

A Spatial Hypertext Editor

Carmen Ximena D'Arlach
John J. Leggett

December 1994
TAMU-HRL 94-005

Table of Contents

Abstract	1
1. Introduction	2
1.1 The SP3 model of hypermedia	2
1.2 The SP3 prototypic architecture	3
1.3 HyperEd and SP3	5
2. Design	6
2.1 The metaphor	6
2.2 Persistent storage	7
2.3 Hypertext capabilities	7
3. HyperEd Functionality	10
3.1 Running HyperEd	10
3.2 Document properties	11
3.3 Document utilities	13
3.3.1 Opening a new document	13
3.3.2 Opening an existing document	13
3.3.3 Saving a document	15
3.3.4 Deleting a document	16
3.3.5 Including a document	16
3.3.6 Importing an ASCII document	16
3.3.7 Exporting an ASCII document	17
3.4 Editing utilities	17
3.4.1 The Keyboard	18
3.4.2 The <i>Edit</i> Menu	18
3.4.3 The Mouse	21
3.5 Link Utilities	25
3.6 Fonts and Colors	29
3.7 Help	30
3.8 Quitting HyperEd	31
4. Future Work	32
5. Conclusion	33
References	34

Abstract

This report discusses the design and implementation of *HyperEd* - a spatial hypertext editor. HyperEd is the first participating application to be fully incorporated to the System Prototype 3 (SP3) architecture under development at the Hypermedia Research Lab at Texas A&M University. The report includes a brief overview of SP3, a detailed description of the functionality of HyperEd, and considerations for future work. HyperEd is one of the first steps in implementing a series of integrated, collaborative, hypermedia based tools that will take advantage of the features provided by SP3.

1. Introduction

This paper discusses HyperEd, a spatial hypertext editor. HyperEd is a participating application of SP3, a prototypical implementation of a hypermedia model being researched at the Hypermedia Research Lab. SP3 is an open-system architecture composed of several layers of clients and servers that allow hypermedia connections to be forged among applications that are able to participate in a distributed link services protocol [Schnase 1992]. In order to give the reader some background on this environment, the remainder of this section will briefly introduce the main aspects of the SP3 model of hypermedia and its prototypic implementation. For further detail, refer to [Schnase 1992], [Schnase 1994], and [Leggett 1994].

1.1 The SP3 model of hypermedia

The SP3 model of hypermedia consists of six elements as shown in Figure 1. *Applications* are programs that allow the user to manipulate different information resources. *Components* represent information managed by applications. *Persistent selections* represent selections within components that persist between application sessions and can be accessed at a later time. *Anchors* and *links* are processes that connect *persistent selections*. *Anchors* can be associated with various *persistent selections* and *links* can be associated with various anchors. Relationships between these elements are termed *associations*. *Associations* may be collected in *association sets*, providing for multiple *contexts*. In this model, the *associations* between *persistent selections* make up the network structure of hypermedia [Leggett et al. 1994].

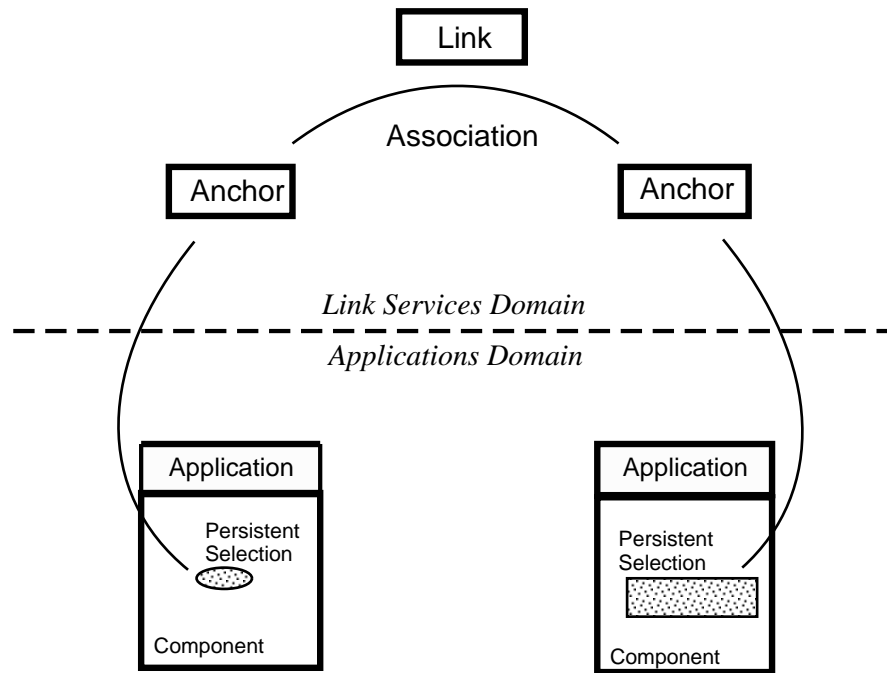


Figure 1. Conceptual model of hypermedia underlying SP3.

