AudioBIFS: The MPEG-4 Standard for Effects Processing

Eric D. Scheirer (1)
Riitta Väänänen (2)
Jyri Huopaniemi (3)

(1) Machine Listening Group, MIT Media Laboratory
eds@media.mit.edu http://sound.media.mit.edu/mpeg4
(2) Laboratory of Acoustics, Helsinki University of Technology
Riitta.Vaananen@hut.fi http://www.acoustics.hut.fi/
(3) CCRMA, Stanford University
Laboratory of Acoustics, Helsinki University of Technology
jyri@ccrma.stanford.edu http://www-ccrma.stanford.edu/

Abstract

We present a tutorial overview of the AudioBIFS system, part of the Binary Format for Scene Description in the MPEG-4 International Standard. AudioBIFS allows the flexible construction of sound scenes using streaming audio, interactive presentation, 3-D spatialization and environmental auralization, and dynamic download of custom signal-processing routines. MPEG-4 sound scenes are based on a model that is a superset of the model in VRML 2.0, and a comparison between the two models is presented. We discuss the use of SAOL, the MPEG-4 Structured Audio Orchestra Language, for writing downloadable effects. The current status of the standard is described.

1 Introduction

ISO/IEC JTC1 SC29 WG11, a subcommittee of the International Organisation for Standardisation that is better known as the Moving Pictures Experts Group or "MPEG", began a new work item in 1995 to standardize low-bit-rate coding tools for the Internet and other bandwidth-restricted delivery channels. This project, now known as MPEG-4, will reach international standard status in December 1998 as ISO 14496. During the period since its inception, the scope of MPEG-4 has expanded; as well as traditional coding methods optimized for low-bitrate transmission, it also contains novel technology that enables the object-based description of synthetic content, audiovisual scenes, and synchronized of synthetic and natural content.

Among these new tools is the Binary Format for Scene Description, or *BIFS*. BIFS enables the concise specification of audiovisual scenes composed of partial pieces of content such as video clips, computer graphics, recorded sounds, and parametric sound synthesis. The part of BIFS controlling the composition of a sound scene is called *AudioBIFS*; AudioBIFS provides a unified framework for sound scenes that use streaming audio, interactive presentation, 3-D spatialization, and dynamic download of custom signal-processing effects.

Many of the BIFS concepts originate from the Virtual Reality Modeling Language (VRML) standard [4], but the audio toolset is built from a different philosophy. AudioBIFS contains significant advances in quality and flexibility over VRML audio.

In this tutorial, we present an in-depth examination of the capabilities of AudioBIFS, the relationship between AudioBIFS and the audio coding techniques in MPEG-4, the relationship between AudioBIFS and audio in VRML, and a discussion of the current status of the MPEG-4 standard.

2 MPEG-4 Audio and AudioBIFS

MPEG-4 is an *object-oriented* standard for multimedia. That is, the particular movie, television program, or interactive multimedia application being transmitted is conveyed as a collection of *media objects*. These media objects may be streaming video segments, streaming video "sprites", still images, streaming audio tracks, synthetic visual graphics, or sound-synthesis instructions. The coding method for each type of media object is specified in the MPEG-4 Audio and MPEG-4 Video standards; in a compliant MPEG-4 application, only MPEG-specified media objects may be contained in the bitstream.

As these elements are received in the terminal, they are composited together into an *audiovisual scene*. It is the scene, not the primitive media objects, which is shown to the person viewing the content. The instructions for composition are conveyed in the special BIFS format, which is specified in the MPEG-4 Systems standard [2].

In this tutorial, we focus only on the sound-compositing capabilities of MPEG-4. The sound coding tools are described in detail elsewhere, both in the technical literature [7][9], and in the MPEG-4 Audio standard itself [3], which is the official reference. There is an equivalent body of work on visual aspects of the standard that is also outside the scope of our presentation.

In this section, we give an overview of the MPEG-4 sound coding tools, discuss the sound-compositing philosophy in MPEG-4, and compare this philosophy with that of the popular VRML standard.

2.1 Sound coding in MPEG-4

There are two major groups of sound coding tools in MPEG-4: the *natural* tools [7] that allow recorded digital audio to be compressed and transmitted, and the *synthetic* tools [9] that allow parametric descriptions of sounds to be transmitted and used for real-time digital synthesis upon receipt.

The natural-audio tools enable the compressed transmission of speech and wideband audio at ranges from 2 kbps to 64 kbps per channel for high-quality multichannel sound. At the upper end of this range, psychoacoustic evaluation has demonstrated that the MPEG-4 algorithms [6] allow *perceptually transparent* coding. That is, even the most skilled listeners under rigorous testing conditions cannot distinguish the coded signal from the original.

