A Virtual Environment Based File Management System: Preliminary Work

Frances L. Van Scoy
West Virginia University, Morgantown, West Virginia 26506-6845, USA

Abstract. The preliminary design of a virtual environment based file management system is described.

1. Background

Current virtual environments model real or imagined scenes at various scales.

It is the premise of this paper that VE can also be used in more abstract settings, in particular in a file management system.

1.1. Pre-Virtual Environment 3-Dimensional Models of Linear Systems

Long before virtual reality, people used 3-dimensional schemes for visualizing abstractions which are essentially linear in nature. For example, the traditional physical card catalog in a library was essentially a linear alphabetical list of records arranged spatially in drawers, cabinets, and rows of cabinets for efficient access.

The home of William Faulkner in Oxford, Mississippi, has been preserved as a museum by the University of Mississippi. Visitors to his home can look into his study in which he wrote much of the novel *The Fable.* Evidence of some of the development of this novel remains on the walls of the room. The novel is organized chronologically over the days of one week. Faulkner outlined the book using grease pencil and regular pencil on the walls of his study. A visitor can imagine Faulkner pacing around the room as he planned the book, pausing now and then to add a note or crossing the room to read an earlier point.

When I was an undergraduate student at Michigan State University in the 1960s, my dorm room had bunk beds. My bunk was the lower one, which gave me a view of the underside of my roommate's mattress and the springs supporting her mattress. These springs each ended in a small hook which curled up towards the mattress. I developed term papers by writing my notes on 3-hole punched notebook paper and hanging the pages from the springs and then organized each paper by moving pages from hook to hook. The general order of sections was maintained by an orderly arrangement of pages on hooks near the foot of the bed, while the parts I was currently working with (the text of the section, pages of rough drawings, notes from references) were suspended around my head within easy reach.

If physical 3-D representations helped in organizing novels and papers before computer visualization systems, then surely modern virtual environments can be used in even better ways.

1.2. The Need for a New File Management System

This paper describes preliminary design work for a virtual environments-based file management system.

What is the need? Storage devices have increased dramatically in size in recent years. Ten years ago a 10 MB hard drive was considered to be large. Today an 8 GB drive is not uncommon. The resulting task of managing thousands (or hundreds of thousands) of files in a simple hierarchical system has become difficult.

For example, a professor might have a subdirectory for each supervised graduate students in which to file documents related to their research and degree progress. There might also be a different subdirectory for papers written by the professor. When co-authoring a paper with a students, in which subdirectory should the professor file it? One solution is to construct a spreadsheet for each kind of file, such as "papers." One row of

the spreadsheet includes attributes of one paper, including title, general topic ("data parallel gossiping" vs. "multiple message broadcasting" vs. "sandpiles"), co-authors, nature of paper (journal article, conference paper, extended abstract, abstract, technical report, tutorial), status (submitted, rejected, undergoing revision, accepted, published), place submitted, year written, and the path and file name of the file. Essentially this approach imposes a relational data base structure on top of the hierarchical directory structure of most contemporary operating systems, although the actual file retrieval is done manually.

The system described in this paper will automate the process somewhat and will use large display devices to assist in navigating the more complex directory structure by showing multiple views of the structure simultaneously.

The primary goals of the system are:

(1) ease of use by the user

(2) implementation of multiple paradigms for viewing logical directory structure

1.3. Context of Project

This project fits within a larger project in its early stage of designing a new VE-based user interface.

We assume that in the future many researchers will have offices that are small CAVE-like systems. Such a system will differ from current systems in several ways. Because the researcher is likely to use the space for 8 to 12 hours each day, we assume the need for a table or desk (and chair) and for a lighter weight (or desktop based) pointing device instead of the typical wand. Because the resolution of current projection-based systems is not high, on the desk will be a high quality monitor which can display either the entire scene (or one wall of it) or a zoomed in version of one part of the scene (for viewing detail).

With such a system as the typical working environment for researchers, the user interface must change from our familiar one of windows, icons, and mouse to something new, with this process of change described by Tom DeFanti as "defenestration," or the removal of windows. [1]

The remainder of this paper assumes the existence of such a system, although the initial prototype is being developed on a Silicon Graphics Octane for demonstration on an ImmersaDesk. Although in the paper we refer to the left, center, and right buttons of a mouse, the system will likely use a somewhat different desk-based pointing device, to be described elsewhere.

2. Terminology

We begin with some definitions.

