MPEG-4

By

M. Habibullah Pagarkar
&
Chinmay Shah

SE-IT
VESIT

TABLE OF CONTENTS

1 INTRODUCTION

MPEG-4 is an ISO/IEC (International Standardization Organization / International Electrotechnical Commission) standard developed by MPEG (Moving Picture Experts Group), the committee that also developed the Emmy Award winning standards known as MPEG-1 and MPEG-2. These standards made interactive video on CD-ROM and Digital Television possible. MPEG-4 is the result of another international effort involving hundreds of researchers and engineers from all over the world. MPEG-4 became an International Standard in the first months of 1999. The fully backward compatible extensions under the title of MPEG-4 Version 2 were frozen at the end of 1999, to acquire the formal International Standard Status early 2000. Some work, on extensions in specific domains, is still in progress.

In particular, MPEG defines the syntax of low bitrate video and audio bitstreams of synthetic and natural sources, descriptions of their structure and content, and the operation of conformant decoders of these bitstreams. The encoder algorithms are not defined by MPEG. This allows for continuous improvement of encoders and their adaptation to specific applications, within the bitstream syntax definition.

MPEG-4 builds on the proven success of three fields:

- Digital television
- Interactive graphics applications (synthetic content)
- Interactive multimedia (World Wide Web, distribution of and access to content)

MPEG-4 provides the standardized technological elements enabling the integration of the production, distribution and content access paradigms of the three fields. MPEG-4 is the next-generation standard in the coding of time-varying digital media consisting of audio-visual (A/V) objects suited for high-efficiency storage and communications.

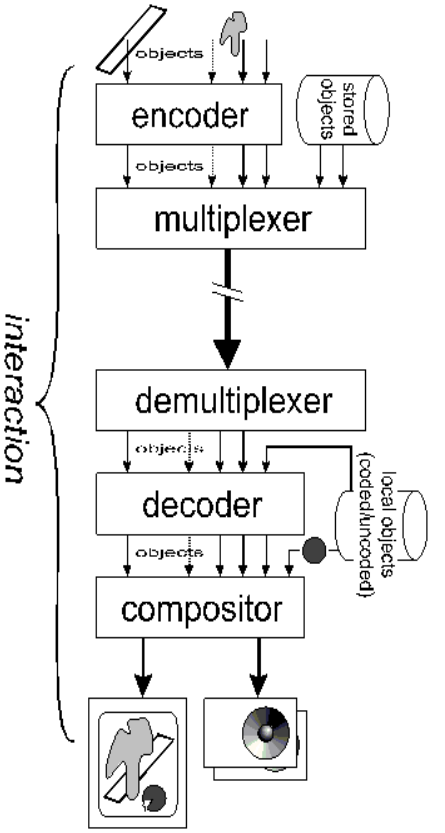


Figure 1-Schematic overview of an MPEG-4 System

The broad objectives addressed by MPEG-4 are high compression, support for service profiles designed for classes of applications, a wide range of channel bandwidths from a few Kbits/sec to many Mbits/sec, content-based access and manipulation, and scalability of A/V objects for different channels and terminal resources.

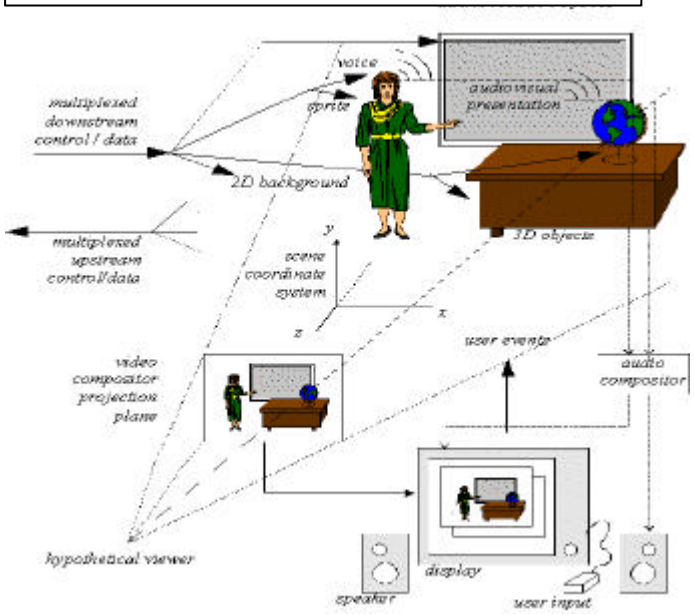
2 FEATURES OF THE MPEG-4 STANDARD

2.1 Coded Representation And Composition Of Objects

MPEG-4 audiovisual scenes are composed of several media objects, organized in a hierarchical fashion such as:

- still images (e.g. as a fixed background)
- video objects (e.g. a talking person - without the background)
- audio objects (e.g. the voice associated with that person)

Figure 2-An example of an MPEG-4 Scene



MPEG-4 standardizes 2D or 3D media objects. In addition, MPEG-4 defines the coded representation of objects as:

- text and graphics
- talking synthetic heads and associated text used to synthesize the speech and animate the head
- synthetic sound

A media object in its coded form consists of descriptive elements that allow handling the object, in an audiovisual scene as well as associated streaming data, if needed. Each media object can be represented independent of its surroundings or background. The coded representation of media objects is efficient while taking into account the desired functionalities such as **error robustness, easy extraction and editing of an object, or having an object available in a scaleable form**.

Figure 2 contains compound media objects that group primitive media objects together. Example: the visual object corresponding to the talking person and the corresponding voice are tied together to form a new compound media object, containing both the aural and visual components of that talking person. The user can:

- place media objects anywhere in a given coordinate system
- apply transforms to change the appearance of a media object
- group primitive media objects in order to form compound media objects
- apply streamed data to media objects, in order to modify their attributes
- change the user's viewing and listening points anywhere in the scene

2.2 Synchronization And Delivery Of Streaming Data

Synchronization of elementary streams is achieved through time stamping of individual access units within elementary streams. The synchronization layer manages the identification of such access units and the time stamping. Independent of the media type, this layer allows identification of the type of access unit in elementary streams, recovery of the media object's or scene description's time base, and it enables synchronization among them. The syntax of this layer is configurable in a large number of ways, allowing use in a broad range of systems. The play-out of the

multiple MPEG-4 objects is coordinated at a layer devoted solely to synchronization. Here, elementary streams are split into packets, and timing information is added to the payload of these packets. These packets are then ready to be passed on to the transport layer containing a two-layer multiplexer.

The first multiplexing layer is managed according to the DMIF specification (Delivery Multimedia Integration Framework). This multiplex may be embodied by the MPEG-defined FlexMux tool, which allows grouping of Elementary Streams (ESs) with a low multiplexing overhead. The "TransMux" (Transport Multiplexing) layer in Figure 2 models the layer that offers transport services matching the requested QoS.

Only the interface to this layer is specified by MPEG-4 while the concrete mapping of the data packets and control signaling must be done in collaboration with the bodies that have jurisdiction over the respective transport protocol.

