zu Werten > 3)
PIXMIN	kleinster Farbwert	
PIXMAX	größter Farbwert	
DUMMY	Füllbyte(s)	
IMAGENAME	Name	Nullterminierter String (80 Zeichen)
COLORMAP	Farbpalette	0 = NORMAL: B/W für einen Kanal, RGB für 3 Kanäle usw. 1 = DITHERED: 1 Farbkanal, RGB-Daten in 1 Byte codiert 2 = SCREEN: 1 Kanal, Pixelwert ist Index in Farbtabelle 3 = COLOMAP: es liegt eine Farbtabelle vor, kein Bild
DUMMY	Füllbyte(s)	

Abb. 3.26 Aufbau SGI

Sind Bilder lauflängen-codiert gespeichert, sind die Offsets der codierten Zeile in den Offset-Tabellen hinterlegt, die Zeilen können nichtsequentiell abgelegt sein.

GIF

Das Graphics Interchange Format (GIF) definiert die Codierung von Rastergrafiken mit bis zu 256 Farben. Die erste Version wurde 1987 als GIF87a definiert, 1989 die erweiterte Version GIF89a. Die verwendete variable Lauflängen-Codierung nach Lempel-Ziv-Welch (LZW) verwendet Codes variabler Länge zwischen 3 und 12 Bits, um Bitmuster der Originaldaten zu ersetzen. Bei der Codierung wird dynamisch eine Übersetzungstabelle aus Bitmustern erzeugt, die in den Originaldaten bisher erkannt wurden. Jedes neue Muster wird in diese Tabelle aufgenommen. Anstelle des Bitmusters wird der gefundene Index in den Datenstrom aufgenommen. GIF stellt einen effizienten one-pass Codierer und Decodierer bereit, Decodierung und Darstellung kann gleichzeitig erfolgen. GIF89a erlaubt mehrere Bilder in einer GIF-Datei. Struktureller Aufbau einer GIF-Datei:



GIF erhält scharfe Bildkanten und besitzt eine hohe Komprimierungsrate für Bilder mit wenigen Farben. Bei photorealistischen Bildern mit weichen Farbübergängen ohne allzu scharfe Kanten ist JPEG vorteilhafter.
Mehrere GIF-Bilder lassen sich auch zu eine GIF-Animation zusammenfassen, wo die einzelnen Bilder hintereinander wie ein Film ablaufen.

TIFF

Das TIFF-Format wurde u.a. von Aldus, Hewlett Packet und Microsoft entwickelt. Vorteile von TIFF sind, dass es ohne Qualitätsverlust gespeichert und wieder geladen werden kann. Außerdem können viele zusätzliche Parameter mit abgespeichert werden. TIFF-Dateien sind vom Ausgabegerät unabhängig und unterstützen mehrere Farbmodi. Allerdings sind TIFF-Dateien relativ groß. Beim Öffnen von TIFF-Dateien können Probleme auftreten, da nicht alle Programme jede Variante des Formats unterstützen.

Eine TIFF Datei besitzt folgenden Aufbau[6]:

[6] vgl. Born, G.: *Referenzhandbuch Dateiformate*, 1997, Addison Wesley

Die Struktur wird durch die Image File Dicectory (IFD) bezeichneten Blöcke geprägt. Sie beinhalten Informationen betreffs des gespeicherten Datentyps der Bilddaten oder des Grafikmodus. Da die Bilddaten in den freien Bereichen innerhalb der Datei gespeichert werden, ist der Aufbau einer TIFF Datei flexibel und es können mehrere Bilder oder verschiedene Varianten eines Bildes innerhalb einer Datei gespeichert werden.
Der Header ist fix und besitzt immer die ersten 8 Byte der Datei:

2 Byte	Byte Order „II“ = Intel „MM“ = Motorola
2 Byte	Version number
4 Byte	Pointer to first IFD

Die ersten beiden Bytes enthalten eine Signatur für das verwendete Speichermodell. In den nächsten 16-Bit befindet sich die Versionsnummer der verwendeten TIFF Datei. Die letzten 4 Bytes des Headers enthalten den Kopfzeiger auf den Begin des ersten IFD. Die Daten in einer TIFF Datei können beliebig angeordnet sein. Dies wird durch die IFDs ermöglicht. Ein IFD fungiert dabei als „Inhaltsverzeichnis“. Alle IFDs sind mit Zeigern verkettet. Der Zeiger auf den nächsten IFD befindet sich innerhalb der IFD-Datenstruktur.

2 Byte	Zahl der Einträge
12 Byte	Tag 0
12 Byte	Tag 1
…	
12 Byte	Tag n
4 Byte	Zeiger nächster IFD

Die Länge eines IFDs ist nicht festgelegt und wird durch die Anzahl der Tag-Einträge bestimmt. Diese ist in den ersten 2 Byte eines IFDs festgehalten. Tags bestehen immer aus 12 Byte. Die letzten 4 Byte repräsentieren den Zeiger auf das nächste IFD. Falls kein IFD folgt wird dieses Feld mit dem Wert 0 besetzt.
Tags dienen zur Aufnahme von Daten über Bildabmessung, Pixelauflösung etc. Sind es zu viele Daten für einen Tag werden diese im freien Bereich innerhalb der Datei ausgelagert. Im Tag befindet sich dann ein Zeiger auf diesen Datenbereich.

2 Byte	Tag Typ
2 Byte	Datentyp
4 Byte	Länge Datenbereich
4 Byte	Zeiger auf Datenbereich oder Wert

Die ersten 2 Bytes geben den Tag Typ (Image Organisations-Tags oder Image Pointer-Tags oder Pixel Description-Tags, usw.) an. In den nächsten 2 Byte folgt den Datentyp (z.B. Byte, ASCII, SHORT, LONG, RATIONAL, usw.) der gespeicherten Werte. Die nächsten 4 Byte speichern die Länge des dazugehörigen Datenbereichs. Die letzten 4 Byte schließlich dienen zur Aufnahme des jeweiligen Wertes (z.B. die Auflösung der X-Achse). Wenn diese 4 Byte nicht ausreichen um den Wert abzuspeichern, enthält das Feld einen 4 Byte großen Zeiger der auf den Datenbereich zeigt, der an einer freien Stelle der TIFF Datei angelegt ist.

BMP
BMPs gibt es in verschiedenen Versionen[7]. Die meisten BMP-Dateien liegen in der Version 3 vor; es gibt keine früheren Versionen. Die späteren Versionen 4 und 5 sind höchst selten anzutreffen. Windows-Bitmaps (der Version 3) erlauben Farbtiefen von 1, 4, 8, 16, 24 oder 32 Bits. Alphakanäle, Farbkorrektur und Metadaten werden nicht unterstützt. Windows-Bitmaps werden entweder unkomprimiert oder verlustfrei mit RLE-Komprimierung (Lauflängenkodierung) gespeichert. Dies ist ein eher schwaches Verfahren, sodass BMP-Dateien wesentlich größer sind als andere Formate und kaum für das World Wide Web genutzt werden. Dafür ist das BMP- Format relativ einfach aufgebaut. BMPs sind vor allem im Windows-Umfeld weit verbreitet; gängige Grafiksoftware

[7] vgl. http://de.wikipedia.org/wiki/Windows_Bitmap

unterstützt das Format problemlos (mit Ausnahme der eher exotischen Farbtiefen 16 und 32 Bits).

Aufbau (Version 3)[8]:
Der „Meta"-Aufbau einer BMP-Datei sieht so aus:

Dateikopf (BITMAPFILEHEADER)
Informationsblock (BITMAPINFO): **Bitmap-Eigenschaften** (BITMAPINFOHEADER) Eventuell: **Farbmasken** Eventuell: **Farbtabelle**
Eventuell: Ungenutzter Platz
Bilddaten
Eventuell: Ungenutzter Platz

Im Folgenden bezeichnet WORD einen 16-Bit-vorzeichenlosen Integer, DWORD einen 32-Bit-vorzeichenlosen Integer und LONG einen im Zweierkomplement codierten 32-Bit-Integer. BMP verwendet die sog. Little-Endian-Konvention (Ein Verfahrung zum effektiven Abspeichern von Bits und wohldefinierten Byte-Reihenfolgen).

BITMAPFILEHEADER (Größe: 14 Byte)			
Offset (Byte)	**Datentyp**	**Name**	**Inhalt**
0	WORD	bfType	ASCII-Zeichenkette "BM" (Dezimalwert 19778).
2	DWORD	bfSize	Größe der BMP-Datei in Byte. (unzuverlässig)
6	DWORD	bfReserved	0
10	DWORD	bfOffBits	Offset in Byte der Bilddaten vom Beginn der Datei an.

[8] ibid.

BITMAPINFOHEADER (Größe: 40 Byte)			
Offset (Byte)	**Datentyp**	**Name**	**Inhalt**
0	DWORD	biSize	40 (Größe des Informationsblocks in Byte)
4	LONG	biWidth	Breite der Bitmap in Pixel.
8	LONG	biHeight	Der Betrag gibt die Höhe der Bitmap in Pixel an. Ist der Wert positiv, so ist die Bitmap eine sogenannte "bottom-up"-Bitmap (die Bilddaten beginnen mit der untersten und enden mit der obersten Bildzeile). Dies ist die gebräuchlichste Variante. Ist der Wert negativ, so ist die Bitmap eine "top-down"-Bitmap (die Bilddaten beginnen mit der obersten und enden mit der untersten Bildzeile).
12	WORD	biPlanes	1 (Stand in einigen älteren Formaten wie PCX für die Anzahl der Farbebenen, wird aber für BMP nicht verwendet)
14	WORD	biBitCount	Gibt die Farbtiefe der Bitmap in bpp an; muss einer der folgenden Werte sein: 1, 4, 8, 16, 24 oder 32. Bei 1, 4 und 8 bpp sind die Farben indiziert. 16 und 32 bpp sind ungebräuchlich.
16	DWORD	biCompression	Einer der folgenden Werte: 0 (BI_RGB): Bilddaten sind unkomprimiert. 1 (BI_RLE8): Bilddaten sind lauflängencodiert für 8 bpp. Nur erlaubt wenn biBitCount=8 und biHeight positiv. 2 (BI_RLE4): Bilddaten sind lauflängencodiert für 4 bpp. Nur erlaubt wenn biBitCount=4 und biHeight positiv. 3 (BI_BITFIELDS): Bilddaten sind unkomprimiert und benutzerdefiniert (mittels Farbmasken) codiert. Nur erlaubt wenn biBitCount=16 oder 32; ungebräuchlich.

20	DWORD	biSizeImage	*Wenn biCompression=BI_RGB:* Entweder 0 oder die Größe der Bilddaten in Byte. *Ansonsten:* Größe der Bilddaten in Byte.
24	LONG	biXPelsPerMeter	Horizontale Auflösung des Zielausgabegerätes in Pixel pro Meter; wird aber für BMP-Dateien meistens auf 0 gesetzt.
28	LONG	biYPelsPerMeter	Vertikale Auflösung des Zielausgabegerätes in Pixel pro Meter; wird aber für BMP-Dateien meistens auf 0 gesetzt.
32	DWORD	biClrUsed	*Wenn biBitCount=1:* 0. *Wenn biBitCount=4 oder 8:* die Anzahl der Einträge der Farbtabelle; 0 bedeutet die maximale Anzahl (16 bzw. 256). *Ansonsten:* Die Anzahl der Einträge der Farbtabelle (0=keine Farbtabelle). Auch wenn sie in diesem Fall nicht notwendig ist, kann dennoch eine für die Farbquantisierung empfohlene Farbtabelle angegeben werden.
36	DWORD	biClrImportant	*Wenn biBitCount=1, 4 oder 8:* Die Anzahl sämtlicher im Bild verwendeten Farben; 0 bedeutet alle Farben der Farbtabelle. *Ansonsten:* *Wenn eine Farbtabelle vorhanden ist und diese sämtliche im Bild verwendeten Farben enthält:* deren Anzahl. *Ansonsten:* 0.

Diese Beispiele mögen für die Rastergrafiken genügen. Natürlich gibt es viele weitere solcher Formate, PNG und JPEG zählen auch dazu.

Vektorgrafik

Vektorgrafiken[9] basieren anders als Rastergrafiken nicht auf einem Pixelraster, in dem jedem Bildpunkt ein Farbwert zugeordnet ist, sondern definieren sich über eine Bildbeschreibung mittels mathematischer Funktionen. So kann beispielsweise ein Kreis in einer Vektorgrafik über Lage des Mittelpunktes, Radius, Linienstärke und Farbe vollständig beschrieben und ohne Qualitätsverlust beliebig skaliert und verzerrt werden, etwa mittels homogener Koordinaten. Die Stärke von Vektorgrafiken liegt daher vor allem im Bereich von Darstellungen,

[9] vgl. http://de.wikipedia.org/wiki/Vektorgrafik

denen geometrische Primitive zugrunde liegen, wie Diagramme oder Logos. Die Erstellung von Vektorgrafiken erfolgt meist mittels eines Vektorgrafikprogramms oder direkt mit einer Auszeichnungssprache. Rastergrafiken können mit gewissen Einschränkungen in Vektorgrafiken umgewandelt werden (Vektorisierung).
Mittlerweile bieten gängige Vektorgrafikprogramme Funktionen an, die es erlauben, Vektorzeichnungen zusammen mit feinen Farbverläufen und Transparenzen zu speichern und somit auch realistischere Ergebnisse zu erzeugen. Die generierten Grafiken bleiben dabei trotzdem skalierbar und veränderbar.

Abb. 3.27 Vektorgrafik vs. Rastergrafik[10]

[10] Quelle: ibid.

Beispiele für Vektorgrafiken sind Schrift-Fonts oder das EPS-Format, welches nachfolgend repräsentativ für allg. Vektorgrafiken näher beleuchtet wird.

EPS

Die Sprache Postscript (PS) wurde ursprünglich von Adobe definiert und mehrfach überarbeitet bzw. erweitert. Encapsulated Postscript (EPS) besteht aus einer Untermenge der Postscript-Sprache. EPS-Dateien können in Postscript-Dateien eingebettet sein. EPS wird auf den Plattformen Macintosh, MS-DOS, MS-Windows, UNIX u.a. unterstützt. Postscript wurde entwickelt als Sprache zur Beschreibung einer Dokumentenseite, die Text, Vektorgrafik und Rastergrafik enthalten kann. Postscript ist eine plattformunabhängige universelle Programmiersprache mit Befehlen zur Generierung einer Grafik auf einem Ausgabegerät, das EPS-Befehle interpretieren kann oder von einem EPS-Interpreter unterstützt wird. Postscript unterstützt Schwarz/Weiß-, Grauton- und Farbbilder in einer Auflösung von 75 bis über 3000 dpi. Alle Informationen werden im 7-Bit ASCII-Format abgelegt.
Den eigentlichen EPS-Befehlen geht ein aus Kommentaren bestehender EPS-Header voraus. Jede mit einem %-Zeichen beginnende Zeile wird als Kommentar betrachtet und normalerweise vom Postscript-Interpreter ignoriert. Eine typischer EPS-Header ist

