George G. Robertson, Stuart K. Card, and Jock D. Mackinlay

INFORMATION VISUALIZATION USING 3D INTERACTIVE ANIMATION

UI innovations are often driven by a combination of technology advances and application demands. On the technology side, advances in interactive computer graphics hardware, coupled with low-cost mass storage, have created new possibilities for information retrieval systems in which UIs could play a more central role. On the application side, increasing masses of information confronting a business or an individual have created a demand for information management applications. In the 1980s, text-editing forced the shaping of the desktop metaphor and the now standard GUI paradigm. In the 1990s, it is likely that information access will be a primary force in shaping the successor to the desktop metaphor.

This article presents an experimental system, the *Information Visualizer* (see Figure 1), which explores a UI paradigm that goes beyond the desktop metaphor to exploit the emerging generation of graphical personal computers and to support the emerging application demand to retrieve, store, manipulate, and understand large amounts of information. The basic problem is how to utilize advancing graphics technology to lower the cost of finding information and accessing it once found (the information's "cost structure").

ILLUSTRATION: RICO LINS STUDIO

APPLE 1012

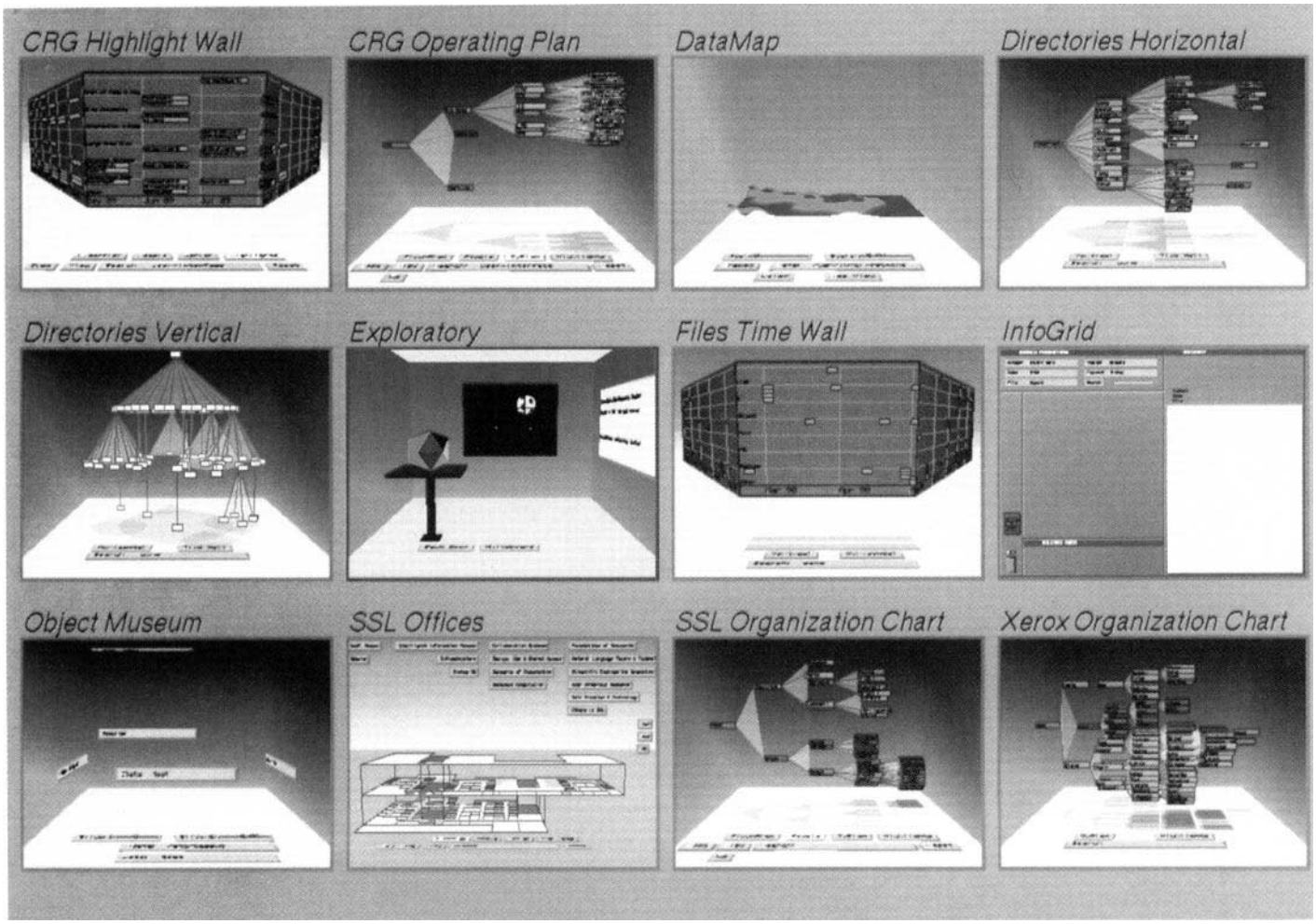


Figure 1. Information Visualizer Overview

We take four broad strategies: making the user's immediate workspace larger, enabling user interaction with multiple agents, increasing the real-time interaction rate between user and system, and using visual abstraction to shift information to the perceptual system to speed information assimilation and retrieval.

Technology Advances

Since the early development of the standard GUI, hardware technology has continued to advance rapidly. Processor and memory technology have far greater performance at far lower cost. Specialized 3D graphics hardware has made it progressively faster and cheaper to do 3D transformations, hidden-surface removal, double-buffered animation, antialiasing, and lighting and surface models. At the same time, software support for real-time operating systems and emerging industry standard open graphics libraries (e.g., OpenGL and PEX) are simplifying the 3D programming task. The trend will bring these technologies to the mass market in the near future.

These technology advances have created many possibilities for user interface innovation. Yet the basic Windows-Icons-Menus-Pointing (WIMP) desktop metaphor has not changed much since its emergence in the Alto/Smalltalk work. Nonetheless, there is a great desire to explore new UI paradigms. Experiments with pen-based notebook metaphors, virtual reality, and ubiquitous computing are proceeding and may eventually influence the mass market. Brown University's Andy van Dam, in several recent conferences has exhorted us to break out of the desktop metaphor and escape flatland and a recent workshop focused on Software Architectures for Non-WIMP User Interfaces [9]. It is this kind of technology change that is driving our research in the Information Visualizer.

Information Access vs. Document Retrieval

Computer-aided access to information is often thought of in the context of methods for library automation. In particular, document retrieval [19] is usually defined more or less as follows: There exists a set of documents and a person who has an interest in the information in some of them. Those documents that contain information of interest are *relevant*, others not. The problem is to find all and only the relevant documents. There are two standard figures of merit for comparing and evaluating retrieval systems: *Recall* is the percentage of all the relevant documents found; and *precision* is the percentage of the documents found that are rel-

evant. While this formulation has been useful for comparing different approaches, we propose extending the document retrieval formulation to take the larger context into account. From a user's point of view, document retrieval and other forms of information retrieval are almost always part of some larger process of information use [2]. Examples are *sensemaking* (building an interpretation of understanding of information), *design* (building an artifact), *decision making* (building a decision and its rationale), and *response tasks* (finding information to respond to a query).

In each of these cases:

1. Information is used to produce more information, or to act directly
2. The new information is usually at a higher level of organization relative to some purpose

If we represent the usual view of information retrieval as Figure 2(a), we can represent this extended view as Figure 2(b). Framing the problem in this way is suggestive: what the user needs is not so much information retrieval itself, but rather the amplification of information-based work processes. That is, in addition to concern with recall and precision, we also need to be concerned with reducing the time cost of information access and increasing the scale of information that a user can handle at one time.

Information Workspaces

From our observations about the problem of information access [2], we were led to develop UI paradigms oriented toward managing the cost structure of information-based work. This, in turn, led us to be concerned not just with the retrieval of information from a distant source, but also with the accessing of that information once it is retrieved and in use. The need for a low-cost, immediate storage for accessing objects in use is common to most kinds of work. The common solution is a *workspace,* whether it be a woodworking shop, a laboratory, or an office. A workspace is a special environment in which the cost structure of the needed materials is tuned to the requirements of the work process using them.

Computer screens provide a workspace for tasks done with the computer. However, typical computer displays provide limited working space. For real work, one often wants to use a much larger space, such as a dining room table. The Rooms system [10] was developed to extend the WIMP desktop to multiple workspaces that users could switch among, allowing more information to reside in the immediate work area. The added cost of switching and finding the right workspace was reduced by adding the ability to share the same information objects in different workspaces. Rooms also had an overview and other navigational aids as well as the ability to store and retrieve workspaces, all to remove the major disadvantages of multiple desktops.

The essence of our proposal is to evolve the Rooms multiple desktop metaphor into a workspace for information access—an *Information Workspace* [2]. Unlike the conventional information retrieval notion of simple access of information from some distal storage, an information workspace (1) treats the complete cost structure of information, integrating information access from distant, secondary or tertiary storage with information access from Immediate Storage for information in use, and (2) considers information access part of a larger work process. That is, instead of concentrating narrowly on the control of a search engine, the goal is to improve the cost structure of information access for user work.