There are two synthetic audio coders in MPEG-4. One provides an interface to text-to-speech systems; the other is a very general music-and-sound-effects synthesis toolset called *Structured Audio* (SA). The Structured Audio coder allows transmission of sound-synthesis algorithms in a new "Music V" language called SAOL ("Structured Audio Orchestra Language") [10], and banks of wavetables in a format called SASBF ("Structured Audio Sample Bank Format"), which was developed in collaboration with the MIDI Manufacturers Association. The algorithmic and wavetable synthesis capabilities may be used in parallel or in sequence in a synthetic soundtrack [12].

The transmission of sound through the real-time execution of synthesis algorithms is a recent development [1][14], and MPEG-4 is the first standard to make use of this capability. The music language SAOL is also important to the audio compositing tools; as we will describe in section 3.2, SAOL is used in MPEG-4 for downloading user-definable effects-processing algorithms. This convergence between structured audio and effects processing in MPEG-4 [11] is one of the elegant and important aspects of the standard.

2.2 Scene graph concepts

Both VRML and MPEG-4 BIFS rely on the *scene graph* to describe the organization of audiovisual material. We briefly outline the important concepts of scene-graph organization here to provide context for the material that follows.

A scene graph represents content as a set of hierarchically-related *nodes*. Each node in the visual scene graph represents a visual *object* (like a cube or image), a *property* of an object (like the appearance of a cube's faces), or a *transformation* of a part of the scene (like a rotation or scaling operation). By connecting multiple nodes together, object-based hierarchies are formed.

Each node has several *fields* that detail the properties of the object. For an object node like a cube, the fields give the size, local rotation, and color of the object. For a property node, the fields specify the particular properties such as the image to be texture-mapped to the cube. For a transform node, the fields specify the set of subsidiary nodes which are affected by the transformation, as well as the details of the transform.

Interaction with the scene graph is accomplished by an *event-routing* model. Objects in the scene may be programmed to transmit events when the user interacts with them. An event is *routed* from one object to another, where it triggers some function of the receiver. For example, a button object can be attached to a **TimeSensor** node. When this button is clicked, the **TimeSensor** sends a event that is routed to the **startTime** field of a sound-playing node, thus triggering the playback of a sound. The content author specifies the particular event mechanisms used in a scene as part of the scene graph.

2.3 Sound scenes in VRML

For the purpose of comparison, we provide a brief outline of the audio capabilities in the VRML (Virtual Reality Modeling Language) standard. VRML is a well-known computer-graphics description language that also has limited capabilities for the creation of interactive sound scenes. The VRML standard defines two nodes (**AudioClip** and **Sound**) that are used to incorporate sound objects into a virtual three-dimensional scene.

The **AudioClip** node specifies audio data to be used by **Sound** nodes; **AudioClip** can be thought of as a property node of the **Sound** node. The **AudioClip** node contains or points to a short PCM–encoded sound file. The standard also recommends, but does not require, that MIDI playback be supported.

AudioClip is a *time-dependent* VRML node, which means that it activates and deactivates itself at specified times; it may also be looped for continuous presentation. **AudioClip** does not *itself* play sound; it only *provides* sound material for use by one or more **Sound** nodes.

The **Sound** node specifies the location (spatial positioning) of a sound object in a VRML scene. The sound object is attached through a field called **source** and can be provided as either an **AudioClip** (for audio only) node or as a **MovieTexture** (for audio synchronized with video) node. The sound that results is located at a point in space and emits sound in a frequency-independent elliptical pattern.

In the last year, VRML implementations have become widely available. There are now several major companies providing VRML plugins for popular WWW browsers on a variety of platforms, and numerous authoring tools available. Serious content providers such as CNN are augmenting their sites with VRML content.

2.4 Sound scenes in MPEG-4

There are two main modes of operation that AudioBIFS, the MPEG-4 audio compositing toolset, is intended to support. We term them *virtual-reality* and *abstract-effects* compositing.

In virtual-reality compositing, the goal is to recreate a particular acoustic environment as accurately as possible. Sound should be presented spatially according to its location relative to the listener in a realistic manner; moving sounds should have a Doppler shift; distant sounds should be attenuated and low-pass filtered to simulate the absorptive properties of air; and sound sources should radiate sound unevenly, with a specific frequency-dependent directivity pattern. This type of scene composition is most suitable for "virtual world" applications and videogames, where the application goal is to immerse the user in a synthetic environment. The VRML sound model described in Section 2.3 embraces this philosophy, with fairly lenient requirements on how various sound properties must be realized in an implementation. The VRML sound nodes offer no functionality for such acoustical phenomena as the Doppler effect, frequency-dependent distance attenuation or more sophisticated directivity modeling.

