

# 3. Film- und Videotechnik und digitale Videobearbeitung

- 3.1 Klassische Filmtechnik
- 3.2 Analoge TV- und Videotechnik
- 3.3 Digitale Videotechnik 
- 3.4 Digitale Videoproduktion
- 3.5 Programmierung für die Videonachbearbeitung

## Literatur:

Ulrich Schmidt: Digitale Film- und Videotechnik, Fachbuchverlag Leipzig 2002  
Johannes Webers: Handbuch der Film- und Videotechnik, 7.Auflage,  
Franzis-Verlag 2002

# Digitalisierung von Video-Signalen

Audio (CD-Qualität):

16 bit Auflösung, Abtastfrequenz 44,1 kHz

Video:

Für Videomonitoring üblich:

8 oder 10 bit Bildwertauflösung (256 bzw. 1024 Farbwerte)

Bei Filmdigitalisierung höchster Qualität:

14 bit Bildwertauflösung (16384 Farbwerte)

Abtastfrequenz (bei Digitalisierung von „Composite Video“):

Farbträgerfrequenz ca. 4,43 MHz, also min. 10 MHz Abtastung

Zur Vermeidung von Interferenzen besser vierfache Frequenz des Farbträgers, d.h. 17,73 MHz

Bitrate:  $17,73 \text{ MHz} * 8 \text{ bit} = 142 \text{ Mbit/s}$

D.h. ca. 1 GByte/Minute ! ( $17,73 * 60 = 1064$ )

→ Digitale Videosignale stellen höchste Anforderungen an Speicherplatz

# Komponentensignal: Chroma-Subsampling

Video-Komponentensignal:  $Y$ ,  $C_R$ ,  $C_B$

4:4:4

Gleichmässige Abtastung von  $Y$ ,  $C_R$ ,  $C_B$

4:2:2

Bei  $C_R$ ,  $C_B$ : Jedes zweite Pixel

Reduzierte Datenrate:  $2/3$

4:1:1

Bei  $C_R$ ,  $C_B$ : Jedes vierte Pixel

Reduzierte Datenrate:  $1/2$

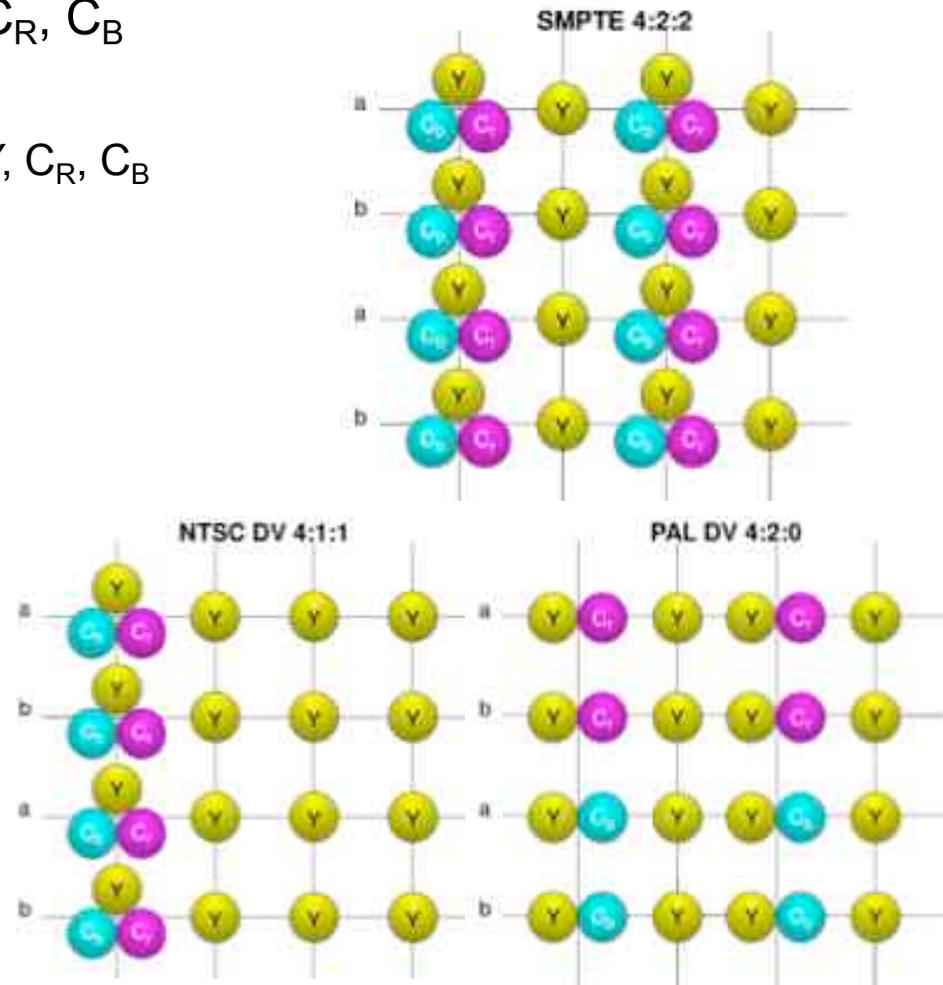
Bei NTSC verbreitet

4:2:0

Bei  $C_R$ ,  $C_B$ : Jedes zweite Pixel,  
abwechselnd  $C_R$  oder  $C_B$

Reduzierte Datenrate:  $1/2$

Bei PAL verbreitet



# Digitales Komponentensignal nach ITU-R 601

Internationaler Standard für digitale Abtastung von Videosignalen  
auch als CCIR-601 bzw. D1 bezeichnet

Systemkompatibel zu:

PAL: 625 Zeilen, 50 Hz Halbbildwechselfrequenz

NTSC: 525 Zeilen, 59,94 Hz Halbbildwechselfrequenz

Abtastfrequenz für Luminanzsignal (Y):

13,5 MHz, d.h. 864 Abtastwerte/Zeile (PAL) bzw. 858 (NTSC)

Berücksichtigung der Austastlücke: 720 Abtastwerte je Zeile  
(*unabhängig vom TV-Standard!*)

Z.B. bei 4:2:2-Chroma-Subsampling:

720 Luminanzwerte + 2 \* 360 Farbwerte je Zeile

576 Bildzeilen (effektiv), d.h. Speicherbedarf je Vollbild 829440 Samples

Datenrate (umfasst auch Daten der Austastlücke):

$13,5 \text{ MHz} * 2 * \text{Samplegrösse}$ , d.h. 216 Mbit/s bei 8 Bit Bildwertauflösung

D.h. ca. 1,3 GByte/Minute !

Bei 4:1:1- oder 4:2:0-Subsampling: 162 Mbit/s

Physikalische Schnittstellen (ITU-R 656):

parallel oder seriell (Serial Digital Interface SDI)

# High-Definition Video - Digital

Hochqualitatives Videosignal:

- Höhere Zeilenzahl (effektiv 1080)

- Höhere Abtastrate (74,25 MHz nach ITU-R 709)

- 1920 Samples/Zeile

- Bildformat 16:9

- Bildwertauflösung 10 bit

Datenrate 1,485 Gbit/s

Digitale Filmproduktion:

- Abtastung mit 2048 oder 4096 Samples/Zeile („2k“ und „4k“)

- Z.B. „MPEG 4 Studio Profile“:

  - Bis zu 4096 x 4096 Pixel

  - Auch 4:4:4 Sampling von RGB-Signalen

  - Datenraten bis zu 2,4 Gbit/s

# Weitere Video-Datenreduktion

## Intraframe-Codierung:

Anwendung der Diskreten Cosinus-Transformation (DCT)  
analog zu JPEG

In vielen Video-Standards verwendet (z.B. in DV = Digital Video)

## Interframe-Codierung:

Basiert auf Prädiktionsverfahren (z.B. Bewegungskompensation)

Differential-Codierung (Differenz tatsächliches Bild - vorhergesagtes Bild)

MPEG-Standard-Familie (derzeit v.a. MPEG-2 und MPEG-4)

Zunehmende Verbreitung als Video-Standard

Problematisch beim digitalen Videoschnitt

# Professionelle Video-Bandgeräte



DVCAM-Recorder



Digital Betacam Recorder

# Digitale Video-Bandaufzeichnung

Digitale Komponenten-Signal-Aufzeichnung (unkomprimiert):

D1-Standard (1985)

Digitales Komponenten-Signal nach ITU-R 601 (227 Mbit/s),  
8 bit Samples

Chroma-Subsampling 4:2:2, sonst unkomprimiert

Diagonale Bandaufzeichnung mit schmalen Spuren

Ähnliches Format mit 10 bit Samples: D5

Digitale Komposit-Signal-Aufzeichnung (unkomprimiert):

D2- und D3-Formate, heute fast bedeutungslos

Digitale Komponenten-Signal-Aufzeichnung (komprimiert) - Beispiele:

Digital Betacam

DCT-Kompression 2:1 (124 Mbit/s)

Digital Video (DV)

DCT-Kompression 5:1 und 4:2:0 Subsampling (25 Mbit/s)  
(d.h. 190 MByte/Minute)

DVCPRO 50

DCT-Kompression 3,3:1 und 4:2:2 Subsampling (50 Mbit/s)

# Trend im professionellen Bereich: MPEG

Z.B. Sony IMX-System

Kompatibel zu MPEG-2 und MPEG-4

Wichtig für Digital Video Broadcast (DVB) und DVD-Video

4:2:2 Subsampling

Reiner I-Frame MPEG-Strom

Damit geeignet für Videoschnitt

Datenrate 50 Mbit/s

Abspielgeräte

(„Multi-Format-  
Player“)

kompatibel mit  
Betacam

„D10“-Standard  
= MPEG-2



# Heimbereich wird semiprofessionell: DV-Standard

Vielzahl analoger Standards für den Heimbereich:  
VHS, S-VHS, VHS-C, Video-8, Hi8

Digitale Standards für den Heimbereich:  
Digital8, DV, ...

## DV

Familie von Formaten mit semiprofessioneller Qualität

Starke DCT-Kompression (5:1) - 25 Mbit/s

Einfach über IEEE 1394  
(FireWire, iLink) übertragbar („DV in/out“)

6,3 mm breite Bänder

Zwei Kassettengrößen  
(Standard und mini)

Professionelle Varianten:  
DVCPRO und DVCAM

Höhere Qualität: DVCPRO 50  
(doppelte Datenrate: 50 Mbit/s)

Heimbereich-DV („miniDV“):

Kleine Kassetten

Fast sendereife Qualität



# Trends im Heimbereich

Direkte Aufzeichnung auf DVD

Direkte Aufzeichnung im MPEG-Format

    Oft auf Flash-Speicherkarten

HD-Auflösung



# Digital ↔ Analog - Wandlung

A → D: Digitalisierung analoger Video-Quellen

Bei Weiterverarbeitung analog vorliegenden Materials (z.B. Videobänder)  
Hardware-Lösungen (z.B. auf Video-Schnittkarte oder „Break-Out-Box“)

A → D: Filmabtastung

Scannen von Filmmaterial

Punktweise („flying spot“), zeilenweise oder bildweise  
Spezialgeräte (z.B. „Telecine“)

D → A: Analoges Rendering digitaler Quellen

Z.B. zur Ausgabe auf TV-Monitor, Aufnahme auf Analog-Videoband,  
Belichten von Film

Hochwertige Lösung: Laserbelichter (z.B. „Arrilaser“)

Einfache Möglichkeit zur Digital↔Analog-Wandlung:

Digitaler Camcorder  
mit analogem und  
digitalem  
Ein-/Ausgang



# Digitales Kino

Literatur: DCI Specification ([www.dcimovies.com](http://www.dcimovies.com), 2008)