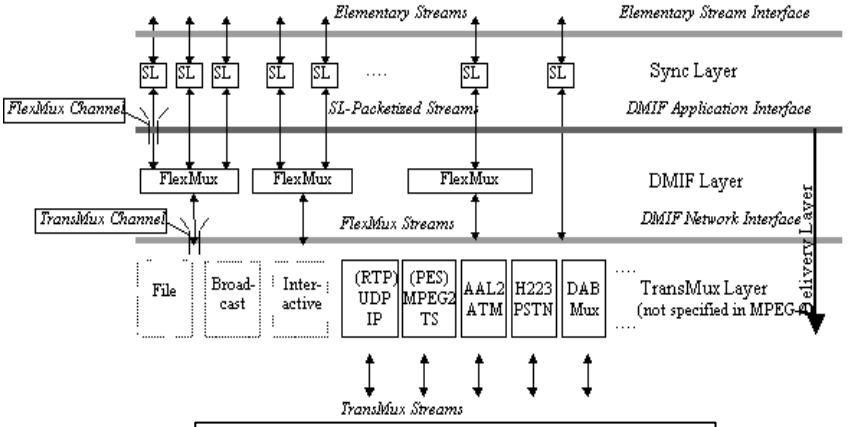


Figure 3-The MPEG-4 System

The choice is left to the end user/service provider, and allows MPEG-4 to be used in a wide variety of operation environments.

2.3 Interaction With Media Objects

In general, the user observes a scene that is composed following the design of the scene's author. Depending on the degree of freedom allowed by the author, however, the user has the possibility to interact with the scene. Operations a user may be allowed to perform include:

- change the viewing/listening point of the scene
- drag objects in the scene to a different position;
- trigger a cascade of events by clicking on a specific object
- select the desired language when multiple language tracks are available

More complex kinds of behavior can also be triggered, e.g. a virtual phone rings, the user answers and a communication link is established.

2.4 Identification Of Intellectual Property

It is important to have the possibility to identify intellectual property in MPEG-4 media objects. Therefore, MPEG has worked with representatives of different creative industries in the definition of syntax and tools to support this. MPEG-4 incorporates identification the intellectual property by storing unique identifiers that are issued by international numbering systems. These numbers can be applied to identify a current rights holder of a media object. Since not all content is identified by such a number, MPEG-4 Version 1 offers the possibility to identify intellectual property by a key-value pair (e.g.:»composer«/»Mr. Gates«).

3 FUNCTIONALITIES IN MPEG-4 VERSION 1

This section contains, in an itemized fashion, the major functionalities that the different parts of the MPEG-4 Standard.

3.1 DMIF

DMIF supports the following functionalities:

- A transparent MPEG-4 DMIF-application interface irrespective of whether the peer is a remote interactive peer, broadcast or local storage media.
- Control of the establishment of FlexMux channels
- Use of homogeneous networks between interactive peers: IP, ATM, mobile, PSTN and Narrowband ISDN.

3.2 Systems

- Scene description for composition of multiple media objects. The scene description provides a rich set of nodes for 2-D and 3-D composition operators and graphics primitives.
- Text with international language support, font and font style selection, timing and synchronization.
- Interactivity, including: client and server-based interaction; a general event model for triggering events or routing user actions; general event handling and routing between objects in the scene, upon user or scene triggered events.
- A tool for interleaving of multiple streams into a single stream, including timing information (FlexMux tool).
- Transport layer independence. Mappings to relevant transport protocol stacks, like MPEG-2 transport stream can be or are being defined jointly with the responsible standardization bodies.
- The initialization and continuous management of the receiving terminal's buffers
- Timing identification, synchronization and recovery mechanisms.
- Datasets covering identification of Intellectual Property Rights relating to media objects.

3.3 Audio

MPEG-4 Audio facilitates a wide variety of applications, which could range from intelligible speech to high quality multi-channel audio, and from natural sounds to synthesized sounds. In particular, it supports the highly efficient representation of audio objects consisting of:

- *Speech signals*: Speech coding can be done using bit-rates from 2 kbit/s up to 24 kbit/s using the speech coding tools. Lower bit-rates, such as an average of 1.2 kbit/s, are also possible when variable rate coding is allowed. Low delay is possible for communications applications. When using the HVXC tools, speed and pitch can be modified under user control during playback. If the CELP tools are used, a change of the playback speed can be achieved by using and additional tool for effects processing.

- *Synthesized Speech*: Scalable TTS coders bitrate range from 200 bit/s to 1.2 Kbit/s which allows a text, or a text with prosodic parameters (pitch contour, phoneme duration, and so on), as its inputs to generate intelligible synthetic speech. It includes the following functionalities.
    1. Speech synthesis using the prosody of the original speech
    2. Lip synchronization control with phoneme information.
    3. Trick mode functionality: pause, resume, jump forward/backward.
    4. International language and dialect support for text. (i.e. it can be signaled in the bitstream which language and dialect should be used)
    5. International symbol support for phonemes.
    6. support for specifying age, gender, speech rate of the speaker
    7. support for conveying facial animation parameter(FAP) bookmarks.
- *General audio signals*: Support for coding general audio ranging from very low bit-rates up to high quality is provided by transform coding techniques. With this functionality, a wide range of bit-rates and bandwidths is covered. It starts at a bitrate of 6 kbit/s and a bandwidth below 4 kHz but also includes broadcast quality audio from mono up to multi-channel.
- *Synthesized Audio*: Synthetic Audio support is provided by a Structured Audio Decoder implementation that allows the application of score-based control information to musical instruments described in a special language.
- *Bounded-complexity Synthetic Audio*: This is provided by a Structured Audio Decoder implementation that allows the processing of a standardized wavetable format.

Examples of additional functionality are speed control and pitch change for speech signals and scalability in terms of bitrate, bandwidth, error robustness, complexity, etc. as defined below.

- The *speed change* functionality allows the change of the time scale without altering the pitch during the decoding process. This can, for example, be used to implement a "fast forward" function (data base search) or to adapt the length of an audio sequence to a given video sequence, or for practicing dance steps at slower play back speed.
- The *pitch change* functionality allows the change of the pitch without altering the time scale during the encoding or decoding process. This can be used, for example, for voice alteration or Karaoke type applications. This technique only applies to parametric and structured audio coding methods.
- *Bitrate scalability* allows a bitstream to be parsed into a bitstream of lower bitrate such that the combination can still be decoded into a meaningful signal. The bitstream parsing can occur either during transmission or in the decoder.
- *Bandwidth scalability* is a particular case of bitrate scalability, whereby part of a bitstream representing a part of the frequency spectrum can be discarded during transmission or decoding.
- *Encoder complexity scalability* allows encoders of different complexity to generate valid and meaningful bitstreams.
- *Decoder complexity scalability* allows a given bitstream to be decoded by decoders of different levels of complexity. The audio quality, in general, is related to the complexity of the encoder and decoder used.
- *Audio Effects* provide the ability to process decoded audio signals with complete timing accuracy to achieve functions for mixing , reverberation, spatialization, etc.

3.4 Visual

The MPEG-4 Visual standard will allow the hybrid coding of natural (pixel based) images and video together with synthetic (computer generated) scenes. This will, for example, allow the virtual presence of videoconferencing participants. To this end, the Visual standard will comprise tools and algorithms supporting the coding of natural (pixel based) still images and video sequences as well as tools to support the compression of synthetic 2-D and 3-D graphic geometry parameters (i.e. compression of wire grid parameters, synthetic text).
The subsections below give an itemized overview of functionalities that the tools and algorithms of the MPEG-4 visual standard will support.

3.4.1 Formats Supported

The following formats and bitrates will be supported by MPEG-4 Version 1:

- bitrates: typically between 5 kbit/s and 10 Mbit/s
- Formats: progressive as well as interlaced video
- Resolutions: typically from sub-QCIF to beyond TV

3.4.2 Compression Efficiency

- Efficient compression of video will be supported for all bit rates addressed. This includes the compact coding of textures with a quality adjustable between "acceptable" for very high compression ratios up to "near lossless".
- Efficient compression of textures for texture mapping on 2-D and 3-D meshes.
- Random access of video to allow functionalities such as pause, fast forward and fast reverse of stored video.