With this system, we use four methods for improving the cost structure of information access:

1. *Large Workspace.* Make the Immediate Workspace virtually larger, so that the information can be held in low-cost storage
2. *Agents.* Delegate part of the workload to semiautonomous agents
3. *Real-Time Interaction.* Maximize interaction rates with the human user by tuning the displays and responses to real-time human action constants
4. *Visual Abstractions.* Use visual abstractions of the information to speed assimilation and pattern detection

Figure 2. (a) Traditional information retrieval formulation and (b) reformulation with context of use

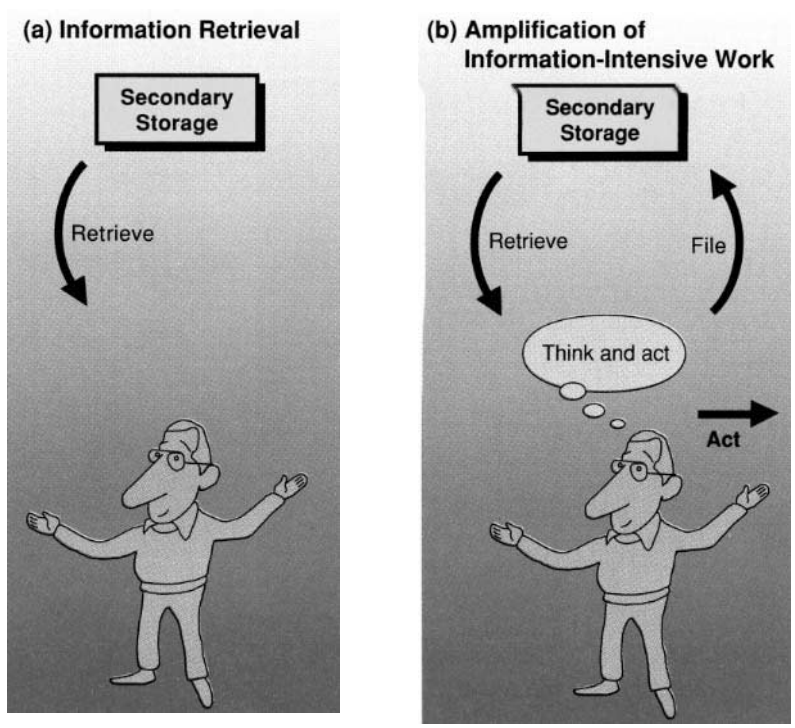
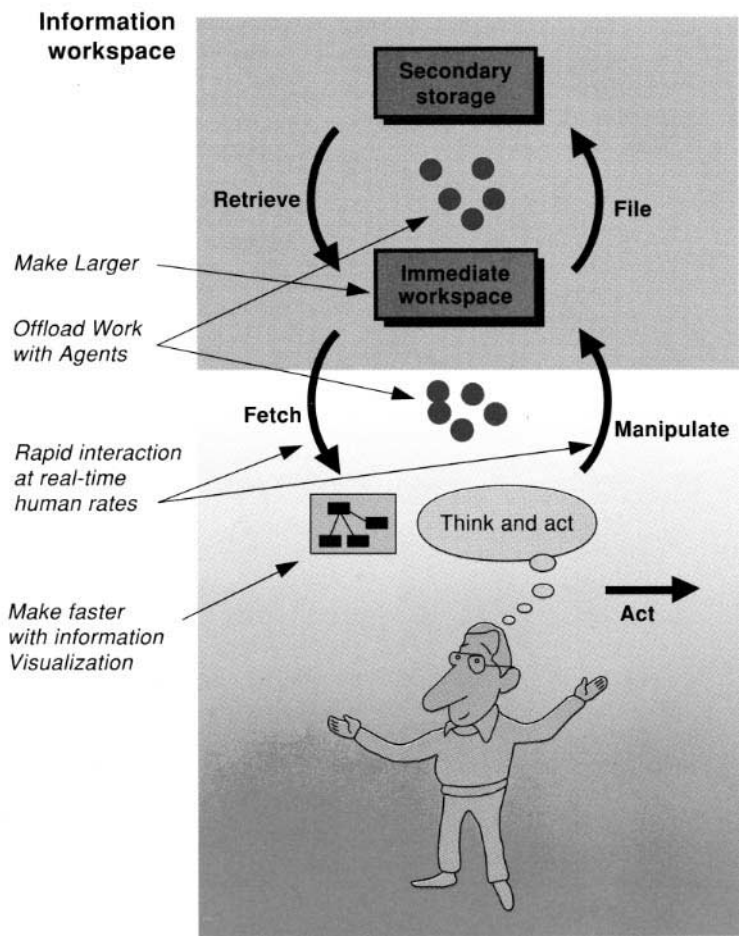


Table 1. Techniques used in the Information Visualizer to increase information access per unit cost

System Goal	Techniques
1. Large workspace to reduce access cost	More screen space → Rooms Denser screen space → Animation, 3D
2. Offload work to agents	search → search agents Organizing → clustering agents Interacting → Interactive Objects
3. Maximize real-time interaction rates	Rapid interaction, Tune to human constants → Cognitive Coprocessor Scheduler and Governor
4. Visual abstraction to speed pattern detection	Information Visualizations: Hierarchical structure → Cone Tree Linear structure → Perspective Wall Continuous data → Data Sculpture Spatial data → Office Floor Plan

Table 2. Information Visualizer solutions to basic UI problems.

Problem	Solution
1. Multiple Agent problem	Cognitive Coprocessor scheduler
2. Animation problem	Cognitive Coprocessor scheduler and governor
3. Interaction problem	Interactive Objects
4. Viewpoint Movement problem	Point of Interest Logarithmic Flier
5. Object Movement problem	Object of Interest Logarithmic Manipulator
6. Small Screen problem	3D/Rooms and 3D visualizations



These define the goals for our UI paradigm. Each of these is intended to decrease the costs for performing information-intensive tasks, or, alternatively, to increase the scope of information that can be utilized for the same cost. Figure 3 shows how these goals are applied to the reformulated information access problem shown in Figure 2(b).

The Information Visualizer system is our experimental embodiment of the Information Workspace concept with mechanisms for addressing each of these system goals (see Table 1): 1) [Large Workspace]. We use two methods to make the workspace larger: We add *more (virtual) screen space* to the Immediate Workspace by using a version of the Rooms system. We *increase the density of information* that can be held in the same screen space by using animation and 3D perspective. 2) [Agents]. To delegate part of the workload, we use agents to conduct searches, to organize information into clusters, or design presentations of information. We manage this by means of a scheduling architecture, called the Cognitive Coprocessor [17], that allows multiple display and application processes to run together. A kind of user interface agent, called Interactive Objects, is used to control and communicate with the system. 3) [Real-time Interaction]. To maximize human interaction rates, we use the properties of the scheduler to provide highly interactive animation and communication with the Interactive Objects. To tune the system to human action times, we require certain classes of actions to occur at set rates. To enforce these rates under varying computational load, we use a Governor mechanism in our scheduler loop. 4) [Visual Abstractions]. To speed the user's ability to assimilate information and find patterns in it, we use visualization of different abstract information structures, including linear structures, hierarchical struc-

Figure 3. Improving the Information cost structure in the Information access model

tures, continuous data, and spatial data.

There have been many systems that have supported interactive animation-oriented UIs, starting with Ivan Sutherland's thesis [23] at the dawn of computer graphics. As with Sutherland's thesis, early examples required specialized and/or expensive computing machinery and were oriented toward specialized tasks. Cockpit simulation systems are a good example. The architectures for such systems share the animation-loop core with our system. The drop in cost for 3D animated systems and the increase in capability has accelerated experiments in using this technology as the basis of a new mass market user interface paradigm. One strategy has been to work up from building blocks. A. Van Dam's group at Brown has been working on an object-oriented framework for interactive animation, 3D widgets [3], and modeling time in 3D interactive animation systems. Silicon Graphics has recently introduced a high-level 3D toolkit, called Inventor. Another tack has been to drive the development by focusing on applications, for example, continuously running physical simulations. M. Green's group at the University of Alberta has developed a Decoupled Simulation Model for virtual reality systems [20]. Their architectural approach is similar to ours, but focuses more on continuously running simulations. D. Zeltzer and colleagues at MIT [25] have built a constraint-based system for interactive physical simulation. Our system, by contrast, is oriented toward the access and visualization of abstract nonphysical information of the form that knowledge workers would encounter.

UI Architecture

In order to achieve the goals set forth in Table 1 we have been led to a UI paradigm involving highly interactive animation, 3D, agents, and visualizations. This is one of the UI regimes now being made practical by current and predicted advances in hardware and software technology. There are several problems, however, which need to be addressed in order to realize such a UI paradigm:

1. *The Multiple Agent Problem.* How can the architecture provide a systematic way to manage the interactions of multiple asynchronous agents?
2. *The Animation Problem.* How can the architecture provide smooth interactive animation *and* solve the Multiple Agent problem?
3. *The Interaction Problem.* How can 3D widgets be designed and coupled to appropriate application behavior?
4. *The Viewpoint Movement Problem.* How can the user rapidly and simply move the point of view around in a 3D space?
5. *The Object Movement Problem.* How can objects be easily moved about in a 3D space?
6. *The Small Screen Space Problem.* How can the dynamic properties of the system be utilized to provide the user with an adequately large workspace?