- Digitale Filmkameras seit ~ 2000 verfügbar
- Digitale Postproduktion weit verbreitet
  - Teilweise seit den 1980ern („The Abyss“)
  - Komplette digitale Postproduktion seit ~2000 (+ „Toy Story“, 1995)



RED ONE, 2007

- ⇒ nächster logischer Schritt: Digitale Distribution und Projektion
- Hindernisse:
  - Henne-Ei-Problem
  - Technik veraltet schnell
  - Sicherheit (Raubkopien, DRM)
  - Zuverlässigkeit
  - Wer zahlt?  
(Kostenvorteile für die einzelnen Verleiher, Ausgaben für Kinos)
  - Wer macht's? (Distributionsinfrastruktur, Abrechnung)

# Produktions- und Distributionskette

Filmproduktion → Verleih → Kinobetreiber

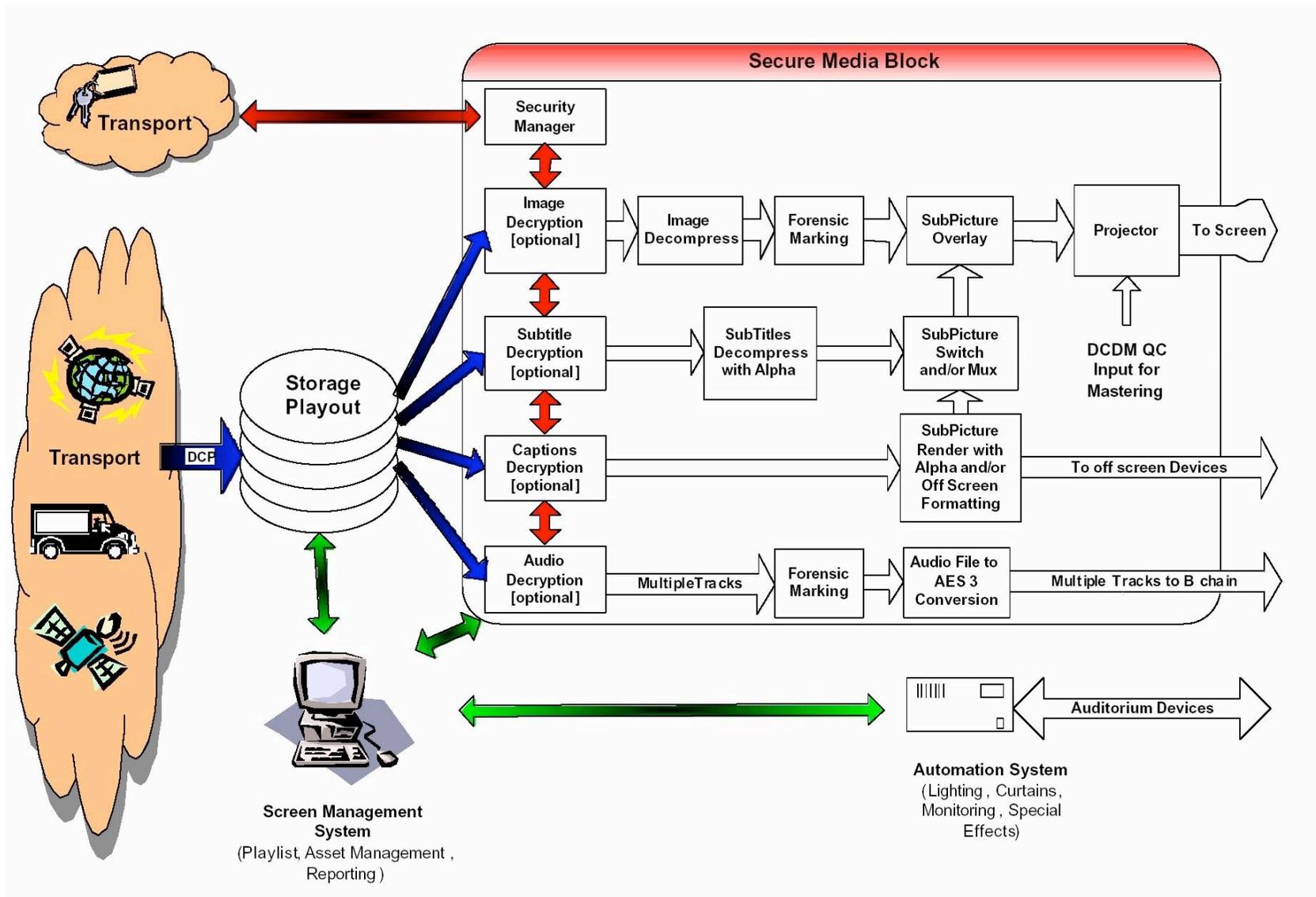
Filmproduktion → Verleih → **Intermediär** → Kinobetreiber

Produktion des Films	Vorfinanzierung	Betrieb Playout-Center
	Promotion	Betrieb Datennetze
	Lokalisierung	Vermietung / Wartung der Kinotechnik
	klassisch:	Abrechnung
	Kopienerstellung	
	Kopienversand	

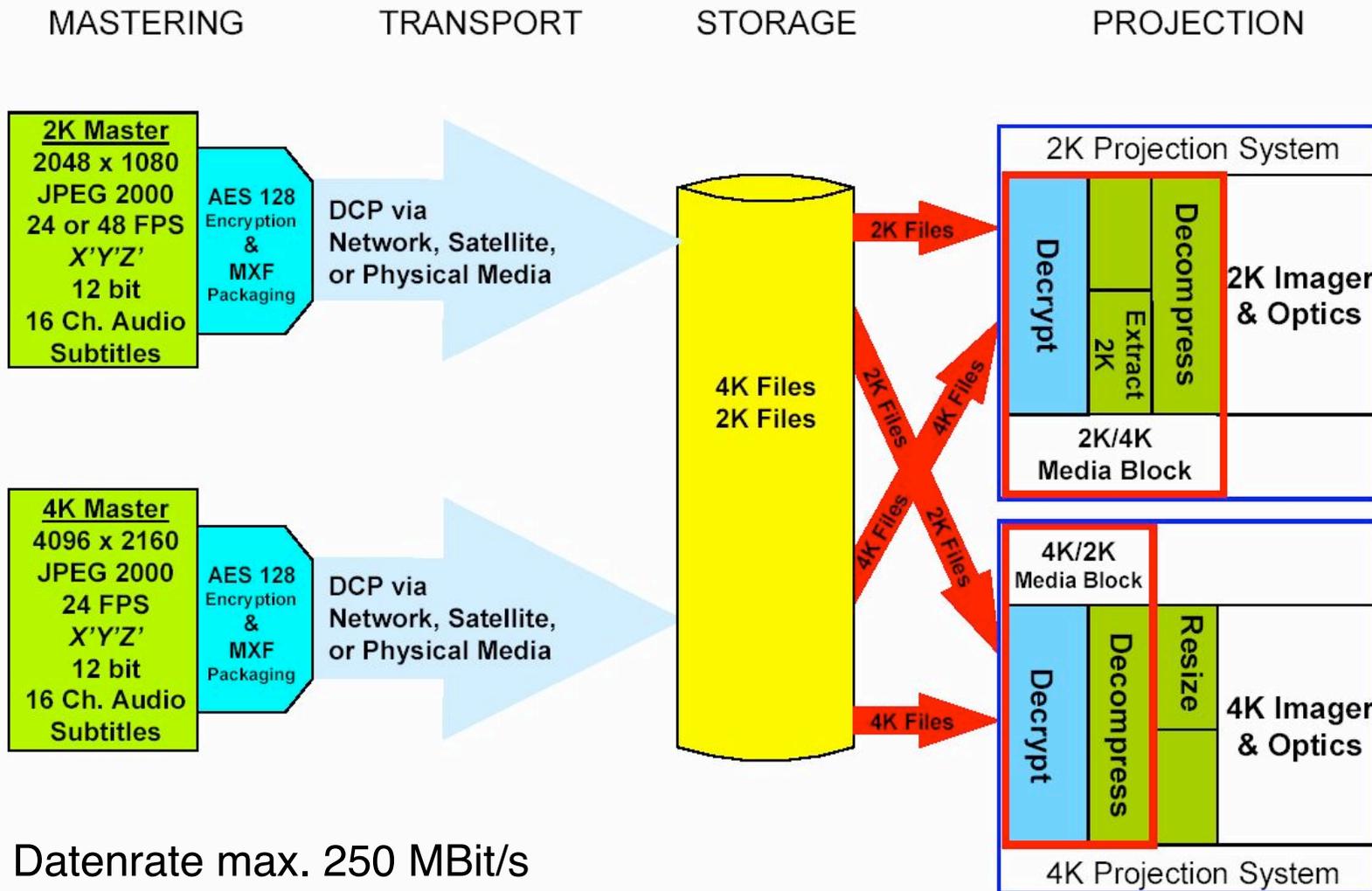
# DCI-Spezifikation

- Von der Digital Cinema Initiative (DCI) entworfen: DCSM -> DCDM
  - Disney, Fox, (MGM), Paramount, Sony, Universal, Warner Bros
- Umfassender Standard einer Vertriebs- und Produktionskette für Digitales Kino
- Letzte Version: 1.2 (März 2008, Errata März 2009)
- Baut auf existierenden Standards auf:
  - MXF (Container)
  - AES (Verschlüsselung)
  - HMAC-SHA-1 (Signatur)
  - JPEG 2000 (Bildkompression)
- Kinobetreiber-freundlich
  - “control lightly, audit tightly”
  - Technik soll nicht erneuert werden müssen
  - Hohe Stabilität (Offline-Betrieb, Fehlertoleranz)

# Distributionskette und Secure Media Block



# Digital Cinema System Workflow



# 3. Film- und Videotechnik und digitale Videobearbeitung

3.1 Klassische Filmtechnik

3.2 Analoge TV- und Videotechnik

3.3 Digitale Videotechnik

3.4 Digitale Videoproduktion

Planung



Aufnahme

Gestalterische Prinzipien der Filmmontage

Technik des digitalen Filmschnitts

3.5 Programmierung für die Videonachbearbeitung

Literatur:

T. Petrasch, J. Zinke: Einführung in die Videofilmproduktion,  
Fachbuchverlag Leipzig 2003

A.H. Müller: Geheimnisse der Filmgestaltung, Schiele&Schön 2003

A. Rogge: Die Videoschnitt-Schule, Galileo Design 2005

# Schritte bei der (digitalen) Filmproduktion

## Produktionsplanung:

Storyboard, Drehbuch, Kalkulation, Casting, ...