1.2 The SP3 prototypic architecture

The SP3 architecture implements this model by moving associations, links and anchors to a separate hypermedia layer. As shown in Figure 2, SP3 consists of five major components: *Participating Applications* (Apps), the *Link Services Manager* (LSM), the *Association Set Manager* (ASM), the *Object Manager* (OM), and the *Storage Manager* (SM).

Applications that are able to interact with the LSM are referred to as *Participating Applications*. As stipulated by the model, *Apps* manipulate components and persistent selections within components. The LSM is a server process that provides run-time support for interapplication linking. It coordinates interprocess communication required to implement the hypermedia

functionality required by participating applications. These activities include the attachment and detachment of links and anchors, browsing operations, and context operations.

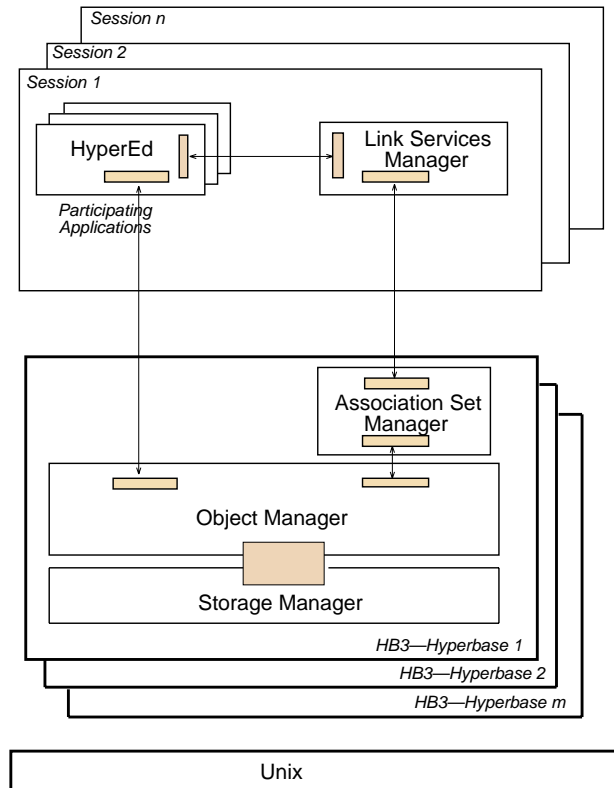


Figure 2. The SP3 architecture and HyperEd.

The ASM, the OM, and the SM make up the *hyperbase management system* (HB3). HB3 is a multi-user multi-hyperbase system. The ASM provides persistent and sharable storage for the connectivity data that link information together. While the LSM provides run-time linking and navigational operations, the ASM manages the static storage of the hypermedia associations. The OM manages simple unstructured components as well as composite components, accommodating a wide range of application requirements. The SM maps HB3's data model into physical storage [Leggett 1994].

1.3 HyperEd and SP3

HyperEd is a participating application of the SP3 architecture, and as such, interacts with all servers of the architecture. As shown in Figure 2, HyperEd interacts directly with the OM and the LSM and indirectly with the ASM and the SM. It uses the OM to handle persistent storage of documents and document components, and the LSM to do interapplication linking. HyperEd communicates with these two servers via the HRLT interprocess communication (IPC) toolkit.

HyperEd was developed on a Unix platform using the C programming language, XWindows, and XView. It runs in an XWindows environment, preferably under an OpenWindows 3 window manager.

2. Design

HyperEd was designed to provide a testbed for SP3's hypermedia model and to exploit its prototypic implementation. A secondary goal was to provide an unobtrusive environment for the realization of these two goals. For this reason, HyperEd's interface was designed to be as simple as possible without losing key functionality.

HyperEd has a "click-to-type" interface. It is mouse and menu driven and, like most text editors, provides a set of commands that allows users to create, store, and retrieve documents. All commands are executed through mouse button combinations or by selecting a menu item. However, there are three characteristics that set HyperEd apart from most common text editors: metaphor, persistent storage, and hypertext capabilities. The next subsections discuss these characteristics in detail.

2.1 The metaphor

"In the worlds of hypertext and hypermedia, a spatial metaphor is the most common means of providing an interface to information. People *navigate* information *spaces* or *worlds*" [Laurel 1994]. HyperEd builds on the spatial metaphor of hypertext by giving the user freedom to place text anywhere on the writing space. There are no pages or preset rows and columns, just an unrestricted space called the *writing space*.

