DESIGNING TO FACILITATE BROWSING: A LOOK BACK AT THE HYPERTIES WORKSTATION BROWSER

BEN SHNEIDERMAN, CATHERINE PLAISANT, RODRIGO BOTAFOGO, DON HOPKINS and WILLIAM WEILAND
Human-Computer Interaction Laboratory, A.V. Williams Bldg., University of Maryland, College Park MD 20742, USA

Since browsing hypertext can present a formidable cognitive challenge, user interface design plays a major role in determining acceptability. In the Unix workstation version of Hyperties, a research-oriented prototype, we focussed on design features that facilitate browsing. We first give a general overview of Hyperties and its markup language. Customizable documents can be generated by the conditional text feature that enables dynamic and selective display of text and graphics. In addition we present:

— an innovative solution to link identification: pop-out graphical buttons of arbitrary shape.

— application of pie menus to permit low cognitive load actions that reduce the distraction of common actions, such as page turning or window selection.

— multiple window selection strategies that reduce clutter and housekeeping effort. We preferred piles-of-tiles, in which standard-sized windows were arranged in a consistent pattern on the display and actions could be done rapidly, allowing users to concentrate on the contents.

INTRODUCTION

Hypertext systems have become widespread and the number of applications is growing. In our projects we, as authors of hypertext documents, have become increasingly aware of the importance of giving proper visual cues to the users who browse the documents. We desire to create a heightened sense of visual engagement that not only catches the users’s attention, but rapidly guides them to the relevant interface actions and the document contents in an appealing manner. Secondly, we recognize the importance of reducing the cognitive load in the user interface so that users can concentrate on the document contents.

Background on Hyperties: We first give a general overview of Hyperties and its markup language. Customizable documents can be generated by the conditional text feature. It enables an author to specify the setting and testing of predicates so that only text relevant to the reader’s task appears on the screen. Then we present several ideas that were developed, implemented, and tested in the SUN version of Hyperties. These ideas are generic and could easily be implemented in most hypertext systems.

Link identification and selection: A key aspect of any hypertext system is the identification of link markers and the process of their
alternatives and observe user behaviour. A PC version of Hyperties has been commercially available from Cognetics Corp. (Princeton Junction, NJ) since 1987, and the Human-Computer Interaction Laboratory has implemented a workstation research version (on a SUN 4 using the NeWS window system) which provides a richer environment to explore advanced features. The versions are conceptually similar and databases are convertible. Some practical features are only implemented on the commercial PC version, while exploratory features (such as irregular shape selectable graphic or multiple windows) are only available on the SUN 4 workstation. This paper focuses on the workstation research version.

FIG. 1. Two articles are displayed. On the right side of the screen the title of the article is 'Participating organizations' which appears at the top left of the article, the page number (Page x of y) in the top right corner. On the left is displayed another article whose title is 'Hubble Space Telescope - Main view'. The bottom of the screen is used for the standard commands (e.g.: NEXT PAGE, BACK PAGE, INDEX, QUIT, etc. . .). When a user clicks once on the highlighted string 'Faint object camera', the corresponding definition appears on the four line area reserved in the low part of the screen. A double click brings up the Full Article on that topic, replacing the current article. When a link is selected with the right button (used to access all menus) a pie menu allows the selection of the window in which the new article should appear. See Section 3 for a description of the pie menu.
Using an encyclopedia metaphor, an Hyperties document consists of a set of articles, each covering a topic (Fig. 1). Each article has a title, an optional list of synonyms (by which the article may be referred to), a short description (definition or abstract) and a content (the article itself). It consists of one or several pages (one page per screen), the size of each article is only a function of the topic covered (down to consisting only of a definition). There is only one type of article, but some articles use a reserved title (such as 'Introduction', 'alphabetical index', 'table of topics' etc.) and are accessible directly by commands on the control panel. For example the 'table of topics' article is accessed by the TOPICS button (Fig. 1).

In Hyperties there are no error messages: only the valid links to existing articles are highlighted, and only the valid control commands are available on the control panel. There are no commands to remember, at any point the user only has to select one item among the selectable items available. Hyperties keeps track of the articles visited. A command 'Return to (name of the previous article)' is always available in the control panel and allows users to reverse their traversal. Aside from following embedded links, several commands allow the user to identify articles of interest. The INDEX command displays the alphabetical list of articles. The TOPICS command displays the table of topics article. The SEARCH gives access to a full text string search which constructs a list of articles containing the desired string(s).

