Frameworks

- Multimedia-Frameworks
- Motivation, Ziele
- Komponenten: Player, Encoder, Server
- Streaming, RTP

- Windows-Media
- Windows Media Player
- Digital Rights Management

- Java Media Framework
- Konzept, Klassenhierarchie, Status
- Registry, Codec-Auswahl
- Codebeispiele

Literatur

Microsoft DirectX 8.1 documentation, msdn.microsoft.com/library/
Bargen, Donnelly, Inside Direct X, Microsoft Press, 1998

Java Media Framework API Guide, Sun Microsystems, 1999
JMF API specification, JMF examples, java.sun.com/products/java-media/jmf
Gordon, Talley, Essential JMF, Prentica Hall, 1999

Internet RFCs, hier: RFC 1889 (RTP/RTCP)

www.microsoft.com/windowsmedia
www.real.com
www.apple.com/quicktime/
java.sun.com/procucts/java-media/

Frameworks: Ziele

Infrastruktur ("Rahmenwerk") für

- Wiedergabe von Multimedia,
- möglichst viele Medientypen
- von lokalen Datenträgern
- oder Streaming über Netzwerke / Internet
- Synchronisation der beteiligten Medien

- Verwaltung der benötigten Codecs und Renderer
- Unterstützung der benötigten Netzwerkprotokolle

zusätzlich

- Unterstützung für Medienproduktion / -capturing
- Rechte-Management (Copyright, Kopierschutz, Pay-per-use, ...)
- Medien-Server

Frameworks: IBM Hotmedia Marketing . . .

A single, easy-to-use authoring environment provides the basis for creating a wide range of multimedia effects, resulting in a *single file* that contains a variety of multiple interactive elements and is easily added to a Web page. Just like that.

With HotMedia, today's Web professionals can take advantage of a number of opportunities in e-business, such as:

- e-tour: *Customers can take a virtual tour of any environment*
- e-care: *…or view and hear operational and assembly instructions online*
- e-tail: *…and interactively experience products and services.*

Frameworks: Beispiele

- Microsoft WindowsMedia
- Real Networks
- Apple Quicktime
- Sun Java Media Framework
  (Linux Simple Direct Layer, aber nur Ansätze zu Tools: xmps, xine, ...)
  (MPEG-4)

  enorme Marktbedeutung, daher:
- Player meistens gratis verfügbar
- Bindung an proprietäre (undokumentierte) Protokolle
- Authoring-Tools für "low quality" frei verfügbar
- aber professionelle Versionen kommerziell
- Medien-Server "schweineteuer"

- zunehmende Bedeutung von "Digital Rights" Tools

Frameworks: Grundfunktionen

- Lesezugriff auf Medien-Dateien oder -Streams
- Erkennung des jeweiligen Datenformats (z.B. AVI oder Qt)
- Bereitsstellen / Zugriff auf Codecs (z.B. MPEG-2)
- zusätzliche Filter (z.B. YUV nach RGB)
- Gerätezugriff zur Ein- und Ausgabe (z.B. Framebuffer, DV-Kamera)

- Player-Applikation w/o GUI
- Player-Utilities: playlists, cddb-Zugriff, ...

- Streaming, Quality of Service (QoS)
- Sicherheits- und Copyright-Funktionen

Frameworks: Synchronisation

- Video 25fps: jedes Frame jeweils 40 ms
- Audio 48KHz: jedes Sample 20.8 µs
  usw.

aber: wie genau muss die Synchronisation passen?

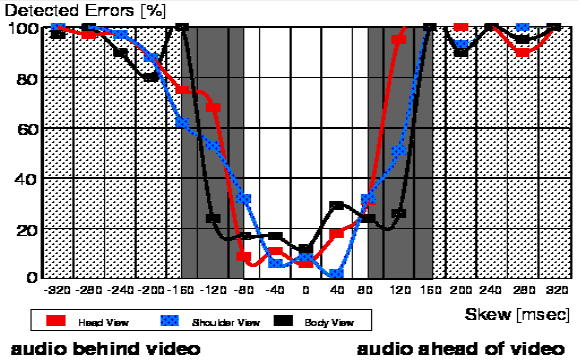
- Puffergrösse / Latenz / QoS (re-transmission) optimieren

=> Experimente mit Testpersonen
- Versatz Video / Musik / Sprache variieren
- individuelle Bewertung der wahrgenommenen Fehler

Empfehlungen (Beispiele, siehe Steinmetz 5.6 für Details):

Audio (Versatz von Stereosamples)	+/- 11 µs
Audio (Dialog mehrerer Personen)	+/- 120 ms
Audio (Hintergrundmusik)	+/- 500 ms
Video (Lippensynchronisation)	+/- 80 ms

Beispiel: Lippen-Synchronisation



Abhängigkeiten:

- Video-Inhalt, z.B. Sprache vs. Ereignisse (z.B. Hammerschlag)
- Video-Hintergrund, wg. Ablenkung
- Kamera-Einstellung (Totale vs. Figur vs. nur Kopf)
- nur Sprache oder Hintergrundgeräusche oder Musik
- Eigenheiten der jeweiligen Sprache

(siehe Steinmetz, Kapitel "media synchronization")

Streaming: Definition

"Streaming Media / Audio" :=

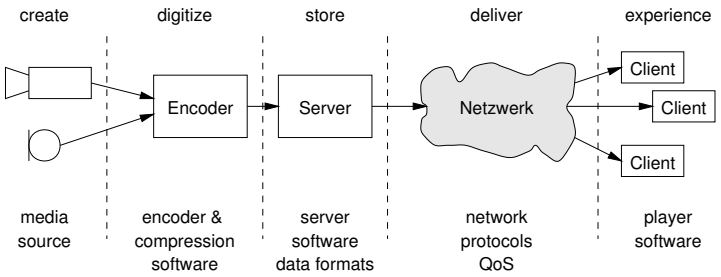
- Echtzeit-Übertragung von Mediendaten
- von einem Server zu einem / vielen Clients

- ohne vorheriges Laden der gesamten Daten
- Beschränkung auf Bitrate des Kanals, mit Schwankungen
- erfordert Fehlertoleranzmechanismen
- Tradeoff Qualität / Robustheit / Bitrate

- Audio auch standalone ("Internet Radio")
- aber meistens Video / Animationen mit sychronem Audio

- großes Marktpotential erwartet ("Video on demand", ...)
- Microsoft WMA, RealAudio, Apple Quicktime, Liquid Audio, ...
- Mediensuche / Tauschbörsen (Napster) / Copyright-Fragen

