SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation

Catherine Plaisant, Jesse Grosjean, Benjamin B. Bederson
Human-Computer Interaction Laboratory
University of Maryland
College Park MD 20782
{plaisant, grosjean, bederson}@cs.umd.edu
(301) 405-2768
www.cs.umd.edu/hcil/spacetree

Abstract
We present a novel tree browser that builds on the conventional node link tree diagrams. It adds dynamic rescaling of branches of the tree to best fit the available screen space, optimized camera movement, and the use of preview icons summarizing the topology of the branches that cannot be expanded. In addition, it includes integrated search and filter functions. This paper reflects on the evolution of the design and highlights the principles that emerged from it. A controlled experiment showed benefits for navigation to already previously visited nodes and estimation of overall tree topology.

Introduction
The browsing of hierarchies and trees has been investigated extensively [Card et al., 1998]. Designers have demonstrated that many alternatives to the traditional node link representation (Figure 1) are possible, but this classic representation of trees remains the most familiar mapping for users and still is universally used to draw simple trees. Our goal was to take another look at this well-known tree representation and see how visualization advances in zoomable user interfaces and improved animation principles could lead to a better interactive tree browser while preserving the classic tree representation. Such a browser might encourage the adoption of visualization by a wider range of users (e.g. families browsing genealogy trees or biology students browsing taxonomies) or by more traditional work environments (organization charts for managers or personal office staff).

We present SpaceTree, a novel interface that combines the conventional layout of trees with a zooming environment that dynamically lays out branches of the tree to best fit the available screen space. It also uses preview icons to summarize the topology of the branches when there isn't enough space to show them in full. This paper reflects on the evolution of the design and highlights the principles that emerged from it. A controlled experiment compares SpaceTree to two other interfaces and analyzes the impact of interface features on the time to perform navigation tasks to new and already visited nodes, and topology evaluation tasks.

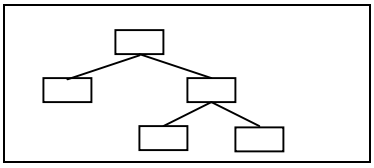

Figure 1: The "traditional" node link representation of a tree. It has a favored direction (here top down). Drawing every nodes makes very poor use of the available drawing space, and would fill up a screen before reaching 100 nodes.

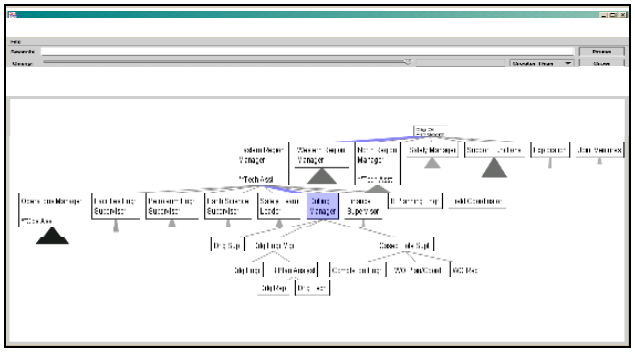

Figure 2: SpaceTree allows large trees to be explored dynamically. Branches that do not fit on the screen are summarized by a triangular preview. When users select a node to change the focus of the layout, the number of levels opened is maximized. In this organization chart example, the 3 lower levels of the hierarchy were opened at once as users clicked on "Drilling Manager" (the colored node in the middle.)

Related work
Two large categories of solutions have been proposed to display and manipulate trees: space-filling techniques and node link techniques. Space filling techniques (treemaps [Bederson et al. 02], information slices [Andrews, 1998]) have been successful at visualizing trees that have attributes values at the node level. In particular, treemaps are seeing a rapid expansion of their use for monitoring, from stock market applications (e. g. www.smartmoney .com), to inventory or network management, to production monitoring. Space filling techniques shine when users care mostly about leaf nodes and their attributes (e.g. outlier stocks) but do not need to focus on the topology of the tree, or the topology of the tree is

trivial (e.g. 2 or 3 fixed levels). Treemap users also require training because of the unfamiliar layout.