HyperEd documents are composed of text items, units of information with the following characteristics:

- (1) they are composed of character strings of any length;
- (2) they have a set of display properties;
- (3) they can be located anywhere on the writing space; and
- (4) they can be persistently selected, anchored, and linked.

Each text item is a unit on its own, an idea inspired by KMS [Akscyn 1989]. Most operations affect whole text items. Text items can be moved, copied, and deleted.

2.2 Persistent storage

The Object Manager (OM) is a server process that provides persistent and sharable storage for objects within HB3. The primary abstraction supported by OM's data model is that of an object. OM objects can be simple or composite. Simple objects are arbitrary-size byte strings having unique identity and optional attribute-value pairs. Composite objects consist of references to simple and composite objects and have a unique identity and optional attribute-value pairs. OM features include: transaction management, support of object *composites*, arbitrary number of *attributes* and storage of objects of unlimited size. Simple object functions implement the fundamental data storage and retrieval operations; composite functions group objects into complex objects; and attribute functions allow applications to handle arbitrary attributes associated with simple and composite objects.

HyperEd takes advantage of this functionality by letting the OM take care of all of its storage needs. Text items are stored as simple objects, with text properties (e.g. font, color) stored as attributes. In a similar way, documents are composite objects made up of the corresponding text items, and their properties are also stored as attributes. OM functions are called upon to retrieve and save documents.

2.3 Hypertext capabilities

The ability to fully participate in Link Services gives HyperEd a unique and distinct feature not found in many other text editors. HyperEd allows its users to create associations among text items in the same document, among text items in different HyperEd documents, and among text items in HyperEd documents and components of other participating applications.

In this implementation, the LSM interface consists of a menu that gives the user access to anchoring and linking capabilities as well as context management functionality [Sanchez *forthcoming*]. The LSM requires that the application be able to uniquely identify its persistent selections within its domain. For this purpose, HyperEd assigns each text item a unique identifier as soon as it is created. Identifiers are implemented as indices to a table that points to text items. Each text item has an entry in this table and each text item has a persistent selection bit. This

combination is used to identify persistent selections when dealing with the LSM. The indices are kept unique by always assigning a new entry to a new text item. Entries corresponding to deleted text items are never reused, thereby maintaining LSM information consistency.

As a participating application, HyperEd has to be able to perform the operations listed in Table 1. Most of these operations affect persistent selections and involve some type of HyperEd-LSM interaction.

Table 1. Operations required to participate in SP3

Making a text item persistent for anchor attachment
Removing persistent selections
Displaying, or highlighting in some fashion, selections and persistent selections
Displaying and hiding persistent selections that have only an anchor attached
Displaying and hiding persistent selections that have linked anchors attached
Identifying persistent selections by assigning each one a unique id
Maintaining persistent selection information in some type of data structure
Loading components on demand and "going to" given persistent selections
Mapping display geometry coordinates to persistent selections

In HyperEd, a user selects an item with a click of the left mouse button. This type of selection is temporary and only one text item can be selected at a time. LSM operations require a different type of selection: a persistent selection. This type of selection will remain until otherwise specified by the user. Furthermore, to indicate which persistent selections should be affected by LSM operations, HyperEd allows the user to mark and unmark text items. This is done by chording the left and right mouse buttons simultaneously on a text item. Only marked items are affected by link operations and several items may be marked at the same time.

Glyphs and colors are used to denote whether items are marked, persistently selected, anchored,

and/or linked. Red glyphs indicate the items are marked and black glyphs indicate that they are unmarked. Figure 11 in Section 3.4 shows the different glyphs used by HyperEd.

3. HyperEd Functionality

This section describes the functionality of HyperEd in detail. It begins with instructions on how to run HyperEd and a description of document properties. The following sections describe the options available in each menu in the same order in which they appear in the interface: *Document*, *Edit*, *Links*, *Fonts*, and *Quit*.

3.1 Running HyperEd

HyperEd runs in the XWindows environment, preferably under an OpenWindows 3 window manager. It is OpenLook compliant.

The following command will start HyperEd:

```
hypered -hbhost <hostname> -hbname <hbase> [-doc_name <doc_name>] [-doc_id <id>]  
[-ps_id <psid>]
```

where,

hostname is the server where the hyperbase is

hbase is the name of the hyperbase where the document is

doc_name is the name of the document to be opened

doc_id is the OM document id of the document to be opened

ps_id is the id of the persistent selection to be displayed

The last three parameters -- *doc_name*, *doc_id*, and *ps_id* -- are optional. If these parameters are not given, the application opens a new document by default.

Figure 3. HyperEd's User Interface.

