KMS: A Distributed Hypermedia System for Managing Knowledge In Organizations

Robert Akscyn, Donald McCracken, Elise Yoder

Knowledge Systems Incorporated
4750 Old William Penn Hwy, Murrysville, PA 15668

ABSTRACT

KMS is a commercial hypermedia system developed by Knowledge Systems for networks of heterogeneous workstations. It is designed to support organization-wide collaboration for a broad range of applications, such as electronic publishing, software engineering, project management, computer-aided design and on-line documentation. KMS is a successor to the ZOG system developed at Carnegie Mellon University from 1972 to 1985.

A KMS database consists of screen-sized WYSIWYG workspaces called frames that contain text, graphics and image items. Single items in frames can be linked to other frames. They may also be used to invoke programs. The database can be distributed across an indefinite number of file servers and be as large as available disk space permits. Independently developed KMS databases can be linked together.

The KMS user interface uses an extreme form of direct manipulation. A single browser/editor is used to traverse the database and manipulate its contents. Over 85% of the user's interaction is direct--a single point-and-click designates both object and operation. Running on Sun and Apollo workstations, KMS accesses and displays frames in less than one second, on average.

This paper describes KMS and how it addresses a number of hypermedia design issues.

INTRODUCTION

For the past six years, we have been developing a commercial hypermedia system (KMS) based on our previous research with the ZOG system at Carnegie Mellon University. This paper describes KMS and how it addresses a number of hypermedia design issues, particularly issues concerning what data model to use. Section 1 provides some historical background on ZOG and KMS. Section 2 gives an introductory description of KMS. Section 3 describes some hypermedia design issues and how KMS addresses them. Section 4 concludes by reiterating the importance of the KMS data model--how it permeates the overall design of the system.

1. BACKGROUND

We have been developing hypermedia systems for over a decade, first at Carnegie Mellon University with the ZOG Project, and now at Knowledge Systems with the commercial development of our Knowledge Management System (KMS).

We have been zealous users of hypermedia. While developing ZOG and KMS, we used them extensively for our work--logging over 10,000 person-hours as users, and creating over 50,000 frames (nodes). Throughout this period we have applied what we have learned to iterate the design of these systems, creating scores of intermediate versions.

Early ZOG efforts at CMU. Work on ZOG began at CMU in 1972. What we now call ZOG-1 was developed for a summer workshop for researchers in cognitive science. It allowed the participants to easily interact with one another's simulation programs by providing a uniform menu-selection interface. After the workshop, ZOG-1 was shelved because the technology used was inadequate (300 baud hard-copy terminals!). Work on ZOG was rekindled in 1975, after Allen Newell and George Robertson observed the PROMIS system at the University of Vermont. PROMIS was a menu system based on rapid-response touch-screen terminals, applied to the task of hospital management [Schu79]. Newell and Robertson were struck by the qualitative difference of the rapid-response PROMIS interface over traditional human-computer interfaces, and began an ONR-sponsored research project to study the general characteristics of large, rapid-response, menu-selection systems. From 1975 to 1980, the ZOG group developed a series of ZOG versions for PDP-10s, Vaxes, and even for an experimental multi-processor machine, C.mmp [Robe81b].

ZOG on the USS CARL VINSON. By 1980 we felt ZOG was sufficiently mature to be tested in the real world. So we embarked on a major ZOG application project--to build a computer-assisted management system for the Navy's newest nuclear-powered aircraft carrier, the USS CARL VINSON. This was a joint project between the ZOG Group at CMU and the officers of the CARL VINSON. The development phase of the project ended in March, 1983, when the CARL VINSON left on her first deployment with a distributed ZOG system running on a network of 28 PERQ workstations. ZOG supported four applications:

- On-line policy manual (Ship's Organization and Regulation Manual)
- Interactive task managment system (for analyzing and tracking complex tasks)
- On-line maintenance manual with interface to videodisk (for weapons elevators)
- Interface to the AirPlan expert system (developed at CMU by McDermott, et.al.)

We continued to work with the crew of the CARL VINSON until the end of the ZOG project in December 1984. The project and some of the lessons we learned are described in [Newe81], [Aksc84b] and [Newe85].

Knowledge Systems and KMS. In 1981, at the request of Westinghouse, we formed a company (now called Knowledge Systems) to develop a commercial version of ZOG. Westinghouse was interested in applying ZOG technology to the problem of providing operators of nuclear power plants access to emergency operating procedures. This initial work led to our first commercial version of ZOG (called KMS) in 1983. Since then we have worked with a number of other organizations to apply KMS to various large-scale knowledge management tasks.

Applications we have explored. We have found ZOG and KMS to be useful in a surprising number of applications over the years. At Knowledge Systems we use KMS for almost everything we do with computers. Below, we list the applications we have explored. More information about these applications can be found in [Aksc84a], [McCr84], and [Newe81].

- Electronic publishing
- On-line technical manuals
- On-line instruction manuals
- On-line help for other software
- Project management
- Issue analysis
- On-line policy manuals
- Group presentations via large screen projectors
- Financial modelling
- User interface to videodisk-based materials
- User interface to expert systems
- Software engineering
- Computer-assisted foreign language translation
- Operating system shell

2. INTRODUCTION TO KMS

Our primary design goal for KMS is to create a general-purpose software environment that helps an organization manage its knowledge. We are concerned not only with the productivity of the individual, but also the productivity of groups--from small workgroups up to an entire organization. We are especially concerned about the problem of building and maintaining large databases, since this activity is often the principal bottleneck in many uses of computers.

We are shaping KMS to exploit what we believe will be the dominant architecture for organizational computing environments of the 1990's: wide-area networks of large-screen, diskless workstations. We believe networked workstations offer quantum leaps in productivity over what is possible with today's personal computers.

In this section, we give a brief overview of the two major components of KMS--its data model (how knowledge is represented in KMS), and its user interface. Additional details about KMS are woven into the following section on hypermedia design issues, where we describe how KMS addresses particular design issues.

