Open Multimedia/Hypermedia Application Development Environment

Sei-Hoon Lee
Dept. of Computer Engineering
Inha Technical J. College, Inchon, Korea
seihoon@dragon.inha.ac.kr

Chang-Jong Wang
Dept. of Computer Science & Engineering,
Inha University, Inchon, Korea
cjwangse@dragon.inha.ac.kr

Abstract : Network technology has been developed rapidly and the use of multimedia data has been increased dramatically. So, we need the multimedia/hypermedia information system that can code multimedia data in order to represent their attributes, save in database and transfer them to the users. In this paper we design distributed multimedia/hypermedia application development environment(MediaADE) so as to satisfy above requirements. The MediaADE use MHEG standard to represent temporal and spatial synchronization, real-time transmission form, final form representation, and interchangeable form of multimedia data. We design MHEG engine to present coded MHEG object to users. Also, we design the interface of communication and database for the purpose of transmission and storing MHEG object in client/server environment. Finally, we design the API that assists programmers to develop distributed multimedia/hypermedia application easily.

Keyword : MHEG, Open System, Multimedia/Hypermedia, Application Development Environment

1. Introduction

It is difficult to standardize the stored and transmitting format of multimedia data through network because multimedia data generally large, are not regular in size, and have difficult in presenting by one keyword[1]. And it is difficult for the existing multimedia/ hypermedia development environment to interchange and reuse information, due to the difference of stored format in dissimilar environments.

Now MHEG(Multimedia and Hypermedia information coding Expert Group) which standardized by ISO/IEC JTC1/SC29/WG12 has defined encoding format for interchanging between multimedia data through network, presenting multimedia/hypermedia objects, spatio-temporal synchronizing, real-time transmission, and presentation of final form, etc[2]. This standard supports functions in order to interchange various type of media. Each media data is coded following international standard such as JPEG, MPEG, etc. MHEG exhibits the presentation of hypermedia and the standardization of exchanging. There are many applications that are possible by using MHEG, such as training/education, simulation/game, sales/advertising, office information system/engineering documentation, etc. To develop these applications, MHEG engine is necessary. The engine has the roles of encoding/decoding multimedia data into MHEG object and interpreting MHEG object to present.

In this paper, we propose the distributed multimedia/hypermedia application development environment based on MHEG standard, MediaADE. It is composed of MHEG engine, database server, and Application Programming Interface(API). In designing database server of client/server environment, ODBC, common database interface, is used in order to solve the discordance of interface between each other database. We also propose the method to solve the discordance of interface between each other network using TCP/IP sockets supported in the most of systems. With the help of our API, programmers can develop applications with easy.

2. Design of MediaADE

2.1 Overview

In this section, we design the MediaADE, an application development environment based on MHEG, the standard of multimedia/hypermedia information representation. MediaADE is made up of three components. One is MHEG engine, another is database server for storing and retrieving MHEG object, and the other is application programming interface(API) for developing application. MediaADE is a client/server environment, locating MHEG engine and API at client, and database at server.

This architecture make it possible for client to maximize calculating ability by interpreting temporal/spatial synchronization information of MHEG object, and for server to minimize overhead. In addition, we locate DBMS and application that interface with database in server to minimize amount of transmission between client and server. Figure 1 represents architecture of MediaADE.