3.4.3 Content-Based Functionalities

- *Content-based coding* of images and video to allow separate decoding and reconstruction of arbitrarily shaped video objects.
- *Random access* of content in video sequences to allow functionalities such as pause, fast forward and fast reverse of stored video objects.
- *Extended manipulation of content* in video sequences to allow functionalities such as warping of synthetic or natural text, textures, image and video overlays on reconstructed video content. An example is the mapping of text in front of a moving video object where the text moves coherently with the object.

3.4.4 Scalability of Textures, Images and Video

- *Complexity scalability in the encoder* allows encoders of different complexity to generate valid and meaningful bitstreams for a given texture, image or video.
- *Complexity scalability in the decoder* allows a given texture, image or video bitstream to be decoded by decoders of different levels of complexity. The reconstructed quality, in general, is related to the complexity of the decoder used. This may entail that less powerful decoders decode only a part of the bitstream.
- *Spatial scalability* allows decoders to decode a subset of the total bitstream generated by the encoder to reconstruct and display textures, images and video objects at reduced spatial resolution. For textures and still images, a maximum of 11 levels of spatial scalability will be supported. For video sequences, a maximum of three levels will be supported.

- *Temporal scalability* allows decoders to decode a subset of the total bitstream generated by the encoder to reconstruct and display video at reduced temporal resolution. A maximum of three levels will be supported.
- *Quality scalability* allows a bitstream to be parsed into a number of bitstream layers of different bitrate such that the combination of a subset of the layers can still be decoded into a meaningful signal. The bitstream parsing can occur either during transmission or in the decoder. The reconstructed quality, in general, is related to the number of layers used for decoding and reconstruction.

3.4.5 Shape and Alpha Channel Coding

- *Shape coding* will be supported to assist the description and composition of conventional images and video as well as arbitrarily shaped video objects. Applications that benefit from binary shape maps with images are content based image representations for image databases, interactive games, surveillance, and animation. Efficient techniques are provided that allow efficient coding of binary shape. A binary alpha map defines whether or not a pixel belongs to an object. It can be 'on' or 'off'.
- *'Gray Scale' or 'alpha' Shape Coding*

An alpha plane defines the 'transparency' of an object, which is not necessarily uniform. Multilevel alpha maps are frequently used to blend different layers of image sequences. Other applications that benefit from associated binary alpha maps with images are content based image representations for image databases, interactive games, surveillance, and animation. Efficient techniques are provided, that allow coding of binary as well as gray scale alpha planes. A binary alpha map defines whether or not a pixel belongs to an object. It can be 'on' or 'off'. A gray scale map offers the possibility to define the exact transparency of each pixel.

3.4.6 Robustness in Error Prone Environments

Error resilience will be supported to assist the access of image and video over a wide range of storage and transmission media. This includes the useful operation of image and video compression algorithms in error-prone environments at low bit-rates (i.e., less than 64 Kbps). Tools are provided which address both the band limited nature and error resiliency aspects for access over wireless networks.

3.4.7 Face Animation

The 'Face Animation' part of the standard allow sending parameters that calibrate and animate synthetic faces. These models themselves are not standardized by MPEG-4, only the parameters are.

- Definition and coding of face animation parameters (model independent):
- Feature point positions and orientations to animate the face definition meshes
- Visual lip configurations equivalent to speech phonemes
- Definition and coding of face definition parameters (for model calibration):
- 3-D feature point positions
- 3-D head calibration meshes for animation
- Texture map of face
- Personal characteristics
- Facial texture coding

3.4.8 Coding of 2-D Meshes with Implicit Structure

- Mesh-based prediction and animated texture transfiguration
- 2-D regular mesh formalism with motion tracking of animated objects
- Motion prediction and suspended texture transmission with dynamic meshes.
- Geometry compression for motion vectors:
- 2-D mesh compression with implicit structure & decoder reconstruction

3.4.9 MPEG-J

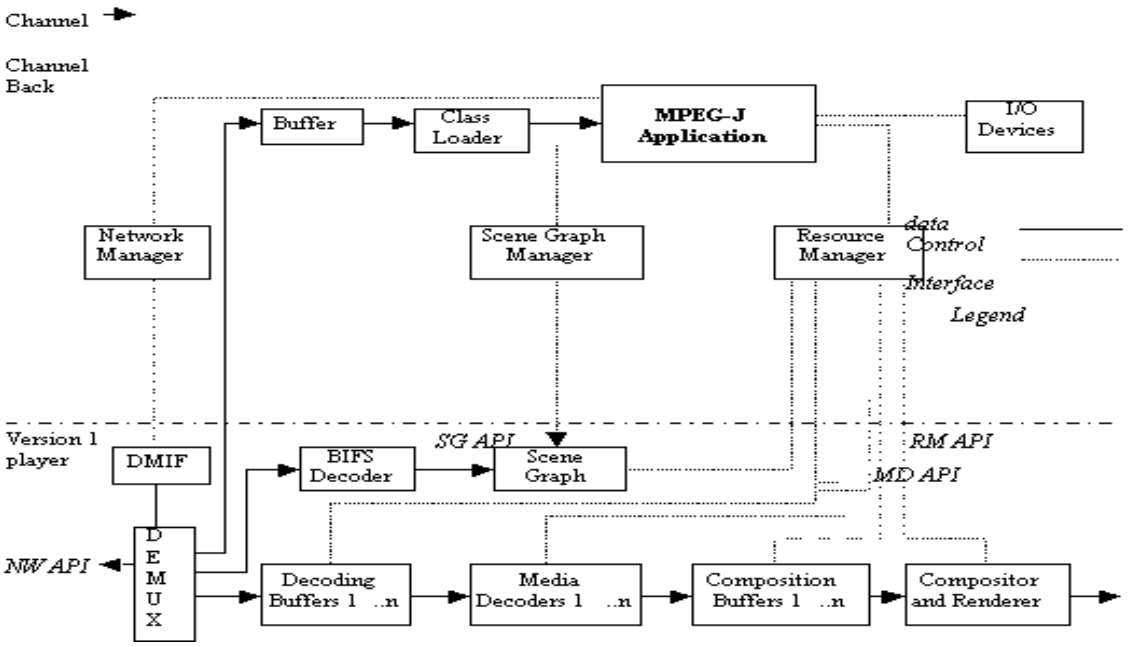


Figure 4-Location of interfaces in the architecture of an MPEG-J enabled MPEG-4 System

The MPEG-J is a programmatic system (as opposed to the parametric system offered by MPEG-4 Version 1) which specifies API for interoperation of MPEG-4 media players with Java code. By combining MPEG-4 media and safe executable code, content creators may embed complex control and data processing mechanisms with their media data to intelligently manage the operation of the audio-visual session. A block diagram of the MPEG-J player in an MPEG-4 system player environment is shown in Figure 4. The lower half of this drawing depicts the parametric MPEG-4 System player also referred to as the Presentation Engine. The MPEG-J subsystem controlling the Presentation Engine, also referred to as the Application Engine, is depicted in the upper half of Figure 4.MPEG-J specifically does *not* support downloadable decoders.