KMS data model

A KMS database consists of a set of interlinked, screen-sized workspaces. These workspaces, called *frames*, may contain any arrangement of text, graphics, and image items. Each individual text item within a frame can be linked to any other frame. As with ZOG, text items may also activate programs. These programs may range from atomic KMS operations to lengthy KMS animations (written in the KMS Action Language), as well as conventional programs that normally run from the operating system.

Frame format. Strong conventions have evolved for the format of frames. These conventions are illustrated by the example frame in Figure 1.

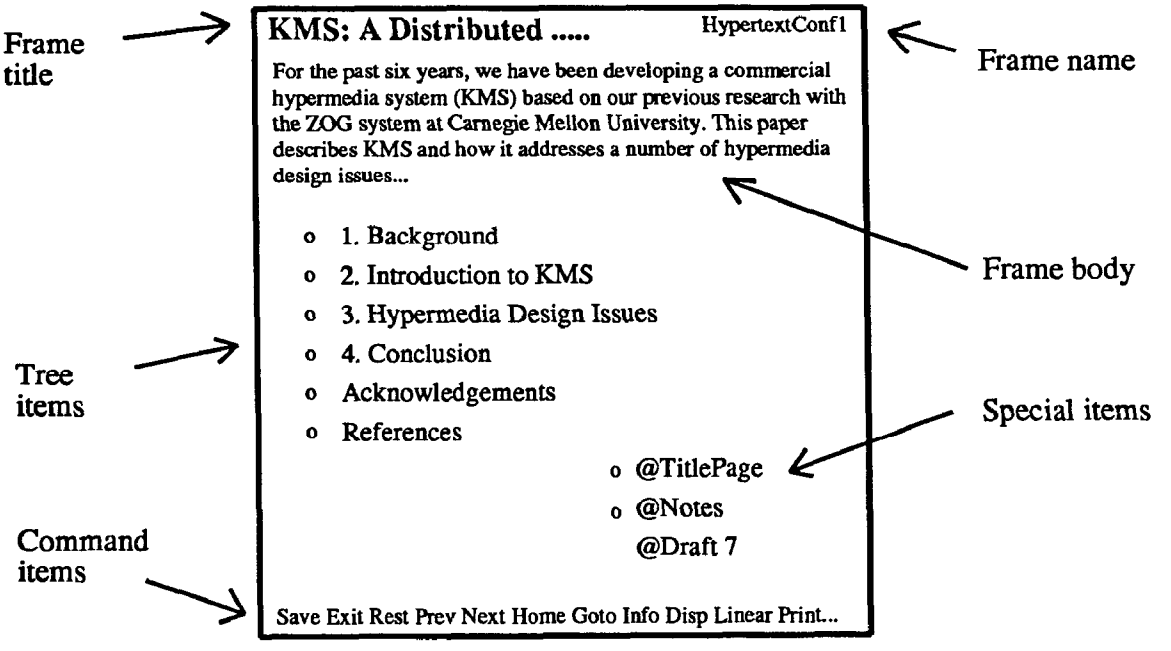


Figure 1: Example KMS frame

Every frame has a unique name. This *frame name* consists of two parts: an alphabetic part and a numerical part. The alphabetic part is the name of the *frameset* of which the frame is a member. (A frameset is the set of frames related to a specific topic as defined by the user, such as an individual document, software program, or project. Users are free to create a new frameset whenever they create a new frame.) The numerical part of the frame name is provided automatically. The frame name in the example above is "HypertextConf1".

The *frame title* is located in the upper left corner of the frame. It provides a short description of the knowledge contained in the frame. The frame title in the example is "KMS: A Distributed ...".

Next comes what we call the *frame body*. For upper-level frames in a hierarchy, the frame body is usually a short paragraph expanding on the topic described by the frame title. Lower-level frames usually contain more text, graphics, and images.

Below the frame body are text items called *tree items*, which are linked to lower-level frames. When a link is present, an item is displayed with a small circle to its left. The tree items in the example are:

- 1. Background
- 2. Introduction to KMS
- 3. Hypermedia Design Issues
- 4. Conclusion
- Acknowledgements
- References

On the lower right side of the frame are the *special items*, which begin with the character "@". Special items are used for miscellaneous purposes such as notes, comments, and document formatting keywords. As a result, special items have the connotation of being meta-level items. Special items are linked to other frames where appropriate. The special items in the example are:

- @TitlePage
- @Notes

@Draft 7

At the bottom of the KMS window (not actually part of the frame) is a customizable command menu containing *command items*. The default menu shown here contains 19 items. These items are used to invoke programs, from simple KMS operations to complex external programs. Invoking programs is discussed further in the following section on the "KMS User Interface."

Linked frames. KMS permits a frame to have an unlimited number of linked items, each of which may be linked to any frame (including itself). This flexibility permits KMS databases to have any structure the users desire, even a 'bowl of spaghetti' structure. In practice, however, KMS databases usually have a hierarchical backbone. This backbone is embellished with meta-level constructs in the form of *special items* such as user comments, formatting instructions, and cross-reference links. The use of hierarchy as the principal organizing paradigm is a strong factor in helping KMS users remain oriented.

Figure 2 illustrates a fragment of a KMS database. In this example we show part of the hierarchy of frames representing this paper.

