# System Architecture for Synthetic/Natural Hybrid Coding and Some Experiments

Jiro Katto, *Member, IEEE*, and Mutsumi Ohta

***Abstract*—This paper presents a system architecture for synthetic/natural hybrid coding toward future visual services. Scene-description capability, terminal architecture, and network architecture are discussed by taking account of recent standardization activities: MPEG, VRML, ITU-T, and IETF. A consistent strategy to integrate scene-description capability and streaming technologies is demonstrated. Experimental results are also shown, in which synthetic/natural integration is successfully carried out.**

***Index Terms*— MPEG-4 synthetic/natural hybrid coding (SNHC), scene description, streaming, system architecture.**

## I. Introduction

RECENTLY, both digital video and computer graphics have evolved rapidly. Interaction capabilities, previously exploited in computer graphics, are now emphasized in digital video [1], [2]. Compression and streaming technologies that were mainly developed in digital video are now focused on computer graphics [3].

MPEG-4 specifies synthetic/natural hybrid coding (SNHC) in their compression algorithms (facial animation and mesh compression) and their binary format for scene (BIFS) description [1], [2]. They collaborate with virtual reality modeling language (VRML) [3], which specifies a file format for describing three-dimensional (3-D) interactive worlds and objects exchanged on the Internet. Streaming technologies are mainly specified in the ITU-T H series, such as H.323 for packet networks including the Internet [4] and H.324 for general switched telephone networks and mobile channels [5]. The Internet Engineering Tasking Force (IETF) also specifies real-time transport protocol (RTP) [6], which is used for streaming applications on the Internet and composes a part of the ITU-T H.323 standard.

However, there have been few discussions about system architecture for the synthetic/natural hybrid coding. MPEG-4 provides a framework but does not specify actual transport protocols. ITU-T and IETF present various protocols but do not give specific considerations on the hybrid coding. Such inconsistency isolates the hybrid coding from many practical systems and causes ambiguities in its implementation.

Manuscript received October 1, 1997; revised November 1, 1998. This paper was recommended by Guest Editors H. H. Chen, T. Ebrahimi, G. Rajan, C. Horne, P. K. Doenges, and L. Chiariglione.

J. Katto was with Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA. He is now with C&C Media Laboratories, NEC Corp., Kawasaki-shi, Kanagawa 216 Japan.

M. Ohta is with C&C Media Laboratories, NEC Corp., Kawasaki-shi, Kanagawa 216 Japan.

Publisher Item Identifier S 1051-8215(99)02333-2.

This paper, therefore, tries to create consistent system architecture that handles both the scene-description capability and the streaming technologies in a unified manner. Transmission of scene-description formats is investigated to incorporate synthetic objects (i.e., computer graphics) into the digital video world and to provide interaction capability. Terminal architecture with a media synchronization mechanism and network architecture considering existing transport protocols are also discussed, followed by considerations on technological gaps between digital video and computer graphics. Last, several experimental results are presented.

## II. Scene-Description Capability

### A. Paradigms

Fig. 1(a) and (b) summarizes classical paradigms for digital video and computer graphics assuming network transmission. Digital video has evolved with compression and streaming technologies. Video signals are encoded and multiplexed with audio at the transmitter side. They are transferred into a network consecutively, then demultiplexed and decoded at the receiver side. Streaming mechanisms play an important role to guarantee synchronized presentation of video and audio [7].

On the other hand, computer graphics has evolved with technologies for synthetic object modeling and rendering. They are stored in the form of rendering programs or by a specified data-base format on a server. They are exchanged through a network, and a scene is generated incorporating synthetic objects at the receiver side. In general, animation is supported by a script or a byte code instead of streaming mechanisms. Interaction capability, thanks to the scene-description information, has been emphasized there [8], [9].

Integration of digital video and computer graphics has been carried out in several ways. An example is a virtual studio, in which composition is done at the transmitter side and the result is encoded as a single video source. An advantage of this approach is that the current paradigm of digital video does not have to be changed. A disadvantage, however, is elimination in the receiver of interaction capability, which is an important property of computer graphics.

Another example is VRML, in which video sources are downloaded along with a scene-description format (i.e., a VRML file) beforehand. An advantage is that interaction capability is fully supported since the scene information is transferred to the receiver. Animation is carried out by scripts embedded in a VRML file. However, the receiver has to wait

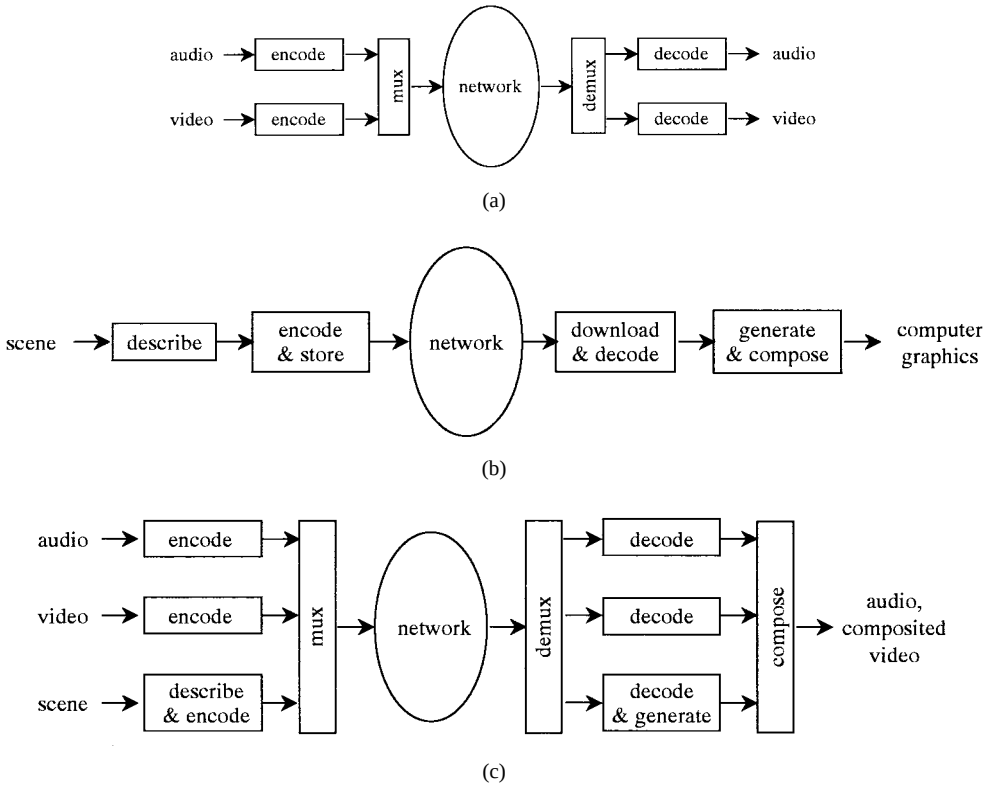



Fig. 1. Paradigms for digital video and computer graphics: (a) digital video, (b) computer graphics, and (c) integration of digital video and computer graphics.

for a long time for huge media sources to be downloaded completely due to the lack of streaming mechanisms.

