Part VI MPEG-4

- Principles
- Architecture
- Systems
- Video
- Audio

Motivation

motivation

a blurring of borders between three distinct service models: communications, interactivity and broadcasting;

objective

- to standardize algorithms for audiovisual coding in MM applications,
- allowing for interactivity, high compression, scalability of video and audio content and
- support for natural and synthetic audio video content

MPEG-4 Overview

- formally ISO/IEC international standard 14496
- defines a *multimedia system* for interoperable communication of complex scenes containing audio, video, synthetic audio, and graphics material.
- combines some typical features of other MPEG standards,
- but aims to provide a set of technologies to satisfy the needs of *authors*, *service providers,* and *end users*.

MPEG-4 Overview

- For authors, MPEG-4 enables the production of content with greater *reusability and flexibility*; also, it permits better management and protection of content owner rights.
- For network service providers, MPEG-4 offers transparent information, interpreted and translated into the appropriate native signaling messages of each network.
- For end users, MPEG-4 enables many functionalities potentially accessible on a single compact terminal and higher levels of interaction with content, within the limits set by the author.

MPEG-4 Principles

- *Audio-visual scenes* made of audio-visual objects composed together according to a scene description:
  - allows interaction with elements within the audio-visual scene,
  - coding scheme can differ for individual objects,
  - allows easy reuse of audio-visual content.

MPEG-4 Principles

- Audio-visual objects can be of different nature:
  - audio (single or multi-channel) or video (arbitrary shape or rectangular),
  - natural (natural audio or video) or synthetic (text & graphics, animated faces, synthetic music),
  - 2D (Web like pages) or 3D (spatialized sound, 3D virtual world),
  - streamed (video movie) or downloaded (audio jingle).

MPEG-4 Principles

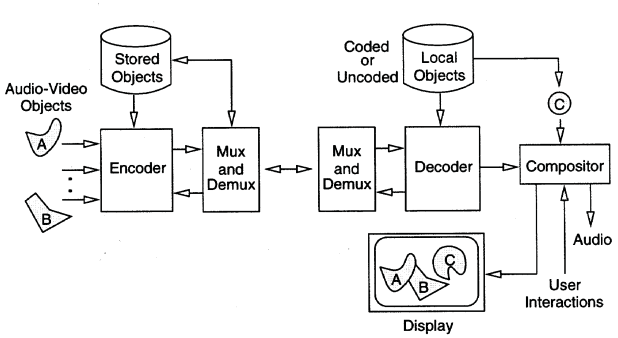
- Scene Description provides:
  - the spatial/temporal relationship between the audiovisual objects (2D, 3D, mixed 2D and 3D scene description),
  - the behavior and interactivity of the audio-visual objects and scenes,
  - protocols to modify and animate the scene in time
- These principles are independent of the bit rate.
- All this information is delivered in a compressed format.

MPEG-4 Principles

MPEG-4 provides

- Coding—representing units of audio, visual, or audiovisual content, called media objects.
- Composition—describing the composition of these objects to create compound media objects that form audiovisual scenes.
- Multiplex—multiplexing and synchronizing the data associated with media objects for transport over network channels providing appropriate QoS.
- Interaction—interacting with the audiovisual scene at the receiver's end or, via a back channel, at the transmitter's end.

MPEG-4 Architecture



MPEG-4 Standard Structure

- Systems
- Visual
- Audio
- Conformance Testing
- Reference Software
- Delivery Multimedia Integration Framework (DMIF).

MPEG-4 Systems

- The Systems subgroup defined the framework for integrating the natural and synthetic components of complex multimedia scenes.
- The Systems level integrates the elementary decoders for media components specified by other MPEG-4 subgroups
- provides the specification for the parts of the system related to composition and multiplex.
  - Composition information consists of the representation of the hierarchical structure of the scene.
  - composition of elementary media objects inspired by the existing Virtual Reality Modeling Language (VRML)

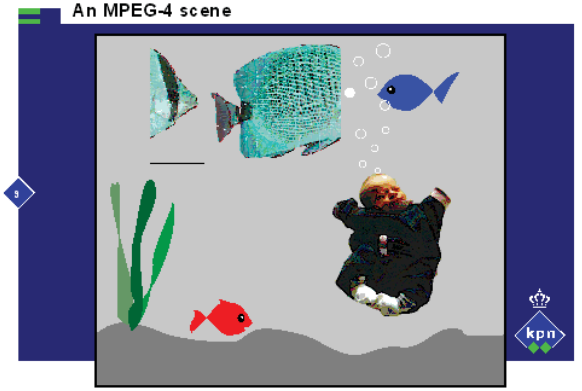
MPEG-4 Systems—Composition

- model to describe the composition of a complex multimedia scene relies on the concepts VRML
- main areas featuring new concepts
  - dealing with 2D-only content, for a simplified scenario where 3D graphics is not required;
  - interfacing with streaming media (video, audio, streaming text, streaming parameters for synthetic objects); and
  - adding synchronization capabilities.
- authors can generate this description in textual format, possibly through an authoring tool.

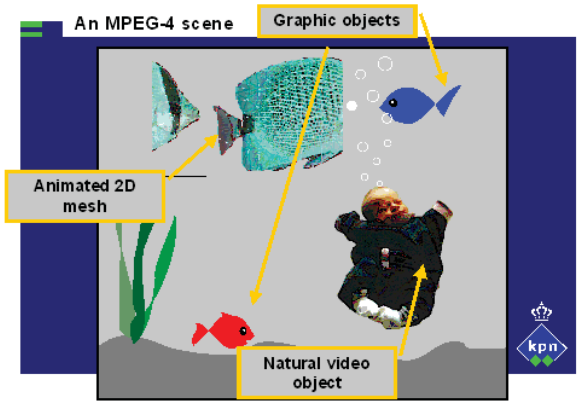
MPEG-4 Systems—Composition

- For efficiency, the standard defines a way to encode the scene description in a binary representation—Binary Format for Scene Description (BIFS).
- Multimedia scenes are conceived as hierarchical structures represented as a graph. Each leaf of the graph represents a media object.
- The initial snapshot of the scene is sent or retrieved on a dedicated stream.
- All the nodes and graph leaves that require streaming support to retrieve media contents are logically connected to the decoding pipelines.

