IMMPS: A Multimedia Presentation Design System

Timothy K. Shih
Tamkang University, Taiwan

Ruth E. Davis
Santa Clara University

Our interactive multimedia presentation development system uses artificial intelligence to specify knowledge inheritance relations between presentation windows. An object-oriented multimedia database organizes resources and presentations, and a database browser facilitates object reuse. The system runs under Windows 95 and can be used for general-purpose presentations or for education, training, or product demonstrations.

As the popularity of multimedia computers increases, software vendors launch more and more multimedia CD-ROM titles. Software systems for producing multimedia titles have also become available, but most CD-ROMs they generate communicate with their audiences in a single direction or provide only limited interaction via pushbuttons or pull-down menus. Few CD-ROM titles support individual tutoring, and different users of the same CD-ROM title often must use the same navigation options even though their backgrounds and learning speeds may differ.

We designed an Intelligent Multimedia Presentation System (IMMPS)[1] that enables a designer to construct an intelligent presentation as a CD-ROM title. Our system provides

1. a presentation specification language,
2. an environment for the specification and learning of audience characteristics, and
3. a multimedia resource and presentation database management system (DBMS).

A rule-based format is used to describe audience background information and enable the system to learn user reactions. A database management system associated with a presentation reuse control interface lets a CD-ROM title designer organize and store multimedia resources and presentations and reuse them partially or fully as desired. IMMPS also supports personalization: The designer can use the system to create a presentation that is customized on the fly based on the changing user information.

We present here the specification language, user interface, and supporting environment for IMMPS. To illustrate these, we describe two sample presentations designed using this system in the sidebar "Sample Presentations" (next page). We also describe the system's features and implementation in detail and discuss how to use IMMPS to design multimedia presentations.

Structuring multimedia presentations

Most presentations produced by multimedia authoring tools are designed as hypermedia documents. Unlike traditional slide presentations, multimedia presentations proceed in a customizable manner via pushbuttons that let the user navigate through different topics.

Presentation windows are related in two ways. First, two windows closely related in content should be linked so that the user can easily move from one to the other by following the links. For instance, a presentation touring Paris may link a picture of the Eiffel Tower with a description—either audio or text—of its history. Second, two presentation windows may share common information. The user's name and background may be stored once and shared among windows needing the information, for example.

Related products and comparison with IMMPS

Several researchers have developed domain-specific presentations using artificial intelligence techniques. For example, Comet (Coordinated Multimedia Explanation Testbed)[2] uses a knowledge base and artificial intelligence techniques to generate coordinated, interactive explanations with text and graphics that illustrate how to repair a military radio receiver-transmitter. WIP[3] generates knowledge-based presentations that explain how to use an espresso machine. Arens[4] integrated knowledge representation systems and a propositional logic theorem prover to create text and map-based illustrations showing a Navy fleet's situations and plans. A piano tutor[5] coordinates video, voice, and graphics to teach beginners how to play the piano.

1070-986X/97/$10.00 © 1997 IEEE

Sample presentations

To demonstrate our system, we designed a number of presentations using IMMPS. We present a Personal Disc Jockey in detail and briefly describe another presentation, a course consultant.

The Personal DJ

This application helps a listener choose songs quickly by classification. The program asks the user about preferences such as period of music (modern, baroque, classical, and so on), style (such as symphony, concerto, chamber, rock, blues, jazz), music author's nationality, and more changeable factors such as occasion (wedding, festival, and so on) or mood (happy, romantic, and so forth). The DJ also asks questions such as, "Are you a music major student, or a musician?" and "Do you need an explanation of the music?" to decide what level of descriptions are necessary. The listener also can choose whether to view a slide show while listening to the music. After the music plays, the DJ asks for keywords so that the listener can reselect the same or similar music quickly in the future.

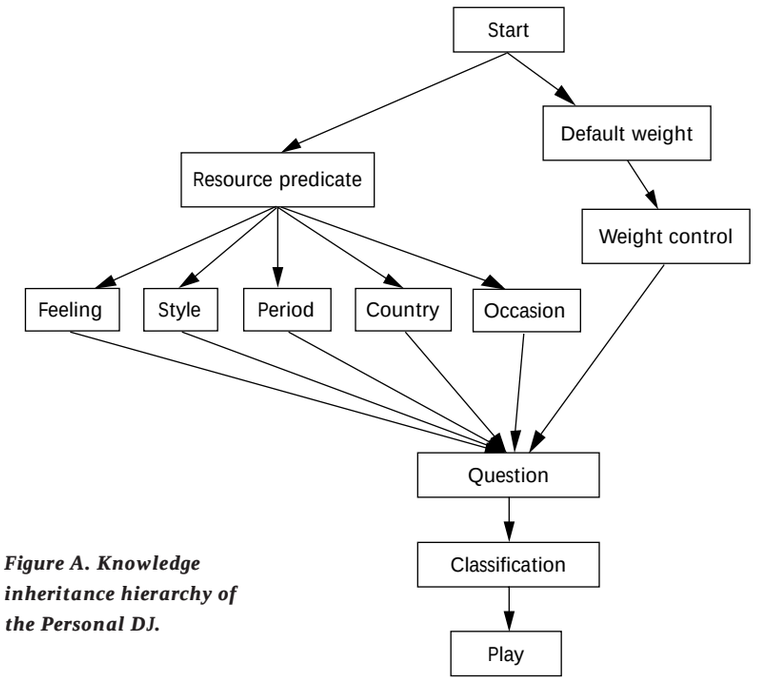

Figure A. Knowledge inheritance hierarchy of the Personal DJ.