In abstract-effects compositing, the goal is to provide content authors with a rich suite of tools from which artistic considerations can be used to choose the right effect for a given situation. As Scheirer [11] discusses in depth, the goal of sound designers for traditional media such as films, radio, and television is not to recreate a virtual acoustic environment (although this would be well within the capability of today's film studios), but to apply a body of knowledge regarding "what a film should sound like." Spatial effects are sometimes used, but often in a non-physically-realistic way; the same is true for the filters, reverberations, and other sound-processing techniques used to create various artistic effects that are more compelling than reality.

MPEG realized in the early development of the MPEG-4 sound compositing toolset that if the tools were to be useful to the traditional content community—always the primary audience of MPEG technology—then the abstract-effects composition model would need to be embraced in the final MPEG-4 standard. However, new content paradigms, game developers, and virtual-world designers demand high-quality sonification tools as well.

MPEG-4 AudioBIFS therefore integrates these two components into a single standard. Sound in MPEG-4 may be post-processed with arbitrary downloaded filters, reverberators, and other digital-audio effects; it may also be spatialized and physically modeled according to the simulated parameters of a virtual world. These two types of post-production may be freely combined in MPEG-4 audio scenes.

The overall integration of synthetic sound, natural sound, virtual-reality post-production, and abstract-effects post-production is termed *Synthetic/Natural Hybrid Coding* of audio, or SNHC audio. MPEG-4 is the first audio standard to enable significant SNHC functionality.

2.5 MPEG-4 Versions

MPEG-4 is being standardized in two *versions*. Version 1 will be completed in October 1998 and published in December 1998; Version 2 will follow a year later. Version 2 will be completely backward-compatible with Version 1 and will provide extensions in certain directions, such as advanced environmental auralization, Java capability, and a more powerful fixed-storage format.

This tutorial focuses mainly on the description of AudioBIFS capabilities in Version 1 (and thus is applicable to both versions); the discussion of Version 2 capabilities is confined to Section 4. Unless specifically mentioned otherwise, any general discussion of MPEG-4 applies to both versions.

3 AudioBIFS Version 1

In this section, we describe the technical operation of the audio scene capabilities of MPEG-4. We begin with a high-level discussion of a model of the overall audio system, and then proceed to list and explain each of the nodes that comprise AudioBIFS.

3.1 The MPEG-4 Audio System

A schematic diagram of the overall audio system in MPEG-4 is shown in *Figure 1* and may be a useful reference during the discussion to follow.

Sound is conveyed in the MPEG-4 bitstream as several *elementary streams* which contain coded audio in the formats described in Section 2.1. There are four elementary streams in the sound scene in *Figure 1*. Each of these elementary streams contains a *primitive media object*, which in the case of audio is a single-channel or multichannel sound that will be composited into the overall scene. In *Figure 1*, the T/F coded stream decodes into a stereo sound, and the other streams into monophonic sounds. The different primitive audio objects may each make use of a different audio decoder, and decoders may be used multiple times in the same scene.

The multiple elementary streams are conveyed together in a multiplexed representation. Multiple multiplexed streams may be transmitted from multiple servers to a single MPEG-4 receiver, or *terminal*. There are two multiplexed MPEG-4 bitstreams shown in *Figure 1*; each originates from a different server. Encoded video content can also be multiplexed into the same MPEG-4 bitstreams. As they are received in the MPEG-4 terminal, the MPEG-4 bitstreams are

