The Design and Implementation of a Visual MPEG-4 Scene-Authoring Tool

Meng-Jyi Shieh, Kuo-Luen Perng, Wen-Chin Chen
Communication and Multimedia Lab.,
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan

E-Mail: {feitian, perng, wcchen}@cmlab.csie.ntu.edu.t

ABSTRACT

We have implemented an MPEG-4 scene-authoring tool for complete profile in our laboratory. This tool is the extension of the result of our laboratory last year. We changed the system architecture and design for 3D scene, the event in/out mechanism, visual editing, and friendlier user interface. We believe that such a tool can allow users to produce MPEG-4 contents easily and thus can push the MPEG-4 standard to be widely used. In this paper, we describe the design and implementation of an MPEG-4 scene-authoring tool, and we also introduce some necessary components for an MPEG-4 scene-authoring tool.

1. INTRODUCTION

The MPEG-4 [1] standard is specified by ISO/IEC 14496. It builds upon three application areas: digital television, interactive graphics applications and interactive multimedia. It aims at establishing a universal, efficient coding of different forms of audio-visual data, which we call audio-visual objects. Moreover, it tries to offer a new natural and synthetic interactive multimedia. This goal will be attained by defining two basic elements:

1. A set of coding tools for audio-visual objects capable of providing support for different functionalities such as object based interactivity and scalability, and error robustness, in addition to efficient compression.
2. A syntactic description of coded audio-visual objects, providing a formal method for describing the coded representation of these objects and the methods used to code them.

The MPEG-4 scene composition defined in a language called BIFS (Binary Format for Scenes). BIFS is based on the Scene-Graph concepts employed in VRML [4]. It is a superset of VRML with adding features such as Facial Animation, Enhanced Audio, native 2D primitives, Streaming and Animation streaming protocols.

In order to allow users to interact with the presented scene, ISO/IEC 14496-1 provides Interactivity mechanisms, which are integrated with the scene description information in the form of linked event sources and targets. These event sources and targets are parts of the scene description nodes, and thus allow close coupling of dynamic and interactive behavior with the specific scene at hand.

A BIFS scene is not likely to be static. It may be added, deleted, or modified later on. An MPEG-4 content server can send a sequence of commands to update BIFS scene. We call those commands BIFS-Commands. A BIFS-CommandFrame consists of some BIFS-commands. However, ISO/OEC 14496 does not specify the storage form of BIFS-Command in server-side. It just specifies the transmit format between server-side and client-side. This implicates that a server-side program may dynamically create a BIFS-CommandFrame and than send it to client.

In this paper, we present the design and architecture of the authoring tool that allow users to visually edit a 2D/3D BIFS scene. Furthermore we introduce some necessary functions and tools that an MPEG-4 editor must hold. The propose system is an extension of our previous work [2] and we will still improve this system in future work.

2. MOTIVATION

MPEG-4, which includes the most of the newest generation of multimedia technologies, is the most complex coding standard in the world now. Owing to its enormous architecture and many system components, which are across many domains, it becomes hard to implement and has no real applications. To push this standard to be a success and widely used, it needs not only a complete implementation of the whole system but also abundant MPEG-4 contents. In order to let content providers create variant, interesting, and interactive contents easily and rapidly, we decided to develop a system, called MPEG-4 scene-authoring tool, for the content editing use. Following are the major issues of our MPEG-4 scene-authoring system design:

- What you see is what you get
- General and fewer restriction
- User friendly

Especially, the first and the third issue have great influences on the users. A system with these characteristics gives the users direct senses about the system manipulation and let users spend less time on learning how to use this system, so the training costs reduced.

From the point of view of system design, the second issue is usually opposite to the third one, because users need to control more detail parameters with operating a general system, and since the system need to provide more user interfaces to access there parameters, it won't be user friendly. To solve this dilemma,

we made many efforts to balance the weights of building a general system and making the system user-friendlier.

3. SYSTEM OVERVIEW

From the system operational aspect, the authoring tool consists of several major components. We briefly describe each component as follows:

3.1 Scene tree structure editor

The normal way to build an MPEG-4 scene is to use a VRML like description language. By using this description language, users can adjust all the detail options of a scene node. Unfortunately, the MPEG-4 standard defines exactly one hundred types of nodes with different functionalities. In addition, the more details defined in the language, the more difficulties for general users to learn and use it.