Each music presentation contains four parts: the music .WAV file, the bitmapped pictures, the explanation text, and the knowledge. Music presentation pieces are reusable. They are presentation window object classes grouped by inheritance links and aggregation links. The simplified Personal DJ consists of 12 presentation windows, with the information inheritance hierarchy shown in Figure A.

IMMPS does not require that every presentation window contain multimedia resources. Some windows may direct dialogs with the user or simply contain rules and facts to be inherited by their children presentation windows, where the multimedia resources are actually presented. In this example, the only window actually presenting multimedia resources is the Play window on the bottom of the knowledge inheritance hierarchy. When the Play presentation window is fired, the knowledge collection process obtains inference rules and facts declared in various parent windows and starts the inference by calling a query predicate.

Each presentation window has inference rules and facts, with an optional query. Due to space limitations, the inference rules and facts are not listed here. Table A summarizes the content and purpose of each presentation window.

The Resource Predicate window knows which resources may be used in the presentation. In the Personal DJ application, we assign each music resource a weight calculated from the user's expressed preferences in the current or previous interactions and from the inference rules declared by the presentation designer. As the inference proceeds, the weight of each music resource may be modified. Finally, the Personal DJ will choose three music resources of the highest weights for presentation.

The default weights of music resources are declared in the Default Weight presentation window. The Weight Control window contains facts that balance different factors in the weight calculation. The five factors include feeling, style, period, country, and occasion. Some factors may be associated with a certainty number or strength (feeling) while others provide definite information (period and country). These facts are defined accordingly in the five presentation windows.

The Question presentation window asks the user preference questions such as "What is the occasion?" or "What is the nationality of the composer?" The answers will be used in the Classification presentation window to decide the weight of each music resource. The knowledge rules designed in

the Classification window ask the user whether she wants to listen to some previously selected music (or perhaps to use the same criteria but find a different set of selections) and proceeds accordingly. This is one place that learning can occur. Finally, the Play presentation window plays the music.

The Start window is the child window of the system-defined Root window and thus is invoked immediately when the presentation is requested. It sends an `open` message to the Play window. The knowledge rules and facts inherited by Play are collected, and the inference process starts with a query that initiates a dialogue with the user. The rules governing the questioning of the user are defined in the Question window. If the user wishes to listen to previously selected music, the Personal DJ selects music from previously selected resources. These selections are available thanks to the assertion of the results of a previous interaction to the knowledge base. The assert predicate of Prolog is used to add facts to the knowledge set of `Play`. If a new dialogue is desired, the Personal DJ may ask the user four types of questions (true-false, definite answer, fuzzy answer, multiple choice).

After interviewing the user, the Questions window calls a standard Prolog predicate (`setof`) to collect music pieces. (A number of help predicates are also called to collect inference results.) Inference results depend on the rules and facts defined in the five presentation windows of the five factors. The weights of these factors are defined and controlled by the Weight Control window. The resource information associated with each music resource is defined in the Resource Predicate window.

The Classification window selects the first three songs of the highest weights and sends three messages to the Play window. After the music is played, the Classification window asks whether the user wants to save these music selections for replay at a later time.

Disk space limitation kept us from storing entire songs, so the application just samples a short bit of each piece of music. The Personal DJ includes about 80 songs and 120 GIF pictures as well as more than 400 rules and facts. Selecting a song according to a particular mood may be difficult, depending on the data available, but the selection becomes easier if the user provides definite information such as country, author, or period. It is relatively more difficult to construct good rules for the DJ than a more deterministic application such as the study plan consultant described briefly next.

A study plan consultant

Computer engineering students at Tamkang University must file a program of study that specifies the courses they plan to take in completing their degree. Different degree programs have different requirements, and most courses have prerequisites. While individual choices are often simple, managing the complexity of a variety of offerings and prerequisite structures together with program requirements can be daunting. To address this challenge, we designed a study plan consultant as an IMMPS presentation.

About 50 facts represent required courses and 60 facts represent prerequisite relations for the 89 available courses (both requirements and electives). We chose keywords to express students' possible interests, such as programming, theory, hardware, and applications. The 89 courses belong to a number of study areas including multimedia computing, distributed systems, software engineering, and compiler and programming languages; two or more areas may share the same course. All of this knowledge is expressed in about 200 rules, and other rules store student information such as names, study plans previously constructed, and frequency of access to the course consultant.

Our early experience shows that most students enjoyed using this system and were surprised that the consultant could help them lay out an initial study plan even before they talked with their advisors.

Table A. Presentation windows and their functions.

Presentation Window	Content and Purposes
Start	Serves as the Root window of the presentation
Resource Predicate	A number of resource specification facts
Default Weight	The default selection weight of each music resource
Weight Control	Inference rules and facts that can change weights of music resources
Feeling	Facts representing the feeling of each music resource
Style	Facts representing the music style of each music resource
Period	Facts representing the time period of each music resource
Country	Facts representing the nationalities of music authors
Occasion	Facts representing the occasions for each music resource
Question	Rules asking the user questions and adding weights to music resources
Classification	Rules to choose three music resources of the highest weights
Play	Presents music resources and a slide show

Table 1. Comparison of authoring systems.