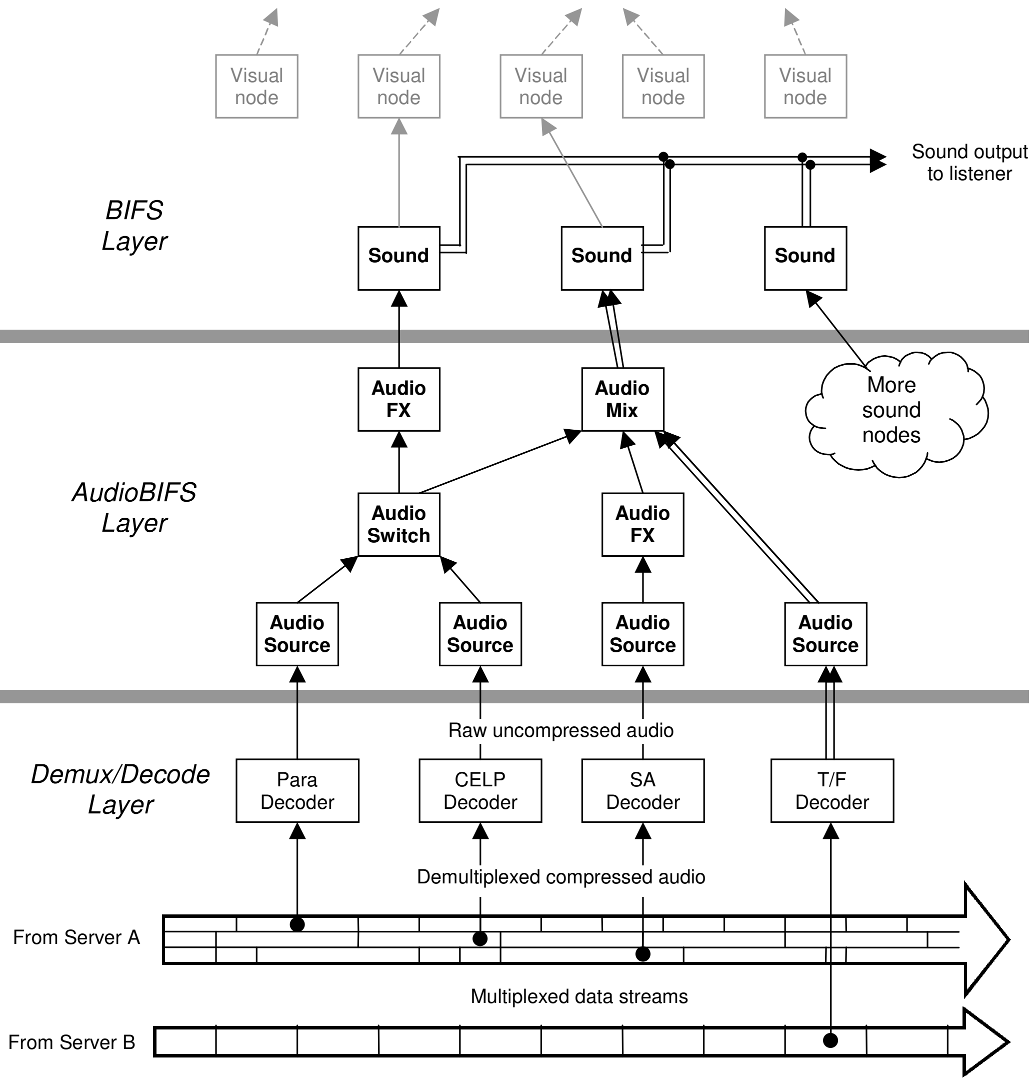

Figure 1. The MPEG-4 audio system. Sound is conveyed in several data streams (bottom); the streams are demultiplexed and the resulting compressed audio data is presented to the decoders. The decoders uncompress the audio data and produce sound streams; of these, some are mono and some are multichannel. The sound streams flow through the AudioBIFS scene graph (middle), where they are mixed and transformed. Ultimately, they arrive at the BIFS layer (top), where **Sound** nodes describe the spatial location of the sounds in relation to the rest of the visual scene. The transformed sounds are spatialized and mixed according to their locations and that of the listener.

demultiplexed, and each primitive media object is decoded. The resulting sounds are not played directly, but rather made available for scene compositing using AudioBIFS.

Also transmitted in the multiplexed MPEG-4 bitstream is the BIFS scene graph itself. BIFS—and AudioBIFS—are simply parts of the content like the media objects themselves; there is nothing "hardwired" about the scene graph in MPEG-4. Content developers have wide flexibility to use BIFS in a variety of ways. In Figure 1, BIFS and AudioBIFS parts of the scene graph are separated for clarity, but there is no technical distinction between them.

Like the rest of the BIFS capabilities, AudioBIFS is comprised of a number of nodes which are interlinked in a scene graph. However, the concept of the AudioBIFS scene graph is somewhat different; it is termed an *audio subgraph*.

Whereas the main (visual) scene graph represents the position and orientation of visual objects in presentation space, and their properties such as color, texture, and layering, an audio subgraph represents a signal-flow graph describing digital-signal-processing manipulations. Sounds flow in from MPEG-4 audio decoders at the bottom of the scene graph; each "child" node presents its results from processing to one or more "parent" nodes. Through this chain of processing, sound streams eventually arrive at the top of the audio subgraph. The "intermediate results" in the middle of the manipulation process are not sounds to be played to the user; only the result of the processing at the top of each audio subgraph is presented.
Audio processing using the scene graph and AudioBIFS is tightly coupled with real-time audio decoding using the MPEG-4 audio tools as described above. The **AudioSource** node (see Section 3.2) connects primitive audio objects, produced by the audio decoders, to the scene graph; sound begins flowing into the scene at each of these nodes. At the top, each audio subgraph is rooted in a **Sound** node, which allows sounds to be attached to visual objects in the world and dynamically moved in response to user interaction. A "sound source" in the scene (that is, from the user's point of view), is the post-processed combination of sounds that attaches to a **Sound** node. Many audio subgraphs may be present in any audio-visual scene; every sound does not have to be attached to a visual object. In Figure 1, there are three sound objects, with the audio subgraph fully expanded for two of them. The two **Sound** nodes with expanded subgraphs are associated with visual objects; the third (the rightmost) does not have any visual correlate in the scene.