For this reason, we provide a visualized Scene Tree Structure Editor. By using this tool, users do not need to know any grammar of the description language and the hierarchically relation of the scene objects. This tool allows users to construct a BIFS scene with only a little BIFS concepts. Therefore, it is necessary for the MPEG-4 scene-authoring tool to provide such a tool.

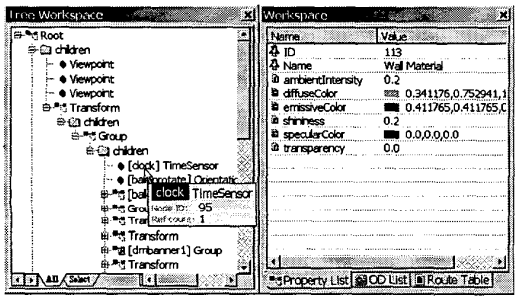

Figure 1. Snapshots of scene the tree structure editor window and the node properties editor window

3.2 Node properties editor

A scene node consists of some fields. Each field has its own data type. Sometimes, these data types are not intuitive and may raise difficulties for uses to assign their data values. For this reason, our system provides a node properties editor. It allows users to see the detail of a node and edit the attributes of the node. The editor provides different interface for assigning the data values to different fields.

3.3 Event (route) editor

Users may provide the "route entry," which is an interactive relation between two nodes in the scene to enable user interactivity. A node may have several "event in" and "event out" fields. The definition about which fields can "event in" or "event out" in a node is defined in ISO/IEC 14496-1:1999 Annex H. There are exactly 100 nodes, and every node has its own special event definitions. Users may get confused with it. Therefore, we must provide a way to give hint to users as to what they can choose.

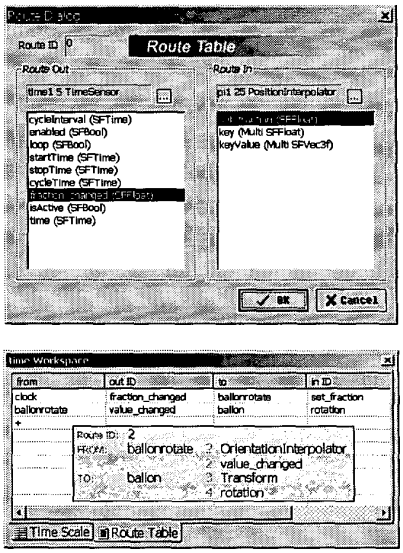

Figure 2. A snapshot of editing event entry and route table

3.4 BIFS-Commands editor

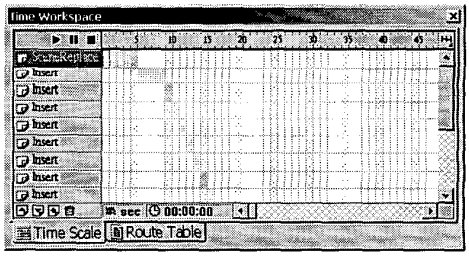

Figure 3. Snapshots of BIFS-Commands window – Every row defines a BIFS-Command, and its column indicates the showing time.

BIFS-Commands are used to modify a set of properties of the scene graph, its nodes and behaviors at a given time. Commands are grouped into CommandFrames in order to be able to send several commands in a single access unit. There are four basic commands, as follow:

1. Replacement of an entire scene
2. Insertion
3. Deletion
4. Replacement

We provide a BIFS-Commands editor to allow users adding BIFS-Command information in a scene. By using this tool users can specify a time to add some new nodes or to modify some attributes of a node in a scene. That can appear more plentiful variation in a scene than just monotone movement.

3.5 Preview and interactive window

In order to achieve the goal of "What you see is what you get," a preview window is provided. Users can directly see the visualized scene result instead of the scene description language scripts. Besides, they can control objects through this window. They may adjust position, size, and rotation angle of the objects by intuition. However, not all the objects in the scene are visible, these invisible nodes cannot be controlled by this way.