Node link diagrams, on the other hand, have long been the plague of information visualization designers because they typically make inefficient use of screen space, leaving the root side of the tree completely empty – usually the top or left of the screen – and overcrowding the opposite side. Even trees of a hundred nodes often need multiple screens to be completely displayed, or require scrolling since only part of the diagram is visible at a given time. Specialized tools can help users manage the multiple pages needed to display those trees (e.g. www.nakisa.com for organizational chart).

Optimized layout techniques can produce more compact displays by slightly shifting branches or nodes (e.g. Graphviz [North, online]), but those techniques only partially alleviate the problem and are often not appropriate for interactive applications.

The coupling of overview + detail views with pan and zoom was proposed early by Beard & Walker [Beard, 1990] and found to be more effective then scrolling. Kumar et al. successfully combined the overview and detail technique with dynamic queries to facilitate the searching and pruning of large trees [Kumar et al., 1995]. The technique allows ranges of depth dependant attribute values to be specified to prune the tree dynamically.

Another approach is to use 3D node link diagrams. Cone Trees [Robertson et al., 1991] allow users to rotate a 3D representation of the tree to reveal its hidden parts. Info-TV [Chignell et al., 1993] allows nodes and labels to be removed from sub trees (leaving the links) to show a more compact view of branches. 3D representations are attractive but only marginally improve the screen space problem while increasing the complexity of the interaction.

A clever way to make better use of screen space is to break loose from the traditional up-down or left-right orientation and use circular layouts [Bertin, 83]. The best known technique is the Hyperbolic tree browser [Lamping et al., 1995] - now available as StarTree from Inxight (www.inxight.com) - which uses hyperbolic geometry to place nodes around the root and provides smooth and continuous animation of the tree as users click or drag nodes to readjust the focus point of the layout. The animation is striking but the constant redrawing of the tree can be distracting. Labels are hard to browse because they are not aligned and sometimes overlap. In addition, the unconventional layout may not match the expectations of users (e.g. it is not appropriate to present the organizational chart of a conventional business.)

Cheops [Beaudouin et al., 1996] overlaps branches of the tree to provide a very compact overview of large trees. Labeling is an issue and interpreting the diagram requires training.

Constrained by limited screen space, WebBrain (www.webbrain.com) chooses to prune the tree to show only a very local view of children and parent of the current selection – and some crosslinks. The nodes have to be reoriented at each selection.

The benefits of pure zooming are illustrated by PadPrints [Hightower et al., 1998], which automatically scales down a tree of visited pages as users navigate the web. The use of fisheye effects to display branches at varying scales in the same display was also explored [Noik, 93] [Hopkins, 89].

Expand and contract interfaces as exemplified by Microsoft Explorer allow the browsing of trees as well. Similarly, WebTOC [Nation et al., 1997] shows how information about size or type could be added to the expandable list of nodes.

Description of the interface

SpaceTree is our attempt to make the best possible use of the traditional node link tree representation for interactive visualization. Figures 3 to 6 show a series of screen captures of the main display area, showing the progressive opening of branches as users refine their focus of interest. Branches that cannot be fully opened because of lack of space are previewed with an icon. Here we describe an initial design using a preview icon in the shape of an isosceles rectangle. The shading of the triangle is proportional to the total number of nodes in the subtree. The height of the triangle represents the depth of the subtree and the base is proportional to the average width (i.e. number of items divided by the depth). The preview icons can be chosen to be relative to the root (for ease of comparison between levels) or to the parent (for ease of local comparison).

Users can navigate the tree by clicking on nodes to open branches, or by using the arrow keys to navigate among siblings, ancestors and descendants. Figure 6 illustrates how SpaceTree maximizes the number of lower levels to be opened.