File and *directory* have their generally accepted meanings.

Each file has associated with it a set of *keys*. Each key has an *attribute* value. A *family* of files consists of those files with the same keys.

We define a *display paradigm* to be the way in which a particular system (e.g., Unix operating system, Smalltalk browser, Macintosh OS) displays the contents of a directory or collection of classes. In this paper we shall take a few liberties in extending these display paradigms.

A *display* is a particular instance of a display paradigm which shows all or part of a family of files and the logical relationships among theses files.

3. Finding a File

The way in which a user finds a file in our system depends on which display paradigm is in use.

3.1. Graphical Unix display paradigm

In the graphical Unix display paradigm we draw a tree which represents part of the directory structure. Our extension has three kinds of nodes in the tree: family, key, and attribute. The root of the tree is the only family node. It indicates the family of files under

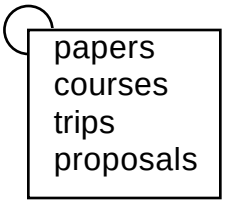

Figure 1
Choice of Family of Files

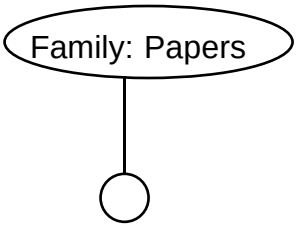

Figure 2
Tree with Family Label for Root

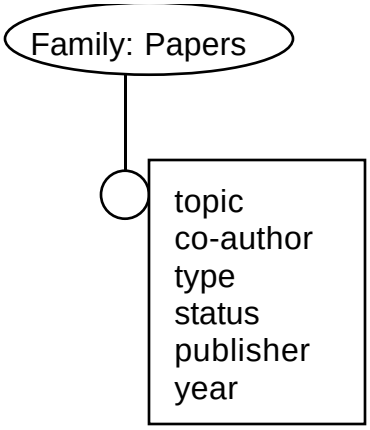

Figure 3
Choice of Key

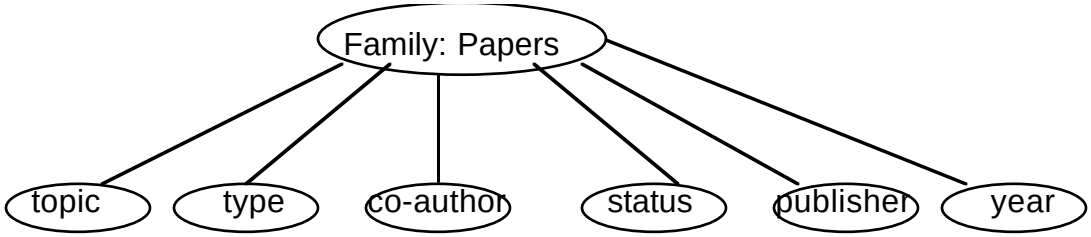

Figure 4
Tree with First Generation of Key Nodes

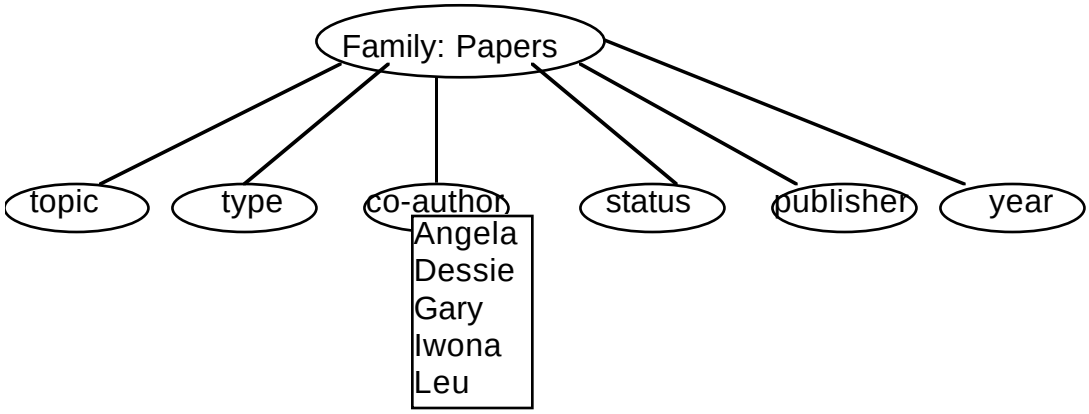

Figure 5
Tree with Popup Menu at Key Node Showing Attributes

consideration. The children of the root are key nodes, with labels corresponding to the keys of files in the indicated family. The grandchildren of the root are attribute nodes, with labels corresponding to attributes. Key and attribute nodes alternate by generation throughout the remainder of the tree.

