An Electronic Book: HyperCard and the Uses of Computers in Literature

Kurt A. Hills

AN ELECTRONIC BOOK: HYPERCARD AND THE USES OF COMPUTERS IN LITERATURE

KURT A. HILLS
BRIGHAM YOUNG UNIVERSITY

INTRODUCTION

HyperCard is a powerful new application for the Macintosh computer. It has been heralded as the biggest break-through in personal computers since the Macintosh itself. While HyperCard has both graphics and text manipulation capabilities, and can use digitized sound, it's most salient contribution to Linguistics and other branches of the Humanities is it's ability to link parts of a document to each other. This is done by way of the flexible and easy-to-learn HyperTalk programming language, which makes it possible to program the HyperCard information containers, called "stacks," "backgrounds," "cards," and "fields," and the information manipulators, which are called "buttons."

HyperCard is an outgrowth of the hypertext theory of information management and text retrieval. Hypertext, which Ted Nelson proposed in the late sixties, maintains that all parts of a document should be accessible through other parts of the same document or other related documents. Nelson's dream is of a computer system "that will provide computer users with instantaneous access to the world's books, magazines, movies, music, and all published documents."[1] HyperCard's contribution to this dream is increased public accessibility through the growing use of microcomputers, as well as it's graphics and sound capabilities, which add new dimension to the current forms of information management.

I chose to subtitle my paper "the uses of computers in literature" because I think this phrase describes a broad spectrum of the possible uses of HyperCard. It is an excellent interactive environment. It is such a "user friendly" programming environment that it would make an ideal "first language" for the computer novice. The teacher who had never touched a computer before could use the simple graphics and English-like HyperTalk to design and create lessons for her students, custom-made, as it were, for the particular needs of her classes. An example of this type of programming would be a stack that taught the constellations in various seasons, or the Indo-European language tree, or Arabic. These ideas, widely varied though they may be, are not only possible, but one is in use now at this University and one other has been developed.

John Sculley, the Chairman of Apple Computers, describes HyperCard and the Hypermedia concept in this way: "In broad terms, hypermedia is the delivery of information in forms that go beyond traditional list and database report methods. More specifically, it means you don't have to follow a predetermined organizational scheme

[1] Ditlea, Steve. "The Big Link," Omni. Sep 1987 v9

when searching for information. Instead, you branch instantly to related facts. The information is eternally cross-referenced, with fact linked to fact, linked to fact."[1]

Of course, this is the opinion of a person describing a product he is eager to sell. "Eternally cross-referenced" is a loaded term, and it needs clarification. There is no inherent "eternal" nature to HyperCard; a HyperCard stack and indeed the whole hypertext concept can only be as cross-referenced as the author makes it. As with any computer system the programmer is always in control of the output, try as he might to convince himself otherwise.

DESCRIPTION OF FEATURES

This is a sample card from the book stack. The Book Stack consists of a field with the book's text in it, as well as buttons that allow you to "drive" through the various cards and stacks. {See Figure 1} The basic structure of HyperCard is as follows: each document, or "stack" is made up of "cards." (HyperCard Stacks are analogous to conventional stacks of 3 x 5 index cards.) The "background" of a card is an consistent design feature shared by similar cards in a stack. Several cards can look exactly the same, since they are all based on the same background. This is the method used in the *Jude* stack. While it appears that only the text window is changing when we click on certain buttons, we are actually changing cards each time we click* the button. Each card has the same background, but the text window is different. After creating the initial background card, the author need merely fill in the fields.

Linking the cards in a HyperCard stack, and indeed other stacks to one another, is partially accomplished by way of "buttons." A button's program is called a script; buttons can have scripts of nearly limitless variety. Internal functions of HyperCard make impressive graphics displays such as zoom simple, and changes from card to card may be quite dramatic. Also, buttons can command the graphics functions of HyperCard as well as do text manipulation, calculations, import/export of outside documents, etc. In Figure 1, the three scripts of the Atlas, Index and Glossary buttons are given. Although each button does essentially the same task, that is, get a selection and match it with a corresponding referent in another stack, each button has to go about this a little differently.