HyperEd's user interface, shown in Figure 3, consists of a menu bar and a view window. The menu bar gives the user access to a set of operations to manage documents and individual text items. The view window displays part of the writing space where documents are created and edited. This window can be resized to be smaller or larger and two scrollbars allow the user to move around the page horizontally and vertically.

3.2 Document properties

All HyperEd documents have a set of properties as described below. The first three -- *Name*, *Type*, and *Protection* -- are under user control and the remaining properties are handled internally by HyperEd.

Name. Document names can be up to twenty characters long. Any alphanumeric string is acceptable. There are only two restrictions: names cannot start with a blank character and they

must be unique for each user. Examples of document names are: *HyperEd: the greatest editor*, *Hello 12345*, and *4th of July*.

Type. A type is just a text description (up to 20 characters long) specified by the user. A document can be of type letter, memo, draft, poem, etc. This is meant to give the user a flexible way to organize documents. The default type is "Doc."

Protection. The owner of the document can use this property to give or deny access to other users. A document can be protected as follows:

None: All users have the same access to open and update the document. Still, the only user allowed to delete this document is the owner. This is the default.

Read: A document that is read-protected can only be opened by its owner. A document that is read-protected is invisible to all users except the owner.

Write: Write protection will not allow other users to save any changes to the document. Other users will be able to open a write protected document and copy it, that is, save it under a new name.

Owner. The *owner* property cannot be edited by any user. As all new documents are always saved under the current user-id (with owner equal to userid), users cannot create documents and save them under user-ids other than their own.

Date created. This is the date when the document was saved for the first time.

Date last modified. This is the date when the document was last modified.

Date last modified by. This field holds the user-id of the last person that updated the document.

3.3 Document utilities

The *Document* menu, shown in Figure 4, provides the user with a set of facilities to handle complete documents. The following sections explain how each of these options work.

Figure 4. The *Document* menu.

3.3.1 Opening a new document

After working in a document, the user might decide to start a new document. The *New* button will close the current document and clear the page for a new one. Another way to start a new document is to start HyperEd without the *-doc_name*, *-doc_id*, and *-ps_id* parameters.

3.3.2 Opening an existing document

The *Open* button under the *Document* menu will bring up the dialog box shown below. This dialog box has multiple purposes. It can be used to browse documents and their properties and to open, include, or delete a document. The *Document* menu presents the user *Include...* and *Delete...* buttons to help the user find this functionality. They will both bring up this same dialog box.

Figure 5. The *Open* dialog box.

The list of existing documents will initially show all the documents that belong to the current user. To see a different set of documents, such as documents that belong to a different user or documents of type **memo** only, the user needs to edit the *Name*, *Owner*, and/or *Type* entries at the top and then press the *Filter* button. The list of documents will be updated with only those documents that match the specific choices.

HyperEd queries the hyperbase for documents that partially match the *Name* entry and that exactly match the *Owner* and *Type* entries. This facility allows the user to organize and locate documents in a flexible manner. If any of these three entries is left blank, HyperEd will, by default, query the hyperbase for all documents that belong to the current user.

The *Properties...* button will bring up a window with information on the currently selected document in the scrolling list allowing the user to browse its properties before opening it.

Figure 6. The *Properties* panel.

As mentioned earlier, once the user has selected a document from the list, he or she has three options:

Open. This option will load the document from the hyperbase and display it in the view window. If the current document is unsaved, the user will be reminded to save first. Only

one document can be open at any time. Therefore, if the user opens a document without saving it first, he or she will lose all unsaved edits.

Include. This option allows the user to combine two documents. It works by adding the text items from a document stored in the hyperbase to the document being edited. The copied text items will become part of the current document while the source document will be left untouched in the hyperbase.

Delete. This option is used to remove a document permanently from the hyperbase. As usual, a warning message will appear asking the user to acknowledge the operation. Once a document has been deleted it is unrecoverable.

3.3.3 Saving a document

There are two options under the *Document* menu to save the current document: *Save* and *Save as...* *Save as...* will save the current document under a new name. This applies to existing documents as well as to new documents. This option will always ask the user for a new document name as shown in Figure 7 below. The user can also edit the default values for the type and protection of this document. Default values are "Doc" and "None," respectively. All three attributes, *name*, *type*, and *protection*, can be changed later by overwriting the existing values.

Save is meant to update documents that have previously been saved. If the document has not been saved yet, it will display the *Save as* dialog box which will ask the user to name the document. As mentioned earlier, document names must be unique within each user's space.

Figure 7. The *Save as* dialog box.

3.3.4 Deleting a document

The *Delete...* button under the *Document* menu is used to remove a document permanently from the hyperbase. As explained in Section 3.3.2, *Opening an existing document*, this button will also display the dialog box shown in Figure 5.

