An Evaluation of Cone Trees

Andy Cockburn[†]
Bruce McKenzie

Department of Computer Science
University of Canterbury
Christchurch, New Zealand

Tel: *+64 3 364 2987 x7768*

Fax: *+64 3 364 2569*

EMail: {*andy, bruce*}*@cosc.canterbury.ac.nz*

URL: `http://www.cosc.canterbury.ac.nz/~andy`

Cone Trees are an appealing interactive 3D visualization technique for hierarchical data structures. They were originally intended to maximise effective use of available screen space and to better exploit the abilities of the human perceptual system. Prior work has focused on the fidelity of the visualization rather than providing empirical user studies. This paper describes the design, implementation and evaluation of a low-fidelity animated and rapidly interactive 3D cone tree system. Results of the evaluation show that our subjects were slower at locating data using cone trees than when using a 'normal' tree browser, and that their performance deteriorated rapidly as the branching factor of the data-structure increased. Qualitative results, however, indicate that the subjects were enthusiastic about the cone tree visualization and that they felt it provided a better 'feel' for the structure of the information space.

Keywords: Cone Trees, visualization, evaluation, navigation, structural views

---

[†] Author for correspondence.

1 Introduction

Interactive visualizations of complex information spaces are designed to improve the ability of users to work with, and navigate through, rich data spaces. Examples include the three-dimensional data displays provided by cone trees (Robertson et al. 1991), the Data Mountain (Robertson et al. 1998), the Perspective Wall (Mackinlay et al. 1991), and the rich 2D displays of the Information Mural (Jerding & Stasko 1998). While the sophistication of visualizations is dramatically increasing, there has been a surprising lack of empirical evaluation to provide evidence of improvements in efficiency and usability.

This paper focuses on cone trees, which were proposed as a visualization technique that would "shift some of the user's cognitive load to the human perceptual system" while maximising the effective use of the available screen space (Robertson et al. 1991). They provide an animated 3D visualization of hierarchical data structures. When a level in the hierarchy is expanded, its contents are arranged around the bottom of an inverted cone. The user's perspective is normally slightly above or below the display, allowing them to 'reach through' the hierarchy to select items. Cones can be rotated to bring data items to the foreground of the display.

Although cone trees have been used in several research systems (for example Cat-a-Cone (Hearst & Karadi 1997) and LyberWorld (Hemmje et al. 1994)) a fundamental limitation of the work on cone trees is the lack of empirical investigation into their usability and effectiveness. As Robertson et al. (1991) state, "Formal user studies are needed". With the exception of a small study by Carriere & Kazman (1995) which compared a cone tree system with a command-line interface, we are unaware of any prior attempt to quantitatively evaluate cone trees.

In this paper we describe the quantitative and qualitative evaluation of a small cone tree system. Although the interface fidelity of our implementation is low when compared to that of prior work, our focus is on creating a rapidly interactive interface that contains the full set of 3D functionality of previous systems. Our research objective is to produce reliable and repeatable results of usability.

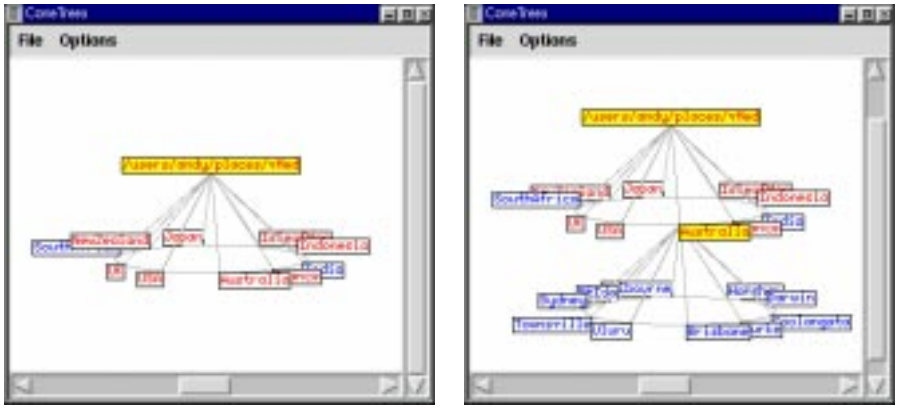
The structure of the paper is as follow. First, we describe the interface and functionality of our cone tree implementation. Next, the evaluation method and results are described, followed by implications for further and related work. Conclusions are then presented.

