LULEÅ UNIVERSITY OF TECHNOLOGY

2004:04

TECHNICAL REPORT

Extending Tree-Maps to Three Dimensions

A Comparative Study

Thomas Bladh, David A. Carr, Jeremiah Scholl

Technical Report

Department of Computer Science and Electrical Engineering
Division of Computer Science

2004:04 - ISSN: 1402-1536 - ISRN: LTU-TR--04/04--SE

Extending Tree-Maps to Three Dimensions: A Comparative Study

Thomas Bladh, David A. Carr, and Jeremiah Scholl

Department of Computer Science and Electrical Engineering
Luleå University of Technology, SE-971 87 Luleå, Sweden
{tbladh, david, jeremiah}@sm.luth.se
http: //www.sm.luth.se/csee

Abstract. This paper presents StepTree, an information visualization tool designed for depicting hierarchies, such as directory structures. StepTree is similar to the hierarchy-visualization tool, Treemap, in that it uses a rectangular, space-filling methodology, but differs from Treemap in that it employs three-dimensional space, which is used to more clearly convey the structural relationships of the hierarchy. The paper includes an empirical study comparing typical search and analysis tasks using StepTree and Treemap. The study shows that users perform significantly better on tasks related to interpreting structural relationships when using StepTree. In addition, users achieved the same performance with StepTree and Treemap when doing a range of other common interpretative and navigational tasks.

1 Introduction

The most common visualization method used for file system hierarchies is the node-and-indentation style used by the Microsoft Explorer and Nautilus (Linux/Gnome) browsers. Tools of this type are well known and recognized by the vast majority of desktop computer users. But, they have well-known disadvantages. In particular, they do not give an effective overview of large hierarchies because only those areas that are manually expanded are visible at any one time. Also, because nodes are expanded vertically, they require a great deal of scrolling to view the entire hierarchy.

An alternative approach for visualizing file systems is the space-filling approach. This approach is employed in a variety of visualization types including tree-maps [10] and SunBurst [11]. The space-filling approach is more efficient at utilizing screen space than node-and-indentation style visualizations, which leave a large amount of white space unused. The space-filling approach is characterized by subdividing a window into parts representing the branches (directories) and leaves (files) of the tree. The area of these parts is often related to some attribute such as size, which can be aggregated. This approach gives a better overview of the entire hierarchy, especially the attribute that is mapped to area.

This paper presents StepTree, a tool for displaying hierarchies that relies on the space-filling method and compares it to Treemap version 4.05, an implementation of tree-maps available from the Human-Computer Interaction Laboratory (HCIL) at the

University of Maryland. StepTree is similar to Treemap in that it constructs space-filling displays using a rectangular technique, but differs from Treemap in that it employs three dimensions by stacking each subdirectory on top of its parent directory. The use of three-dimensional space is intended to more clearly convey the structural relationships of the hierarchy to users and gives StepTree an appearance similar to boxes laid out on a warehouse floor, as opposed to the two-dimensional map of rectangles commonly associated with tree-maps.

The rest of this paper is organized as follows: In the next section we discuss related work. This is followed by a more detailed description of StepTree in Section 3. In Section 4 we describe an empirical study of 20 users performing tasks with StepTree and Treemap. Finally, we summarize and discuss possible future work in Section 5.