Fig. 1(c) provides a new paradigm for this integration purpose. A scene-description format is attached to video/audio streams. They are encoded, multiplexed, and transferred to the receiver, and a scene is generated consecutively. Video and audio streams are handled similarly to the case of Fig. 1(a). Interaction capability is preserved since the scene-description data is transferred, similar to Fig. 1(b). A technical challenge is that both decoding and rendering have to be carried out continuously within an expected frame rate.

### *B. A Scene-Description Format for Streaming Media*

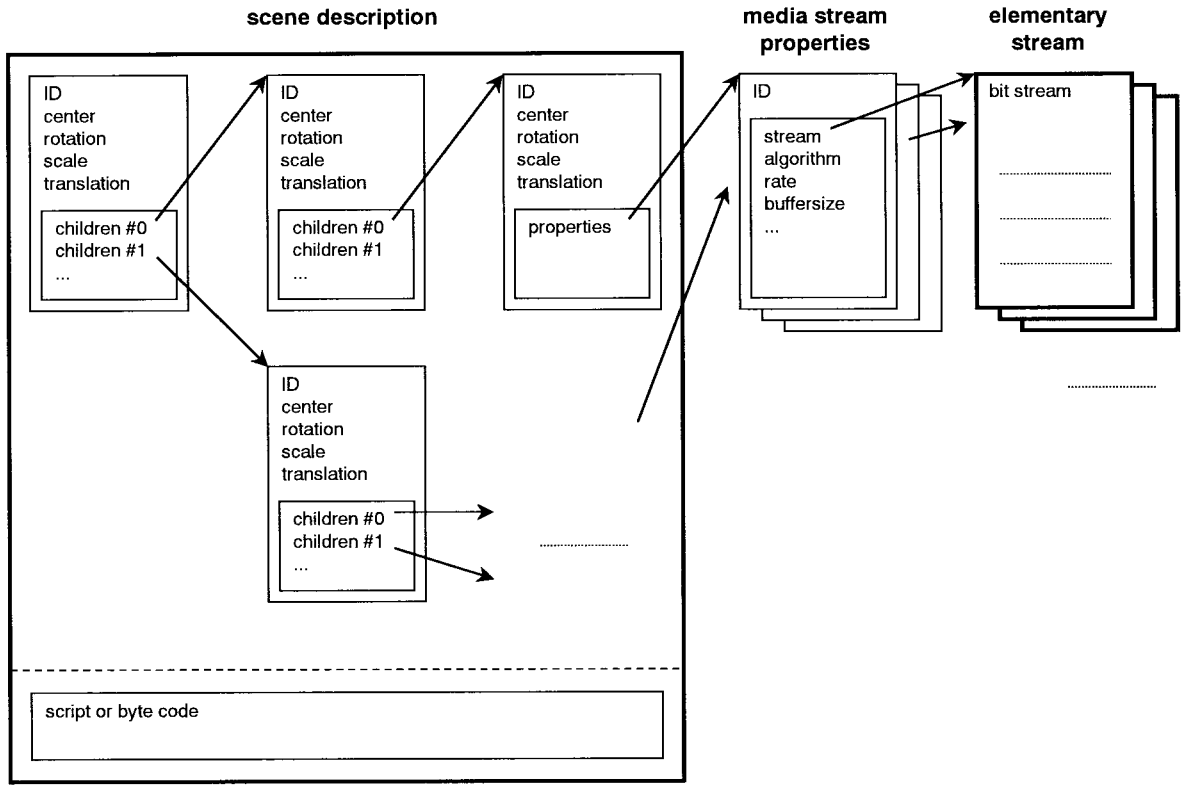
Scene-description formats, such as HTML and VRML, specify spatial/temporal relationships of objects inside a scene. In the case of HTML, spatial relationship is determined implicitly according to appearance order. In the case of VRML, it is decided explicitly by the specified fields: center, translation, rotation, and scale. In both cases, temporal relationship is presented by attached byte codes or scripts, known as Java or ECMAScript, which contribute to interaction and animation.

Standards for streaming applications, such as MPEG and ITU-T H series, specify compression algorithms and related control (system) parameters. The compression results are called elementary streams. The control parameters are used to manage elementary streams: multiplexing, buffer control, synchronization, and so on. They may be signaled on a different channel (i.e., control channel), compose an independent elementary stream, be attached to an access unit that is a piece of an elementary stream (e.g., time stamps for a video frame), or be transformed to other parameters according to transport protocols.

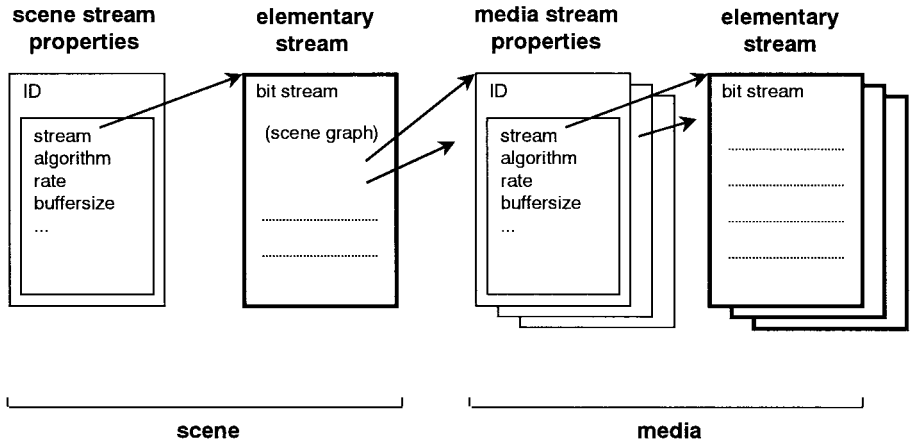
Fig. 2(a) demonstrates a general method to incorporate streaming media into a scene-description format. It consists of scene description, media stream properties, and elementary streams. The scene-description part is composed of a scene graph and an optional program (script or byte code). A scene is described in a hierarchical manner by cascading nodes containing VRML-like parameters: center, translation, rotation, and scale. The node is a minimal unit of the scene graph, to which a unique identification number is assigned. The node is typically classified as a group node and a leaf node; the former has children but the latter does not. The group node is used to group and reuse children inside a scene. The leaf node is used to specify detailed properties of synthetic objects, as well as video textures and audio sources, which are mapped onto objects. In VRML, the former corresponds to Transform and Group nodes, and the latter to Sphere, Box, IndexedFace-Set, MovieTexture, AudioClip, and so on.

The media stream properties are containers of decoder control parameters. The parameters indicate compression algorithm, bit rates, buffer sizes, random access information, pointers to the elementary streams (e.g., URL's), and so on. A unique identification number is assigned to the container and is used to associate the media to the scene description.