The MPEG-4 Systems standard contains specifications for the resampling, buffering, and synchronization of sound in AudioBIFS. Although we will not discuss these aspects in detail, for each of the nodes described in Section 3.2 there are precise instructions for the associated resampling and buffering requirements written in the standard. These aspects of MPEG-4 are *normative*; that is, every MPEG-4 terminal must implement them the same way. This makes the behavior of an MPEG-4 terminal highly predictable to content developers.

3.2 AudioBIFS nodes

There are eight BIFS nodes which comprise the AudioBIFS toolset; in addition, a few of the general-purpose BIFS nodes have sound behavior associated with them. This section discusses each of the AudioBIFS nodes, giving their syntax and semantics and describing their function in an audio scene.

As described in Section 2.2, each node has several *fields* which specify the parameters of operation of the node. In MPEG-4 BIFS, these fields and their operating range are carefully quantized and transmitted in binary data format for maximum compression of the scene graph information; here, we give a more conceptual description using the nonnormative textual names for the fields.

AudioSource
The **AudioSource** node is the point of connection between real-time streaming audio and the AudioBIFS scene. The **AudioSource** node attaches an audio decoder, of one of the types specified in the MPEG-4 audio standard, to the scene graph; audio flows out of this node.

The **AudioSource** node has *time-sensitive* fields (**startTime** and **stopTime**) which allow the playback of sound data to be started, stopped, paused and rewound, when the transmission scenario allows such functions. Fields named **pitch** and **speed** allow the playback pitch and speed to be controlled for decoders which allow this functionality (the Parametric and Structured Audio decoders). A field named **numChan** specifies how many channels of audio, from those produced by the decoder, should be used. A field called **phaseGroup** allows the content developer to specify *phase relationships* among the several channels of audio produced by the decoder – that is to say, to declare that, from a seven-channel decoded stream, the first two channels are a stereo pair, the next four are a quadraphonic set unrelated to the first two, and the final channel is not related to any of the first six. This information is important for executing effects on multichannel sets and producing spatial audio.

AudioMix

The **AudioMix** node allows M channels of input sound to be mixed into N channels of output sound through the use of a mixing matrix. The M channels of input may be all from the same child source, all from different children, or any desired combination. If the child sound sources are at different sampling rates, all of the input data is resampled to the highest of the sampling rates of the children before mixing.

The fields of the **AudioMix** node are **children**, which attaches the child AudioBIFS nodes; **matrix**, which contains the mixing matrix; **numInputs**, containing the number of input channels (needed so that the shape of **matrix** is known) and **numChan** and **phaseGroup**, which are as in **AudioSource** – they identify the grouping parameters of this node.

AudioSwitch

The **AudioSwitch** node allows N channels of output to be taken as a subset of M channels of input, where M ≤ N. It is equivalent, but easier to compute, to an **AudioMix** node where M ≤ N and all matrix values are 0 or 1. This node allows efficient selection of certain channels, perhaps on a language-dependent basis. As with **AudioMix**, input sounds are resampled to the same rate before selection occurs.

The fields of the **AudioSwitch** node are **children**, which attaches the child nodes; **whichChoice**, which specifies the particular subset of channels to pass through; and **numChan** and **phaseGroup**, which are as in **AudioSource**.

AudioDelay

The **AudioDelay** node allows several channels of audio to be delayed by a specified amount of time, enabling small shifts in timing for media synchronization. As with **AudioMix** and **AudioSwitch**, if the input channels are not all at the same sampling rate, they are resampled before delay is computed.

The fields of **AudioDelay** are **children**, which attaches the child nodes; **delay**, which specifies the amount of time delay; and **numChan** and **phaseGroup**, which are as in **AudioSource**.

AudioFX

The **AudioFX** node allows the dynamic download of custom signal-processing effects to apply to several channels of input sound. A special sound-processing language called SAOL [10] is standardized in the audio toolset; this language allows arbitrary effects-processing algorithms to be transmitted in the scene graph.

The use of SAOL to transmit audio effects means that MPEG does not have to standardize the "best" artificial reverberation algorithm (for example), but also that content developers do not have to rely on terminal implementors and trust in the quality of the algorithms present in an unknown terminal. Since the execution method of SAOL algorithms is precisely specified, the sound designer has control over exactly which reverberation algorithm (for example) is used in a scene. If a reverb with particular properties is desired, the content author transmits it in the bitstream; its use is then guaranteed.

