Cone Trees: Animated 3D Visualizations of Hierarchical Information

G. G. Robertson, J. D. Mackinlay and S. K. Card

UIR-R-1991-06

Cone Trees: Animated 3D Visualizations of Hierarchical Information

George G. Robertson, Jock D. Mackinlay, and Stuart K. Card
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
415-494-4755, robertson.parc@xerox.com

Abstract

The task of managing and accessing large information spaces is a problem in large scale cognition. Emerging technologies for 3D visualization and interactive animation offer potential solutions to this problem, especially when the structure of the information can be visualized. We describe one of these *Information Visualization* techniques, called the *Cone Tree*, which is used for visualizing hierarchical information structures. The hierarchy is presented in 3D to maximize effective use of available screen space and enable visualization of the whole structure. Interactive animation is used to shift some of the user's cognitive load to the human perceptual system.

Keywords: User-Interface Design Issues: *visual output strategies, interface metaphors, graphic presentations, screen layout.* Analysis Methods: *analysis of contents of particular domains.* Domain Specific Designs: *information retrieval.*

1 Introduction

Scientific Visualization allows scientists to make sense out of intellectually large data collections. It does so by exploiting the human perceptual system, using animation and visualization to stimulate cognitive recognition of patterns in the data. A similar set of problems exists in the information world, where the volume of information we deal with expands at an astonishing rate. Information access and management is difficult in large information spaces, because it is hard to visualize what is there and how the various parts are related. Emerging technologies for 3D visualization and interactive animation offer potential solutions to these "large scale cognition" tasks, especially when the structure of the information can be visualized and thus exploited. We call these techniques *Information Visualization*, analogous to Scientific Visualization.

The SemNet[2] 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.

In this paper, we focus on visualizing hierarchical information structures rather than arbitrary graphs. 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. We also use interactive animation in addition to 3D visualization to aid in information management and access tasks. Interactive animation reduces cognitive load by exploiting the human perceptual system.

Our animated 3D visualization of hierarchical structures, called *Cone Trees*, is implemented in a prototype system, called the *Information Visualizer*[10]. In addition to describing the look and feel of Cone Trees in this paper, we describe some of the human perceptions evoked by Cone Trees and indicate where these techniques contribute to solving the large scale cognition task in information access and management.

2 A Framework for Information Visualization

The Information Visualizer provides a framework for developing visualization techniques. The framework is driven by the *Cognitive Coprocessor Architecture*[10], which supports management of multiple asynchronous agents and smooth interactive animation. It includes mechanisms for 3D navigation[7, 6] and object manipulation[7], *Interactive Objects* for building the visualizers, and *3D Rooms* for managing information workspaces.

This system is similar to artificial reality systems[3] (in fact, could run in an artificial reality system). However, it works in more conventional 3D workstation environments with conventional 2D input devices (mouse). The user still enters into the 3D environment and is engaged in navigation and manipulation as in artificial reality systems, but rather than wear special equipment (a helmet and glove), the user is drawn into the world by the nature of perceptual cues and interactive animation. Information visualization is intended for an audience (office workers, for example) that is not likely to be willing to wear special equipment.

The 3D Rooms system supports a collection of 2D and 3D rooms. The user walks around these rooms and interacts with artifacts which represent information or information structure. The information workspace is a simulation of a physical space. This familiar metaphor makes it easy to learn and use the system. Workspace management is similar to the Rooms system[5], except that a visualization replaces a collection of windows.

Information access is made ubiquitous by embedding retrieval mechanisms in the *desktop*, much as *Editor Top Level* systems make editing operations ubiquitous. We call this a *Retrieval Top Level* system. Each visualizer supports retrieval, often in the form of relevance feedback search through underlying text databases[1].

The prototype has visualizers for three classes of information: unstructured, linear structures, and hierarchical structures. Linear structures are visualized by folding a 2D layout onto a malleable three-segment 3D wall, called the *Perspective Wall*[8]. Hierarchical structures are visualized with Cone Trees.