2 Related Work

Shneiderman [10] describes a theoretical foundation for space-filling visualization of hierarchies, including some initial algorithms. Tree-maps are basically nested Venn diagrams where the size of each node (in relation to the whole) is proportional to the size of the file or directory it represents. Tree-maps display hierarchies through enclosure, unlike node-link diagrams, which display hierarchies through connections. Using the two-dimensional, space-filling approach is a clever and simple way of displaying a hierarchy as it allows the contents of an entire structure (or a great deal of it) to be displayed at once. Johnson and Shneiderman [5] offered a more user-centered view of tree-maps that introduced them as an alternative method for viewing large file systems. Their work also introduced basic usability issues requiring additional research. These included the general categories of aesthetics, interactivity, comprehension, and efficient space utilization, which cover topics such as: layout, filtering, zooming (including traversing the hierarchy), coloring and labeling of files. Turo and Johnson [12] present an empirical study demonstrating the advantages of tree-maps. Their paper includes an experiment analyzing 12 users performing tasks with tree-maps in comparison to the Unix tcsh shell, and also an experiment with employees at General Electric Network for Information Exchange using tree-maps on a product hierarchy as compared to using traditional financial reports. Tree-maps outperformed the alternative in both cases. Since their introduction, tree-maps have been used to visualize a wide range of hierarchical structures such as stock portfolios [7], tennis matches [4], and photo collections [1].

After the initial research, two problems remained to be solved. First, the original "slice-and-dice" layout method often presented files of the same size in vastly different shapes having the same area. This made comparisons of size problematic. Second, the flat layout often made it difficult to truly perceive the hierarchy.

A number of improved layout algorithms have been developed to present equal areas in nearly identical shapes. Bruls et al. [3] presents the "squarification" algorithm which packs each directory's rectangle as nearly as possible with rectangles of the same aspect ratio. Squarification uses a greedy approach beginning with the largest children. Figures 1 and 2 show the same data set using the slice-and-dice and squari-

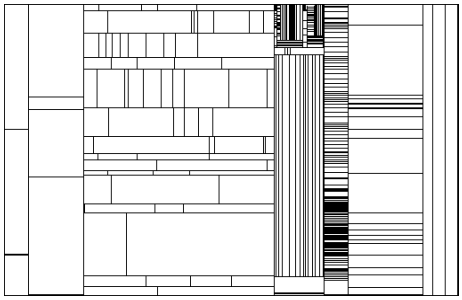

Fig. 1. Tree-map using slice-and-dice layout

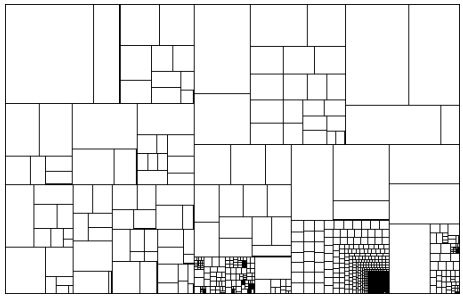

Fig. 2. Tree-map using squarified layout

fication methods. Bedersen, et. al. [1] present "ordered" tree-maps, which use a family of algorithms based on recursive division of the rectangle into four parts where one is a "pivot" element. Pivots are chosen based on various criteria. Bedersen's paper also summarizes and compares other layout algorithms including quantum tree-maps that are designed to lay out image thumbnails of a standard size.

In order to overcome problems perceiving the hierarchy, van Wijk & van de Wetering propose a shading technique called "cushioning" [14]. Cushioning presents tree-map rectangles as pillows and shades them to enhance edge visibility. This makes the hierarchy more apparent. The SunBurst visualization [11] constructs a radial, space-filling display (Figure 3). It offers users an advantage over tree-maps by more clearly displaying the structure of the hierarchy. SunBurst layers the levels of the hierarchy successively so that the innermost layer corresponds to the tree root and the outermost layer corresponds to the lowest level in the hierarchy. A comparative study showed that SunBurst outperformed tree-maps in tasks related to structural interpretation (e.g., locating the deepest directory). However, SunBurst does have disadvantages when interpreting size as radial slices are harder to compare than are rectangles.