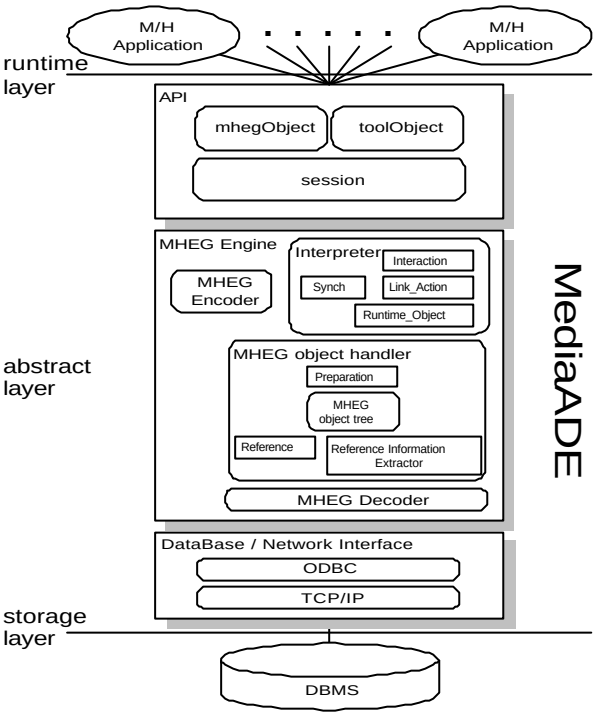

Figure 1. Architecture of MediaADE

As shown above, MediaADE has three layered architecture. Abstract layer includes MHEG engine and manipulates real multimedia/hypermedia information. MHEG engine manages all of hypermedia objects and plays anchoring and presentation specification in Dexter. It encodes media objects to MHEG objects and transmits them to storage layer. And it decodes MHEG objects transferred from storage layer transmits to presentation application. Storage layer has two components. First, hypermedia database which is comprised database for MHEG objects and file system for media files and a interface module. Second, network interface for transferring media files and MHEG objects on communication and database server. We use relational DBMS and design same interface regardless of the kind of DBMS. API provides functions that maps run-time objects generated in interpreter to physical device by using presentation information, interprets user's interaction and transfers it to MHEG engine.

We implement MediaADE based on the Windows NT. In the authoring application, users can communicates only with API. All functions of MediaADE use NT kernel by using Windows NT's Win32 API. Kernel communicates with real physical devices and makes it possible to use the functions of devices.

2.2 MHEG engine

MHEG standard defines coded representation of multimedia/hypermedia information for final-form representation and interchange between multimedia/ hypermedia applications in heterogeneous platforms on network. This standard places in a category interaction, multimedia synchronization, real-time presentation, real-time interchange, and final-form representation. Especially, it focus on real-time information transmission and interchange on network[3].

Using object-oriented approaches, the standard defines MHEG object for coded representation of multimedia/ hypermedia information and that defines MHEG object class for a set of multimedia object with specially consistent structure.

In this paper, we designed the MHEG engine that is composed of MHEG encoder, MHEG decoder, MHEG object handler, and Interpreter. Figure 2 is MHEG engine diagram showing component modules and the flow of MHEG objects through MHEG engine.

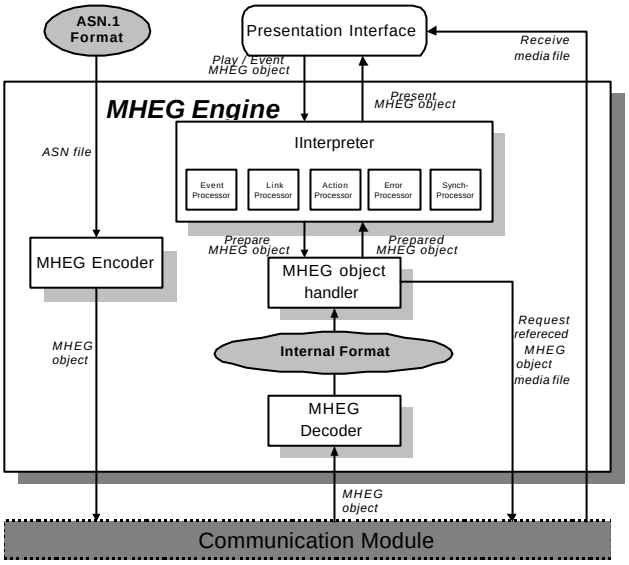

Figure 2. MHEG engine diagram

(1) ASN formatter generates ASN file from each MHEG object. MHEG engine receives ASN file which is described by ASN.1 notation as input and transmits this to MHEG encoder[4, 5].

(2) MHEG encoder transmits the MHEG object of byte-stream form converted by ASN.1 encoding rules to communication module. And then communication module stores MHEG object at local database.

(3) In order to process the presentation of MHEG object, presentation interface transmits at first the message(Play MHEG object) for representation to Interpreter. For interpreting this MHEG object, Interpreter

transmits the message(Prepare MHEG object) for object request to MHEG object handler.

(4) MHEG object handler loads MHEG object through communication module from local database to prepare specified MHEG object and maintains it by using the structure of tree form. When the preparation of object is completed, MHEG object handler transmits the message(Prepared MHEG object) to Interpreter. In case of that prepared MHEG object includes another MHEG object and refers external media files, MHEG object handler request for these to communication module.

(5) Requested MHEG object by MHEG object handler is converted into internal format and then is loaded into MHEG object handler. External media file is stored directly into memory scope without processing through MHEG engine.

