A Rendering Module of MPEG-4 System Based on VRML97 for Virtual and Natural Scene Integration

Kuo-Luen Perng, Yu-Chung Lee, Wei-Ru Chen, Yu-Li Huang, An-Lung Teng, and Ming Ouhyoung

Communication and Multimedia Lab
Dept. of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

Abstract

An architecture design and implementation of an MPEG-4 rendering module is proposed. MPEG-4 is an object-based international compression standard established by ISO, and this standard has many significant features that make it very suitable for Internet applications with variable or very low bit rate. In addition, the object-oriented characteristic allows greater user interaction than before. To obtain the powerful and attractive features of MPEG-4, the rendering module has to interpret the scene structure in MPEG-4 scene description language, which is based on VRML97 – the most famous industrial standard to describe a 3D scene on the Internet. Various nodes are implemented – geometry nodes, non-geometry nodes, route nodes, texture nodes, etc. As an example, the system is able to show three moving cubes with each face containing a video running at 70 fps.

Because MPEG-4 is a highly extensible standard, new features are possibly added as objects of the scene. The system should have good flexibility to include new features. We'll take a panoramic image viewer as an example to show the ability of our system to integrate new features.

Figure 1: Using our rendering module to browse an MPEG-4 scene, which is composed of natural and synthetic objects.

Key words: MPEG-4, DirectShow, VRML97, BIFS

1. Introduction

There are lots of multimedia standards in storage and communication usage established by the organizations such as ISO or ITU. But when being applied to environment differs from its original purpose, most of them will lead to unpredictable outcomes. In the recent years, technology in communication and multimedia field is making great progress. Various new applications are appearing in different fields rapidly, which are with different bandwidths, different computation powers, different transmission error rates, etc. Obviously, old multimedia standards are becoming unable to satisfy applications in different environments. MPEG-4[1] [3][4] is a standard with new ideas in many aspects. First, to compare with previous frame based standards, MPEG-4 takes "object" as the basic unit of the scene. Each object could be edited or adjusted individually, and could be treated with dif-

ferent codecs. It brings great flexibility and freedom to the content authors, service providers, and end users. Another significant feature of MPEG-4 is the ability of Synthetic Natural Hybrid Coding (SNHC). This not only enriches the content of MPEG-4 scenes, but also leads to more reasonable manipulation of limited bandwidth. To accomplish the above features, MPEG-4 must draw up a scene description language to describe the structure of the scene. The language takes VRML97[2] as the basis and adds some new nodes for other purposes. The rendering module composites and renders the scene according to the structural information and the media samples dealt by the visual codec. Furthermore, the rendering module has to implement several important mechanisms so that the MPEG-4 system can bring its ability into full play, such as navigation in the scene, changing the viewpoint, individually adjusting playing quality of video objects, and the animation mechanism.

1.1 System Overview

In essence, our system is an implementation of a VRML browser under the MPEG-4 architecture. The difference between other VRML browsers and ours is the VRML scene data acquired through the BIFS Decoder (Binary Format for Scene Stream Decoder). The video/audio data required by scenes are processed through a video/audio decoder in our system.

Our rendering module consists of the following tasks. Two of them are about composition and displaying the scene onto a screen, and others are about cooperation with other modules in the system:

1. To control the 2D/3D rendering engine.
2. To interpret the scene tree structure, compose the scene, and set up the geometry framework.
3. To support the node definition and the structural mechanism of scene description language.
4. To link up with the media codec, get the visual media sample, and manage buffers.
5. To interact with users, provide navigation ability, and feedback users' requests to the system.

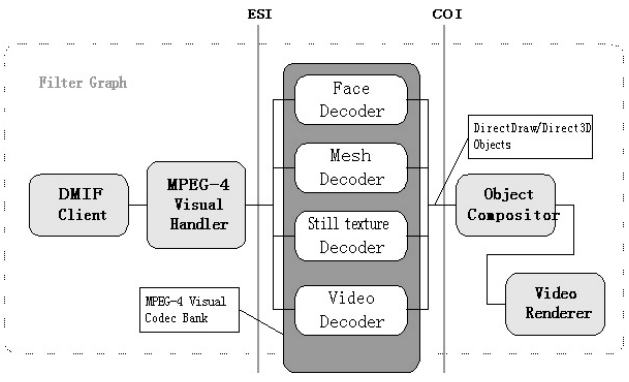

Figure 2: The figure above is the implementation of our rendering module which is the part to the right of the line of COI (Composition Interface).

Before the final MPEG-4 system integration currently, we have our own independent testing environment. In this testing system, MPEG-4 scenes are described in the VRML grammar, and then are interpreted by the parser. The decoder for still images/video can read the necessary texture data in advance for testing.