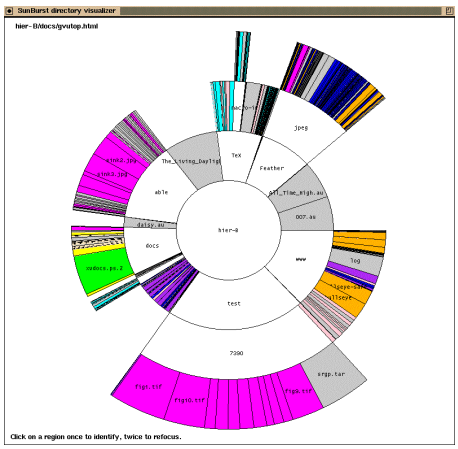

Fig. 3. SunBurst (courtesy of John Stasko)

Finally, utilizing the third dimension has been suggested as another method to help users perceive hierarchal relationships. Two early, 3-dimensional, tree-map-like implementations are FSV [8] and VisFS [9], but neither has been experimentally tested for usability. StepTree was designed specifically to act as a test bed for performing experimental evaluations on the benefits of 3D in tree-map-like graphs. Thus, StepTree follows the design of the Treemap application more closely than FSV and VisFS in order to reduce the number of variables that may alter experimental results.

3 The StepTree Application

StepTree is essentially a tree-map extended into three dimensions by the simple expedient of stacking levels of the tree on top of each other in 3D space. It utilizes the OpenGL API and was developed specifically for the display of file system hierarchies. It currently displays visual mappings of file system metrics such as file size; file and directory changes, and file type. StepTree is intended for use on traditional windows desktops and does not require any special hardware.

Figure 4 is a screen capture from StepTree. In addition to size, the display depicts change history and file type. In this particular case, files that have been changed within the last year are solid. Directories that have not recently changed are represented with wire frames ("ghosted"), and files not recently changed are simply not drawn. Note that directories (gray) containing recently changed files are also solid. File type is associated with color, a mapping which was set as close as possible to that of the Treemap application.

StepTree was originally developed to investigate ways of enriching space-filling visualization so that size is less dominant. Often when relationships are depicted by mapping size to area, this is at the expense of all other mappings. As the areas of nodes tend to be linked directly to this relationship, some nodes may dominate the view while others may be completely drowned out. If one wants to display change, changes to small files may be just as important as to large files. A solution to this problem is the optional use of gradual equalization of sibling nodes provided by StepTree's layout algorithm (Section 3.1).

StepTree uses a ghosting technique to display changed files. If a file has been modified within a specified range, then the node is drawn as a solid. If it has not, it is either ghosted or hidden. Ghosting unchanged nodes can be extremely effective, and hiding even more so when trying to spot changes to the file system. Changed files are effectively singled out. Changes are also propagated up in the hierarchy so that a directory is considered changed at the same date as it's most recently modified descendant. This is necessary as StepTree sometimes does not display nodes that are deep in the hierarchy in order to maintain good interactive response. Consequently, undisplayed files that have changed are represented by solid parent directories.

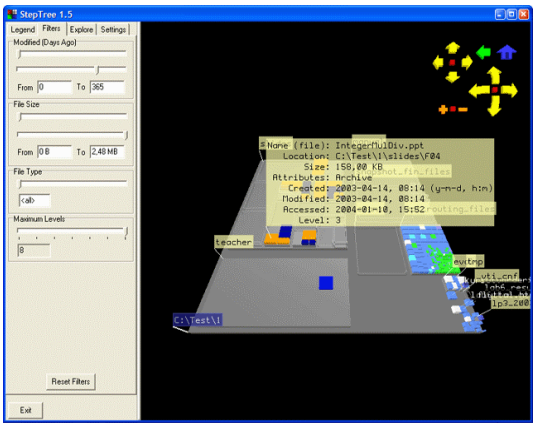

Fig. 4. StepTree

In adapting StepTree for the user study, we devised a new and more restrictive method of interaction with the 3D scene (Section 3.2), added a sidebar with a file type legend tab, a tab for dynamic-query filters, a tab for a traditional file system browser (coupled to the 3D tree-map), and a tab for settings. In addition labels were made translucent.

