Browsing Data in 3D: An Immersive First Person File Browser

GEOFFREY ULMAN
Rose-Hulman Institute of Technology

ADVISOR: DR. J.P. MELLOR
Rose-Hulman Institute of Technology

Draft: 3-23-2005

Abstract

As the data typical computer users receive and store daily increases, traditional hierarchical file browsers are becoming less and less useful in helping organize and retrieve that data. People do not have the time to create sufficiently "deep" hierarchies which accurately reflect the semantic structure of their data. More fundamentally, file hierarchies impose a single organization on the data, meaning interaction with the data is based on how the data was stored, not which properties pertain to the current task. Numerous solutions to this problem have been proposed, ranging from complicated search and indexing schemes (Google Desktop Search) to Microsoft Longhorn's WinFS file system and others. This paper, however, focuses on addressing the question "where is my data?" by fundamentally changing the file browser metaphor so that interaction with computer data takes place in a 3-D virtual space which simulates an actual physical desk/office. First, current similar solutions using 3-D techniques are discussed and evaluated. Second, possible benefits of 3-D file browser interfaces are suggested. Finally, a list of design principles for use in 3-D user interface development was created by condensing research from HCI, human cognition and perception, VR, and UI design. These form the basis of the prototype immersive 3-D first person file browser currently under development.

---

1. Introduction

Windows, Mac OS, Linux—when most people think of using a computer they think of the operating system they use to store and view their files. For most users a significant portion of their time is spent organizing, storing, and retrieving their files using the GUIs that these operating systems provide. However, data organization is becoming increasingly complicated as both the volume of data we receive increases and as the sources of that data become more diverse. Cached internet sites, downloaded information, instant messenger conversations, emails, personal documents, data from PDAs, phones, mp3 music players, digital cameras, and more all find their way onto our computer en masse. Traditional GUI file browser interfaces with hierarchical file organizational schemes are becoming less and less viable for organizing such diverse data:

- Large amounts of data
- Data received over time
- Data used in different types of tasks received from various sources

File hierarchies are efficient means of referencing and storing files on a disk. They prevent namespace conflicts by separating files into directories, and most importantly, allow for efficient searching of documents by taking advantage of the *semantic* structure which the organization of the hierarchy maps onto its files (Dourish 134). Each directory divides the file-space according to some attribute (“Personal Documents” and “Work Documents” for example) which lets users apply a simple reduce-and-conquer search. By pruning irrelevant portions of the hierarchy according to the file characteristics mapped to the various directories, users must only search through a small fraction of the total documents.

However, studies have made it increasingly clear (Marsden 122, Dourish 135) that average computer users today do not employ the “deep” filing system that allows for efficient (order log n) reduce-and-conquer searches. The designers of Windows 95 acknowledged the confusion hierarchies cause by creating “patches” like the “My Documents” folder which often acts as a flat dump for user files.

Hierarchies are also extremely inflexible. They organize data into a semantic structure according to a specific set of characteristics which then cannot be easily modified. They lock the user into a specific organizational scheme which may not be appropriate for all situations. Therefore, interaction with the data is based on the way in which the data was stored, not necessarily on the properties of the data which pertain to the user’s task.

These difficulties that file hierarchies face in dealing with the types of data computer users accumulate every day require solutions beyond the tried and true hierarchical file systems accessed by GUI file browsers using window-icon-menu-pointer (WIMP) interfaces. There have been numerous responses to this important issue—both in the form of theoretical research in operating system design and human-computer

interaction, and also in the form of actual implementations of new file browsers and visualization tools. However, even though much research has been done in human perception, human cognition, advanced interaction technologies involving sound synthesis, speech synthesis and recognition, haptic feedback, and more, we still lack a systematic way for incorporating this body of knowledge into effective user interfaces. Dr. Frederick Brooks of the University of North Carolina at Chapel Hill identifies "User Interface Design for Computer Systems" as one of his "Three Great Challenges for Half-Century-Old Computer Science," stating that not only do we have no systematic or disciplined way of integrating this body of knowledge into interfaces, but we have no "reliable ways even of predicting whether a proposed specific interface design will be good" (Brooks 26).

One intriguing potential improvement to the traditional WIMP file system interface is the use of three dimensions. Driven by ever increasing processor power and 3-D graphics hardware, there has been an amazing amount of research in many aspects of this broad field of study. However, the design space for 3-D user interfaces is much more expansive than 2-D (Bowman 106), encompassing everything from totally immersive virtual reality and mixed reality systems to 3-D worlds navigated through traditional flat screen/keyboard/mouse interfaces to traditional interfaces and systems employing some form of depth or perspective. This variety has made the development of cohesive paradigms for developing and interacting with 3-D interfaces every bit as difficult as the general case which Brooks discusses above.

The motivation for working with 3-D interfaces despite these difficulties lies in the potential for superior data visualization, more natural data organization and representation, better use of screen real estate, and the exploitation of human's natural spatial cognitive abilities that 3-D interfaces may be able to provide (see *Three Dimensional Interface Advantages* section). This research proposes a 3-D file browser which allows users to navigate a virtual office whose objects (papers, books, shelves, etc…) represent the files and programs on the user's computer. The design for this file browser is based on a proposed set of design principles for 3-D interfaces compiled from the aggregate research in this field. Thus, not only is a specific solution to the organizational difficulties of the average PC user proposed, but it is built on top of a general framework which is developed to answer Brooks' call for an integration of the body of knowledge into more compact interface guidelines. Note, that to limit the complexity of this undertaking, the guiding principles presented are targeted specifically to 3-D file system interfaces, although many of the principles discussed could be applied elsewhere as well.

In his "Introduction to 3-D User Interface Design," Dr. Doug Bowman acknowledges (in agreement with Dr. Frederick Brooks) that the design techniques and paradigms for 3-D interfaces are far from mature, but provides some basic guidelines for the creation of general interface paradigms which this research attempts to follow. First, interface design should be solidly grounded in existing human factors research. The list of principles for 3-D interface design presented in section 6 *Proposed 3-D Interface Design Paradigm Principles* clearly relates the proposed principles to large amounts of research in the field. Secondly, Bowman suggests that design principles should reuse interaction techniques developed by researched and reuse or incorporate existing design models and strategies. For this reason, section 5 *Current Work* presents a thorough

practical review of the current attempts at various kinds of 3-D file systems with discussion about their relationship to the proposed 3-D file system and the proposed set of interface principles. This review also provides the reader with an understanding of the current state of the field and an understanding of where the proposed system fits.

Therefore, this paper will answer the following questions, building a logical progression from the motivation for the development of a 3-D user interface paradigm (specifically applied to the organization and visualization of files on a personal computer) to current 3-D interface work, and finally to the specific suggested specifications of the proposed system:

1. Why are consistent/cohesive interface paradigms important for interface development and usability?
2. Why should a consistent/cohesive *3-D* user interface paradigm be developed? What are the advantages (and disadvantages)?
3. What current systems exist? i.e. what is the current state of the design space which this new paradigm would exist in?
4. What should a 3-D user interface paradigm accomplish? What general design principles should it be based on?
5. Finally, what are the specifics of the prototype system suggested based on this research?

The background section of this thesis is broken down according to theses five key questions. The remainder of the paper discusses implementation details of the specific prototype system which was constructed along with success criteria and metrics as well as a discussion of the current state of the field and suggestions for future work and extensions.

2. Interface Metaphor and Design Paradigms

Before arguing for why exploring the design possibilities of 3-D file system interfaces (and indeed 3-D interfaces in general) is worthwhile, it is instructive to comment on the power of the “desktop” metaphor and explain why this basic interaction metaphor is left virtually untouched in the proposed 3-D file browser.

Metaphors are mappings from one domain to another. They allow domain knowledge from one area to be applied to something completely separate (Marcus 48). Most office workers do not know how a computer’s file system manages their data, but they readily understand filing of documents into folders. Thus, the *structural* metaphoric connection between computer data and paper and manila folders allows them to interact with the computer despite no real knowledge of how it actually functions. The traditional desktop metaphor also conveys *operational* knowledge, mapping actions performed at a real desktop (such as throwing away a piece of paper) to deleting data on a computer system. Finally, well crafted metaphors like the desktop metaphor are *pragmatic*—able to work in reverse to allow the user to gain understanding about the abstract system.

By providing the user with a familiar environment, allowing the user to make inferences about the functionality of the system by assuming the consistency of the metaphor, and flattening the learning curve of the system, metaphors provide real

advantages to users. The desktop metaphor proves powerful in this regard since the tasks performed at ones desk on a computer very closely mimic those performed on a computer (organizing and retrieving files, working with documents, etc…). This limits the impact of one of the main costs identified by Aaron Marcus in "Metaphor Design in User Interfaces" associated with employing metaphor in interface: metaphorical references, based tightly on technological, cultural, or aesthetic references, require constant maintenance as those references naturally shift. The desktop metaphor, however, has become so engrained and attached GUI interfaces that it has become second nature to many users. For example, people do not even consciously apply "domain knowledge" anymore—no one consciously thinks back to how they interact with a desk to determine how to interact with their computer. This fact also would allow a 3-D file browser interface to become much more transparent. By allowing interaction with computer systems to model interaction with the physical world, users will be able to focus more on the task rather than the steps necessary to achieve that task through the provided interface. This is the basic principle discussed further in section 6.10 *Interface Transparency*.

Once committed to exploring the design space of 3-D interfaces using physical world metaphors, there are other benefits which can be taken advantage of to further strengthen the system.

3. Three Dimensional Interface Advantages

Google develops complicated search algorithms and indexing schemes to fight (or perhaps organize) clutter. Microsoft Longhorn and others are developing new database oriented models of data storage to better accommodate multiple views of data by the attributes users care about. The problem "where is my data?" is being attacked from multiple angles. Another alternative is to approach the problem from a user interface standpoint by asking the question: "how can I design an interface so that users can easily store and recover data so they don't lose track of it in the first place?" Three dimensional interfaces are an ideal solution not only because of the advantages enumerated below, but also because there is a massive yet largely unorganized mass of human factors, human-computer interaction, human psychology, and user interface research from which to draw. The following section uses the research already done in this area to describe how 3-D might be able to help solve the problem "where is my data?"

3.1 Extend Familiar Interaction Metaphors

The choice of the metaphor underlying a computer interface is clearly critical to its success (see *Object Manipulation—Combine "Magic" and Metaphor* as well as *Unifying Conceptual Framework*). The "desktop" metaphor is one which has proven successful in traditional 2-D WIMP interfaces, creating a virtual space for users which acts in ways very similar to their own physical desktops. The importance of metaphor in interface design was discussed previously in section 2, and the fact that making the switch of a 3-D interface will only strengthen the already powerful desktop metaphor is therefore a clear advantage. By drawing even stronger parallels between the tasks a user performs (as well as the way those tasks are performed) at their physical desk and virtual desktop, the user will not only have more queues from their own experiences which they may apply to the

operation of the computer interface, but also have a greater sense of security and familiarity with the new interface (Marcus 48).

This project suggests an extension of the traditional and successful 2-D desktop interface metaphor to 3-D. Hopefully, the sense of a personal space in which traditional physical rules and desktop organization principles can be relied on can be preserved, while adding the additional benefits enumerated below

3.2 Flexible and Natural Ad-Hoc Data Organization

As personal computers store more and more data of widely varying formats and functions—email, instant messenger conversations, music and media files, traditional text documents, application data files of every kind, HTML documents, and more—the problem of finding and organizing data is becoming more and more critical. While browsing, users need both context (information about their file path in a hierarchical system, for example) and detail (information about available files as well as options for browsing further into the file system) (Faichney 13).

Current hierarchical user interfaces, because they hide details about the directory structure beyond the current folder, encourage users to create shallow/broad directory structures rarely deeper than 6 levels (Faichney 19). However, as Paul Dourish of the Xerox Palo Alto Research Center points out, this shallow directory structure prevents full utilization the hierarchical structure. Directory hierarchies work through “progressively focus[ing] on a smaller and smaller set of documents” by categorizing the documents according to their characteristics (Dourish 134). Thus, a deeper directory structure should allow more accurate categorization and thus more accurate search. However, while hierarchies are efficient when well organized and maintained, the often unmanageable influx of data makes this upkeep task difficult. Also, hierarchies allow only one immutable organization of the data. If the top level of a directory is split into semantic categories such as “work documents,” “picture files,” and “personal correspondence,” it is impossible to rearrange all the documents in the hierarchy to be grouped by size or created date, for instance.

The attention that large companies like Google and Microsoft are giving to this issue is testament to its seriousness. Google recently released *Google Desktop Search* (http://desktop.google.com/) which indexes the hard drive of your computer and allows a Google-like search of emails, documents, websites visited, and instant messenger conversations (see *Google Desktop Search* section for more information). Google recognizes that people do not have the time to sort and organize information which they may never use again, and instead provides an easy way to search that mass of information according to any of its characteristics. This solution circumvents file hierarchies’ disadvantage of allowing organization according to only a set group of characteristics (represented by the directory structure and the semantic meaning of the names given to the directories).

Longhorn, Microsoft’s newest incarnation of Windows, tackles the same problem with WinFS. This new file system uses an XML schema to tag data according to its meaning and purpose, allowing searches with even more in depth parameters than *Google Desktop Search* can provide. Not only can items be searched for according to any of their properties, but by borrowing the Unix concept of organizing files in a directed acyclic

graph, files can also be search for according to the properties of items related to them (Rodriguez).

I propose a different solution to this data organization problem—although it should be noted that the most powerful solutions developed as more research is conducted in this area will probably be flexible hybrid solutions, allowing users to organize and sort their data according to their specific needs. A 3-D visualization of computer data provides some very unique organizational advantages. Studies such as those by Cockburn and McKenzie as well as Parush and Berman and others have clearly indicated that humans very readily remember and understand spatial relationships. Thus, one clear advantage of a 3-D spatially oriented interface is the ability to exploit spatial cognition (see the separate section *Exploit Spatial Cognition* below) to remember the location of files. This decreases the need for explicit organization of files as well as for complicated search techniques. If the user can simply remember the location of a file, there may be large gains in decreased frustration and increased ease of use.

Perhaps the most efficient way to store those ten documents that you use every day is to stack them up and toss them right in the middle of your virtual floor. Not only would "real-life" organizational techniques like piles, filing cabinets, drawers, and boxes give users flexibility in representing the organization of their data, but the objects themselves would provide landmarks in the virtual environment. As indicated by Parush in his study, these landmarks would further serve to increase spatial memory and ease navigation through the virtual world (Parush 391).

3.3 Enhanced Data Visualization

Humans have the amazing ability to very efficiently skim vast amounts of data in parallel to search for a specific piece of information. We only need to remember the last trip to the library, bookstore, or grocery store. While searching for an item, we rapidly scan the shelves, not reading every label, but instead looking for key patterns which cause us to slow our search and examine certain items more closely. While this may not be the most efficient way to locate a file on a computer system if the exact name of the file is known, it may vary well be more efficient for browsing files when there is no specific file being searched for.

Organizing files in 3-D, and displaying them in virtual organization devices (like bookshelves) would facilitate this kind of quick scanning of files. It would also allow the user to take advantage, to some extent, or peripheral vision. While the user will generally concentrate on the center of the display, the surrounding objects will still be visible, and if relevant, may catch the attention of the user.

The data visualization advantages which may be gained can also be made much more acute by the incorporation of specific visualization and creativity support tools into the interface. For more information about the types of creativity support/visualization tools that might be included, see *Integrated Creativity Support Tools/Data Visualization*.

3.4 Exploit Spatial Cognition

Modern graphical users interfaces already strongly rely on the power of human capabilities for spatial cognition. GUI interfaces are efficient because we can easily recognize and remember that the folder we want is in the upper left corner of the window, or that a window can always be closed by the 'x' button in its upper right hand corner. In

Evaluating Spatial Memory in Two and Three Dimensions, Cockburn and McKenzie motivate their study by citing "several research projects…that have shown that measures of spatial cognition are strongly correlated with performance in a variety of user interface tasks" (Cockburn 360). In fact, they explicitly state that "it may be possible to leverage human spatial capabilities by providing computer-generated 3-D scenes that better reflect the way we perceive our natural environment" (Cockburn 360). Among the advantages of leveraging these spatial capabilities is improved memory (Czerwinski 163), which Cockburn asserts should lead to improved user performance.

Cockburn sites Data Mountain as a specific example of 3-D spatial organization of data being used aid user memory. Data Mountain performs a function similar to Internet Explorer's favorites feature (storing and organizing bookmarks) but does so by allowing the user to organize thumbnails of the web pages in a 3-D space. Even after not using the system for 4 months, and without the aid of thumbnails, users found favorite web pages faster using Data Mountain than Internet Explorer (Czerwinski 163).