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: Demo einfacher Befehle
%%Creator: M.Wetzer
%%CreationDate: 01/10/98 13:20:00
%%Bounding Box 50 50 500 700
%%EndComments
```

Ein EPS-Header beginnt stets mit %!PS-Adobe-3.0 EPSF-3.0 und endet mit %%EndCommments. Dem Text PS-Adobe folgt die Kennzeichnung des unterstützten Levels der Strukturkonventionen für PS-Dokumente, z.B. 2.0 oder 3.0. Die Versionsnummer hinter dem Text EPSF zeigt den unterstützten Level der EPS-File Specifications an, z.B. 4.0. Die Bounding Box gibt die Bildgröße in PICA an, die Koordinaten beziehen sich auf den Nullpunkt links unten.
Postscript Befehle werden in umgekehrt polnischer Notation (=klammerfreie Ausdruckweise für Formeln) angegeben. Die Befehle umfassen die Ausgabe grafischer Elemente, Transformationen des Koordinatensystems, Textausgabe, Ausgabe von Bitmaps uvm. Als Programmiersprache besitzt Postscript Datentypen, Variablen, Kontrollstrukturen wie if- und for-Anweisungen, Funktionen usw. Ein Charakteristikum von Postscript sind Pfade: Ein Pfad definiert eine Grafik. Zeichenoperationen werden zunächst nicht ausgeführt, sondern in einem

Pfad gespeichert, Zeichenposition ist der letzte Punkt. Ein Pfad endet mit dem letzten Befehl vor der Ausgabe mit stroke oder er wird mit closepath geschlossen (letzte Linie wird automatisch hinzugefügt). Nach stroke wird der Pfad gelöscht, der Bezugspunkt ist wieder undefiniert. Der Befehl stroke gibt den Pfad in den Bildspeicher aus, erst showpage druckt die so aufbereitete Seite, löscht die Seite im Bildspeicher und setzt alle Parameter auf Standardwerte. Hierzu ein einfaches Beispiel:

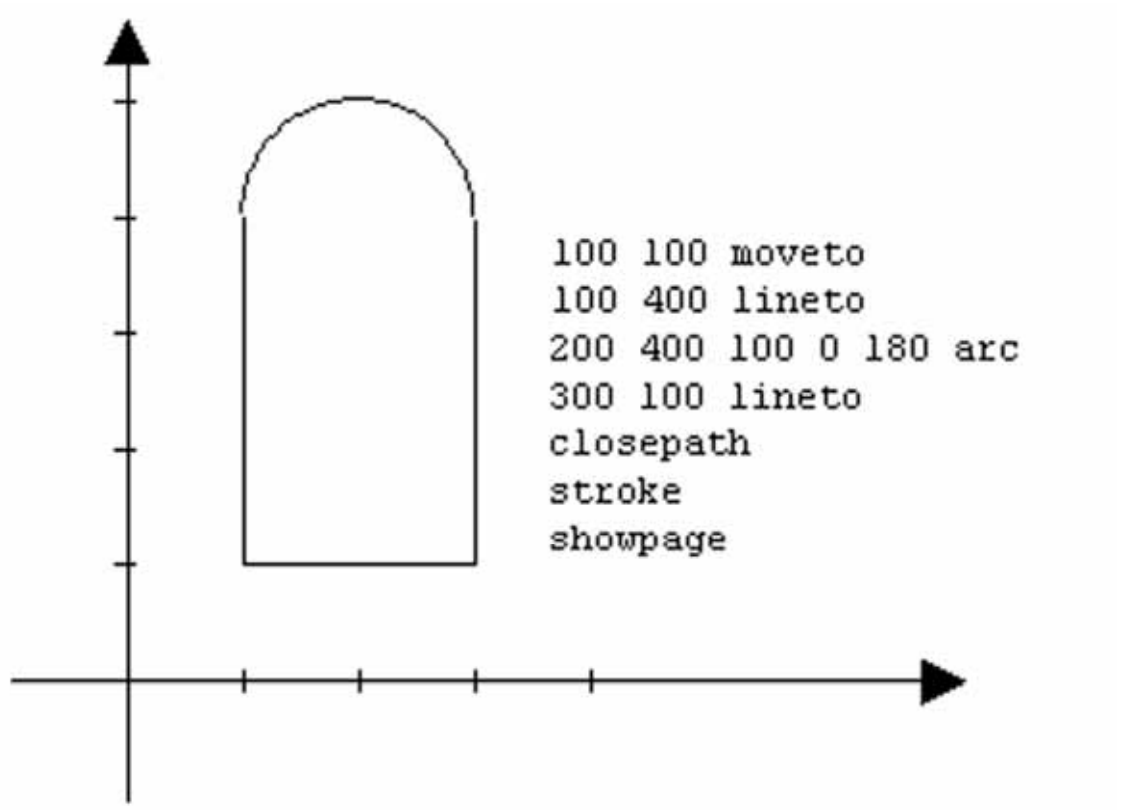


Abb. 3.28 EPS-Grafik

Weitere Video- und Audio-Formate

Abschließend werden noch einige Begriffe und Formate angegeben, welche speziell im Audio- und Video-Sektor eine wichtige Rolle spielen.

Indeo und DVI

Grundlage des Intel Video (Indeo) Codec ist die Digital Video Interactive Technology (DVI). Die DVI-Technologie (siehe unten) erstreckt sich auf Daten, Grafiken, Einzelbilder, Bildfolgen und Audiodaten, legt ein gleichnamiges Datenformat, eine Benutzerschnittstelle und Codier- und Decodieralgorithmen fest. DVI-Systeme existieren auch als Hardware-Lösung. An solche DVI-Systeme können Signale verschiedener Normen angelegt sein wie FBAS oder RGB. Analoge NTSC- und PAL-Signale können konvertiert und gespeichert werden. Intern arbeitet ein DVI-System mit einem modifizierten YUV-Signal in 9-Bit,

16-Bit oder 24-Bit YUV-Format. DVI erlaubt mit Unterstützung eines entsprechenden Hardware-Chipsatzes eine Datenrate von 30 Frames/s mit bis zu 1920x1200 Pixel. Die Software-Komprimierung kennt 3 Qualitätsstufen, die abhängig von der vorhandenen Hardware verwendet werden können. Eine wichtige Eigenschaft ist daher die Skalierbarkeit der Qualität: je mehr CPU-Leistung, um so besser das Ergebnis, die Frame-Rate hingegen kann stabil gehalten werden. DVI unterscheidet zur Codierung von Bewegtbildern die Verfahren PLV (Production Level Video) und RTV (Real Time Video). Die Komprimierungsrate kann bei bis zu 160:1 liegen. In QuickTime Video for Windows ist Indeo lediglich mit dem RTV-Verfahren integriert.
PLV ist ein asymmetrisches, verlustbehaftetes Verfahren zur Komprimierung. Nach PLV codierte Videos besitzen eine hohe Qualität, benötigen aber einen hohen Zeitaufwand zur Codierung.
RTV ist ein symmetrisches Verfahren, das mit Hardware oder Software in Echtzeit arbeitet.

RIFF
Das herstellerspezifische Resource Interchange File Format (RIFF) von Microsoft stellt einen Formatrahmen für Multimediadaten dar, der verschiedene Formate für Audio- Video- oder Grafikdaten aufnehmen kann. RIFF-Dateien können neben den Audio- und Videodaten auch Informationen über Ein- und Ausgabegeräte enthalten. Welcher Art Daten in einer RIFF-Datei abgelegt sind, wird auch durch die Erweiterung im Dateinamen festgelegt:

- **AVI** Audio- und Videodaten im audio-video-interleaved Format für VfW
- **WAV** Audio Wave (Windows Wave Files)
- **RDI** Bitmap
- **RMI** MIDI (siehe unten)
- **BND** Verbund aus anderen RIFF-Dateien

RIFF-Dateien besitzen einen strukturierten Aufbau aus Informationseinheiten (Chunks) ohne feste Position. Chunks enthalten Strukturbeschreibungen, auf Wortgrenze ausgerichtete Datenströme und eventuelle andere Informationseinheiten, „Subchunks“ genannt. Jeder Chunk besteht aus einer Chunk-Identifikation, der Angabe der Chunk-Größe in Byte und den Chunk-Daten selbst. Eine RIFF-Datei stellt ebenfalls einen Chunk dar mit einem Header, gefolgt von Subchunks mit den eigentlichen Daten. Der RIFF-Header enthält die Signatur, Dateigröße in Byte und den RIFF-Typ. Die Signatur ist 'RIFF', wenn die Daten im Intel-Format (Little Endian) gespeichert wurden und 'RIFX' für Daten im Motorola-Format (Big Endian).

AVI

AVI Dateien, eingeführt mit Video for Windows, entsprechen der RIFF-Spezifikation und enthalten den RIFF-Header mit RIFF-Typ 'AVI' gefolgt von mehreren Chunks:

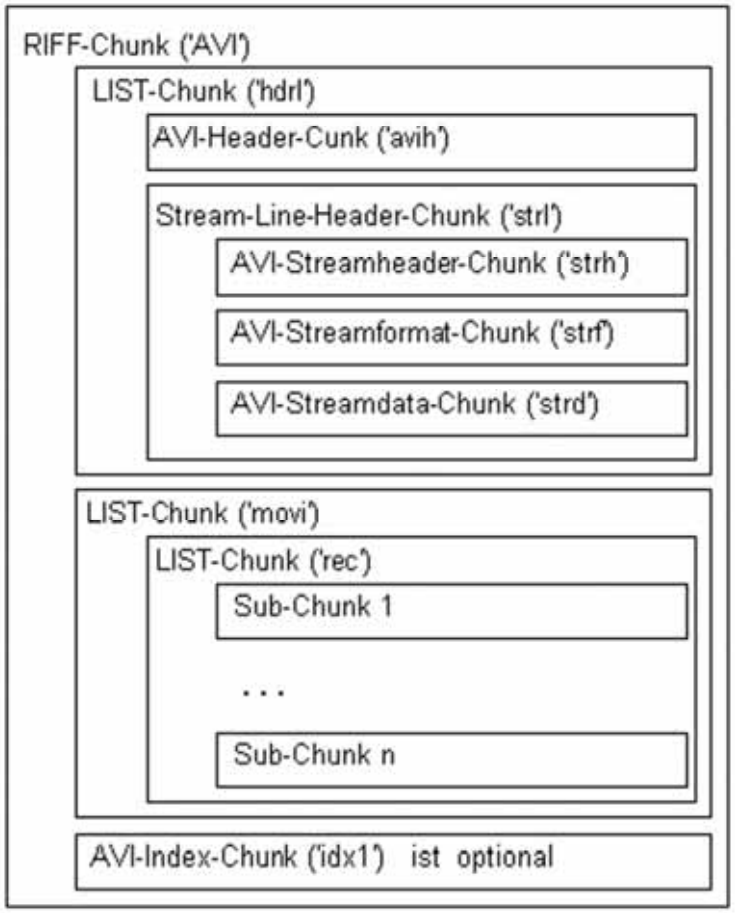


Abb. 3.29 AVI

Der AVI-Header-Chunk ('avih') enthält Informationen über Framerate, Framegröße, Zeitinformationen, Zahl der enthaltenen Streams usw. Im Stream Line Header-Chunk sind Informationen zu den einzelnen Streams in den Sub-Chunks des movi-Chunks der AVI-Datei enthalten. In den Subchunks kennzeichnet die Signatur 'XXwb' Wavedaten, 'XXdb' unkomprimierte und 'XXdc' komprimierte Bitmapdaten (XX identifiziert den Stream). Der optionale Index-Chunk am Ende enthält Informationen über die Chunk-Positionen und erlaubt wahlfreien Zugriff auf die einzelnen Chunks.

DVI

Das Digital Video Interleaved Format (nicht zu verwechseln mit Digital Video Interface, was eine Monitor-Schnittstelle bezeichnet, siehe Kapitel 2, und auch nicht zu verwechseln mit dem Ausgabeformat für Tech-Dateien!) ist Bestandteil

der DVI-Technologie von Intel, ursprünglich am David Sarnoff Research Center entwickelt. Dateien im DVI-Format können wie beim RIFF-Format Daten verschiedenen Typs enthalten, Einzelbilder wie Bewegtbilder oder Audiodaten. Die Dateiendung beschreibt den in der Datei abgelegten Typ der Daten. Dabei bedeuten die Endungen:

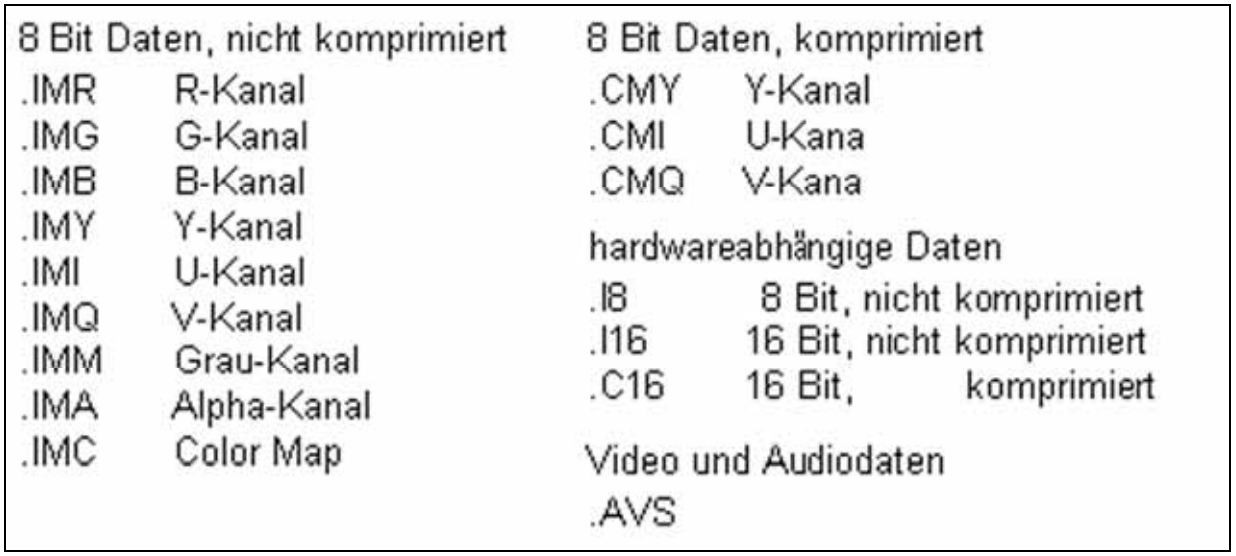


Abb. 3.30 DVI

AVS (Audio Video Stream)

Man erkennt aus Abb. 3.30, dass im DVI-Format für Bilder die Farbebenen getrennt in Dateien abgelegt sind. Einzelbilder werden mit 3 Farbkanälen und optional mit Alpha-Kanal und Farbtabelle gespeichert. Video- und Audiodaten werden zusammen in einem AVSS-Format(Audio Video Stream Standard) abgelegt mit der Dateiendung AVS oder dem Dateityp 'AVSS'. Eine AVSS-Datei enthält Daten in mindestens einem Bitstrom (Stream). AVSS Dateien besitzen die Strukturelemente

Abb. 3.31 AVSS

Im Standard-Header steht in der Signatur 'VDVI' für Videos und 'VIM' für Einzelbilder. Der sich daran anschließende AVL-Header beschreibt alle weiteren

Datenstrukturen, die in der Datei enthalten sind. Zu jedem Stream ist im entsprechenden Stream Headers hinterlegt, welche Art Daten im Stream abgelegt und wie diese zu interpretieren sind. Unterschieden werden komprimierte und nicht komprimierte Videodaten, komprimierte Audiodaten und mit Frames verknüpfte Daten. Streams selbst können zu Gruppen zusammengefasst sein. Jeder Stream enthält einen Substream Header.

Der Audio-Substream Header enthält alle notwendigen Angaben zur Interpretation und Wiedergabe der Audiodaten wie die Lautstärke einzelner Kanäle, Geschwindigkeit oder den zur Komprimierung benutzten Algorithmus usw. sowie die Anzahl Frames im Stream. Auch der Video-Substream Header enthält alle zur Interpretation und Wiedergabe notwendigen Parameter. Im Stream Header ist zudem für die Videodaten der Subtyp angegeben. Der Subtyp beschreibt, ob der Video Stream nur einen Kanal des YUV-Farbraums enthält oder ob alle Farbebenen entweder in YUV-Form oder YVU-Form (für komprimierte JPEG-Bilder) codiert sind. In den Substream Headers ist zudem jeweils der Name der Datei angegeben dem die Audio- bzw. Videodaten entstammen. Daran schließen sich die Frames mit jeweils eigenem Frame Header an. Die Position der einzelnen Strukturelemente ergibt sich entweder aus ihrer in den Elementen selbst abgelegten Länge oder es wird die Position im Header angegeben. So ist im Frame Header ein Zeiger auf den vorhergehenden Frame enthalten, die Position jedes Frames wird in einem Frame Directory gespeichert.

QuickTime

Das QuickTime Movie Resource Format (QTM) ist das Standardformat zur Speicherung von Video- und Audiodaten auf der Macintosh Plattform. QuickTime bezeichnet nicht nur das Dateiformat sondern auch die Multimedia Systemsoftware überhaupt. Apple entwickelte QuickTime auch für Windows-Umgebungen als plattformübergreifende Multimedia-Lösung. Das Dateiformat steht für den Macintosh mit der Dateiendung MooV und auf PCs mit der Dateiendung QTM zur Verfügung. Dateien im QTM-Format müssen jedoch erst in das MooV-Format mit getrennten Data fork und Resource fork konvertiert werden und umgekehrt. QTM besitzt wie AVI oder DVI Atome genannte Blöcke. Jedes Atom besitzt besteht aus den Teilen Länge, Typ, Struktur und Daten.
Die Atome können nach ihrer Struktur unterschieden werden in Container-Atome, die weitere Atome, auch Container-Atome, enthalten und Daten-Atome, die nur Daten enthalten. Atome können in bis zu fünf Ebenen geschachtelt sein. Auf eine detailliertere Darstellung wird verzichtet, jedoch die Struktur einer Datei in QTM-Format aufgezeigt:

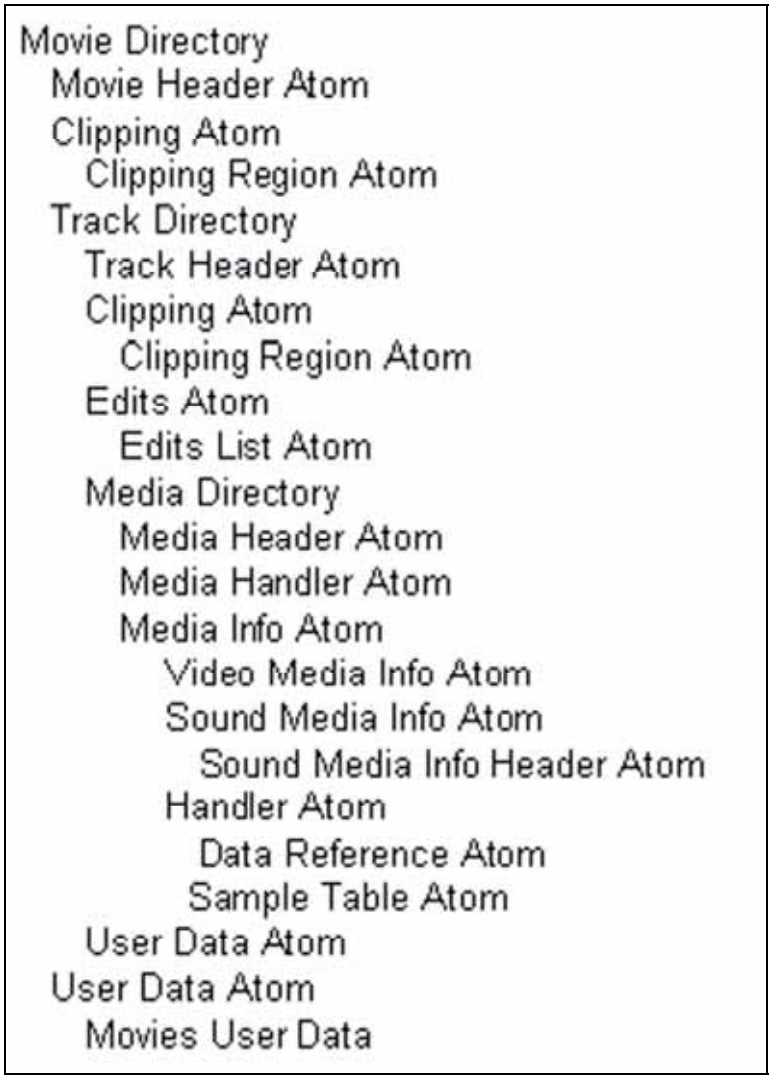


Abb. 3.32 QTM

MJPEG

Für MPEG wie für M-JPEG sind keine einheitlich standardisierten Formate definiert worden. Bei MPEG sind alle notwendigen Informationen zur Wiedergabe eines Videos direkt im Datenstrom codiert. Daher sind interne Strukturen wie Header oder Blockstrukturen (bisher) nicht nötig.

Dateien im M-JPEG Format (Motion JPEG) enthalten Videos als Folge einzelner mit JPEG codierter Bilder. Zur Wiedergabe werden die Bilder nacheinander decodiert und angezeigt. Für Audiodaten wird ein Standardverfahren zur Komprimierung verwendet. Die Vorteile von M-JPEG liegen in der schnellen, in Echtzeit möglichen Komprimierung und in der Tatsache, dass keine Interpolation oder Bewegungskompensation erfolgen muss. M-JPEG-Dateien sind jedoch deutlich größer als MPEG-Dateien und die Wiedergabe ist langsamer, da jedes Bild vollständig decodiert werden muss. Der größte Nachteil liegt jedoch in der

Qualität der Wiedergabe, wenn hohen Komprimierungsraten gefordert sind.

Es wurden hier nur einige Formate besprochen, und im Abschnitt über Video und Audio (Kapitel 5) werden hierzu weitere Formate (wie z.B. MIDI) skizziert. Man hätte zwar diese Formate auch hier mit beschrieben können, doch mir erschien es sinnvoll, dass diese Formate im Zusammenhang mit den benutzten Anwendungssoftwaren untersucht werden. Das hätte man zwar mit einigen Formten dieses Kapitels auch machen können, doch aufgrund der unmittelbaren Anwendbarkeit spezieller Formate (wie MIDI in Tonstudio-Applikationen) erscheint mir die vorliegende Aufteilung sonnvoll.

Das nächste Kapitel beschäftigt sich nun ein wenig mit den Grundlagen digitaler Bildbearbeitung.

Übungen zum Selbsttest

1. Kodieren Sie folgenden Datenstrom mit RLE (ggf. Recherchieren!), wobei Sie als Kontrollzeichen ein „X“ verwenden:

   3 5 5 5 5 7 8 8 8 7 7 7 4 4 4 4 4 4 X 9 9

2. In Abb. 3.7 wurde der Aufbau eines MPEG2-Datenstroms, wie er für z.B. Filme auf DVDs verwendet wird, dargestellt.
   Welche Vor- und Nachteile ergeben sich durch diese Art der Komprimierung in Hinblick auf
   a) wenig Aktionen im Film (z.B. Nachrichtensprecher)
   b) viele Aktionen im Film (z.B. dauernder Bildwechsel innerhalb einer GOP, schnelle Schwenks etc.)
   c) Rotationen und Zooms des Bildes

3. Wie könnte die Konvertierung einer Rastergrafik in eine Vektorgrafik (Vektorisierung) sinnvoll gemacht werden?

4. Digitale Bildverarbeitung

Digitalisierte Bilder kann man editieren, d.h. den Kontrast verändern, Farben anpassen, Weichzeichnen oder Verschärfen und vieles mehr. In diesem Abschnitt sollen einige Grundbegriffe sowie repräsentativ einige Verfahren hierzu betrachtet werden.
Grundsätzlich wird ausgenutzt, dass die digitale Bildinformation durch Zahlenwerte für jeden Pixel repräsentiert ist. Im einfachsten Fall kann dies z.B. so geschehen, dass jeder Bildpunkt den Wert eines Byte (also 8 Bit) besitzt, in dem z.B. die Grauwerte des betreffenden Pixels gespeichert sind. Dies ergibt in diesem Fall dann 256 mögliche Grauwerte, z.B. könnte das Byte mit dem Wert 0 der Farbe Schwarz und der Wert 255 einem weißen Bildpunkt entsprechen. Möchte man neben dem Grauwert noch einen Farbwert abspeichern, so benötigt man hierfür eben mehr als 8 Bit pro Pixel, z.B. 32 Bit (True Color). In diesem Fall wird der Grauwert des Schwarzweiß-Bildes praktisch zum Helligkeitswert der entsprechenden Farbe. Es hat sich eingebürgert, bei Farbbildern diesen Farbhelligkeitswert *Luminanz* und den eigentlichen Farbwert selbst als *Chrominanz* (oder einfach nur Chromanz) zu bezeichnen. Diese Trennung erweist sich insbesondere deswegen vorteilhaft, weil das menschliche Auge weniger Farb- als Helligkeitsempfindlich ist. Dadurch kann bei der Bildkompression für den Farbanteil ein „gröberes“ Kompressionsverfahren (also mit mehr Verlusten) benutzt werden als bei der Kompression des Helligkeitsanteils. Des weiteren kann ein Farbbild leicht in ein Grauwertbild umgewandelt werden: man lässt einfach die Farbinformation weg.
Der Einfachheit halber demonstrieren wir nachfolgend die beschriebenen Verfahren meistens an einfachen Grauwertbildern (8 Bit pro Pixel); die Prinzipien können leicht auf Farbbilder übertragen werden.
Man unterscheidet in der digitalen Bildverarbeitung zwischen *lokalen* und *globalen* Operationen. Lokale Operationen beziehen sich auf die jeweiligen Pixel, während globale sich auf das ganze Bild (oder eine Teilmenge davon) beziehen. Solche Anpassungen werden entweder durch mathematische Funktionen oder durch Matrizen, die auf das Urbild angewendet werden, erreicht. Hierfür nachfolgend nun einige gängige Methoden, die natürlich im Rahmen dieses Buches nur eine kleine Auswahl darstellen können.

Kontrastanhebung durch Grauwertspreizung
Durch einen einfachen mathematische Ausdruck (lineare Funktion) lässt sich der Kontrast eines Ausgangsbildes anheben: Man weist dem Wert des „dunkelsten“ Bildpunktes den Wert „schwarz“ (also 0) des Zielbildes und dem „hellsten“ Bildpunkt des Urbildes den Wert „weiß“ (255) des Zielbildes zu. Durch einfa-

chen Dreisatz kann damit jeder „Zwischenwert“ sofort errechnet werden (vgl. Abb. 4.1):

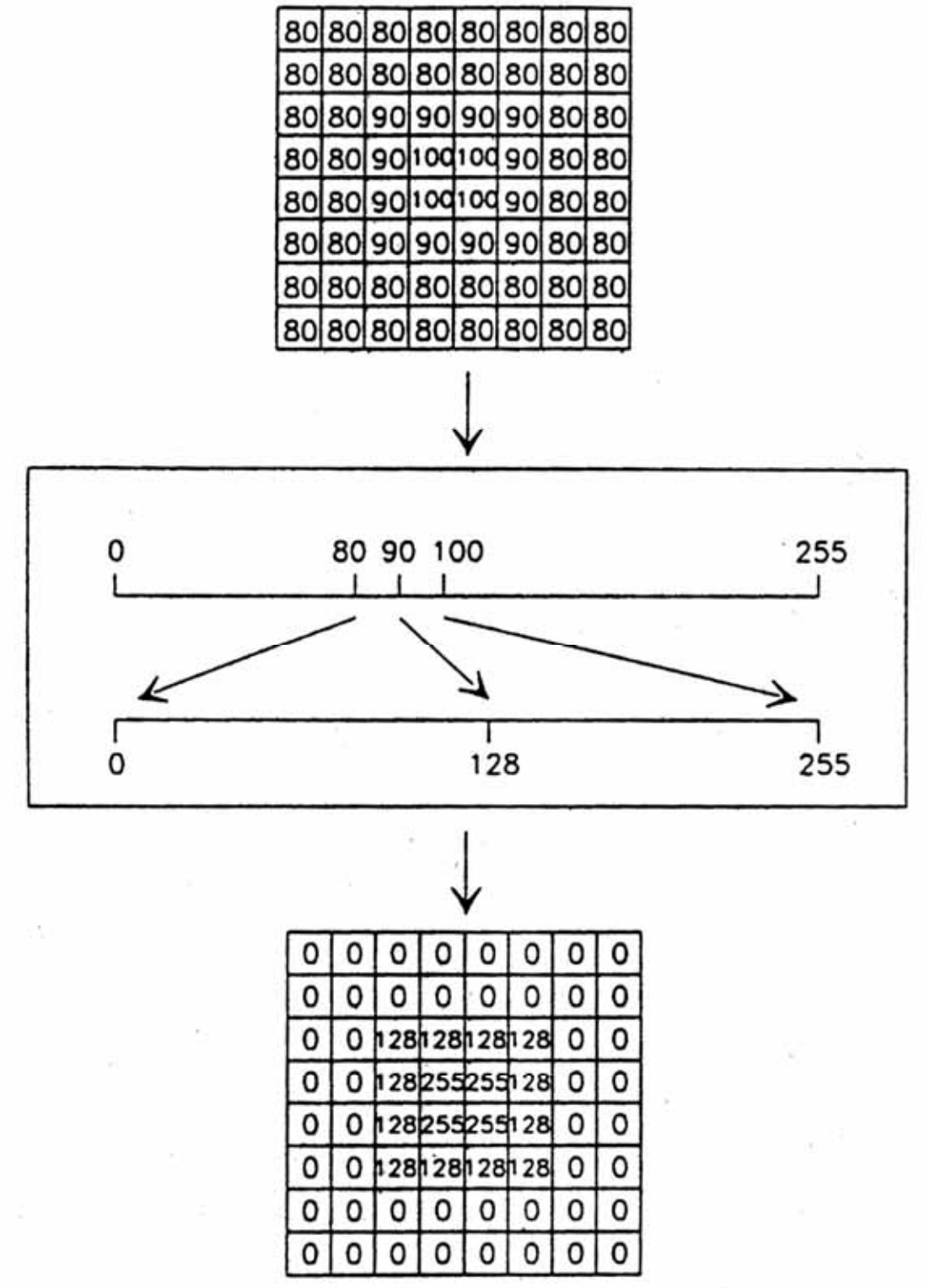


Abb. 4.1 Kontrastanhebung

In Abb. 4.1 sehen wir ein Bild mit 8x8 Pixel, wobei der dunkelste Grauwert des Ausgangsbildes 80 und der hellste 100 beträgt. Durch die Grauwertspreizung entsteht daraus das Zielbild, wo der dunkelste Bildpunkt jetzt den Wert 0 (also schwarz) und der hellste den Wert 255 (also weiß) angenommen hat.
Umgekehrt lässt sich der Kontrast eines Bildes absenken, in dem man eine Grauwertstauchung vornimmt. Hierzu kommt einfach die mathematische Umkehrfunktion zum Tragen: Der hellste Bildpunkt des Ausgangsbildes wird auf einen dunkleren und der dunkelste Punkt auf einen helleren abgebildet (und alle Zwischenwerte wieder per Dreisatz linear angepasst). Neben linearer Grauwertspreizung können auch nichtlineare Funktionen benutzt werden. Dies wird z.B. bei der sog. *Tonwertkorrektur* ausgenutzt. Dabei handelt es sich um die Anhebung oder Absenkung bestimmter festgelegter Grauwertbereiche (also nicht notwendigerweise das gesamte Bild), um Unregelmäßigkeiten z.B. durch bestimmte Druckverfahren auszugleichen.

Falschfarbenbilder
Es lassen sich auch Falschfarben mit der Methode der linearen Grauwertspreizung erzeugen. Hierzu werden bestimmten Graubereichen des Ausgangsbilds feste Farbwertbereiche zugeordnet (vgl. Abb. 4.2):

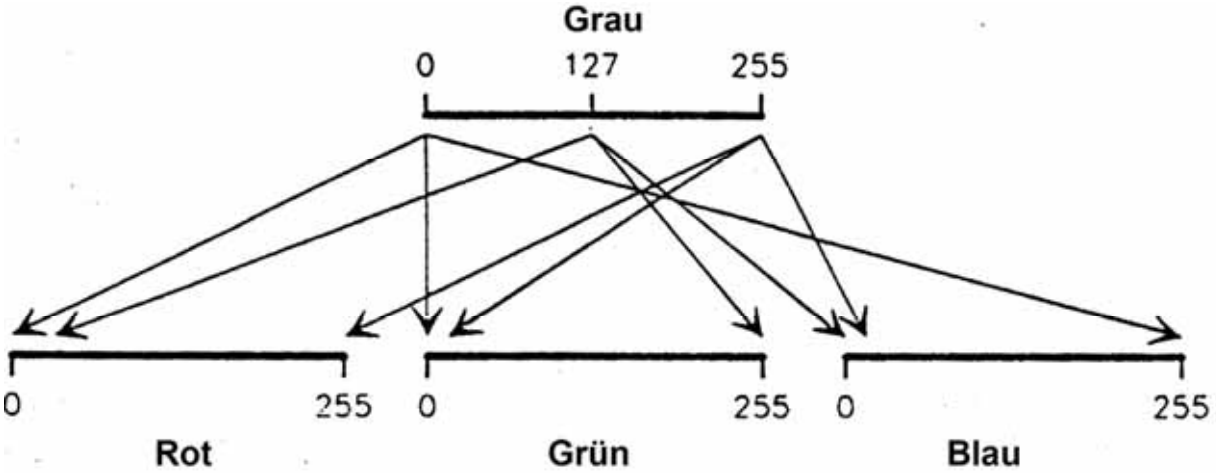


Abb. 4.2 Beispiel für Falschfarbenzuordnungen

Falschfarbenbilden werden z.B. für Ultraschallaufnahmen im medizinischen Bereich eingesetzt, da man hier Gewebeveränderungen besser als in der Schwarzweißaufnahme erkennen kann. Falschfarben eignen sich übrigens nicht zum „Umwandeln“ von Schwarzweisfotos in realistische Farbbilder, da gleiche Grauwerte auf dem Bild verschiedene Farben in der Realität darstellen können (ein Grauwert eines Schwarzweisbildes entspricht ja nur der Helligkeit einer Farbe und gibt keine Auskunft über die Farbe selbst). Wenn man solches durch-

führen möchte, sind wesentlich komplexere Verfahren erforderlich (vgl. Kaptitel 8).

Objektkantenermittlung durch den LaPlace-Operator

Bei bestimmten Verfahren werden auch lokale Matrix-Operatoren benutzt. Diese arbeiten so, dass z.B. eine 3x3-Operatormatrix mit festen Zahlenwerten über die Pixel des Ausgangsbilds gleitet (vgl. Abb. 4.3). Ein neu zu berechnender Bildpunkt ergibt sich dabei z.B. aus der Matrizenmultiplikation desjenigen „alten" Bildpunktes, der im mittleren Matrixelement der Operatormatrix steht. So gleitet die 3x3-Operator-Matrix Zeile für Zeile über die Pixel des Ausgangsbilds und berechnet z.B. durch spezielle Matrix-Multiplikation (das wäre dann ein linearer Operator) immer den sich in der Mitte der Operatorenmatrix befindliche Bildpunkt des Ausgangsbildes neu. Die Randelemente des Ausgangsbildes werden dabei künstlich um je einer Zeile und Spalte, die nur aus Nullen besteht, erweitert (nur für die Berechnung). Hinterher wird i.d.R. noch durch einen Normierungsfaktor dividiert um wieder in den ursprünglichen Wertebereich zu gelangen

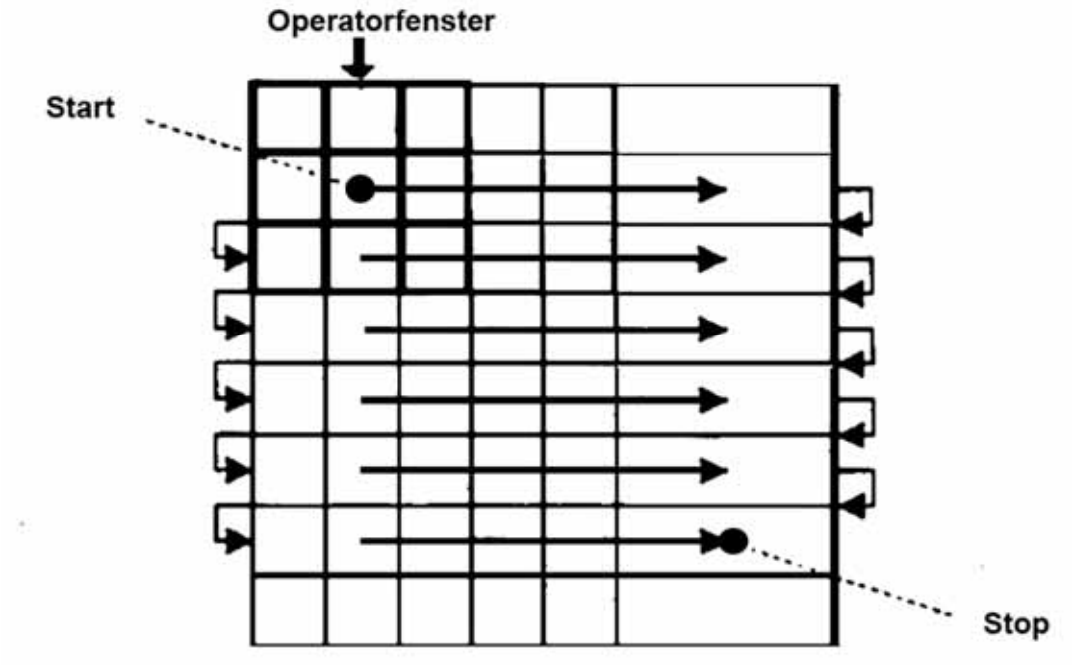


Abb. 4.3 3x3-Operatormatrix gleitet über Ausgangsbild

Ein Beispiel dafür ist der LaPlace-Operator. Er lautet:

0	1	0
1	-4	1
0	1	0

Mathematisch entspricht dieser LaPlace-Operator einer zweifachen Anwendung des Gradienten (als Faltung), bestimmt also in gewissem Sinn die zweite Ableitung und so die Objektkanten (vgl. Abb. 4.4).

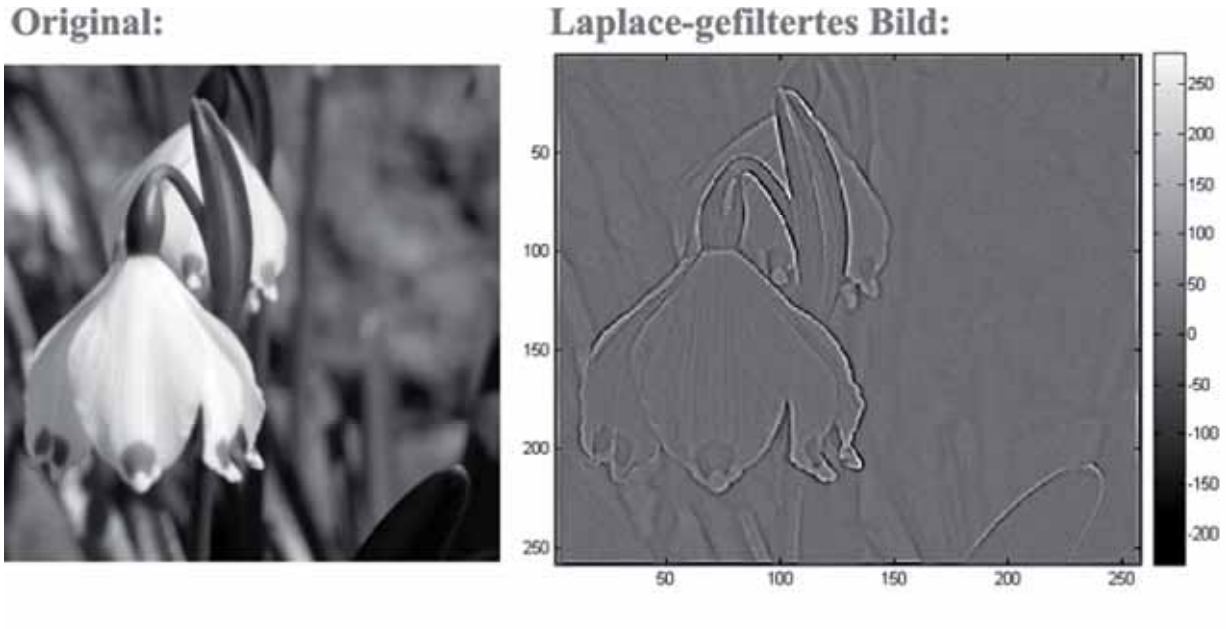


Abb. 4.4 Anwendung des LaPlace-Operators[11]

Hier noch ein mathematisches Beispiel dazu:

1	2	3	1	1	...
1	1	2	2	1	
2	3	**10**	2	2	
1	1	3	3	1	
1	1	2	1	3	
...					

•

0	1	0
1	-4	1
0	1	0

→

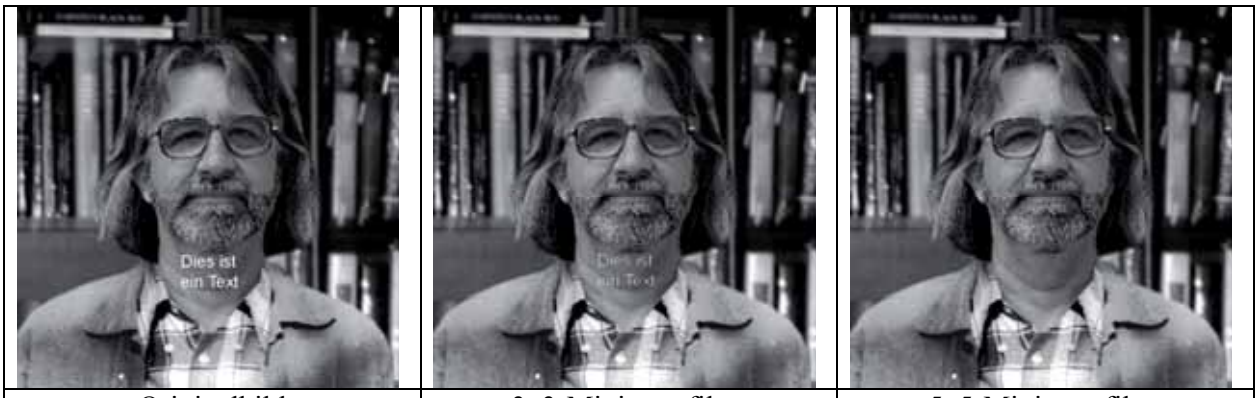
0	0	0	0	...
0	-4	-8	2	
0	-2	**-30**	-9	
0	-4	-4	-5	
...				

Beispielsweise errechnet sich die Zahl -30 im Zielbild aus der Zahl 10 des Ursprungsbildes wie folgt:
$0 \cdot 1 + 1 \cdot 2 + 0 \cdot 2 + 3 \cdot 1 - 10 \cdot 4 + 2 \cdot 1 + 1 \cdot 0 + 3 \cdot 1 + 0 \cdot 3 = 2 + 3 - 40 + 2 + 3 = -30$
Die Ergebniswerte sind später noch entsprechend in das Zielintervall (z.B. [0,255]) zu transformieren.

Glättung und Störungsentfernung durch Rangordnungsfilter
Rangordnungsfilter zählen zu den nichtlinearen Filtern. Das Prinzip ist folgendes: Im Falle einer 3x3-Operatormatrix werden alle Pixel des aktuellen Bereichs, über dem die Matrix gerade liegt, sortiert. Danach wird beim *Minimum-Filter*

[11] Quelle: http://de.wikipedia.org/wiki/Bild:Laplace_beispiel.png

der im Zentrum liegende Original-Pixel durch das Minimum der sortierten Pixel ersetzt, beim *Maximum-Filter* durch das Maximum und beim *Medianfilter* durch den Mittelwert. Es können neben 3x3-Operatoren auch 5x5-, 7x7-etc. angewendet werden, was die Ergebnisse entsprechend verbessern kann.



Originalbild	3x3-Minimumfilter	5x5-Minimumfilter

Abb. 4.5 Anwendung des Minimumfilters

In Abb. 4.5 sehen wir eine Anwendung des Minimumfilters. Da die weißen Pixel der eingeblendeten Schrift die größten Werte besitzen, werden diese beim Minimumfilter ersetzt durch Werte der „dunkleren" Umgebung. Gleichzeitig leidet die Bildschärfe etwas darunter. Wäre die Schrift schwarz gewesen, hätte ein Maximumfilter angewendet werden müssen, um diese Schrift zu beseitigen.

A Abb. 4.6 Anwendung des Medianfilters[12]

[12] Quelle: http://upload.wikimedia.org/wikipedia/commons/1/1d/Medianfilterp.png

Abb. 4.6 zeigt eindrucksvoll, wie mit Hilfe des Medianfilters eine mit erheblichen Störungen behaftete Fotografie „entstört“ werden kann.
Professionelle Bildbearbeitungsprogramme wie z.B. Adobe Photoshop® ermöglichen das Erstellen eigener linearer Filter-Matrizen (vgl. Abb. 4.7).

A Abb. 4.7 Eigene Filter-Matrizen

Morphologische Operationen

Manchmal steht man vor der Aufgabe, „ausgefranselte“ Bildelemente zu glätten. Dies kann auf zwei Arten geschehen: Mann kann die Ausfranselungen entfernen, so dass nur „gerade“ Konturen bleiben, oder man kann die Ausfranselungen auffüllen. Zu diesem Zweck eignen sich sog. morphologische Operatoren, von denen ich zwei Repräsentanten vorstellen möchte.
Morphologische Operationen verwenden Methoden der Nachbarschaftsfindung, um das Umfeld eines Bildpunktes in Betracht zu ziehen und daraus folgend Operationen durchzuführen. So wird z.B. versucht, eine (relativ zum Bild kleine) Bitmaske (das sog. *Strukturelement*) so auf ein Bild zu legen, dass alle darunter liegenden Bildpunkte des Ausgangsbildes denselben Wert wie diejenigen haben, die in der Maske definiert sind. Man löscht alle Punkte, die nicht Zentrum der irgendwo passend aufgelegten Maske sind.
Die Grundoperation *Erosion* wird dabei mit Hilfe einer solchen Strukturmaske realisiert. Sie besteht aus einer kleinen Teilmenge des Gesamtbildes, die zum Überstreichen des zu untersuchenden Bildes benutzt wird. Für jede Maske wird ein Bezugspunkt definiert, welcher das Platzieren der Maske an einer bestimm-

ten Pixelposition erlaubt. Die eigentliche Operation besteht aus der pixelweisen Verschiebung der Strukturmaske über das Gesamtbild. Es wird geprüft, ob das strukturierende Element vollständig in die Menge passt. Wenn ja, so gehört das Pixel des Bildes, an der Stelle an der sich der Bezugspunkt der Strukturmaske befindet, zur erodierten Menge.
Eine weitere morphologische Methode ist die *Dilation* (manchmal auch *Dilatation* genannt), bei der Voraussetzung für eine Aktion ist, dass mindestens ein Pixel des Strukturelements ein Pixel des Urbilds berührt. In diesem Fall werden im Originalbild die Bildpunkte um das Zentrum um die in der Maske definierten Bildpunkte erweitert.

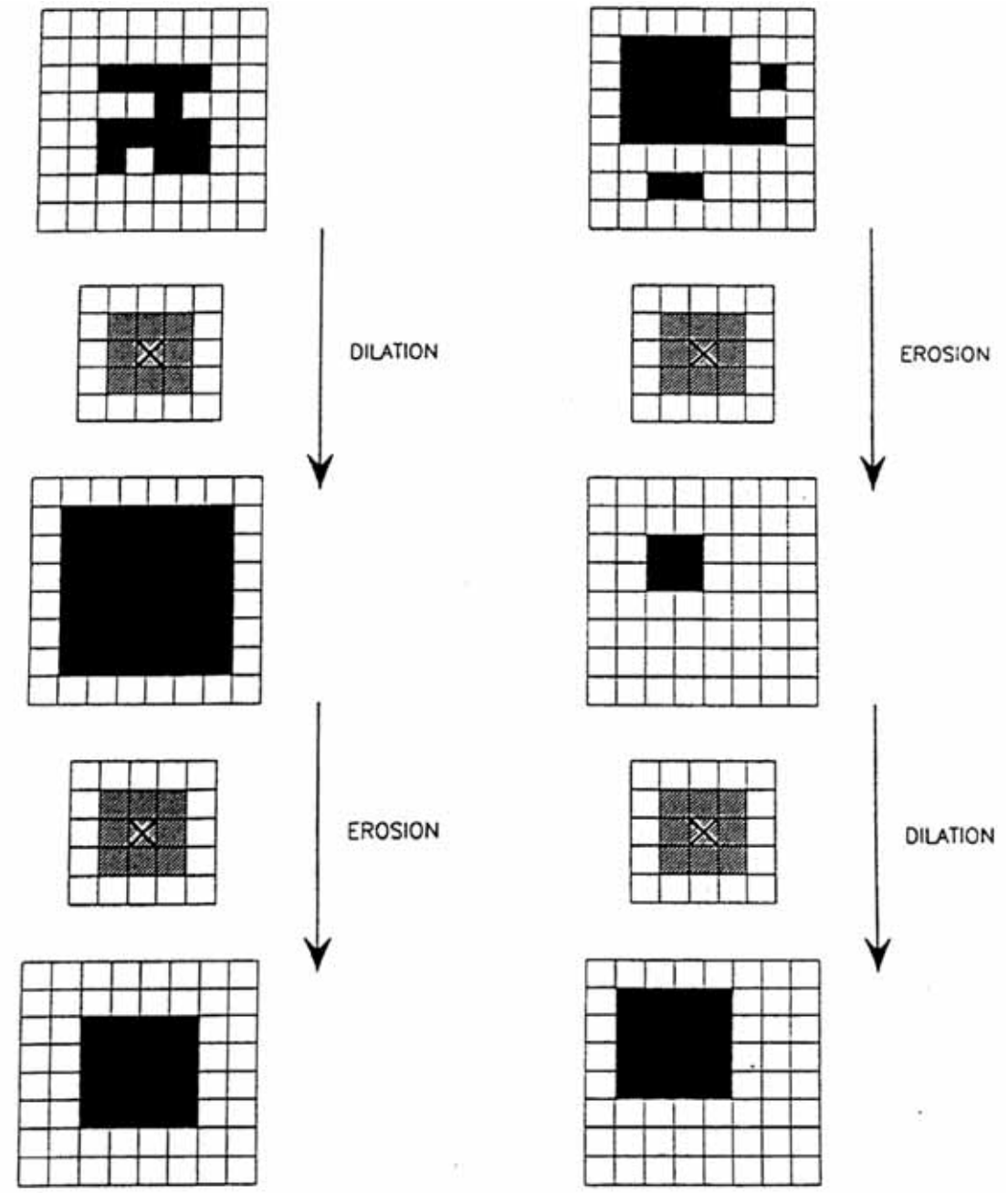


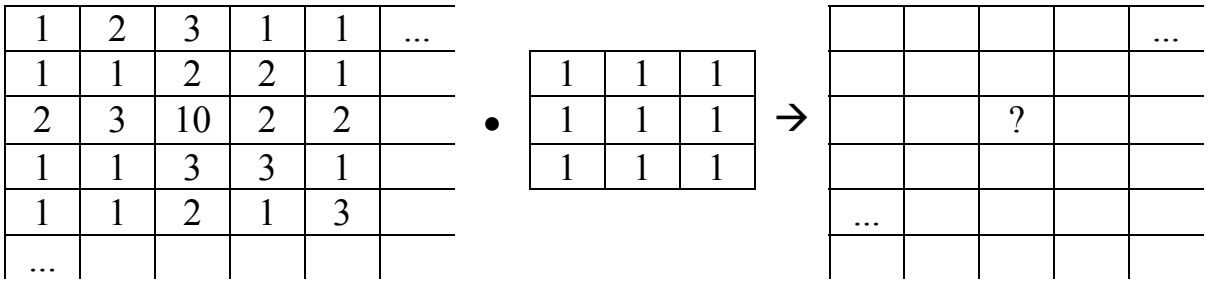
Abb. 4.8 Closing und Opening mittels morphologischer Operationen

Abb. 4.8 zeigt, dass durch Hintereinanderausführung der Erosion- und Dilationsoperationen ein sog. „closing“ (links im Bild) oder „opening“ (rechts im Bild), je nach Reihenfolge der Anwendung, geleistet werden kann.

Diese Beispiele sollen uns hier genügen. Im nächsten Kapitel wollen wir uns speziell den Audio- und Video-Anwendungen zuwenden, vor allem in praktischer Hinsicht.

Übungen zum Selbsttest

1. Welche Wirkung hat folgender linearer Operator:



2. Schreiben Sie einen (Pseudo-Code)-Algorithmus zur Binarisierung eines Graubildes mit einem frei wählbaren Schwellwert „s“.

3. Es sein ein Grauwertbild gegeben, dessen hellster Bildpunkt den Helligkeitswert g_{max} hat und dessen dunkelster Punkt den Wert g_{min} hat. Wie lautet die Formel für einen beliebigen Bildpunkt G einer „Auto-Kontrastwertkorrektur“, die eine Grauwertspreizung auf das Intervall [0,255] leistet?

5. Audio- und Video-Verarbeitung

In diesem Kapitel wollen wir noch einige theoretische Anmerkungen zu Formaten und Video-Normen machen, welche sich vorwiegend auf die Bearbeitung von Audio –und Video-Dateien, Herstellung von Ton- und Bildträgern (CD, Video-DVDs etc.) beziehen. Daneben werden einige Standard-Verfahren zur Audio- und Videoverarbeitung behandelt.

Spezielle Audio-Formate
Wir beginnen mit dem Audio-Bereich. Hierzu haben wir schon die gängigen Formate WAV und mp3 kennen gelernt (vgl. Kapitel 3). Allerdings gibt es gerade im Bereich der digitalen Videoverarbeitung noch das Audio-Format AC-3, das wir schon in Kapitel 3 kurz angesprochen haben und was hier jetzt vertieft werden soll.

Dolby Digital und AC-3
Dolby Digital[13] (auch ATSC A/52 und AC-3) ist ein Mehrkanal-Tonsystem der Firma Dolby, das in der Filmtechnik, auf Laserdiscs, DVDs und in der Fernsehtechnik zum Einsatz kommt. Dolby Digital unterstützt bis zu sechs diskrete Kanäle und verwendet ein psychoakustisches, verlustbehaftetes Verfahren zur Datenkompression.
Das Format wurde vom Advanced Television Systems Committee mit der Dokumentnummer A/52 international standardisiert und trägt offiziell den Namen ATSC A/52. Dolby Digital ist der Marketingname (oft abgekürzt als DD). AC-3 schließlich bezeichnet das Bitstream-Format (Adaptive Transform Coder 3) und hat sich ebenfalls als Bezeichnung eingebürgert. Daher kommt auch die typische Dateiendung *.ac3. Auch leicht abgewandelte Bezeichnungen wie Dolby Stereo Digital oder Dolby SR-Digital und einige andere werden verwendet.
1995 wurde Dolby Digital als Audioformat für die DVD festgelegt. Außerdem erschien die erste Laserdisc mit diesem Tonformat. Dolby Digital umfasst bis zu sechs Kanäle. Im Einzelnen sind das:

- Vorne links und rechts,
- Vorne Mitte,
- Hinten links und rechts (Surroundkanäle),
- LFE (Low Frequency Effects; Subwoofer).

Die häufigsten Kanalkonfigurationen sind:

- Dolby Digital 5.1 mit allen sechs Kanälen. Häufigstes Format für die Haupttonspuren einer DVD.

[13] vgl. http://de.wikipedia.org/wiki/Dolby_Digital

- Dolby Digital 2.0 mit zwei Kanälen (Stereo), teilweise sind auch Dolby-Pro-Logic-Informationen enthalten. Häufigstes Format für zusätzliche Tonspuren der DVD, wie Audiokommentare.
- Dolby Digital 2.1 mit drei Kanälen, davon zwei (Stereo), und dem *LFE (Low Frequency Effects; Subwoofer) -Kanal.
- Dolby Digital 1.0 mit einem Kanal (Mono). Manchmal anzutreffen als Audiospur für die DVD-Ausgabe sehr alter Filme.

Die volle Kanalausstattung wird als 5.1-Ton bezeichnet, da nur die ersten fünf Kanäle das komplette mögliche Frequenzspektrum von 20 Hz bis 20 kHz wiedergeben. Der LFE-Kanal ist auf Tieftoneffekte zwischen 20 und 120 Hz beschränkt.
Dolby Digital arbeitet mit diskreten Kanälen, d.h. alle Kanäle sind vollständig und prinzipiell unabhängig. Im Gegensatz dazu steht z.B. Dolby Pro Logic, das in einem Stereosignal per Matrixcodierung vier Kanäle verschlüsselt.
Aufgrund der großen Kanalanzahl, der hohen Abtastrate von 48 kHz und der möglichen Auflösung zwischen 16 und 24 Bit fällt in unkomprimierter Form eine enorme Menge an Audiodaten an. Für die Tonspur eines zweistündigen Films liegt der Platzbedarf in der Regel bei 4 GB, was etwa der halben Speicherkapazität einer Double-Layer-DVD entspricht. Deshalb arbeitet Dolby Digital mit einer verlustbehafteten Datenkompression. Das verwendete Kodierungsverfahren basiert – wie auch bei z.B. MP3, Vorbis und AAC – auf der Tatsache, dass das menschliche Ohr bestimmte Toninformationen nicht wahrnimmt. AC-3 unterstützt Bitraten zwischen 32 und 640 kbit/s (Kilobit pro Sekunde). Auf einer DVD werden für 5.1-Ton gewöhnlich 384 oder 448 kbit/s verwendet, für Stereoton 192 oder 224 kbit/s. Im Kino liegt die Datenrate bei 320 kbit/s.

MIDI

Musical Instrument Dgital Interface (MIDI) ist ein Industrie-Standard zur Darstellung von Sound in binärem Format, ursprünglich als Standard für Synthesizer-Schnittstellen definiert. MIDI stellt kein Audio-Format für digital aufgenommenen (gesampelten) Sound im eigentlichen Sinn dar, sondern enthält eine *Beschreibung* der Audiodaten und nicht die Audiodaten selbst. Genau genommen werden die Aktionen des Musikers z.B. an einer Klaviertastatur („MIDI-Keyboard“) aufgezeichnet. Also z.B. die Note der gedrückten Taste, die Anschlagsstärke, wie lange die Taste gedrückt wird etc.; diese Aktionen werden i.d.R. jeweils in 8-Bit-Worten codiert. Deshalb sind MIDI-Dateien relativ klein, denn der Sound selbst wird nicht abgespeichert, sondern erst in einem entsprechenden MIDI-fähigen Empfangsgerät erzeugt. Dadurch kann es passieren, dass die Güte der Wiedergabe z.B. eines Pianos von dem abspielenden Gerät abhängig ist, da dieses erst den Klang aus den Aktionen der MIDI-Datei erzeugt. Dabei werden die einzelnen Datenworte seriell übertragen, sodass in jedem Byte

noch eine Identifikation, um welche Art Daten es sich gerade handelt, mitgeliefert werden muss. Abb. 5.1 zeigt beispielsweise den allg. Aufbau solcher MIDI-Botschaften. Dabei werden 16 sog. *MIDI-Kanäle* benutzt, d.h. es kann im Prinzip an 16 Geräte gleichzeitig Information geschickte werden (so das z.B. max. 16 verschiedene Instrumente gleichzeitig angesprochen werden können, jedes mit eigener „Melodie“).

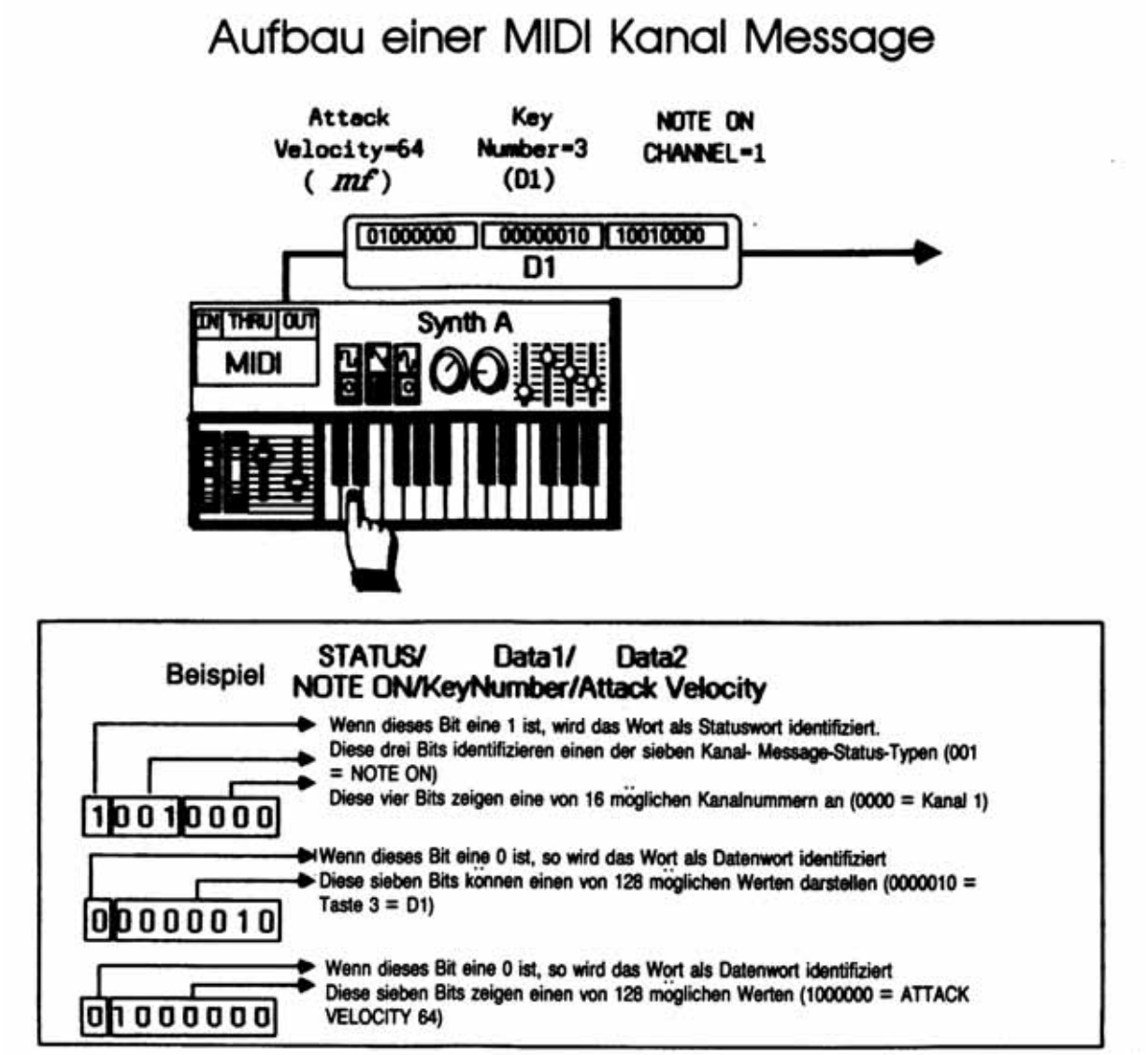


Abb. 5.1 MIDI-Kanal-Messages

Drückt der Musiker also auf dem Keyboard eine Taste, wird zuerst das Statuswort „NoteOn“ geschickt (welches u.a. die zuvor selbst einstellbare MIDI-Kanalnummer zwischen 1 und 16 mitliefert), gefolgt von einer Tastennummer (die der gedrückten Note, was hier dem „D“ entspricht) und danach ein Byte mit dem Wert , der angibt, wie „fest“ die Taste herunter gedrückt wurde („Anschlagsstärke“).

In Abb. 5.2 sehen wir weitere Möglichkeiten für MIDI-Messages:

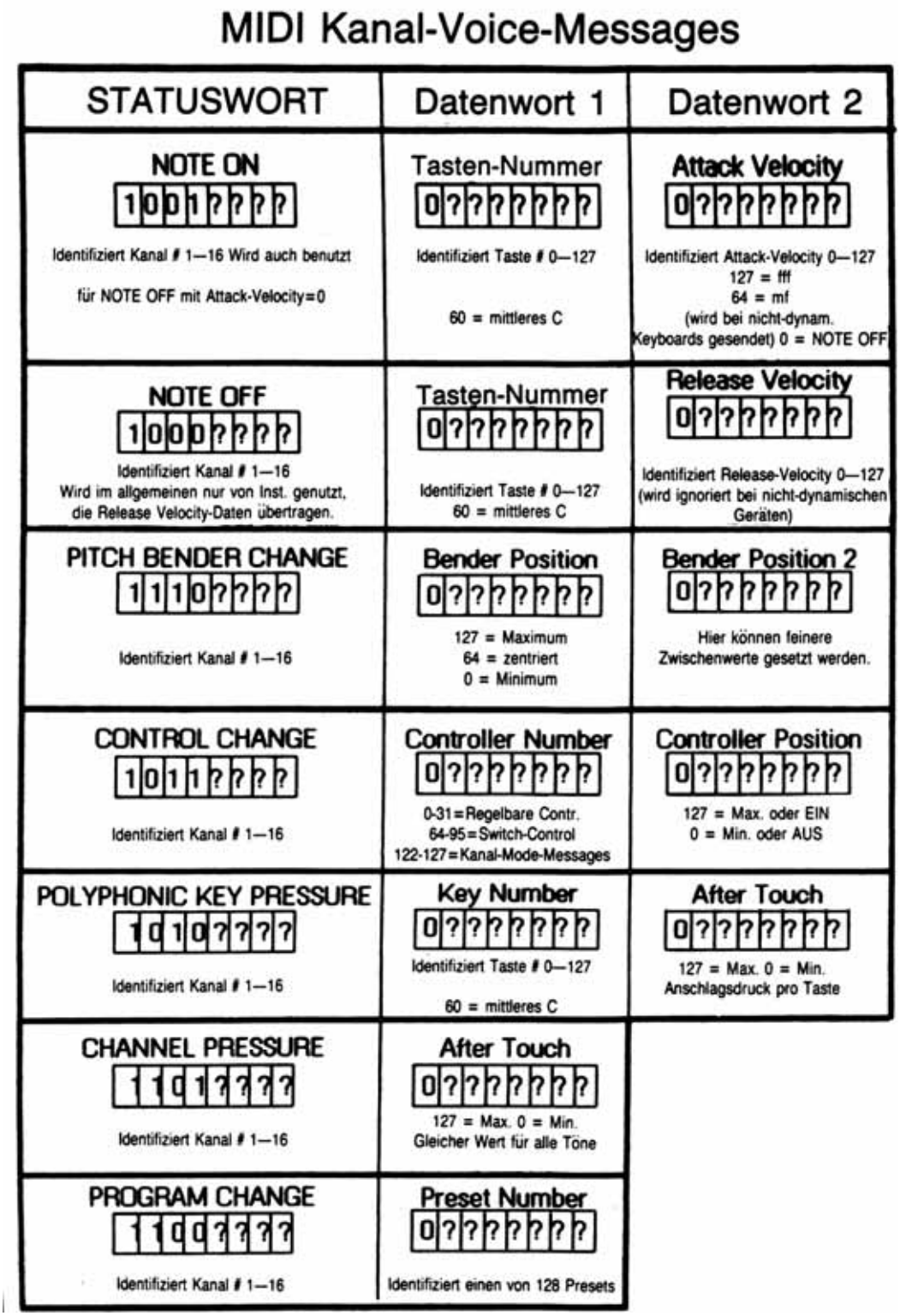


Abb. 5.2 MIDI-Kanal-Voice-Messages

Zum Austausch der mit unterschiedlichen Geräten aufgezeichneten Daten wurde das systemunabhängige Standard MIDI-File Format (SMF) definiert. MIDI-

Dateien enthalten also eine Reihe von Control-Messages, die ein Soundereignis mit Tonhöhe, Zeitdauer und Lautstärke beschreiben. An MIDI-kompatible Geräte gesandte Control-Messages werden dort interpretiert und der Sound dort reproduziert. MIDI beschreibt dazu die Hardware-Schnittstellen beteiligter Geräte und Protokolle zum Austausch von MIDI-Daten. Die MIDI-Daten können wie andere binäre Daten komprimiert werden und benötigen keine der für Audiodaten speziell entwickelten Verfahren der Codierung und Komprimierung.
Das Zeitformat kann verwendet werden für die Geschwindigkeit der Wiedergabe oder zur Synchronisation von Sound und Videodaten. Messages (oder Befehle) werden auch als Events bezeichnet, die unterschieden werden nach MIDI-Events, System-Events und Meta-Events:

- **MIDI-Events:** Gruppe von Befehlen zur Kommunikation zwischen Geräten, Befehle wie das Ein- und Abschalten von Tönen, Festlegung der Anschlagstärke von Tasten, aber auch Befehle zur Programmierung von Synthesizer-Kanälen oder der Zuordnung von Instrumenten zu Kanälen.
- **System-Exclusive-Events:** Mit dem System-Exclusive-Befehlen werden über den Standard hinausgehende Funktionen möglich