Many of these problems are well known. The Multiple Agent and Animation problems are less obvious, and since they define the basic organization of the Information Visualizer, we describe them in more detail.

The Multiple Agent Problem. We want our architecture to support multiple agents to which the user can delegate tasks. In fact, we have previously argued [17] that T.B. Sheridan's analysis of the supervisory control of semiautonomous embedded systems [21] can be adapted to describe the behavior of an interactive system as the product of the interactions of (at least) three agents: a *user,* a *user discourse machine* (the UI), and a task machine or *application.* These agents operate with very different time constants. For example, a search process in an application and the graphical display of its results may be slow, while the user's perception of displayed results may be quite fast. The UI must provide a form of "impedance matching" (dealing with different time constants) between the various agents as well as translate between different languages of interaction. The application itself may be broken into various agents that supply services, some of which may run on distributed machines (e.g., an agent to filter and sort your mail). Even the UI may itself contain agents (e.g., presentation agents). These additional agents have their own time constants and languages of interaction that must be accommodated by the UI.

Impedance matching can be difficult to accomplish architecturally because all agents want rapid interaction with no forced waiting on other agents, and the user wants to be able to change his or her focus of attention rapidly as new information becomes available. For example, if a user initiates a long search that provides intermediate results as they become available, the user should be able to abort or redirect the search at any point (e.g., based on perception of the intermediate results), without waiting for a display or search process to complete. The UI architecture must provide a systematic way to manage the interactions of multiple asynchronous agents that can interrupt and redirect one another's work.

The Animation Problem. Over the last 65 years, animation has grown from a primitive art form to a very complex and effective discipline for communication. Interactive animation is particularly demanding architecturally, because of its extreme computational requirements.

Smooth interactive animation is particularly important because it can shift a user's task from cognitive to perceptual activity, freeing cognitive processing capacity for application tasks. For example, interactive animation supports object constancy. Consider an animation of a complex object that represents some complex relationships. When the user rotates this object, or moves around the object, animation of that motion makes it possible (even easy, since it is at the level of perception) for the user to retain the relationships of what is displayed. Without animation, the display would jump from one configuration to another, and the user would have to spend time (and cognitive effort) reassimilating the new display. By providing object constancy, animation significantly reduces the cognitive load on the user.

The Animation Problem arises when building a system that attempts

to provide smooth interactive animation *and* solve the Multiple Agent problem. The difficulty is that smooth animation requires a fixed rate of guaranteed computational resource, while the highly interactive and redirectable support of multiple asynchronous agents with different time constants has widely varying computational requirements. The UI architecture must balance and protect these very different computational requirements.

In fact, the animation problem is one aspect of a broader Real-Time Interaction problem. Services need to be delivered under real-time deadline, under varying load, while simultaneously handling the Multiple Agent problem.

The Cognitive Coprocessor

Table 2 summarizes the Information Visualizer's solutions to each of the problems described earlier. The next few sections describe these solutions.

The heart of the Information Visualizer architecture is a controlled-resource scheduler, the Cognitive Coprocessor architecture, which serves as an animation loop and a scheduler for Sheridan's three agents and additional application and interface agents. It manages multiple asynchronous agents that operate with different time constants and need to interrupt and redirect one another's work. These agents range from trivial agents that update display state to continuous-running simulations and search agents. This architecture provides the basic solution to the Multiple Agent and Animation problems.

The Cognitive Coprocessor is an impedance matcher between the cognitive and perceptual information processing requirements of the user and the properties of these agents. In general, these agents operate on time constants different from those of the user. There are three sorts of time constants for the human that we want to tune the system to meet: perceptual processing (0.1 second) [1], immediate response (1 second) [15], and unit task (10 seconds) [15].

The perceptual processing time constant. The Cognitive Coprocessor is based on a continuously running scheduler loop and double-buffered graphics. In order to maintain the illusion of animation in the world, the screen must be repainted at least every 0.1 second [1]. The Cognitive Coprocessor therefore has a *Governor* mechanism that monitors the basic cycle time. When the cycle time becomes too high, cooperating rendering processes reduce the quality of rendering (e.g., leaving off most of the text during motion) so that the cycle speed is increased.

The immediate response time constant. A person can make an unprepared response to some stimulus within about a second [15]. If there is more than a second, then either the listening party makes a back-channel response to indicate that he is listening (e.g., "uh-huh") or the speaking party makes a response (e.g., "uh...") to indicate he is still thinking of the next speech. These serve to keep the parties of the interaction informed that they are still engaged in an interaction. In the Cognitive Coprocessor, we attempt to have agents provide status feedback at intervals no longer than this constant. Immediate response animations (e.g., swinging the branches of a 3D tree into view) are designed to take about a second. If the time were much shorter, then the user would lose object constancy and would have to reorient himself. If they were much longer, then the user would get bored waiting for the response.

The unit task time constant. Finally, a user should be able to complete some elementary task act within about 10 seconds (say, 5 to 30 seconds) [1, 15]. This is about the pacing of a point and click editor. Information agents may require considerable time to complete some complicated request, but the user, in this paradigm, always stays active. A user can begin the next request as soon as sufficient information has developed from the last request or even in parallel with it.

The basic control mechanism (inner loop) of the Cognitive Coprocessor is called the *Animation Loop* (see Figure 4). It maintains a *Task Queue*, a *Display Queue*, and a *Governor*. Built on top of the Animation Loop is an information workspace manager (and support for 3D simulated environments), called *3D/Rooms;* supports for navigating around 3D environments; and support for *Interactive Objects*, which provide basic input/output mechanisms for the UI. The task machine (which, for the Information Visualization application, is a collection of visualizers) couples with the Cognitive Copocessor in various ways. More details of this architecture can be found in [17].

Interactive Objects

The basic building block in the Information Visualizer, called *Interactive Objects,* forms the basis for coupling user interaction with application behavior and offloading work to an agent to handle user interaction. Interactive Objects are a generalization of Rooms Buttons [10]. They are used to build complex 3D widgets that represent information or information structure.

Rooms Buttons are used for a variety of purposes, such as movement, new interface building blocks, and task assistance. A Button has an appearance (typically, a bitmap) and a selection action (a procedure to execute when the Button is 'pressed'). The most typical Button in Rooms is a door—when selected, the user passes from one Room to another. Buttons are abstractions that can be passed from one Room to another, and from one user to another via email. Interactive Objects are similar to Buttons, but are extended to deal with gestures, animation, 2D or 3D appearance, manipulation, object-relative navigation, and an extensible set of types.

An Interactive Object can have any 2D or 3D appearance defined by a draw method. The notion of selection is generalized to allow mouse-based gestural input in addition to simple 'pressing'. Whenever a user gestures at an Interactive Object, a gesture parser is involved that interprets mouse movement and classifies it as one of a small set of easily differentiated gestures (e.g., press, rubout, check, and flick). Once a gesture has been identified, a gesture-specific method is called. These gesture methods are specified when the Interactive Object is created. The ges-

ture parser can be easily extended to allow additional gestures and gesture methods, as long as the new gestures are easily differentiated from other gestures.

There are a number of types of Interactive Objects. In the current implementation, these include static text, editable text, date entry, number entry, set selection, checkmark, simple button, doors, sliders, and thermometers (for feedback and progress indicators). The basic set of 3D widgets supported for Interactive Objects can be easily extended.

Interactive Objects are generalized to the point that every visible entity in the simulated scene can be an Interactive Object (and should be, so that object-relative navigation is consistent across the scene). Thus, the surfaces of the 3D Room (the walls, floor, and ceiling) are Interactive Objects. All the controls (e.g., buttons, sliders, thermometers, text, and editable text) are Interactive Objects. And finally, the application-specific artifacts placed in the room are Interactive Objects.

Search Agents

Search agents are also used to offload user work. The Information Visualizer uses an indexing and search subsystem [5], which allows search for documents by keyword or by iterative "relevance feedback" (e.g., find the documents most like this document). Associative retrieval based on such linguistic searches can be used to highlight portions of an information visualization. Thus we can combine traditional associative searches with structural browsing.