SAOL has many useful algorithms built in, such as comb and allpass filters, multitap fractional delay lines, digital FIR and IIR filters, chorus and flanging operations, a time-shifter, and multiband compression and equalization. It is arbitrarily extensible to include new algorithms; SAOL is not a "suite of digital effects" but a *language* for describing synthesis and digital-effects algorithms. Any algorithm for digital sound manipulation can be written in SAOL.

As with other AudioBIFS nodes, multiple child nodes may be attached to the **AudioFX** node; if these children are running at different sampling rates, the input data is resampled before being presented to the SAOL signal-processing algorithms. The **phaseGroup** fields of the children are made available to the SAOL orchestra, and the algorithms in SAOL may thereby depend on the particular phase-relationships of the inputs. The position of the **Sound** node in the overall scene, as well as the position of the listener (see below), are also made available to the **AudioFX** node, so that the effects-processing may also depend on the spatial locations (relative or absolute) of the listener and source.

The fields of the **AudioFX** node are **children**, which attaches the child nodes; **orch**, which specifies the SAOL orchestra; **score**, which specifies the SASL script, if any is needed; **params**, which allows scene-graph-level interaction control of effects (see Section 3.3) and **numChan** and **phaseGroup**, which are as in the other nodes.

AudioClip

The **AudioClip** node allows a segment of audio to be excerpted from a stream, and then triggered and played back interactively. Unlike the VRML node of the same name, the **AudioClip** node does not itself contain any sound data. Instead, it records the first *n* seconds of sound produced by its children. It captures this sound into an internal buffer. Then, it may later be triggered interactively (see the section on interaction below) to play that sound back.

This function is most useful for "auditory icons" such as feedback to button-presses. It is impossible to make streaming audio provide this sort of audio feedback, since the stream is (at least from moment to moment) independent of user interaction. The back-channel capabilities of MPEG-4 are not intended to allow the rapid response required for audio feedback.

The fields of **AudioClip** are **children**, which attaches the child nodes; **length**, which specifies how much sound to record; **startTime** and **stopTime**, which control interactive playback of the sound; and **numChan** and **phaseGroup**, which are as in the other nodes.

There is a special function of **AudioClip** which allows it to cache samples for sampling synthesis in the Structured Audio decoder. A special field of the **AudioSource** node named **children** may only be used when the **AudioSource** node is attached to a Structured Audio decoder; the **children** must all be **AudioClip** nodes. The sounds recorded in the **AudioClip** nodes are made available to the Structured Audio decoder attached to the **AudioSource** for use in the synthesis process.

This technique allows perceptual compression to be applied to sound samples, which can greatly reduce the size of bitstreams using sampling synthesis.

Sound
The semantics of the **Sound** node in MPEG-4 are similar to that of the VRML standard, i.e., the sound attenuation region (fields **direction**, **minBack**, **maxBack**, **minFront**, **maxFront**) and spatialization (fields **location**, **spatialize**) are defined in the same way as in the VRML standard to create an elliptical model of attenuation. This node is used in MPEG-4 to attach sound to 3-D audio scenes.

In contrast to VRML, where the **Sound** node accepts raw sound samples directly and no intermediate processing is done, in MPEG-4 any of the AudioBIFS nodes may be attached. Thus, if an **AudioSource** node is the child node of the **Sound** node, the sound as transmitted in the bitstream is added to the sound scene; however, if a more complex audio scene graph is beneath the **Sound** node, the mixed or effects-processed sound is presented. The spatialization effects may be added to sound whether or not complex processing has taken place; however, spatialization is not applied to multiple channels of sound that have phase interactions among them (as specified using the **phaseGroup** field of the child).

All of the spatial and non-spatial sounds produced by the **Sound** node(s) in the scene are summed and presented to the listener. The methods of synthesizing spatial cues and presenting them to the listener are not normative in MPEG-4.

Sound2D
The **Sound2D** node is used to attach sound to 2D BIFS Scenes. The source of audio is the same as in the **Sound** node, with the similar possibility to route the audio through an audio subtree. The spatialization in this node is carried out in a 2D plane.

Group (and other grouping nodes)
Several general BIFS nodes allow multiple nodes to be grouped together; these include **Group**, **Group2D**, **Transform**, and **Transform2D**. When grouping nodes are used in the general scene to group together multiple **Sound** nodes, the sounds represented in each are summed together. When nodes such as **Transform** are used, they modify the location and direction of the (spatially presented) sounds grouped under them. Thus, the **Transform** node can be used conveniently to move or rotate a group of sound objects in a scene.

