Chapter Five

Path Making Facility: Technical Details

Introduction

This chapter contains details about the design of the paths facility. The structures containing the path's information and the reasons behind design decisions are offered. Problems encountered during the design, and the implemented solutions, are discussed. The use of the History list is discussed and it's use in creating new paths is described. Problems concerning HyperCard's limitations and how they affect the paths facility are described. Finally, the problem of maintaining path integrity, and the lack of this feature in the paths facility, is discussed.

Path Structures

Each path is stored on one HyperCard card. The name of the card is the name of the path. There are three fields on each card — one contains the path information, one contains the meta-information and one contains the history list.

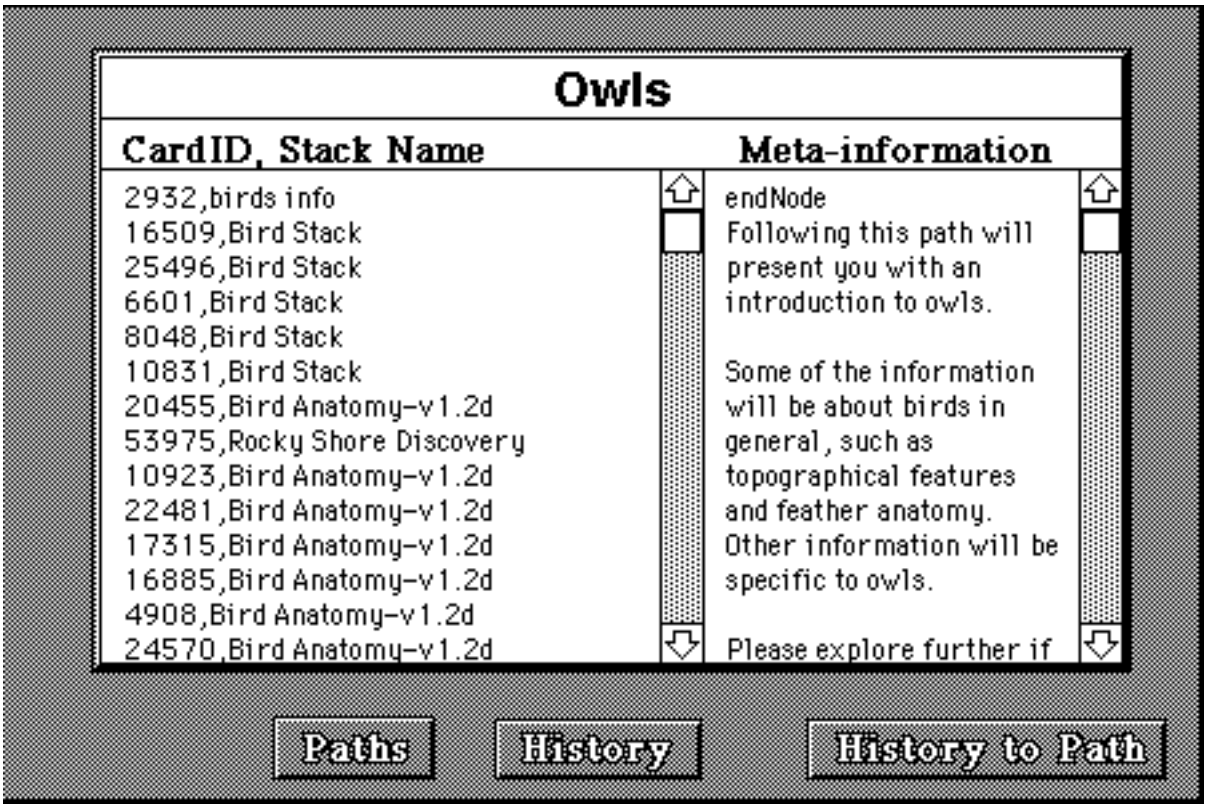

Figure 5.1 *Path card showing the path information and the meta-information*

Path Data

In the paths facility, the following data is required in order to navigate the system.

- *Card identification*
- *Stack name*

This information is stored on one line of the field. Each line represents one node and many lines of this information make up the path. The links can be from one card in a stack to any other card in any other stack. Each card will be uniquely identified by its ID number (which is produced by HyperCard itself)

within a stack. Using the name of the card was considered but this was rejected due to the fact that it would not be unique to a stack, and it also required the stack authors to name the cards which does not always happen. So in order to be unintrusive, the card identification number was used to uniquely identify a card in a stack. Using the card identification number is also much faster in searching than using the card name. Of course, the stack name is also required to uniquely identify the card within the system (all the stacks available). In a networked version, some location information would also need to be stored.

Figure 5.2 *Path card showing the History List.*

Meta-information Structure

The meta-information is stored in a separate field and is structured like this:

Delimiter
Meta-information for node 1
Delimiter
Meta-information for node 2
Delimiter
…

Meta-information for node n
Delimiter

Delimiters are on separate lines by themselves. The delimiter that has been used is '*EndNode*'. Using a whole word rather than a specific character was implemented because any character might be used in the meta-information itself, so using a whole word reduced the possibility of confusing the delimiter with the meta-information. Thus the meta-information may contain blank lines and any characters at all — the only restriction is that it may not contain the delimiter on a separate line by itself, although the delimiter may be used within a line and will then not be recognised as such.