Fig. 3(a) shows an example of scene-description syntax. The *ID* provides a unique identifier to a node in a scene graph, corresponding to that in the scene description in Fig. 2(a). The *type* informs a node type, e.g., Transform or Sphere in the case of VRML. The node type determines subsequent




(a)




(b)

Fig. 2. Scene description incorporating streaming applications: (a) basic structure and (b) elementary stream of scene description.

data structure according to the node definition. The *count1* and *count2* present the number of fields and the number of children nodes, respectively. Field values are provided along with a fieldID, which identifies a field complying with the node definition. Children nodes are specified by ID values, which are also defined in their own node descriptions. A nesting structure like VRML is not assumed here, but it is straightforwardly done by replacing the ID fields by node descriptions of children.

### C. Scene Update

Scene description itself can be considered as an elementary stream by defining its update syntax. Fig. 2(b) depicts this concept, in which an elementary stream of a scene graph and its control parameters are provided. Based on this mechanism, another animation mechanism analogous to the digital video paradigm (I/P-pictures) is provided.

Fig. 3(b) shows the corresponding syntax for scene update, to which time stamps may be attached when necessary. The ID indicates a node in a scene graph to be updated. The mode determines an action taken by the update command to the indicated node. The count specifies the number of subsequent components, of which data structure is determined by the mode field.

Currently, four modes are considered: *replace*, *append*, *insert*, and *remove*. The former two are used to change field

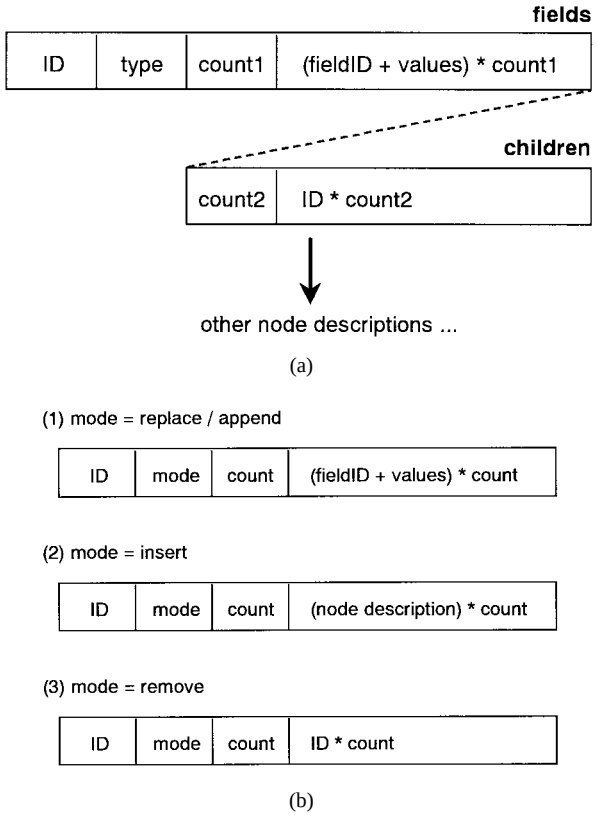



Fig. 3. Syntax for scene description: (a) node description and (b) scene update.

values in the selected node. Replace sets new values, and append adds values to previous ones. For each of them, a fieldID is attached to identify a field in the node to be updated. Insert and remove are used to manage children nodes. The former inserts new children nodes. When the inserted child node has its own children, corresponding insert commands should be issued afterwards. Remove erases children nodes, which are identified by their ID values. It is assumed that all of the children are removed when their parent node is removed.

Special cases have to be taken into account when insert and remove commands are applied to a node associated with media streams. In the case of insertion, media stream properties have to be informed along with the insert command. Insertion is carried out only after necessary operations (decoder initialization and channel opening) are completed. In the case of removal, it is applied only after corresponding channels are closed successfully. These problems are related to the control protocols with negotiation such as ITU-T H.245 [10].

### D. Relationship to Existing Standards

This section utilizes a generalized form to discuss scene-description formats necessary to incorporate synthetic objects and to provide interaction capability. It follows a scene-graph concept adopted by VRML. Therefore, it can be applied to VRML and can be used as its streaming extension. In this case, scene description is done in a nesting manner, and node identification is executed through its naming mechanism.

The MPEG-4 system provides BIFS and object descriptors, which correspond to the scene-description format and the scene/media stream properties discussed in this section, respectively. Differences lie in the compactness of the proposed method. The current MPEG-4 specification provides more functionalities that have not been considered in this paper. For example, MPEG-4 provides two update mechanisms: BIFS-Update, which happens at a given time instant, and BIFS-Anim for continuous change of node parameters. This separation contributes to saving bits in addition to their functional efficiency.

## III. Terminal Architecture

### A. Browser (Decoder) Configuration

A browser receives scene-description data along with video/audio streams. Control parameters are also attached for the decoder/compositor control. Note that the term "browser" is used instead of the classical term, decoder, since the decoder is a part of the browser, as shown in Fig. 4.

Fig. 4 demonstrates an example of browser configuration. Streams are separated by a demultiplexer into video, audio, scene-description, and control data. They are stored in buffers and decoded. The results are stored in memories, and their composition begins. A scene with synthetic objects is generated according to the scene-description format. Video and audio signals are then composed into a scene, and the result is presented to a viewer. The viewer may try to interact with the scene. The interaction, such as viewpoint movement, is reflected in subsequent scenes produced by the scene generator. A controller indicates the start of decoding and composition to decoders and a compositor according to control parameters, respectively.

Stream properties of the scene and media have to be communicated to the browser when starting a session. They are used to initialize decoders and to open necessary channels. When a new object to which new media streams are associated is inserted during a session, a similar procedure is applied.

Buffers store compressed data. Their recommended sizes should be specified through the media stream properties in order to avoid buffer overflow and underflow. Memories store decoding results. Their sizes may be implicitly determined by picture sizes or audio frame periods that are contained in the stream properties or elementary streams themselves. However, when variable delay networks are assumed and a predownload mechanism of elementary streams is required, both buffer and memory sizes may be controlled by a server through explicit indications.

### B. Server (Encoder) Configuration

An example of a server (encoder) configuration is presented in Fig. 5. It takes the style of an authoring tool, in which user interaction is applied to a scene generator and the change is transferred to an encoder.