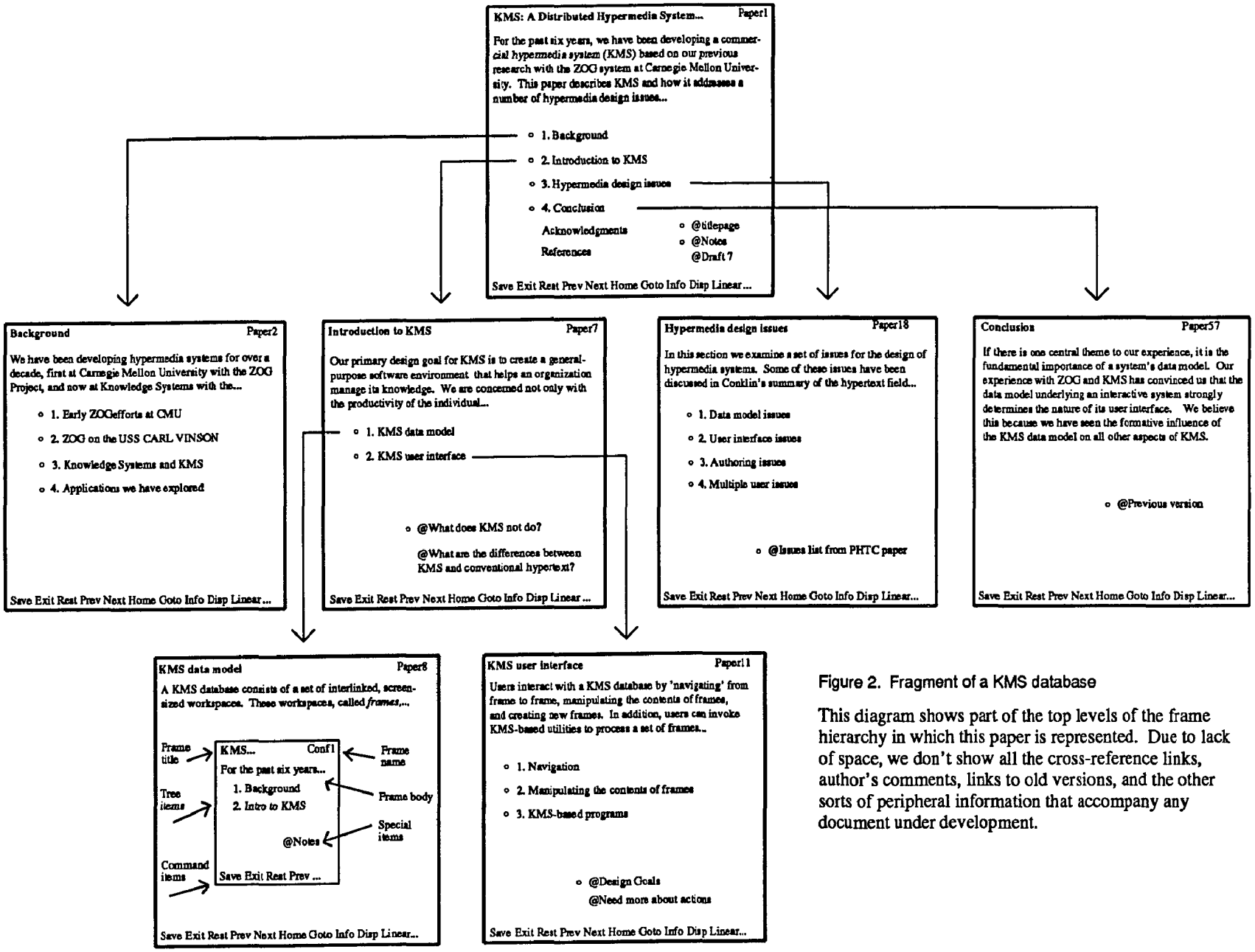


Figure 2. Fragment of a KMS database

This diagram shows part of the top levels of the frame hierarchy in which this paper is represented. Due to lack of space, we don't show all the cross-reference links, author's comments, links to old versions, and the other sorts of peripheral information that accompany any document under development.

KMS User Interface

Users interact with a KMS database by 'navigating' from frame to frame, manipulating the contents of frames, creating new frames, and invoking programs.

Navigation. The central metaphor in KMS is that the database is like a universe of connected spaces through through which users rapidly travel, like pilots navigating spacecraft in the real universe. Users navigate from frame to frame by pointing the mouse cursor at an item linked to another frame and clicking one of the mouse buttons (KMS uses a 3-button mouse). KMS accesses the designated frame and displays it in the same window in less than one second, on average. Thus, KMS is replacing the currently displayed frame as though the user had physically travelled to a new location in the real universe.

Manipulating the contents of frames. A user can directly manipulate the contents of a frame at any time. This is done by moving the mouse cursor to the desired location on the screen and clicking buttons on the mouse, or in the case of text input, typing keys on the keyboard. There is no mode boundary between navigation and editing.

KMS makes use of contextual distinctions so that users can invoke the most frequent operations with a single point-and-click of the mouse. The location of the cursor (for instance, whether it's in empty space or inside a text item) determines which functions are available via the mouse buttons. As an aid to users, the cursor images include text labels indicating which function is currently available on each button.

Three years of testing this approach shows that users have little trouble knowing what functions the mouse buttons perform--they can always read the labels. KMS novices rely heavily on these cursor labels to learn the system. KMS experts continue to rely on the labels, but their attention is sublimated to a perceptual level. Figure 3 illustrates several KMS cursors.

Reduced command set. The move command is an example of how we have tried to streamline the KMS user interface by unifying multiple operations into a small set of commands. Pointing the cursor at an object causes the "Move, Copy, Delete" cursor to appear. Clicking the Move button at this point causes the cursor to latch onto the object. The user can drag the object around--not only within the current frame, but also across the window boundary into the other frame. In addition, while in this dragging state, the user can still perform some top-level commands, such as typing text, moving to other frames, and even creating new frames. This unification eliminates the need for a clipboard construct and the operations of cutting and pasting.