Several layout options allow adjustments of the spacing between nodes, alignment, icon options etc. The choice of overall orientation of the tree layout, allows designers or users to match the layout to the natural orientation of the data. For example organizational charts are often oriented top down (suggesting power), while the evolution of species is more likely to be show left to right (suggesting time) or bottom up (suggesting progress). Figures 7 and 8 show examples of a left to right orientation. The choice of the most space efficient orientation depends on the tree topology and the aspect ratios of the labels and the window.

Search and filter

SpaceTree also includes integrated support for search and filter. As users type a string, the location of results is highlighted on the tree. Then users can navigate the tree,

or click on the "prune" button to see a filtered view of the tree showing only the paths to the matching nodes.

We also implemented dynamic queries [Shneiderman, 1994] to illustrate how dynamic queries allow the rapid pruning of the tree when attributes are available at the node level. As users manipulate a slider to limit the value of an attribute, leaves or branches of the tree are dynamically grayed out to show the effect of the query. (Note that the current version supports rudimentary dynamic queries with only one attribute, but the principle applies to any number of attributes such as income of employees, year in the company, or language spoken, etc. for our organizational chart example).

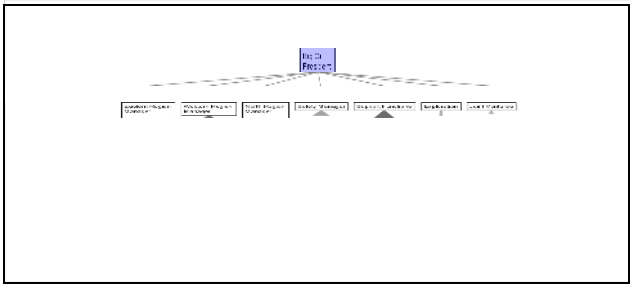

Figure 3: Top level overview. The triangular preview icons summarize the branches that cannot be opened. When room is available, two or more levels might be opened at once. Darker icons correspond to branches with more nodes. Taller icons (in this top-down layout) correspond to deeper branches, and wider icons correspond to a higher average branching factor.

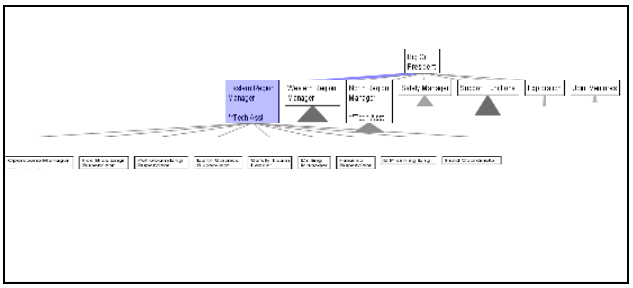

Figure 4: As users change the focus of the layout (i.e. click on a node – shown darker), more detail is revealed.

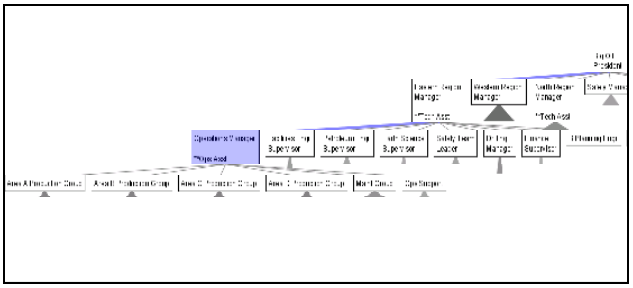

Figure 5: The tree is animated to its new layout in tree separate steps: trim, translate and expand (trim and translate is only done when needed).

