Multimedia II

Konrad Froitzheim, Helge Bahmann

Programm

0. Überblick
1. Networked Graphics and Multimedia
   - X-Window Grundbegriffe
   - MurX
2. Multimedia (Dokumenten-)Formate
   - Quicktime, MHEG
   - MPEG 4, 7, 21
3. Digital Music
   - Grundbegriffe aus Physik, Akustik und Musik
   - Midi, Sequencer, …
   - Software
4. Multimediaproduktion
   - Drehbuch, Aufnahme
   - Digital Compositing
   - Verteilung

Literatur

- Bahmann, H.: A Streaming Multimedia …; Diplomarbeit, FG 2002.
- Brinkmann, R.: The Art and Science of Digital Compositing; MKP, 1999.
- Chapman, Chapman: Digital Multimedia, 2000.
- Froitzheim, K.: Multimediale Kommunikationsdienste, 1996.
- Halsall: Multimedia Communications; Addison-Wesley, 2001.
- Homann, J.-P.: Digitales Colormanagement; Springer, 2000.
- Jones, O.: Einführung in das X-Window System.
- Li, Drew: Fundamentals of Multimedia; Prentice-Hall, 2004.
- Scheifler, Gettys, Newman: X Window System C Library and Protocol Reference.

Formales

Termine:

Vorlesung:	Dienstag	9:15 - 10:45
Übung:	Freitag	11:00 - 12:30

Dramatis Personae:

Prof. Dr. Konrad Froitzheim
03731/393939
frz@informatik.tu-freiberg.de

Helge Bahmann
bahmann@informatik.tu-freiberg.de

Vorlesungsarchiv:

http://ara.informatik.tu-freiberg.de/vorlesungen/2MM2007.doc
http://ara.informatik.tu-freiberg.de/vorlesungen/MurX/

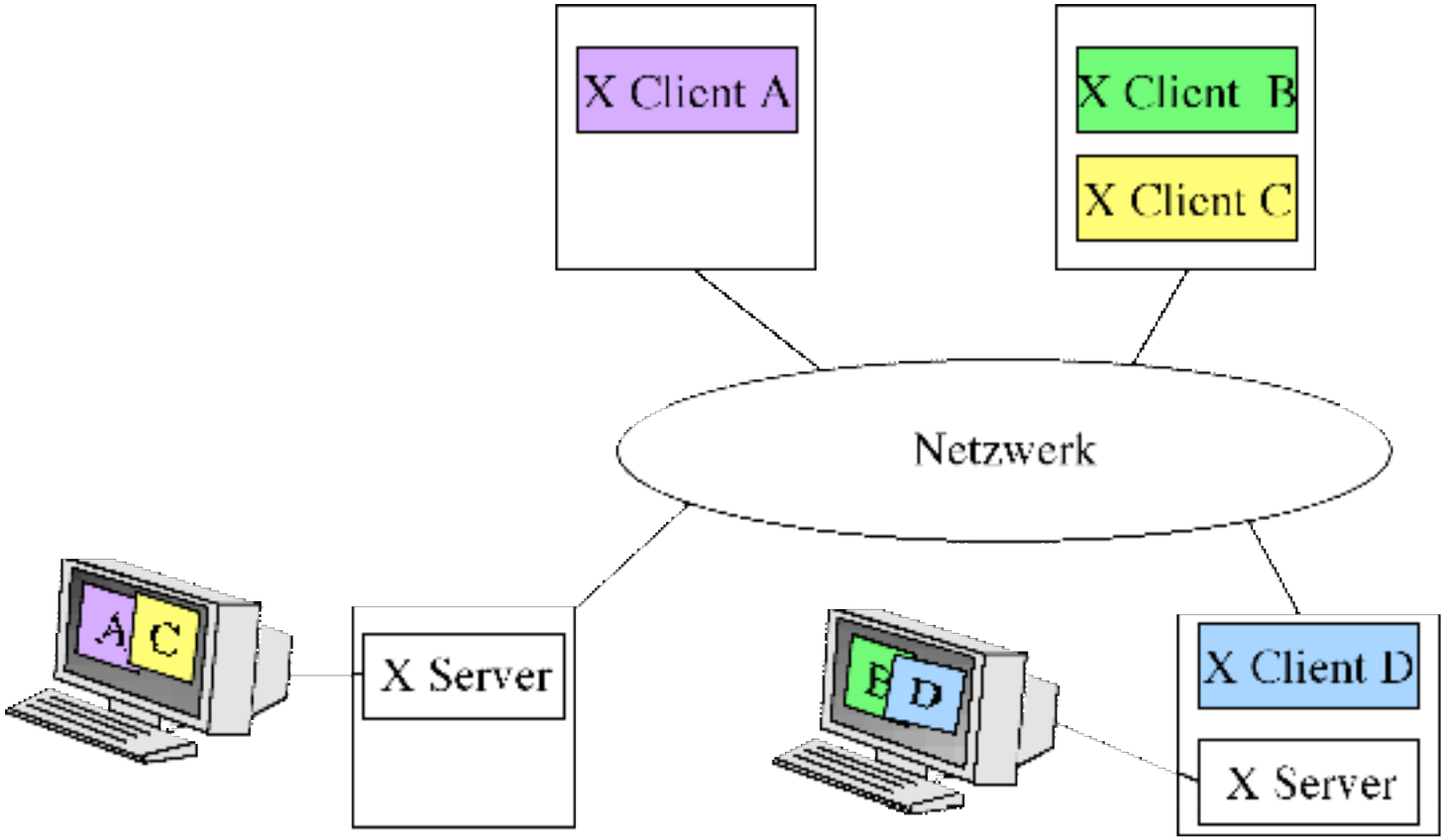
1. Networked Graphics and Multimedia

1.1 X-Window

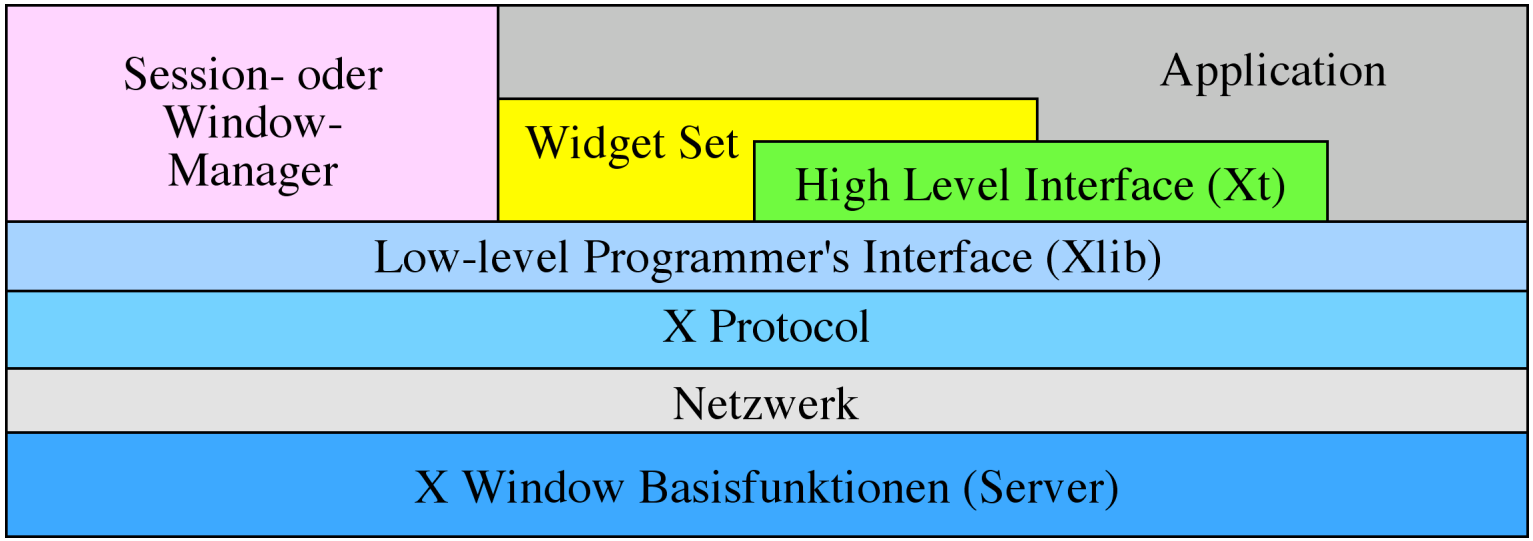
- De-facto Standard für Grafik auf UNIX-Systemen und Linux
- Herstellerunabhängig: X-Consortium am MIT (Scheiffler)
- Kein vorgegebenes 'Look and Feel' für User-Interface
- Netzwerkunabhängig
  - verlangt zuverlässigen, verbindungsorientierten Transportdienst
  - typischerweise auf TCP/IP
  - andere Netze möglich
- Geschichte:
  - M.I.T.: Project Athena
  - Ziel: herstellerunabhängiger Grafikstandard
  - unterstützt von DEC und anderen
  - gegenwärtiger Stand: X11R6Vn
  - Entwicklung Lightweight X …
- X Window System, X Window, X11, X
- ca. 20 'Standard'-Extensions
  - Shared Memory, Render, OpenGL, …

1.1.1 X Szenarien

- Clients und Server (Vorsicht, die Bezeichnungen sind 'verdreht')
  - X-Client ist die Applikation
  - X-Server ist das 'Terminal', schreibt in Bildwiederholspeicher
    - passive_open auf Port 6000+display



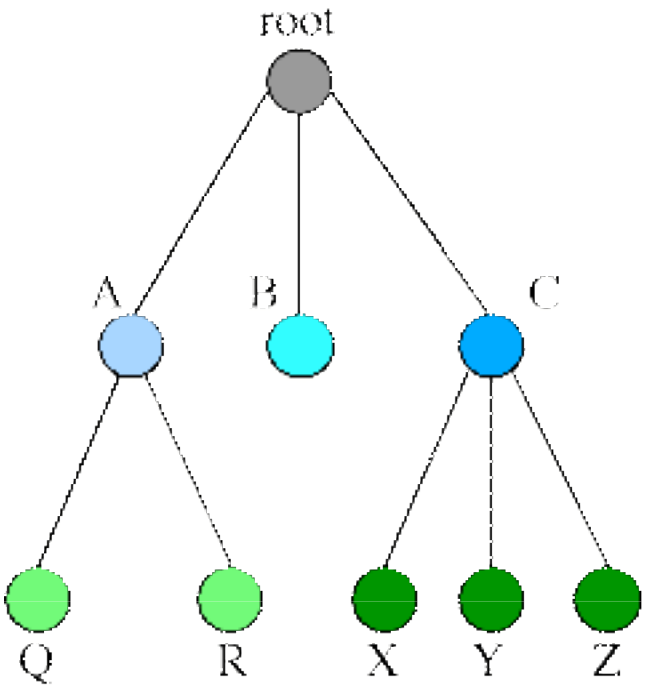
1.1.2 X Architektur

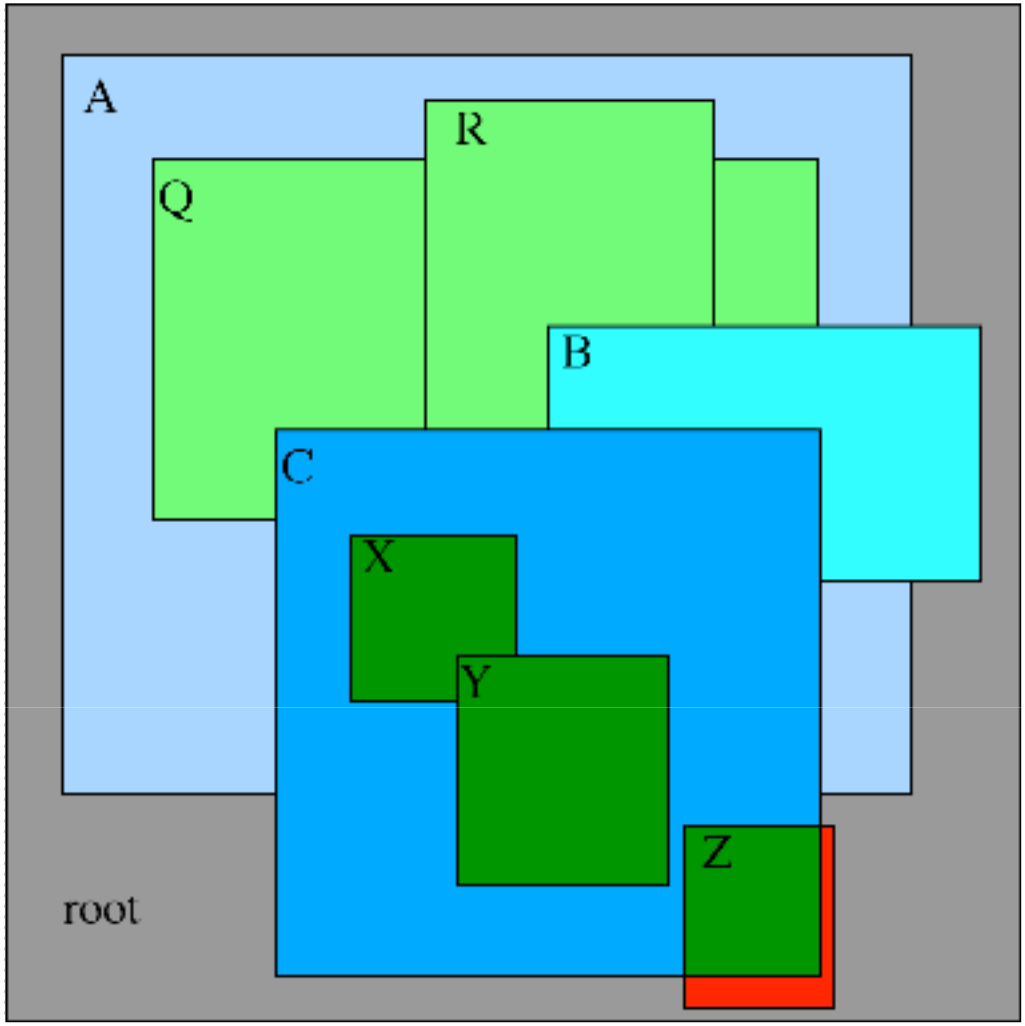


- Mechanisms
  - X Protocol : X-Befehle, Kodierungsvorschrift
  - Xlib : Application Programmers IF für X-Protocol
  - Xt Intrinsics : Rahmen für Widgets
- Policy (look and feel)
  - Window Mngr. : Fensteranordnung, Clipping, Fensterdekoration, UI für Fenstermanagement
  - Widgets : Buttons, Menus, Scrollbars, Dialogboxen, …

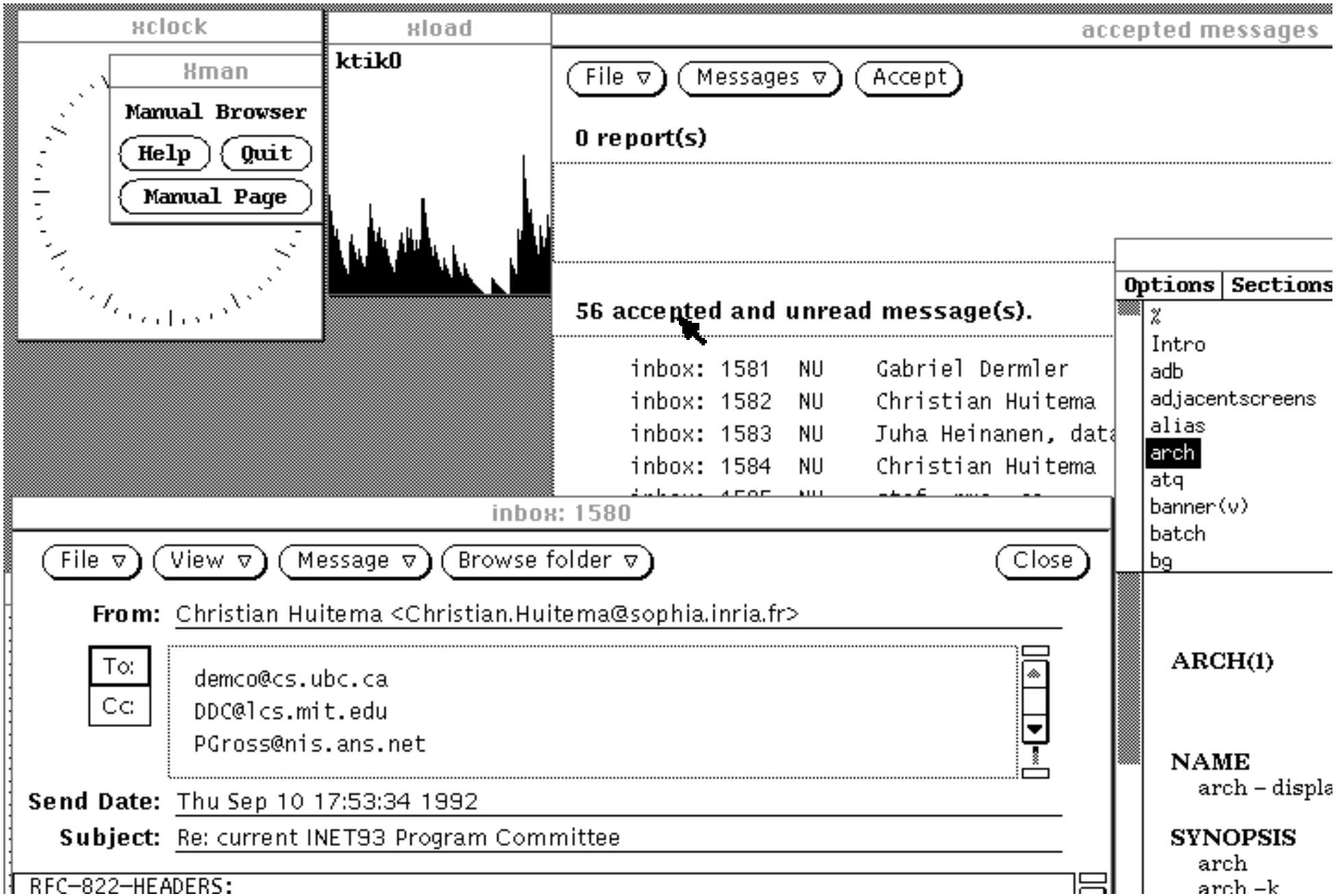
X-Szenarien (eine Maschine, 2 Maschinen, n …)

- Fensterhierarchie





- Stacking Order bestimmt das Abdecken durch 'Geschwister'


xclock
xload
ktik0
Xman
Manual Browser
Help
Quit
Manual Page
accepted messages
File
Messages
Accept
0 report(s)
56 accepted and unread message(s).
inbox: 1581 NU Gabriel Dermler
inbox: 1582 NU Christian Huitema
inbox: 1583 NU Juha Heinanen, data
inbox: 1584 NU Christian Huitema
Options
Sections
%
Intro
adb
adjacentscreens
alias
arch
atq
banner(v)
batch
bg
inbox: 1580
File
View
Message
Browse folder
Close
From: Christian Huitema <Christian.Huitema@sophia.inria.fr>
To:
Cc:
demco@cs.ubc.ca
DDC@lcs.mit.edu
PGross@nis.ans.net
Send Date: Thu Sep 10 17:53:34 1992
Subject: Re: current INET93 Program Committee
RFC-822-HEADERS:
ARCH(1)
NAME
arch - displa
SYNOPSIS
arch
arch -k

1.1.3 X programmieren

- Display

  Ein Server kontrolliert ein Display
  Zum Display gehören Screen(s), Tastatur und Zeiger (Maus)
  Name eines Displays: <Maschine>:<DispNr>.<ScreenNr>
  Beispiel: xclock -display someXMachine:0.0

- Drawable

  Window, Pixmap (Bitmap)
  Ort für Ausführung von Zeichenoperationen

- Grafikkontext (GC)

  XGCValues-Datenstruktur:
  function, foreground, background, line_width, line_style,
  font, fill_style, …

```
myGC = XCreateGC(display, drawable, mask, &values);
…
XChangeGC(display, myGC, mask, &values);
```

  Nicht alle Felder besetzt: `mask = GCForeGround | GCLineStyle;`

  Spezialfunktionen zur Veränderung einzelner Felder:

```
XSetFont(display, myGC, emphasisFont);
```

- Resources:
  - Window, Drawable, Pixmap, Graphic Context, Font, Color, …
  - Objekte liegen im Server
  - Server Resource Identifier (XID)
  - Ergebnis von XCreate…
- Namensgebung:
  - Xlib Funktionen und Datenstrukturen beginnen mit großem X

```
XCreateWindow()
XDrawString()
```

  - Xt Funktionen mit Xt:

```
XtCreateWidget()
```

  - Auch bei Widgets gibt es Namenskonventionen

    z.B. beginnen Motif Funktionen mit Xm:

```
XmStringDraw()
```

• Xlib Funktionen (Auswahl)

- Verwaltungsarbeit:

  Display öffnen: `myDisp = XOpenDisplay(theName);`

  Window kreieren:

  ```
  myWin = XCreateWindow(myDisp,prnt,x,y,width,hgt,…);
  ```

  Window sichtbar machen: `XMapRaised(myDisp,myWin);`

  Weitere Funktionen um Windows und Displays zu schließen etc.

  Hints sagen dem Windowmanager, wie das Fenster aussehen soll

- Grafische Objekte

```
XDrawPoint(myDisp, myWin, myGC, x, y);
XDrawLine(myDisp, myWin, myGC, x1, y1, x2, y2);
XFillRectangle(myDisp, myWin, myGC, x, y, width, height);
XDrawArc(myDisp, myWin, myGC, x, y, width, height,
                angle1, angle2);
```

  Randbedingungen in myGC

- Text

  Text anzeigen: `XDrawString(myDisp,myWin,myGC,x,y,str,len);`

  Font in den Server laden: `myFnt=XLoadFont(myDisp, fontName);`

  ```
  /* Do not forget XSetFont() */
  ```

  Eigenschaften des Fonts: `theFntDesc=XQueryFont(myDisp,myFnt);`

1.1.4 Events

- Ereignisse am Server werden dem Klienten gemeldet
  - Zustandsveränderungen
  - Taste, Maus, Expose
  - beziehen sich auf ein Window
  - Warteschlange
  - Beispiele:

```
ButtonPress, ButtonRelease, KeyPress,
MotionNotify, EnterNotify, LeaveNotify,
Expose, …
```

- Datenstruktur XEvent
  - gemeinsamer Teil für alle Events: XAnyEvent-Record

    type, display, window, …

```
event.xany.window
```

  - eventspezifischer Teil; Beispiel XKeyEvent-Record

```
time, x, y, state, keycode, …
theChar = XLookupString(event.xkey.keycode);
```

  - Anzahl wartende Expose-Events: `event.xexpose.count`

- Verarbeitung von Events
  - Klienten verlangen Events:

```
XSelectInput(display,window,Evt1Mask|Evt2Mask|…|EvtnMask);
```

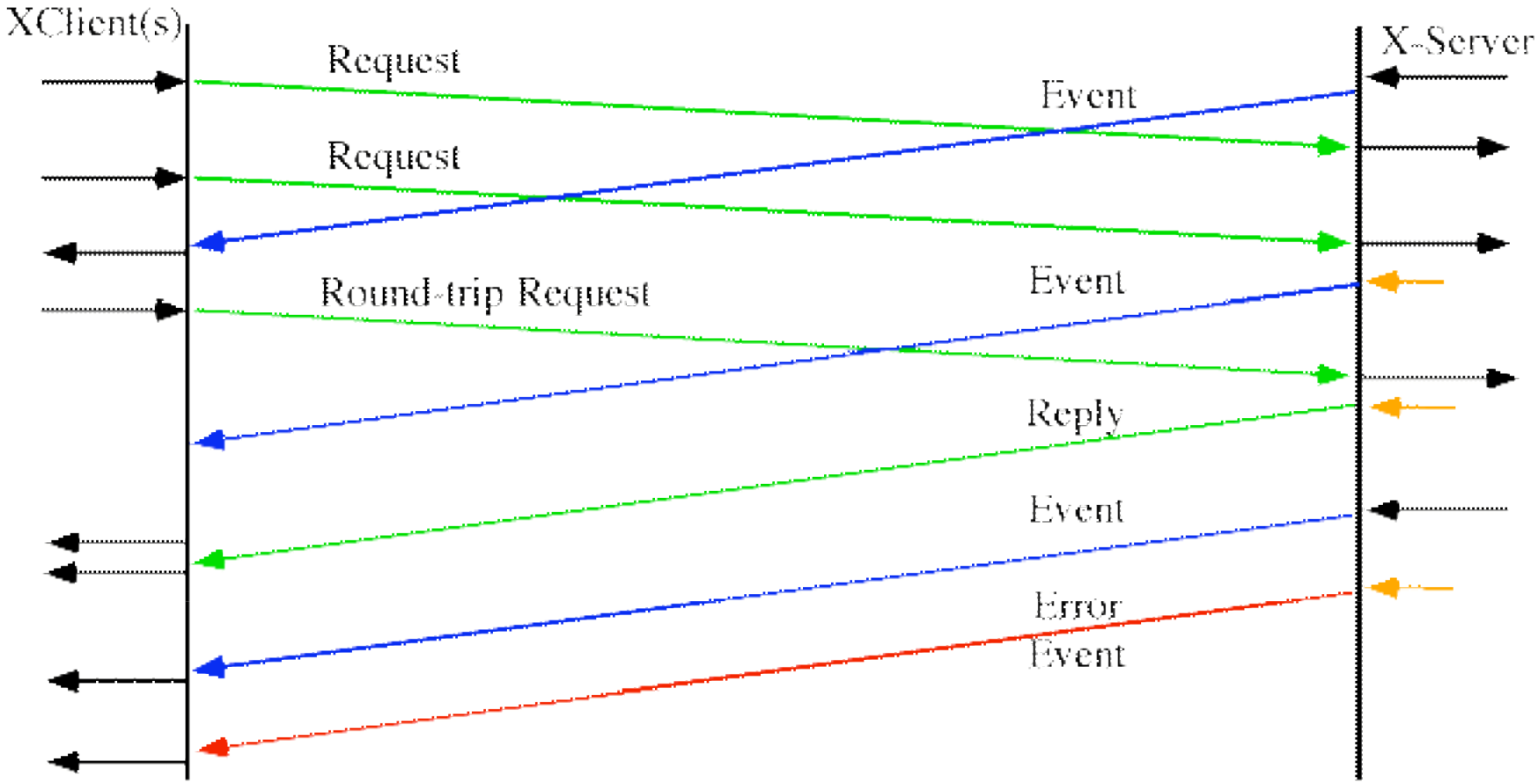
  - Maskenkonstanten:

```
KeyPressMask, ButtonPressMask,
PointerMotionMask, KeyPressMask,
KeyReleaseMask, ExposureMask,
EnterWindowMask, LeaveWindowmask, …
```

  - switch Statement

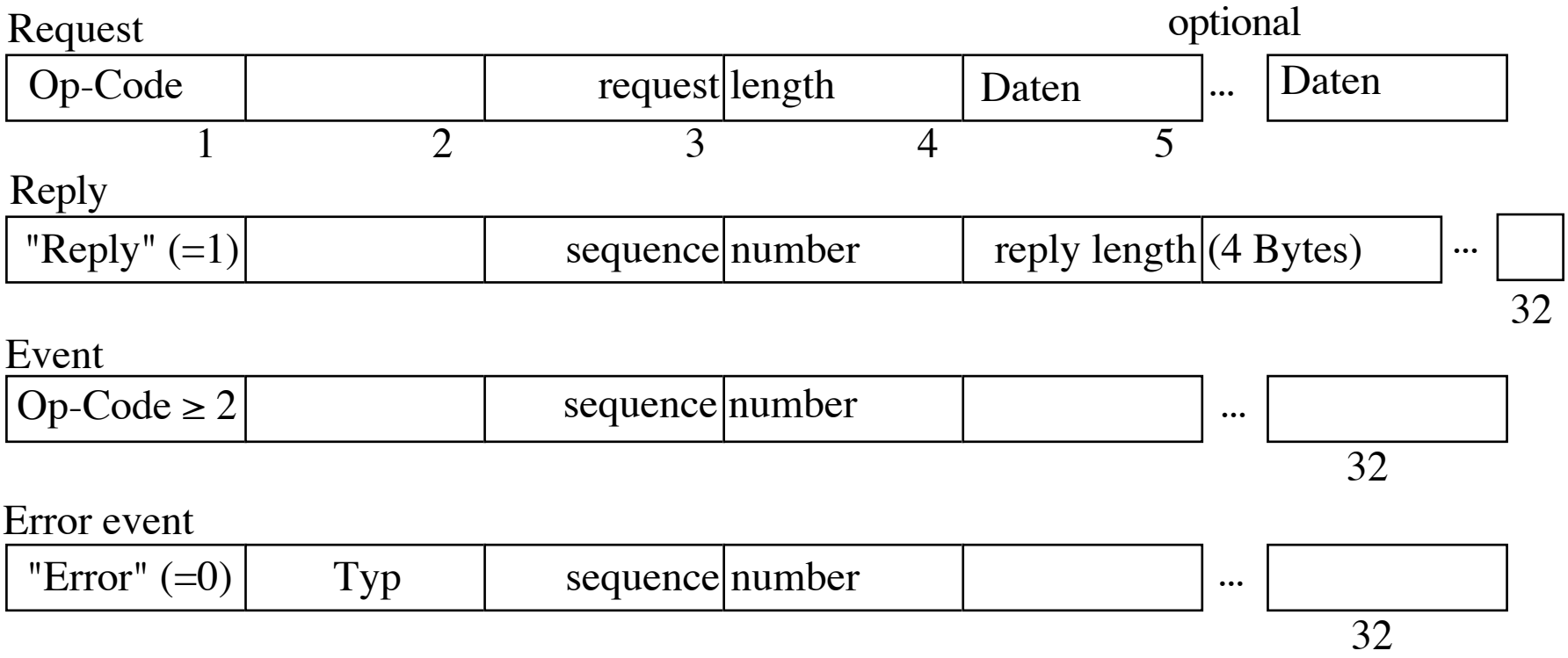
1.1.5 X Window Protocol

- Request (Klient -> Server) und Reply (Server -> Klient)
- Events (Server -> Klient), Fehlermeldungen ("Error-Events")



- eintreffende Events kommen in Warteschlange
- werden allen interessierten Klienten angeboten
- werden von Klienten abgenommen

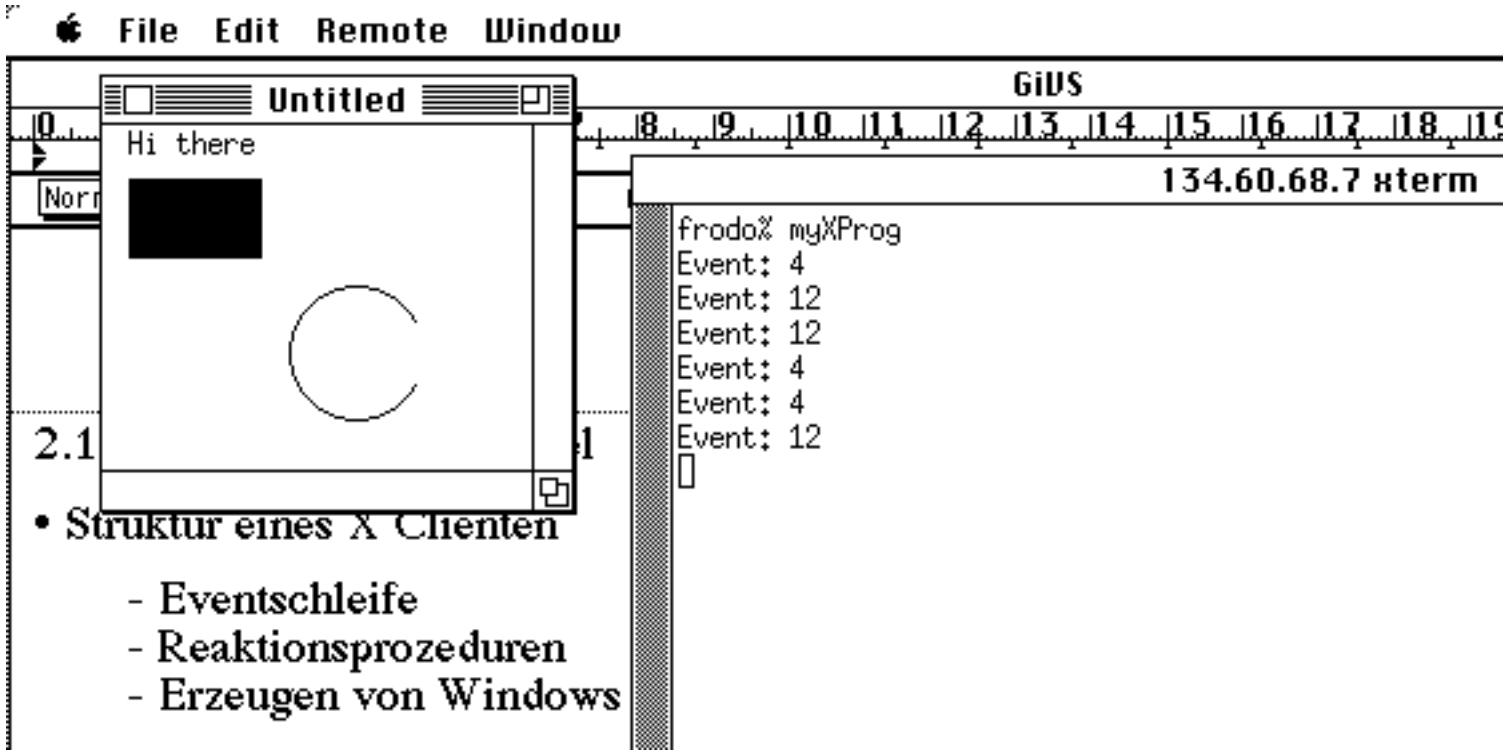
- Paketformat



- Request-Codes entsprechen teilweise Xlib-Funktionen
  - Context-Änderungen werden auf ein Protokoll-Element abgebildet
  - es gibt Protokollelemente ohne Xlib-Funktion
  - statisch alloziert für Standard-Operationen
- Allozierbare *-Codes für Extensions

1.1.6 Programmierbeispiel

- Struktur eines X Clienten
  - Eventschleife
  - Reaktionsprozeduren
  - Erzeugen von Windows etc.