The Atlas button gets the selection and places it in a global variable. A global variable is one that will keep the same value until something changes it. It will also maintain its existence even after the program in which it is found stops. HyperCard then switches stacks, going from the Book Stack to the Map Stack (here "Goodmap 1.1.") It then searches for a button with a name that matches the value of "target," the global variable. If "target" is empty (value = 0 or "") then it asks the user to "Please choose a town." The Index button is very similar to the Atlas button. It too gets a selection and places it in

---

[1] Goodman, Danny. The Complete HyperCard Handbook. Toronto: Bantam Books, 1987 p. xvii

* Click in this case means *one* push of the mouse button, not the normal "double click" of the Macintosh interface. HyperCard scripts are often triggered by an "on mouseUp" command. HyperCard waits, or "idles," until some type of user action occurs. (This could be a touched key, mouse movement, or "mouseUp.") HyperCard then responds to the action by either performing a script that the author has previously created or by doing nothing.

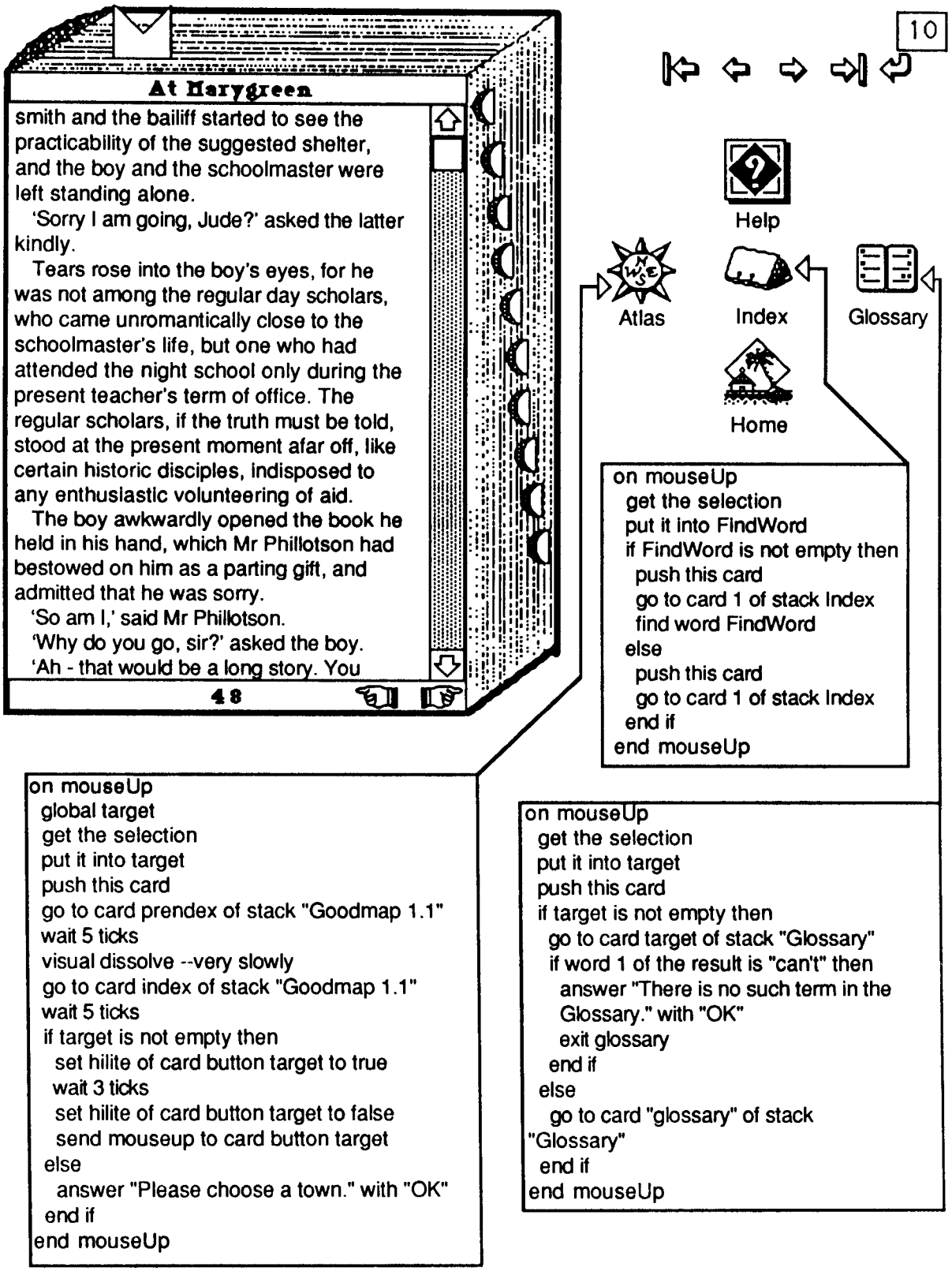


Figure 1. A sample card, including some buttons and their scripts.