ListeningPoint
This node allows the user to set the listening point in a scene to be different from the viewpoint, which is specified in the visual scene. The listening point is the position which the spatial positions of sources are calculated relative to. By default (if no **ListeningPoint** node is used), these are the same.

TermCap
The **TermCap** node is not an AudioBIFS node specifically, but provides capabilities which are useful in creating terminal-adaptive scenes. The **TermCap** node allows the scene graph to query the terminal on which it is running, to discover hardware and performance properties of that terminal. For example, in the audio case, **TermCap** may be used to determine the ambient signal-to-noise ratio of the environment. The result can be used to control "switching" between different parts of the scene graph, so that (for example) a compressor is applied in a noisy environment such as an automobile, but not in a quiet environment such as a listening room.

Other audio-pertinent resources that may be queried with the **TermCap** node include: the number and configuration of loudspeakers, the maximum output sampling rate of the terminal, and the level of sophistication of 3-D audio functionality available.

3.3 Interactive Audio Scenes

The AudioBIFS nodes described in the previous section may be used in static presentations, in which all of the parameters are downloaded in a fixed scene graph and a single piece of content is played back. Facilities in MPEG-4 also allow the construction of sophisticated *interactive* content.

Most of the fields in the AudioBIFS nodes are termed *exposed* fields; that is, they may be changed by other parts of the scene graph. These changes may be driven by user interaction with an interface or other external commands. Thus, if a user interface is created which allows the listener to manipulate the values in the **matrix** field of an **AudioMix** node, the result is to give the listener control over the "fader levels" in post-production.

Each of the important control parameters is exposed for each node; the **params** field of the **AudioFX** node allows scene-level interaction to control parameters of downloaded effects-processing algorithms. The semantics of the **params** field change from application to application.

4 AudioBIFS Version 2

In this section we describe the features proposed for AudioBIFS in MPEG-4 version 2, which will become an ISO standard in January 2000. The AudioBIFS extensions to the first version of the MPEG-4 standard concern audio environment modeling in a manner more natural than is possible in the current BIFS and VRML standards. We briefly discuss the concepts behind virtual audio environments, and then give an introduction to their creation in Version 2 MPEG-4 audio scenes.

4.1 Modeling of Acoustic Environments

By *modeling of acoustic environments* we mean processing sound so that the resulting acoustic response corresponds to the visual scene. This involves modeling individual sound reflections off the walls, modeling sound propagation through objects, simulating air absorption, and rendering of late diffuse reverberation, in addition to the 3D-spatialization of the sound location. This kind of "environmental spatialization" is often referred to as *auralization* [5].

Auralization consists of rendering the whole of sound propagation from the sound source to the listener. Therefore, it involves not only the properties of sound transmission in the acoustic medium, but also direction-dependent sound radiation at the source and the arrival of sound at both ears of the listener. This *source-medium-receiver* model is the basis of the current draft of a more natural-sounding virtual-reality rendering of MPEG-4 scenes. Other examples of such systems, in which models of room acoustics are dynamically computed according to sound source and listener positions in a computer-modeled room, can be found in the literature [8].

4.2 Extensions to AudioBIFS in MPEG-4 Version 2

In version 1 of the MPEG-4 standard, as in VRML, the virtual-reality sound-source model only provides techniques for placing sound sources in 3D space and coarse simulation of sound source directivity by the elliptical attenuation model.

To improve this model, the spectral content of the sound should change as a function of distance; this occurs in natural environments because of the low-pass filtering effect of air absorption. Another improvement is to enable more flexible simulation of the frequency-dependent radiation patterns of real sound sources. For example, a brass instrument has more high-frequency spectral content in front than behind. Finally, the current BIFS model does not take effects of the environment and the medium into account. Among these are the occlusion of sound by physical objects and the Doppler effect caused by the coupling of the sound propagation delay to the relative movement of the sound source and the listener.

For the second phase of the standard, three new nodes have been proposed for advanced auralization of the sound (see [13]). Their textual names are **AcousticScene**, **DirectiveSound** and **AcousticMaterial**; together they can be used to form natural-sounding audio environments.

AcousticScene and **AcousticMaterial**

The **AcousticScene** node is a grouping node that is used to bind together an entire auralization process. It may be used to group together any audio or visual nodes, but additionally it has fields that define common properties for the acoustic space. The field **reverbtime** defines the frequency-dependent late reverberation time for its children **DirectiveSound** nodes. Boolean fields **useAirabs** and **useAttenuation** allows the attenuation mode of the child sounds of the child sounds to be selected—that is, whether distance-dependent lowpass filtering is applied, or distance-dependent attenuation is rendered, or the sounds are *ambient* (have a constant level independent of the distance from the sound source).