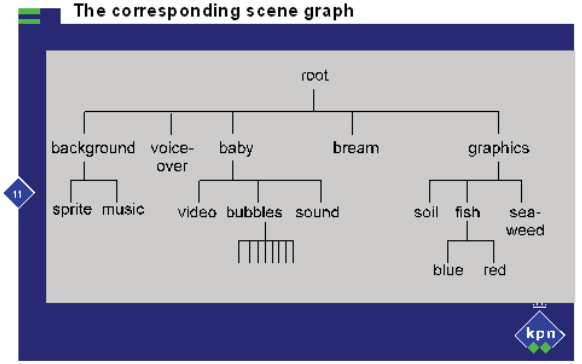
Example Scene



Example Scene



Scene Graph



Spatial Composition

- The *composition stream* is treated differently from others because it provides the information required by the terminal to set up the scene structure and map all other elementary streams to the respective media objects.
- Spatial relationships—Each elementary media object is represented by a leaf in the scene graph and has its own local coordinate system. The mechanism to combine the scene graph's nodes into a single global coordinate system uses spatial transformations associated to the intermediate nodes, which group their children together.

Temporal Composition

- The composition stream (BIFS) has its own time base.
- Even if the time bases for the composition and for the elementary data streams differ, they must be consistent except for translation and scaling of the time axis.
- Time stamps attached to the elementary media streams specify at what time the access unit for a media object should be ready at the decoder input (*DTS, decoding time stamp*), and at what time the composition unit should be ready at the compositor input (*CTS, composition time stamp*).
- Time stamps associated to the composition stream specify at what time the access units for composition must be ready at the input of the composition information decoder.

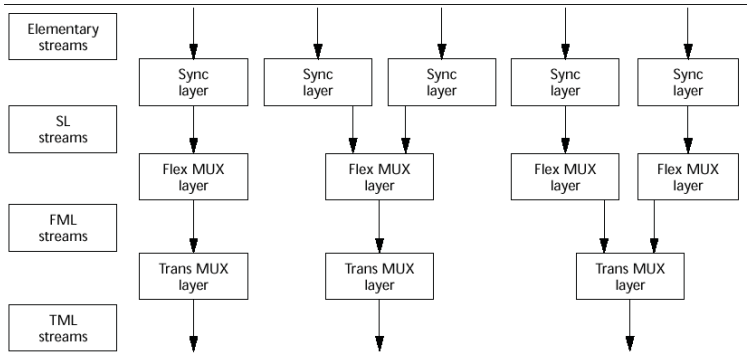
Temporal Composition

- In addition to the time stamps mechanism (derived from MPEG-1 and MPEG-2), fields within the scene description also carry a time value. They indicate either a *duration in time* or an *instant in time*.
  - represent a relative time with respect to the time stamp of the BIFS elementary stream.
  - For example, the start of a video clip represents the relative offset between the composition time stamp of the scene and the start of the video display.

Multiplex

- Because MPEG-4 is intended for use on a wide variety of networks with widely varying performance characteristics, it includes a three-layer multiplex separating the functionality of
  - adding MPEG-4-specific information for timing and synchronization of the coded media (synchronization layer);
  - multiplexing streams with very different characteristics, such as average bit rate and size of access units (flexible multiplex layer); and
  - adapting the multiplexed stream to the particular network characteristics in order to facilitate the interface to different network environments (transport multiplex layer).

Multiplex Structure



Multiplex Structure

- Elementary streams are packetized, adding headers with timing information (clock references) and synchronization data. They make up the *synchronization layer* (SL) of the multiplex.
- Streams with similar QoS requirements are then multiplexed on a content multiplex layer, termed the *flexible multiplex layer* (FML). It interleaves data from a variable number of variable bit-rate streams.
- A service multiplex layer, known as the transport *multiplex layer* (TML), can add a variety of levels of QoS and provide framing of its content and error detection

Synchronization Layer

- The header attached by sl contains fields specifying:
  - *Sequence number*, *Instantaneous bit rate*, *OCR (object clock reference), DTS (decoding time stamp), CTS (composition time stamp).*
- The information contained in the SL headers maintains the correct time base for the elementary decoders and for the receiver terminal, plus the correct synchronization in the presentation of the elementary media objects in the scene.
- The clock references mechanism supports timing of the system, and the mechanism of time stamps supports synchronization of the different media.

Flexible multiplex layer

- wide range of possible bit rates associated to the elementary streams—ranging from 1 Kbps to more than 100 Mbps
  - intermediate multiplex layer provides more flexibility
- The intermediate (optional) flexible multiplex layer provides a way to group together several low-bit-rate streams for which the overhead associated to a further level of packetization is not necessary or introduces too much redundancy
- With conventional scenes, like the usual audio plus video, this optional multiplex layer can be skipped;

Transport multiplex layer

- The multiplex layer closest to the transport level depends on the specific transmission or storage system on which the coded information is delivered.
- The Systems part of MPEG-4 doesn't specify the way SL packets (when no FML is used) or FML packets are mapped on TML packets. The specification simply references several different transport packetization schemes.
- The "content" packets may be transported directly using e.g. an ATM Adaptation Layer 2 (AAL2) scheme, an MPEG-2 transport stream packetization or transport control TCP/IP.

MPEG-4 Video

- MPEG-4 Video supports:
  - *Content-based interactivity*
    - content-based multimedia data access tools,
    - content-based manipulation and bit-stream editing,
    - hybrid natural and synthetic data coding, and
    - improved temporal random access.
  - *Compression*.
    - improved coding efficiency and
    - coding of multiple concurrent data streams.
  - *Universal access*
    - robustness in error-prone environments and
    - content-based scalability.

