KMS: A DISTRIBUTED HYPERMEDIA SYSTEM FOR MANAGING KNOWLEDGE IN ORGANIZATIONS

Developers of hypermedia systems face many design issues. The design for KMS, a large-scale hypermedia system for collaborative work, seeks improved user productivity through simplicity of the conceptual data model.

ROBERT M. AKSCYN, DONALD L. McCRACKEN, *and* **ELISE A. YODER**

For the past seven years we have been developing KMS (Knowledge Management System), a distributed hypermedia system for workstations, based on our previous research with the ZOG system at Carnegie-Mellon University. KMS supports organization-wide collaboration for a broad range of applications, such as electronic publishing, on-line documentation, project management, software engineering and computer-aided instruction.

The heart of KMS is its conceptual data model. A KMS database consists of screen-sized WYSIWYG workspaces called frames which contain text, graphics and image items. Individual items can be linked to other frames or 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.

The KMS user interface employs a form of direct manipulation designed to exploit a three-button mouse. A combined browser/editor is used to traverse the database and manipulate its contents. Over 90 percent of the user's command 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 half a second, on average.

Definition of Hypermedia

Hypermedia is a generalization of the hypertext concept, now over 40 years old. Although there is no generally accepted definition for hypermedia, most hypermedia and hypertext systems can be characterized by the following features:

- Information is "chunked" into small units, variously called *notecards*, *frames*, *nodes*, etc. Units may contain textual information. In *hypermedia* systems, units may also contain other forms of information such as vector graphics, bitmapped images, sound and animation.
- Units of information are displayed one per window. (Systems vary in the number, size and arrangement of windows they permit.)
- Units of information are interconnected by *links*. Users *navigate* in a hypermedia database by selecting links in order to travel from unit to unit.
- By creating, editing, and linking units, users build information structures for various purposes (e.g., authoring documents, developing on-line help systems).
- In *shared* hypermedia systems, multiple users may simultaneously access the hypermedia database. Shared systems may be implemented as *distributed* systems, in which portions of the database are distributed across multiple workstations and file servers on a network.

For a complete historical perspective of hypertext, readers are referred to [6, 8–14, 16, 17, 21].

History of ZOG and KMS

We have been developing hypermedia systems for over a decade, first at Carnegie–Mellon University (CMU) with the ZOG Project, and now at Knowledge Systems with the development of KMS. While developing ZOG and KMS, we have used them extensively for our work—personally 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.

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 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 [26]. Struck by the qualitative difference that rapid response brings to menu-selection interfaces, Newell and Robertson began a research project sponsored by the Office of Naval Research (ONR) 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, VAXs, and even for an experimental multi-processor machine, C.mmp [25].

By 1980 we felt ZOG was sufficiently mature to be tested in the real world. So we embarked on a major 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 Captain Dick Martin and other officers of the *Carl Vinson*. The joint group used ZOG as a collaborative environment for project management, document production, and software engineering. 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. This version of ZOG supported four applications:

- On-line policy manual (Ship's Organization and Regulation Manual)
- Interactive task management system (for analyzing and tracking complex tasks)
- On-line maintenance manual with interface to videodisk (for weapons elevators)
- Interface to the AirPlan expert system (Airplan 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 [3], [19] and [20].

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 provide operators of nuclear power plants with rapid 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.

Over the years we have found ZOG and KMS to be useful in a surprising number of applications. At Knowledge Systems we use KMS for almost every aspect of our work, including administration, product support, document production, product design and software engineering. Below, we list the applications we have explored. More information about these applications can be found in [2], [16], and [20].

1. Electronic publishing
2. On-line manuals
3. Electronic mail and bulletin boards
4. On-line help for other software
5. Project management
6. Issue analysis
7. Financial modeling and accounting
8. User interface to videodisk-based materials
9. User interface to other programs (e.g., expert systems)
10. Software engineering
11. Computer-assisted foreign language translation
12. Operating system shell

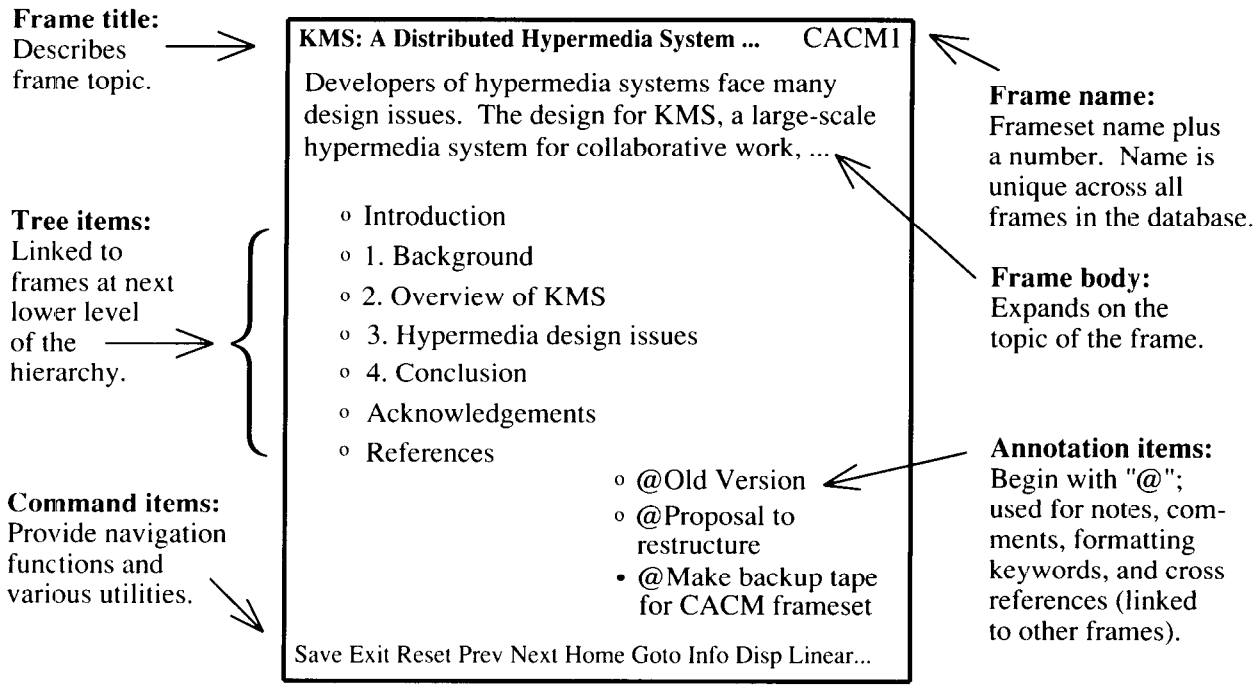
OVERVIEW OF KMS

KMS is designed to help organizations manage their knowledge. We are concerned not only with the productivity of the individual, but also the productivity of groups—from small work groups up to an entire organization. Consequently, we have been 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 [1]. In particular, we are trying to reduce the effort required to build and maintain corporate databases, since these activities are often the principal bottlenecks in many uses of computers.

KMS Data Model

A KMS database consists of a set of interlinked, screen-sized workspaces called *frames*. There is only one type of frame in KMS. Frames may contain any combination of text, graphics and image items, each of which may be linked to another frame or used to invoke a program.