In addition, clustering agents are used to organize information. Using a near-linear clustering algorithm [4], which allows interactive use of clustering, a structure can be induced on an unstructured (or partially structured) body of information. There are several ways this can be of use. For unstructured information, a user can induce a subject hierarchy, which can then be browsed with our hierarchy visualization tools. For information that already has a structure, the clustering results sometimes reveal problems with the existing structure. In general, if a

The heart of the Information Visualizer architecture is a controlled resource scheduler, the Cognitive Coprocessor architecture, *which serves as an animation loop and a scheduler for Sheridan's three agents and additional application and interface agents.*

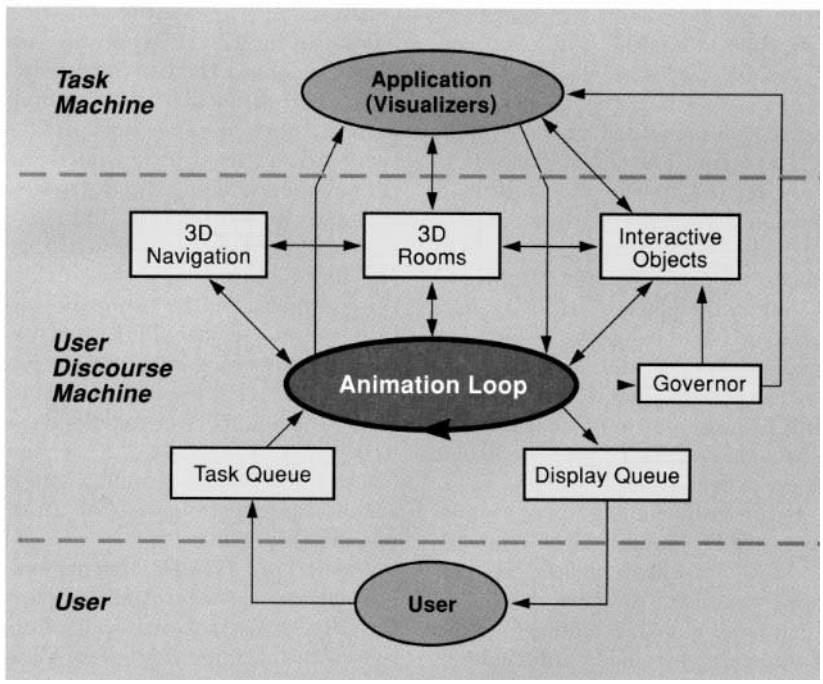


Figure 4. Cognitive Coprocessor Interaction architecture

user is unsure about the content of a corpus, and therefore unsure of what kinds of queries to make, clustering can provide an overview of the content of that corpus.

3D Navigation and Manipulation

In virtual 3D workspaces, techniques are required for moving the user (the viewpoint) and objects around the space. The Information Visualizer currently has five of these as building blocks, with others under development:

1. The Walking Metaphor
2. Point of Interest Logarithmic Flight
3. Object of Interest Logarithmic Manipulation
4. Doors
5. Overview

Walking Metaphor. The 'Walking Metaphor' [13] has virtual joystick controls superimposed as heads-up displays on the screen and controlled by the mouse. The controls are operators related to the way a human body might be moved (one control for body motion forward, backward, turn-left, or turn-right; a second for motion in the plane of the body: left, right, up, or down; and a third for rotating the head left, right, up, or down). This scheme is fairly general and works well for exploratory movement, which has no particular object as its target.

Large information spaces, however, involve numerous objects and/or highly detailed objects that require the user to move back and forth from global, orienting views to manipulate detailed information. Therefore, an important requirement for such systems is a movement technique that allows the user to move the viewpoint (1) rapidly through large distances, (2) with such control that the viewpoint can approach very close to a target without collision. We call this the problem of *rapid and controlled, targeted 3D movement* [12].

Point of Interest (POI) Logarithmic Flight. Our second navigation technique uses a point of interest logarithmic movement algorithm for very rapid, but precise movement relative to objects of interest [12]. Current techniques for moving the viewpoint [13] are not very satisfactory for targeted movement. They typically exhibit one or more of the following three problems: (1) inefficient interactions and movement trajectories, typically caused by 2D input devices; (2) difficulties controlling high velocities when the technique is based on flying or steering the viewpoint through the workspace; and (3) limits on human reach and precision when the technique is based on directly positioning the viewpoint.

Most viewpoint movement techniques focus on schemes for directly controlling the six degrees of freedom of viewpoint movement (3 position and 3 orientation) or their rate derivatives—a complex control task. Our solution is to have the user select a point of interest (POI) on the surface of an object and use the distance to this POI to calculate a logarithmic motion function. Two keys on the keyboard are used to indicate logarithmic motion along the ray toward and away from the POI. The viewpoint is automatically oriented during the flight to face the surface being approached by using the surface normal at the POI. Another control allows movement perpendicular to the surface normal. This allows for scrolling over extended objects (for example, a virtual blackboard) or circumnavigation around spherical objects (for example, a virtual globe.)

Object of Interest Logarithmic Manipulation. Logarithmic motion can also be used to manipulate objects with the same UI as POI viewpoint movement. The mouse cursor is used to control a ray that determines the lateral position of the object of interest (given the viewpoint coordinates) and the same keyboard keys are used to control the position of the object on the ray. However, the user must be able to control object position at a distance, where logarithmic motion is not effective. The solution is to use an acceleration motion clipped by a logarithmic motion. The object moves slowly at first (allowing control at a distance), then accelerates toward the viewpoint, and finally moves logarithmically slower for control near the viewpoint.

POI logarithmic flight and object of interest logarithmic manipulation both allow simple, rapid movement of the viewpoint and of objects in a 3D space over multiple degrees of freedom and scales of magnitude with only a mouse and two keyboard keys. We believe these techniques provide a mouse-based solution for the viewpoint movement and object movement problems that are as good or even better than those requiring special 3D devices. The chief advantage of a mouse-based solution is that mice are ubiquitous. Also, many users of information visualization (office workers, for example) are not likely to be willing to wear special equipment (such as gloves and helmets). Even so, the techniques could be adjusted to work with 3D devices such as the glove.

Doors. The 3D/Rooms system supports Doors that allow a user to move from one room (or workspace) through to a home position in another room. The Door is an Interactive Object that supports either manual control or scripted animation of opening and walking through to the other room.

Overview. As with Rooms, 3D/Rooms contains an *Overview* (see Figure 1) allowing the user to view all the 3D workspaces simultaneously. This is a navigation technique that lets the user view all the rooms and go to any room directly. In 3D/Rooms the user can also reach into the Rooms from the Overview, move about in them, and manipulate their objects.

3D/Rooms

3D/Rooms extends the logic of our Rooms system to three dimensions. In the classical desktop metaphor and the original Rooms system, the view of a Room is fixed. In 3D/Rooms, the user is given a position and orientation in the Room, and can move about the Room, zoom in to examine objects closely, look around, or even walk through doors into other Rooms. Thus 3D/Rooms is the same as Rooms, except that visualization artifacts (implemented as Interactive Objects) replace a collection of

windows, and users can have arbitrary positions and orientations in the Rooms.

The effect of 3D/Rooms is to make the screen space for immediate storage of information effectively *larger* (in the sense that the user can get to a larger amount of ready-to-use information in a short time). The effect of rapid zooming, animation, and 3D is to make the screen space effectively *denser* (in the sense that the same amount of screen can hold more objects, which the user can zoom into or animate into view in a short time). By manipulating objects or moving in space, the user can disambiguate images, reveal hidden information, or zoom in for detail—rapidly accessing more information. Both the techniques for making the Immediate Storage space virtually larger and the techniques for making the space virtually denser should make its capacity larger, hence the average cost of accessing information lower, hence the cost of working on large information-intensive tasks lower.

Information Visualization

Recent work in scientific visualization shows how the computer can serve as an intermediary in the process of rapid assimilation of information. Large sets of data are reduced to graphic form in such a way that human perception can detect patterns revealing underlying structure in the data more readily than by a direct analysis of the numbers. Information in the form of documents also has structure. *Information visualization* attempts to display structural relationships and context that would be more difficult to detect by individual retrieval requests. Although much work has been done using 3D graphics to visualize physical objects or phenomena, only a few systems have exploited 3D visualization for visualizing more abstract data or information structure.

The SemNet [6] system is an early example of the exploitation of 3D visualization of information structures. The structures visualized in SemNet were mostly large knowledge bases, and were often arbitrary graphs. The results tended to be cluttered, and the cognitive task of understanding the structure was still quite difficult.

The *n*-Vision system [7] exploits 3D to visualize *n*-dimensional business data. Multivariate functions are displayed in nested coordinates systems, using a metaphor called worlds-within-worlds. Although *n*-Vision focuses on continuous multivariate functions, it does exploit the human 3D perceptual apparatus to visualize abstract data.

Silicon Graphics has recently released an unsupported system, called File System Navigator (FSN) [24], which explores what they call Information Landscapes. In this system the file system hierarchy is laid out on a landscape, with each directory represented by a pedestal which has boxes representing individual files on top of it. They effectively use the 3D space to present structure, while using box size to represent file size and color to represent age. FSN uses a technique called 'artificial perspective', a form of fisheye effect [8], to make more effective use of screen space.

In the Information Visualizer, we have explored 3D visualizations for some of the classical data organizations:

1. Hierarchical: The *Cone Tree* visualization [18] (see following description).
2. Linear: The *Perspective Wall* visualization [14] (see following description).
3. Spatial: The spatial structure of a building (see Figure 5) can be used as a structural browser for people. Selecting an organization will produce the names and pictures of its members and select their offices. Clicking on offices retrieves their inhabitants.
4. Continuous Data: In the Data Sculpture (see Figure 6), the user can walk around or zoom into this visualization containing over 65,000 sampling points as if it were a sculpture in a museum.
5. Unstructured: The Information Grid [16] is a 2D visualization for unstructured information.

These visualizations use interactive animation to explore dynamically changing views of the information structures. More visualizations are visible in the 3D/Rooms Overview of Figure 1. The visualizers attempt to present abstractions of large amounts of data tuned to the pattern detection properties of the human perceptual system. For example, they use color, lighting, shadow, transparency, hidden surface occlusion, continuous transformation, and motion cues to induce object constancy and 3D perspective.