This single Move operator can perform the KMS equivalents of the following functions in other computing environments:

- Rearranging text and graphics within a diagram or page
- Moving a text string to another location in text
- Rearranging the order of sections in a document
- Moving data from one file to another
- Moving a directory or file to another directory

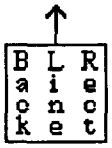
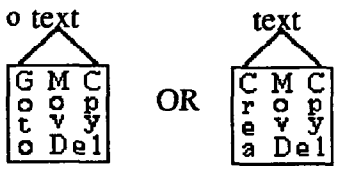
Cursor	Context	Available Functions
	When the cursor is not pointing at any item on the frame	Left: Go **back** to the previous frame Center: Create **line** or point Right: Create **rectangle**
 if linked if not linked	When cursor is inside of a text item (the cursor will only sit in between characters)	Left: **Goto** frame (if linked) **Create** frame (if not) Center: **Move**--latch onto the item(s) for movement Right: **Copy** the item(s) Center & Right: **Delete** the item(s)

Figure 3: Several KMS mouse cursor images

Invoking programs. Another category of user interaction is invoking programs by clicking on items linked to programs. These programs range from simple KMS operations, such as those provided by the customizable command menu at the bottom of a KMS window, to large conventional programs that normally run from the operating system.

A common function of KMS-based programs is to process a hierarchy of frames. For example, the program called **Linear** takes the contents of a hierarchy of frames and paginates it into a linear document, while automatically creating a table of contents, index, etc. Another program does a text search through a hierarchy of frames. With this program, a user can first narrow a search down, by going to the top of the appropriate hierarchy, before invoking the program.

3. HYPERMEDIA DESIGN ISSUES

In this section we examine a set of issues for the design of hypermedia systems. Some of these issues have been discussed in Conklin's summary of the hypertext field [Conk87], and in papers describing specific systems, such as Intermedia [Garr87], NoteCards [Hala87], Neptune [Deli86] and TIES [Shne86]. Other issues on our list haven't received as much discussion in the literature, but they have become important for the development of ZOG and KMS. We have concentrated on issues that highlight differences between KMS and other hypermedia systems. The issues are organized into four categories:

Hypermedia Design Issues

Data Model Issues

1. What is the appropriate data model for a node?
2. What is the best size for a node?
3. What types of nodes should there be?
4. What sort of data object should be used as the source for a link?
5. What sort of data object should be used as the destination of a link?
6. What types of links should there be?
7. Should a link have internal structure?
8. How can data be aggregated into large structures?

User Interface Issues

9. What style of user interface should be used?
10. How should the information in nodes be presented on the display?
11. How should a link source/destination be represented on the display?
12. How fast should the system respond when following a link?
13. How should the system support browsing?
14. Should graphical representations of the node linkage structure be provided?
15. How can disorientation be prevented or reduced?

Authoring Issues

16. How can authoring of large databases be facilitated?
17. How can material from a database be converted to paper form?

Multiple User Issues

18. How can information be jointly authored and shared by multiple users?
19. How can interference between multiple users be prevented?
20. How can access to sensitive data be restricted?

Data Model Issues

[1] What is the appropriate data model for a node?

You will recall that KMS uses a screen-sized, two-dimensional space for a node (called a frame), containing any arrangement of text, graphics and image items. This capability is flexible enough, when combined with the ability to link frames together, to allow users to represent a wide variety of knowledge structures (documents, drawings, programs, etc).

One source of the flexibility of KMS is the way it treats space. Like space in the real world, space in frames 'exists' whether or not anything occupies it. Thus a frame may be completely empty. This is distinct from the degenerate way space is represented by most text-oriented programs (e.g., word processors and mail systems). Space to the left of text is usually some mixture of space characters and tabs, while space to the right usually has no representation at all.

The spatial nature of frames is a fundamental feature of KMS that has important implications for the user:

It helps chunk items. White space provides a visual aid to perceiving the separate items on the frame. By convention, each individual text item is surrounded by white space, and therefore is easy to recognize as a separate chunk. In some hypermedia systems, a link is embedded in a larger piece of text, thus requiring some form of highlighting for the link.

It is easy to reposition items. The space in a frame provides a background on which objects can be positioned independently of one another, as is the case with graphics programs. Rearranging objects within a frame (for instance, linked text items representing a document's sections) is a moment's work.

It provides room for peripheral items. Empty space in a frame provides a handy place for peripheral items such as a note or a reviewer's comment about the contents of the frame. In KMS, creating such items is as simple as moving the cursor to an empty area in the frame and starting to type. As long as the note or comment is not unduly large, it can happily coexist with the other items in the frame. In the case of a lengthy note, the bulk of it can be placed on additional linked frames. (By convention, these items are prefaced with "@", which suppresses them from appearing in the hardcopy version of the material.)

It maps directly onto white space on paper. We know that white space on paper is a good thing, so why not in hypermedia? This is especially useful when we want to print out hypermedia-based material. If the desired white space for the printed version cannot be directly represented, how will it be supplied? (See also Issue [17].)

It provides a convenient command context. Since space in KMS is an actual construct, it provides another context for interaction. KMS exploits this context by using it as a means for creating new objects. Thus, when the cursor is in empty space, the user can directly create new points, lines, rectangles and text items. The navigation command "Back" is also available when the cursor is in empty space. In many systems, input in empty space is an error condition!

[2] What is the best size for a node?

KMS fixes the size of a frame to a width of 1140 pixels and a height of 820 pixels. These limits allow a whole frame to be displayed on most large screen displays, with some room left for window boundaries and a small message window.

The main reason we limit the size of a frame is to avoid scrolling, which we feel is an inefficient way to navigate in a database.

[3] What types of nodes should there be?

KMS has only one type of node--the frame. We have not found it necessary to have more than one type of node because of the generality of frames. Frames can contain any arrangement of text, graphics, and images. This generality plus the ability to link frames together (especially into hierarchies) makes it straightforward to represent a broad range of knowledge structures such as documents, programs, drawings, and conversations.

Implicitly, users can define frame types by placing distinctive data items in the frames and by developing distinctive frame formats. However, since KMS won't enforce these informal frame types, any processing of these frames that relies on this 'typing' is subject to error. We think it is a good tradeoff to accept the possibility of such errors, in return for the simplicity of a single system-supported node type.