2 A Minimal Cone Tree Implementation

Our cone tree system is designed to support users navigating through hierarchical file systems[‡]. Figure 1(a) shows a single cone displaying the contents of the directory "`/users/andy/places/vMed`". Directories are distinguished from files by text colour; directories are red, files are blue. Directory contents are arranged around the cone in alphabetical order in an anti-clockwise direction from the front[§]. To explore the contents of any directory (or to hide its contents) the user clicks on the directory name with the middle mouse-button, as shown in Figure 1(b). To help the user

[‡]It would be trivial to modify the system to work with almost any hierarchical data.

[§]This order is the default produced by the Unix command `ls` which is used as our control in the evaluation.



(a) One cone. (b) Expanding one directory.

Figure 1: Basic cone tree interface.

identify the text-tags associated with expanded directories in cluttered displays, their background is coloured yellow ("`/users/andy/places/vMed`" and "`Australia`" in Figure 1(b)). When a parent cone is contracted and re-expanded, its previous display state is immediately restored, including the rotation and expansion state of all child cones in the hierarchy.

User controlled cone rotation is a fundamental requirement in supporting exploration of the data contained in the visualization. It is essential that rotation is both rapid and animated. Rapidity is necessary to ensure that users see rotation as a 'lightweight' operation that does not disrupt their interaction with the system. Animation is needed to help the user maintain their sense of orientation within the data-space by easing the mapping between the pre- and post-rotation states. To rotate a cone, the user clicks on a text item with the left mouse button. The clicked item immediately turns green (to highlight the target), and the cone is smoothly rotated in the direction yielding the smallest angle change until the target is at the front of the cone. A configuration option allows users to choose whether child cones are dynamically rotated with the parent (providing a smooth animated display of the rotation of the entire hierarchy) or are redisplayed only when the animation of the parent cone is complete. This is a user-controlled trade-off between the fidelity of the visualization and the speed of interaction. In our evaluations, all rotations took less than half a second, and the dynamic rotation option was disabled.

The presentation, placement and management of text identification tags in cone trees raises several complex interface design trade-offs. Our system uses a variety of techniques to ease the conflict between display-space clutter and support for item identification. By default, every item in every cone has a non-transparent text label that is implicitly raised above all other labels when the mouse cursor enters it. Non-transparent labels have the disadvantage that they can occlude other items, but our

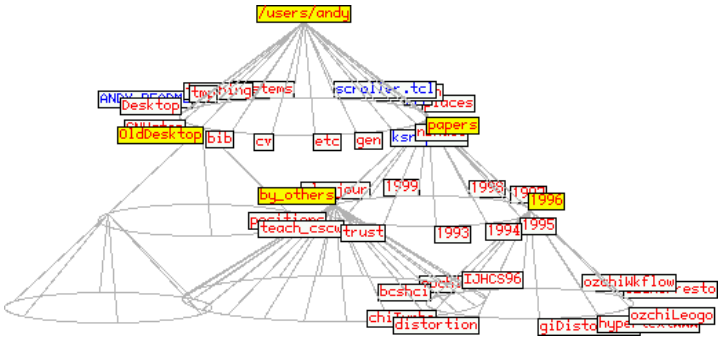


Figure 2: A hierarchy viewed with the “One named cone per level” option.

(a) Side-on. (b) Tilted. (c) Vertically above.