Hyperties separates the authoring and browsing of a database. An article is represented by a collection of UNIX files, one of which serves as the primary description of the article's format and content: the 'storyboard'. The storyboard contains a human readable text description of an article in the database. Other 'associated files' contain graphical images (or other data, such as digitized sound, that will not normally be represented as human-readable text) and eventually a compiled version to allow rapid presentation. One special file, the master index, provides the association between the article files within the database.

The storyboard files are broken into five fields: title (the identifier of the article), synonyms (alternate identifiers by which the article may be referenced), description (a brief textual, graphical, etc. piece, usually used to summarize the document), content (a lengthy multimedia piece — the article proper), and notes (an optional collection of textual remarks maintained by the author). The content (and possibly the description) may contain references to other documents. Since the storyboard is a pure text document, it uses the Hyperties Markup Language (HML) to indicate page layout, font types, specify the inclusion of graphics, etc.

To Write as Clearly as You Think

Concentrate on simplifying sentence structures in business writing. It's the easiest way to say what is meant and to make sure the message gets across. Remember three basic rules:

(a) Keep sentences short. They should be no more than 17 to 20 words.
(b) Vary the length of sentences. The 17 or 20 word rule is the average. When sentences drone on at unvarying lengths, the reader's attention begins to wander.
(c) Vary the punctuation. Include plenty of commas, as well as a sprinkling of semicolons, to go with the necessary periods. Reason: Clarity of communication. Well-placed punctuation is a road map, leading the reader comfortably and accurately through the message.


FIG. 2. Formatting using macros and blocks
Fig. 2 shows many HML features that allow authors to create visually appealing pages while maintaining a consistent database format. First, many commands (.style, .size, .para, etc.) control the formatting of text and its position on the screen. Second the HML is a full programming language, allowing the creation of nested blocks, with local variables and macros. Although variables created inside a block are usually local, it is possible to create and make changes to global variables. Blocks are very useful in creating structural formatting, e.g. section headings are tagged, rather than having explicit commands for larger font, boldface, and line spacing. In Fig. 2 the macro .indent makes use of blocks to obtain the desired formatting. Blocks together with macros are very useful to create style files, a set of macro definitions that can be used in any article in the database. In Fig. 2, chap_name, sec_title and indent could have been defined in a style file. The use of any of them in any article would then guarantee consistency throughout the database.

Style files guarantee consistency, thereby reducing the burden for authors and browsers. Also thoughtful use of style files could give further cues to lessen disorientation, e.g. major articles would be in larger fonts, while secondary articles might be in smaller fonts.

Generating customizable documents

To achieve effective formatting we created the Hyperties Markup Language (HML) in 1988. It includes standard markup language features and conditional text to easily customize the document based on user actions. Although traditional Generalized Markup Languages, the Interleaf document preparation system, and scripting languages such as Hypertalk enable authors to specify conditional appearance of text, we feel that this feature should appear as a regular part of hypertext systems. Our empirical studies have shown that by limiting the amount of text on the screen, users can more rapidly perform their tasks.

Each selection of a highlighted link can set a variable. Predicates on these variables can be tested to decide on whether one string or another is displayed. This permits optional text to be suppressed, selection of national languages, Ted Nelson's 'stretch text', an outliner, and other features. Fig. 3 gives an example of conditional text for a software installation manual (in our example, 'Magic Paint'). This software package can be installed either on a Macintosh or an IBM PC. It provides for different configurations on each machine, different floppy disk sizes, one or two floppies, and hard disk options. When this storyboard is interpreted, users should already have indicated what configuration they possess, thereby setting the proper variables. Suppose the user has a Macintosh with two floppy drives, then the variables .macintosh and .dual_floppy would be set to true while the other variables would be set to false. This will cause the storyboard interpreter to generate a screen with information relevant to this user's task.

Conditional text also allows the creation of conditional links. Suppose one is writing a textbook where each chapter is followed by a quiz, the quiz should be taken only after the whole chapter is read, and the following chapter may not be read before taking the quiz. One structure that might be used is shown in Fig. 4. Each node of Chapter One will have a conditional link to the quiz and this link will only be active when all the other nodes of the chapter have already been read. Conditional text can enable authors to specify some of the browsing semantics.

Since the HML interpreter functions as the central controller of the Hyperties browser, the functionality of the interface may be changed by redefining control commands passed from the interface to the interpreter. When a button is activated, it generates an HML command string to be
interpreted. In one existing database, intended for use in an information
kiosk, it was desired to simplify the interface by eliminating the
presentation of brief definitions when a link was selected, jumping
directly to the intended article. This could be accomplished by simply
redefining the command that displays definitions to the one that displays
articles (via the macro facility).