Visualizing Hierarchical Structure: Cone Trees

Hierarchies are almost ubiquitous, appearing in many different applications, hence are good information structures to exploit. In some cases, arbitrary graphs can be transformed into hierarchies (with auxiliary links), so the utility of hierarchy visualization is further enhanced.

Cone Trees are hierarchies laid out uniformly in three dimensions. Figure 7 is a snapshot of a simple Cone Tree. Nodes are drawn as 3- × 5-inch index cards. The top of the hierarchy is placed near the ceiling of the room, and is the apex of a cone with its children placed evenly spaced along its base. The next layer of nodes is drawn below the first, with their children in cones. The aspect ratio of the tree is fixed to fit the room. Each layer has cones of the same height (the room height divided by the tree depth). Cone base diameters for each level are reduced in a progression so that the bottom layer fits in the width of the room. The body of each cone is shaded transparently, so that the cone is easily perceived yet does not block the view of cones behind it. The display of node text does not fit the aspect ratio of the cards very well, hence text is shown only for the selected path. Figure 8 shows an alternative layout, which is horizontally oriented and has text displayed for each node.

When a node is selected with the mouse, the Cone Tree rotates so that the selected node and each node in the path from the selected node up to the top are brought to the front and highlighted. The rotations of each substructure are done in parallel, following the shortest rotational path, and are animated so the user sees the transformation at a rate the

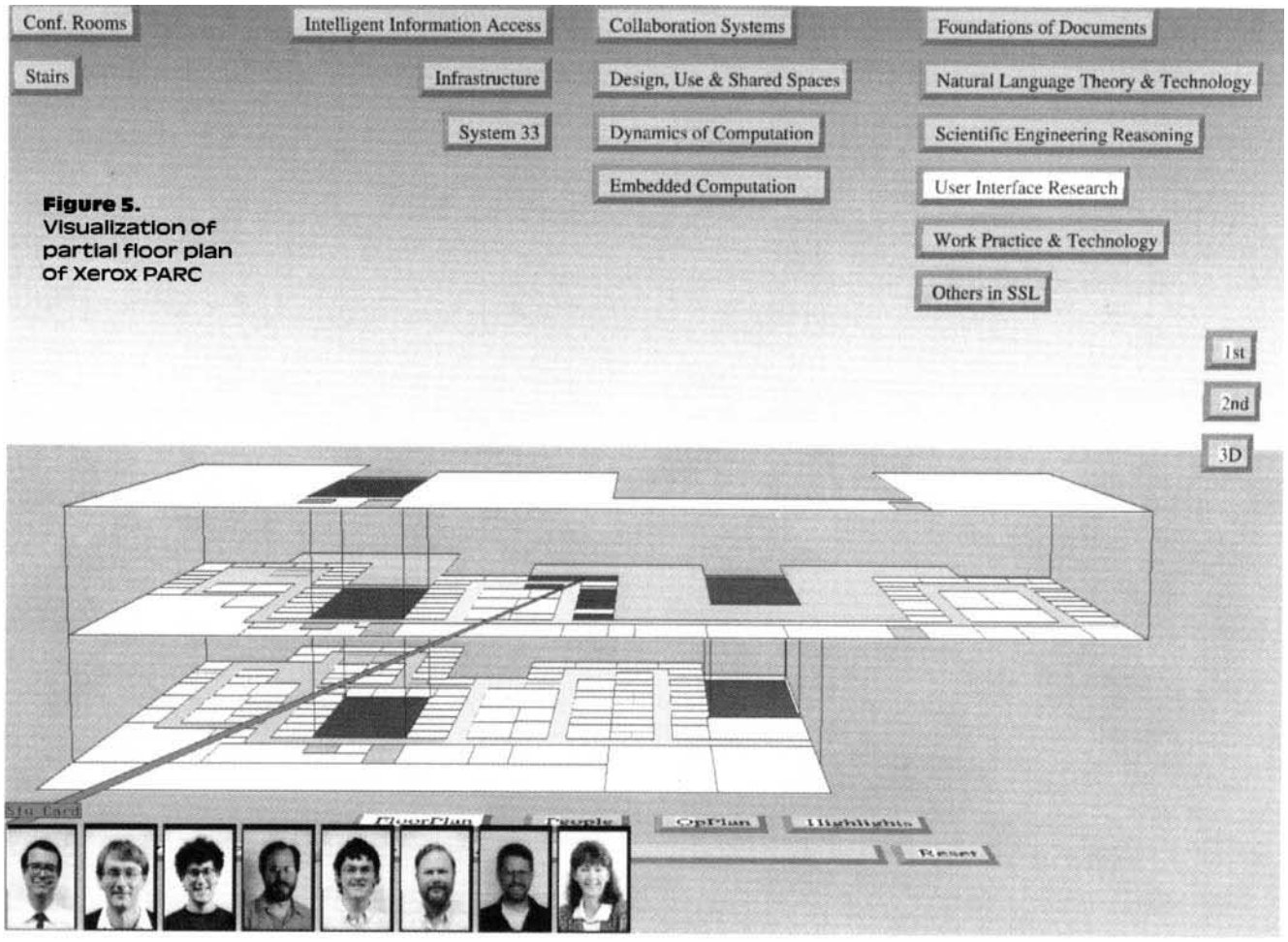


Figure 5. Visualization of partial floor plan of Xerox PARC

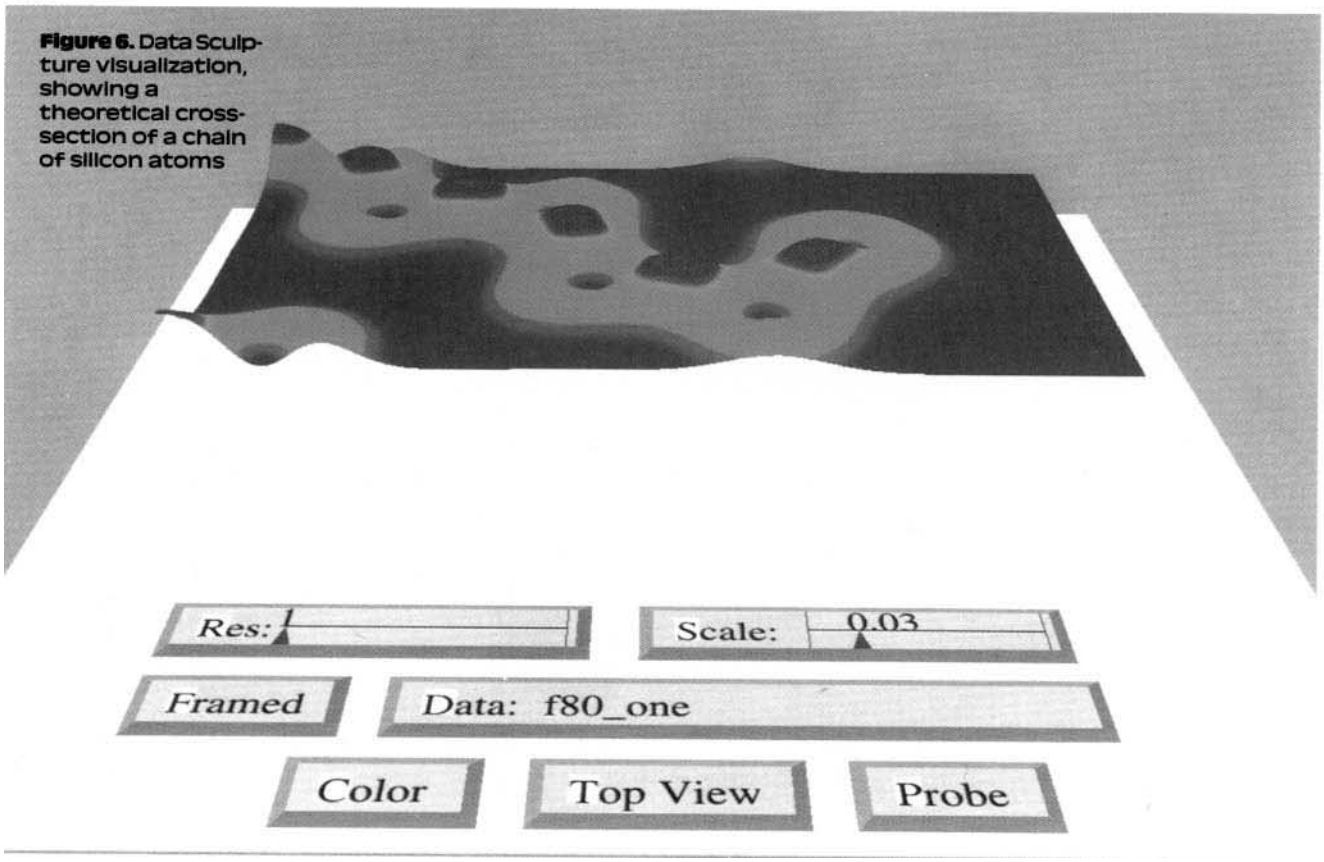


Figure 6. Data Sculpture visualization, showing a theoretical cross-section of a chain of silicon atoms