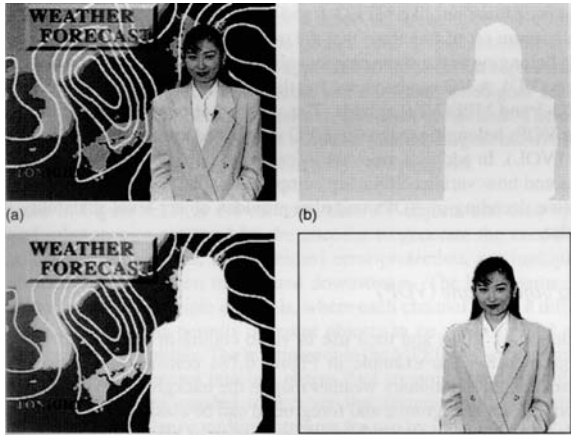
Video Codec Structure

- syntax allows coding of *rectangular* as well as *arbitrarily shaped video objects* in a scene and supports both nonscalable and scalable coding.
- The scalability syntax enables reconstructing useful video from pieces of a bit stream by structuring the total bit stream in two or more layers, starting from a *stand-alone base layer* and adding a number of *enhancement layers*.
- The ability to access individual objects requires achieving a coded representation of their shape. A natural video object consists of a sequence of 2D representations (VOPs).

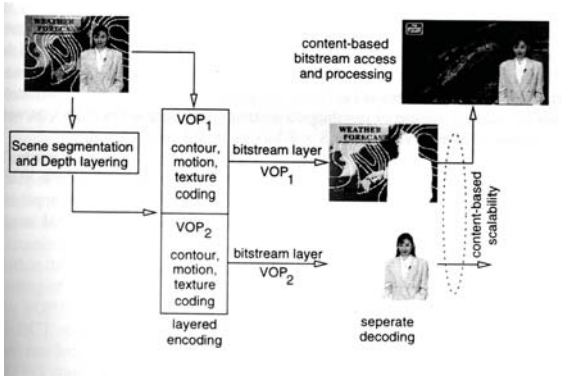
MPEG-4 Video Coding

- each video frame is segmented into a number of arbitrary shaped regions, called video object planes (VOP).
- successive VOPs belonging to the same physical object in a scene are referred to as video objects (VO).
- the shape, motion and texture information of the VOPs belonging to the same VO is encoded into a separate video object layer (VOL).
- relevant information needed to identify each of the VOLs and how various VOLs are composed is also encoded. This allows for selective decoding and provides object-level scalability.

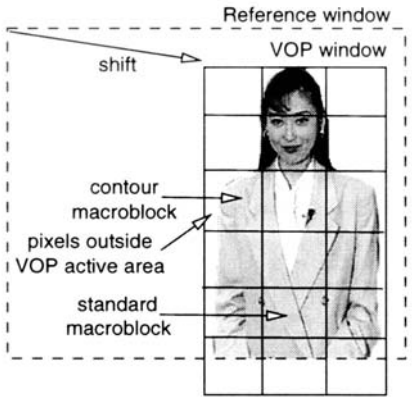
Video Object Plane (VOP)



VOP Coding Process



Macroblock Grid for MC



Interactivity

- important feature: approach doesn't explicitly define a temporal frame rate; encoder and decoder can function in different frame rates, which don't even need to stay constant
- Interactivity between user and encoder or decoder:
  - at the encoding level, e.g. in coding control to distribute the available bit rate between different video objects or to influence the multiplexing to change parameters such as the composition script at the encoder.
  - In case no back channel is available, or when the compressed bit stream already exists, the user may interact with the decoder by acting on the compositor to change either the position of a video object, its display depth order or by requesting the processing of a portion of the bit stream only.

Media Integration of Text and Graphics

- MITG provides a Layout node to specify the placement, spacing, alignment, scrolling, and wrapping of scene objects.
- *Still images* or *video objects* can be placed in a scene graph in many ways, and they can be texture-mapped on any 2D object.
- The most common way, though, is to use the Bitmap node to insert a rectangular area in the scene in which pixels coming from a video or still image can be copied.
- The 2D scene graphs can contain *audio sources* by means of the Sound2D nodes.
- *Text* can be inserted in a scene graph by the Text node. Text characteristics (font, size, style, spacing, and so on) can be customized by means of the Font Style node.

Media Integration of Text and Graphics

- *3D graphics*—BIFS 3D nodes—an extension of the ones defined in VRML—allow the creation of virtual worlds.
- Like in VRML, it's possible to add behavior to objects through Script nodes.
- MPEG-4 allows the creation of much more complex scenes than VRML, of 2D/3D hybrid worlds where contents are not downloaded once but can be streamed to update the scene continuously.

Face Animation

- Focuses on parameters for face animation and definition. It has a very tight relationship with hybrid scalable text-to-speech synthesis for speech-driven avatars.
- the face animation work is the first attempt to define in a standard way the sets of parameters for synthetic anthropomorphic models.
- Face animation is based on the development of two sets of parameters:
  - facial animation parameters (FAPs) and
  - facial definition parameters (FDPs).

Face Animation

- FAPs allow having a single set of parameters regardless of the face model used
- Most FAPs describe atomic movements of facial features; others (expressions and visemes) define much more complex deformations.
- Visemes define the position of the mouth (lips, jaw, tongue) associated with phonemes.
- In the context of MPEG-4, the expressions mimic the facial expressions associated with human primary emotions like joy, anger, fear, surprise, sadness, and disgust.
- Animated avatars' animation streams fit very low bit-rate channels (about 4 Kbps).

Face Animation / Texture Coding

- FAPs can be encoded either with arithmetic encoding or with discrete cosine transform (DCT).
- FDPs are used to calibrate (that is, modify or adapt the shape of) the receiver terminal default face models or to transmit completely new face model geometry and texture.
- Texture coding—MPEG-4 supports an ad-hoc tool for encoding textures and still images based on a wavelet algorithm that provides spatial and quality scalability, content-based (arbitrarily shaped) object coding, and very efficient data compression over a large range of bit rates.
- Texture scalability comes through many (up to 11) different levels of spatial resolutions, allowing progressive texture transmission and many alternative resolutions

MPEG-4 Text-to-Speech

- MPEG-4 doesn't define a specific text-to-speech technique but rather the binary representation of a TTS stream and the interfaces of an MPEG-4 text-to-speech (M-TTS) with the other parts of an MPEG-4 decoder.
- An M-TTS stream may contain many different information types about the synthetic voice apart from text: gender, age, speech rate, language code, prosody, and lip shape information.
- It may contain fields that allow trick mode (fast forwarding, pausing, playing, or rewinding the synthetic speech).
- An M-TTS can also carry the International Phonetic Alphabet (IPA) coded phonemes with their time duration.