• Hauptschleife fragt Events ab und reagiert entsprechend

```
void Loop4Events ()
{
    XEvent theEvent;
    quit = False;
    XSelectInput(myDisp,myWin,(ButtonPressMask|KeyPressMask|
                                        ExposureMask));
    while (quit == False)
    {  XNextEvent(myDisp, &theEvent);
       if (theEvent.xany.window==myWin)
         switch (theEvent.type)
         {
           case ButtonPress  : ReDraw(); break;
           case KeyPress     : quit = True; break;
           case Expose       : ReDraw(); break;
}  }     }
```

• Reaktion auf ExposeEvent, Zeichnen des gesamten Fensters

```
void ReDraw ()
{
    XSetForeground(myDisp, myGC, WhitePixel(myDisp, myScreen));
    XFillRectangle(myDisp, myWin, myGC,0,0,150,120);
    XSetForeground(myDisp, myGC, BlackPixel(myDisp, myScreen));
    XDrawString(myDisp,myWin, myGC,10,10,"Hi there",8);
    XFillRectangle(myDisp, myWin, myGC,10,20,50,30);
    XDrawArc(myDisp, myWin, myGC,70,60,50,50,30*64,300*64);
}
```

• Hilfsfunktion zum Erzeugen eines einfachen Fensters

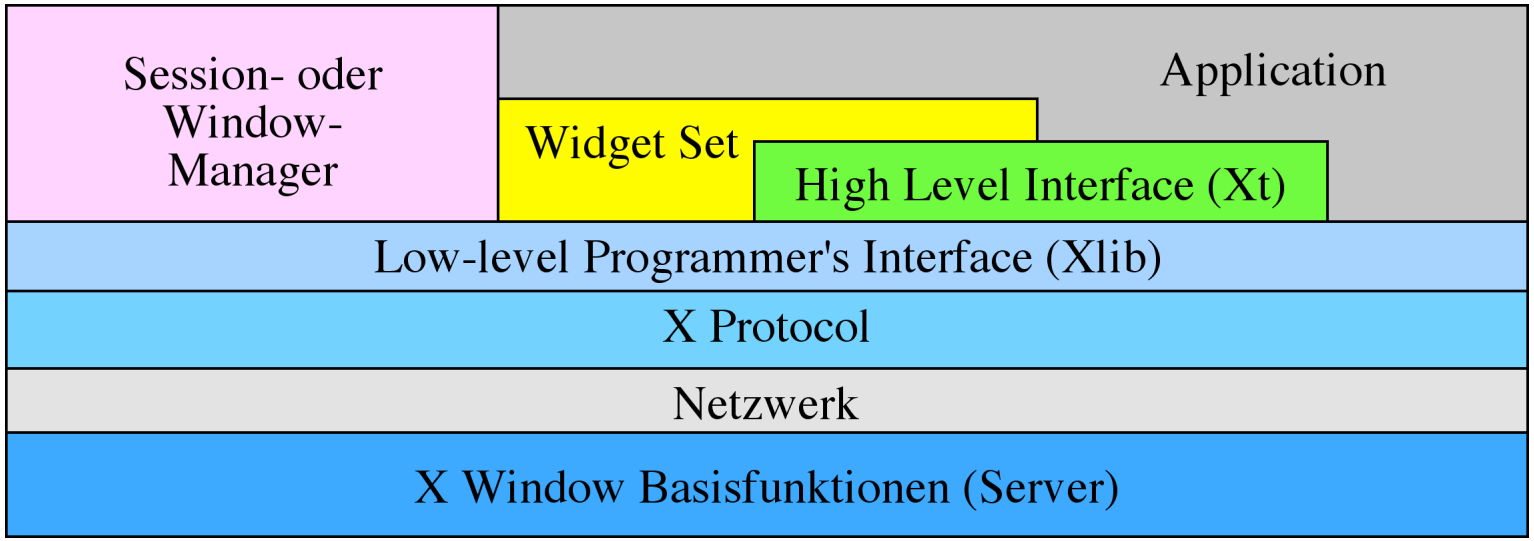
```
Window MakeWindow(name,x,y,w,h)
    char   *name;
    int x, y, w, h;
{
    Window              aWin;
    unsigned long       black, white;
    XGCValues           gcVals;

    aWin = NULL;
    myDisp = XOpenDisplay(name);
    myScreen = DefaultScreen(myDisp);

    myParent = RootWindow(myDisp, myScreen);
    black = BlackPixel(myDisp, myScreen);
    white = WhitePixel(myDisp, myScreen);

    aWin =
     XCreateSimpleWindow(myDisp,myParent,x,y,w,h,0,black,white);
    if (aWin != NULL)
        { XMapRaised(myDisp, aWin);
          gcVals.foreground = black;
          gcVals.background = white;
          myGC = XCreateGC(myDisp, aWin,
                           (GCForeground|GCBackground),&gcVals);
        }
    return(aWin);
}
```

• Programmrahmen

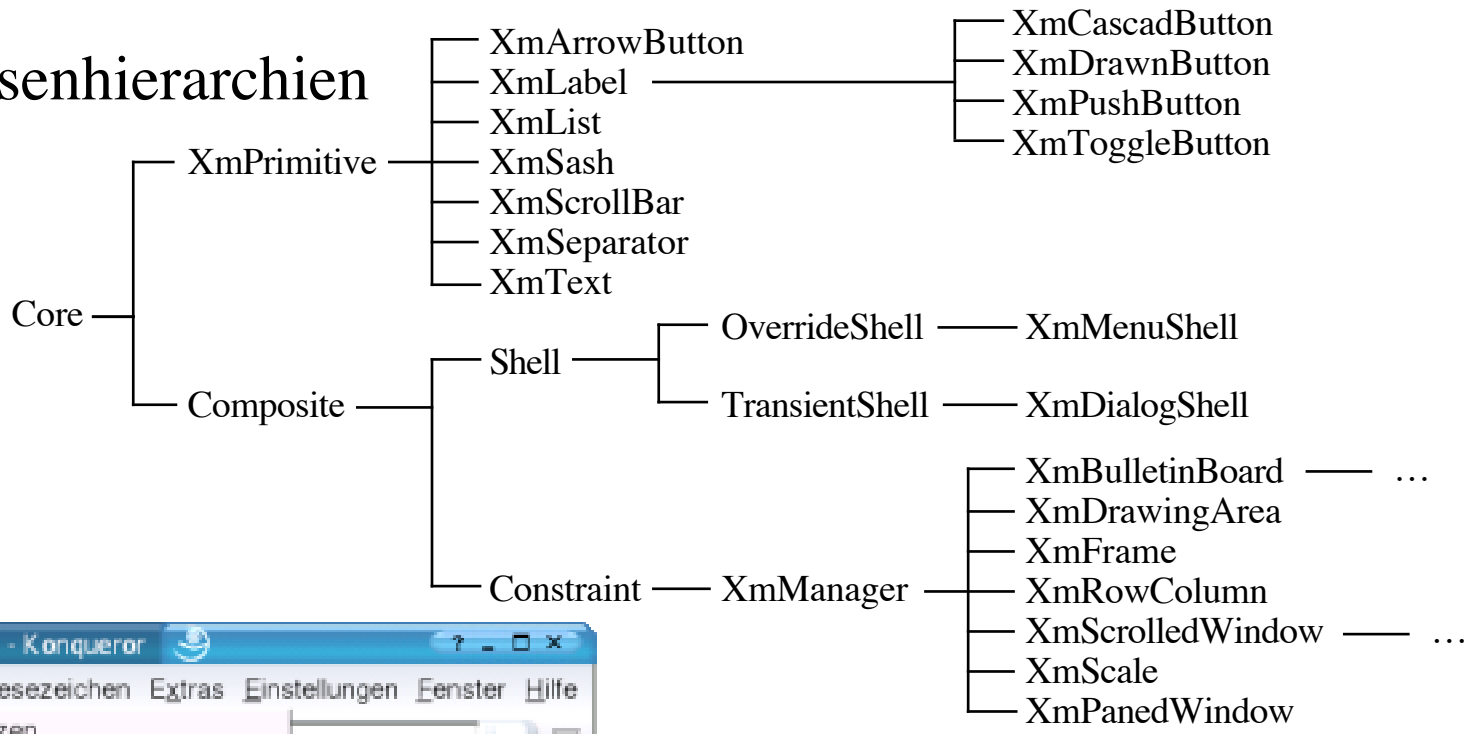
```
#include <X11/Xlib.h>
#include <stdio.h>

Display   *myDisp;
Window     myWin, myParent;
int        myScreen;
GC         myGC;
Bool       quit;


Window MakeWindow(name,x,y,w,h)
…

void ReDraw ()
…

void Loop4Events ()
…

main()
{
    myWin = MakeWindow("134.60.70.20:0.0",100, 50, 150, 120);
    if (myWin != NULL) Loop4Events();
}
```

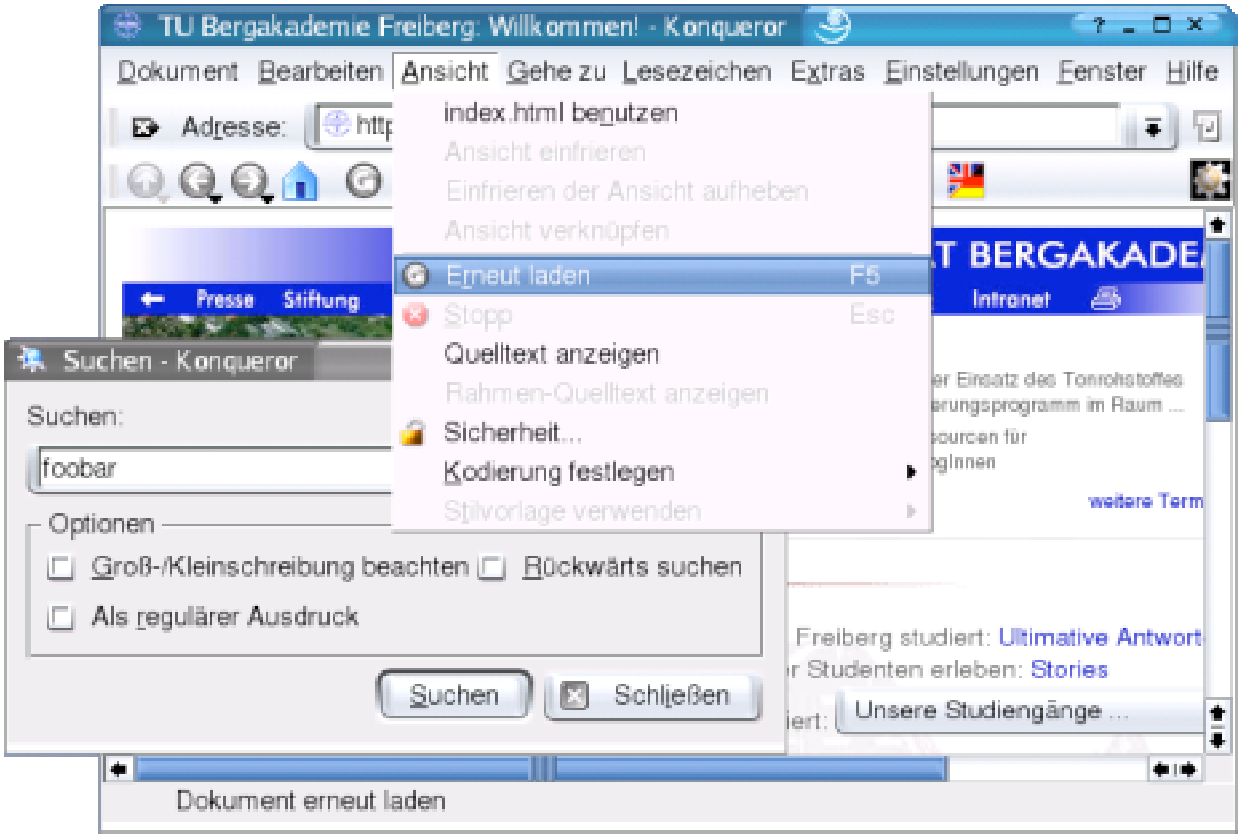
1.1.7 Widgets und Toolkits

- Oft gebrauchte grafische Elemente
  - Buttons mit besonderem Look, Dialogboxen, …
  - standardisiertes UI
    -> Policy für grafische Elemente
  - MOTIF, Athena, …
  - Gtk (Gimp, Gnome)
  - Qt (KDE)



- Widgetsets sind Klassenhierarchien





- Xt als Programmvereinfachung
  - objektorientierte Architektur
  - Verwalten der Widgets
  - Widgets werden in Klassen organisiert
  - XtCreateWidget erzeugt Instanz einer Klasse

```
top = XtInitialize(…,cls,NULL,0,&argc,argv);
                /* initialize toolkit and display */
                 /* besser XtAppInitialize(…) */

myScr=XtCreateWidget("Scroller",XmScrollBarWidgetClass,top,NULL,0);
XtRealizeWidget(top);
    /* show widget top and all children */
```

- Eventhandler
  - übernimmt Event-'Routing' (dispatch)
  - callbacks (prozedural oder objektorientiert)

```
…void do_something();
int okbutton;
…
okbutton=XCreateWindow(…);
…



XEvent ev;
while(1) {
  XNextEvent(display, &ev);
  …
  if (ev.xany.window==okbutton) {
    if (ev.type==ButtonPress) {
      do_something();}
    …
  …
}
```

```
…
void do_something();
Widget okbutton;
…
okbutton=XmCreatePushButton(…);
XtAddCallback(okbutton,
  XmNactivateCallback,
  do_something, 0);
…

XEvent ev;
while(1) {
  XNextEvent(display, &ev);
  XtDispatchEvent(&ev);
}
```

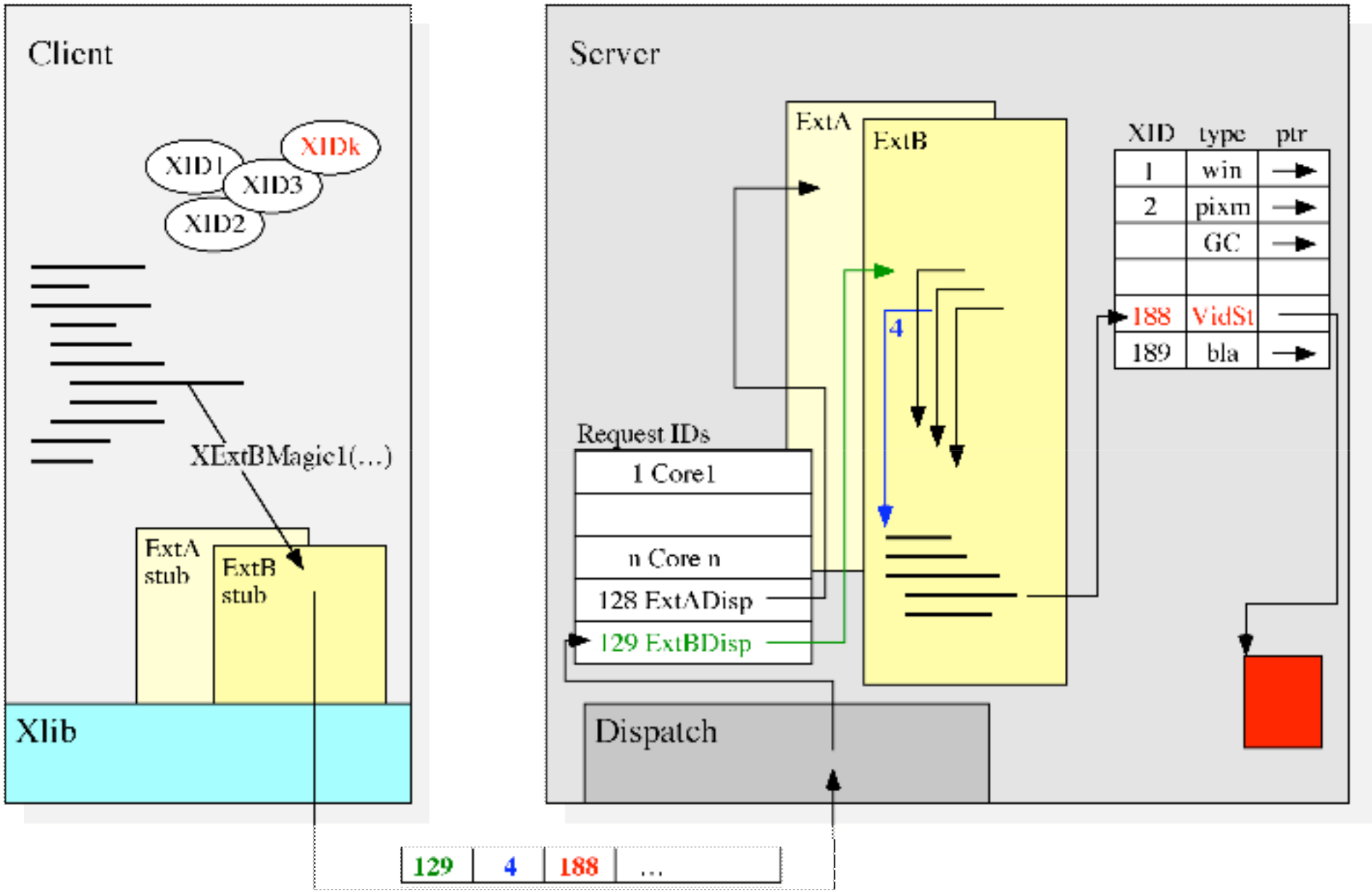
1.1.8 X Extension-Mechanismus

- X flexibel erweitern
  - artfremde Funktionalität (Audio, Video, OpenGL, JPEG, …)
  - Patches und Hacks
  - platformabhängige Funktionen (Hardwarefeatures)
  - Konsortialkonflikte
  - <-> Xn+1? (Render, Compositing, …)
- Randbedingungen
  - Mitbenutzung von Mechanismen (Protokoll, Adressierung)
  - Zugang zu relevanten Ressourcen
  - Management von Ressourcen (XID!)
  - Interaktion mit 'Kern'
  - dynamisches Binden
- 'Standard'-Extensions
  - bei fast allen X-Servern mitgeliefert
  - BIG-REQUESTS, MIT-SHM, GLX, SECURITY

- Programmieren und Benutzen
  - Shared Memory
  - R&R (RandR): Rotate und Resize
- Grafik
  - Render
  - GLX (OpenGL), …
- Brauchen wir das wirklich (noch)?
  - SHAPE, RECORD, XTEST
  - TOG-CUP
- Hacks
  - BIG-REQUESTS
  - XC-MISC
  - MIT-SUNDRY-NONSTANDARD (X11R3)
- XPrint?

```
number of extensions:    28
    BIG-REQUESTS
    DOUBLE-BUFFER
    DPMS
    Extended-Visual-Information
    FontCache
    GLX
    LBX
    MIT-SCREEN-SAVER
    MIT-SHM
    MIT-SUNDRY-NONSTANDARD
    RECORD
    RENDER
    SECURITY
    SGI-GLX
    SHAPE
    SYNC
    TOG-CUP
    XC-APPGROUP
    XC-MISC
    XFree86-Bigfont
    XFree86-DGA
    XFree86-DRI
    XFree86-Misc
    XFree86-VidModeExtension
    XInputExtension
    XKEYBOARD
    XTEST
    XVideo
```

- Extensions beim XServerstart gebunden
  - /etc/X11/XF86Config
  - Server lädt Extension
- Extension.Initialisierungsroutine
  - Resourceklassen registrieren
  - Request-ID besorgen (Code -> Ptr)
  - Event-Code besorgen
- XServer verteilt Requests auf Extensions
  - `(*dispatchertable[requestID])(packetdata)`
  - jede Extension hat Unter-Dispatcher
  - `(*evtdispTbl[packetdata->minorrequestID])(packetdata->rest)`
  - Prozedur dekodiert Paketdaten
  - Resourcen werden über XIDs gefunden
  - `myPixelPtr=SecurityLookupByType(XID,RT_PIXMAP,…)`
- Xlib: ListExtensions und QueryExtension
- Extension Stub im Client
  - Abbildung Extension -> RequestID pro Display
  - Stub kodiert Minorrequest und Paketdaten
  - Xlib zum Versand

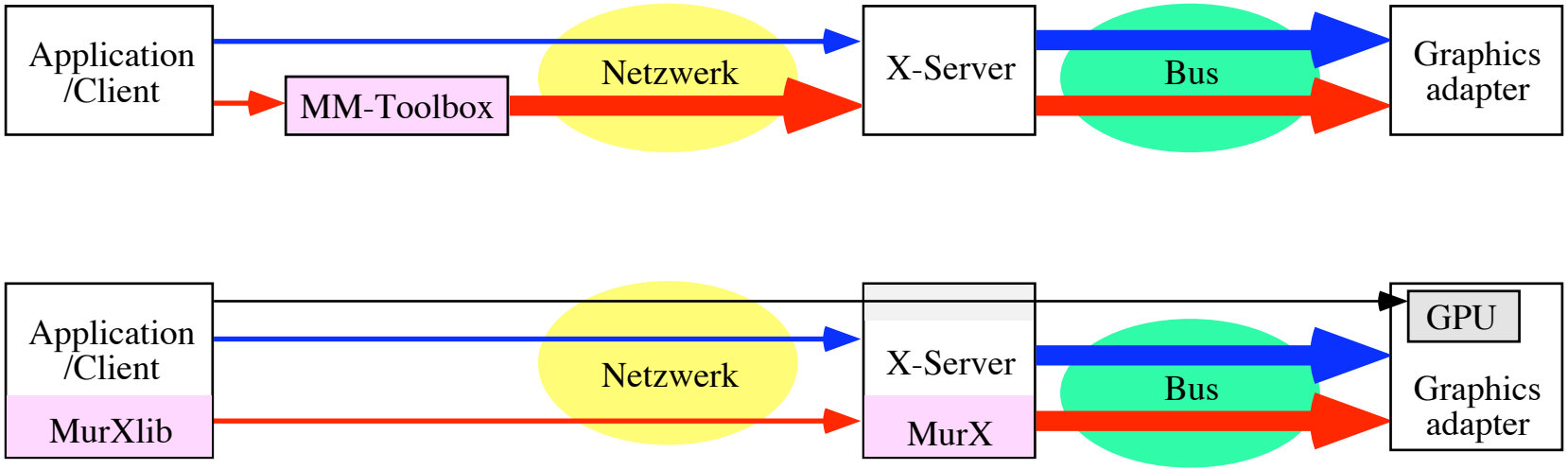

Client
XIDk
XID1
XID3
XID2
XExtBMagic1(...)
ExtA stub
ExtB stub
Xlib
Server
ExtA
ExtB
XID
type
ptr
1
win
2
pixm
GC
188
VidSt
189
bla
4
Request IDs
1 Core1
n Core n
128 ExtADisp
129 ExtBDisp
Dispatch
129
4
188
...

- Beispiel: Anlegen einer PIXMAP im Shared Memory

```
XShmSegmentInfo s; /* handle für shm Segment */
Pixmap p;
/* Legende: Extension, Betriebssystem, Applikation */

if (XShmQueryExtension(mydisp)) { /* extension vorhanden ? */
  /* shared memory segment erzeugen (syscall) */
  s.shmid=shmget(0, 16*1024*1024, IPC_CREAT|userRead|userWrite);

  /* einblenden im Prozess, liefert Addresse */
  s.shmaddr=shmat(s.shmid, 0, 0 /* read-write */);

  /* XServer über Segment informieren und als Pixmap verwenden */
  p = XShmCreatePixmap(mydisp, myrootwindow, 0, &s,
    1024 /*width*/, 1024 /*height*/, 16 /*depth*/);

}
else { /* ohne shared memory ueberleben */ }
```

1.2 MurX

- Network Multimedia Toolbox
  - QuickTime, Windows ?
  - viele Player-Applikationen für Linux



- Multimedia-Erweiterung für XWindow
  - Eingebettet in X (Extension-Mechanismus)
  - *nicht* expandierte Datenströme zwischen Client und Server
  - Audio, Video, …
  - verteilte Berechnungslast
  - Steuern – Übertragen - (De-)Kompression – Präsentation
  - Abstrakte Medientoolbox (Sequence Grabber, Player, …)

- Medienströme
  - komprimierte Pixel, Soundframes
  - X-Grafik
  - Bildkonstruktionskommandos
  - Synchronisationsstrom
- Synchronisation am playout-Point
  - Synchronisationsinformation vom Client (policy)
  - Synchronisation im MurX-Server (mechanism)
- Klientenfunktionen
  - kodierte Ströme zerlegen
  - zeitliche und räumliche Bezüge herstellen und kodieren => Policy
  - Toolbox mit ähnlichen Aufgaben wie QT
- Serverfunktionen
  - Synchronisation
  - Dekompression
  - Pixelmanipulation, Sampletransformation (Filter, …)
  - Blend, Audiomixer

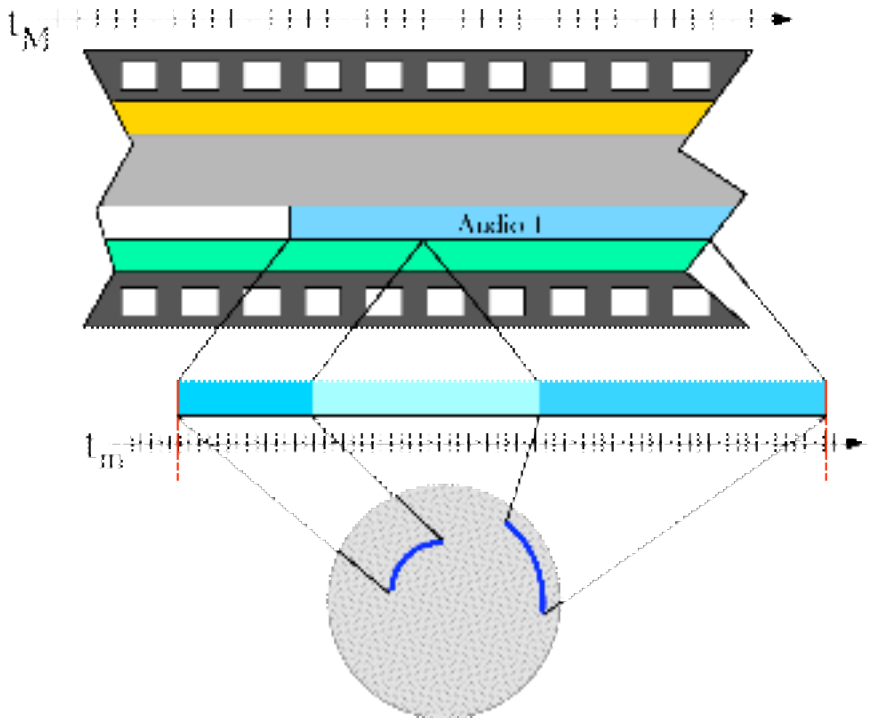
-> Einschub MurX

2. Multimedia (Dokumenten-)Formate

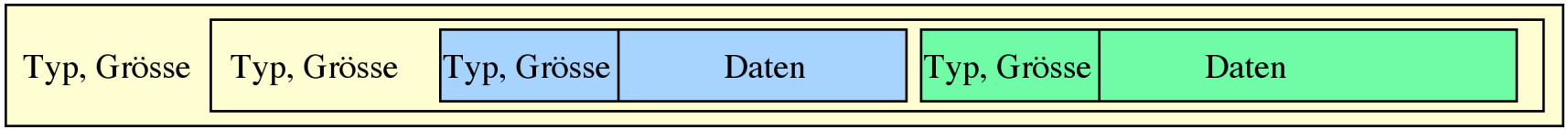
- Speicherung
  - viele heterogene Medienelemente (2D, 3D, 2.1D, … , 3.3D)
  - Synchronsiation und Komposition
  - streamingfähig
- Medien
  - Samples, Pixel, DCT-Blöcke, Chunks, …
  - Zeichenoperationen, Text, …
  - variable oder konstante Größe der Stücke
  - Attribute und Metainformation (Zeit, Kompression, …)
- Interaktion
  - Navigation (FF, …)
  - inhaltsbasiert -> Präsentationen, HyperMedia
  - programmgesteuert: extern, intern, objektorientiert
- Missverständnisse
  - MM-Dokumentenformat ≠ Kompressionsverfahren
  - mp4, QMF, .avi, RealVideo, …
  - xvid, DivX, 3ivx, …
  - Flash, Shockwave

2.1 QMF - QuickTime Movie Format - Revisited

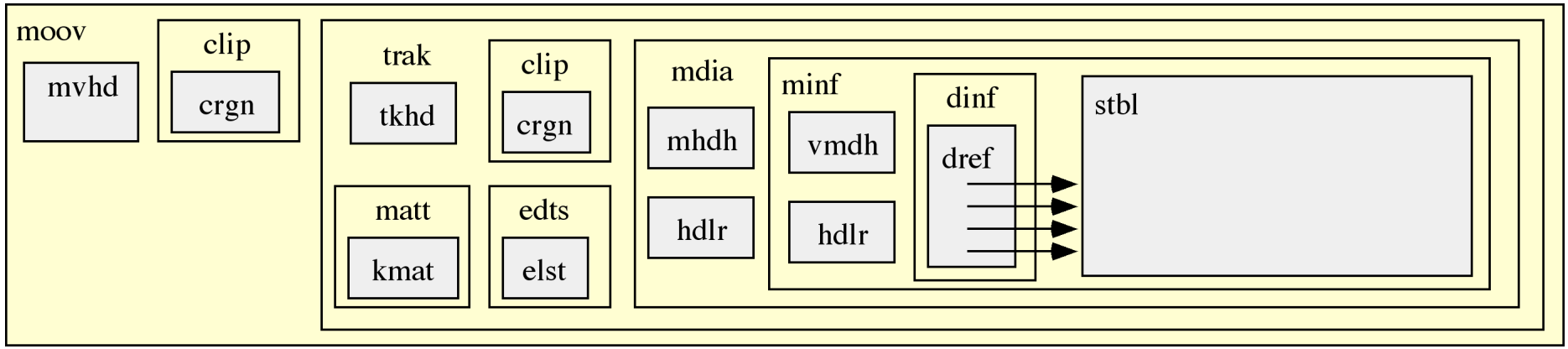
- Dokumentenformat Apple's QuickTime
  - Multimedia-Filme
  - Video, Audio, Text, Grafik, …
  - Präsentationsinformation (Transformationen, Zeitbezüge, …)
- MPEG-4 Dateiformat
  - Speicherung auf dem Server
  - Austausch zwischen Servern
  - ≠ Stromformat
- Movie ist eine *Datenstruktur*
  - Metadaten
  - Anzahl Tracks, Kompressionsverfahren, Zeitbezüge
  - Verweise auf Mediendaten (Samples, Video-Bilder)
  - Mediendaten im File oder in anderen Dateien



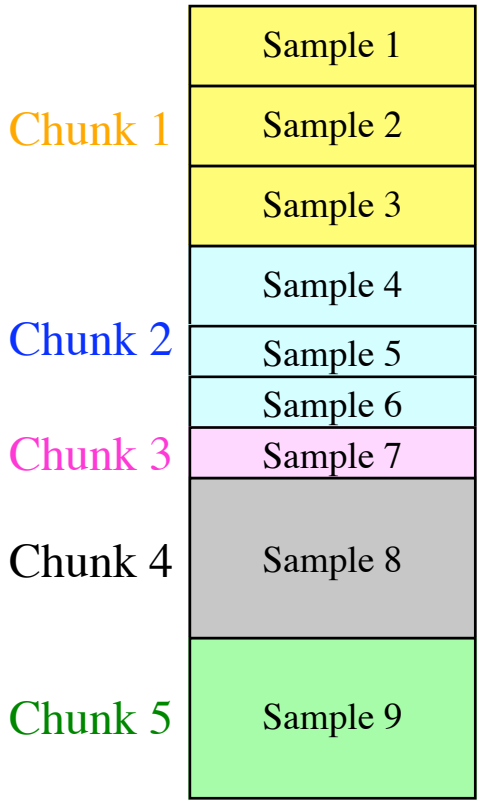
- Information in Atomen
  - Behälter-Atom, Blatt-Atom