[4] What sort of data object should be used as the source for a link?

The source for a KMS link is an individual text item in a frame. The text of the item describes what it's linked to. Although the text can range from a single character to a whole paragraph of text, it is most common to use a single line of text.

Links are not embedded within text as in traditional hypertext. Experimental work with TIES purports to show the superiority of embedded links over separate links [Shne87]. However, these results probably do not apply to ZOG and KMS, where the links are always visible on the same screen as the text, due to the small grain size of the node.

Using individual items as link sources decouples the contents of a frame from the links to other frames. In systems where links are embedded in text, the phrases in the text must fit into the context of the material as well as serve as links to other nodes. Also, if the material is to be transformed into a linear document, the linked phrases must appear in the order required in the document. These constraints make it more difficult to author the material. In KMS, the links can be treated separately, and can be given whatever text seems appropriate for them.

[5] What sort of data object should be used as the destination of a link?

The destination for a KMS link is a whole frame. Some hypermedia systems use an individual point within a node, or a 'region' within a node. We have never felt the need for such a capability within KMS.

[6] What types of links should there be?

KMS users do not think in terms of links per se, but rather in terms of *linked items* --that is, items that are linked to other frames. There are two types of linked items. *Tree items* have the connotation of being linked to lower-level frames in a hierarchy, such as a chapter of a book, or a procedure within a program. *Special items* are linked to peripheral material, such as comments and cross-references. These items are simply prefaced with the "@" character, which makes it easy to change the type of a linked item. The "@" is used by KMS utility programs to distinguish between the two types of links, especially for the common case of processing a hierarchy of frames.

[7] Should a link have internal structure?

In some systems, links are objects with internal structure which provides more information about the destination of the link. In KMS a link is not an object, but rather a property of a text item. Thus

links do not have any internal structure, other than the frame name representing the destination of the link. We have found that the text of the linked item can provide enough information about the destination of the link. This avoids the need for mechanisms to view and edit the internal structure of links. In addition, we feel the rapid response of KMS makes it just as practical to follow the link as it would be to see a 'preview' of the destination.

[8] How can data be aggregated into larger structures?

In KMS, aggregates can be built from regular frames. The primary way of aggregating data is to create hierarchies of frames by linking them together via *tree items*. Since frames can perform the indexing role normally provided by directories as well as the content-holding function of files, KMS users need not employ operating system directories as a mean of organizing their work. Users find this approach very natural. Many KMS utility programs are designed to work on hierarchies as input and create hierarchies as output.

User interface issues

User interface issues have always been a major focus of our work. In fact, we usually referred to ZOG as a "human-computer interface system." The ZOG Group created a User Studies Laboratory and conducted detailed studies of ZOG users. Some of this work is reported in [Robe81a], [Robe81b], and [Yode84]. Both ZOG and KMS are instrumented to collect low-level usage data. Over the years we have collected data on nearly 400,000 user sessions.

Below we discuss several important user interface issues that apply generally to hypermedia systems:

[9] What style of user interface should be used?

Because of the potential for innovation, we believe that the user interface for a hypermedia system should be designed from scratch. Consequently, we have attempted to leave behind most of our biases about user interfaces. Instead of adopting an existing style such as multiple, overlapping windows on a desktop with pull-down menus and icons, we have tried to completely open up the design of the user interface. This has proved extremely difficult.

Thus KMS today is the result of slowly unlearning many concepts and assumptions. Mostly, this has meant learning to do without things that seemed necessary before. We are trying to provide the KMS user with an environment in which there are few conceptual distinctions. Thus we dispensed with the distinction between files and directories, use a single node type, and restrict the explicit link types to just two. Also, we eliminated the mode boundary between navigating and editing, thereby dispensing with a separate "editor." Users may make changes to a frame at any time; when they leave the frame, the changes are saved automatically.

We have chosen to develop a user interface for KMS based on the direct manipulation paradigm. We have also chosen to develop the interface around the capabilities of the three button mouse. By exploiting every contextual distinction we thought natural, we have developed an interface in which over 85% of the user's interaction requires just a single point-and-click (i.e., no intermediate menu selection). As a result, KMS users can interact more than twice as efficiently as with interfaces dominated by menu selection.

[10] How should the information in nodes be presented on the display?

There are two approaches commonly used by other hypermedia systems: (1) Each node in a separate window, with multiple overlapping windows, perhaps of different sizes; and (2) A single, linear text display, where each node that is represented is expanded "in place."

KMS's choice is distinctly different: Two nodes, each taking up a full half of the display surface, or, at the user's option, one node taking up the entire display. There are no other possibilities. When a user selects an item linked to another frame, the currently displayed frame is replaced by the new frame. Because KMS can follow a link very quickly, you can think of it as using the time dimension to keep linked nodes close together, rather than trying to keep them visible on the display at the same time.

[11] How should a link source/destination be represented on the display?

Some hypermedia systems use various forms of highlighting to represent a link source on the display, e.g., italics, boldface, color, video-reversing, or a box. Unfortunately this usurps the normal use of such highlighting by the author.

Systems that use an embedded icon of some kind are prone to clutter. By themselves, icons often do not provide enough information to enable the user to make a good decision about whether or not to follow the link. In addition, these icons are often small targets, which require more time to select.

KMS uses whole text items as link sources. A linked item is displayed with a small circle to its left indicating the existence of a link. Since the text item is normally surrounded by a sea of empty space, the range or region of the link source is defined implicitly. The content of the text item can provide as much semantic information about the link as is needed. Also the average size of linked items makes them easy to point to them.

Since KMS links are one-way, and the destination of a link is a whole frame, there is no need to denote the destination of a link.

[12] How fast should the system respond when following a link?

