Interactive Graphics Interfaces in Hypertext Systems

William J. Weiland
Ben Shneiderman*
Human-Computer Interaction Laboratory
Department of Computer Science
University of Maryland
College Park, MD 20742

February 23, 1989

Abstract

One of the key design aspects of hypertext systems is the rapid selection of items displayed on the screen in a direct manipulation manner. The user can select next or back page turning icons or buttons, or select another item that might jump to a remote destination. Authors and browsers are confronted with the problem of recognizing and selecting these choices in textual and graphic databases. This paper discusses the problems and offers a variety of solutions.

*Address correspondence to Ben Shneiderman

(To appear at ACM DC Technical Symposium Aug 24, 1989)
Introduction

Hypertext connotes a technique for organizing textual information in a complex, nonlinear way to facilitate the rapid exploration of large bodies of knowledge [Nelson 1974; Conklin 1987]. Conceptually, a hypertext database may be thought of as a directed graph, where each node of the graph is a (usually short) chunk of text, and where the edges of the graph connect each text chunk to other, related text chunks. An interface is provided to permit a user to view the text in such a database, traversing links as desired to explore new areas of interest as they arise, check background information, and so forth.

The Hyperties system (formerly TIES, [Shneiderman 1986, 1987, 1989], available from Cognetics Corp., Princeton Jct., NJ), developed at the University of Maryland, provides a particularly simple model of database interaction to the user. This is based on the idea of embedded menus -- i.e. links are represented by words or phrases ("targets") appearing in the text itself, and are associated with other entries ("references") in the database [Koved 1986]. When the user selects a target, the system provides a brief description of the referenced entry in a separate window; the user may choose to follow the reference, in which case the referenced (linked to) entry replaces the referent (linked from) entry on the display. A master index of entries may be obtained at any time, from which any entry may be accessed by selecting its title. In addition, a table of contents, path history, and string search facility are available.

The hypertext concept may be generalized to hypermedia, in which case the entries in the database are not restricted to pure text, but may also contain graphics, video images, and sound, among other possibilities. This article describes the design considerations for an extension of the Hyperties system. Originally, a text-only version of Hyperties for PC's was ported to the SUN workstation, to allow exploration of a wider range of interface techniques. The new, extended Hyperties, which
now runs in the NeWS operating environment, incorporates graphics while preserving the embedded menu approach of the older system. A displayed page can freely intermix text and graphics while allowing arbitrarily-shaped regions to be designated as targets, which provide links to other entries. The addition of graphics provides significant gains in information access. Information that is structured in the form of charts, graphs, maps, and images may be explored with the same facility as text. This has numerous applications: in the realm of computer-aided instruction, pictures may be used to summarize ideas, with more detailed information (in the form of text or pictures) readily accessible through the image itself; one can envision interactive diagnostic systems, with extensive pictorial information, based on hypertext; if authoring and browsing are integrated into one environment, the technique can be used to organize extensive, growing pictorial databases, as for medical, art, or satellite images.

However, the incorporation of graphics into hypertext systems also presents considerable problems in interface design, both for the user (the browsing system interface), and the database creator (the authoring system interface). The text domain allows relatively simple assumptions to be made about the nature of selectable items, and permits a simple, stereotypical approach to their presentation to the user (e.g. displaying them in a bold typeface or color). Much can be done to automate the author's job of establishing links between entries. In Hyperties, entries are known by specific names and synonyms. Whenever the author indicates a reference in an entry, the system needs merely to examine the global list of entry names and synonyms for the given text string, and generate a link. In the graphics domain, 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).

Another potentially important use of graphics, now under development for Hyperties, is as a "navigational aid" for the user. A difficulty inherent in all hypertext systems is that it can be difficult to conceptualize the form of the network that comprises the database, which may mean missing
important entries, or even getting lost, in conducting a search for specific information. Graphically displaying the database network can alleviate some of these difficulties; however, providing a comprehensible picture of a large, complex, heavily interlinked database is far from a solved problem. Several systems have attempted to implement this kind of facility, notably Intermedia [Yankelovich 1985], MCC's PlaneText [Conklin 1987], and gIBIS [Conklin 1988].

Design of browsing systems

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 [HyperCard 1987] 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 [Yankelovich 1985], 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 [Guide 1986], which provides three 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.

It appears, then, that an important problem to be addressed is that of indicating 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 (preferably no) volitional effort on the part of the user. A number of such schemes are proposed below; in essence, they all involve providing a visual emphasis to the selectable elements of a graphic, when activated by certain user actions (mouse movements or special keystrokes).

