An Application Framework for Distributed Multimedia System Development

Ying-Hong Wang, Chi-Ming Chung and Hong-Zu Lin

Department of Computer Sciences and Information Engineering
Tamkang University
Tamsui, Taipei Hsien, 25137 Taiwan, R.O.C.
E-mail: inhon@mail.tku.edu.tw

Abstract

In this paper, a distributed multimedia application framework is proposed. The framework is based on Model-View-Controller of Smalltalk-80. Multimedia applications are developing as a popular. However, development of highly interactive multimedia applications for today's high-powered computer is complex and time consuming. An application framework is typically composed of mixture of abstract and concrete classes along with a model of interaction and control flow among the classes. The application framework has "hooks" to allow an application programmer to plug in objects that represent the functionality unique to this application. The distributed processing of multimedia information enables advanced application areas like video conferencing, video on demand and improves the performance in other areas. Thus, it is necessary that an application framework be built for distributed multimedia.

Key words: Distributed multimedia, object-oriented programming, application framework, smalltalk, model-view-controller

1. Introduction

Multimedia applications are developing as a popular and demanding field since the potential of new multimedia technologies is challenging a variety of group user [16]. It is complex and time-consuming to develop a high interactive multimedia application for today's high-powered computers. Recently, although multimedia presentation toolkits are used to improve the developed complexity of multimedia application. However, the functionality of these toolkits is still inadequate for substantially easing the multimedia application building process. There is an apparent lack of existing tools to integrate the multimedia functionality. The existing tools can not provide mechanisms for building interactive, complex multimedia applications. Besides, there is rarely toolkits that are supporting distributed processing of multimedia information [4,7,9,15,13,19].

An object-oriented application framework is typically composed of a mixture of abstract and concrete classes along with a model of interaction and control flow among the classes. The basic idea of an application framework is to take the applications one step further and provide a skeleton for implementing an application or application subsystem. The application framework has "hooks" to allow an application programmer to plug in objects that represent the functionality unique to this application [8]. An application framework is used to deriving new concrete classes from existing classes and configuring a set of objects by providing parameters to each object and connecting them.

Multimedia application presents interaction to the combination of text, graphics, images, audio, video, and animation [2]. Recently, multimedia applications are more popular in many software systems. However, many multimedia applications have similar skeleton of design. Maybe they are using different type of media, or different operations on the media. The application programmers are coding similar applications repeatedly. Therefore, an application framework for developing multimedia

system is necessary. Through a multimedia application framework, the programmers of multimedia application will build a multimedia system more efficiently and less time-consumed.

Some frameworks are proposed for multimedia presentation system [3,5,12,16]. However, most of they can not provide enough high flexibility, high reusability and maintaining. To improve these drawbacks, the Model-View-Controller (MVC) strategy of Smalltalk-80 provides an appropriate idea. Thus, an application framework based on MVC for multimedia application is proposed [17,18]. The primary feature of this multimedia application framework is separating the format structure of various media and their interface operations. This feature makes for the framework keeps flexibility, reusability and maintaining. Furthermore, the proposed framework includes a networking abstract class to handle the relevant operations of distributed, heterogeneous environment.

The paper is organized as follows. In section 2 the basic concept and the advantages of MVC are introduced. Section 3 shows the related researches. The configuration of proposed application framework for distributed multimedia is presented in section 4 and section 5 concludes and gives an outlook on future works.

2. The Basic Concept of MVC

The MVC paradigm originated as a Smalltalk-80 user interface design strategy for representing information [10]. The MVC provides the framework for most Smalltalk-80 user interface [1,11,14]. MVC is a fundamental principle in object-oriented user interface design because it is flexible and can be applied in a variety of ways. MVC framework separates user interface applications into three components. The ***model*** class contains application-specific data, the ***view*** class displays that data on the screen, and the ***controller*** class handles user interactions that affect both models and views.

Whenever the model is changed, its representation must be updated. For this updating to be automatic, a dependency must be established between the view and the model that it represents. In addition to the dependency of the view on the model, the view and the controller must be aware of one another and aware of the model on which they are to work [6]. The relationships between a model, its view and its controller are show in the figure 2-1.