  - Grösse in 32 bit
  - Typ 4 Zeichen ('clip', 'trak', 'moov', …, 'free', 'skip')
- QT-Atoms komplexer und flexibler
  - Typ, Grösse, Nummer: 32 bit
  - Anzahl Kinder: 16 bit
- Atome unsortiert in der Datei
  - Strukturinformation im Data- oder Resource-Fork
  - Sample-Daten im Data-Fork

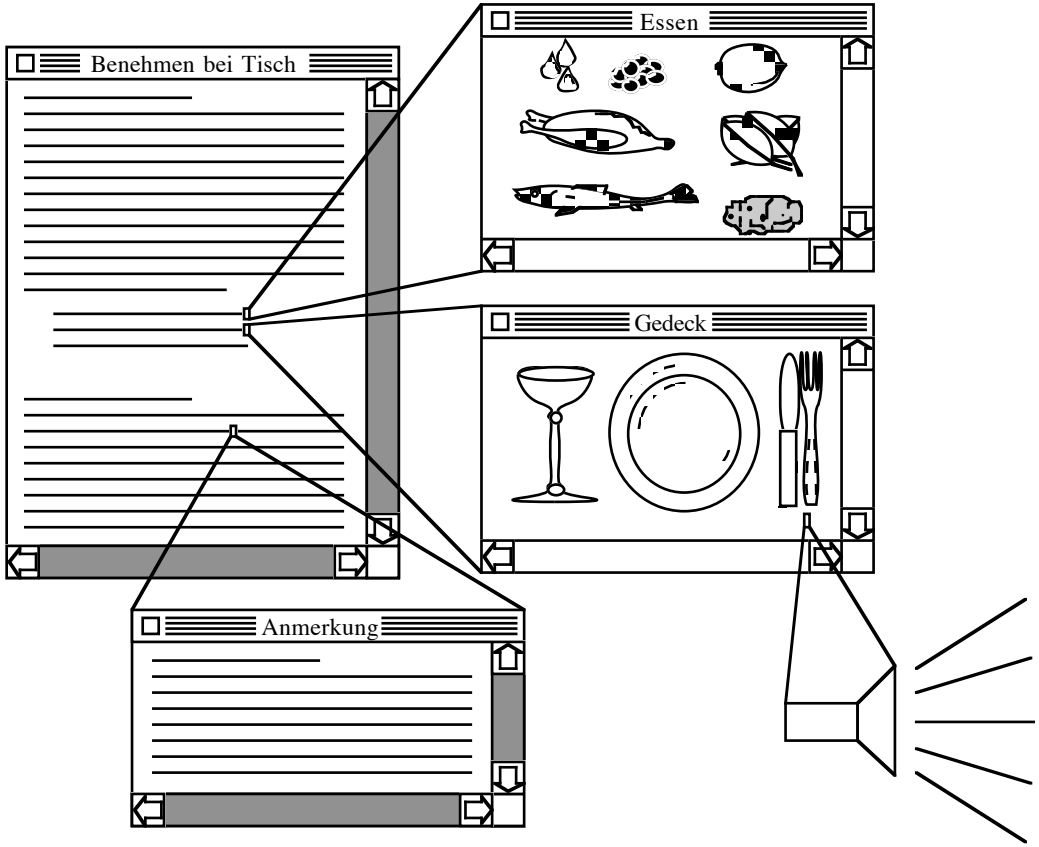


- Movie Header Atom
  - version, flags, creation time, modification time
  - duration, preferred rate, preferred volume, matrix
  - preview
  - Variablen: selection, current time, next track
- Edit List
  - Medienteile zur Präsentation
  - offset und Dauer
- Data Reference Atoms ('dref')
  - Anzahl Einträge, Referenzen
  - Referenz: 'alis'|'rsrc', flags
  - Pfadname oder Res-Typ und Res-Id
- Sample Table Atom ('stbl')
  - time-to-sample (Anzahl Samples, Sample-Dauer)
  - sync samples (Indices der Keyframes)
  - sample-to-chunk
  - sample size (Tabelle mit Sample-Größen)
  - chunk offset (Tabelle mit chunk-Anfang in Bytes)

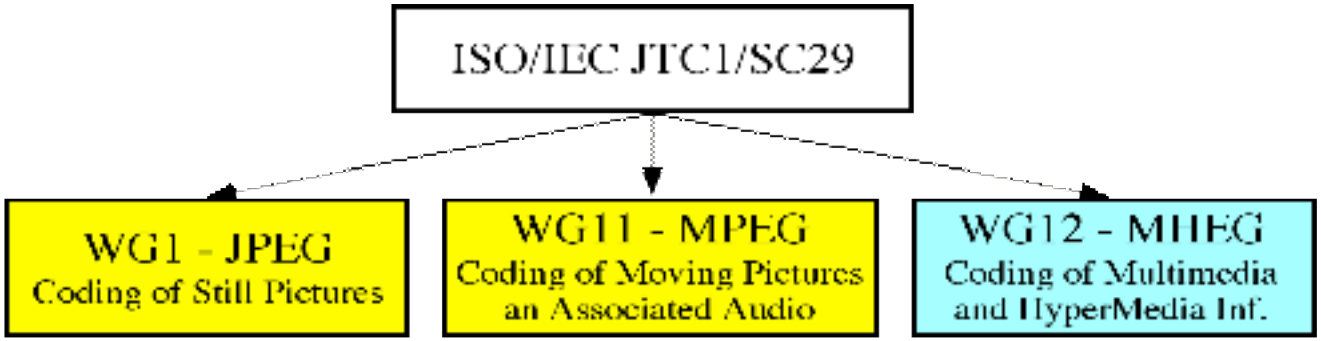


2.2 Multimedia Hypertext Expert Group (MHEG)

- Hypertext/Hypermedia
  - Nichtlineare Dokumente
  - logische Dokumentenstruktur = Graph
  - Knoten enthalten Inhalt:
    Medien (Text, Grafik, Video, Audio, …)
  - Kanten verbinden Knoten
    Links zwischen Elementen
    Knöpfe, "sensitive" Worte
  - Navigation
    Kanten verfolgen
    Backtrack

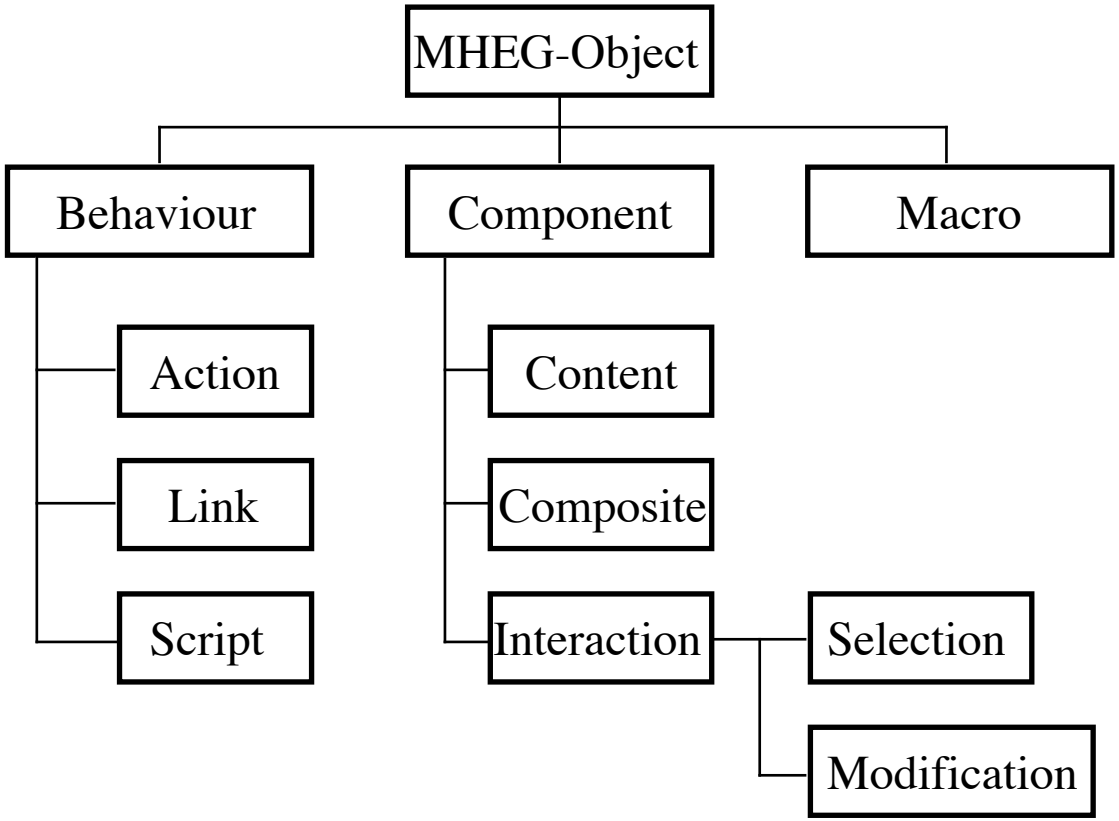


- ISO-Working Group



- Platform-unabhängig
  - minimale Hardwarekonfiguration
- Präsentationsformat
- Container für Daten: MPEG, JPEG, AIFF, …
- ASN.1 (http://www.oss.com/asn1/dubuisson.html) oder SGML für Austausch
- Heterogen
  - Byte-ordering
  - Datentypen
  - externe Referenzen
- 4-dimensionaler Raum (generic space)
- Objekt-Modell

- MHEG - Objekte
  - Content Class: Monomedia
    Parameter: Kodierung, Applikationsdaten
  - Selection Class: Button, Menu
  - Interactive object
    (Zusammengesetztes Objekt mit Selection und Content Objekten)
  - Hyperobject (composite object)
    Selection und Content Objekte mit Links

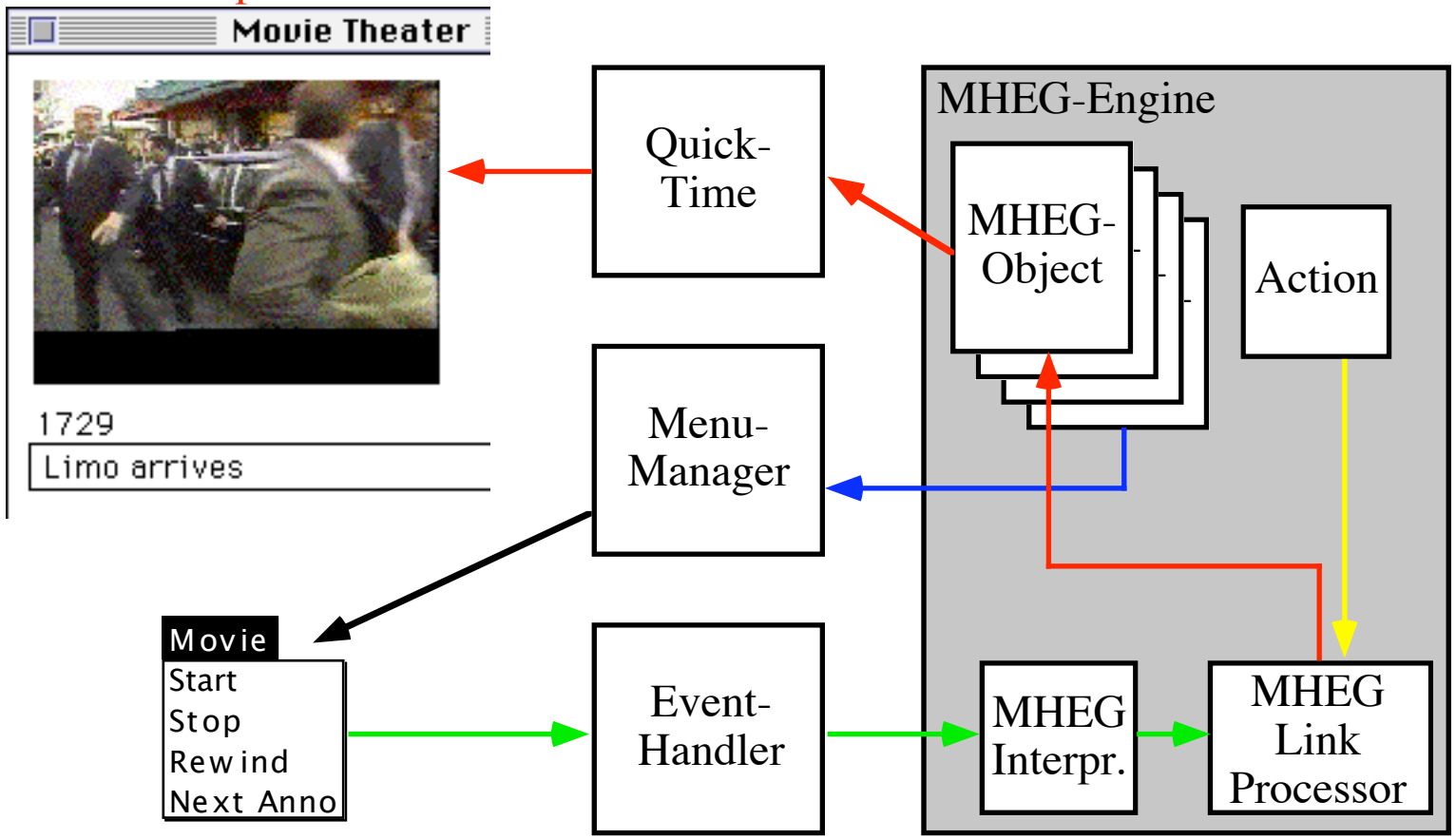


- Behaviour: Action-Class
  - Elementare Aktionen
  - set position, set size, set volume (…, transit duration)
  - run, …
  - parallel oder seriell
- Behaviour: Link-Class
  - Beziehung Trigger - Action

- Component: Interaction
  - Selection: Auswahl (Buttons, Menu, …)
  - Modification: Werteingabe (Slider, Scroller, Eingabefeld…)
- Component: Composite
  - Zusammengesetzte Objekte
  - Referenzen auf andere Objekte

- Beispiel:

  Selection erzeugen, Menu ableiten und Anzeigen
  Menuauswahl durch User, Interpretieren, Link überprüfen
  Action auswerten
  Run-Action an Content-object
  Dekodieren und abspielen



2.3 SMIL

- Synchronized Multimedia Integration Language
  - 'presentation level interactive multimedia'
  - W3C; aktuell SMIL 2.0
  - XML-Sprache
  - Hypermedia -> linkbasiert
  - http://www.smilguide.com/guide/tutorial/learning-to-smil
  - Player: Ambulant, GRINS, RealPlayer10

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
                      "http://www.w3.org/TR/REC-smil/SMIL20.dtd">
<smil xmlns:"http://www.w3c.org2001/SMIL20/Language">
  <head> … </head>
  <body> … </body>
</smil>
```

- Trennung Präsentationsinformation - Inhalt
  - present.sml, xyz.smil
  - Anordnung in Zeit und Raum (Fenster)
  - Verhalten von Hyperlinks gesteuert
  - <ref> zeigt auf Medienobjekte
  - img, video, audio, animation, text, textstream Spezialfall von <ref>

```
<video src="movie.mpg" clipBegin="10s"/>
```

- Grundregeln
  - Tag + Attribute (src, clipBegin, Begin, …)
  - id="…"
  - parent-children
- Raumaufteilung: Regionen
  - Höhe+Breite oder Links+Oben+Rechts+Unten

```
<region id="CIF-PAL" width="352px" height="288px"/>
```

  - fit-Attribut: hidden, scroll, fill, meet, slice
  - z-Index
  - source-Tags füllen Regionen mit Inhalt

```
<video src="movie.mpg" region="CIF-PAL" clipBegin="10s"/>
```

- Raumaufteilung: Layout
  - mehrere Regionen
  - in <head>

```
<head>
  <layout>
    <region id="CIF-PAL" width="352px" height="288px"/>
    <region id="photo" width="6in" height="4in"/>
    <region id="cntr" left="25%" top="25%" width="50%"
            height="50%"/>
  </layout>
</head>
```

- Serielle Präsentation
  - <seq> {Medienreferenzen} </seq>
  - eventuell verschachtelt auch mit <par>
  - meist implizites Timing

```
<body> …
 <seq>
  <video src="car.mpg"/>
  <video src="razor.mpg"/>
  <video src="phone.mpg"/>
 </seq>
… </body>
```

- Parallele Präsentation
  - <par> {Medienreferenzen} </par>
  - eventuell verschachtelt, auch mit <seq>
  - Timing auch unter einander abgeleitet (Dauer, Beginn)

```
<par>
  <video id="clip1" src="car.mpg"/>
  <video src="wheels.mpg" begin="clip1.end+00:01"/>
  <video src="gas.mpg"/>
</par>
```

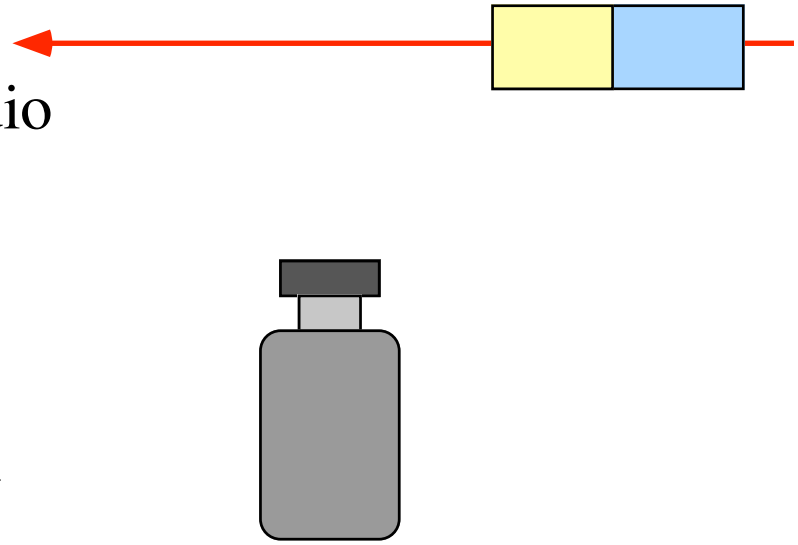
- Anchor-tag
  - <a> ähnlich HTML
  - Attribute: sourcePlaystate, destinationPlaystate, sourceLevel, …
  - andere Smile-Datei: `<a href="second.smil" show="replace">`
  - zu Konstrukt im selben SMIL-File mit id:

```
<a href="#hint" sourcePlaystate="stop"/ >
…
<par id="hint">
   <text region="hintreg" src="texts/bla.txt"/>
   …
</par>
```

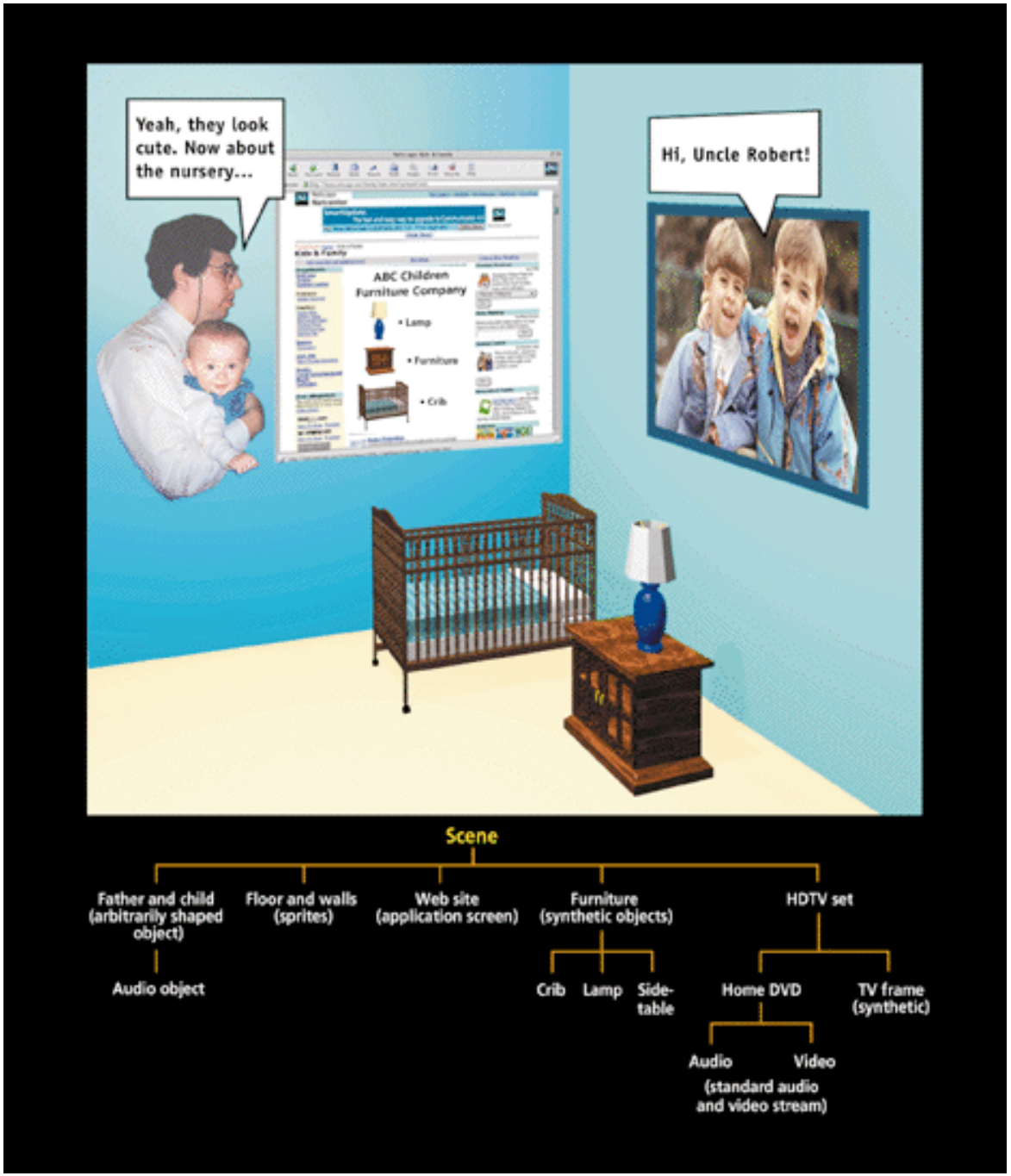
  - Weblink
- Area-tag
  - ähnlich image-map
  - mehrere Links in einem Objekt (Video, Bild, Grafik)
- Profile
  - Geräte (~Player) unterstützen nicht alle Elemente und Attribute
  - SMIL 2.0 Language: volle Funktionalität
  - SMIL 2.0 Basic: low-power devices, Minimalfunktionen
  - 3GPP Mobile: ohne Animation und eingeschränkte Interaktion
  - XHTM+SMIL, SMIL 1.0

2.4 MPEG-4

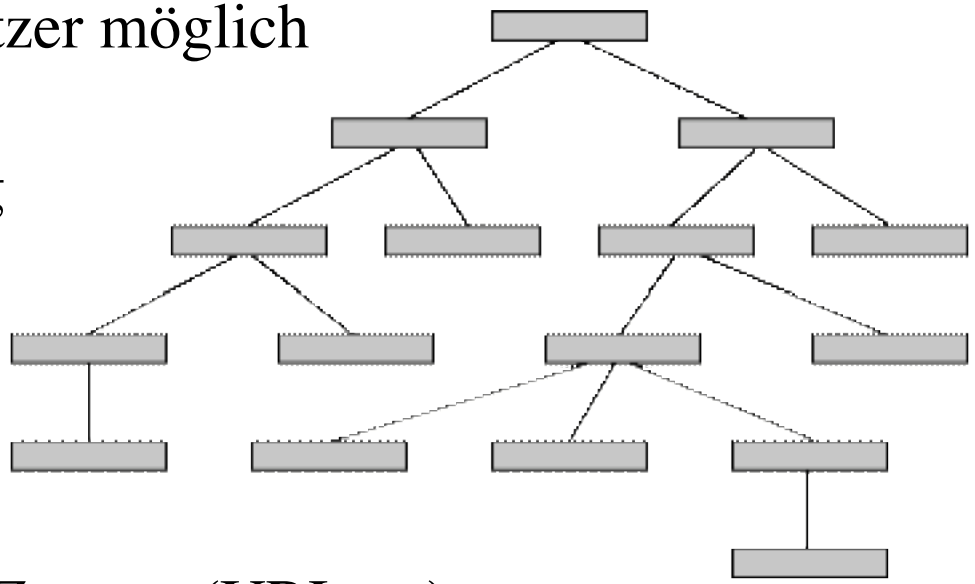
- MPEG
  - JPEG + Differenzkodierung + mpn-Audio + Multiplexer + …
- MPEG-1
  - Kompressionstechniken für Video und Audio
  - Stromformat
  - 'implizite' Synchronisation
- MPEG-2
  - flexibleres Stromformat und Profile
  - Synchronisation durch einfache Zeitachsen
- Wie gut ist die 'Bewegungskompensation'?
  - Objekte bewegen sich Ansichtswinkel-konstant zur Kamera
  - akzeptabel für Kameraschwenk+Landschaft
  - gleiche Behandlung für langsame und schnelle Bildteile
- Kompressionsleistung
  - Objektzerlegung und Semantikwissen
  - verbesserte Kodierung (Prädiktion …)



- Szenen im *Dekoder*
  - Menge von Objekten
  - künstliche Objekte
  - Video-Objekte
  - Bilder
  - Grafiken
  - Audioströme
  - Interaktionselemente
- Komposition
  - 4D-Position
  - Transformation
  - Programm
- Kodierung
  - fehlertolerant
  - skalierbar



- Szene
  - Objekthierarchie
  - Hintergrund
  - mehrere Vordergrund-Objekte
  - 'composition information'
  - Objektanordnung sogar durch Benutzer möglich



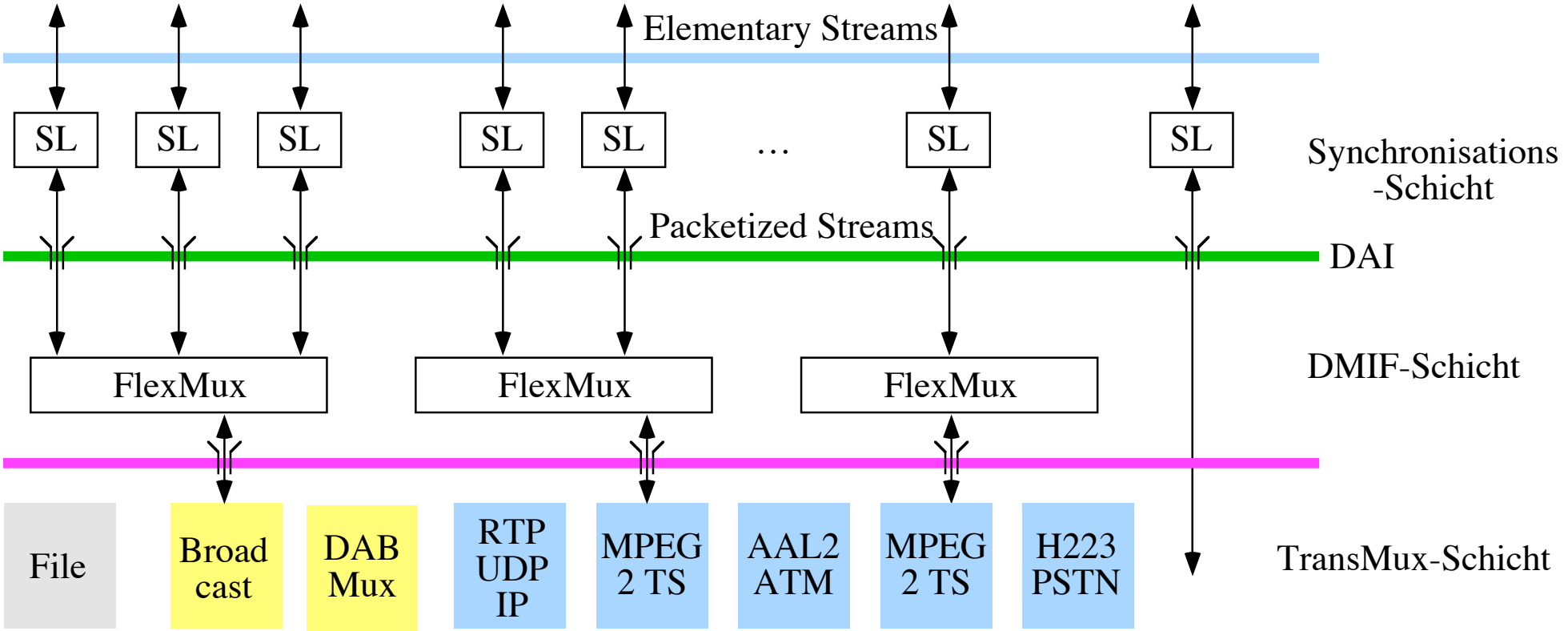
- Media objects (audio-visual objects)
  - zeitliche und räumliche Ausdehnung
  - hörbar und sichtbar
  - Text und Graphik
  - synthetisches Audio
  - zusammengesetzte Objekte
- Object descriptor
  - Objectbeschreibung: ES, Semantik, Zugang (URL …)
  - Strombeschreibung: Decoder, QoS, …
  - SDL: Parserbeschreibungs-Sprache