MPEG-4 Text-to-Speech

- Handed to the face animation engine in the MPEG-4 player, the coded phonemes can produce speech-driven face animation.
- In this case the face animation system doesn't receive a FAP stream from the MPEG-4 demultiplexer; instead it converts phonemes into visemes and uses them to perform the face model deformations.
- The phoneme duration synchronizes model animation and speech.
- Interestingly, applications require a tiny channel bandwidth—from 200 bps to 1.2 Kbps.

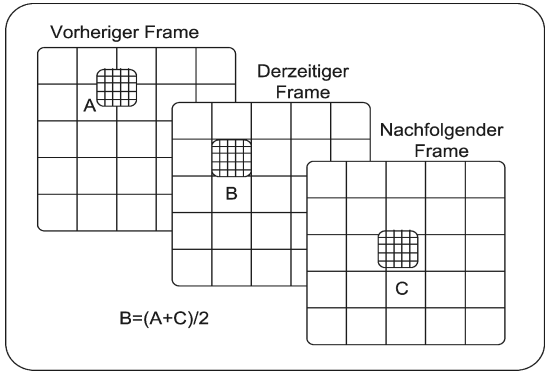
Allgemeine MPEG-4 Video Konzepte

- 4 Modi für B-Framekodierung
- Adaptive Quantisierung
- Bewegungsvektoren mit bis zu Viertel-Pixel Auflösung
- Globaler Bewegungausgleich
- Wahl der Quantisierungsmethode
- Variable Blockgröße
  - 1 MV Mode – 1 Vektor/16x16 Block (MPEG-2)
  - 4 MV Mode – 4 Bewegungsvektoren/16x16 Block, entspricht 1 Vektor/8x8 DCT-Block

MPEG B-Frames

- MPEG-2 bietet 3 Möglichkeiten der B-Frame Kodierung
  - Vorwärts Modus – Referenzframe ist vorhergegangener I- oder P-Frame, kodiert wird ein Vektor + Fehlerbild (Fehlerbild = Prädiktionsfehler)
  - Rückwärts Modus – Referenzframe ist nachfolgender I- oder P-Frame, kodiert wird ein Vektor + Fehlerbild
  - Interpolationsmodus – Beide Vektoren (aus Rückwärts- und Vorwärtsprädiktion) werden übertragen, übertragendes Fehlerbild resultiert aus Interpolation beider Referenzwerte

MPEG-2 B-Frame Interpolationsmodus



MPEG-4 B-Frame: Direkter Modus

- MPEG-4 bietet zusätzlich vierten Modus: „Direkter Modus“
- In die Kodierung werden wie im Interpolationsmodus sowohl Rückwärts- wie auch Vorwärtsprädiktionswerte mit einbezogen
- Die beiden Vektoren können allerdings von einem einzigen Vektor abgeleitet werden:
  - Bewegungsvektor im nachfolgendem Frame, gleiche Position wie Makroblock im betrachteten B-Frame
  - Vektor wird linear skaliert, je nach zeitlichen Abständen der 3 Frames (vorh. Frame – B-Frame – nachfolg. Frame)
  - Kodiert wird nur ein Delta Vektor (entsp. Fehlerkorrektur)

Adaptive Quantisierung

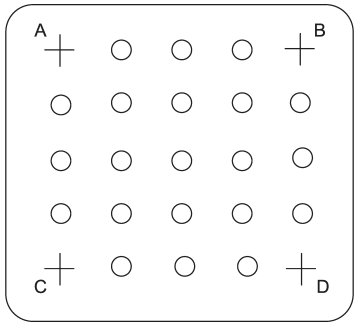
- Adaptive Quantisierung (MPEG-4) – Jedem Makroblock wird unter Berücksichtigung psychovisueller Aspekte individuelle Quantisierung zugeordnet
- Einfaches Beispiel (bei Xvid angewandt)
  - Artefakte an sehr hellen/dunklen Bildstellen für menschliches Auge weniger sichbar
  - Entsprechende Blöcke werden stärker quantisiert
- Adapitive Quantisierung heisst bei DivX „Psychovisuelle Verbesserung“

Viertel-Pixel Bewegungskompensation

- Bewegungskompensation mittels Blockmatching
  - Encoder sucht Makroblock im Referenzbild
  - Kodiert wird Bewegungsvektor (x-, y-Koordinaten) und das Fehler-Bild (Prädiktionsfehler)
  - Je genauer Bewegungsvektor, desto kleiner der Fehler
- Bewegungsvektor mit „Ein-Pixel“- Genauigkeit
  - Nur ganzzahlige Vektorkomponenten möglich
  - Reale Bewegung nur unzureichend beschrieben
  - Hoher Prädiktionsfehler, schlechte Kompression
  - Verwendet in H.261

Viertel-Pixel Bewegungskompensation

- (Virtuelle) Viertel-Pixel Werte mittels Interpolation berechnet
- Bewegungsvektoren werden im Viertel-Pixel Netz ermittelt



Viertel-Pixel Bewegungskompensation

- Bewegungsvektor mit „Halb-Pixel“ - Genauigkeit
  - Vektorkomponenten sind ganze oder halbe Pixelwerte
  - Halbpixelwerte durch Interpolation errechnet
  - Kleiner Prädiktionsfehler
  - Verwendet in MPEG-1, MPEG-2 und H.263
- MPEG-4 bietet Möglichkeit der Viertel-Pixel Genauigkeit
  - Realitätsnahe Beschreibung von Bewegung möglich
  - Signifikante Qualitätssteigerung bei niederen Datenraten
  - Interpolationsalgorithmen komplex