The virtues of MVC are following:

(1).multiple views of the same model are possible;

(2) .views are built from subviews, allowing reuse of components; and

(3) .controllers are easily interchanged to modify application behavior.

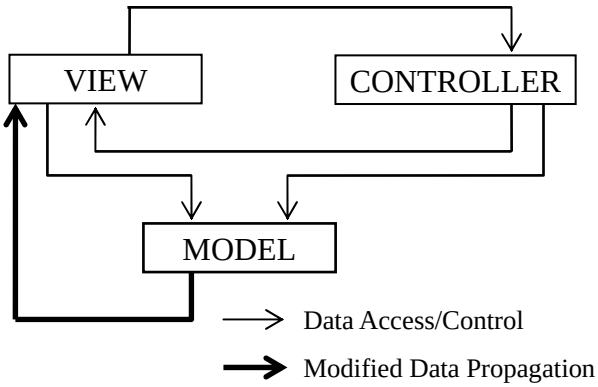

Figure 2-1. The relationships between model, view and controller

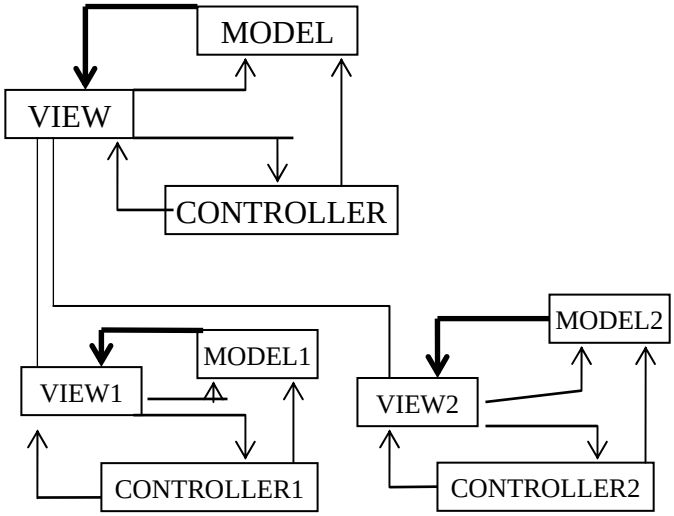

Figure 2-2. A hierarchy of views

The instances of the class *view* are windows. Two of the instance variables of this class are model and controller, which refer respectively to the *model* class and the *controller* class assigned to the view class. When several objects are to be represented in the same view, the view can be divided into several subviews. Thus, a hierarchy of views is defined, since a subview is a completely separate view. To implement this hierarchy, each view has a list of subviews and one superview. The figure 2-2 shows the relationships between models, views and controllers in the case where a view is made up of several subviews.

3. Related Researches

Before we design our framework, we have surveyed both academic researches and industrial software packages. Four previous related researches are introduced as specified below.

A multimedia presentation model is proposed by Blakowski [3]. The model is consisted of three types of object classes. They are:

1. ***Information*** classes: Instances of information classes are primary multimedia objects.
2. ***Presentation*** classes: Instances of presentation classes are objects with the ability to display themselves on a node. They exist only for the purpose of presentation and they are always generated by information objects.
3. ***Transport*** classes: Instances of transport classes are objects in a state for transport, in general with data in a compressed form. If information or presentation objects have to migrate, they always convert themselves into transport objects for the duration of the migration. Transport objects are only used for this purpose.

The main shortcoming is the information class includes all data structures and methods of the media data. Presentation class is inherited from information class and handles the presentation context information. This will restrict the flexibility and reusability. Because of the methods of presentation are included in Information class, the presentation class can not independent to information class.

Natarajan and Slawsky proposed framework architecture for multimedia network [12]. This architecture is applied to telecommunication system. This architecture is consisted of a logical model and an object class. The ***logical model*** defines the logical structure of information networking application. The ***object*** class define all encapsulates data and processing in the networking system. This framework is specific applied in an INA architecture for telecommunication network. Furthermore, it is same as Blakowski's model that Object class includes all data structure and presentation methods. It still restrict the flexibility and reusability.