```
align bit(32) picture_start_code=0x00000100
```

- Elementary Streams (ES)
  - Datenströme für Objekte (Samples, Animation, Text, Szenen, …)
  - Zeitstempel

2.4.1 Transportmodell

- Synchronisation Layer
  - Zeitstempel und Individual Access Units

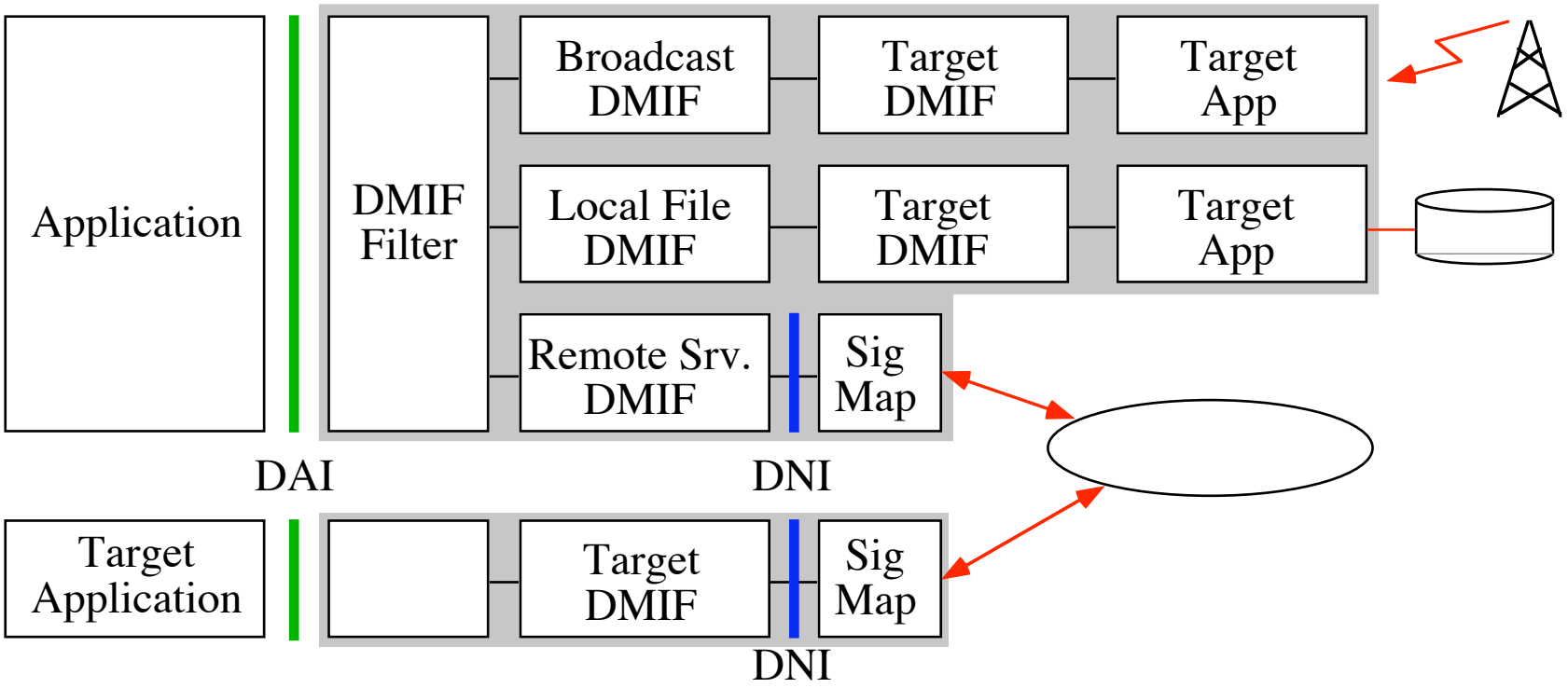


- Elementary Streams
  - Medien-Daten
  - Kommando-Transport (Play, Pause, …)

- Delivery Multimedia Integration Framework (DMIF)
  - Abstraktion für Netzwerke, Disks, Broadcast
  - DAI: DMIF (Delivery) Application Interface
  - Session-Protokoll mit Local-DMIF, Remote-DMIF, Remote-App
  - Remote-Komponenten evtl. local simuliert (Disk, Broadcast)
  - FlexMux-tool, andere Tools möglich
  - interleaving von ES's zu sinnvollen Gruppen
  - z.B. QoS-gesteuert
  - wenige Netzwerkverbindungen
- TransMux-Schicht
  - erbringt Transportleistung
  - nur das Interface ist in MPEG-4 spezifiziert
  - leer, leicht- oder schwergewichtig
- Sync Layer
  - liefert Datenelemente (Elementary Streams) zur richtigen Zeit
  - Zeitstempleauswertung
  - Füllen der Decoder-Buffer und Buffer Management

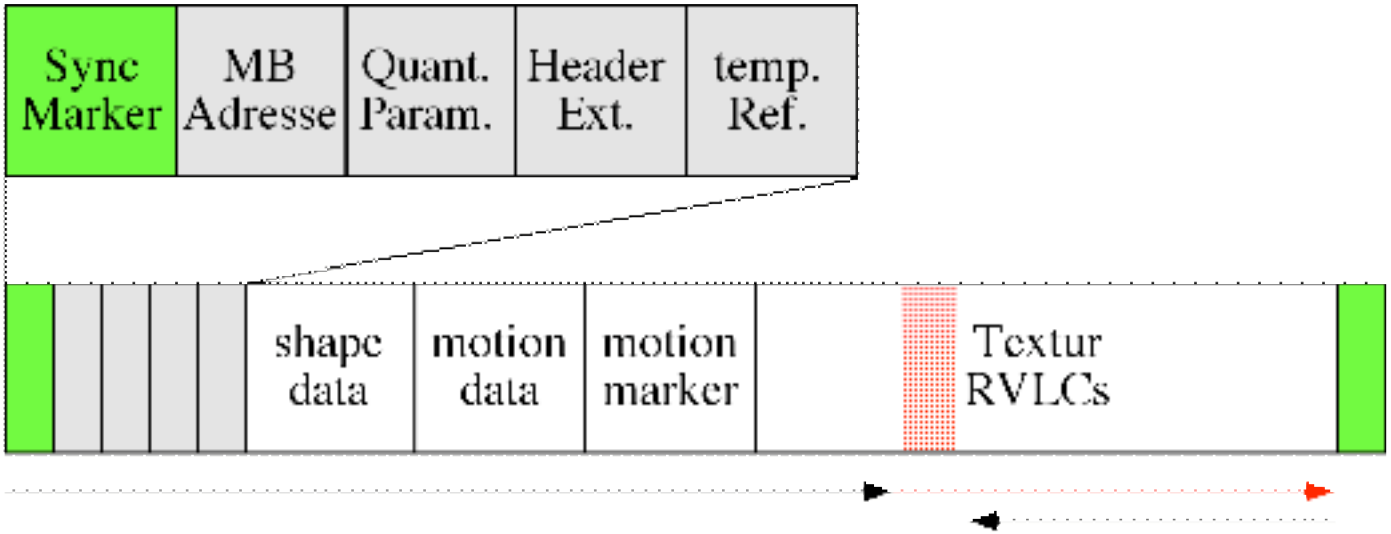
- Service
  - **DA_ServiceAttach(ParentSessionId, URL, DataIn,SessionId, …)**
  - **DA_ServiceAttachCallback()**
  - **DA_ServiceDetach*()**
- Channel
  - **DA_ChannelAdd(SessionId, ChannelDesc,channelHandle, …)**
  - **DA_ChannelAddCallback()**
  - **DA_ChannelDelete*()**
- QoS Monitoring
  - **DA_ChannelMonitor(channelHandle, …)**
  - **DA_ChannelEvent(qosReport)**
- User Command
  - **DA_UserCommand(channelHandle, …)**
  - **DA_UserCommandCallback()**
  - **DA_UserCommandAck()**
  - **DA_UserCommandAckCallback()**
- Data
  - **DA_Data(channelHandle, streamDataBuffer, streamDataLen)**
  - **DA_DataCallback(channelHandle, streamDataBuffer, streamDataLen)**
  - **DA_Data(channelHandle, …, appDataBuffer, …)**
  - **DA_DataCallback(channelHandle, …, appDataBuffer, …)**

- DMIF Network Interface



- Dienstelemente (primitives)
  - Session (setup, release)
  - Service (attach, detach)
  - TransMux (setup, release, config)
  - Channel (add, added, delete)
  - User command (command, ack)

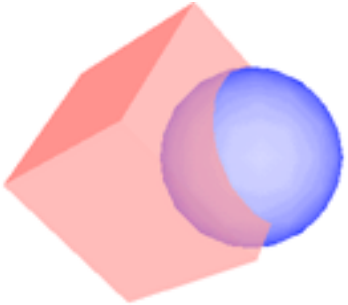
- Methoden zur Fehlererholung
  - Marker zur Resynchronisation
  - Reversible VLCs
  - Interleaving
  - Kodierung von Schichten (layered coding)



- Richtungsunabhängig dekodierbare Symbole (RVLC)
  - VLC: variable length codes
  - VLCodes mit fester Anzahl '1' (Hamming-Gewicht)
  - RVLC = Präfix+VLC+Suffix
  - Vorwärts dekodieren bis Bitfehler
  - zum Marker scannen
  - Rückwärts dekodieren bis Bitfehler

2.4.2 Szenenbeschreibung

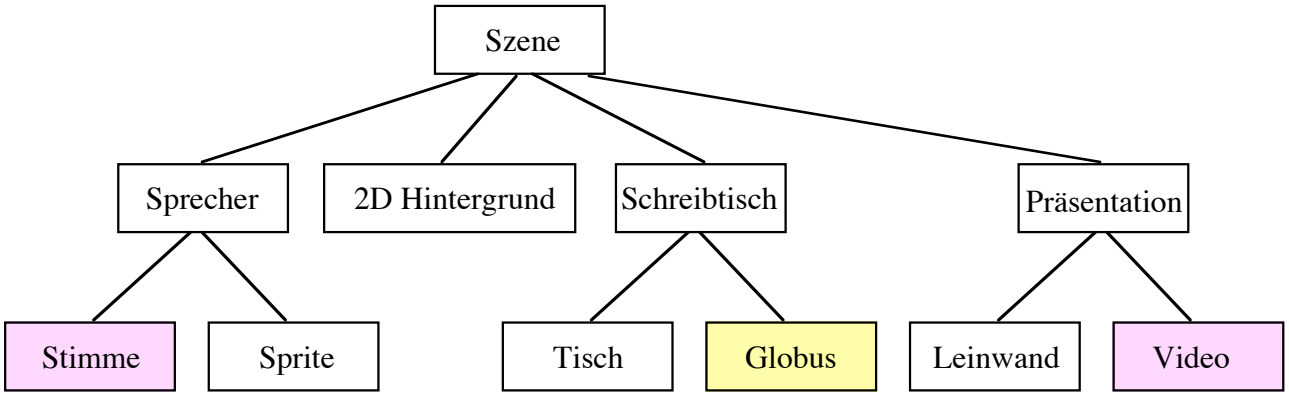
- Szenenbeschreibung
  - "Programmiersprache"
  - Objekte definieren
  - Eigenschaften ändern (Farbe, Leuchtkraft, …)
  - Objektgruppen mit relativer Position zueinander
  - Kamera, Beleuchtung
- VRML bzw. Web3D bzw. X3D
- Virtual Reality Markup Language
- Einfache ASCII-basierte Syntax
- VRML 97 (2.0)
  - dynamische Welten, Interaktive Kontrolle
- External Authoring Interface (EAI)
  - 3D-Welt aus externem Programm fernsteuern
  - Sensoren zur Kommunikation mit Programm

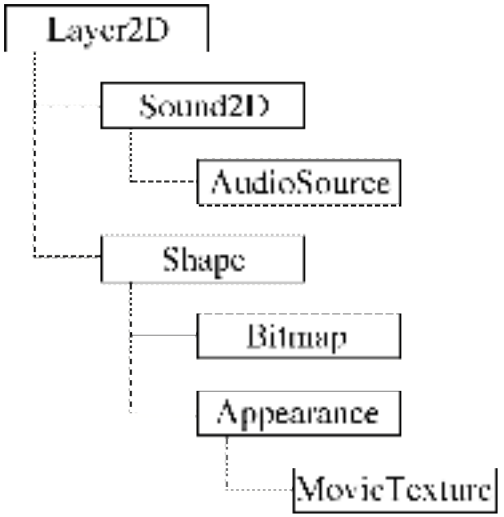


- Eine einfache VRML-Welt

```
#VRML V1.0 ascii
Separator {
   Separator {
      Transform {
         rotation 0 1 1  1.2
      }
      Material {
         emissiveColor 1 0 0
         transparency 0.3
      }
      Cube {}
   }
   Separator {
      Transform {
         translation 1 0 0
      }
      Material {
         emissiveColor    0 0 1
      }
      Sphere {}
   }
}
```

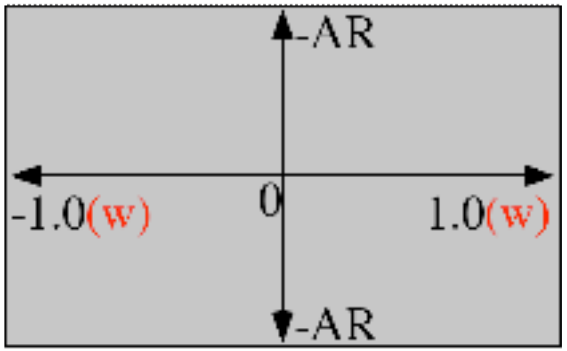
- BIFS: Binary Format for Scenes
  - 3D-Szenenbeschreibung
  - binär kodiert statt VRML-ASCII-Text
  - dynamische Szenen: Objekte hinzufügen und entfernen
  - Positionsveränderung ≠ motion vector



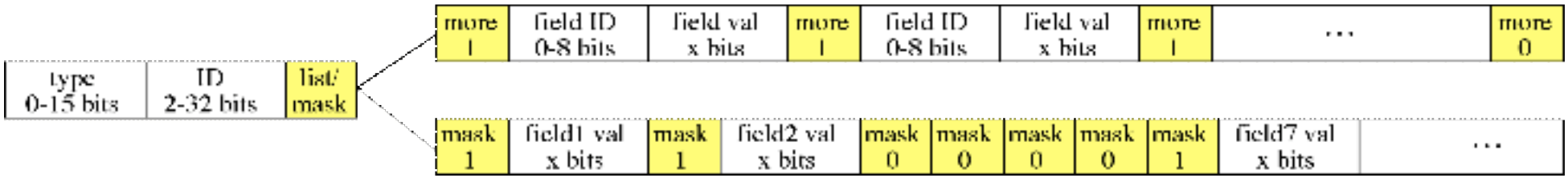


- Objektposition
  - globales und lokal Koordinatensysteme
  - Ausdehnung in Zeit und Raum
- Verändungen der Objektattribute
  - MPEG-J und Information im Elementary Stream
- XMT: eXtensible MPEG-4 Textual format: VRML/X3D bzw SMIL

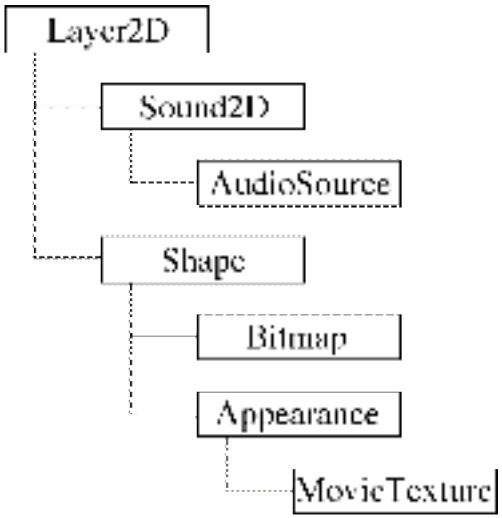
- Rendering-Kontext
  - Layer2D: root-Objekt für 2D-Szenen
  - Layer3D: root-Objekt für 3D-Szenen
  - CompositeTexture2D (…3D) off-screen
  - Transform2D, Transform
  - Koordinaten metrisch oder pixel



- Sound, Sound2D Knoten
- Shape-Knoten
  - geometry-Feld: `Rextangle, Circle, Box, Bitmap`
  - appearance-Feld: `Material, ImageTexture, MovieTexture`
- Sensoren, `TimeSensor`, Interpolatoren
- BIFS-Kommandos
  - `Insert/Delete` Knoten oder Felder in Knoten
  - `Replace` Szenen, Knoten, Felder, Routen
- Binärkodierung
  - Command Frames in Elementary Stream Access Units



• Einfacher Szenengraph in XMT-Form



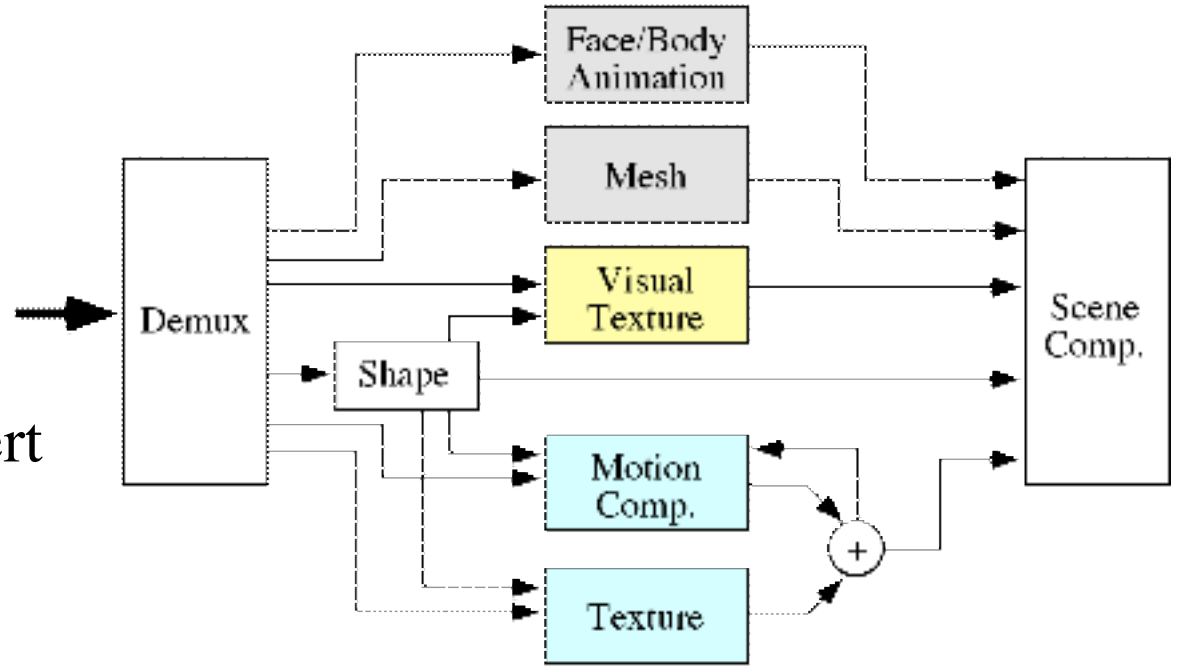

```
 1 <Replace> <Scene>                                          10
 2  <Layer2D> <children>                                      12
 3   <Sound2D> <source>                                       11
 4    <AudioSource url='od:3'
 5       startTime="0.0" stopTime="-1.0"/>                    94
 6   </source> </Sound2D>                                      1
 7   <Shape>                                                   8
 9    <geometry> <Bitmap/> </geometry>                         7
10     <appearance>                                           1
11      <Appearance> <texture>                                6
12       <MovieTexture url="od:4" loop="false"
13          <startTime="0.0" stopTime="-1.0"/>                93
14      </texture> </Appearance>                               2
15     </appearance>                                          1
16   </Shape>                                                 1
17  </children> </Layer2D>                                    1
18 </Scene> </Replace>                                       2
```

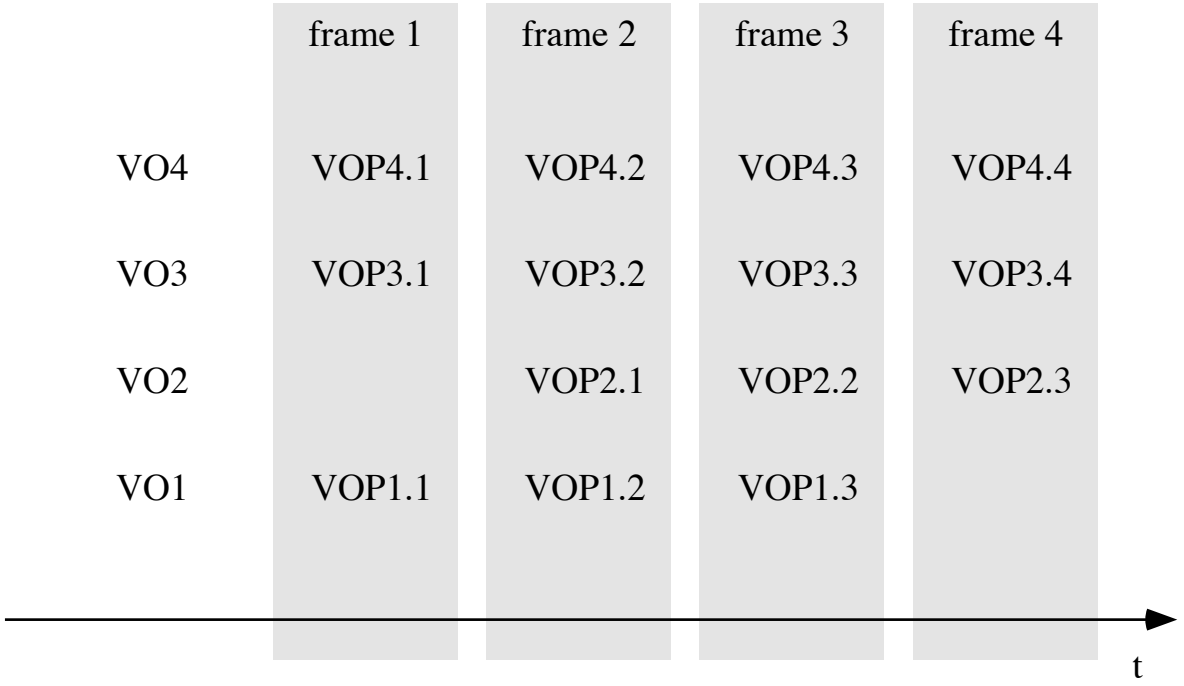
- BIFS-Grösse 250 Bit - 32 Byte
- Objektsdeskriptoren 3 und 4 enthalten Medienströme

2.4.3 Visual Objects

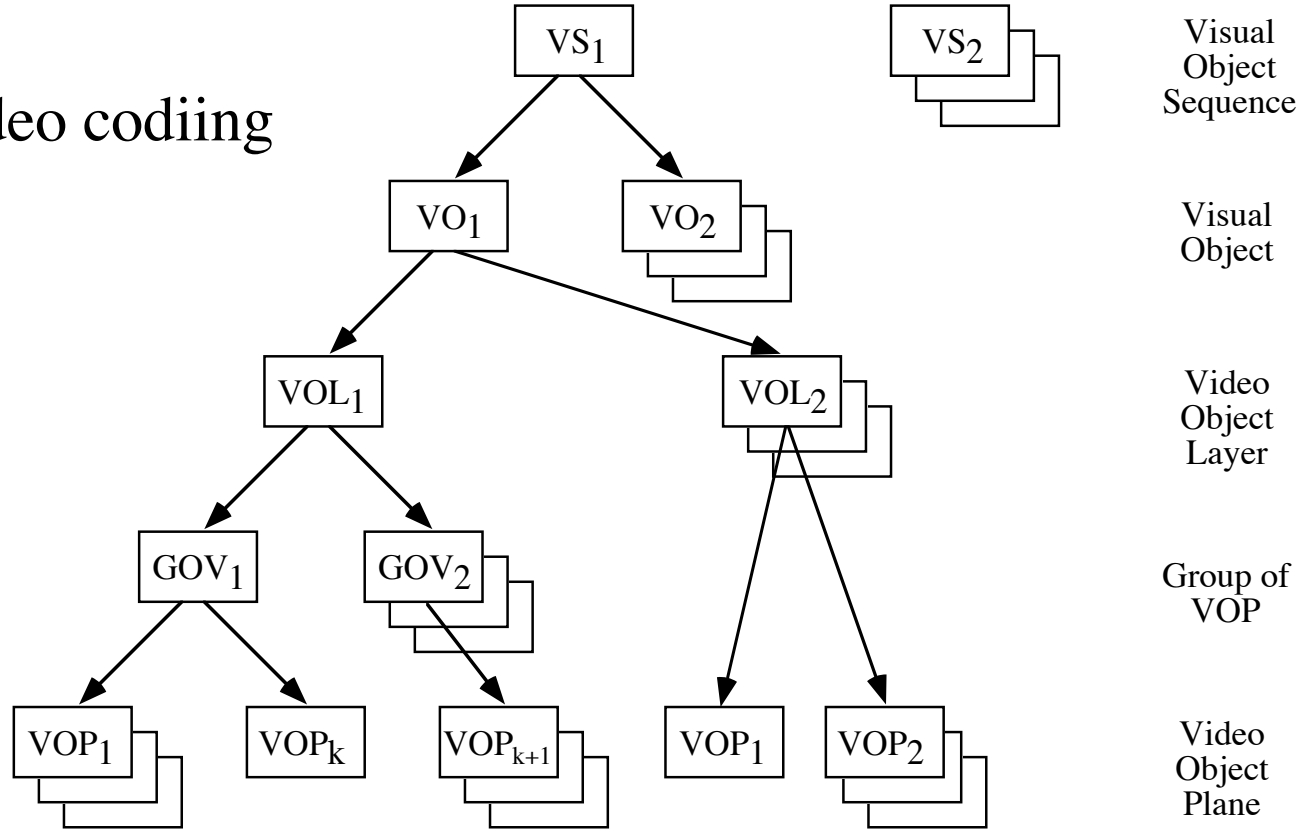
- Szenen-Komponente
  - individueller Zugang
  - einzeln manipulierbar
- Texture = Pixelinformation
- Video object
  - Texture wie MPEG1/2 kodiert
  - Bewegung (motion)
  - Form (shape)
  - trivale und komplexe Form

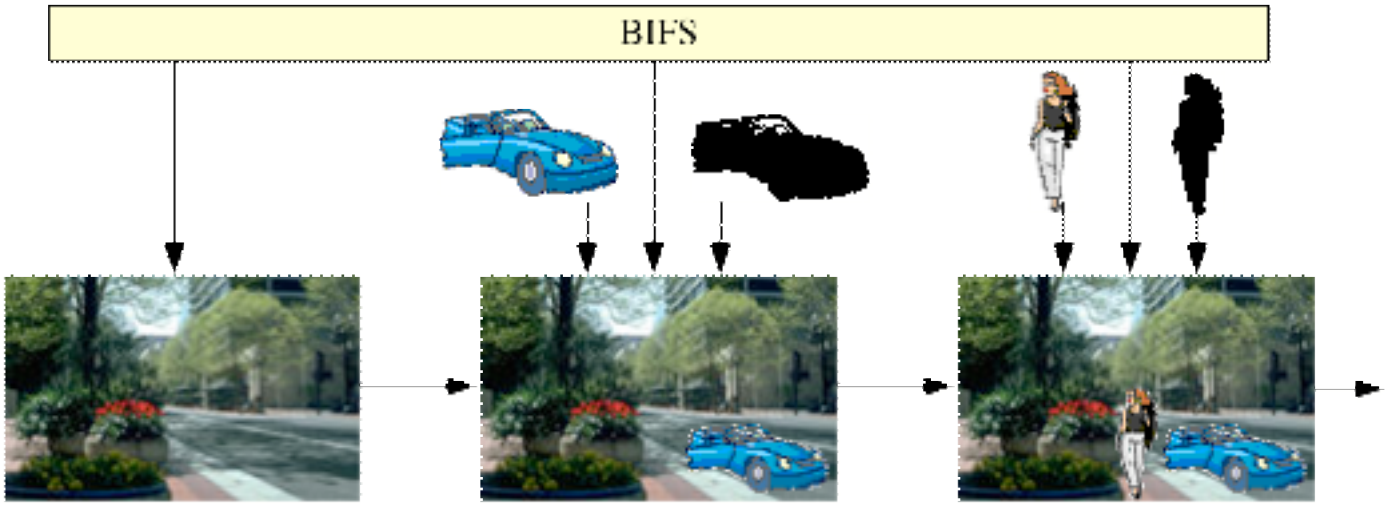
- Framekonstruktion
  - aus Video-Object-Planes
  - 'Prioritäten'
  - Hintergrund



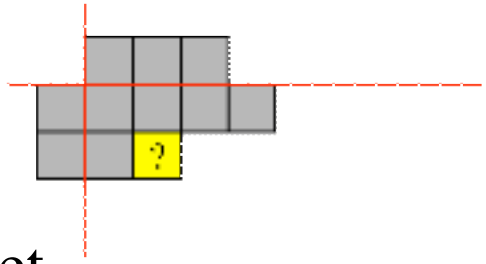


- Video Object Layer
  - mehrer bei scalable video codiing
  - Auflösung
  - Bildqualität
- GOP
  - Random Access
  - Re-Synchronisation



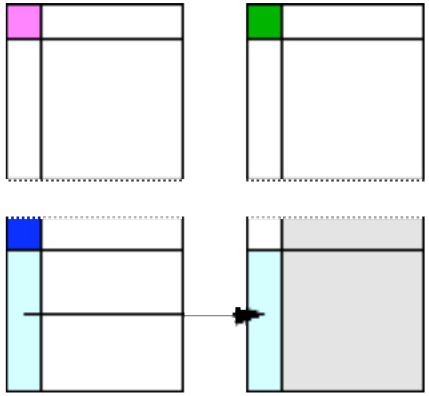


- Alpha-Kanal (shape)
  - kontrolliert Mischen der Videoobjekte
  - einfache Maske als Bitmap: Pixel einsetzen oder nicht
  - oder Transparenzwert
  - oder Funktion zur Pixelverknüpfung
- Shape-Kodierung
  - alpha-plane als Bitmap
  - umfassendes Rechteck und Position im Frame
  - evtl. globaler, nicht-binärer alpha-Wert
  - transparente und deckende Makroblöcke
  - Rand-Makroblöcke: binary alpha blocks
  - bab mit Praediktionstechnik vorhergesagt
  - Wahrscheinlichkeit wird vom ArithKoder verwendet

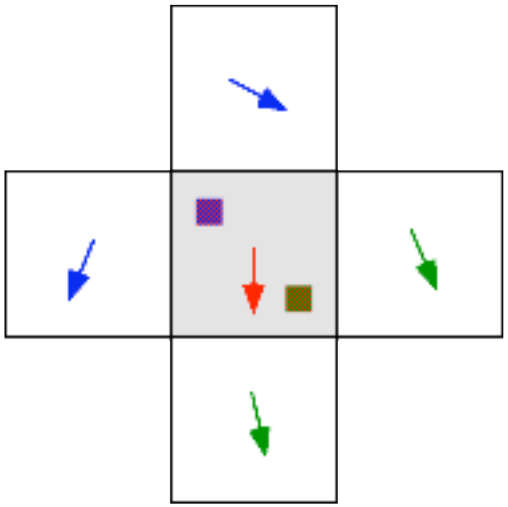


- Kodierung für 'echten' alpha-Kanal
  - binäre Hilfsmaske wie Bitmap-Maske kodiert
  - Texture-Coding für alpha-Blöcke
  - Rand-Makroblöcke mit besonderen Techniken

- Texture-Kodierung
  - I-VOP, B-VOP, P-VOP
  - GOV: Group of VOPs
  - inter- und intrakodiert
- Prädiktion in Intra-Bildern
  - für eine AC-Koeffizienten Spalte/Zeile
  - Auswahl durch DC-Differenzen (Gradient)
  - Praediktionsfehler für Spalte/Zeile kodiert
  - Rest vom Block DCT-kodiert
  - verschiedene Linearisierungs-Sequenzen



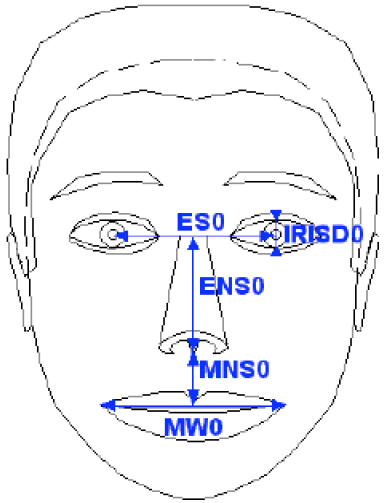
- 1 oder 4 Motion-Vektoren pro Makroblock
  - VOP-global motion compensation: 4 globale MVs
- Overlapped Block Motion Compensation
  - Pixelprädiktion und Bewegungsvektoren
  - Pixelprädiktion mit Nachbarvektoren
  - Pixel:= a_i*p[$nvec_1$]+ b_i*p[$nvec_2$]+ c_i*p[blockvec]-Rest
  - Nachbarvektorenwahl abhängig vom Quadranten
  - DCT über Reste
  - nicht in den gängigen Profilen

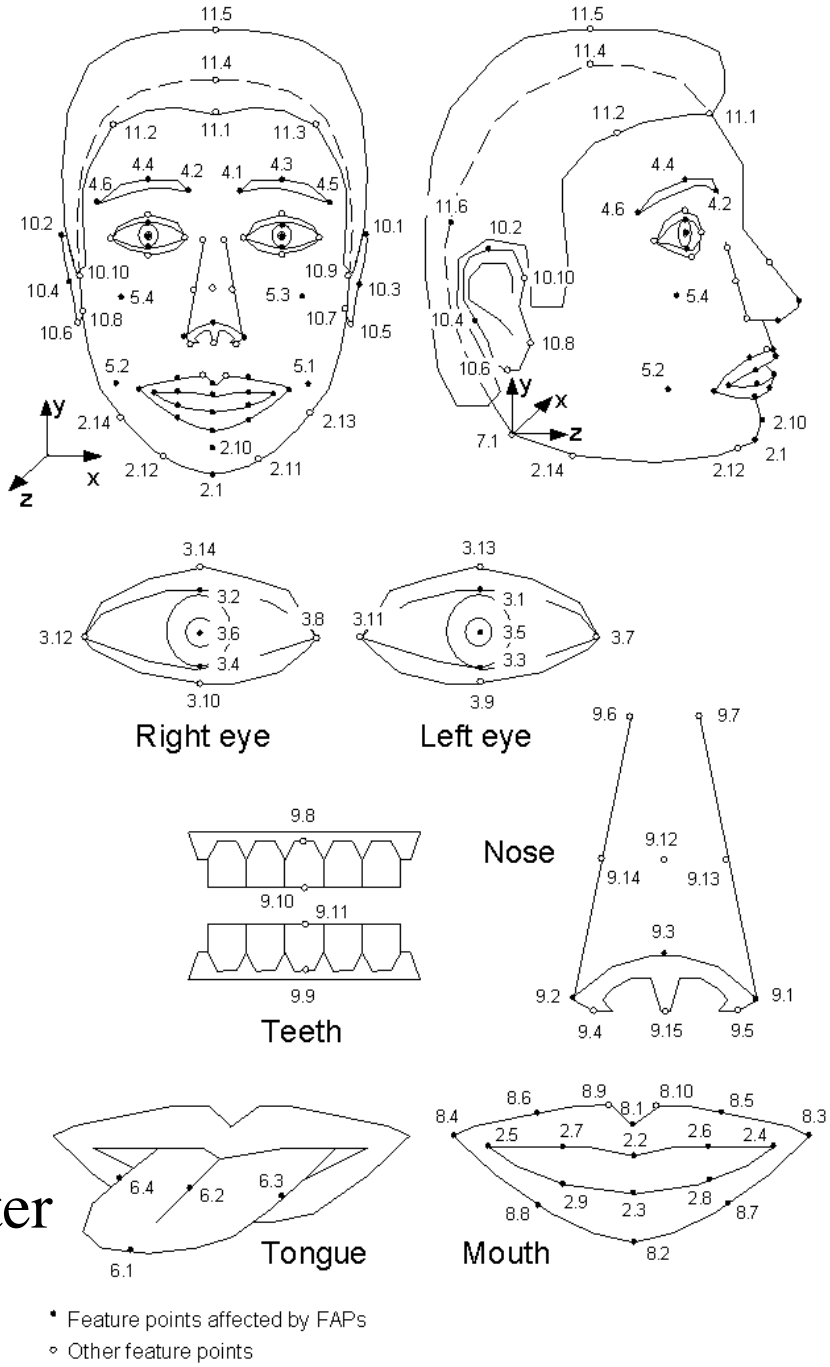


- Still texture object
  - Wavelet-Kodierung möglich
  - layered bitstreams: base, enhancement(s)
- Sprites

  - Sprite ändert sich nicht in der Zeit
  - übertragen in I-VOP
  - Ausschnitt im Bild sichtbar
  - static video object planes (S-VOP): 3-D-Vektoren für Sprite
  - Schwenk, Zoom, …
  - ->Hintergrund verändert sich scheinbar
- 2-dimensionale Gitter (2D-Mesh)
  - Dreiecks-Netz
  - Vektoren für Gitterpunkte -> verzerrtes Gitter
  - Prädiktion für Gitterpunktvektoren …
  - I-VOP-Textur wird mit dem Gitter verzerrt
  - P-VOP-Fehler-Textur addiert
- 3D-Meshes siehe BIFS/VRML

- Face animation object
  - Modell mit generischem Gesicht
  - FDP: Face Definition Parameters
  - BIFS-Mechanismen zur Modellübertragung
  - Default Modell im Decoder
  - FAP: Face Animation Parameters (68)
  - Übertragung der Parameter in FAPU

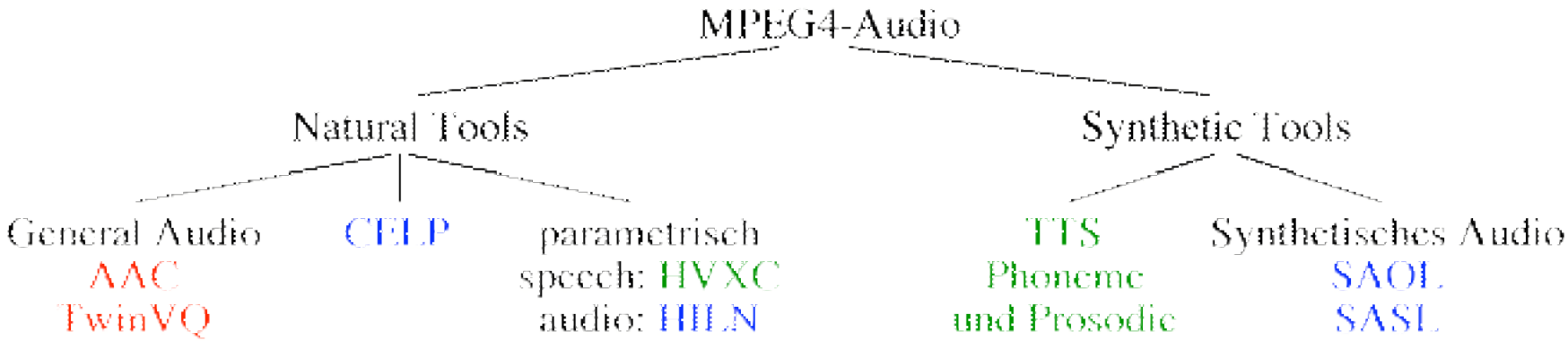




- Body animation object
  - 3D-Polygonnetze
  - BDP: Oberfläche, Grösse, Textur
  - Body Animation Parameters (168)
  - Standardposition frontal stehend
  - Mechanismus zur Übertragung der Parameter

2.4.4 MPEG-4 Audio (Überblick)

- AudioBIFS (siehe 2.4.6.1)
- Coding Tools
  - De-Kompressionsverfahren: parametrisch oder psychoakustisch
  - Synthese



- Bitraten pro Kanal : 2-4, 4-16, 16-64 [kbit/s]
- Siehe 2.4.6

2.4.5 Profile in MPEG-4

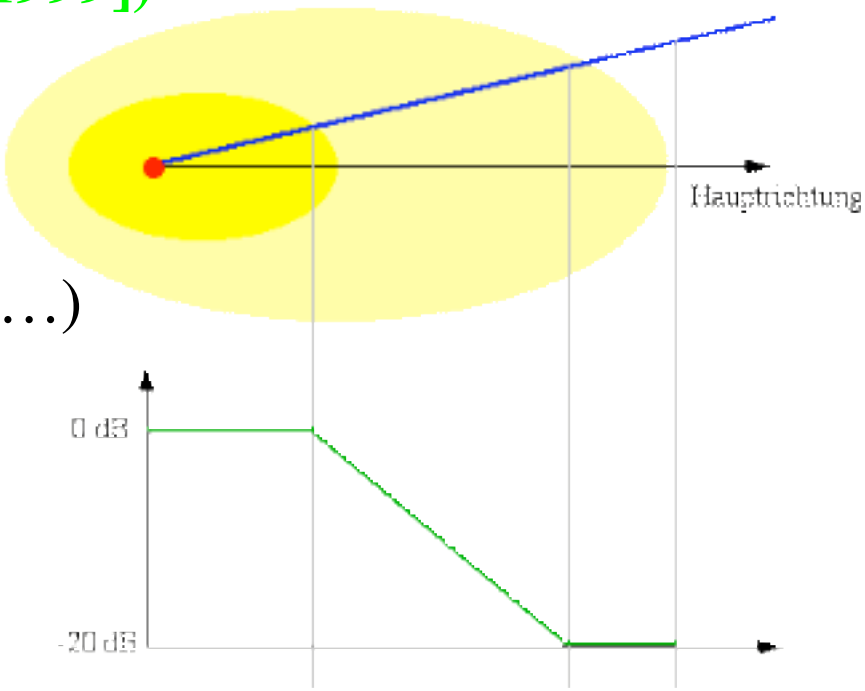
- Untermengen der Kodiermöglichkeiten (Tools)
  - Endgeräte-Klassen und Szenarien
  - Conformance-Tests
- Visual Profiles für Natural Video
  - simple: rechteckig, fehlertolerant (z.B. Mobilfunk)
  - simple scalable: Auflösung (Raum, Zeit) skalierbar
  - Advanced simple: B-Frames+1/4 pel+GMC -> Internet-Video bis TV
  - advanced real-time simple (ARTS): Videotelefonie, Fehlertoleranz
  - core: Objekte beliebiger Form, sprites, Interaktivität
  - core scalable
  - main: interlaced, transparent, Sprites; Broadcast und DVD
  - advanced coding efficiency: core+1/4 pel+GMC -> 'mobile broadcast'
  - n-bit: andere Bit-Tiefen für Überwachungskameras etc.
  - simple studio (nur I-Frames, 180-1800 Mbit/s)
  - core studio (900 Mbit/s)

- Visual Profiles für Synthetic Video
  - simple face animation, simple face and body animation
  - scalable texture (advanced scalable texture)
  - basic animated 2-D texture
  - hybrid: natural und synth. Video
- Audio Profiles
- General Audio
  - Mobile Audio Interworking: AAC, TwinVQ
- Speech und general Audio
  - scalable
  - high quality (AAC LTP, CELP),
  - natural: alle natural-tools
- Synthetic:
  - SAOL, midi, wavetable, TTSI
- Hybrid natural/synthetic audio
  - speech: 1-20 Objekte, CELP, HVXC, TTSI
  - main: alle tools
  - low delay audio: HVXC, CELP, AAC, TTSI

- Graphics Profiles
  - simple 2-D, complete 2D, basic 2D, core 2D, advanced 2D
  - complete: volle BIFS-Funktionalität
  - X3D interactive
  - 3D audio
- Scene Graph Profiles
  - simple facial animation
  - scalable texture
  - simple face and body animation
- MPEG-J Profiles
  - personal: PDAs, gameboys, etc.; nur network, scene, resource API
  - main: personal+decoder+ decoder function+… APIs
- Levels für Profiles: Auflösung

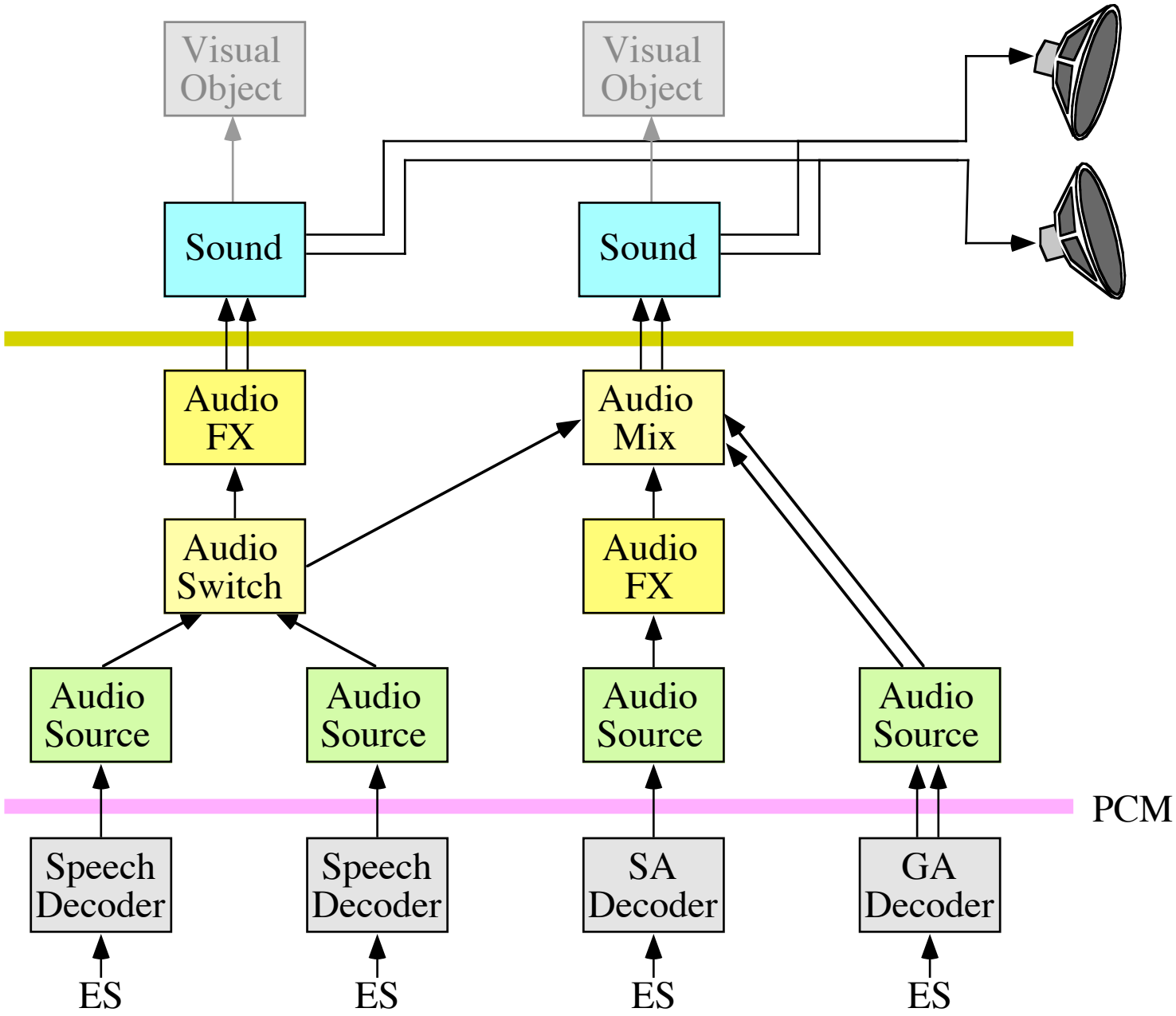
2.4.6. Audio Objects (siehe [Scheirer et al., 1999])

2.4.6.1 Audio-Szenen



- AudioBIFS
  - Subgraph mit Signalverarbeitungsfluss
  - Objekt in der Szene (3D-Position, Ellipse, …)
  - AudioSource: Decoder, URL,
    startTime, stopTime, numChan, …
  - AudioMix und AudioSwitch:
    m Kanäle -> n Kanäle
  - AudioFX (SAOL-Programm)
  - Group, Transform, …
  - ListeningPoint
- MPEG-4V2: Virtuelle Akustik
  - Modellierung der (künstlichen) akustischen Umgebung
  - Modelle für Quelle, Umgebung, Wiedergabeumgebung
  - Dopplereffekt
  - AcousticScene
  - AcousticMaterial: Transferfunktionen Durchlässigkeit und Reflektion
  - DirectiveSound: spektrale Auswirkungen der Position

- Beispiel für Audio-Szenengraph

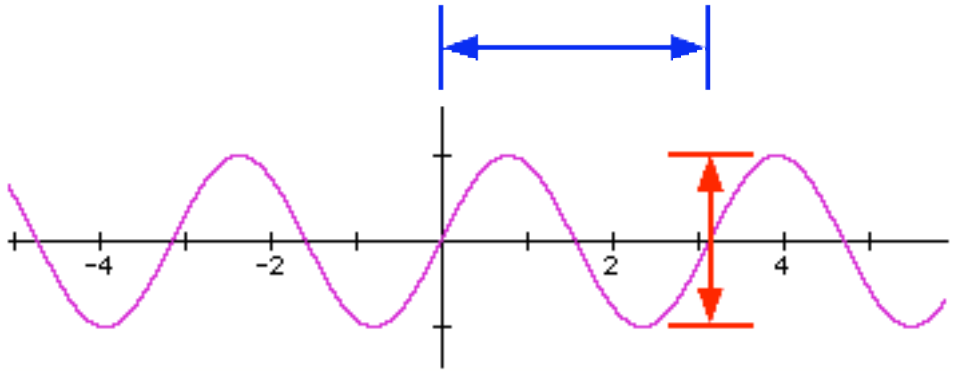


• Beispiel AudioBIFS [Scheirer et al., 99]