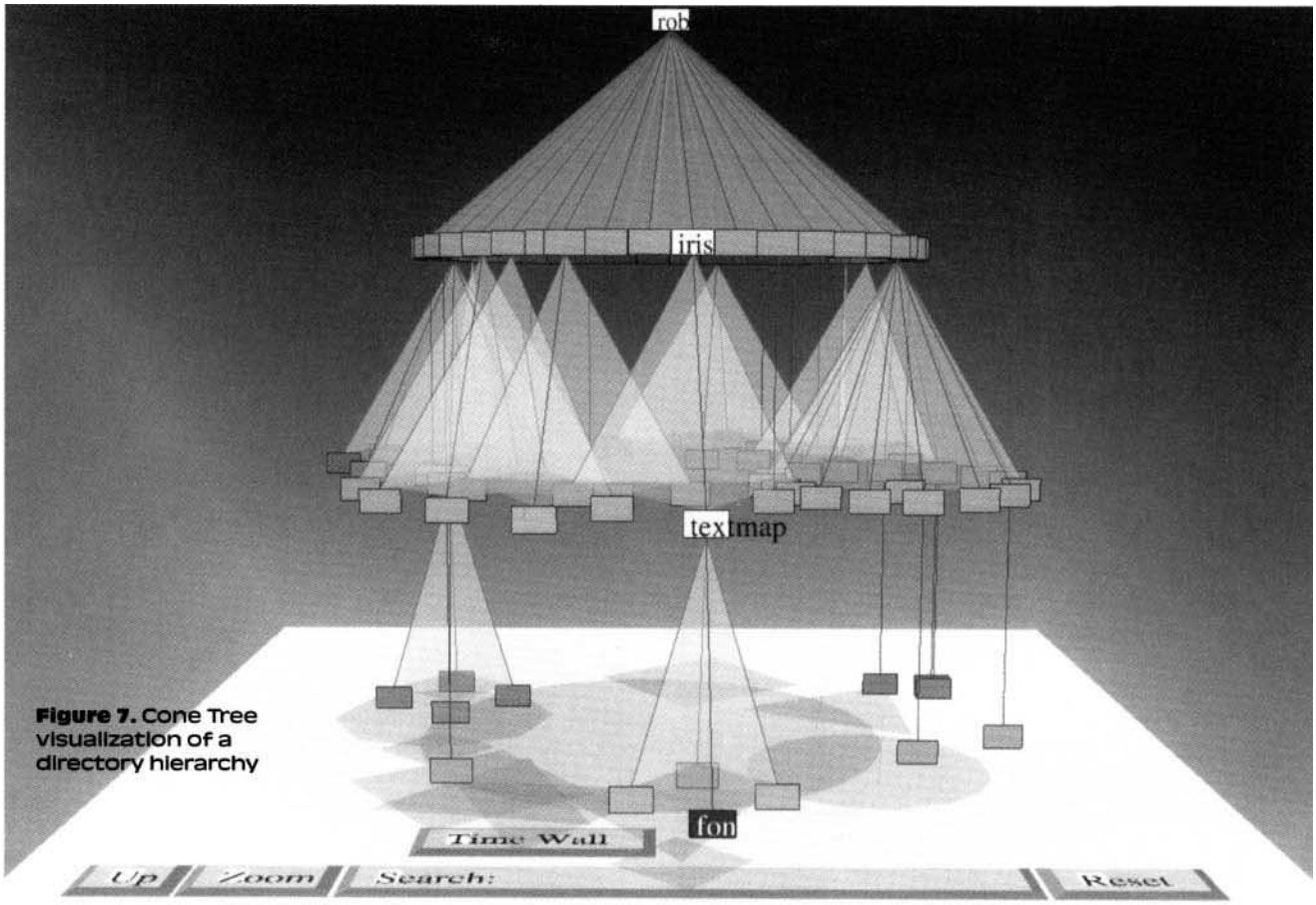


Figure 7. Cone Tree visualization of a directory hierarchy

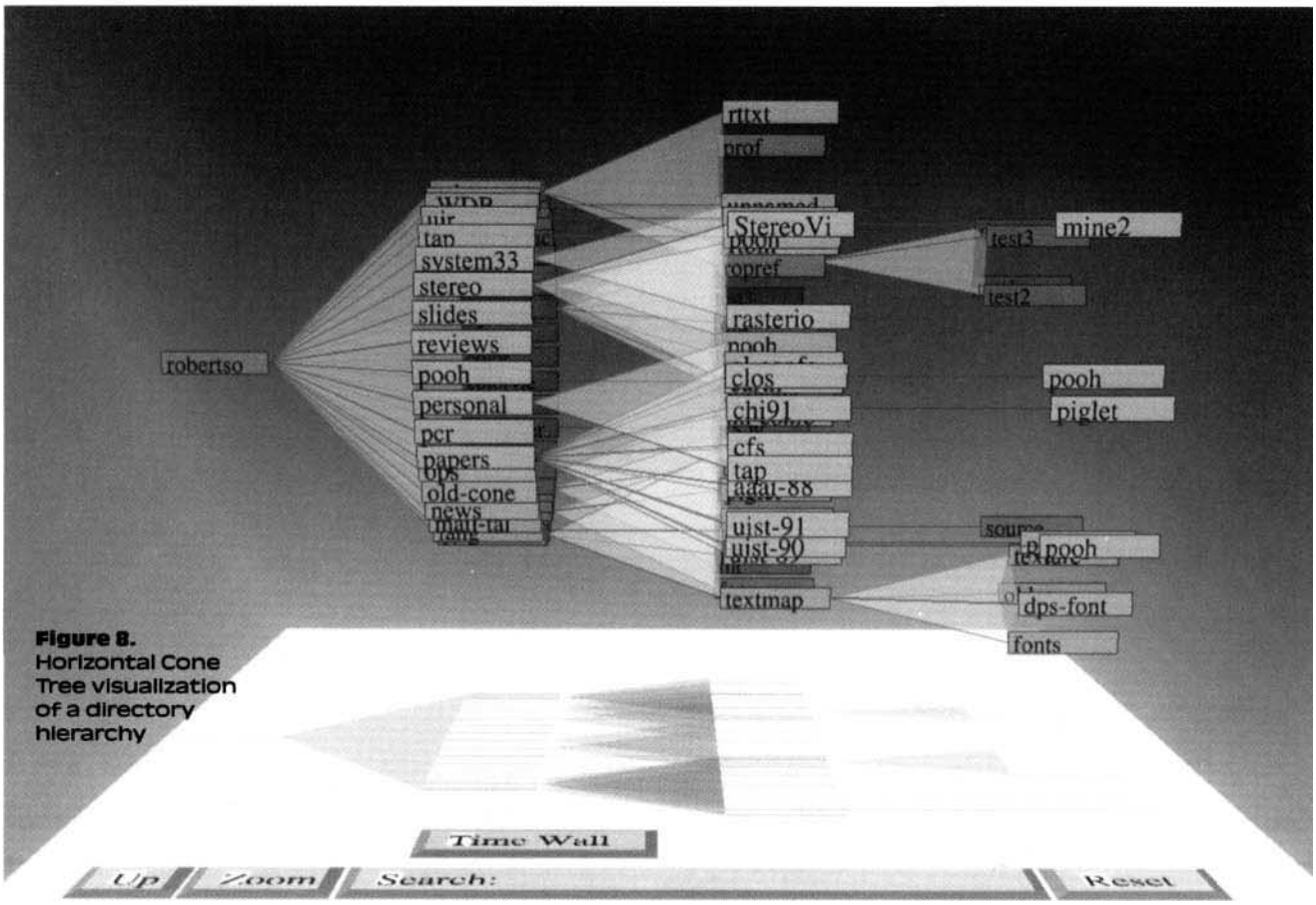


Figure 8. Horizontal Cone Tree visualization of a directory hierarchy

perceptual system can track. Typically, the entire transformation is done in about a second. The tree can also be rotated continuously to help the user understand substructure relationships.

The hierarchy is presented in 3D to maximize effective use of available screen space and enable visualization of the whole structure. A 2D layout of the same structure using conventional graph layout algorithms would not fit on the screen. The user would have to either scroll through the layout or use a size-reduced image of the structure. Most hierarchies encountered in real applications tend to be broad and shallow. This typical hierarchy aspect ratio is problematic for 2D layouts (a size-reduced image may look like a line with little detail). A 3D layout uses depth to fill the screen with more information.

To see this effect analytically, consider the aspect ratio of a 2D tree, ignoring the size of the nodes. If there are l levels and the branching factor is b, the width of the base is b^{l-1} and the aspect ratio is b^{l-1}/l. Aspect ratio for 2D trees increases nearly exponentially, and is much worse as the branching factor gets larger. Figure 9 shows what happens for small branching factors ($b = 2$ and $b = 3$). In contrast, the Cone Tree aspect ratio is fixed to fit the room by adjusting level height and cone diameters to fit. The line near the bottom of Figure 9 is a typical aspect ratio of four to three. Although fixing the aspect ratio introduces a limitation on the number of levels that can be effectively displayed (about 10), it makes Cone Trees independent of the number of nodes, branching factor, and number of levels (until the limit is reached).

In addition to perceptual effects already mentioned, the 3D perspective view of Cone Trees provides a fisheye view [8] of the information, without having to describe a degree of interest function, as in general fisheye view mechanisms. The selected path is brighter, closer, and larger than other paths, both because of the 3D perspective view and because of coloring and simulated lighting. SemNet [6] also reported a fisheye view effect from their use of 3D perspective. Our fisheye view effect is further enhanced by selection rotation, because the user can easily select a new object of interest and have the structure quickly reconfigure to highlight it.