Authoring Systems	Inference Engine	Layout Design	Navigation Control	Information Sharing	Template or Wizard	Database Support	Resource Editing
Authorware	No	Interactive	Flow line	No	Yes	Yes	No
Director	No	Interactive	Timetable, script language	Yes	No	Yes	Yes
Viewer	No	Progressive control	Script language	Yes	No	No	Yes
Toolbook	No	Interactive	Script language	Use background	Yes	No	No
Hypermedia	No	Interactive	Timetable	No	No	Yes	No
Action!	No	Interactive	Timetable	No	Yes	No	Yes
IMMPS	Yes	Interactive	Message passing	Yes: Knowledge inheritance	No	Yes	No

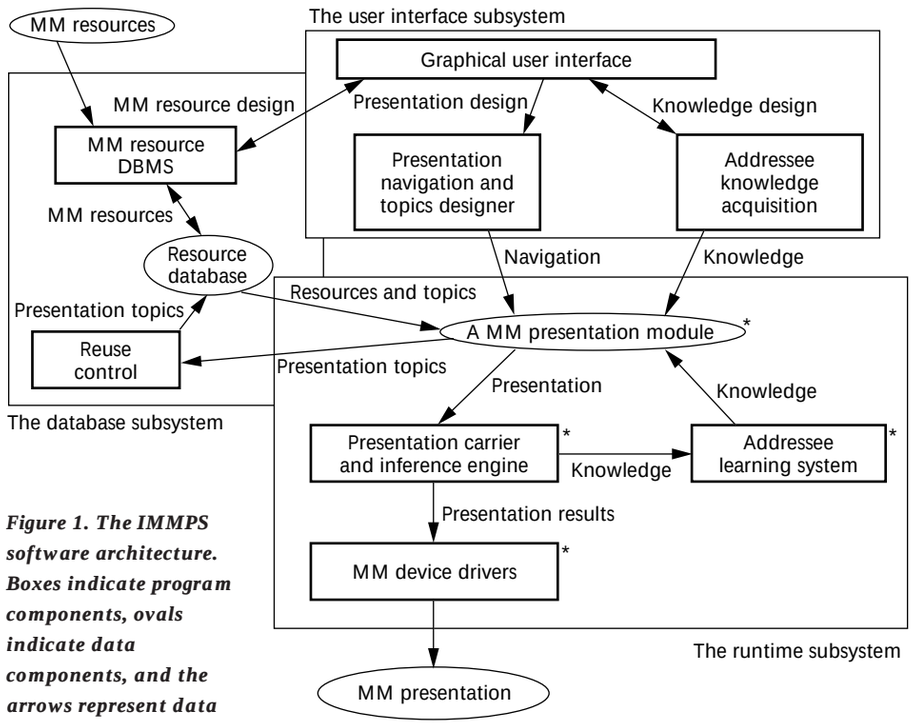

Figure 1. The IMMPS software architecture. Boxes indicate program components, ovals indicate data components, and the arrows represent data flow between components. A multimedia presentation module consisting of the components marked with an asterisk would be part of the runtime system included on a CD-ROM with a presentation.

To support the creation of good multimedia presentations, many researchers suggest starting with a multimedia database that supports fast indexing and synchronization. Little and Ghafoor, for example, presented a mechanism for formal specification and modeling of multimedia object composition.[6] Their work also considered the temporal properties of multimedia resources.

We also investigated the following commercial authoring and presentation design products: Macromedia's Authorware Professional, Director, and Action! packages; Microsoft's Multimedia Viewer; Multimedia ToolBook by Asymetrix; and the Hypermedia Authoring and Playback System by ITRI (Taiwan).

Authorware's event control flow diagram allows the designer to specify presentation objects and controls that can be decomposed into several levels in a hierarchical structure. Authorware provides a simple script language for calculation and data manipulation; other systems also provide script languages and application program interface (API) functions. Hypermedia Authoring and Playback System, Action!, and Director use a time-line table that lets designers drop actions or objects in a particular time slot.

Most systems let designers cut and paste presentation objects or actions via button click. All of these systems let you link presentation objects using a script language. None of the above systems, however, allows a presentation to learn user-selected interactive sequences. Our system uses this information to infer appropriate options to be offered later in a presentation or the next time the same user views the presentation.

Table 1 compares IMMPS to other authoring systems. IMMPS differs from other systems in its knowledge inference engine and its support for reusable objects. Hypermedia System provides a simple resource management system. Director achieves information sharing by declaring a global object and lets users edit simple bitmapped graphics and text. Action! also allows the user to edit simple graphic objects.

Director and Authorware both provide a connection to ODBC (Open Database Connectivity), a database interface introduced by Microsoft and

used in some commercial database systems (mostly relational, such as MS Access and MS FoxPro). These do not achieve efficient presentation object reuse; in our early development of IMMPS, we used ODBC and the FoxPro database system but encountered difficulty designing for object reuse, and the performance of the integrated system (Windows 3.1, VB, ODBC, and FoxPro) was not very good.

The IMMPS architecture

Figure 1 illustrates our system's software architecture. The database subsystem, described in more detail later, consists of a multimedia resource DBMS together with an object reuse control system to facilitate presentation reuse. The system stores and allows reuse of both multimedia resources and presentations. The database is built on top of a commercial object-oriented database and runs under Microsoft's Windows 95.

Two subsystems support the IMMPS GUI: the *presentation navigation and topics designer* and the *addressee knowledge acquisition tool*. The first allows the designer to drag and drop presentation topics such as picture or music boxes in a presentation window. You can also include pushbuttons and hot-spot areas to give the user some control over a presentation. The second subsystem lets you enter knowledge represented as facts and rules similar to those used in Prolog. When a presentation runs, the presentation carrier and knowledge inference engine play the multimedia resources, control the sequence of navigation, and perform knowledge inference. Inference results may cause some facts or rules to be asserted to or retracted from the presentation. The presentation program thus updates itself as communication proceeds between user and computer.