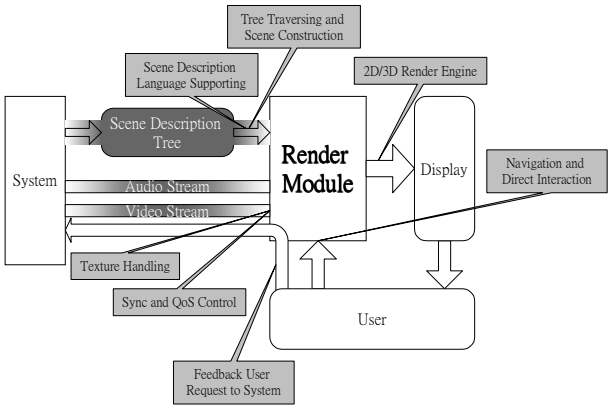

Figure 3: Illustration of the I/O flow of the whole rendering module.

In the implementation of MPEG-4, we use the Microsoft multimedia architecture, Directshow. From software points of view, the kernel of DirectShow is a modularized pluggable system, based on the usage of the so-called filters. The most significant advantage of DirectShow comes from its ability to make the multimedia application design more clear and easy. By carefully dividing the work into connected filters in the DirectShow architecture, each filter can be implemented by different program developers. Another advantage of DirectShow is the filter re-use, which speedups the developing of new multimedia applications. So our program of rendering can be independent from other parts in the system, and is wrapped to be a filter according to the DirectShow architecture.

2. Implementation

The rendering module is developed on the Microsoft Windows 98/2000 platform. OpenGL and DirectX are used to implement rendering. In order for the convenience of cross-platform compatibility, we wrapped our program in a new interface for the use of OpenGL and DirectX. In actual implementation, when there are more video textures in the scene, we can have greater performance by adopting DirectX for rendering, because we can take advantage of hardware acceleration on most video cards that supports the DirectX hardware abstraction layer. Currently there are no special functions designed for 2D image processing in OpenGL, so acceleration for 2D image processing is processed purely by software. If there are not many video textures, the per-

formance in adopting DirectX or OpenGL is nearly the same.

2.1 Scene Tree

Scenes are depicted by various nodes in VRML. Our system records all of the nodes in the scene by constructing a tree, and the tree structure is called "Scene Tree". In the integrated system, the scene tree is constructed by the BIFS Decoder. In the testing platform prior to the final system integration, a VRML parser sample provided by SGI is used to interpret the scene tree after reading the VRML file.

Each time a scene needs to be re-rendered, the whole scene tree will be re-traversed. When a node is met, the corresponding handle function is executed. After traversing all the nodes in the scene tree, the rendering of the scene is completed.

In the rendering module, traversing a scene tree will be iterated, not just done once at first. It is because that in the MPEG-4 system, the tree structure will be updated in run time, and it's possible for the route mechanism to change the data of the nodes in the tree all the time.

2.2 Geometry Node

In the initial stage, the nodes such as sphere, cone, circle, and cylinder will need that all required triangular data be pre-calculated. Our program provides users to adjust the level of details of these figures.

In the future, we will add a function to have the program itself adjust the level of details. So when objects are far away, the program will automatically draw these nodes with fewer triangles.

The mechanism of text-showing nodes is different in Direct3D and OpenGL. Because until so far, the program still cannot get the vector data of letter forms. In Direct3D, every character is implemented as a rectangular polygon mapped with a texture including alpha blending data. But OpenGL provides functions to convert letter forms to OpenGL lists, so the real vector characters can be drawn.

2.3 Non-geometry Node

Some nodes are not designed for drawing, like interpolator nodes, sensor nodes, and transform nodes used to change positions of geometry nodes.

Interpolator nodes are utilized to produce the animation effects of geometry nodes. Each time when an interpolator node is traversed, the interpolated key value is established by the route mechanism. The handle function of the interpolator node is to compute the result of interpolation through the key value.

The handle function of the time sensor is to compute the value of the time counter during each execution. The handle function of the transform node is to record the transformation matrix by a stack mechanism while drawing.

Based on the new transformation data, new matrix can be produced. OpenGL itself provides a stack mechanism, but for Direct3D a stack mechanism was implemented.

2.4 Texture

There are three types of textures: still images, video and CompositeTexture.