3.1 Layout and Labeling

The graph is laid out by a recursive function where the initial call specifies the root node of the file system subset and the coordinates and dimensions of the box for layout. This function then calls itself once for every child, placing child nodes as dictated by the squarification layout algorithm detailed in [3]. If equalization (redistribution of area), is enabled, it is applied before the node is actually laid out. The equalization step is followed by an "atrophication" step (size reduction of successive levels), in which the child nodes are shrunk to enhance level visibility.

Equalization is implemented in StepTree as a method of redistributing area from large nodes to smaller ones within a group of siblings. Equalization does not change the total area of the group. The equalization function, below, is applied to all members in a sibling group, adjusting their size depending on the equalization constant, ε ($0 \leq \varepsilon \leq 1$). The same equalization constant is used for the entire tree.

$$\nu_{eq} = (1 - \varepsilon)\nu + \varepsilon\alpha \qquad \text{(Equalization function)}$$

ν is the initial area of the child as a fraction of the area of the parent, α is the average child area for the sibling group, and ν_{eq} is the equalized area fraction. Setting the equalization constant to 1 will result in a completely equalized sibling group where all nodes are given the same fraction of the parent's area. Setting the constant to 0 will result in no change in area distribution.

Small files and empty directories would not be visible without equalization or a similar redistribution function. Equalization, however, distorts the visualization. Two files of equal size on disk might appear to have different sizes if they have different parent directories. Note that in our implementation, the equalization step is followed by an atrophication step where the area used by children is shrunk by a set fraction in relation to the parent in order to expose underlying structure. Both steps can be disabled. Equalization is but one of the many types of distortions that could be applied to a space filling visualization. Previous uses of distortion include for example the application of exponential weight functions to exaggerate size differences [13].

The final layout issue is to ensure adequate performance when rotating and moving in real time. While current systems can handle about 5,000 nodes, file systems are often considerably larger. Therefore, we were forced to introduce node pruning. However, we did not want to display partial levels. So, the depth of the displayed portion of the hierarchy is limited by processing time and an upper limit on the number of visible nodes. If the node limit or the time limit is reached, the currently incomplete level of the visualization is not displayed.

Labels in StepTree are implemented as text flags that always face the observer and always have the same size and orientation. This helps to ensure a minimum level of legibility regardless of how the visualization has been rotated. Labels affixed directly to the surface of the nodes are often arbitrarily truncated and distorted by perspective projection. In order to avoid a forest of labels where few individual labels are legible, permanent flags are only assigned to the root and its immediate children. All flags are translucent to enable the user to see parts of the visualization that would otherwise be obscured. Label translucency also seems to make more labels legible when they overlap.

3.2 Navigation and Interaction

A common problem with 3D visualization is getting an unusable view or being lost in the "desert fog" [6]. From this view it is difficult if not impossible to draw conclusions as to where to navigate next. This was a problem in StepTree's early versions where view navigation allowed unconstrained flight. The user would frequently navigate into an unfavorable position, be pointed in the wrong direction, and see nothing but a blank screen. To correct this, we elected to use object-centric manipulation of the 3D scene, treating the graph as an object to be manipulated and inspected. Furthermore, we limited the user's freedom to position the graph. It can be rotated around two axes, x and z, but limited to a maximum rotation of 160 degrees. Rotation is also constrained so that some part of the graph is always at the center of the display. The viewpoint's position on the (x, z) plane can be panned. Panning is also constrained so that some part of the graph is always centered. Zooming is accomplished by moving closer to or farther away from the graph along the y-axis and is limited by a set of distance bounds. This combination of constraints on rotating, panning, and zooming solved the desert-fog problem for all but a handful of degenerate cases.

4 User Study

The primary motivation for our study was the relative lack of experiments comparing two- and three-dimensional visualization tools. In order to determine directions for further research on three-dimensional extensions to the tree-map concept, it is important to find out exactly what the difference in user performance between two-dimensional and three-dimensional tree-maps is.