Incorporating presentation perspectives

A designer should consider a presentation from two different perspectives: the control view and the knowledge view. Figure 2 shows the presentation navigation subsystem and knowledge inference subsystem that facilitate these two views. The user's navigation information passes from the window I/O control system to the navigation system, and Microsoft's Media Control Interface (MCI) functions present the presentation content. The navigation subsystem sends messages to the inference subsystem that assert or retract knowledge so that the computer can learn the user's behaviors. The inference subsystem can then send messages to the navigation subsystem that affect further options presented to the user.

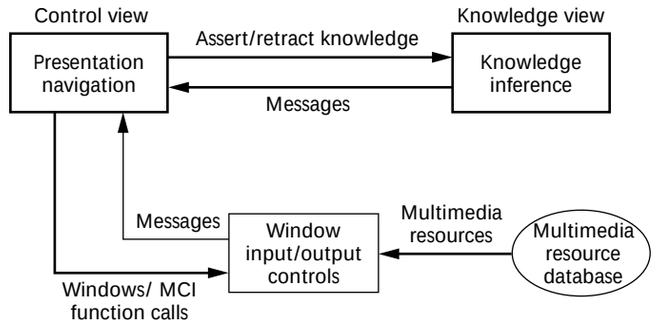

Figure 2. A presentation from different points of view, the control view and the knowledge view.

From a control view, a presentation is a graph with nodes as presentation windows and edges as communication links. From the knowledge representation view, a presentation is a hierarchy of windows sharing information such as the user's identity and background knowledge. We used a directed acyclic graph (DAG) for our knowledge inheritance structure. Figure 3 illustrates these two structures, showing a presentation window (PWin) as a composite object that represents a topic a designer wants to present. A presentation window may contain pushbuttons, one or more multimedia resources to be presented, and knowledge rules ($k1$, $k2$, $k3$, and so on). A message ($m1$, $m2$) with optional parameters can be passed between two presentation windows (or back to the same presentation window). The graph edges representing communication links appear in the figure as thin lines, with message names as labels. The DAG edges representing knowledge inheritance appear as thick lines without labels.

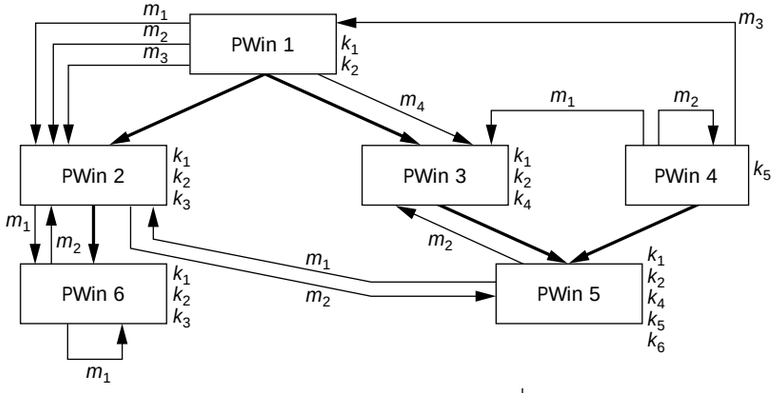

Figure 3. Graph and directed acyclic graph representations of a multimedia presentation.

To the right of each presentation window, Figure 3 shows the knowledge rules that can be used in the presentation window. Knowledge

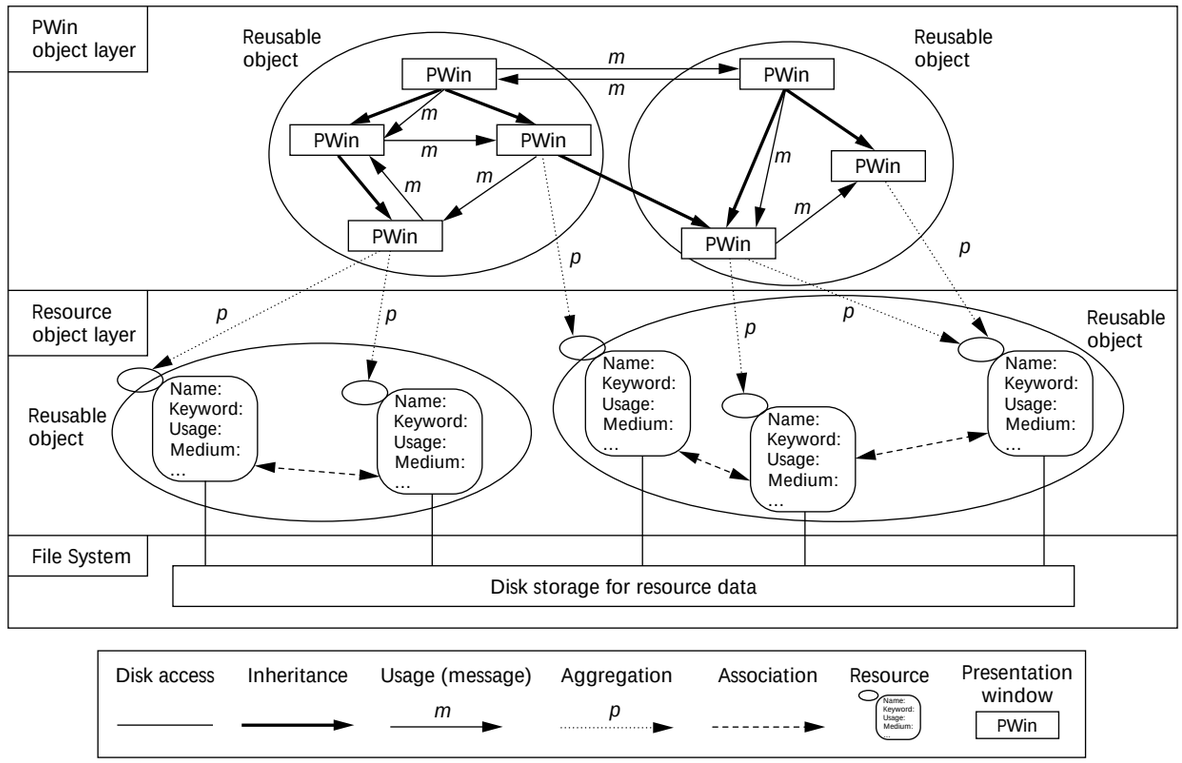

Figure 4. The database hierarchy consists of two layers that handle objects and manage resources.