4 PROFILES IN MPEG-4

MPEG-4 provides a large and rich set of tools for the coding of audio-visual objects. In order to allow effective implementations of the standard, subsets of the MPEG-4 Systems, Visual, and Audio tool sets have been identified, that can be used for specific applications. These subsets, called 'Profiles', limit the tool set a decoder has to implement. For each of these Profiles, one or more Levels have been set, restricting the computational complexity. A Profile@Level combination allows:

- a codec builder to implement only the subset of the standard he needs, while maintaining inter-working with other MPEG-4 devices built to the same combination, and
- checking whether MPEG-4 devices comply with the standard ('conformance testing').

Profiles exist for various types of media content (audio, visual, and graphics) and for scene descriptions. MPEG does not prescribe or advise combinations of these Profiles, but care has been taken that good matches exist between the different areas.

4.1 Visual Profiles

The visual part of the standard provides profiles for the coding of natural, synthetic, and synthetic/natural hybrid visual content. There are five profiles for natural video content:

1. 1. The *Simple* Visual Profile provides efficient, error resilient coding of rectangular video objects, suitable for applications on mobile networks
2. The *Simple Scalable* Visual Profile adds support for coding of temporal and spatial scalable objects to the Simple Visual Profile, It is useful for applications which provide services at more than one level of quality due to bit-rate or decoder resource limitations, such as Internet use and software decoding.
3. The *Core* Visual Profile adds support for coding of arbitrary-shaped and temporally scalable objects to the Simple Visual Profile. It is useful for applications such as those providing relatively simple content-interactivity (Internet multimedia applications).
4. The *Main* Visual Profile adds support for coding of interlaced, semi-transparent, and sprite objects to the Core Visual Profile. It is useful for interactive and entertainment-quality broadcast and DVD applications.
5. The *N-Bit* Visual Profile adds support for coding video objects having pixel-depths ranging from 4 to 12 bits to the Core Visual Profile. It is suitable for use in surveillance applications.

The profiles for synthetic and synthetic/natural hybrid visual content are:

1. The *Simple Facial Animation* Visual Profile provides a simple means to animate a face model, suitable for applications such as audio/video presentation for the hearing impaired.
2. The *Scalable Texture Visual* Profile provides spatial scalable coding of still image (texture) objects useful for applications needing multiple scalability levels, such as mapping texture onto objects in games, and high-resolution digital still cameras.
3. The *Basic Animated 2-D Texture* Visual Profile provides spatial scalability, SNR scalability, and mesh-based animation for still image (textures) objects and also simple face object animation.

4. The *Hybrid Visual* Profile combines the ability to decode arbitrary-shaped and temporally scalable natural video objects (as in the Core Visual Profile) with the ability to decode several synthetic and hybrid objects, including simple face and animated still image objects. It is suitable for various content-rich multimedia applications.

4.2 Audio Profiles

Four Audio Profiles have been defined in MPEG-4 V.1:

1. The *Speech Profile* provides HVXC, which is a very-low bit-rate parametric speech coder, a CELP Narrowband/Wideband speech coder, and a Text-To-Speech interface.
2. The *Synthesis Profile* provides score driven synthesis using SAOL and wavetables and a Text-to-Speech Interface to generate sound and speech at very low bit-rates.
3. The *Scalable Profile*, a superset of the Speech Profile, is suitable for scalable coding of speech and music for networks, such as Internet and Narrow band Audio DIgital Broadcasting (NADIB). The bit-rates range from 6 kbit/s and 24 kbit/s, with bandwidths between 3.5 and 9 kHz.
4. The *Main Profile* is a rich superset of all the other Profiles, containing tools for natural and synthetic Audio.

4.3 Graphics Profiles

Graphics Profiles define which graphical and textual elements can be used in a scene. These profiles are defined in the Systems part of the standard:

1. **Simple 2-D Graphics Profile**: provides for only those graphics elements of the BIFS tool that are necessary to place one or more visual objects in a scene.
2. **Complete 2-D Graphics Profile**: provides two-dimensional graphics functionalities and supports features such as arbitrary two-dimensional graphics and text, possibly in conjunction with visual objects.
3. **Complete Graphics Profile:** provides advanced graphical elements such as elevation grids and extrusions and allows creating content with sophisticated lighting. The Complete Graphics profile enables applications such as complex virtual worlds that exhibit a high degree of realism.

4.4 Scene Description Profiles

Scene Description Profiles, defined in the Systems part of the standard, allow audiovisual scenes with audio-only, 2-dimensional, 3-dimensional or mixed 2-D/3-D content. The 3-D Profile is called 'VRML', as it optimizes inter-working with VRML material:

1. The *Audio* Scene Graph Profile provides for a set of BIFS scene graph elements for usage in audio only applications. The Audio Scene Graph profile supports applications like broadcast radio.
2. The *Simple 2-D* Scene Graph Profile provides for only those BIFS scene graph elements necessary to place one or more audio-visual objects in a scene. The Simple 2-D Scene Graph profile allows presentation of audio-visual content with potential update of the complete scene but no interaction capabilities. The

Simple 2-D Scene Graph profile supports applications like broadcast television.

3. The *Complete 2-D* Scene Graph Profile provides for all the 2-D scene description elements of the BIFS tool. It supports features such as 2-D transformations and alpha blending. The Complete 2-D Scene Graph profile enables 2-D applications that require extensive and customized interactivity.
4. The *Complete* Scene Graph profile provides the complete set of scene graph elements of the BIFS tool. The Complete Scene Graph profile will enable applications like dynamic virtual 3-D world and games.

4.5 MPEG-J Profiles

Two MPEG-J Profiles exist: Personal and Main:

Personal - a lightweight package for personal devices.

The personal profile addresses a range of constrained devices including mobile and portable devices. Examples of such devices are cell video phones, PDAs, personal gaming devices. This profile includes the following packages of MPEG-J APIs

1. Network
2. Scene
3. Resource

Main - includes all the MPEG-J API's.

The Main profile addresses a range of consumer devices including entertainment devices. Examples of such devices are set top boxes, computer based multimedia systems etc. It is a superset of the Personal profile. Apart from the packages in the Personal profile, this profile includes the following packages of the MPEG-J APIs.

1. Decoder
2. Decoder Functionality
3. Section Filter and Service Information

5 DETAILED TECHNICAL DESCRIPTION OF THE MPEG-4 STANDARD

This Section contains a detailed overview of the MPEG-4 Standard. It first described the entire system, and then describes all parts of the system in subsections.

Figure 5 shows how streams coming from the network (or a storage device), as TransMux Streams, are de-multiplexed into FlexMux Streams and passed to appropriate FlexMux de-multiplexers that retrieve Elementary Streams. The Elementary Streams (ESs) are parsed and passed to the appropriate decoders. Decoding recovers the data in an AV object from its encoded form and performs the necessary operations to reconstruct the original AV object ready for rendering on the appropriate device. Audio and visual objects are represented in their coded form. The reconstructed AV object is made available to the composition layer for potential use during scene rendering. Decoded AVOs, along with scene description information, are used to compose the scene as described by the author. The user can, to the extent allowed by the author, interact with the scene, which is eventually rendered and presented.