Figure 3: Changing the user’s perspective.

experiments with opaque and translucent labels showed that they greatly aid text legibility. In dense cones, many text labels can be displayed in close proximity, particularly at a cone’s left and right extremities. This makes the implicit item raising feature difficult to use. In such circumstances the user can ‘shuffle’ the stacking order of text labels by right-clicking an item with the mouse.

A variety of configuration options allow the user to tailor the cone visualization, including the following:

Constant/Variable cone radius — The variable cone radius option makes each cone’s radius dependent on the number of items contained in the cone. This reduces the occurrence of occluded text items within each cone, but increases the occurrence of overlapping cones when several dense cones are displayed at the same level in the hierarchy.

One named cone per level — When the user displays more than one cone at any level in the hierarchy, portions of the cones are likely to collide. Although layout algorithms can reduce collisions between cone text labels (see (Carriere & Kazman 1995)), collisions cannot be completely avoided because of the effects of cone rotation. For instance, child cones that are displayed with maximum separation

(the left and right extremities of the cone) will collide and occlude each other at the front and back positions if the parent is rotated through 90 degrees.

To ease this problem, the user can select a "One named cone per level" option which removes the text labels from all but the most recently expanded cone at each level. To maintain the user's sense of structure in the data space, the construction lines of all cones are displayed as normal (Figure 2).

User's perspective angle — To enhance the 3D interaction capabilities of the system, the user can shift their vertical angle of perspective. This is controlled by a slider widget that ranges from a side-on view (Figure 3(a)), through tilted views (Figure 3(b)), to a vertically downward view (Figure 3(c)).

Cone height — The final configuration option allows users to control the height of each cone.

Our cone tree interface also allows the user to view and manipulate the hierarchy using a normal holophrasting (Smith et al. 1984) tree browser similar to those used for file management in Windows Explorer and MacOs (see Figure 4). All interface controls and representation schemes are the same in both interfaces: the middle mouse button expands and contracts directories, text labels for directories are coloured red while those for files are coloured blue, and the background of the text labels of expanded directories are coloured yellow. This interface is used in the control for the experiment described in the following section.

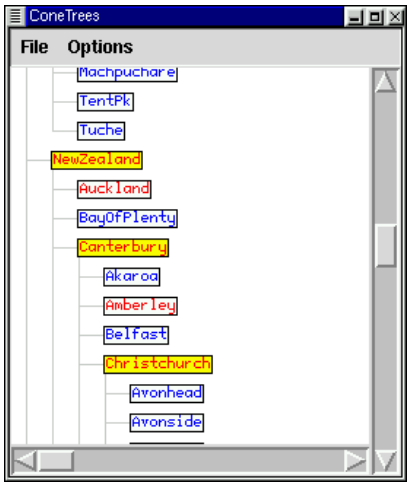


Figure 4: The 'tree browser' control interface.

3 Evaluation

The aim of the evaluation is to compare the efficiency of the cone tree interface for simple hierarchical navigation tasks with that of 'normal' tree interfaces, and to detect qualitative usability issues. We are also interested in the support provided by

cone trees for navigating through hierarchies of different depths (number of levels in the hierarchy) and densities (number of items in each cone).

The experiment is a three-way factorial design with the following independent variables: depth, density, and interface type. The depth variable has two levels: 'shallow' for search paths of length two (for example, "Find California in the USA"), and 'deep' for search length four (for example, "Find Newport in Los Angeles in California in the USA"). The density variable has three levels: 'sparse', 'medium' and 'dense' for fan-out values of 6, 10 and 20 files or directories contained within each directory. Robertson et al. (1991) stated that cone trees become cluttered and ineffective at a branching factor of 30, so our choice of 20 as a maximum branching factor is fairly conservative. The two levels in the interface type variable are 'cone tree' and 'normal tree'.