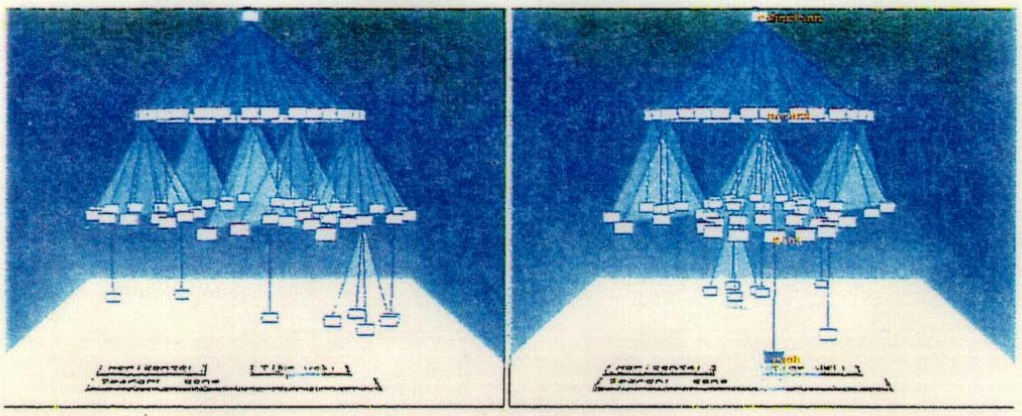

Figure 1: Layout of a simple Cone Tree, before and after selection.

3 Cone Trees: Basic Approach

Cone Trees are hierarchies laid out uniformly in three dimensions. The left side of Figure 1 is a snapshot of a simple Cone Tree. Nodes are drawn like 3x5 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 which insures 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.

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 perceptual system can track. Typically, the entire transformation is done in about one second. The right side of Figure 1 shows the Cone Tree after a selection rotation is completed. The tree can also be rotated continuously to help the user understand substructure relationships.

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 2 shows an alternative layout, called the *Cam Tree*, which is horizontally oriented and has text displayed for each node. The right side of the figure shows the appearance after selection rotation, where highlighting is done with node color rather than text.

Interactive animation is used to shift some of the user's cognitive load to the human perceptual system. Consider what would happen if node selection displayed the rotated structure without animation. Since the rotations are complex, the user would take several seconds to reassimilate the relationships between substructures. However, animation allows the perceptual system to track the rotations. The perceptual phenomenon of object constancy enables the user to track substructure relationships without thinking about it. When the animation is completed, no time is needed for reassimilation.

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

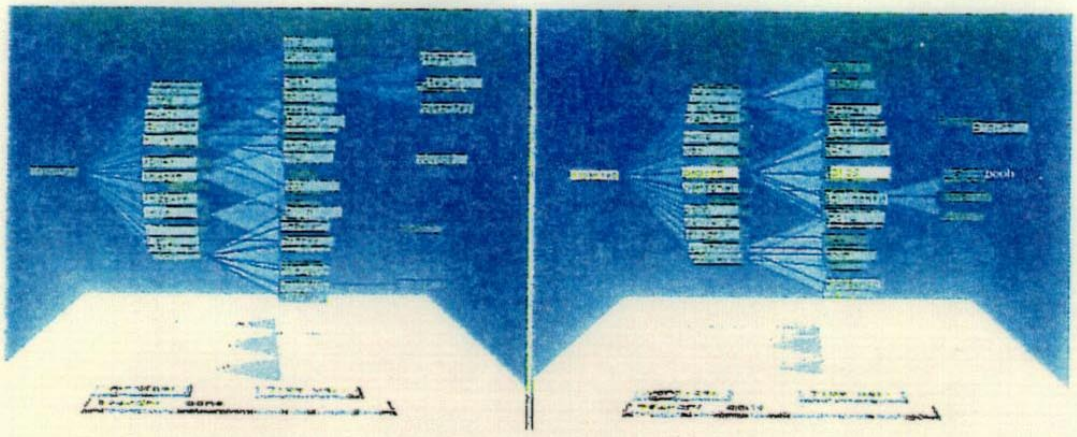

Figure 2: Layout of a simple Cam Tree, before and after selection.

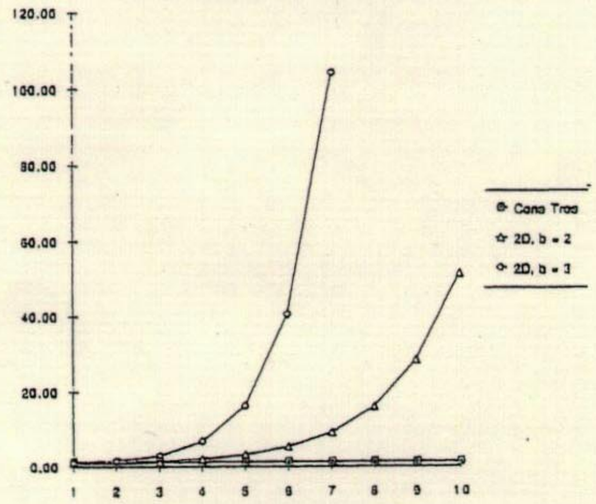

Figure 3: Aspect Ratio of 2D and 3D Trees.

conventional graph layout algorithms[9] 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 tends to be 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 3 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 3 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).