"FindWord," a local variable. FindWord works like "target" does—it finds the first match of its contents. If there is no match, the default is the title card of the Index stack.

The Glossary button works in the described fashion, with one exception. If a word not in the Glossary is selected and the Glossary button is clicked, then the user is informed, "There is no such term in the Glossary."*

These are, of course, merely examples of HyperCard buttons. This paper is meant as an introduction to HyperCard, and as an attempt to generate more interest in it, especially from people in language-related fields. This is most easily and effectively done through examples.

Text in HyperCard is stored in "fields." A field can have a script like a button's. It can be smaller than one word or as large as the Macintosh screen, or larger if a scrolling field is used, and it can be use any type or size font contained in the system folder. However, this is a good place to point out one of HyperCard's shortcomings—in its present version it only allows one font and style per field. In other words, you can't use Plain Text and Underline in the same field, or Geneva and Times, or any other combination. This is extremely vexing, and has led to many different "kludge" approaches, such as overlays, where the plain text of one field is duplicated and underlined in another, and then placed over the original text. It is hoped that in future versions this problem will be dealt with.

Each text field has a scroll bar that allows more actual data to packed into each page, and each page of the stack corresponds roughly to the same page in the paper book.

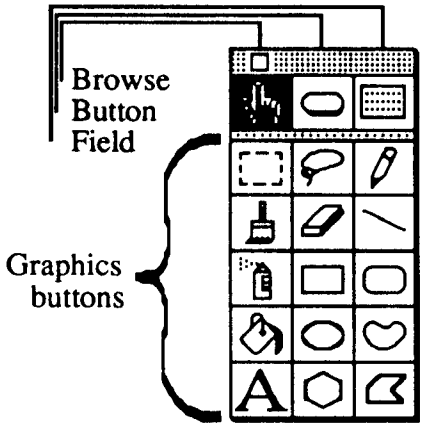


Figure 2. The HyperCard "tear-away" Tool Box

While the purpose of this paper is to highlight the linguistic possibilities of HyperCard, there is no reason to skip over graphics; indeed, for teaching in an interactive environment graphics are very useful, since they can help focus the attention of the student and make the

* If these buttons appear increasingly "sophisticated," then you are to be congratulated on your observational skill. Not only do they reflect the work of two different authors (myself and Charles D. Bush) they also show a gradual increase in understanding HyperTalk As we learned more about the language our scripts naturally became tighter and more productive.

subject exciting. HyperCard's graphics were designed by the application's creator, Bill Atkinson. Atkinson also created MacPaint, and took the opportunity provided by HyperCard to update and revitalize the nearly outdated MacPaint graphics system in his new creation. HyperCard's graphics are easily accessible from the "tear-away" Tool Box. {Figure 2} The graphics tools are very useful in creating cards, since it eliminates many time-consuming trips to SuperPaint or Cricket Draw some other graphics application and the subsequent importing of the outside documents. The book cover and the majority of the backgrounds in *Jude* were created within HyperCard; some of the more detailed work was copied with a scanner and imported.

ACTUAL STACKTUAL

The normal procedure in using one of the *Jude* buttons is: 1. select a word from the text, and 2. click the appropriate button. Since this paper does not include a computer, we will have to pretend. {See Figure 3a-d} We select a place name in the text by positioning the cursor over the word we intend to use and double clicking the mouse button. We then move the cursor to the Atlas button and single click there. This takes us to the Map Stack,where first we see a demonstration of the graphics abilities of HyperCard when teamed with a scanning digitizer.

After we see the confirmation of our selected placename, we then change cards to a map of a portion of Hardy's Wessex and the position on that map of the town we have selected. Since Wessex is based on the real English countryside, a click on the "Show England" button changes our view to the corresponding view of real England. (You should be imagining the names of the Wessex towns dissolving and being replaced by the names of the English towns. You can go back now, if you want.*) Clicking on the town name sends the user to an information card that contains the page numbers of the text on which our placename is found. Clicking the return button sends us back to the text.