Each subject completed seven paired tasks, with the pairing repeating the same task for the cone and normal tree interfaces. Experimental conditions for each task are summarised in Table 1. The first six task pairs covered all combinations of the independent variables, and they tested the subjects' ability to navigate through a stated series of directories to locate a named file. Tasks 1 to 6 all start from a fully contracted state with only the top-level directory name showing. Task 7 interleaved the subjects' completion of Task 6 (the deep and dense condition). In contrast to Tasks 1 to 6, Task 7 starts from an expanded system state, testing the subjects' ability to return to a previously visited file (the one visited in Task 1). The interleaved order between Tasks 6 and 7, then, was 6.1, 7.1, 6.2, 7.2, and the system state is only reset to a fully contracted state prior to Tasks 6.1 and 6.2.

The interface order used for each task (cone, then normal, or vice-versa) was varied between each task and between each subject to control any learning effect. In addition, we tested the response times to ensure that order had no significant effect on task completion times (as summarised in the results).

In selecting the navigational paths for each task, three sample paths were generated for each depth (see right hand column of Table 1). These tasks were then rotated round the three levels of the density independent variable for each of the subjects. We could not use random selection of navigational paths because of the difficulty of providing a familiar hierarchy with fan-out of 20 to four levels of depth (requiring 160,000 meaningful place names).

Task	**Depth**	**Density**	**Paths (rotated between Tasks 1–3 and 4–6)**
1	Shallow	Sparse(6)	France to Buoux
2		Medium(10)	Australia to Sydney
3		Dense(20)	IsleofMan to Chasms
4	Deep	Sparse(6)	UK to Yorkshire to York to Shambles
5		Medium(10)	USA to California to LosAngeles to Newport
6		Dense(20)	NewZealand to Canterbury to Christchurch to Sumner
7	Shallow	Dense(20)	Return to the Task 1 file

Note: Task 7 is interleaved with Task 6

Table 1: Conditions for the seven pairs of tasks.

Task completion times were measured by the software which recorded all user actions and their timing.

3.1 Subject Details and Treatment

Twelve volunteer subjects, all Computer Science professionals or graduate students, took part in the evaluation. Each subject participated in a single half-hour session.

The first ten minutes of each session was spent introducing the subject to the two interfaces. Most of the subjects were familiar with the normal tree style of interface, but they were given several sample tasks to ensure they were familiar with the mouse bindings used to display and hide directory contents. The subjects were also given sample tasks in the cone tree interface, and they were encouraged to experiment with the interface mechanisms for rotating, expanding and contracting cones, as well as the techniques for shuffling text-labels and scrolling round the display. The various configuration options, such as the ability to change the user's angle of perspective, were neither described nor used in the evaluation. The introductory session continued until the subjects reported that they felt comfortable with both interfaces.

The subjects were informed that they would carry out fourteen tasks involving following, as quickly as possible, a specified series of directories to reach a particular file. They were told that the file structure was a geographical hierarchy of countries, regions, cities and places.

Each task was introduced to the subject via a graphical user interface that controlled the experimental condition (the selection of the navigational path and the order of the interfaces). It displayed the path that the subject had to navigate through in large red text, and the subjects were asked to read the path out loud prior to being told to start.

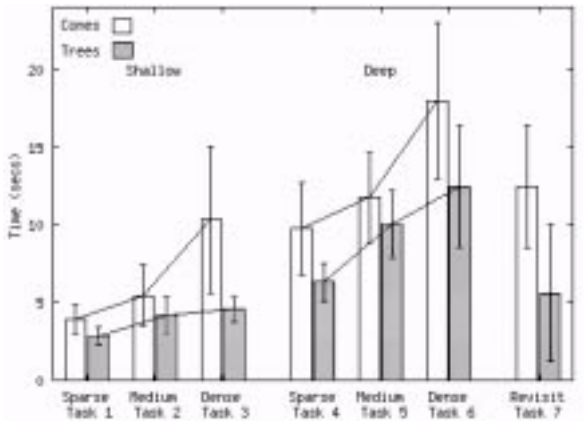
After each task the subjects recorded their level of agreement with two statements: "It was easy to see the data needed", and "Overall the normal/cone tree interface was effective for the task." Responses were recorded using a five-point Likert scale. Once all of the tasks were completed, the subjects were asked for comments about the interfaces.