To remove a document, the user must first select the document from the scrolling list and then press the *Delete* button. As usual, a warning message will appear and the user will be asked to acknowledge the operation. Once a document has been deleted it is unrecoverable.

3.3.5 Including a document

As explained in Section 3.3.2, *Opening an existing document*, this button will also display the dialog box shown in Figure 5. This feature allows the user to combine two documents. It works by adding the text items from a document stored in the hyperbase to the document being edited. The copied text items will become part of the current document while the source document will be left untouched in the hyperbase.

3.3.6 Importing an ASCII document

HyperEd provides this very simple facility to convert an ASCII document to HyperEd format. This option creates a new HyperEd document by converting each paragraph in the source document into a HyperEd text item. The new document can then be edited and saved like any other HyperEd document.

Figure 8 shows the *Import Document* dialog box. The user needs to provide the full path to the directory where the file is located and the file name. Pressing the *Import* button will either display the converted file or warn the user that it could not find the file.

Figure 8. The *Import* dialog box

3.3.7 Exporting an ASCII document

This facility is provided as a complement to the import facility and as a primitive tool to print the contents of a HyperEd document. The *Export* dialog box shown in Figure 9 allows the user to send an ASCII version of the document to a Unix file or directly to the printer. Unfortunately, all colors and fonts are lost. Text items are placed one after the other from top to bottom. A fully integrated printing facility will be incorporated in a future version HyperEd.

Figure 9. The *Export* dialog box.

3.4 Editing utilities

Most editing operations are performed directly on items using the keyboard, the *Edit* menu, or the mouse. **Keyboard** operations include inserting and deleting characters, and using the arrow keys to move the caret -- a text cursor that looks like a dash line -- within the text item. The *Edit* menu includes operations such as delete, undelete, cut, paste, copy, move, and find. The **mouse** is mainly used to select options from menus and dialog boxes, to select text items, and to control the caret in the writing space. Moreover, different combinations of mouse button presses provide some of the functionality of the *Edit* menu -- select, copy, move, cut, and paste -- and of the *Links* menu -- mark, unmark, and follow link. These functions are identical and perform the same operations regardless of which way they are carried out. This redundancy in the implementation is aimed at

giving the user some added flexibility. It is expected that experienced users will prefer using the mouse to invoke functions rather than using menu selection. The following sections explain in detail how each set of functions work.

3.4.1 The Keyboard

As in most text editors, the **keyboard** is the main source of input in HyperEd. Text items are created by first clicking the left mouse button on an empty space of the view window to position the caret and then typing the text item. The **Back Space** is used to delete characters one at a time.

some text was deleted --- retype

3.4.2 The *Edit* Menu

To use the functions of the *Edit* menu, the user is expected to first select a text item or an empty location in the page and then press an *Edit* menu button. Figure 10 shows the *Edit* menu and its options. The remainder of this section presents a detailed explanation of each *Edit* menu item.

Figure 10. The *Edit* menu.

Delete and Undelete. A text item can be deleted by first selecting it with the left mouse button and then pressing the *Delete* button in the *Edit* menu. This operation does not delete text items permanently until the document is saved. Instead, deleted items are placed in a temporary buffer from where they can be restored in two ways: one by one in the reverse order in which they were deleted or all together in a single operation. These two operations can be found under the *Undelete* button in the *Edit* menu: *Undelete last* and *Undelete all*.

Cut. This operation will remove the item and place it in a buffer from where it can be restored later by the *Paste* operation. To cut an item using the *Edit* menu, the user first selects the text item using the left mouse button. An enclosing rectangle appears around the selection. Next, the user presses the *Cut* button in the *Edit* menu and the item disappears.

Paste. To use the paste option in the *Edit* menu, the user must select an empty space on the page, i.e. with a click of the left mouse button, and then select paste from the edit menu. The item stored in the buffer will be restored at the current mouse pointer location. Repeating these two steps will restore copies of the item any number of times. If the user places the pointer on top of an existing text item, paste will simply place the restored item on top of it.

Move. To use this option, the user must first select the desired item and then press the *Move* button in the menu. This operation will attach the item to the mouse pointer, change the mouse pointer to reflect the operation (Table 2), and display a footer message giving instructions on how to drop the item. The text item will then follow the mouse pointer as it moves around the page until the user presses the middle mouse button to drop it at its destination. If the user drops the item on top of another, the text of the moving item will be inserted between the two characters of the non-moving item where the click occurred. The two items are then *merged* together into a single item where the properties of the bottom item prevail. The *move* cursor changes to a *merge* cursor when the pointer passes through other items to let the user know that a merge operation is enabled.