We believe that fast system response to selecting a link is one of the most important parameters in a hypermedia system. Even though the average time a user spends at a node will usually be many seconds, there will be frequent bursts of rapid navigation, when response time becomes critical. Our experience with a variety of hypermedia systems has shown that the difference between one system with a response of several seconds and another with sub-second response is so great as to make them seem qualitatively different. Our design goal for KMS is to be able to access and display a random frame across a wide-area network in less than .25 seconds on average.

In the early 1970's, researchers at the PROMIS laboratory produced a hypermedia system capable of 0.25 second reponse 70% of the time, using specialized hardware [Schu79]. Our early versions of ZOG, created in 1976, ran on DEC time-sharing machines with 1200 baud terminal links and provided response times of 5 to 10 seconds. When we graduated to 9600 baud around 1979, response was improved to about 2 or 3 seconds, and it seemed like a major breakthrough for users. Our PERQ version of ZOG, completed in 1983, gave an average response of about 0.7 seconds for frames local

to the machine, and 1.5 seconds for frames accessed over the Ethernet. Users again experienced a dramatic improvement over the previous version, but they quickly adapted to the new speed and still hungered for more.

We have also had experience with response speeds at the very fast end of the scale--0.05 to 0.1 seconds. In 1978, as part of the ZOG effort at CMU, we built two special ZOG terminals using a high-speed vector graphics display, a touch screen, and a fast drum, attached to one of the DEC PDP-11 processors in the experimental multiprocessor called C.mmp. We were not able to study the use of this system in any detail, because it had no editor available, it was difficult to download material from our main working environment on a PDP-10, and the hardware was unreliable. But we did satisfy ourselves that we had bounded the optimal response time from below. In fact, without some explicit cue, 0.05 second response may be too fast--we had trouble noticing whether or not the screen had changed, especially if we blinked at the wrong time!

In making the initial leap from ZOG to KMS, we took a step backward in response speed. This happened because frames tended to become larger and more complex as we took advantage of larger bit-mapped displays. Also, we began using a separate file for each frame, for added flexibility. Fortunately, KMS has benefitted greatly from the faster hardware now available, so that KMS once again has sub-second average response times.

KMS's responsiveness is mostly a function of the amount of material in the average frame (1 Kbyte), the graphics performance of the window system, and the speed of the storage device. Interestingly, frames stored remotely on a file server with a fast disk can often be accessed more quickly than frames stored locally on a slower disk.

The larger memories now available in workstations (4 Mbytes being typical) have allowed us to implement a frame caching mechanism that further speeds the response by eliminating file accesses for frames already in the cache. In Figure 4 we show typical response times for KMS running on a Sun-3/50 with 4 Mbytes of memory, using a locally-attached small disk.

	Small frame (~ 0.4 Kbytes)	**Med. frame** (~ 1.6 Kbytes)	**Large frame** (~ 3.5 Kbytes)
From disk	0.34	1.02	2.60
From cache	0.20	0.28	0.30

Figure 4: Time for KMS to access and display a frame (in seconds). The average size for KMS frames is 1 Kbyte.

[13] How should the system support browsing?

We believe that the ability to browse quickly in a hypermedia system is critical to its usability. This is particularly true of larger-scale hypermedia databases, where it's necessary to 'travel' longer distances. Although system response time is an important factor for browsing, there are other aspects as well:

Standard frame layout. The conventions for the layout of a frame make it easier for the user to assimilate the information on the frame. As a result, it takes less time to decide what to do next.

Time user takes to select. On average, linked items are large in size (compared with embedded icons) and segmented spatially. This reduces the time it takes users to point the cursor at them.

No mode boundary between editing and navigation. The user need not cross a mode boundary in order to switch between editing and navigating. Navigation and editing commands are simultaneously available.

Fast backtrack command. Backtracking is a frequent activity--for every movement forward there tends to be a compensating move back. In KMS, the Back command is available as one of the buttons of the 'empty space' cursor. The user need only move the cursor to an empty area of the frame to get into the proper context and click the Back button. On average this takes .7 seconds to do, partly because the cursor often doesn't need to be moved. This compares with 1.5 seconds to click on a menu command (such as the command items at the bottom of a KMS window). This small difference adds up since the Back command may be used several hundred times per hour. From the user's perspective, it's not just the time savings, but the reduced mental and physical effort.

[14] Should graphical representations of the node linkage structure be provided?

Periodically we consider providing additional views in KMS such as a graph of a portion of the network. But each time we retreat. We believe such views are unnecessary, except perhaps for large, essentially non-hierarchical structures. Our own experience indicates that our 'mind's eye' sees KMS structures as time-travel through familiar frames, rather than as some graphical representation of the structure. This view is supported by our ZOG user studies, which revealed that users rarely made use of the multi-node views that were available. The 'overview-like' nature of frames, plus being able to travel in the database rapidly--seems to substantially reduce the need for such structures in KMS.

[15] How can disorientation be prevented or reduced?

The classical hypermedia problem is the "getting lost problem," which becomes more severe as the database grows larger. However, we have found that getting lost is not much of a problem for KMS users. KMS has characteristics that help users stay oriented, plus some features that help users re-orient themselves if they do get lost.

Hierarchical backbone. KMS strongly encourages a top-down, stagewise refinement approach to organizing material in the database. The resulting hierarchical "backbone" in the database helps users build a coherent mental model of the database. Also, multiple hierarchies can be constructed to provide alternative paths through the database.

Special navigation commands. KMS provides several commands that let users go directly to specific locations in the database. The Goto command lets a user go directly to any named frame. The Home command displays a user's home frame. The Info command displays a frame with links to KMS documentation and utilities.

Marking the item just returned from. KMS flags the item linking to the frame from which the user has just backtracked with a temporary asterisk.

Richer frames. The use of larger frames provides a richer context in which to assimilate knowledge. Also, since there are fewer frames, less travel is required.

Fast response. The ability to navigate quickly from frame to frame makes exploration less risky for users, since they can always backtrack quickly to return to a familiar frame.