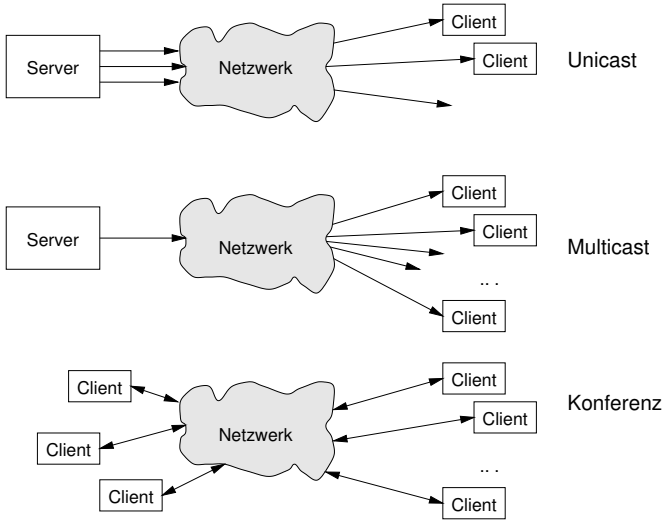
Streaming: System



funktionierendes System erfordert alle Komponenten:

- alle Anbieter liefern "integrierte Lösung"
- Encoder + Server + Management + Player (+ Verschlüsselung)
- Microsoft, RealNetworks, Apple, ...
- Live-Übertragung nur mit Echtzeit-Encoder

Streaming: Szenarien



Streaming: Bandbreiten

"streaming" ist stark von verfügbarer Bandbreite abhängig:

Modem	28 .. 56	Kb/s
ISDN	64	Kb/s
DSL	128 .. 768	Kb/s
Intranet	10 .. 100	Mb/s
MPEG-2 (DVD)	4 .. 9	Mb/s
MPEG-1, CDDA	1.5	Mb/s
MPEG-4 (DivX)	1.5	Mb/s
MP3 Audio	64 .. 320	Kb/s
RealAudio 8	10 .. 160	Kb/s

(bis ca. 4 Mb/s)

=> selbst MP3 nicht per Modem streaming-fähig

=> Bildtelefonie (H.263 QCIF 15fps) erfordert mindestens ISDN

Streaming: Internet

Multimedia-Übertragung im Internet ("IP"):

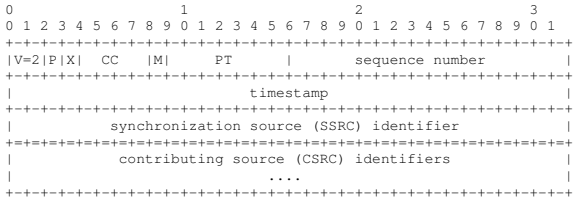
- hohe Bandbreiten erforderlich (s.o.)
- für Unicast / Multicast- Übertragungen
- nur Paketvermittlung, keine Punkt-zu-Punkt-Verbindungen
- Echtzeitanforderungen: z.B. Latenz < 250ms für Telephonie

=> "klassische" Protokolle (ftp) für Streaming ungeeignet
=> neue Protokolle notwendig

- verlorene Daten (dropped frames) oft tolerierbar
- verlorene Pakete meistens bei überlastetem Netzwerk
- wiederholte Übertragung verschlimmert das Problem

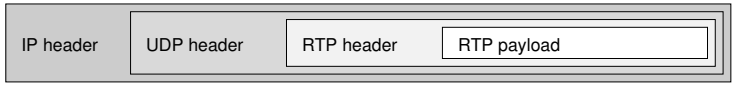
=> auf IP/UDP aufsetzen, nicht auf TCP

RTP: Packet-Header



V:	version number (2)
P:	padding
CC:	CSRC count (>1 if payload mixes data from several sources)
M:	marker, e.g. "frame boundary"
PT:	payload type
sn, ts	sequence number, timestamp

RTP



"real time transport protocol", RFC1889 / RFC1890

- setzt auf UDP auf — aber auch RTP auf IP direkt möglich
- timestamps — für Streaming / Echtzeitwiedergabe
- sequence numbers — weil UDP keine Reihenfolge garantiert
- payload type identifier — z.B. PCM, MP3, MPEG-2, H.261
- source identification — z.B. Sprecher bei einer Telekonferenz
- encryption

- in Quicktime / RealAudio als Transportprotokoll verwendet

RTCP: RTP control protocol

- periodische Übertragung von Kontrolldaten
- zwischen allen Beteiligten (Server <-> Clients)

Funktionen:

- Statusmeldungen der Clients (Verlustrate, Jitter, ...)
- eindeutige Kennzeichnung aller Quellen (CNAME)
- Adaption der RTCP Paketrate (wegen Skalierbarkeit)
- Session-Control

Pakettypen:

SR	sender report	transmission statistics
RR	receiver report	perception statistics
SDES	source description	incl. CNAME
BYE	end of participation	
APP	application specific function	

RTCP: Receiver Report

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|    RC   |   PT=RR=201   |             length            | header
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     SSRC of packet sender                     |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                 SSRC_1 (SSRC of first source)                 | report
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ block
| fraction lost |       cumulative number of packets lost       |   1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           extended highest sequence number received           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      interarrival jitter                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         last SR (LSR)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   delay since last SR (DLSR)                  |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                 SSRC_2 (SSRC of second source)                | report
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ block
:                               ...                             :   2
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                  profile-specific extensions                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```


WindowsMedia: Ziele, Features

"deliver digital media, such as songs and videos,
through the Internet in a protected and secure manner. [...]
Provides tools for protecting digital files so you distribute
them and maintain copyright protection. [...]
Solutions are flexible enough to be easily adapted to your
current business model or to create new ones."