Globaler Bewegungsausgleich

- Für ein VOP werden bis zu 4 globale Bewegungsvektoren berechnet und im Header kodiert
- Effektive Kompression bei Kamerabewegungen
  - Geradliniger Schwenk – Ein Globaler Bewegungsvektor
  - Allgemeine Bewegung – Globaler Bewegungsausgleich mittels 4 Bewegungsvektoren (an jeder Ecke einer)
- Für jeden Makroblock wird zusätzlich Bewegungsvektor berechnet
- Encoder entscheidet, für welchen Makroblock welcher Bewegungsvektor (oder -kombination) kodiert wird

Alternative Scan Modi

- Alternativer horizontaler Scan

0	1	2	3	10	11	12	13
4	5	8	9	17	16	15	14
6	7	19	18	26	27	28	29
20	21	24	25	30	31	32	33
22	23	34	35	42	43	44	45
36	37	40	41	46	47	48	49
38	39	50	51	56	57	58	59
52	53	54	55	50	61	62	63

Alternative Scan Modi

- Alternativer vertikaler Scan

0	4	6	20	22	36	38	52
1	5	7	21	23	37	39	53
2	8	19	24	34	40	50	54
3	9	18	25	35	41	51	55
10	17	26	30	42	46	56	60
11	16	27	31	43	47	57	61
12	15	28	32	44	48	58	62
13	14	29	33	45	49	59	63

Wahl der Quantisierungsmethode

- MPEG-4 bietet User Möglichkeit, Quantisierungsmethode selbst zu wählen
  - Wahl zwischen H.263, MPEG oder Erstellen eigener MPEG-Quantisierungstabellen (Inter- und Intramatrix)

Part VII
Multimedia Programming Abstractions

- Motivation
- Media Modelling Abstractions

MM Programming Environments

framework selection:

- what programming techniques, methodologies, abstractions, etc. are needed
- what basic “services” are provided
- which media and standards are supported

Why Object-Oriented Multimedia Programming?

- encapsulation
- no software legacy (new area)
- need for extensibility (specialization can accommodate new hardware, media formats, standards, compression techniques, etc.)
- need for cross-platform development (abstract interfaces reduce platform dependence)

Framework Development

- identify and define *abstractions* that cover metaphors and mechanisms found in audio and video production
- separate generic parts of a solution and structure them as *collaborating objects*
- provide a general *architecture* of how objects cooperate in a well-defined way (framework)
- *implement classes* that are necessary for the specific purpose and the specific hw platform

Abstractions

- time-object
- timed streams / media elements (“samples”)
- interpretation
- derivation
- composition (often called “synchronization”)
- media processing elements (MPEs, “components”)
- configuration
- quality of service (QoS)

Time-Object

- umbrella for media processing elements and time-related media
- should be system supported yet user extensible and distribution proof
- should support high-level semantics
- should support QoS and synchronization

Time-Object Definitions

- *continuous time value*
  A continuous time value is a measurement of time using some agreed upon units.

- *discrete time coordinate system* (*DTCS*)
  A DTCS, D, is a mapping from discrete time values to continuous time.
  $D_f : i \rightarrow (1/f)i$, where *f* is called the *frequency* of the time system.
  Examples: D_{30} (video), D_{25} (video), D_{24} (film), and D_{44100} (CD audio).

Timed Streams

- streams and stream interfaces are widely used abstractions proposed in different contexts: communications protocols, operating system support and application design
- communication channel with real-time guarantees and continuous transmission of media elements (samples) without application intervention
- should be multicast and cover both asynchronous and isochronous transmission

Timed Streams

media type and media element

definition *media type*:

A *media type* specifies the encoding of data for some medium. Examples of media types include encodings for audio, video and image data.

definition *media element*:

A *media element* is an autonomous unit of data encoded by some media type. Each element has a *size* (length in bytes) and a *descriptor*, which identifies the media-specific and representation-specific parameters.

Timed Streams

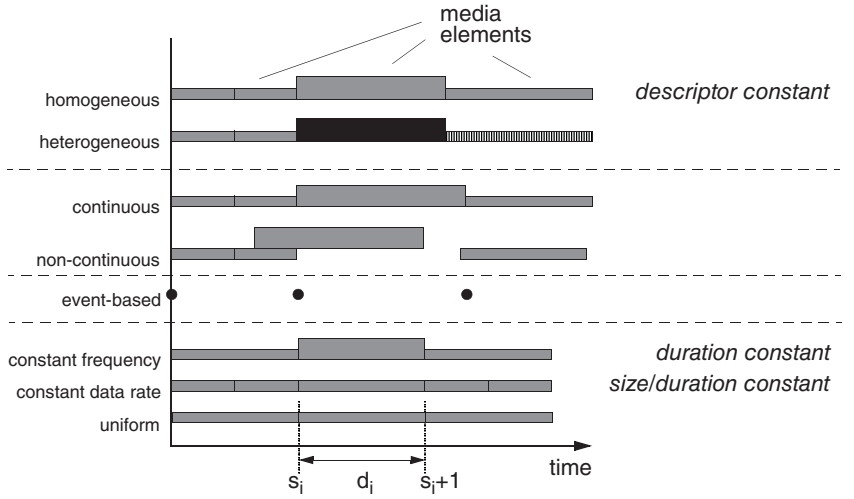
definition *timed stream* (*media object*):

A *timed stream* is a finite sequence of tuples of the form: $<e_i, s_i, d_i>$, $i=1, \ldots n$. Each media sequence is based on a media type *M* and a discrete time coordinate system *D.*

definition *timed-stream type*:

A *timed-stream type* is a tuple *T*= <*M, D, R*> where *M* is a media type, *D* a discrete time coordinate system type and *R* is a set of rules that can be considered as integrity constraints for media sequences over *M* and *D*.