## Aufnahme:

Studio/Außenaufnahmen

Selten in Abspielreihenfolge (Wetter, Schauspieler)

## *Postproduktion:*

Filmschnitt (meist digital gesteuert)

### *Compositing*

(traditionell im Kopierwerk, heute meist digital)

Zusammenkopieren von Filmmaterial aus  
verschiedenen Quellen

Farbnachbearbeitung

Tricks und Animation

Heute fast ausschließlich digital

# Beispiel: Storyboard

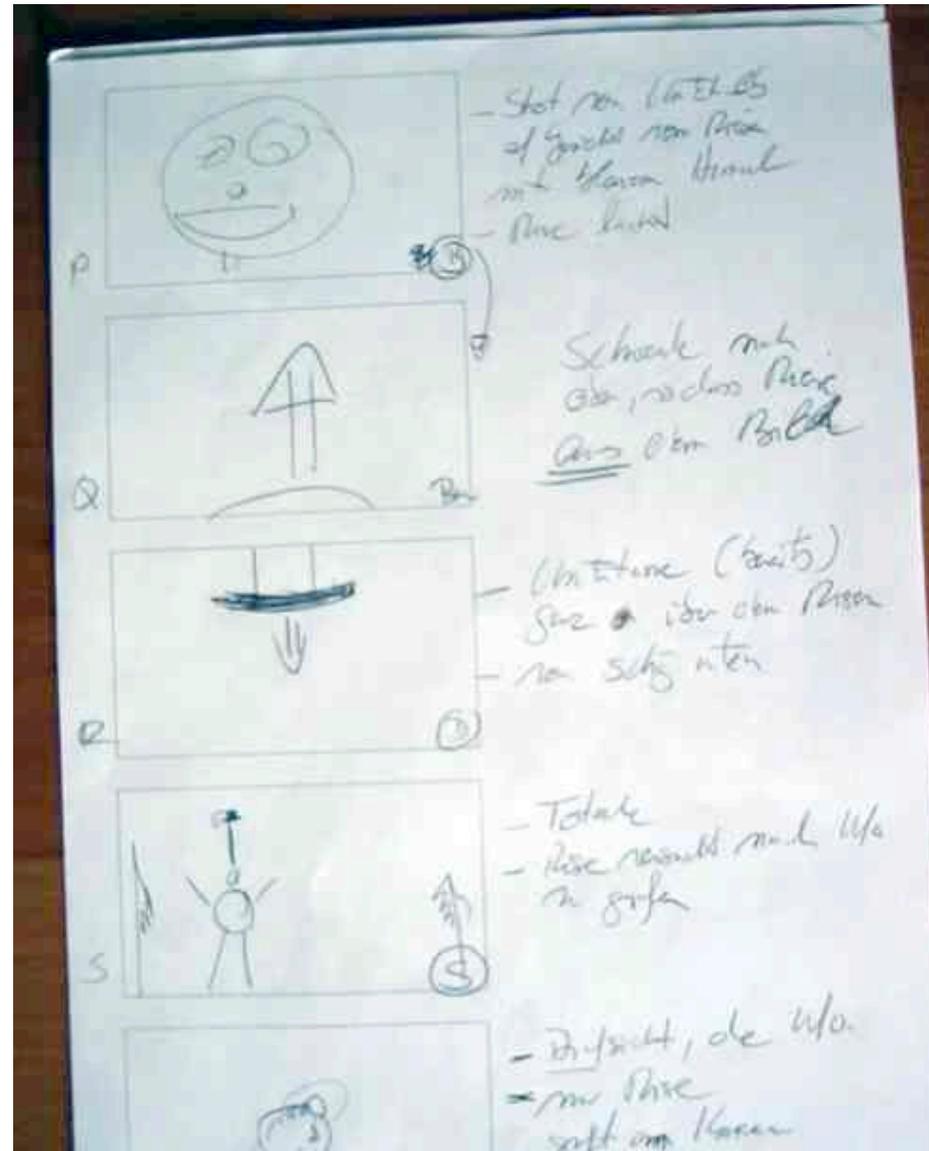
Aus dem Projekt „Riesen im Englischen Garten“ 2003  
(Raphael Wimmer et al.)

Wichtigste Szenen als  
Zeichnungen

Blickwinkel, Kamera-  
perspektive, Bewegungen

Veränderungen durch Pfeile  
beschrieben

Strichmännchen oder  
Scribbling oder  
Animationsprogramm  
(z.B. *Poser*)



# Exposé, Treatment, Drehbuch

## Exposé (*outline*):

Kurze Inhaltsbeschreibung der Filmidee

Wenige Seiten Text, Angaben zu Genre, Dauer etc.

## Treatment:

Angabe von einzelnen Komplexen und Sequenzen (mit grober Dauer)

Z.B 50 Seiten für einen Spielfilm

## Drehbuch:

Fertiger Film mit Worten beschrieben

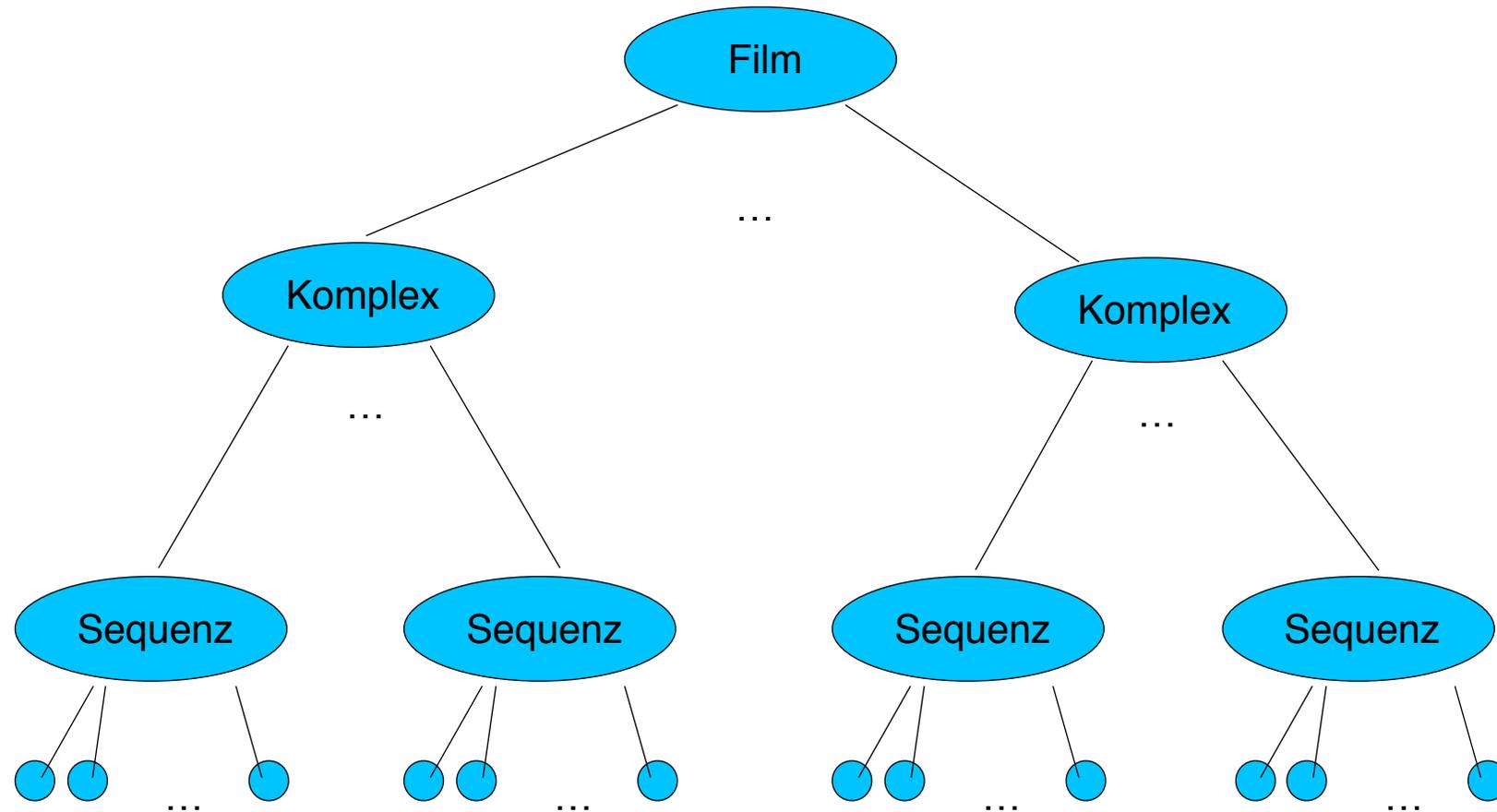
Aufteilung in linke Hälfte (Bild) und rechte Hälfte (Ton)

Aufteilung in Akte analog Schauspiel bei Spielfilmen

Basis für Produktionstagebuch

Abhaken bereits gedrehter Einstellungen

# Struktur eines Films



Einstellungen

Sequenz = kleinste dramatische Einheit

### 3. Film- und Videotechnik und digitale Videobearbeitung

3.1 Klassische Filmtechnik

3.2 Analoge TV- und Videotechnik

3.3 Digitale Videotechnik

3.4 Digitale Videoproduktion

Planung

Aufnahme



Gestalterische Prinzipien der Filmmontage

Technik des digitalen Filmschnitts

3.5 Programmierung für die Videonachbearbeitung

Literatur:

T. Petrasch, J. Zinke: Einführung in die Videofilmproduktion,  
Fachbuchverlag Leipzig 2003

A.H. Müller: Geheimnisse der Filmgestaltung, Schiele&Schön 2003

A. Rogge: Die Videoschnitt-Schule, Galileo Design 2005

# Aufnahme: Checkliste

Material für die spätere Montage bereitstellen

Lieber zu viel als zu wenig

Bildausschnitt

Totale, Halbtotale, Halbnahe, Amerikanisch, Nahe, Groß, Detail

Bildkomposition

Bildachsen, Schwerpunkt, statisch/dynamisch

Verschiedene Kamerapositionen

U.U. mehrere Kameras

Veränderungen der Bildausschnitts

Schwenk

Ruhig, am besten mit Stativ, evtl. „Steadycam“

Ruhephase am Anfang und Ende

Zoom

selten: „Ausrufzeichen der Bildsprache“ (P. Kerstan)

Aufzieher = Zoom + Schwenk

Kamerafahrt

# Zoom vs. Kamerafahrt

## Kamerafahrt:

Objektivbrennweite bleibt gleich

Abstand zum Objekt verändert sich

Veränderung der Größenverhältnisse:

Vordergrundmotiv stärker vergrößert als Hintergrund

## Zoom:

Objektivbrennweite verändert sich

Abstand zum Objekt bleibt gleich

Ähnlicher Effekt zur Ausschnittvergrößerung

Jedoch: Zusätzlich Veränderung der Schärfentiefe

Zoom wirkt generell unnatürlicher

# Aufnahme-Kontinuität

Gedrehte Einstellungen müssen später nahtlos kombinierbar sein

Lichtverhältnisse

Position von Darstellern und Objekten

Details von Darstellern und Objekten

    Kleidung, Accessoires

    Herumliegende Objekte

    Hintergrund

Hintergrundgeräusche

...

### 3. Film- und Videotechnik und digitale Videobearbeitung

3.1 Klassische Filmtechnik

3.2 Analoge TV- und Videotechnik

3.3 Digitale Videotechnik

3.4 Digitale Videoproduktion

Planung

Aufnahme

Gestalterische Prinzipien der Filmmontage 

Technik des digitalen Filmschnitts

3.5 Programmierung für die Videonachbearbeitung

Literatur:

T. Petrasch, J. Zinke: Einführung in die Videofilmproduktion,  
Fachbuchverlag Leipzig 2003

A.H. Müller: Geheimnisse der Filmgestaltung, Schiele&Schön 2003

A. Rogge: Die Videoschnitt-Schule, Galileo Design 2005

# Der Kuleshov-Effekt

Lev Kuleshov 1919, Moskau: Erste Filmhochschule

Systematische Experimente zur Wirkung von Bildmontage

Subjektive Wahrnehmung eines identischen Gesichts

Abhängig von vorhergehenden Bildern

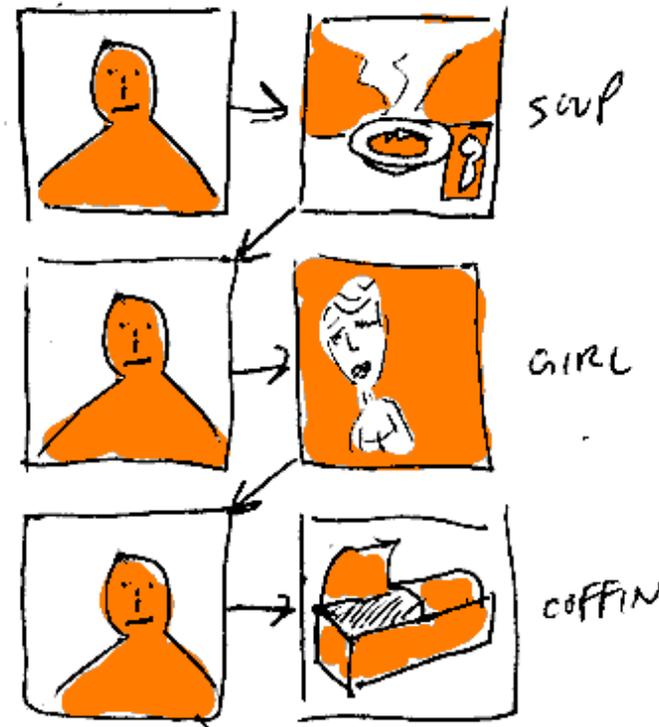
Hunger, Freude, Trauer

Weitere Effekte:

Verschmelzung verschiedener Schauplätze durch Handlung

Verschmelzung von Detailansichten einer Person („die ideale Frau“)

AUDIENCE THINKS THE EXPRESSION CHANGES



BUT THE EXPRESSION REMAINS THE SAME

[www.austinkleon.com](http://www.austinkleon.com)

# Wo der Film entsteht...

Der Film entsteht im Kopf des Zuschauers!