(Microsoft Media Rights Manager 7 SDK)

Komponenten des WindowsMedia Konzepts:

- Windows Media Player
- Windows Media Tools
- Windows Media Rights Manager

- Player auch für nicht-Windows Plattformen verfügbar
  derzeit: Macintosh, Solaris (!), Pocket-PC, "palm-size PC"

Windows Media Homepage

(07.01.2002)

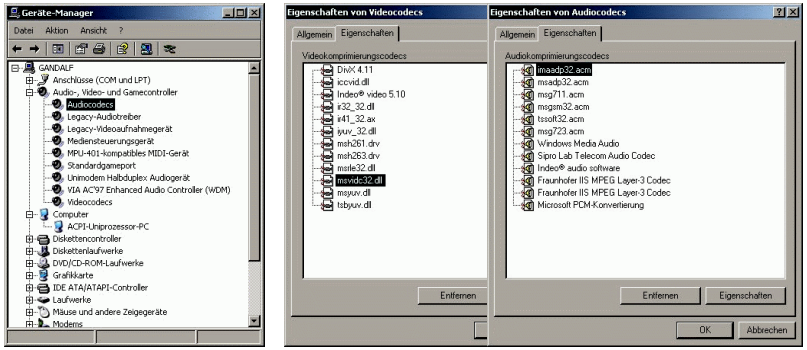
WindowsMedia: Media Player

Skins :-(

wesentliche Veränderungen seit Player 6.x auch unter der Oberfläche:

- online feedback über Mediennutzung
- unterstützt Windows Media Rights Manager
- z.B. "counted operations", "expiration", ...

Windows Media: Medientypen

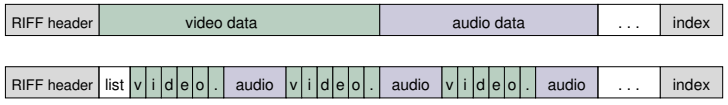


- vorinstallierte Codecs in Windows XP Home (plus OpenDivX)
- u.a. MPEG-1, -2, -4, H.26x, Indeo 5, diverse Audioformate
- automatischer Download weiterer Codecs

AVI: Beispiel

```
(0x00000000)     ID:<RIFF>   Size:0x000fe964                    RIFF: AVI
                 Form Type = <AVI >
(0x0000000c)         ID:<LIST>   Size:0x000007d4                Header
                     List Type = <hdrl>
(0x00000018)             ID:<avih>   Size:0x00000038                [c't 11/94 327ff]
(0x00000058)             ID:<LIST>   Size:0x00000474
                         List Type = <strl>
(0x00000064)                 ID:<strh>   Size:0x00000038
(0x000000a4)                 ID:<strf>   Size:0x00000428
(0x000004d4)             ID:<LIST>   Size:0x0000005c
                         List Type = <strl>
(0x000004e0)                 ID:<strh>   Size:0x00000038
(0x00000520)                 ID:<strf>   Size:0x00000010
(0x00000538)             ID:<vedt>   Size:0x00000008
(0x00000548)             ID:<JUNK>   Size:0x00000298
(0x000007e8)         ID:<LIST>   Size:0x000fd184
                     List Type = <movi>                         Daten (MOVI):
(0x000007f4)             ID:<LIST>   Size:0x000005b0            Audio / Video / ...
                         List Type = <rec >
(0x00000800)                 ID:<01wb>   Size:0x000005a4
...                          /* einige Male nur Sound */
(0x00004c98)             ID:<LIST>   Size:0x000019da
                         List Type = <rec >
(0x00004ca4)                 ID:<00dc>   Size:0x00001422
(0x000060ce)                 ID:<01wb>   Size:0x000005a4
...                          /* diverse Male Sound und Bild */
(0x000db804)             ID:<LIST>   Size:0x00002d72
                         List Type = <rec >
(0x000db810)                 ID:<00dc>   Size:0x00002d66
...                          /* einige Male nur Bild */
(0x000fd974)         ID:<idx1>   Size:0x00000ff0                Index
```


AVI: Audio Video Interleaved



normal / interleaved

- RIFF-Dateiformat für Multimedia / Videos
- eingeführt mit Video for Windows (Win 3.11)
- definiert ca. 20 Chunk-Typen
- List-Chunk erlaubt "verwobene" Daten
  - Audiodaten, Videoframes (BMP), Audiodaten, ...
  - Datei muß vor Abspielen nicht voll geladen werden
- Spezifikation in Windows API, Übersicht in c't 94/11 S.327
- mittlerweile von ASF bzw. WMA/WMV abgelöst (s.u.)

ASF: Advanced Streaming Format

"... an extensible file format designed to store synchronized multimedia data.
It supports data delivery over a wide variety of networks and protocols,
while still proving suitable for local playback. The explicit goal of ASF
is to provide a basis for industry-wide multimedia interoperability,
with ASF being adopted by all major streaming solution providers."

(Microsoft, Real Networks 1998)

- RIFF-ähnliche Struktur: header, index, interleaved data
- Chunks (=:"objects") per GUID gekennzeichnet
  - bei Bedarf Registrierung der GUIDs bei Microsoft
  - erlaubt z.B. Codec-Download
  - auch der Player identiziert sich über seine GUID
  - sehr feine Copyright / Nutzungskontrolle
- siehe ASF-Spezifikation (Version 1.0, 26.02.1998)
- aber aktuelle Version (WMA/WMV) nicht mehr dokumentiert

ASF: GUID

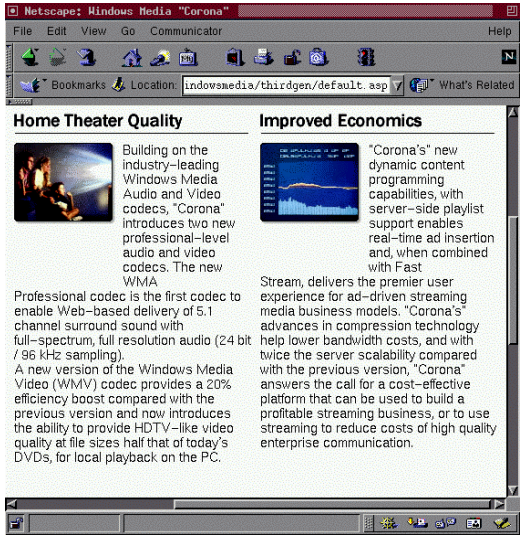
GUID/UUID := "globally/universally unique IDs"

- 128-bit Signatur
- entwickelt für NCS (Apollo), übernommen in OSF/DCE und Windows
  - Ethernet-MAC Adresse plus Zeitmarke (eindeutig)
  - oder 47-bit Zufallsadresse plus Zeitmarke
  - Zeit mit 100ns Auflösung, Sequenznr. zur Korrektur (reboot)
  - eindeutig bis 3400 n.Chr.