The handling of video and audio streams is carried out in a similar way to the classical digital video paradigm. The streams are encoded and put into buffers. They are locally decoded and stored in memories. The data in the memories are used for encoding of subsequent access units and for

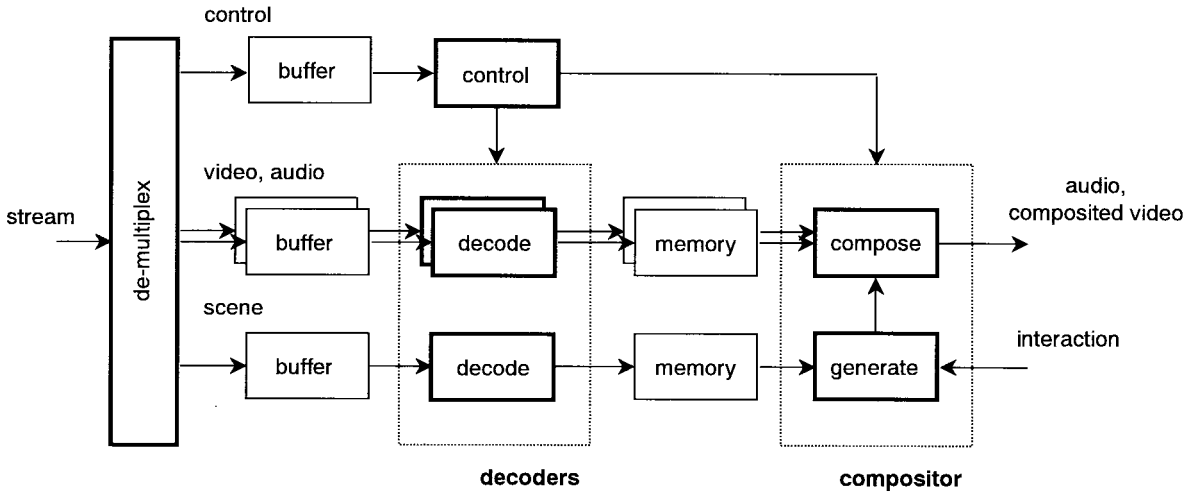



Fig. 4. Browser configuration.

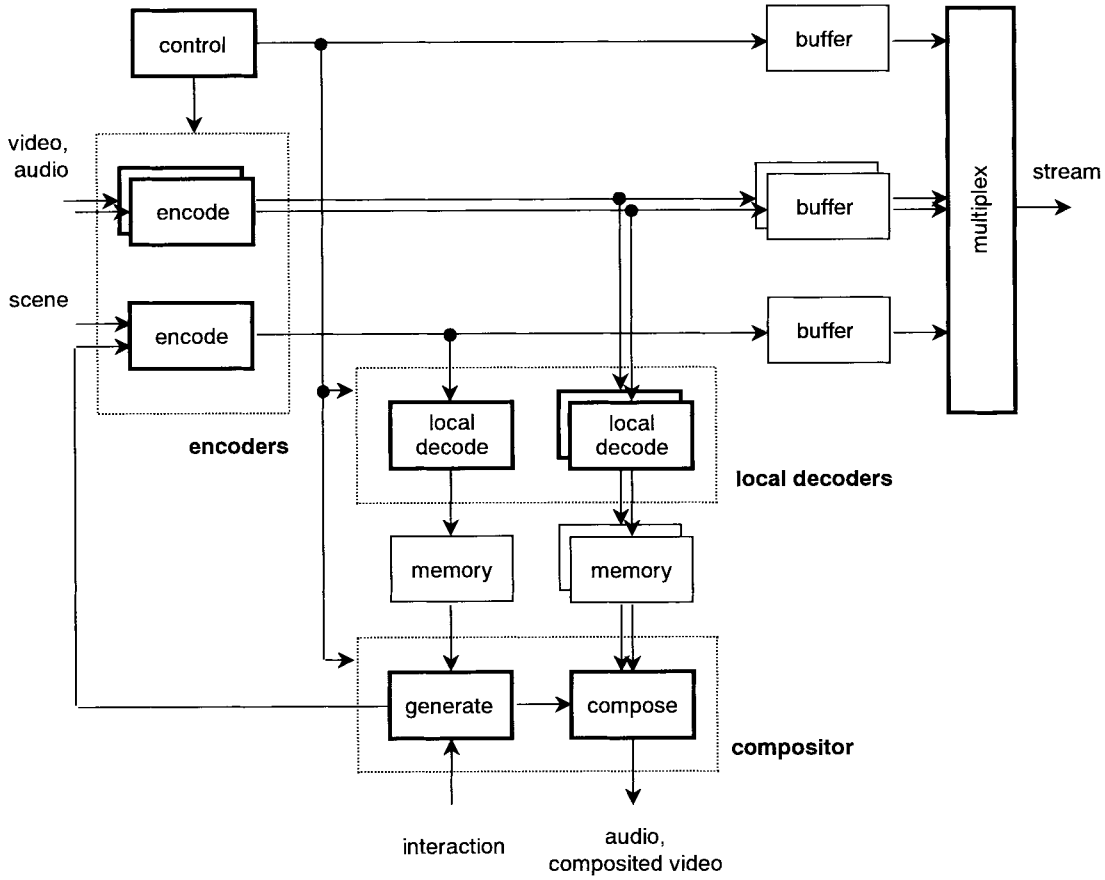



Fig. 5. Server configuration.

composition by the compositor. Scene-description formats are also encoded and put into a buffer. They are locally decoded (when necessary) and stored in a memory. A scene generator feeds the scene data and produces a new scene graph. This generator also accepts user interaction (via a keyboard or a mouse) and transfers the modification to the encoder. A composer gets the scene graph and video/audio signals and composes them into a scene. Note that the configuration of local decoders and a compositor is exactly the same as that of a browser.

### C. Synchronization Mechanism

Fig. 6 depicts a constant delay model, which is obtained by extending that of the MPEG-2 system [7]. In the case of the MPEG-2 system, it is assumed that the buffers at

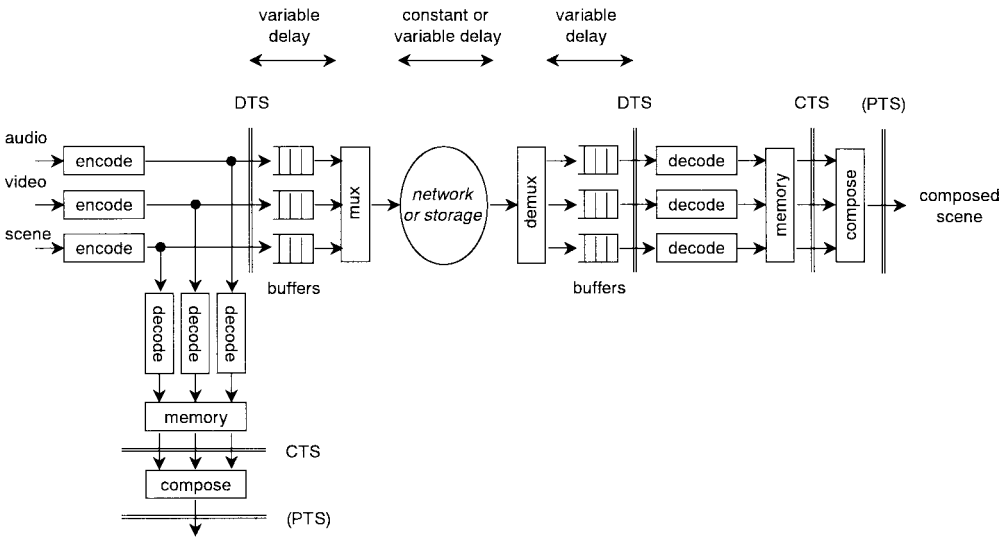



Fig. 6. An extended constant delay model for media synchronization: DTS, CTS, and PTS.

both the encoder side and the decoder side cause variable delays. Then two time stamps, the decoding time stamp (DTS) and presentation time stamp (PTS), are specified to indicate synchronization points among different streaming media. They are attached to access units at the encoder side and retrieved by the decoder to know adequate decode and presentation timing. When necessary, time bases furthermore are transferred from the encoder to the decoder, the so-called clock references.