Vazirgiannis and Mourlas proposed an object-oriented model for multimedia presentation [16]. This model is implemented as an object-oriented hierarchy. The base class of this hierarchy is called item class. The media classes are inherited from item and specific a media type -- text, audio, video, and image, respectively. The operations of presentation are implemented in the programs that use specific media objects. The main advantage of this object-oriented model is separating several operations of multimedia presentation with media classes. There are three types of presentation operation that are multimedia composition, multimedia temporal composition, and multimedia synchronization. However, the other operations of control and presentation are still included in media classes. It restricts the flexibility and reusability to develop multimedia applications. Another flaw is this model has no supporting networking functions.

A multimedia framework is presented by Gibbs and Tsichritzis [5]. This framework comprised four basic classes – media class, transform class, format class, and component class. The media class corresponds a media type; transform class represents media operations in flexible and extensible manner; format class deals with external representations of media data; and component class represents the devices and processes that modify media data. This framework provides flexible ways of extending developed multimedia applications. There are two primary drawbacks it also lacks networking functions and it has no clear definitions of the control classes and presentation classes separately. The second shortcoming will make the same media data supports to a view hierarchy as MVC model.

We also looked at the following commercial products related to multimedia authoring or presentation designs:

1. Authorware by Macromedia, Inc.;
2. Multimedia Viewer by Microsoft;
3. Multimedia Toolbook by Asymetrix Corporation;
4. Hypermedia System by ITRI (Taiwan);
5. Action! by Macromedia, Inc.;
6. Audio Visual Connection by IBM
7. Astound by Gold Disk Inc.
8. MFC class hierarchy of MS Visual C++;
9. Active-X of Microsoft;

None of the above system, however, allows the programmer create distributed multimedia applications easily and flexibly. Thus, an application framework for distributed multimedia is necessary.

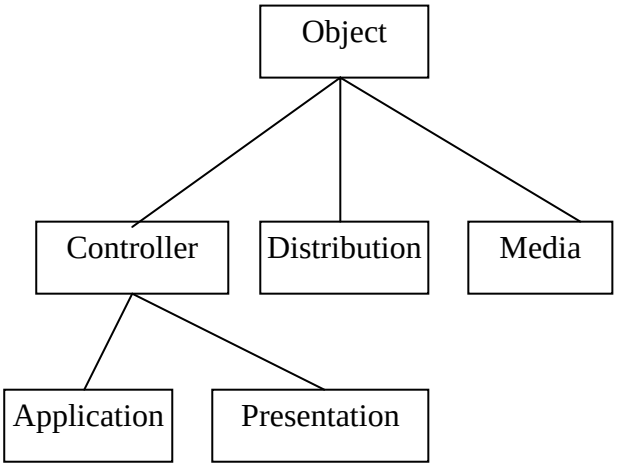

Figure 4-1. The Primary Hierarchy of Distributed Multimedia Application Framework

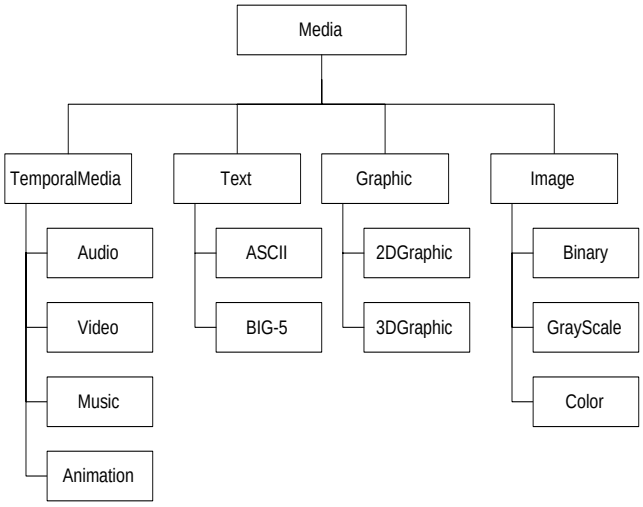

Figure 4-2.The Media class sub-hierarchy

4. A Distributed Multimedia Application Framework

The aim of this paper is to provide a high flexibility, high reusability, and maintaining application framework for developing multimedia applications. Furthermore, the framework supports distributed functionality. The requirements for an adequate application framework are:

- An integrated support for all kinds of media should be given.
- Separating data structure of media data with its control and presentation operations need be provided.
- Transport services have to be offered. The distribution and heterogeneity should be as transparent as possible.