(6) Interpreters interpret link, action and synchronization information from prepared MHEG object. And, Presentation Interface represents presentation object that is related with the interpreted information to user through screen.

(7) In presentation, the events from user may happen. These are entered into Interpreter through Presentation Interface. Interpreter interprets appropriately that and has an influence on MHEG object or presentation.

(8) When Presentation Interface completes the presentation process of a specific MHEG object, Interpreter transmits message for the destruction of MHEG object to MHEG object handler and starts for the presentation of new MHEG object.

2.3 Database Server

In this section, we design the database server in client/server environment for the purpose of saving and retrieving MHEG objects. Database server can save and retrieve MHEG objects in database using each MHEG object identifier as a primary key. And database server can send and receive MHEG objects with respond to client's request.

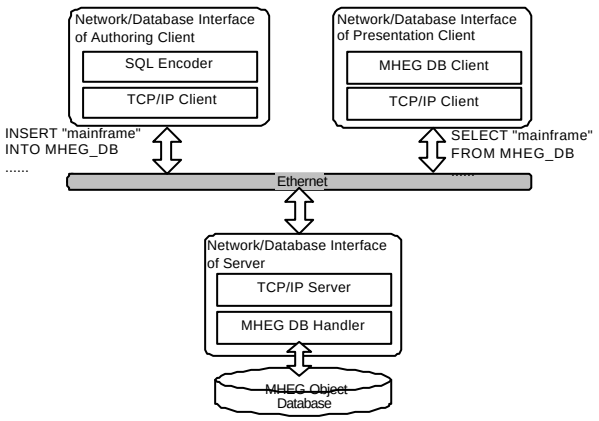

Figure 3. Network/database interface for Client and Server

We use ODBC common database interface and TCP/IP socket network interface to cope with heterogeneous DBMSs which generally use different network protocol and interface.

Figure 3 depicts the interface between client and server. When the client sent query to server, Query Preprocessor in server analyzes the query whether it requests MHEG object or media file. If the query request the MHEG object, MHEG database manager is called, otherwise media file manager is called. After searching relevant objects or files, MHEG database manager and media file manager send these to the client.

Some classes are used in interchange among applications, ACTION, LINK, CONTENT, and COMPOSITE. And we define the frame class which derived from COMPOSITE class. The frame class is used for representing multimedia/hypermedia data as display unit. So, the frame class includes the all MHEG objects which are played in the same screen.

2.4 Application Programming Interface(API)

In this section, we design API using object-oriented method. All objects have their own method for creating, destroying, preparing and executing. Created object is translated by MHEG engine, and shown to user by using its own execution method during presentation.

We define classes of multimedia/hypermedia objects, attributes and methods of those classes for supplying efficient API. That is possible, for object-oriented designing. Figure 4 depicts hierarchical structure of API class.

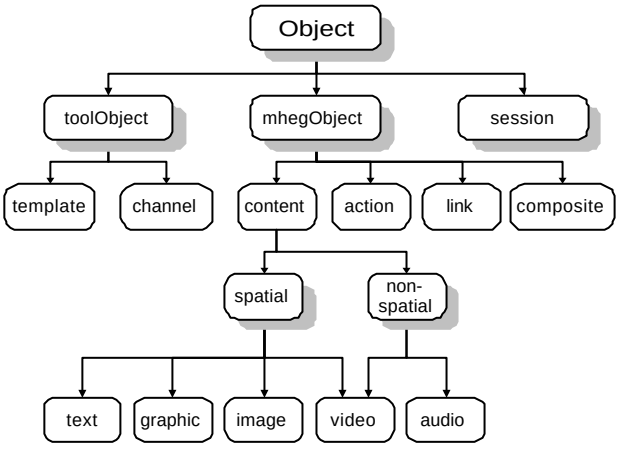

Figure 4. Hierarchical structure of API class

Object class, in topmost level, is higher class of all other classes. It has only one role, that is inheriting attributes. Session class enables users to connect database, to verify their username and password, to give rights associated level of them. There is only one session object in one application, it must be created after

application starts and must be destroyed before application closes. In toolObject class, it is declared that attributes used in channel and template that are used for developing application. Channel and template classes assist author for developing application. Channel maps media to physical devices. Template defines structures of hypermedia for reusing. MhegObject is the highest class of all classes that associate with application programming, storing and transferring. In this class, methods for creating, destroying, preparing and executing are declared. In addition, encoding and decoding method for transferring is also declared. Content class, for media objects, would be played in screen or speaker. Action class defines all behaviour resulted from link operation. Actions are transferred in the shape of message, to MHEG engine in order to be interpreted by engine. Methods about message transfer, object construction and destruction are defined in this class. Link class is executed when users select objects prepared for link operation. Composite class represents a frame. As it is declared recursively and can embed other mhegObject class objects, it manages those objects as a list structure. When a object of this class is executed, it sends messages to its embedded objects.