This framework can be applied to the current case, in which a compositor is newly introduced. Buffers cause variable delays. Networks or storage devices cause constant or variable delays; circuit switching networks and storage devices cause constant delays, and packet networks such as the Internet cause variable delays. It is then assumed that, as far as any operations are carried out on the time line with constant delays to the synchronization points, media synchronization is possible. Accordingly, three time stamps are introduced; DTS, composition time stamp (CTS), and PTS.

The DTS is used to indicate decode timing of each access unit. In the ideal case, each access unit is extracted from a buffer, decoded, and put into a memory at DTS. The CTS is used to specify composition timing that corresponds to the PTS in the MPEG-2 systems. In the ideal case, each decoded signal is read from the memory and is composed into a scene at CTS. Following the MPEG-2 scenario, the DTS may be omitted when no B-pictures are used in the video compression algorithm. In that case, the DTS is assumed to be same as the CTS. The PTS is optionally used to indicate presentation timing of a composed scene. It also can be utilized to specify rendering points in time of continuous animation caused by script or byte code and to specify actual rendering delays when required.

In practice, the CTS/DTS is used to align any events on the time line with constant delay. A browser manages its resources to schedule adequate delays to let the decode/render operations be done in a constant rate. When excessive operations happen due to a terminal's insufficient abilities, there should be escape mechanisms, e.g., temporarily stopping subsequent rendering operations. Note that no strict buffer requirements are forced that are dissimilar to the MPEG-2 case because variable delay networks require loose synchronization there.

## IV. Network Architecture

### A. Basic Scenario

Fig. 7 shows a mapping scenario of the proposed data structure to actual transport mechanisms. There are already many transport protocols: for example, MPEG-2 transport stream [7], H.223 [11], H.225 [12], and RTP [6]. They introduce their own terms to classify data streams according to functionality and different quality-of-service requirements. In Fig. 7, they are classified into either system layer and media layer or control channel, media channel, and data channel. The system layer handles the control aspects and is equivalent to the control channel. The media layer conveys actual streaming data and corresponds to the media channel. The data channel is optionally used to transfer private data that may be inserted into the system layer. It is generally assumed that both the system layer and the control channel are protected by some error-handling strategies: retransmission or error-correction codes. This is because they convey important parameters that should not be lost.

Scene and media stream properties are necessary to initialize decoders and a compositor. Therefore, they are assumed to be transmitted by the system layer or by the control channel. Scene elementary streams convey scene-description data, which also need reliable transmission. Therefore, they are assumed to be transferred by the system layer as private data or by the data channel with some error protection. Media channels may be also utilized when adequate error protection is applied. Media elementary streams convey compression streams for audio, video, or geometry animation such as facial animation or object transformation dynamics. Their quantity is sometimes quite large, and they should be conveyed by the media layer or by the media channel without heavy error protection.

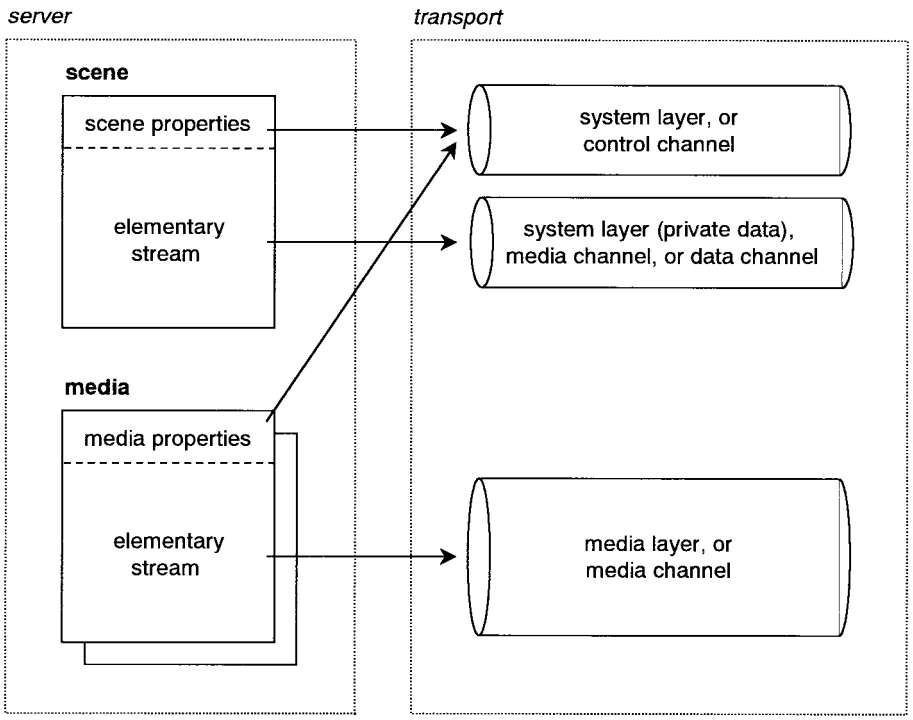



Fig. 7. Mapping to transport mechanisms.

The detailed relationship to the typical existing standards is presented as follows.

*1) MPEG-2 Transport Stream/H.222:* A system layer and media layers are provided. A system layer also provides a mechanism to transport private data. Media layers are used for transmission of video/audio compression streams. They are split into 188-byte fixed-length packets, called transport packets, and then multiplexed and transmitted.

*2) H.223:* A control channel, media channels, and data channels are provided. There are three layers known as AL1, AL2, and AL3 (adaptation layers). Generally speaking, AL1 is used for control/data channels, AL2 is used for audio transmission, and AL3 is prepared for video transmission. Annexes specify various error-protection strategies differently applied to these adaptation layers. They are multiplexed and transmitted with small overheads, which indicate multiplex patterns specified through a control channel.

*3) H.225:* Control channels, media channels, and data channels are provided. There are two types of control channels, assuming reliable transmission control protocol/Internet protocol (TCP/IP) transmission (H.245) and unreliable user datagram protocol (UDP)/IP transmission (registration, admission, and status). Data channels use TCP/IP transmission. Media channels utilize UDP/IP transmission with RTP headers attached. Multiplexing of each stream is done at the IP transmission level.

*4) RTP:* Media channels are assumed. This is a transport protocol for streaming applications on the Internet, and it constructs a part of ITU-T H.225, described above. RTP headers are attached to each packet transported on the media channel. Real-time transport control protocol (RTCP) packets are returned to report network status and other useful information. They are assumed to be transported on UDP/IP channels.