```
Group {                                              // in ascii übersetzt
   children [
      sound {
         spatialize FALSE
         source [
            AudioMix {
               numChan 2                             // 4 Eing., 2 Ausg.
               phaseGroup [1,1]
               matrix [0.8 0.4 1.0 0.1 0.4 0.8 0.1 1.0]
               children [
                  AudioSource {
                     url "<string>"
                     numChan 2                       // 2 Eing.
                     phaseGroup [1,1]
                  },
                  AudioFX {
                     orch "<SAOL bytecode>" // mono -> stereo
                     numChan 2                       // 2 Ausg.
                     children [
                        AudioSource {
                           url "<string>"
                           numChan 1                 // 1 Eing.
} ] } ] } ] } ] }
```

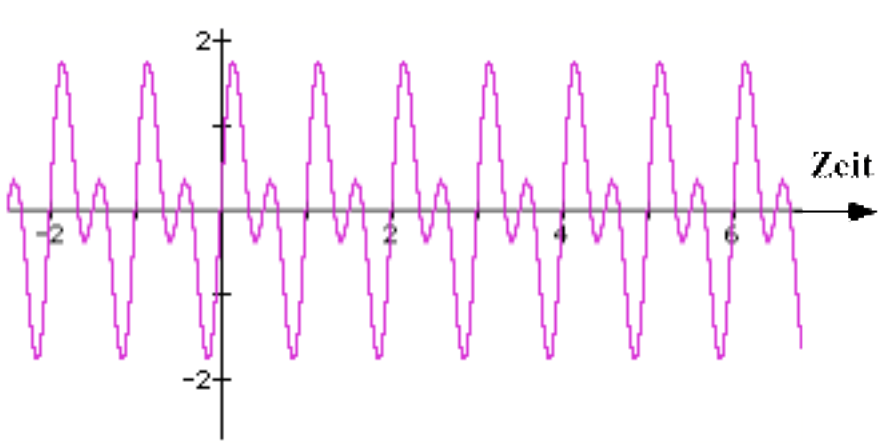
2.4.6.2 Kodier-Werkzeuge

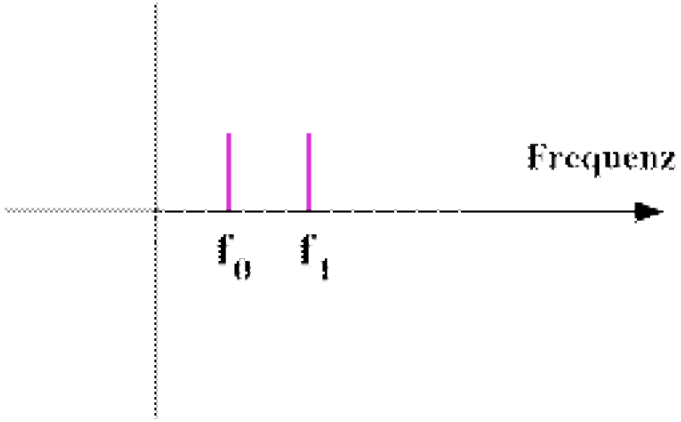
- Wiederholung: Audioeigenschaften
- Frequenz und Amplitude



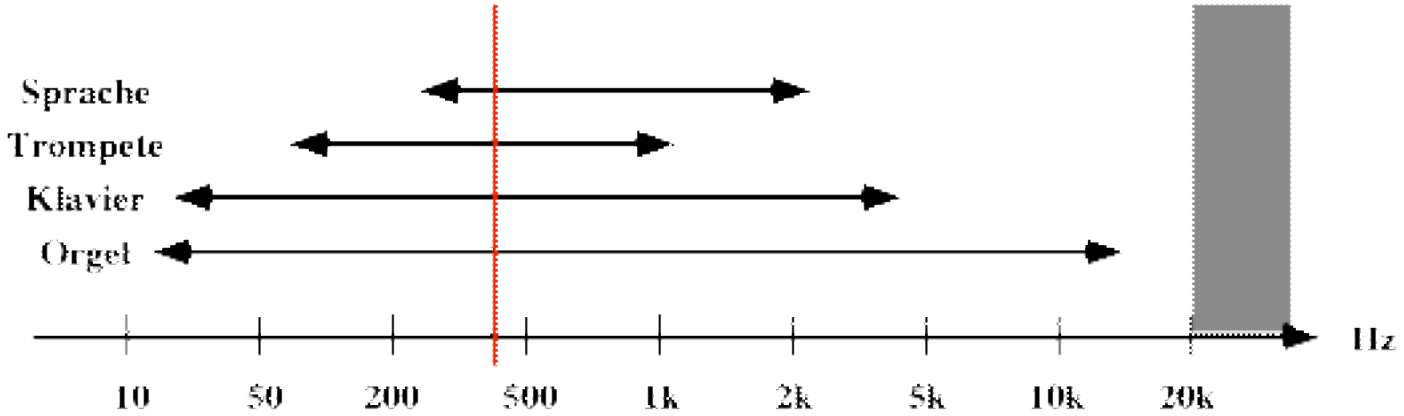
  - Amplitude -> Lautstärke (gemessen in dB)
  - Frequenz (1m/Wellenlänge) -> Tonhöhe

- Fourier: Jede Schwingung kann als Summe von Sinusschwingungen dargestellt werden: $f(x) = \sin 2\pi x + \sin 4\pi x$



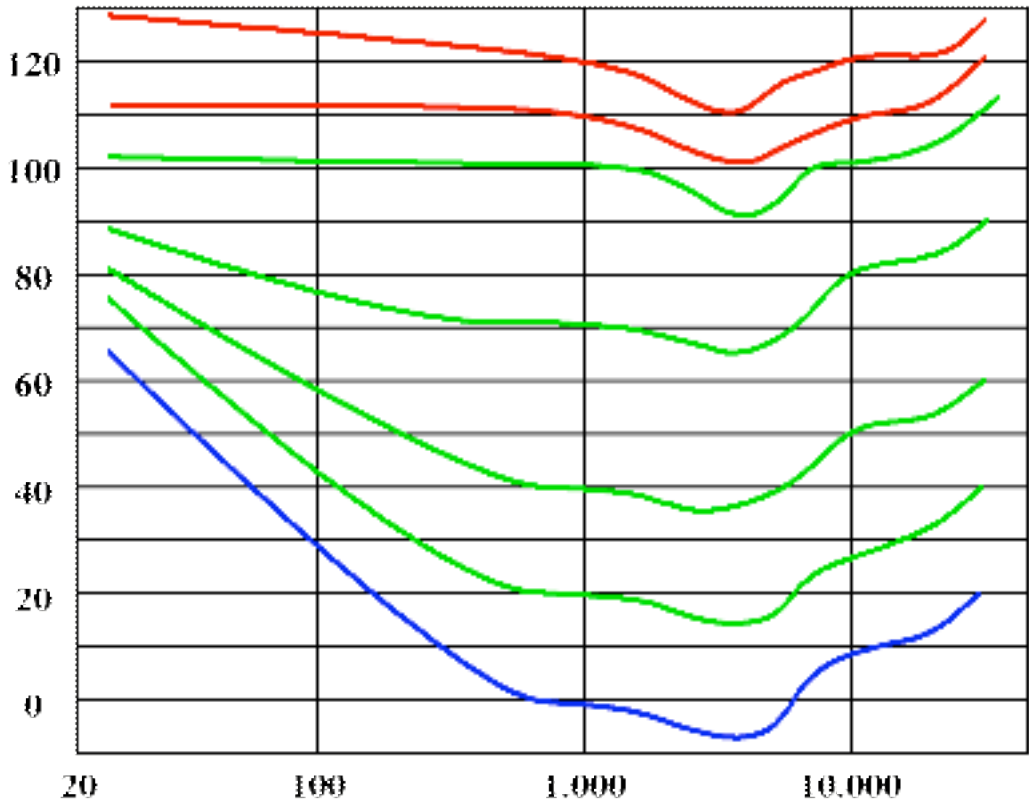


- Typische Frequenzbereiche

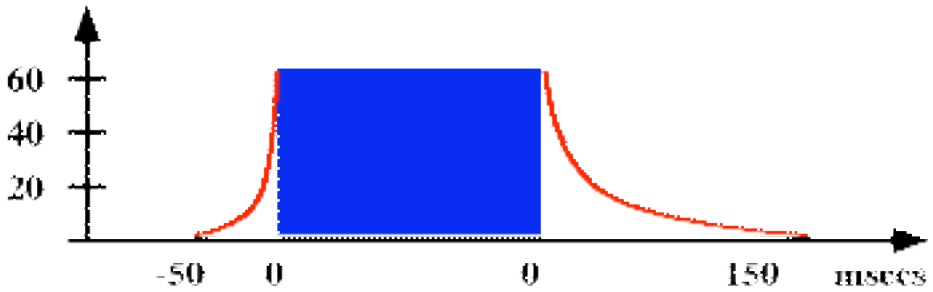


  - Telefon 300Hz - 3.400 Hz
  - Heimstereo 20 Hz - 20.000 Hz
  - UKW (FM) 20 Hz - 15.000 Hz
- Menschliches Hörvermögen
  - 20 - 20.000 Hz
  - hohes zeitliches Auflösungsvermögen
  - logarithmisch bezüglich Amplitude

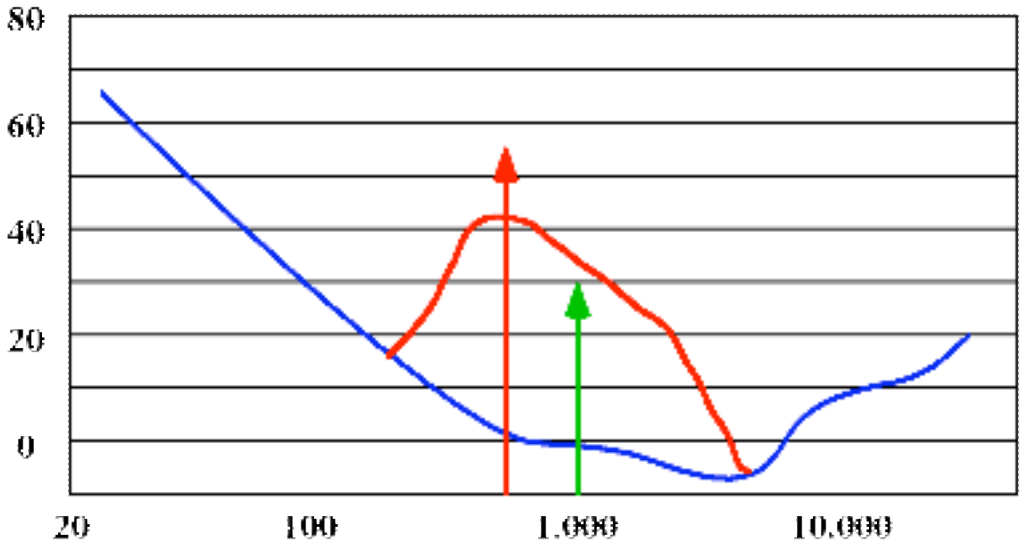
- Lautstärkeempfinden nach Fletcher und Munson



- Abschattung
  - Zeit

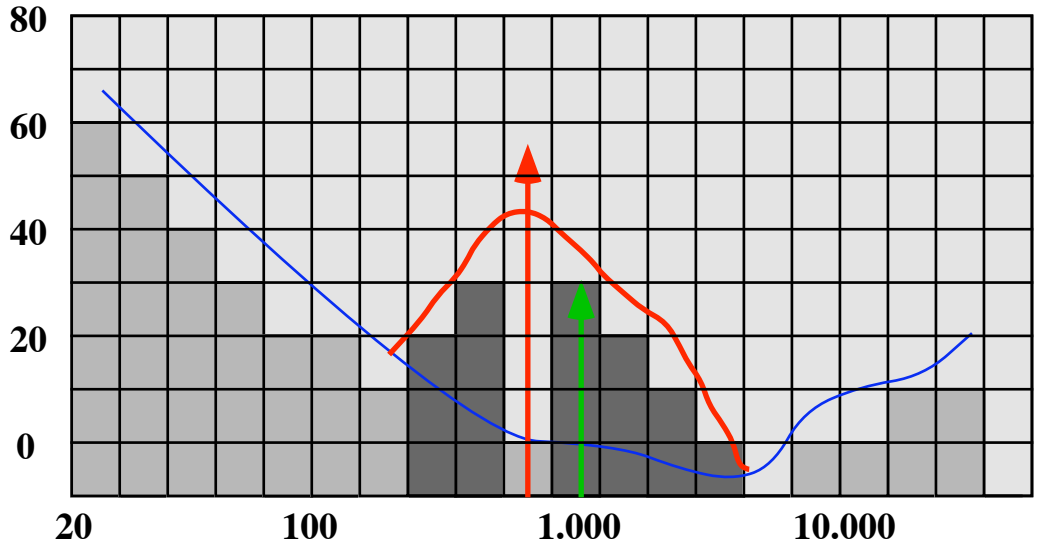


Frequenz

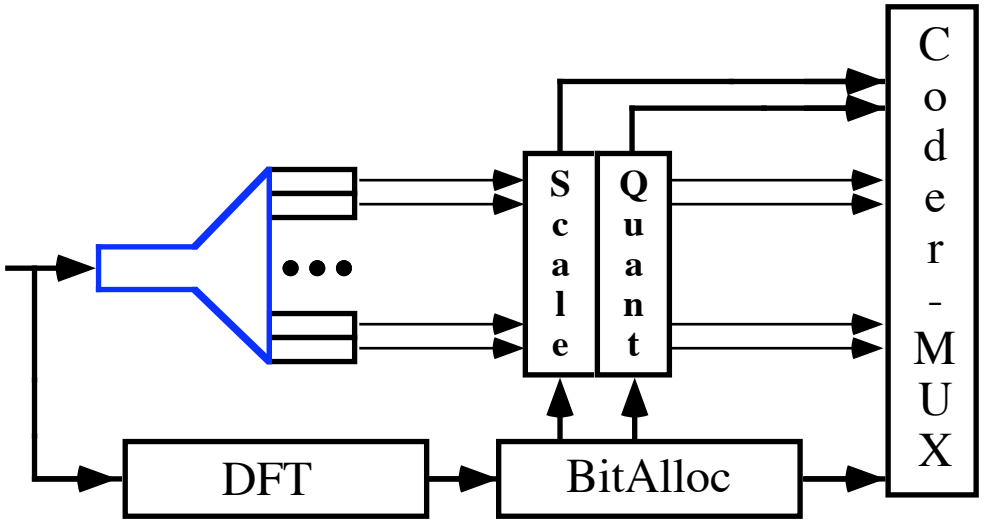


- Perceptual Coding
  - Filterbänke zerlegen in Bänder
  - Maskierung im Band und zwischen Bändern
  - Wahrnehmungsmodell liefert Quantisierung für Bänder
  - Transformation temporale in spektrale Darstellung
  - Entropiekoderung

- MP3
  - MPEG - Moving Pictures Expert Group
  - Video
  - Audio-Layers 1, 2, 3 => mp3
- Frequenzspektrums in Bänder zerlegen
  - Filterbank mit n Filtern
  - Bandbreite F/n
  - Bsp.: DAT 48 kSamples/s, 32 Bänder (MPEG), Bandbreite = 750 Hz
- Nur hörbare Bänder übertragen
  - normale Fletcher-Munson-Kurve
  - Maskierung durch andere Bänder

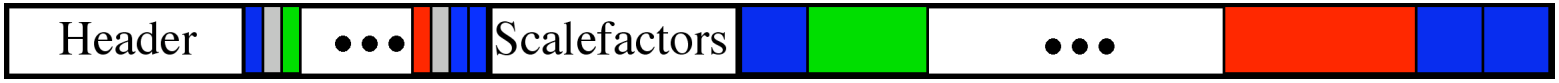


- Beispielkoder
  - Bandaufteilung mit Analyse-Subband-Filter
  - Rahmen (Frame): 384 = 12 * 32 Samples => 12 Samples pro Subband
  - Maskierungwerte (Hörschwelle, andere Bänder) berechnen
  - Frequenzbänder aussondern
  - Skalieren (pro Frame und Band)
  - Quantisieren
  - iterative Bitzuweisung an Bänder, konstante Bitrate (Bitreservoir)



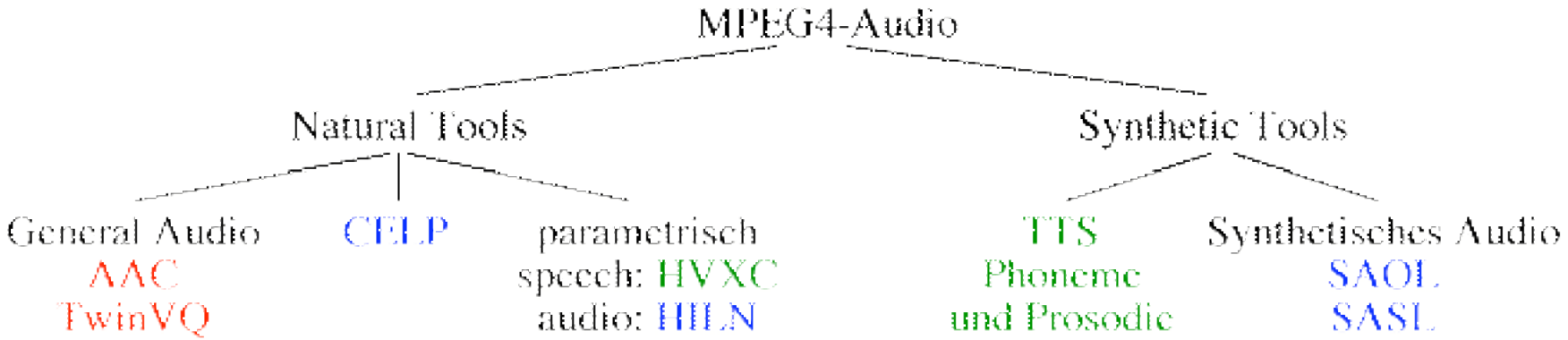
Siehe z.B. N. Fliege: Multiraten-Signalverarbeitung, S. 219 ff.
Aufwärtsabtastung, Filtern, Summieren

- Analyse-Subband-Filter
  - 32 neue Werte in X[0..511] shiften (Modulationsvektor)
  - mit Fenster gewichten
  - mit Modulationsmatrix multiplizieren
  - 1 Wert pro Band, 32 Bänder
- Psychoakustisches Modell
  - Fourier Transformation
  - Schalldruck in jedem Band bestimmen
  - Masken pro Band bestimmen
  - Signal-to-Mask-Ratio SMR berechnen
- Datenstrom



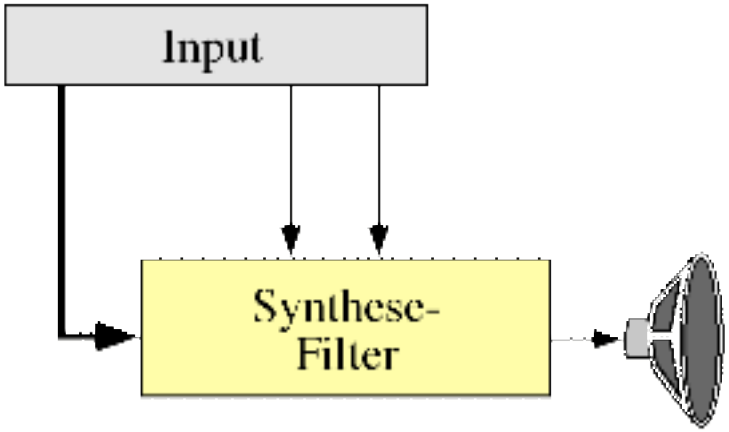
- MPEG Audio Layers
  - I: mit einfacher Filterbank
  - II: mehr Samples, feinere Samplekodierungsmöglichkeiten
  - III: Hybridfilter (Subband, MDCT), adaptive Segmentierung
    MDCT: Modified DCT for Audio Layer III
- Unterstützte Abtastraten: 32, 44.1 und 48 kSamples/s
- Stereo
- MPEG-Audio Datenstrom
  - Layer I: n*32 kbit/s, n = 1, …, 14
  - Layer II: 32, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256, 320, 384 kbit/s
  - Layer III: 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256, 320 kbit/s
- Ab 192 kbit/s kein Unterschied zum Orginal hörbar
- MPEG-2: Advanced Audio Codec (AAC)
  - in MPEG-2 definiert
  - basiert auf mp3: MDCT nach der Banderzeugung
  - Verfeinerung in Details zur besseren Kompression
  - Joint Stereo Coding

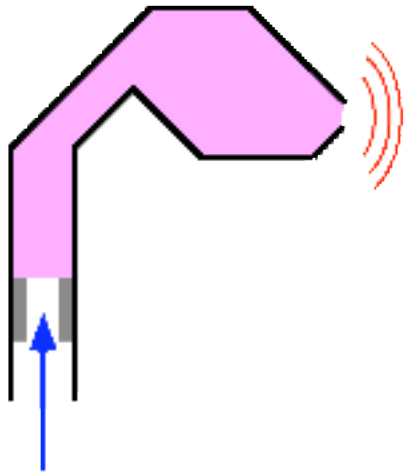
- MPEG-4



- verbessertes AAC
  - bandweise Noise Substitution
  - Long Term Prediction (LTP) und Differenzkodierung
- TwinVQ: Transform-domain Weighted Interleave Vector Quantization
  - alternative Kodierung der spektralen Koeffizienten
  - Normalisierung und Vektorquantisierung (inkl. Codebuchauswahl)
  - guten Ergebnisse bis zu 6 kbit/s
- Low Delay Audio Coding abgeleitet von AAC (Qualität vgl. mp3)
  - MPEG-4 GA bei 24 kbit/s: 110+210 msec Verzögerung
  - Low Delay AC: 20 msec
  - halbe Framelänge: 512/480 Abtastwerte und halbe Fenster
  - kleines oder kein Bit-Reservior

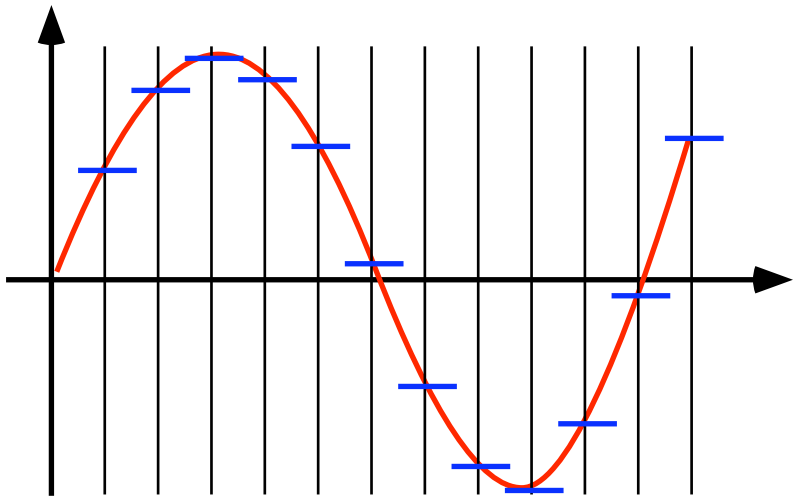
- Einschub: Parametrische Kodierung
  - niedrige Bitraten von 1.2 – 18 kbit/s
  - Decoder: Filter zur Sprachsynthese
  - Filterparameter beschreiben das Modell
  - Bitstrom mit Anregungsdaten für Filter
- Modell für Sprachapparat
  - Luftproduzent (Lunge)
  - Schwingungserzeugung (Stimmbänder)
  - Lautformung in 'Röhre' (Mund, Lippen, Nase)
  - stimmliche Laute durch Schwingung und Formung
  - stimmlose Laute mit Rauschen und Formung
- Sprachkodierung
  - Unterscheidung stimmlos / stimmlich
  - Tonhöhe für stimmliche Laute
  - Formanten bestimmen (-> Parameter)
  - Residuum kodieren (-> Anregungsdaten)



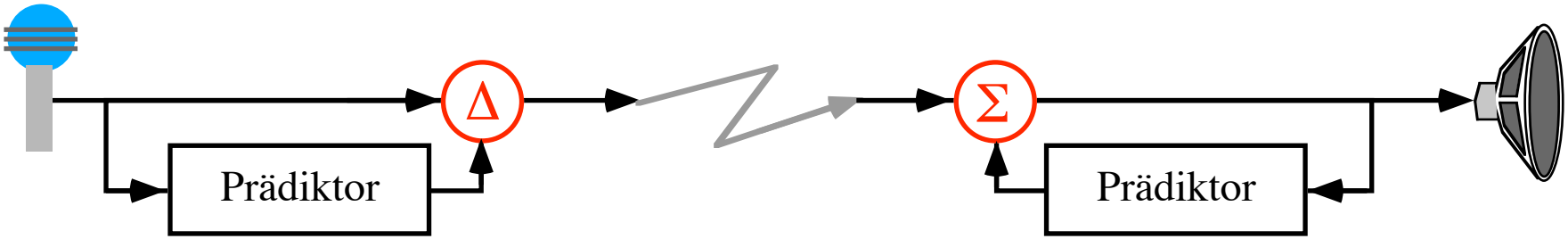


- CELP: Code Excited Linear Prediction
  - 6, 8.3, 12, 18 kbit/s
  - Analyse durch 'Synthese'
  - Erregungssignal ergänzt Generatorsignale
  - wird durch Ausprobieren bestimmt
  - Codebuch für Erregungssignal
  - adaptive Codebuchergänzung
  - Silence Insertion Descriptor und Comfort Noise
  - Wideband-CELP: 16 kHz
- HVXC: Harmonic Vector Excitation
  - 2 und 4 kbit/s; VBR: 1.2-1.7 kbit/s
  - Erregung für stimmliche Laute: Spektrum-Hüllkurve vektorquantisiert
  - Erregung für stimmlose Laute: Codebuch ähnlich CELP

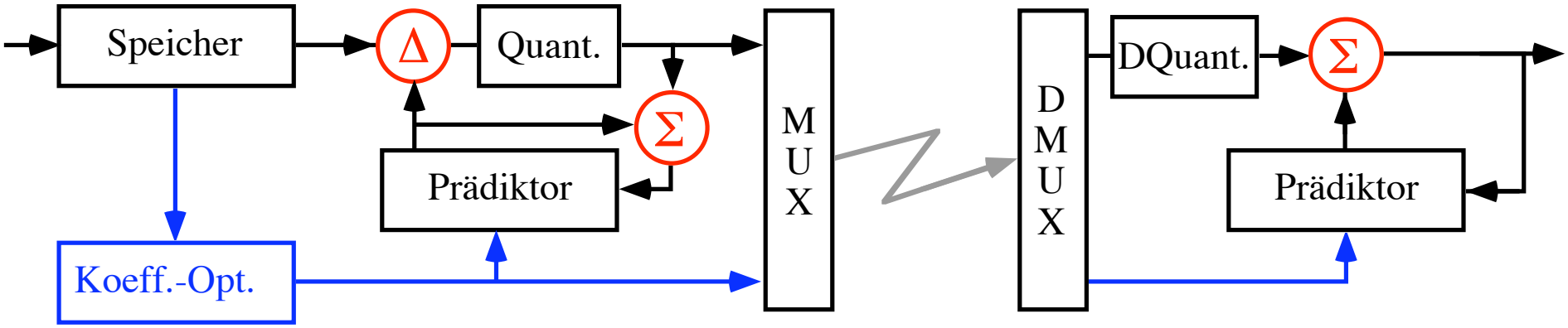
- Stetige, glatte Kurve => nächstes Sample 'in der Nähe'
  - $E(s_i - s_{i-1}) << E(s_i)$
  - Schluß von s_{i-1} auf s_i
  - Übertragung der Differenzen zwischen Samples
  - Variable Bitlänge des Codes
  - Delta-Modulation
  - eventuell nur ±1 pro Sample



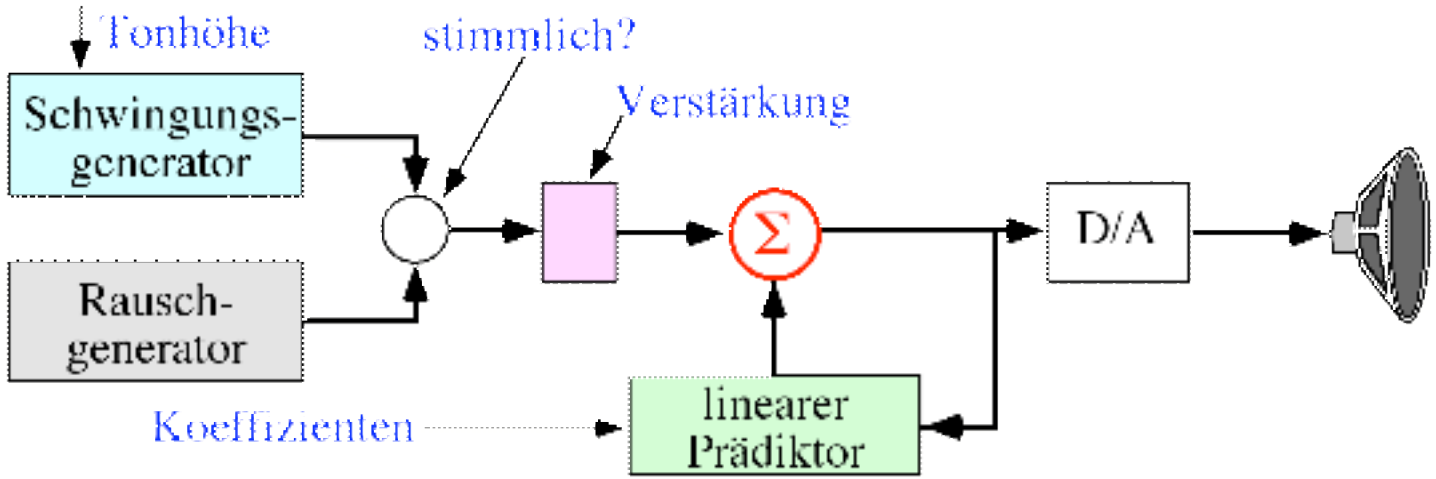
- Nächstes Sample in ähnlicher Richtung
  - $E(f'(s_i) - f'(s_{i-1})) < \varepsilon$
  - Schluß von $(s_{i-n}, \ldots, s_{i-1}, s_i)$ auf s_{i+1}
  - Vorhersage des nächsten Samples
  - Übertragung des Fehlers der Vorhersage
  - Differential PCM (DPCM)



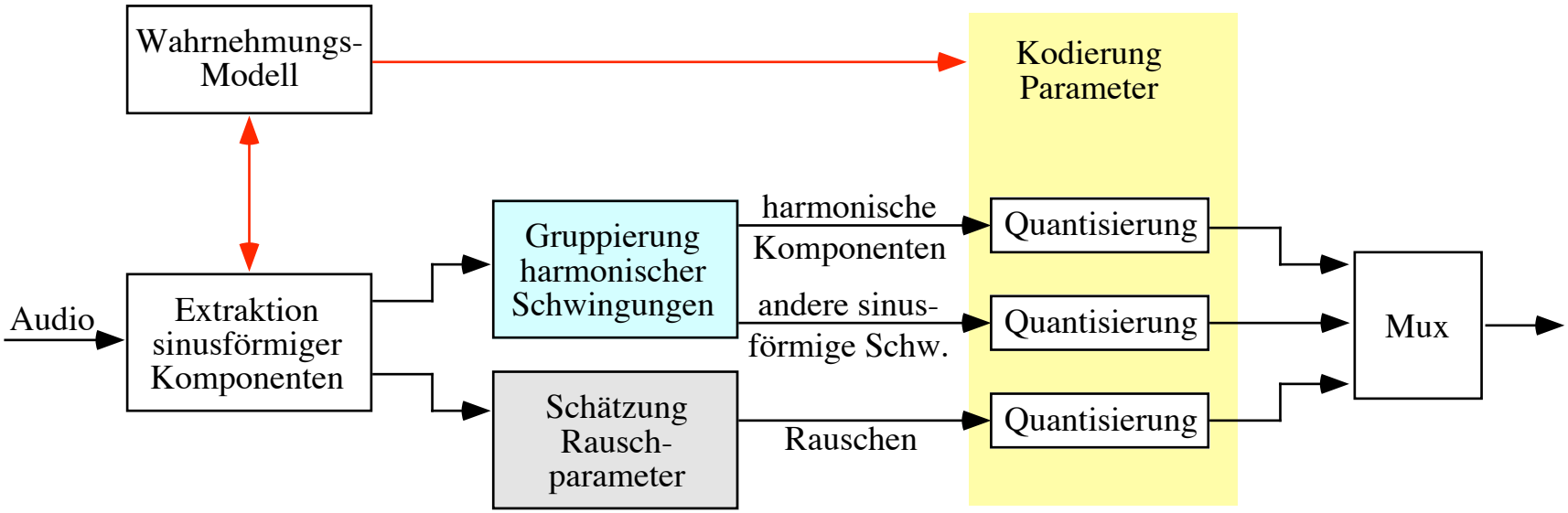
- Adaptive DPCM (ADPCM)
  - Optimierung der Prädiktionskoeffizienten: $d_i = s_i - s_i^*$ minimal
  - Intervallweise Optimierung