These initial results lend promising credence to the idea of moving these results from the realm of web browsers and favorites to file browsers. Clearly, a key metric in measuring the effectiveness of a file browser is the ease with which a user can retrieve a desired file. If a 3-D design can aid user's memory though tapping into their spatial cognition abilities, it may be worth pursing further.

3.5 Increased Screen Real Estate

Closely linked to data visualization, the ability of 3-D interfaces through tricks of perspective, information in peripheral vision, and emersion of the user inside the workspace (it exists all around them instead of merely in a plane in front of them) allows them to make more efficient use of limited screen real estate than traditional WIMP user interfaces. In his study of this very problem, D. Henderson and Stuart Card of the Xerox Palo Alto Research Center define the concept of "Window Locality Sets" based on the observation that users tend to use specific subsets of the total open windows together. When the number of simultaneously viewable windows required to complete a certain task becomes too large or the screen real estate becomes too small, the user is forced to constantly switch windows on and off the screen, resulting in "window thrashing" (derived from, and very similar in nature to, the process thrashing which occurs in virtual memory paging systems with insufficient main memory and too many processes).

By allowing the user to change "workspaces" by simply turning to the side or moving to a new location, 3-D interfaces essentially provide "virtual" screen real estate by providing the user with a simple means of switching workspaces. Many traditional WIMP interfaces have implemented similar concepts, usually by providing the user with multiple virtual desktops. However, 3-D interfaces have the distinct advantage of providing a natural relationship between the various virtual desktops (here a "virtual desktop" in a 3-D space can be thought of as any view of the space). The spatial relationships between these various views may make switching between them (achieved by navigating through the 3-D space) more intuitive and more fluid. Also, users can combine sections of various workspaces. If the user has a desk next to a bookshelf in the 3-D space they could concentrate on each individually. Alternatively, they may look between the two, seeing half the desk and half the bookcase and allowing easy transfer of information between the two.

3.6 Increased User Satisfaction

In his evaluation of the effectiveness of 3-D document management systems, Andy Cockburn of the University of Canterbury suggests that 3-D interfaces may not provide significant increases in efficiency, but do significantly impact the user's enjoyment of a system. His study compared two similar document management systems (used to store web bookmarks represented as thumbnail images) which differed only in the number of dimensions available. In one version, users organized bookmark thumbnails on a 2-D surface, in the other users were also able to push bookmarks backward or bring them forward, giving the system a sense of depth. Note that unlike the document management system proposed in this paper, Cockburn's system employed a fixed perspective for both the 2-D and 3-D systems. Although he found no statistically significant differences in information retrieval time, he does comment that "many users of the 3-D interface also commented that the interface felt 'natural' and 'a good way to organize bookmarks'" (Cockburn 439).

4. Three Dimensional Interface Disadvantages

Despite the above advantages, three dimensional interfaces are far from free of issues. These range from problems that arise simply from the fact that traditional WIMP interfaces are so prevalent and are so integrated with existing software, to issues that arise from the 3-D medium itself.

4.1 Need for a 3-D Window Manager

Even assuming that an extremely successful and efficient 3-D file browser was developed, it would be extremely difficult for it to become widely used because all applications are still developed with a 2-D design paradigm in mind. A 3-D window manager, while not critical for the development of a functional 3-D file browser, would probably be necessary before such a system could become incorporated into commercial operating systems. It may be too cumbersome and unintuitive to interact with files inside programs in a different way than one interacts with files in the file browser. For example, what would the save dialogue in a non 3-D enabled programs look like? It would still have a hierarchical file list even though outside the program the data is displayed and storied fundamentally differently.

Since this project is merely a proof of concept of the viability of a 3-D file browser using a completely non-WIMP interface metaphor, this hurdle is not within the scope of my consideration. This project concentrates on the question of whether and what advantages a 3-D file browser offers over current implementations. If advantages are found to exist, then perhaps the next steps would be to examine these implementation details.

4.2 Limited Feedback

We live in a three dimensional world. Three dimensional manipulation and interaction is second nature to us. Yet mainstream WIMP interfaces remain solidly two dimensional. This is partly because of our 2-D output and input devices (mice and screens) and partly because computers lack the ability to accurately represents the wealth of sensory cues that the real world provides (Bowman 96, Mine). Therefore, despite living in 3-D space our entire lives, manipulating objects and navigating *virtual* 3-D environments is still

much harder for humans than the real thing. Dr. Hinckley of the University of Virginia proposes that this difficultly in interacting with virtual 3-D spaces arises from the fact that “people do not innately *understand* three dimensional reality, but rather they *experience* it.” Spatial relationships are understood through interaction and experimentation—through grasping an object and moving it around in the space—though butting it up against other objects and seeing how they interact. Without the complicated, multi-sensory feedback that physical objects provide, users of virtual 3-D interfaces must make spatial decisions based on visual observation of the environment.

5. Current Work

Design always consists of a synthesis of new ideas with established techniques, and interface design is no exception. The following section explores current systems which employ some 3-D element to accomplish some interface task. The purpose and design of the system is summarized, and then the applicability of their specific application of 3-D interface technology to the proposed first person 3-D file browser is discussed. The second part of this section deals with alternative ways of organizing and searching through files on disk. This section focuses less on the user interface and more on how modifying the internal representation of files on disk can improve search and file storage efficiency. The discussion, however, still revolves around how a 3-D file browser interface could be designed to support these features.

5.1 Approaches to Three Dimensional User Interfaces

5.1.1 3D-Desktop

http://desk3d.sourceforge.net/

The *3d-Desktop* project is an open source desktop switching tool for Linux that allows for 3-D visualization of and switching between the multiple virtual desktops which Linux supports. The software does not change the desktop interface at all, simply allowing the user to “step back” from the current desktop into a view where multiple virtual desktops

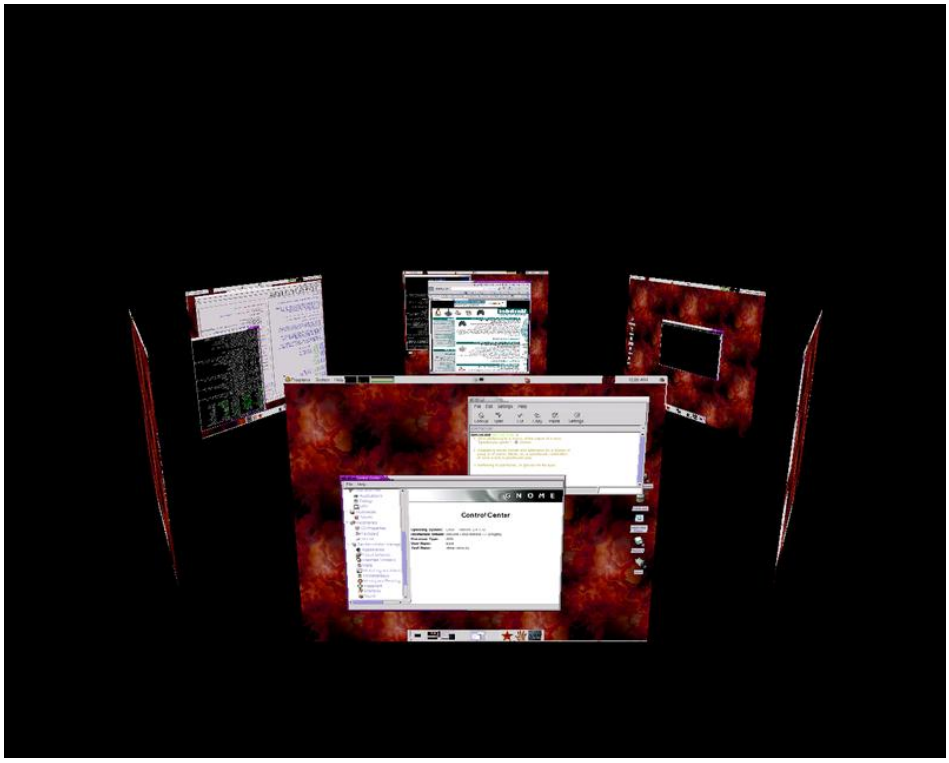

Figure 1 Using 3d-Desktop to switch virtual desktops

are simultaneously displayed. One of those desktops can then be selected and zoomed in on, bringing the user back to a 2-D desktop view.
While this project varies greatly in its function from the 3-D file browser proposed, it does exploit a few key features of 3-D visualization. The 3-D view allows users to see numerous desktops in a much smaller space than would be possible with a 2-D display, hopefully allowing for quicker selection of the desired desktop. *3d-Desktop* also provides an example of how transitioning between 2 and 3 dimensional interfaces might be done—a key consideration when developing any 3-D interface which must cooperate with the host of non 3-D applications available.

5.1.2 3DNA Desktop

http://www.3dna.net/products/desktop.htm
3DNA Desktop is a commercial product which envisions the desktop in much the same way as the browser proposed in this paper. The user begins with a 3-D "world" that has been rendered and designed by a 3-D graphics artist (choosing from themes such as Villa, Xenon, Basestation, Undersea Lab, and Chateau). When you launch the program, you are taken to this 3-D world which has been populated with elements from your computer—the idea being that you are now "living" in this dream world. In addition to being able to create 3-D objects which allow you to load the files and programs associated with them, you have access to 3-D games, internet surfing, or simply exploring the world that 3DNA Desktop has created.

Figure 2 Features of the 3DNA Desktop

This desktop program is clearly very similar to the proposed file browser. It scans the users desktop to create a 3-D world for them which allows them to access content on their computer from inside that 3-D world. The user has a virtual desk (see figure 2) which give the user access to traditional desktop tools like a calculator or address book (much in the same way that the Macintosh desktop first cemented the desktop interface metaphor by providing similar options). The user is able to arrange shortcuts to file and programs in various spaces above the desk as well. In this way, *3DNA Desktop* is taking advantage of

both the enhanced data visualization and spatial cognition which a 3-D interface should be able to provide (see *Exploit Spatial Cognition* and *Enhanced Data Visualization* in the *Three Dimensional Interface Advantages* section). It also embraces the idea, which I have also supported, of extending the desktop metaphor and keeping the interface in the realm of what is understandable to the user. We spend all our lives in rooms, thus the user of *3DNA Desktop* automatically has some understanding of where things might be in the 3-D space. This consistency allows users to pick up and being using the interface immediately, with no frustrating learning curve that might cause them to give up entirely.

However, while *3DNA Desktop* is an excellent example of an actual product based on many of the interface principles described in this paper, it is lacking in a number of key areas. The organization of the room and the virtual objects it contains is much too structured to be useful as a general purpose file browser. It would be very difficult to imagine a future version of *3DNA Desktop* replacing Windows Explorer as a fully functional file browser. There are very specific locations where files, shortcuts, and icons can be placed. One of the great benefits that I envision the system proposed in this paper having is the ability to organize files and programs in much the same way as we organize objects in the real world. If a file is used all the time, it can be placed on a pile right at the front of the desk. Less important notes can be thrown in a drawer, or a storage box next to the desk. You can "conjure" a filing cabinet and make a drawer for tax information. Only by allowing "freeform" organization of the 3-D space will the purported benefits in the form of improved spatial cognition and easier, more accurate ad-hoc data organization.

Figure 3 The user's virtual desk

5.1.3 Rooms 3D

http://www.rooms3d.com.sg/index.html

Rooms 3D is an approach fairly similar to 3DNA Desktop and is even more focused on providing a surrealistic, exciting world to explore rather than a functional user interface. It is worth mentioning only as another example of an existing application which views to files on a computer as objects in a 3D space. However, it is limited to showing objects (program shortcuts or files) as blocks floating in space. The blocks can be moved to various locations in the world, but that is the extent of the ability of the user to organize or modify the environment. Thus, Rooms 3D is far from a functional file browser, and falls more into the realm of an interesting experiment or entertaining diversion.

Figure 4 Rooms 3D Scenes

5.1.4 SphereXP

http://www.hamar.sk/sphere/

SphereXP falls into the broad category of 3-D interfaces which stick to the traditional WIMP metaphor but attempt to extend it in some way to three dimensions. In the case of SphereXP, like many applications in this category, the goal is to make better use of limited screen real estate. SphereXP does this not only by allowing windows to be viewed at an angle and to partially occlude other windows, but by mapping the user's entire desktop onto a virtual sphere and allowing the user to rotate their view around that sphere. Thus, the user's desktop workspace essentially becomes a 360 degree spherical surface surrounding the user. SphereXP focuses on solving the issue of limited screen real estate (also discussed as an advantage of 3-D systems: *Increased Screen Real Estate*). Navigation is augmented by providing the user with a "mini-map" showing a wire-frame representation of the surface of the sphere.

The use of overview maps for both 2-D and 3-D navigation is the subject of quite extensive research. Dr. Hornbaek of the University of Copenhagen suggests that there is a basic tradeoff between user satisfaction (increased by employing an overview map) and task completion time (actually deceased by the inclusion of an overview). He suggests that the cognitive overhead associated with using both a detail and an overview map for navigation can be decreased by eliminating navigation commands dealing specifically with the overview map and only allowing navigation in the main map (which is what SphereXP does, using the mini sphere only for reference). He finds that simply allowing the main map to zoom to multiple levels of detail actually provides more effective results. This scaling technique is similar to the "scaled-world grab" object manipulation

technique discussed in section 6.8.1 *Magic Manipulation*. Zoomable interfaces are also another active area of UI research, discussed for example in Dr. Benjamin Bederson's work on zoomable interfaces in Java.

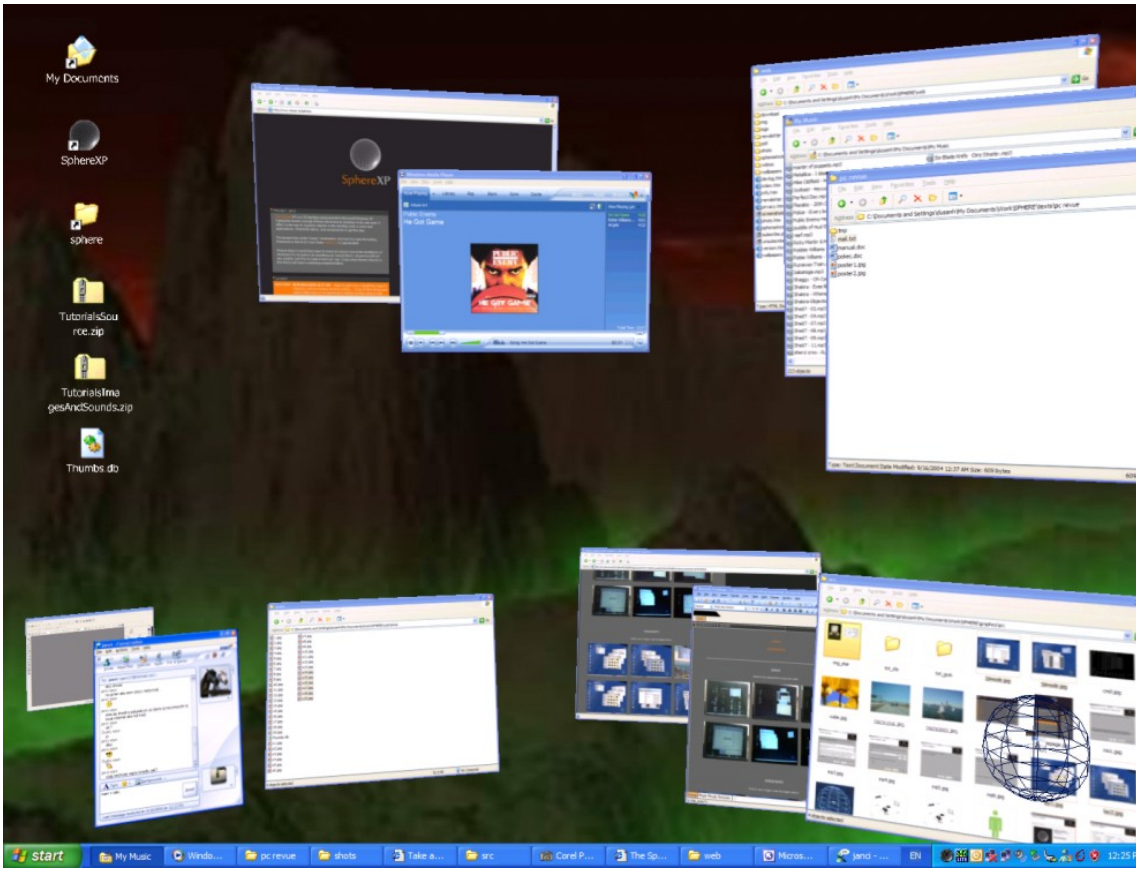

Figure 5 SphereXP desktop

5.1.5 XCruiser

http://xcruiser.sourceforge.net/

XCruiser is an interesting file system visualization utility which represents the files on a computer as a "universe"—allowing the user to fly around. Directories are represented as Galaxies, files are individual planets, symbolic links are wormholes. The positions of files are determined by the length of the filename. The program only acts as a visualization tool, not allowing any manipulation of the files or directory structure.