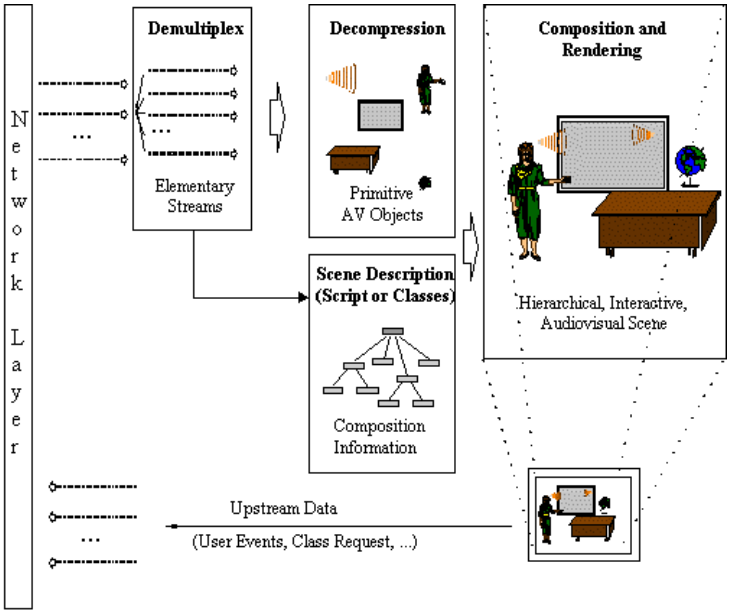


Figure 5-Major components of an MPEG-4 terminal (receiver side)

5.1 DMIF

DMIF (Delivery Multimedia Integration Framework) is a session protocol for the management of multimedia streaming over generic delivery technologies. In principle it is similar to FTP. The only (essential!) difference is that FTP returns data, DMIF returns pointers to where to get (streamed) data

When FTP is run, the very first action it performs is the setup of a session with the remote side. Later, files are selected and FTP sends a request to download them, the FTP peer will return the files in a separate connection.

Similarly, when DMIF is run, the very first action it performs is the setup of a session with the remote side. Later, streams are selected and DMIF sends a request to stream them, the DMIF peer will return the pointers to the connections where the streams will be streamed, and then also establishes the connection themselves.

Compared to FTP, DMIF is both a framework and a protocol. The functionality provided by DMIF is expressed by an interface called DMIF-Application Interface (DAI), and translated into protocol messages. These protocol messages may differ based on the network on which they operate. The Quality of Service is also considered in the DMIF design, and the DAI allows the DMIF user to specify the requirements for the desired stream. It is then up to the DMIF implementation to make sure that the

requirements are fulfilled. The DMIF specification provides hints on how to perform such tasks on a few network types, such as the Internet.

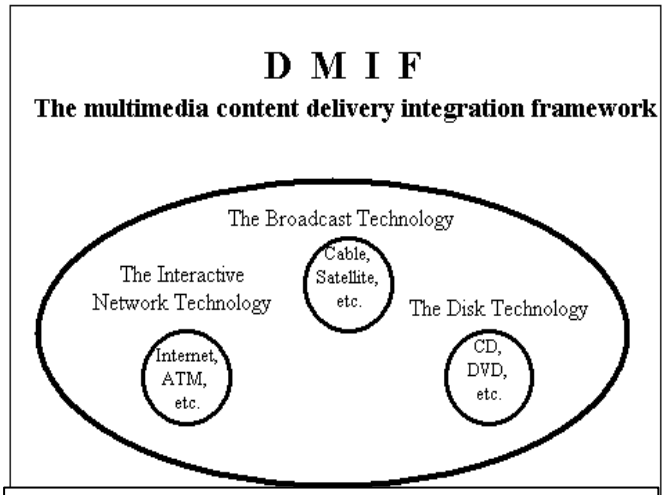


Figure 6-DMIF addresses the delivery integration of three major technologies

The DAI is also used for accessing broadcast material and local files, this means that a single, uniform interface is defined to access multimedia contents on a multitude of delivery technologies.
As a consequence, it is appropriate to state that the integration framework of DMIF covers three major technologies, interactive network technology, broadcast technology and the disk technology; this is shown in the Figure 6 on the left.

The DMIF architecture is such that applications that rely on DMIF for communication do not have to be concerned with the underlying communication method. The implementation of DMIF takes care of the delivery technology details presenting a simple interface to the application.
Figure 6 represents the above concept. An application accesses data through the DMIF-Application Interface, irrespective of whether such data comes from a broadcast source, from local storage or from a remote server. In all scenarios the Local Application only interacts through a uniform interface (DAI). Different DMIF instances will then translate the Local Application requests into specific messages to be delivered to the Remote Application, taking care of the peculiarities of the involved delivery technology. Similarly, data entering the terminal (from remote servers, broadcast networks or local files) is uniformly delivered to the Local Application through the DAI.

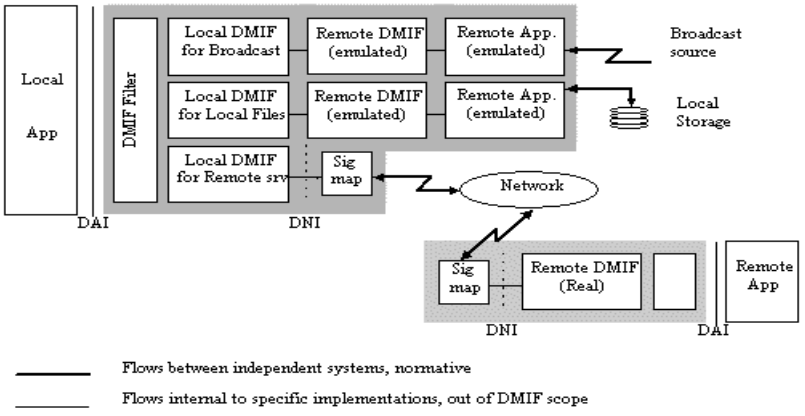


Figure 7-DMIF communication architecture

Different, specialized DMIF instances are indirectly invoked by the Application to manage the various specific delivery technologies, this is however transparent to the Application, that only interacts with a single "DMIF filter". This filter is in charge of directing the particular DAI primitive to the right instance. DMIF does not specify this mechanism, just assumes it is implemented. This is further emphasized by the shaded boxes in the figure, whose aim is to clarify what are the borders of a DMIF implementation, while

the DMIF communication architecture defines a number of modules, actual DMIF implementations only need to preserve their appearance at those borders.
Conceptually, a "real" remote application accessed through a network e.g., IP- or ATM-based, is no different than an emulated remote producer application getting content from a broadcast source or from a disk. In the former case, however, the messages exchanged between the two entities have to be normatively defined to ensure interoperability (these are the DMIF Signaling messages). In the latter case, on the other hand, the interfaces between the two DMIF peers and the emulated Remote Application are internal to a single implementation and need not be considered in this specification. Note that for the broadcast and local storage scenarios, the figure shows a chain of "Local DMIF", "Remote DMIF (emulated)" and "Remote Application (emulated)". This chain only represents a conceptual model and need not be reflected in actual implementations (it is shown in the figure totally internal to a shaded box).