  - Übertragung von Koeffizienten und Fehlerkorrektur zum Empfänger



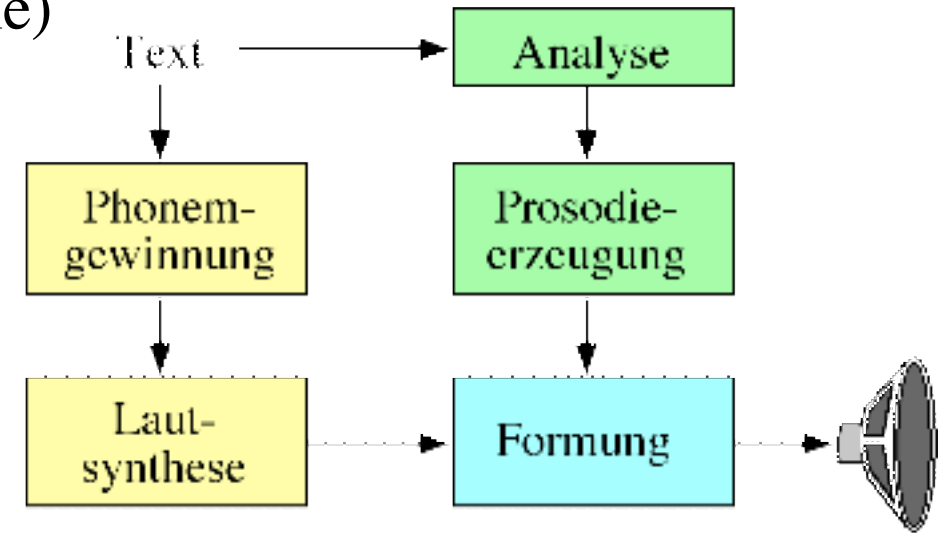
  - Rekonstruierte Werte zu Prädiktorberechnung
- Linear Predictive Coding (LPC)
  - 'ADPCM ohne Prädiktionsfehlerübertragung'



- HILN: Harmonic and Individual Lines plus Noise
  - general audio bei 4 kbit/s
  - Zerlegung in Komponenten
  - Rauschen: Amplitude und spektrale Hüllkurve
  - sinusförmige Anteile: Frequenz und Amplitude
  - harmonische Anteile: Grundfrequenz und spektrale Hüllkurve



- Synthesized Sound
  - Structured Audio Tool
  - SASL: Structured Audio Score Language (-> 3.5)
  - SAOL: Structured Audio Orchestra Language (-> 3.5)
  - TTS: text-to-speech: ASCII + 'Ausspracheinformation'
  - TTS zusammen mit Gesichtsanimation
- Text-to-Speech
  - Text als Buchstaben-Strom
  - Identifikation der Phoneme, phonembasierte Synthese
  - Face-Animation-Parameter aus Phonemen gewinnen
  - Parameter für das Synthesemodell
  - Prosodie-Information (Satzmelodie)



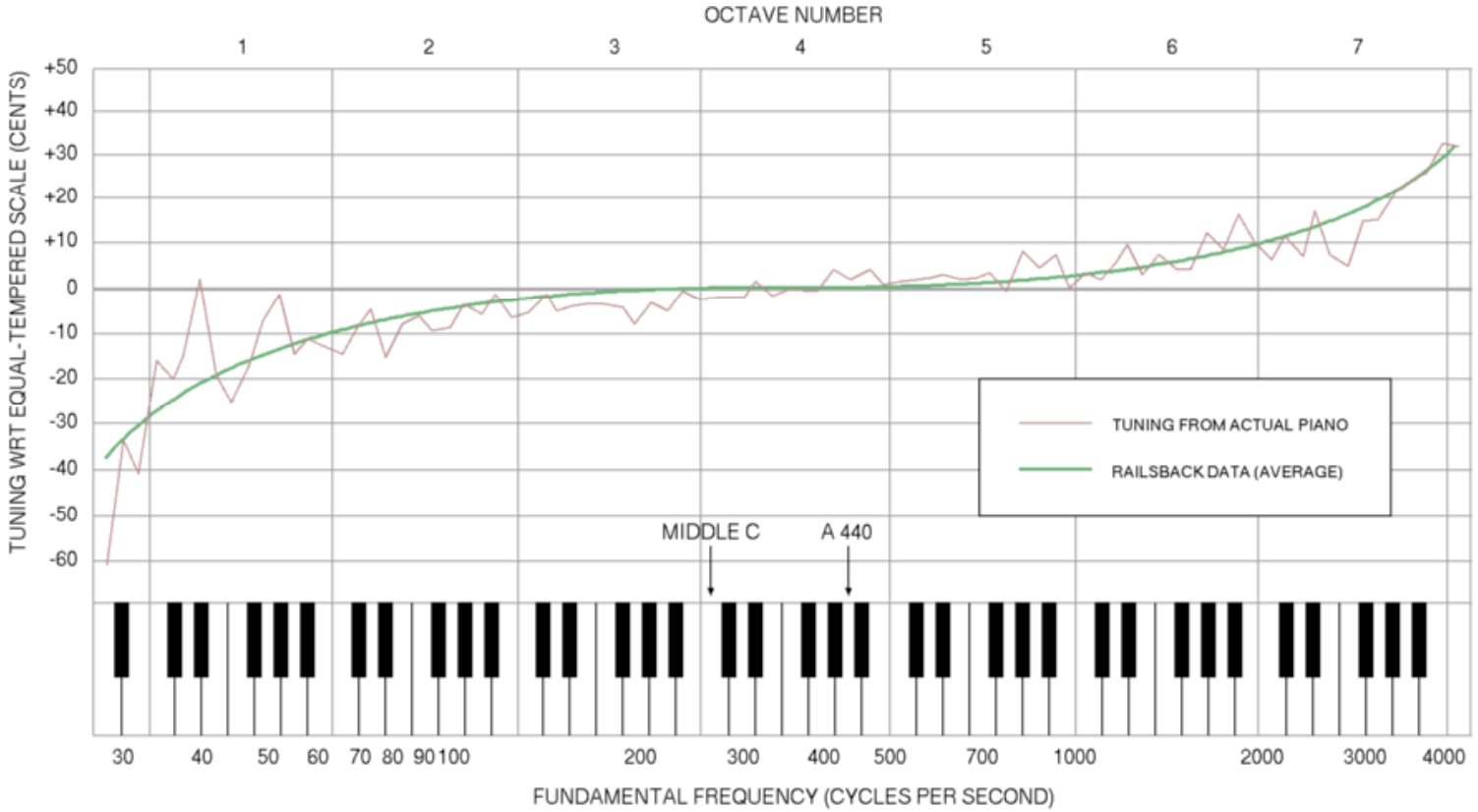
2.5 MPEG-7 und MPEG-21

- Inhaltsbeschreibung
  - Suchen und Finden, Navigation
  - technische Information austauschen
  - Filtern von Content
  - verbesserte Speicherung (-> content management)
- MPEG-7: The Generic Multimedia Content Description Language
  - Descriptor: Syntax und Semantic von 'Features'
  - Description Scheme
  - Description Definition Language: Erweiterung von XML-Schema
  - System Tools
- MPEG-21
  - 'Multimedia-Framework'
  - 'Vision, Technology, Strategy'
  - Digital Item Declaration, Identification, Adaptation
  - IPMP: Intellectual Property Management and Protection
  - Rights Expression Language, Rights Data Dictionary

3. Digital Music

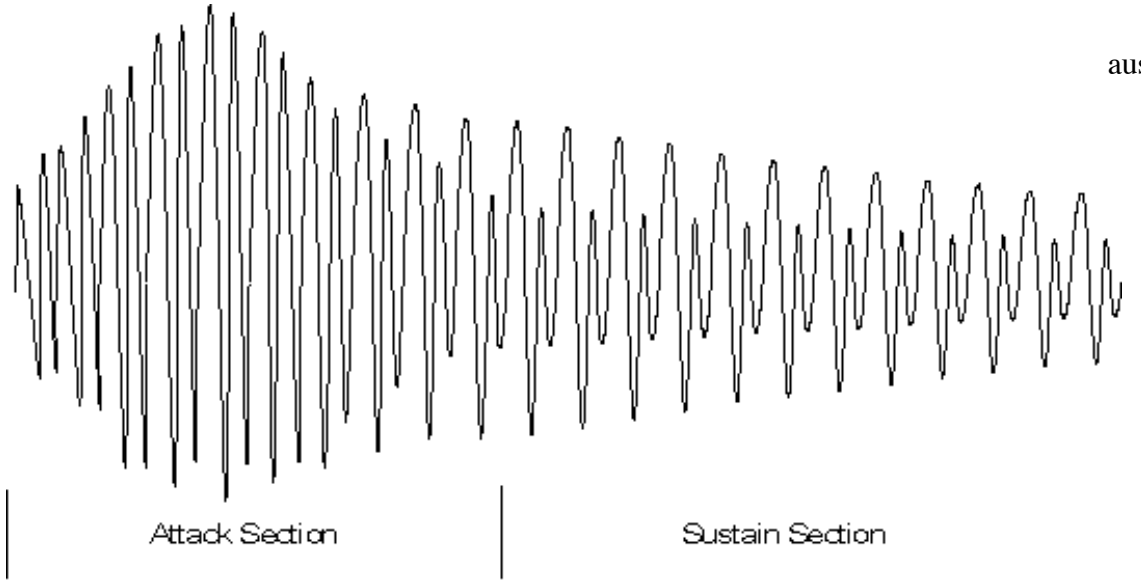
3.1 Instrumentenklang

- Natürliche Instrumente
  - Schwingung anregen (Blasen, Streichen, Zupfen)
  - Resonanzraum
  - Fundamentalfrequenz und Oberwellen (harmonische Frequenzen)



aus:Wikipedia

- Harmonie
  - Zusammenwirken mehrerer Töne
  - Fundamentalfrequenzen und Oberwellen, 'fusing'
- Phasen des Tons
  - Basisfrequenz mit Oberwellen
  - Attackphase: Amplitude und Spektrum schnell verändert
  - Sustainphase: Spektrum wenig verändert, Amplitude nimmt ab

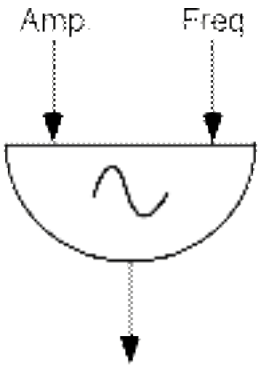


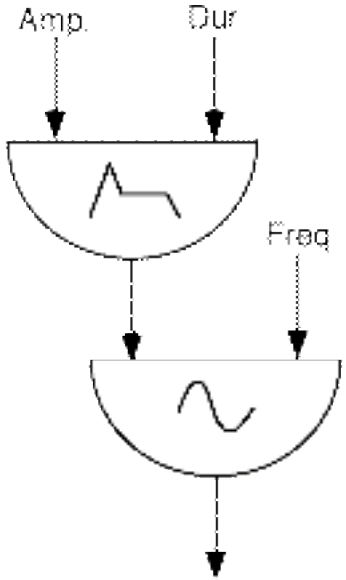
aus: [Heckroth, 1995]

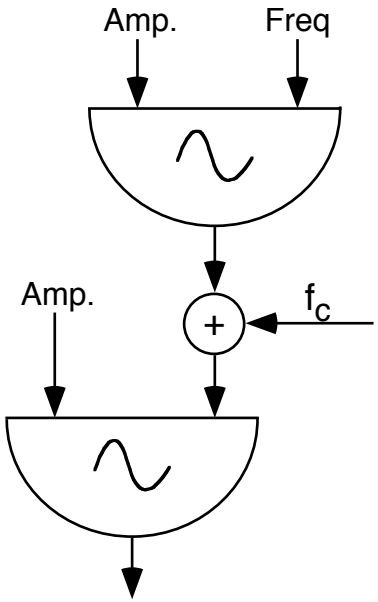
3.2 Geräte der Digitalen Musik

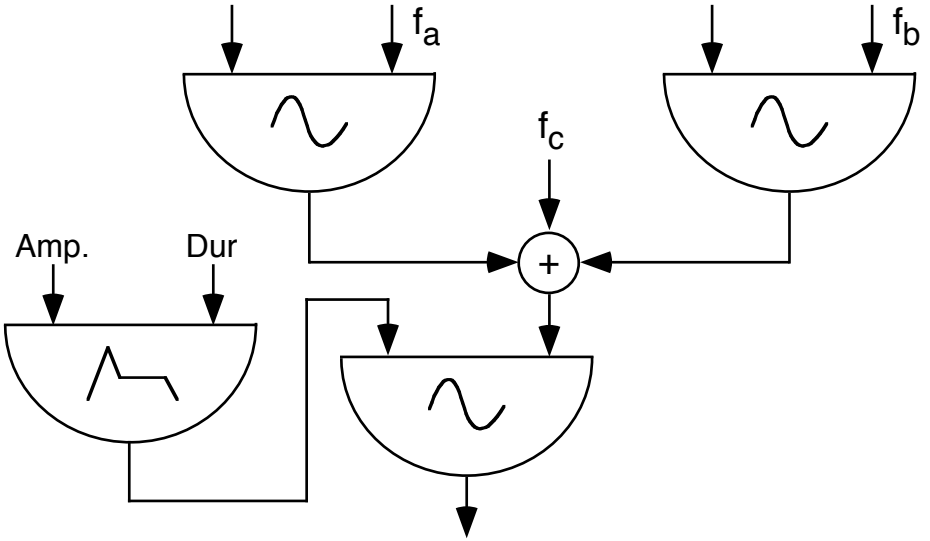
3.2.1 Instrumentensynthese

- Oszillatoren
  - Eingabe: gewünschte Frequenz und Amplitude
  - Hüllkurve für Amplitude
- Frequenzmodulation
  - komplexere Funktionen
  - reiche Spektra, Bessel-Funktionen
  - Basis für Keyboards der ersten Generation



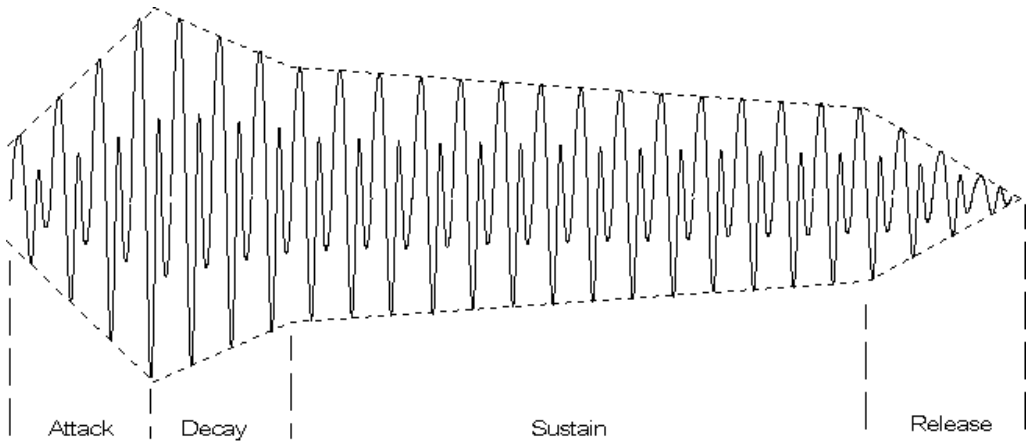






3.2.2 Instrumenten-'Synthese' mit Samples

- Aufnahme (sampling)
  - Attack-Phase (relativ kurz)
  - Sustain-Beispiel -> Loop
  - Sample-Länge Vielfaches der Grundfrequenz
  - evtl. sanft und hart angeschlagen (velocity)
  - Editieren der Samplesequenzen sichert Konkatenierbarkeit
  - 'One-Shot'
- Kompression der Samples ähnlich G.711
- Abspielen der Samples
  - ADSR: Attack - Decay - Sustain - Release
  - Attack - Decay Samplesequenz mit fester Länge abspielen
  - Sustain aus Loops zusammensetzen
  - Hüllpolygon sorgt für Ausklingen (Release)
  - Lookup Table

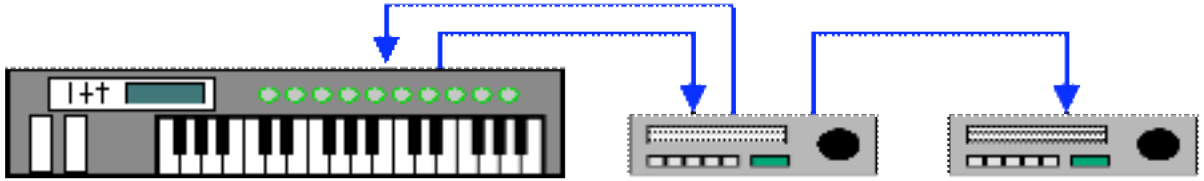


aus: [Heckroth, 1995]

- Tonhöhenanpassung (pitch shift)
  - Samplewiedergabe mit veränderter Geschwindigkeit
  - Beispiel Klavier: C -> C# und D
  - Geschwindigkeit*1,05946 -> Halbton höher
  - Sample-Interpolation nötig
  - evtl. Oversampling der Samplesequenz
  - Vibrato: niedrigfrequente Schwingung modifiziert pitch-shift-Faktor
  - Tremolo: niedrigfrequente Schwingung modifiziert Hüllpolygon
- Splits
  - Tonmenge in Untermengen einteilen: pitch-splits (key-split)
  - pro Untermenge 1 Repräsentant
  - pitch shift produziert andere Töne im split
  - velocity-splits (Anschlagstärken-Repräsentanten)
- Digitale Filter zur Nachbearbeitung
  - Tiefpass entfernt pitch-shift-Rauschen
  - Piano: stark angeschlagene Töne 'heller'
  - Tiefpass verändert Spektrum entsprechend Anschlagstärke
  - variabler Tiefpass über Tondauer
  - besondere Filter an den Split-Grenzen

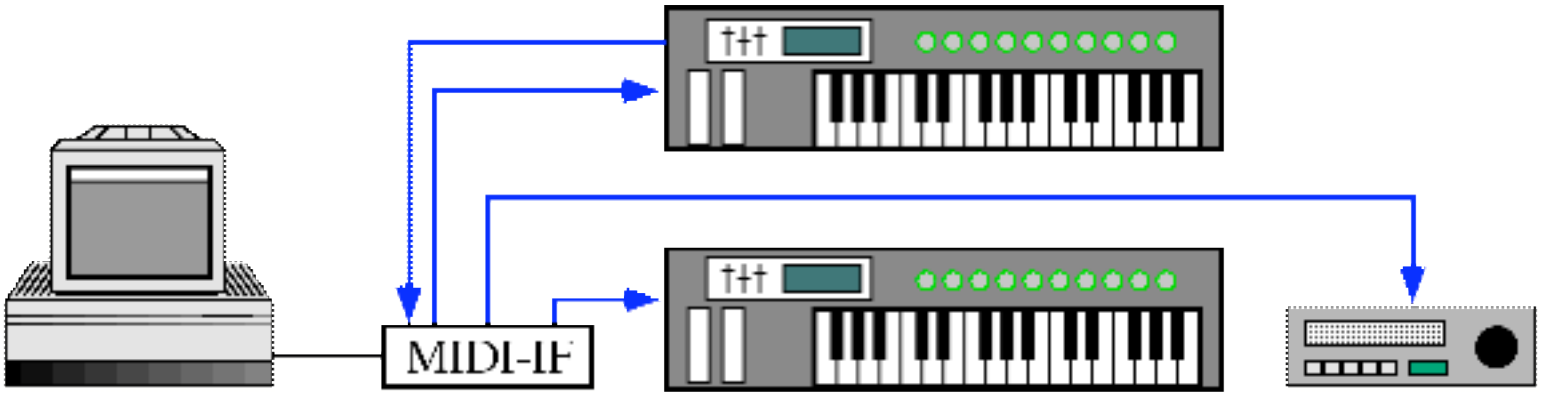
3.2.3 Midi - Musical Instrument Digital Interface

- 1983
- Kommunikation zwischen Steuergeräten und elektronischen Instrumenten
  - Keyboard
  - Sequencer: Aufzeichnung, Editieren und Wiedergabe von Tönen
  - Synthesizer
  - Instrumente mit Interface: Yamaha Grand Piano, …
  - Interface für Instrumente: Gitarren-IF
  - Computer
- Verbindung zwischen Musikinstrumenten



- Format zur Musik-Spezifikation ≠ Samples

- Verbindung der Musikinstrumente mit Computer



- Schnittstelle
  - 31.250 bit/s
  - 5 mA current-loop, 15 m, twisted pair
  - Startbit, 8 Datenbits, Stopbit
  - gerichtet (simplex)
  - 2 Schnittstellen für Vollduplexbetrieb (In und Out)
  - Durchgang für weitere Geräte
- Controller
  - wird 'gespielt'
  - erzeugt Midi-Datenstrom
- Sequencer: aufzeichenen, speichern, editieren, abspielen
- Synthesiser: erzeugt Audio

- 16 logische Kanäle pro physischem Kanal
  - Kommandos mit Tonhöhe, Modulation, Pedale, Balance, Lautstärke
  - Channelcontroller zur Kanalmodussteuerung (122-127)
  - im Kanal 1
  - polyphone Synthesizer spielen mehrere Töne gleichzeitig
- Noten und Anschlag
  - 128 Töne, 21 - 108 entsprechen Klavier
  - 60 entspricht C
  - Anschlagstärke (velocity)
- Nachrichten
  - Channel-messages

    Voice Messages steuern die Stimmen: Ton ein / aus, Anschlag
    Mode Messages vom und zum Controller
  - System Messages

    Common, z.B. Song Select
    Real-Time: Timing, Start, Stop
    Exclusive, Herstellerspezifische Daten

- Protokoll

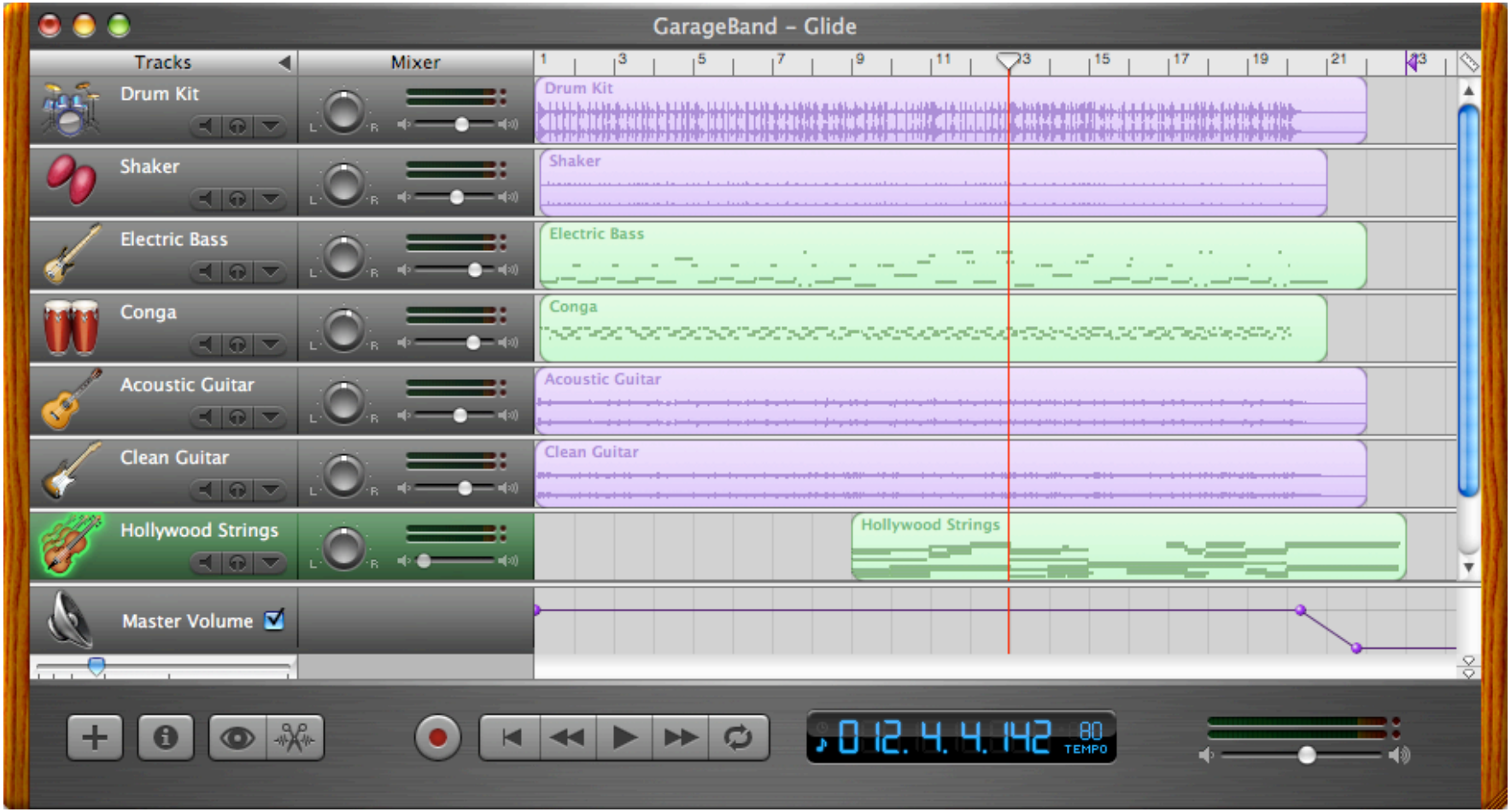
  - Status: Nachricht + Kanal bzw. Systemnachrichten
  - 2 Datenbytes

- Beispiel

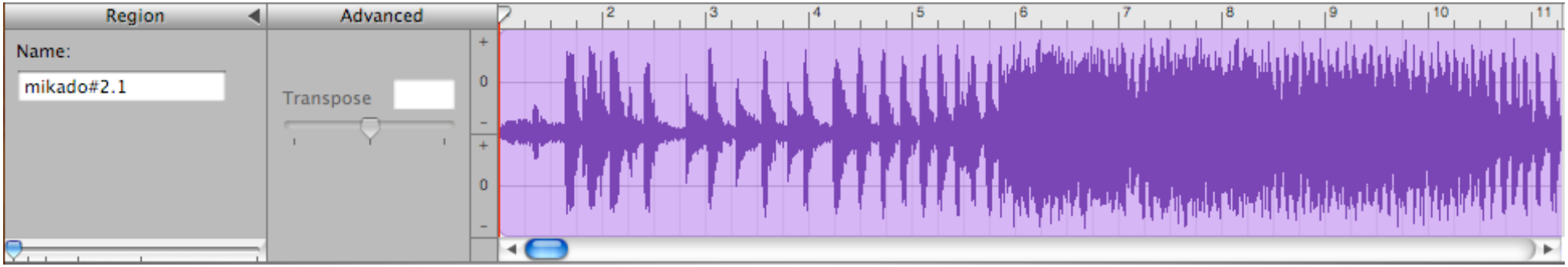
(NoteOn, #, Vel),
(NoteOn, #, Clock, Vel),
(NoteOn, #, Vel),
(NoteOff, #, Vel),
(Cntl, Tremolo, 42), …

3.3 Audio Compositing (GarageBand, CuBase, …)

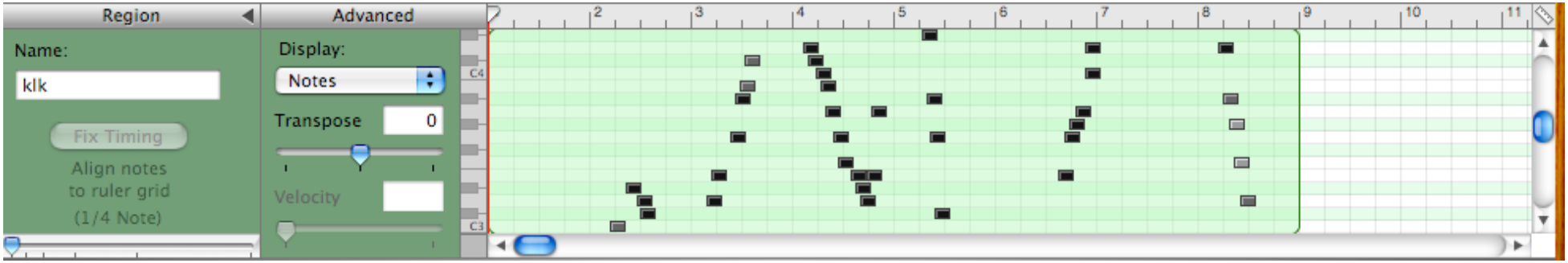
- Spurkonzept
  - Audiokomponenten: Instrumente, Stimme, Geräusche
  - Einspielen: AD-Konverter oder digital (MIDI)
  - Software-Instrumente mit 'Keyboard' (z.B. USB)
  - Loops



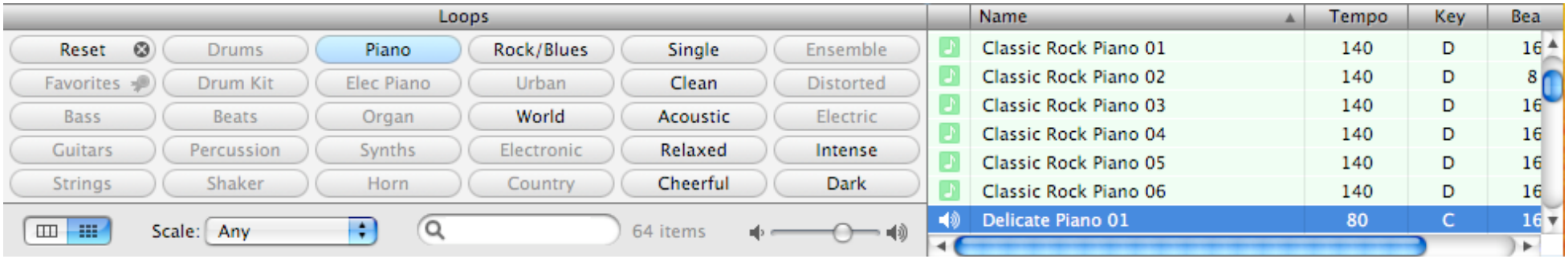
- Track-Editor
  - nachbearbeiten der Spur
  - digitalisiertes Audio: Sample-Editor



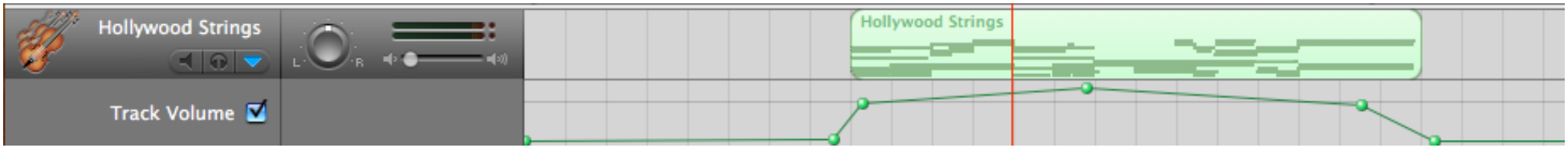
  - 'Software Instruments': Key-editor



- Loops
  - mitgeliefert Audiokomponenten
  - Charakterisierung oder Anreicherung der eigenen Musik
  - rythmische Wiederholung möglich



- Mischen
  - Spuren in Zeit und Raum anordnen
  - Spurlautstärkekurve



3.5 Musik-Programmiersprachen

- Signalverarbeitungskomponenten
- Oszillatoren, Filter, Hüllkurven