Copy. Copying a text item requires almost the same steps needed to move it. First, the user must select the text item and then press the *Copy* button in the *Edit* menu. Again, this operation will attach a copy of the item to the mouse pointer, switch to the copy cursor, and display a footer message indicating how to drop the duplicate.

The copy will follow the mouse pointer until a second click of the right mouse button occurs, which will drop the copied text item at the location chosen by the user. The new text item inherits all of its properties from the original text item except for links, anchors, or marks. Unlike the move operation, an item that is dropped inside of another will just be placed on top of it. They will not be merged together.

Find. This function will take any alphanumeric string of characters and search for it throughout the document. The string may consist of a single character, a word, or a sentence. Blanks are allowed. An enclosing box will highlight the first text item containing the string. Each time the *Find* button is pressed it will search for the next occurrence of the string, highlight the item, and make it visible in the view window.

Figure 11. The *Find* dialog box.

3.4.3 The Mouse

As mentioned earlier, HyperEd has a *click-to-type* interface. This means that the user must click the mouse to position the caret. Each time the user clicks the left mouse button, the caret moves to the corresponding position and the user is ready to type. If the mouse points to an existing text item, the caret is placed at an insertion point and the text typed is inserted before the character indicated by the caret. If the user clicks the left mouse button at an empty location and starts typing, a new text item is created with the default properties.

There are four context-sensitive mouse pointers in HyperEd as shown in Table 2 below. Each cursor guides the user in different contexts of operation.

Operation	Pointer
Select a text item or menu buttons	
Move a text item	mov
Copy a text item	cpy
Merge a text item	mrg

Table 2. Different mouse pointer shapes.

The select operation -- clicking the left mouse button to select text items for further processing -- is the most used operation. Other operations, also available in either the *Edit* menu or the *Links* menu, have been assigned to all possible combinations of mouse button presses. Expert users are expected to take advantage of the speed that this feature brings to the system. The following table shows the equivalence between mouse button press combinations and their effects on a text item.

Mouse Buttons			
Left	Middle	Right	
			none
			copy
			move
			cut
			select
			mark/unmark
			paste
			follow link

Table 3. Mouse buttons and their corresponding operations.

Select. The user selects an item by clicking the left mouse button on top of a text item. Selection is used to make an item the focus of attention for all succeeding operations, like inserting/deleting characters or changing an item's properties.

When an existing text item is selected, the caret is placed at the location where the mouse click occurred. This allows the user to edit the text by inserting or deleting characters. At this time, the item is considered to be selected and operations such as changing display properties or marking the item are enabled.

Whenever the user performs a select operation on empty space, the caret is moved to the corresponding pointer location where the user can start typing a new text item or change default properties for future new text items (see Section 3.6: Fonts and Colors).

Move. Text items can be moved by clicking the middle mouse button on top of a text item. This will attach the cursor to the top left corner of the text item and will cause the text item to move

around the writing space following the mouse pointer. The mouse cursor changes to reflect the operation, and a message is displayed in the footer giving instructions to guide the user. Pressing the middle mouse button again will drop the text item at the current mouse pointer position.

If the attached item is dropped inside the body of another item, both items will be merged into a single one with the properties of the fixed item. The *move* cursor changes to a *merge* cursor when the pointer passes through other items to let the user know that a merge operation is enabled.

Copy. Copying a text item requires almost the same steps needed to move it. Text items are copied by clicking the right mouse button on top of an item. This operation will attach a copy of the item to the mouse pointer, as reflected by the mouse cursor. The copy will follow the mouse pointer until a second click of the right mouse button occurs, which will drop the copied text item at the location chosen by the user. The new text item inherits all of its properties from the original text item except for links, anchors, or marks.

Cut. This operation will remove the item and place it in a temporary location, the cut buffer. The paste operation can be used to restore a copy of this item several times. A new "cut" item overwrites the current one. To cut a text item, the user must click the middle and right mouse buttons on the desired item. The item will disappear.

Paste. This operation will restore the item stored in the cut buffer at the current mouse pointer location. To paste an item, the user merely places the mouse pointer at an empty spot of the page and presses the left and middle buttons simultaneously. Repeating this operation will restore the previously cut item a number of times.

If the user places the pointer on an existing text item, paste will simply place the restored item on top of it. It will not merge both items into a single one.

Mark. This operation will mark the item so that it will be affected by linking operations. To mark an item, the user must click the left and right mouse buttons simultaneously on top of the item. A hollow red bullet will appear next to its top left corner if the item is not persistently selected, anchored, or linked. Otherwise, persistent selection, anchor, and link glyphs will become red. Only marked items will be affected by linking operations.

