MPEG-4's Binary Format Codec for Scene Description

A. Kaczor, S. Maćkowiak

Poznań University of Technology, Institute of Electronics and Telecommunications,
Piotrowo 3A, 60-965 Poznań, POLAND
e-mail: kaczor@multimedia.edu.pl, smack@et.put.poznan.pl

Abstract – **In order to facilitate the development of authoring, editing and interaction tools, scene descriptions are coded independently from the audio-visual media that form part of the scene. This permits modification of the scene without having to decode or process in any way the audio-visual media. The following clauses detail the binary format codec for scene description that is compatible with ISO/IEC 14496-1.**

Keywords: BIFS encoder, scene description, MPEG-4

I. Introduction

The systems based on MPEG-4 standard let to independent encoding of audio-visual objects. The MPEG-4 scene description is based on VRML97 language and offers all properties of the VRML97. VRML is Virtual Reality Modelling Language, open 3D programming language became an international standard in 1997. (ISO/IEC 14772-1:1997) VRML files are typically stored in human-readable text format.

MPEG-4 [3-6] uses a client-server model (Fig.1). An MPEG-4 client (or player, or browser) contacts an MPEG-4 server, asks for content, receives the content, and renders the content. A scene can be composed of several media elements whose types can be of natural video, natural or synthetic audio, 2D and/or 3D vector graphics or text. The way all this different data is to be combined at the receiver for display on the user's screen or playback in the user's speakers is determined by the scene description. The spatial and temporal organisation of the different elements composing the scene is specified using a Multimedia Description Format or Language. MPEG-4 [1] provides a binary Multimedia Description Format called BIFS (Binary Format for Scene).

The MPEG-4 scene description standard was developed in several steps: BIFS Version 1 (about 45 new nodes), BIFS Version 2 (12 new nodes), BIFS Version 3 (4 new nodes).

Once a scene is in place, the server can further modify it by using MPEG-4's scene update mechanism, called BIFS-Command. This powerful mechanism can be used to do many things: for example, the objects can be manipulated by the server within a scene.

In this paper, the authors present the binary format encoder and decoder for scene description. This codec is a part of the Interactive Television System, which is implemented among the several partners under the iTVP project [8].

The scene description is done by the commands, which include information about objects, their sizes, colours and their properties. The MPEG-4 standard defines four commands, which have following meanings: insert an object into the scene, delete object from the scene, replace object in the scene and replace all objects in the scene.

The scene description lets to introduce the interaction between the user and the objects in a virtual scene. The user can interact with objects by rotating the objects, changing a view, scaling and moving the objects etc.

BIFS scene is composed of a collection of nodes arranged in a hierarchical tree. Each node represents, groups, or transforms an object in the scene and consists of a list of fields that define the particular behaviour of the node.

The tree structure consists of one root and much leafs. Such a structure is easy to present in syntax of XML language [7]. In a BIFS scene, every node, which is a termination of the branch, represents a multimedia object like a text, 2D and 3D objects etc. Every node in the tree has a strictly defined a number of the parameters that represent it.

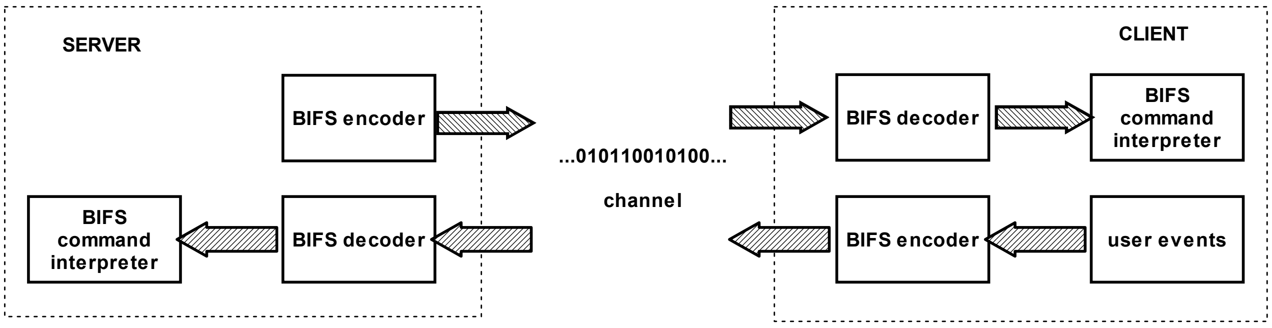

Fig.1. MPEG-4 client-server model and location of the BIFS encoder/decoder in the MPEG-4 system.