```
output = oscillator(freq,time)*envelope(time)
```

- Trennung Signalverarbeitung und Partitur
- SAOL: Structured Audio Orchestra Language
  - C-ähnliche Programmiersprache
  - Instrument: Variablen und Signalverarbeitungskette
  - Instrumente werden von SASL benutzt
  - http://www.cs.berkeley.edu/~lazzaro/sa/book/index.html

```
global {srate 32000; krate 1000;}    // sample- und control-rate
instr beep (pitch,amp) {
      asig sound;                    // Audiosignal
      ksig env;                      // Kontrollsignal
      table wave(harm,2048,1);       // wavetable generation
                                     // dur Instrumentenvariable
      env = kline(0,0.1,1,dur-0.1,0); // huelle erzeugen
      sound = oscil(wave,pitch)*env;
      output(sound*amp);             // play sound
} // aus Scheirer, Vercoe: SAOL; 1999
```

- 3 Zeitachsen
  - sample rate (a-rate)
  - control rate (k-rate): Noten- und Klangfarben-Änderungen
  - instrument-rate (i-rate): Initialisierung
- Signalvariablen
  - asig, ksig, ivar
  - steuern Ausführungsfrequenz ihrer Statements
  - Signal-Arrays
- SAOL-Statements
  - Ausführungsfrequenz (implizite Schleife)
  - hängt von enthaltenen Signalvariablen ab
  - arithmetisch, Prozeduraufruf, Verzweigung, …
- Op-Codes
  - eigentlich zeitabhängige Funktionen
  - ausgeführt mit Rate: iopcode, kopcode, aopcode
  - Ausführung mit höherer rate liefert Anfangsergebnis
  - rate-polymorphic Opcodes (sin(), pow(), …)
  - können interne Zustände haben

- Wavetables
  - Audio-Samples aus Puffern (Tabellen)
  - Datentyp table
  - Generatoren können wavetables füllen
  - table mysine(harm,256,1,4)
  - abspielen mit {lldlk}oscil(Tabelle,freq)
  - interner Zeiger auf nächstes sample, evtl. Interpolation
- Op-Codes Überblick
  - math: int, frac, sgn, exp, log, sqrt, sin, cos, tan, …
  - pitch converters: cpsmidi (midi->Hz), octcps, …
  - Wavetable spielen: oscil, loscil, doscil, koscil
  - Wellen erzeugen: harm, harm_phase, periodic, buzz, …
  - weitere Wellen: step, lineseg, spline, …
  - Hüllen erzeugen, plucked strings, …
  - filter: hipass, lopass, bandpass, fir, iir, …
  - noise generators, spectral analysis
  - sample rate conversion, delays, tempo control, gain control
  - Effekte: reverb, chorus, flange, speedt
  - sample liest AIFF oder WAV-Datei in Tabelle ein

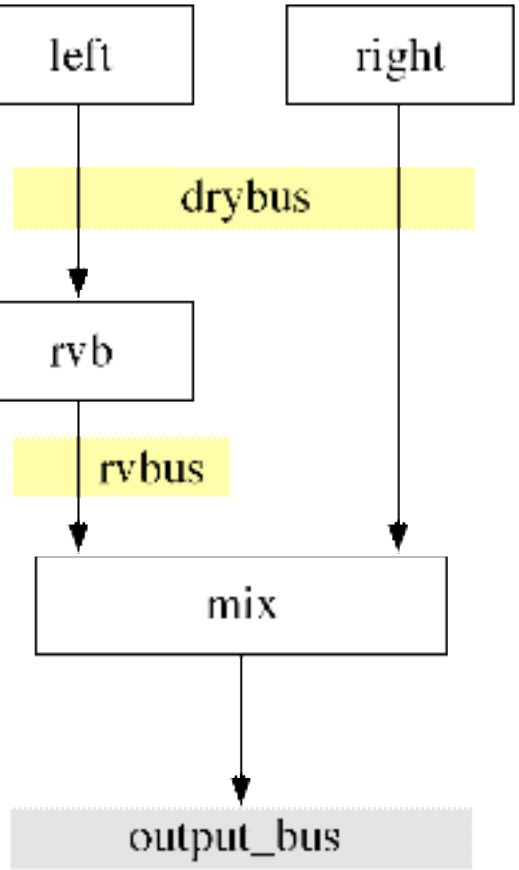
- Ausgabe: output_bus
  - Breite: Kanalanzahl
  - output(sigexp [,sigexp2, sigexp3, …])
  - in Datei oder auf in DAC geschreiben

```
global { outchannels 2;  // stereo output }

instr mono () {
asig a;
output(a);    } // a wird in beide Kanäle geschreiben

instr stereo () {
asig b[2];      // b[0] wird zu Kanal 1 addiert, b[1] zu Kanal 2
output(b);    } // evtl clipping
```

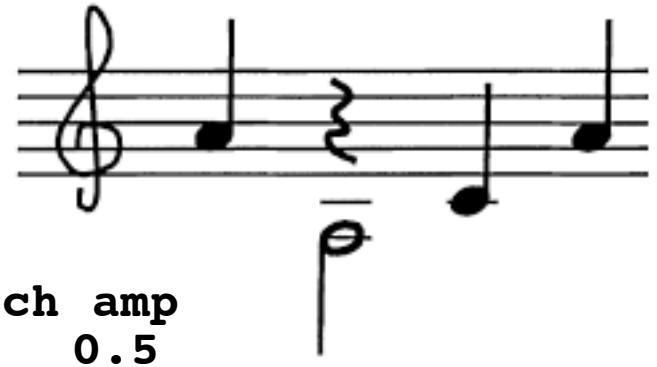
- Eingabe: input_bus
  - input[0], input[1], …

- Routing
  - output wird an selbstprogrammierten Bus gesendet
  - z.B. Mixer mit Anordnung im Stereo-Feld

```
global { outchannels 2;        // Echo im linken Kanal
  route(drybus, left, right);
  send(rvb; ; drybus);          // man beachte die semis
  route(rvbus, rvb);
  send(mix; 0.2, 1 ; rvbus, drybus); }

instr left() { output(arand(0.2));}
instr right(){ output(arand(0.2));}

instr rvb() {output(reverb(input[0]+input[1],5));}

instr mix(revAmp, dryAmp) {
asig out[2];
  out = revAmp*input[0];
  out[0] = out[0]+ dryAmp*input[1];
  out[1] = out[1]+ dryAmp*input[2];
  output(out);
}
```



- SASL: Structured Audio Score Language
  - Partitur
  - Folge von Instrumenten-Aufrufen