4.1 Experiment Procedure

Twenty students in a Human-Computer Interaction class at Luleå University of Technology volunteered to participate in the experiment. Of these twenty students, one participant was later excluded because he was color-blind. This student was replaced by a member of the computer support group at the university who had comparable age, education, and computer experience. The participants were between 21 and 35 years old with an average age of 23.3. Most were in their third or fourth year at the university. They had used computers for an average of 10.5 years and currently used computers on an average of 30 hours per week. All but one of the participants had experience with 3D games averaging slightly less than one hour per week. All participants were right-handed; three were female and 17 were male.

The tests were conducted on a 1.7 GHz Pentium 4 workstation with 256 Megabytes of RAM and running the Windows 2000 operating system. Both Treemap and StepTree were run at a resolution of 1024 by 768 pixels on a 21-inch CRT. For the test, the equalization distortion was disabled as it is not available in Treemap, and the atrophication distortion was set to a level of 10%. Participants used only the mouse.

The test leader conducted a tutorial session for each tool just before each participant performed the related tasks. Each tutorial session took approximately ten minutes to complete and was followed by a five minute, free-form exploration period during which each participant could try the tool and ask any questions. The actual test began after the five minutes had passed, or earlier if the participant indicated readiness. Before the test the timing procedure was explained to the participant.

Each task was first read out loud followed by the phrase "and you may start now" to indicate the start of timing. At this time the task in question was provided on paper, which was especially important when the task description contained complicated path information. Answers to questions could be given by pointing with the mouse and using a phrase such as "this one", or the answer could be given verbally by naming a file or directory. In addition, a challenge-response procedure was used when an answer was indicated. All verbal interaction and written material was in Swedish, the native language of all test participants and the test leader.

Each participant performed a set of nine tasks with both Treemap and StepTree. Two distinct but structurally similar data sets of around a thousand nodes were used, one with each tool in the test. The order of the visualizations and the mapping between data set and visualization tool were counterbalanced. The tasks were:

1. Locate the largest file.
2. Locate the largest file of a certain type.
3. Locate the directory furthest down in the hierarchy structure.
4. Locate a file with a certain path.
5. Determine which of two given directories contains the most files including subdirectories?
6. Determine which of two given directories is the largest?
7. Name the most common file type?
8. Determine in which directory the file I'm pointing at is located?
9. a) Locate the largest file in a certain directory
   b) Locate the largest file of the same type in the whole hierarchy.

The tasks were chosen as a representative sampling of the types of perceptual and navigational problems a user might run up against when browsing a file system. Tasks were also classified and distributed evenly along the two broad categories of topological tasks and content-related tasks. Tasks 1, 2, and 6 are clearly content-related tasks while tasks 3, 4, and 8 are clearly topological – task 3 strongly so. The remaining tasks 5, 7, and 9 belong in both categories. Exact classification of tasks can be fuzzy. As the number of files grows, and they become more densely packed, one tends to perceive the pattern rather than the exact structural placement of each entity. Topology becomes content. Therefore for tasks 5, 7, and 9, we can argue for both interpretations.

4.2 Hypotheses

Our first hypothesis for the experiment was that Treemap, the two-dimensional tool, would be faster overall and result in fewer errors. This is mainly based on the assumption that the added complication of three-dimensional navigation would have

significant adverse effects. In slight contradiction to this overall hypothesis, we hypothesized that the task with a pronounced topological component (Task 3) would benefit from the three-dimensional view, resulting in shorter task times and fewer errors when StepTree is used on this specific task.