We begin with the root of the file system. An empty node representing the root is displayed in the window. Clicking on the left mouse button causes the display of a pop-up menu containing a list of family names, as shown in Figure 1. When the user chooses one of these keys, the label of the empty root node changes to the name of the family. A child node of the root is then displayed along with an edge from the root node to that child, as shown in Figure 2. Moving the cursor to that child node and then clicking on the left mouse button causes the display of a pop-up menu containing a list of key names, as shown in Figure 3. When the user chooses one of these keys, the label of the empty current child node changes to the name of the key. If there are n keys then n-1 sibling nodes appear, each with the name of one of the other keys, with about half of these nodes displayed on either side of the original child node of the root, as shown in Figure 4. Similarly, clicking on the child node with the left mouse button causes a popup menu with all currently

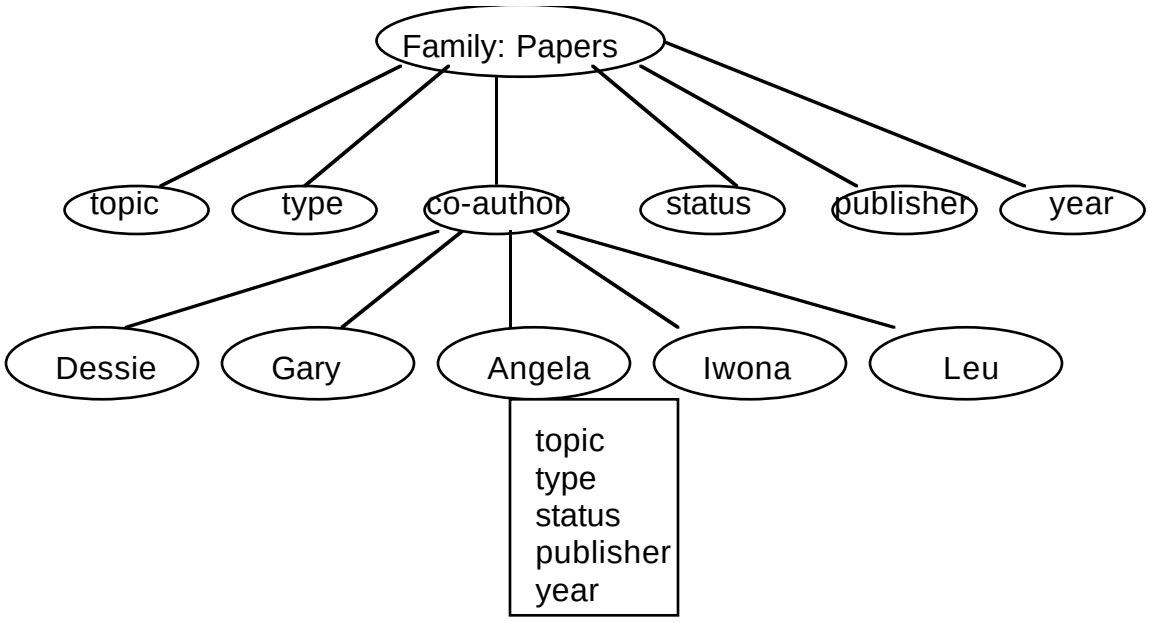

Figure 6
Tree with One Generation of Key Nodes and One Generation of Attribute Nodes