API functions define as methods of API classes. API functions are separated to creation, preparation, execution and destroy functions. Constructor in C++ language are used for functions associated with creation of objects. When media objects are created, those objects are initialized with handles of their own media files. Preparing functions of objects give attributes associated with created objects. Waiting for starting the execution functions, prepared objects are allocated to channel in order to be shown to users. Execution functions are different from those of each other. Static media(i.e. text, still image) end with only one execution, but dynamic media(i.e. sound, video) is not ended until duration because of it's dynamic properties. Because global attributes which are up to various media form are pre-defined in channels, each media objects just have attributes that are used during they are presented. All objects are executed through channels and execution functions of channels are called in order to execute some objects. Function execute() is inherited from mhegObject class, defining presenting method in accordance to media form. Because of composite class object can embed other classes objects in lower level, execute() function is playing these objects by using depth-first search. Content class objects in leaf level are executed immediately. For link class objects, Ready() function is called, and that class objects are waiting for user's interaction. Selecting link object in ready state, action class objects embedded in selected object are executed. For destroy functions, destructor in C++ is used. In case of composite class object, especially, that kind of object must be destroyed after embedded objects are destroyed.

Session class plays a role of bridge between MediaADE and users. Users cannot use API without session. Application sends request to session API for creation of session object. Then API create session object, allocate it to requested application. After that, all requests of application is made through session object. Session object has a mechanism for processing all of generated events or messages, and methods for handling any events that application and MHEG engine generate. In addition, verification process at connection time and handling request for transferring data from MHEG engine are processed by session object. Session object send data to server in order to verify identifier and password. Session object also create virtual channel from client to database server.

4. Conclusion

In this paper, we design and implement MediaADE as a model of distributed multimedia/Hypermedia applications development environment based on MHEG standard. MediaADE consists of API, MHEG engine, and database server. API is used effectively for developing and presenting. MHEG engine manages and controls these information totally. Database server transmits the encoding information through network and stores them in it. Also it transfers MHEG object when client requires information retrieval.

MediaADE is a client/server environment and it places MHEG engine and applications to client and database to server. This environment makes it possible to minimize overhead of server and to develop a various applications in client site.

MediaADE has the following merits. First, because of its hierarchical structure, connection between heterogeneous system is made easily by changing of interface. Second, MHEG engine is designed as structure in which the internal format after decoding of transmitted MHEG object is mapped into presentation structure of object-oriented method. By using this internal format structure it becomes easier to decode and encode MHEG object at presentation interface, to make dynamic reference to external objects or data files, and to manage sub objects. Third, it is possible to use most database without modification of application program by using common database interface(ODBC). Fourth, interface for application development is a form of object-oriented API and makes it possible to use multimedia object by providing intuitive method to application programmer and user. Fifth, It is possible to reuse multimedia/hypermedia object and useful classes efficiently through object-oriented design method.

The MediaADE would be useful as platform for distributed multimedia/hypermedia applications development on Korea Information Super-Highway.

References

[1] F. Kretz, F. Colaitis, “Standardizing Hypermedia Information Objects”, IEEE Communications Magazine, vol.1, no.5, pp. 60-70, 1992.

[2] ISO/IEC DIS 13522-1 Information technology - Coding of Multimedia and Hypermedia information - Part 1 : MHEG object representation - Base notation(ASN.1), 1994.

[3] R. Price, “MHEG: An Introduction to the Future International Standard for Hypermedia Interchange”, Proceedings of the 1st ACM International Conference on Multimedia, Anaheim(CA), USA, pp. 121-128, 1993.

[4] ISO/IEC IS 8824 Specification of Abstract Syntax Notation One (ASN.1). Second edition. 1990.

[5] ISO/IEC IS 8825 Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). Second edition. 1990.