```
// Zeitstempel  Instrument  Dauer  Parameter:  pitch  amp
      0.0         beep       1.0                440    0.5
      1.0         beep       2.0                220    0.2
      2.0         beep       1.0                264    0.5
      3.0         beep       1.0                440    0.5
      4.0         end
```

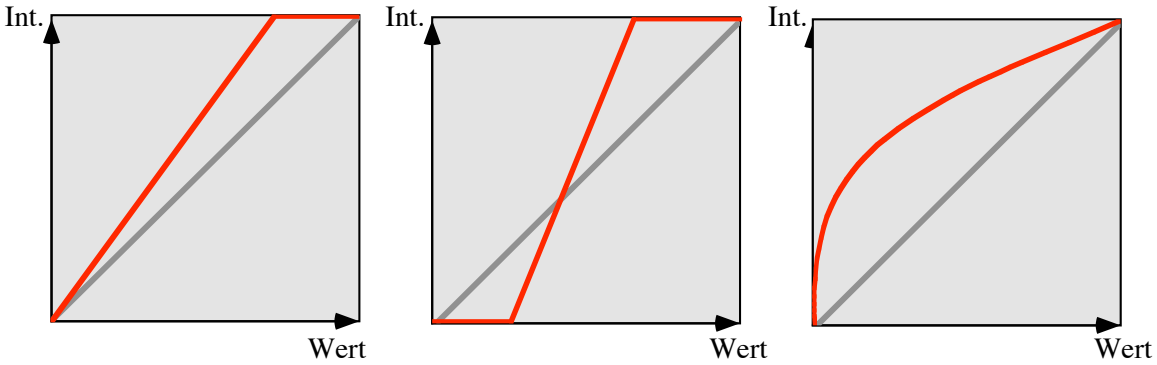
- SASL-Kommandos
  - instr: `[label:] trigger name dur parameters`
  - trigger Startzeit in 'beats'
  - tempo ändert beats/min
  - end
  - control ändert SAOL-Variable
  - table: Wavetable erzeugen und an SAOL schicken/ändern

4. Digital Compositing

4.0 Audio siehe oben

4.1 Bilder

- Einfache Bildbearbeitung
  - Helligkeit: R = P*y
  - Kontrast: R = (P-x)*y
  - Gamma: $R = P^{1/Gamma}$
- Transformationen
  - Verschieben (pan)
  - Rotieren
  - Skalieren
  - Verzerrren (Warp, Gitterpunkte transformieren)



- Filter
- Konvolution (Faltung)
  - Kern
  - typisch 3x3 oder 5x5

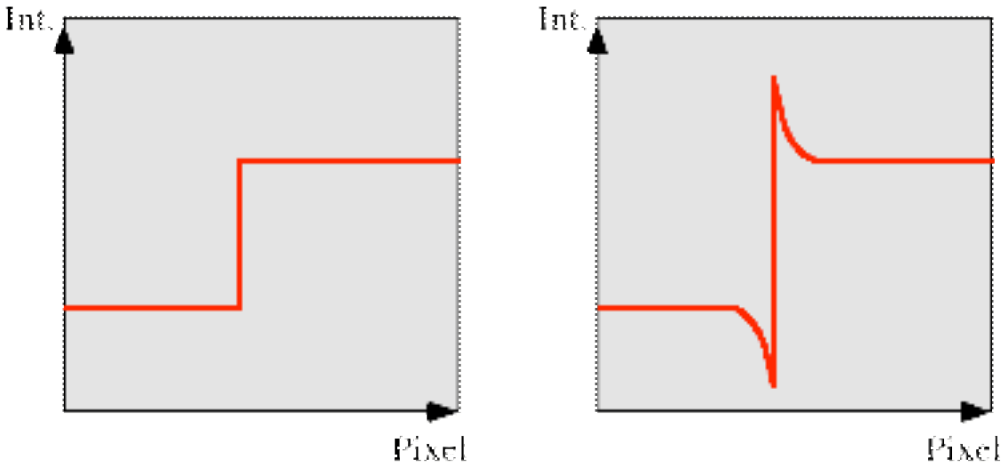
0.6	0.6	0.9	0.9	0.9
0.6	0.6	0.9	0.9	0.9
0.6	0.6	0.9	0.9	0.9
0.6	0.6	0.6	0.9	0.9
0.6	0.6	0.6	0.6	0.9

-1	-1	-1
-1	8	-1
-1	-1	-1

0	0.9	0
0	1	0.3
0	0	0.9

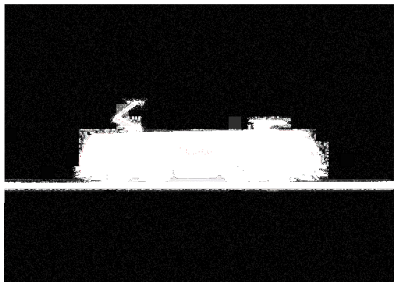
- Verwischen (blur)
  - Durchschnittsbildung mit Umgebungspunkten
- Schärfen

1	1	1
1	1	1
1	1	1



- Median-Filter: Pixel-Störungen beseitigen

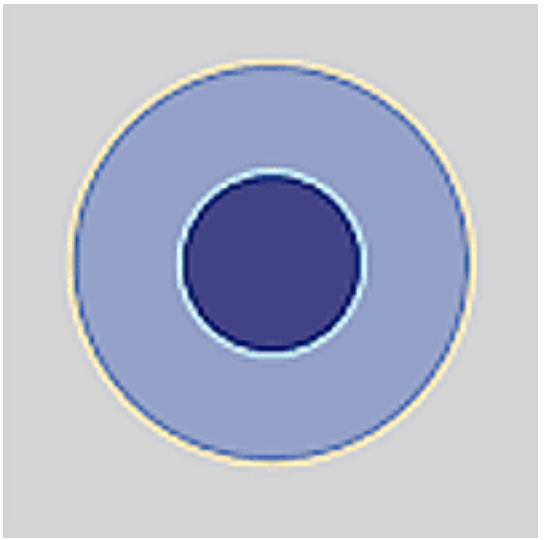
- Komposition
  - mehrere Quellbilder
  - Operatoren verknüpfen Quellen
- Maske (matte)
  - steuert Komposition
  - Graustufenbild als Zusatz zum Orginal
  - auch als weiterer Kanal im Farbbild: alpha-Kanal: P_{rgba}
- Mehrquellen-Operatoren
  - Over: $R = A*M + B*(1-M) = A_{rgb}*A_a + B_{rgb}*(1-A_a)$
  - Mix mit MischVerhältnis: $R = A*mv + B*(1-mv)$
  - Überblendung: mv(t)
  - Subtraktion: R = A-B (Clipping oder Betrag)
  - In: $R = A*B_a$
  - Out: $R = A*(1- B_a)$
  - Atop: R = (A over B) over B
- Maske erzeugen (keying)
  - Luminanz-keying
  - Chrominanz-keying
  - Differenz



4.1.1 Photoshop und ähnliche

- Bildverbesserung
  - Retouching
  - Kontrast, Helligkeit, Farbsättigung, Histogramm-Equalization
  - Kanten bearbeiten (Schärfe, …)
- Selektion
  - Rechteck, Ellipse und Lasso, 'magnetisches' Lasso
  - Zauberstab
  - farbgesteuert
- Filter
  - Störungen entfernen (z.B. Durchschnitt mit Nachbarn)
  - Effekte
- Zeichenwerkzeuge
  - Stift, Pinsel, Farbeimer, Radiergummi
- Schichten zur Anordnung
  - Transparenz, arithmetische Verknüpfung

- Farbkalibrierung
- Digitalisierte Bilder sind verfälscht
  - CCD-Rauschen, unterschiedlich stark in den Farbkanälen
  - Helligkeit und Kontrast
  - Farben schlecht (Farbtemperatur des Lichtes ...)
- Rauschen entfernen
  - Lab Farbraum
  - blauer Kanal besonders verrauscht
- Nachschärfen
- Kurven (Helligkeit, Kontrast, Farbkanäle)
  - Histogramme
  - Farbkorrektur
- Effekte
  - als Filter implementiert
  - Verwischen (blur, PSF), Kanten schärfen, …
- Kai's Powertools
  - lokale und globale Transformationen

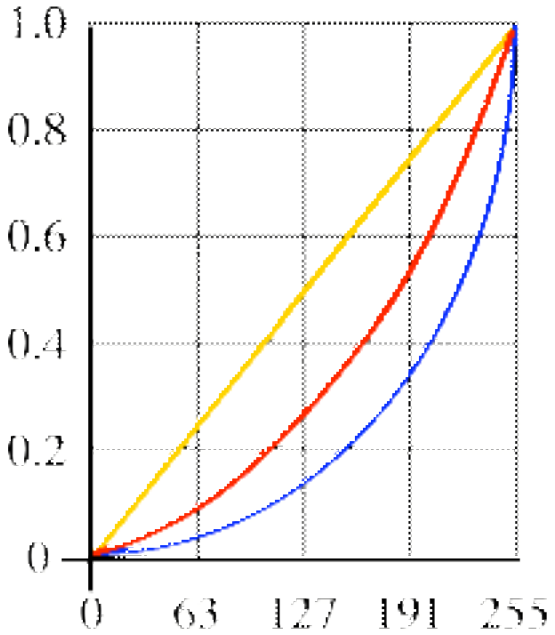


0.01	0.1	0.25	0.1	0.01
0.1	0.35	0.5	0.35	0.1
0.25	0.5	1.0	0.5	0.25
0.1	0.35	0.5	0.35	0.1
0.01	0.1	0.25	0.1	0.01

- Räumliche Anordnung der Bildelemente
  - Layer
  - Organisation des Bildes
  - wichtige Elemente in Layern
  - Transparenz
  - arithmetische Verknüpfung
- Druck-Ausgabe
  - Farbanpassung an Drucker
  - Farbseparationen

4.1.2 Color Management

- Farbdarstellung in der Verarbeitungskette
  - Orginal - Scanner - Bildschirm – Proof – Drucker
  - Lab, CMYK, RGB
  - Farbtemperatur, -Intensität gerätespezifisch
  - Wiedergabekurve Darstellungsgeräte (Phosphor, …)
  - Gamma-Kurve
- Beispiel Bildröhre
  - Messung der Luminanz pro Kanal für Werte 0..255
  - Abbildung Farbwert -> Ansteuersignal
    => Gammakorrekturtabelle im Displayadapter
    => oder in der Strahlsteuerung



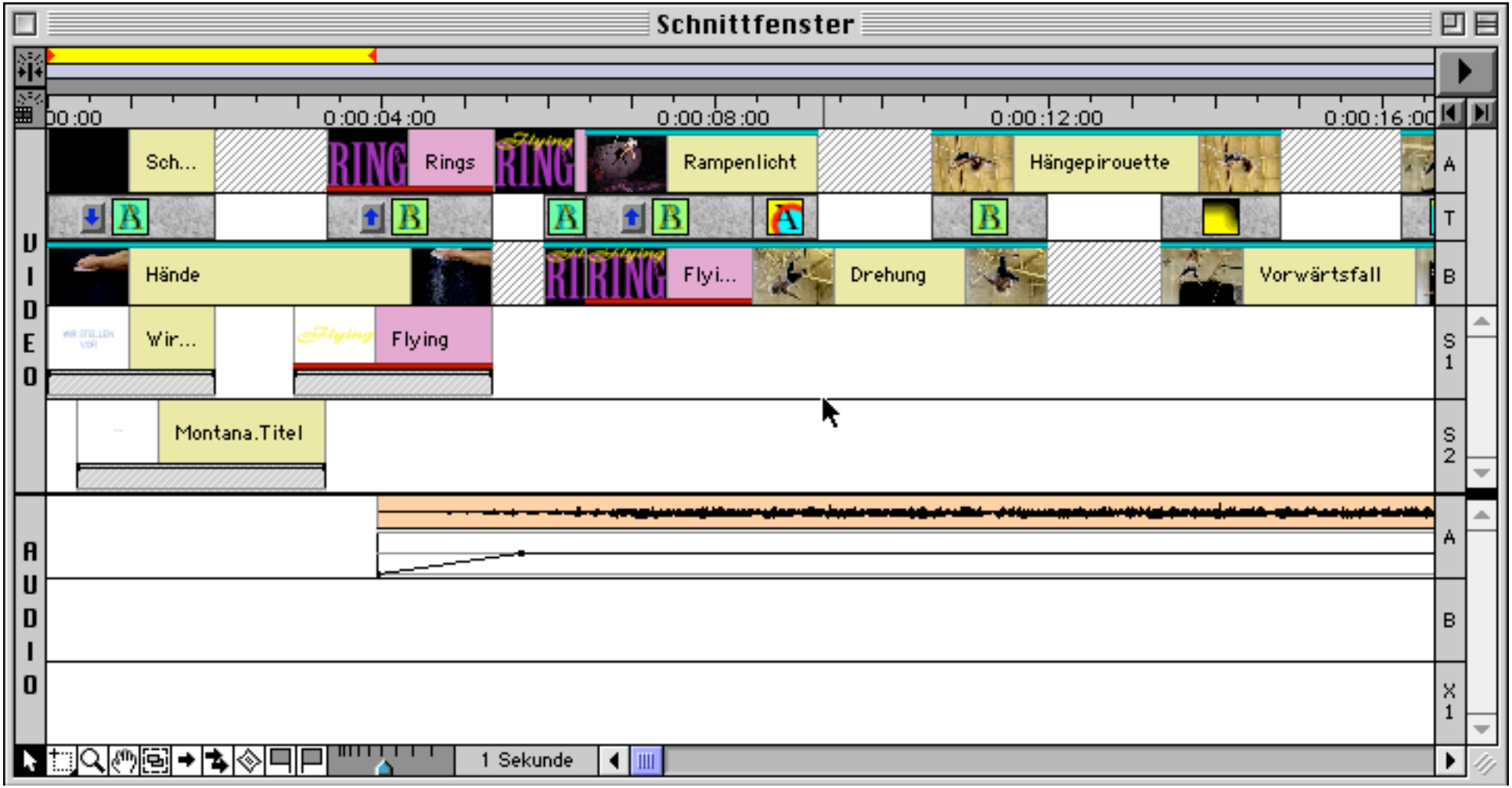
4.2 Videoschnitt

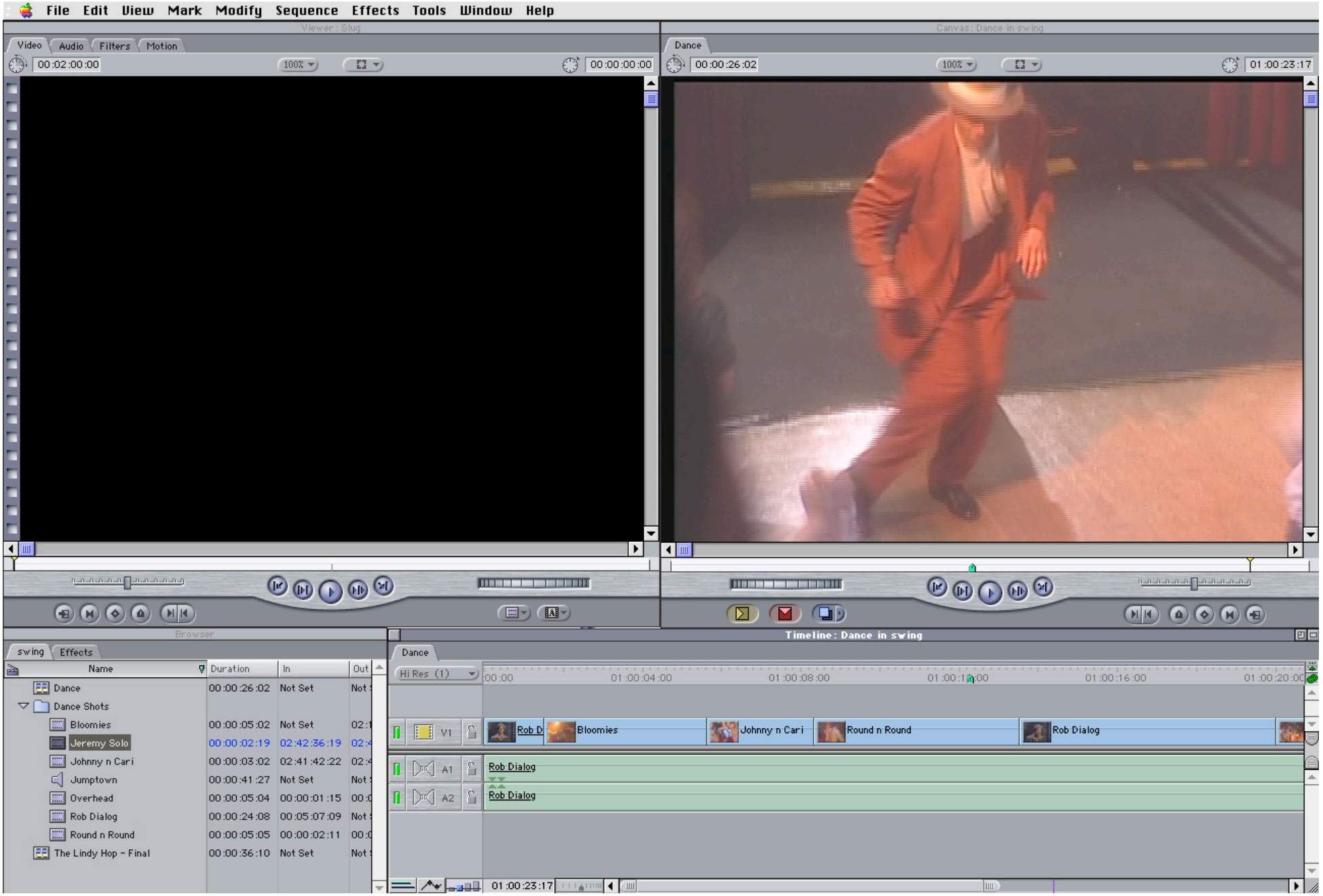
- Digital Compositing mit Zeitachse
  - Sequentielle Anordnung (Schnitt)
  - Räumlich Anordnung mit Zeitkomponente (Animation)

4.2.1 Adobe Premiere, Apple Final Cut Pro

- Projekt
  - technische Parameter
  - Menge von Komponenten: Audio, Video, Titel
  - Digitalisierung
  - Import
- Medienströme bearbeiten
  - editieren (In, Out)
  - Präsentationseigenschaften ändern
  - Filter ähnlich Bildern
  - einfache Effekte: Bewegung, Zoom, Drehung
- Produktion des Filmes
  - Movie-Datei in vielen Formaten
  - Edit Decision List für Schnittsysteme

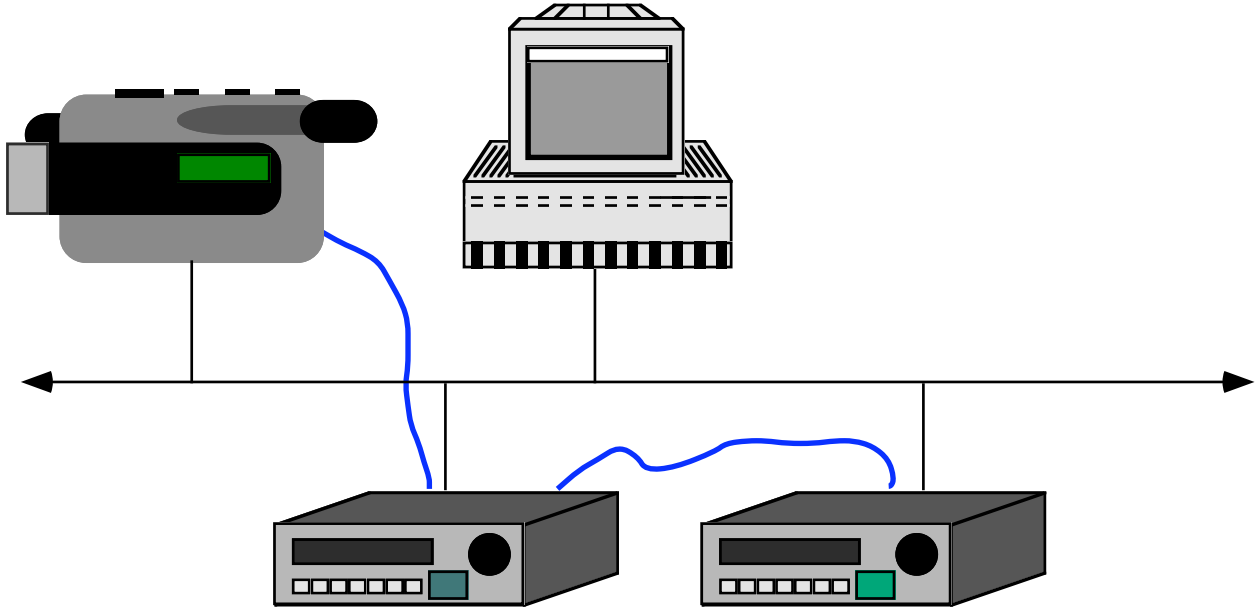
- Mehrere Spuren
  - zeitliche Anordnung
  - Übergänge gestalten




File Edit View Mark Modify Sequence Effects Tools Window Help
Viewer: Slug
Video Audio Filters Motion
00:02:00:00
100%
00:00:00:00
Canvas: Dance in swing
Dance
00:00:26:02
100%
01:00:23:17
Browser
swing Effects
Name Duration In Out
Dance 00:00:26:02 Not Set
Dance Shots
Bloomies 00:00:05:02 Not Set
Jeremy Solo 00:00:02:19 02:42:36:19
Johnny n Cari 00:00:03:02 02:41:42:22
Jumptown 00:00:41:27 Not Set
Overhead 00:00:05:04 00:00:01:15
Rob Dialog 00:00:24:08 00:05:07:09
Round n Round 00:00:05:05 00:00:02:11
The Lindy Hop - Final 00:00:36:10 Not Set
Timeline: Dance in swing
Dance
Hi Res (1)
01:00:04:00
01:00:08:00
01:00:16:00
01:00:20:00
V1
Rob D
Bloomies
Johnny n Cari
Round n Round
Rob Dialog
A1
Rob Dialog
A2
Rob Dialog
01:00:23:17

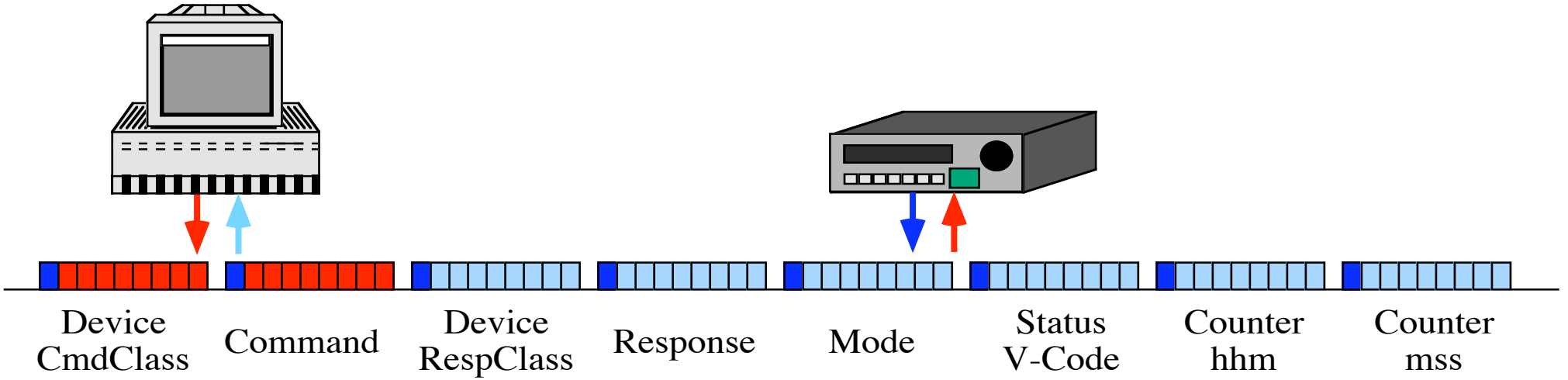
4.2.2 Rekorderkontrolle

- Ansteuerung von Videorekordern, -Kameras, …



  - Framegenau
  - Vorlauf, Rücklauf
  - Aufnahme, Wiedergabe, Stop, Pause
- ViSCA
- Control-S
  - VHS-VCR

- Control-L (LANC)
  - 8mm, Hi-8, Sony-VCR
  - serieller Bus, 8 Bit, 1 Startbit, 1 Stopbit, 9600 bit/s
  - CSMA/CD falls mehrere Geräte angeschlossen sind
  - Commander: Computer, Editor, Videorekorder mit Edit-Controller
  - Slave: Videorekorder, Camcorder, Kamera
  - pro Bild (Frame) ein Paket (50 bzw. 59,94 pro Sekunde)
  - Pakete mit 8 Byte
    - 2 Byte Commander -> Slave
    - 6 Byte Slave -> Commander

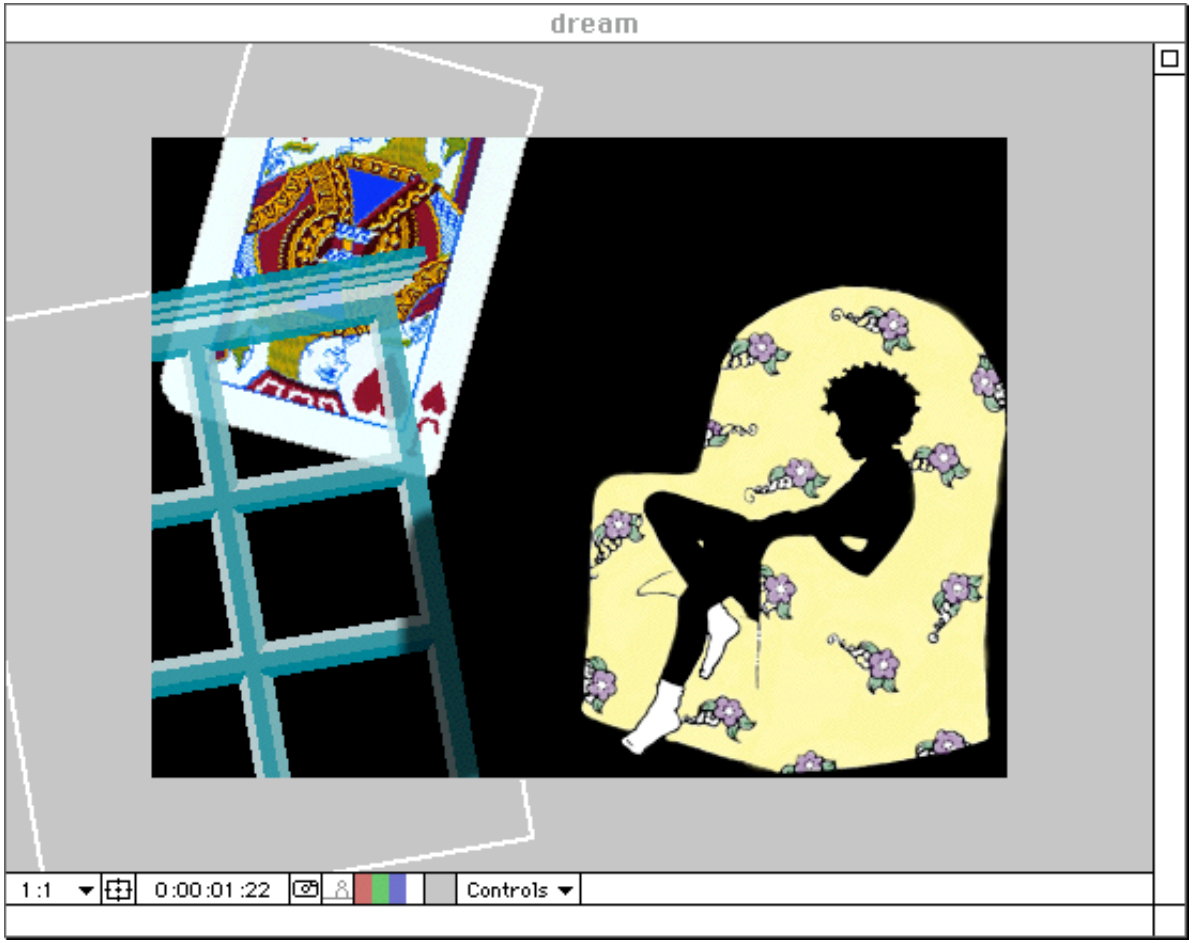


- Kommando in Byte 1
  - muß fünfmal wiederholt werden
  - Strom ein/aus, Stop, Pause, Play, Rewind, FF, Record, Frame advance +/-
- VTR-Mode: Tape Ejected, Stopped, FF, Rwd, REC, Playback, Still

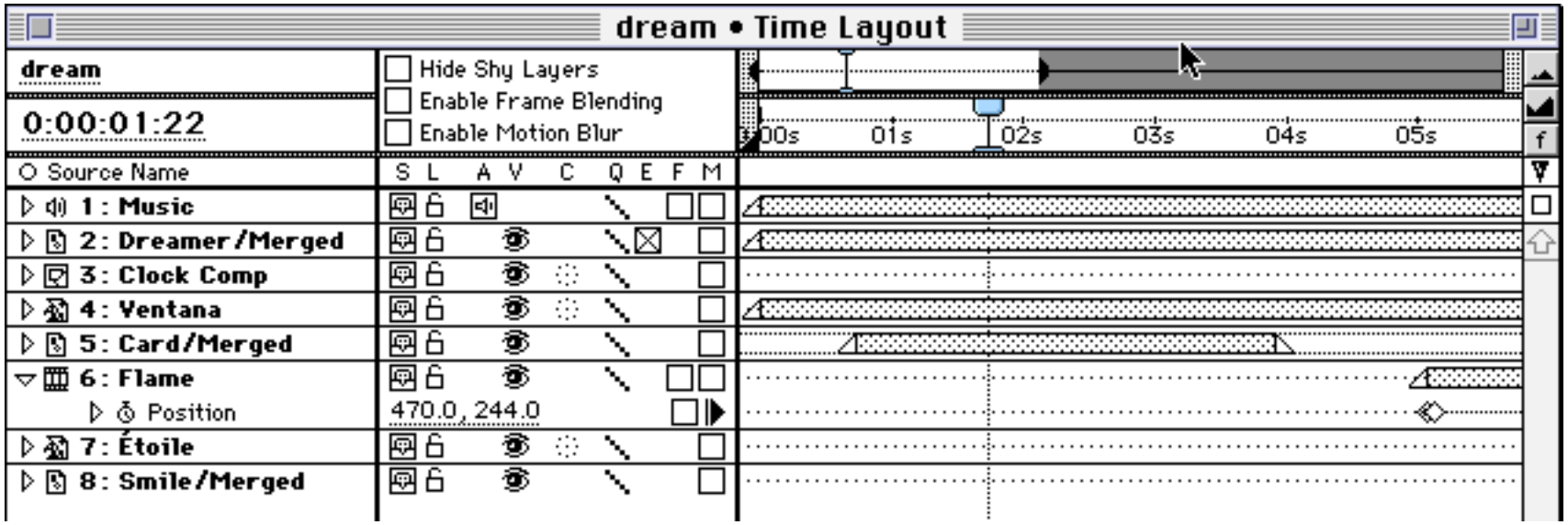
4.3 Effekte und Animation

4.3.1 After Effects

- Bilder lernen Laufen
  - Animationen
  - Zeichnungen, Text
- Objekte dynamisch anordnen
- Änderungen in der Zeit
  - Eigenschaften
  - Anordnung
- f(Objekt, t)



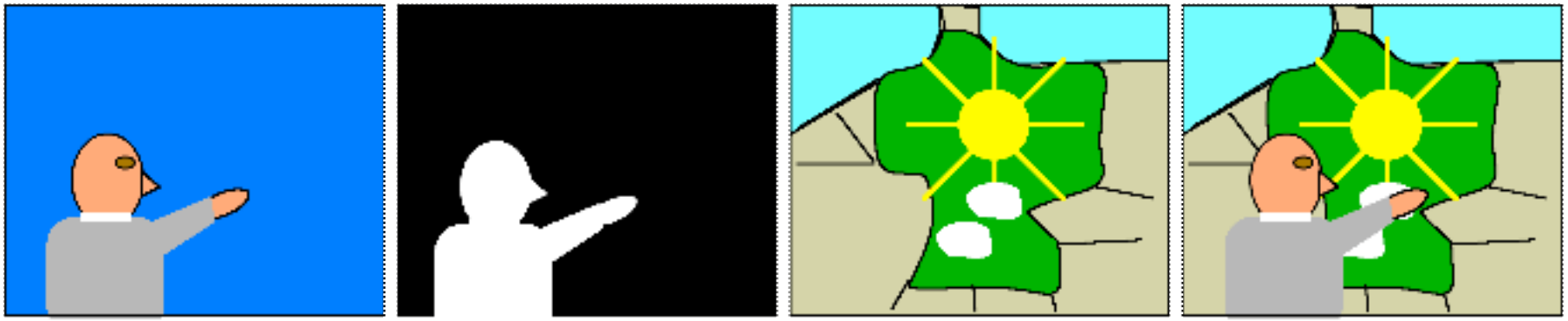
- Schichten (Layer) überlagern
  - Darstellung zeitlich begrenzt (In, Out)
  - Maske
- Zeitachsen und Keyframes



- Keyframes haben Schichteigenschaften als Attribute
  - Übergang zwischen Keyframes
    => gradueller Übergang von A1 nach A2
  - lineare oder komplexe Übergangsfunktion

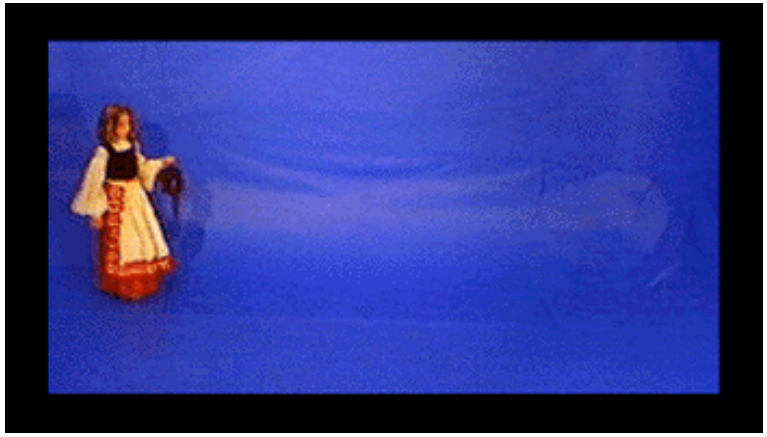
- Geometrische Attribute der Schichten in der Zeit
  - Größe
  - Bewegung entlang eines Pfades (Gerade, Bezier-Spline)
  - Rotation, Ankerpunkt
  - Bewegungsverzerrung
  - Durchsichtigkeit (Ein- und Ausblenden)
- Farb-Attribute
- Filter in der Zeit
  - Verschwimmen
  - Schatten
  - Textur
  - Erhebung, 3D, …
- Überblenden zwischen Schichten
  - Transparenz
  - Überblendeffekte (Dissolve, Wipe, Vorhang, …)

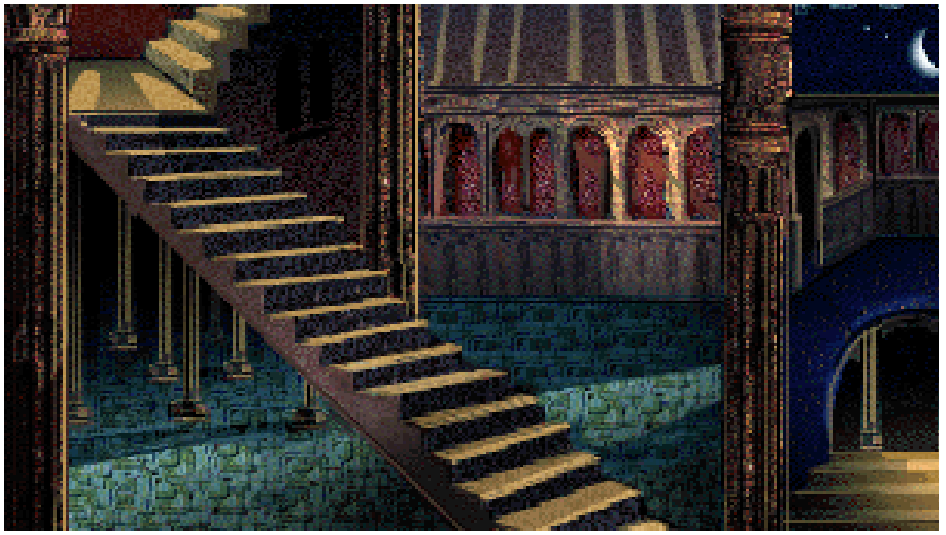
- Motion Pack
  - Identifizieren bewegter Elemente
  - Bewegungen glätten
  - Bewegungen 2. Ordnung
- Keying Pack
  - Wetterbericht



  - Teile des Orginalbildes werden 'transparent'
  - einfache Keys: Farbe, Helligkeit
  - scharfe Grenzen
  - Wertebereich der Transparenz
- Verzerrung von Objekten
  - perspektivisch
  - wellenförmig, Polarkoordinaten, Wirbel, …

- La Reine des Neiges
- Film mit Kind und gezeichneter Raum

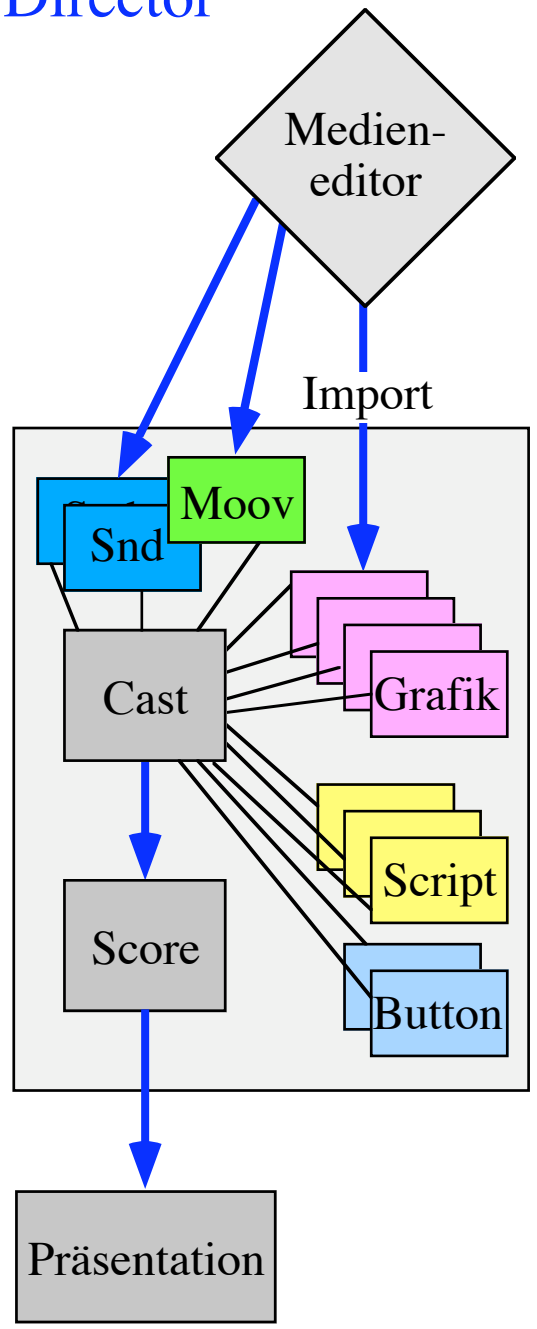




- Chroma-Key, Bewegungsfilter
- Komposition

4.4 Multimedia-Präsentationen erstellen: MacroMedia Director

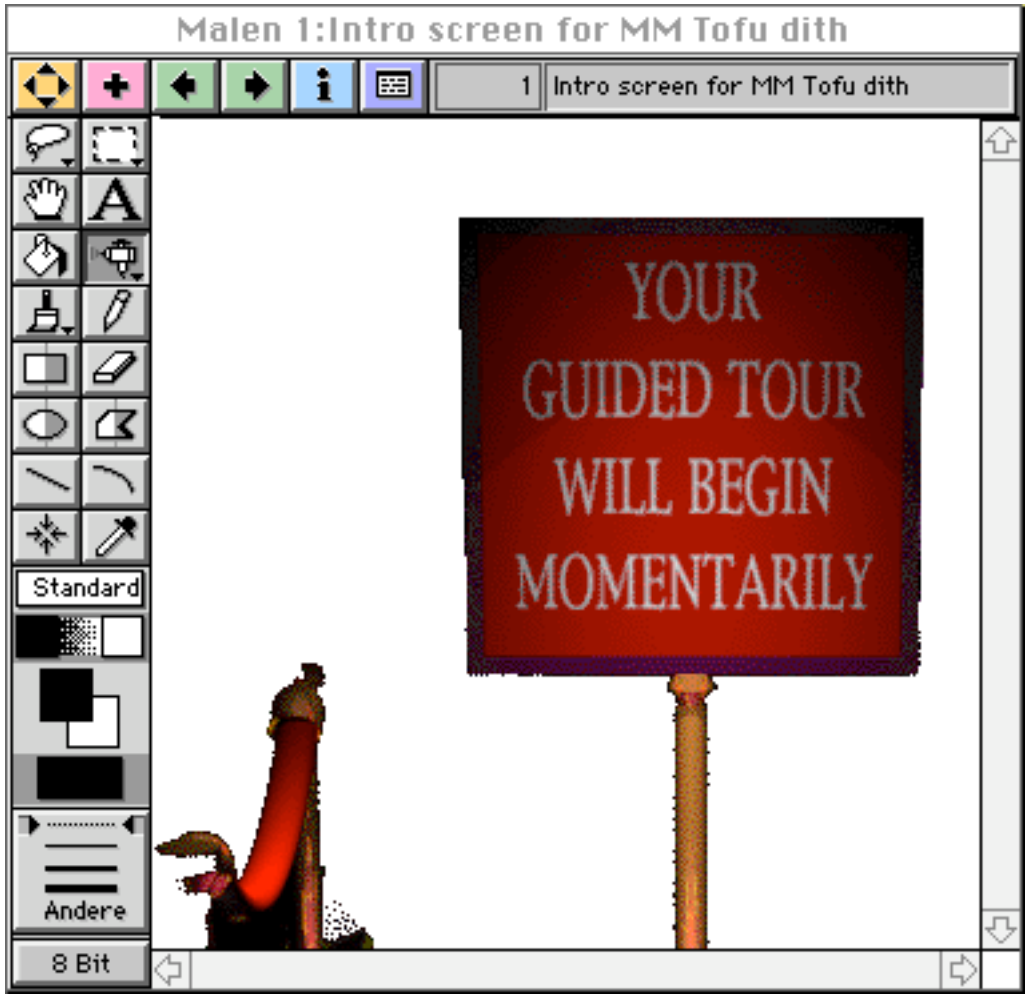
- Präsentationen
  - Animationen (nicht extrem anspruchsvoll)
  - Abspielen von Video und Sound
- Grafische Elemente
  - Text, Grafik, Video, Sound
  - 16 oder 256 Farben, verschiedene Paletten
  - Animation mit Sprites
- Benutzungsschnittstelle
  - simple Navigation mit Knöpfen
- Begriffe
  - Darsteller (actors) und Besetzung (cast)
  - Drehbuch (score)
  - Programmiersprache Lingo (scripts)



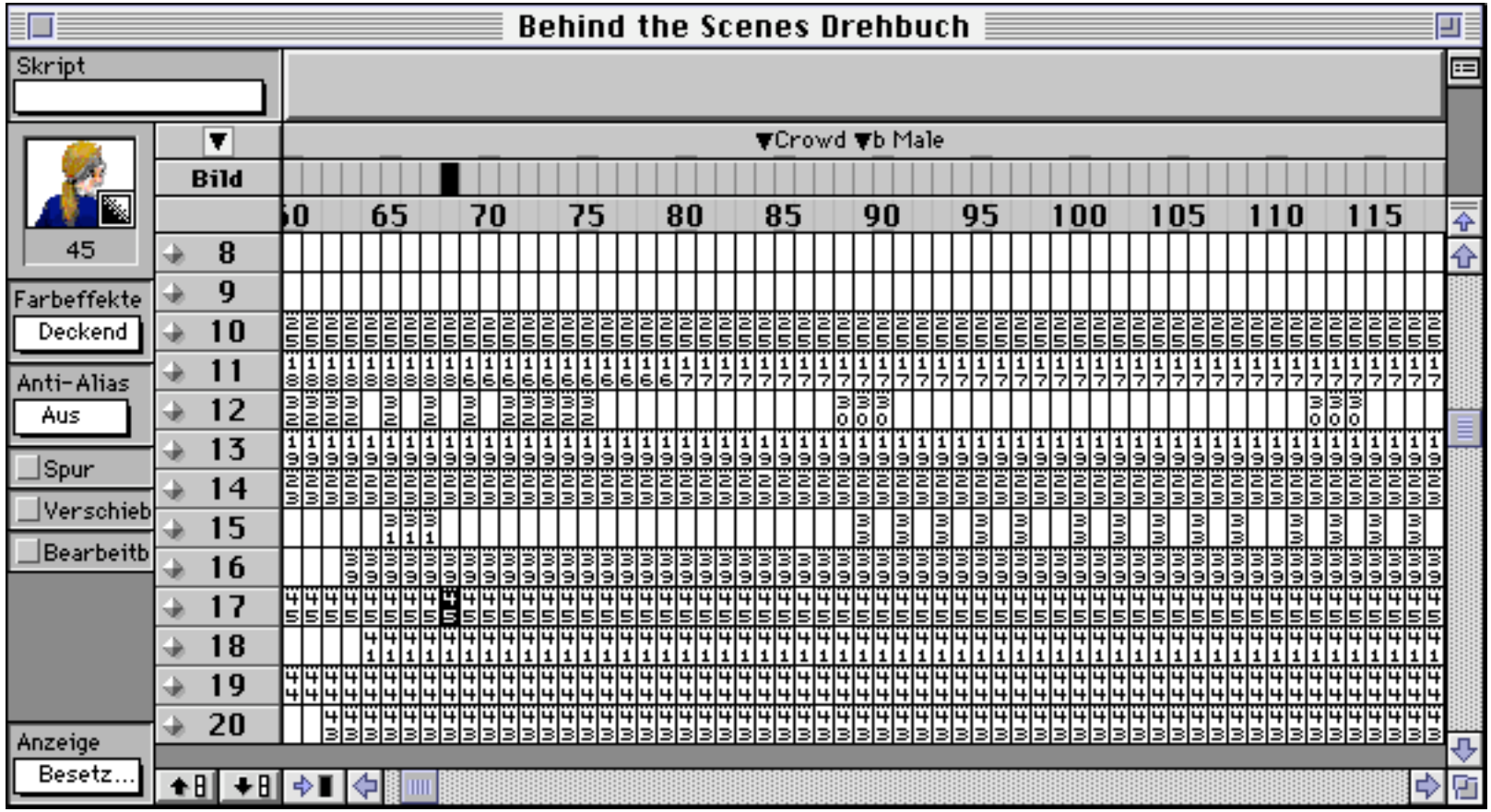
- Darsteller (actors) und Besetzung (cast)

- Grafische Objekte (PICT, PICT-Sequenz, Bitmap)
- QuickTime Movie
- Sound
- Button
- Script

- Zeichenwerkzeug integriert
  - Nachbearbeitung von importierten Bildern
  - Grafiken können auch Skripts haben



- Drehbuch (score)



- Zeit horizontal (Frames)
- vertikal 'Spuren'
- Nummern aus cast

- Programmiersprache Lingo (scripts)
  - erweitertes HyperTalk
  - Objektorientierung als Nachgedanke
- Basis-Lingo
  - on <Event> … end <Event>
  - set <Variable> of <Objekt> to <Wert>
  - set <Variable> = <Wert>
  - if <condition> then <clause> else <clause> end if
  - go <Identifier>
  - repeat with <Variable>=<Wert> to <Wert>
    …
    end repeat
  - <Identifier> ruft Prozedur
  - globale Variable
- Handler
  - mouseUp, mouseDown, startMovie, stopMovie
  - selbstdefiniert

```
on mouseUp
    go to frame "start"
end mouseUp
```

- Objektorientierung
  - Darstellerscript = Klasse
  - property-variable = Instanzvariable
  - ancestor-variable = Inheritance
  - birth ist Konstruktor (on birth)

```
MECH ,  76
global gMyPegBoard,gConstrainSprite,gIsRunning,gHandCursor,gSpeedCntrSprite
property   PEG_columns,PEG_rows,myHPegs  ,myVPegs,myAnimator,myButtons
on birth me
    set PEG_columns = 16                -- we have 16 columns of pegs
    …
    -- build the vert peg list
    set myVPegs = []                  -- empty list
    set count = 2
    setAt myVPegs,1,50                  -- set first array elt
    repeat while count < PEG_rows
        set pos = 50 + ((count - 1) * 30)
        setAT myVPegs , count,pos
        set count = count + 1
    end repeat
    …
    return me
end birth
```