4 Results

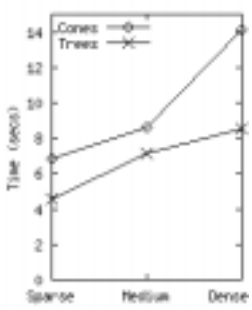
This section reports the quantitative results of the experiment. Qualitative results are reported in the following section.

All of the subjects were able to complete the tasks fairly rapidly. The mean time for task completion across all conditions was 8.3 seconds, and the minimum and maximum times across all tasks were 1.9 seconds (using the normal tree in Task 1) and 29.2 seconds (using the cone tree in Task 6). Figure 5(a) shows the means and standard deviations for the two interfaces in each of the tasks.

Three-way analysis of variance (ANOVA) with repeated measures reveals that the main effects for each of the independent variables are significant. The mean task times for the normal tree and cone tree interfaces were 6.7 and 9.9 seconds, providing a reliable difference ($F(1,11) = 71.98$, $p < .001$). The means for the shallow and deep conditions were 5.2 and 11.4 seconds ($F(1,11) = 133.9$, $p < .001$). Finally, the means



(a) Mean task completion times and standard deviations in seconds for the cone tree and normal tree interfaces across density and depth independent variables.



(b) The interaction between factors interface type and density.

Figure 5: Results.

for the three levels of the density factor were also significantly different at 5.7, 7.9 and 11.3 seconds for the levels sparse, medium and dense ($F(2,22) = 64.1$, $p < .001$). Summarising the main effects: the subjects followed the paths more quickly using the normal-tree interface than when using the cone tree interface; unsurprisingly, they took more time to follow deep navigational paths than shallow paths; finally, the subjects took longer to follow paths in densely populated data sets.

There is a significant interaction between the density and type independent variables ($F(2,22) = 12.8$, $p < 0.001$). The effect of the interaction is clear in Figure 5(b): using the cone interface, the time taken for the dense condition increases dramatically from that for the medium and sparse conditions, while using the normal tree interface, there is a more gradual increase between the sparse, medium and dense conditions. A Tukey honest significant difference for this interaction is 7.2 (maintaining α at the 0.05 level), revealing that only the mean for the 'dense' and 'cone' condition yields significant differences from any other condition.

There is no interaction between the depth and type factors ($F(1,11) = 0.77$, $p=0.4$). This shows that while the subjects took longer to complete deep tasks than shallow ones, there were no significant differences in the rate of degradation in their performance between the two interfaces.

The main effects and interactions detailed above describe the subjects' behaviour in Tasks 1 to 6, which involved following a single path starting with a single unexpanded directory in an otherwise clear display. In contrast, Task 7 investigates the differences in the support for following paths when starting from a partially explored representation of the data-structure.

The mean time for completing Task 7 with the cone interface (12.5 seconds, s.d. 3.9) was more than twice that using the normal tree interface (5.6 seconds, s.d. 4.4), providing a significant difference (paired T-Test, $t(11) = 4.4$, $p < 0.01$).

Our experimental design assumed that the interface order used for each task would not significantly influence the subjects' task completion time. Our control for order involved varying the interface that each subject used first, both between tasks and between subjects. To confirm that order was not a confounding factor in our experiment, we calculated a per-subject difference between task completion time using the cone and normal tree interfaces for each task. We then compared these values between the subjects that used the cone interface first, and those that used the normal tree interface first. None of these differences were significant at the .05 level (unpaired t-tests, p values 0.06, 0.73, 0.13, 0.36, 0.57, 0.95, 0.36 for Tasks 1 to 7).

The subjects' responses to the questionnaire also yielded significant results. The mean responses to the question "It was easy to see the data needed" were significantly different between interfaces for all tasks (see Table 2). Similarly, in all but Tasks 5 and 6 there were significant differences in the responses about the overall effectiveness of the interfaces. Figure 6 shows that the subjects rated both interfaces more poorly as the data density increased; however, the ratings decreased more rapidly for the cone interface. Note that subjects rated the overall effectiveness of the cone tree interface higher than their assessment of the ease of seeing the data.