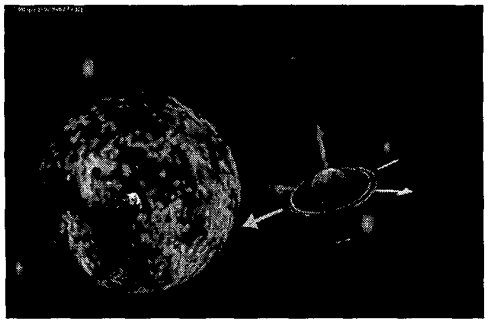

Figure 4. Snapshots of preview and interactive window – In the scene of the solar system, a planet is been editing by user mouse actions, and the bounding box and some control points are appearing in the preview window.

3.6 Import object

There are several reasons why we must provide such functionality. First, an author may have previously created some 3D objects in a different format. With this functionality, he may reuse those 3D objects in a new scene. Second, an author may be accustomed to certain other 3D model editors, such as Maya or 3D Studio Max. He can still edit a static scene with his custom 3D model editor, and then imports this static scene into our system.

3.7 Export to server side storage format

ISO/IEC 14496-1 does not specify the scene storage format of a scene at the server side. It only specifies the format of the bitstream, which is transmitted from the server-side to the client-side. This implicates that the bitstream of BIFS may be produced dynamically by an MPEG-4 server-side program. Therefore, we are developing an interactive MPEG-4 streaming server. In this project, the streaming server uses server-side JAVA script to produce the bitstream of the BIFS-CommandFrame. The current authoring tool saves the scene data into the format of the new interactive MPEG-4 streaming server.

4. DESIGN OF THE SYSTEM ARCHITECTURE

ISO/IEC 14496-1 specifies 31 Node data types, 100 kinds of nodes, and 20 basic data types. This is not a simple case for object-oriented programming designed for MPEG-4 scene editing and rendering. Not only many classes need implementing, but also some mechanisms need designing in kernel.

4.1 The system modules

In high-level view, our architecture consists of three modules: **BIFS Editor**, **Scene Render**, and **Scene Graph Manager**, as illustrated in Figure 5. The actions of the user will affect the organization of the **BIFS scene tree** by going through both **BIFS Editor** and **Scene Render**.

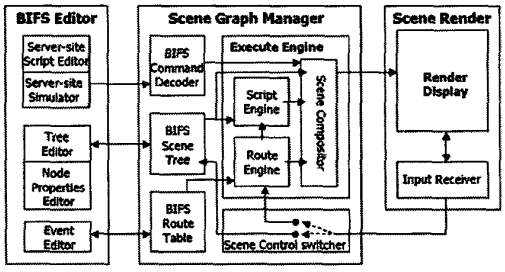

Figure 5. The System Architecture

The **BIFS Editor** module can show BIFS Scene Description in a user-friendly way. It provides a visual, non-language, and instantly error-checked mechanism. It contains 4 submodules: **Tree Editor**, **Node Properties Editor**, **Event Editor**, and **Server-site Script Editor** modules. Users can directly read and modify scene tree organization in **Tree Editor**, and modify the attribute of the selected node in **Tree Editor** by using **Node Properties Editor**. In addition, the entry of **BIFS Route Table** can be illustrated and modified by using **Event Editor**.

The **Scene Graph Manager** module, which consists of some data and controllers, is the kernel of our system. While the data of **BIFS Scene Tree** and **BIFS Route Table** are the most critical in this module, **Execute Engine** is the most important controller in this module. **Scene Compositor** would duplicate the data of **BIFS Scene Tree,** which will be used or modified later by other controllers. Once an event has occurred, **Route Engine** will route this event to its target. Meanwhile, the data of **Scene Compositor** might be modified or some script function executing in Script Engine could be triggered.

To fulfill What-You-See-Is-What-You-Get, the scene tree will be sent to the **Scene Render** module for previewing after composing by **Scene Compositor**. The action of users will be sent to **Scene Control Switcher**, which will then deliver the event to different block according to the current editing mode. On one hand, if the editor is in the preview mode, the UI message will be sent to **Route Engine**; on the other, when editor is in edit mode, the UI message will be sent to **BIFS Scene Tree.**

Moreover, **BIFS Scene Tree** could be modified according to the user actions.