rules *k*1 and *k*2 are shared among PWin 1, PWin 2, PWin 3, PWin 5, and PWin 6, but they are stored only once in PWin 1. Note that multiple inheritance is also allowed, as PWin 5 inherits knowledge rules from both PWin 3 and PWin 4.

We applied some restrictions to our message passing and knowledge inheritance systems. First, a message passed between two presentation windows has a unique name, source, and destination. Second, a child presentation window inherits from its parent windows all knowledge rules and facts as well as their ancestors, since the relation of knowledge inheritance is transitive. The inheritance architecture is acyclic, that is, a presentation window cannot be an ancestor or descendant of itself. If a child presentation window contains a knowledge rule that has the same name as one of the rules it inherits (directly or indirectly), the rule defined in the child overrides the one from its parent.

An object-oriented multimedia database

High-quality multimedia resources are essential to good multimedia presentation design, as is the ability to reuse predesigned presentation pieces. Our research[7] has emphasized the development of such a multimedia database from the beginning. Strategies for storing multimedia resources follow four basic approaches. They can

- rely on a regular file system,
- use a relational DBMS with the support of an object-oriented interface,
- use an object-oriented DBMS with user interface support, or
- use object-oriented concepts and design the database from scratch.

The first approach relies on users to manage multimedia resources and provides no support for presentation reuse, much less the ability to cut and paste portions of a presentation. The limitations of a general-purpose file system, such as inflexible object composition and sharing, make this strategy inappropriate for multimedia resource management. The second approach relies on a relational DBMS—less useful for organizing multimedia presentations than object-oriented methods.[8-10] The relational and object-oriented DBMS differ in organization and focus, thus making a hybrid construction unnatural for a multimedia database system.[9]

The object-oriented DBMS, though not designed specifically for multimedia data, does provide a binary data type useful for storing pictures, sound, video, and so on. Designing from scratch might result in the most efficient system but probably would not be worth the substantial effort required. We thus took the third approach and used a commercial object-oriented DBMS.

The database hierarchy

Because the object-oriented database offers data sharing via property inheritance and the convenience of object reuse, our DBMS supports the integration of multimedia resources and presentation designs. The multimedia database hierarchy consists of two layers: the presentation window object layer and the resource object layer. Figure 4 gives an overall view of the database. Objects are connected by links, and two object types are implemented: presentation windows and resources. Boxes denote presentation windows and ovals represent resources, their properties given in an attached rounded box.

IMMPS specifies four types of links. An inheritance link, denoted in Figure 4 by a thick straight line, represents knowledge inheritance between two presentation windows. Inheritance links are used to share knowledge with an activated presentation window before any inference in the presentation window proceeds.[1] A usage link, denoted by a labeled thin straight line, represents a message being passed between two presentation windows. An aggregation link, shown in the figure as a labeled dotted line, indicates that the presentation window (at the line's tail) is using the resource (at the arrowhead). An aggregation link connects window objects in one layer with the resource objects they contain from the database's other layer.

An association may also exist between two resources. For example, an animation resource may be associated with a MIDI resource used as background music. An association link appears in Figure 4 as a dashed line between two correlated resources. A resource's actual data is stored in the commercial multimedia file format on the hard disk. In its current implementation, our database still relies on the file system; we are, however, seeking techniques that will allow fast indexing of multimedia resources.

Multimedia objects and their attributes

In the presentation window object layer, each presentation window is associated with attributes:

- a unique *name* for the presentation window;
- one or more *keywords* describing the presentation window;
- *inheritance links* to child windows, which inherit properties from the source presentation window;
- *usage links* to indicate that messages (possibly with parameters) may be sent from source windows to destination windows;
- *aggregation links* to resources used in the source presentation window;
- *presentation knowledge* such as logic facts, rules, and queries used in the presentation window when open[1] (other presentation attributes can also be represented here as logic facts); and
- *presentation window layouts* indicating screen coordinates of resources.

The IMMPS GUI (described later) lets you create presentation knowledge and presentation window layouts.

Multimedia resources can be recorded or captured (via still camera, tape recorder, or video camera), digitized, saved on disk, and reused in different presentations. A resource is associated with attributes:

- a unique *name*;
- one or more *keywords* to describe the resource (for instance, the city name is a keyword for a bitmapped picture of Paris);
- resource *usage* (such as background, navigation, or focus);
- the resource *medium* and the device needed to carry it out (such as sound, video, MPEG, or graphics hardware);
- the *model* of how the resource is presented (such as table, map, chart, or spoken language)
- the resource's *temporal endurance* in a presentation (for example, 20 seconds, or permanent);
- the users' likely *synchronization tolerance* (for instance, a user typically expects an immediate

response after pushing a button for the next page of text but might be able to tolerate a video playback delay of two seconds);

- the resource's *detectability*, that is, its ability to attract a viewer (high, medium, or low);
- the *startup delay*, that is, the time between sending a message and presentation of the corresponding resource, especially important when the resource is on a network-connected remote computer;
- any *hardware limitation* (MPC level 1, level 2, level 3, or other limitations);
- the *version*;
- the *date/time* when the resource file was created;
- the resource's *resolution*, specified by $X \times Y$ screen units (or 0×0 if visual display is irrelevant);
- the *start/end time* for nonpermanent resources, specifically the starting and ending cycles of the video, sound, or other resource used in a presentation (a cycle can be a second, tenths of a second, or a video or animation frame number);
- a logical *resource descriptor* to a physical data segment on disk; and
- *association links* to other resources that have a coexistence relation with the current resource.