4 User Perceptions

In addition to perceptual effects already mentioned, several other user perceptions enhance the effectiveness of Cone Trees. For example, the 3D perspective view of Cone Trees provides a fisheye view[4] 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 that other paths, both because of the 3D perspective view and because of coloring and simulated lighting. SemNet[2] 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.

The user perception of 3D depth is enhanced in a number of ways. Obviously 3D perspective transformation results in size changes to reflect distance from the viewer. Lighting cues, like lighter coloring of closer nodes and links, also enhances 3D depth perception. Finally, shadows of cones and nodes are cast onto the floor. These are idealized shadows, which provide a 3D depth cue and also convey additional structural information about the hierarchy. This additional information is different for Cone Trees and Cam Trees. In Figure 1, the Cone Tree shadow conveys information about clustering in the hierarchy. Cam Tree shadows (Figure 2) convey more direct information about the hierarchy in the form of a simple 2D projection. While users do not seem to focus directly on the extra information in shadows, it still appears to help in understanding the structure, perhaps subliminally.

Interactive animation's primary perceptual effect, as described above, is the reduction of cognitive load by exploiting the human perceptual system. It also brings the interface and the information to life, making the tasks more enjoyable. In addition, it helps the user to understand the information structure more completely. By observing continuous rotations of the structure, the user gains insights into the relationships between substructures. This technique is also used in CAD/CAM systems, which provide ways to rotate objects to gain insights into their structure.

5 Gardening, Manipulation, and Search

The basic look and feel of Cone Trees and some of their perceptual properties have been described. In this section, we describe several additional operations that can be performed on Cone Trees, including techniques for viewing parts of trees, restructuring trees dynamically, and searching through underlying information. These allow the user to further explore and manipulate the structure of the information being visualized.

For a complex hierarchy, users often need ways of hiding selected parts to focus on a particular substructure. We provide operations for pruning and growing the view of the tree, collectively called *gardening* operations. Prune and grow operations are done either by menu or by gestures directed at a node. If a user flicks a node toward the top node, all of its descendants are hidden from view. The pruned node is modified so that a *grow tab* appears below it (or to its right for Cam Trees). Growing the children of a pruned node back into view is done either by flicking that node away from the top node or by clicking on the grow tab. To focus on one particular substructure, the *prune others* menu operation prunes all the siblings of the selected node, leaving only the desired substructure visible. These gardening

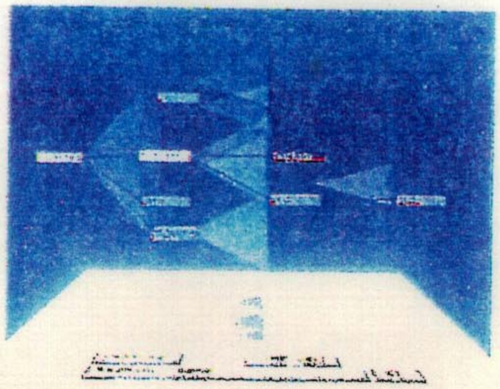

Figure 4: Result of a Search Operation.

operations make it easier to manage and understand large, complex hierarchies.

For applications where the user wishes to change the structure of the hierarchy dynamically, techniques are needed for directly rearranging the Cone Tree. The 3D manipulation techniques described in [6] are used to grasp a node and drag the substructure rooted at that node to a new position. Feedback is provided by highlighting the body of the target cone. When the grasped node is released, it is moved to its new position.

As mentioned earlier, information retrieval is always available. Search parameters are provided either by node selection or parameters typed into a pop-up property sheet. Search is initiated by menu command, and is restricted to currently visible nodes. The search operation typically takes place in another process to avoid degrading interactive animation and to allow the user to continue work during the search. When a search starts, all nodes are made invisible. During the search, a progress indicator shows how much of the search remains, and partial search results are shown by making nodes visible. A node is highlighted with a red bar, whose size indicates its relative search score. Figure 4 shows the result of a search in a file browser for files with the string "cone" in the file name. When the search is completed, the node with the highest search score is selected.

6 Application Examples

There are numerous applications with large hierarchical information structures that can take advantage of Cone Trees. We have developed three applications which we will use to illustrate these techniques.