While I am not convinced that this particular representation of a file structure has merit (using it to find a specific file would clearly be tedious and frustrating), the underlying concept of providing an alternate ways of viewing data is similar to the strategies discussed in the *Integrated Creativity Support Tools/Data Visualization* section dealing with the Multimedia Bulletin Board project. Any file system representation, even a straightforward hierarchical organization like that provided by Microsoft Explorer, has strengths and weaknesses. When viewing files with Microsoft explorer, for example, you have a very clear view of your path back to the root directory, but no indication of the structure of other branches of the tree, or even of the structure of the next directories you will be visiting. Hyunmo Kang's Multimedia Bulletin board explores the concept of

providing multiple views of data so that the user may choose whichever view best fits the task at hand.

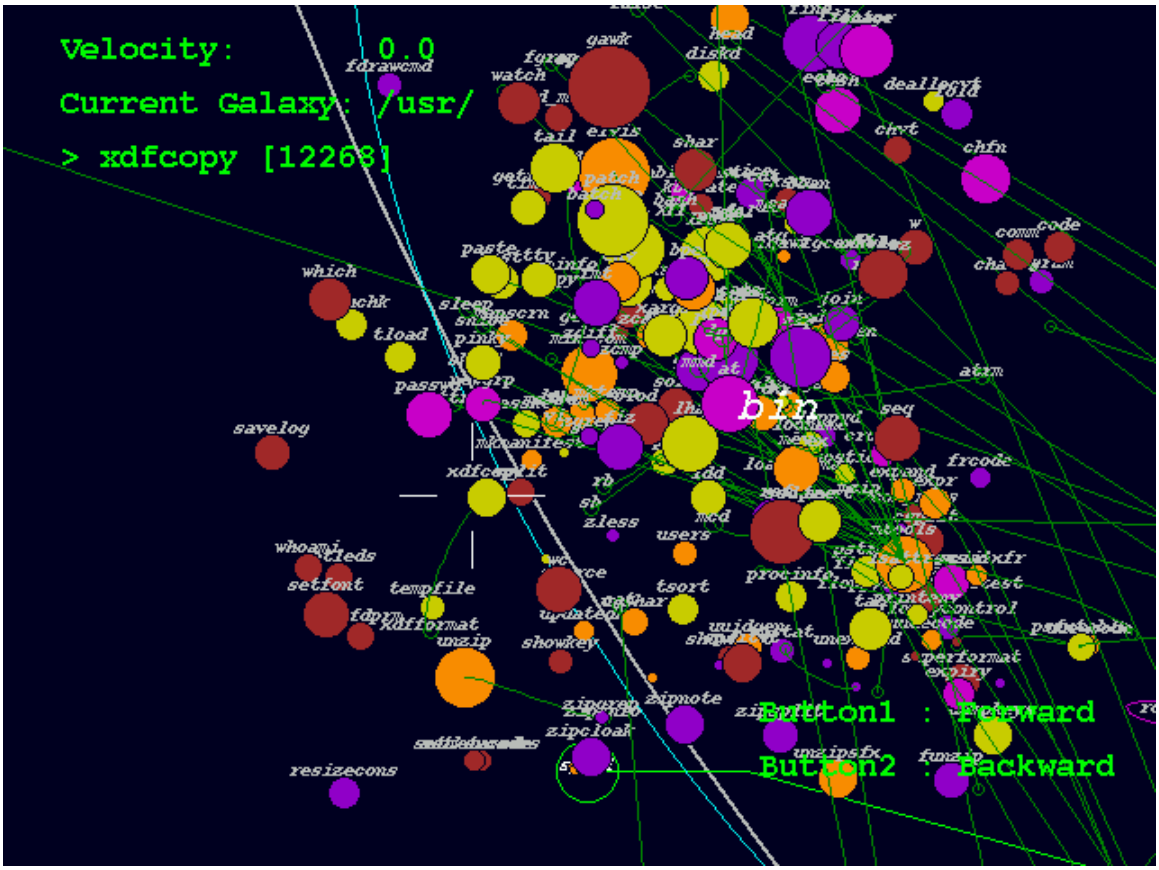

Figure 6 Visualizing the files in the /bin directory

Thus, while *XCruiser* is clearly not going to replace Windows Explorer, the power of providing alternate displays for data is clearly evident even in mainstream file browsers. Both Macintosh OSX and Windows XP provide numerous “views” which determine how a window displays its contents. Files can be browsed as thumbnails, emulating the *Enhanced Data Visualization* advantages discussed in the *Three Dimensional Advantages* section. The “detailed” view allows users to sort by attributes of the files, i.e. name, date created, size, and more. The concept of multiple user levels, only showing minimal features for less advanced users (discussed further in the *Evaluation Cost of Interface* section), is also tied in with data views. Most modern operating systems are not designed with a “level structured” interface in mind, however they often do at least provide a simple and intuitive mode for viewing and selecting files and folder. Both Windows and Mac OSX, for example, provide a “thumbnail” view which displays folders and files as large boxed icons which cannot be selected or moved and can be opened with a single click. This eliminates such advanced features and dragging and dropping files, right clicking to access pull-down menus, and viewing detailed information about files. Thus, by providing adjustable data views, the interface designer can not only make data easier to use in certain situations, but can also make it easier to use by users of different skill levels.

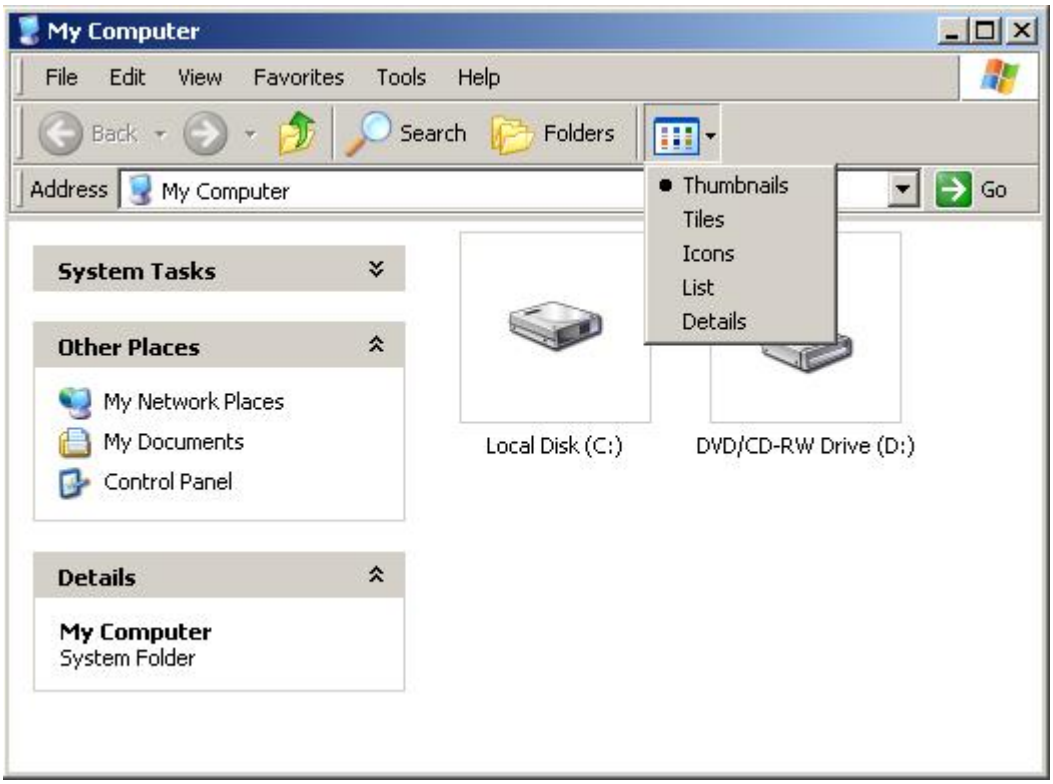

Figure 7 Windows file browser *view* options

5.1.6 Project Looking Glass

http://www.sun.com/software/looking_glass/

Project Looking Glass differs from many of the above examples in its scale more than anything else. Project Looking Glass is being developed by Sun Microsystems to show off the capabilities of their Java language:

> Java technology-based developments will bring 3D windowing capabilities to the desktop to offer a far richer user experience for work and play. But, it's not only about looks, it's about creating an engaging user experience, one that can make communications and collaboration even easier…We are moving beyond the boundaries of old environments to revolutionize the use of the desktop.

The operating system user interface is chock full of ways to extend traditional WIMP desktops to exist in a 3-D space. Windows can be flipped over to reveal program options or a space for writing notes. Windows can be pushed to the side or rotated out of the way to allow more applications to coexist on the screen at once. In a manner similar to *3-D Desktop*, the user can pan between virtual desktops by clicking on the side of the screen. Finally, alternate views which are ideal for quick browsing of data are included for songs and media files (see figure 6 below).

Figure 8 Sun's Project Looking Glass

Clearly, Sun is taking a very different approach to developing a viable 3-D user interface than the approach proposed in this paper, but they are leveraging many of the same advantages and principles. The icon bar at the bottom of the screen, similar in function to "the dock" in Mac OSX, provides the same type of virtual proprioceptive feedback described in the *Proprioceptive Feedback* section of the *Interface Design and Human Factors Principles* section. It links functions, tools, and shortcuts to a specific portion of the screen (relative not to the user's body as in true proprioceptive virtual reality systems, but to the user's view of the screen). The function is the same however. It provides a constant (independent of the current state of the rest of the workspace) region for accessing critical and frequently used functions in an intuitive manner, a virtual tool belt.

Also attempting to answer the problem of screen real estate, *Project Looking Glass* takes a radically different approach than *SphereXP*. It employs a single workspace like a traditional WIMP interface, but uses 3-D to introduce perspective and distortion, allowing some less important windows to be angled away from the user. Thus they remain visible, but take up far less screen real estate.

5.1.7 3DOSX

http://www.acm.uiuc.edu/macwarriors/projects/3dosx/

3DOSX is a 3-D file browser designed for Mac OSX which, similarly to many of the above interfaces, attempts to maintain the traditional WIMP interaction/display metaphor while adding a 3-D perspective. In 3DOSX, folders are "platters" which can be rotated

similarly to a lazy-susan. Icons sit around the edges of the platters and the directory structure is indicated by overlaps and interlinking between the platters.

Like the other interfaces above, the main purported benefit of 3DOSX is the data visualization which it provides. Since all folders are open simultaneously, the user is able to gain a much more holistic view of the directory structure. This technique, coined a "Goldleaf" browser by Dr. Jolon Faichney of Griffith University who developed a very similar system, allows more fluid interaction between various levels of the hierarchy than would otherwise be possible. Displaying all directories simultaneously solves another problem with traditional hierarchical file browsers like Windows Explorer. The path of directories you have taken to get to a given directory is easy to see with the collapsing list view that Explorer provides. However, you are given no indication of the directory structure beyond that point. With 3DOSX, the user can "look ahead," enabling them to navigate more quickly and more accurately to the correct folder, even skipping over directories if they see the target directory far in the background a few folders deep. Finally, 3DOSX may actually also provide tangible speedups when performing tasks like moving or copying files and folders. Since all directories are simultaneously available, the user is freed from having to juggle various windows to perform the copy or move operation.

Figure 9 3DOSX Screenshot

5.2 Approaches to Document Organization and Search

5.2.1 Google Desktop Search

http://desktop.google.com/

Making sense of and organizing the massive amount of data we interact with and store is quickly becoming an unmanageable task. Google is tackling the problem of "where is my

stuff" with the same powerful search algorithms and indexing techniques they employ to search the web. Useful features include automatic caching of IM and email conversations, old versions of files, and websites which may have been modified since you last visited them. Google explains the value of this technique:

> Since you can easily search information on your computer, you don't need to worry about organizing your files, email, or bookmarks. You can just do a quick search for what you remember seeing, instead of having to remember exactly what file, email, or web page had that information, and where that item is now located on your computer.

However, the most intriguing part of *Google Desktop Search* as it relates to this study of file browsing techniques is that to some extent it removes the need for file browsers in the first place. Perhaps it would be impractical in its current form as a complete replacement for traditional file browsers or hierarchical organizational techniques, but it does provide an interesting alternative. To date, search functionality built into operating systems like Windows and Mac OS have been unwieldy at best. *Google Desktop Search* is perhaps the first such offering that makes Google's "search don't sort" mantra applicable to file browsing.

Indeed, a major motivation behind the proposed first person 3-D file browser is the massive influx of data which we receive every day through channels like email, IM, web info, and simply creating documents. By providing a natural way to organize this data (instead of forcing people into unnatural and unintuitive hierarchical file systems—see Dourish 135) by mimicking the real world. I believe that combining an effective ah-hoc organization scheme such as this (which allows very quick access for recently used and frequently used data—thereby taking advantage of a form of the principal of locality) with an intuitive search mechanism for more complicated queries will provide the best results for future file management systems.

A search mechanism similar to Google's could easily be combined with the proposed file browser through the techniques discussed in the *Flexible/Task-Based File Organization* section. For example, a "search" could have the effect of dynamically rearrange the organization of the objects in the virtual space, allowing the user to take advantage of the *Enhanced Data Visualization*

5.2.2 Presto/Vista

📖Paul Dourish, "Presto: An Experimental Architecture for Fluid Interactive Document Spaces"

Presto is a document management system which supports file browsers (such as Vista) that are set up to allow "rich interaction with documents through meaningful, user-level document attributes" (Dourish 133). By tracking specific document attributes in a manor similar to the XML-like tags that Microsoft's WinFS file system uses, Presto allows users to "rapidly reorganize their document space for the task at hand" (Dourish 133). In this sense, Presto is an attempt to create a file browser based on the same principles underlying Hyunmo Kang's Multimedia Bulletin Board discussed in the *Integrated Creativity Support Tools/Data Visualization* section. That is, both are based on the concept of allowing fluid reorganization of the way data is visualized to fit the task at hand. Presto is important because it applies this concept directly to file browser systems.

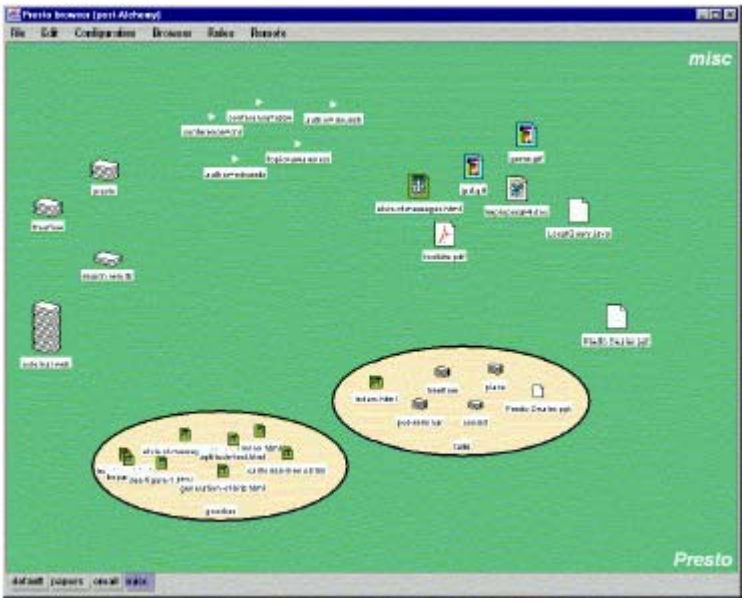

Figure 10 Vista Browser

The creators of Presto base their motivation in part on research by Barreau and Nardi which suggests that most users creating hierarchical organizations of their data do not employ deep or complicated systems, but rather create simply, shallow structures whose main purpose is simply to group related files together (either on the desktop or in folders). Presto provides mechanisms for easily creating groups of data instead of hierarchies, either automatically based on their characteristics or manually by dragging files into and out of the group (Dourish 151). Thus, files are not only spatially grouped according to their relationships, the "multivariate" nature of documents is recognized because individual files can be part of multiple groups. The key advantage of Presto then, is that "the structure of the document space reflects changes in the state of the documents, rather than simply their state when they were filed" as it would in a standard hierarchical file system (Dourish 137). Simply put, as files change their membership in various groups is automatically updated.