Timed Streams - Examples



Timed Streams - Properties

- homogeneity: element descriptors are constant
- continuity: $s_{i+1} = s_i + d_i$, for $i = 1, \ldots n-1$
- frequency: d_i = constant, for $i = 1, \ldots n-1$
- data rate: element size/duration = constant
- uniformity: element size and duration are constant

The Dual Nature of Time-Based Media

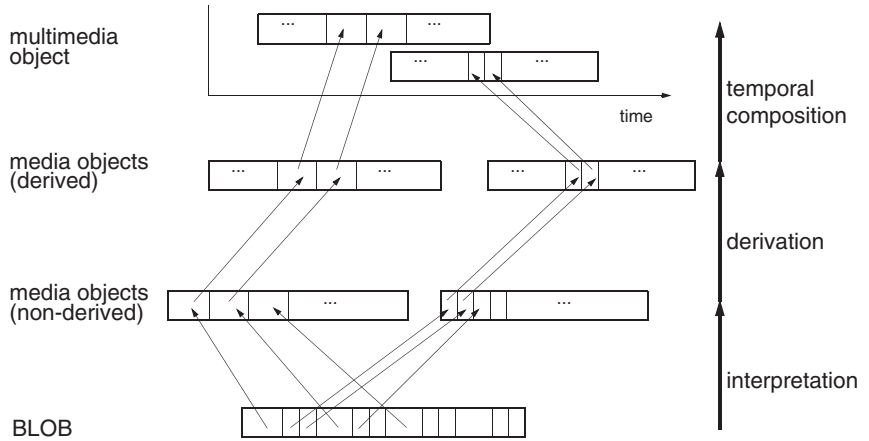
Time-based media can be viewed as:

- *pools,* for media-independent operations, e.g. copying
- *streams,* for media-dependent operations, e.g. editing

Necessary for stream “view”:

- timing information for *inter*- and *intra*-stream synchronization
- structuring information of timed streams

Interpretation—Derivation—Composition



Interpretation

Definition: interpretation

An interpretation of a stream (BLOB) *S*, is a mapping from *S* to a set of media elements.

complicating factors

- heterogeneity
  - variable-sized elements
  - several encoding methods, parameters
- out-of-order elements
- interleaving and padding
- non-destructive editing and scalability

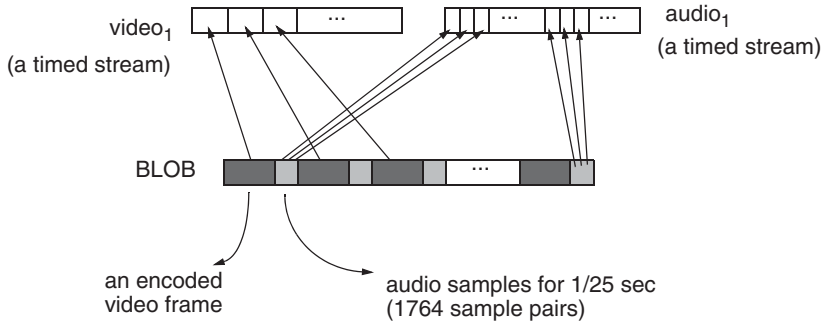
Interpretation

Reasons for Interpretation:

- structural information must not be separated from data
- permits sophisticated queries
- presenting time-based data requires timing information

Interpretation

Example 1



Interpretation

interpretation descriptor example

video$_1$ descriptor = {
category = homogeneous, constant frequency
quality factor= “VHS quality”
duration = 10 minutes
frame rate = 25
frame width = 640
frame height = 480
frame depth = 24
color model = RGB
encoding = YUV 8:2:2, JPEG }

audio$_1$ descriptor = {
category = homogeneous, uniform
quality factor = “CD quality”
duration = 10 minutes
sample rate = 44100
sample size =16
number of channels = 2
encoding = PCM }

Interpretation

mapping from element number to BLOB placement

- tables for example 1
  - video$_1$(elementNumber, elementSize, blobPlacement)
  - audio$_1$(elementNumber, blobPlacement)
- table for a heterogeneous video stream
  - video$_1$(elementNumber, startTime, duration, elementDescriptor, elementSize, blobPlacement)

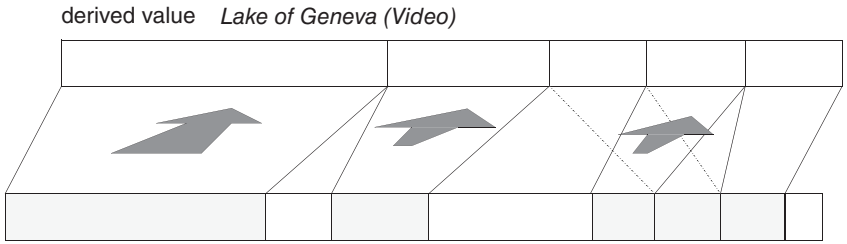
Derivation

definition: derivation, derived object, derivation object

- The *derivation* (D) of a media object o_1 from a set of media objects O is a mapping of the form $D(O, P_D) \rightarrow o_1$, where P_D is the set of parameters specific to D.
- o_1 is called the *derived object*.
- *derivation object*:
  - references to the media objects
  - and parameter values used.

Derivation

translation concatenation extraction conversion ...



Derivation

Advantages introduced by derivation:

- ❑ reduces storage and networking costs.
- ❑ allows modification operations to be performed more efficiently.
- ❑ provides data independence.

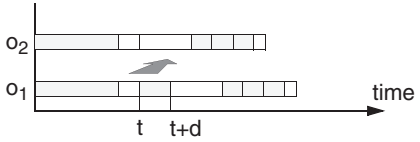
Derivation

examples of derivations

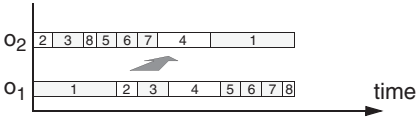
Derivation Name	Argument Type(s)	Result Type	Category
FilterVideo	Video	Video	change of content
FilterImage	Image	Image	change of content
FadeClip	TimeBasedMedia	TimeBasedMedia	change of content
ExtractClip	TimeBasedMedia	TimeBasedMedia	change of timing
CutClip	TimeBasedMedia	TimeBasedMedia	change of timing
InsertClip	TimeBasedMedia	TimeBasedMedia	compound
SynthesizeVideo	Animation	Video	change of type
Synthe-sizeCDAudio	MIDI	CDAudio	change of type