Bewegung und Nahaufnahme werden als wichtig erkannt

Lücken in der Darstellung werden „aufgefüllt“

Z.B. A. Hitchcock: Darstellen durch Nicht-Zeigen

Codes: Zeichen mit vereinbarter/antrainierter Bedeutung

Z.B. Ausgestreckte Beine hinter der Couch

Z.B. Bewegungen zweier nacheinander gezeigter Objekte;

in gleicher Richtung: Verfolgung

In entgegengesetzter Richtung nach innen: Kampf, Konflikt

Tendenz zur Verkürzung von Codes

Z.B. Aufzugfahrt

# Perspektive (1): Erzählperspektive

## Grundperspektiven:

- Auktorialer oder allwissender Erzähler

- Personale Erzählperspektive

  - Oft im Wechsel verschiedener Personen

## Varianten der Ich-Form:

- Auktorialer Ich-Erzähler

  - Prinzipiell widersprüchliche Konstruktion

- Personaler Ich-Erzähler

  - Einschränkung der mitteilbaren Information

## **Perspektive (2): Sichtweisen der Kamera**

### Objektive Kamera:

Neutrale Betrachtung (wie durch Unbeteiligte)

Steter Wechsel des Blickwinkels

Totale, Vogelperspektive

### Subjektive Kamera:

Personale Erzählung

Zuschauer nimmt im Kopf einer Figur Platz

Nahaufnahmen, Durchblicke

## Beispiel (1)

Leere Landschaft, kleines Anwesen am Horizont

Lange Heranfahrt, Stop bei Hauptperson  
(Frau sitzt auf Terrasse)

Flugzeug von außen im Flug

Passagier (männlich)

Aussicht aus dem Fenster

Zurück zur Frau auf der Terrasse:

Was passiert nun?

## Beispiel (2)

Auto von außen

Fahrer frontal durch die Windschutzscheibe

Straße aus Sicht des Fahrers

(a) Ein Hindernis!

(b) Cockpit eines Flugzeugs

# Montage-Techniken

## Klassische Montage-(=Schnitt)-Typen

### Continuity-Schnitt

Fortlaufendes „Erzählen“ einer Geschichte

Vermeiden von drastischen Kontrasten z.B. in Farbe, Schwenkrichtung

### Compilation-Schnitt

Dokumentation, Nachrichten

Zusammenhalt durch Sprecher im „Off“

### Kreuz- oder Wechselschnitt

Laufend wechselnde Darstellung paralleler Geschehnisse

## Progressive Montage:

Vom Allgemeinen (Horizont) mit Übergang (Transit) zum Einzelnen (Fokus)

Total - Halbnahe - Groß - Halbnahe (Reorientierung)

## Regressive Montage:

Vom Einzelnen (Fokus) mit Übergang (Transit) zum Allgemeinen (Horizont)

# Konturenfehler, Kopf-auf-Kopf

Ähnliche Konturen aber verschiedene Objekte/Personen

Nicht direkt schneiden

Gleiche Person, aber verschiedene Position

Nicht direkt schneiden

Klassisches Problem: Interview in Ausschnitten

Lösungen:

Zwischenschnitt (z.B. Publikum) in Bild und Ton

Zwischenschnitt nur im Bild, Ton weiterlaufend

Überblendung im Bild

Schwarz- oder Weissblende

# Erzählkontinuität

## Kontinuität von Raum

Progressive oder regressive Montage

## Kontinuität von Zeit

Nicht vollständig realisierbar

Anschlüsse, Zwischenschnitte

## Kontinuität von Handlung

Anspruchsvoll, wenn Zeit- und Raumkontinuität nicht gegeben

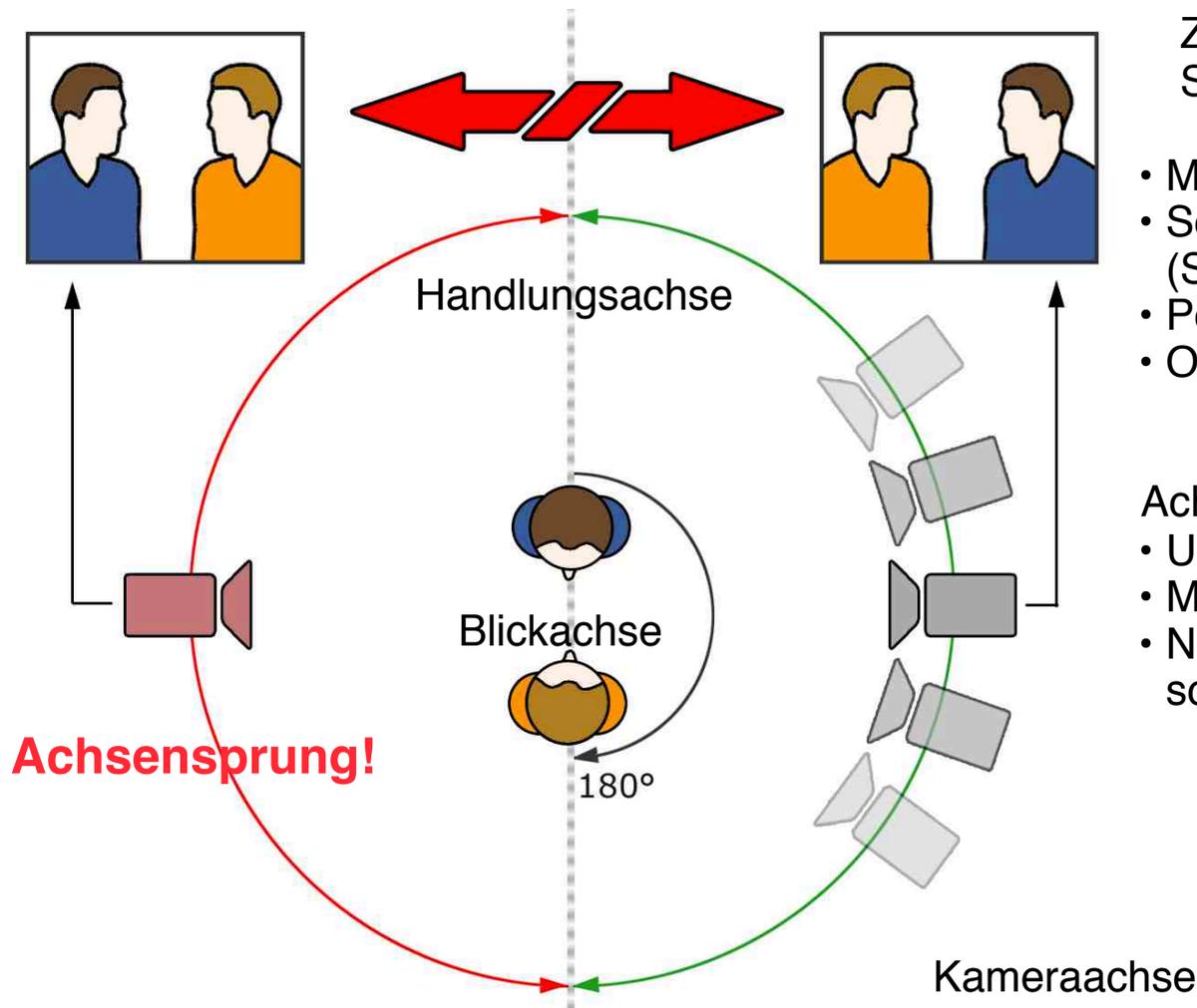
Weitere Stilmittel: Bekleidung, Analogien der Tätigkeit, ...

## Diskontinuitäten:

Stilmittel

Bewusster Einsatz zur klaren Trennung von Sequenzen

# Achsen im filmischen Raum



Zulässige  
Standard-Einstellungen:

- Master shot
- Schuss - Gegenschuss (SRS)
- Point of View (POV)
- Over-Shoulder

Achsenwechsel:

- Umfahrt
- Mitgehen
- Neutraler Zwischenschnitt (*cut-away*)

# 3. Film- und Videotechnik und digitale Videobearbeitung

3.1 Klassische Filmtechnik

3.2 Analoge TV- und Videotechnik

3.3 Digitale Videotechnik

3.4 Digitale Videoproduktion

Planung

Aufnahme

Gestalterische Prinzipien der Filmmontage

Technik des digitalen Filmschnitts 

3.5 Programmierung für die Videonachbearbeitung

Literatur:

T. Petrasch, J. Zinke: Einführung in die Videofilmproduktion,  
Fachbuchverlag Leipzig 2003

A.H. Müller: Geheimnisse der Filmgestaltung, Schiele&Schön 2003

A. Rogge: Die Videoschnitt-Schule, Galileo Design 2005

# Klassifikation von Schnitt-Techniken

Linear - nichtlinear:

- *Linearer* Schnitt: Kopieren von Material-Sequenzen auf eine „Master-Kopie“  
(Digitale) Steuerung von Geräten durch Timecode-Sequenzen  
Änderungen bereits kopierter Sequenzen kaum möglich
- *Nichtlinearer Schnitt (non-linear editing)*:  
Zusammenstellung von *Referenzen* auf in das Schnittsystem digital  
*eingeladenes* (importiertes) Material  
Wesentlich flexibler für nachträgliche Änderungen im Master

Online - Offline:

- *Online* = direkte Bearbeitung des hochqualitativen Videomaterials
- *Offline* = Festlegung der Schnittentscheidungen anhand Darstellung in  
niedrigerer Bildqualität

Destruktiv - nicht-destruktiv:

- *Destruktiv* = Manipulationen an Original-Material irreversibel

Echtzeit:

Direkte Beurteilung des Ergebnisses bei Mischung, Effekten etc.

# Konventioneller Film- und Videoschnitt

## Hauptfunktionen des Schnitt-Arbeitsplatzes:

Wiedergabe verschiedener Ausschnitte vorhandenen Materials (Bild und Ton)

Beurteilung der Wirkung verschiedener Kombinationen

Erstellung einer „Schnittkopie“ für die weitere Bearbeitung im Kopierwerk

„Negativ-Schnitt“ vollzieht den an Positiv-Kopien entwickelten Schnitt am Negativ nach und erstellt den Rohfilm („Null-Kopie“) für die Massenkopierung

## Wichtige Kriterien:

Beurteilbarkeit des Ergebnisses

Absolute Synchronizität, vor allem bei Bild/Ton

Die absolute Bildqualität entscheidet sich erst beim Negativschnitt (d.h. „Offline“-Bearbeitung)

Videoschnitt: Etwas einfacher durch magnetische Aufzeichnung

Beim (häufigen) Kopieren entstehen durch analoge Technik sich akkumulierende Fehler

# Überblendungen und AB-Verfahren

Einfache („harte“) Schnitte:

Ein oder mehrere Quellgeräte und ein Zielgerät  
(„Einzelspurverfahren“)

Überblendungen:

Benötigen Bildinformation aus zwei *überlappenden*  
Materialsequenzen

(Ungünstige) Realisierung: Erstellen von Kopien überlappender  
Sequenzen beim Schnitt

Bessere Realisierung: *AB-Verfahren*

Aufteilen des Materials in zwei Rollen (A und B)

Erstellen der endgültigen Überblendung im Kopierwerk  
(vom Original-Negativ)

AB-Verfahren heute noch in professionellen digitalen  
Schnittsystemen zu erkennen!