ASF-Header:	`D6E229D1-35DA-11DA-9034-00A0C90349BE`
ASF-Data:	`D6E229D2-35DA-11DA-9034-00A0C90349BE`
ASF-Index:	`D6E229D3-35DA-11DA-9034-00A0C90349BE`
usw.	`time-low-mid--high|seq.|ethernet----`

- u.a. auch jeder Windows-Rechner eindeutig identifzierbar

Windows Media: 3rd generation



(www.microsoft.com, 07.01.2002)

Windows Media: Codec-Auswahl

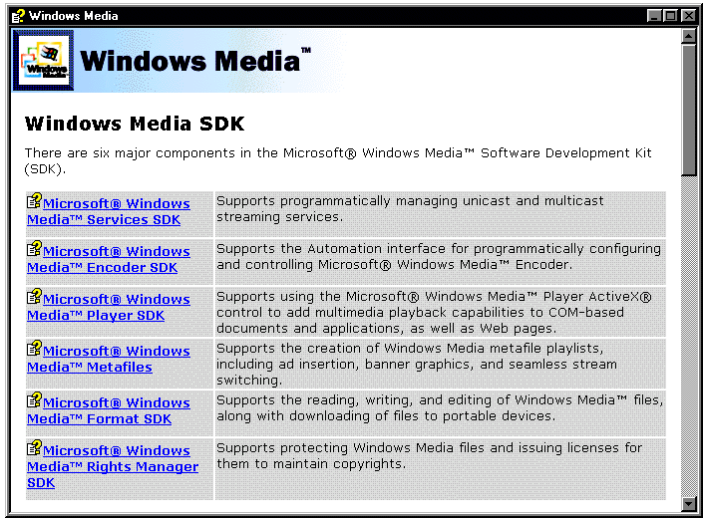
AVI-Format (seit Windows 3.11):

- RIFF-Datei, einzelne Chunks mit "fourcc" Kennung
- Auswahl zum "fourcc" passender Codecs via Registry
- Microsoft verwaltet zentrale Liste aller registrierten Codecs
  www.microsoft.com/hwdev/archive/devdes/fourcc.asp
  www.webartz.com/fourcc/fcccodec.htm

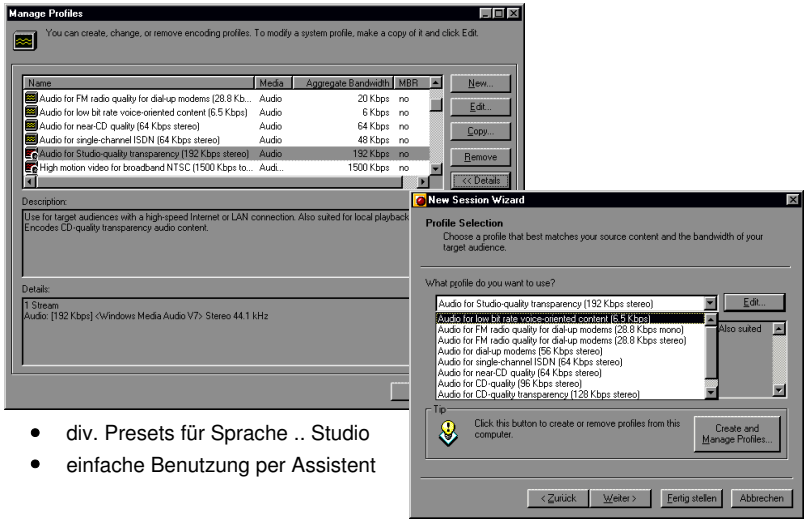
ASF-/WMx-Format:

- Codecs-Verwaltung via GUID
- optionale Chunks für Codec-Downloadseiten
- automatischer Download via www.microsoft.com
- Player-Update entfernt "nicht untersützte" Codecs ...

WindowsMedia: Authoring



WindowsMedia: Audio Profiles, Wizard



- div. Presets für Sprache .. Studio
- einfache Benutzung per Assistent

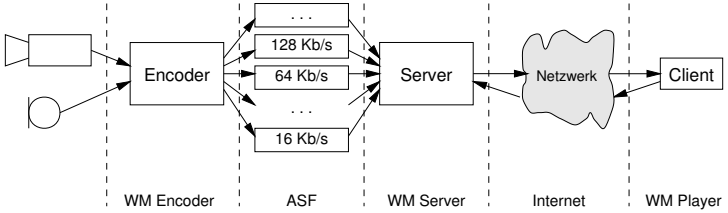
WindowsMedia: Security

- mögliche Restriktionen für WMA/WMV-Dateien:
  - play on PC
  - counted play
  - start date
  - expire date
  - burn to Audio CD
  - counted CD burn
  - transfer to portable device
  - ... and many more
- Player überprüft vorhandene Lizenzen
- oder versucht, neue Lizenz zu erhalten:

  "might issue a licence silently, so the consumer is unaware of the process"

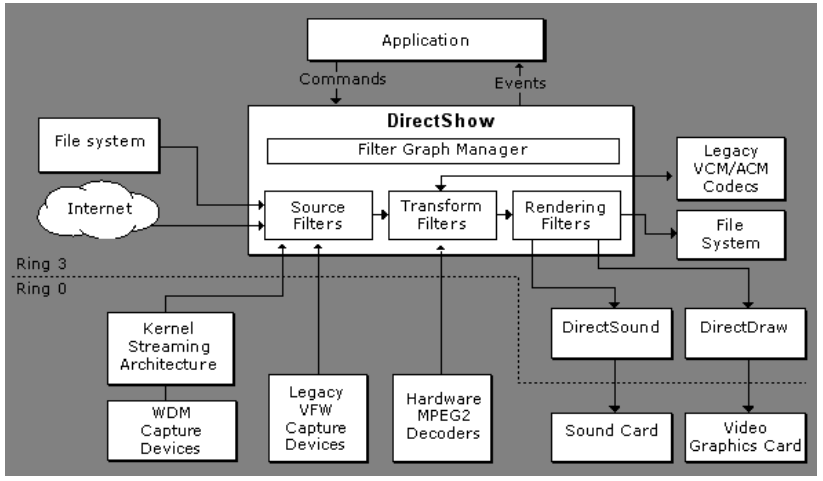
  "... supports pre-delivered licences"

WindowsMedia: Intelligent Streaming



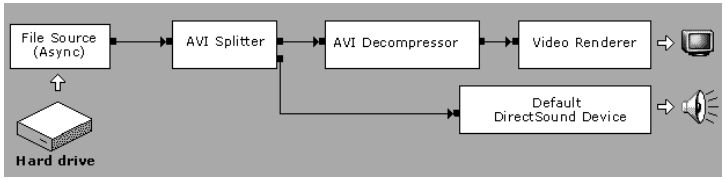
- Encoder erzeugt (optional) spezielle ASF-Datei:
  - mehrere parallele Datenströme
  - mit verschiedenen Bitraten
  - entsprechende Qualitätsstufen
- Player sendet Feedback über Bandbreite und Paketverluste
- Server sendet nur den Datenstrom der max. möglichen Qualität