Although frames may contain any arrangement of items, strong conventions have evolved for the format of frames, and these conventions have been remarkably stable for over a decade. Figure 1 shows a typical frame that illustrates these conventions.



This is the "top" frame of the frame hierarchy that represents this article. A frame may contain any combination of text, graphics, and image items. Each individual item can be linked to any other frame (indicated by a small hollow circle) or to a program (small solid circle).

FIGURE 1. Typical KMS Frame

Users typically represent a small "knowledge artifact," such as a one-page letter, in a single frame. However, a larger artifact, such as a technical manual or software module, is usually represented as a hierarchy of frames. Figure 2 shows a small fragment of a KMS-based document—in this case, the top levels of the frame hierarchy that represent an earlier draft of this article.

A KMS database may have as many frames as disk space permits, distributed across any number of servers. Figure 3 shows an abbreviated example database distributed across a network—here a combination of workstations with attached disks (file servers) and workstations without disks.

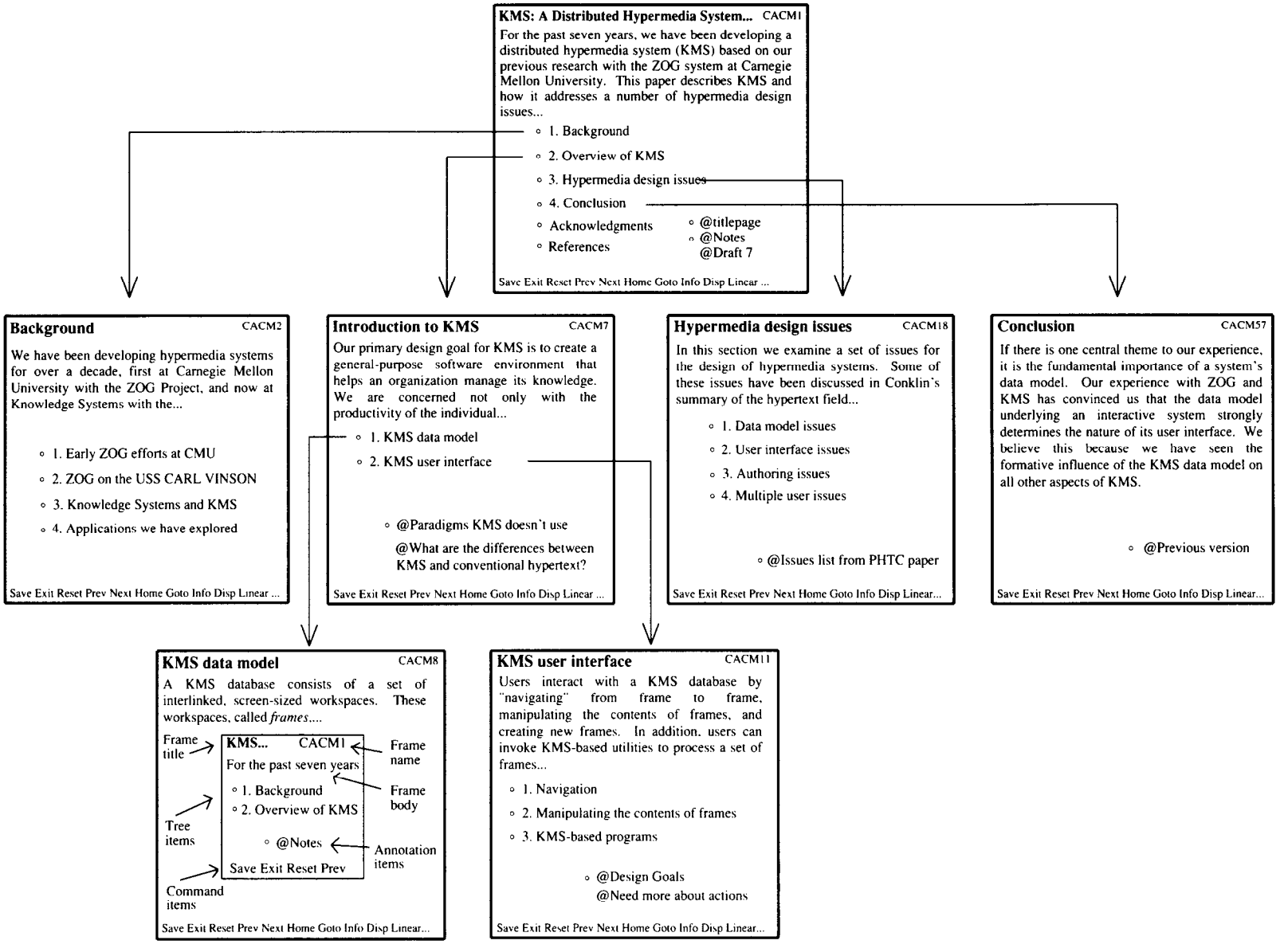
Figure 3 also illustrates the strong hierarchical orientation of most KMS databases (although KMS databases may have any structure that their creators desire). There is a multi-level hierarchy that acts as a "skeleton" for the entire database. The top levels of this skeleton serve primarily as an index to the entire database. The skeleton reaches down, at lower levels, to smaller hierarchies of frames that represent documents, project plans, and other task-related groups of information. Users freely supplement the hierarchies with links to cross-references, comments, old versions, and shared information (such as boilerplate language).

KMS User Interface

The KMS screen. The workstation screen is normally split into two windows, each of which shows the left half of a full-screen frame (see Figure 4). This size is sufficient for one page worth of material. Full screen frames are used mainly for complex diagrams, such as Figures 2, 3, 4, and later in 6.