**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. One possibility is to require that textual references be made from textual targets (the labels in a graphic), and that graphical references be made from graphical targets (elements of the actual graphics), but this imposes excessive constraints on the layout of graphical and textual entries. Different forms of highlighting could be used for the different types of targets (see below).

**Searching in textual and graphic databases:** An intriguing possibility, but one beyond the scope of the current investigation, is that of search in a graphical database. In pure hypertext systems, string search is often employed as a means of speeding retrieval of pertinent information, but analogs to this technique in the graphics domain are lacking. The simplest approach is merely to search the database for graphics which contain particular components, on the basis of text strings supplied by the user: this is possible since targets in a graphic are associated with the names (or synonyms) of the entries they reference. This could be further extended by allowing an author to add keyword "tags" to entries. In this way, graphical entries in a database could be accessed quickly on the basis
of names of things which appear in them. More ambitious approaches involve simple shape matching through the graphics in a database, and even the application of image recognition techniques. The former is especially suited to situations where much of the graphics consists of "clip art," i.e. reusable, stock image components. In this case, an image component can be selected graphically (from an image, or perhaps a "catalog" of components), and searched for in other graphical entries — the search can look for exact bit-by-bit matches, or perhaps by classes of image components that have been defined to be "synonymous." Also possible is matching by rough similarities of appearance of the images sought [Christodoulakis 1986].

Design of authoring systems

Designing a satisfactory user interface for a graphics browser requires careful attention to the issues described above; however, none of these present insurmountable difficulties given the state of graphics technology. Unfortunately, the authors of graphical databases face one problem which no amount of careful design can completely alleviate: creating the necessary graphics, and designating the selectable targets, is far more work than writing the text for a textual database. In order that hypermedia systems will actually find significant use, it is important that the authoring problem be addressed thoughtfully.

In designing any extensive document, whether it be a book or a hypermedia database, it is desirable to maintain a consistent style throughout, because the stylistic conventions condition the user's expectations. In a graphics browsing system, not only should the "look" of the individual images be consistent, but the conventions for emphasizing selected targets, and the shape of those targets should be consistently applied. This means that a system requiring all images (and targets) to be specified as pure bitmaps (or an equivalent representation), which leaves the matter of style
completely open, imposes an enormous burden on the author to maintain whatever stylistic conventions have been adopted. Furthermore, specifying the targets in an image becomes a laborious task.

For this reason, it is deemed better to avoid thinking in terms of a simple text/graphics split, instead examining the specific requirements of the types of data to be illustrated pictorially. In this way, the author of a database entry can specify only the minimum information needed to convey what is intended, and leave much of the work of formatting and specifying the selectable targets to the authoring and/or browsing systems. To better illustrate this, much of the information in hypertext databases can be categorized as follows:

- Simple - e.g. function buttons, whose location and appearance are completely unspecified by the author.

- Lexical - e.g. text, where the author specifies only a sequence of symbols; placement and appearance are determined by the system, according to predetermined formatting rules.

- Topological - e.g. node-link diagrams, organization charts, etc., where the author specifies only the connections between objects, and leaves arrangement on the display to the system.

- Functional - e.g. graphs, histograms, time lines, where only numerical relationships need to be specified.

- Geometric - e.g. maps, which may be scaled by the system.
• Absolute - e.g. images, which are to be displayed at a fixed size.

Since some use of absolute image information is unavoidable, especially in the cases where existing pictures (photographs, paintings, etc.) are to be digitized, it would be useful to consider techniques for machine-aided specification of the targets in an image -- these can range from relatively simple ideas, like a graphics editor equipped with a large variety of general-purpose drawing and region-filling tools, to sophisticated image recognition techniques. Currently, Hyperties supports the specification of only lexical, geometric, and absolute information by the author.

Interface techniques

A typical interaction sequence with a graphical database entry might proceed as follows. An image is displayed, without any special indication as to its selectable targets. The user requests, by a keypress, that the targets be revealed, at which point all targets are highlighted in some way. The user moves the mouse cursor to the desired target, at which point it alone becomes highlighted as the designated target. A click of the mouse button selects the target and causes its reference's description to be shown. A rapid double click activates the target, whence its full reference is displayed.

This sequence suggests five (or perhaps, six) states of the interface that may need to be distinguished visually to the user.

1) display is the base state in which the entry is shown unmodified.

2) reveal: permits all selectable targets to be seen, allowing the user to determine available options for obtaining more information (these targets form the "embedded menu").
3) **designate:** where the user is "pointing to" a target, whether or not he actually intends to do anything with it.

4) **select:** means that the user has actively chosen the currently designated target, at which point the description of the entry associated with the target should be displayed.

5) **activate:** means that the user wants to obtain the full entry relating to that target.