The Index works similarly. {See figure 4} We select a word from the text and click on the Index button. We see a graphic display that tells us we have arrived, and then the word is found with the typical HyperCard "find box" outlining it. If we choose to find another word in the text we can select it and click on the text button; we could see another reference of the same word by getting the page number and clicking the text button, or we can just click the text button and return to our page.

The Glossary button takes us to a stack of cards containing words with different than normal or rare usages. Each card contains two fields that are immediately apparent, a "head word" field and a "text" field, which together give the general appearance of a 3 x 5 card.

---

* If you would like to see the real thing, please send a stamped, self-addressed envelope and a blank Macintosh disk to:

JUDE STACKS
Humanities Research Center
3060 JKHB
Brigham Young University
Provo, Utah 84602

(But don't wait too long.)

Step 1. Select a placename from the text and click on the Atlas button.

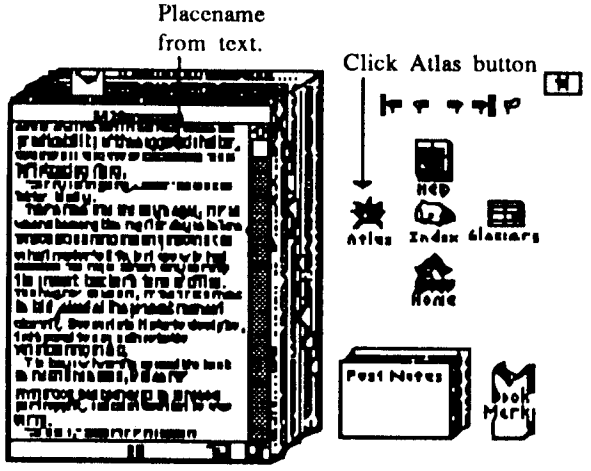


Figure 3a. A page from the Book stack

Step 2. See the Atlas IndexCard and confirm chosen town

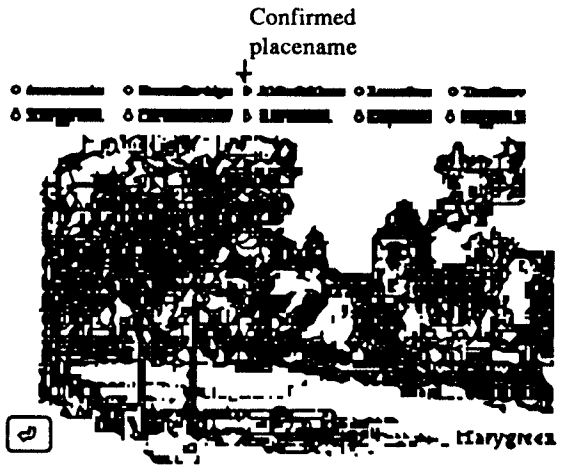


Figure 3b. The Index card.

Step 3. See town flash on map. If desired, click on town name for InfoCard.

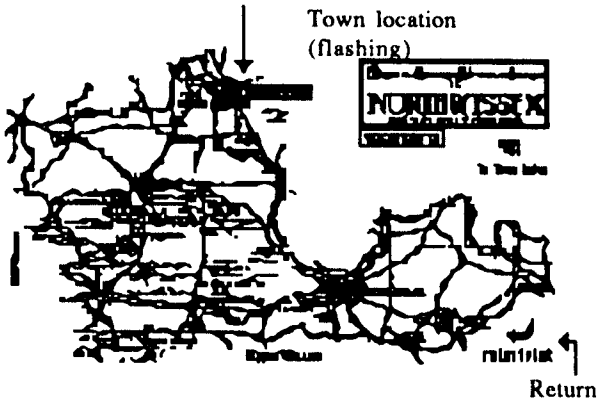
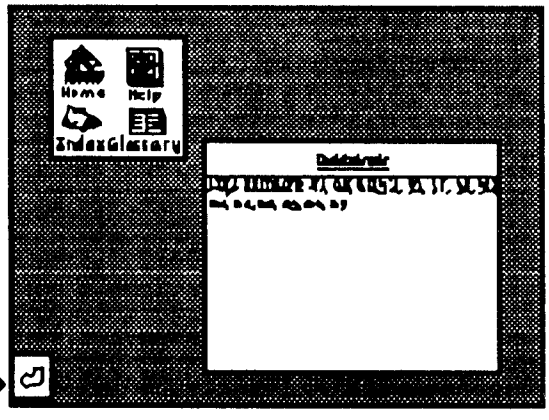


Figure 3c. The Map card.