DirectShow: Übersicht



(msdn.microsoft.com/library/default.asp?url=/library/en-us/dx8_c/directx_cpp/htm/directshowsystemoverview.asp)

DirectShow: Filter

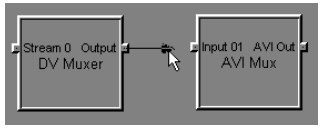


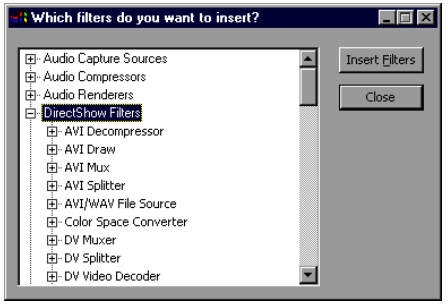
"Media Streaming":

- DirectShow besteht aus einzelnen (COM-) Komponenten:
- Codecs, Multiplexer/Splitter, Renderer, usw.
- Verkettung nur über definierte Schnittstellen ("pins")
- zentraler FilterGraphManager zur Synchronisation

- Applikation kombiniert einfach die einzelnen Filter

DirectShow: GraphEdit



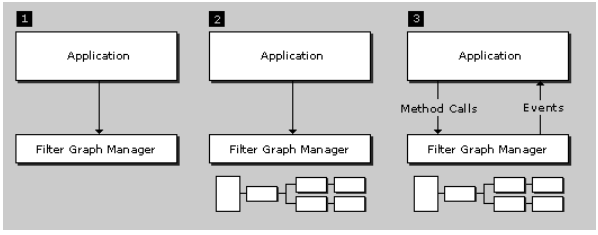


GraphEdit:

- interaktiver Editor zum Aufbau der Filtergraphen
- "drag and drop"
- Zugriff auf alle installierten Codecs / Filter
- direktes Austesten des Graphen
- schreibt Quelltext für FilterGraphManager

(msdn.microsoft.com/library/default.asp?url=/library/en-us/dx8_c/directx_cpp/htm/usinggraphedit.asp)

DirectShow: FilterGraphManager



Konstruktion des Filter-Graphen:

- "per Hand", Applikation gibt die Filter fest vor
- automatisch via FilterGraphManager:
  - Suche aller passenden Filter für Eingabe- / Ausgabeformate
  - Einbau evtl. benötigter Formatkonverter
  - Algorithmus terminiert, da gerichtete/zyklenfreie Graphen

DirectShow: Simple Player

```
#include <dshow.h>  // simple player demo, DirectX 8.1
void main(void)
{
  IGraphBuilder *pGraph;
  IMediaControl *pMediaControl;
  IMediaEvent   *pEvent;
  CoInitialize(NULL);  // initialize COM DLLs

  // Create the filter graph manager and query for interfaces.
  CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC_SERVER,
                   IID_IGraphBuilder, (void **)&pGraph);
  pGraph->QueryInterface(IID_IMediaControl, (void **)&pMediaControl);
  pGraph->QueryInterface(IID_IMediaEvent, (void **)&pEvent);

  // Build the graph. IMPORTANT: Change string to a file on your system.
  pGraph->RenderFile( "C:\\Example.avi", NULL);

  // Run the graph.
  pMediaControl->Run();

  // Wait for completion.
  long evCode;
  pEvent->WaitForCompletion(INFINITE, &evCode);

  // Clean up.
  pMediaControl->Release();
  pEvent->Release();
  pGraph->Release();
  CoUninitialize();
}
```

JMF: Java Media Framework

- Java-API zur Verarbeitung und Präsentation zeitbasierter Medien
- JMF 1.0 (nur Wiedergabe) von Sun, Intel, SGI
- JMF 2.0 (inkl. Capturing) von Sun und IBM

- diverse Audio-/Video-Codecs
- pure-Java oder "performance pack" Versionen
- Streaming via RTP/RTCP

Ziele:

- Integration von "Medien" in Java-Applikationen und -Applets
- einfache Programmierschnittstelle

- JSR 135: Multimedia API for J2ME

(www.javasoft.com/products/java-media/jmf)

JMF: Browser-Unterstützung

```
...
<applet code=SimplePlayerApplet width=320 height=300>
  <param name=file value="sun.avi">
</applet>
...
```

JMF läuft auch als Applet:

- in Java-fähigen Browsern (JDK 1.1+)
- erfordert lokale Installation des JMF
- kein ständiger Codec-/Code-Download
- allerdings: Class-Download << Medien-Download ...

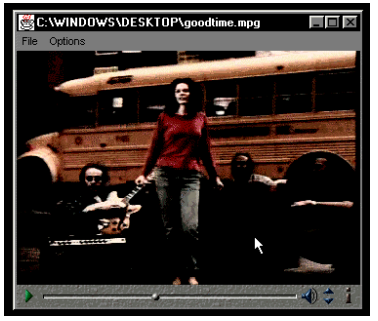
- SecurityManager regelt Zugriff auf (Capture-) Devices

JMF: Medientypen

		"cross platform"	"performance packs"	
		pure-Java	Solaris	Windows
PCM,ADPCM	.wav	D,E	D,E	D,E
G.711 (u-law)	.wav	D,E	D,E	D,E
GSM	.gsm	D,E	D,E	D,E
MP2,MP3	.mp3	D	D/E	D,E
MIDI	.midi	--	D	D
Windows ACM		--	--	D,E
JPEG	.jpg	D	D/E	D/E
Qt Cinepak	.mov	D	D/E	D
MPEG-1	.mpeg	--	D	D
H261/H263		- / D	D / D,E	D / D,E
Flash	.swf	D	D	D
Hotmedia	.mvr	D	D	D
Windows VCM	.avi	--	--	D/E

(in JMF 2.1.1, java.sun.com/products/java-media/jmf/)