Each resource may have many attributes, making it cumbersome to require every query to specify all of them. We simplified query specification by defining several inference rules that allow some attributes to be deduced from others. Each rule describes a relation between two attributes; you thus do not need to specify all attributes of a resource when searching for it.

Some of the rules used in our database include

```
If usage = focus then
  detectability = high
If model = map then medium = picture
If medium = picture then
  temporal_endurance = permanent
If medium = MPEG then
  hardware_limitation = MPEG_Card
```

Object reuse

Object reuse in IMMPS follows the concept of *object groups*, collections of objects that serve as basic presentation units. For example, a presentation segment showing the history of computers and consisting of several presentation windows associated with resources used in them comprises an object group and can be declared a reusable object. You could then reuse this segment in other computer science-related lectures.

Our database defines two types of groups: presentation window groups and resource groups (both denoted by large circles in Figure 4). Both can be stored as reusable objects; you simply declare a group via the GUI. A declaration abstracts an object group into an object class. This presentation window object class or resource object class can be reused when instantiated later on.

When an object group is declared a reusable object class, some links will be lost and some maintained. For the windows in a presentation window group, the usage (or inheritance) links are divided into two parts: internal and external. An internal link has both its source and destination windows belonging to the presentation window group. An external link has its destination window outside the presentation window group.

When a presentation window group becomes a presentation window object class, IMMPS keeps the internal inheritance links, the internal usage links, and the aggregation links. However, the system must lose the heads of all external links of the presentation window group. It marks the heads of such links "unknown," since they may have been moved before one is ready to instantiate the object class for use in another presentation. A similar process applies to a resource group such that when a resource group becomes a resource object class, the internal association links are kept and all external association links are marked "unknown."

Instantiating a class to create an object group requires allocating memory or disk storage for the newly instantiated object group and defining the location of other objects to be accessed through the presentation's external links. You must, however, restore the external usage and inheritance links for a presentation window group instantiation and the external association links for a resource group instantiation. The instantiation process creates a new object group with links connected to other objects in the presentation. (This does not duplicate information that can be shared among instances, such as presentation window layouts or the actual resource data stored on the disk.)

Our presentation reuse control interface and resource browser make managing reusable objects and presentations convenient and systematic. The following section presents some details on using IMMPS to design presentations.

Designing a presentation

You can design a presentation using our internal language or (for nonprogrammers) the system's graphical user interface. We defined a specification language that allows you to design a presentation made up of a collection of presentation windows with messages and inheritance links. When you use the GUI to design a presentation, the system collects information you specify in different windows and produces an internal specification program for each presentation designed. Once generated, the presentation program runs under the presentation carrier and inference engine subsystem.

Specifying a presentation

To create an intelligent presentation, you need to provide

- presentation resources,
- presentation knowledge,
- navigation rules, and
- presentation window layouts.

Presentation resources are standard multimedia files, such as a Windows AVI file (AVI is a Microsoft video file format), collected or created using commercial tools. These resources must be stored in the database before they can be used in a presentation. Presentation knowledge includes user background features (such as familiarity with certain terms) that you want to consider in customizing the presentation. Also, as the presentation proceeds, the presentation knowledge base incorporates the user's demonstrated knowledge. A presentation intelligence window allows you to specify a query, facts, and rules for each window of the presentation. For example, a knowledge rule might take the following form:

```
case Predicates -> Predicate $
Predicates -> Predicate $
…
Predicates -> Predicate $
true -> Predicate
```

where each **Predicate** represents a condition or relationship that may or may not hold. **Predicates** is a list of one or more such conditions, separated by commas, representing the conjunction of those conditions. That is, a predicate list holds if and only if all predicates in the list hold. The case rule is an extension of an if-then rule. If any of the **Predicates** (to the left of **->**) holds, the corresponding predicate (following the **->** in the same line) will also hold.

The last branch, **true -> Predicate**, is optional and allows a default case to be declared. A fact is simply a stated **Predicate**. IMMPS translates these Prolog-like knowledge rules and facts to Prolog statements before knowledge inference proceeds. Regular Prolog statements are also allowed in the knowledge declaration.

Knowledge inheritance and system learning

Rules and facts represent knowledge that can be inherited. You declare knowledge inheritance relations by specifying the parent window of each presentation window. A special presentation window, Root, is a place holder; only one presentation window (the first to be invoked in a presentation) has Root as its parent. Each presentation window must have a unique starting query.

The learning process of a presentation in progress relies on two Prolog predicates: the assert and the retract system predicates. You use these predicates to anticipate and react to user actions, adding or subtracting knowledge rules (or facts) from the presentation knowledge base.

To facilitate user interaction, we allowed four kinds of dialogue boxes to be invoked in the knowledge rules:

1. the user is asked a yes/no question and indicates the answer by clicking on a yes or no button;
2. the user is asked to supply a definite (brief) answer, which is then bound to a text string;
3. the user is asked to supply a degree of certainty or strength of agreement with a statement; or
4. one or more multiple choice questions may be posed at once, with allowable answers displayed in a window.