Elementary Derivation - Examples

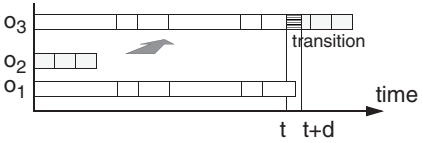
CutClip(o_1, t, d) → o_2



PermuteSamples(o_1, f) → o_2

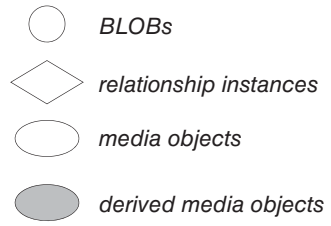


Transition(o_1, o_2, t, d, f) → o_3



Derivation - Data Diagram

Graphical Representation of:

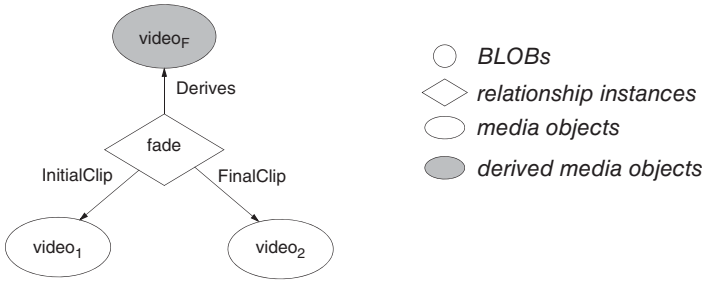


Purpose:

- ❑ visualization of interpretation and derivation
- ❑ optimization of graphs according to derivation properties

Derivation

data diagram example



fade = new FadeClip(initialClip: video$_1$, startInInitialClip: 1:00,
finalClip: video$_2$, startInFinalClip: 0:0, duration: 0:10)
video$_F$ = fade.Derive()

Composition

- spatiotemporal arrangement of presentation
- currently mainly an authoring issue
- should be intuitive, entity conserving, distribution proof
- should be suited for translation into a playout schedule
- should be suited for unlimited and unknown durations with flexible degree of synchrony.

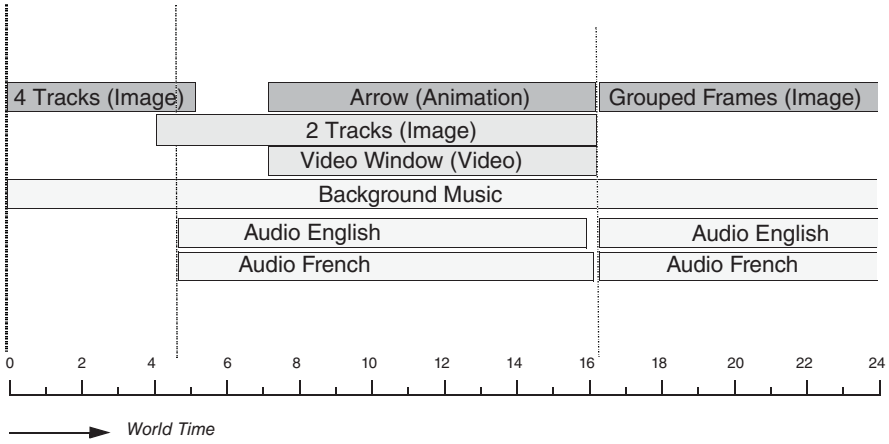
Composition

definition: composition

Composition is the specification of temporal and/or spatial relationships between a group of media objects.

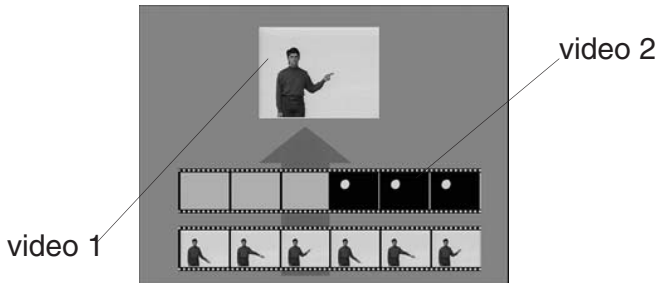
The result of composition is called a *multimedia object*, the original elements are called its *components*.

Temporal Composition - Timeline Diagram



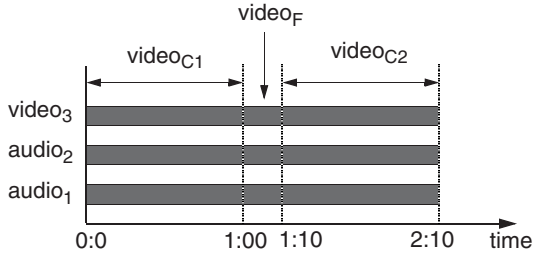
Spatial Composition

- specification of layout for visual data types
  - presentation region (window)
  - clipping and composition operations
- spatial transformations (scaling etc.)

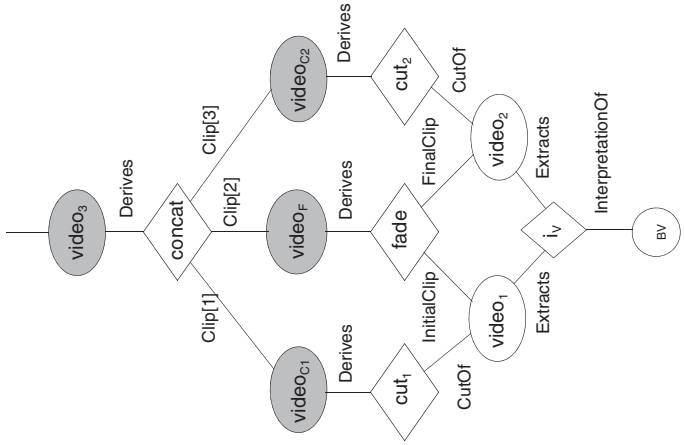


Example Composition

m = new Multimedia
c_1 = new TemporalComp(component: $audio_1$, start: 0:0, stop: 2:10)
c_2 = new TemporalComp(component: $audio_2$, start: 0:0, stop: 2:10)
c_3 = new TemporalComp(component: $video_3$, start: 0:0, stop: 2:10)
m.Add(composite: c_1)
m.Add(composite: c_2)
m.Add(composite: c_3)