Cone Tree: Examples

In Figures 7 and 8, Cone Trees are used for a file browser, showing one user's directory hierarchy, with each node representing a directory in a Unix file system. Information access is done on file names and file contents. We have also visualized an entire Unix directory hierarchy, which contained about 600 directories and 10,000 files. To our knowledge, when we did this in 1989, it was the first time anyone had ever visualized an entire Unix file system. The directory hierarchy was surprisingly shallow and unbalanced. Since then, both FSN [24] and TreeMaps [11] (a 2D visualization technique) have been used to visualize entire file systems.

Cone Trees have also been used as an organizational structure browser. Search is done in a database of facts about each person (e.g., title or office location) and a database of autobiographies. Users can search for other people with biographies similar to a selected person's biography. We have implemented several organization charts. The largest contained the top 650 Xerox Corporation executives. Since this requires 80 pages on paper, this is the first time the organization chart could be seen in one visualization.

We also have used Cone Trees to visualize a company's operating plan. Text narratives describe each portfolio, program, and project, and are augmented with project highlights (brief statements of milestones and achievements) from the previous year. A typical search finds all projects related to a selected project. Cone Tree manipulation mechanisms are used during early stages of operating plan definition to reorganize the plan to a desired structure.

Other potential applications include software module management, document management (library structure and book structure), object-oriented class browsers, and local area network browsers.

Visualizing Linear Structures: Perspective Wall

Case studies indicate that tasks often involve *spanning* properties (such as time) that structure information linearly [14]. This linear structure results in 2D layouts with wide aspect ratios that are difficult to accommodate in a single view. The principal obstacles to a visualization of linear information structures are (1) the large amount of information that must be displayed and (2) the difficulty of accommodating the extreme aspect ratio of the linear structure on the screen. These problems make it difficult to see details in the structure while retaining global context.

A common technique for viewing linear information while integrating detail and context is to have two simultaneous views: an overview with a scale-reduced version of a workspace, and a detailed view into the workspace where work can be accomplished. The overview typically contains an indication of the detailed view's location that can be manipulated for rapid movement through the workspace. However, a uniform scale reduction of the workspace causes it to appear very small. Furthermore, important contextual information, such as the neighborhood of the viewing region, is just as small as unimportant details. Finally, if the display space for the overview is increased to make the workspace appear larger, the space for the working view becomes too small.

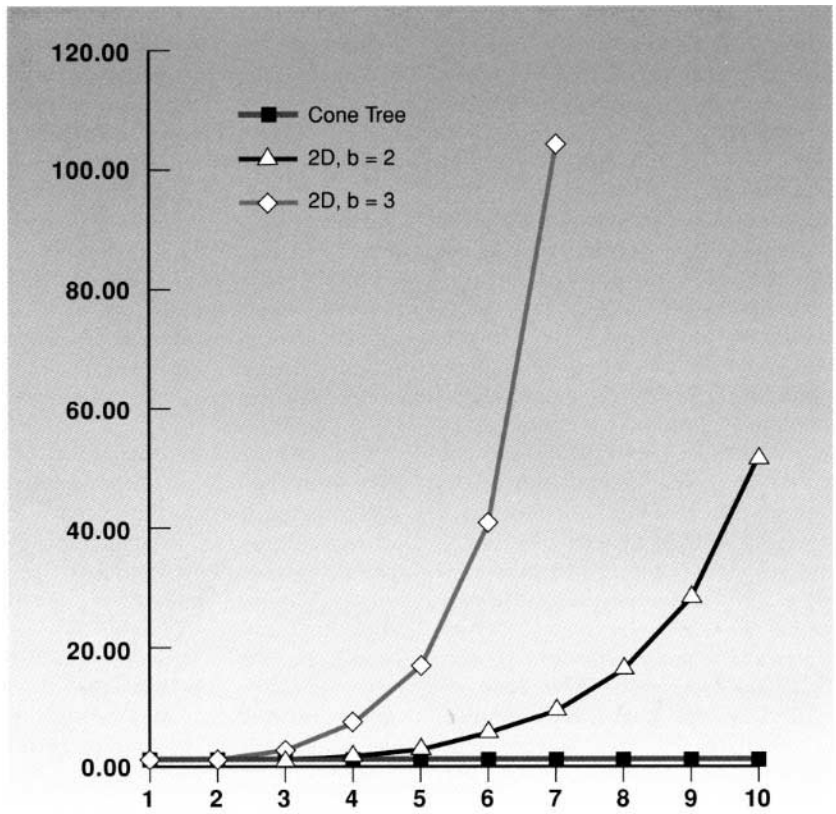
Rather than a uniform overview of a workspace, an effective strategy is to distort the view so that details and context are integrated. Fisheye views [8] provide such distorted views by thresholding with Degree of Interest functions to determine the contents of the display. However, thresholding causes the visualization to have gaps that might be confusing or difficult to repair. Furthermore, gaps can make it difficult to change the view. The desired destination might be in one of the gaps, or the transition from one view to another might be confusing as familiar parts of the visualization suddenly disappear into gaps.

Spence and Apperley developed an early system called the Bifocal Display that integrates detail and context through another distorted view [22]. This 2D design is a conceptual ancestor of the Perspective Wall system. The Bifocal Display was designed for professional offices that contain information subdivided into a hierarchy of journals, volumes, issues and articles. Abstractly, the workspace consists of information items positioned in a horizontal strip. The display is a combination of a detailed view of the strip and two distorted views, where items on either side of the detailed view are distorted horizontally into narrow vertical strips. For example, the detailed view might contain a page from a journal and the distorted view might contain the years for various issues of the journal.

The *Perspective Wall* integrates detailed and contextual views to support the visualization of linearly structured information spaces, using interactive 3D animation to address the integration problems of the Bifocal Display. The Perspective Wall folds a 2D layout into a 3D wall that smoothly integrates a central region for viewing details with two perspective regions, one on each side, for viewing context (see Figure 10). This intuitive distortion of the layout provides efficient space utilization and allows smooth transitions of views. Space utilization analysis of the Perspective Wall technique indicates at least a three-fold improvement over simple 2D visualization (see [14] for details).

Perspective Wall: Implementation

The Perspective Wall's physical metaphor of folding is used to distort an arbitrary 2D layout into a 3D visualization (the wall), while automatically retaining any 2D task-specific features. More important, no special large- and small-scale versions of items must be designed (as in the Bifocal Display). The perspective panels are also shaded to enhance the perception of 3D. This intuitive visualization provides efficient space utilization for 2D layouts with wide aspect ratios. In addition, the vertical dimension of the wall can be used to



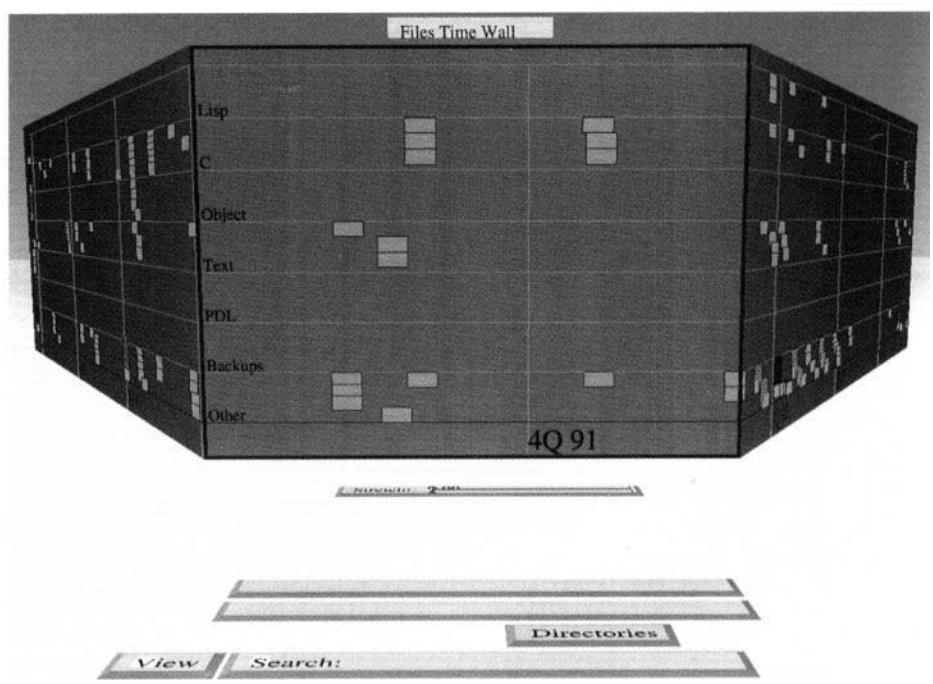


Figure 9. Aspect ratio of 2D and 3D trees

Figure 10. Perspective Wall visualization of files

visualize layering in an information space. The Perspective Wall in Figure 10 holds cards that represent files in a computer system that are structured by modification date (horizontally) and file type (vertically). The perspective view has the further advantage that it makes the neighborhood of the detailed view larger than more distant parts of the contextual view.