5.2 De-multiplexing, Synchronization And Description Of Streaming Data

Individual Elementary Streams have to be retrieved on the delivery layer from incoming data from some network connection or a storage device. Each network connection or file is homogeneously considered a TransMux Channel in the MPEG-4 system model. The de-multiplexing is partially or completely done by layers outside the scope of MPEG-4, depending on the application. The only multiplexing tool defined by MPEG-4 is the FlexMux tool that may optionally be used for low delay, low overhead multiplexing and for saving network connection resources.
For the purpose of integrating MPEG-4 in system environments, the DMIF Application Interface is the reference point at which elementary streams can be accessed as sync layer-packetized streams. The DMIF Network Interface specifies how either SL(Sync Layer)-packetized streams —no FlexMux used— or FlexMux Streams are to be retrieved from the TransMux Layer. This is the interface to the transport functionalities not defined by MPEG. The data part of the interfaces is considered here, while the control part is dealt with by DMIF.
In the same way that MPEG-1 and MPEG-2 describe the behavior of an idealized decoding device along with the bitstream syntax and semantics, MPEG-4 defines a System Decoder Model. This allows the precise definition of the terminal's operation without making unnecessary assumptions about implementation details. This is essential in order to give implementers the freedom to design real MPEG-4 terminals and decoding devices in a variety of ways. These devices range from television receivers, which have no ability to communicate with the sender, to computers that are fully enabled with bi-directional communication. Some devices will receive MPEG-4 streams over isochronous networks, while others will use non-isochronous means (e.g., the Internet) to exchange MPEG-4 information. The System Decoder Model provides a common model on which all implementations of MPEG-4 terminals can be based.
The specification of a buffer and timing model is essential to encoding devices which may not know ahead of time what the terminal device is or how it will receive the encoded stream. Though the MPEG-4 specification will enable the encoding device to inform the decoding device of resource requirements, it may not be possible, as indicated earlier, for that device to respond to the sender. It is also possible that an MPEG-4 session is received simultaneously by widely different devices; it will, however, be properly rendered according to the capability of each device.

5.2.1 Demultiplexing

Demultiplexing occurs on the delivery layer that is modeled as consisting of a TransMux layer and a DMIF layer. The retrieval of incoming data streams from network connections or storage media consists of two tasks. First, the channels must be located and opened. This requires a transport control entity that manages, among others, the tables that associate transport channels to specific elementary streams. Stream map tables link each stream to a Channel Association Tag that serves as a handle to the channel that carries this stream. Resolving Channel Association Tags to the actual transport channel as well as the management of the sessions and channels is addressed by the DMIF part of the MPEG-4 standard.

Second, the incoming streams must be properly de-multiplexed to recover SL-packetized streams from downstream channels (incoming at the receiving terminal) to be passed on to the synchronization layer. In interactive applications, a corresponding multiplexing stage will multiplex upstream data in upstream channels (outgoing from the receiving terminal).

The generic term 'TransMux Layer' is used to abstract any underlying multiplex functionality – existing or future – that is suitable to transport MPEG-4 data streams. The TransMux Layer is assumed to provide protection and multiplexing functionality, indicating that this layer is responsible for offering a specific QoS. Protection functionality includes error protection and error detection tools suitable for the given network or storage medium.

In any concrete application scenario one or more specific TransMux Instances will be used. Each TransMux demultiplexer gives access to TransMux Channels. The requirements on the data interface to access a TransMux Channel are the same for all TransMux Instances. They include the need for reliable error detection, delivery, if possible, of erroneous data with a suitable error indication and framing of the payload, which may consist of either SL-packetized streams or FlexMux streams. These requirements are summarized in an informative way in the TransMux Interface, in the Systems part of the MPEG-4 Standard. An adaptation of SL-packetized streams must be specified to each transport protocol stack of interest according to these requirements and in conjunction with the standardization body that has the proper jurisdiction. This is happening for RTP and mobile channels at the moment.

The FlexMux tool is specified by MPEG to optionally provide a flexible, low overhead, low delay method for interleaving data whenever this is not sufficiently supported by the underlying protocol stack. It is especially useful when the packet size or overhead of the underlying TransMux instance is large, so that a waste of bandwidth or number of network connections would result otherwise. The FlexMux tool is not itself robust to errors and can either be used on TransMux Channels with a high QoS or to bundle Elementary Streams that are equally error tolerant. The FlexMux requires reliable error detection and sufficient framing of FlexMux packets (for random access and error recovery) from the underlying layer. These requirements are also reflected in the data primitives of the DMIF Application Interface, which defines the data access to individual transport channels. The FlexMux demultiplexer retrieves SL-packetized streams from FlexMux Streams.

5.2.2 Synchronization And Description Of Elementary Streams

The sync layer has a minimum set of tools for consistency checking, padding, to convey time base information and to carry time stamped access units of an elementary stream. Each packet consists of one access unit or a fragment of an access unit. These time stamped access units form the only semantic structure of elementary streams that is visible on this layer. Time stamps are used to convey the nominal decoding and composition time for an access unit. The sync layer requires reliable error detection

and framing of each individual packet from the underlying layer, which can be accomplished, e.g., by using the FlexMux. How data can be accessed by the compression layer is summarized in the informative Elementary Stream Interface, which can be found in the Systems part of the MPEG-4 Standard. The sync layer retrieves elementary streams from SL-packetized streams.

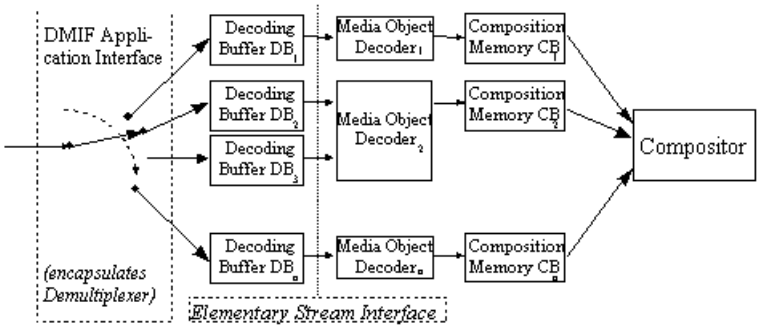


Figure 10-Buffer architecture of the System Decoder Model

To be able to relate elementary streams to media objects within a scene, object descriptors are used. Object Descriptors convey information about the number and properties of elementary streams that are associated to particular media objects. Object descriptors are themselves conveyed in one or more elementary streams, since it is possible to add and discard streams (and objects) during the course of an MPEG-4 session. Such updates are time stamped in order to guarantee synchronization. The object descriptor streams can be considered as a description of the streaming resources for a presentation. Similarly, the scene description is also conveyed as an elementary stream, allowing modifying the spatio-temporal layout of the presentation over time.

5.3 Coding Of Audio Objects

MPEG-4 coding of audio objects provides tools for both representing natural sounds and for synthesizing sounds based on structured descriptions. The representation for synthesized sound can be derived from text data or so-called instrument descriptions and by coding parameters to provide effects, such as reverberation and spatialization. The representations provide compression and other functionalities, such as scalability and effects processing. With the intent of identifying a suitable digital audio broadcast format to provide improvements over the existing AM modulation services, several codec configurations involving the MPEG-4 CELP, TwinVQ, and AAC tools have been compared to a reference AM system. It was found that higher quality can be achieved in the same bandwidth with digital techniques and that scalable coder configurations offered performance superior to a simulcast alternative. Additional verification tests were carried out by MPEG, in which the tools for speech and general audio coding were compared to existing standards.

5.3.1 Natural Sound

MPEG-4 standardizes natural audio coding at bit-rates ranging from 2 kbit/s up to and above 64 kbit/s. When variable rate coding is allowed, coding at less than 2 kbit/s, such as an average bitrate of 1.2 kbit/s, is also supported. The presence of the MPEG-2 AAC standard within the MPEG-4 tool set provides for general compression of audio in the upper bitrate range. For these, the MPEG-4 standard defines the bitstream syntax and the decoding processes in terms of a set of tools. In order to achieve the highest audio quality within the full range of bit-rates and at the same time provide the extra functionalities, speech coding techniques and general audio coding techniques are integrated in a common framework:

- Speech coding at bit-rates between 2 and 24 kbit/s is supported by using Harmonic Vector eXcitation Coding (HVXC) for a recommended operating bitrate of 2 - 4 kbit/s, and Code Excited Linear Predictive (CELP) coding for an operating bitrate of 4 - 24 kbit/s. In addition, HVXC can operate down to an average of around 1.2 kbit/s in its variable bitrate mode. In CELP coding, two sampling rates, 8 and 16 kHz, are used to support narrow band and wide band speech, respectively. The following operating modes have been subject to verification testing: HVXC at 2 and 4 kbit/s, narrow band CELP at 6, 8.3, and 12 kbit/s, and wide band CELP at 18 kbit/s. In addition various of the scalable configurations have been verified.
- For general audio coding at bit-rates at and above 6 kbit/s, transform coding techniques, namely TwinVQ and AAC, are applied. The audio signals in this region typically have sampling frequencies starting at 8 kHz.