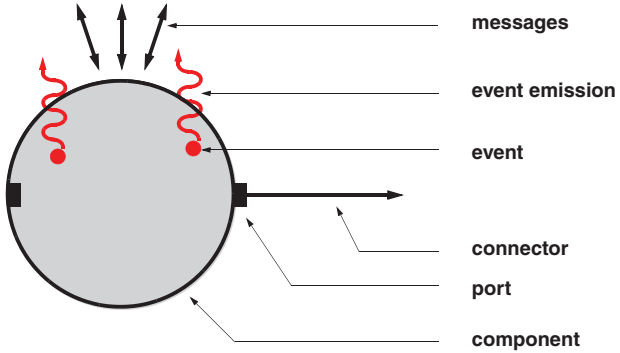
Derivation/Composition Diagram



Components (MPEs)

- produce | consume | transform ***timed media streams***
- streams enter and leave through ***ports***
- connected ports must be ***plug compatible***
- three interfaces:
  - synchronous
  - asynchronous
  - isochronous

Component Interfaces



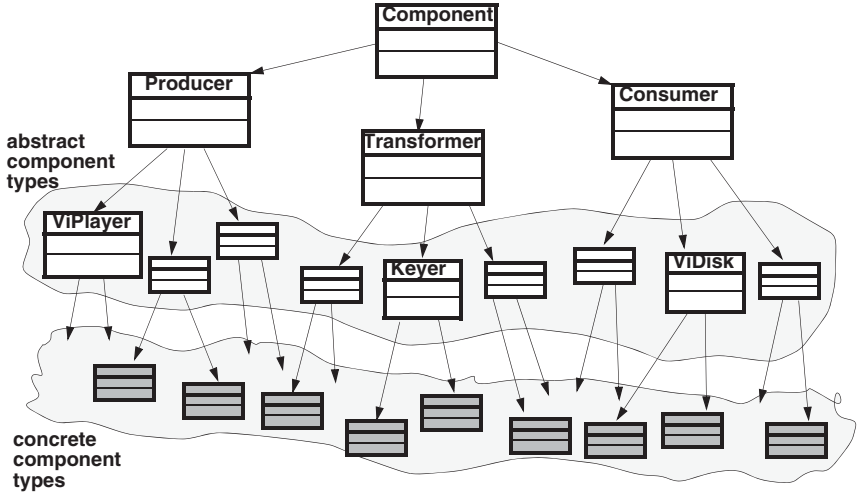
Components - Examples

- a video camera — produces video
- a video codec — transforms video
- a 3D renderer — consumes geometry, produces video
- a MIDI keyboard — produces MIDI
- an audio recorder — consumes audio
- a dataglove — produces tracking data

Component Classes—Port Connection

- Two ports can be connected provided:
  - one is an output port and the other an input port
  - the two ports are *plug compatible*
  - adding the connection does not exceed the *fan-limit* on either port

Component Classes



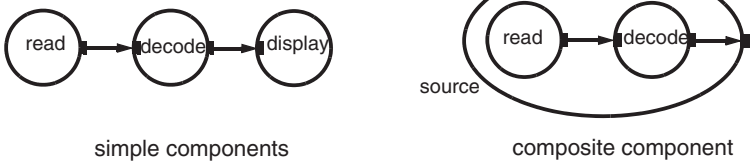
Application Example: A Virtual Studio

media object	input ports	output ports	implementation
3DRenderObj	TrackSeq VideoSeq VideoSeq (mask)	VideoSeq (back-ground) VideoSeq (key)	server on SGI Onyx
TrackObj		TrackSeq	server on SGI Onyx
KeyerObj	VideoSeq (foreground) VideoSeq (background) VideoSeq (key)	VideoSeq (mix) VideoSeq (key)	server on any SGI
VideoPlayerObj		VideoSeq	server on any SGI
CompObj	VideoSeq	MPEGSeq	server on any SGI
DecompObj	MPEGSeq	VideoSeq	server on any SGI

Configuration

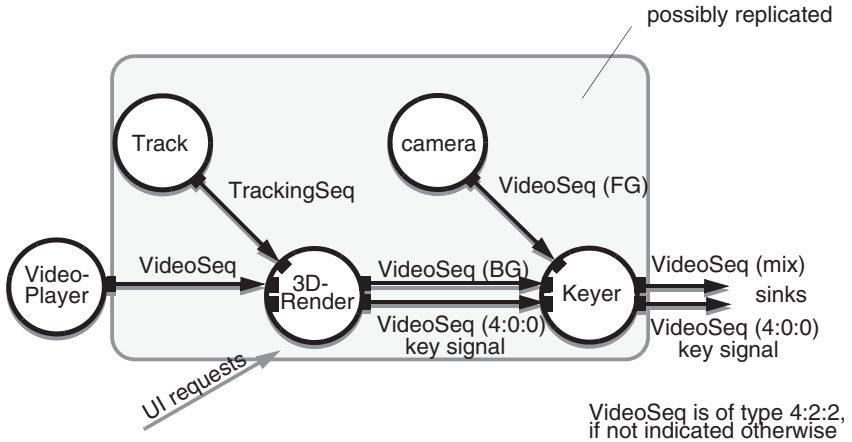
- composition of components (MPEs), streams and media into an application
- should support dynamically changing configurations with arbitrary numbers of components
- should support distribution, authoring and programming

Configuration - Example



simple components composite component

Virtual Studio Configuration



Quality of Service (QoS)

- term used to represent the application requirements for a given resource
- “fuzzy” or statistical description of parameters of service delivery
- reflects tolerance in human perception
- enables resource management and trade-offs between conflicting goals
- on the user level, it should be intuitive and controllable
- on all levels, it should be compatible, negotiable and accessible for system-wide optimization

QoS

- typical QoS parameters: minimum and maximum resolution, allowed error rate, acceptable jitter and delay bounds
- some application require hard or deterministic guarantees of service; strong guarantees can be given as long as certain factors as load, buffer space and schedule can be monitored and controlled.
- other applications will need only statistical guarantee of some range of performance (probability)