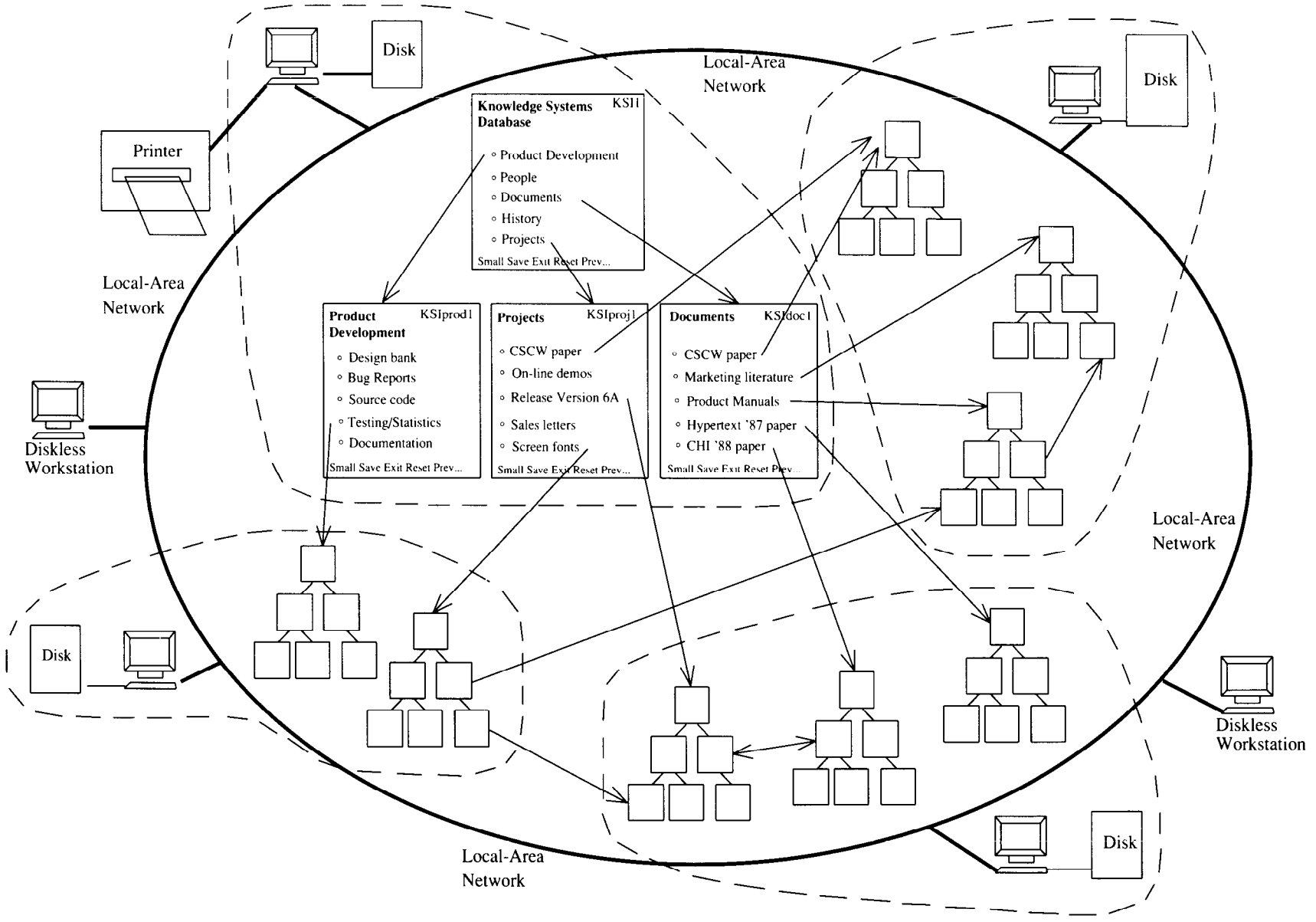
Navigating. The central KMS metaphor is that the database is a universe of connected spaces through which users rapidly travel, like pilots navigating spacecraft in the real universe. Users navigate from frame to frame by pointing the mouse cursor at an item linked to another frame and clicking the left mouse button. KMS accesses the linked frame and displays it within the same window—in less than half a second, on average.

Editing frames. There is no mode boundary between navigation and editing operations. 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. Creating a new frame is also easily done, by moving the mouse cursor to an unlinked item and clicking the left button to create a new frame linked



This diagram shows the top levels of the frame hierarchy representing a version of this article. For clarity, we've omitted many of the cross-reference links, author's comments, links to old versions, and the other sorts of peripheral information that accompany any document under development. All of Knowledge Systems documents (letters, memos, product literature and documentation, papers, reports, etc) are represented in this form.

FIGURE 2. Fragment of a KMS Database



Each little hierarchy of frames represents a task-related group of frames, such as a document, group of sales letters, issue analysis, etc. The dotted lines indicate which **framesets** are stored on which disks.

FIGURE 3. Hypothetical KMS Database Distributed Across a Network

[KMSMsg149] Saved changes to Conference30.
[KMSMsg2] Please enter name of frame to go to:

Registration packets Conference24

By May 1 we'll need to send out the advance registration packets for exhibitors. Here's a checklist of items for the packets:

- **Cover letter** from conference organizers
- **Promo piece**--"Logical Software '87"
  - Photographs from last year
  - Data sheet: last year's attendance, number of orders placed, breakdown of job categories
  - Quote from president's address
- **Booth registration form**
- **Diagram of exhibit hall**
- **Price list** for booth space and rental equipment

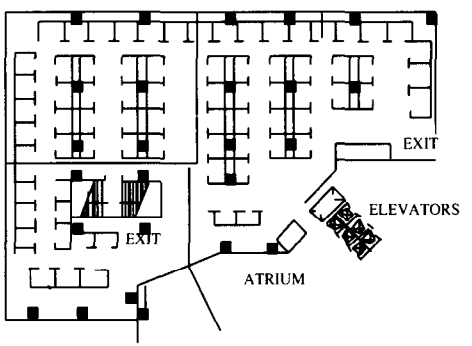
◦ @Information about Exhibit Hall

Save Exit Reset Prev Next Home Goto Info Disp Linear Print Fmt UnDel Scale Extr

Exhibit Hall Conference30

The hotel has provided us with a diagram of the exhibit space. We will need to number the booths and send the diagram to the exhibitors so they can sign up for specific spaces.

The diagram is shown below:



◦ 1. Individual booth sizes
◦ 2. Availability of electrical power
◦ 3. Storage spaces
◦ @Problems at previous Exhibit Hall

Save Exit Reset Prev Next Home Goto Info Disp Linear Print Fmt UnDel Scale Extr

The item in the shaded area on "Registration Packets" frame (in left window) is linked to "Exhibit Hall" frame (in right window).

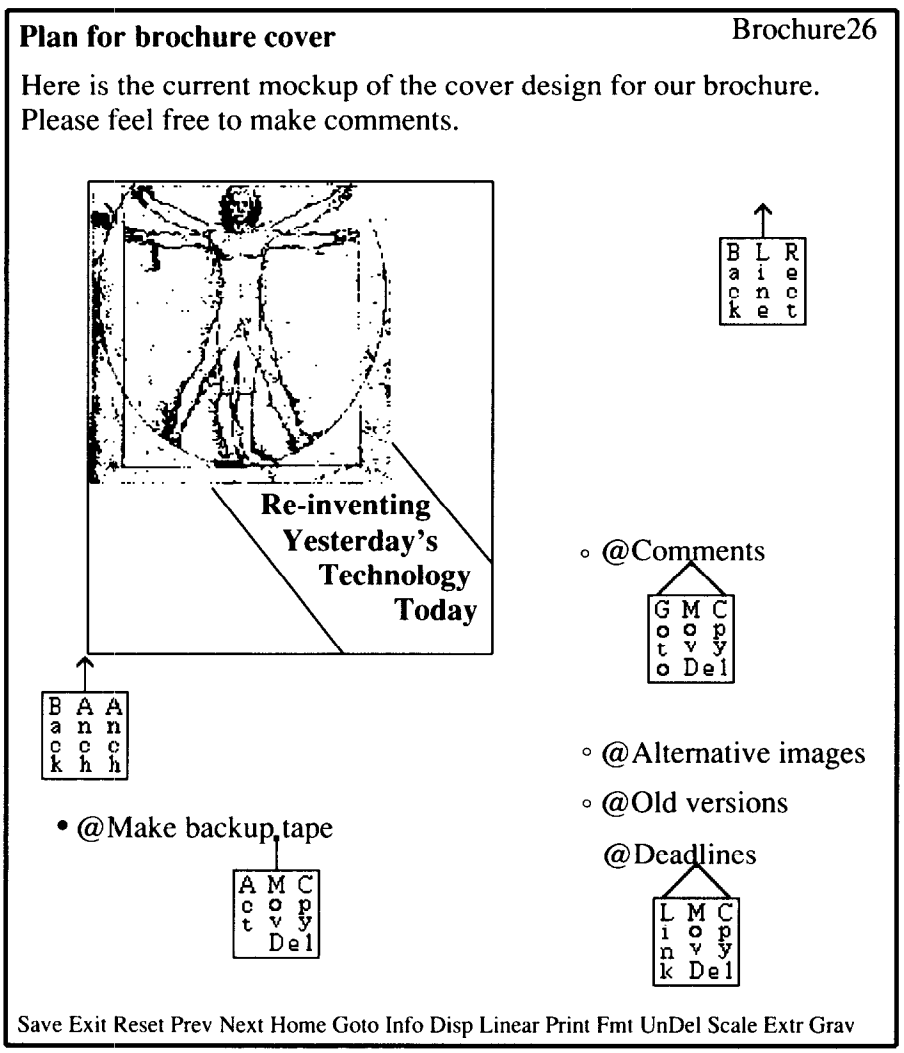
FIGURE 4. Two Example Frames on Workstation Screen