In this paper, an object-oriented application framework for distributed multimedia is presented. This framework specifies interfaces, leaving open their implementation. By adding new classes through specialization the interfaces can be extended. This ability makes the framework well suited to constructing open-ended software environments for multimedia applications.

The distributed multimedia framework comprises what may first appear to be a large number of unrelated classes and methods. There is, however, an overall structure is implemented as an object-oriented hierarchy whose superclass is Object. Object is the root class of the class hierarchy and all other classes shown in figure 4-1 are subclasses of the abstract class, Object. There are five primary abstract classes -- ***media*** class, ***presentation*** class, ***controller*** class, ***distribution*** class and ***application*** class. The media, presentation and controller classes are similar as model, view, and controller of MVC paradigm, respectively.

Object class defines a base method, Clone( ), and three abstract methods: ReadFrom( ), WriteTo( ), and Equal( ). Clone( ) copies the object and its instance variables exactly and can be directly inherited by subclasses. ReadFrom( ) and WriteTo( ) are overridden in subclasses to read and write an object to and from disk. The Equal( ) method is overridden in subclasses to compare objects.

The **media** class abstracts mainly administrative features (like author, format, date of creation/modification etc.) of multimedia objects. The detail class sub-hierarchy is shown as figure 4-2. It supports the MVC paradigm by maintaining a list of **presentations** dependent on its data and broadcast changes to these presentations whenever their data change. Media defines the methods: Changed( ) and Notify( ) for implementing change propagation. In this framework, the media classes such as text and video, are subclasses from the Media class. For this reason, it is usually unnecessary to subclass the Media class directly, unless you need a media of unusual data. The data structures are provided with methods to allow modification of the object's data. When these methods are invoked, they call the Changed( ) function which in turn calls Notify( ). Notify( ) sends the MediaUpdated( ) message to each dependent presentation.

There are two types of media, static media and temporal media. The static media defines common features of time-independent media such as text, graph, and image, and so on. Temporal media defines common features of time-dependent media such as video, audio and animation.

The **presentation** class is an abstract class, which is responsible for the representation of media data. Presentation defines abstract methods of multimedia

operations such as Play( ), Stop( ), Pause( ), etc. When a media sends the MediaUpdated( ) message to a presentation, the presentation sends a PresentUpdated( ) message to all sub-presentations classes that inherited from this presentation class. The design idea of presentation class inherited from controller class is the user is able to handle the specific media data when it is presenting.

The **controller** abstract class is forming a root for any classes that handle input from mouse, keyboard, or other input devices for media data. Controller defines abstract event handling methods, such as DoKeyDown( ) and DoMouseCommand( ), for subclasses like Application class and Presentation class. All methods defined in the controller classes are expected to be overridden in subclasses.

Distribution class is responsible to migrate media data between workstation and network. The distributed environment may be homogenous or heterogeneous. The transport class defines abstract services of multimedia communication. The best suitable and available service has to selected and overridden when distributed multimedia are developed. Currently, it provides Microsoft windows socket functionality. The figure 4-3 presents the winsocket message management structure. The class hierarchy for distribution superclass is represented in figure 4-4.