- **Meta-Events:** Gruppe von Events mit Befehlen, die nicht unmittelbar der Aufzeichnung oder Wiedergabe dienen, sondern erlauben, Tracks zu nummerieren, Instrumenten Namen zuzuordnen, Songtexte abzulegen, Taktart und Geschwindigkeit zu modifizieren usw.

Audio-Bearbeitung in der Praxis
Im Bereich der Audioverarbeitung ist also zu unterscheiden zwischen Wave- und MIDI-Dateien. Der Vorteil der MIDI-Files liegt auf der Hand: Von einem Musiker eingespielte MIDI-Aktionen können leicht mit einem entsprechenden Editor bearbeitet werden. So können nachträglich Noten hin- und hergeschoben werden, die Tonhöhe einzelner Töne durch Drag- und Drop verändert werden, also Spielfehler notenweise korrigiert werden, Lautstärken und Längen einzelner Noten nachträglich verändert werden etc.; theoretisch kann man mit Hilfe eines MIDI-Editors Musik machen, ohne dass man auch nur ein Instrument spielen kann, z.B. in dem man einfach nacheinander die einzelnen Noten per Mausklick eingibt.
Im professionellen Tonstudio-Bereich werden beide Verfahren, also MIDI- und WAVE-Dateien, kombiniert eingesetzt: Wo immer möglich, werden MIDI-Aktionen benutzt (wegen der guten Editierbarkeit und der kleinen Datenmengen, was auch die CPU-Belastung extrem reduziert). Wenn MIDI nicht eingesetzt werden kann, so werden Audio-Aufnahmen gemacht (wie z.B. beim Gesang). Audio- und MIDI-Events werden dann gleichzeitig oder auch hintereinander aufgenommen und können gleichzeitig wiedergegeben werden. Man zerlegt so ein Musikstück dabei in einzelne Tonspuren, z.B. für jedes MIDI-Instrument

eine Spur und für jedes Audioereignis eine. Erst wenn das Lied fertig ist, wird eine „Endabmischung“ gemacht, wo alles zusammen auf 2 Stereo-Audio-Spuren „heruntergemischt“ wird. Professionelle Audio-Softwaren bieten dafür auch virtuelle Mischpulte mit den in Tonstudios üblichen Effekten (wie Hall, Echo etc. für jede Spur einzeln oder gemeinsam) an.

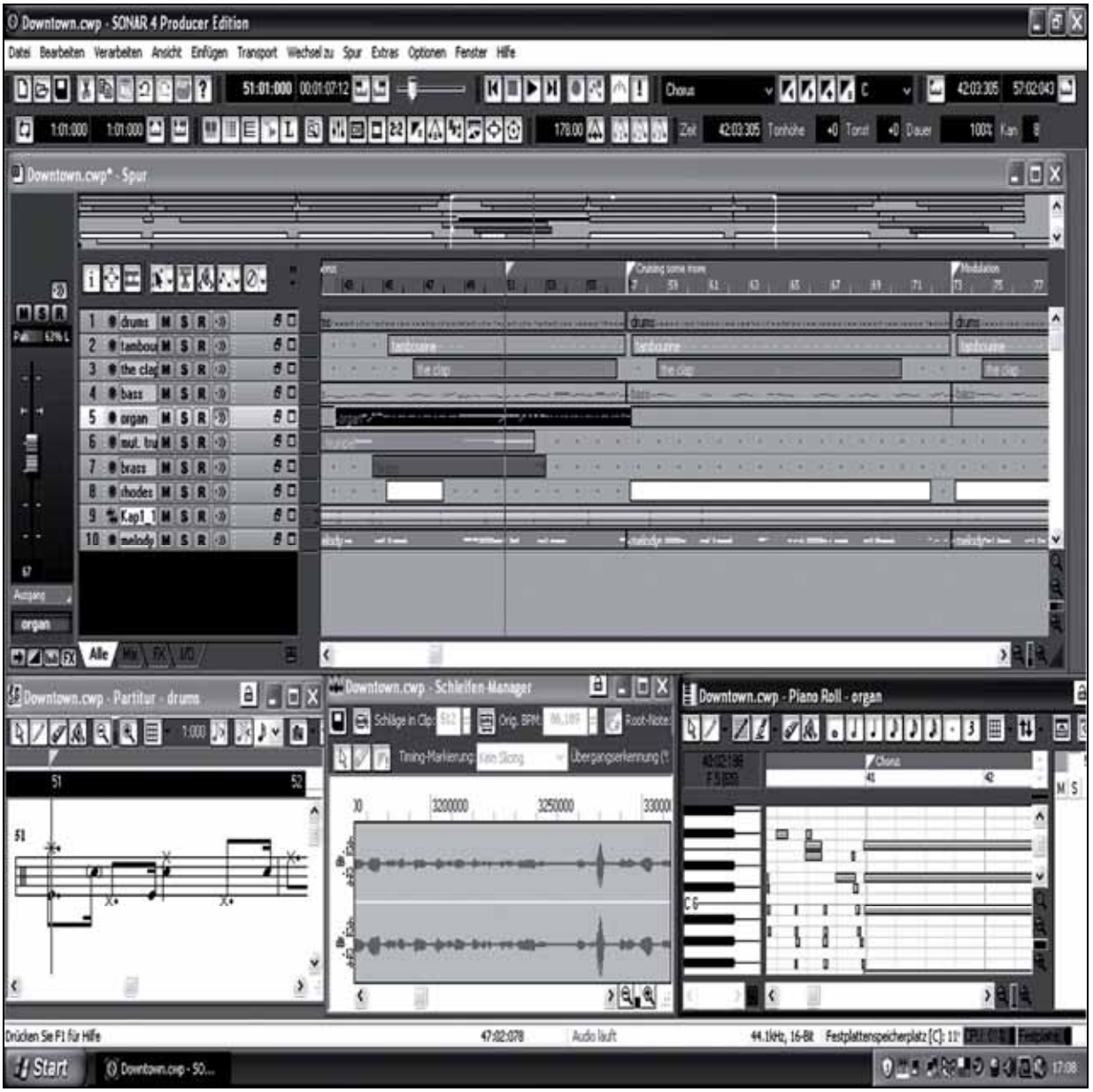


Abb. 5.3 Audio- Be- und Verarbeitung mit Sonar®

In Abb. 5.3 sehen wir diese typischen Merkmale, wie sie z.B. im Audio-Bearbeitungsprogramm Sonar® der Firma Cakewalk vorhanden sind. Bei den Spuren 1-8 und 10 handelt es sich um reine MIDI-Spuren, während Spur 9 eine WAVE-Audio-Spur darstellt. Im unteren Bereich sieht man beispielhaft einzelne dieser Spuren geöffnet. Dabei kann man verschiedene Ansichten wählen, bei MIDI-Files z.B. die Notenansicht (links unten) oder eine Klaviatur-Ansicht

(rechts unten). Im Fenster unten Mitte sieht man die Darstellung einer WAVE-Audio-Spur. In allen Fenstern kann man die Daten editieren, also Teile herausschneiden, verschieben, Lautstärken oder Tonhöhen ändern und so weiter. Dies kann auch in Echtzeit geschehen, also während das Stück abgespielt wird. Dadurch kann eine Abmischung optimal durchgeführt werden.
Werden MIDI-Spuren benutzt, so müssen natürlich MIDI-fähige Geräte eingesetzt werden. Dazu können eine oder mehrere Soundkarten benutzt werden, die MIDI-Instrumente „On Board“ haben. Oder es können externe Synthesizer angeschlossen werden, die über MIDI-Anschlüsse (i.d.R. DIN-Buchsen) verfügen. In neuere Zeit haben sich auch sog. „Software-Synthesizer“ etabliert. Dabei handelt es sich um Software, die MIDI-Hardware emuliert. Die Töne, die dabei durch MIDI angesprochen werden, sind in Wirklichkeit kleine WAVE-Dateien, welche je einen gesampelten Ton beinhalten, der bei Drücken der entsprechenden MIDI-Taste auf einem MIDI-Keyboard von der Software wiedergegeben wird. Diese Samples kann man mit geeigneter Sample-Software („Sampler“) auch selbst erstellen und so z.B. einen Aaahh-Chor selbst einsingen, in dem man für jeden Ton auf der MIDI-Tastatur die passende Note singt. Diese wird abgespeichert und später über die MIDI-Tastatur aufgerufen. Dadurch können auf dem Computer Kompositionen entstehen, deren Musikinstrumente gesampelt waren und nicht von „Life-Einspielungen“ zu unterscheiden sind.

Melodyne
Der Münchner Peter Neubäcker stellte auf der Musikmesse Frankfurt 2001 auf einem kleinen Stand sein Computerprogramm "Melodyne" vor, mit dem sich fertig aufgenommenes, einstimmiges Tonmaterial nachträglich in Zeit und Tonhöhe manipulieren lässt, ohne den Klangcharakter zu verlieren. Zur Vermarktung von Melodyne gründete Peter Neubäcker zusammen mit seiner Frau Hildegard Sourgens und Carsten Gehle die Firma Celemony[14].
Auf der Musikmesse 2008 stellte die Firma den Prototyp der nächsten Melodyne-Version vor, mit dem sich in mehrstimmiges Klangmaterial eingreifen lässt. Die Technik dahinter geht ebenfalls auf Peter Neubäckers Forschungen zurück. Er nennt sie, weil sie es zum Beispiel ermöglicht, bei einem aufgenommenen Gitarrenakkord direkt in einzelne Töne einzugreifen, "Direct Note Access" oder DNA. Zahlreiche Journalisten lobten diese Technik als Durchbruch, manche hielten sie für die Sensation der Messe.
Per Direct Note Access lassen sich also erstmals auch mehrstimmige Audio-Aufnahmen in einer Weise editieren, die man bislang nur von MIDI-Daten her kannte. Während einer Pressekonferenz wurde einem erstaunten Publikum vorgeführt, wie man beispielsweise in eine Aufnahme einer Jazz-Combo eingreifen

[14] http://www.celemony.com/cms/index.php?id=news&L=1

und per Keyboard die Solo-Trompete in einer anderen Melodie erklingen lassen kann. Gitarrenakkorde sind in einzelne Töne zerlegbar und lassen sich separat verändern, von einem Streichquartett gespielte Melodien können nachträglich per Software verändert werden – ein durch die Komplexität des notwendigen Erkennungsvorganges bislang unmöglicher Eingriff. Die erkannten Noten lassen sich in MIDI-Events umwandeln und damit auch von anderen Instrumenten spielen oder als Partitur ausdrucken.

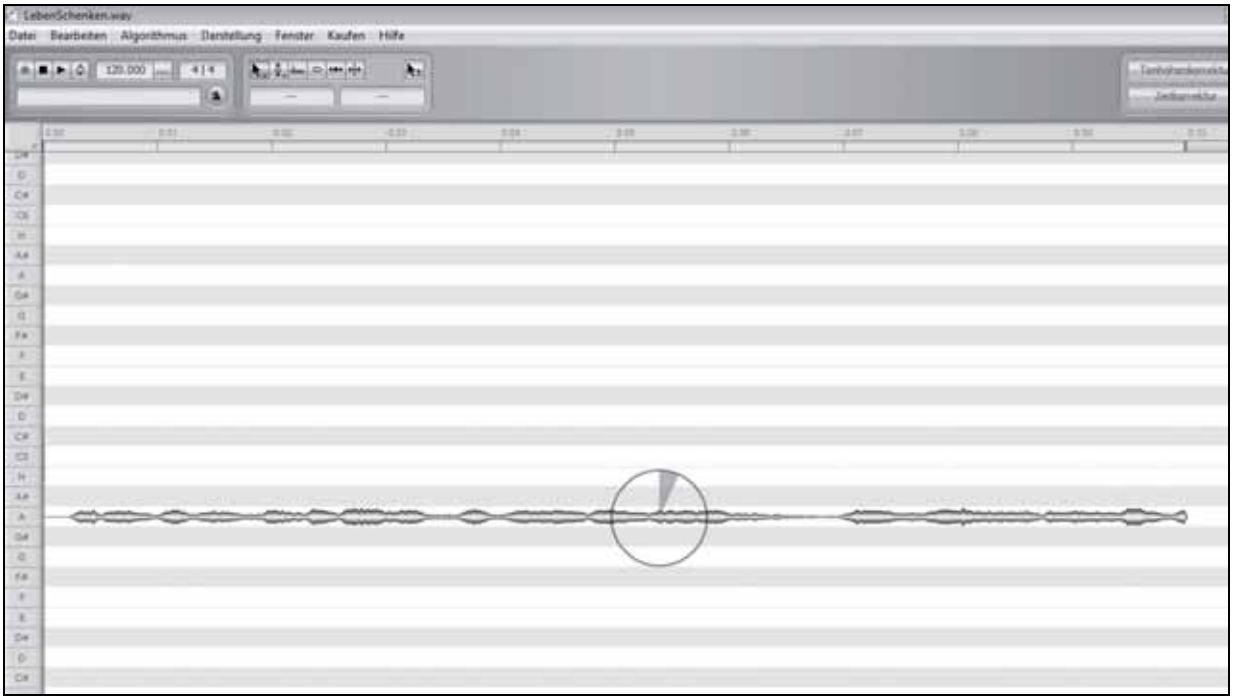


Abb. 5.3a Audio- Be- und Verarbeitung mit Melodyne®, Analyse der Wave-Datei

In Abb. 5.3a sehen wir ein Beispiel für DNA. Eine Wave-Datei wurde dabei in das Editor-Fenster gezogen und wird zunächst vom Algorithmus analysiert.

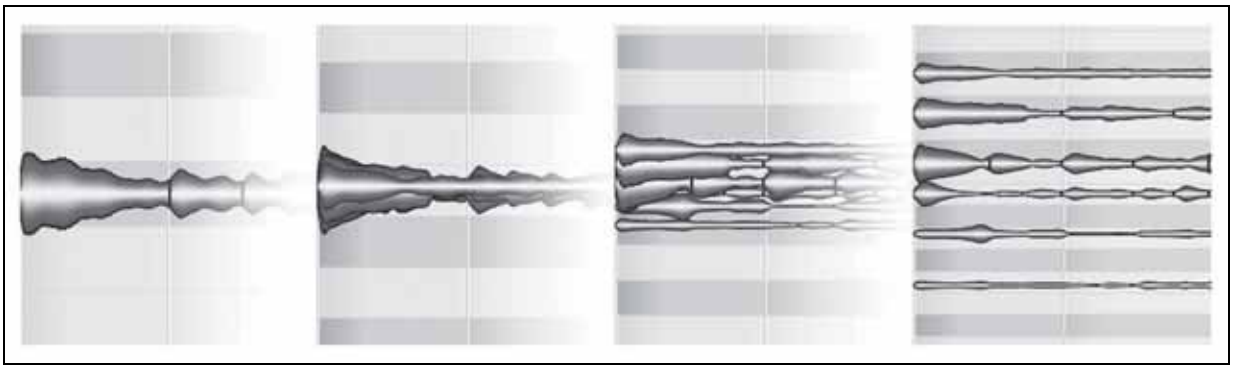


Abb. 5.3b Zerlegung der Wave-Datei mit Melodyne® in einzelne Noten-Waves

In Abb. 5.3.b sehen wir den Zerlegungsvorgang und in Abb. 5.3c das Ergebnis.

Die Wave-Datei wurde so zerlegt, dass einzelne Audio-Waves einzelner Instrumente sichtbar sind. Wie man sieht, hat der Algorithmus die kleinen Audio-Fragmente gleich an ihrer „richtigen“ Tonhöhe (vgl. linke Leiste) platziert.
Es ist jetzt möglich, mit der Maus die Höhe diese Töne zu verändern (durch hoch- und runterschieben), beliebig zu Platzieren und auch deren Länge zu ändern.

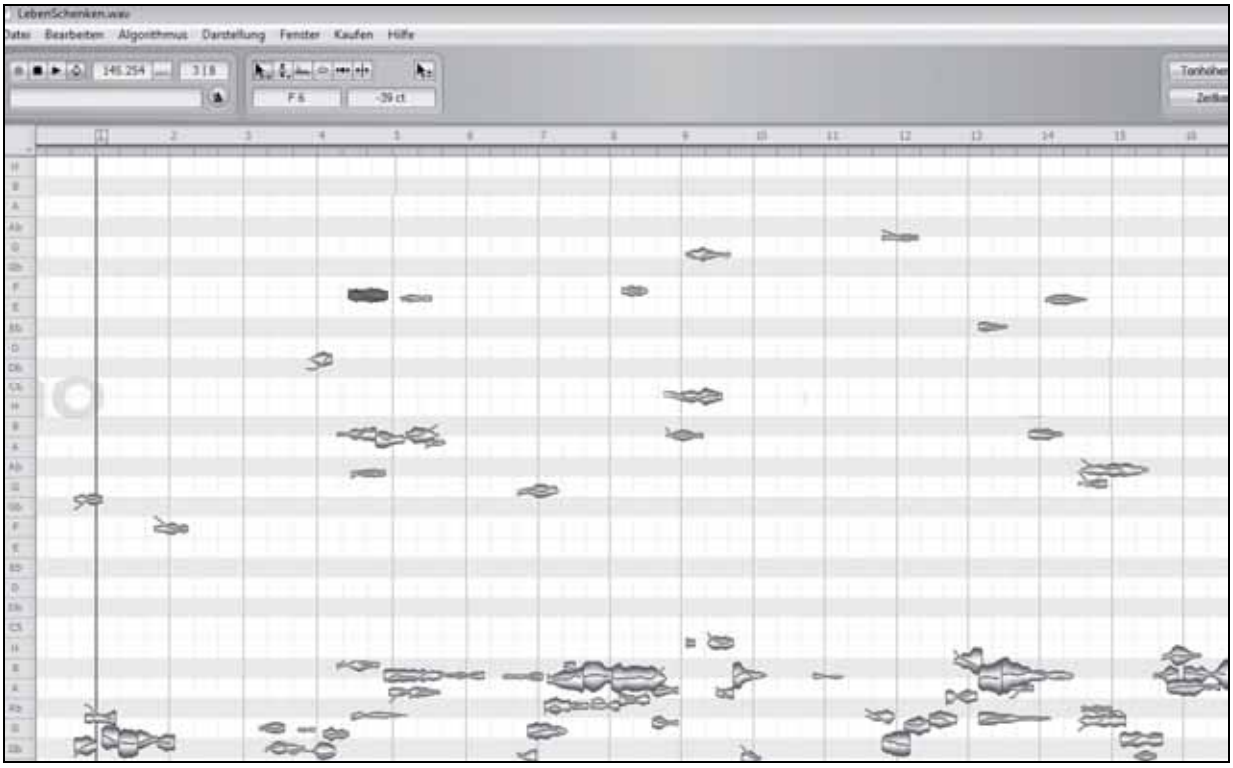


Abb. 5.3c Audio- Be- und Verarbeitung mit Melodyne®, zerlegte Wave-Datei

Der Melodyne zugrunde liegende Algorithmus ist „Betriebsgeheimnis“ von Celemony. Es werden wohl die Zahlenmuster der beteiligten Frequenzen der Originaldatei auf bekannte Zahlenmuster der Einzelfrequenzen hin untersucht und eine Zerlegung damit vorgenommen. In der Physik der Frequenzen und der Harmonielehre gibt es große Gemeinsamkeiten: man kann zeigen, dass nur bestimmte Überlagerungsfunktionen für das Ohr „harmonisch“ klingen. Einer Hypothese von Neurowissenschaftlern zu folge liegt das daran, dass das Blut im Gehirn nach einem bestimmten Muster fließt, welches offenbar bei Gehörtem, was diesem Muster entspricht, angenehme Gefühle auslöst: Eine Melodie oder Harmonie gefällt. Es ist sogar so, dass in verschiedenen Kulturen diese Muster des Blutflusses verschieden sind, was erklärt, dass verschiedene Kulturen verschiedene Musik als „schön“ empfinden.

Spezielle Video-Formate und Normen
Für die Be- und Verarbeitung von Video-Signalen sind neben den bereits in Kapitel 3 besprochenen Video-Formaten noch andere Dinge zu berücksichtigen. So spielen spezielle Übertragungsverfahren, Signalarten und Aufzeichnungsverfahren und –Medien eine besondere Rolle. Auf einige davon soll nun eingegangen werden.

Signalarten

RGB
Das in Primärfarben ROT-GRÜN-BLAU (daher RGB) zerlegte Signal wird in Grafikkarten und zur Wiedergabe an Monitoren verwendet. Zur Weiterverarbeitung wird das RGB-Signal selten benutzt, denn wenige Geräte verfügen über einen RGB-Eingang. Ein Signal im RGB-Format liegt bei der Aufnahme durch Kameras oder Wiedergabe über eine Kamera vor. Die Übertragung von Videodaten geschieht für jede Primärfarbe auf einer eigenen Leitung. Zur Synchronisation der Zeilen oder der Spalten wird meist eine getrennte Leitung verwendet. Werden Horizontal- und Vertikalsynchronisation getrennt geführt, braucht man 5 Leitungen. Es wird in manchen Fällen auch über die Grün-Leitung synchronisiert. Die Bandbreite des RGB-Signal beträgt 5MHz und höher für jedes Primärsignal.

YUV
Das YUV-Signal (auch Komponenten- oder Betacam-Signal genannt) eignet sich für professionelle analoge Aufzeichnungen mit bester Bildqualität (siehe auch Kapitel 3). Das YUV-Signal bildet zudem eine Zwischenstufe bei der Aufbereitung des RGB-Signals zum sog. FBAS-Signal (s.u.). Das RGB-Signal kommt wie das YUV-Signal ohne Farbhilfsträger aus. Die Transformation geschieht arithmetisch mit
$Y = 0.299R + 0.587G + 0.114B$
$U = 0.493(B-Y)$
$V = 0.877(R-Y)$
Die Y-Komponente repräsentiert die Helligkeitsinformationen (Luminanz), die U-Komponente (auch Cb) und die V-Komponente (auch Cr) die Farbinformationen (Chromanz). Die Übertragung geschieht über drei Koaxialkabel. Das Synchronisationssignal ist in Y enthalten. Zur Wiedergabe in schwarzweiß wird nur das Y-Signal genutzt. Das YUV-Signal kommt in der Studiotechnik zum Einsatz, wo Mischer und Kameras im Betacam-Standard arbeiten. Eine Rücktransformation ist ebenso arithmetisch möglich, die Signalqualität bleibt erhalten. Das YUV-Signal kann im 4:2:2 Format oder 4:2:0 Format digitalisiert wer-

den, und bietet damit eine beträchtliche Reduktion der Datenmenge. Die Bandbreite im 4:2:2 Format beträgt für U- und V-Signal standardmäßig jeweils 2.75MHz.

Y/C
Das Y/C-Signal, wegen seiner Weiterverarbeitung auch S-VHS-Signal genannt, verwendet einen Farbhilfsträger. Zur Übertragung sind 2 Leitungen erforderlich. Das Y-Signal entspricht dem des YUV-Signals und enthält lediglich Luminanz-Informationen sowie Signale für die Synchronisation. Die Farbinformation wird im Prinzip durch Addition des U- und V-Signals gewonnen und einem Farbhilfsträger überlagert. Der Farbhilfsträger hat eine Frequenz von 4.43MHz, die zu übertragenden Farbinformationen bewirken eine Amplitudenmodulation. Die Bandbreite des Y-Signal beträgt auch hier 5MHz, die des C-Signal nur noch 1.3MHz. Dadurch ergibt sich eine schlechtere Farbauflösung. Verwendet wird diese Signalart im semiprofessionellen Bereich in Kameras, Recordern und auch Videokarten.

FBAS
Das FBAS-Signal (Farb, -Bild- Austast- und Synchronisationssignal) entsteht aus dem Y/C-Signal durch Addition von Y und C. Zur Signalübertragung ist nur eine Leitung notwendig. Die Mittelwerte der Schwingungen entsprechen den Luminanzwerten. Die Synchronisationsimpulse entstammen dem Y-Signal. Die Verschmelzung der Signale zieht Störungen nach sich, vor allem Helligkeitsschwankungen in der Nähe des Farbhilfsträgers (4,43MHz). Die Erzeugung von Videokopien verschlechtert das Signal weiterhin. Die drahtlose Übertragung von Analogsignalen erfolgt als FBAS-Signal. VHS-Recorder können oft nur FBAS-Signale aufzeichnen, viele Videokarten für den Amateurbereich beschränken sich auf FBAS-Signale.

Digitale Signale
Das sogenannte D1-Signal ist das digitalisierte YUV-Signal nach CCIR 601 (4:2:2 Format), das D2-Signal die digitalisierte Form des FBAS-Signals. Die Übertragung kann verlustfrei erfolgen. Die Abtastung erfolgt mit 13,5 MHz für die Y-Komponente und 6,75 für die Farbkomponenten. Die einzelnen Komponenten werden mit 8 Bit quantisiert.

Aufzeichnungsverfahren

Unabhängig von der Signalart ist zur Speicherung auf einem Bandmedium eine Signalaufbereitung notwendig. Die Unterschiede liegen in den Bandformaten,

deren Aufteilung und Beschreibung sowie Signalaufbereitung. Neben den in Studios verwendeten 1-Zoll-Systemen existieren eine Reihe von Standards für die Aufzeichnung analoger sowie digitaler Signale. Die gebräuchlichsten Verfahren waren bzw. sind:

VHS
VHS wurde von JVC entwickelt. Ein VHS-Recorder ist nur für die Signalart FBAS ausgelegt. Bei der Aufzeichnung erfolgt zunächst eine Trennung in Luminanz- (Y) und Chrominanzsignal (C). Das Y-Amplitudensignal wird in ein Frequenzsignal FM übertragen. Das FM-Signal wird von einer 4.3MHz Frequenz überlagert. Die Chrominanz wird mit einer Frequenz von 627Khz überlagert. Stereoton wird mit 1.4 bzw. 1.8 MHz überlagert. Diese Frequenzen werden zusammen übertragen und beim Empfänger wieder decodiert. Keine eigene (Stereo-)Tonspur, keine (Stereo-)Nachvertonung, Bandgeschwindigkeit beträgt 2.34cm/s, die Lesegeschwindigkeit 4.87 m/s.

S-VHS
Das Super-VHS Verfahren entspricht weitgehend dem VHS-Verfahren, jedoch werden Y- und C-Signal durchgängig getrennt verarbeitet. Dadurch ist eine wesentlich bessere Bildqualität zu erreichen.

Video Hi-8
Für Camcorder entwickeltes System mit kleiner Kassette. Die Weiterentwicklung des Camcorderformats Video-8 nutzt dafür entwickelte ME-Bänder für große Schreibdichte. Video Hi-8 genügt semiprofessionellen Anwendungen und entspricht sonst technisch dem S-VHS-Verfahren.

High-Band U-matic
In Studios verwendeter Standard, der spezielle Kassetten verlangt. Das Verfahren ähnelt S-VHS, aber gestattet HiFi-Tonqualität. Die Bandgeschwindigkeit beträgt 9.5cm/s, die Lesegeschwindigkeit 10.7 m/s. Die Y- und C-Komponente des Y/C-Signals werden zusammen abgelegt. Es existiert eine eigene Timecode-Spur.

Betacam
Entwickelt von Sony für Studiosektor mit bester Bild und Tonqualität. Einer der wichtigsten Standards bei professionellen Anwendungen. Die YUV-Signale werden getrennt verarbeitet. Bandgeschwindigkeit beträgt 10.15cm/s, die Lesegeschwindigkeit 5.75 m/s. Bildgenaues Schneiden durch Timecode-Spur ist möglich.

Digitale Aufzeichnung
Die digitale Aufzeichnung geschieht i.d.R. nach dem 4-2-2 Standard. Bei digitalen Systemen wird das Y-Signal doppelt so häufige abgetastet (13,5 MHz) wie das U- oder V-Signal (6,75 MHz), weil das menschliche Auge die Luminanzinformation besser auflösen kann als die Chrominanzinformation. Audiodaten werden mit einer Abtastrate von 48 KHz zwischen den Videospuren abgelegt.
Neuerdings werden die bereits beschriebenen Formate MPEG2 und MPEG4 benutzt um Videos aufzuzeichnen. Dabei kann auf Kassettenbänder oder DVD oder Speicherkartenmedien zurückgegriffen werden.

Farbübertragungsverfahren

PAL
(Phase Alternation Line) Ein Standard der Farbübertragung. Unterschiede durch Netzfrequenz und Eigenentwicklung einzelner Länder bedingt. Es handelt sich um ein 625/50 System mit 625 Bildreihen und Bildablenkung von 50 Hz. Eigentliche Bildgröße beträgt 768x576 Bildpunkte, sichtbar ist daher nur ein Teil.

NTSC
(National Television System Committee) Der 1953 in den USA entwickelte Standard ist auch Grundlage des PAL-Systems. Ein Vollbild weist 525 Zeilen auf, die Bildablenkung arbeitet mit 60 Hz. Das Luminanzsignal wird wie bei PAL gebildet, das Chrominanzsignal etwas anders. Die eigentliche Bildgröße beträgt 640x480 Bildpunkt, sichtbar ist nur ein Teil.

SECAM
(secuentielle a memoire) Secam wird in Frankreich und einigen anderen europäischen Ländern eingesetzt. Die Bild- und Zeilenfrequenz entsprechen dem PAL-Standard, die Aufbereitung der FBAS- und Y/C-Signale sind jedoch zu PAL verschieden.

Durch den neuen HDTV-Standard (vgl. Kapitel 3) verschwinden die nationalen Unterschiede zunehmend. Außerdem können heute praktisch alle digitalen Wiedergabegeräte (wie DVD-Player) alle Normen in Echtzeit so wandeln, dass es keine Rolle mehr spielt, mit welchem Farbübertragungsverfahren eine Aufnahme getätigt wurde und in welchem sie wiedergegeben werden soll.
Damit kommen wir zu den praktischen Anwendungen der Video-Verarbeitung.

Video-Bearbeitung in der Praxis

Über dieses Thema kann man dicke Bücher schreiben, so dass hier nur auf die wichtigsten Bearbeitungsanwendungen und Problemfelder eingegangen werden kann. Es gibt zahlreiche Video-Bearbeitungsprogramme unterschiedlicher Professionalität auf dem Markt. In allen Fällen gilt aber: Je schneller der PC, je mehr RAM vorhanden und je besser die Grafikkarte ist, desto besser lassen sich Videos bearbeiten. Dennoch ist es kaum möglich, ein Video in voller Auflösung in Echtzeit zu bearbeiten. Alle Bearbeitungsprogramme bedienen sich eines Tricks: In den Vorschaufenstern wird eine kleinere Auflösung benutzt, und mit dieser werden die Bearbeitungsschritte angezeigt. Auch werden z.B. herauszuschneidende Filmteile oder Effekte nicht wirklich an „Ort und Stelle“ getätigt, sondern die jeweiligen Parameter zwischengespeichert und in „abgespeckter“ Version angezeigt. Wenn z.B. ein Werbeblock aus einem Film herausgeschnitten wird, so werden in Wirklichkeit die Anfangs- und Endpunkt der Schnittstelle nur markiert und bei der Vorschauanzeige überspringt die Wiedergabe einfach die so markierte Stelle. Erst wenn das Video abgespeichert wird, werden die markierten Szenen wirklich herausgenommen und Effekte etc. hinzugefügt. Diesen Vorgang nennt man *rendern.* Die meisten Video-Bearbeitungsprogramme verfügen über ein Feature, das *Smart-Rendering* genannt wird. Dabei werden nur die Teile neu berechnet, die tatsächlich verändert wurden. Die anderen Teile werden einfach nur kopiert. So wird z.B. beim Herausschneiden eines Werbeblocks im Falle des Smart Renderings einfach nur die Datei bis zum Werbeblock kopiert, der Block selbst weggelassen und direkt dahinter der Rest hinzugefügt. Also keine Neuberechnung des gesamten Films, sondern nur ein Hintereinanderkopieren der sonst unveränderten Filmteile.

Beim Rendern kann es passieren, dass das Ergebnis schlechter aussieht als das Original. Meine Erfahrung zeigt, dass es sehr große Qualitätsunterschiede bei den Softwareherstellern gibt, je nach benutzten Codecs für die Umwandlungen (gilt jetzt für wirkliche Neuberechnungen). So gibt es gute und weniger gute Algorithmen, die die Hersteller in ihren Bearbeitungsprogrammen untergebracht haben. Meistens sind Eingriffe wie Kontrastanhebungen, Helligkeitsanpassungen oder Weichzeichnen relativ unkritisch. Doch sobald es um Größenveränderungen wie z.B. Verändern der Auflösung, Änderung der Durchsatzrate oder der TV-Norm (z.B. PAL nach NTSC) geht, treten eklatante Qualitätsunterschiede bei verschiedenen Herstellern auf, insbesondere auch beim Wandeln von Formaten (z.B. von MPEG2 nach MPEG4 etc.). Wenn solche Probleme vorhanden sind, sollte man die Codecs eines anderen Herstellers ausprobieren.

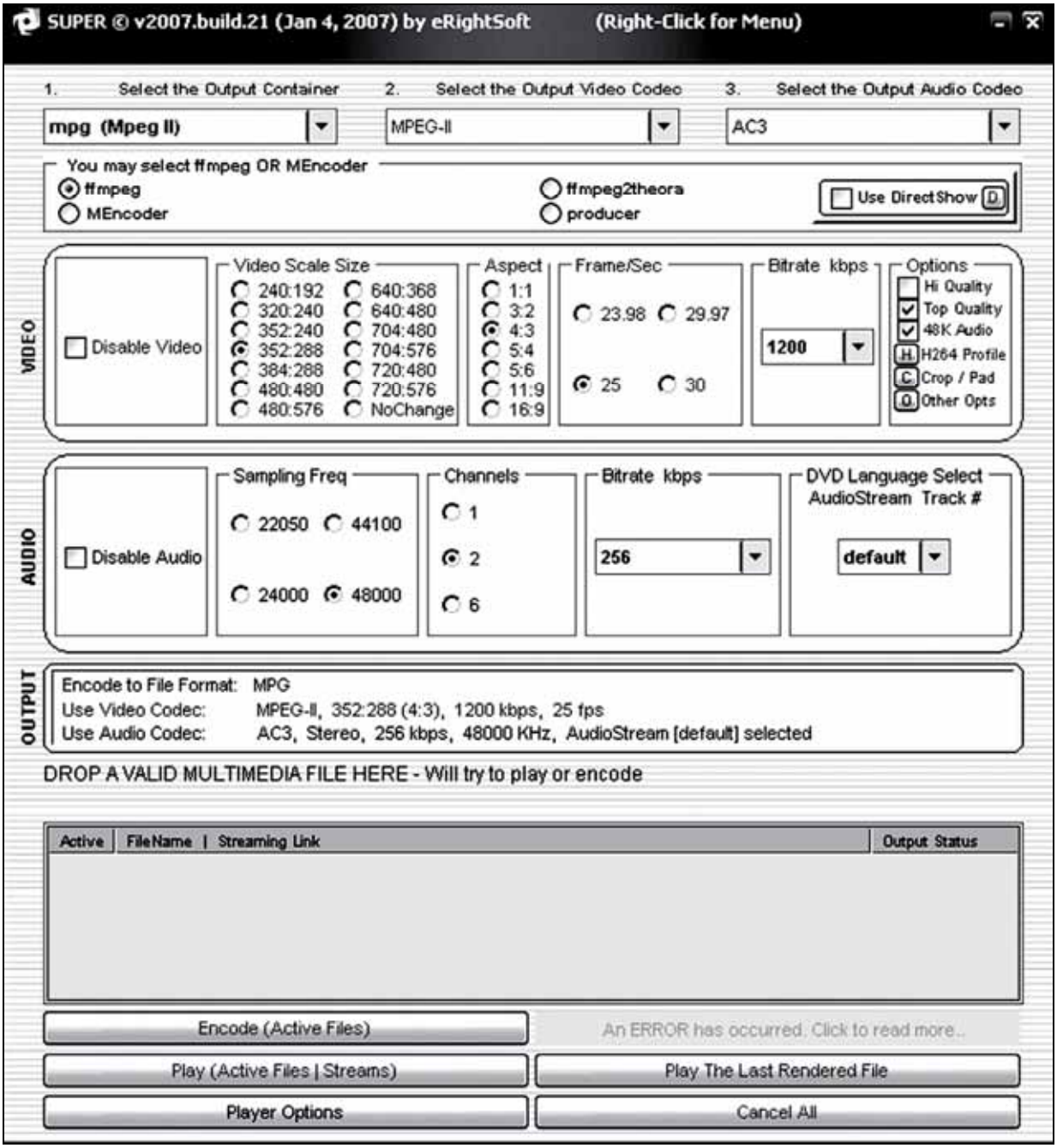


Abb. 5.4 Formatwandlung mit SUPER®

In Abb. 5.4 ist die Oberfläche eines recht universell einsetzbaren Audio- und Video-Formatwandlers zu sehen: der Super®-Video-Konverter der Firma eRightSoft[15]. Er ist in der Lage, aus einer Vielzahl von Audio- und Video-Formaten hin- und herzuwandeln, wobei die Parameter in großem Umfang festlegbar sind.

[15] kostenlos unter http://www.erightsoft.com/home.html

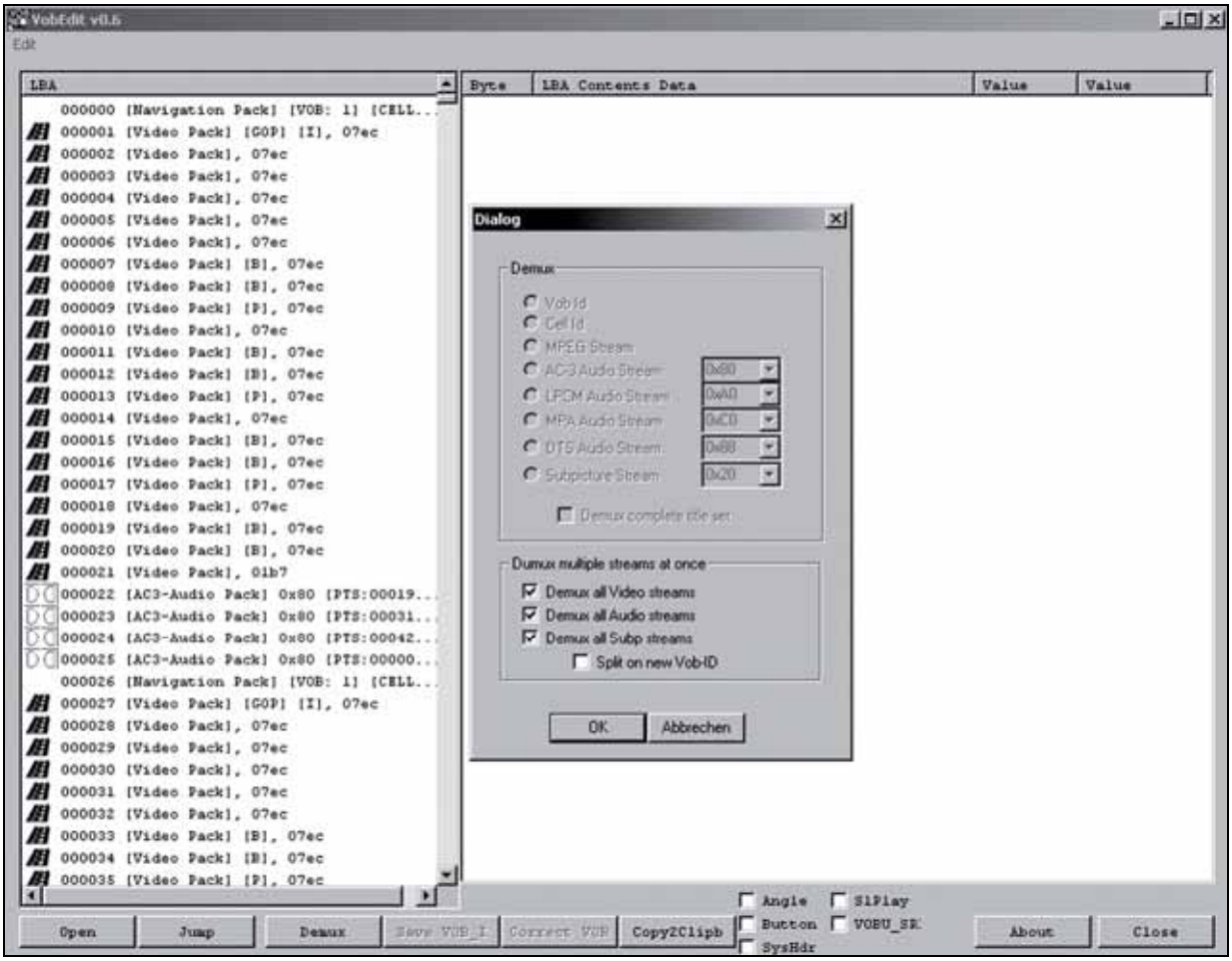


Abb. 5.5 VOB-Edit zum Umwandeln von VOB in MPEG2-Streams

Ein wichtiger Begriff ist der des *Multiplexing* und *De-Multiplexing*. Letzterer Begriff bezeichnet das Zerlegen eines kombinierten Video-Audio-Streams in den Video-Anteil und den Audio-Anteil, während das Multiplexing das Zusammenfügen entsprechender Video- und Audio-Dateien bedeutet.
Es sind z.B. auf DVD häufig Filme in mehreren Sprachversionen vorhanden. Mit dem De-Multiplexing kann man das Video und die einzelnen Audio-Streamings getrennt erhalten. Die MPEG2-Videos sind auf einer DVD im sog. VOB-Format (VOB=Video-Object) abgelegt. Dabei handelt es sich zwar um MPEG2-Dateien, doch diese sind um bestimmte zusätzliche Parameter erweitert und können mehrere Audio-Streams (z.B. für mehrere Sprachen) enthalten. Außerdem ist eine VOB-Datei nicht größer als 1048 MB. Ist ein MPEG2-File ursprünglich größer gewesen, wird er in mehrere VOBs zerlegt. Dieser Sachverhalt ist auf die ursprüngliche Beschränkung von Datenfiles auf max. 4 GB Größe unter dem Dateisystem FAT32 bzw. der ISO-Norm bei DVDs notwendig gewesen. Diese Beschränkung wurde erst mit NTFS aufgehoben.
VOB-Dateien können mit Freeware wie z.B. „VOB-EDIT“ von Decision Developement in MPEG2-Streams umgewandelt werden. Diese Software setzt

dabei automatisch zusammengehörige VOB-Files zu einem MPEG2-File zusammen (vgl. Abb. 5.5), trennt dabei aber durch De-Multiplexing auch die Video-Spur von allen Audiospuren. Will man dann diese Spuren wieder zusammensetzen, so muss man einen Multiplexer benutzen. Auch dafür gibt es reichlich Freeware und auch fast alle Video-Bearbeitungsprogramme verfügen über ein entsprechendes Werkzeug. Audio- und Video-Streams sind dabei mit einem Zeitstempel versehen, sodass keine Synchronisationsprobleme auftreten sollten.
Sollte letzteres doch passieren, kann man die Audio- und Videospur durch De-Multiplexing zerlegen und mit einem geeigneten Video-Bearbeitungsprogramm getrennt bearbeiten (z.B. die Audiospur gegenüber der Bildspur verschieben bis die Synchronisation wieder stimmt). Dies sei an einem kleinen Beispiel demonstriert, wo man auch sieht, welche Möglichkeiten in solchen Video-Bearbeitungsprogrammen stecken.
Wir benutzten dafür das Programm „MPEG Video Wizzard“ der Firma Womble Multimedia[16]. Dieses Video-Bearbeitungsprogramm zeichnet sich durch seine leichte Handhabbarkeit aus, verfügt allerdings nicht über so umfangreiche Features wie z.B. das Profi-Programm Adobe® Premiere[17].
Angenommen, ein MEPG2-Stream hat eine Tonspur, dessen Audio etwa ab der Mitte des Films plötzlich um ca. eine halbe Sekunde zu früh kommt (im Verhältnis zu den Lippen des Sprechers, d.h. das Bild ist ab hier „zu spät dran“). Was tun? Nun, zunächst wird das Video in den Wizzard geladen und dann in die Video-Zeitleiste gezogen (vgl. Abb. 5.6). Danach zieht man den gleichen Video-Clip auf eine der Tonspuren unterhalb der Video-Spur (vgl. Abb. 5.7, Spur mit dem Gitarrensymbol). Dadurch wird dort noch mal (nur) die Tonspur des Clips wiedergegeben. Doch der Originalton aus der Video-Spur ist auch noch vorhanden, so dass man diesen durch Betätigen der rechten Maustaste auf die Videospur mittels „Mute“ abschalten muss. Starten man nun das Abspielen, sieht man die Video-Spur und hört die Audiospur (des Gitarrensymbols) dazu (welche noch unverändert ab der Mitte des Films unsynchron ist).

[16] www.womble.com
[17] www.adobe.com

Abb. 5.6 Unsynchrones Video in MPEG Video Wizzard

Abb. 5.7 Audiospur duplizieren und Originalton abschalten

Als nächstes positioniert man den Navigations-Cursor an die Stelle, ab der der Ton unsynchron ist, klickt mit der rechten Maustaste auf die Tonspur und wählt „Split“ aus. Dieses führt dazu, dass die Tonspur jetzt hier in zwei Teile zerlegt wird (Abb. 5.8).

Abb. 5.8 Audiospur splitten

Danach lässt sich der rechte Teil der Audio-Spur beliebig verschieben, insbesondere so, dass er wieder lippensynchron zur Videospur wird (Abb. 5.9).

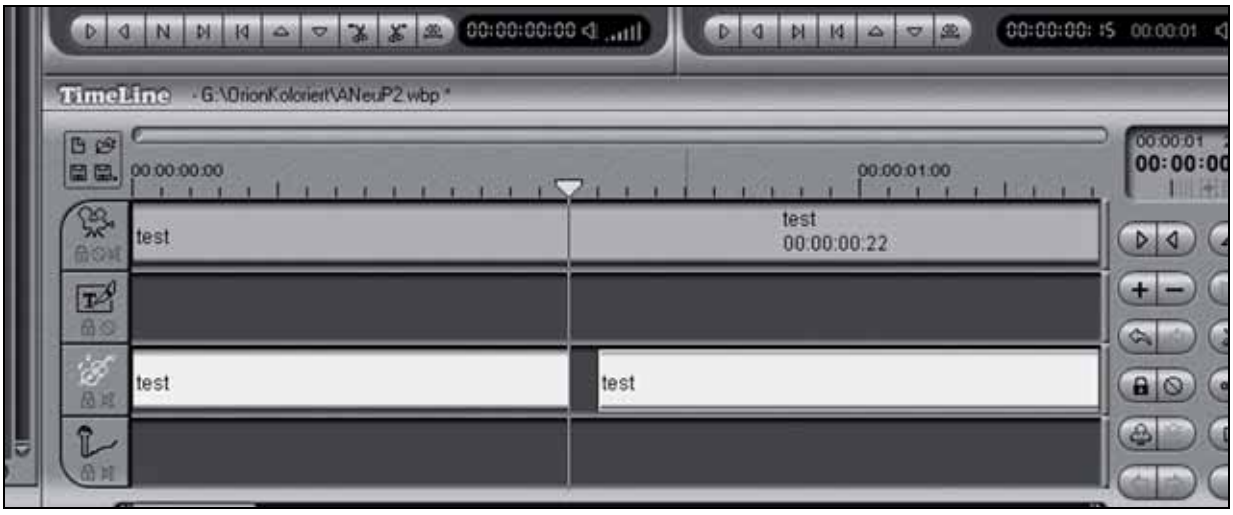


Abb. 5.9 Teil der Audiospur verschieben

Man kann den Film jetzt durch Abspeichern in der neuen Version rendern und erhält damit ein komplett lippensynchrones Video. Natürlich können auch die Lautstärkeverhältnisse etc. der Audiospur geändert werden, ebenso wie z.B. Kontrast oder Farbstärke im Video. Hierfür ist aber immer ein komplett neues Rendering des Films notwendig (was zeitlich häufig länger dauert wie die Filmlänge selbst in Echtzeit beträgt).
Ich möchte nun noch eine kleine Tabelle angeben, wo häufig auftretende Probleme beim Rendern solcher bearbeiteten Videos durch entsprechende „Gegenmaßnahmen“ gelöst werden können.

Problem	**Mög. Ursache**	**Lösung**
Video „ruckelt“ bei der Wiedergabe	Schlechter Software-Player	Anderen Player ausprobieren, z.B. kostenlosen Mplayer (http://www.mplayerhq.hu), bringt alle Codecs selbst mit
	Zu wenig RAM	Mehr RAM installieren
	Langsame Grafik-Karte	Andere Grafikkarte benutzen
Video ruckelt erst nach dem Rendern, Original ist flüssig	Falls Auflösung vergrößert wurde, evtl. gleiche Gründe wie oben	Kleinere Auflösung ausprobieren beim Rendern oder Video z.B. auf DVD brennen und auf DVD-Player anschauen, evtl. dort kein Ruckeln mehr
	Manche Kompressionsalgorithmen kommen mit scharfen Kanten oder verrauschten Bildern (z.B. alte Videos von Kassette o.ä.) nicht zurecht (häufig bei MPEG2 und MPEG4, DivX und WMV zu beobachten)	Andere Kompressionsalgorithmen probieren oder: Originalvideo unverändert neu rendern, wobei aber dabei etwas „Blur“ (Unschärfe) hinzufügt werden muss (mit Video-Bearbeitungsprogramm), so dass Kanten unschärfer und Rauschen geringer wird. Danach nochmals rendern in gewünschtes Format
Nach dem Rendern entstehen kleine „Klötzchen“ (Artefakte), vor allem bei Bewegungen	Zu geringe Bitrate	Bitrate erhöhen
	Evtl. gleiche Ursachen wir beim Ruckeln (s.o.)	Siehe Lösungen beim Ruckeln (s.o.)

Erstellen von Video-DVDs

Wie bereits erwähnt, werden DVDs, die auf dem heimischen DVD-Player abspielbar sein sollen, in einem bestimmten Format erzeugt. Die MPEG-Filme (im MPEG1 oder MPEG2-Format) müssen in sog. VOB-Files konvertiert werden; dabei findet allerdings keine wirkliche Formatwandlung statt, sondern die vorhandenen MPEG-Files werden mit Zusatz-Information ausgestattet, es werden evtl. mehrere Audio-Spuren (wie verschiedene Sprachen) hinzugefügt und das ganz in Vielfache von max. 1048 MB zerlegt (falls größer als 1048 MB).

Schließlich wird noch weitere Information für die Menügestaltung und die Kapitelinformation abgelegt.
Die Verzeichnisstruktur[18] und Dateinamen einer DVD sind ebenfalls standardisiert. Folgende Struktur muss im Dateisystemformat UDF gegeben sein, damit eine maximale Kompatibilität mit jedem DVD-Player sichergestellt ist:
AUDIO_TS (Audio Title Sets). Dieses Verzeichnis ist für die Kompatibilität mit einer DVD-Audio nötig. Meistens ist dieses Verzeichnis vorhanden, aber leer.
JACKET_P (Jacket Picture). Dieses Verzeichnis gehört nicht zur offiziellen DVD-Video Spezifikation, ist aber oft vorhanden. Bestimmte DVD-Player nutzen dieses Verzeichnis, um aus ihr eine Grafikdatei auszulesen, z.B. zum Anzeigen eines Logos. Das Logo muss mehrfach für verschiedenen Auflösungen und Fernsehnormen abgelegt werden:

- J00___5L.MP2 Bilddatei in großer Auflösung von 720×480 Pixel für NTSC-Fernseher.
- J00___5M.MP2 Bilddatei in mittlerer Auflösung von 176×112 Pixel für NTSC-Fernseher.
- J00___5S.MP2 Bilddatei in kleiner Auflösung von 96×64 Pixel für NTSC-Fernseher.
- J00___6L.MP2 Bilddatei in großer Auflösung von 720×576 Pixel für PAL-Fernseher.
- J00___6M.MP2 Bilddatei in mittlerer Auflösung von 176×144 Pixel für PAL-Fernseher.
- J00___6S.MP2 Bilddatei in kleiner Auflösung von 96×80 Pixel für PAL-Fernseher.

Die typische DVD-Video-Struktur ist **VIDEO_TS** (Video Title Sets); dieses Verzeichnis enthält die eigentlichen Videodaten einer Video-DVD. Folgende spezifizierte Dateien sind dort u. a. anzutreffen:

- VIDEO_TS.IFO: Diese Datei enthält Informationen zum Video Manager Menu (VMGM) für Aufbau und Navigation sowie Informationen zur Wiedergabe der VIDEO_TS.VOB-Dateien.
- VIDEO_TS.BUP: Backup der VIDEO_TS.IFO-Datei (auch VMGI_BAK für „Video Manager Information Backup“ genannt).
- VIDEO_TS.VOB: enthält die Video-Objekte für das Titel-Menü und gemultiplexte Video-, Untertitel- und Audiodateien. Diese Datei wird auch als VMGM_VOBS für „Video Manager Information Video Object Set“ bezeichnet.

[18] http://de.wikipedia.org/wiki/DVD-Video

- VTS_01_0.IFO: (Video Title Set Information; VTSI) enthält Informationen über das Video Title Set und das Video Title Set Menu. Die erste Zahl (01) gibt die Title Set Nummer an, die zweite Zahl (0) ist immer 0.
- VTS_01_0.BUP: (VTSI_BAK) Backup der VTS_01_0.IFO-Datei.
- VTS_01_0.VOB: (VTSM_VOBS) enthält die Video-Objekte des VTS-Menüs. Diese Datei ist nur vorhanden, wenn dieses Title Set ein Menü besitzt. Die erste Zahl (01) gibt die Title Set Nummer an, die zweite Zahl (0) ist bei Title Set Menu VOBs immer 0.
- VTS_01_1.VOB: (VTSTT_VOBS) enthält die Video-Objekte der Titel. Die erste Zahl (01) gibt die Title Set Nummer an, die zweite Zahl (1) die Dateinummer (die maximale Dateigröße auf DVD-Videos ist 1 GB, weshalb es –wie bereits erwähnt- notwendig werden kann, die Daten in mehrere Dateien aufzuteilen).

Das Erstellen von Video-DVDs geschieht normalerweise mit einer entsprechenden Authoring-Software. Diese sind sowohl kostenlos als Freeware zu erhalten[19] wie auch in professioneller Ausstattung im kommerziellen Bereich. Dabei sind zwei Varianten zu unterscheiden: Es gibt DVD-Authoringsysteme, die lediglich die bereits „fertigen“ MPEG2-Files entsprechend aufbereiten (also die VOBs daraus erzeugen und die Menüstruktur etc.), und welche, die auch Transkodierungen durchführen. Darunter versteht man das komplette neu rendern der Dateien, meistens verbunden mit vorherigem Demultiplexen und anschließendem Multiplexen. Letzteres kostet allerdings vieles an Rechenzeit, oft mehr, als die DVD zeitlich gesehen beim Abspielen dauert. Der Vorteil ist dabei aber, dass das Ergebnis i.d.R. wirklich DVD-Player-kompatibel ist, da hierfür die Normen und Formatvorgaben des offiziellen DVD Forums (eine Vereinigung von mehreren Hundert Firmen, die einen Quasi-Standard für DVD-Formate festlegten) eingehalten sind.
Abb. 5.10 zeigt beispielsweise die DVD-Entwicklungsoberfläche der Freeware „DVDStyler“, welche selbst keine Neuberechnungen vornimmt, d.h. die MPEG-Files müssen bereits in der „richtigen“ Auflösung und Größe vorliegen. Man zieht so einen File einfach in die untere Leiste und kann dann den vorher in dem Hauptmenü zugefügten „Knöpfen“ den Film zuweisen, wobei Farben, Beschriftungen etc. geändert werden können. Durch Doppelklick auf den Titel unten kann man auch die Zeitmarken für die Kapitel verändern. Somit sind also die wichtigsten Tätigkeiten für das Erstellen einer Video-DVD vorhanden. Möchte man jedoch selbst Untertitel oder mehrsprachige Filme unterbringen, so muss

[19] siehe z.B. http://www.dvdstyler.de

man sich einer kommerziellen Software (wie z.B. DVD Encore® von Adobe) bedienen.

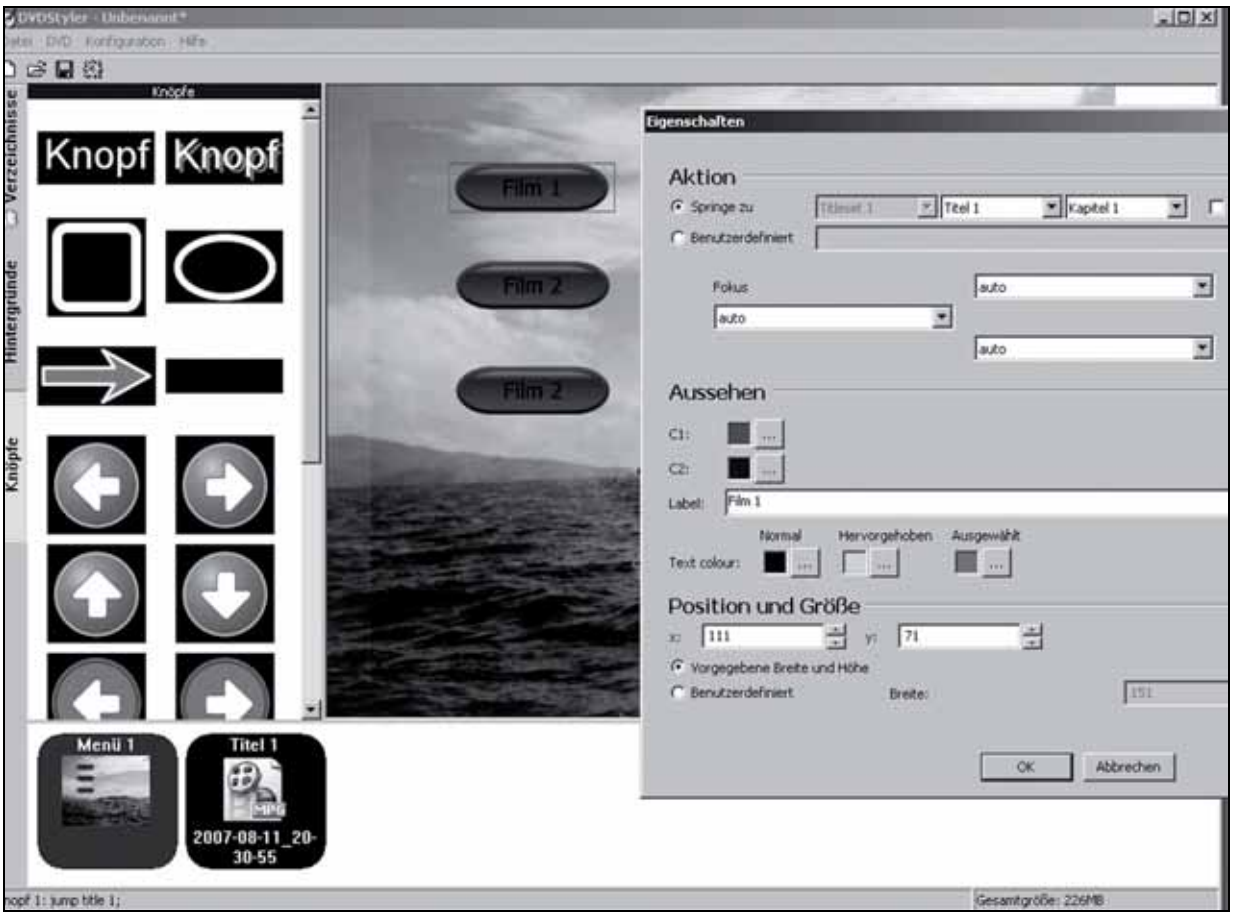


Abb. 5.10 DVD erstellen mit Freeware DVDStyler

Das nächste Kapitel beschäftigt sich mit einer weiteren wichtigen Anwendung von Multi-Media-Systemen: Animationen und Computergames.

Übungen zum Selbsttest