from that item. (This is the same button used to navigate existing links, so creation feels like a special form of navigation to the user.) When users navigate away from a frame they have edited, their changes are automatically saved.

Invoking programs. Another category of user interaction is invoking programs by clicking on items that have attached programs. These programs can range from simple KMS operations, such as those provided by the command menu at the bottom of a KMS window, to large, conventional programs that are normally run from the operating system shell. A common use of KMS-based programs is to process a hierarchy of frames using any frame as a starting point. For instance, documents are not explicit objects in KMS; they can be created from any point in a hierarchy of frames. This enables users to print out just the section of the document that they want.

Context-sensitive cursor. Users can invoke most KMS operations with a single point-and-click. The context of the cursor (for instance, whether it is in empty space or inside a text item) determines which operations are currently available via the mouse buttons. As an aid to users, the cursor images include vertical text labels indicating what each mouse button does. KMS novices rely on these cursor labels to learn the system. Experience has shown that KMS experts continue to rely on the labels, but in a subliminal way. Figure 5 illustrates several KMS cursors.

Unified command set. KMS can be viewed as a *reduced instruction set interface*, somewhat akin to a reduced instruction set computer (RISC) in that we seek "performance through simplicity." The Move command is an example of how we have tried to streamline the KMS user interface by unifying related operations. For instance, pointing the cursor at an item causes the "Move, Copy, Delete" cursor to appear. Clicking the Move button attaches the item to the cursor. The user can then drag the item around—not only within the current frame, but also across the window boundary into the other frame. Before anchoring the item, the user can



The KMS mouse cursor changes dynamically to indicate the operations currently available on each of the 3 mouse buttons. Here we show the principal KMS cursors sprinkled on a frame to illustrate the contexts in which they appear. The operations performed by the buttons for the cursors shown above are:

- Act Invoke **action** (KMS program)
- Anch **Anchor** the item being dragged by the mouse
- Back Go **back** to the previous frame (dragging attached items, if any)
- Cpy **Copy** the item
- Del **Delete** the item (middle and right button combination)
- Goto **Go to** the frame to which this item is linked
- Mov Latch onto the item for **moving** (dragging)
- Line Create first endpoint of a **line** (starts rubberbanding)
- Link Create a **link** to a new frame or existing frame
- Rect Create first corner of a **rectangle** (starts rubberbanding)

FIGURE 5. Frame Showing KMS Cursors in Different Contexts

type text, move to other frames (dragging the item along), and even create new frames. This eliminates the need for a KMS "clipboard," with separate operations for "cutting" and "pasting." The single KMS Move operation performs the equivalents of the following operations in other computing environments:

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

HYPERMEDIA DESIGN ISSUES

Here 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 [8], and in papers describing specific systems, such as Intermedia [11, 29], NoteCards [14], Neptune [9] and HyperTIES [27, 28]. Other issues on our list haven't received as much discussion in the literature, but they have been important in the development of ZOG and KMS. We emphasize issues that highlight differences between KMS and other hypermedia systems. We have organized the issues into four categories: data model, user interface, collaboration and miscellaneous.

DATA MODEL ISSUES

What is the appropriate data model for a node?

The hypermedia node in KMS is the *frame*, a screen-sized workspace upon which the user can place text, graphics and image items. Each of these items can be linked to another frame or used to invoke a program. Thus, a *link* in KMS is a component of a node, similar to links in NoteCards and HyperCard, but different from systems that represent links and nodes separately (e.g., InterMedia's *webs*).

The most important characteristic of a KMS frame is its spatial nature. Like space in the real world, space in a frame "exists" whether or not any objects occupy it. Thus a frame may be completely empty. This is different 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 KMS frames has the following important implications for the user:

Makes it easy to recognize items. By convention, each individual text item is surrounded by white space, and therefore is easy to recognize as a separate chunk. Since the individual item is the default scope for cursor-based operations, users can visually identify the default scope of the operation.

Makes it 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.

Provides a natural command context. Since space in KMS is part of the data model, it provides a natural context for interaction. KMS uses this context for creating objects. When the cursor is in empty space, the user can directly create points, lines, rectangles and text items. The navigation command Back is also available. In many systems, user input in empty space is considered an error condition!

Provides room for annotation. Empty space in a frame provides a handy place for peripheral items such as an author's 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. If a note is lengthy, the bulk of it can be placed on additional linked frames. By convention, comments are represented as annotation items, which makes their meta-level nature easily identifiable.

What size should a node be?

KMS fixes the size of a frame to a width of 1132 pixels and a height of 805 pixels. This choice may seem arbitrary, but it allows 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 reduce reliance on scrolling, which is an inefficient way to navigate in a database. Most large knowledge artifacts can easily be represented as hierarchies of frame-sized units, and can then be accessed via navigation rather than scrolling.

What types of nodes should there be?

Most hypermedia systems have a variety of node types (e.g., NoteCards, InterMedia). KMS has only one frame type, similar in that respect to HyperCard. Variety is provided at the level of individual items within a frame—the text, graphics, and images. The generality of frames, plus the ability to link them together (especially into hierarchies), enables users to represent a broad range of knowledge artifacts such as documents, programs, drawings, conversations and indexes.

Using a single frame type has the following implications:

Reduces the number of concepts. Frames unify into a single construct what are often distinct levels in traditional computing environments. For example, the functions of a "desktop," directories, files, clipboards, menu bars and pull-down/pop-up menus are all provided by frames. This unification simplifies the user's conceptual model considerably, and eliminates the need for many commands (e.g., commands for manipulating directories).

Reduces the number of command contexts. Having a single frame type encouraged us to develop a single editor that could operate on all types of objects in a frame. Furthermore, we merged navigation with editing, providing a single major command context in which all commonly used commands are directly available to the user. In fact, we no longer describe KMS as having an editor. Having one frame type and one major command context greatly simplifies the user's model of the system.

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; links are not embedded within text as in conventional hypertext systems. The text of the item describes what it is linked to. Although the text can range from a single character to an entire frame's worth of text, it is most common to use a single line of text for a linked item (see Figure 1).

The use of whole text items as the source for links has the following implications:

Avoids highlighting for link source. The extent of each link source is readily apparent, since the corresponding text item is surrounded by white space. No text highlighting is necessary to denote the link source. KMS needs only to denote whether or not an item is a link source. (KMS does this by displaying a small circle icon to the left of each linked item.)