FIG. 4. A schema illustrating the use of conditional links: Dotted links will only be active
when all articles of Chapter One have been read.

Since storyboards are text files, they can be created and edited in any
text editor as well as be manipulated by UNIX facilities (spelling
checkers, sort, grep, etc...). On our SUN version Unipress Emacs
provides a multiple windows, menus and programming environment
to author a database. Graphics tools are launched from Emacs to create or
edit the graphic components and target tools are available to mark the
shape of each selectable graphic element. The authoring tool checks the
links and verifies the syntax of the article markup. It also allows the
author to preview the database by easily following links from Emacs
buffer to buffer. Author and browser can also be run concurrently for
final editing.

Similarly, storyboards can be automatically generated. We are
exploring automatic importation techniques to rapidly create databases
from existing large set of documents. Our experience and others11,12 shows
that this can be achieved if the information source exhibits a regular
structure. An automatic importation tool has been added and successfully
applied by Cognetics in the PC version.

LINK IDENTIFICATION AND SELECTION
The use of purely textual articles in hypertext databases provides for a
very simple scheme for specifying links and referring to articles, which
Hyperties has adopted from its inception. As has been described above,
each article has a unique title, and possibly a number of synonyms. These
provide names by which articles are known to the system. In order to
construct a text link, an author needs merely to use one of these names in
the text of an article and surround it by tildes (~) to identify it as a link.
When the article is subsequently presented to a user, the links appear
embedded in the text, but presented in a different style (generally,
boldface, although this is not a fixed requirement). For example, in Fig. 1
the boldface string 'Scientific instruments' indicates the link to the article
whose title is 'Scientific instruments'. Selecting one of these highlighted
phrases causes the system to find the article named by the phrase, and
display its definition (and full content, if so desired). The transparency of
the link gives the user a good idea of the reference's nature and a sense
of continuity. It also leads to simplified authoring.

The incorporation of graphics presents a challenge in interface design.13
In the graphic domain, the notion of a link is less obvious than in the text
case. Are links represented by points on the image, or by extended
regions representing objects? There is no simple technique for indicating
visually the selectable elements (links) of a graphic that is acceptable in
all cases. The task of the author in creating graphics and graphical links
is significantly more complex than for pure text documents.

Implications of graphics in hypertext
Hyperties incorporates graphics while preserving the embedded menu
approach used for text-only documents. A displayed page can mix text
and graphics while allowing arbitrarily-shaped regions to be designated
as targets, which provide links to other articles. The addition of graphics
provides significant advantages.14 Information that is structured in the
form of charts, graphs, maps, and images may be explored with the same
facility as text. But the use of graphics in hypertext requires more work
on the part of the author to produce comprehensible documents. There is
no simple technique for emphasizing the targets that is acceptable in all
cases, and the author must laboriously link targets to their references
(they are not 'self-naming,' as in the text case). In the Sun version of
Hyperties rudimentary tools have been developed to simplify the author's
job of establishing graphical links between entries. These consist of
editors for designating arbitrary regions of an image using rectangles or
polygons, associating names with these regions (which are used by the
system to locate references), designating their appearance when
highlighted, as well as overall management facilities for keeping track of
graphics that have been produced.
Identifying selectable items

Several existing hypertext systems permit the browsing of mixed graphics and text, but none has carefully addressed the problems inherent in this more complex realm. Apple Computer’s HyperCard\(^{13}\) makes minimal distinction between textual and graphical elements of a database entry; anything can be selectable (linked to a reference) if the author so chooses, but it is left to the author to provide hints to the user about what is selectable. (However, it is possible to obtain a temporary display of selectable regions, highlighted by bounding rectangles, via a modifier-key combination.) Brown University, in its Intermedia system\(^{14}\), has, in a sense, eliminated selection of graphical and textual elements entirely; instead, all links are indicated by special icons, which the author may place at will. This scheme has the advantages of simplicity and clarity, but risks obscuring pertinent information in a welter of special symbols, and may introduce ambiguities of reference: for example, in a map of the United States, does a particular icon refer only to New York City, or to the entire state of New York? A slightly different approach is taken by Guide,\(^{17}\) which provides four different types of linkages; the presence of a link and its type are indicated by changes in the appearance of the screen cursor as it is moved about the display and encounters selectable objects.