An appropriately designed interaction can make the presentation more responsive to the needs of individual users.

Figure 5. The main window of IMMPS.

We allow designers to take a hypermedia traversal approach to presentation navigation. You may define three object types for each presentation window: button definitions, topic definitions, and message definitions. A button definition associates each button with a property list and one or more actions to be applied when the button is pushed. Typical actions include sending a message, enabling or disabling a button, asserting or retracting information, and calling a procedure such as an external C procedure that makes an MCI (Media Control Interface) call to the underlying Windows 95 multimedia drivers.

A topic definition resembles a button definition except that a topic cannot send out a message. However, a topic can receive a message (play, stop, forward) and respond with an appropriate action.

A message definition is a collection of message-action pairs specifying a message name, its parameters, and a list of actions. Each pair has the following format:

```
on message: Message_name(Parms) do
   [Actions]
```

A message-action pair might be defined as

```
on message: play(AVI_file_name) do
   [proc_call("play_resource(AVI_
   file_name)")]
```

where `AVI_file_name` is the name of a video file.

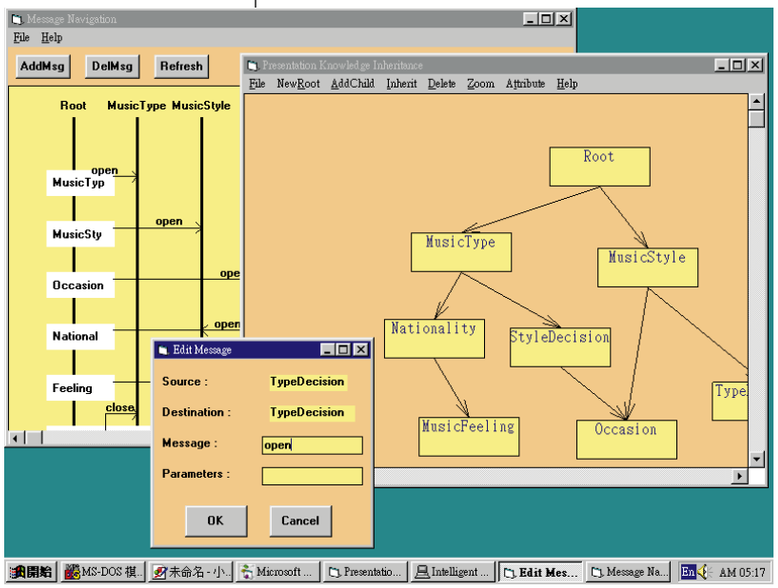

Figure 6. The presentation knowledge inheritance window and the presentation message passing window.

Graphical user interface

Figure 5 shows the IMMPS main window, which lets you drop various presentation topics such as text or audio in the design area shown in the subwindow (labeled Root in the figure). Tool bar functions permit saving or loading files, creating new presentation windows, and running a presentation. You can use the main window in conjunction with the message passing and knowledge inheritance windows (shown in Figure 6)—bring these up by clicking on Navigation and Intelligent, respectively, in the main window menu bar. The functions shown in the inheritance window menu bar permit construction of the knowledge inheritance hierarchy; declaring inheritance relations creates inheritance links among presentation window objects.

To add a usage (or message) link, access the presentation message passing window. A vertical bar represents the presentation window indicated by the name on the top of that bar. Messages are passed when the user pushes a button in a presentation window (or as a side effect of knowledge inference[1]). Pushbuttons appear as boxes on vertical bars. Each message, its name displayed as the label of a usage link (shown as a horizontal arrow), is associated with zero or more parameters entered in the Edit Message window shown in Figure 6.

You can use a presentation intelligence window to add rules or facts (discussed elsewhere[1]) to the knowledge set of a presentation window. In an Action Control window you can specify the actions[1] that should take place when a presenta-

tion window object receives a message. The Presentation Button Controls window lets you specify messages to be sent out when a button is pushed.

The multimedia resource browser (shown in Figure 7) lets you retrieve resources ready for inclusion in presentations. Select or enter resource attributes in different components of the window. When you push the topic button, another dialogue box comes up that lets you select a topic of a presentation window to link with the highlighted (or available) resource in the list box. The resource browser also allows you to preview various kinds of resources. The video window on the upper left side permits video previewing; the picture box on the upper right side lets you view text files, pictures, or animations. The MCI control bar shown below the picture box lets you listen to sound and MIDI files. Specialized windows assist with object reuse and the creation of association links between resources.

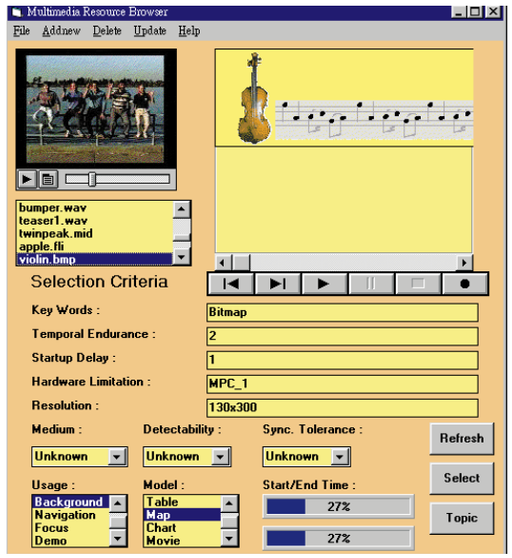

Figure 7. The multimedia resource browser.