Provides natural default operand scope. Representing text as separate items provides a natural default operand scope for common operations such as moving, copying and deleting links. Over 90 percent of the commands invoked in KMS (by frequency) are invoked on the default scope, i.e., a whole item.

Decouples main text and links. In systems where links are embedded in text, the phrases in the text must fit into the context of the prose as well as serve as links to other nodes. Also, if these links are followed to transform the text 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.

What sort of data object should be used as the destination for a link?

The destination for a KMS link is a whole frame, similar to NoteCards and HyperCard. Some hypermedia systems, such as InterMedia, use an individual point or region within a node. We have never felt the need for such a capability within KMS, probably because a frame is a small enough logical unit that the whole frame can sensibly serve as the link destination. In a system where nodes are entire documents, the need for a finer discrimination of link destinations is obvious.

What types of links should there be?

Most hypermedia systems provide for different link types. In some systems there is a predefined set of link types; in other systems they may be defined by users. The typing information may be visually denoted or hidden. The major purpose of link types is to provide users and programs with more information about the destination of the link (e.g., to indicate it is a "counter-argument").

Since the source for links in KMS is a whole text item (i.e., a link is a property of an item), users think in terms of linked items. In KMS there are two types of linked items: *tree items* and linked *annotation 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. Linked annotation items point to peripheral material, such as comments and cross-references. Annotation items are denoted by having "@" as their first character. This makes it convenient for users to change the type of a linked item.

Having these two types of links distinguishes between structural relationships (e.g., chapters of a book) and purely associative relationships. This distinction helps users stay oriented while they are navigating in a large hypermedia database.

Should a link have internal structure?

In some systems, links are objects with internal structure that 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. 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 usually provides enough information about the destination of the link. This reduces the need for mechanisms to view and edit the internal structure of links. 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.

How can nodes be aggregated into larger structures?

In KMS, aggregate structures are built by linking frames together. The primary way of aggregating data is to create hierarchies of frames by linking them together via *tree items*. This is how KMS represents the hierarchical structure of artifacts such as documents and programs. In addition, tree items enable frames to serve the organizing role normally provided by directories, as well as the content-holding function of files. Many KMS programs process hierarchies of frames as input and create hierarchies of frames as output.

How can versioning be supported?

Knowledge artifacts with long life cycles (e.g., aircraft maintenance manuals, system software) require support for multiple versions. KMS supports versioning for frame hierarchies, which are the natural KMS structures for representing an object such as a document or

software module. A KMS utility program can "freeze" all the frames in a hierarchy as belonging to a particular version. Whenever a frozen frame is subsequently modified, a copy of its frozen contents is automatically saved in a new frame, which is placed at the head of a linked list of frames representing earlier versions. Users can follow these links to find a specific version of a frame, but the links are chiefly intended for programs to reconstruct earlier versions.

USER INTERFACE ISSUES

User interface issues have always been a major focus of our work. In fact, we initially 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 [15], [23], [24], and [30]. Both ZOG and KMS are instrumented to collect low-level usage data. Over the years we have collected data on over 400,000 user sessions.

Here we discuss several important user interface issues that apply to hypermedia systems.

What style of user interface should be used?

Because of the potential for innovation, we believe 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.

Hence, KMS today is the result of rethinking many user interface design issues. Mostly, this has meant learning to do without things that previously seemed so necessary to us. We are trying to provide the KMS user with an environment in which there are few concepts to learn. For instance, we dispense with the distinction between files and directories, use a single node type, and restrict the explicit link types to two. We also eliminate the mode boundary between navigating and editing and dispense with the notion of an editor.

The KMS user interface is based on the direct manipulation paradigm and the three-button mouse. By exploiting natural contextual distinctions, we have developed an interface in which over 90 percent of the user's interaction requires just a single point-and-click (i.e., no intermediate menu selection). This reduces the average time per operation to less than half what it is with typical menu-selection interfaces.

How should 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; (2) A single linear display, where each node is expanded in place.

KMS's choice is distinctly different: Two nodes, each taking up a full half of the screen, or, at the user's option, one node taking up the entire screen. 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, we think of it as using the time dimension to keep linked nodes close together (*time multiplexing*), rather than trying to keep them visible on the display at the same time (*space multiplexing*)—these terms are due to Card [7].

How should a link source/destination be presented on the display?

Some hypermedia systems use text highlighting to present a link source on the screen (italics, boldface, color, video-reversing, underlining, boxing, etc.). Unfortunately this usurps the normal use of highlighting by authors.

Another approach is to use embedded icons. By themselves, icons often do not provide enough information for the user to make a good decision about whether or not to follow the link. To reduce the degree of clutter, these icons are often small, thus requiring more time for the user to select with the mouse.

KMS uses whole text items as link sources. A linked item is denoted by a small circle to its left. Since a text item is normally surrounded by white space, the range of the link source is defined implicitly. Most linked items are at least one line of text, making them easy to select. Furthermore, text items can be as large as a full screen, thus allowing the author to provide as much semantic information about the link as is necessary.

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 as in NoteCards and HyperCard.

How fast should the system respond when following a link?

We believe that fast system response to selecting a link is the most important parameter of a hypermedia system. Although 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 shows 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 0.25 seconds on average. (Newell has argued that response faster than 0.25 seconds will not benefit users [18]).

In the early 1970s, researchers at the PROMIS laboratory produced a hypermedia system capable of 0.25 second response 70 percent of the time, using specialized hardware [26]. 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 to 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 became 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 and file systems now available, so that KMS once again has sub-second average response times.

KMS's responsiveness is mostly a function of the average frame size (1 Kbyte), the graphics performance of the window system, and the speed of secondary storage devices. Ironically, 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 (typically 4 Mbytes) have allowed us to implement a frame caching mechanism that further speeds the response by eliminating file accesses and generation of text display images for frames already in the cache. In Table I we show typical response times for KMS running on a Sun workstation, accessing frames from a remote machine.

How should the system support browsing?

Browsing is a canonical activity for hypermedia users. We believe the ability to browse quickly in a hypermedia system is critical to its usability. This is especially true for large-scale hypermedia databases, where it is necessary to navigate through more nodes. Although system response time is the most important, other factors are significant as well. The following factors are relevant to KMS users.

TABLE I. Time (in seconds) for KMS to access and display a frame using a Sun 386i model 250 Workstation. The average size for KMS frames is 1 Kbyte.

	Small frame (0.5 Kbytes)	Med. frame (1.5 Kbytes)	Large frame (4.5 Kbytes)
From remote disk	0.22	0.48	1.46
From cache	0.12	0.18	0.22

Standard frame layout. The relative homogeneity of KMS frames (fixed size, layout conventions, etc.) gives frames a visual regularity that makes it easier for users to perceive the components of the frame, interpret them, and make a decision about what to do next.

Larger targets for selection. On average, linked items are large in size (compared with embedded icons) and spatially distinct. This reduces the time it takes users to point the cursor at them.

Fast backtrack command. Backtracking is a frequent activity—for every move 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 mouse cursor. The user need only move the cursor to an empty area of the frame and click the Back button. Because there is usually abundant space on frames to "hit," invoking the Back command takes 0.7 seconds on average, compared with 1.5 seconds to click on a menu or a "close box." This small difference adds up since the Back command may be used several hundred times per hour. From the user's perspective, it is not just the time saved, but the reduced mental and physical effort.