Step 4. InfoCard



Figue 3d. The Info card.

When the Example button is clicked, an example of the word in context is shown. Clicking the button again hides the example.

The Glossary stack also contains another dimension of information, sound. The "Hear it!" button allows the user to experience the terms of certain cards aurally. The button is made

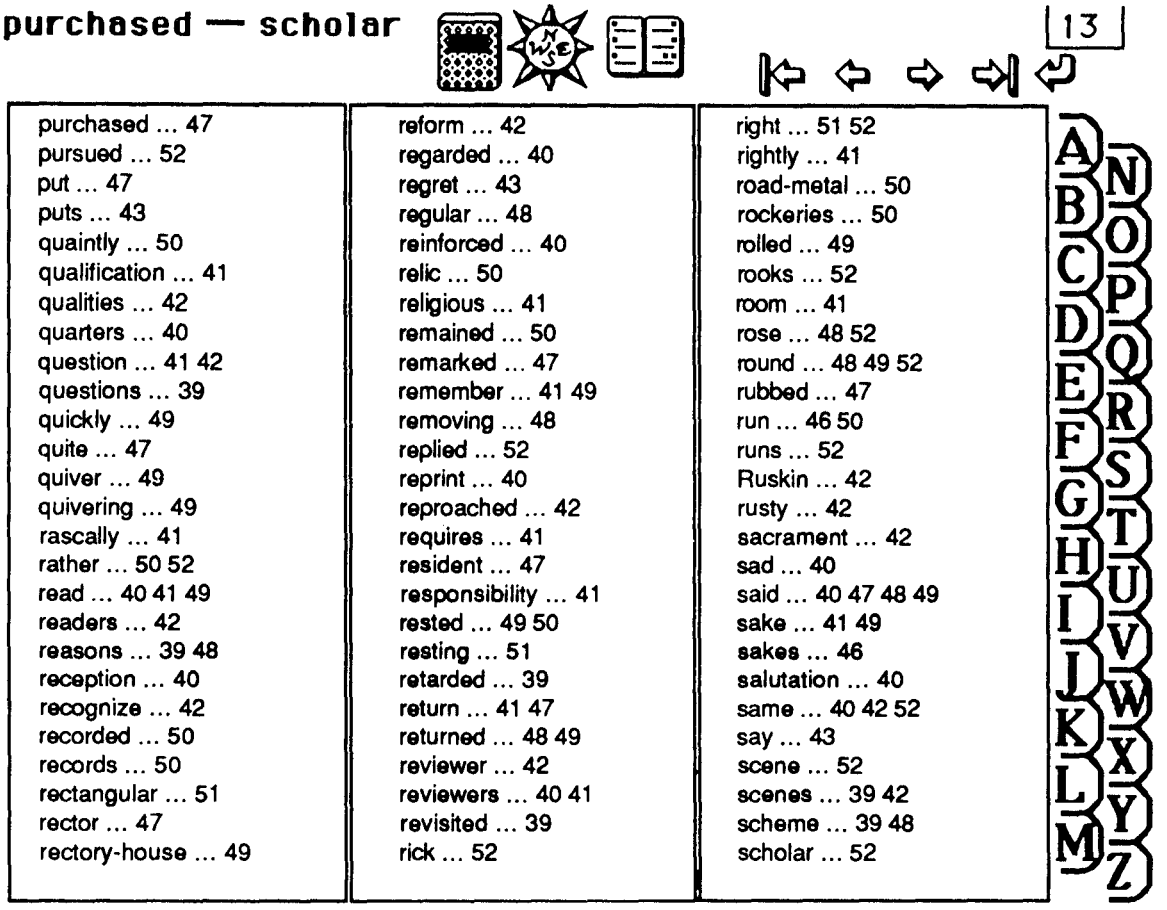


Figure 4. A typical index card

by sampling sound with a digitizer, and then inserting the sound into the stack with "SoundCapToRes," a HyperCard utility. A music generator standard in HyperCard also allows the author to compose music in at least the three mid-range octaves.

None of the *Jude* buttons require a selection to be made in order to work. If you click on the Index button without selecting text, you are free to browse through the Index Stack, picking references as you go. If you click on the Atlas button without a selection, you are prompted to choose a town. If you click on the Glossary button , you arrive at the first page of the glossary, where you are prompted to make a new card. The Glossary is user expandable, and it is always possible to write notes on existing cards.