History List

When a user is exploring, their path history is automatically stored. This history is saved only when the path is saved. Their path history is their linear trail through the system over time. So that includes nodes that were visited off the path as well as ones that were visited on it. The history list can be used to navigate also — any of the nodes on the list are selectable so a node can be re-visited just by selecting one of the lines in the list.

The data stored on each line of the history list is as follows:

Card identification

Stack name

Time of visit

Length of visit

As in the path field, the card identification number and the stack name are required in order to identify the card. The time of visit is stored for contextual purposes — perhaps to see when a particular card was last visited. The length of the visit is the amount of time in seconds that a particular card was viewed.

The length of visit might be used in a number of ways. Firstly as a contextual clue — it can be seen whether or not a card was examined in detail and perhaps gain a clue as to its interest level. Secondly, this length of visit data could be used as a criterion in creating a new path from the history data. As a basic assumption, it could be said that the length of time that a card is attended to corresponds in some way to its interest level. Thus, one quick and easy way of

creating a new path would be to filter the history list using the length of visit data as the criterion.

Using the history list to quickly create a path has been implemented. On each path card there is the path field, the meta-information field and the history field. There is also a button called '*History to Path*'. Clicking on this button will enable a new path card to be created using the current card's history list as a base.

Creating a New Path from the History List

To create a new path from the History List, select the '*History to Path*' button on the path card whose history list is going to be used. The threshold value will then be asked for. The *threshold value* is the cutoff value (in seconds) for adding a card from the history list to the new path. With each history is the length of time that a particular card has been visited and if this is greater than or equal to the Threshold value then it will be added to the new path.

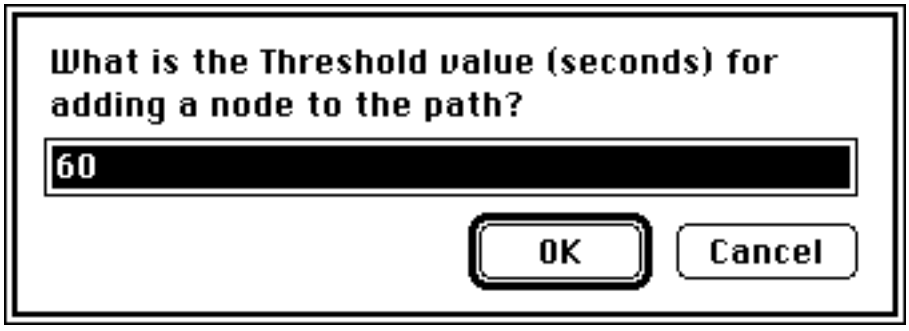

Figure 5.3 *Enter the threshold value dialog*

When all the cards on the history list have been examined, if some of them have been added to the new path, then a name will be required for the new path. A name should be entered describing the new path as in Figure 4.5 (or it can be named later — it is the name of the card), and a new card will be created with the path field filled with the cards satisfying the threshold criterion. The meta-information field will also be ready to accept new meta-information.

Problems Encountered & Solutions

While it would have been possible to store all the information on one field, it is a better breakdown by functionality to store the meta-information separate from the path information. The meta-information, while important, is a separate entity from the path itself. Having it separate also assists in lessening the effects of HyperCard's limitations. More specifically, HyperCard limits the amount of text in a field to 30000 characters. So the length of a path, as well as the amount of meta-information, is limited by this. The meta-information is more likely to come up against this limitation.

The history list might also come up against this limitation. If a user interacts with the system for a long time and covers many nodes then the history list will get very large. For example, assume each node in the history list takes up 50 characters:

card id	— 5 chars
stack name	— 25 chars
time	— 10 chars (max.)
length of visit	— 10 chars (max.)

This means that a maximum of 640 nodes may be added to the history list. Of course this is highly dependent on the stacks that are visited — if the average stack name was only 15 characters long then the number of nodes that could be visited before reaching the limit would be 800. Another way of reducing the size of the history list would be not to store the time of the visit. This does not seem to be particularly helpful unless a user comes back at a later time and can see when a node was last accessed [Utting, 1989a]. Perhaps this information should be stored in the path itself and could then be displayed on the path palette.

Storing the meta-information separate from the path information does complicate processing somewhat, especially when modifying a path, as it results in two areas that need to be searched before modification can occur. In a different implementation the information for each node might be stored together to ease processing and make it more functional. By storing all the information for a node on a path together, modification of the path would become easier. This would make a graphical tool to manipulate the path simpler to develop. It

would also enable the extension of the path tool to encompass more types of documents rather than just HyperCard stacks and cards. Conceivably the path tool could be made generic and different types of documents (for example, graphics, movies, text, sounds) could be linked. Then each document could be opened with its own application with the meta-information being available as a side-note in a separate window.

Integrity

There is currently no control over the deletion of cards and stacks or the movement of stacks to different locations. There is no way of ensuring that a node in a path is actually there. If a stack is moved to a folder that is not in the HyperCard search path then the stack will not be found and the user must locate it in the system. Similarly, if a card is deleted then all the paths that it belongs to will not be updated to reflect the change. What happens is that when the non-existent card is the current card, the system will report that the particular card does not exist any more. This can be a problem for novice users if they are not familiar with the Macintosh file system. If they are asked to find a stack then they may not know what to do. This needs some sort of attention and probably would be best addressed at the system level in a similar manner to the IRIS Hypermedia Services. That is where the system itself provides basic support for nodes and linking and manages them relatively seamlessly, so that the integrity of the system components is maintained without additional effort being required from users of the system.