6) **wait:** corresponds to the lag between the users activation of a target and the system's display of the new entry; in some implementations, this may be a significant interval, and indicating that the system is working, not waiting for input or "hung," may reduce user frustration.

Distinct from the issue of distinguishing interface states is the manner in which the interface transitions from state to state. These transitions occur in response to input actions from the user. The various types of input actions that can be used in controlling an interface have different characteristics, which must be considered in designing the interface. Movements of the mouse, although time-consuming over distance, require little conscious attention. Single and double clicks of one mouse button are effective and fast, requiring no mouse movement and no decision about which button (left, middle, or right) to use. On-screen "light-buttons" (i.e. labeled mousable areas on the display which work like function keys) are slow, requiring much extra mouse movement and conscious effort, and should be used sparingly. The use of a keyboard in conjunction with the mouse, is to be avoided, since switching between devices is time-consuming and often awkward; however, a keyboard alone is fast and positive when interface options can be discretized. The touchscreen provides an interesting alternative to these devices: it allows any part of the display to be reached with relative ease, and so lessens the need to optimize movements of the pointing device in designing the interface.
In general, a "light" interface, where the state transitions occur with little input action, is preferable to a "heavy" one where an extensive and tedious sequence of input is required for each transition, as long as this does not result in excessively frequent disruptive actions, like displaying a new entry by mistake. In a light interface, most of the state transitions would occur on the basis of mouse movements or touchscreen touches, and require little use of light-buttons, where a heavy interface would require extensive mouse clicking and light-button pressing. For example, a light version of the browser might go from display to reveal state as soon as the mouse cursor enters the image window, and designate a target as soon as the cursor touches it. Selecting the designated target might require a single click of the mouse button, activating it, a double click. In fact, on a computer system with sufficient performance, the target could be selected as soon as it is designated, so that descriptions would appear automatically whenever the mouse touched a target, and clicking would only be necessary to activate a target -- an even lighter interface! A heavy version of the same interface could require pressing a light-button (or holding down one of the mouse keys) to reveal the targets, clicking on a target to designate it, and then pressing light-buttons to select and then activate the designated target. The advantages of the former are speed and ease of use; the advantages of the latter are that interface options are clearly spelled-out on the display, and that unintentional state changes are unlikely. Certainly, it is possible to support multiple styles of interaction in one interface, to cater to the needs of different types of users.

One issue which has arisen is that requiring specific action to reveal the targets is cumbersome, but revealing them automatically whenever the mouse enters the image window may be annoying, since this can obscure detail in the image. However, typical user behavior when confronted with an image with hidden targets is to sweep across the image until a target is highlighted (becomes designated), or try what appear to be targets until one is found. This suggests that the interface could reveal 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.
Keyboard input, as was mentioned previously, can be very effective: some studies have shown, more effective than mouse input [Ewing 1986]. In order for this to be used, the notion of a "current designated target" becomes important. A single highlight moves about the display in response to cursor-key presses. This, incidentally, makes revelation somewhat less important, since all targets can be seen by pressing a cursor-key repeatedly. For text, the technique works very well, because there is an unmistakable left-to-right ordering of targets in the text stream, but this is not so clear in the graphics domain, where targets can potentially surround other targets. The problem here is to construct some type of ordering function for targets that doesn't defy user expectations. Most cases can likely be handled by defining a target's location to be the midpoint (or lower left corner, etc.) of its horizontal and vertical extent, and then ordering the targets top-to-bottom and left-to-right. The cases where a target's midpoint lies inside another target can be treated specially by taking the target with the greater topmost (if necessary, smaller leftmost) extent as the first. This cursor motion approach was implemented in the PC version of Hyperties, and proved convenient.

Display techniques

The preceding describes a set of possible states of the information display, some or all of which may need to be distinguished visually to a user. It is worth considering the types of techniques that may be used to accomplish this. There is no single, correct solution to the problem of providing visual emphasis to a graphical (or textual) entity on a display screen: each technique has advantages and drawbacks, and the choice may be greatly affected by the available hardware, and specific characteristics of the information being displayed.
Some of the possible classes of techniques are:

- **Fixed symbols** - "handles" which may be embedded in the image. Although this is very simple to implement, and requires little hardware support, these tend to obscure information in the image. Also, in some cases, they may be very difficult to discern.

- **Dynamic indicators** - most interactive systems make use of a cursor to permit the user to indicate what part of the display is being addressed. The appearance of this cursor can be manipulated to reflect hidden properties of information underneath. This technique is used by Guide, among others, to show that the cursor is positioned over a selectable object, and to indicate what type of information is associated with that object. This approach is simple to implement, but requires the user to "hunt" for information, and requires unusual attention to the cursor, which is normally taken for granted. Another possibility is to provide a fixed textual status line, or even a visual map, which continuously displays information about the state of the interface. This has the drawback of requiring the user to divide attention between two separate, but parallel, sources of information.