Starting with a coder operating at a low bitrate, by adding enhancements to a general audio coder, both the coding quality as well as the audio bandwidth can be improved.

Bitrate scalability, often also referred to as embedded coding, allows a bitstream to be parsed into a bitstream of lower bitrate that can still be decoded into a meaningful signal. The bitstream parsing can occur either during transmission or in the decoder. Bandwidth scalability is a particular case of bitrate scalability whereby part of a bitstream representing a part of the frequency spectrum can be discarded during transmission or decoding.

Encoder complexity scalability allows encoders of different complexity to generate valid and meaningful bitstreams. The decoder complexity scalability allows a given bitstream to be decoded by decoders of different levels of complexity. The audio quality, in general, is related to the complexity of the encoder and decoder used. Scalability works within some MPEG-4 tools, but can also be applied to a combination of techniques, e.g. with CELP as a base layer and AAC for the enhancement layer(s).

The MPEG-4 systems allows codecs according to existing (MPEG) standards, e.g. MPEG-2 AAC, to be used. Each of the MPEG-4 coders is designed to operate in a stand-alone mode with its own bitstream syntax. Additional functionalities are realized both within individual coders, and by means of additional tools around the coders. An example of such a functionality within an individual coder is speed or pitch change within HVXC.

5.3.2 Synthesized Sound

MPEG-4 defines decoders for generating sound based on several kinds of 'structured' inputs. Text input is converted to speech in the Text-To-Speech (TTS) decoder, while more general sounds including music may be normatively synthesized. Synthetic music may be delivered at extremely low bit-rates while still describing an exact sound signal.

Text To Speech.

TTS coders bit-rates range from 200 bit/s to 1.2 Kbit/s, which allows a text or a text with prosodic parameters (pitch contour, phoneme duration, and so on) as its inputs to generate intelligible synthetic speech. It supports the generation of parameters that can be used to allow synchronization to associated face animation, international languages for text and international symbols for phonemes. Additional markups are used to convey control information within texts, which is forwarded to other components in synchronization with the synthesized text.

Score Driven Synthesis.

The Structured Audio tools decode input data and produce output sounds. This decoding is driven by a special synthesis language called SAOL (Structured Audio Orchestra Language) standardized as a part of MPEG-4. This language is used to

define an "orchestra" made up of "instruments" (downloaded in the bitstream, not fixed in the terminal) which create and process control data. An instrument is a small network of signal processing primitives that might emulate some specific sounds such as those of a natural acoustic instrument. The signal-processing network may be implemented in hardware or software and include both generation and processing of sounds and manipulation of pre-stored sounds.
MPEG-4 does not standardize "a single method" of synthesis, but rather a way to describe methods of synthesis. Any current or future sound-synthesis method can be described in SAOL, including wavetable, FM, additive, physical-modeling, and granular synthesis, as well as non-parametric hybrids of these methods.
Control of the synthesis is accomplished by downloading "scores" or "scripts" in the bitstream. A score is a time-sequenced set of commands that invokes various instruments at specific times to contribute their output to an overall music performance or generation of sound effects. The score description, downloaded in a language called SASL (Structured Audio Score Language), can be used to create new sounds, and also include additional control information for modifying existing sound. This allows the composer finer control over the final synthesized sound. For synthesis processes that do not require such fine control, the established MIDI protocol may also be used to control the orchestra.
Careful control in conjunction with customized instrument definition allows the generation of sounds ranging from simple audio effects, such as footsteps or door closures, to the simulation of natural sounds such as rainfall or music played on conventional instruments to fully synthetic sounds for complex audio effects or futuristic music.
For terminals with less functionality, and for applications which do not require such sophisticated synthesis, a "wavetable bank format" is also standardized. Using this format, sound samples for use in wavetable synthesis may be downloaded, as well as simple processing, such as filters, reverbs, and chorus effects. In this case, the computational complexity of the required decoding process may be exactly determined from inspection of the bitstream, which is not possible when using SAOL.

5.4 Coding Of Visual Objects

MPEG-4 Video offers technology that covers a large range of existing applications as well as new ones. The low-bit rate and error resilient coding allows for robust communication over limited rate wireless channels, useful for e.g. mobile videophones and space communication. There may also be roles in surveillance data compression since it is possible to have a very low or variable frame rate. At high bit-rates, tools are available to allow the transmission and storage of high-quality video suitable for the studio and other very demanding content creation applications. It is likely that the standard will eventually support data-rates well beyond those of MPEG-2.
A major application area is interactive web-based video. Software that provides live MPEG-4 video on a web page has already been demonstrated. There is much room for applications to make use of MPEG-4's object-based characteristics. The binary and grayscale shape-coding tools allow arbitrary-shaped video objects to be composed together with text and graphics. Doing so, a rich interactive experience for web-based presentations and advertising can be provided; this same scenario also applies to set-top box applications. Additionally, it is possible to make use of scalability tools to allow for a smooth control of user experience with terminal and data link capabilities.

MPEG-4 video has already been used to encode video captures with a hand-held camera. This form of application is likely to grow in popularity with its fast and easy transfer to a web page, and may also make use of MPEG-4 still-texture mode for still frame capture. The games market is another area where the application of MPEG-4 video, still-texture, interactivity and SNHC shows much promise, with 3-D texture mapping of still images, live video, or extended pre-recorded video sequences enhancing the player experience. Adding live video of users adds to the user experience multi-player 3-D games, as does use of arbitrary-shaped video, where transparency could be combined artistically with 3-D video texture mapping.

5.4.1 Natural Textures, Images and Video

The tools for representing natural video in the MPEG-4 visual standard provide standardized core technologies allowing efficient storage, transmission and manipulation of textures, images and video data for multimedia environments. These tools allow the decoding and representation of atomic units of image and video content, called "video objects" (VOs). An example of a VO could be a talking person (without background), which can then be composed with other AVOs (audio-visual objects) to create a scene. Conventional rectangular imagery is handled as a special case of such objects.

In order to achieve this broad goal rather than a solution for a narrow set of applications, functionalities common to several applications are clustered. Therefore, the visual part of the MPEG-4 standard provides solutions in the form of tools and algorithms for:

- efficient compression of images and video 2-D meshes
- efficient compression of textures for texture mapping on 2-D and 3-D meshes
- efficient compression of time-varying geometry streams that animate meshes
- efficient random access to all types of visual objects
- extended manipulation functionality for images and video sequences
- content-based coding and scalability of images and video
- spatial, temporal and quality scalability
- error robustness and resilience in error prone environments

5.4.2 Synthetic Objects

Synthetic objects form a subset of the larger class of computer graphics, as an initial focus the following visual synthetic objects will be described:

- Parametric descriptions of
    1. a synthetic the face and body (body animation in Version 2)
    2. Static and Dynamic Mesh Coding with texture mapping
- Texture Coding for View Dependent applications

5.4.2.1 facial animation

The ‘facial animation object’ can be used to render an animated face. The shape, texture and expressions of the face are controlled by Facial Definition Parameters (FDPs) and/or Facial Animation Parameters (FAPs). Upon construction, the face object contains a generic face with a neutral expression. This face can already be rendered. It can also immediately receive the animation parameters from the bitstream, which will produce animation of the face: expressions, speech etc. Meanwhile, definition parameters can be sent to change the appearance of the face from something generic to a particular face with its own shape and (optionally) texture. If so desired, a complete face model can be downloaded via the FDP set.