	Task 1	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Task 6**	**Task 7**
			"It was easy to see the data needed"				
Cone: Mean(st.dev.)	3.92(0.9)	3.42(0.79)	2.42(1.0)	3.75(0.97)	3.08(0.9)	2.67(1.5)	3.17(1.47)
Tree: Mean(st.dev.)	4.58(0.51)	4.5(0.52)	4.08(0.9)	4.5(0.52)	3.92(0.67)	3.83(0.84)	4.33(0.49)
Paired T(11)	-2.35	-4.17	-4.43	-2.69	-3.08	-2.55	-2.76
p	0.039	0.002	0.001	0.021	0.01	0.03	0.019
			"Overall the interface was effective for the task"				
Cone: Mean(st.dev.)	4.17(0.72)	3.67(0.78)	2.83(0.94)	3.67(0.98)	3.33(0.78)	3.0(1.54)	3.17(1.34)
Tree: Mean(st.dev.)	4.75(0.45)	4.67(0.49)	4.17(0.83)	4.58(0.52)	3.83(0.72)	3.75(0.45)	4.25(0.45)
Paired T(11)	-2.55	-3.63	-4.0	-2.93	-2.17	-1.83	-2.6
p	0.027	0.004	0.002	0.014	0.053*	0.095*	0.025

Table 2: Responses to the Likert questions (1 = strongly disagree, 5 = strongly agree). * marks differences that are *not* significant at the 0.05 level.

5 Discussion

The quantitative results show that the subjects took longer to complete their tasks when using the cone interface, and that they rated the cone interface more poorly than the normal one for seeing and interacting with the data structure. There were, however, many positive comments about the cone interface, and half of the subjects performed at least one of the tasks more rapidly using cones.

The most common positive comment about the cone interface was that it made it easier to see the structure and population of the explored data-space (reported by five of the subjects). Four of the subjects felt that their lack of familiarity with the cone interface was a major factor in their task completion times, and they believed they would be able to become much more efficient with more experience. The most

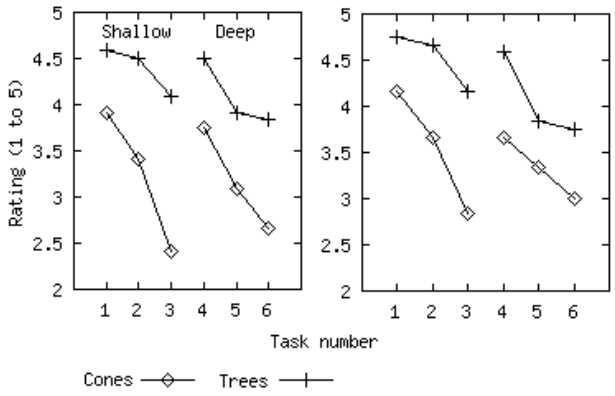


Figure 6: Likert responses by task for the two interfaces: “It was easy to see the data needed” (left); “Overall the interface was effective for the task” (right).

common negative comment concerned the difficulty of finding the desired text label amongst the clutter of overlapping labels in dense tasks.

The ability to rotate cones received both positive and negative comments. Three participants commented that rotation was a good alternative to scrolling, with one adding that he disliked scrolling. In contrast, one subject stated that he found rotation disorienting, stating that that it was “easy to lose landmarks”.

Despite the fact that our results show that cone trees were not as effective for the tasks in our study, the subjects’ comments indicate that several factors could have contributed towards an overly negative set of results. These factors are as follows:

Experimental tasks and conditions — The tasks in our study are characterised by ‘text-based search and locate’. These tasks were selected because of the predominantly text-based nature of most file-organisation schemes. However, it seems likely that the 3D organisation of cone trees is inherently poor at representing text labels. Our subjects’ comments indicated that cone trees may be more effective in providing users with a sense of structure and population. Therefore, there is reason to suspect that cone trees may perform relatively better in tasks such as “find the most densely populated directory” or “find the deepest directory”.