This concept of an "aspect" oriented file system arranged based on relationships and similarities between files is tightly integrated and expanded upon in the proposed file browser. The idea of grouping data by attributes, for example, goes hand in hand with context sensitivity in file systems. Context, like when a file was accessed, what files were used simultaneously with that file, and (for a 3-D virtual reality or mixed reality system) information like the position and view of the user, can all be considered attributes which can be used to arrange the view of the file system (see section 6.12 *Context Awareness for Task Automation*). Presto's philosophy also combines very neatly with the Multimedia Bulletin Board described by Hyunmo Kang at the University of Maryland in section 6.11 *Integrated Creativity Support and Data Visualization Tools*). In essence, he attempts to reconcile the power of automatically organizing data according to attributes with the flexibility and emergent organization that comes from letting a user manually organize their data. By saving multiple views of the data and allowing those views to be dynamically switched and restored, Dr. Kang reconciles the power of *Ad-Hoc Data Organization* discussed in section 3.2 with a Presto-like attribute based data view.

As mentioned, the synthesis of these ideas is tightly integrated into the design of the proposed file browser. By interacting with their files in a 3-D virtual environment, the system will have access to much more contextual information than would otherwise be possible. That information will be combined with Presto-like attributes attached to files. The user will then be able to automatically re-arrange their virtual space based on any combination of the stored attributes. For example, the user could select a target file, and then have all the other files arrange themselves in stacks radially around the user's position based on how often that file was open at the same time as the selected target file.

5.2.3 Microsoft Longhorn (WinFS)

http://www.c-sharpcorner.com/Longhorn/WinFS/WinFSDataModel.asp

One of the features in Microsoft's upcoming major upgrade to Windows is WinFS—a new file systems which attempt among other things to provide search capabilities rivaling *Google Desktop Search* and similar in function to *Presto* by structuring data with an XML schema that includes the "meaning" and "purpose" of the data. This closely matches *Presto's* strategy of including file attributes to facilitate organization and search. In fact, WinFS goes beyond a traditional file system or relational database, providing not only the ability to easily and efficiently search and organize information, but to share it with others.

6. Proposed 3-D Interface Design Paradigm Principles

Developing formal user interface design paradigms and principles is not a new idea. While the field is perhaps not as mature as mature as it should be (Brooks 2), much research has gone into how user interfaces should be designed, specified, and tested. Dr. Ben Shneiderman of the University of Maryland for example lists "the eight golden rules of interface design:"

1. Strive for consistency.
2. Cater to universal usability.
3. Offer informative feedback.
4. Design dialogs to yield closure.
5. Prevent errors.
6. Permit easy reversal of actions.
7. Support internal locus of control.
8. Reduce short-term memory load.

In addition to high level principles such as these, specific UI development style guides have been developed for a number of different platforms. Some examples include IBM's *Common User Access Guide to User Interface Design*, Apple's *Inside Macintosh* and *Macintosh Human Interface Guidelines*, as well as Microsoft's *The Windows Interface: An Application Design Guide* (Rogers 512). These style guides promote usability by ensuing consistency. In general, they accomplish this by providing very low level specifications covering the look-and-feel of every individual piece of the interface as opposed to providing general design principles.

Currently there exist no comprehensive style guides or lists of general design principles for 3-D user interfaces. The goal of this section is to present a reasonably comprehensive list of general design principles which could be applied to any 3-D user interface (but which are specifically geared toward the specific application being

considered—a 3-D file browser implemented as a virtual reality environment on a desktop machine). These principles are heavily based on previous research in user interface design, 3-D graphics, human-computer interaction, and psychology. Each section subheading describes a general principle. The body text of each principle's section contains a discussion of the background research which suggests its importance as well as comments on how the proposed 3-D file browser will adhere to the guideline.

Proposed 3-D User Interface Guidelines

1. Provide a unifying conceptual framework.
2. Optimize evaluation cost of the interface.
3. Provide customizability and configurability.
4. Incorporate proprioceptive feedback.
5. Allow multimodal interaction.
6. Provide aids to acquisition of spatial knowledge.
7. Allow multiple granularities of navigation.
8. Allow flexible object manipulation ("magic" techniques).
9. Allow imprecise interaction.
10. Make the interface transparent.
11. Limit the user's degrees of freedom.
12. Integrate creativity support and data visualization tools.
13. Provide task automation through context awareness.
14. Organize files by attributes in a flexible and task-based manner.

These principles were developed specifically to guide the design of the proposed 3-D file browser (this was necessary due to the lack to adequate existing design frameworks), but may also be applicable to a range of other 2-D and 3-D applications.

6.1 Provide a Unifying Conceptual Framework

Mark Mine of the University of California at Berkeley suggests two main causes for the current difficulty users experience when interacting with 3-D virtual reality systems. One of these causes is the lack of a unifying interaction framework like the window, icon, menu, pointer (WIMP) metaphor which has become so prevalent in traditional systems (Mine 19). Although Mine focuses more on immersive virtual reality systems, the issue of developing a conceptual framework is central to my investigation of alternative three dimensional metaphors for file browsing interfaces. Today, I can sit down at a Windows, Macintosh, or UNIX based operating system and generally find my way around because of the conceptual framework that the WIMP metaphor provides. It is powerful because it allows consistency across multiple platforms. To delete a file, I know to look for a trash can icon regardless of what type of system I am using. In his discussion of the general guidelines that traditional user interface designers should follow in *Designing the User Interface*, Shneiderman lists "standardize task sequences" and "consistency of data display" as two of his top considerations (Shneiderman 63). Thus, I hope to ease the switch to three dimensions by employing an "interface" which is already familiar to users and has predefined methods of data display and predefined task sequences. Users will instinctively know that to access files, one must open the correct drawer in the filing cabinet, for example.

The ability of users to quickly learn a variety of systems based on general similarities when systems are designed by following general paradigms (such as the WIMP metaphor) relies on the powerful concept of storing the function of objects "in the world" (Mine 20). Returning to the trash can example, on a Macintosh system the trash

can provides a clear contextual clue as to its function—and since it has been adopted as a standard user interface feature, its function is immediately recognized in any application which uses a trash can.

Because Mine is considering interaction problems in immersive virtual reality interfaces, some of his interaction solutions are very explicitly tailored to such interfaces. However, he presents a number of novel solutions which could be adapted to less immersive 3-D interfaces. Proprioceptive ("body-centered") interaction, which is discussed in more detail in its own section below, provides an excellent example. Proprioception in virtual reality systems provides a powerful way of using the user's body as a means to "anchor" physical mnemonics. For example, objects can be "thrown away" by grasping them and making a motion like tossing something over one's shoulder. In a less immersive desktop 3-D environment this same type of body centered interaction is not possible, but many of the same techniques can still be applied by using the edges of the screen as a physical anchor point. The top of the screen could be used a buffer for storing files. Moving the mouse to the bottom of the screen could bring up tool options, much like reaching into the "pockets" of the user's avatar. In short, the basic premise of providing tools and ways to invoke commands and access data independent of one's position in a virtual environment can be adapted from fully immersive virtual environments to non immersive ones. The various manipulation and navigation techniques (scaled-world grab, etc…) which are discussed later as well are also examples of adaptable techniques from studies of immersive virtual reality environments.

6.2 Evaluation Cost of Interface

The evaluation cost of an interface refers to "the degree of effort the user must put into finding the function of [the items]" in a particular user interface (Ehret 211). From a purely common sense perspective, one of the goals of a user interface designer should be to keep the evaluation cost of the interface as low as possible. Indeed, many of the principles presented in this section, including creating a unifying conceptual framework, developing transparent interfaces, and incorporating strong metaphors into interface design, work toward this common sense goal of making interfaces easier to use. However, in his 2002 paper on "Location learning in graphical user interface," Ehret suggests that users actually learn the location of user interface objects more effectively when the evaluation cost is higher. Thus, there may be some advantage to organizing an interface so that users must spend some initial time exploring to find the locations of items.

This result seems to fit well with the psychological research which has been conducted regarding human spatial learning. In his summary on spatial knowledge acquisition, for instance, Parush points out that "spatial cognition changes as a function of the direct experience in the environment: more experience led to more survey knowledge" (Parush 378). He also points out that "little is reported [in the literature] about the actual characteristics of the learning phase and how performance during learning can be related to navigation and orientation performance after the learning" (Parush 378). His study, which compared subject's ability to learn the locations of objects in an immersive virtual environment, compared subjects who were given maps of the environment to subjects given specific directions to various points of interest. The subjects using maps took more time to find the objects because they had to orient the map

to their view on the screen, but were just as efficient as the other subjects by the end of the learning phase. Further, once the navigation aids (both the map and rout lists) were removed, the subjects who used the map actually performed better. Thus, Parush provides a clear example of the benefit that increasing the evaluation cost of an interface can provide.

Benjamin Bederson of the University of Maryland provides another example of the benefits of increased evaluation cost in his discussion of interface design aimed at creating the "optimal human experience" or putting users "in the flow" (Bederson 1). One of the key traits which he attributes to tasks which are conducive to optimal experiences is that they must require effort to first acquire and then apply skills. Bederson suggests that this may contribute to the appeal of complicated applications such as Adobe Photoshop and Emacs. Clearly the wide range of features appeals to users, but the very fact that significant effort must be expended to learn the programs may in fact contribute to their popularity. Of course, Bederson points out that increased evaluation cost clearly does not guarantee increased overall usability of the interface/application. It clearly is very easy for an application to become frustrating and cause a user to give up. The key with applications like Photoshop and Emacs, he claims, is that once the evaluation cost has been "payed," the application's interfaces become essentially transparent—allowing power users to concentrate conceptually on complicated tasks instead of the series of steps needed to accomplish them. In fact, interface transparency is such an important concept in interface design that it is further discussed in its own section below.

Clearly, there are two conflicting principles being presented here. On one hand, user interfaces should be approachable and understandable. They should be logically organized according to some consistent metaphor which users can fall back upon to fill gaps in their knowledge about the location of interface items. However, users should also be forced to actively participate in this learning process since this participation is the most efficient way to build knowledge of the interface.

One interesting approach which has been taken to attempt to satisfy both design goals has been called a "level-structured" approach. Level-structured interfaces allow the user to set the level of complexity of the interface, learning features at their own pace in a simplified and controlled environment and then choosing when/if to give themselves access to more complicated features. This strategy prevents users from overwhelming him/herself and makes the interface easier to user by presenting new features only a few at a time. However, it does not preclude each level of the user interface from being designed with a heightened evaluation cost in mind.

Providing customization options is another way to effectively raise the evaluation cost of an interface while ensuring concrete benefits for the user. By requiring users to physically place interface items, they are forced to "explore" the interface. Bederson likens configuration to the "turning off of a telephone ringer, or clearing [of] a desk" in order to focus. He claims that in addition to helping users become better acquainted with the interface, customization allows users to have the "optimal experience" he is concerned with.

6.3 Customization and Configurability

To say that user interfaces should be customizable refers to many aspects of interface design simultaneously. At one level, Shneiderman and others argue that interfaces should

be designed with “universal usability” in mind. One of the main ways in which this is accomplished is to include extra custom features to aid hearing and vision impaired users.

Customizability also aids in allowing the complexity of the interface to scale with the level of comfort of the user. This ensures that the evaluation cost of the interface (see previous section *Evaluation Cost of Interface*) is ideal. Users will neither be too frustrated or too bored with such an interface.

Finally, one of the basic concepts behind the proposed 3-D file browser is that the underlying interaction metaphor mimics the physical world. One thing which that entails is that the user is able to rearrange their virtual space to best suit their needs. By giving the user the freedom to define what their virtual space looks and feels like, they will hopefully be more productive and more happy using the system.

6.4 Proprioceptive Feedback

Since 3-D environments currently lack any such unifying metaphors or interaction styles, Mine suggests we develop such an interaction style using the person’s natural sense of the position and orientation of their own body. He explores three ways to use body-related actions to develop a general set of natural interaction techniques: direct manipulation, physical mnemonics, and gestural actions. Direct manipulation is essentially the concept that we can work with objects which are closer to our bodies or held directly in our hands than we can with object manipulated at a distance or attached to a fishing rod (See *Object Manipulation*). Physical mnemonics involve associating hand or body motions with specific common tasks. A file could be deleted, for example, by the user “throwing” it over their shoulder. A writing “widget” could be accessed by reaching into one’s left breast pocket or above one’s ear. Menus or other controls could also be stored out of view and “pulled down” onto the screen by the user reaching above their head. Finally, gestural actions would allow users to recall objects or initiate commands with common hand gestures.

The focus of Mine’s work is on immersive 3-D virtual reality interfaces where the user uses their entire body as an input mechanism and therefore has a much greater opportunity to use proprioceptive context clues to aid in their navigation through and interaction with the world. While powerful, it is unclear whether a discussion of these interaction techniques has any place in a 3-D system where the user, sitting in front of a conventional computer screen, does not have the same sense of the motion and/or position of his/her virtual body inside the environment. Motions like reaching behind one’s head or into one’s virtual pocket would be much less natural to perform with a mouse and keyboard. However, I argue that the ideas behind the use of physical mnemonics and gestural actions can and should be effectively applied to less immersive 3-D interfaces.

One of the key differences between a conventional computer environment and a virtual reality environment is that “interface controls must move with the user as he moves through the environment and be made easy to locate” (Mine 20). While this realization leads Mine to suggest proprioceptive interaction techniques, I believe that similar results can be accomplished by using the sides of the screen in much the same way. A user could grab a file and drag it to the bottom of the screen in order to delete it. Dragging the same file to the top of the screen might place it in the clipboard. Moving the mouse to the top of the screen again would allow the user to retrieve the file.

Interaction based on the boundaries of the environment is actually an adaptation of proprioception with a concept which has been used in conventional interfaces for some time now. Beginning with Apple's OS X, users could bring up "The Dock" by simply rolling their mouse over the lower section of their screen—giving them quick access to open windows as well as frequently used programs ("apple.com/macosx/"). Hidden menu bars which can be accessed by rolling the mouse over the top of the screen have also been used to decrease onscreen clutter.

By taking advantage of the natural bounds which the edges of the screen provide, we can turn a disadvantage in our output medium into a tool for standardizing and streamlining interaction. When combined with other object manipulation techniques discussed later (See *Object Manipulation*) the 3-D interface user will be provided with a power set of mobile and easy to access tools for intuitively and quickly accessing and working with their files.

6.5 Multimodal Interaction

Interacting with a 3-D environment can be a puzzlingly challenging problem. We interact 3-D objects and navigate 3-D environments every day (a realization which fundamentally motivates the development of virtual reality systems which allow interacting with a computer to parallel interaction with the real world), yet current virtual reality environments are difficult to navigate. Mark Mine of the University of California at Berkeley suggests a number of causes, chief among them: the lack of haptic feedback, limited input information, and limited precision of movement all make object manipulation difficult.

6.5.1 Input Devices

The input and output devices available to personal computers today create a fundamental barrier to accessing the full benefits that a 3-D interfaces might provide. The mouse provides a 2-D mode of data entry, which is restrictive when attempting to use it to navigate a three dimensional world. There is an essential tradeoff between providing the user with a sufficiently expressive means of interacting with the environment and ensuring that navigation does not require extensive attention (see section 6.10 *Interface Transparency*). In order to achieve this flexibility and ease of use, it may be necessary to combine two or more disparate input techniques to capture the advantages and mutually cover for defects of both.

Since 3-D provides a unique navigation challenge, many of the current solutions have been multimodal input approaches. On more dedicated virtual reality systems "trackers, 3-D pointing devices, and whole-hand devices allowing gestural input" are attempts to provide more natural interaction with 3-D worlds (Bowman 96). In many computer game interfaces today "mouse-look" has become an increasingly popular means of 3-D navigation. It succeeds by using the keyboard keys to provide an extra degree of freedom to the user. Four keyboard keys are used to move forward, backwards, and side to side (generally 'w' 'a' 's' and 'd' are chosen because their relative positions on the keyboard suggests their function). As the name implies, the mouse is then used only to control the direction (both along the up-down and right-left axes) in which the user is facing. Mouse-look has become the de-facto standard for 3-D games. It provides

relatively intuitive steering with an easy learning curve. The only difficulty for new users is the necessity to use two hands in coordination.