4.3 Results and Discussion

Contrary to what we expected, we found no statistical significance in favor of Treemap for all tasks combined. An ANOVA on the effect of tool, tool order, data set, and data set order for total task times, showed $p > 0.05$ by a significant margin in all cases. We did, however, find a statistical significance for the effect of tool type on time ($p = 0.0091$), when we performed an ANOVA looking at the effect of the same factors for just Task 3. Users were in this case significantly faster when they used StepTree. The same test also found a significant effect for data sets ($p = 0.0153$), on task times for Task 3. This effect can be explained by the fact that the deepest directory in data set 2 is less isolated, and thus harder to pick out, than the one in data set 1. Error rates on task 3 also differed significantly ($\chi^2 = 14.54$, df = 1, $p < 0.001$), with the fewest errors being made when StepTree was used. Seventeen participants got this question wrong with Treemap, while only five participants were unable to complete this task correctly with StepTree.

Except for performance with Task 3, the performance on the tools was very similar (Figures 5 and 6.) It would seem that mapping depth in the hierarchy to height in the visualization is an effective method for visualizing the topological component of file systems. Users were both faster and less error prone using StepTree when looking for the deepest subdirectory.

We also noticed a much higher error rate for Task 7 on data set 1 than on data set 2. In data set 1 the most common file type (gif) consists primarily of small files. As the participants were inexperienced with space-filling visualizations, many picked the predominate color and answered the question, "Which file type uses the most space?" This illustrates that

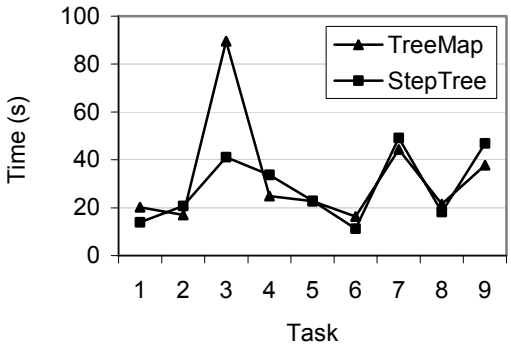

Fig. 5. Average completion time (seconds) by task

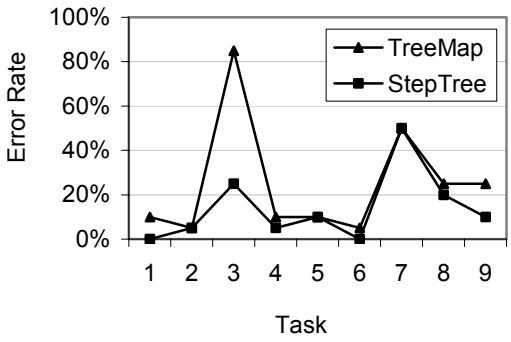

Fig. 6. Error rate by task

both visualizations can be misleading and that a greater understanding of the visualization is required to correctly answer some questions.

5 Conclusions and Future Work

The equivalence of StepTree and Treemap on most tasks was unexpected, since 3D interfaces often result in longer task times. However, we may find an explanation for these results in that the 3D interface used in StepTree was designed to be more restricting and forgiving than traditional 3D interfaces. The limits imposed on zoom, pan, and rotation seem to have been effective in preventing users from getting lost. In addition, the fact that 19 out of the 20 users had previous experience playing 3D games may have helped equalize performance. The gap in usability between 2D and 3D interfaces may close as the average computer user becomes more experienced with 3D. While a clear conclusion of this trend cannot be made from our experiment, it is an interesting topic for future study.

The explicit display of hierarchical depth by StepTree resulted in a clear advantage over Treemap on the question regarding depth in the hierarchy. This illustrates an area where 3D may have an advantage over 2D. However, SunBurst also explicitly displays depth by mapping it to radius. It would be worthwhile to compare SunBurst and StepTree.

The study group offered several interesting comments about StepTree that may be useful in improving future versions of the tool. One frequent complaint participants made during the tests was the lack of rotation around the y-axis (vertical axis). Their preconception seemed to be that dragging sideways should rotate the object around the y-axis much like a potter's wheel. This was indicated by the participant's actions – an ineffective, sideways dragging motion – just prior to voicing the complaint. Manipulation of this sort should be added in future versions of the StepTree software.