The constant screen real-estate used in both interface conditions is also likely to have detrimentally affected the subjects’ performance with the cone interface. The need for vertical scrolling could have been removed in the cone tree interface if we had allowed the subjects to maximise the window. However, if we had allowed this for the cone interface, we would also have had to allow it for the normal tree interface, with a consequent reduction in the amount of scrolling required in both conditions.

Familiarity — As mentioned above, with more training, it is likely that the subjects

would have improved their task completion times using the cone tree interface.

Interface fidelity — Previous work with cone trees has focused primarily on the fidelity of the 3D cone representation. Our interface is crude by comparison, but this compromise allowed us to achieve rapid animated interaction, even for large data-structures. Two of the subjects (who had prior experience with VRML representations of cone trees) commented on the relatively poor rendering of cones, but no subjects stated that their performance was detrimentally affected by this. Enhanced cone representation, such as shading and shadowing, might improve subject performance, and future evaluations should determine if this is the case.

6 Conclusions

Cone tree visualizations of hierarchical data structures were first described by Robertson et al. (1991). Since then, there have been numerous refinements on the concept, but there has been a notable absence of empirical studies to demonstrate their effectiveness. This paper provides an empirical foundation for future work on the usability of cone tree interfaces. Results show that our subjects were significantly slower at locating named files in a hierarchical data structure when using our cone tree interface than when using a 'normal' tree interface. Subject performance also deteriorated rapidly with cone trees as the density (branching factor) of the data structure increased.

Despite relatively poor task performance times, many subjects were enthusiastic about the cone tree interface. In particular, several subjects stated that cone trees provided a better feel for the structure of the data-space. In our further work we will investigate the effectiveness of using cone trees in different types of navigation tasks, and we will explore the effects of enhancing the interface's 3D visual cues.

Availability

The cone tree interface is written in Tcl/Tk and is available on request from the first author.

References

Carriere, J. & Kazman, R. (1995), Visualizing huge hierarchies: Beyond cone trees, *in* 'Proceedings of the 1995 IEEE Symposium on Information Visualization, October 1995, Atlanta, Georgia', pp. 74–81.

Hearst, M. & Karadi, C. (1997), Cat-a-cone: an interactive interface for specifying searches and viewing retrieval results using a large category hierarchy, *in* 'Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval. July 27–31, 1997, Philadelphia, PA USA', pp. 246–255.

Hemmje, M., Kunkel, C. & Willet, A. (1994), Lyberworld–a visualization user interface supporting fulltext retrieval, *in* 'Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval. Dublin, Ireland', pp. 249–259.

Jerding, D. & Stasko, J. (1998), 'The information mural: A technique for displaying and navigating large information spaces', *IEEE Transactions on Visualization and Computer Graphics* **4**(3), 257–271.

Mackinlay, J., Robertson, G. & Card, S. (1991), Perspective wall: Detail and context smoothly integrated, *in* 'Proceedings of CHI'91 Conference on Human Factors in Computing Systems New Orleans, May', pp. 173–179.

Robertson, G., Czerwinski, M., Larson, K., Robbins, D., Thiel, D. & van Dantzich, M. (1998), Data mountain: Using spatial memory for document management, *in* 'Proceedings of the 1998 ACM Conference on User Interface Software and Technology, November 1–4. San Francisco, California.', ACM Press, pp. 153–162.

Robertson, G., Mackinlay, J. & Card, S. (1991), Cone trees: animated 3d visualizations of hierarchical information, *in* 'Proceedings of CHI'91 Conference on Human Factors in Computing Systems New Orleans, May', pp. 189–194.

Smith, S., Barnard, D. & Macleod, I. (1984), 'Holophrasted displays in an interactive environment', *International Journal of Man-Machine Studies* **20**, 343–355.