6.5.2 Output Devices

Delivering information about the environment back to the user can be equally problematic. Again, when developing 3-D applications for traditional personal computers we must project a view of a 3-D scene onto a 2-D screen. Since a suggested benefit of developing computing metaphors based on real-world interaction is the ability of users to draw upon their experience navigating through and interacting with the real world (see section 2 *Interface Metaphor and Design Paradigm*), barriers to using this experience are of great concern. Bowman sums up the sources of disorientation in 3-D as a lack of sufficient output or feedback: “Although we live and act in a 3-D world, the physical world contains many more cues for understanding and constraints and affordances for action that cannot currently be represented accurately in a computer simulation” (Bowman 96). In fact, he uses these difficulties as motivation to suggest that traditional WIMP and similar user interface paradigms cannot be simply adapted to 3-D. Instead, we must develop new metaphors for computer interaction either based on real world interaction or other new metaphors (Bowman 96).

6.6 Aids to Spatial Knowledge Acquisition

In order to take advantage of human spatial abilities in user interface design, virtual worlds must be designed not only to maximize our ability to use these abilities to solve tasks, but also to maximize our ability to acquire spatial knowledge. When a user is placed into a new environment, their first task is to explore the environment. In his investigation of spatial knowledge acquisition in 3-D environments, Dr. Avi Parush of Carleton University divided the problem into two components: navigation or wayfinding, and orientation. Orientation described the ability to continually gain and update spatial knowledge through exploration (Parush 376). In his study of how users acquire spatial knowledge in an on-screen virtual environment, Parush studied the effect of various navigation aids such as landmarks, maps, and route lists. Since successful and rapid acquisition of spatial/orientation knowledge is critical for an effective workspace, understanding how the construction of the environment can aid acquisition is important.

Parush found that survey or analog navigation aids such as overhead maps (as opposed to procedural descriptions of navigation steps) made exploration a slower process, but developed a more holistic cognitive map of the environment. Users who used overhead maps as navigational aids were better able than users given procedural route lists to perform tasks in the environment once those aids were taken away (Parush 390). However, when landmarks (distinctive objects with no function within the application—a potted plant for example) were added to the environment, procedural rout lists which referenced those landmarks became equally as effective as those user who employed maps of the environment. These varied findings led Parush to suggest that more adaptive approaches to training spatial knowledge of an environment be employed—taking advantage of either landmarks and specific directions or general overview maps depending on the task and user (Parush 393).

Cognitive research by Thorndyke and Hayes-Roth strongly supports the suggestion to combine these two techniques. Thorndyke classifies spatial knowledge as

consisting of landmark, procedural, and survey knowledge—a scheme which vindicates Parush's use of maps, landmarks, and procedural descriptions to teach users about an environment (Thorndyke 560).

Because the proposed 3-D file browser will allow complete customization of the user's virtual environment, users will be able to set up their virtual space to give themselves landmark clues. The features document included in the appendix lists a number of proposed ways in which users will be able to use their data to orient themselves in the environment. Image files can be turned into pictures to be hung on the wall (or wallpaper or rugs on the floor). Files can be represented as paper pages or books. A radio placed at once corner of the room could play music from the user's mp3 selection. Also, various pre-created furniture pieces can be used both as landmarks and organizational aids. The bookshelves, desks, tables, filing-cabinets, boxes, coffee tables, etc… which the user can place in their environment will not just organize their data—they will also provide visual cues to aid in locating that data.

6.7 Navigation Granularity: Steering and Target-based Travel

All user interfaces for file systems require some form of navigational scheme. In traditional systems, a hierarchical file structure divided into folders or directories is navigated by issuing text commands or selecting icons with a pointing device. These actions move between levels of the hierarchy until the desired files are found. In a 3-D file system interface based on real world interaction, however, navigation has the potential to become much more invasive. Clearly, if the user has to walk a long way or navigate obstacles to reach a desired file, the efficiency of the interface will suffer greatly. If we wish to take advantage of the increased spatial awareness that a 3-D world may provide (see section 3 *Three Dimensional Interface Advantages*), we must provide a lightweight and non-invasive means of navigating through the world (Bowman 96). In other words, means of navigation which feel natural enough to the user that they can concentrate on the task they are trying to perform instead of the individual actions required to navigate the environment. Thus, we wish to apply the principles discussed in section 6.10 *Interface Transparency* to navigation as well as other aspects of the interface.

Bowman divides navigation tasks into three broad categories: exploration, search, and maneuvering (Bowman 96). Exploration is the process of developing spatial knowledge of the environment by moving through it. Searching draws upon this stored knowledge to find a particular location or object. Finally, maneuvering is short-range and high-precision navigation meant to orient the user to make manipulating a particular object easier.

Bowman suggests a number of different navigation schemes including: physical movement, manual viewpoint manipulation, steering, target-based travel, and route planning (Bowman 96).

Bowman's classification of navigation tasks into three categories is intended for a more general virtual reality context where special input and output hardware literally immerses the user in a life-like environment. However, the categorization is still useful in a 3-D desktop file system interface because the user will be faced with many of the same navigational challenges. In order to take advantage of a 3-D space, the user will have to be allowed some form of free movement within that space—essentially the ability to

freely explore. When the user has a specific task or goal in mind, the environment should be arranged so that the target location or object can be easily reached. Finally, when the user finds an object, they must be able to perform useful actions on it. For Bowman, this third component of navigation focuses on positioning the viewpoint to make manipulation tasks (rotating an object, moving it, etc…) simpler. Although tasks performed in a file system user interface will be more function oriented (deleting the file/object, renaming the file/object, opening the file/object, copying the file/object), Bowman's maneuvering component of navigation will still play a role since many of those function oriented tasks will be performed through physical analogies. One of the main reasons for using a real world metaphor is the ability to draw on experience interacting with physical objects. Therefore, allowing the user to easily navigate to a position where object manipulation is simple is a critical concern.

In a 3-D file system interface, it is clear from above that exploration, search, and maneuvering tasks are all important. However, means of navigation which may be useful for exploration of an environment are not as useful when the user need to travel to a specific object to perform a specific task. First, a general navigation system which enables the user to travel to any point in the world is necessary to allow exploration. As suggested in the *Multimodal Interaction* section, "mouse-look" appears to provide the most intuitive and flexible system for general navigation. However, directing every detail of the path taken to reach an object can be inefficient when the destination is much more important than the path taken (when the user is traveling to a specific object and maneuvering him/herself for manipulation). For this type of navigation, target-based travel will be implemented—allowing the user to select any visible object in the scene and be automatically moved to that object. This has the advantage, because of its simplicity, or allowing the "user to focus on the more relevant task" (Bowman 96).

6.8 Object Manipulation: Combining Magic and Metaphor

The lack of a cohesive interface paradigm or interaction metaphor is perhaps one of the most serious hurdles preventing 3-D computer interfaces from becoming more widespread. As has been mentioned in the earlier discussion on the importance of developing a unifying conceptual framework for 3-D interfaces, the lack of an accepted paradigm (similar to the WIMP metaphor for current graphical file browsers), the relative youth of the idea, as well as the significantly larger design space that 3-D offers, all make the task extremely difficult. Applications range from entertainment to CAD modeling, from training simulations to file organization. Input and output technologies are changing at a rapid pace, all bringing new opportunities and challenges. However, by widely researching techniques being used in all these application domains, and further grounding that research in human psychology and human-computer interaction research, but then restricting the problem to specifically file browsing systems for personal computer use, I feel that there is hope of contributing to the development of a new metaphor of interaction.

In this specific case, the problem space is much better defined than the larger problem space of 3-D interfaces in general. For the time being, at least, the output and input devices are fixed, and thus our metaphor will not need to be constantly revaluated as new technologies become available. Yet the domain is not so small as to be insignificant, or supportive of ad-hoc application specific solutions. Section 5 *Current*

Work showed the number and scope of current projects exploring various aspects of this problem space. However it is clear that each of these projects treats the idea of a 3-D file browser very differently. Some, like Sphere XP or Sun Microsystems's Project Looking Glass are merely trying to extend the traditional WIMP metaphor into 3-D. Others, like d3DNA Loft and Rooms 3D actually take an approach similar to the system advocated in this paper.

Examining the development of the now pervasive WIMP metaphor makes it clear that the key role that the "grounding metaphors" behind computer interfaces play in determining "how some possible directions become viable while others fade and eventually disappear" (Hamilton 250). In her study on the importance of metaphoric discourse in information technology, Anne Hamilton of Monash University uses the "desktop" metaphor as an example of a robust conceptual framework which was successful because it gave the users a "set of expectations to apply to computer environments. People readily associated with the folders, calculator, and address book all sitting on their virtual desktop. Hamilton argues that choosing a powerful and informative underlying metaphor is one of the most important factors determining the success or failure of a computer interface.

Therefore, by reducing the scope of my inquiry to file browser interfaces for personal computers, and developing that interface from a familiar metaphor which has already been successful in similar 2-D systems, I hope to lay a strong foundation upon which to construct a consistent, useful, and intuitive interface.

6.8.1 Magic Manipulation

Manipulation and navigation in 3-D, especially an input device as limited as a mouse, is clearly a non-trivial problem. Thus, while employing a "real-world" interaction metaphor may provide benefits beyond traditional WIMP interfaces, there may be benefits to breaking the metaphor to make certain types of interaction simpler. In his overview: "An Introduction to 3-D User Interface Design," Doug Bowman of Virginia Polytechnic Institute specifically points out that "often, nonrealistic techniques have better performance than those based on the real world" (Bowman 107). The combination of a lack of haptic feedback (when manipulating actual objects we constantly rely on tactile feedback from solid surfaces in the environment) and the large number of degrees of freedom needed to accurately position an object, makes it necessary to allow common object manipulations, which might be difficult to perform manually, to be performed by "magic."

One observation about object manipulation, which motivates much of the discussion on "magic" manipulation techniques, is the ease with which we manipulate close-by objects (Mine 3). Thus, a major category of "magic" interaction techniques deals with automatically bringing objects close to the user to facilitate easier interaction. Scaled-world grab is one such interaction technique. Essentially, every time the user selects an object, the entire world except for the user and that object (or that object and the objects around it) are scaled down. The user is then free to manipulate the object and place it back in its distant location, or to drop the object next to the user. Either way, this not only simplifies the manipulation task (as the object has been enlarged and centered on the screen by the scaling), but also eliminates the need for any navigation (the user is stationary—there is no need to walk to the object, pick it up, and walk back again). As a

side note, this method of object manipulation is also useful for navigation. Instead of pulling an object toward oneself, the user can be given the option to pull him/herself to the object. Object centric navigation, and other techniques for easing 3-D navigation challenges are discussed in the navigation section.

Other navigation techniques suggested by mine include “Go-Go” object manipulation—named after Inspector Gadget’s extending arms. The idea is to reduce the navigation load on users by allowing them to interact with any object in the environment that they can see (and not requiring them to physically move their avatar to that object. While this slightly breaks the physical interaction metaphor, the huge savings in efficiency more than compensates. This is especially true since it has already been shown that a large disadvantage of 3-D interfaces is the difficulty of navigation and the possibility for disorientation.

6.9 Imprecise Interaction

Especially when manipulating objects in 3-D, exactly specifying orientations, paths, and positions can be time-consuming and frustrating—especially when constrained to a 2-D input surface (such as a mouse) and a 2-D output device (the computer’s monitor). For some drafting and deign applications this difficulty is unavoidable and must simply be mitigated through complex interaction techniques and input devices. However, as George Fitzmaurice of the University of Toronto discovered while observing subjects sorting colored lego bricks (gaining insight into the “motor-action vocabulary” necessary to describe how users interacted with his Graspable UI). One of his findings was that subjects often used their entire hand as a bulldozer to move entire piles of blocks. Since the individual location of any particular block in the pile did not matter, they could be moved much more efficiently as a group.

For tasks that users would be likely to perform in a 3-D file browser, where the exact location of files is not as important as their relative positions, allowing for simpler but more imprecise interaction could yield large usability gains. Implementing some degree of accurate physics/object collision detection would allow users to move a whole row of adjacent files by simply pushing on the side of one of them. Similarly, users could make room on a table surface by simply sweeping the existing papers onto the floor. This prevents the user from having to take the time to individually determine specifically where each file should go.

6.10 Interface Transparency

A key goal for any user interface is the achievement of *transparency* or the ability of the user to “apply intellect directly to the task” so that “the tool itself seems to disappear” (Rutkowski 291). If a user feels that they are directly involved with “a world of objects rather than communicating with an intermediary” then they truly feel that they are manipulating a representation of reality instead of handling a clumsy tool which carries out their instructions for them (Heckel). Shneiderman even suggests that the closer an interface matches the reality it manipulates, the more users are excited to use that interface (Shneiderman 202). Thus, he argues that interfaces should match the physical environment they manipulate as closely as possible.

The 3-D UI employed by Dr. Shirehjini of the Fraunhofer Institute for Computer Graphics to control media devices in a conference room is an excellent example of

interface transparency through matching interface form to function (Shirehjini 669). In order to solve the problem of controlling the myriad projectors, lights, interactive TVs, audio devices, etc… in a modern media room, Shirehjini developed a palm pilot like *Personal Environment Controller* which, when held in front of a user in the room, displayed a virtual 3-D image of the portion of the room which the user was facing. The elements of this 3-D image (the identical physical analogues of which lie right beyond the user) can then be clicked on and modified. Files can be loaded onto projectors or TVs simply by dragging the file from one device in the virtual image to another. This extension of drag-and-drop from the traditional user interface to interaction directly between computers and the physical devices which they ultimately control makes interaction easy because the interface becomes transparent. Putting a file up on a monitor is no harder than slapping a transparency on an overhead projector.

The same basic principle which underlies the *Personal Environment Controller* is one of the basic principles behind using real world metaphors in 3-D file system interfaces. By providing a conceptual bridge between physical world and the virtual world, we can develop an interface whose objects behave exactly like the objects we interact with every day. As Shirehjini suggests, this makes interaction with the computer system easier and more intuitive. Of course there are differences between Shirehjini's application domain and ours. When developing an interface to explicitly control physical objects in a 3-D space, having the interface be a virtual model that space provides clear benefits. However, a file system user interface manipulates objects which have no physical representation. Therefore achieving a transparent file system interface could be considered impossible since the interface itself defines the representation of the objects being manipulated. There is no physical analogy to model. For this reason, metaphor we use to represent files on a computer is of critical importance. It defines how we conceptualize and interact with the data on our computer. The window, icon, menu, pointer (WIMP) interface paradigm makes sense in this light as an attempt to provide familiar physical representations to computer data. Related chunks of computer data became files, shown on screen as tiny sheets of paper—a physical analogy for a discrete bit of information. Files could be grouped in directories, represented as folders on the screen—again providing a clear physical analogy which gave strong suggestions as to their function.

Clearly the designers of the WIMP interface metaphor were onto something. Today essentially all modern operating systems provide some similar type of interface. This success must be due at least in part to the power of the metaphor to provide a convincing enough analogy with the real world and therefore achieve some degree of interface transparency. We are able to move data around in the file system simply by dragging icons from one folder to another, mimicking the steps one might go through in the physical world to achieve a similar result. The idea of an immersive 3-D file system attempts to expand on this interaction metaphor by making interaction with data even more like interaction with physical papers and files. Ideally this change would allow the 3-D interface to actually *become* the work environment, not simply an interface used to retrieve files. Complete interface transparency, achieved when the user of a system can accomplish a task by simply performing the logical steps needed to achieve that task (instead of transforming those logical steps into actions which must be performed using the interface in order to accomplish the task), might become an achievable goal.

6.11 Combine 2-D/3-D Interaction to Limit Degrees of Freedom

Simply because an interface is 3-D does not mean that all tasks performed by the user must be performed in three dimensions. In many cases, the added degrees of freedom are simply unnecessary and distracting. In his discussion of 3-D interface design, Bowman even concedes that "by taking advantages of both 2-D and 3-D interaction techniques, we can create interfaces for 3-D applications that are easier to user and more intuitive for the user" (Bowman 96). Ken Hinckley of the University of Virginia also notes that "the general issue of constructing *hybrid interfaces* which combine 2-D and 3-D interaction in a unified framework…remains largely unexplored" (Hinckley 129).