# Analoger Videoschnittplatz

Kontrollmonitore



Schnittcomputer

Audio-Mischpult

Quelle:  
Uni Koblenz

Trick-Mischpult

Schnittsteuer-Pult

Analoge  
Video-Player und  
-Recorder

Video-  
Switcher  
incl.  
Effektberechnung



# Digitaler Videoschnittplatz 2003



Kontrollmonitore

E/A Geräte

Rechner mit Schnittsoftware

# Arbeitstechnik beim digitalen Videoschnitt

## Bereitstellung des Materials

Importieren von Quellen (Clips, Standbilder, Sound)

Ablage in Datenarchiv

## Rohschnitt

*In-* und *Out-Marken* für Clips zur Bestimmung des relevanten Ausschnitts

## Zusammenstellung

„Montage“ der Bestandteile in ihrem zeitlichen Ablauf

Einfügen von Effekten

## Feinschnitt

Detail-Bearbeitung der Grenzen von Bestandteilen

Einfügen von Übergängen

## Tonmischung

Klassische Audio-Mischung (Mischpult-Metapher)

Unterscheidung: Material mit synchroner Tonspur oder separate Tonquellen

# Grundlegende Interaktionsformen

## Schnittmonitor

- Orientiert sich am klassischen Schnitt-Arbeitsplatz
- Ein oder zwei Monitorfenster
- Steuerung des Schnitts weitgehend mit Tastatur möglich
- Geeignet für harte Schnitte



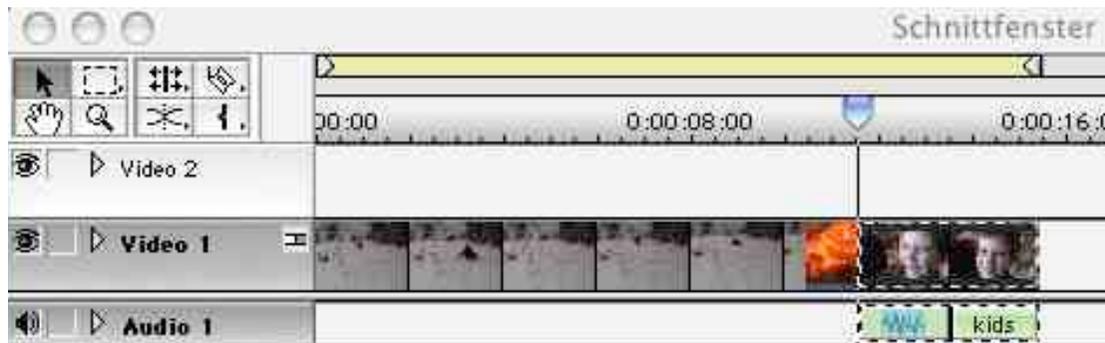
## Zeitlinienorientierter Schnitt

- Darstellung des zeitlichen Verlaufs entlang einer Zeitachse
- Geeignet für komplexe Schnitte mit Überblendungen

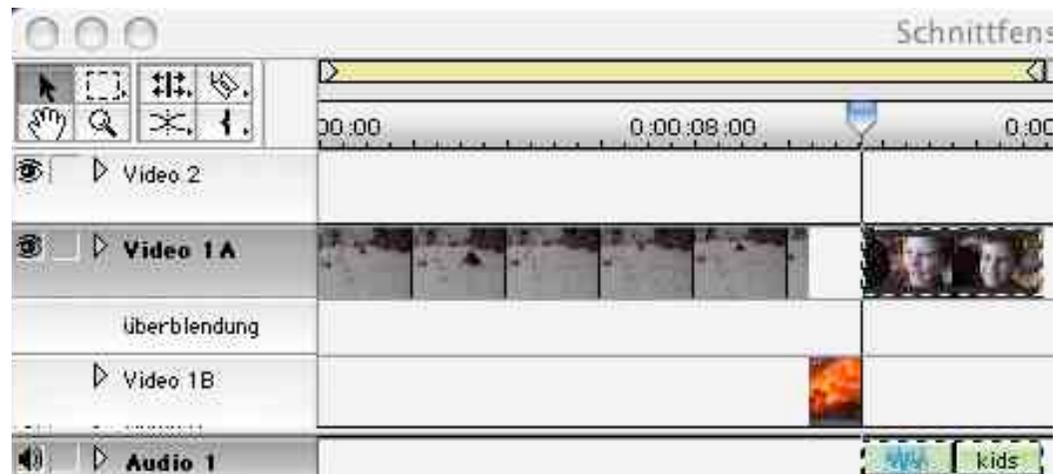


# Einzelspur- und AB-Bearbeitung

Einzelspurbearbeitung



AB-Bearbeitung



# Feinschnitt: Trimmen

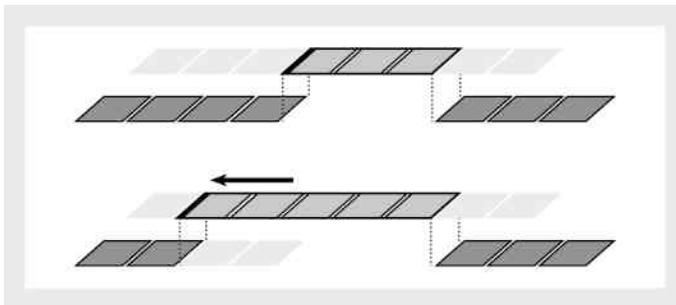
Übergänge zwischen Sequenzen  
bildgenau schneiden

Dazu evtl. Bilder weglassen oder aus  
den Ursprungsclips wieder  
hinzufügen  
(nichtlinearer Editiervorgang)

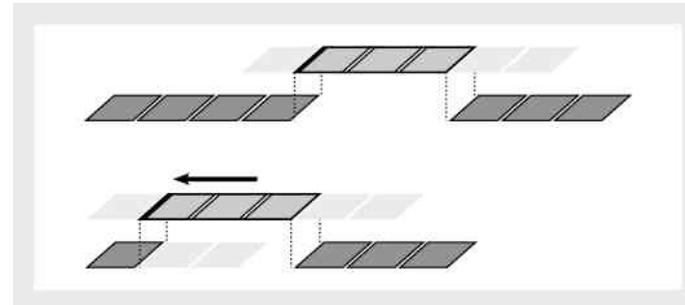
Schnittprogramme wie Premiere  
bieten spezielle Editieransichten  
dazu („Trimmen“)



„Rollen“  
(gleichbleibende Programmlänge)



„Löschen und Lücke schliessen“  
(veränderte Programmlänge)



# Effekte

Filmsequenzen können mit verschiedenen Effekten überarbeitet werden

Analogie zur Standbildbearbeitung  
(z.B. Weichzeichner, Farbanpassung etc.)

Dynamisierung von Effekten

„Schlüsselbilder“ (*key frames*) mit manuell definierten  
Effekteinstellungen und automatische Interpolation  
dazwischen  
(Analogie zu Macromedia Flash u.a.)

Effekte sind

sinnvoll bei der Herstellung eines einheitlichen  
Gesamteindrucks (z.B. Helligkeitsanpassung)

insgesamt eher zurückhaltend zu verwenden

Effekte oft erst nach „Rendering“ in der  
Programmorschau sichtbar  
(keine Echtzeiteffekte)



# Transparenz, Keys

- „Tricks“ entstehen oft durch Überlagerung von Videosequenzen
- Sequenzen mit Alpha-Kanal für Transparenz
  - meist nur auf speziellen „Trickspuren“ möglich (Video 2 und höher)
- *Key*:
  - Definition von Kriterien, anhand derer einzelne Teile einer Videosequenz transparent gemacht werden
  - Beispiele:
    - » Chroma-Keying: anhand des Farbtöns
    - » Blue Screen und Green Screen:  
anhand eines speziell definierten Farbtöns
    - » Bewegte Maske: individuell erstellte Maske  
z.B. zur Verfolgung eines Objekts
- Deckkraft kann u.U. in der Zeitleiste dynamisch geregelt werden  
(Analogie zu Audio-Hüllkurven)
- Überlagerungen oft erst nach „Rendering“ in der  
Programmorschau sichtbar

# „Blue Screen“-Technik

## Überlagern zweier Filmszenen

Vordergrund unabhängig vom Hintergrund aufgenommen

Neutraler, einfarbiger (meist blauer) Hintergrund

Anwendungsbeispiele:

Sprecher vor Hintergrundbild oder -Film (Bsp. Fernsehnachrichten)

Trickszenen im Film

## Konventionelle Realisierung:

Spezialfilm, der auf bestimmten Blauton unempfindlich ist

Zusammenkopieren mit Trickkopiermaschine

## Digitale Realisierung:

Vordergrundmotiv: Aufnahme vor blauem Hintergrund

Bestimmung von Schwellwert/Toleranz zur Umwandlung in Transparenz

Überlagerung mit Hintergrund (auf Video 1-Spur)

# Überblendung („Blenden“)

Erzeugen eines möglichst zum Gesamteindruck beitragenden Übergangs zwischen aufeinanderfolgenden Videosequenzen

Abblenden und Aufblenden:

Kontinuierlicher Übergang zu Schwarz bzw. von Schwarz

Überblendung:

Überlagerung der Bilder beider Videos und kontinuierlicher Übergang (Veränderung der Transparenz)

Schiebeblende:

Neues Bild „schiebt“ altes Bild weg

... und viele weitere Varianten,

z.B. Zoomblende, Unschärfeblende, Tür, Jalousie, ...

Generell mit Zurückhaltung anzuwenden und Wirkung auf den Betrachter berücksichtigen

# Export und Edit Decision List (EDL)

Ergebnis des Videoschnitts: *Edit Decision List*

Festlegung des Materials, seiner Eigenschaften und der durchgeführten Manipulationen

Vollständige und präzise Erfassung des zeitlichen Programmverlaufs über Timecodes

Als Austauschformat vor allem für die separate Master-Produktion in Spezialgeräten

Verschiedene Industriestandards

Erzeugen von weiterverarbeitbarem Videomaterial durch Export:

z.B. Ausgabe auf Videoband

z.B. Ausgabe als Videodatei (QuickTime, MPEG, ...)