With execution efficiency in mind, textures of still images are constructed by the mip-map method, but not for video. In implementation only single texture is needed for video, because it will refresh constantly, unlike still images. The constructed textures for a still image can be used repeatedly while in video the case is different. It will be a waste of time to construct video textures by the mip-map method. The way to handle video textures is: if the required image-space size isn't large, a smaller space size of a video texture can be constructed for it. Since the video part of MPEG-4 contains the function of scalability, the decoder will be informed that lower quality data is required in this case.

CompositeTexture is a texture created from an image of VRML scene, then pasted upon the geometry node. In OpenGL and Direct3D, we use a similar process. We allocate a part of memory, and assign the render target of OpenGL/Direct3D to the memory, and then convert the image of the memory mentioned above to a texture, before pasting it to the geometry node.

2.5 Route Node

The data of route nodes is unique. It's not stored in the scene tree, and we use a separate table to record those nodes which will influence the value of other nodes.

```
struct
{
        NODE *SourceNode;
        FIELD *FieldNode;
        NODE *DestNode;
        FIELD *FieldNode;
};
```

Generally speaking, source nodes can be categorized in two types of nodes: sensor nodes and interpolator nodes.

During traversing a scene tree, the values of source nodes will be computed first. After finishing traversing, all the data of the corresponding source fields will be copied to the destination fields. Thus, next iteration during rendering scenes, new data will be used to derive the animation effects.

2.6 Selecting an Object

The touch sensor provides the function of selecting objects by using a mouse. After pasting a video texture on the geometry node, with the user interface, we can select an object at any time and perform play, stop or pause to the video on the object.

The way to select objects is very different in OpenGL and Direct 3D. OpenGL provides a mechanism of selecting objects. There is no such function in Direct3D. The way we do it is each time after drawing a geometry node, the ZBuffer value of the mouse cursor will be checked. And if the value changes, it means the mouse has pointed to the node. After drawing all the nodes, the last one selected is the real selected node by the mouse.

2.7 User Interaction

Since MPEG-4 Scenes are composed of many 3D objects, the system should provide functions that users can roam or rotate/translate objects in the scene. In the near future, functions for users to insert/delete objects will be added.

3. Applications

Besides MPEG's conventional function of playing video, MPEG-4 rendering can be used for new applications mentioned below.

3.1 Panoramic Image

The first application is to show a spherical panoramic view with arbitrary viewing angles. In implementation, in addition to the most essential function of changing viewing angles, we draw a big sphere, and paste ready-made panoramic image on the big sphere. Thus, our system can be a tool to view a panoramic scene.

Figure 4: Use our rendering module to combine a panoramic image and a VRML scene together.

3.2 Virtual Meeting

MPEG-4 defines the face node to render human heads. We can render a talking head in real-time from the description of Facial Animation Parameters/Facial Destination Parameters node conveyed from the remote end.

Figure 5: Use our rendering module to display a scene which contains a human head model with facial animation and other synthetic objects.

3.3 Caption Mechanism

In MPEG-1, let caption be part of the image is the only way of showing the caption during the movie is playing. But in MPEG-4, "Text" is an independent geometry node. We can record the subtitles of the movie by real characters, instead of images. Users can change the font size, or the language.

3.4 Non-video Movie

Through the timer mechanism of the time sensor and the act of the interpolator node, a route mechanism can generate the effect of geometry objects moving around in a MPEG-4 scene. We can utilize this function to produce pure 3D animation movies. Users can choose by which viewpoint they want to see a movie, and the required bandwidth may be much lower than the original bandwidth required of video. The resolution of the video is fixed, but the effect of instant 3D rendering can be easily appreciated as users' computer performance raised.

3.5 Multimedia Hyperlink

MPEG-4 scenes provide anchor nodes. Users can attain the function of hyperlinking by anchor nodes. For example, at the beginning, there is a huge television wall and each TV set is broadcasting a different program. After a user selects any of the TV set, this selected TV screen will be maximized to a full-screen view for the user to watch.

4. Conclusion

We have designed and implemented the architecture of an MPEG-4 rendering module. To be integrated with other modules, it will play a key role in a MPEG-4 player or a scene editor. Because of modulization on system design, new features can easily be added into the system.

Our research is running toward the third year, which is a part of an industrial academic collaboration project sponsored by National Science Council (NSC) of Taiwan. Although many research institutions around the world

are devoted into MPEG-4 implementation, our laboratory is among the few ones that proposed a total solution from the deliver layer to the composite layer. The demo system is put on the WWW, and could be downloaded at http://www.cmlab.csie.ntu.edu.tw/cml/g/Projects99.html

An alternative version of our system that is wrapped as an ActiveX control is under developing. The MPEG-4 rendering module can then be combined with Internet, and will bring more fancy applications in various fields.