In fact, one of the major design issues facing 3-D development is the rather unintuitive observation that "people do not innately *understand* three dimensional reality, but rather they *experience* it" (Hinckley 214). In other words, we are very good at observing three dimensional objects and discovering spatial relationships between objects through moving those objects around and experimenting how they interact with each other, but we are very poor at simply visualizing a 3-D scene or relationship if we have no way to experience/interact with it. Hinckley supports these general claims with studies performed by Shepard-Metzler which concluded that in order for most people to visualize an object for various perspectives, we must actually imagine a rigid body rotation occurring on that object. Most people cannot simply look at an object and know what it will look like in another orientation—they must (at least in their head) actually rotate the object. In his research, Hinckley uses these conclusions to underscore the importance of advanced interaction and feedback techniques like two-handed interaction, multi-sensory feedback, head tracking and more. All these techniques are aimed at giving the user of a 3-D virtual reality system the tools needed to overcome some of the inherent difficulties humans have with understanding complex 3-D relationships without the aid of the rich auditory and haptic feedback that the real world provides us. However, when constrained to viewing a 3-D environment on a flat monitor and interacting with that environment using only a mouse, the observation about how humans experience/understand 3-D reality may actually lead us toward *less* complex interaction techniques.

For example, one possible conclusion from this observation that people do not actually readily understand three dimensional reality is that we should actually limit the tasks which users must perform in three dimensions. Many tasks that users would perform with a file browser may be made much more cumbersome if the user must constantly be interacting with a 3-D environment. When choosing a specific file from a stack of papers sitting on the user's virtual desk, the 3-D spatial queues would help the user remember which stack he/she wanted to look in, but when actually choosing the specific file, a 2-D interface might be simpler. For example, the user could click on the pile to cause the papers to spread out in a 2-D grid directly in front of the user's screen. This would ensure none of the papers occlude each other (ensuring that the file names or thumbnails are always visible) and clearly indicate to the user that a choice is required.

Navigation techniques also benefit from a reduction in the degrees of freedom given to the user. Even when navigating a real room, we are essentially limited to two dimensional movement along the floor. Therefore, it makes sense in a three dimensional simulation of an actual room, that the user's movement would be similarly constrained.

Most of the work done with hybrid interfaces combines a physical 2-D input device with a 3-D immersive virtual reality environment. Mark Mine of the University of

North Carolina mentions multiple studies of tablets being used as a drawing/manipulation surface (Mine 25). In advocating limiting the degrees of freedom available to a user, Hinckley also notes that 2-D surfaces make excellent natural restraints to a user's freedom of motion. He lists "the physical surface of the user's desk, the glass surface of the user's monitor, [and] a hand-held palette or clipboard" as possible constraining 2-D surfaces (Hinckley 216). Ali. Shirehjini of the Fraunhofer Institute for Computer Graphics also employs the 2-D surface of a tablet PC as the input device for his 3-D multi-media room control interface. In Shirehjini's case, not only does the device allow simple selection of objects in the 3-D world, but also allows the user to change his/her view of the virtual world simply by changing the physical location of the tablet.

The situation to which I want to apply this concept is slightly different than the above examples. My proposed 3-D file browser will not be an immersive virtual reality environment and it will not use any specialized input devices. Output will be provided through the computer systems' 2-D screen and the mouse and keyboard will serve as input devices. Therefore, already the range of motion of the user is restricted by the surface of the desk upon which the mouse rests. However, the basic concepts of the hybrid interface and of simultaneously taking advantage of the simplicity of 2-D and spatial intuitiveness of 3-D are still important.

A critical design consideration will be the extent to stick precisely to the three dimensional physical room metaphor (for both interaction and navigation), and when to use "magic." By which I mean either the environment reacts in ways which a physical environment would not (objects lighting up to indicate they are selected for example) or the user is able to navigate the environment in ways he/she otherwise could not (teleporting from place to place for example). The strictness with which the physical room metaphor should be applied (at least in terms of the means of interacting with the environment) is the subject of the *Object Manipulation* section, but our choice of 2-D versus 3-D is another place where this concept applies. Although it may dilute the physical room metaphor to have a stack of papers present themselves for selection in a line floating in the air in front of the user, it may actually make the interface more intuitive.

6.12 Integrated Creativity Support and Data Visualization Tools

In his article "Creating Creativity: User Interfaces for Supporting Innovation," Ben Shneiderman proposes a user interface framework for supporting and expanding user creativity which he calls the genex framework (for: generator of excellence). It proposes that creative work consists of four phases, and Shneiderman claims that it is important because by understanding how creative people function, user interfaces can better support creativity. The four phases of the genex framework are:

- Collect: learn from previous works stored in libraries, the Web, etc.
- Relate: consult with peers and mentors at early, middle and late stages
- Create: explore, compose, and evaluate possible solutions
- Donate: disseminate the results and contribute to the libraries
  (Shneiderman 120)

Shneiderman explicitly states that the goal of the framework is "to suggest improvements to Web-based services and personal computer software tools…by reducing the distraction caused by poorly designed user interfaces" (Shneiderman 122). Essentially, he states

what we already know: searching the World Wide Web for relevant information is *hard*. Oftentimes finding files we need on our own computer is likewise hard. This is because interfaces and the data they present are not organized as effectively as they could be.

One of the most intriguing suggestions that Shneiderman makes based on this framework is actually a return to the consistent theme of creating a unifying conceptual framework. In this instance Shneiderman argues that common, compatible sequences of actions such as cut-copy-paste or open-save-close are very powerful, allowing smoother coordination and better interaction when users are working with a variety of different programs or systems. He then takes this concept a step further and suggests that higher level/more abstract actions, such as "annotate-consult-revise" or "collect-explore-visualize" could be standardized in a similar manor. Essentially, Shneiderman is arguing for the actions performed with a computer to be more associated with and defined by the specific tasks we are performing. Thus, instead of thinking of creating a research paper in terms of opening programs, entering search terms, copying relevant data, saving files, etc... the interface would instead abstract these actions. For example, he suggests that if users "see an unfamiliar term they should be able to click it and get an English definition, a French translation, or a medical dictionary report, all in a predictable screen location" (Shneiderman 122). Essentially, interfaces and databases like the web should have more meta-data associated with information so that they can provide intelligent context related to the users task. Interfaces should allow the user to focus on the task, not series of actions needed to accomplish that task.

Shneiderman concludes his discussion with the suggestion that creativity support tools built into user interfaces should allow at least eight main activities that support the genex framework for creative work. His list includes:

- Searching and browsing digital libraries
- Consulting with peers and mentors
- Visualizing data and processes
- Thinking by free associations
- Exploring solutions—what-if tools
- Composing artifacts and performances
- Reviewing and replaying session histories
- Disseminating results

(Shneiderman 123)

Finally, and perhaps most relevant to the motivations behind this project, is Shneiderman's conclusion that "visualizing objects and processes is the activity that seems most pervasive and could appear in every genex phase" (Shneiderman 124). As will be discussed in the *Applicability and Context* section of this principle, one of the main advantages to a 3-D file browser is the enhanced visualization aids it could provide.

Visualizing objects is easy, however, compared to visualizing processes and the relationships between objects. Shneiderman directly correlates creative thought with the ability to form new associations between existing ideas (Shneiderman 126). He lists examples of numerous visualization tools such as MindManager (www.minman.com), The Brain (www.thebrain.com), IdeaFisher (www.ideafisher.com) and more. All of these serve essentially as "dictionaries of associations" with the intent of stimulation creative thought. This leads to the question: what do people want to know about the associations between files on their computers? Also, how can the tried and true visualization

techniques used in these tools and others be applied to finding and presenting the associations between a users files? Although much more study and research needs to be put into this area, see the *Problem Statement* and other design documents for the proposed file browser in the *Appendix* section for ways in which file associations could be identified and exploited. With a more attribute based approach to file browsing already envisioned (see section 5.2.2 *Presto/Vista*) it is also possible that comparisons between these attributes could be used to easily find associated files.

Creativity support and visualization tools are becoming more and more common today. One especially interesting example, however, which suggests interesting possibilities for the proposed 3-D file browser, is a Multimedia Bulletin Board described by Hyunmo Kang at the University of Maryland. The stated purpose of the project is to produce an "asynchronous communication system that enables rich communication and collaboration between users of multimedia objects such as text, image, moving picture, sound, voice, web, office document, and other files" (Kang 51). Essentially, the system allows users to post any of the data listed above in a 2-D space, allowing the x-y position, size, partial occlusion, explicit links, and implicit links based on spatial proximity of data to convey information about the "user dynamics" occurring on the board. However, this system suffers a fundamental conflict between two opposing requirements. First, the system wishes to use the "random" user-defined organization of the board to convey information which might be impossible to otherwise represent. Essentially, the users are able to create meaning through the organization of their space (Kang 51). Second, users should be able to quickly identify specific and useful information—a task which may be made difficult or impossible with bits of data spread all across the bulletin board. This system's innovative solution to this problem was to allow the users to dynamically rearrange the contents of the board using predefined or custom dynamic layout templates which would automatically arrange the data according to some criteria. At any time, the user could go back to the "natural" layout of the board (Kang 51). Both this multimedia bulletin board and a 3-D file browser deal with organizing and arranging data which may be poorly organized into a useful and understandable format. Thus, we may be able to apply many of the lessons learned from data visualization systems like the Multimedia Bulletin Board directly to 3-D file systems.

In his discussion of how the genex framework can be applied to actual user interfaces, one of the key considerations that Shneiderman addresses is the visualization of data. Specifically, he claims that while data *searching* capabilities should be improved (via meta data tagged to files using technologies such as XML), rapid data browsing to support less specific *exploration* of a topic. He likens the power of this type of searching to browsing shelves at a library or bookstore. One may not be looking for a specific book, or may have a general topic or set of criteria in mind, but not enough to generate explicit search terms—but the layout of a bookstore allows rapid scanning of hundreds or thousands of possibilities (Shneiderman 125). This ability is exactly the kind of power that a 3-D representation of computer files and directories would provide. By taking advantage of the user's peripheral vision and laying out data in structures like bookshelves and piles, we essentially increase the amount of data which can be scanned by the user. By using physical analogies (like bookshelves or filing cabinets) we are building off of a system which has proved successful and useful at least in physical contexts.

The situation of the multimedia message board is also very closely tied to that of a 3-D file system. Personal computer data is often very randomly entered and stored, often making retrieval of particular files quite a difficult task. Another issue in implementing a 3-D file system is the initial placement of a user's files within the world. Perhaps the solution to both of these problems is an approach similar to the multimedia bulletin board. Have a basic "free-form" layout where the user can position files as he/she sees fit. Perhaps there will be a default location where newly created files are placed (an "inbox" of sorts). The user would then also be able to access a number of preset arrangements (similar to the "dynamic layout templates which Kang discusses) which would automatically rearrange the room (virtual furniture and containers as well as files) based on some predefined organization. Perhaps the files would be sorted from largest to smallest along the north-south axis of the room and alphabetically by name along the east-west access. Or, files could be organized radially about the user, with file attributes changing based on radial distance and angle. Because of the 3-D room metaphor, more unique automatic arrangements could also be implemented. Perhaps all the files in the room could be arranged into stacks or boxes by file name or type. The shelves on a bookshelf could be rearranged based on different criteria.

6.13 Context Awareness for Task Automation

One of the powerful impetus behind the development of pervasive computer interfaces (interfaces which extends beyond the computer system, drawing information from the physical world through cameras and sensors) is that they allow for extensive task automation through increased awareness of the physical context of the user. For example, a multimedia room controlled by a context aware pervasive computer system could sense a person standing at the front of the room by the projection screen and automatically dim the lights and turn on the projector in preparation for their presentation. If that person had a device like a PDA, the room could communicate with that device to download and display the presentation (a situation not unlike the Multimedia Room Controller discussed in the *Interface Transparency* section).

Context aware systems can be as large as a room, or be a much more confined workspace. Hideki Koike of the University of Electro-Communications, for example, developed an "augmented desk" which uses cameras and sensors to track documents and hand motions on the desk and a projector to display digital information pertaining to the objects and actions being performed on the desk. This linking of physical and virtual objects is another form that the context provided by pervasive computer systems can take (Koike 310).

In "An application of a context-aware file system," Christopher Hess of the University of Illinois lists three main advantages of a file system with access to contextual information (like the location of its users, their identity, the activity they are performing, the time, and the devices they are using):

> "[Contextually aware systems can:] 1) automatically make personal storage available to applications, conditioned by user presence, 2) organize data to simplify locating data important for applications and users, and 3) retrieve data in a format based on the context of user preferences or device characteristics" (Hess 341).

It appears on initial consideration that “context aware” systems and “pervasive” systems are one in the same and thus that the ideas cannon really be applied to a file browser which will be run completely internally on a personal computer. However, since the user of the proposed 3-D browser will be moving around the virtual file system space in the same manner as they might move around an actual room, I believe it may be possible to make use of their context in the virtual world to provide task automation.

By combining information about a user’s location in the environment, with tags on the files in the system similar to those proposed by the Presto file management system (see *Presto/Vista* section), a 3-D file system could imitate to some extent the behavior of a pervasive context aware system without the hardware and expense of outfitting a room with sensors. The system could, for example, simply remember the user’s position in the world (and perhaps the time and data) when a file was accessed, and then provide a smart “quick links” window which would suggest files that the user is likely to access based on their current position in the world.

6.14 Flexible and Task-Based File Organization

The basic organization of data on a hard disk—as essentially a tree data structure with directories indicating branches and data files representing the tree’s leaves—has not changed since the 1970’s. Of course the schemes for keeping track of the ever growing amounts of data we store have become much more complicated and advanced, but the underlying idea of a file hierarchy still persists. However, as Gary Marsden of the University of Cape Town points out, there are two major flaws with this system: it is unintuitive for all but the most advanced computer users, and it is extremely inflexible (Marsden 122). Marsden points to the inclusion of application default folders such as the “My Documents” folder as “hacks” to work around the flaws of a hierarchical file structure. The “My Documents” folder essentially creates a space for users to dump their files, often just all in the main level or in a very shallow hierarchy, betraying the fact that it is either too much work to create an efficient directory structure, or that most users do not know how (Dourish 135). Alternatively, this behavior could result from the inherent inflexibility of a hierarchical organization. Even if the user takes great care to create a meaningful file hierarchy, those files can only be searched through according to those characteristics that the hierarchy was based on. For example, if you create a directory hierarchy of MP3 files based on the music genre, then want to find the newest file in your collection, there is no simple way to reorganize your hierarchy to accommodate this task.

Marsden, as well as the designers of systems like Presto (see *Presto/Vista* section) and Hess’s context aware file system (see *Provide Context Awareness for Task Automation* section) and others agree that the best solution to this problem is to tag files with meta data and allow the file system to function more like a traditional relational database. Instead of organizing data in one set fashion, data views should be flexible and able to change based on the current task. The problem, however, as Marsden points out, is the creation of the meta data in the first place (since users may tend to not take the time to add meaningful attributes, making such a system about as useful as the “My Documents” folder).

7. Proposed First Person File System Description

The motivation behind the creation of the above 3-D design principles list, as well as the enumeration of possible advantages of 3-D interfaces and the review of existing 3-D interfaces was to eventually create a prototype file browser system. The Appendix section 12.2 contains a detailed problem statement document with a explicit feature list and analysis of the form/function/economy/time factors. This section is intended to give a higher level summary of the basic components of the design.

[section currently incomplete—expand high level description--place features doc in appendix--look over methods for specifying UIs, choose one]

8. Implementation Details

8.1 Java3D

The prototype system was developed using Java3D technology. It extends the capabilities of Java which allow rapid development of cross-platform applications which can be easily deployed over the web to include interactive 3-D graphics. Java3D is an API which provides the programmer with high level constructs for describing a scene, and then uses a lower level graphics API such as Direct3D or OpenGL (there are Java3D implementations using both these APIs as a back end) to actually render the scene. Java3D uses a scene-graph programming model. The programmer builds the 3-D world as a series of tree nodes which represent the geometry, appearance, lighting, affine transformations, and other aspects of the scene. Java3D also abstracts the interface between the scene description and the output device. Thus, Java3D programs can be viewed with stereoscopic 3D goggles just as easily as on a traditional computer monitor. As Aaron Walsh summarizes in *Java 3D API Jump-Start*, "Java3D allows developers to focus on *what* to draw, not *how* to draw." This makes it an ideal platform for rapid application development and prototyping, and thus makes it ideal for creating a quick functional demonstration mockup of the proposed 3-D user interface.