AcousticScene may also contain geometrical descriptions of acoustically responsive surfaces in its children. This way it is possible, for example, to build a room model from walls with reflective properties, or to provide information about the attenuation caused by a surface appearing between the sound source and the listener. The **AcousticMaterial** node attaches the needed reflection and transmission properties to polygonal surfaces.

A BIFS scene may contain several **AcousticScenes**, and thus it is possible to have several rooms (or more generally regions) with different acoustic responses in a single virtual environment.

DirectiveSound

The **DirectiveSound** node enables the flexible definition of frequency-dependent directivity modeling of sound sources. The **directivity** field of this node specifies the frequency-dependent gain as a function of azimuth angle around the source. The sound attenuation is defined by a -60 dB attenuation distance, within which the sound is linearly attenuated in the dB scale; outside it is not heard. The strength of Doppler effect is defined by the value of the **speedOfSound** field specific to each **DirectiveSound** node.

5 Conclusions

We have described essential concepts of AudioBIFS, the MPEG-4 standard for effects processing and audio scene description. AudioBIFS is a powerful, flexible format which serves the needs of virtual-world builders and traditional media developers alike. The tools and techniques in MPEG-4 Structured Audio and Version 1 of AudioBIFS have been developed by the MIT Media Laboratory and donated to ISO and the audio community at large. The Media Laboratory maintains no patent rights or proprietary control over the direction of MPEG-4. A public-domain source code implementation of the SAOL and AudioBIFS tools is available from the web site <http://sound.media.mit.edu/mpeg4>.

It is our hope that acceptance and implementation of the MPEG-4 audio standard will help to drive forward the marketplace for advanced digital-audio technology on personal computers. As such, we encourage collaboration and comment from any interested developers; we are happy to share any materials that make such collaboration fruitful.

It is important to note that this tutorial does not itself represent a standard or the views of the standardization body ISO/IEC JTC1/SC29/WG11, but only the opinions of three individuals involved in technical aspects of the standardization process. Certain elements described herein, particularly those pertaining to Version 2 of MPEG-4, may change between the time of this writing and final standardization. The MPEG-4 process is an open standards process; suggestions for improvement are welcome from any party at any time.

References

[1] Casey, M., and Smaragdis, P. 1996. “Netsound" Proc. Int. Comp. Music Conf, Hong Kong.

[2] Eleftheriadis, A., Herpel, C., Rajan, G., and Ward, L., Editors. 1998. ISO 14496-1 (MPEG-4 Systems) Final Committee Draft. MPEG Document W2201, Tokyo.

[3] Grill, B., Edler, B., Kaneko, I., Lee, Y., Nishiguchi, M., Scheirer, E., and Väänänen, M., Editors. 1998. ISO 14496-3 (MPEG-4 Audio) Final Committee Draft. MPEG Document W2203, Tokyo.

[4] ISO 14472. 1997. Virtual Reality Modeling Language (VRML). International Organisation for Standardisation, Geneva.

[5] Kleiner, M., Dalenbäck, B.-I., and Svensson, P. 1993. "Auralization -- an Overview", J. Audio Eng. Soc., 41(11) pp. 861-875.

[6] Meares, D., Watanabe, K., and Scheirer, E. Report on the MPEG-2 Advanced Audio Coding Stereo Verification Tests. MPEG Document W2003, San Jose, CA, USA.

[7] Quackenbush, Q. 1998. “Coding of Natural Audio in MPEG-4”.Proc. IEEE ICASSP, Seattle.

[8] Savioja, L., Huopaniemi, J., Lokki, T., and Väänänen, R. 1997. “Virtual environment simulation - Advances in the DIVA project.” Proc. Int. Conf. Auditory Display (ICAD'97), Palo Alto, California, USA.

[9] Scheirer, E. D. 1998. “The MPEG-4 Structured Audio standard”. Proc. IEEE ICASSP, Seattle.

[10] Scheirer, E. D. 1998. “The MPEG-4 Structured Audio Orchestra Language”. Proc. ICMC, Ann Arbor, MI, USA.

[11] Scheirer, E. D. In press. “Structured Audio and effects processing in the MPEG-4 multimedia standard". To appear in ACM Multimedia Sys. J.

[12] Schreider, E.D. In press. “Structured Audio and effects processing in the MPEG-4 multimedia standard”. To appear in ACM Multimedi Sys.J.

[13] Väänänen, R., Singer, D., Belknap, W., Shamoon, T., Herpel, C. 1998. ISO 14496-1 (MPEG-4 Systems version 2) Verification Model. MPEG Document W2359, Dublin.

[14] Vercoe, B. L, Gardner, W. G., and Scheirer, E. D. 1998. “Structured audio: The creation, transmission, and rendering of parametric sound representations”. Proc. IEEE 86(5) pp. 922-940.