IMMPS implementation environment

We used Arity Prolog as the underlying implementation language of our inference engine. We also used some C functions as stub functions that communicate between Prolog predicates and Basic procedures. The GUI is written in Visual Basic (VB). The database management system is implemented in Visual C(VC). About 9,000 lines of VB, 4,000 lines of VC, and another 4,000 lines of Prolog code were implemented. The supporting software includes the ObjectPro/ODB database management system and Windows 95/3.1. Figure 8 illustrates the implementation environment of our prototype system.

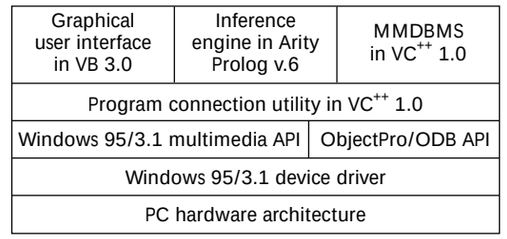

Figure 8. The implementation environment of IMMPS.

We spent about two years developing the system. In the first half year, we surveyed other presentation systems and research before documenting a draft specification. The functional specification and design of IMMPS took about nine months. Two PhD students and three Master students spent almost a year in the implementation before the prototype was tested by a few undergraduate students for one month.

Conclusions and continuing work

IMMPS contributes important new ideas in intelligent tutoring, and we are seeking ways to improve it. For instance, we might use a truth maintenance system technique to check the consistency of knowledge rules in a presenter's design. We are designing the second version of our GUI to be more friendly and powerful. Also, we are analyzing commonly used presentation rules. IMMPS does not provide templates or wizards that could guide a designer in developing a presentation; we hope to provide these in a future version. IMMPS also does not come with resource editing tools such as a bitmapped graphics editor, video editor, and sound editor, which typically are included with multimedia hardware products.

We are also looking at the semantic modeling of multimedia presentations and see the need to consider the temporal properties of multimedia resources. Using interval temporal logic will let us express the temporal properties of multimedia resources and solve application-level synchronization problems. Finally, future work will investigate database indexing techniques for the fast retrieval of multimedia data. MM

Acknowledgement

We thank Ying-Hong Wang, Chun-Chia Wang, Kuan-Shen An, Wen-Shan Yu, and Julian B. Chang for their implementation of IMMPS. The prototype system would not have been successfully constructed without their hard work.

References

1. T.K. Shih et al., "Formal Specification of Multimedia Authoring," *Proc. 1996 IEEE Workshop on Multimedia Software Development*, IEEE Computer Society Press, Los Alamitos, Calif., 1996, pp. 128-137.
2. S.K. Feiner and K.R. McKeown, "Automating the Generation of Coordinated Multimedia Explanations," *Intelligent Multimedia Interfaces*, M.T. Maybury, ed., American Assn. for Artificial Intelligence, London, 1993, pp. 117-138.
3. E. Andre, "WIP: The Automatic Synthesis of Multimodal Presentations," *Intelligent Multimedia Interfaces*, M.T. Maybury, ed., American Assn. for Artificial Intelligence, London, 1993, pp. 75-93.
4. Y. Arens, "Presentation Design Using an Integrated Knowledge Base," *Intelligent User Interfaces*, J.W. Sullivan and S.W. Tyler, eds., ACM Press, New York, 1991, pp. 241-258.
5. R.B. Dannenberg and R. L. Joseph, "Human Computer Interaction in the Piano Tutor," *Multimedia Interface Design*, M.M. Blattner and R.B. Dannenberg, eds., Addison-Wesley, Reading, Mass., 1992, pp. 65-78.
6. T.D.C. Little and Arif Ghafoor, "Synchronization and Storage Models for Multimedia Objects," *IEEE J. Selected Areas in Comm.*, Vol. 8, No. 3, 1990, pp. 413-427.
7. T.K. Shih, C-H. Kuo, and K-S. An, "An Object-Oriented Database for Intelligent Multimedia Presentations," *Proc. IEEE Int'l Conf. on Systems, Man, and Cybernetics*, IEEE Computer Society Press, Los Alamitos, Calif., 1996, pp. 2904-2909.
8. C.Y.R. Chen et al., "Design of a Multimedia Object-Oriented DBMS," *ACM Multimedia Systems*, Vol. 3, 1995, pp. 217-227.
9. T. Ozsu et al., "An Object-Oriented Multimedia Database System for a News-on-Demand Application," *ACM Multimedia Systems*, Vol. 3, 1995, pp. 182–203.
10. M. Vazirgiannis and C. Mourlas, "An Object-Oriented Model for Interactive Multimedia Presentations," *The Computer J.*, Vol. 36, No. 1, 1993, pp. 79-86.

Ruth Davis is professor of computer engineering and David Packard Mentor Scholar at Santa Clara University. She earned her BS and MS degrees in mathematics at SCU and San Jose State University and a PhD in information sciences at the University of California, Santa Cruz in 1979. She received the ACM Doctoral Forum Award for her dissertation, *Generating Correct Programs from Logic Specifications*. Her technical interests include formal methods in software engineering in a variety of contexts. She is a senior member of IEEE and SWE (Society of Women Engineers), and a member of the IEEE Computer Society, ACM, and Computer Professionals for Social Responsibility (CPSR).

Timothy Shih is an associate professor in computer engineering at Tamkang University, Taiwan. His research interests include multimedia computing, software engineering, and formal specification and verification. Shih received his BS and MS in computer engineering at Tamkang University in 1983 and California State University, Chico in 1985, respectively. He received his PhD in computer engineering from Santa Clara University in 1993.

Contact Davis by e-mail at rdavis@sunset.scu.edu. Contact Shih at the Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taipei Hsien, Taiwan 251, ROC.