An important problem is how to indicate to a user the selectable elements of a graphic. It is clear that some scheme is needed to do this — expecting a user to hunt after the targets in an image by trial and error is apt to cause frustration with the system and limit its use. Any scheme chosen must satisfy certain requirements: it must unambiguously identify the location and scope of the target, it must not itself interfere with comprehension of the image, and it must require little effort on the part of the user. Possible solutions include highlighting the targets by dynamic indicators, outlines, inverse video, fixed symbols, color, shading, and image manipulation.\(^{13,14}\)

The approach taken in Hyperties has been to allow arbitrarily-shaped regions of an image to define these link targets. Normally, these regions are not distinguished in any way (other than visual cues that an author may choose to provide); however, when the mouse cursor passes over a target region, it becomes highlighted. The highlighting scheme developed is novel, and has met with favorable comments from users: it is referred to as ‘pop-out,’ and consists of offsetting the highlighted object vertically and horizontally by a small amount, and placing a drop-shadow beneath. This gives the appearance of having the object pop out of the screen (Fig. 5a); in addition, the slight movement of the object makes it readily detectable to the eye. Because of the use of arbitrary regions, there is no ambiguity of reference; because highlighting only occurs in response to user demand, the image itself is not cluttered; and because the targets highlight automatically in response to mouse movements, the interface requires minimal effort.

One remaining problem is that it is not possible to identify the links in an image at a glance. However, certain user behaviours were noticed: when confronted with an image with hidden targets, they tend to sweep across the image until a target is highlighted (becomes designated), or try to select what they think might be a target until one is found. This suggests that the system could highlight all of the targets automatically (for a short time) whenever it appears that the user is searching for targets, as when sweeping the display, or clicking in non-target areas. This latter strategy has been implemented in the NeWS version of Hyperties — whenever a user attempts to select in a non-selectable region (like the background), all targets are revealed (Fig. 5b). This technique was found effective and generally very well received by users.

Description of the reference

Certain subtler issues also crop up in the graphics domain. Before following a reference, a user customarily wants some idea of the nature of the reference (is it an article or a diagram?) because this may bear on its usefulness. Examining a full reference is a somewhat disruptive
occurrence, in that it requires a shift of the user's attention from the context of the current entry. Certainly, entry type information could be incorporated in the entry's description, but it would be preferable to make this apparent from the display of the targets themselves. An authoring convention was adopted (though not enforced by the system) where textual references are made from textual targets (text labels in a graphic), and graphical references are made from graphical targets (elements of the actual graphics). In text, links to graphics are generally identified as such through naming (e.g., see Figure A . . . ). In addition, different forms of highlighting can be used for different types of targets — generally, links to text in a graphic are simply highlighted by inverse video, as are standard textual links.

**PIE MENUS TO PERMIT LOW COGNITIVE LOAD ACTION**

The Hyperties browser uses pie menus as accelerators, to make commonly used commands quickly and easily available. A pie menu is a type of pop up menu whose selections are laid out in a circle around the menu center. The menu pops up centered on the cursor, so that each selection is adjacent to the cursor but in a different direction (Figs. 1 and 6). A selection is made by moving the cursor in the direction of the desired selection, and clicking. Experienced pie menu users can make selections from familiar menus quickly and reliably without even having to look at the menu, because the menu selection depends on the direction between the two mouse clicks that invoke and select from the menu. The distance of cursor motion does not affect the selection, but the further away from the center the cursor is, the more precise the control of the selection is.

The browser has a control panel at the bottom of the screen, with buttons showing the names of available commands, to turn the page, return to the previous article, show the index, etc. When users are browsing a document by pointing and clicking on highlighted text links in the main contents window, they move the cursor down to the bottom of the screen to press buttons in the control panel, and back up to continue browsing. The permanent display of those controls is important for the novice and occasional users. On the other hand, pop up menus reduce the distraction of moving the cursor by making these commands available wherever the cursor currently is. This reduces perceptual and motor load. Pie menus are arranged with their items in easy to remember directions. For example the BACK page turning commands are to the left (and the NEXT page is to the right) (Fig. 6). This arrangement facilitates gestural input and encourages development of muscle memory.

Experienced users can make gestural selections from these menus so comfortably and rapidly that it is often unnecessary to display the menu. This is called 'mouse ahead display suppression', and its point is to reduce the perceptual distraction.