A major advantage of the Perspective Wall is that its intuitive 3D metaphor for distorting 2D layouts allows smooth transitions among views. When the user selects an item, the wall moves that item to the center panel with a smooth animation, as if it were a sheet in a player piano moving selected notes to the center of view. This animation helps the user perceive object constancy, which shifts to the perceptual system work that would otherwise have been required of the cognitive system to reassimilate the view after it had changed. Furthermore, the relationship between the items in the detail and context is obvious. Items even bend around the corner.

The Perspective Wall has the additional feature that the user can adjust the ratio of detail and context. This is quite important when the detailed view contains a lot of information. The metaphor is to stretch the wall like a sheet of rubber.

The Perspective Wall has been used to visualize various types of information. Figure 10 represents files in a file system that are classified by their modification date and file type. Vacations and other work patterns are clearly visible. The technique has also been used for corporate memoranda and reports, which also have a useful linear structure. The technique is particularly effective when combined with a retrieval technique that allows the user to select an item and find similar related items. The Perspective Wall makes it easy to visualize the results of such retrievals because it shows all similar items simultaneously and in context.

Summary

To summarize, we believe that the structure of information, the emerging technologies of 3D and interactive animation, and the human perceptual system can be effectively exploited to improve management of and access to large information spaces. There is a large class of applications for which these techniques work. It seems clear that interactive animation can effectively shift cognitive processing load to the perceptual system. And it seems plausible (but not yet proved) that 3D can be used to maximize effective use of screen space. Formal user studies are needed to verify and expand on these conclusions.

The Information Visualizer we have described is an experimental system being used to develop a new UI paradigm for information retrieval, one oriented toward the amplification of information-based work. It is based on our analysis of several aspects of information use that have led us to reframe the information retrieval problem as a problem in the cost structuring of an information workspace. This, in turn, has led us to evolve the computer desktop metaphor toward (1) the Cognitive Coprocessor interaction architecture (to support highly coupled iterative interaction with multiple agents), (2) 3D/Rooms (to manage information storage cost hierarchies), and (3) information visualization (to increase the level of information abstraction to the user). Collectively these techniques alter the cost of retrieving information from secondary storage and the cost of using it in workspace. Future work will focus on how these techniques aid in sensemaking, and will continue to explore the rich space of techniques for visualizing information.

Acknowledgments

We would like to thank a number of people who have been involved in the use and development of the Information Visualizer: Doug Cutting, Peter Pirolli, Kris Halverson, Walt Johnson, Jan Pedersen, Ramana Rao, Dan Russell, Mark Shirley, Mark Stefik, and Brian Williams.

References

1. Card, S.K., Moran, T.P. and Newell, A. *The Psychology of Human-Computer Interaction.* Erlbaum, Hillsdale, New Jersey, 1983.
2. Card, S.K., Robertson, G.G. and Mackinlay, J.D. The Information Visualizer: An information workspace. In *Proceedings of SIGCHI'91,* 1991, pp. 181–188.
3. Conner, D.B., Snibbe, S.S., Herndon, K.P., Robbins, D.C. and Zeleznik, R.C. Three-dimensional widgets. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics,* 1992, pp. 183–188.
4. Cutting, D.R., Karger, D.R., Pedersen, J.O. and Tukey, J.W. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of SIGIR'92,* 1992, pp. 318–329.
5. Cutting, D.R., Pedersen, J.O. and Halvorsen, P.K. An object-oriented architecture for text retrieval. In *Proceedings of RIAO'91, Intelligent Text and Image Handling,* 1991, pp. 285–298.
6. Fairchild, K.M., Poltrock, S.E. and Furnas, G.W. Semnet: Three-dimensional graphic representations of large knowledge bases. In *Cognitive Science and its Applications for Human-Computer Interaction,* Guindon, R. Ed, Lawrence Erlbaum, 1988.
7. Feiner, S. and Beshers, C. Worlds within worlds: Metaphors for exploring *n*-dimensional virtual worlds. In *Proceedings of the UIST'90,* 1990, pp. 76–83.
8. Furnas, G. W. Generalized fisheye views. In *Proceedings of SIGCHI'86,* 1986, pp. 16–23.
9. Green, M. and Jacob, R. SIGGRAPH'90 Workshop Report: Software architectures and metaphors for non-WIMP user interfaces. *Comput. Graph. 25,* 3 (July 1991), 229–235.
10. Henderson, D.A. and Card, S.K. Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Trans. Graph. 5, 3,* (July 1986), pp. 211–243.
11. Johnson, B. and Shneiderman, B. Space-filling approach to the visualization of hierarchical information structures. In *Proceedings IEEE Visualization '91,* 1991, pp. 284–291.
12. Mackinlay, J.D., Card, S.K. and Robertson, G.G. Rapid controlled movement through a virtual 3d workspace. In *Proceedings of SIGGRAPH '90,* 1990, pp. 171–176.
13. Mackinlay, J.D., Card, S.K. and Robertson, G.G. A semantic analysis of the design space of input devices. *Human-Computer Interaction, 5, 2–3,* 1990, pp. 145–190.
14. Mackinlay, J.D., Robertson, G.G. and Card, S.K. Perspective wall: Detail

and context smoothly integrated. In *Proceedings of SIGCHI'91,* 1991, pp. 173–179.

15. Newell, A. *Unified Theories of Cognition.* Harvard University Press, Cambridge, Mass., 1990.
16. Rao, R., Card, S.K., Jellinek, H.D., Mackinlay, J.D. and Robertson, G.G. The information grid: A framework for information retrieval and retrieval-centered applications. In *Proceedings of UIST'92,* 1992, pp. 23–32.
17. Robertson, G.G., Card, S.K. and Mackinlay, J.D. The Cognitive Coprocessor architecture for interactive user interfaces. In *Proceedings of UIST'89,* 1989, pp. 10–18.
18. Robertson, G.G., Mackinlay, J.D. and Card, S.K. Cone Trees: Animated 3D visualizations of hierarchical information. In *Proceedings of SIGCHI'91,* 1991, pp. 189–194.
19. Salton, G. and McGill, M.J. *Introduction to modern information retrieval.* McGraw-Hill, New York, 1983.
20. Shaw, C., Liang, J., Green, M. and Sun, Y. The decoupled simulation model for virtual reality systems. In *Proceedings of SIGCHI'92,* 1992, pp. 321–328.
21. Sheridan, T.B. Supervisory control of remote manipulators, vehicles and dynamic processes: Experiments in command and display aiding. *Advances in Man-Machine System Research 1,* 1984, JAI Press, 49–137.
22. Spence, R. and Apperley, M. Data base navigation: An office environment for the professional. *Behavior Inf. Tech. 1,* 1 (1982), 43–54.
23. Sutherland, I.E. Sketchpad: A man-machine graphical communication system. MIT Lincoln Laboratory Tech. Rep. 296, May 1965. Abridged version in SJCC 1963, Spartan Books, Baltimore, Md., 329.
24. Tesler, J. and Strasnick, S. FSN: 3D Information Landscapes. Man pages entry for an unsupported but publicly released system from Silicon Graphics, Inc. Mountain View, Calif. Apr. 1992.
25. Zeltzer, D., Pieper, S. and Sturman, D. An integrated graphical simulation platform. In *Proceedings of Graphics Interface '89,* 1989.

CR Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*retrieval models*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*interaction styles*; I.3.6 [**Computer Graphics**]: Methodology and Techniques—*interaction techniques.*

Additional Key Words and Phrases: Information visualization, information visualizer, information access, information cost structure

About the Authors:

GEORGE G. ROBERTSON is a principal scientist at Xerox Palo Alto Research Center.

STUART K. CARD is a research fellow and manager of the user interface research group at Xerox Palo Alto Research Center.

JOCK D. MACKINLAY is a member of the research staff at Xerox Palo Alto Research Center.

Authors' Present Address: Xerox PARC, 3333 Coyote Hill Road, Palo Alto, CA 94304; email: {robertson, card, mac kinlay} @parc.xerox.com

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© ACM 0002-0782/93/0400-056 $1.50

in-ter-op-er-a-bil-i-ty \,int-ər-,äp-(ə-)rə-'bil-ət-ē\ ***n.*** (1993)
1. the existence of consistent processes and interfaces across disparate computer platforms. Extremely difficult to achieve in distributed applications.
Until now.

ADA DEVELOPERS

Introducing Network Broker. The environment for developing distributed applications.
Network Broker manages the complexities of inter-communication among heterogeneous or homogeneous networks. Complexities that are often unavoidable such as different vendor's protocols, operating systems, and host computers. By using an intelligent software environment — the Virtual Network — Network Broker exploits the inherent concurrency in distributed systems, and insulates your working applications from their underlying platforms.

The result — connectivity and interoperability. Pure and simple.
And that means less development time. Since Network Broker handles the communications aspect of the programming task, your time is spent building your application. So there's less coding in the front door, and reduced maintenance in the back door.

First implemented in 1988, Network Broker is a proven development tool designed specifically for use with the ADA programming language. Network Broker for C - language developers will be available later this year.

To find out how Network Broker can simplify the design, implementation, integration, and maintenance of your distributed applications, call our Software Engineering Division directly, (410) 312-2259.

Ideas
7120 Columbia Gateway Dr.
Columbia, MD 21046
(410) 312-2000
FAX (410) 312-2250

Circle #97 on Reader Service Card