Functionally, Java3D acts as a layer on top of more native low-level rendering APIs like OpenGL and Direct3D. While it does allow the programmer to specify geometry by building from individual triangles and specifying normals and texture coordinates, it is more useful in its ability to abstract away most of those rendering details. Also, it allows for easier integration of 3-D content into existing traditional java applications (simply add a 3DCanvas Component object to any JFrame). Compared to other scene graph APIs such as OpenInventor and Performer, Java3D also provides a number of advantages. According to Doug Gehringer of Sun Microsystems, Java3D is able to perform far more scene graph optimizations than OpenInventor because its scene graph structure ensures that the state of a node is effected only by the nodes above it in the tree. Java3D also provides better portability and can be used on a larger number of systems than Performer, which is only available on SGI and Linux platforms. Java3D uses an extensible file loader interface to allow the importing of 3D content in a variety of different file formats, make content creation easy and flexible.

The Java3D scene graph is a directed acyclic graph: edges between nodes form a parent child relationship and the graph structure cannot contain loops. Nodes may, however, share certain aspects of their appearance and geometry. The basic tree structure consists of Group nodes which act as containers for sets of nodes. Environment nodes

model effects like light, fog, sounds. Transform groups apply transformations to the all the geometry below it in the scene graph. In general, nodes may have only one parent. However, special shared groups are supported which may be a child of multiple groups through special link nodes. Various capability bits associated with the nodes control which aspects of the node can be modified, allowing Java3D to perform optimization of the scene graph based on the contract set by the capability bits that certain aspects of the scene graph will not be modified.

8.2 Application Details

The following sections elaborate on specific technologies, resources, API's, etc… employed in the construction of the prototype. It is intended to give the reader an overview of how the implementation was accomplished.

8.2.1 File Browser Interface

The file browser will get information about the underlying file structure on the disk through Java's system independent API for accessing file and directory information. The Java File object will then be wrapped into a File3D class which extends Java3D's BranchNode class and thus can be placed directly into the scene graph. The File3D class will contain information needed by Java to access the file through the underlying file system as well as the information needed about the position of the file's representation within the file browser UI.

8.2.2 Collision Control

Collision control is implemented using an extension of the picking supported provided by Java3D and is based on sample code provided in Sun Microsystems's "FlyThrough" sample program. When the camera is moved, if collision detection is enabled, the current position and the desired new position are passed into a collision detection controller class which has been registered with the scene graph and has rights to read information about the geometry of the various objects in the scene. The vector between the current and desired position is used as an input to Java3D's picking methods. These methods return information about the geometry which intersects with the provided ray. This information is then possibly used to modify the desired position by only allowing it to move up to the boundary of the nearest geometry object and no farther.

The height of the viewer is also calculated in a similar manner. A vector is projected down from the current position of the viewer and the first object which that vector intersects with is taken to be the 'floor.' The viewer's height is then set as a static distance above the floor.

This straightforward collision control implementation has a number of drawbacks, but the important advantages that it is simple to implement and provides the basic functionality necessary to create a convincing virtual world. Some notable drawbacks include that collision detection is checked only when the viewers position changes. Therefore, moving objects cannot 'collide' with the viewer; it must collide with other objects. However, since most objects in the scene will not move on their own (only when they are manipulated by the user), this is not a serious drawback.

8.2.3 File Data Management

The prototype implementation of the proposed 3-D file browser will be a user interface layer only. It will rely on the underlying file system to actually keep track of files on the hard disk. While this will limit the ability to implement some of the more advanced "Presto-like" features (see section 5.2.2 *Presto/Vista*) which would require an completely different and more database-like file storage paradigm to be implemented effectively, the first stages of the prototype system developed for this research will not incorporate these features anyway.

However, this means that the system will have to store its own meta-data for each file which it is keeping track of (information such as the position of the file's representation in the UI) in its own data structure as opposed to integrating this information into the file itself (which would be ideal). However, the Java serialization functionality can be used to simply store the entire Java3D scene graph (by serializing the root SimpleUniverse node). While this method is clearly not the most efficient approach, it should provide an adequate solution for the prototype system. All the meta-data about the position of the various files and the objects which they represent will then be stored in the scene graph nodes themselves.

8.2.4 Native System Calls

Although Java does contain an interface to the local file system which allows java programs to access and manipulate files and their contents, it does not provide a similarly simple mechanism for opening other native applications. This necessary because the Java-based 3-D file browser must be able to open the programs necessary to read various supported file formats. This functionality will be accomplished using the Java Native Interface (JNI) which is a standard part of the Java SDK. A small portion of c code will be written for interfacing with Windows and opening the specified file with its default application. This code will then be called from within Java using the JNI framework.

8.2.5 Image Loaders

Java3D provides an extensible API for creating loaders which allow 3D geometry files in various formats to be imported into Java3D. Although programmers can create their own loaders if necessary, there are a large number available for typical file formats. Both Sun's Java3D FlyThrough package and the Web3D Consortium's Xj3D Browser contain loaders for VRML objects. There are also a large number of freely available VRML objects available on the Princeton 3D model search engine (http://shape.cs.princeton.edu/search.html) and other sites. This allowed for very rapid content creation.

9. Success Criteria and Evaluation Metrics

The proposed 3-D file system was intended to solve the specific problems with file organization and recall discussed in the introduction. However, it was also intended as a test of the effectiveness of the interface guidelines developed through this research. Therefore, having an effective plan to measure the success of the prototype system was quite important.

9.1 Expert Review and Usability Testing

In choosing a review technique, a number of different options were considered. For example, in *Designing The User Interface* Ben Shneiderman begins his discussion of user interface evaluation with various “expert” or “peer” evaluation techniques. This type of evaluation by experts knowledgeable in the user interface field will occur through the thesis evaluation process and has been occurring iteratively through conversations with Dr. Mellor, my research advisor.

Informal expert reviews must be coupled with a more formal usability test of the system (once a sufficiently complete prototype is developed). When discussing usability testing, Ben Shneiderman makes a clear distinction between academic testing under controlled conditions attempting to validate or reject a hypothesis at a given confidence level and usability testing intended to develop lists of recommendations to improve a specific system. This usability test plan falls into the second category. It attempts to get a general feel for the usability of an entire system. This will be accomplished by developing a specific set of tasks for user to complete using the prototype system. These tasks will involve organizing a given set of files, searching for specific files, performing basic manipulations on the file organization (copying, moving, and deleting files), getting information about individual files or groups of files. In short, the task list will attempt to cover all basic tasks that a user would perform with a modern file browser. While completing these tasks, a *think-aloud protocol* will be encouraged. Before beginning the tasks users will be asked to verbalize their thoughts (which will then be recorded). Often this informal, spontaneous, and on-the-spot feedback mechanism generates farm more numerous suggestions for improvements than a post-experiment survey (Shneiderman 147). After the users have completed the task list, they will then complete a survey rating various aspects of the system on a 1-9 scale where range represents a value between two bipolar semantically anchored items (pleasing versus irritating for example). Finally, each section of the survey will also contain a section for freeform comments asking the user about the most and least useful parts of the interface.

9.2 Comparative Testing

Any alternative user interface must inevitably be compared against the standard hierarchical file browser with a WIMP interface. The goal of this comparative testing will be to determine both which user interface users prefer in a qualitative evaluation of measures such as “ease of use” and “enjoyment” as well as measuring the users efficiency. The comparison of efficiency will be performed by giving users a specific set of data and a number of tasks to perform with that data. They will perform the same tasks using a baseline WIMP hierarchical file browser (yet to be chosen) and the prototype of the proposed system. Quantitative metrics such as the time to complete each task and the number of mouse clicks required will be measured during these trials.

This comparative test against a baseline traditional hierarchical file system will be a critical test of the claims and research presented in the Introduction section. Problems with how hierarchical file structures are used today were presented as motivation for seeking alternative ways to display, organize, and access computer data. Thus, comparing these alternative browsers against traditional browsers will provide critical data on the correctness of these motivating claims.

The exact nature of the metrics collected will be determined in part by the nature of the tasks chosen for the test. However, they will need to be numerical data which is readily collected and gives some measure of the effectiveness of the interface in allowing the user to complete the assigned tasked. Metrics such as the time to complete a task or even the number of mouse clicks required might be useful in this regard.

10. Conclusions and Future Work

Although the motivating idea behind this research was the development of a 3D file browser using a physical interaction metaphor, much research and effort was spent developing the theoretical framework to develop such a system. The enumerated principles, motivations for and advantages of 3D interfaces are all direct results of this work.

Development is currently progressing on the prototype system based on these principles. The system is being implemented using the Sun's Java3D API. Comparative usability tests will provide a numerical comparison of the system against a baseline hierarchical file browser using metrics like task completion time. Qualitative usability data will also be gathered through user surveys.

11. Works Cited

"3d-Desktop." *SourceForge.net.* 2005. http://desk3d.sourceforge.net/ (5 Jan 2005).

"3DNA Desktop." *3DNA.* 2005. http://www.3dna.net/products/desktop.htm (5 Jan. 2005).

"Home of the 3D Chat, Virtual Reality Building Platform." *Activeworlds.* http://www.activeworlds.com/ (29 Oct. 2004).

"Project Looking Glass." *Sun Microsystems.* 2005. http://www.sun.com/software/looking_glass/ (5 Jan 2005).

"3DOSX" *Mac Warriors.* 2005. http://www.acm.uiuc.edu/macwarriors/projects/3dosx/ (5 Jan 2005).

"Tiger: Change Your Stripes." *Apple Computer.* 2005. http://www.apple.com/macosx (5 Jan. 2005).

"Google Desktop." *Google.* 2005. http://www.desktop.google.com/ (5 Jan. 2005).

"Rooms 3D Desktop." http://www.rooms3d.com.sg/index.html (5 Jan 2005).

"SphereXP." http://www.hamar.sk/sphere/ (5 Jan 2005).

A. Cockburn and B. McKenzie. "Evaluating spatial memory in two and three dimensions." International Journal of Human-Computer Studies 61.3 (2004): 359-373.

Avi Parush and Dafna Berman. "Navigation and orientation in 3D user interfaces: the impact of navigation aids and landmarks." International Journal of Human-Computer Studies 61.3 (2004): 375-396.

Beaudouin-Lafon, Michel. "Instrumental interaction: an interaction model for designing post-WIMP user interfaces." Conference on Human Factors in Computing Systems The Hague, The Netherlands: ACM Press, 2000. 446-453.

Bederson, Benjamin B. "Interfaces for Staying in the Flow." University of Maryland Human-Computer Interaction Lab 2003.

Ben Schneiderman and Catherine Plaisant. "Designing the User Interface." Boston: Addison Wesley, 2005.

Ben Shneiderman and Harry Hochheiser. "Universal Usability as a Stimulus to Advanced Interface Design." Behaviour & Information Technology 20.5 (2001): 367-376.

Benjamin B. Bederson, Jon Meyer, and Lance Good. "Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java." CHI Letters 2.2 (2000): 171-180.

Bowman, Doug A. "An Introduction to 3-D User Interface Design." Presence: Teleoperators & Virtual Environments 10.1 (2001): 96-109.

Brooks, Frederick P. Jr. "Three Great Challenges for Half-Century-Old Computer Science." Journal of the ACM 50.1 (2003): 25-26.

Brown, Iain M. "A 3D user interface for visualisation of Web-based data-sets." 6th ACM international symposium on Advances in geographic information systems Washington, D.C., United States: ACM Press, 1998. 100-105.

Chin, Robert. "Three-dimensional file system browser." Crossroads 9.1 (2001): 16-18.

Cockburn, Andy and McKenzie, Bruce. "3D or not 3D?: evaluating the effect of the third dimension in a document management system." Conference on Human Factors in Computing Systems ACM Press, 2001. 434-441.

Czerwinski, M. "The contribution of thumbnail image, mouse-over text and spatial location memory to web page retrieval in 3D." Proceedings of the Interact '99 Edinburgh, Scotland: ISO Press, 1999. 163-170.

Dourish, Paul. "Presto: an experimental architecture for fluid interactive document spaces." ACM Transactions on Computer-Human Interaction 6.2 (1999): 133-161.

Faichney, Jolon and Gonzalez, Ruben. "Goldleaf hierarchical document browser." 2nd Australasian conference on User interfaces 13

Foley, James. "Future directions in user-computer interface software." ACM Press, 1991. 289-297.

George W. Fitzmaurice,Hiroshi Ishii, and William Buxton. "Bricks: Laying the Foundations for Graspable User Interfaces." Conference on Human Factors in Computing Systems Denver, Colorado, United States: 1995. 442-449.

Gorayska, Barbara. "Cognitive Technology: In Search of a Human Interface." Amsterdam: Elsevier, 1996.-420.

Grudin, Jonathan. "The case against user interface consistency." Communications of the ACM 32.10 (1989): 1164-1173.

H. Rex Hartson and Deborah Hix. "Human-Computer Interface Development: Concepts and Systems for Its Management." ACM Computing Surveys 21.1 (1989): 1-88.

Hamilton, Anne. "Metaphor in Theory and Practice: the Influence of Metaphors on Expectations." ACM Journal of Computer Documentation 24.4 (2000): 237-253.

Heckel, Paul. "The Elements of Friendly Software Design: The New Edition." San Francisco: SYBEX, 1991.

Henderson, D. Austin Jr. "Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface." ACM Transactions on Graphics 5.3 (1986): 211-243.

Hess, Christopher K. and Campbell, Roy H. "An application of a context-aware file system." Personal and Ubiquitous Computing 7.6 (2003): 339-352.

Hetherington, Robina E. "Adding a fourth dimension to three dimensional virtual spaces." 3D technologies for the World Wide Web Monterey, California: ACM Press, 2004. 163-172.

Hiroshi Ishii and Brygg Ullmer. "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms." Conference on Human Factors in Computing systems 1997. 234-241.

Jean Vanderdonckt, Chow Kwok Chieu, Laurent Bouillon, and Daniela Trevisan. "Model-based Design, Generation, and Evaluation of Virtual User Interfaces." 3D technologies for the World Wide Web Monterey, California: ACM Press, 2004. 51-60.

Kang , Hyunmo and Ben Shneiderman, and Gregory J Wolff. "Dynamic Layout Management in a Multimedia Bulletin Board." Proceeding of IEEE 2002 Symposia on Human Centric Computing Language and Environments 2002. 51-53.

Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. "A survey of design issues in spatial input." Symposium on User Interface Software and Technology 1994. 213-222.

Marcus, Aaron. "Metaphor design in user interfaces." ACM SIGDOC Asterisk Journal of Computer Documentation 22.2 (1998): 43-57.

Mark R. Mine, Frederick P. Brooks Jr., and Carlo H. Sequin. "Moving objects in space: exploiting proprioception in virtual-environment interaction." International Conference on Computer Graphics and Interactive Techniques 1997. 19-26.

Marsden, Gary. "Improving the usability of the hierarchical file system." ACM International Conference Proceeding Series South African Institute for Computer Scientists and Information Technologists, 2003. 122-129.

Olwal Alex and Feiner Steven. "Unit: modular development of distributed interaction techniques for highly interactive user interfaces." Ed. Computer graphics and interactive techniques in Austalasia and South East Asia ACM Press, 2004. 131-138.

Preece, Jenny, et al. "Human-Computer Interaction." Edinburgh Gate: Addison-Wesley Longman Limited, 1994.-775.

Robert J. K. Jacob, Leonidas Deligiannidis, and Stephen Morrison. "A Software Model and Specification Language for Non-WIMP User Interfaces." ACM Transactions on Computer-Human Interaction 6.1 (1999): 1-46.

Rodruguez, Jesus. “WinFS Data Model.” http://www.c-sharpcorner.com/Longhorn/WinFS/WinFSDataModel.asp (15 Jan. 2005).

Rutkowski, Chris. "An introduction to the Human Applications Standard Computer Interface, Part 1: Theory and Principles." Byte 7 11 (1982): 291-310.

Schwesig, Carsten. "Gummi: user interface for deformable computers." Conference on Human Factors in Computing Systems Ft. Lauderdale, Florida: ACM Press, 2003. 954-955.

Sciammarella, Eduardo. "VisualFlow a media browser." Conference on Human Factors in Computing Systems Seattle, Washington: 2001. 33-34.

Serollipinho, Márcio et al. "A User Interface Model for Navigation in Virtual Environments." Cyberpsychology & Behavior 5.5 (2002): 443-449.

Shinyama, Yusuke. “XCruiser.” *http://xcruiser.sourceforge.net/* (5 Jan 2005).

Shirehjini, Ali. A. Nazari. "A novel interaction metaphor for personal environment control: direct manipulation of physical environment based on 3D visualization." Computers & Graphics 28.5 (2004): 667-676.

Shneiderman, Ben. "Creating Creativity: User Interfaces for Supporting Innovation." ACM Transactions on Computer-Human Interaction 7.1 (2000): 114-138.