No scrolling. KMS provides fast navigation as an alternative to scrolling.

Should graphical views of the database structure be provided?

In addition to the breadth-first view of frames, KMS provides only one other view of the contents of the database—a linear view of a hierarchy of frames. With this view, users can create well-formatted documents from material in KMS, both on-line and in hardcopy. We do not provide a graphical browser.

Periodically we consider providing additional views in KMS, but each time we retreat. We believe such views are limited in value, except perhaps for large, essentially non-hierarchical structures. This belief is supported by our ZOG user studies. These studies revealed that users rarely made use of the multi-node views that were available. The breadth-first view of frames, combined with rapid system response, seems to serve well for navigation within the database. Additional graphical views might be useful, but we are not convinced they are worth the additional complexity.

How can disorientation be reduced?

The classic hypermedia problem is the "getting lost problem," a problem that 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 skeleton. KMS strongly encourages a top-down, stagewise refinement approach to organizing material in the database. The resulting hierarchical "skeleton" in the database helps users build a coherent mental model of the database. They can remain oriented when navigating because they can always see whether they are selecting a hierarchical link or a cross-reference (annotation items are prefaced by "@").

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. In addition, if a user has tunneled over to an unfamiliar part of the database, he can follow links ascending up through the hierarchy to learn what the global context is.

Flagging previous selections. KMS flags the item linking to the frame from which the user has just backtracked.

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

How can the user search for information other than browsing?

In a large hypermedia database with thousands of nodes users can have trouble locating information solely by browsing, either because the structure of the database is not well suited to their search, or because they have forgotten where they stored something.

KMS addresses this issue by providing a program that can search for text strings within any hierarchy of frames. This method of searching is particularly useful because it lets users make use of whatever knowledge they have about the location of some information to constrain the scope of the search.

The results of a search are represented in frame form, with items linked to the frames where matches were found. (Selecting one of these items displays the frame with the matched text strings highlighted.)

We are currently exploring the use of a global index that indexes every frame by its meaningful terms. Each frame is automatically re-indexed whenever it is modified. We expect this mechanism, combined with fuzzy matching criteria (e.g., relevance measures, synonyms, etc.), will significantly improve the user's ability to locate particular frames by content.

How can the user interface be tailored?

Our experience shows that users have a strong desire to tailor a user interface to suit their individual preferences. To accommodate this desire, we provide, like NoteCards, many parameters the user can customize. The parameters affect such aspects of KMS as mouse cursor movement, visual cues to indicate type of navigation transition between frames (e.g., forward or backward), and what frames initially appear in the windows.

COLLABORATION ISSUES

From the beginning, both ZOG and KMS have been designed to support a community of users, where users can jointly develop and share data rather than simply exchange it. Collaboration takes the form of shared access to a database of knowledge artifacts such as articles, plans, reports and memos. We liken ourselves to workers at a large construction site, each making contributions to the global structure being built. We don't think of ourselves as "collaborating"; we just see ourselves as knowledge workers working side by side.

Here we examine some issues for supporting collaborative work.

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

KMS provides a community of users with 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 on a single time-sharing system, but with vastly improved response and display bandwidth.

Multiple users can work simultaneously on a common project such as developing a large proposal. Users can easily see what others have done, make comments, print out any hierarchical section of the proposal at any time, etc. There is no database administration—users simply evolve the database as their common sense and group norms dictate.

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, which holds the location of all framesets. This location information is itself represented in KMS frames, making it straightforward to update it from within KMS. (All file servers containing a portion of the KMS database have automatically-maintained backup copies of this location information, to be used if the master is unavailable.)

How can interference among multiple users be reduced?

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? KMS provides more "elbow room" for people to work, because a large artifact such as a document is broken into many frames, and people do not interfere with each other at all when they are editing

different frames. Then, since interference is rare, KMS can use an optimistic concurrency control mechanism to reduce the inefficiencies of locking.

More elbow room. In KMS, the unit of representation is small—frames average two paragraphs of content. As a result, the number of frames in a KMS database rapidly outnumbers its users. Since updating of the database occurs at the individual frame level, conflicts between users can occur only when they attempt to modify the same frame at the same time. Our experience shows that conflicts rarely occur since users usually are working in different areas of the database, even when they are working on the same document. For instance, there were several occasions when all three authors of this article were reading and modifying its frames simultaneously, yet conflicts were rare, because there are over 200 frames that make up the article.

Optimistic Concurrency. 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 *optimistic concurrency control*, which makes 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 a user edits a frame, one will necessarily be able to save the changes without any problem. At the time a user attempts to save the changes, he may be informed that someone else has already saved changes to the same frame. This means that the tentative changes cannot be saved, because they would revoke the other user's changes. What KMS does in this case is to temporarily save the user's changes in a newly created frame, so that he can then map them into the new version of the original frame. (To reduce the time window during which the user can be tripped up, KMS double-checks the current version number of the frame at the time the user first makes a change to the frame.)

In those rare instances when users are working on the same small set of frames, they can cope by using informal frame locking conventions. Namely, they can place 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, a user can employ this informal locking to alert others of plans to work in some area of the database over an extended period of time.

Optimistic concurrency control may seem unwise, since it is not a foolproof mechanism for preventing interference between users. But adopting it allows us some benefits that we feel well outweigh its drawbacks. The most important benefits are that it facilitates eliminating the mode boundary between navigating and editing, and avoids the complexity of locking mechanisms for a wide-area network of heterogeneous machines.

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 (initially 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 cannot access the frame.

An intermediate form of protection allows users to add annotation items to a frame, but not modify existing items. In practice, however, we often leave frames unprotected to encourage our colleagues to make minor corrections such as typos. We simply rely on their good will not to damage our frames.

How can communication be supported?

We do not use special purpose systems for communication such as electronic mail or bulletin boards. Instead we perform these activities using KMS, in a manner quite different from conventional mail and bulletin boards. The flexibility of KMS frames allows them to easily function as mailboxes, bulletin boards, or ad hoc discussion areas (see Figure 6). Rather than sending or posting messages, KMS users "grow" a conversation at some location in the database while simultaneously preserving the global structure of the communication. By comparison, mail and bulletin board systems fragment conversations.