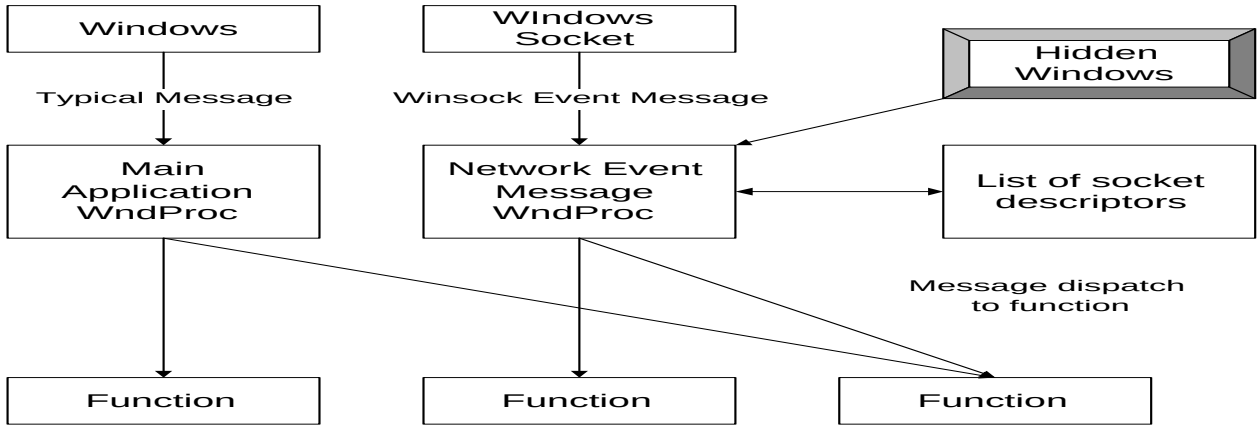

Figure 4-3. Winsock message management structure

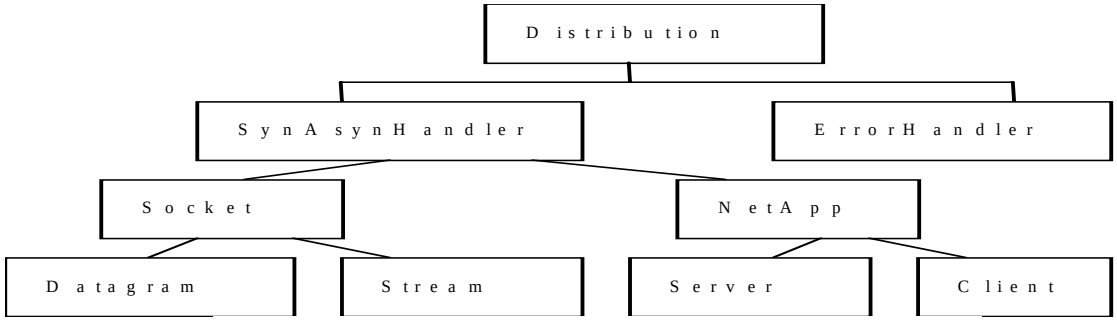

Figure 4-4 The class hierarchy of Distribution Superclass

Application class is a hook for user main program plugging. It runs the main event loop and dispatches events to the appropriate objects to handle them. The programmer will always subclass Application and each program will have one, and only one, instance of that subclass. The abstract method Initialize( ) is usually overridden in the subclass of Application to perform domain-specific initialization. As the application program starts the template method Run( ) of the application class invokes the Initialize and CreateMenu( ) methods and then starts the main event loop.

5. Conclusions and Future Works

Multimedia applications are more popular. The distributed processing of multimedia information enables new application areas like teleconferencing, video on demand, virtual reality, and improves the performance in other areas such as a multimedia CAI course promises as much better learning efficiency [7]. Existing environments and research in multimedia programming have heavily influenced the application framework for multimedia. In this paper, an application framework for distributed multimedia based on MVC paradigm is proposed. The main

features of the framework are (1) supporting networking capability; (2) offering a skeleton for implementing a multimedia application or application subsystem, (3) providing high flexibility, reusability, and maintaining to extend.

There are two directions in the future works. One is to concrete more common classes and builds a library. This library is able to assist programmers implement a multimedia application more efficiently and conveniently. Another is to build a browser that can aid the programmers to find proper classes. By plugging and specializing new classes to these proper classes, this ability makes the framework well suites to constructing open-ended multimedia software and can be ease to extend.

Acknowledgements

This distributed multimedia framework is developing in TESG laboratory of Dept. of CS&IE, TamKang University. It is supported by NSC, R.O.C. The project number is NSC 89-2213-E-032-011.