JMF: Player / PlayerBean



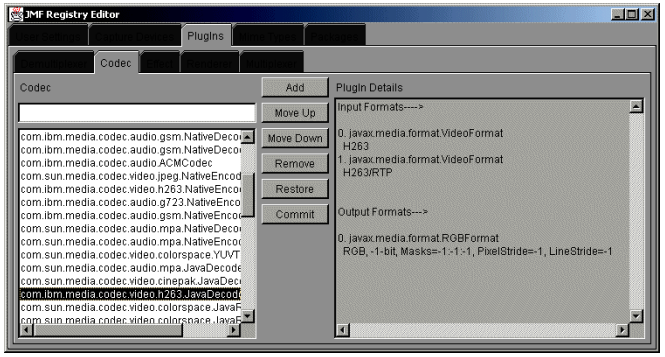
media location
name
size, position
aspect ratio
audio volume
show control panel
show caching controls
start, stop, search, loop

- einfacher Medienplayer, minimale GUI:
- Transportkontrolle, Lautstärkeregelung, Medieninfo
- auch als Komponente zur direkten Integration in Anwendungen: javax.media.bean.playerbean.MediaPlayer

JMF Player: Beispielcode

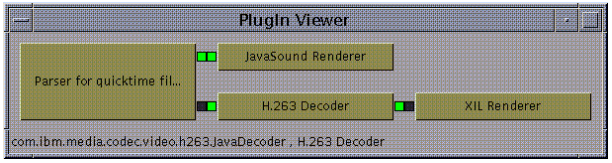
```
public class SimplePlayerApplet extends Applet implements ControllerListener {
  Player              player = null; // the media player
  Component  visualComponent = null; // the (video) display AWT component
  Component controlComponent = null; // AWT component for the controls

  public void init() {
    String mediaName = getParameter( "FILE" );
    URL    mediuURL  = new URL( getDocumentBase(), mediaName );
    MediaLocator mrl = new MediaLocator( url.toExternalForm() );
    player = Manager.createPlayer(mrl);  // create a player
    player.addControllerListener(this);  // we handle the player events
  }
  public void start() { // starts the player (including prefetch)
    if (player != null) player.start();
  }
  public void stop() {
    if (player != null) { player.stop(); player.deallocate(); }
  }
  public void destroy() {
    player.close();
  }
  public synchronized void controllerUpdate(ControllerEvent event) {
    if (event instanceof RealizeCompleteEvent) { // player ok, create GUI now
      controlComponent = player.getControlPanelComponent();
      visualComponent = player.getVisualComponent();
      panel.add(controlComponent); panel.add(visualComponent);
    } else if (event instanceof CachingControlEvent) { // update progress bar
    } else if (event instanceof ...) {                 // handle other events
    }
  }
}
```


JMF: Registry



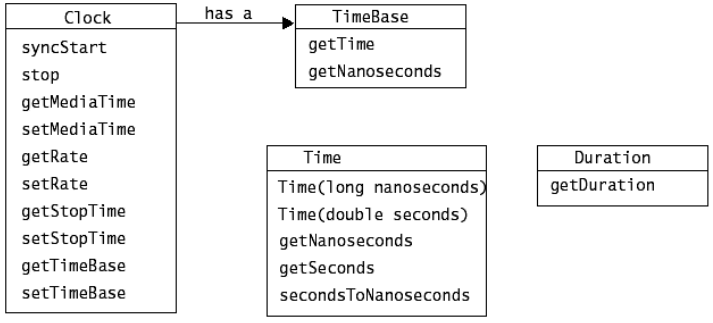
- Liste der Default-Codecs in JMF 2.1.1 ("Windows performance pack")
- JMF "registry" enthält Liste der vorhandenen Codecs
- zusätzlich Download via Classloader

JMF: Codec-Graph



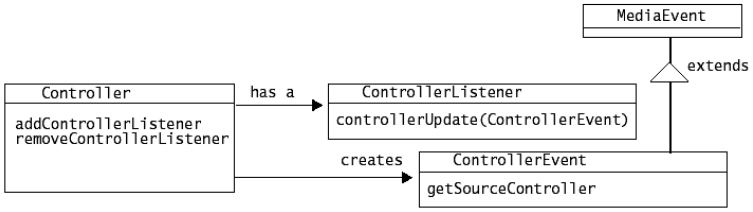
- Konzept wie bei DirectShow:
- Demultiplexer, Codec, Effect, Multiplexer, Renderer
- jeweils als abgeleitete Klasse von "Player" bzw. "Processor"

JMF: Clock, TimeBase, Time



- interne Zeitberechnung mit 1ns Auflösung
- Time, Duration: Abstraktion von Zeit bzw. Zeitintervallen
- Clock: Klasse zur Synchronisation

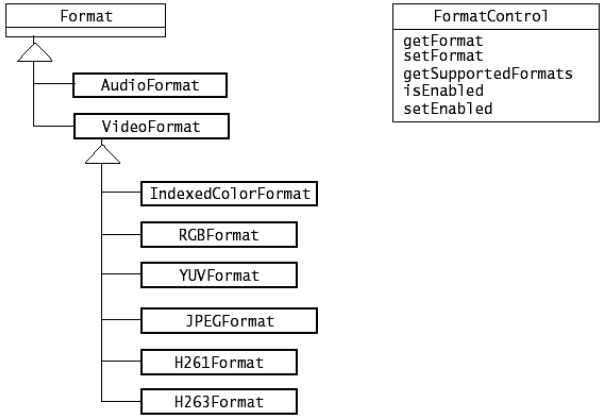
JMF: Events



ControllerListener / ControllerEvent

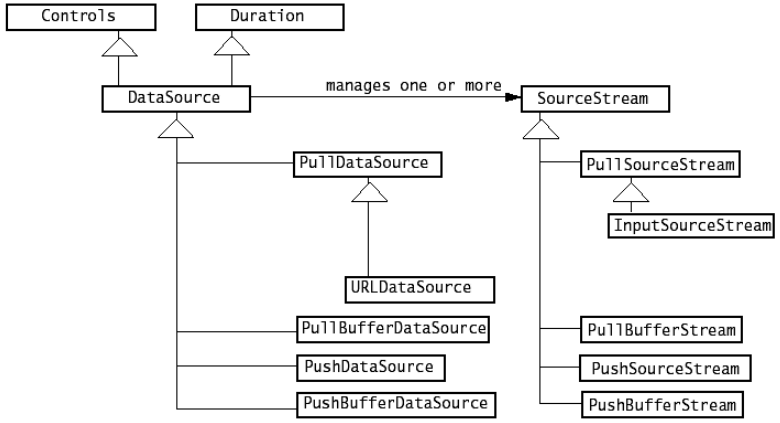
- Abstraktion für alle (!) Medien-relevanten Ereignisse, z.B:
- Player fertig initialisiert
- Filmende
- uvm.