Authoring Issues

Below we discuss a couple of issues dealing with how hypermedia databases can be created. These are important because database creation is a severe bottleneck. A user's ability to assimilate information far outstrips his ability to generate it.

[16] How can authoring of large databases be facilitated?

Small databases are of limited interest. This poses an economics problem for hypermedia system designers to solve. If it's too inconvenient to build a hypermedia database users will avoid doing it. Listed below are some of the approaches we have taken to encourage the development of large-scale databases:

Rapid navigation. Users need to be able to move around rapidly in order to get to where they wish to build. This is also important in the frequent case of moving objects to a different place in the database.

No editing/navigation mode transition. KMS does not have a mode transition between navigation and editing (see Issue [9]).

Rapid creation of new frames. To create a frame, all a user has to do is click on an unlinked item. Typically, the user can be editing a new frame less than two seconds after deciding to create it.

Default operand scope. Editing in KMS is dominated by manipulating individual items. Since the default scope for operations is the whole item pointed to by the cursor, the vast majority of operations can be invoked directly, without any explicit scope designation.

Implicit saving of changes. Tentative modifications are limited to the currently displayed frames. If there are any changes to a frame those changes will be automatically saved when the user moves to another frame. This default works well in practice. Not only does it eliminate most explicit save invocations, but it reduces the complexity of the user's model of the current state of the system.

Use of schemas. Schemas are chunks of data (e.g., a frame or tree of frames) that contain variable parts. Schemas can be used to build data objects that have some common parts, simply by copying the schemas and filling in the variable parts manually. Ramakrishna [Rama81] developed schema mechanisms for ZOG and studied their use experimentally.

Tools for importing external databases. KMS provides a number of tools for mapping in material from other sources (e.g. text files and bitmap files).

Support for multiple users. KMS is a distributed hypermedia system designed to support simultaneous building of a KMS database by multiple users. (Please see "Multiple User Issues")

No restriction on the size of the database. KMS databases may be as large as available secondary memory and may be distributed across any number of workstations and file servers.

Database merging. Independently developed KMS database are easily joined together to form a single database.

[17] How can material from a database be converted to paper form?

One of the major forces guiding our design efforts has been the desire to create well-formatted documents from material in the database. Since frames provide a local WYSIWYG view, there is a natural process for paginating the material: concatenating the contents of frames from a hierarchy in depth-first order. This default can be supplemented by additional formatting commands (e.g., "@NewPage," "@Figure," etc.) that are placed on frames to specify additional formatting constructs. Typically, these items, like other meta-level items such as notes and comments, are placed off in the corner to keep them out of the reader's way. This approach is a hybrid between pure WYSIWYG document systems (in which little structure is represented explicitly) and structured formatting systems (Scribe and TEX). KMS also offers the flexibility of applying the document formatting process at any level of a hierarchy of frames, thereby enabling users to get just the portion of a document they want.

Multiple User Issues

Both ZOG and KMS have been designed from the beginning to support a community of communicating users, where users can jointly develop and share data, rather than simply exchange it. Below we discuss several of the relevant issues.

[18] How can information be jointly authored and shared by multiple users?

KMS provides to a community of users a single, logical database, physically distributed across multiple workstations and file servers on a network. The actual physical location of data can be completely transparent to the users--as if they were all users on a single time-sharing system, but with vastly improved response and display bandwidth.

Our first real implementation of a distributed system was the version of ZOG running on the PERQ network on board the USS CARL VINSON, which was completed in 1983. It implemented special ZOG network servers that managed the locking of individual ZOG frames for modification by one user at a time. It had a location database, managed by one of the machines designated as the *master*, with information about which machine each ZOG frameset was actually located on.

Our current version of KMS uses Sun's Network File System (NFS) to provide access to frames that reside on remote machines. As with ZOG, there is a *master* file server that holds the location of all frameset. (All file servers containing a portion of the KMS database have automatically-maintained copies of this location information, to be used in case the master is unavailable).

One of the major benefits of KMS is the ability of multiple users to work simultaneously on a common project such as proposal or conference paper. Working together on a single document (or single area of the database), users can easily see what others have done, make comments, print out any part of the material at any time, etc. There is no strict coordination concerning the evolution of the database similar to that required in conventional database systems.

This communal approach makes it possible to communicate electronically in a way quite different from conventional electronic mail. In KMS, conversations simply 'grow' in some area of the database, thus preserving their logical structure.

[19] How can interference between multiple users be prevented?

How can we prevent multiple users from losing changes due to interference, yet avoid the inefficiencies of locking users out from making changes for long periods of time? In ZOG, we provided for the locking and unlocking of a frame when the user entered and exited the editor, respectively. In KMS, we do not lock frames in this way. Instead, we use a weaker form of concurrency control, called 'optimistic concurrency control.' We make the optimistic assumption that since frames greatly outnumber users, a conflict between users editing the same frame is rare.

All 'optimistic concurrency' guarantees is that a KMS user who has successfully saved changes to a frame cannot subsequently have those changes revoked by another user who had been editing the same version of the frame. It does not guarantee that if you edit a frame, you will necessarily be able to save the changes without any problem. At the time you attempt to save your changes, you may be informed that someone else has already saved changes to the same frame. This means that your tentative changes cannot be saved, because they would revoke the other user's changes. What KMS does in this case is to temporarily save your changes in a newly created frame, so that you can then map them into the new version of the original frame.

Our experience shows this situation rarely occurs since users are normally working in different areas of the database, even when they are working on the same document (for example, this paper is represented by over 200 frames). Whenever users find they are 'bumping elbows,' they can cope by using informal frame 'locking' conventions--namely, by placing a text item on a frame that warns the other users who come to that frame that editing is in progress. Besides being more personal, this informal locking can be used to alert others you plan to do some work in this area of the database.

On the face of it, optimistic concurrency control may seem unwise, since it is not a foolproof mechanism for preventing interference between users. But adopting it allowed us some benefits that we feel well outweigh its drawbacks. The most important benefit was that it facilitated eliminating the mode boundary between navigating and editing (see Issue [9]).