Reference

[1] Adams, S. S., "MetaMethods: The MVC Paradigm," *HOOPLA*, Vol. 1, No. 4 (1988).

[2] Agnew, P. W. and Kellerman, A. S., "Distributed Multimedia," *ACM press* (1996).

[3] Blakowski, G., "Supporting Multimedia Information Presentation in Distributed Heterogeneous Environment," *IEEE 0-8186-2088-9/90/0000/0029*, pp. 29-35 (1990).

[4] Chung, C. M., Shih, T. K., Kuo, C. H. and Wang, Y. H., "On the Construction of Intelligent Multimedia Presentations," *Information Sciences: An International Journal*, Vol. 89, No. 1&2, Feb., pp. 131-155 (1996).

[5] Gibbs, S. J. and Tsichritzis, D. C., *Multimedia Programming*, Addison-Wesley (1994).

[6] Goldberg,A. and Robson,D., *Smalltalk-80: the Language and Its Implementation*, Addison-Wesley (1983).

[7] Hodges, M., Sasnett, R. and Ackermann M., "A Construction Set for Multimedia Applications," *IEEE Software*, Jan. (1989).

[8] Keh, H., Wittel, W. and Lewis, T. G., "Speedcode: A C++ framework for the Mac. Frameworks," *The Journal of Macintosh Object Program Development*, pp. 26-38 (1991).

[9] Khalfallah, H. and Karmouch, A., *An architecture and a data model for integrated multimedia documents and presentational applications*, Multimedia Systems, Springer-Verlag, pp. 238-250 (1995).

[10] Knolle, N. T., "Variations of Model-View-Controller," *JOOP*, pp. 43-46 (1989).

[11] Krasner, G. E. and Pops, S. T., "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80," *JOOP*, Vol. 1, No. 3, pp. 26-49 (1988).

[12] Natarajan, N. and Slawsky, G. M., "A Framework Architecture for Multimedia Information Networks," *IEEE Communications Magazine*, pp. 97-104 (1992).

[13] Schloss, G. A. and Wynblatt, M. J., "Presentation Layer Primitives for the Layered Multimedia Data Model," *in Proceedings of the IEEE 1995 International Conference on Multimedia Computing and Systems*, May 15-18, Washington DC, pp. 231-238 (1995).

[14] Shan, Y. P., "An Event-Driven Model-View-Controller Framework for Smalltalk," *OOPSLA'89 proceedings*, pp. 347-352, Oct. (1989).

[15] Shih, T. K., Kuo, C. H., Chung, C. M., Keh, H. C., Wang, Y. H., Jiang, D. R. and Pai, W. C., October 12-15, Orlando, Florida, U.S.A., "A Stepwise Refinement Approach to Multimedia Presentation Designs," *IEEE International Conference on System*, Man, and Cybernetics (1997).

[16] Vazirgiannis, M. and Mourlas, C., "An Object-Oriented Model for Interactive Multimedia Presentations," *The Computer Journal*, Vol. 36, N0. 1, pp. 78-86 (1993).

[17] Wang, Y. H., Keh, H. C., Chung, C. M., Lin, H. Z. and Lee, C. J., "Object-Oriented Application Framework for Distributed Multimedia Design," *Proceedings of The Fourth Joint Conference On Information Sciences*, N. C., U.S.A. (1998).

[18] Wang, Y. H., Shih, T. K., Kuo, C. H., "MScript -A Type System and Language for the Composition of Multimedia Presentation Objects," *accepted by The International Institute for Advanced Studies in Systems Research and Cybernetics* (1997).

[19] Wang, Y. H., Chung, C. M., Shih, T. K., Keh, H. C. and Lee, C. J., "Distributed Framework for Multimedia Application Design," *Proceedings of The Fifth International Workshop on Distributed Multimedia System*, Tamshui, Taiwan, pp. 42-47 (1998).

Manuscript Recceived: Oct. 08, 1999
Revision Received: Nov. 10, 1999
And Accepted: Nov. 16, 1999