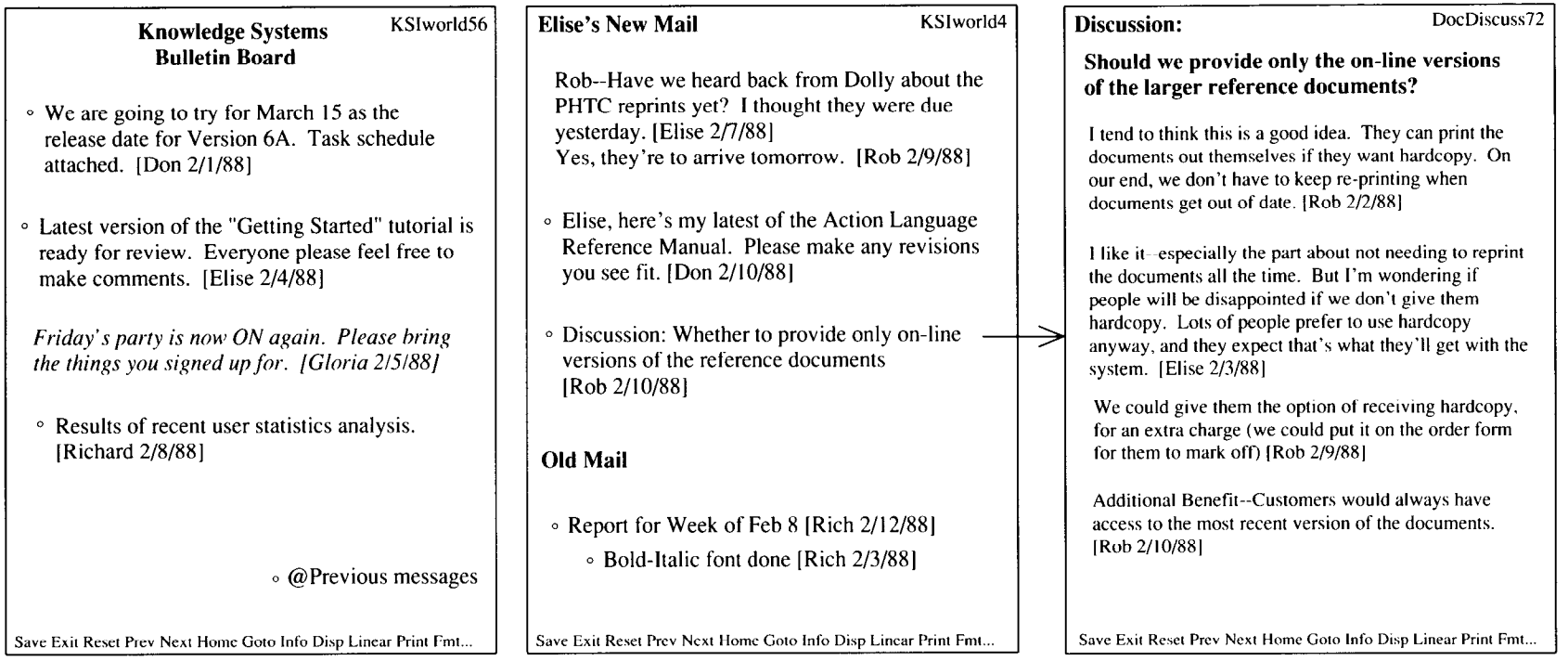
How can annotation be supported?

A principal form of collaboration is commenting on the work of others. KMS supports such comments in an unusual way, allowing annotations to be placed directly on the frame they refer to.

Non-disruptive annotation. When a typical knowledge artifact is represented as a hierarchy of KMS frames, each frame ends up with only two paragraphs on average. This means that even when a frame is displayed using half the screen, there is usually ample space for comments by other users, as well as additional notes by the author. Thus comments and notes can be authored and read in the context of what they refer to without obscuring this material. By convention, users represent comments as *annotation items* so that they will be ignored by KMS programs that process hierarchies; i.e., they will not appear in the author's "official" document.

One benefit of non-disruptive annotation is that authors are encouraged to give other people early access to their documents while they are still in progress, because they can continue to work while others review and make comments.

Relaxed group norms. Since it is so easy to place a comment on a frame, people working in KMS usually feel free to make comments on other people's frames. In fact, they often feel free to reposition other people's comments to make room for their own, and to respond directly to other comments already on the frame. Sometimes an author revisits a document frame and finds that a debate has broken out among the commentators.



Bulletin Board frame

- People come to the frame and make their contribution, usually "signing" it with their name and the date.
- A bulletin board entry may be linked to another frame. In the example above, there are entries linked to a task schedule, an on-line tutorial document, and an analysis of user statistics.

Mail box frame

- This example shows one person's mail box frame. People place their messages directly on the frame.
- As with the bulletin board, messages may be linked to other documents, or to a frame elaborating on the message. (Confidential information usually is placed on a protected frame that is linked to the message.)
- Messages can be passed back and forth, with a new response appended each time. After 2 or 3 messages are appended, the people involved usually create a discussion frame for that topic. (In this example, one item is linked to the Discussion frame at the right.)

Discussion frame

- A discussion frame is similar to a bulletin board frame. People place their comments directly on the frame. Prior comments are read in context since the history of the discussion can be read on the frame. People exploit this context by responding directly to earlier comments as if they were holding a conversation.

FIGURE 6. KMS Frames Being Used for Communication

Increased incentive to comment. The convenience of placing comments on the same frame as the material being commented on provides considerably more incentive to users than if they were forced to put their comments on another frame. This is equally true for an author's own notes, especially for quickly jotting down a stray thought while fresh in the mind. Later, at a more convenient time, the stray thought can be moved to a more appropriate home in the database.

MISCELLANEOUS ISSUES

The following are several issues that do not readily fit into our major categories, yet are too important to disregard.

How can the functionality of the system be extended?

To extend the functionality of KMS, we provide a general-purpose block-structured programming language. The language has a simple syntax and control structure, but it is powerful because it has high-level primitives for creating and manipulating KMS structures. The language is similar in scope to HyperTalk, the programming language in HyperCard [4], [12].

How can development of large databases be facilitated?

Small hypermedia databases are of limited interest, but large, interesting databases are difficult to create. This poses an efficiency problem for hypermedia system designers to solve. If it is too inconvenient to contribute to 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:

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

Rapid creation of new frames. To create a frame, the user just clicks on an unlinked item. The user can be editing a new frame less than two seconds after deciding to create it.

Ease of editing structure. We find that users make many more structural changes to their KMS-based documents than they do to documents based in text files. We attribute this behavior to the structural clarity of the breadth-first view and the ease of making structural changes. For instance, users can more readily perceive that the subsections of a given section are incomplete or out of order.

Rapid navigation. Users need to be able to move around rapidly in order to get to where they wish to build. Rapid navigation is also important for revision of the database, since restructuring is done by navigating while the cursor is dragging an item or set of items.

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 [22] developed elaborate, built-in schema mechanisms for ZOG and studied their use experimentally.

Tools for importing external databases. KMS provides a number of utility programs for mapping in material from other sources (e.g., text and bit-map files, other databases).

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 storage devices.

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

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 printed documents from material in a KMS database. Since frames provide a local WYSIWYG view, there is a natural process for composing the document: concatenating the contents of frames from a hierarchy in depth-first order. This document composition program can be invoked at any level of a hierarchy of frames, thereby enabling users to get just the portion of a document they want.

The default format for composing documents can be modified by placing keyword annotation items on frames (e.g., "@NewPage," "@Figure," "@Index: Copying" etc.). These items, like other meta-level items such as notes and comments, are usually placed in a corner of the frame 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 markup systems such as Scribe and TEX. (HyperScribe, a program recently developed by Scribe Systems, integrates KMS and Scribe for large-scale document engineering.)

CONCLUSION

Hypermedia has a rich design space, especially when implemented for networks of large-screen workstations. The good news is that hypermedia designers can expect lifetime employment (our design database for KMS contains well over a thousand potential enhancements and grows larger every day). The bad news is that the rich design space for hypermedia presents designers with many agonizing tradeoffs. In the thrill of implementation, designers may avoid facing up to these tradeoffs and thus permit the system to become too complex. Our experience with these tradeoffs has encouraged us to practice a design philosophy of *voluntary simplicity*, striving to make do with fewer concepts and mechanisms.