The first example is a file browser. The Cone Tree is used to present the directory structure, with each node representing a directory in a Unix file system. Information access is done on file names and file contents. Figures 1, 2, and 4 show one user's directory hierarchy. We have also visualized an entire Unix directory hierarchy, which contained about 600 directories and 10,000 files. To our knowledge, this is the first time anyone has ever visualized an entire Unix file system. The directory hierarchy was surprisingly shallow and unbalanced.

The second example is a 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.

There are a number of potential applications, including software module management, object-oriented class browsers, document management at the library level and at the book structure level, and local area network browsers.

7 Discussion

Based on analysis, demonstration, and use experience, there are several observations and tentative conclusions we can make, and several issues we can raise.

The clearest win in this technology is interactive animation. It is easy to demonstrate that animation shifts cognitive load to the human perceptual system. While it is difficult to quantify the time it takes to cognitively reassimilate structural relationships after a tree transformation *without* animation, it is clear from a simple demonstration that it is many seconds, and perhaps tens of seconds depending on the complexity of the hierarchy. The time it takes to animate the rotational transformations can be fixed (using the *governor* in the Cognitive Coprocessor Architecture[10]). We typically slow it down to about one second so that the perceptual system can easily track the movement. While the optimal rotation rate with animation is unknown, it is clear that rates which work well result in much shorter times than cognitive reassimilation time without animation. Another clear win with interactive animation is the aid it provides in understanding complex structural relationships. The added insights gained from watching complex structures rotate is well known in the CAD/CAM community.

Using our current techniques, there are limits to the size of the hierarchy that can be effectively displayed. The display becomes cluttered and its effectiveness is reduced with more than about 1000 nodes, 10 layers, or a maximum branching factor of 30. It also appears that Cone Trees are more effective for unbalanced hierarchies. A rotating balanced tree is hard to track because of uniform appearance of substructures. In practice, however, many hierarchies are broad, shallow and unbalanced, and thus fall within these limits.

How much screen space does an optimized 2D layout of the same structure take? In the 650 node organization chart mentioned above, the largest layer has 292 nodes. Assuming an over-optimized 2D layout which ignores link placement, this structure would take about three feet of screen space! This is not surprising considering the aspect ratio analysis described earlier. The user of the 2D layout must scroll through the hierarchy, and cannot see the whole structure at once (unless it is size-reduced, but then the details get obscured). However, it could be argued that rotation of the Cone Tree is just another form of scrolling. Future formal evaluation studies will resolve this question, and reveal the actual gains of using a 3D layout.

How ubiquitous are hierarchies? Many examples of simple hierarchies come to mind, and some have been described here. It is also possible to take any directed graph and, through a depth-first traversal, detect and cut the cycles in the graph. If the cut links are saved and displayed in another form (like dotted lines shown on demand), then a Cone Tree can be used to visualize the resulting tree. For arbitrary graphs with a lot of cycles, this technique may not be satisfactory. However, for graphs with only a few cycles, this may work quite well. A better characterization of the class of graphs adaptable for Cone Tree display is a topic for future work.

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 and access of 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 proven) that 3D can be used to maximize effective use of screen space. Formal user studies are needed to verify and expand on these conclusions.

References

[1] Cutting, D. R., and Pedersen, J. O. Optimizations for dynamic inverted index maintenance. *Proceedings of SIGIR'90,* Brussels, Belgium, ACM Press, September, 1990.

[2] 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.

[3] Foley, J. D. Interfaces for advanced computing. *Scientific American,* October, 1987.

[4] Furnas, G. W. Generalized fisheye views. *Proceedings of CHI'86 Human Factors in Computing Systems,* 16-23. New York: ACM, 1986.

[5] 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 Transactions on Graphics, 5, 3,* 211-243, July, 1986.

[6] Mackinlay, J. D., Card, S. K., and Robertson, G. G. Rapid controlled movement through a virtual 3d workspace. *SIGGRAPH '90 Conference Proceedings* (Dallas, Texas, August 1990). In *Computer Graphics, 24, 4* (August 1990), 171-176.

[7] 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,* 145-190, 1990.

[8] Mackinlay, J. D., Robertson, G. G., and Card, S. K. Perspective wall: detail and context smoothly integrated. Submitted to CHI'91.

[9] Messinger, E. B. Automatic layout of large directed graphs. Technical Report No. 88-07-08, Department of Computer Science, University of Washington, Seattle, Washington, July 1988.

[10] Robertson, G. G., Card, S. K., and Mackinlay, J. D. The Cognitive Coprocessor Architecture for Interactive User Interfaces. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology,* November, 1989, ACM, Williamsburg, Virginia, 10-18.