### **3. Film- und Videotechnik und digitale Videobearbeitung**

3.1 Klassische Filmtechnik

3.2 Analoge TV- und Videotechnik

3.3 Digitale Videotechnik

3.4 Digitale Videoproduktion

3.5 Programmierung für die Videonachbearbeitung 

#### Literatur:

Ulrich Schmidt: Digitale Film- und Videotechnik, Fachbuchverlag Leipzig 2002  
Johannes Webers: Handbuch der Film- und Videotechnik, 7.Auflage,  
Franzis-Verlag 2002

# Java Media Framework

Kostenloses, plattformunabhängiges und objektorientiertes Medien-Framework

Hauptfunktionen:

- Abspielen von Medien

- Verarbeitung von Mediendaten in Echtzeit

- Erfassung von Datenströmen

- Speichern von Mediendaten

- Strombasierte Übertragung (*streaming*) von Mediendaten

Geschichte:

- JMF 1.0 (Sun, SGI, Intel): 1998

- JMF 2.0 (Sun, IBM): 1999

- Aktuelle Fassung JMF 2.1.1: 2001

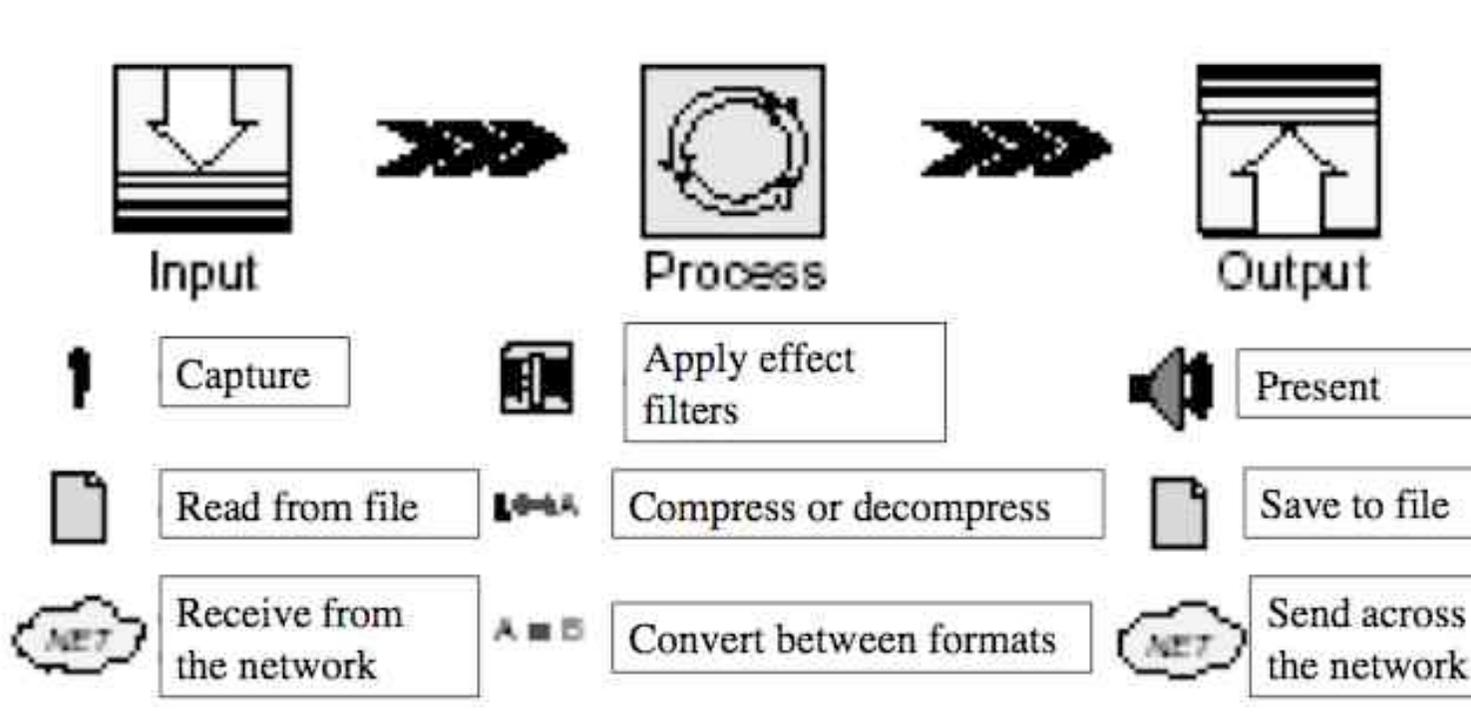
  - realisiert Funktionalität von 2.0

- Grundlage für Java Mobile Media API (für Java ME)

JMF *nicht* Bestandteil der Standard-Java2-Distribution

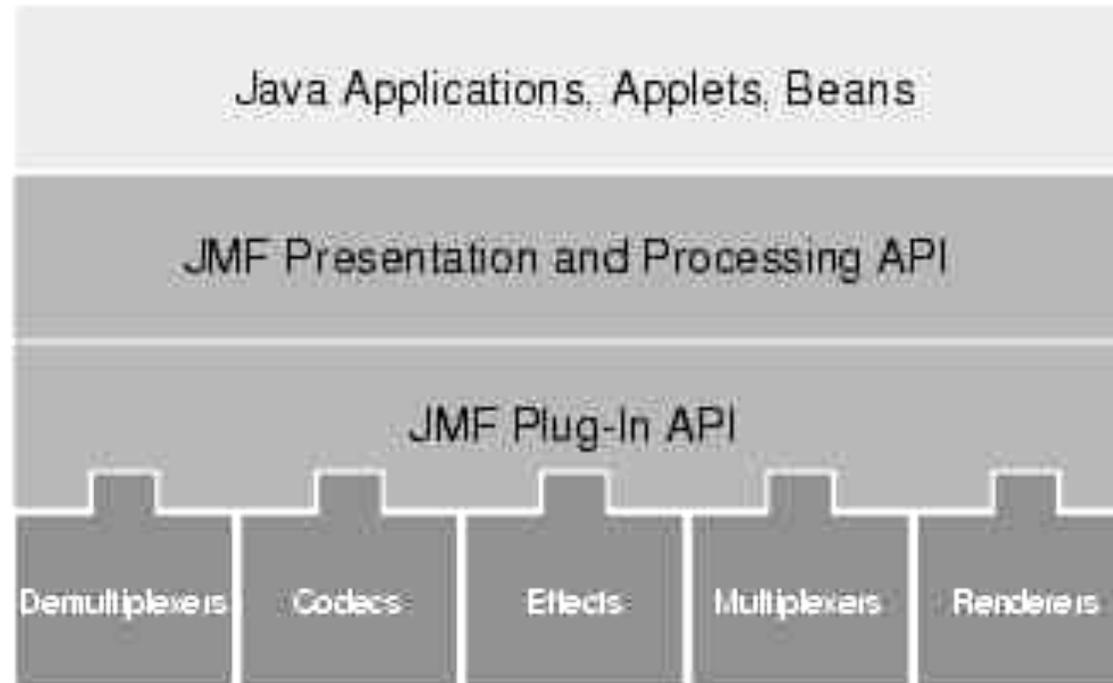
Cross-Platform-Implementierung und „Performance Packs“

# Verarbeitungsketten-Modell in JMF



Prinzipiell ist jede Kombination der möglichen Eingabeoptionen, Verarbeitungsschritte und Ausgabeoptionen möglich

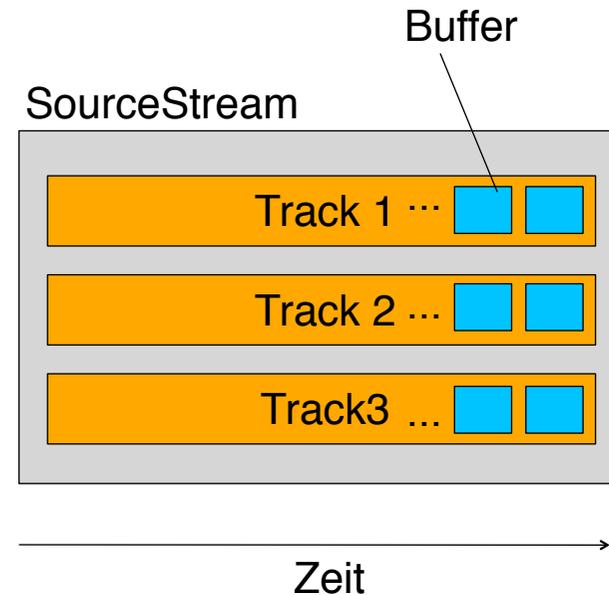
# High-Level-Architektur von JMF



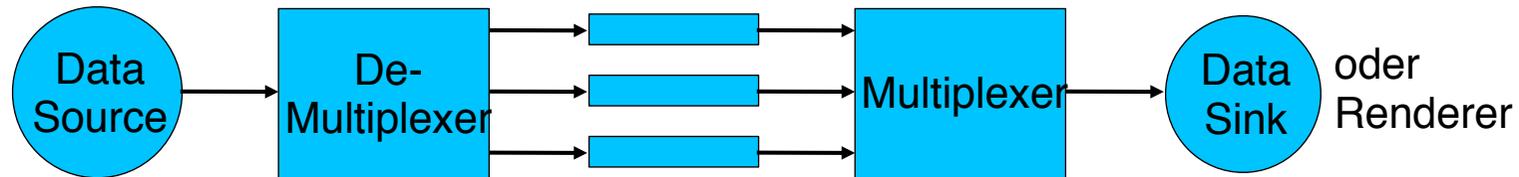
Durch „Plug-Ins“ sehr flexibel für die einheitliche Unterstützung verschiedener Medientypen und für nachträgliche Erweiterungen

# Packungsgrad von Medien

- **SourceStream:**
  - Kapselt Medium
  - Beschrieben durch `ContentDescriptor`
- **Track:**
  - Einzelkomponente eines Stroms (z.B. Video-, Audiospur)
  - Zugriff auf Mediendaten
- **Buffer:**
  - Einzelner Datenblock eines **Track**
  - Wird zur Weitergabe von Daten in Verarbeitungsketten genutzt
  - Detaillierte Beschreibung: `Format-Objekt`
    - » `AudioFormat`
    - » `VideoFormat`
      - `RGB`, `YUV`, `JPEG`, ...



# Verarbeitungsketten in JMF



## Kette von Transformatoren in JMF:

Echtzeitanforderung: Bei Zeitüberschreitung wird **Buffer** gelöscht

Alle Transformatoren passiv

Wer steuert die ganze Kette (und gibt z.B. den Zeittakt vor)?

- **Controller-Schnittstelle**

Schnittstelle für Steuerung von Verarbeitungsketten

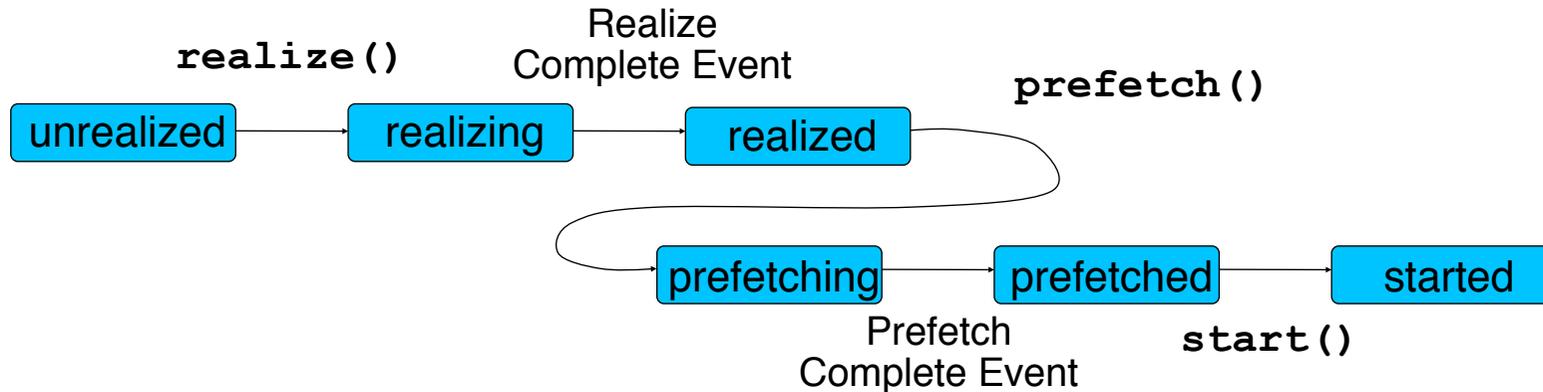
Spezialfälle:

- » **Processor**: Allgemeine Verarbeitungskette

- » **Player**: Einfacher Fall mit trivialer Transformation und Rendering

- **DataSink**: Dateiausgabe, ähnlich aber nicht von **Controller** abgeleitet

# Zustandsmodell von Player



- *Unrealized:*  
Anfangszustand
- *Realizing:*  
Medienabhängige Teile des Players werden bereitgestellt
- *Prefetching:*  
Eingabestrom wird soweit gelesen wie nötig, um Puffer zu füllen
- *Started:*  
Verarbeitung läuft