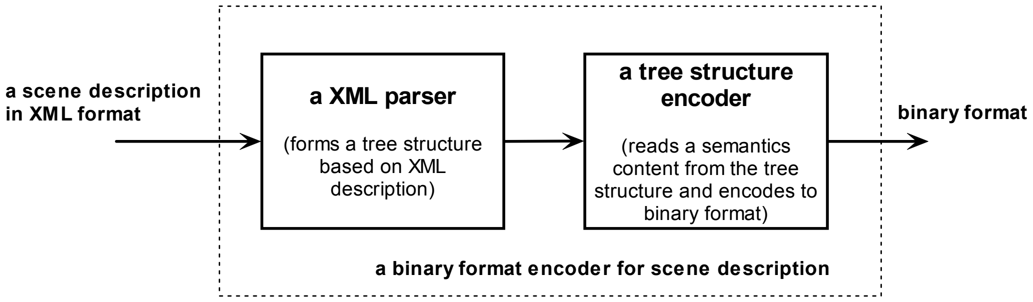
a scene description in XML format
a XML parser
(forms a tree structure based on XML description)
a tree structure encoder
(reads a semantics content from the tree structure and encodes to binary format)
binary format
a binary format encoder for scene description

Fig.2. The block diagram of scene description encoder.

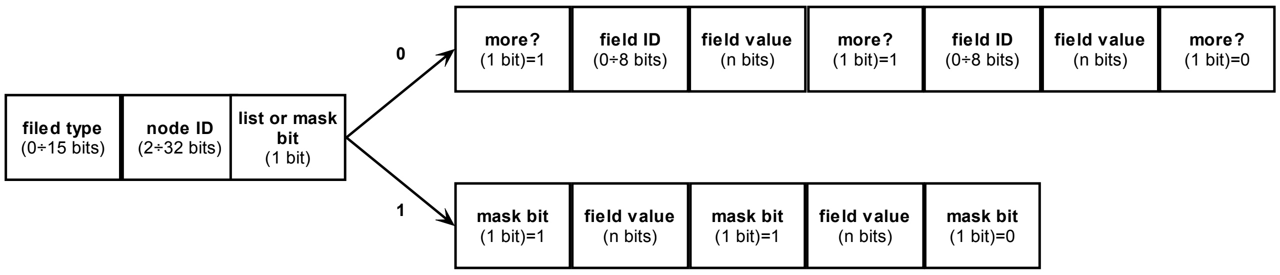

Fig.3. The example of the node encoding as a vector or a list.

II. THE CONCEPT AND STRUCTURE OF SCENE DESCRIPTION ENCODER

The implementation of the reference encoder (14496-5) is not optimised and it encodes only the scenes that are pseudo-VMRL language described. The XML parser and XML language converter have been never implemented in the reference software.

A scene description encoder is implemented in C++ language. The encoding process is realized in two steps, which are related to two general modules. The first module reads and analyzes a scene description in XML format (a block denotes as a XML parser in the Fig. 2), the second module encodes the tree structure of scene description to the BIFS bitstream (a block denotes as a tree structure encoder in the Fig. 2). The single encoding process is executed only for one BIFS command, which is a little part of a scene description. This encoding method limits a required size of buffer data in a program memory and can decrease an encoding time. Parsing the whole scene description and next encoding will not be efficient.

Architecture of source code of the tree encoder is very complicated. There are two main problems: many nodes to encode (the MPEG-4 standard defines about 120 different nodes) and a context based encoding algorithm. Every node could be different encoding accordingly to the parent. Therefore the tree encoder consists of many methods in several classes.

The fields of the node could be encoding as the vector or the list (Fig.3.). In the case of the vector, a mask bit set on 1 denotes that after this bit is encoded field. The bit 0 denotes that concrete node has not a more field (parameters). In the case of the list, bit 'more' defines appearance of the next fields. Therefore one field is encoded as a set of 3 parameters (bit 'more', field ID, field value).

The process of encoding of scene description strictly depends on encoder configuration. The configuration could be influence on compression ratio because the encoder configuration defines the number of bits that the encoder uses to encode a header of the nodes.

The configurations of the encoder and the decoder are set independently for every MPEG-4 elementary stream. The configuration process of the encoder/decoder must be done before the encoding/decoding process. The configurations are set up in the descriptor *ES_Descriptor*.

The compression ratio of BIFS encoder is typically about 10-15. The efficiency of the encoder also depends on a tree structure and quantization level.

III. The bitstream decoder

The bitstream decoder is analogically implemented as the encoder. The main difference is an inverse tasks execution. First the bitstream is translated to the tree structure and next the tree is converted to the text stream (a XML format).