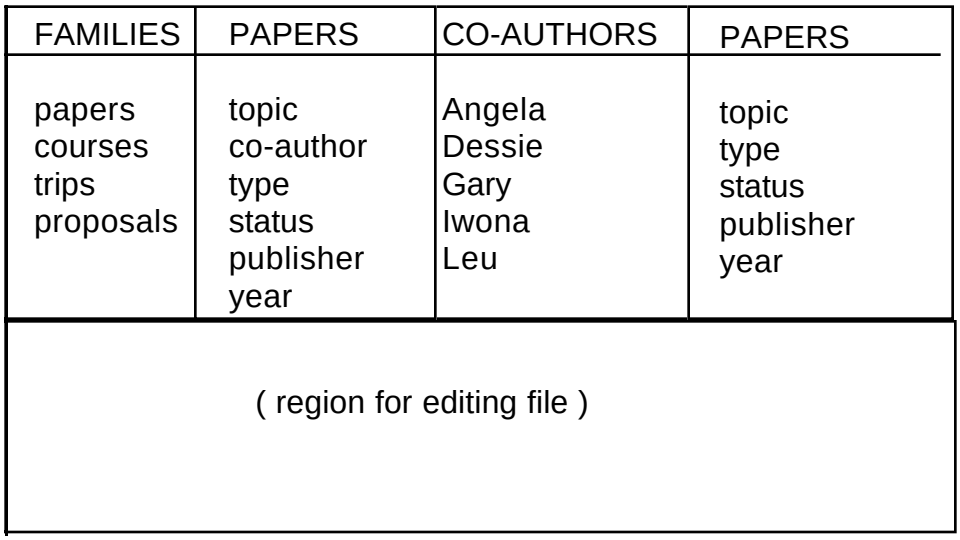

Figure 7
Smalltalk Display Paradigm

possible attribute values of that key to be displayed, as shown in Figure 5. The tree is built generation by generation in a similar manner, except each set of sibling key nodes has one less node than the set of sibling key nodes two generations above it, as shown in Figure 6. That is, the kth generation of key nodes has n+1-k nodes. Also the popup menu associated with a particular key node is context sensitive and shows only attribute values for which, based on the other attributes on the path from the root, there is at least one file in the family.

The right mouse button is used to edit the appearance of the tree. Clicking on the right mouse button causes the display of a popup menu which allows increasing or decreasing the number of nodes displayed. Commands include: show siblings, show cousins, hide siblings, hide cousins. Other commands allow for toggling between showing both the kind and attribute generations or only the attribute generations.

The center mouse button is used for rearranging nodes at a particular level by a "drag and drop" technique.

There are some additional minor points of interest. For example the logical file structure is potentially a directed acyclic graph because of (1) the possibility of multiple co-authors and (2) the capability of expanding more than one node in a given generation. In the latter case, the labels on one path from the root to a leaf are a permutation of the labels

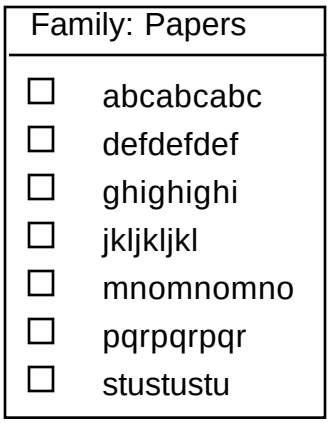

Figure 8
Macintosh Display Paradigm
Showing All Files in Family

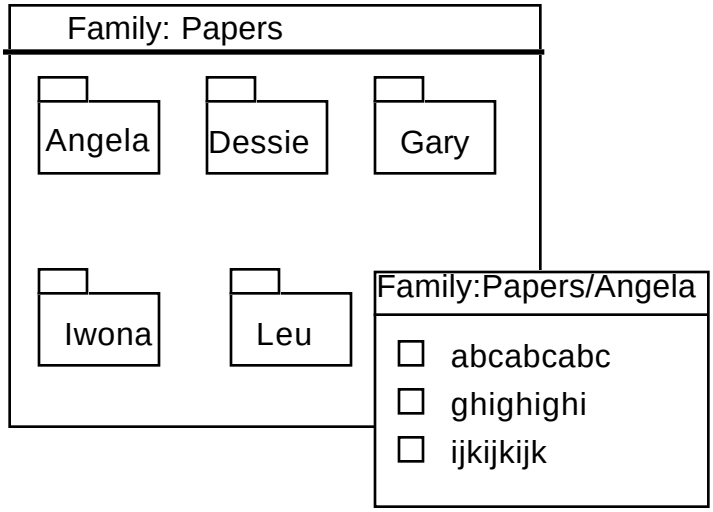

Figure 9
The Result of Choosing Key = Papers
and Attribute = Anglea

on another path from the root to a leaf. In each case we have multiple leaf nodes with the same label, and when the mouse is over one such node or one of the nodes is selected, all nodes with that label are highlighted.

3.2. Smalltalk display paradigm

In the Smalltalk paradigm, the basic display is that of a window divided into a set of scrolling frames directly above one larger frame. The user moves through the upper frames from left to right, each time making a selection from the scrolling list. When a specific file has been identified the appropriate application which created the file or can edit it is called, and the file is displayed in the frame at the bottom of the window and can be edited there. Figure 7 shows a snapshot of this paradigm.