4.2 Event route mechanisms for virtual editing

To realize the objective of virtual editing, some mechanisms must be added to the scene tree node classes and the render module. An object in a render window consists of some nodes. Therefore, if a user wants to move an object or resize an object by using the mouse in the preview window, the system may adjust field values in different nodes.

To make this problem more general, our system uses event route mechanisms. In editing mode of preview window, when users choose a node in the BIFS scene tree presentation window or click an object in the preview window, a sub-tree will be chosen, and we say this sub-tree is in focus state. After that, the preview window computes the bounding box and then shows some control points on the screen, as in Figure 4 and Figure 7. Every control point has its individual significance in different editing modes, such as **Move**, **Resize**, and **Rotation**, and so forth. When a user drags a control point, the control point will throw an event with some bearing information to the focused sub-tree. The event will be delivered recursively until a node accepts it. This node will decide to apply this event to its attributes or selectively deliver this event to its children nodes.

Figure 6 is an example of a box object. If an event arrives to the transform node that is the root of the sub-tree, when this event is **Move**, **Scale**, or **Rotation**, the transform node will accept this event and apply this to its attributes. But if this event is **Resize**, transform node will deliver this event to its children nodes. After that, the shape node accepts this event and simply delivers it to the geometry node. Finally, geometry node receives it and applies to size field.

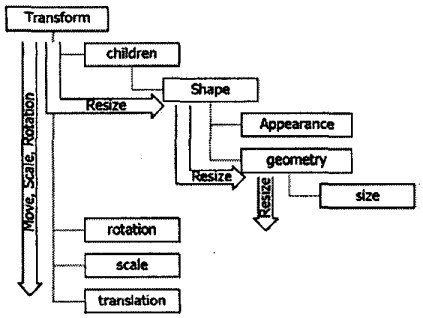

Figure 6. An example of the edit message route mechanism – This diagram shows the scene tree of a box, which receives an editing message and decides to send this message to children or hold it and applies it.

5. RESULTS

We implemented our system in Windows 2000 with VC++ 6.0, BCG UI library, and Direct3D/OpenGL library. Figure 7 is a Screen shot of the result. At the ending final, the authoring tool allows users to save these scene data into the data format of the MPEG-4 interactive server, which was also developed in our laboratory.

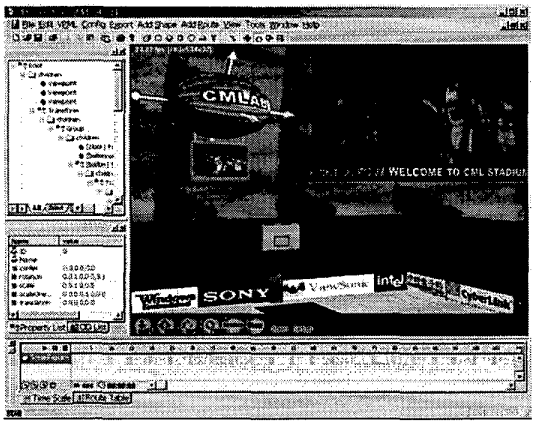

Figure 7. A snapshot of our system – You can see a hot balloon is in selected state and some control points are showing.

6. CONCLUSION AND FUTURE WORK

A new interactive authoring tool is currently under development. It allows editing client-site and server-side JAVA-script and simulating its executing result. We will use JAVA-Script solution to support BIFS commands and the user interactivity.

7. REFERENCE

[1] ISO/IEC 14496 AM 1, Part 1: Systems, Part2: Visual, Part 3: Audio, Part 6: DMIF, International Organization for Standardization, 2000.

[2] D.R. Liu, M.J. Shieh, Y.C. Lee, W.C. Chen, "On the Design and Implementation of an MPEG-4 Scene Editor", ICCE 2000 Digest of Technical Papers.

[3] Y.S. Tung, J.L. Wu, and C.C. Ho, "Architecture Design of an MPEG-4 System", ICCE 2000 Digest of Technical Papers.

[4] VRML (IS 14772-1), Virtual Reality Modeling Language, April 1997.