The algorithm of binary format decoder processing is presented on Fig.4. The same algorithm for the execution of all types of command is impossible to present because every command have a different semantic. In the algorithm of scene description decoding are not implemented the modules, which eliminate the transmission errors. The transmission errors are detected by the higher layer of the MPEG-4 system.

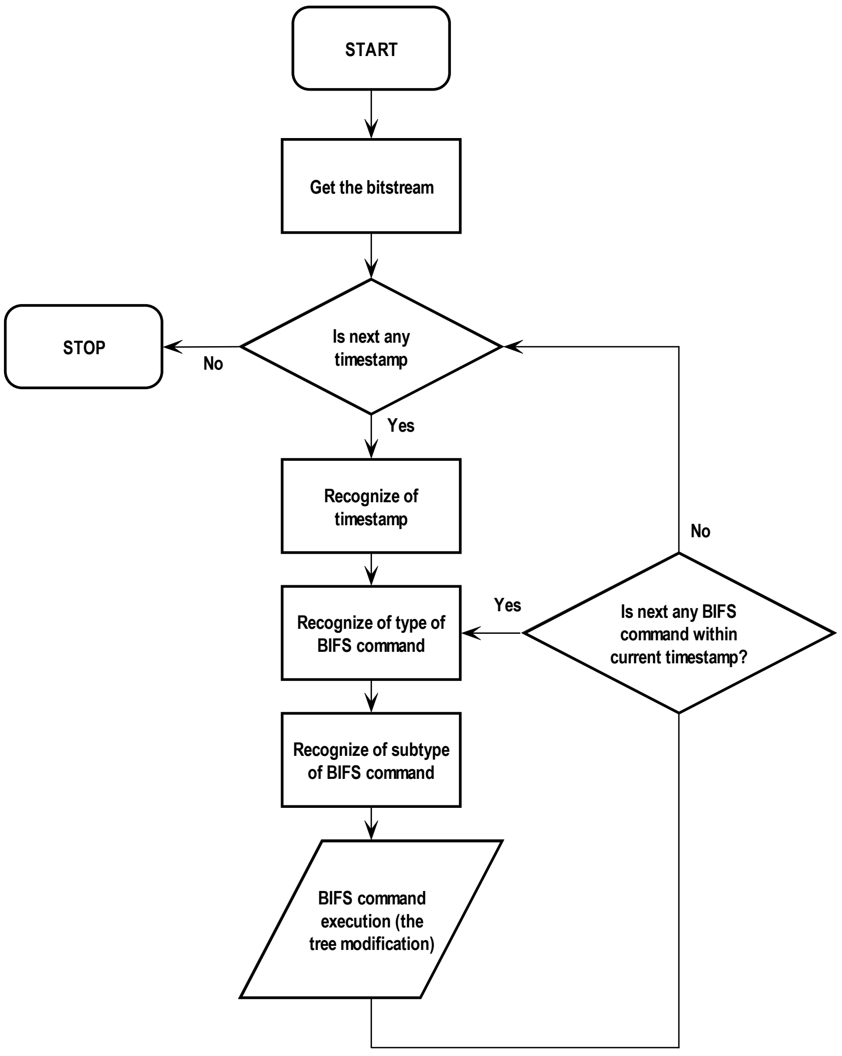
START
Get the bitstream
Is next any timestamp
No
STOP
Yes
Recognize of timestamp
No
Recognize of type of BIFS command
Yes
Is next any BIFS command within current timestamp?
Recognize of subtype of BIFS command
BIFS command execution (the tree modification)

Fig.4. The algorithm of binary format decoder processing.