Face Animation in MPEG-4 Version 1 provides for highly efficient coding of animation parameters that can drive an unlimited range of face models. The models

themselves are not normative, although (see above) there are normative tools to describe the appearance of the model. Frame-based and temporal-DCT coding of a large collection of FAPs can be used for accurate speech articulation. Expression parameters are used to code specific speech configurations of the lips and the mood of the speaker.

The Systems Binary Format for Scenes provides features to support Face Animation when custom models and specialized interpretation of FAPs are needed:

1. The Face Definition Parameters (FDP) in BIFS (model data downloadable to configure a baseline face model pre-stored in the terminal into a particular face before FAP decoding, or to install a specific face model at the beginning of a session along with the information about how to animate it);
2. The Face Animation Table (FAT) within FDPs (downloadable functional mapping from incoming FAPs to feature control points in the face mesh. This provides piecewise linear mappings of incoming FAPs for controlling facial movements).
3. The Face Interpolation Technique (FIT) in BIFS (downloadable definition of mapping of incoming FAPs into a total set of FAPs before their application to feature points, through weighted rational polynomial functions invoked by conditional evaluation of a Face Interpolation Graph). This can be used for complex cross-coupling of FAPs to link their effects, or to interpolate FAPs missing in the stream using the FAPs that are available in the terminal).

These specialized node types in BIFS effectively provide for tailored face models including calibration of an established face model in a terminal or downloading of a fully custom model including its shape, texture, and color.

5.4.2.2 2-D animated meshes

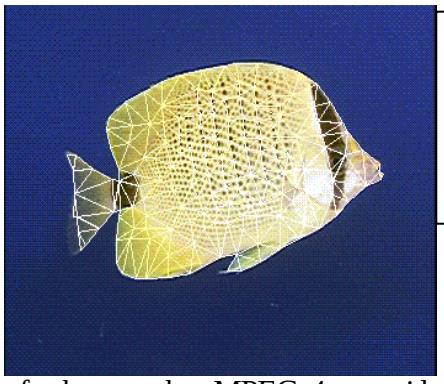


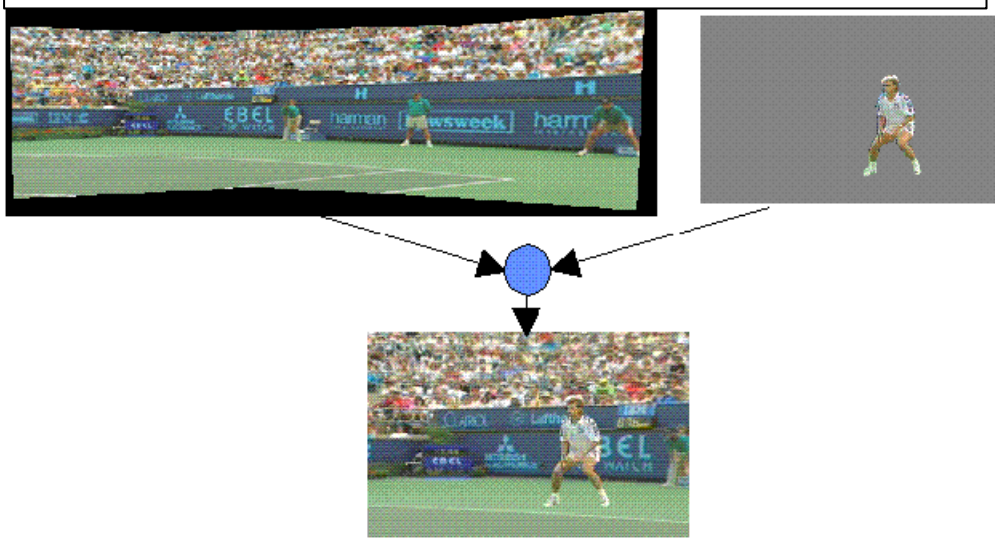
Figure 9-2-D mesh modeling a fish. By deforming the mesh, the fish can be animated very efficiently, and be made to 'swim'. Also, a logo could be projected onto the fish, and made to move in accordance with the fish

A 2-D *mesh* is a partition of a 2-D planar region into polygonal patches. The vertices of the polygonal patches are referred to as the *node points* of the mesh. MPEG-4 considers only triangular meshes where the patches are triangles. A 2-D dynamic mesh refers to 2-D mesh geometry and motion information of all mesh node points within a temporal segment of interest. Triangular meshes have long been used for efficient 3-D object shape (geometry) modeling and rendering in computer graphics. 2-D mesh modeling may be considered as projection of such 3-D triangular meshes onto the image plane.

Figure 10 depicts the basic concept for coding an MPEG-4 video sequence using a sprite panorama image. It is assumed that the foreground object (tennis player, image top right) can be segmented from the background and that the sprite panorama image can be extracted from the sequence prior to coding. (A sprite panorama is a still image that describes as a static image the content of the background over all frames in the sequence). The large panorama sprite image is transmitted to the receiver only once as first frame of the sequence to describe the background – the sprite remains is stored in a sprite buffer. In each consecutive frame only the camera parameters relevant for the background are transmitted to the receiver. This allows the receiver to reconstruct the background image for each frame in the sequence based on the sprite. The moving

foreground object is transmitted separately as an arbitrary-shape video object. The receiver composes both the foreground and background images to reconstruct each frame (bottom picture in figure below). For low delay applications it is possible to

Figure 10-Example of Sprite Coding of Video Sequence



transmit the sprite in multiple smaller pieces over consecutive frames or to build up the sprite at the decoder progressively.

6 TERMS USED

API	Application Programming Interface
BIFS	Binary Format for Scene description
DMIF	Delivery Multimedia Integration Framework
ES	Elementary Stream: A sequence of data that originates from a single producer in the transmitting MPEG-4 Terminal and terminates at a single recipient, e.g. an media object or a Control Entity in the receiving MPEG-4 Terminal. It flows through one FlexMux Channel.
FlexMux stream	A sequence of FlexMux packets associated to one or more FlexMux Channels flowing through one TransMux Channel
FlexMux tool	A Flexible (Content) Multiplex tool
HVXC	Harmonic Vector Excitation Coding
MPEG-J	Framework for MPEG Java API's
QoS	Quality of Service
SNHC	Synthetic- Natural Hybrid Coding
TransMux	Generic abstraction for any transport multiplex scheme

7 REFERENCES

- L. Chiariglione (Convenor), "MPEG-4 project description", Document ISO/IEC JTC1/SC29/WG11 N1177, Munich MPEG meeting, January 1996
- http://www.cselt.it
- http://www.sait.samsung.co.kr
- http://www-elec.enst.fr
- http://www.es.com
- http://www.iso.ch
- http://www.tnt.uni-hannover.de
- http://www.sv.philips.com
- http://www.mpeg.org
- IEEE Spectrum February 1999 Volume 36 Number 2