*5) MPEG-4:* No actual transport mechanisms are specified. They are called TransMux and should be determined in a flexible manner according to applications. Officially, when usage of the ITU-T framework is assumed, additional H.245 entries [10] are required. Otherwise, mapping of the scene/media properties to the H.245 control messages will be a promising choice.

### *B. Broadcasting/Multipoint Scenarios*

Fig. 8(a) presents an extended data structure to support broadcasting applications. A session description table is attached, specifying that multiple sessions (programs) are transferred simultaneously. This mechanism is exactly similar to the program map table used in the MPEG-2 transport stream [7]. A receiver chooses a program, initiates necessary decoders and a compositor, and only decodes elementary streams corresponding to the chosen program. In this broadcasting scenario, it is noted that interaction should be limited inside a receiver, i.e., it should not be returned to a server. Otherwise, a server suffers from overly heavy interaction requests.

Fig. 8(b) shows another extension mechanism, which supports multipoint communications. The scene-description format includes pointers to the media stream properties located at remote servers in addition to local ones. This mechanism is already supported by VRML through its utilization of a URL field. Therefore, this is a small extension to incorporate streaming applications. Remote entries in the scene description are provided by the server itself or may be appended by a new attendant.

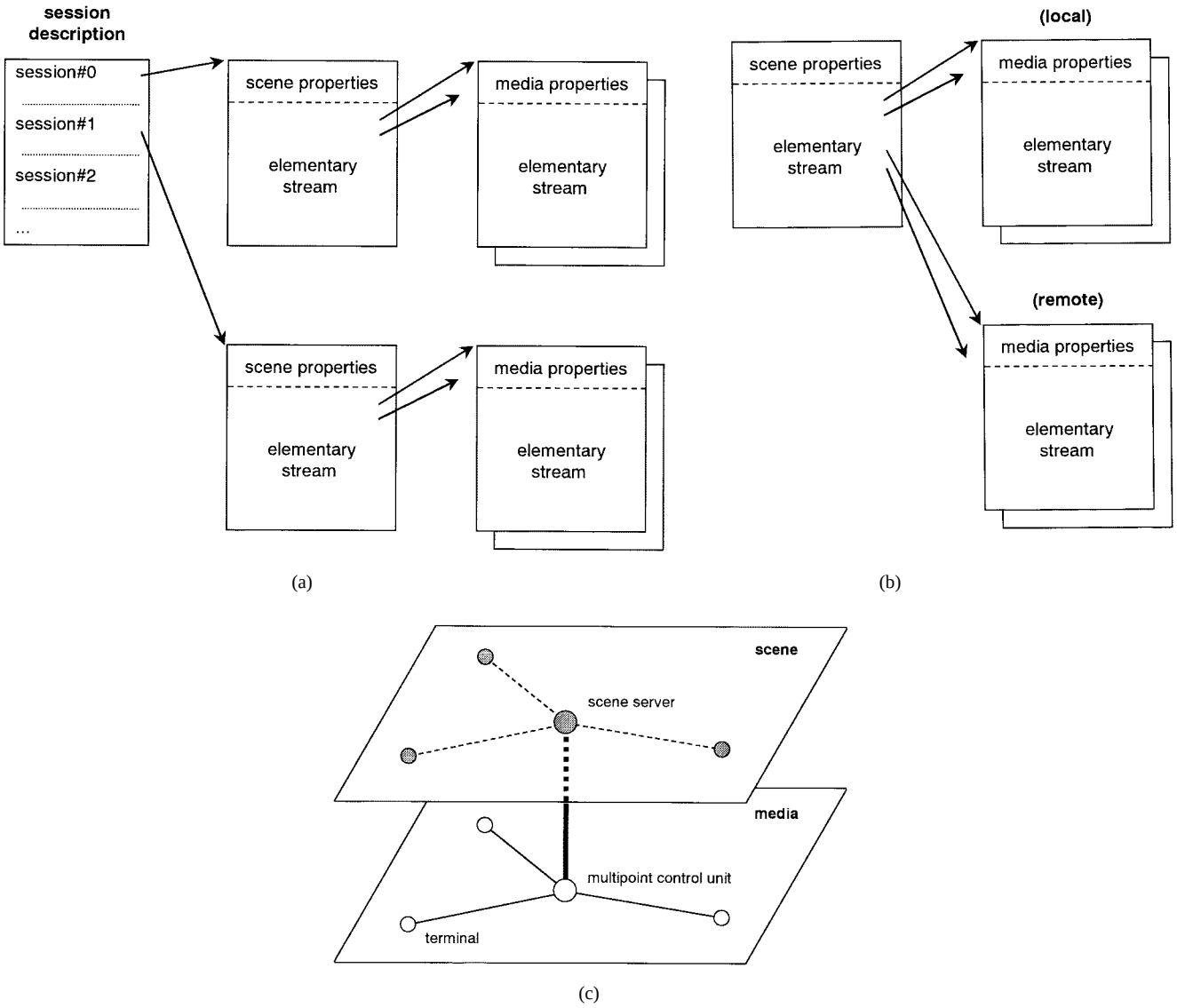



Fig. 8. Applications of scene description: (a) a broadcasting scenario, (b) a multipoint scenario, and (c) usage of a multipoint control unit.

Fig. 8(c) supports this scenario from the viewpoint of the ITU-T framework. The multipoint control unit (MCU) manages any streams emitted from attendant terminals. It also handles a scene-description format and works as a scene server. When a new terminal wants to join a session, it tries to talk to the MCU, opens necessary channels, and obtains scene data and media streams. This centralization mechanism unifies scene description and stream management in a consistent manner.

## V. Discussions

There are several new problems raised by the different cultures surrounding digital video and computer graphics. This section tries to summarize some of them and to provide possible solutions (compromises).

### A. Coordinate System

Computer graphics introduces a world coordinate system, which is independent of display devices (device coordinate systems), when describing a scene [8], [9]. Rendering means a mapping operation from the world coordinate system to the device coordinate system. On the contrary, a pixel coordinate system used in digital video generally coincides with the display devices, and subsequent aspect-ratio transformation is implied. When synthetic objects are defined in this pixel coordinate system, distortions may happen due to the aspect ratio transformation (i.e., a circle will be an ellipse).

This paper assumes usage of a world coordinate system for scene description similar to the conventional computer graphics approach. The aspect-ratio transformation is assumed to be managed in the scene description by adequately specifying the size of a rectangle onto which video textures are mapped.

### B. Conformance

Computer graphics has been evaluated only by subjective impression. Furthermore, there are a lot of evaluation criteria. One is rendering speed, which is accomplished by simplification of rendering algorithms. Another is picture quality, which is achieved by complicated algorithms with special techniques (ray tracing, radiosity, etc.) and little simplification.

Accordingly, it is very difficult to set conformance points to the computer graphics. Pixel-by-pixel conformance of resultant pictures, which has been forced on the digital video cases, will not be applied because there are no reference pictures.