Shneiderman, Ben. "Designing the User Interface: Strategies for Effective Human-Computer Interaction." Reading, Massachusetts: Addison-Wesley Publishing Company, 1992.-573.

Stuart E. Middleton, Nigel R. Shadbolt, and David C. De Roure. "Capturing interest through inference and visualization: ontological user profiling in recommender systems." International Conference On Knowledge Capture 2003. 62-69.

Thorndyke, P., and Hayes-Roth, B. "Differences in spatial knowledge obtained from maps and navigation." Cognitive Phychology (1982): 560-589.

Udell, John. "Space, Time, and Data." Infoworld.com 2004.

Vanderdonckt, Jean. "Model-based design, generation, and evaluation of virtual user interfaces." 3DTechnologies for the World Wide Web Monterey, California: ACM Press, 2004. 51-56.

Vildan Tanriverdi and Robert J.K. Jacob. "VRID: A Design Model and Methodology for Developing Virtual Reality Interfaces." ACM Symposium on Virtual Reality Software and Technology Banff, Alberta, Canada: 2001. 175-182.

Weiss, Greg. “Nooface: In Search of the Post-PC Interface.” *http://nooface.com/* (29 Oct. 2004).

Y. Lin and W. J. Zhang. "Towards a novel interface design framework: function–behavior–state paradigm." International Journal of Human-Computer Studies 61.3 (2004): 259-297.

Yamaguchi, Toshihiro. "On a web browsing support system with 3d visualization." International World Wide Web Conference New York, NY, USA: ACM Press, 2004. 316-317.

Zhai, Shumin. "The partial-occlusion effect: utilizing semitransparency in 3D human-computer interaction." ACM Transactions on Computer-Human Interaction (TOCHI) 3.3 (1996): 254-284.

12. Appendix

12.1 User Evaluation Survey

This survey is based on the *Questionnaire for User Interaction Satisfaction* developed by Ben Shneiderman and available for reference at http://www.lap.umd.edu/quis/. The basic form and some of the questions and categories suggested by Shneiderman are used, but the overall survey has been heavily modified to suit the specific needs of this project.

PART 1: Overall User Reactions

Please Circle the numbers which most appropriately reflect your impressions about using this computer system. Not Applicable = NA.

1.1 Overall reactions to the system:	terrible									wonderful	
	1	2	3	4	5	6	7	8	9		NA
1.2	frustrating									satisfying	
	1	2	3	4	5	6	7	8	9		NA
1.3	dull									stimulating	
	1	2	3	4	5	6	7	8	9		NA
1.4	difficult									easy	
	1	2	3	4	5	6	7	8	9		NA
1.5	inadequate power									adequate power	
	1	2	3	4	5	6	7	8	9		NA
1.6	rigid									flexible	
	1	2	3	4	5	6	7	8	9		NA
1.7	ugly									aesthetically pleasing	
	1	2	3	4	5	6	7	8	9		NA

Please write your comments on the overall system here:

PART 2: Learning

2.1 Learning to operate the system: difficult easy

1 2 3 4 5 6 7 8 9 NA

2.1.1 Getting started: difficult easy

1 2 3 4 5 6 7 8 9 NA

2.1.2 Learning advanced features: difficult easy

1 2 3 4 5 6 7 8 9 NA

2.1.3 Time to learn to use the system: long short

1 2 3 4 5 6 7 8 9 NA

2.2 Exploration of features by trial and error: discouraging encouraging

1 2 3 4 5 6 7 8 9 NA

2.2.1 Exploration of features: risky safe

1 2 3 4 5 6 7 8 9 NA

2.2.2 Discovering new features: difficult easy

1 2 3 4 5 6 7 8 9 NA

2.3 Tasks can be performed in a straightforward manner: never always

1 2 3 4 5 6 7 8 9 NA

2.3.1 Number of steps per task: too many just right

1 2 3 4 5 6 7 8 9 NA

2.3.2 Steps to complete a task follow a logical sequence: never always

1 2 3 4 5 6 7 8 9 NA

2.3.3 Feedback on the completion of sequence of steps: unclear clear

1 2 3 4 5 6 7 8 9 NA

Please write your comments on learning here:

PART 3: System Capabilities

3.1 System speed: too slow fast enough

1 2 3 4 5 6 7 8 9 NA

3.1.1 Response time for most operations: too slow fast enough

1 2 3 4 5 6 7 8 9 NA

3.1.2 Rate information is displayed: too slow fast enough

1 2 3 4 5 6 7 8 9 NA

3.2 Correcting your mistakes: easy difficult

1 2 3 4 5 6 7 8 9 NA

3.2.1 Correcting typos: difficult easy

1 2 3 4 5 6 7 8 9 NA

3.2.2 Correcting misplaced objects or navigation errors: difficult easy
1 2 3 4 5 6 7 8 9 NA

3.2.3 Ability to undo operations: inadequate adequate
1 2 3 4 5 6 7 8 9 NA

3.3 Ease of operation depends on your level of experience: never always
1 2 3 4 5 6 7 8 9 NA

3.3.1 You can accomplish tasks knowing only a few commands: with difficulty easily
1 2 3 4 5 6 7 8 9 NA

3.3.2 You can use features/shortcuts: with difficulty easily
1 2 3 4 5 6 7 8 9 NA

Please write your comments on system capabilities here:

PART 4: Usability

4.1 Navigation: difficult easy
1 2 3 4 5 6 7 8 9 NA

4.1.1 You were able to use the mouse to change facing and select objects on the screen: with difficulty easily
1 2 3 4 5 6 7 8 9 NA

4.1.2 You were able to use the keyboard to navigate: with difficulty easily
1 2 3 4 5 6 7 8 9 NA

4.2 Selection: difficult easy
1 2 3 4 5 6 7 8 9 NA

4.2.1 You were able to select desired files: with difficulty easily
1 2 3 4 5 6 7 8 9 NA

4.2.2 You were able to select menu options and controls: with difficulty easily
1 2 3 4 5 6 7 8 9 NA

4.3 Organization: difficult easy
1 2 3 4 5 6 7 8 9 NA

4.3.1 Ease of organizing data: difficult easy
1 2 3 4 5 6 7 8 9 NA

4.3.2 Time to find required data: slow fast
1 2 3 4 5 6 7 8 9 NA

Please write your comments on system capabilities here:

12.2 Problem Statement for Immersive First Person File Browser

Revision History

Date	Version	Description	Edited by
January 10, 2005	1.0	Initial draft	Geoffrey Ulman
March 21, 2005	2.0	Filled in holes in initial draft. Editing changes.	Geoffrey Ulman

High Level Problem Summary

Summary of Primary Success Criteria

- Program design grounded in research on current attempts to create 3-D user interfaces, principles which lead to good interface design (both 2-D and 3-D), and the advantages which 3-D interfaces can provide.
- Program demonstrates some of the advantages and principles enumerated in the research.
- Preliminary usability testing is conducted.
- Research, program design, and results are compiled into a finished thesis.

Scope

The Immersive First Person File Browser will provide a file browsing interface which will allow users to view, manipulate, and access the files on their personal computer's hard disk. The system will place users in a virtual environment which they will experience from the first person (the navigate the environment as if they themselves were walking around the room with the screen displaying what they would see). The files stored on their computer will be represented by papers, cassette tapes, virtual CD's, framed pictures, and other objects (which will provide a visual cue about the type of the file). These files will be stored in virtual imitations of real world storage medium such as boxes, piles, bookshelves, desks, and filing cabinets.

The user of the system will be able to customize their virtual space by moving these objects and placing their various files inside of them. Ideally, this system will provide all the features that a current commercial file browser such as Windows Explorer would provide, along with features specifically designed to take advantage of some of the unique advantages of 3-D and implement the interface principles discussed in the accompanying paper. However, these features are categorized by their importance and only the critical features necessary for a basic proof of concept will be concentrated on initially. The specification of the other features is provided as a guideline for future work.

Not included in the scope of this project are:

- Management of the underlying file system. Although this project attempts to do away with the concept of the hierarchical storage of data, it will (at least in this early implementation) rely on the underlying Windows file system to actually keep track of files on disk.
- This project will not act as a window manager for applications. Although the interface would feel much more natural if a 3-D window manager was provided which worked in concert with the 3-D file browser, this feature is not within the scope of this project.
- Any OS functions besides the graphical display of information on the hard disk are not within the project scope.

- Full integration with the OS. This application, for simplicity, will run as a separate application which will affect the file structure through calls to the OS. It will store and meta-data relating to files (the position of their representation in the 3-D environment, for example, in a separate data structure maintained by the program).
- Creation of a standard AIP allowing other programs to interact with this program (i.e. allow the "save" dialogue of a program to use this file browser).
- Thorough help documentation beyond what is provided in this project description and research paper will not be included.

Detailed Problem Statement

1.0 Function

Key Features

Feature No.	Feature Name
	File Modification Features
1.1	Import a file on the disk into the virtual environment of the file browser.
1.2	Auto import sets of files (for example the Windows "My Documents" folder).
1.3	Create new .txt files inside the virtual environment.
1.4	Delete files from the OS file system from within the virtual environment.
1.5	Allow files to be deleted using "proprioceptive" motions (for example by dragging the file off of the bottom of the screen).
1.6	Allow editing of text files from within the virtual environment.
1.7	Expand file types which can be created inside the virtual environment.
1.8	Display files of unrecognized file type with a unique virtual object. This virtual object will be a piece of paper which contains the icon for that file and the file's name.
1.9	Display .txt files with a unique virtual object. This virtual object will have multiple forms depending on where the user wishes it to be displayed: piece of paper, books, possibly others. These should display the title of the file as well as a portion of its text to allow quick scanning of the contents of multiple files.
1.10	Display .jpg and .gif images with a unique virtual object. This virtual object will have multiple forms depending on where the user wishes it to be displayed: a piece of paper (with the image displayed on it), a framed picture placed on the wall, or possibly wallpaper or texturing (upholstery, etc..) on various objects.
1.11	Display .avi (+ other java supported sound files) with a unique virtual object.
1.12	Allow extensions to add support for other types of virtual objects.
1.13	Allow users to change the name of files in the virtual environment.
	Navigation Features
2.1	Implement "mouse-look" and keyboard movement for basic navigation in the virtual environment.
2.2	Implement object based navigation (exact form to be decided later – see *Navigation—Steering and Target-based travel* section).
2.3	Implement collision detection (allow ability to turn on/off).
2.4	Provide non-functional decorative objects to the environment (make the virtual space more aesthetically pleasing as well as providing landmarks to aid in spatial memory acquisition).
	Virtual Object Manipulation Features
3.1	Allow users to *pick up* and *put down* virtual objects.
3.2	Implement an *inventory* for the *user*, indicating the *objects* the user has *picked up*.
3.3	Provide automatic arrangements of the virtual space based on certain characteristics of the file. Allow the user to easily return to their default/manual arrangement.
3.4	Support "imprecise" actions (sweeping a number of *piles* of *objects* onto the floor to make

	room on the desk, for example) by allowing objects to affect each other (i.e. allow pushing one *pile* to cause other adjacent piles to move). Another alternative would simply be to allow the group selection of multiple objects.
Virtual Object Organization Features	
4.1	Allow users to create *piles* of *objects*.
4.2	Allow the users to expand piles of objects and select objects from the expanded view.
4.3	Allow users to automatically create piles of objects based on attributes.
4.4	Allow users to create *storage units* which contain collections grouped objects which are functionally equivalent to *piles*. The form of storage units may vary (desks, filing cabinets, etc…), although they should share a common interface.
4.5	Allow users to place *piles* inside *storage units* and remove *piles* or individual *objects* from them.
4.6	Allow users to place objects onto the sides/top of *storage units* (i.e. allow users to place documents on the top of a desk as well as in its drawers.
4.7	Allow users to split piles into multiple stacks.
4.8	Allow users to take *files/objects* out of *storage units* as individual files or in a stack.
Virtual Object Display Features	
5.1	Allow users to create framed *paintings* from image file objects and hang them on the walls of the virtual space.
5.2	Allow users to specify sound files to play in the virtual space.
5.3	Allow users to expand the contents of a *storage unit*, allowing easier visualization/scanning of its content.
5.4	(Example of above feature: A bookshelf might allow users to quickly scan the titles of files stored on multiple shelves. Then, clicking on a shelf could cause the files on that shelf to float in the air in front of the bookshelf facing the user, thus allowing for easy selection of an individual file).
5.5	(Example of above feature: A box could expand its contents in a similar manner—a files arranged in the 2-D plane floating in front of the user—but with files near the bottom of the box displayed on the bottom of the screen and those near the top at the top of the screen).
Other Features	
6.1	Allow creation of objects which represent shortcuts to programs.

Key interfaces

Feature No	**Feature Name**
	The Windows XP file system. The file browser must interact with the operating system to get information about the files on the disk. This interface is handled through the Java System and I/O classes.
	The graphics card / OpenGL graphics libraries. Java3D is the AIP which will be used to interact with the systems 3-D graphics resources.

2.0 Form

Key attributes

Performance and Capacity (secondary concern)

- The system should be stable and scalable. It should be able to handle a typical number of personal documents (less than 10,000).
- However, only a small subset of these need to be readily available through the main organizational objects which a user has created (only the 100 most used files need to be easily organized into piles, easily accessible drawers in desks, etc—less used files can be stored in mass storage devices like boxes or a large "misc. items" filing cabinet or simply not be imported into the virtual environment and only brought in when they will be needed).
- The system must respond in real time without visible pauses to respond to user actions.

Reliability (secondary concern)

- The system should be relatively bug free and provide accurate error messages when unable to perform operations.

Usability (primary concern)

- See *Three Dimensional Interface Advantages* section for some of the usability benefits which this system may exhibit over traditional WIMP file browsers.
- The main purpose of this study is to test whether these benefits can be actualized, thus project success is not determined by whether or not these benefits are achieved, but rather by whether or not the question of if they can be achieved or not has been answered.
- In addition:
    - Users should report a more enjoyable experience (as measured by a numerical metric obtained by surveying the users after usability testing)
    - The system should provide faster or comparable times for a user to scan files for a particular item.
    - The system should provide a more natural file organization (numerical measurement obtained by survey).
    - The system should provide faster or comparable times for a user to organize files into an organization of their choice.

Security (secondary concern)

- The operating system will manage any security issues. This is outside the scope of this project.

Modifiability, maintainability, and customizability

- The system should be customizable and modifiable enough to support the iterative development plan set forth here. However, in the interest of saving time, and because of the fact that the developer will be learning the Java3D API concurrently with development, some degree of rigidity in the program structure will be allowed. Note: Customizability here refers to programmatic customizability—not modification of the user's virtual environment within the system.
- Maintainability is also a secondary concern. Code should be well documented, however beyond this no explicit maintainability aids will be included.

Testability

- Debugging and correctness testing will be done concurrently with development.
- Usability testing (covered above) will be conducted separately.

Hardware and Software Constraints

The system will be developed in Java with the Java3D API for implementing the 3-D graphics. Java should abstract most of the operating system dependence (such as file access calls) but in the interest of time the system will be implemented and tested solely using the Windows XP file system.

Required Standards
No official standards. However, the final system should demonstrate the principles enumerated and discussed in this thesis.

3.0 Economy

Being a student developed "proof of concept" application, economic considerations play a minimal role. However, there are a few considerations worth noting.

Business Context
The *Approaches to Three Dimensional User Interfaces* and *Approaches to Document Organization/Search* sections of this paper describe the current approaches which are being taken to develop 3-D file system interfaces. After conducting this thorough review of the current practical work in the field, I feel that my application fills a niche which has not been thoroughly explored. It takes a graphical approach every similar to *3DNA Desktop*, but provides much more extensive file organization and viewing options similar to applications like *Presto/Vista*. It aims not to entertain the user with the virtual environment, but provide them with a functional workspace.

Customer Organizational Constraints
Not applicable.

Development Organizational Constraints
Not applicable.

Key Risks and Uncertainty
- I do not have complete working knowledge of the Java3D API which will be used to do the graphics for the application. This will be researched and learned as development progresses, but still is a project risk.

4.0 Time

Historical Context and Current Context
See the *Approaches to Three Dimensional User Interfaces* and *Approaches to Document Organization/Search* sections for current and previous approaches to 3-D user interfaces.

Future Context
This application could possibly be extended to include interfaces to the financial aid applications and for department heads and instructors to view class interest in planning for the number of sections of courses.

Development Time
The program development will begin the week of January 10th and continue until approximately April 11th. This deadline will allow enough time for finalization of the thesis paper as well usability testing of the program.