If there is one central lesson from 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 its "look and feel" [5]. We believe this because we have seen the formative influence of the ZOG and KMS data models on all other aspects of these systems. For instance, the properties of a frame—its fixed size, its spatial nature, how links are represented within it, its homogeneous format, etc.—contribute significantly to the global nature of the system.

One of the most pressing problems of human–computer interaction is the ever-growing complexity of software systems. We believe that this complexity is caused largely by the complexity of the underlying data models. The problem is compounded when users must cope with a multitude of systems that have inconsistent data models. Standardizing the user interface produces, at best, a superficial similarity, but does not truly address the problem. The underlying differences still remain, and as programs become more sophisticated (e.g., distributed versions) these differences are even more troublesome for users.

To reduce these differences, we would like to see standards develop for data models, and greater freedom preserved for users to customize interfaces to suit their preferences. Perhaps the hypermedia data model, with its potential for simplicity, can provide a basis for such standardization. If so, hypermedia may eventually replace the desktop metaphor as the reigning human–computer interaction paradigm.

Acknowledgments. We wish to acknowledge the contributions of many people over the years. Those who were involved with ZOG at CMU: Allen Newell (our ZOGfather), 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

1. Akscyn, R. The next computer revolution? In *Pittsburgh High Technology*. Pittsburgh High Technology Council, Pittsburgh, Penn. Nov./Dec. 1987.
2. Akscyn, R., and McCracken, D. The ZOG approach to database management. In *Proceedings of the Trends and Applications Conference: Making Database Work* (Gaithersburg, Maryland, May). 1984.
3. Akscyn, R., and McCracken, D. ZOG and the USS *Carl Vinson*: Lessons in system development. In *Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84)* (London, England, Sept.). Elsevier Science Publishers B.V., Amsterdam, Netherlands, 1984.
4. Akscyn, R., and McCracken, D. *A Comparison of KMS and HyperCard*. Knowledge Systems, Murrysville, Penn., 1988.
5. Akscyn, R., Yoder, E., and McCracken, D. The data model is the heart of interface design. To appear in *Proceedings of CHI '88* (Washington, D.C., May). ACM, New York, 1988.
6. Bush, V. As we may think. *Atlantic Monthly 176*, (July 1945). Reprinted in *CD ROM: The New Papyrus*, Microsoft Press, 1986.
7. Card, S.K., Pavel, M., and Farrell, J.E. Window-based computer dialogues. In *Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84)* (London, England, Sept.). 1984.
8. Conklin, J. A survey of hypertext. *IEEE Computer* (Sept. 1987).
9. Delisle, N., and Schwartz, M. Neptune: A hypertext system for CAD applications. In *Proceedings of ACM SIGMOD International Conference on Management of Data* (Washington, D.C., May). 1986, pp. 132–143.
10. Engelbart, D. A conceptual framework for the augmentation of man's intellect. *Vistas in Information Handling*, 1, P.W. Howerton and D.C. Weeks, Eds. Spartan Books, 1963.
11. Garrett, L., Smith, K., and Meyrowitz, N. Intermedia: Issues, strategies, and tactics in the design of a hypermedia document system. In *Proceedings of the Conference on Computer-Supported Cooperative Work* (Austin, Texas, Dec.). 1986, pp. 163–174.
12. Goodman, D. *The Complete HyperCard Handbook*. Bantam Books, New York, N.Y., 1987.
13. Gullichsen, E., D'Souza, D., Lincoln, P., and The, C. The PlaneTextBook. MCC Tech. Rep. STP-333-86(P). 1986.
14. Halasz, F. Reflections on NoteCards: Seven issues for the next generation of Hypermedia systems. *Commun. ACM 31*, 7 (July 1988), 836–852.
15. Mantei, M. A study of disorientation behavior in ZOG. Ph.D. dissertation. University of Southern California, 1982.
16. McCracken, D., and Akscyn, R. Experience with the ZOG human-computer interface system. *International J. Man-Machine Studies 21* (1984), 293–310.
17. Nelson, T. Replacing the printed word: A complete literary system. *Information Processing 80*, S.H. Lavington, Ed. IFIP, 1980.
18. Newell, A. Notes for a model of human performance in ZOG. Tech. Rep., Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, Penn., 1977.
19. Newell, A. An on-going case study in technological innovation. In *Advances in Information Processing in Organizations*, L. Sproull and P. Larkey, Ed. 1985.
20. Newell, A., McCracken, D., Robertson, G., and Akscyn, R. ZOG and the USS *Carl Vinson*. In *Computer Science Research Review*, Carnegie-Mellon University, Pittsburgh, Penn., 1981, pp. 95–118.
21. *Proceedings of Hypertext '87* (Chapel Hill, North Carolina, Nov.). 1987.
22. Ramakrishna, K. Schematization as an aid to organizing ZOG information nets. Ph.D. dissertation. Computer Science Dept., Carnegie-Mellon University, Pittsburgh, Penn., 1981.
23. Robertson, C.K., and Akscyn, R. Experimental evaluation of tools for teaching the ZOG frame editor. In *Proceedings of the International Conference on Man/Machine Systems* (Manchester, England, July). 1982.
24. Robertson, C.K., McCracken, D., and Newell, A. Experimental evaluation of the ZOG frame editor. In *Proceedings of the 7th Canadian Man-Computer Communications Conference* (Waterloo, Ontario, June). 1981, pp. 115–123.
25. Robertson, G., McCracken, D., and Newell, A. The ZOG approach to man-machine communication. *International J. Man-Machine Studies* (1981).
26. Schultz, J., and Davis, L. The technology of PROMIS. In *Proceedings of the IEEE* (Sept. 1979), pp. 1237–1244.
27. Shneiderman, B. User interface design and evaluation for an electronic encyclopedia. Tech. Rep. CS-TR-1819. Dept. of Computer Science, University of Maryland, College Park, Maryland, Mar. 1987.
28. Shneiderman, B., and Morariu, J. The interactive encyclopedia system (TIES). Dept. of Computer Science, University of Maryland, College Park, Maryland, June 1986.
29. Yankelovich, N., Smith, K.E., Garrett, L.N., and Meyrowitz, N. Issues in designing a hypermedia document system. In *Interactive Multimedia*, S. Ambron and K. Hooper, Eds. Microsoft Press, 1988.
30. Yoder, E., McCracken, D., and Akscyn, R. Instrumenting a human-computer interface for development and evaluation. In *Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84)* (London, England, Sept.). 1984.

CR Categories and Subject Descriptors: H.3.3 **[Information Storage and Retrieval]**: Information Search and Retrieval—*Search Process*; H.3.5 **[Information Storage and Retrieval]**: On-Line Information Services; H.4.1 **[Information Systems Applications]**: Office Automation; H.4.3 **[Information Systems Applications]**: Communication Applications
General Terms: Human Factors
Additional Key Words and Phrases: Conceptual data model, direct manipulation, human–computer interaction, hypermedia, hypertext

Authors' Present Addresses: Robert M. Akscyn, Donald L. McCracken, Elise A. Yoder, Knowledge Systems, Inc., 4758 Old William Penn Highway, Murrysville, PA 15668.