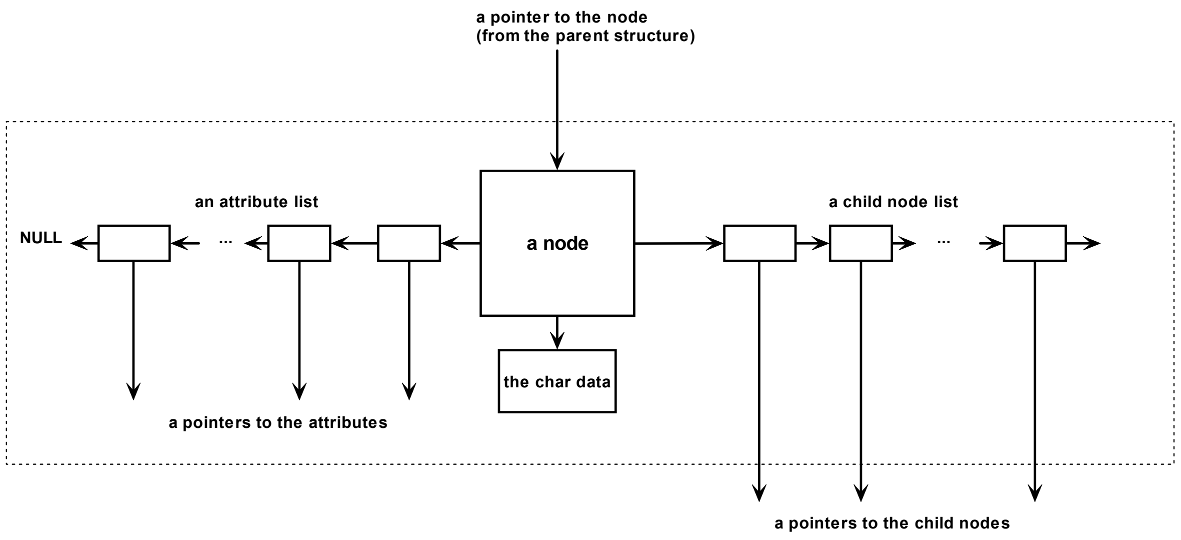
a pointer to the node
(from the parent structure)
an attribute list
NULL
...
a node
a child node list
...
the char data
a pointers to the attributes
a pointers to the child nodes

Fig.5. A scheme of the node in a memory allocated by the XML parser.
The pointers to the nodes have a communication role between nodes in the tree.

IV. XML PARSER

The XML parser module reads a scene description stream which is XML formatted and converts data to the tree structure. If the algorithm accepted the bits as the node flag in the stream, the structure presented on the Fig. 5 is created in a memory. In iTVP project the XMT-A standard (version of XML) is implemented.

Many nodes have fields that hold other nodes – this is what gives the scene its tree structure. Whereas VRML has only two node types, *SFNode* and *MFNode*, MPEG-4 has a rigidly typed collection of nodes that specifies exactly which nodes can be contained within which other nodes. A node may have more than one type, however, when it can be contained as a child node in different contexts.

In proposed codec, every node consists of two lists: the first is the attribute list with values and second that is the child node list. The authors applied the lists since at the moment of the node creation the, numbers of the attributes and child nodes are not defined. Implementation based on lists lets to add the attributes and child nodes without any restriction.

V. CONCLUSIONS

The whole codec is implemented as a DLL library. Therefore the tests of encoder/decoder modules in the early version are possible.

Up to now, the XML parser and the basic tree encoder/decoder are implemented. Currently the authors implement the set of the advanced option of the codec. The advanced options consist of definition mechanism and reusing of nodes (For example, once a wheel is defined as a collection of geometric nodes collected inside a Group node, it is possible to reuse the wheel elsewhere in the scene, rather than copying it explicitly wherever it is to appear), the mechanism of prototype creation (*PROTO* command enables the definition of new interfaces to user-constructed scene components), and possibility of the script execution (*SCRIPT* node - scripts can manipulate the scene directly, for example by routing output values of the script to other parts of the scene). In the future, the authors plan to implement the BIFS-Anim encoder, which is more efficient than BIFS-Update encoder. (the BIFS-Anim encoder offers much better a compression ratio and time synchronization between video and audio objects in the case of lip or face animation).

REFERENCES

[1] ISO/IEC 14496-1, Coding Of Audio-Visual Objects: Systems, Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2501, October 1998.

[2] ISO/IEC JTC1/SC29/WG11 N2611, "MPEG-4 Systems Version 2 WD 5.0," December 1998.

[3] ISO/IEC 14496-2, Coding Of Audio-Visual Objects: Visual, Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2502, October 1998.

[4] ISO/IEC JTC 1/SC 29/WG 11 N 2501+COR1+AMD1, Information technology – Coding of audio-visual objects – Part 1: Systems, 2000.

[5] F. Pereira, T. Ebrahimi, "The MPEG-4 Book", IMSC Press.

[6] A. E. Walsh, M. Bourges-Sevenier,"MPEG-4 Jump-Start", Prentice Hall, 2002.

[7] http://gpac.sourceforge.net/tutorial/bifs_intro.htm MPEG-4 BIFS and XMT Tutorial, Cyril Concolato © 2002, Jean Le Feuvre © 2003

[8] K. Warmiński, Z. Pióro, R. Sitnik, „Concept of Interactive Television (iTVP) Platform", IWSSIP 2004, Poznań, 2004