CONCLUSION

In conclusion, please allow a brief editorial comment. Most people take programmers for granted. We rarely realize that someone somewhere lost hair over the creation of the word processing program we so merrily type away at. When we paint a poster we don't feel it important to remember that inside the computer there are bits and bytes whizzing around at a dizzying speed.

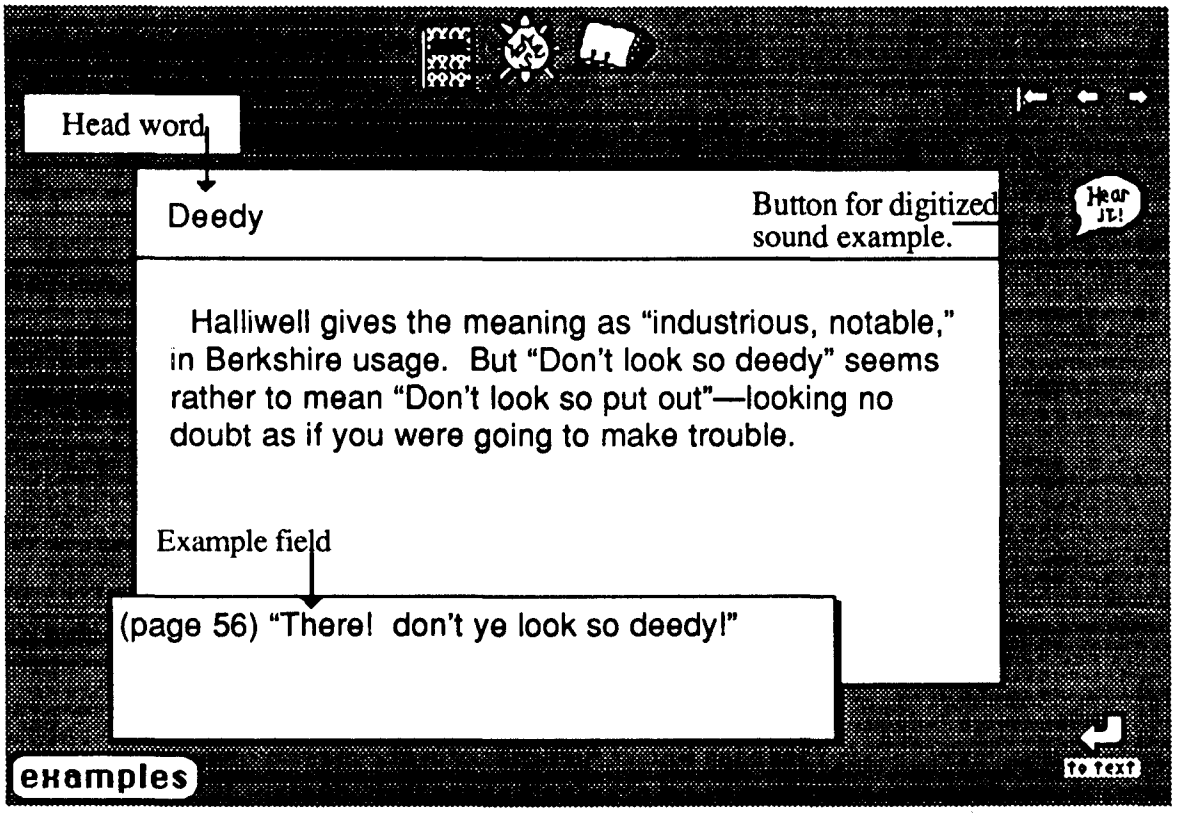


Figure 5. A card from the Glossary Stack

Fortunately, HyperCard exploits this attitude shamelessly. The joy of creating HyperCard stacks is that the majority of the behind-the-scenes "computing" is taken care of by the application itself, allowing us to be as imaginative as possible.

Also, since the program is free, it is probable that there will be many new resources and advances in stack design in the future.

My experience with *Jude* and HyperCard in general has shown me that with a little study and some creative effort an effective information management system can be created. As HyperCard becomes more familiar to us, we will expect to see it everywhere: in libraries, in the classroom, in the officeplace, in the home, *ad infinitum*. HyperCard is a powerful new force in the computer world.

ACKNOWLEDGEMENTS

Thanks, Chuck. Also, a special thanks to my wife Elaine, who put up with all the dumb linguistics lab stuff.