This use of a classical window display from the 1970s may seem to contradict the statement made earlier in this paper that we are moving towards a defenestrated environment, an environment without windows. However, some of the power of this new VE-based file management system is due to the ability to link together multiple displays of the same file family (or related file families).

3.3. Macintosh display paradigm

In the Macintosh display paradigm all files in a family are displayed initially as if they were all in a single folder for that family. Clicking on the (left) mouse button displays a popup menu with the names all keys belonging to that family, as shown in Figure 8. Choosing a key causes a sub menu to be displayed with all valid argument values for that key. Choosing an argument value causes
(1) a set of folders, one for each argument value of the key, to be displayed in the family window,
(2) a copy of each file which has a matching attribute to be (logically) placed in the appropriate folder, and
(3) the window for the selected attribute to be opened. (If the selected key is "co-authors" and if one paper has multiple co-authors a logical copy of that paper is placed in the folder for each co-author.) This is shown in Figure 9.

Within any folder a user is allowed to color file icons, to connect icons with lines (with zero, one, or two arrowheads) and to draw rectangles around groups of icons. These lines and groupings indicate user specified relationships among files and persist until deleted no matter how the icons themselves are moved within the window. Both the lines and rectangles can be given user-defined labels. An example of this is shown in Figure 10. This feature has potential for allowing users to define a hierarchy of files with "multiple inheritance" of properties and relationships.

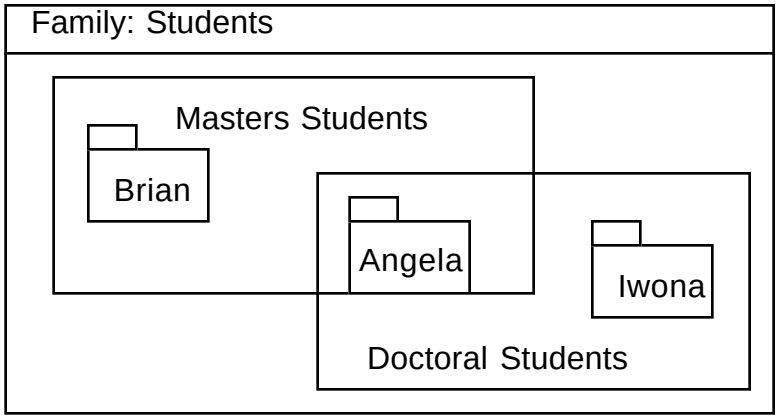

Figure 10
User-Defined Relationships in the Macintosh Display Paradigm

3.4. Manipulating displays

Some of the power of this file management system comes from the ability of the user
(1) to examine multiple displays of the same family simultaneously, with visual cues to identify the same file in multiple designs;
(2) to save properties of a display for use in other displays.
Here is where the interrelationship between the 3-d projection and the high resolution monitor is important. From the 3-d system the user can get multiple overviews of where related belong logically in the file family and then zoom in for a closer view on the monitor.

4. Placing a File into the System

The previous section shows in some detail how the various display paradigms are extended and used in our system. While this may be useful for viewing and navigating through a large collection of files, the necessary associating of attributes with files will not happen if this is too burdensome.

The user is prompted for attribute values when a new file is saved for the first time. The Save (as well as the Save As) dialog box contains a popup menu which lists all families belonging to this user. (A button for creating a new family is also provided.) By default a new file is placed in the General family, a system-defined family with only one key: file name. Once the user has selected a family, a series of scrolling windows, similar to those in Prieto-Diaz's faceted scheme. Each window corresponds to a key belonging to that family and contains a scrolling list of all attribute values assigned to that key for any file in the current family as well as a "new attribute" value for inserting a new value. If a user does not choose an attribute for a particular key, a default value may be assigned (depending on how key is defined in that family) or the user may be reminded (but not required) to choose an attribute when clicking on the Save button in the dialog box. Attribute values for a file may be changed at any time through the Save As command.

5. Implementation of the Prototype

During the academic year 1998-1999 we anticipating implementing a prototype version of the system whose design is described in this paper.

References

[1] T. DeFanti, D. Sandin, and M. Brown, The Coming Defenestration: Immersive Environments Without Windows, *IEEE Multimedia,* **3** (1996) 6-9.