JMF: Media format



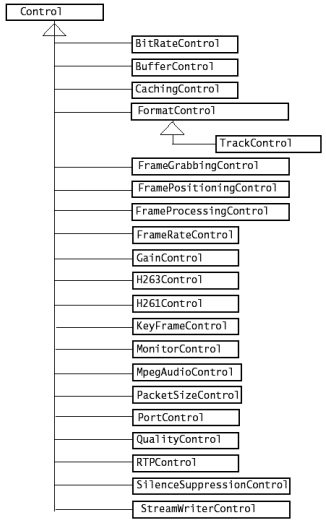
- kleiner Ausschnitt aus der "Format" Klassenhierarchie
- entsprechende Hierarchie für die Audio-/ weitere Datentypen

JMF: Data model

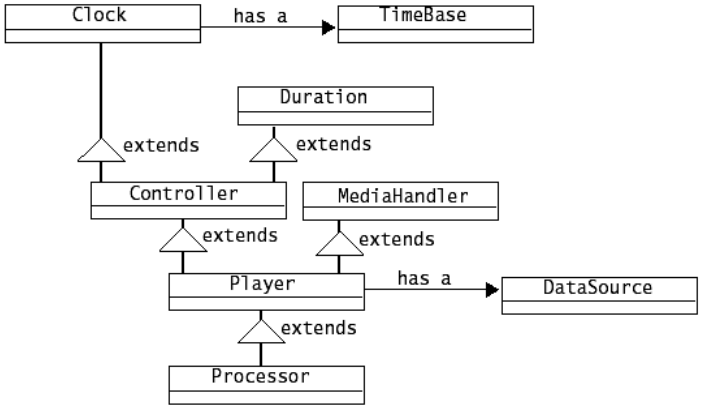


- Datenquellen sowohl als push- oder pull-Typen

JMF: Control

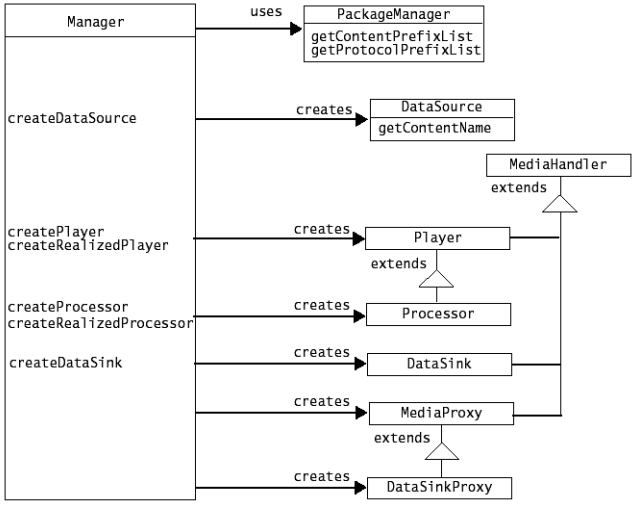


JMF: Controller



- Player: nur Medien-Wiedergabe (aus einer DataSource)
- Processor: Daten-Manipulation (z.B. Filterung)

JMF: Manager

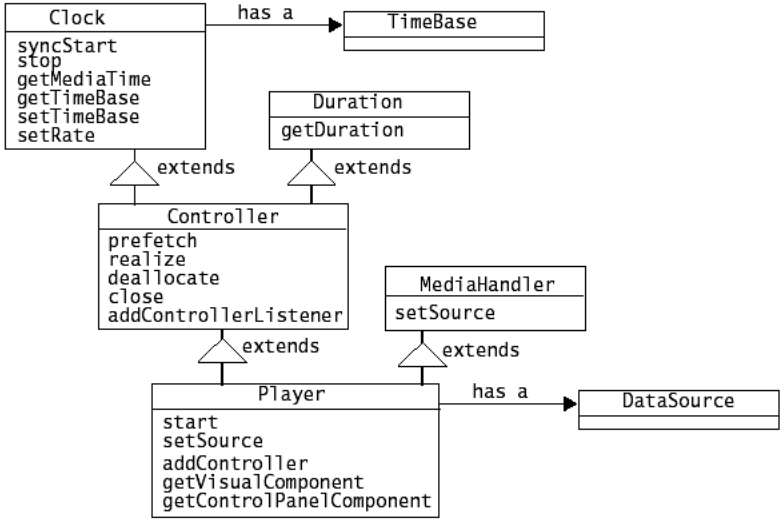


JMF: Managers

JMF uses four managers:

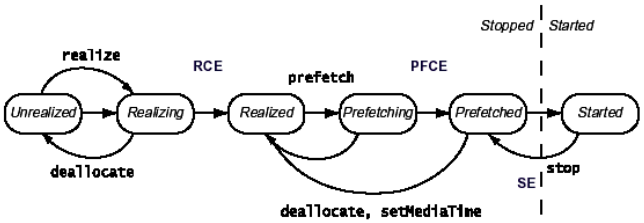
- `Manager`—handles the construction of `Players`, `Processors`, `DataSources`, and `DataSinks`. This level of indirection allows new implementations to be integrated seamlessly with JMF. From the client perspective, these objects are always created the same way whether the requested object is constructed from a default implementation or a custom one.
- `PackageManager`—maintains a registry of packages that contain JMF classes, such as custom `Players`, `Processors`, `DataSources`, and `DataSinks`.
- `CaptureDeviceManager`—maintains a registry of available capture devices.
- `PlugInManager`—maintains a registry of available JMF plug-in processing components, such as `Multiplexers`, `Demultiplexers`, `Codecs`, `Effects`, and `Renderers`.

JMF: Player class diagram



JMF: Player states

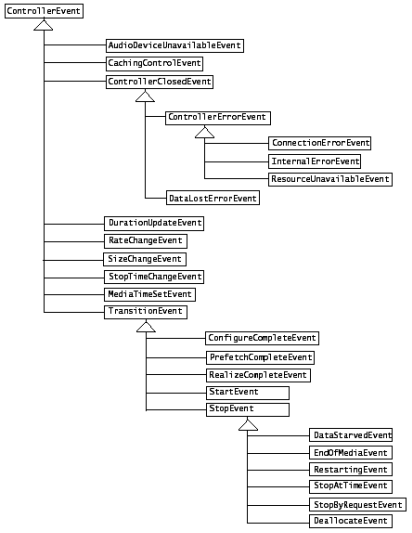
A `Player` can be in one of six states. The `Clock` interface defines the two primary states: *Stopped* and *Started*. To facilitate resource management, `Controller` breaks the *Stopped* state down into five standby states: *Unrealized, Realizing, Realized, Prefetching*, and *Prefetched*.