**WINDOW SELECTION STRATEGIES**

There are many solutions to the window selection problem for hypertext documents. The simplest solution is to have only one window and avoid any screen clutter or housekeeping effort related to multiple windows. This strategy was the basis for the IBM PC version of Hyperties and other systems. The simplicity is appealing both from visual clutter and from cognitive load perspectives, and this strategy works effectively with novice browsers. However as task complexity and user sophistication increase the desire to see multiple documents becomes greater. Several systems offer user managed multiple windows in which the users specify the windows' location and size (e.g. Notecards and Intermedia) in a cluttered desktop of partially overlapping windows. This two and a half dimensional world can be visually appealing and functional, but the additional perceptual, cognitive, and motor load can undermine the advantages. To overcome these problems we sought to make low cognitive load window selection strategies that reduced the number of decisions and operations made by users.

Our strategy was to allow fixed size windows arranged in piles-of-tiles. Fig. 1 shows a two window browser while Fig. 7 shows a four window browser, both with user control over the placement of each successive window as a tile on top of the current window. Automatic placement strategies such as replacing the least recently used or most recently used window were implemented and tested but proved to be less attractive. Other strategies included a sequential placement of
successive windows and panning strategies which moved windows to make space for the newest window. A cascading strategy in which the first window appeared in the upper left and then successive windows appeared slightly below and slightly to the right was also considered. Our preference for the non-overlapping strategies is based on the assumption that readers of these databases may be novice users and that the additional effort in manipulating overlapping windows would be strongly distracting from the central task of absorbing the contents of the articles.

Our empirical testing focussed on two versions of the four window browser: user controlled placement of piles of tiles vs. automatic panning in a Z-pattern (all selections were made on the lower right window, which moved to the lower left, to make place for the new window. The previous lower left window moved to the upper right, the upper right moved to the upper left, and the upper left moved off-screen to an invisible stack). Both versions enabled recall of previous windows. Our testing was done using 106 articles on Austria and Holocaust, using a version of Hyperties for the SUN workstation. The tasks included three simple fact questions that could be answered with a single article, e.g. How many nations attended the Evian conference in 1938? There were also three difficult questions that required comparisons of the contents of several articles, e.g. In which order where the three Social Democratic presidents elected?

The results of the study with 23 subjects showed a statistically significant preference rating for user controlled placement of the piles-of-tiles approach but no speed advantage. We believe that the subjects had inadequate time to develop effective strategies using the multiple windows and that longer experience might be necessary for the full benefits to be detectable.

The more recent NeWS version of Hyperties on the SUN workstation uses two large windows that partition the screen vertically. Each window can have links and users can decide whether to put the destination article on top of the current window or on the other window. The pie menus made it rapid and easy to permit such a selection. When users click on a selectable target a pie menu appears (Fig. 1) and allows users to specify in which window the destination article should be displayed (practically users merely click then move the mouse in direction of the desired window. This strategy is easy to explain to visitors and satisfying to use. An early pilot test with four subjects was run, but the appeal of this strategy is very strong and we have not conducted more rigorous usability tests.

In the author tool, we employ a more elaborate window strategy to manage the 15-25 articles that an author may want to keep close at hand. We assume that authors on the SUN/Hyperties will be quite knowledgeable in UNIX and Emacs and therefore would be eager to have a richer set of window management features, even if the perceptual, cognitive, and motor load were greater. Tab windows have their title bars extending to the right of the window, instead of at the top. When even 15 or 20 windows are open, the tabs may still all be visible for reference or selection, even though the contents of the windows are obscured. This is a convenient strategy for many authoring tasks, and it may be effective in other applications as well.

CONCLUSIONS
The documents written with Hyperties have been used by hundreds of users at demonstrations, and some components have been studied in controlled experiments. A full usability study with a realistic application to compare Hyperties/SUN documents with other electronic forms or paper is a desirable goal. We believe that the components are important contributions and encourage other developers to try them and refine them further.

We see further opportunities for improving the hypertext browsing environment with ideas such as multiple indexes to permit partitioning of large databases, graphic browsing capabilities (not a node-link diagram,
but a higher level representation), various bookmarks, and still larger, sharper, and faster displays. Improved search facilities would include flexible string searching, search for graphic images, search within a restricted distance from the current node, weighted search results, intersections/union of searches, and search by node-link structures. Our recent efforts have been to develop structural analysis software tools and metrics to provide authors with suitable feedback to better guide the document creation process. The future of hypertext will be brighter if the user interface for browsing can be made more attractive and effective.

ACKNOWLEDGEMENTS

We appreciate the support of NCR Corporation in providing funding for this research and SUN Microsystems in donating equipment.

REFERENCES