- **Inverse video** - the object or a surrounding rectangle are inverted bit-by-bit. Although this requires reasonably fast hardware (or low resolution), it requires no additional image memory, since it is non-destructive — inverting again restores the image. This technique is likely to be visually overwhelming, especially when applied to large objects.

- **Outlines** - a surrounding rectangle or conforming outline is drawn to highlight the object. The outline should actually be drawn in two different colors, or black and white, to ensure that it will be visible on any image. Inverting the outline is also workable, but may be difficult to recognize on a complex background. In any case, the technique will not work well for complex line images. It also requires
significant processing power if the outline is to conform to arbitrary-shaped objects.

- Color - this may serve as a highlighting mechanism by itself (i.e. the object is recolored in some easily distinguished way), or may be used to augment one of the other methods (fixed symbols or outlines). This technique can be quite striking, to the point of being overwhelming in some cases, but works well with images that are essentially monochrome, or at least, restricted in color use. It is not as well suited to arbitrary images, because an image may contain the highlighting color. Since color is supported by many personal computers, it is widely available, and does not require high resolution displays, as would monochrome shading, for example.

- Shading - this can take at least two forms: either half-toning on monochrome displays, or intensity shading on color (or at least, multiple-bit-per-pixel) displays. The latter technique is appealing for arbitrary full-color images, since it can be used to lighten or darken areas of an image without obscuring information. The former requires fairly high resolution to be effective, and sufficient processing power to handle the bit-blt (bit-block transfer) operations necessary to implement this.

- Image-manipulation - various visual effects are possible which rely on illusory three-dimensional changes to objects in the image. The "pop-out," which has been implemented in the Hyperties system at Maryland, causes the object to be redrawn slightly displaced vertically and horizontally, with an apparent shadow underneath (see Figure 1). This gives the appearance of having the object pop out of the screen; in addition, the slight movement of the object makes it readily detectable to the eye. This is especially effective where arbitrary-shaped objects are to be highlighted, as in maps, and does not clutter the image. Other, similar techniques can be envisioned, for example "bulging," in which pixels of the image are displaced radially from a central point, to give the appearance of bulging the image outwards (as through a distorting lens, perhaps). This class of techniques requires
both high resolution and significant processing power to be effective — it is currently in use on a monochrome SUN workstation. An interesting possibility would be to use stereoscopic displays, which would make this type of effect even more visually compelling. A recently announced home video-game system offers such stereoscopic displays, through the use of synchronized liquid-crystal glasses: this may herald the appearance of similar devices for personal computers, where true 3-D could open up many exciting possibilities for interface designers.

- Dynamic effects - many of the above techniques may be augmented through the use of animation effects. This may consist of crawling theater-marquee outlines, or simply alternating rapidly between the highlighted and unhighlighted states. The former is used in HyperCard to reveal targets, when requested by use of a keyboard sequence. This can be used to enhance the visibility of some of the subtler visual effects, since it adds the stimulus of apparent motion, which is readily detectable to the visual system.

Since the interface described above must distinguish several display states, these highlighting methods may be used in a progression, to indicate successive states. For example, the primary form of highlighting used to indicate designated targets might be pop-out. Then, selection of a designated target could cause further pop-out, or might just cause the target to be shaded. Similarly, activation could take either of these a further step, or might cause yet a third type of highlighting to occur (perhaps flashing). However, the highlighting used should form a reasonable progression, either in amount of a particular type, or overall visual impact.
Summary

This article has described a number of considerations in the design of author and browser interfaces for multi-media hypertext systems, and presented a wide (though not exhaustive) range of interface techniques. Several of these have been put to use in Hyperties, and have met with wide approval. However, much remains unexplored, and offers tremendous opportunities for the designers of such systems. In particular, graphical navigation and search, as well as more thorough support of the full range of information types described above, appear promising.
References

[Christodoulakis 1986]

[Conklin 1987]

[Conklin 1988]

[Ewing 1986]

[Guide 1986]

[HyperCard 1987]
HyperCard software for Macintosh, contact: Apple Computer, Inc., 20525 Mariani Avenue, Cupertino, CA 95014.

[Koved 1986]

[Nelson 1974]

[Shneiderman 1986]

[Shneiderman 1987]

[Shneiderman 1989]

[Yankelovich 1985]
A Hyperties graphic, with highlighted target

(Figure 1)