Follow Link. Linked text items are associated to other text items or other applications' components. To traverse the link that ties them together, the user must click all three mouse buttons on top of the linked item. HyperEd will then notify the LSM. The LSM will take care of displaying the components at the other end of the link. This operation is also available in the *Links* menu.

3.5 Link Utilities

Link utilities allow users to create associations among text items and other applications' components. Associations are created by a four step process involving the *Links* menu and the *LSM* menu (Figure 13).

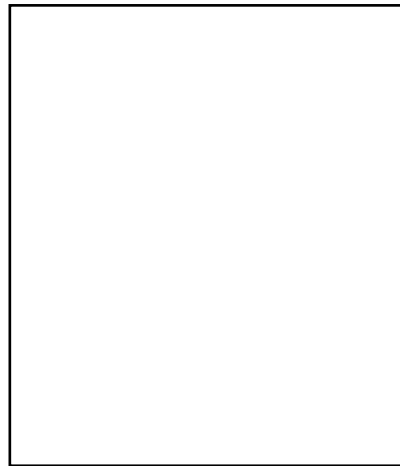


Figure 12. HyperEd Glyphs.

A text item can be marked, persistently selected, anchored, and/or linked. The glyphs shown in Figure 12 are used to denote the state of an item. Red glyphs indicate that the item is marked and will be affected by subsequent link operations such as make/unmake persistent, attach/detach anchor, and attach/detach link. Black glyphs indicate that the item is not marked and therefore will not be affected by any of these operations.

The last item in the figure is both linked and marked. This will occur when the user wants an item to become part of more than one association. In order to create the second association, the item must be anchored a second time. The two glyphs are used to represent this state. There is actually no limit to the number of anchors and associations that an item can be a part of.

Figure 13. The *Links* menu and the *LSM* menu.

The *LSM* menu and the *Links* menu in HyperEd provide the user with a set of operations to create and manage associations. For any of these operations to affect a text item, the item must first be marked. The set of linking operations includes:

In the *Links* menu: *Make / unmake persistent*
Follow link

In the *LSM* menu: *Attach / Detach anchor*
Attach / Detach link
Create / Delete / Open / Close / Lock /Unlock context
Turn anchor / link markers on / off

To mark a text item, the user must click the right and left mouse buttons simultaneously on top of the desired item. A red bullet will appear next to its top left corner. If the item is already persistent, anchored, or linked, the corresponding glyphs will be turned red. Otherwise these items will be unmarked and the glyphs will turn black. Items remain marked until explicitly unmarked by the user.

This section will first explain how each particular operation is performed. Next, we will take a look at the steps needed to create an association in detail.

Making a persistent selection. To make an item persistent, the user must first mark one or more items and then press the *Make Persistent* button in the *Links* menu. All marked glyphs will be replaced by persistent selection glyphs. The items remain marked or active as reflected by the red color of the anchor glyphs.

Attaching/detaching an anchor. To attach an anchor, the user needs to press the *Attach Anchor* button in the *LSM* menu. If the operation is successful, an anchor marker should replace the persistent selection glyph of all currently marked items, i.e. all red persistent selection glyphs.

To detach an anchor, the user needs to first mark the item to be detached. Next, the user needs to press the *Detach anchor* button on the *LSM* menu.

Attaching/detaching a link. All the user has to do to attach a link is to press the *Attach Link* button in the *LSM* menu. Links will be attached to all marked unlinked anchors. A text item may participate in several associations; therefore, text items may be anchored and linked several times.

To remove a link, the user needs to first make sure the desired linked items are marked. Next, the user needs to press the *Detach Link* button in the *LSM* menu.

Following a link. To traverse a link, the user needs to simultaneously click all three mouse buttons on top of a linked text item, i.e. an item with a link marker next to it. The LSM will bring up the document and components at the other end of the link. The LSM will probably start up some applications to display these documents. The *Follow Link* button in the *Edit* menu will perform the same operation.

3.6 Fonts and Colors

HyperEd provides a variety of fonts and colors. Text has the following properties: family, style, size, underline, and color. The font **family** is restricted to a small set of fonts. The font **style** can have three values: normal, bold, italic. Some families will not support all three different styles. Font **sizes** range from 8 pts to 100 pts, although not all sizes are available in all font families. All text can be **underlined** as well. The thickness of the line is under user control with values ranging from zero (no underline) to five points. All text items can have any **color** and color can be specified by name or by RGB values. A slider may be used to select the desired color.

Any time the user changes any of the items in the *Fonts...* dialog box, the selected font and color are displayed in the preview window. This allows the user to preview the text before applying the new values.

The fonts dialog box shown in Figure 14 allows the user to change the properties of a single item or the default properties for new items. If the user selects empty space before pressing the apply button, the default properties are changed. If the user selects an item, only the properties of that item are affected.