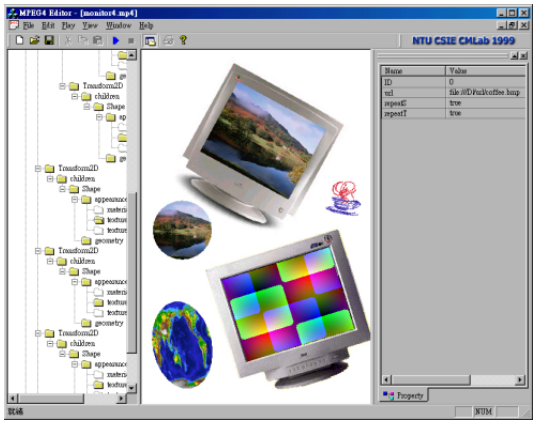

Figure 6: Snapshot of the rendering module, which is integrated with other modules to show 2D nodes. The monitors may contain video objects. A user can individually change the attributes of objects in the scene, e.g. rotate or resize the monitor where a video is attached to it.

5. Future work

Our system hasn't supported all of the MPEG-4 nodes yet, and more functions are under development. If the integration with decoders can be improved, we can accelerate the performance of video playing. For example, let the decoder write data directly into the texture memory. At present, there is only front-half face of the head drawn by the face node, so it seems to be normal only when the head approximately faces to users.

Figure 7: Snapshot of the rendering module, which is integrated with other modules to show 3D nodes. In this picture, there are three moving cubes with each face containing a video, and the background texture is also a video object. The average performance is about 70 fps.

Acknowledgement

This project is partially funded by National Science Council (Taiwan) at CyberLink Co. under the grant NSC88-2622-E-002-002.

References

[1] ISO/IEC FDIS 14496, Information Technology – Generic Coding of Audio-Visual Objects – Part 1: System, Part 2: Visual, International Organization for Standardization, 1998.

[2] ISO/IEC 14772-1:1998, Information technology – Computer graphics and image processing – The Virtual Reality Modeling Language – Part 1: Functional specification and UTF-8 encoding.

[3] Yi-Shin Tung, Ja-Ling Wu, Chia-Chiang Ho "Architecture Design of an MPEG-4 System", Proceedings of IEEE International Conference on Consumer Electronics, June 2000(ICCE 2000), pp. 122-123.

[4] Deng-Rung Liu, Meng-Jyi Shieh, Yu-Chung Lee, and Wen-Chin Chen "On the Design and Implementation of an MPEG-4 Scene Editor", Proceedings of IEEE International Conference on Consumer Electronics, June 2000 (ICCE 2000), pp. 120-121.

[5] I-Chen Lin, Chien-Feng Huang, Jia-Chi Wu, Ming Ouhyoung "A Low Bit-rate Web-enabled Synthetic Head with Speech-driven Facial Animation", in Workshop on Computer Animation and Simulation 2000 (EuroGraphics CAS' 2000), pp. 29-40.

[6] ISO/IEC IS 11172, Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s – Part 2: Coding of Moving Picture Information, Part 3: Audio, International Organization for Standardization, 1991.

[7] ISO/IEC IS 13818, Information Technology – Generic Coding of Moving Pictures and Associated Audio Information – Part 2: Video, Part 3: Audio, International Organization for Standardization, 1994.

[8] CCITT SG XV, Recommendation H.261 – Video Codec for Audiovisual Services at px64 kbit/s, COM XV-R37-E, International Telecommunication Union, August 1990.

[9] ITU-T IS, Recommendation H.263 – Video Coding for low bit-rate communication, International Telecommunication Union, November 1995.

[10] ISO/IEC FDIS 14496, Information Technology – Generic Coding of Audio-Visual Objects – Part 1: System, Part 2: Visual, Part 3: Audio, Part 6: DMIF, International Organization for Standardization, 1998.

[11] Video For Window, Microsoft Corporation.

[12] DirectX Media: Multimedia Services for Microsoft Internet Explorer and Microsoft Windows, Microsoft Corporation, October 1998.

[13] PC Video Synchronization and Playback, Microsoft Corporation, November 1998.

[14] DIS 10918 (JPEG), Information Technology- Digital Compression and Coding of Continuous Tone Images – Part 1: Systems, International Organization for Standardization, 1992.

[15] RFC 2205: Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification. R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin. September 1997.

[16] RFC 1899: RTP: A Transport Protocol for Real-Time Applications. Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick & V. Jacobson. January 1996.

[17] International Standard ISO/IEC 14772-1:1997 – VRML 97: The Virtual Reality Modeling Language.

[18] DirectX 7.0 Programmer's Reference, Microsoft Corporation.