Transition Events:

RCE	RealizeCompleteEvent
PFCE	PrefetchCompleteEvent
SE	StopEvent

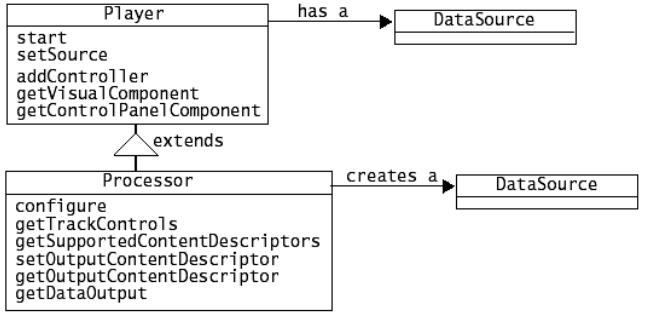
JMF: Controller event hierarchy



JMF: Player state transitions

Method	Unrealized Player	Realized Player	Prefetched Player	Started Player
addController	NotRealizedError	legal	legal	ClockStartedError
deallocate	legal	legal	legal	ClockStartedError
getControlPanelComponent	NotRealizedError	legal	legal	legal
getGainControl	NotRealizedError	legal	legal	legal
getStartLatency	NotRealizedError	legal	legal	legal
getTimeBase	NotRealizedError	legal	legal	legal
getVisualComponent	NotRealizedError	legal	legal	legal
mapToTimeBase	ClockStoppedException	ClockStoppedException	ClockStoppedException	legal
removeController	NotRealizedError	legal	legal	ClockStartedError
setMediaTime	NotRealizedError	legal	legal	legal
setRate	NotRealizedError	legal	legal	legal
setStopTime	NotRealizedError	legal	legal	StopTimeSetError if previously set
setTimeBase	NotRealizedError	legal	legal	ClockStartedError
syncStart	NotPrefetchedError	NotPrefetchedError	legal	ClockStartedError

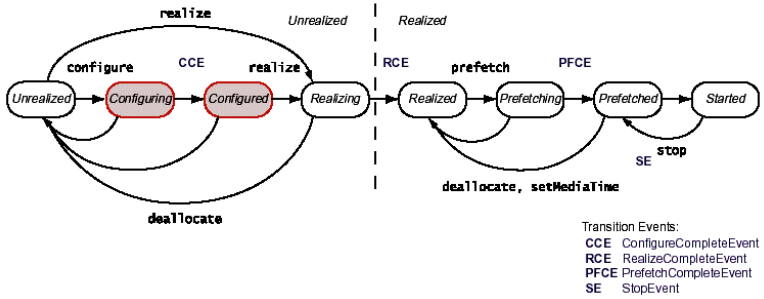
JMF: Processor



- Player liest Mediendaten aus seiner DataSource
- Processor liest und schreibt (manipulierte) Daten
- Basisklasse für alle Filter und Codecs

(Zitat)

JMF: Processor states



- "unrealized" / "realized": "noch nicht fertig" / "benutzbar"
- "prefetching" Füllen der internen Puffer
- "start" / "stop" Abspielen / Stopp
- "setMediaTime" Suche / Navigation im MediaStream

JMF: Lightweight Player in Swing

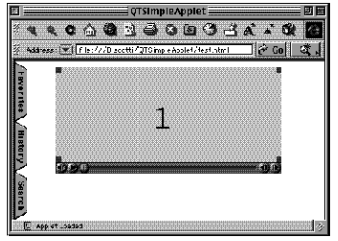
```
import javax.media.*;
...
public class MDIApp extends Frame {
  JMFrame jmframe = null;
  JDesktopPane desktop;
  Player player = null;

  public MDIApp() {
    super("Java Media Player");
    setLayout( new BorderLayout() );
    desktop = new JDesktopPane();
    desktop.setDoubleBuffered(true);
    add("Center", desktop);
    setMenuBar(createMenuBar());
    setSize(640, 480);
    setVisible(true);

    UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
    Manager.setHint(Manager.LIGHTWEIGHT_RENDERER, new Boolean(true));
    ...
    player = Manager.createPlayer(url);
  }


  public void controllerUpdate(ControllerEvent ce) {
    ...
    if (ce instanceof PrefetchCompleteEvent) {
      desktop.add( player.getVisualComponent() );
    }
    ...
  }
}
```


JMF: FrameSeek

```
import javax.media.control.FramePositioningControl;

class Seek extends Frame implements ControllerListener, ActionListener {
  ... // GUI stuff
  Player p; FramePositioningControl fpc;

  // given a DataSource, create a player and use that player for playback
  public boolean open(DataSource ds) throws Exception {
    p = Manager.createPlayer(ds);
    p.addControllerListener(this);
    p.realize();

    fpc = (FramePositioningControl)p.getControl(
            "javax.media.control.FramePositioningControl");
    totalFrames = fpc.mapTimeToFrame(duration);
    System.err.println("Total # of video frames: " + totalFrames);
    ...
    int currentFrame = fpc.mapTimeToFrame(p.getMediaTime());
    int randomFrame = (int)(totalFrames*Math.random());
    randomFrame = fpc.seek( randomFrame );
  }

  public static void main(String [] args) throws Exception {
    MediaLocator ml = new MediaLocator( argv[0] );
    DataSource   ds = Manager.createDataSource( ml );
    Seek       seek = new Seek();
    if (!seek.open(ds)) System.exit(0);
  }
}
```


Quicktime: Java API

```
public class QTSimpleApplet extends Applet {

private Drawable myQTContent;
private QTCanvas myQTCanvas;

public void init () {
  try {
    QTSession.open();
    setLayout (new BorderLayout());
    myQTCanvas = new QTCanvas (QTCanvas.kInitialSize, 0.5F, 0.5F);
    add (myQTCanvas, "Center");

    QTFile file = new QTFile (getCodeBase().getFile() +
    getParameter("file"));
    myQTContent = QTFactory.makeDrawable (file);
  } catch (Exception qtE) { ... }
}

public void start () {
  try {
    if (myQTCanvas != null)
      myQTCanvas.setClient (myQTContent, true);
  } catch (QTException e) { ... }
}

...
```



- "arguably simpler than C code"
- "C code differs for Mac and Windows"