# Ereignis-Konzept in JMF

Ereignisse werden wie in AWT/Swing durch *callback* realisiert

Bei einem **Player** werden Objekte mit **addControllerListener** registriert, die Controller-Ereignisse interpretieren

```
public interface javax.media.ControllerListener {  
    public void controllerUpdate(ControllerEvent event)  
}
```

Beispiele für Controller-Ereignisse  
(Unterklassen von **ControllerEvent**):

- **RealizeCompleteEvent**
- **PrefetchCompleteEvent**
- **StartEvent**
- **StopAtTimeEvent**
- **EndOfMediaEvent**
- **FormatChangeEvent**
- **RateChangeEvent**
- **StopTimeChangeEvent**

# Manager-Klassen in JMF

Vermittlerobjekte zwischen den zahlreichen Interfaces und Implementierungen davon:

– *Manager*

zur Verwaltung und Konstruktion von Controller-Objekten  
(*Player*-, *Processor*-, *DataSource*-, und *DataSink*- Objekte)

Beispiel: `Manager.createPlayer(DataSource d)` ;

– *PackageManager* (Verwaltung aller Pakete, die die JMF-Dateien enthalten)

– *CaptureDeviceManager* (Verwaltung aller vorhandenen Eingabegeräte)

– *PluginManager* (Verwaltung aller verfügbaren Plug-Ins, z.B. Multiplexer, Codecs)

– *RTPManager* (Verwaltung von Streaming-Sitzungen)

## **MediaLocator**

Erwartet Information zu Protokoll, Hostname, Dateiname in URL-ähnlicher Syntax

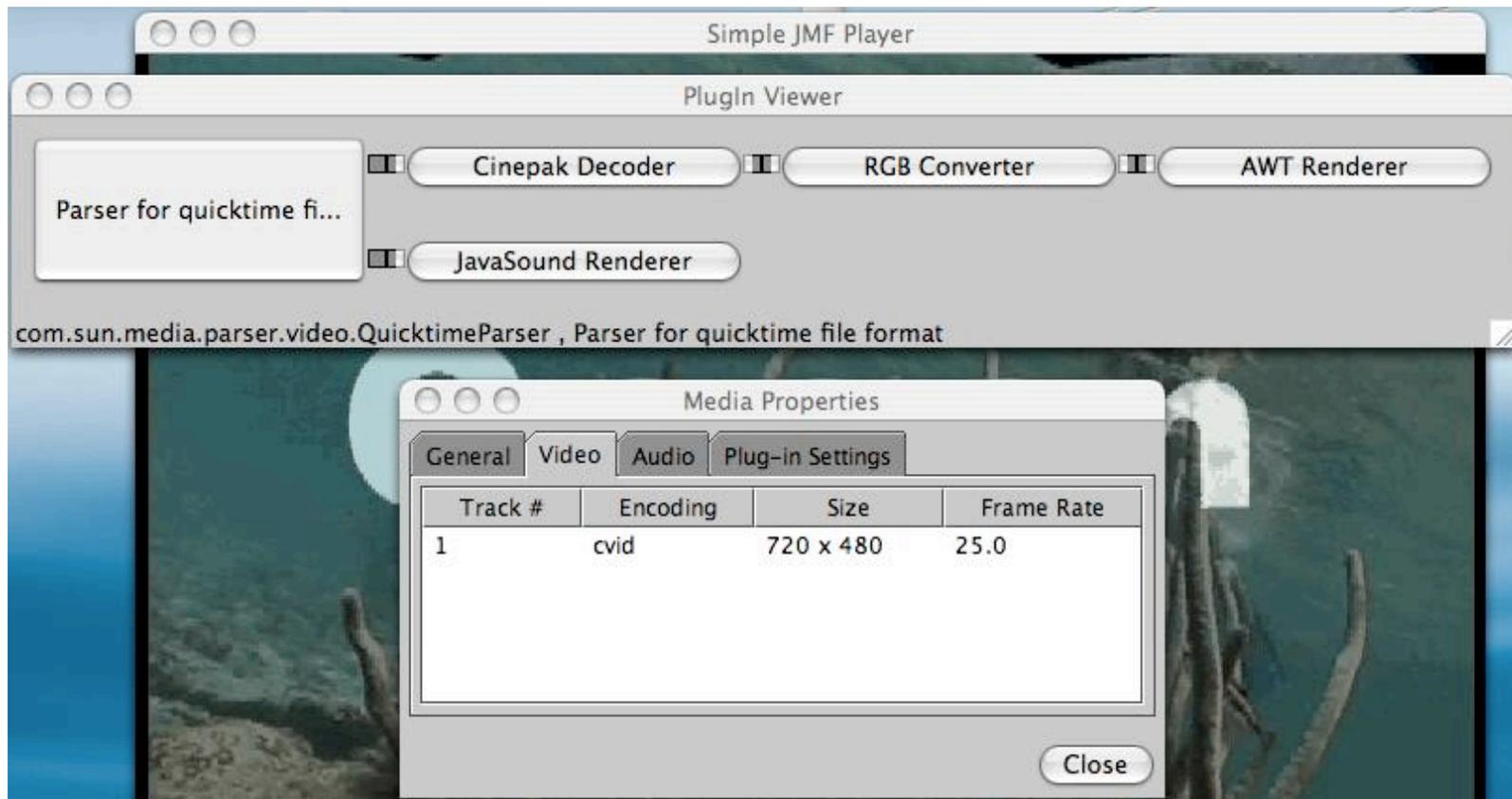
Lokalisiert Medienquelle und richtet notwendige Verwaltung (z.B. Pufferung) ein

(Viele JMF-Funktionen erlauben auch direktere Angabe von Medienquellen)

# Beispiel: Logo-Effekt



# Verarbeitungsketten



# Processor

- **Processor** ist die Abstraktion einer medienverarbeitenden Einheit

Funktionsmöglichkeiten:

Nimmt Datenquellen als Eingabe

Führt beliebige benutzerdefinierte Transformationen aus

Liefert bearbeitete Daten ab

Auf Ausgabegerät (Rendering, analog Player)

Als **DataSource**

Vom **Processor** gelieferte **DataSource** kann weiterverarbeitet werden

In einem weiteren **Processor**

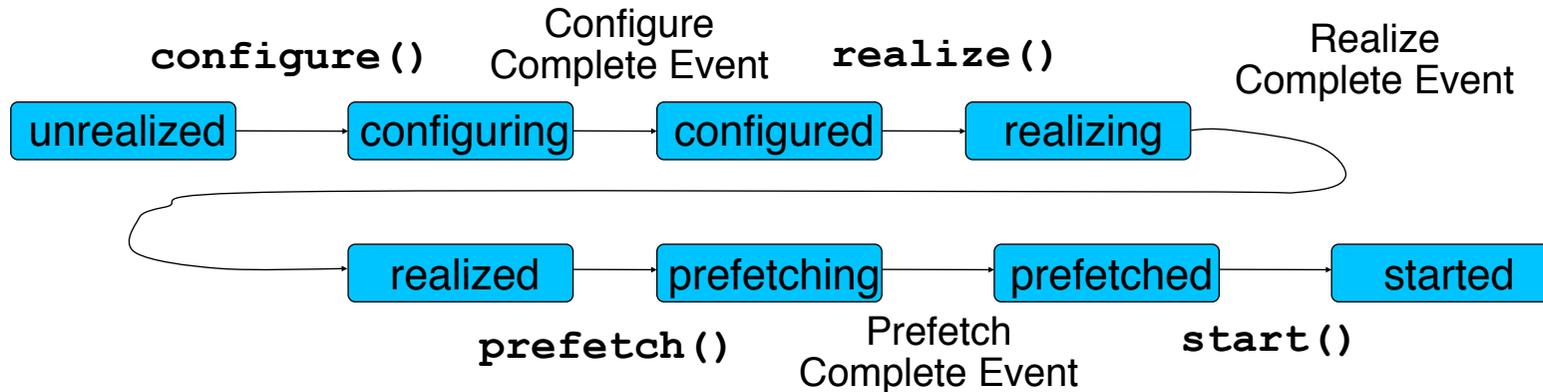
In einer **DataSink** (z.B. Speicherung in einer Datei)

Wichtigster Unterschied zu **Player**:

Separate Bearbeitung verschiedener Tracks der Quelle

- **Processor** wird erzeugt mit  
`Manager.createProcessor (DataSource ds)`

# Zustandsmodell von Processor



- *Configuring:*  
Die Eingabe wird auf die enthaltenen Medien (Spuren, *tracks*) analysiert
- *Configured:*  
Bearbeitung für die einzelnen Spuren kann separat definiert werden
- *Realizing:*  
Wie beim Player: Verarbeitungskette wird abhängig von den konkreten Mediendaten bereitgestellt
- *(alles andere wie beim Player)*

# Beispiel: Logo-Effekt (1)

```
class videoPlayerFrame extends JFrame
    implements ControllerListener {

    Processor p = null;
    Player player = null;

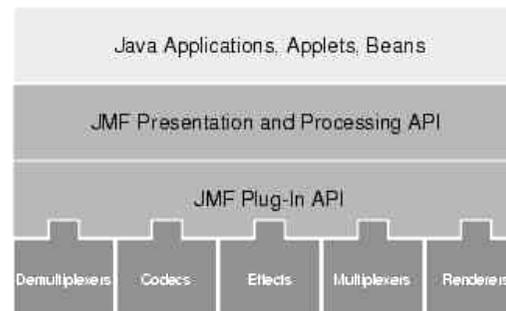
    public videoPlayerFrame(String file) {
        setTitle("Logo Effect");
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                p.stop();
                p.deallocate();
                System.exit(0);
            }
        });
        try {
            p = Manager.createProcessor
                (new MediaLocator("file:"+file));
            p.addControllerListener(this);
            p.configure();
        } catch(Exception e) { ...}
    }
    ...
}
```