[20] How can access to sensitive data be restricted?

To prevent access to sensitive data, KMS implements protection of individual frames. Every frame has an owner--originally, the person who created it. The owner can protect the frame so that others may access it but not make any modifications, or so that others can't even access the frame.

An intermediate kind of protection (called *annotation access* by the Intermedia researchers [Garr87]) seems like a useful capability to add to KMS. It would allow users to add new items to a frame without allowing them to modify any of the existing items. For now, we generally leave frames unprotected to allow free annotation, and simply rely on the good will of users to not delete the work of other users without permission.

4. CONCLUSION

If there is one central theme to our experience, it is the fundamental importance of a system's data model. Our experience with ZOG and KMS has convinced us that the data model underlying an interactive system strongly determines the nature of its user interface. We believe this because we have seen the formative influence of the KMS data model on all other aspects of KMS.

In the case of KMS, the properties of a node--its fixed size, its spatial nature, how links are represented within it, its standard format, etc.--contribute significantly to the global nature of the system and distinguish it strongly from other hypermedia systems.

Consequently we recommend that interactive systems be developed from the inside out--from the data model to the user interface--rather than the other way around. This view contrasts sharply with the philosophy that the user interface should be the dominant system component and thus standardized across programs. Perhaps hypermedia, with the structural richness it has to offer human-computer interaction, may eventually overshadow the reigning HCI paradigm, the desktop interface.

ACKNOWLEDGMENTS

We wish to acknowledge the contributions of many people over the years. Those who were involved with ZOG at CMU: Allen Newell, George Robertson, Kamila Robertson, Peter Lieu, Sandy Esch, Patty Nazarek, Marilyn Mantei, Kamesh Ramakrishna, Roy Taylor, Mark Fox and Andy Palay. Those officers from the USS CARL VINSON who worked with us at CMU: Mark Frost, Paul Fischbeck, Hal Powell, Russ Shoop, and Rich Anderson. Captain Richard Martin, Cdr. Ted Kral, Lt. Brian MacKay, and other officers and crew of the USS CARL VINSON. Finally, we would like to thank the Office of Naval Research for sponsoring the 10 years of the ZOG Project.

REFERENCES

[Aksc84a] Akscyn, R. and D. McCracken, "The ZOG Approach to Database Management," *Proceedings of the Trends and Applications Conference: Making Database Work*, Gaithersburg, Maryland, May 1984.

[Aksc84b] Akscyn, R. and D. McCracken, "ZOG and the USS CARL VINSON: Lessons in System Development," *Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84)*, London, U.K., September 1984.

[Conk87] Conklin, J., "A Survey of Hypertext," MCC Technical Report STP-356-86, Rev. 1, February 1987. To appear in *IEEE Computer*, September 1987.

[Deli86] Delisle, N. and M. Schwartz, "Neptune: A Hypertext System for CAD Applications," *Proceedings of ACM SIGMOD International Conference on Management of Data*, Washington, D.C., May 1986, pp. 132-143.

[Garr87] Garrett, L., K. Smith and N. Meyrowitz, "Intermedia: Issues, Strategies, and Tactics in the Design of a Hypermedia Document System," *Proceedings of the Conference on Computer-Supported Cooperative Work*, Austin, Texas, December 1986, pp. 163-174.

[Hala87] Halasz, F., T. Moran and R. Trigg, "NoteCards in a Nutshell," *Proceedings of the ACM Conference on Human Factors in Computing Systems*, Toronto, Canada, April 1987.

[Mant82] Mantei, M., *A study of Disorientation Behavior in ZOG*, PhD thesis, University of Southern California, 1982.

[McCr84] McCracken, D. and R. Akscyn, "Experience with the ZOG Human-Computer Interface System," *International Journal of Man-Machine Studies*, Vol. 21, 1984, pp. 293-310.

[Newe85] Newell, A., "An On-Going Case Study in Technological Innovation," in *Advances in Information Processing in Organizations*, Sproull, L. and P. Larkey (eds.), 1985.

[Newe81] Newell, A., D. McCracken, G. Robertson and R. Akscyn, "ZOG and the USS CARL VINSON," *Computer Science Research Review*, Carnegie-Mellon University, 1981, pp. 95-118.

[Rama81] Ramakrishna, K., *Schematization as an Aid to Organizing ZOG Information Nets*, PhD thesis, Computer Science Department, Carnegie-Mellon University, 1981.

[Robe82] Robertson, C.K. and R. Akscyn, "Experimental Evaluation of Tools for Teaching the ZOG Frame Editor," *Proceedings of the International Conference on Man/Machine Systems*, Manchester, U.K., July 1982.

[Robe81a] Robertson, C.K., D. McCracken and A. Newell, "Experimental Evaluation of the ZOG Frame Editor," *Proceedings of the 7th Canadian Man-Computer Communications Conference*, Waterloo, Ontario, June 1981, pp. 115-123.

[Robe81b] Robertson, G., D. McCracken and A. Newell, "The ZOG Approach to Man-Machine Communication," *International Journal of Man-Machine Studies*, 1981.

[Schu79] Schultz, J. and L. Davis, "The technology of PROMIS," *Proceedings of the IEEE*, September 1979, pp. 1237-1244.

[Shne86] Shneiderman, B. and J. Morariu, "The Interactive Encyclopedia System (TIES)," Department of Computer Science, University of Maryland, College Park, MD, June 1986.

[Shne87] Shneiderman, B., "User Interface Design and Evaluation for an Electronic Encyclopedia," Technical Report CS-TR-1819, Department of Computer Science, University of Maryland, March 1987.

[Yode84] Yoder, E., McCracken, D., and R. Akscyn, "Instrumenting a Human-Computer Interface for Development and Evaluation," *Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84)*, London, U.K., September 1984.