A compromise is alleviation of conformance requirements according to complexity of rendering algorithms. When two dimensional, it is required that each object is presented at a correct place with specified colors. When three dimensional, it is required that each object is presented by a correct depth order with specified colors. When fast rendering algorithms are applied, some distortion may be permitted.

### C. Complexity

Rendering time greatly changes depending on the complexity of scenes. It also changes according to time events and user interaction that may suddenly happen. In the case of video compression, picture sizes and compression functionalities determine the decoder's complexity and lead to the concept of profiles and levels introduced in MPEG-2. Decoding time is almost stable as long as the decoder conforms to the profile and level. Therefore, there should be a similar measure that dominates the rendering time.

A possible solution is to limit the number of nodes as discussed in the VRML specification [3]. A more predictable solution to constraining the complexity of rendering, achieved by progressive mesh or MIP mapping, for example, could also limit the coverage factor of rendering images. These limitations enable the browser to predict the worst case rendering time. This enables the browser to allocate constant delays for possible rendering operations to keep the synchronization mechanism discussed in Section III-C.

## VI. Experiments

### A. Overview

Fig. 9(a) shows software architecture commonly used in the following experiments. It consists of stream interfaces, decoders, a compositor, and a controller corresponding to Fig. 4. The stream interfaces receive bitstreams that are transferred via networks or stored in local library files. The decoders accept bitstreams compressed by MPEG and the ITU-T H.26x series. The compositor produces a scene through application programming interface (API) functions provided by typical graphic libraries such as OpenGL, DirectX, and RenderWare. This part may be substituted by VRML/HTML browsers or other animation systems according to applied scene-description methods. The controller manages these resources through designated API's, including protocol handling, buffer size management, and media synchronization.

A number of primitive experiments are carried out: a) object extraction for scene composition, b) layered approach for scene composition, c) Java implementation of a two-dimensional (2-D) browser, and d) C++ implementation of a three-dimensional (3-D) browser. Experiments a) and b) demonstrate new concepts of communication with the help of innovative image-processing techniques and real-time rendering engines. The first one exploits multiple cameras and the other one uses motion information in order to extract object shapes from natural image sequences, which are put into synthetic scenes. Their practicality is proved by their real-time performance on inexpensive PC's. Experiments c) and d) focus on new kinds of content creation *aimed at Internet applications especially*. The first one composes a two-dimensional scene, and the second one a three-dimensional scene. Contents are transferred on the Internet in a streaming manner and are composed at the receiver side without waiting. These results prove the effectiveness of the proposed framework, which enables media integration of text and graphics onto streaming media. The details are as follows.

### B. Object Extraction for Scene Composition

We had developed an object-extraction algorithm using multiple camera inputs (typically four) [13]. This method utilizes focusing information and disparity information simultaneously and extracts stable shapes of target objects. This method, followed by scene composition, provides a result as shown in Fig. 9(b).

A bear is extracted from the four camera inputs. It is then put into a scene that is described by VRML. A viewer can move around inside the scene. The program is written in C++, and the RenderWare graphics library [14] is utilized to implement a VRML browser.

### C. Layered Approach for Scene Composition

The layered approach [15] produces multiple objects with shapes, which are extracted from the original video sequences. This method utilizes motion information in an efficient manner, and its successful tracking results in very impressive object extraction with motion. We followed this approach with modification by incorporating region information, and we put the results into a 3-D scene as demonstrated in Fig. 9(c).

A man and an array of monitors are extracted by the layered approach applied to our original video sequences. They are put into a scene that is also described by VRML. They move inside the scene as they do in the original sequences. A viewer can pick an object inside the scene. The program is also written in C++ with the support of the RenderWare graphics library.

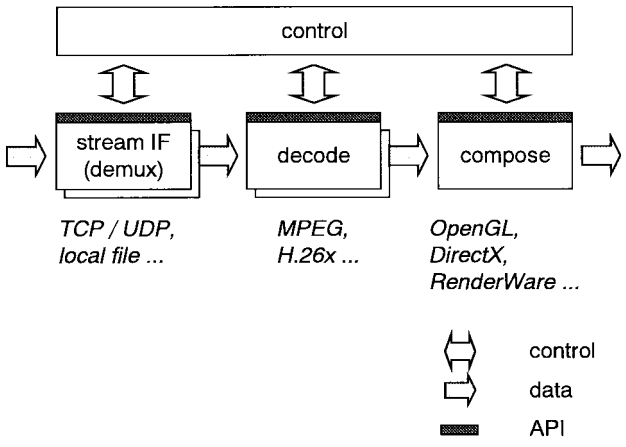
### D. Java Implementation of a 2-D Browser

Fig. 9(d) shows a result of a 2-D browser, which feeds MPEG-1 video sources and a scene-description source that was originally defined. The syntax of the scene source enables text overlay and its animation on the decoded movies.

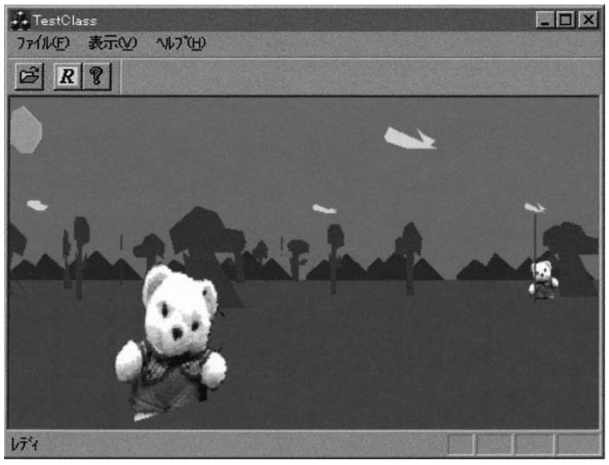
Since a program is fully written in Java, byte codes are downloaded beforehand and the compositor works on any platforms. The browser continuously requests and decodes access units through the HTTP connection. Streaming is successfully carried out as long as heavy network congestion does not occur.

### E. C++ Implementation of a 3-D Browser

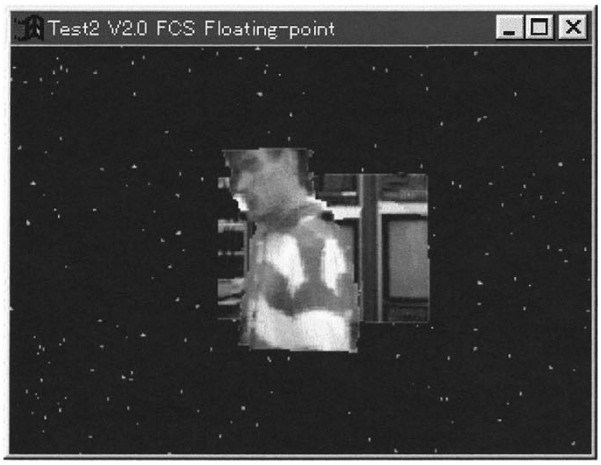
Fig. 9(e) demonstrates the result in a 3-D browser, which feeds MPEG-1 video sources and a scene-description source, BIFS, specified by current MPEG-4 activity. The BIFS stream