Another annoyance perceived by the participants was the lack of tight coupling. If a filter had been applied so that the visualization only showed ".gif" files, then many participants assumed that the reported number of nodes in the visualized directory had been updated as well. This is not the case in either application and should be included in both StepTree and Treemap.

After completing both tests, one participant complained about the tool-tip flag in Treemap. This flag was in his words, "always obscuring something". The same person remarked that in StepTree the corresponding flag did not appear immediately and was translucent, which reduced occlusion. On the other hand, a source of complaints was that StepTree's tool-tip often spilled over the edge of the screen and was unreadable. Future versions should take into account the physical dimensions of the view port and not arbitrarily place labels.

6 Acknowledgement

We would like to thank John Stasko for providing the SunBurst figure. Thanks also go to Carl Rollo for proofreading this paper. Finally, special thanks go to the 21 anonymous participants who helped us in our study.

References

1. Bederson, B. B., Shneiderman, B., Wattenberg, M., Ordered and quantum treemaps: making effective use of 2D space to display hierarchies, *ACM Transactions on Graphics*, 21(4), Oct. 2002, 833-854.
2. Bederson, B. B., PhotoMesa: A zoomable image browser using quantum treemaps and bubblemaps, *Proceedings of the 2001 ACM Symposium on User Interface Software and Technology, CHI Letters 3(2),* Orlando, FL, 11-14 Nov. 2001, 71-80.
3. Bruls, M. Huizing, K. van Wijk, J. J., Squarified treemaps, *Proceeding of Joint Eurographics and IEEE TCVG Symposium on Visualization*, Amsterdam, the Netherlands, 29-30 May 2000, 33-42.
4. Jin, L., Banks, D. C., TennisViewer: a browser for competition trees, *IEEE Computer Graphics and Applications,* 17(4), July/Aug. 1997, 63-65.
5. Johnson, B., Shneiderman, B., Tree-maps: A space-filling approach to the visualization of hierarchical information structures. *Proceedings of IEEE Visualization '91*, San Diego, CA, Oct. 1991, 284-291.
6. Jul, S., Furnas, G. W., Critical zones in desert fog: aids to multiscale navigation, *Proceedings of the 1998 ACM Symposium on User Interface Software and Technology,* San Francisco, CA, Nov. 1998, 97-106.
7. Jungmeister, W. A., Turo, D., Adapting treemaps to stock portfolio visualization, 1992, University of Maryland Center for Automation Research, technical report CAR-TR-648.
8. Richard, G. D., File System Visualizer (FSV), http://fsv.sourceforge.net, Jan. 2004.
9. Schmidt, H., Visual File System (VisFS), http://www.heiko-schmidt.info/project/visfs/visfs_de.html, Jan. 2004.
10. Shneiderman, B., Tree visualization with tree-maps: a 2-D space-filling approach, *ACM Transactions on Graphics,* 11(1), Jan. 1992, 92-99.
11. Stasko, J. Catrambone, R. Guzdial, M., McDonald, K., An evaluation of space-filling information visualizations for depicting hierarchical structures, *International Journal of Human-Computer Studies,* 53(5), Nov. 2000, 663-694.
12. Turo, D., Johnson, B, Improving the visualization of hierarchies with treemaps: design issues and experimentation, *Proceedings of IEEE Visualization '92*, Boston, MA, 19-23 Oct. 1992, 124-131.
13. Turo, D., Hierarchical visualization with Treemaps: making sense of pro basketball data, *ACM CHI '94 Conference Companion*, Boston, MA, 24-28 April 1994, 441-442.
14. van Wijk, J., van de Wetering, H., Cushion treemaps: visualization of hierarchical information, *Proceedings of INFOVIS'99*, San Francisco, CA, 25-26 Oct. 1999, 73-78.