# Plugin

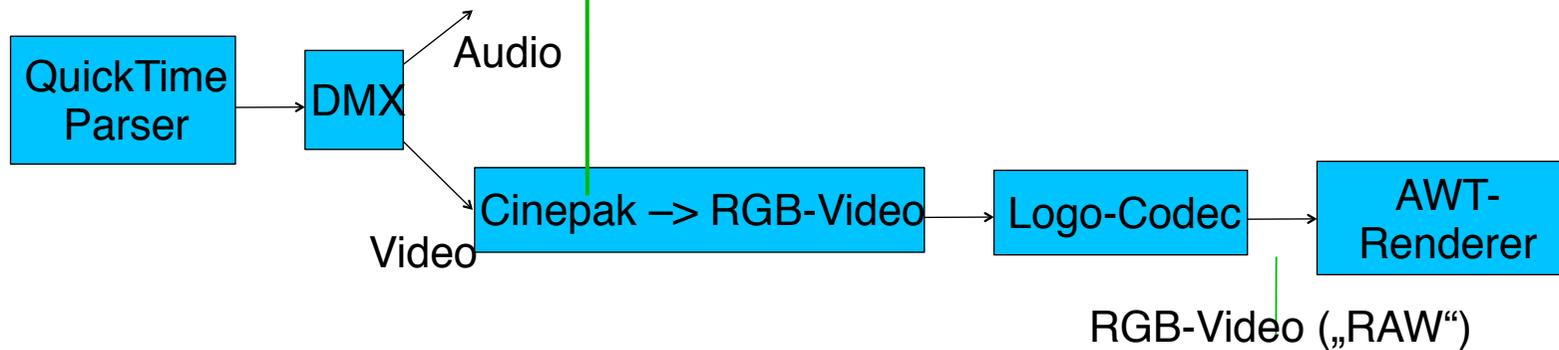
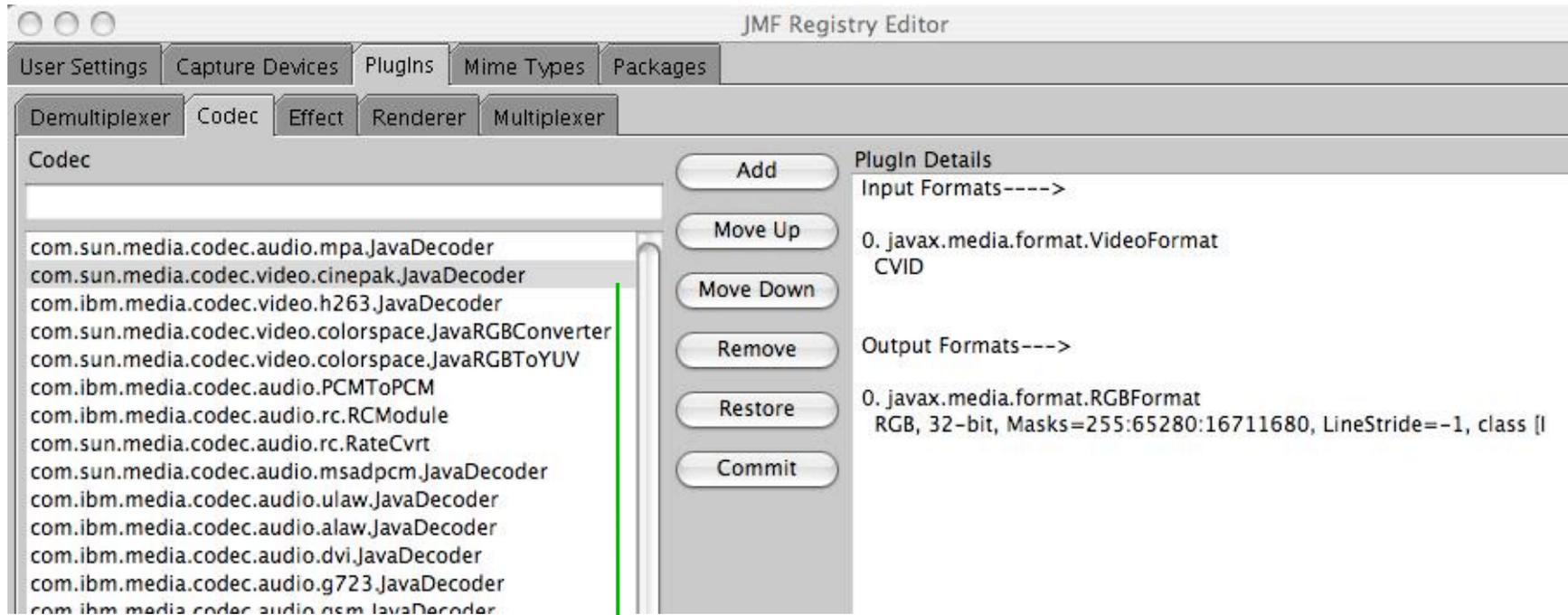
Basisschnittstelle für alle Elemente der Verarbeitungskette, die Daten in einem bestimmten Format ein- und/oder ausgeben

Unterschnittstellen:

- **Codec**  
Unterschnittstelle **Effect**
  - **Multiplexer, Demultiplexer**
  - **Renderer, VideoRenderer**
- **Plugin-Schnittstelle** unterstützt allgemeine Operationen **open ()** , **close ()** , **reset ()** , die bei Konfiguration aufgerufen werden  
Weitere Details in den Unterschnittstellen definiert



# Verarbeitungskette für Logo-Effekt



# Track, TrackControl

- **Track :**

- Ergebnis eines Demultiplexers

- Strom von gleichartigen Mediendaten (**Buffer**)

- **TrackControl:**

- Schnittstelle zur Steuerung der Verarbeitung eines einzelnen Tracks

- `Processor.getTrackControls()`

- Liefert Array von `TrackControl`-Objekten

- `void setCodecChain() (Codec[] codecs)`

- Spezifikation einer Kette von anzuwendenden `Codec`-Plugins

## Beispiel: Logo-Effekt (2)

```
public synchronized void controllerUpdate
(ControllerEvent event) {
    if (event instanceof ConfigureCompleteEvent) {
        TrackControl[] tracks = p.getTrackControls();
        boolean found = false;
        for(int i=0; i < tracks.length; i++) {
            if (!found && tracks[i].getFormat()
                instanceof VideoFormat) {
                Codec[] videoConversion = new Codec[] {
                    new com.sun.media.codec.video.cinepak.JavaDecoder(),
                    new logoCodec()
                };
                logoCodecControlIF lcc = (logoCodecControlIF)
                    (videoConversion[1].getControl("logoCodecControl"));
                lcc.setAlphaChannel(0.5);
                try {
                    tracks[i].setCodecChain(videoConversion);
                    found = true;
                } catch (Exception e) {...}
            }
        }
    };...
```

spezifisch  
für neuen  
„Logo-Codec“

## Beispiel: Logo-Effekt (3)

```
        if (found) p.realize();
        else { Fehlerbehandlung }
    }
else if (event instanceof RealizeCompleteEvent) {
    try {
        player = Manager.createRealizedPlayer(p.getDataOutput());
        getContentPane().setLayout(new BorderLayout());
        getContentPane().add(player.getControlPanelComponent(),
            BorderLayout.SOUTH);
        getContentPane().add(player.getVisualComponent(),
            BorderLayout.NORTH);
        setSize(720,525);
        setVisible(true);
        player.start();
        p.start();
    } catch(Exception e) { Fehlerbehandlung }
}
else if (event instanceof EndOfMediaEvent) {
    player.stop();
    p.stop();    ... Z.B. Neustart
}
}...
```

*Hauptprogramm: Erzeugen eines videoPlayerFrame-Objekts*

# Implementierung eines eigenen Codec/Effect

Implementierung der Schnittstelle `Codec` bzw. ihrer trivialen  
Unterschnittstelle `Effect`

Selbstbeschreibung bezüglich Strom-Datenformaten

Realisierung der Konfigurations-Operationen `open()`, `close()`,  
`reset()`

Fachlicher Kern: `process()`

Implementierung mindestens eines `Controls` zur Steuerung

```
public interface Codec { // Auszug!  
    Format[] getSupportedInputFormats()  
    Format[] getSupportedOutputFormats()  
    int process(Buffer input, Buffer output)  
    ...  
    Object getControl(String controlType)  
    ...  
}
```

## Beispiel: Logo-Effekt (4)

```
public class logoCodec implements Effect {

    private Format[] inputFormats;
    private Format[] outputFormats; ...

    private logoCodecControl control =
        new logoCodecControl();
    private int [] logoArray      = null;

    private String logoFile      = "logo.jpg";
    private int logoWidth        = 60;
    private int logoHeight       = 80;
    private int offsetX          = 10;
    private int offsetY          = 10;
    private double alphaChannel  = 1.0;

    public void open() {
        ...
        JPEGImageDecode decoder= JPEGCodec.createJPEGDecoder
            (new FileInputStream(logoFile));
        ... Einlesen von logoArray
    }
    ...
}
```

## Beispiel: Logo-Effekt (5)

```
public void close() {}
public void reset() {
    close();
    open();
}
public synchronized int process
(Buffer in, Buffer out) {
siehe später
}
public synchronized Object getControl
(String controlType) {
    return(control);
}
private class logoCodecControl
implements logoCodecControlIF {
    siehe nächste Folien
}
}
```

# Control

- **Control:**

Basisschnittstelle für alle Funktionen, die die Verarbeitung steuern

```
public interface Control {  
    java.awt.Component getControlComponent()  
}
```

- `getControlComponent` liefert eine Swing-Komponente, die als GUI der Steuerung dient, kann `null` sein, wenn kein GUI vorhanden

Jedes Plugin (also auch jeder Codec) muss anbieten:

```
Object getControl (String classname)
```

Das Resultat-Objekt muss `Control` implementieren und `classname`-Objekte steuern können

Fachliche Funktionen für das spezifische Plugin (meist Setzen von Einstellungen) in der Steuerungsschnittstelle

Beispiel:

```
logoCodecControlIF lcc = (logoCodecControlIF)  
    (videoConversion[1].getControl("logoCodecControl"));  
lcc.setAlphaChannel(0.5);
```

# „Trick“: Erhöhung der Flexibilität in Frameworks

Bei Sprachen, die *reflektive* Sprachmittel anbieten  
(z.B. Ermittlung des Klassennamens als Zeichenreihe)

Statt einer speziellen (tatsächlich geforderten) Operation:

```
class logoCodec {  
    logoCodecControl getControl ()..  
}
```

wird eine allgemeinere Operation realisiert bzw. gefordert:

```
Object getControl (String classname)
```

Aufruf z.B.:

```
lc.getControl ("logoCodecControl");
```

– logoCodecControl ist Klasse, die (mindestens) Control implementiert

Idealerweise wird zur Laufzeit die Typrichtigkeit überprüft

## Beispiel: Logo-Effekt (6)

```
public interface logoCodecControlIF
    extends javax.media.Control {

    public java.awt.Component getControlComponent();
    public String getLogoFileName();
    public void setLogoFileName(String fileName);
    public int getLogoWidth();
    public void setLogoWidth(int w);
    public int getLogoHeight();
    public void setLogoHeight(int h);
    public int getXOffset();
    public void setXOffset(int ox);
    public int getYOffset();
    public void setYOffset(int oy);
    public double getAlphaChannel();
    public void setAlphaChannel(double ac);
}
```

aus Interface  
javax.media.  
Control

spezifisch  
für neuen  
„Logo-  
Codec“

## Beispiel: Logo-Effekt (7)

```
private class logoCodecControl implements logoCodecControlIF
{
    public java.awt.Component getControlComponent() {
        return(null);
    }

    public int getLogoHeight() {
        return(logoHeight);
    }

    public void setLogoHeight(int h) {
        logoHeight=h;
        reset();
    }

    ... Usw. (analog)
}
```

Beispiel: `lcc.setAlphaChannel(0.5);`

## Beispiel: Logo-Effekt (8)

```
public synchronized int process
    (Buffer in, Buffer out) {

    out.copy(in);
    int [] data = (int[]) out.getData();

    RGBFormat inFormat = (RGBFormat)in.getFormat();
    int redMask = inFormat.getRedMask();
    int greenMask = inFormat.getGreenMask();
    int blueMask = inFormat.getBlueMask();

    Dimension inSize = inFormat.getSize();
    int w = (int)inSize.getWidth();
    int h = (int)inSize.getHeight();

    int x,y, offBuffer,offLogo, pixelData;
    ...
}
```

## Beispiel: Logo-Effekt (9)

```
...
for(y=0; y<logoHeight; y++) {
for(x=0; x<logoWidth; x++) {
    offBuffer = (y+offsetY)*w + offsetX + x;
    offLogo = (y * logoWidth + x);

    pixelData = ((int)
        ((data[offBuffer] & redMask) * (1-alphaChannel)
        + (logoArray[offLogo] & 0x000000ff)
        * (alphaChannel))) & redMask;
    pixelData += ((int)
        ((data[offBuffer] & greenMask) * (1-alphaChannel)
        + (logoArray[offLogo] & 0x0000ff00)
        * (alphaChannel))) & greenMask;
    pixelData += ((int)
        ((data[offBuffer] & blueMask) * (1-alphaChannel)
        + (logoArray[offLogo] & 0x00ff0000)
        * (alphaChannel))) & blueMask;

    data[offBuffer] = pixelData;
    }
}
return BUFFER_PROCESSED_OK;
}
```