Figure 14. The *Fonts* dialog box.

3.7 Help

When the user presses this button, the SP3 Help Browser utility is called upon to display online help. Initially, this utility displays a table of contents where each entry is linked to a frame containing either more topics or an explanation on the selected topic.

Help documents are organized hierarchically to help the user find the desired topic. The title item of the frame is linked to the previous frame allowing the user to traverse back and forth in the help world. Link glyphs look like a hand and are placed next to the top left hand corner of the linked text item. They can be blue or maroon. Maroon glyphs indicate that the link was traversed earlier.

Figure 15. The *Help* window.

3.8 Quitting HyperEd

To exit HyperEd the user can either press the *Quit* button or select *quit* from the window menu. Both options will prompt the user to confirm the operation. However, the *Quit* button option will warn the user of an unsaved document and give him or her a chance to save it before exiting.

4. Future Work

HyperEd provides the user with a nice interface and some key functionality; however, the present version might be improved as outlined below.

- *Smaller granularity.* HyperEd operations currently handle a text item as a unit, i.e. editing, formatting, and linking operations cannot be applied to substrings of the text item. This would be a nice improvement since it would allow for copying and extracting parts of a text item, for multiple fonts and colors in a single text item, and for link embedding.
- *Allow for several documents to be open at the same time.* The current implementation of HyperEd allows for only one open document at any one time. HyperEd could be enhanced by allowing for multiple open documents and by providing a facility to quickly switch between them. This facility could be used to give users the choice to overlay the document at the other end of the link on top of the current one instead of running another instance of the editor to display it.
- *OM performance.* HyperEd is currently very slow when interacting with the LSM and the OM due to the low performance of *Postgres*. A future version will be implemented with *Illustra*, the improved commercial version of *Postgres* that promises to be much faster.
- *Printing facility.* The *Export* facility provides a means of printing a HyperEd document by converting it to an ASCII format before sending it to the printer. The drawback is that all formatting is lost. This problem could be solved by implementing a facility that would take a HyperEd document and convert it to a postscript format before sending it to the printer.

5. Conclusion

HyperEd is a spatial editor for creating and editing hypertext documents. It provides the user with most of the basic functionality available in today's editors, as well as a consistent interface with which the user can explore the flexibility of a true hypermedia system. However, HyperEd has barely scratched the surface.

As the first participating application to be fully integrated into SP3, HyperEd provides a flexible tool to test and refine the current implementation of this prototype and its underlying hypermedia model. This version of HyperEd has raised some questions about SP3 and we expect that refinements introduced from the application side will result in further improvements for the whole system.

Ongoing work will allow integration of other useful features such as formatting capabilities that go beyond a single document, basic drawing functionality, links of finer granularity, versioning of documents, and a fully integrated printing facility. Most importantly, HyperEd will help to further research in open hypermedia systems.

References

Akscyn, R., McCracken D., and Yoder, E. 1987. KMS: A distributed hypermedia system for managing knowledge in organizations. *Proceedings of the Hypertext '87 Conference*, (Chapel Hill, NC, November), pp. 1-20.

Laurel, B. 1991. *Computers as Theatre*. Addison-Wesley, Reading, MA.

Leggett, J. J., Schnase, J. L. 1994. Viewing Dexter with open eyes. Communications of the ACM, Vol. 37, No. 2, (February), pp. 76-86.

Sánchez, J. A., and Leggett, J. L. (forthcoming). HyperActive: Extending an open hypermedia architecture to support agency. ACM Transactions on Computer-Human Interaction.

Schnase, J. L., Leggett, J. J., Hicks, D. L., Nürnberg, P.J., Sánchez, J.A. 1993. HB1: Design and implementation of a hyperbase management system. Electronic Publishing: Origination, Dissemination and Design, Vol. 6, No. 1, (March), pp. 35-63.

Schnase, J. L. 1992. HB2: A hyperbase management system for open, distributed hypermedia system architectures. Dissertation, Department of Computer Science, Texas A&M University, College Station, TX, August.

Schnase, J. L., Leggett, J. J., Hicks, D. L., Nürnberg, P. J., and Sánchez, J. A., "Open architectures for integrated, hypermedia-based information systems", *Proceedings of the 27th Annual Hawaii International Conference on System Science (HICSS'94)*, pp. 386-395, Weilea, HI, January, 1994.

Schnase, J., Leggett, J., Hicks, D., Szabo, R. 1993. Semantic data modeling of hypermedia associations. ACM Transactions on Information Systems, Vol. 11, No. 1, pp. 27-50.

Shah, A., and Leggett, J. 1993. Image Manager. Department of Computer Science Technical Report No. TAMU-HRL 93-003, Texas A&M University, College Station, Texas.