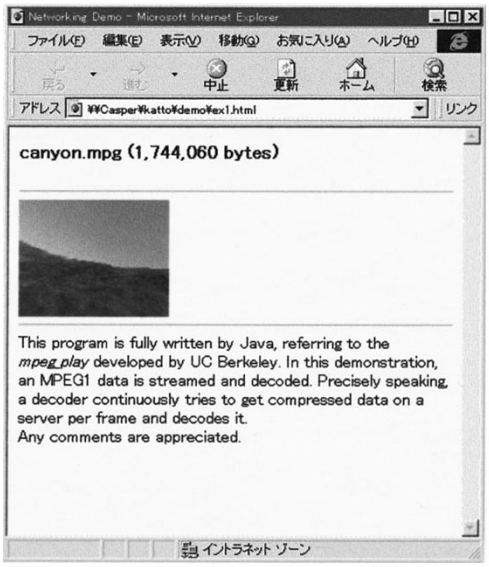

(a)



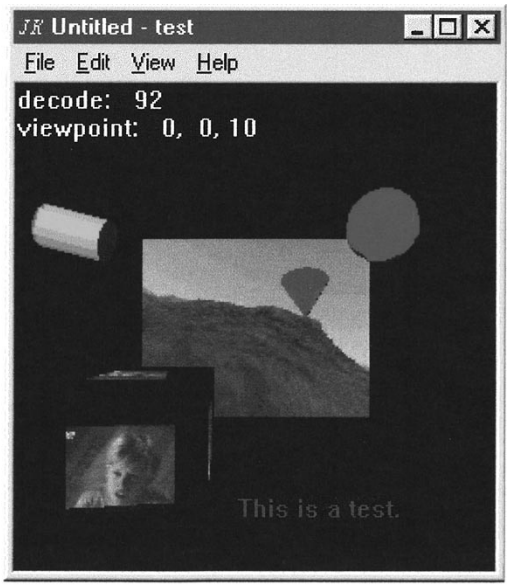

(b)




(c)




(d)




(e)

Fig. 9. Experimental results: (a) software architecture, (b) object extraction for scene composition, (c) layered approach for scene composition, (d) Java implementation of a 2-D browser, and (e) C++ implementation of a 3-D browser.

is created by an encoder that transforms a VRML file into a binary format, and the specified animation data are appended. The browser continuously feeds the streams and updates a scene at a specified rate.

MPEG-1 movies are texture mapped onto a square and a cube. Synthetic objects are also inserted into a scene through the VRML description. They are animated by the specified update mechanism. A viewer can change his viewpoint within

the terminal session using a pointing device. A program is written in C++ with the support of the OpenGL graphics library [16].

## VII. Conclusions

This paper describes system architecture for synthetic/natural hybrid coding toward future visual services. Scene-description capability, terminal architecture, and network architecture are discussed by taking account of recent standardization activities. Possible solutions to technological gaps between digital video and computer graphics are also mentioned. Some primitive but promising experiments are carried out by incorporating image-processing, video-compression, and computer graphics techniques into a single framework.

The synthetic/natural hybrid environment seems to be quite a natural direction for the very near future. There are already a closed-caption and a virtual studio, which are very simple examples. The Internet is another example, in which texts, graphics, audio, speech, and videos are integrated through the HTML/VRML descriptions. Practical problems arise owing to the limitations of computational power, which has to manage complicated compression algorithms and heavy rendering operations simultaneously. A breakthrough brought by the evolution of hardware and development of fast algorithms is expected and is regarded as future work.

## Acknowledgment

The authors would like to acknowledge that discussion with Prof. B. Liu and Prof. M. Orchard was most helpful. They also would like to thank members in the MPEG-4 SNHC and Systems subgroups for their fruitful discussion.

## References

[1] ISO/IEC JTC1/SC29/WG11, "Information technology—Very-low bitrate audio-visual coding—Part 1: Systems," ISO/IEC CD 14496-1, 1997.
[2] ———, "Information technology—Very-low bitrate audio-visual coding—Part 2: Visual," ISO/IEC CD 14496-2, 1997.
[3] ISO/IEC JTC1/SC24, "The virtual reality modeling language specification," ISO/IEC IS 14772-1, 1997.
[4] ITU-T, "Packet based multimedia communication systems," ITU-T Rec. H.323, 1998.
[5] ———, "Line transmission of nontelephone signals: Terminal for low bitrate multimedia communication," ITU-T Rec. H.324, 1995.
[6] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," RFC 1889, Internet Engineering Task Force, 1996.
[7] ISO/IEC JTC1/SC29/WG11, "Information technology—Generic coding of moving pictures and associated audio: Systems," ISO/IEC 13818-1/ITU-T Rec. H.222.0, 1995.
[8] F. James, A. V. Dam, S. Feiner, and J. Hughes, *Computer Graphics: Principles and Practice*, 2nd ed. Reading, MA: Addison-Wesley, 1990.
[9] D. Hearn and M. Pauline Baker, *Computer Graphics*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1997.
[10] ITU-T, "Line transmission of nontelephone signals: Control protocol for multimedia communication," ITU-T Rec. H.245, v. 2, 1996.
[11] ———, "Line transmission of nontelephone signals: Multiplexing protocol for low bitrate multimedia communication," ITU-T Rec. H.223, 1996.
[12] ———, "Line transmission of nontelephone signals: Call signaling protocols and media stream packetization for packet based multimedia communication systems," ITU-T Rec. H.225.0, v. 2, 1997.
[13] J. Katto and M. Ohta, "Novel algorithms for object extraction using multiple camera inputs," in *Proc. IEEE ICIP'96*, Sept. 1996, pp. 863–866.
[14] R. F. Ferraro, *Learn 3D Graphics Programming on the PC.* Reading, MA: Addison-Wesley, 1996.
[15] J. Y. A. Wang and E. H. Adelson, "Representing moving images with layers," *IEEE Trans. Image Processing*, pp. 625–638, Sept. 1996.
[16] J. Neider, T. Davis, and M. Woo, *OpenGL Programming Guide.* Reading, MA: Addison-Wesley, 1993.

**Jiro Katto** (S'89–M'92) was born in Tokyo, Japan, in 1964. He received the B.E., M.E., and Ph.D degrees in electrical engineering from the University of Tokyo in 1987, 1989, and 1992, respectively.

Since joining C&C Research Laboratories, NEC Corp., in 1992, he has been engaged in research on digital video compression, image processing, and multimedia communication systems.

Dr. Katto is a member of the IEICE of Japan.

**Mutsumi Ohta** was born in Osaka, Japan, in 1958. He received the B.S. and M.S. degrees in applied mathematics and physics from Kyoto University, Kyoto, Japan, in 1981 and 1983, respectively.

Since joining C&C Research Laboratories, NEC Corp., in 1983, he has been engaged in research on digital compression of television signals. He is now with the C&C Media Research Laboratories of NEC.

Mr. Ohta is a member of the IEICE of Japan and the Institute of Image Information and Television Engineers.