Angenommen, es sei ein Audio-Datenstrom (Wave-Datei) mit einer Samplingrate von 8 kHz und einer Quantisierung von 8 Bit eine Sekunde lang aufgenommen. Wie viele Sekunden könnte man (ungefähr) für die gleiche Datei-Größe in einer MIDI-Datei codieren, wenn man annimmt, dass bei einem 4/4-Takt mit durchschnittlich 120-Viertelnoten pro Minute aufgenommen wurde (Overheads etc. braucht man nicht zu berücksichtigen, jede Note stelle einfach einen MIDI-Event dar)?

6. Computer Games und Animationen

Aufgrund des beschränkten Platzes kann dieses Thema hier natürlich nur gestreift werden. Es wird daher hier eine (nicht notwendig repräsentative) Auswahl bestimmter Techniken getroffen.
Zunächst sind einige Begriffe zu klären, welche in diesem Zusammenhang vorkommen. Wir haben in Kapitel 3 bereits DirectX eingeführt und die damit verbundenen Technologien wie DirectDraw, Direct3D etc. kurz angeschnitten.
Abb 6.1 zeigt das Zusammenspiel dieser Komponenten:

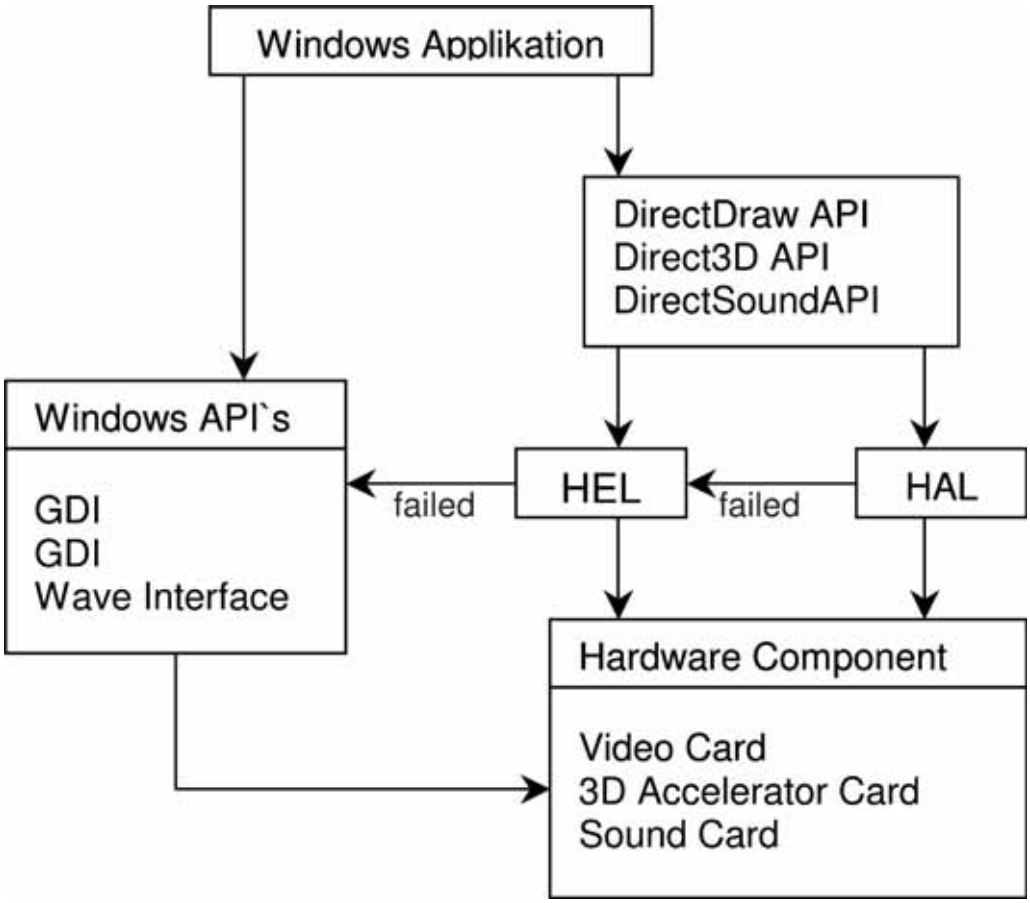


Abb. 6.1 DirectX-Pipeline[20]

Ganz oben in Abb. 6.1 steht die Windows Applikation, die diverse DirectX API's (im Beispiel: DirectDraw**,** Direct3D und DirectSound) einbindet. Die Methoden dieser API's versuchen dann, die aufgerufene Funktionalität durch direkte Hardwareunterstützung, *Hardware Abstraction Layer* ***HAL***, zu erzeugen. Darunter versteht man eine architektonische Ebene größerer Computerpro-

[20] Quelle: Kiana Nemati, „3D-Spieleprogrammierung mit DirectX und C++", Diplomarbeit am Studiengang Informatik, Fachhochschule Frankfurt am Main, 2005

gramme oder Systeme solcher Programme, die dafür sorgt, dass andere Software-Komponenten nicht auf die Einzelheiten der Hardware Rücksicht nehmen müssen, auf der das Programm läuft. Wird die verlangte Funktionalität von der vorhandenen Hardware nicht unterstützt, dann versuchen die API's diese durch Softwareemulation zu simulieren, welches die Aufgabe des Hardware Emulation Layers HEL ist. Je mehr Features emuliert werden müssen, desto mehr Performance geht dabei verloren. Wenn der Versuch der Softwareemulation fehlschlägt, so greift die API auf die letzte Möglichkeit zurück: die Windows Standard API's, wie beispielsweise das GDI oder das Wave Interface. Da diese API's integrale Bestandteile von Windows sind, können die nachgefragten Funktionalitäten auf alle Fälle ausführt werden. Im letzten Schritt wird die ausgeführte Funktionalität auf die Hardware ausgegeben, also zum Beispiel Bilder durch die Grafikkarte und Sound durch die Soundkarte. Am schnellsten wird die nachgefragte Funktionalität durch das HAL erledigt; insbesondere bei der Direct3D API ist eine 3D Beschleunigerkarte, die DirectX unterstützt, sehr empfehlenswert. Wesentlich langsamer ist das HEL und die nicht zu empfehlende Alternative sind die Standard Windows API's.
Zur Modellierung einer 3D-Welt benötigt man eine sog. 3D-Engine. Darunter versteht man dasjenige Modul eines Computerprogramms, welches für die Aufbereitung aufwändiger Computergrafik zuständig ist, meist für möglichst realitätsgetreue 3D-Computergrafik, wie Gegenstände, Umwelt und Personen. Sie bietet einem Programmierer eine große Palette von grafischen Funktionen und Effekten (geometrische Objektbeschreibung, Oberflächentexturen, Licht und Schatten, Transparenz, Spiegelungen usw.), so dass er für seine spezielle Anwendung diese nicht stets neu programmieren muss.
Bezogen auf Computerspiele wird 'Engine' also als Motor verstanden und bildet die Grundlage und die Schnittstelle zwischen dem Spiel und dem Spieler. Sie nimmt durch die Programmierbibliothek, welche immer wieder auftretende Abläufe automatisiert, dem Programmierer viel Arbeit ab. In dieser Bibliothek befinden sich die Strukturen, Funktionen und Algorithmen, die zur Bildschirmausgabe dienen (d.h. 3D-Objekte möglichst effizient auf einer 2D-Fläche darzustellen). Hier werden auch alle Berechnungen und Operationen durchgeführt, welche mit der Benutzerschnittstelle (d.h. Abfragen der Maus/Tastatur) und Ton (d.h. Abspielen von Sound-Dateien, Effekte) zusammenhängen. Unterschiede zwischen der vielfältigen 3D-Hardware werden aufgelöst. Aufgrund der langen Entwicklungszeit für eine ausgereifte 3D-Engine und wegen der großen Basisfunktionalität, welche jedes 3D-Programm benötigt, werden 3D-Engines nicht für jedes Programm neu erstellt, sondern lediglich angepasst und erweitert. 3D-Objekte werden mit externen Programmen erstellt und dann in das Programm eingelesen. Eine 3D-Engine stellt geeignete Parser und Funktionen bereit, mit

denen Daten schnell und bequem ausgelesen und in das Programm eingebunden werden können.
Die Engines dienen u.a. zum sachgerechten Verabreiten der 3D-Grafiken. Diese setzten sich in der Regel aus sog „3D-Primitiven“ zusammen. Dies sind einfache geometrische Figuren, die zunächst als „Drahtgitter“ vorliegen und aus denen sich schließlich dann komplexe Gebilde zusammensetzten können. Eine 3D-Primitive ist also eine Menge an Vertexe, die ein Objekt bilden (ein Vertex ist ein einzelner Punkt im Raum). Die einfachste Primitive ist ein einzelner Punkt. Primitive können aber auch Dreiecke bzw. Dreiecksaneinanderreihungen sein. Oftmals sind Primitive auch Polygone. In der Computergrafik sind Polygone geschlossene 3D-Figuren – wiedergegeben durch mindestens drei Vertexe. Flächen, umgrenzt von geschlossenen Linien, werden dabei verwendet, um räumliche Elemente zu beschreiben. Die Repräsentation erfolgt in Vektorform.

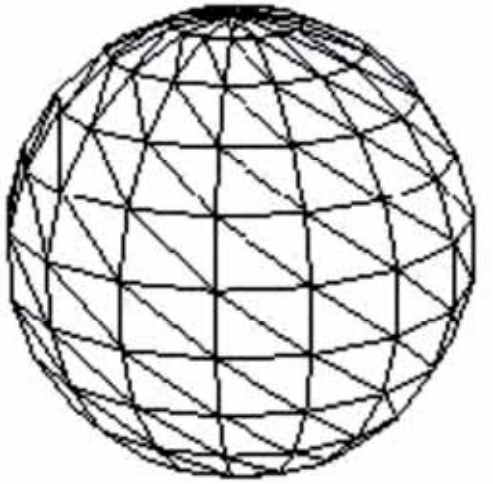


Abb. 6.2 Polygon aus Primitiven „Dreieck“ gebildet[21]

Mit Hilfe spezieller Grafikprogramme kann ein Polygon aus beliebigen einzelnen Eckpunkten und Kanten/Segmenten zusammengefügt werden. Sobald ein Linienzug zu einem einfachen Polygon geschlossen wird, verschwinden alle übrig gebliebenen Punkte und Segmente und nur das einfache Polygon bleibt bestehen. Das einfachste Polygon ist ein Dreieck. Direct3D benutzt Dreiecke, um Polygone zu bilden, weil alle drei Vertexe in einem Dreieck garantieren, dass sie eben ein Dreieck bilden. Alle Dreiecke zusammengefasst formen dann ihrerseits ein Primitiv wie die Kugel aus Abb. 6.2, deren Oberflächen mit Texturen und Materialien belegt werden können.
Wie gut ein Computerspiel ist, bemisst sich zu einem großen Teil nach der Anzahl der gleichzeitig darstellbaren Bildpunkte, Farben, bewegten Flächen und Lichtreflexe. Je mehr solche Attribute ohne sichtbare Zeitverzögerung für jede Perspektivänderung zu berechnen sind, desto räumlicher und interessanter wird

[21] Quelle: ibid.

ein Spiel. Voraussetzung für eine wirklichkeitsnahe Wiedergabe am Bildschirm ist ein schneller Prozessor. Denn je schneller der Prozessor, desto mehr Polygone können berechnet werden. Die Playstation2 zum Beispiel kann theoretisch 70 Millionen Polygone pro Sekunde verarbeiten.
Jeder Vertex kann über eine Normale verfügen (siehe Abb. 6.3). Eine Normale ist ein Vektor, der angibt, wie ein Vertex ausgerichtet ist. Dies ist notwendig, wenn Lichtquellen mit ins Spiel kommen. Dann kann anhand der Ausrichtung der Normalen zur Lichtquelle ermittelt werden, wie stark ein Vertex bzw. die damit verbundene Fläche angestrahlt wird.

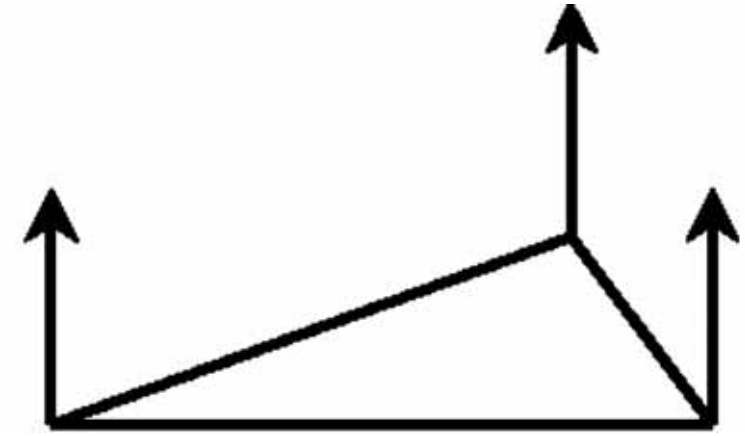


Abb. 6.3 Normale auf Dreieck

Normalen müssen nicht unbedingt orthogonal zur Fläche verlaufen. Wenn Sie in eine nicht orthogonale Richtung zur Dreiecksfläche zeigen, lassen sich Wölbungen vortäuschen. Die beiden unteren Kugeln in Abb. 6.4 demonstrieren den Unterschied.

Abb. 6.4 Normale auf Kugeln[22]

[22] Quelle: ibid.

Die linke Kugel in Abb. 6.4 benutzt keine Normal-Vektoren. Direct3D rendert daher jede Fläche mit einer ganzheitlichen Schattierung. Unter geschickter Benutzung von Normal-Vektoren kann man jedoch diese Schattierung interpolieren lassen, so dass man den Eindruck erhält, die Fläche wäre gewölbt.
Auch die Reihenfolge der Verbindung der Vertexe kann ausschlaggebend für das Aussehen der damit erzeugten Figur sein.
Direct3D benutzt Matrizen, um Transformationen bzw. Umformungen durchführen zu können. Es gibt drei Arten von Umformungen: Rotation, Translation und Skalierung. Man kann dabei jeden Punkt in einen anderen transformieren, wenn man eine 4x4-Matrix verwendet. Im folgenden Beispiel wird eine Matrix dazu benutzt, aus einem Punkt (x,y,z) einen anderen (x‘,y‘,z‘) zu machen:

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix}$$

Führt man die folgenden Operationen an (x,y,z) und der Matrix durch, so erhält man den Punkt (x‘,y‘,z‘):

$$x' = (x \times M_{11}) + (y \times M_{21}) + (z \times M_{31}) + (1 \times M_{41})$$

$$y' = (x \times M_{12}) + (y \times M_{22}) + (z \times M_{32}) + (1 \times M_{42})$$

$$z' = (x \times M_{13}) + (y \times M_{23}) + (z \times M_{33}) + (1 \times M_{43})$$

Die Matrizenberechnung muss nicht selbst programmiert werden, sondern man kann sich meistens geeigneter Libraries bedienen, wie z.B. des Moduls *math.bas*, welches in DirectX enthalten ist.
Auch kann jedes dreidimensionale Objekt um die X-, Y-und Z-Achse rotiert werden mit Methoden wie *RotateXMatrix*, *RotateYMatrix* und *RotateZMatrix*. Auch für die Translation und für die Skalierung gibt es passende Methoden.
Benutzt man ein 3D-Programm, besteht die Möglichkeit, die erstellten 3D-Modelle in sein Spiel direkt zu integrieren. Es ist jedoch auch möglich, die aus 3D-Modellen berechneten Bilder als zweidimensionale Grafiken zu verwenden, was beispielsweise bei Spielen wie *Age of Empires* und *Command and Conquer* realisiert wurde. Inzwischen geht der Trend langfristig zu echter 3D-Grafik über. Das heißt, dass die Spielegrafik in Echtzeit in drei Dimensionen berechnet und dann auf den Monitor (2D) gebracht wird. In Direct3D besteht die Möglichkeit, *.3Ds Dateien in so genannte *.X Dateien zu konvertieren. Diese *.X Dateien enthalten alle Informationen über die Objekte, welche mit 3D-Anwendungssoftware erstellt wurden. Diese .X Dateien basieren auf Vorlagen (Templates), die als Abschnitte der Datei vorstellbar sind und einen bestimmten

Teil des 3D-Modells beschreiben. Das Dateiformat der X-Dateien kann als Text oder binär codiert vorliegen.

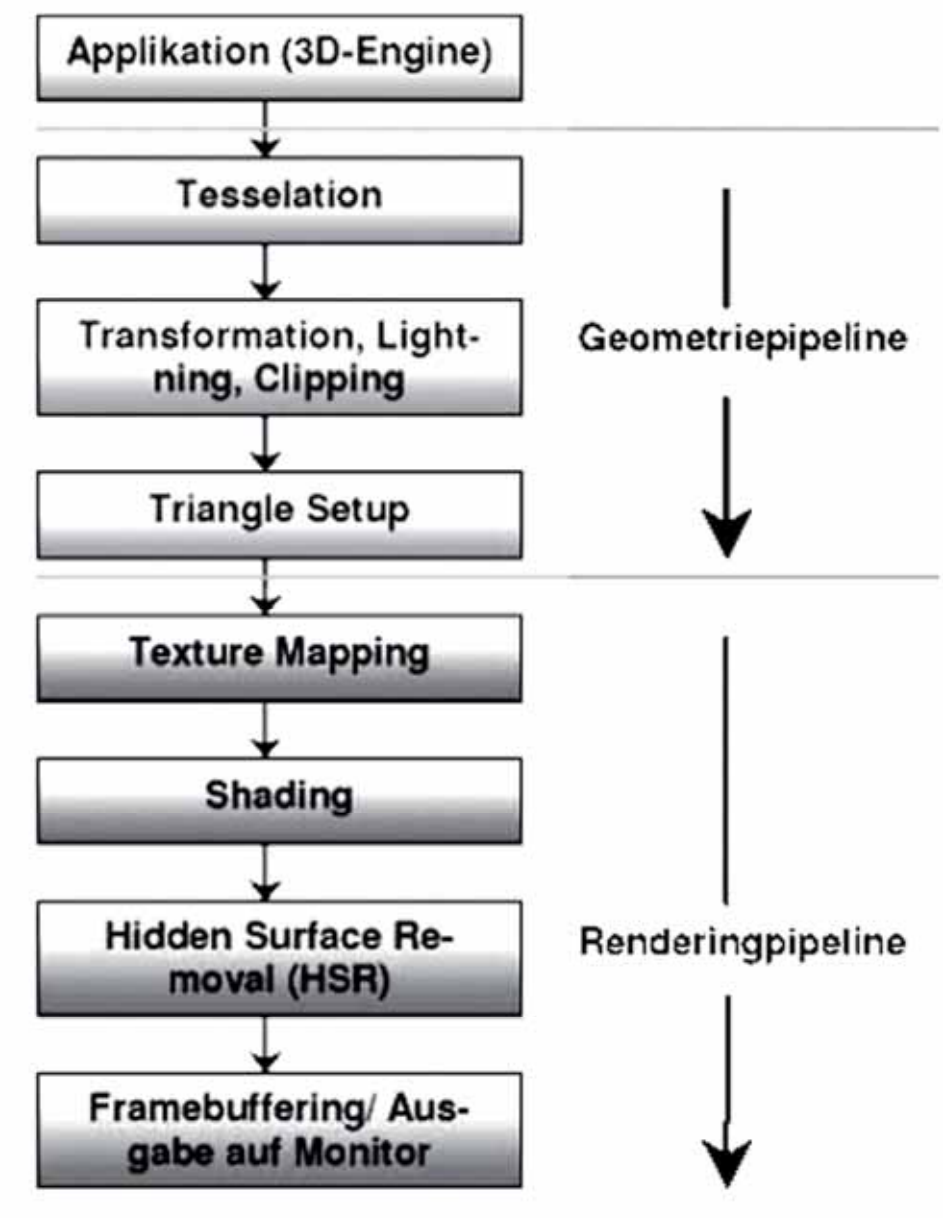


Abb. 6.5 3D-Pipeline[23]

Eine „3D-Pipeline“ erzeugt aus einer dreidimensionalen Umgebung eine zweidimensionale Darstellung und regelt die Zusammenarbeit zwischen CPU und Grafiksystem. Nachdem im Rechner 3D-Objekte definiert wurden, liegen diese als Drahtmodell im Speicher des Rechners vor. Nun müssen weitere Schritte durchgeführt werden, damit diese Modelle am Bildschirm erscheinen können. Die Software und Hardware, die 3D-Modelle verarbeiten und auf den Bildschirm übertragen, werden 3D-Pipeline genannt. Diese Pipeline kann man sich

[23] Quelle: ibid.

als Leitung vorstellen, durch die alle Modelle fließen müssen um zum Bildschirm zu gelangen. Die Manipulation und Darstellung eines 3D-Bildes erfordert eine große Menge mathematischer Berechnungen, damit alle Details der Szene richtig dargestellt werden. Für diese blitzschnellen Berechnungen ist die Pipeline verantwortlich. Sehen wir uns diese einzelnen Pipelines nun genauer an.
Wie in Abb. 6.5 zu sehen, lässt sich die 3D-Pipeline grob in drei Abschnitte unterteilen: Den Applikations-Bereich, die folgende Geometriepipeline und die Renderingpipeline mit anschließender Ausgabe des Bildes auf den Monitor. Diese Abschnitte lassen sich wiederum in weitere Schritte unterteilen, auf die ich im Folgenden genauer eingehen werde. Bevor mit dem Erscheinen des *Voodoo1-Chips* die dreidimensionale Revolution begann, wurden all diese Schritte in Software erledigt. Da 3D-Berechnungen extrem rechenintensiv sind, wurden Teile der Pipeline auf speziell dafür ausgelegte Chips verlegt.
Zu Beginn eines Renderingvorgangs steht immer die Applikation (in den meisten Fällen ein Spiel), die man -neben dem User -auch als oberste Instanz betrachten kann: Die so genannte Spielengine reagiert auf Eingabebefehle des Benutzers, indem sie diese zuerst auswertet (z.B. „linke Cursortaste gedrückt = Spieler dreht sich um die eigene Achse nach links“ usw.) und dann der Hardware die entsprechenden Befehle erteilt (z.B. „Spieler dreht sich um die eigene Achse nach links = drehe die Kamera um die eigene Achse nach links“ usw.). Zusätzlich gibt die Engine noch andere Anweisungen an die Hardware weiter, die dem Benutzer meist verborgen bleiben (z.B. „Objekt X befindet sich sehr weit vom Betrachter entfernt = Setze Detailgrad für Objekt X auf niedrig um Rechenzeit zu sparen“ usw.). Bereiche wie Physikberechnungen oder die Kollisionsabfrage werden ebenfalls von der Engine gesteuert.

Die Geometriepipeline
Bis das fertige Bild den Monitor erreicht, ist es ein langer Weg. Als Erstes muss die Geometrie der 3D-Szene berechnet werden. Im folgenden Beispiel wird alles bis zum Lighting von der CPU berechnet – die restliche Arbeit erledigt dann der Grafikchip. Dieses Beispiel stellt allerdings nur eine Möglichkeit dar. Je nach Engine und Grafikkarte sieht die 3DPipeline etwas anders aus, so kann beispielsweise das Clipping an einer anderen Position erledigt werden etc.

Tesselation
Zu Beginn ist die gesamte 3D-Landschaft in einzelnen Scheitelpunkt -Vertexen dargestellt. Während der Tesselation werden nun die einzelnen Punkte zu Polygonen verbunden. Alle aktuellen 3D-Beschleuniger arbeiten hierbei mit Dreiecken (Triangles) als geometrische Grundfigur. Diese Zerlegung der Szene in

Dreiecke vermindert die Datenmengen und somit auch den Rechenaufwand extrem.

Transformation

Der Vorgang der Verteilung der Modelle im Raum in ihrer richtigen Stellung zueinander wird auch Transformation genannt. Transformation ist die Manipulation der Koordinaten in einem Modell, um deren Position und Ausrichtung zu bestimmen. Die erste Transformation ist die »Welt-Transformation«, womit das Objekt im 3D-Raum positioniert wird. Jedes Objekt wird dabei typischerweise einer anderen Transformation unterzogen, um dessen Platz in der Szene zu identifizieren. In der folgenden „Ansichts-Transformation“ wird die Position der Objekte relativ zum Betrachter bestimmt. Schließlich werden die Objekte einer dritten Transformation unterworfen, der „Perspektiven-Transformation“. Sie ist verantwortlich für die Formung der Szene in eine gewünschte Perspektive (ähnlich einer Kameralinse). Durch die Auswahl verschiedener Perspektiven-Transformationen kann das Sichtfeld des Betrachters und dessen Tiefeneindruck verändert werden. Man stelle sich z.B. eine kleine Szenerie mit drei Objekten vor: Ein Monitor, eine Teekanne und eine Tischlampe. Von jedem dieser Objekte existiert eine genaue Beschreibung in einer bestimmten Größe und Position. Zuerst müssen sie eine relativ zur Position des Betrachters korrekte Größe erhalten (also: Teekanne < Lampe < Monitor) und in die richtige Position gebracht werden (z.B. Teekanne steht vor dem Monitor, Lampe steht neben dem Monitor usw.). Dies wird durch grundlegende mathematische Operationen wie Transformation (=Verschiebung), Skalierung (=Vergrößerung bzw. Verkleinerung) und Drehung der Objekte erreicht. Sind diese Schritte für alle Objekte der Szenerie durchgeführt, hat man eine perspektivisch korrekte 3D-Welt.
In einem zweiten Schritt wird dann die 3D-Welt auf eine zweidimensionale Fläche projiziert – hier ein virtuelles Abbild der Monitorebene. Hierbei wird für jeden Vertex in der 3D-Welt (mit X,Y und Z-Koordinate) eine passende (X,Y)-Koordinate auf der noch virtuellen Monitorfläche berechnet und zugewiesen. Bei der darauf folgenden Ansichts-Transformation werden diese Werte noch für die vom Benutzer gewählte Auflösung angepasst (z.B. 924x768). Da sich die Positionen der Objekte bei jeder noch so kleinen Bewegung verändern und sie somit neu ausgerichtet sowie anschließend wieder auf die Monitorebene projiziert werden müssen, wird die Transformation für jedes einzelne Frame neu durchgeführt.

Lighting

Um einen größeren Realismus in die Szene zu bringen, können die Objekte von Lichtquellen in der 3D-Welt beleuchtet werden. Im Gegensatz zur relativ schnellen Ausführung der Transformation durch die CPU, ist es beim Lighting offen-

sichtlich komplizierter – Beleuchtung auf Geometrieebene ist extrem rechenaufwendig und selbst Grafikkarten mit Lighting-Unterstützung in Hardware haben mit mehreren Lichtquellen ihre Mühe. Das einzige Geometrie-Lighting, das momentan im großen Stil genutzt wird, ist ambient Lighting, bei dem unabhängig von Art und Entfernung einer Lichtquelle jedem Vertex ein Ligthing-Wert zugewiesen wird.

Clipping

Bevor die Engine die Geometriedaten an die Grafikkarte zum Rendern schickt, muss als Nächstes bestimmt werden, was nun im sichtbaren Bereich des Bildschirms liegt. Für jedes Objekt wird dabei geprüft, ob es außerhalb oder innerhalb des Bildschirms liegt. Objekte, die teilweise im sichtbaren Bereich liegen, werden an den Bildschirmrändern abgeschnitten, ein Vorgang, der Clipping genannt wird. Das Blickfeld muss man sich dabei als eine Pyramide vorstellen, die sich von der Monitorebene weg nach „hinten“ öffnet. Alles, was sich nicht innerhalb dieser Pyramide befindet, wird bei den folgenden Berechnungen nicht mehr berücksichtigt. Die zu berechnende Datenmenge verringert sich dadurch enorm.

Triangle Setup

Bevor die Geometrie nun dem Rasterizer übergeben werden kann, folgt noch ein letzter wichtiger Schritt: Das Triangle Setup. Dies gehört zwar genau genommen immer noch zur Geometriepipeline, wurde jedoch bereits in den Anfängen der 3D-Beschleunigung teilweise oder ganz auf die PC-Grafikchips verlegt. Nach der Transformation und dem Lighting existieren immer noch Eckpunkte (Vertixe) mit x-, y-und z-Koordinaten. Nun muss diese Szene für die zweidimensionale Darstellung auf dem Bildschirm vorbereitet werden. Die zweidimensionale Welt besteht letztendlich nur aus Pixel. Deren Anzahl wird von der eingestellten Auflösung (z.B. 924 x 768) festgelegt. Im Triangle Setup werden die einzelnen Polygone für das anschließende Rendering vorbereitet: Damit der Rasterizer nämlich irgendwas zeichnen kann, muss er für jede Bildzeile sowohl Anfangs- als auch Endpunkt des zu zeichnenden Dreiecks kennen. Die Triange-Setup-Prozedur verbindet nun die vorliegenden Eckpunkte der Polygone bzw. Dreiecke und füllt diese mit Pixel. Die Triangle Setup Unit verarbeitet der Reihe nach jedes Dreieck. Diese bestehen aus drei Eckpunkten, die selbst durch eine x-, y- und z-Koordinate gekennzeichnet sind. x und y stellen die spätere Position auf dem Bildschirm dar, während die z-Koordinate die Tiefeninformation erhält. Jedes der innerhalb eines Dreiecks liegenden Pixel bekommt eine Lichtinformation zugewiesen und wird dann an die Pixel-Rendering-Einheit geschickt.

Die Renderingpipeline
Das Rendering ist der rechenaufwendigste Teil der Pipeline und daher die Aufgabe spezieller Grafikkarten. Beim Rendern wird nun die Szene, die bis jetzt genau genommen nur in gigantischen Zahlenkolonnen existierte, als Bild berechnet, und am Ende auf den Monitor ausgegeben. Da aber ein Rendern der in der Geometrie-Pipeline berechneten Daten nur ein Drahtgittermodell zeigen würde, wird die Szene zuvor noch mit diversen Mitteln verschönert. Auch hier kann die Art und Reihenfolge, die in den folgenden Schritten absolviert werden, von Engine zu Engine und Grafikkarte zu Grafikkarte leicht variieren.

Rasterisierung
Die Rasterisierung ist für das eigentliche Zeichnen (Rendern) der Primitiven (Dreiecke, Punkte, Linien) zuständig. Sie ist der letzte Schritt in der Darstellungs-Pipeline. Hierbei werden 3D-Oberflächen in Pixel auf dem Bildschirm umgewandelt, nachdem sie transformiert, geclippt, beleuchtet und texturiert wurden.

Texture Mapping
Ein großer Schritt zur Verschönerung der 3D-Szene stellt das Texture Mapping dar. Hierbei wird eine Textur (= zweidimensionales, quadratisches Bild) auf ein Polygon „gepappt“. Dies lässt die gesamte Szenerie mit einem Schlag realistischer wirken. Hierfür existieren verschiedene Arten von Texturen. Der wohl am meisten verwendete Texturing-Effekt ist das Lightmapping. Dabei wird über die Basistextur ein weiteres Bild als Lightmap gelegt, mit welchem auf der Basistextur die Intensität der einzelnen Texturpunkte bestimmt wird. Damit wird eine realistische Schattierung gewährleistet. In den meisten Fällen werden dabei schwarz/weiss-Texturen in relativ kleinen Auflösungen verwendet, wodurch die Texturgröße klein bleibt. Mit Lightmaps kann eine 3D-Szene ohne großen Rechenaufwand verschönert werden.

Abb. 6.6 Texture-Blending[24]

[24] Quelle: ibid.

Andere Maps sorgen für andere Effekte: BumpMaps verleihen der eigentlich flachen Textur eine Struktur, ohne dabei die Geometrie zu verändern, EnviromentMaps sorgen für realistische Spiegelungen.

Shading

Die Flächen sind texturiert, aber nun gibt es noch ein Problem: Die verschiedenen Polygone machen auf den Betrachter einen stark voneinander abgegrenzten Eindruck und wollen nicht so recht zu einem Ganzen verschmelzen. Die Ursache liegt darin, dass jedes Polygon nur einen einzigen Helligkeitswert besitzt und daher die Übergänge bei zwei Polygonen mit verschiedener Helligkeit sehr gut zu erkennen sind. Um die Übergänge unkenntlich zu machen, muss ein adäquates Shading-Verfahren her: Das simpelste ist das Flat-Shading (nur ein Helligkeitswert pro Polygon); es wird jedoch in heutigen Spielen nicht mehr verwendet. Stattdessen setzen alle Spiele mittlerweile auf das Gouraud-Shading, das derzeit den besten Kompromiss zwischen Qualität und Geschwindigkeit bietet. Beim Gouraud-Shading kommen endlich die bereits zu Beginn durch das Ambient-Lighting zugewiesenen Farbwerte der Vertexe zum Zuge. Diese werden über das ganze Polygon interpoliert, wodurch die Helligkeitsübergänge etwas verwischt werden und folglich nicht mehr so stark ins Auge stechen. Die Qualität ist dem des Flat-Shadings klar überlegen.

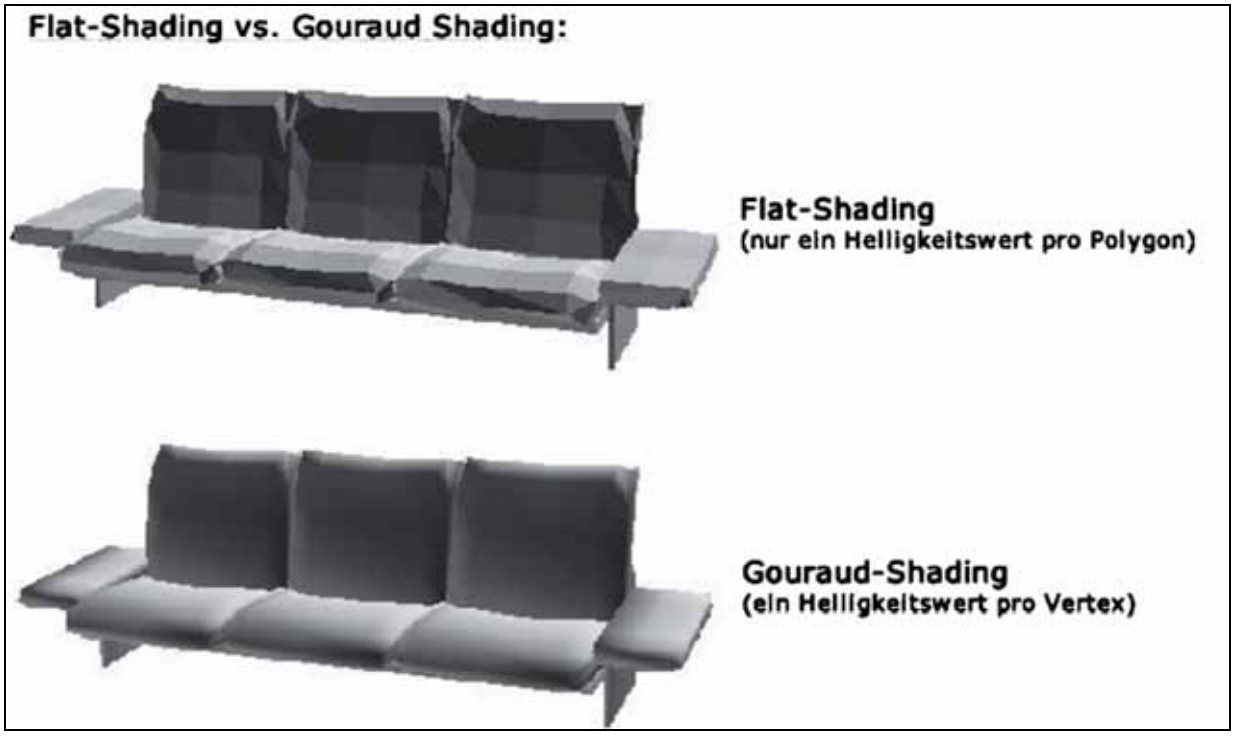


Abb. 6.7 Shading[25]

[25] Quelle: ibid.

Hidden Surface Removal

Nun sind die Polygone texturiert und gefiltert und das Shading sorgt für weiche Übergänge. Bevor das Bild jedoch fertiggestellt werden kann, muss noch das sog. *Hidden Surface Removal* (HSR) durchgeführt werden. Wie der Name bereits zu erkennen gibt („Entfernung verborgener Oberflächen"), wird beim HSR dafür gesorgt, dass verdeckte Bereiche vor der Fertigstellung des Bildes entfernt werden. Dies wird mit einem so genannten Z-Buffer realisiert. In diesem Teil des Grafikspeichers werden die Tiefenwerte (Z-Werte) der gerenderten Pixel abgelegt. Ein Beispiel: Pixel A befindet sich an der Bildschirmkoordinate (59 / 200) und hat einen Tiefenwert von 450. Es wird gerendert und der berechnete Farbwert im Grafikspeicher abgelegt, zudem wird der Z-Wert (450) im Z-Buffer gespeichert. Kurz darauf soll für das gleiche Bild ein weiterer Pixel mit Koordinate (59 / 200) gerendert werden. Dies wird erledigt, und danach wird der Z-Wert des neuen Pixels mit dem im Z-Buffer gespeicherten Wert des „alten" Pixels verglichen. Beträgt der Z-Wert des neuen Pixels z.B. 700, befindet sich das Pixel tiefer im Bild als das alte Pixel mit Z-Wert 450, wird somit von Letzterem verdeckt und ist für den User unsichtbar. Das soeben gerenderte Pixel wird in diesem Falle gleich wieder gelöscht. Beträgt der Z-Wert aber z.B. 200, befindet sich das neue Pixel vor dem alten. In diesem Fall wird das nun verdeckte, alte Pixel durch das Neue ersetzt. Auch der Z-Buffer Eintrag wird durch den Z-Wert des neuen Pixels ersetzt, da dieses nun als Referenzpunkt für die Z-Checks an der Koordinate (59 / 200) gilt. Der Z-Buffer muss sehr genau sein (mindestens 24 Bit, besser 32 Bit), weil bei zu ungenauen Z-Buffer-Werten es schnell vorkommen kann, dass eigentlich verdeckte Objekte plötzlich für kurze Zeit sichtbar werden.

Framebuffer

Die fertig berechneten Pixel werden nun zum Schluss in den Framebuffer (ein Teil des Grafikspeichers) geschrieben. Dieser ist unterteilt in Back- und Frontbuffer. Das Bild wird zuerst im Backbuffer aufgebaut. Befindet sich jedes Pixel eines Frames darin, wird dieses mittels Flipping in den Frontbuffer geschrieben, der das Bild entweder an den RAMDAC oder bei LCD-Monitoren unmittelbar an den digitalen DVI-Ausgang sendet. Der RAMDAC (Random Access Memory Digital/Analog-Converter) wandelt die digitalen Daten in analoge Signale um, die über den VGA/TV-Ausgang an den Monitor/TV wandern, wo schließlich das fertige Bild dargestellt wird. Ist das nächste Bild fertig gerendert, wird nochmals das Flipping durchgeführt: Im Frontbuffer wird das alte Bild durch das neue ersetzt und der Backbuffer für das nächste Bild bereitgestellt usw.; Frame-Buffering ist insofern wichtig, da die benötigte Zeit für die Berechnung eines Bildes stark variieren kann (je nach Komplexität der Szenerie). Durch ein Zwischenspeichern im Framebuffer wird eine mehr oder weniger

stabile Bildwiederholrate garantiert. Damit der User das Geschehen auf dem Monitor als flüssig wahrnimmt, müssen mindestens 30 Bilder in der Sekunde über den Monitor laufen, ideal wären jedoch 60 Bilder.

VRML

VRML[26] bedeutet Virtual Reality Modeling Language und ist eine Beschreibungssprache für 3D-Szenen, deren Geometrien, Ausleuchtungen, Animationen und Interaktionsmöglichkeiten. VRML wurde ursprünglich als 3D-Standard für das Internet entwickelt. Die meisten 3D-Modellierungswerkzeuge ermöglichen den Im- und Export von VRML-Dateien, wodurch sich das Dateiformat auch als ein Austauschformat von 3D-Modellen etabliert hat.

Eine VRML-Darstellung (zum Beispiel innerhalb eines Web-Browsers oder einer virtuellen Realität) wird vom Computer des Betrachters in Echtzeit generiert. Das bedeutet, dass der Computer jedes einzelne Bild aus den vorhandenen Geometriedaten, sowie dem Verhalten und den Bewegungen des „Besuchers" ständig neu berechnet. Aus diesem Grund scheiden (ohne Einsatz von Supercomputertechnologien) fotorealistische Darstellungen mit rechenaufwendigen Raytracing-Verfahren, „echten" Spiegelungen und Schattenwurf weitgehend aus. Es werden auch beim Benutzen von vordefinierten Betrachterpositionen (viewpoints) beim Wechsel zwischen diesen Punkten und bei Kamerafahrten keine fertigen Bilder aus Filmsequenzen abgespielt. Komplexe VRML-Szenen stellen so unter Umständen hohe Anforderungen an die Hardware. Wie schnell, beziehungsweise wie flüssig die Bewegungen erfolgen, hängt vom Prozessor und vor allem von der Grafikkarte des wiedergebenden Computers ab.

VRML-Dateien erkennt man an der Dateierweiterung „.wrl" (world), sie sind im Klartext (ASCII bzw. UTF-8) geschrieben und können auch in einem einfachen Texteditor erstellt werden. Es finden sich auch mit Gzip verpackte VRML-Dateien unter der Dateierweiterung „.wrl", obwohl dafür eigentlich die Dateierweiterung „.wrz" vorgesehen ist.

Da sich eine VRML-Szene aus mehreren Knoten zusammensetzt, hier einige wichtige Knotentypen in VRML:

- Für Geometriegrundkörper wie Quader, Zylinder, Kegel und Kugel sind jeweils eigene Knotentypen vorhanden.
- Komplizierte Grafikobjekte bauen auf einer Liste aus Punkten und damit beschriebenen Flächen (IndexedFaceSet), Linien (IndexedLineSet) oder Punkte (PointSet) auf.
- Die Körper können hierarchisch durch Transform-Knoten zusammengefasst werden. Auf alle Knoten unterhalb dieses Knotens können Transformationsoperationen wie Skalierung, Rotation oder Translation angewendet wer-

[26] siehe http://de.wikipedia.org/wiki/VRML

den. Durch die Baumstruktur in der VRML-Datei ist es leicht, eine vorwärts gerichtete Kinematik zu erzeugen. So bewegt sich dann ein dargestellter Finger mit, wenn der Arm bewegt wird.

- Materialeigenschaften können den geometrischen Körpern zugeordnet werden. So sind mit Hilfe von PNG-Bildern auch transparente Texturen möglich.
- Die Lichtquellen sorgen dann durch das Beleuchtungsmodell (zumeist Gouraud Shading) für die entsprechende Schattierung der Objekte.
- Sensoren reagieren auf Benutzeraktionen und der Time-Sensor dient für Animationen.
- Interpolatoren können dann z. B. eine Rotation in einen beliebigen Farbwechsel umwandeln oder mit fortschreitender Zeit ändert sich die Lage eines Objektes
- Der Skriptknoten wird aktiviert über Verbindungen (Route) durch definierte Ereignis-Ausgänge von Objekten und es wird ein Java-Script oder Java-Programm gestartet. Dieses kann beliebige Berechnungen durchführen und die Ergebnisse durch weitere Verbindungen an die Eingänge von Objekten liefern.
- Der USE-Befehl dient zum Wiederverwenden von schon mittels „DEF“ definierten Skriptknoten.
- Der PROTO-Knoten ist wesentlich flexibler als der USE-Befehl und ermöglicht z. B. die Schaffung eines Torus-Geometrieknotens, welcher laut Standard eigentlich nicht definiert ist. Etliche Protos sind im Internet frei zugänglich.
- Durch sogenannte Anker und Inline kann man durch das Anklicken von Objekten in eine andere Welt gelangen oder andere VRML-Objekte in die eigene Welt mit einbauen. Dieses ist hilfreich um den VRML-Text übersichtlich zu halten.
- LOD (Level of Detail) ermöglichen die vereinfachte Darstellung, wenn sich der Benutzer in großer Entfernung befindet, um die Performance zu erhöhen.
- Billboards sind „Tafeln“, die dem Benutzer immer ihre Breitseite zudrehen.
- Zusätzliche Knoten beschreiben die Schrittgeschwindigkeit und Augenhöhe des Nutzers und auch die Hintergrundfarbe der Welt.

Wie beim Skript-Knoten schon erwähnt, besitzt ein VRML Viewer eine integrierte ereignisorientierte Simulation, d. h. jedes Objekt kann ein Ereignis aussenden. Dabei handelt es sich um einzelne Werte oder ganze Listen von Werten. Diese Werte können Zeiten, Zahlen, Zeichenketten, Farben, Vektoren, Bilder oder ganze Knoten sein. Sie werden dann vom System weiterverarbeitet und ermöglichen somit sogar die Simulation von einfachen physikalischen Vorgängen.

Die Kollisionserkennung des VRML Browsers gehört zum Standard. Eine Kollisionserkennung wird benötigt, damit man nicht durch Wände läuft.
Zur externen Steuerung der VRML-Szenen durch den Browser kann die Programmiersprache Java über die EAI-Schnittstelle (External Authoring Interface) nach ISO/IEC 14772-2 verwendet werden. Wie jede Textdatei kann VRML auch durch serverseitige Skriptsprachen (z. B. PHP, Perl, Python) vom Server erzeugt werden. Neben der Anwendung im Browser wird VRML auch in Umgebungen virtueller Realität eingesetzt.
Es kann hier nicht auf die gesamte Syntax von VRML eingegangen werden. Es ist aber so, dass mit relativ wenig „selbstsprechenden“ Anweisungen beeindruckende Animationen programmiert werden können, in denen sich der User interaktiv bewegen kann. Hier als Beispiel ein einfaches VRML-Programm:

```
#VRML V2.0 utf8
# Landschaft mit Burg
Background {
  skyColor [ 0.0 0.7 0.8 , 0.0 0.5 0.8 , 0.0 0.3 0.8 ]
  skyAngle [ 0.785 , 1.571]
  groundColor [ 0.0 0.8 0.2 , 0.0 0.7 0.3 , 0.0 0.5 0.4 ]
  groundAngle [ 0.785 , 1.571 ]
}
Transform {
  children [
    DEF Turm Shape {
      appearance Appearance {
        material Material {
          diffuseColor 1.0 1.0 0.0  #gelb, beleuchtet
        }
      }
      geometry Cylinder {
        radius 1.0
        height 3.0
      }
    }
  ]
  translation -3.0 0.5 0.0
}
Transform {
  children [
    DEF Dach Shape {
      appearance Appearance {
        material Material {
          diffuseColor 1.0 0.0 0.0  #rot, beleuchtet
        }
      }
      geometry Cone {}
    }
  ]
```

```
  translation -3.0 3.0 0.0
}
DEF Mauer Shape {
  appearance Appearance {
    material Material {
      diffuseColor 0.4 0.25 0.1 #braun, beleuchtet
    }
  }
  geometry Box {
    size 4.1 2.0 0.5
  }
}
Transform {
  children [USE Turm]
  translation 3.0 0.5 0.0
}
Transform {
  children [USE Dach]
  translation 3.0 3.0 0.0
}
Transform {
  children [USE Turm]
  translation 3.0 0.5 -6.0
}
Transform {
  children [USE Dach]
  translation 3.0 3.0 -6.0
}
Transform {
  children [USE Turm]
  translation -3.0 0.5 -6.0
}
Transform {
  children [USE Dach]
  translation -3.0 3.0 -6.0
}
Transform {
  children [USE Mauer]
  translation -3.0 0.0 -3.0
  rotation 0.0 1.0 0.0 1.571
}
Transform {
  children [USE Mauer]
  translation 0.0 0.0 -6.0
}
Transform {
  children [USE Mauer]
  translation 3.0 0.0 -3.0
  rotation 0.0 1.0 0.0 1.571
}
```

Die Befehle sind relativ selbstsprechend, so dass hier nicht näher darauf eingegangen wird. Das Ergebnis dieses einfachen Programms ist eine Burg mit Türmen und Dächern, und der User kann mit der Maus leicht in 3D-Manier die Burg von allen Seiten betrachten, zoomen und so weiter (vgl. Abb. 6.8).

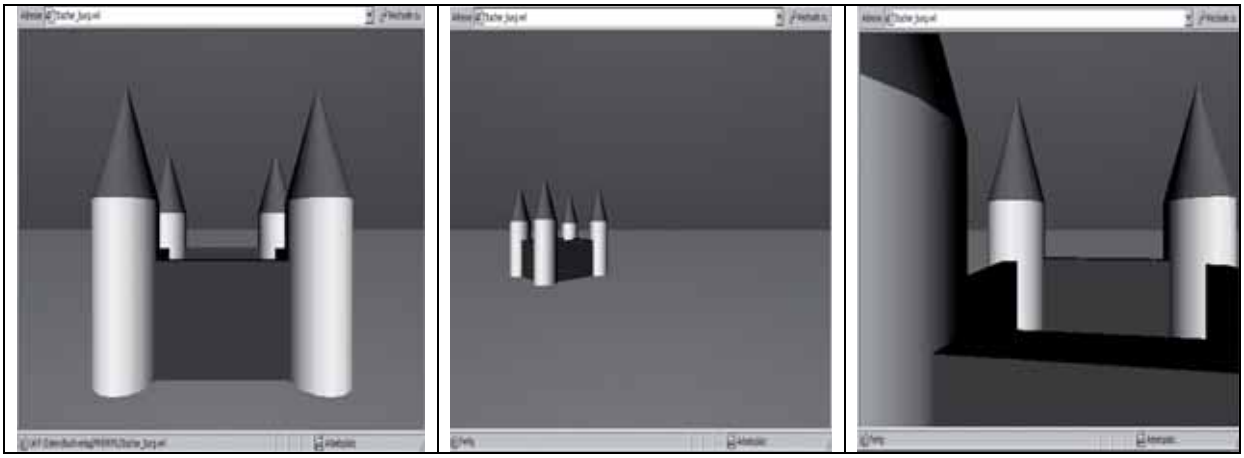


Abb. 6.8 Navigation in einem VRML-Programm

Morphing

Morphing[27] ist ein computergenerierter Spezialeffekt bei Ton- oder Bildaufzeichnungen. Beim Morphing werden zwischen zwei Einzelbildern bzw. zwei Klängen Zwischenübergänge berechnet. Im Gegensatz zur Überblendung (Film) wird beim Morphing ein Bild in ein anderes Bild durch Einsatz von zusätzlichen gezielten Verzerrungen überführt. Dabei versucht man, ausgehend von einem Quellbild, einen möglichst realistischen Übergang zu einem Zielbild zu erzeugen. Der typische Morphing-Prozess besteht deshalb darin, markante Bildelemente (z.B. Gesichtszüge wie Mund und Augen oder Objektränder) in Quell- und Zielbild auszuwählen und so zu verzerren, dass ihre Konturen zur Übereinstimmung gebracht werden können. Um möglichst realitätsnahe Effekte zu erzielen ist es wichtig, dass sich Quelle und Zielbild nicht zu sehr voneinander unterscheiden, beispielsweise ist es leichter, ein menschliches Gesicht in ein anderes menschliches Gesicht umzuwandeln als ein Gesicht in ein Bügeleisen.
Als Film abgespielt, erwecken diese Bilder den Eindruck einer stetigen Transformation. Morphing wird hauptsächlich in der Filmindustrie eingesetzt. In der Vergangenheit mussten Übergänge von Filmen durch oftmaliges Aufbauen und Fotografieren einer Szene (Slow-Motion Capturing) in Verbindung mit aufwendigen Filmschnitten erzeugt werden. Durch die ständige Steigerung der Rechenleistung ist man in der Filmindustrie immer mehr dazu übergegangen, Filme mit digitaler Technik (Computern und digitalen Bildverarbeitungsprogrammen) zu entwickeln. Übergänge von Bildern werden dadurch nicht mehr mit Slow-

[27] siehe http://de.wikipedia.org/wiki/Morphing

Motion Verfahren sondern mit Morphing realisiert. In den frühen Tagen des Morphing wurden einfache und wenig realistisch wirkende Effekte, wie Überblendung (langsames Überblenden der RGB-Werte von Ursprungsbild zum Zielbild) und Fading (langsames Aus- und Einblenden der RGB-Werte) verwendet.

Abb. 6.9 Morphing mit einem Zwischenschritt[28]

Weitere Anwendungsgebiete des Bilder-Morphings außerhalb der Filmindustrie sind beispielsweise der Einsatz in der Biologie und Chemie um Modelle zu verändern, oder in der Kriminalistik um nach vermissten Personen zu suchen. Bei Letzterem werden Fotos von diesen Personen durch Morphing verändert um das Aussehen dieser Menschen ihrem Alter entsprechend anzupassen.
Beim Morphing von Klängen werden ebenfalls Zwischenschritte berechnet, so dass sich der Ausgangsklang langsam über die neu generierten Zwischenklänge in den neuen Klang verändert.
Ein komplexer Morphing-Vorgang besteht aus drei Teilschritten: *Warping*, *Tweening* und *Cross-Dissolving*.
Unter *Warping* versteht man das Drehen und Verzerren (Strecken bzw. Dehnen) eines Bildes. Dabei wird jeder Position eines Punktes im Quellbild eine neue Position zugeordnet. Diese neue Position ist abhängig von den bereits oben erwähnten ausgewählten, markanten Bildelementen, die später als Referenzlinienpaare bezeichnet werden.
Der zweite Teil des Morphingalgorithmus benutzt das *Tweening*. Dabei handelt es sich um eine einfache lineare Interpolation, welche die Position jedes Punktes

[28] Quelle: http://de.wikipedia.org/wiki/Bild:Bush-Arnie-morph.jpg

im Ursprungsbild an seine neue Position überführt. Wie bereits erwähnt wird beim Warping jedem Punkt des Ursprungsbildes eine neue Position im Zielbild zugeordnet. Durch das Tweening wird jeder Punkt im Ursprungsbild in die neue, durch das Warping berechnete Position überführt. Durch lineare Interpolation kann man eine Animation erzeugen. Dabei wird durch eine berechnete Intervallgröße schrittweise auf die neue Position zugesteuert.
Der dritte und letzte Teil des Morphings besteht aus dem so genannten *Cross-Dissolving* oder *Farb-Morphing*. Das Cross-Dissolving findet parallel zum Tweening statt. Dabei werden die einzelnen RGB-Werte jedes Pixels linear interpoliert und vermischt. Die Interpolation wird in Abhängigkeit von Quell- und Zielbild durchgeführt, d.h. während der ersten Interpolationsschritte ist der Anteil des Quellbildes deutlich höher als der des Zielbildes. Dann jedoch nimmt der Anteil des Zielbildes immer mehr zu, bis beim letzten Interpolationsschritt nur noch die RGB-Werte des Zielbildes enthalten sind und somit das Morphing beendet ist.

Anwendungsbeispiel für Morphing
Eine der vielen Anwendungen von Morphing sei am Beispiel in Verbindung mit Animationen kurz aufgezeigt. Dabei handelt es sich um die Animation einer Fotografie, z.B. dem Gesicht einer Person, welche in sehr realistischer Weise beliebige, gesprochene Audio-Files offensichtlich lippensynchron spricht. Die Idee dabei ist folgende: Es wird ein Referenz-Drahtmodell eines Kopfes entwickelt, welches über aufwendige Programmierung fest zugeordnete Phoneme lippensynchron darstellen kann. Nun kann man eine Fotografie einer Person in gewisser Weise wie eine Gummimaske über das virtuelle Drahtgestell ziehen.

Abb. 6.10 Zuordnung von Gesichtsmerkmalen zum Referenzmodell[29]

[29] hier mit dem Programm „Talking Show 2.0“ von CyberLink® erstellt

Dies erfolgt durch Zuordnung der Gesichtsmerkmale (wie Lippen-Endpunkte, Nase, Augen-Endpunkte, Gesichtskontur-Punkte etc.) zu den entsprechenden Punkten des Referenzmodell (vgl. Abb. 6.10 und Abb. 6.11).

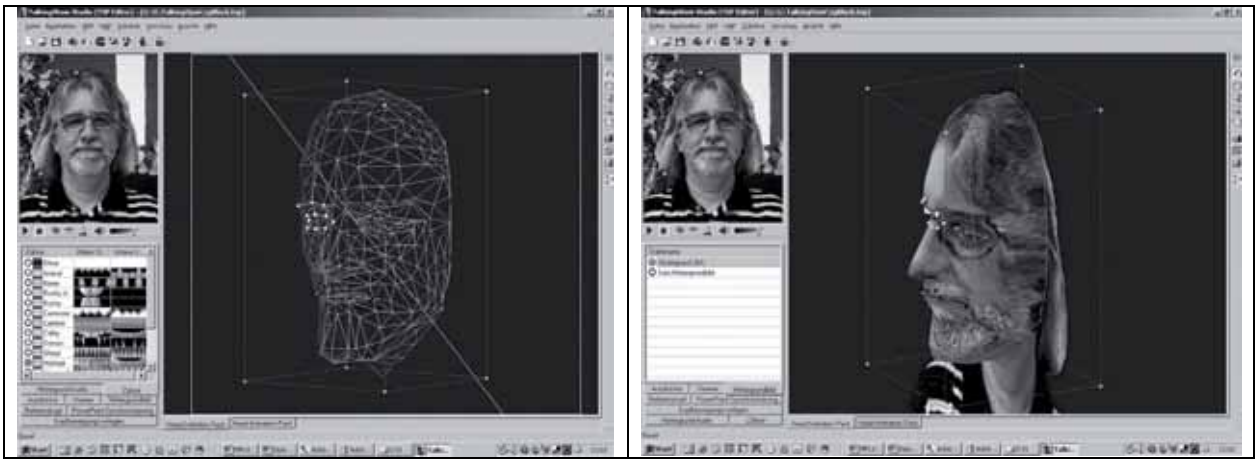


Abb. 6.11 Drahtmodell ohne und mit Gesichtstextur des Originalbildes

Nach der Modellierung des Gesichtsmodells kann dann eine beliebige Audio-Datei zusammen mit der Fotografie zu einem Videoclip kompiliert werden (vgl. Abb. 6.12).

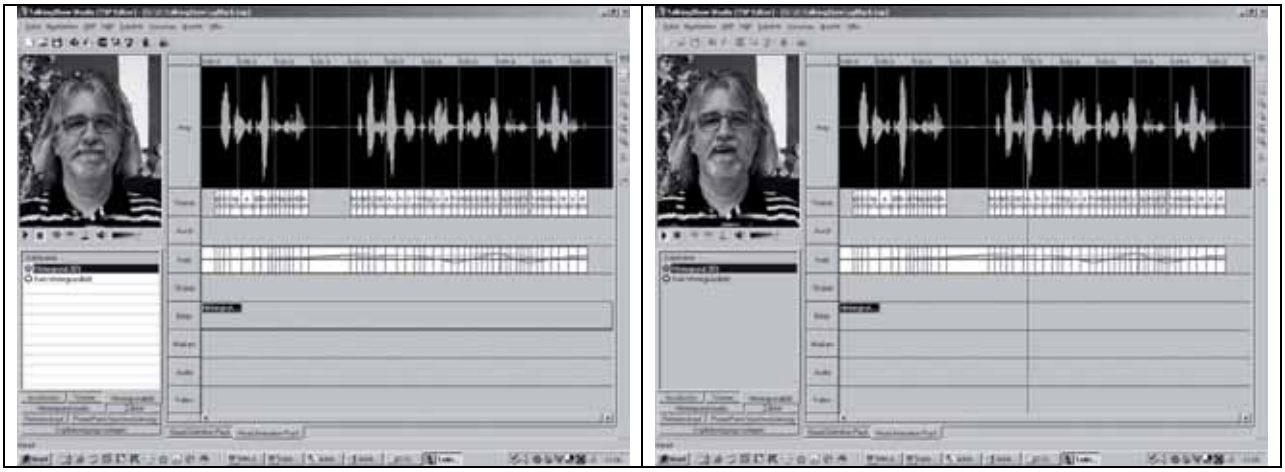


Abb. 6.12 „Sprechende“ Fotografie

Neben der Lippensynchronität wird durch Morphing ein zufälliges (editierbares) Bewegen des Kopfes erzeugt, zusammen mit realistischem Augenblinken. In Abb. 6.12 sieht man links das Originalbild mit einem Audiofile und rechts daneben eine Stelle, wo das Audio gerade ein Phonem präsentiert, bei dem der Sprecher den Mund passend öffnet. Das ganze kann schließlich als AVI- oder WMV-Videoclip exportiert werden.

Eine Anwendung dieser Technologie ist neben dem eLearning wieder in der Filmindustrie zu finden. So gibt es z.B. eine Dokumentation, bei der Schauspie-

ler die Taten verstorbener Politiker nachspielen. Das Gesicht des Schauspielers wurde anschließend mit oben genannten Methoden durch eine sprechende, 3D-animierte Fotografie des verstorbenen Politikers ersetzt, so dass der sehr realistische Eindruck entsteht, dass der Politiker selbst hier zu sehen ist, spricht und agiert. In ein paar Jahren wird es möglich sein, alle Szenen komplett durch 3D-Animationen zu ersetzen, so dass ein am Computertisch entstandener, vollständig mit Computergrafik animierter Film nicht von einem Film, der mit realen Schauspielern und realen Szenen gedreht wurde, zu unterscheiden ist.

Übungen zum Selbsttest

Erklären Sie den Weg von der Erstellung der Primitiven bis zur Ansicht auf einem Monitor (über eine 3D-Pipeline).

7. eLearning und Internet

Sollen Anwendungen für das Internet entwickelt werden, so gelten natürlich die auch die allgemeinen Prinzipien des Softwareengineerings. Wobei wir davon ausgehen, dass nicht nur einfache HTML-Seiten entworfen werden sollen, sondern Datenbank- oder datenbankähnliche Anwendungen oder auch Multimedia-Anwendungen wie z.B. eLearning.

Datenbankzugriffe über das Internet
Damit ist gemeint, dass z.B. ein User über das Internet Daten auf einem Server über eine Abfrage abrufen kann. Nachfolgend seien einige Gesichtspunkte hierfür verwendeter Internet-Strukturen näher betrachtet[30].
Neben HTML hat sich ein neuer Standard entwickelt, der schon in diese Richtung geht: XML (*Extensible Markup Language*). Dabei handelt es sich ebenso wie bei HTML um eine Markierungssprache. Beide sind abgeleitet von der „*Standard Generalized Markup Language*“, kurz SGML. Beide besitzen daher eine sehr ähnliche Syntax. Doch während XML eine Untermenge von SGML darstellt, ist HTML lediglich eine Anwendung derselben.
Im Unterschied zu HTML stellt XML nämlich einen Weg dar, Daten zu strukturieren und zu beschreiben, ohne ihnen jedoch eine Bedeutung zukommen zu lassen. Somit sind die reinen Informationen eines XML-Dokuments zunächst nicht viel Wert, wenn nicht klar ist, wie sie zu deuten sind. Die wahre Stärke von XML liegt in den zahlreichen Technologien und Sprachen, welche die Daten erst interpretieren und ihnen somit einen Sinn geben. Für HTML hingegen gilt dies nicht: Jede einzelne Markierung hat hier eine bestimmte Bedeutung. Der Browser „weiß“, wie er die verschiedenen Tags darstellen muss. Beispielsweise wird er einen Textblock, der in <i>- Tags gekapselt ist, *kursiv* darstellen. Um eine HTML-Datei anzuzeigen, werden somit keine weiteren Sprachen benötigt, es bedarf lediglich eines Agenten, der HTML Dokumente darstellen kann. Der große Nachteil dieser Art der Datenstrukturierung liegt auf der Hand: HTML trennt nicht zwischen den reinen Daten und den Informationen, die gebraucht werden, um jene entsprechend darzustellen. Dies erschwert das Herausfiltern von Informationen aus dem zusammengeworfenen Datenbrei. Auch Tags können nicht derart definiert werden, dass der Browser in diesem Fall wissen würde, wie er die Daten zu deuten hat. Es gibt außerdem auch noch syntaktische Unterschiede: Während es z.B. auch bei HTML gleichgültig ist, ob die Tags groß oder klein geschrieben werden oder ob Tags, wie z. B. <td> oder <tr>, nicht durch deren End-Tags geschlossen sind, da in solchen Fällen der Agent die

[30] vgl. Zöller-Greer, P.: *Softwareengineering für Informatiker und Ingenieure*, Vieweg Verlag 2002, Seite 297ff.

fehlenden Tags einsetzt, so ist XML da wesentlich strikter. XML unterscheidet zwischen Groß- und Kleinschreibung, und der jeweilige *XML-Parser* gibt sofort eine Fehlermeldung aus, falls ein Tag nicht durch ein entsprechendes Abschluss-Tag geschlossen wird. Diese rigidere Art kommt jedoch im Endeffekt dem Programmierer zugute, da Fehler bereits von Anfang an unterbunden werden. Schwächen von HTML, wie die mangelnde Flexibilität und die Vermischung von Information und Darstellung, sind bei XML besser gelöst.
Dies heißt jedoch nicht, dass XML das alternde HTML ersetzen wird. HTML wird auch weiterhin bestehen bleiben, und zwar als eines jener Werkzeuge, dessen sich XML bedient: Wie bereits angesprochen, enthält ein XML-Dokument reine Daten, ohne Informationen, wie diese angezeigt werden. Warum also nicht mit Hilfe einer Parser-Sprache wie XSL (*eXtensible Stylesheet Language*) die reinen Informationen in ein HTML-Dokument übersetzen? Dies könnte der Browser dann wiederum problemlos darstellen. Und tatsächlich ist jene Vorgehensweise bereits gängige Praxis und demonstriert ein Paradebeispiel für die Anwendung von XML. Dennoch lässt sich mit XML sehr viel mehr bewerkstelligen als nur die reine Erzeugung von HTML-Code. Zunächst soll der Aufbau eines XML Dokuments näher untersucht werden.
Die erste Zeile des Dokuments enthält stets die Angabe, dass es sich um ein XML Dokument handelt und nennt zugleich auch die Version und den verwendeten Zeichensatz, z.B.:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Im darauf folgenden Block wird die Struktur des Dokumentes festgelegt: Welche Tags an welcher Stelle erlaubt sind, welche Attribute für einzelne Tags zulässig sind und dergleichen. Diese Informationen werden benötigt, um zu überprüfen, ob die Daten im Dokument der vordefinierten Struktur genügen. Beispielsweise kann ein XML-Dokument Datensätze mit Adressinformationen von Mitarbeitern einer Firma enthalten. Dabei ist es wichtig, dass jeder Datensatz auch die gesamte Bandbreite der geforderten Informationen, also Name, Vorname, Straße, Ort usw. enthält und die einzelnen Daten korrekt in den entsprechenden Tags gekapselt sind.
Der gängige Weg, solche Richtlinien zum Aufbau eines XML Dokuments anzugeben, besteht in der Bereitstellung einer so genannten „Document Type Definition“, abgekürzt DTD. In der Tat gehört die Syntax der DTD zum Sprachumfang von XML. Eine mögliche Ausprägung der DTD für das oben angesprochene Adressbeispiel könnte in etwa so aussehen:

```
<!– DTD für das Adressbeispiel –>
<!DOCTYPE Adressen [
<– Definition der einzelnen Tags –>
```

```
<!ELEMENT Adressen (Datensatz)+>
<!ELEMENT Datensatz (Name, Vorname,
StrasseNr, OrtPLZ)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Vorname (#PCDATA)>
<!ELEMENT StrasseNr (#PCDATA)>
<!ELEMENT OrtPLZ (#PCDATA)>
]>
```

Allerdings wird die DTD nicht unbedingt benötigt und daher oft weggelassen. Man spricht in einem solchen Fall von „wohlgeformtem“ XML – unter der Annahme, dass ansonsten die Gesetzmäßigkeiten von XML erfüllt sind, im Gegensatz zu „gültigem“ XML, das eine DTD enthält und dieser auch genügt. Auf die Definition der DTD folgen die eigentlichen Daten, deren Syntax stark an die von HTML erinnert. Tatsächlich werden in beiden Sprachen die Tags und ihre zugehörigen Attribute in gleicher Weise notiert. Mit dem Unterschied allerdings, dass in HTML nur auf ein bestehendes, statisches Kontingent an erlaubten Tags zurückgegriffen werden darf. Wie schon erwähnt, ist XML diesen Einschränkungen nicht unterworfen. Die Tags müssen lediglich der zuvor in der DTD festgelegten Struktur entsprechen, falls diese existiert.
Z. B. folgende Daten sind gültiges XML in Bezug auf die zuvor definierte DTD:

```
<Adressen>
<Datensatz>
<Name>Engel</Name>
<Vorname>Sibylle</Vorname>
<StrasseNr>Fichtenweg 10</StrasseNr>
<OrtPLZ>63764 Aschaffenburg</OrtPLZ>
</Datensatz>
</Adressen>
```

Eine andere Möglichkeit, Daten über das Internet zugänglich zu machen, besteht in der Bereitstellung eines echten Datenbanksystems auf einem Web-Server sowie der Programmierung entsprechender Anwendungen, mit denen dann der User auf die Server-Datenbank zugreifen kann. Hier gibt es viele verschiedene Möglichkeiten, und es wird später anhand eines Beispiels aufgezeigt, wie über einen Apache-Webserver und einer MySQL-Datenbank mittels der Sprache PHP Anwendungen zum Datenzugriff über das Internet ermöglicht werden können. Zunächst seinen aber die benutzten Begriffe im Einzelnen näher beleuchtet.
Vor der Installation eines Web-Servers steht natürlich die Auswahl der Hardware-Plattform und die damit verbundene Entscheidung für das zugrunde liegende Betriebssystem. Für den Apache-Webserver selbst ist diese vorbereitende Phase nicht so entscheidend, da er für alle gängigen Plattformen in fertig geschnürten Paketen zur Verfügung steht. Diese reichen von Desktop-Plattformen wie den Microsoft-Betriebssystemen über Unix-Server-Plattformen wie IBM’s

AIX, HPUX oder Sun's Solaris bis hin zu Großrechner-Systemen wie OS/390. Entscheidender sind zu diesem Zeitpunkt die Anforderungen an die Belastbarkeit und die Stabilität der Basis. Hat man sich für eine Plattform entschieden, ist die einfachste Möglichkeit, den Apache-Webserver auf dem eigenen Rechner zu installieren, was der Download einer so genannten Binär-Distribution von der Apache-Web-Site ermöglicht. Diese hat gegenüber den ebenfalls verfügbaren Apache-Sourcen den Vorteil, dass man sich den Aufwand der Übersetzung sparen kann und dafür auch keine Programm-Entwicklungsumgebung benötigt.
Das Surfen im World Wide Web funktioniert nach dem Client-Server-Prinzip. Ein Client (hier: der Web-Browser) fordert von einem (Web-)Server ein Dokument an, beispielsweise http://www.fh-frankfurt.de/index.htm. Der Web-Server empfängt diese Aufforderung und sucht auf der Festplatte des Servers nach der Datei index.htm. Der Inhalt dieser Datei wird an den Client, den Web-Browser, zurückgeliefert. Bei serverseitigen Script-Sprachen wird das etwas anders gehandhabt. Nehmen wir an, der Web-Browser fordert die Datei http://www.fh-frankfurt.de/ index.php an. Der Web-Server sucht nun nach der Datei index.php, weiß aber auf Grund der Dateiendung (.php ist die Standardendung für sog. PHP-Dateien, siehe unten), dass es sich um ein PHP-Skript handelt. Deswegen gibt er nicht die PHP-Datei zurück, sondern sorgt dafür, dass der PHP-Interpreter aufgerufen wird, der die Datei (mit PHP-Code) in HTML umwandelt. Dieser HTML-Code wird an den Web-Browser zurückgeliefert. Um den Web-Server für die Unterstützung von PHP zu konfigurieren, reicht es nicht aus, nur PHP zu installieren, sondern der Web-Server muss erkennen, dass er Dateien mit der Endung .php an den PHP-Interpreter weiterleiten muss. Je nach verwendetem Web-Server funktioniert das unterschiedlich.
PHP wurde einst als Hobbyprojekt von Rasmus Lerdorf entwickelt. Er stellte eine erste Version im Internet öffentlich zur Verfügung und bat um Kommentare und Anregungen. Später begannen eine Reihe von Gleichgesinnten das PHP (damals noch PHP/FI „*Personal Homepage Tools/Form Interpreter*“ genannt) zu testen und selbst Code und Erweiterungen beizusteuern. PHP konnte mit dem Erscheinen von Version 3 erstmals kommerziell ernst genommen werden. Die Dokumentation, bei vielen Open-Source-Projekten ein Manko, war passabel (und ist mittlerweile exzellent). Erstmals wurde zudem nicht nur der Quellcode veröffentlicht, sondern es wurden auch ausführbare Dateien für die im Privatbereich wichtige Windows-Plattform zur Verfügung gestellt. Im Jahr 2000 erschien die Nachfolgeversion PHP4. Diese wurde nicht mehr von Rasmus Lerdorf als Hauptausführendem gestaltet und bietet eine ganze Reihe von neuen Konzepten. PHP als Skript-Sprache kann man durch folgende Eigenschaften und Merkmale beschreiben: PHP ist eine leistungsfähige Skriptsprache, primär dazu ausgelegt im Web-Umfeld verwendet zu werden. PHP geniest seit seiner vierten Version (bzw. vierten Auflage) eine breite Unterstützung, einerseits durch Web-

server-Einbindungen, andererseits aber auch durch zahlreiche Funktionen, die inzwischen im PHP implementiert wurden. PHP ist einfach zu erlernen, benutzt die vielfach verwendete ANSI-C Syntax und unterstützt von Haus aus viele Datenbanken. Die für das Webscripting unumgänglichen String-Operationen hat es zum größten Teil von Perl übernommen. Die Einbindung des PHP Codes mit HTML erfolgt analog zu Microsofts ASP (Applikations-Service-Provider). Auf diese Weise wird von allen bekannten Sprachen und Techniken „das Beste“ verwendet und in PHP vereint.
MySQL ist eine kleine, aber im Web-Umfeld weit verbreitete Datenbank. Ihr Vorteil ist u.a. die große Unterstützung in der Web Open Source Gemeinde. Es gibt unzählige Projekte, die MySQL als Datenbank nutzen. Dies liegt nicht zuletzt auch daran, dass MySQL sehr einfach zu handhaben ist. In diesem Zusammenhang kann man das Projekt *phpMyAdmin* erwähnen. Es ist das bekannteste Frontend für die MySQL Datenbank, die auf PHP basierend eine sehr komfortable Oberfläche zur Datenbankadministration bietet. Es ist sehr einfach zu installieren und zu benutzen. Gerade im Webbereich ist der Einsatz problemlos möglich. Verwendet werden kann es mit MySQL ab Version 3.21.x, mit geringen Einschränkungen auch mit älteren Versionen. Folgende Funktionen sind per Mausklick ausführbar:

- Erzeugen und Löschen einer Datenbank
- Erzeugen, Löschen, Leeren und Kopieren von Tabellen
- Löschen, Bearbeiten und Hinzufügen von Feldern
- Ausführen von SQL-Kommandos und Stapelanweisungen
- Bearbeitung von Schlüsseln, Indizes usw.
- Export und Import von Textdateien in die Datenbank
- Administration einer Datenbank

Es sei nachfolgend ein praktisches Beispiel zur Demonstration der Zusammenarbeit von php-Eingaben mit MySQL auf dem Webserver angegeben.
In diesem Beispiel sollen Studenten über die Eingabe ihrer Personalien später Zugang zu Klausurergebnissen haben. Diese Ergebnisse unterliegen natürlich dem Datenschutz, so dass ein Student oder eine Studentin erst nach Registrierung über ein individuelles Passwort die Klausurergebnisse abrufen kann. Abb. 6.13 zeigt, wie so eine Registrierungsseite aussehen könnte. Man erkennt, dass die URL die besagte Endung .php besitzt, womit klar ist, dass also hier ein php-Skript abläuft. Dasselbe sammelt die gemachten Eingaben und schreibt diese in die MySQL-Datenbank auf dem Apache-Server. Die Matrikelnummer wird später als Passwort für den Zugang des Studenten genutzt.

Abb. 6.13 Eingabemaske eines php-Formulars

Nach Registrierung kann der Klausurteilnehmer dann die Noten erfahren (Abb. 6.14).

Abb. 6.14 Abfrage eines Klausurergebnisses

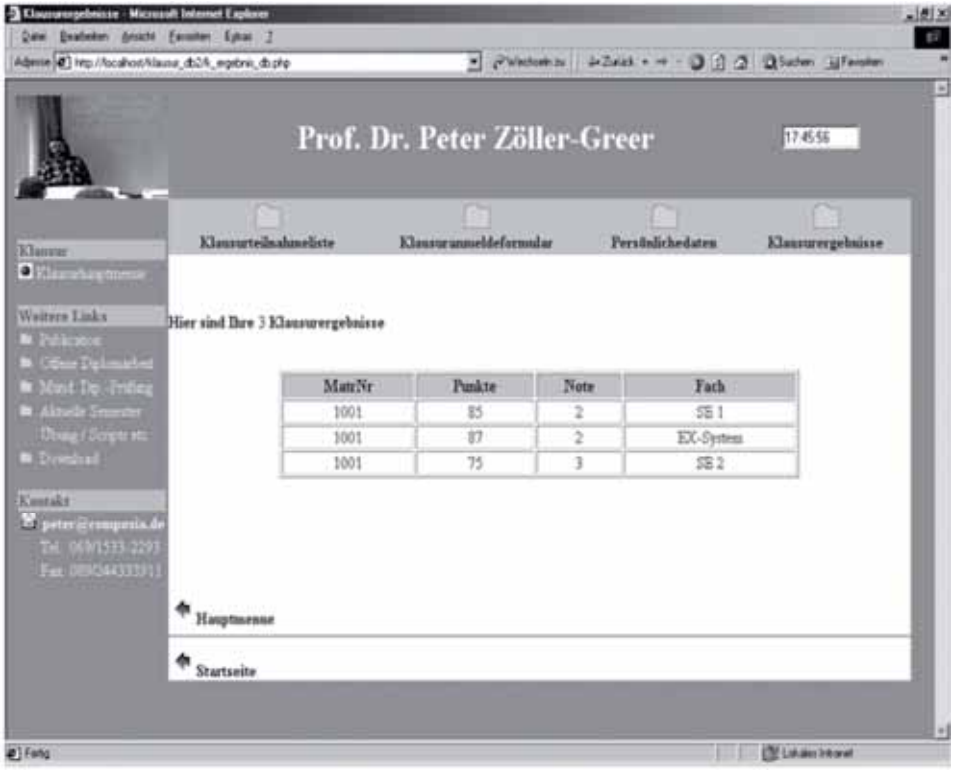


Abb. 6.15 Rückgabewerte des Servers

Als Beispiel sei nachfolgend das php-Skript der Seite „k_Ergebnis_sehen.php“, welches Abb. 6.14, also der Abfrage der Ergebnisse zu einer Matrikelnummer, entspricht:

```
<html>
<head><title>Klausurergebnisse sehen</title>
<?php
  include("layout_oben.template");
?>
<?php
  include("layout_menue.template");
?>
<br><br><br><br>
<h2>Klausurergebnisse sehen</h2>
<hr>
<br><br><br><br>
<form action="k_ergebnis_db.php" method="post">
<table border=0>
<tr>
       <td><b>Matrikel-Nr:<b>    </td>
       <td><input type=text name="MatrNr" maxlength=6 size=10></td>
</tr>
<tr><td>  </td><td>  </td></tr>
<tr>
       <td></td><td><input type="submit" value=  "  OK
"> </td>
```

```
        <td><input type="reset" value="Abbrechen"></td>
</tr>
</table>
</form>
<?php
  include("layout_unten.template");
?>
</body>
</html>
```

Ohne jetzt auf den Syntax jeder Zeile im Einzelnen einzugehen ist erkennbar, dass über den Befehl *input* die Matrikelnummer eingelesen und mittels des Befehls *submit* an der Server geschickt wird. Die Rückgabe der gefilterten Klausurergebnisse, wie sie in Abb. 6.15 zu sehen ist, wird durch folgende php-Seite erreicht:

```
<html>
<head><title>Klausurergebnisse</title>
<?php
  include("layout_oben.template");
?>
<?php
  include("layout_menue.template");
?>
<br>
<hr>
<br><br>
<?php
if ($MatrNr!=0)
{
 include("dbconnect.inc");
    if($link !=0)
    {
    //Jan added this Comment
    $anfrage="SELECT Fach,Semester,Note,BelegNr From k_ergebnis
WHERE MatrNr=\"$MatrNr\"  ";
    //$anfrage="show tables";
    //$ergebnis=mysql_query($anfrage) or die ("Fehlermel-
dung=".mysql_error());
    $ergebnis=mysql_query($anfrage);
          //Jan added this IF
          if (!$ergebnis )
            {
                print "Datenbank abfrage nich erfolgreich  <br>";
                print "Fehlermeldung = ";
                print mysql_error();
            } // end if
          // Jan
          else
            {
            //Original Part
```

```
    echo mysql_num_rows($ergebnis)."<b> Klausur Ergebnisse
f&uuml;r Studnet(in) mit Matrikelnummer: $MatrNr</b><p> ";
    echo "<TABLE border='1' width= '70%'align='center'>";
     for ($i=0;$i< mysql_num_fields($ergebnis);$i++)
          { echo "<TH bgcolor=#cccccc>".
          mysql_field_name($ergebnis,$i)."</TH>"; };
               while ($zeile=mysql_fetch_row($ergebnis))
                     { echo "<TR align='center'>";
       for ($i=0;$i<mysql_num_fields($ergebnis);$i++)
               { echo "<TD>".$zeile[$i]."</TD>";
               };
                  echo "</TR>";
                    };
               echo "</TABLE>";
               mysql_free_result($ergebnis); mysql_close($link);
           } // end else
    }// end if

    else
          {
           echo "<font color=#ff0000><b>Die eingegebene Matrikel-
Nummer ist Fehlerhaft !!!</b></font>";
         echo"<br>";
         }; // end else
}; // end if
?>
<br><br>
<?php
  include("layout_unten.template");
?>
</body>
</html>
```

Auch hier kann nicht auf die Syntax im Einzelnen eingegangen werden, doch es ist auch so ersichtlich, dass Skript-Befehle zum Aufbau der Ergebnis-Tabelle sowie ein *Select*-Befehl zum Ausfiltern der Ergebnisse auf die eingegebene Matrikelnummer etc. vorhanden sind. Auch der C^{++}-ähnliche Aufbau für das Ansprechen von MySQL innerhalb dieses php-Skirpts ist erkennbar.

eLearning-Anwendungen

In jüngster Zeit bieten Industrie und Hochschulen zunehmend die Möglichkeit, Weiterbildung über das Internet als Fernstudium verfügbar zu machen. Dabei ist es natürlich sinnvoll, alle multimedialen Möglichkeiten auszuschöpfen, sich also nicht nur darauf zu beschränken, einfach schriftliche Ausarbeitungen als HTML-Seiten oder zum Download anzubieten (wobei dies schon ein wichtiger Teil eines eLearning-Konzepts sein kann). Die Attraktivität des eLearning besteht gerade darin, z.B. Video-Clips, Sounds, Animationen sowie Texte und Grafiken miteinander zu kombinieren. Die Erstellung solcher virtuellen Vorlesungen ist

entsprechend aufwändig. Es gibt Untersuchungen, nach denen –je nach Aufwand- für eine Stunde multi-medialen virtuellen Unterricht 100-300 Zeitstunden investiert werden müssen. Gerade wenn Animationen und Video-Clips integriert sind, wird diese Zeitspanne schnell erreicht. Videos beispielsweise müssen aufgenommen werden, wobei die Licht- und Sound-Verhältnisse entsprechend erzeugt werden müssen. Dekorationen müssen ggf. angeschafft werden, (mind.) ein Kameramann muss filmen, das Ganze muss auf Video aufgezeichnet und nachbearbeitet werden (Schnitte, Überblendungen etc.). Für den genauen Ablauf sind oft die Texte zuvor zu schreiben, und es muss ein detailliertes Drehbuch vorhanden sein, welches ja auch jemand erstellen muss. Das fertige Video wird schließlich in ein internetgerechtes Format gebracht (z.B. in das Real®-Format) und in eine Web-Seite integriert, welche ebenfalls zu designen und zu programmieren ist.

Noch aufwändiger wird es, wenn z.B. synchron zu einem Videoclip noch Texte, Animationen und/oder Grafiken eingeblendet werden müssen. Prinzipiell kann man hierfür zwei Vorgehensweisen benutzen. Zum Einen könnte man alle erforderlichen Grafiken und Texte etc. einfach als Video filmen und digitalisieren und mit geeigneter Schnittsoftware in den ursprünglichen Clip an der „richtigen" Stelle einfügen. Das Ganze hat aber verschiedene Nachteile. Zunächst muss das Video-Fenster relativ groß sein, damit Texte und Grafiken überhaupt erkannt werden können. Soll das Ganze über das Internet laufen, dann ist so was bei den heutigen Übertragungskapazitäten allenfalls im Breitbandbereich möglich. Die Videos brauchen viel Speicherplatz, und werden z.B. längere Textteile eingeblendet, so verschwendet man diesen Platz für Standbilder, die für sich allein genommen, wären sie kein Video, eigentlich viel weniger Speicherplatz benötigen würden. Ein weiterer Nachteil besteht darin, dass durch das Einblenden einer Szene (z.B. eines Textes oder einer Grafik) etwas anders ausgeblendet werden muss (z.B. der Moderator). Deswegen ist die zweite Variante zu bevorzugen: Texte und Grafiken bleiben dies und werden synchron zu einem Video-Clip, der in einem eigenen Fenster abläuft, ein- bzw. ausgeblendet. Ist im Clipfenster beispielsweise nur der Moderator zu sehen, so kann dieses recht klein gehalten werden, was dazu führt, dass man nur wenig Speicherplatz für den entsprechenden Videofile benötigt. Das Problem ist dann natürlich, die Ein- und Ausblendungen der Texte und Grafiken synchron mit dem Video an den „richtigen" Stellen hinzukriegen.

Eine Möglichkeit hierfür bietet die Skriptsprache SMIL (*Synchronized Multimedia Integration Language*), welche von Real Networks Inc. (USA) entwickelt wurde. Sie basiert auf der Wiedergabe mit Hilfe des von der gleichen Firma entwickelten und weit verbreiteten RealPlayers®. Dieser benutzt das so genannte Streaming zum Übertragen von Videos. Diese Methode erlaubt, dass das Video in „Chunks" übertragen werden kann, also nicht erst heruntergeladen werden

muss bevor man es ansehen kann. Beim Abspielen im RealPlayer wird zunächst der erste Chunk heruntergeladen, und während dieser läuft, wird der nächste nachgeladen. Es muss also nicht wie sonst üblich erst die ganze Videodatei auf einmal heruntergeladen werden, sondern durch Buffering können Teile davon geladen und gleich angeschaut werden, während die nächsten Chunks am Herunterladen sind. Damit passt sich die Wiedergabe auch an die Geschwindigkeit des Rechners und der Übertragung selbständig an. Außerdem sind Real-Videos extrem komprimiert und besitzen damit einen relativ kleinen Speicherplatzbedarf. Das Programm RealProducer® dient zur Umwandlung von digitalen Videofiles in das Realformat. Das Video kann dann vom eigenständigen RealPlayer, welcher von den Standardbrowsern automatisch aufgerufen wird, betrachtet werden. Geeignete PlugIns ermöglichen es auch, den RealPlayer voll in eine HTML-Seite zu integrieren.

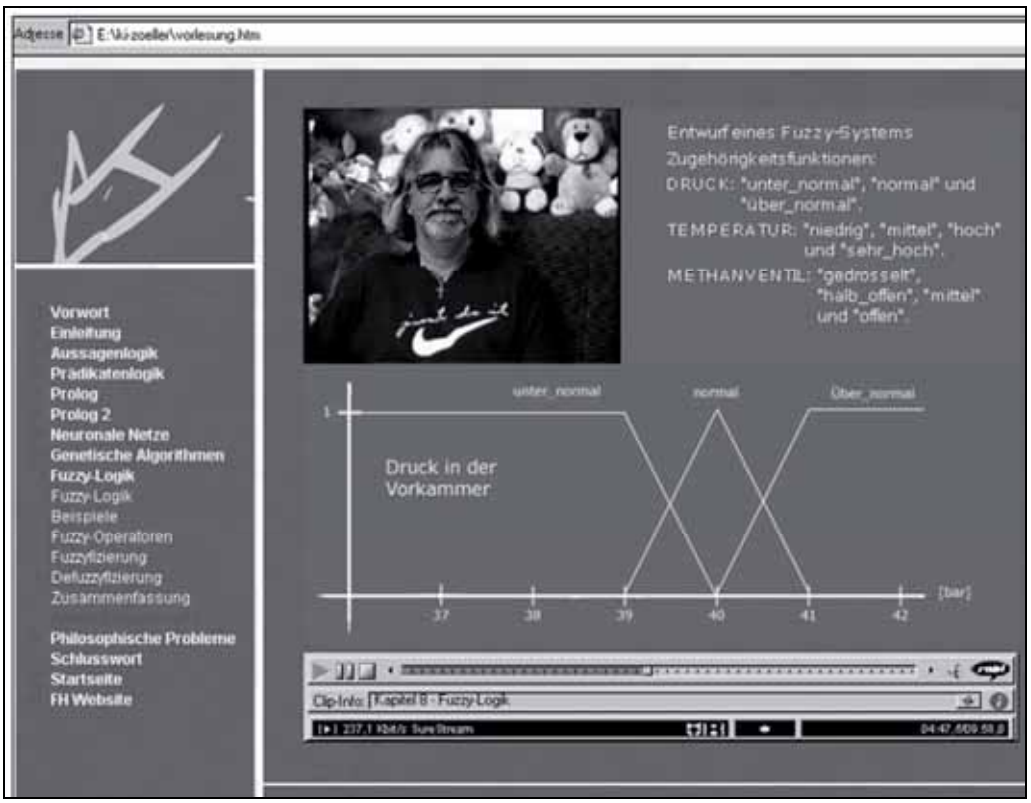


Abb. 6.16 Multimediale virtuelle Vorlesung mit SMIL erstellt

Abb. 6.16 zeigt den Ausschnitt einer virtuellen Vorlesung über „Künstliche Intelligenz“, wie sie vom Autor des Buches angeboten wird. Dieser Screen-Shot des Internetbrowsers zeigt, wie die verschiedenen Multimediakomponenten integriert sind: In der linken oberen Ecke läuft eine Flash-Animation ab (es drehen sich dreidimensional die zwei Buchstaben „KI“), links darunter ist das Auswahlmenü der jeweiligen Lerneinheiten. Im Hauptfenster sieht man im unteren Teil die Steuerkonsole des RealPlayers, welcher hier als PlugIn integriert ist. Das Video kann jederzeit angehalten, vor- und zurückgespult und an jeder belie-

bigen Stelle fortgesetzt werden. Links oben im Hauptfenster ist der eigentliche Videoclip zu sehen. Rechts daneben steht der „Tafel-Anschrieb“, also wichtige Textteile, über die der Moderator gerade spricht. Darunter wird passend eine Grafik eingeblendet, auf die sich der Moderator ebenfalls gerade bezieht. Texte und Grafiken werden dann an den entsprechenden Stellen ein- und wieder ausgeblendet, während das Video des Moderators läuft. Selbst mit einem analogen Modem (56k) kann dies noch einigermaßen bequem betrachtet werden.
Das synchrone Ein- und Ausblenden der Texte und Grafiken etc. wird mit einem SMIL-Skript erledigt. SMIL-Skripts sind relativ einfach aufgebaut und ähneln der HTML-Syntax. Nachfolgend wird ein Teil des SMIL-Skripts angegeben, welches sich auf Abb. 6.16 bezieht:

```
<smil>
    <head>
      <meta name="title" content="Kapitel 8 - Fuzzy-Logik" />
      <meta name="author" content="Till Wagner" />
      <meta name="copyright" content="fh©" />
      <meta name="keywords" content="" />
      <meta name="description" content="" />
      <meta name="robots" content="all" />
      <layout type="text/smil-basic-layout">
      <root-layout width="560" height="411" background-
color="#006699"/>
      <region id="title_region" left="0" top="0" width="600"
height="450" z-index="4" />
      <region id="movie" left="0" top="0" width="272" height="200"
colour background-color="#006699" z-index="3" />
      <region id="text_region" left="272" top="0" width="288"
height="200" colour background-color="#006699" z-index="2" />
        <region id="txtpic_region" left="0" top="200" width="560"
height="222" colour background-color="#006699" z-index="1" />
      </layout>
    </head>
    <body>
      <par>
        <seq>
          <par>
            <video src="k8_clip1_video.rm" region="movie"/>
            <text src="k8_clip1_text.rt" region="text_region"/>
            <text src="k8_clip1_pic.rp" region="txtpic_region"
fill="freeze"/>
          </par>
        </seq>
      </par>
    </body>
</smil>
```

In diesem Skriptteil werden vornehmlich die Positionierungsbereiche der Clips, Texte und Grafiken im Realfenster bestimmt. Wichtig sind die drei Befehle

```
<video src="k8_clip1_video.rm" region="movie"/>
```

```
<text src="k8_clip1_text.rt" region="text_region"/>
<text src="k8_clip1_pic.rp" region="txtpic_region"
fill="freeze"/>
```

Hier werden nämlich die Dateien für den Video-Clip angegeben (*k8_clip1_video.rm*), für den textuellen Teil (*k8_clip1_text.rt*) sowie für eingeblendete Bilder *(k8_clip1_text.rp*). Das SMIL-Skript *k8_clip1_text.rp* sei nachfolgend beispielhaft angegeben:

```
<imfl>
        <head
          title="A.I. Kapitel 8"
          copyright="fh frankfurt am Main"
          timeformat="dd:hh:mm:ss.xyz"
      background-color="#006699"
          preroll="10.0"
          bitrate="12000"
          width="560"
          height="222"
          duration="9:58.0"
           pos="center"
        />
<!-- Assign handle numbers to images -->
  <image handle="1" name="pics/k8_1.jpg"/>
  <image handle="2" name="pics/k8_2.jpg"/>
  <image handle="3" name="pics/k8_3.jpg"/>
  <image handle="4" name="pics/k8_4.jpg"/>
  <image handle="5" name="pics/k8_5.jpg"/>
  <image handle="6" name="pics/k8_6.jpg"/>
<!-- These effects define the timeline of the presentation -->
  <fill start="0" color="#006699"/>
  <fadein start="4:31.0" duration="1.0" target="1" aspect="true" />
  <crossfade start="4:54.0" duration="1.0" target="2" aspect="true"
/>
  <crossfade start="5:18.0" duration="1.0" target="3" aspect="true"
/>
  <fadeout start="5:39.0" duration="0" target="3" aspect="true" />
  <fill start="5:39.0" color="#006699" />
  <fadein start="8:06.0" duration="1.0" target="4" aspect="true" />
  <crossfade start="8:24.0" duration="1.0" target="5" aspect="true"
/>
  <crossfade start="8:56.0" duration="1.0" target="6" aspect="true"
/>
  <fadeout start="9:27.0" duration="0" target="4" aspect="true" />
  <fill start="9:27.0" color="#006699" />
</imfl>
```

Wie man sieht, werden den physikalischen Bildern so genannte Target-Nummern (*Image Handles*) zugewiesen, auf die sich dann später bezogen wird. Die Befehle *fadein, fadeout, crossfade* etc. beziehen sich auf die durch das Tar-

get angegebenen Grafiken, wobei genau die Zeitwerte, wann die jeweilige Aktion relativ zum Start des Videos erfolgen soll, angegeben werden. Analog geschieht die Einblendung von Texten.
Es sind noch weitere Techniken zum Erzeugen vergleichbarer multimedialer Anwendungen verfügbar, wie z.B. Flash® von Macromedia. Damit kann auch die Zeit für die Erstellung solcher Anwendungen verkürzt werden, denn Flash bietet z.B. weitgehende grafische Editiermöglichkeiten zum Erstellen solcher Applikationen.

In jüngster Zeit gibt es auch sog. Virtuelle Klassenzimmer, deren Aufbau ähnlich wie in Abb. 6.16 beschrieben aussieht. Entscheidender Unterschied: Es wird keine Konserve abgespielt, sondern der Unterricht erfolgt „live“ und eine Kommunikation mit den Teilnehmenden ist während der Veranstaltung z.B. über Video, Audio und/oder Chat möglich (siehe Abb. 6.17).

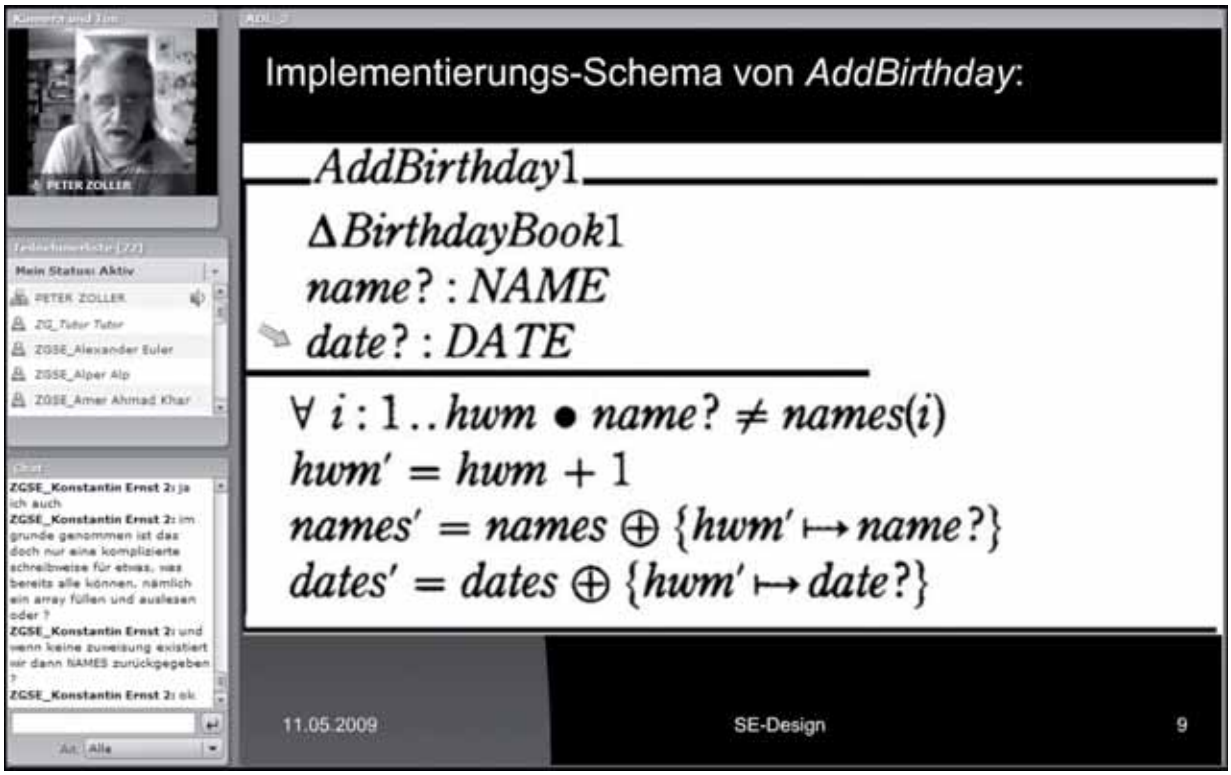


Abb. 6.17 Virtuelles Klassenzimmer (Adobe Connect Pro)

Übungen zum Selbsttest

Wie könnte man ein Software-Werkzeug erstellen, welches die Skripte für die beschriebenen SMIL-Dateien automatisch erstellt? Geben Sie geeignete Eingabe-Maske an für „Nicht-Programmierer“ in einer userfreundlichen Anwendung.

8. Spezialanwendung: Nachkolorieren von s/w-Filmen

Naturgemäß gibt es viele Anwendungen von Multi Media Systemen, und es sei hier eine herausgegriffen, welche auch die Komplexität demonstriert, die damit verbunden sein kann.
Immer wieder sind im Fernsehen alte Dick- und Doof-Filme in Farbe zu sehen. Meistens sind diese Filme nachkoloriert, d.h. es handelte sich ursprünglich um Schwarzweißfilme, die hinterher in einem aufwändigen Verfahren mittels Computer nachkoloriert wurden.
Das Nachkolorierungsverfahren, welches ich hier vorstellen möchte und nach den Autoren als „Levin-Lischinski-Weiss-Methode[31]“ bezeichne, sieht drei Schritte vor:

1. Zuordnung von Farben zu den s/w-Objekten
2. Objekterkennung und Einfärben dieser Objekte
3. Objektverfolgung in einer Filmszene und richtiges Einfärben der bewegten Objekte

Die Punkte 2. und 3. stellen hohe mathematische Anforderungen an die Algorithmen. Die Einzelbilder eines Filmes („Frames“) werden als Matrizen aufgefasst (mit Millionen von Elementen), die invertiert und potenziert werden müssen etc.; so wird auch einem Rechner viel abverlangt.
Am Einfachsten ist Punkt 1. Ein Farbbild besteht bekanntermaßen aus der Grauwert-Helligkeitsinformation, auch Luminanz genannt (dies entspricht dem s/w-Anteil eines Bildes) und den Farbanteilen jedes Pixels (auch Chromanz genannt). Aus dem s/w-Bild kann man natürlich nicht auf die Farbinformation schließen, denn z.B. ein grauer Pullover kann blau, grün, rot oder eine Kombination davon gewesen sein. Ein Grauwert im s/w-Bild hängt nur von der Helligkeit, nicht aber von der Farbe ab. Würde man bestimmten Grauwerten einfach immer bestimmte Farben fest zuordnen, so erhielte man sog. „Falschfarben-Bilder“, wie Sie in verschiedenen naturwissenschaftlichen Bereichen zu Analysezwecken eingesetzt werden. Diese sind jedoch aus den genannten Gründen nicht für realistische Farbeindrücke brauchbar.
In unserem Fall ist das Problem so gelöst: Es wird mit Hilfe eines Zeichenprogramms (wie Paint® oder Photoshop®) die einem Objekt zughörige Farbe in das Objekt hinein „gekritzelt“ (engl. „scribble“). Also ein blauer Strich in den Pullover, ein orange-brauner Strich in das Gesicht einer Person, ein blauer Strich in den Himmel usw., die Farbintensität der Scribbles ist dabei egal (also z.B. ob

[31] Anat Levin, Dani Lischinski, Yair Weiss: *Colorization Using Optimization*, in http://www.cs.huji.ac.il/~yweiss/Colorization/

tiefrot oder rosa), denn der Helligkeitswert der Farbe wird später aus dem s/w-Bild übernommen.
In nächsten Schritt (Punkt 2.) sucht nun ein aufwändiger mathematischer Algorithmus die Objektgrenzen und färbt die gescribbelten Objekte entsprechend ein. Auf dem rückseitigen Umschlagbild dieses Buch ist dafür ein Beispiel zu sehen: Der „Input" besteht aus dem originalen s/w-Bild sowie dem gescribbelten Bild. Die Ausgabe des Algorithmus liefert dann das nachkolorierte Bild.
Ist das Aufsuchen und Einfärben von Objekten in einzelnen Bildern schon eine sehr aufwendige Sache, wird im Falle von s/w-Filmen das ganze noch komplizierter. Man will ja nicht jedes Einzelbild einfärben, sondern nur ein „Startbild" und hofft, dass ein Algorithmus erkennt, welche Objekte sich im Film bewegen. Das geht natürlich nur, solange kein Szenenwechsel vorliegt. In diesem Fall muss der neue Startframe wieder farblich neu zugeordnet werden.
Zur Objektverfolgung werden sog. „Flow-Algorithmen" benutzt. Diese erkennen die Bewegung von Objekten durch deren Vektorisierung. In der Praxis muss dennoch manchmal nachkorrigiert werden. Das benutzte Verfahren hat aber den Vorteil, dass nur wirklich in diesen „Zwischenscribbles" Striche für die Korrekturen da sein müssen, es muss also nicht noch mal das ganz Bild neu gescribbelt werden.
Das Ganze kann als ein 3D-Problem aufgefasst werden: Man hat die 2 Dimensionen der Frames und die Zeitachse als 3. Dimension. Eine Bildfolge entspricht dann einer großen räumlichen Matrix (mit den Pixel als Matrixelemente). Das Problem der Objekterkennung verschmilzt dann mit dem der Objektverfolgung zu einem mathematischen Optimierungsproblem im 3D-Raum. Daher nennen Levin, Lischinski und Weiss ihre Lösung auch „Colorization by Optimization". Es wird dabei davon ausgegangen, dass in dieser räumlichen 3D-Matrix benachbarte Pixel ähnlicher Intensität (sowohl in der Ebene wie auch in der Zeit) gleiche Chromanzwerte zugewiesen bekommen. Der Wechsel von einer zur nächsten Farbe geschieht über den Objektwechsel, der durch einen Gradientenoperator (und damit über die Objekt-Konturen) ermittelt wird. Im einzelnen geht der Algorithmus wie folgt[32]:
Der Algorithmus bezieht sich auf dem YUV Farbraum, der gewöhnlich zur Codierung der Farbe bei der Videobearbeitung verwendet wird, wobei Y der monochrome Luminanzanteil ist. Hier wird Y einfach als Intensität genutzt, während U und V die Chrominanzanteile darstellen. Es gibt bei diesem Algorithmus ein „Intensitätsvolumen" Y(x, y, t) als Eingabe und zwei „Farbvolumen" U(x, y, t) und V(x, y, t) als Ausgabe. Um die Beschreibung zu vereinfachen, werden folgende Abkürzung verwendet: Fettgedruckte Buchstaben, z.B. **r,**

[32] ibid.

s sind Symbole für Tupel (x, y, t). Auf diese Art ist Y(**r**) die Intensität eines bestimmten Bildpunktes zu einer Zeit t.
Es wird vorausgesetzt, dass die zwei benachbarten Bildpunkte **r**, **s** dann ähnlichen Farben haben sollen, wenn auch ihre Intensitäten ähnlich sind. Auf diese Weise soll der Unterschied zwischen der Farbe U(**r**) am Bildpunkt **r** und dem gewichteten Durchschnitt von den Farben an Nachbarbildpunkten minimiert werden. Als Ansatz gilt:

$$J(U) = \sum_r \left(U(r) - \sum_{s \in N(r)} w_{rs} U(s) \right)^2 \qquad (1)$$

wobei w_{rs} eine Gewichtungsfunktion darstellt, deren Wert dann groß ist, wenn Y(**r**) und Y(**s**) ähnlich sind, und deren Wert klein ist, wenn die zwei Intensitäten stark verschieden sind.
Ähnliche Gewichtsfunktionen werden oft in Algorithmen für Bildsegmentierung verwendet, wo sie normalerweise *Verwandtschaftsfunktionen* genannt werden.
Zwei Gewichtungsfunktionen werden häufig verwendet. Die einfachste Variante wird meist von Bildsegmentierungsalgorithmen benutzt und basiert auf dem Quadrat des Differenzbetrags zwischen den zwei Intensitäten:

$$w_{rs} \propto e^{-(Y(r)-Y(s))^2 / 2\sigma_r^2} \qquad (2)$$

Eine zweite Gewichtungsfunktion basiert auf der normalisierten Korrelation zwischen den zwei Intensitäten:

$$w_{rs} \propto 1 + \frac{1}{\sigma_r^2}(Y(r) - \mu_r)(Y(s) - \mu_r) \qquad (3)$$

wobei μ_r und σ_r das Mittel und die Varianz der Intensitäten in einem Fenster um **r** sind.
Die Korrelationsverwandtschaft kann auch von einer Annahme der lokalen linearen Relation zwischen Farbe und Intensität hergeleitet werden.
Die Formel setzt voraus, dass die Farbe an einem Bildpunkt U(**r**) eine lineare Funktion der Intensität Y(**r**) ist: U(r) = aiY(**r**) + b_i und die linearen Koeffizienten a_i, b_i gleich sind für alle Bildpunkte in einem kleinen Fenster um **r**.

Diese Annahme kann empirisch begründet werden. Intuitiv bedeutet es, wenn die Intensität konstant ist, dass die Farbe auch konstant sein soll, und wenn die Intensität ein „Rand“ ist, auch die Farbe ein „Rand“ sein sollte (obgleich die Werte auf den zwei Seiten des Randes zwei verschiedene Zahlen sein können). Während dieses Modell dem System ein Paar Variablen pro Bildfenster hinzufügt, erbringt eine einfache Beseitigung der a_i, b_i eine Gleichung, die mit Gleichung (1) einer auf Korrelation basierende Affinitätsfunktion gleichwertig ist.

Die Darstellung $\mathbf{r} \in N(\mathbf{s})$ bezeichnet die Tatsache, dass **r** und **s** benachbarte Bildpunkte sind. In einem einzelnen Frame des Filmes werden zwei Bildpunkte als Nachbarn definiert, wenn ihre Positionen „nahe“ ist. Zwischen zwei aufeinanderfolgenden Frames werden zwei Bildpunkte als Nachbarn definiert, wenn ihre Positionen, nach Berechnung der Objektverfolgung „nahe“ ist.

Formal werden $v_x(x, y)$ und $v_y(x, y)$ als *optischer Fluss* bezeichnet und der zum Zeitpunkt t errechnet wird. Dann ist der Bildpunkt (x_0, y_0, t) ein Nachbar des Bildpunktes $(x_1, y_1, t+1)$, wenn:

$$\left\| (x_0 + v_x(x_0), y_0 + v_y(y_0)) - (x_1, y_1) \right\| \angle T \qquad (4)$$

Das „Ablauffeld“ $v_x(x_0)$, $v_y(y_0)$ ist mit Hilfe eines Standard-Algorithmus für Bewegungseinschätzung berechnet. Der optische Fluss wird hier verwendet, um die Nachbarschaft jedes Bildpunktes zu definieren.

Gegeben sei jetzt eine Menge von lokalen $\mathbf{r}_i$, in dem die Farben durch den Benutzer spezifiziert werden: $u(\mathbf{r}_i) = u_i$, $v(\mathbf{r}_i) = v_i$, und J(U), J(V) sollen zu diesen Begrenzungen minimiert werden.

Da die Kostenfunktionen quadratisch sind und die Begrenzungen linear, erzeugt dieses Optimierungsproblem ein System linearer Gleichungen, die mit Standardmethoden gelöst werden können.

Dieser Algorithmus ähnelt Algorithmen, die für andere Aufgaben bei der Bildbearbeitung verwendet werden. In Bildsegmentierungsalgorithmen beispielsweise, die auf normalisierten Schnitten basieren, versucht man, den zweiten kleinsten Eigenvektor der Matrix D-W zu finden, in der W eine *npixel x npixel* Matrix ist, deren Elemente die paarweisen Affinitäten zwischen den Bildpunkten darstellen (d.h., die Eingabe **r**, **s** der Matrix ist $w_{\mathbf{rs}}$). D ist hierbei eine diagonale Matrix, deren Diagonal-Elemente die Summe der Affinitäten sind (in unserem Fall ist dies immer 1).

Der „zweite“ kleinste Eigenvektor jeder symmetrischen Matrix A ist ein Einheitsvektor x, der xAx minimiert und zum ersten Eigenvektor orthogonal ist. Die

quadratische Form, die durch normalisierte Schnitte minimiert wird, ist gerade die Kostenfunktion J, d.h. $x^T(D-W)x = J(x)$. So minimiert dieser Algorithmus die gleiche Kostenfunktion, aber zu unterschiedlichen Begrenzungen.

In der Praxis erweist sich das ganze natürlich trotz Software-Unterstützung noch als recht mühselig. So ist es ein nicht-triviales logistisches Problem, dass gleiche Leute in einem Film immer die gleiche Haarfarbe, gleiche Augenfarbe und Wände in einem Raum, welcher z.B. nur am Anfang und am Ende in einem Film vorkommt, gleiche Farben zugewiesen bekommen. Außerdem ist der einfärbende Mensch auch ein stückweit kreativ als Künstler gefordert, denn die Farben sollen ja auch einem gewissen ästhetischen Anspruch genügen.
Technisch ist auch eine Menge zu machen: Es muss der Film zunächst digitalisiert werden, dann müssen Bild und Ton getrennt werden, der Video-Anteil muss in einzelne Bitmap-Frames zerlegt werden. Die Bitmaps werden dann Szenenweise wie beschrieben eingefärbt, die eingefärbten Szenen müssen wieder zusammengesetzt und am Schluss muss der Ton wieder lippensynchron hinzu gefügt werden. Die aufwändigen mathematischen Berechnungen dauern auf einem PC recht lange: Um 1 Sekunde eines Film einzufärben, braucht es je nach Rechnerleistung zwischen 30 Min. bis zu 2 Stunden Rechenzeit! Dabei ist die Handarbeit des Farbezuordnens nicht mitgerechnet. Und natürlich auch nicht die des Zerlegens des Filmes in einzelne Szenen und das wieder Zusammensetzen.
Dann ist da noch das urheberechtliche Problem: Farbe einem Film hinzuzufügen gilt als künstlerischer Eingriff am Werk und ist daher nicht ohne weiteres zulässig. Für private und wissenschaftliche Zwecke dürfen „kleine Teile“ ohne besondere Genehmigung nachkoloriert werden. Diese Filmteile dürfen aber anschließend ohne Genehmigung der Urheber nicht veröffentlicht oder gar gesendet werden. Sie dürfen allenfalls nur zu „Demonstrationszwecken der benutzten Techniken“ und nicht etwa zum Vergnügen angeschaut werden (!).
Das kann man auch verstehen, wenn man so schöne alte Filme wie etwa „Casablanca“ anschaut: Das Schwarzweiß gehört einfach dazu. Es kann daher schon in gewissem Sinne unästhetisch sein, so was nachträglich einzufärben.

Lösungen der Übungen:

Kapitel 2:

Es sind zu berechnen:

Anzahl	Was	Wie lange
10	Einfacher Videoclip	Je 5 Min.
10	Audio (gekauft)	Je 5 Min.
4	Animation mit 5 Extremitäten	Je 10 Sek.
3	Photo	
1	Grafik	

Das ergibt

Video:

$$10 \cdot s_4 = 10 \cdot \gamma \cdot \frac{t}{\omega} = 10 \cdot 1 \cdot \frac{5}{10} = 5 \text{ Personentage}$$

Audio:

$$10 \cdot s_1 = 10 \cdot \gamma \cdot \frac{t}{\omega} = 10 \cdot 1 \cdot \frac{5}{30} = 1{,}67 \text{ Personentage}$$

Animation:

$$4 \cdot s_2 = 4 \cdot ß \cdot t \cdot \gamma \cdot (\sum_{i=1}^{p} \psi_i)^2 = 4 \cdot 0{,}167 \cdot 1 \cdot (\sum_{i=1}^{1} 5)^2 = 16{,}7 \text{ Personentage}$$

Photo:

$$3 \cdot s_5 = 3 \cdot \frac{m}{2} = 3 \cdot \frac{1}{2} = 1{,}5 \text{ Personentage}$$

Grafik:

$$1 \cdot s_3 = 1 \cdot \frac{g}{2} = 1 \cdot \frac{1}{2} = 0{,}5 \text{ Personentage}$$

Gesamtaufwand: Mind. 25,37 Personentage (nur für die Einzelkomponenten, d.h. ohne Planungszeit und Kompositionszeit der Komponenten)

Kapitel 3:

1. Man verwendet z.B. den nachfolgenden Algorithmus:
   - definiere ein Zeichen (Kontrollzeichen Bsp. **X**), was kaum vorkommt
   - schreibe alle Zeichen der Originaldaten in gleicher Reihenfolge auf, solange keines zwei Mal hintereinander vorkommt, wenn doch, dann:
     - Schreibe das Kontrollzeichen, gefolgt von der Anzahl der Wiederholungen des betreffenden Zeichens und anschließend das Zeichen selbst
   - Kommt das Kontrollzeichen (**X**) selbst vor so schreibe: **X**[Anz.Wdhlg]**X**

   Dies ergibt den Ergebnis-Datenstrom RLE:
   3 **X**4 5 7 **X**3 8 **X**3 7 **X**6 4 **X**1 X **X**2 9

2. a) Hier funktioniert MPEG2 optimal, weil die P-Frames fast identisch mit den I-Frames sind und so kaum Abweichungen abgespeichert werden, da hohe zeitliche und räumliche Bildredundanz vorliegen
   b) Hier kommt MPEG2 an seine Grenzen: Da kaum Redundanz vorhanden ist, wird die Bildfolge entweder „ruckelig“ aussehen und/oder mit „Artefakten“ (kleine „Klötzchen“) übersäht sein
   c) Reine Rotationen und/oder Zoomen werden durch die Vektorisierung der P-Frames relativ gut abgefangen und auch die interpolatorischen B-Frames können das gut darstellen.

3. Das Problem besteht darin, dass die Rasterdaten nur als Datei (im Ganzen) für eine Software erkennbar sind. Es gibt also keine einzeln adressierbaren Objekte. Einer automatischen Vektorisierung muss es gelingen, das Rasterbild fachgerecht zu interpretieren, aus der Ansammlung von Pixel Objekte zu erkennen und diese Objekte dann in DV-technisch adressierbare Objekte im Vektorformat umzuformen. Dahinter verbirgt sich also das Problem der Objekterkennung (vgl. Kapitel 4). Eine Möglichkeit wäre, die Konturen eines Objekts in der Rastergrafik zu bestimmen (z.B. durch einen Gradientenoperator), und dem Pixel der Konturen jeweils einen Vektor zuzuordnen. Wird eine solche Grafik dann z.B. vergrößert, werden die Koordinaten einfach passend umgerechnet und entstehende „Leerstellen“ zwischen den neuen Vektorendpunkten geeignet interpoliert.

Kapitel 4:

1. Es handelt sich um den sog. „Mittelwert-Operator“:

1	2	3	1	1	...
1	1	2	2	1	
2	3	10	2	2	
1	1	3	3	1	
1	1	2	1	3	
...					

•

1	1	1
1	1	1
1	1	1

→

0	0	0	0	...
0	25	26	24	
0	24	27	26	
0	24	26	27	
...				

2. Algorithmus: Wähle einen Schwellwert „s“:

```
For i to AnzPixelBildbreite
    For j to AnzahlPixelBildhöhe
        If
            Imagepixel(i,j)<s then Imagepixel(i,j)=0
        Else
            Imagepixel(i,j)=255
```

3. Für jeden Grauwert $g \in [g_{\min}, g_{\max}]$ des Originalbildes wird auf ein $G \in [0,255]$ linear abgebildet durch die Formel:

$$G(g) = 255 \bullet \frac{g - g_{\min}}{g_{\max} - g_{\min}}$$

Kapitel 5:

Bei 8 kHz und 8 Bit (=1 Byte) ergibt sich ein Datenstrom von ca. 8 kB/s. Damit wäre die Wave-Datei ca. 8 kB groß.
Ein MIDI-Event benötigt durchschnittlich 3 Byte, so dass in 8 kB ca. 2600 Events „passen“. Bei 120 Events pro Minute würde man ca. 21 Minuten = 1260 Sekunden in einer Datei der Größe 8kB unterbekommen.

Kapitel 6:

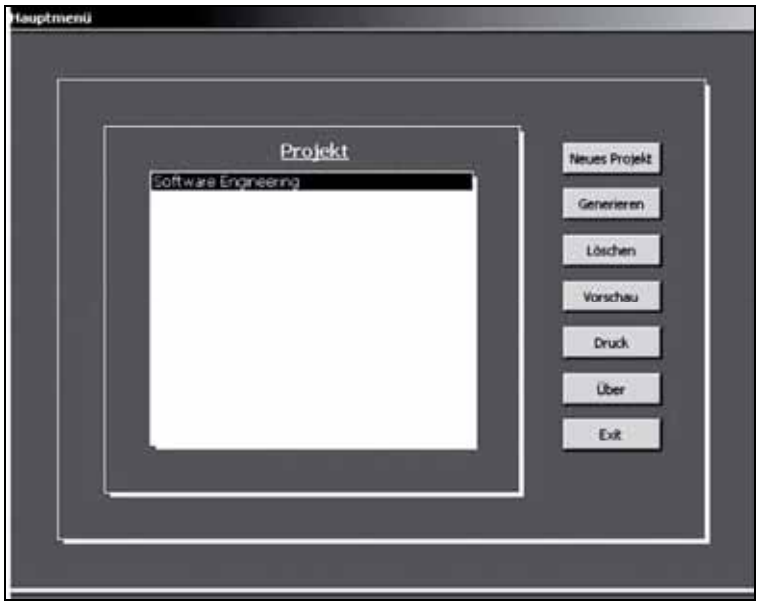
Eine „3D-Pipeline“ erzeugt aus einer dreidimensionalen Umgebung eine zweidimensionale Darstellung und regelt die Zusammenarbeit zwischen CPU und Grafiksystem. Zu Beginn eines Renderingvorgangs steht immer die Applikation

(in den meisten Fällen ein Spiel), die man -neben dem User- auch als oberste Instanz betrachten kann. Zusätzlich gibt die Engine noch andere Anweisungen an die Hardware weiter, die dem Benutzer meist verborgen bleiben (z.B. „Objekt X befindet sich sehr weit vom Betrachter entfernt = Setze Detailgrad für Objekt X auf tief um Rechenzeit zu sparen“ usw.). Bereiche wie Physikberechnungen oder die Kollisionsabfrage, werden ebenfalls von der Engine gesteuert. Dabei passieren in dieser Reihenfolge nachfolgende Vorgänge:

- **Tesselation**
- **Transformation**
- **Lighting**
- **Clipping**
- **Triangle Setup**
- **Rasterisierung**
- **Texture Mapping**
- **Shading**
- **Hidden Surface Removal**
- **Framebuffer**

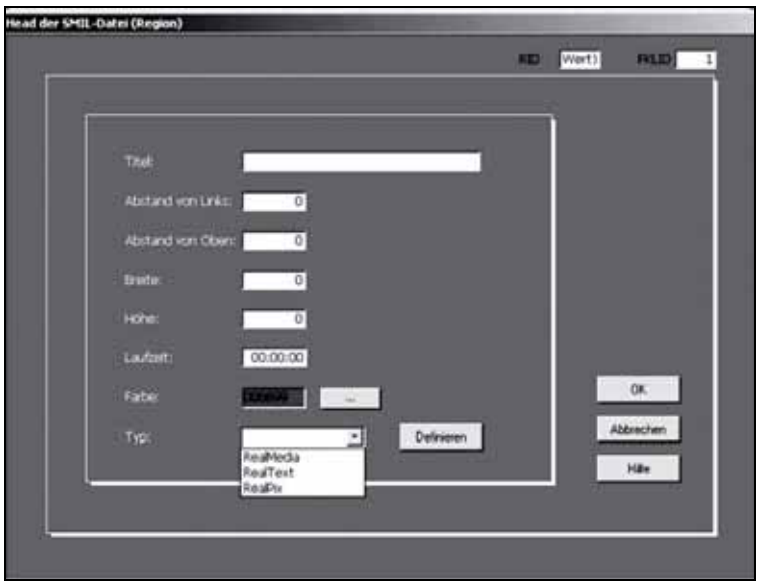
Kapitel 7:

Eine Idee wäre, dass man über eine zu schreibende Software-Anwendung die wichtigsten Parameter über benutzerfreundliche Eingabemasken aufnimmt und dann daraus die SMIL-Skripte zusammensetzt. Dazu kann man z.B. Visual Basic (unter MS-Access) verwenden. Hier ein paar Maskenentwürfe:



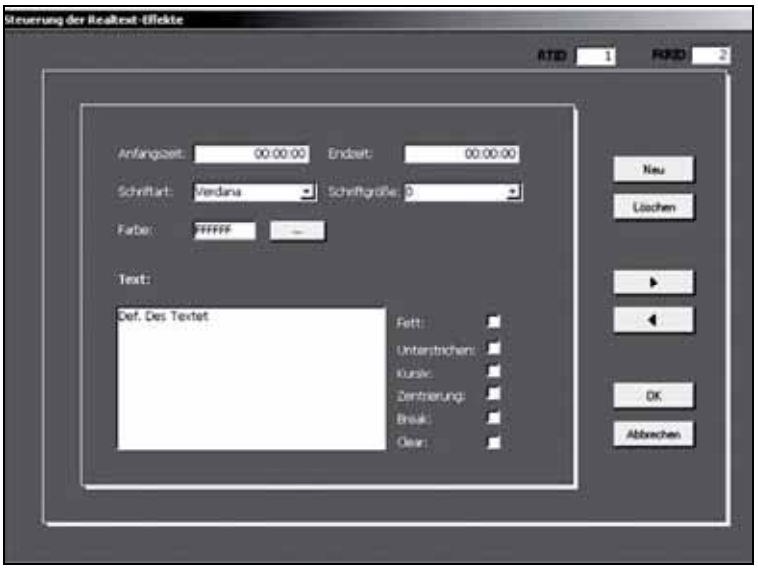
Hauptmenu

Durch Doppelklick auf den Projektnamen kommt man in ein Definitionsfenster für die Positionierung der jeweiligen Elemente:



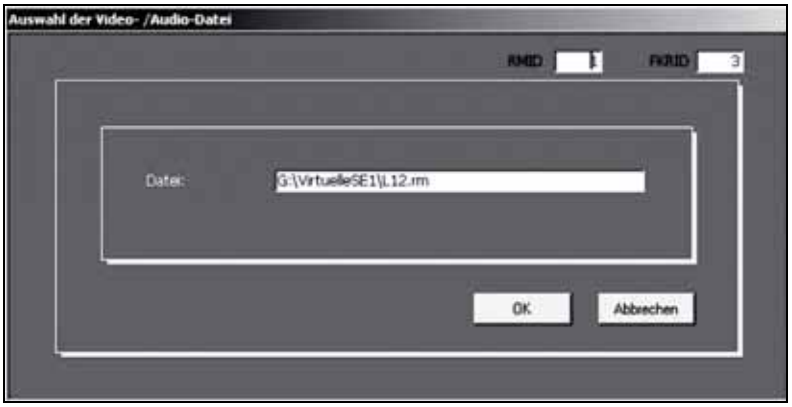
Positionierung

Durch „Definieren“ kann man dann Video, Texte oder Grafikeinblendungen festlegen:



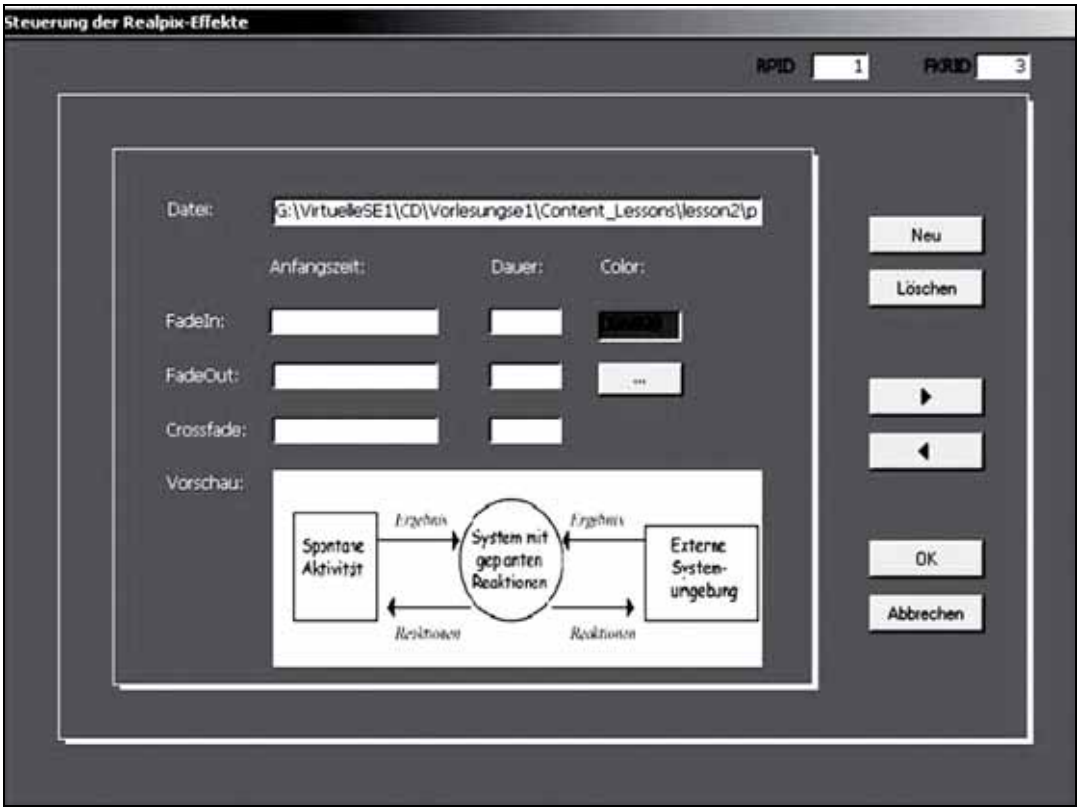
Ein- und Ausblenden z.B. eines Textes

Das Hauptvideo wird hier festgelegt:



Angabe des Ortes eines Video-Clips

Die Grafiken können hier definiert werden:



Grafiken festlegen

Wenn man dann auf „Generieren“ im Hauptmenu drück, so wird daraus das SMIL-Skript für eine einfache HTML-Seite generiert, die alle angegebenen Elemente enthält, die dann an den gewünschten Stellen während des Video-Clips ein- und ausgeblendet werden.

Index:

In der gleichen Reihe erschienen :

Peter Zöller-Greer

Software-Architekturen

Grundlagen und Anwendungen

Mit einer Einführung in
Architekturbeschreibungssprachen (ADLs)
UML, Object-Z, OCL, CORBA, IDL
Entwurfsmuster, Architektursichten
Architekturmuster, DDD, Architektur-Dokumentation
Komplexitätsprobleme, Standard-Architekturen
SOA, TOGAF, RM-ODP
Software Factories

Reihe *Wissen & Praxis >kompakt<*

Composia Verlag

Peter Zöller-Greer

Software - Architekturen

sind Baupläne für Software. Es gibt -wie im Baugewerbe- verschiedene (An-)Sichten für verschiedene Projektbeteiligte. Jede Ansicht beherbergt eigene Darstellungsbeschreibungen. Die wichtigsten werden im vorliegenden Werk behandelt.

In der gleichen Reihe erschienen:

Über 1 Jahr
fast ununterbrochen
No. 1 - Bestseller bei amazon.de
z.T. in 3 Kategorien
gleichzeitig!

Prof. Dr. rer. nat. Peter Zöller-Greer
ist Mathematiker und unterrichtet die Fächer Künstliche Intelligenz, Software-Engineering und Multi Media Systeme an der FH Frankfurt am Main-University of Applied Sciences.

ISBN 978-3-9811639-3-3

Die Reihe *Wissen & Praxis >kompakt<* nimmt sich komplexer Themen an und versucht diese so einfach wie möglich, beschränkt auf das Wesentliche, für Studium und Praxis gleichermaßen geeignet darzustellen.

Composia Verlag