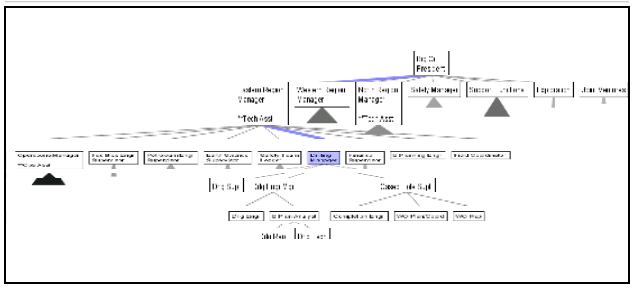

Figure 6: Upon each refocusing, the maximum number of levels that fit is opened (here 3 levels could fit so they were opened at once when user selected "drilling manager").

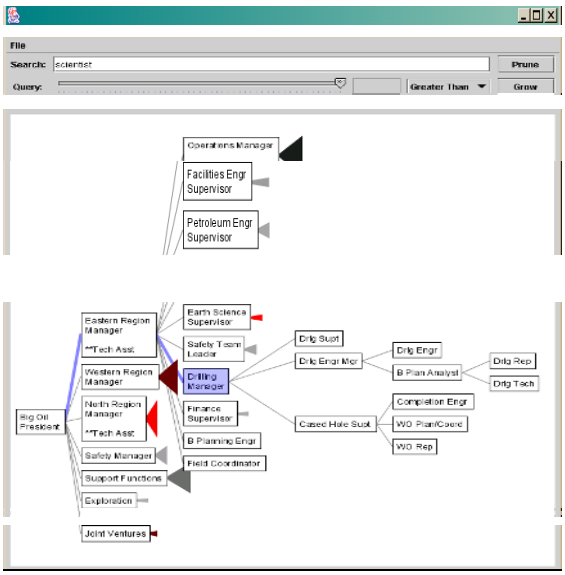

Figure 7: The tree shown in Figure 6 has been rotated to a different orientation, then a search for "scientist" was performed and the location of search results is shown in red.
(not visible in a black and white prints

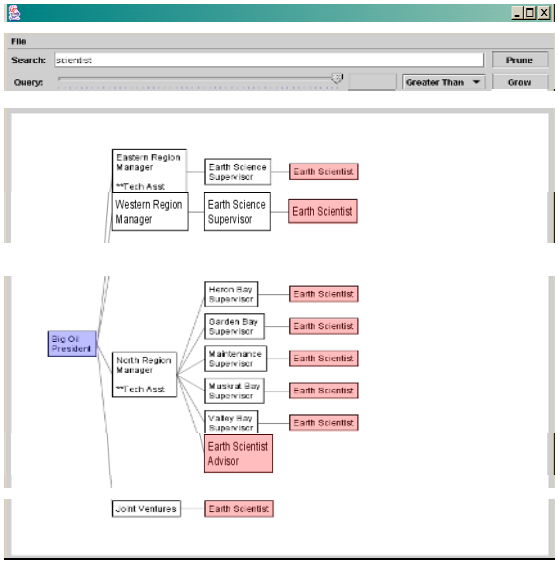

Figure 8: A click on the "Prune" button displays a filtered view of the tree, revealing only the branches that lead to scientists, opened as space permits.

SpaceTree was developed in Java using TinyJazz, a new toolkit that is an optimized subset of Jazz [Bederson et. al., 2000], and the tree layout is inspired from [Walker 1990] and [Furnas, 81]

Reviewing early versions and emerging design guidelines

The SpaceTree was designed with continuous feedback from our sponsors who had a particular need for hierarchy browsing at the time of the project. This included monthly discussions and exchange of prototypes. Through progressive refinement (about 10 versions were discussed) we learned lessons that we summarize here as guidelines for designers.

Semantic zooming is preferred over geometric scaling (i.e. "Make it readable or don't bother showing the nodes".)

Our first designs attempted to use fixed progressive scaling down of the nodes – providing a nice overview of the tree (Figure 9) and continuous geometrical zooming to allow users to progressively reveal details of lower levels of the tree. The result was a smooth fly through of the tree (Figure 9 and 10) but was rejected bluntly by our users who rightly noted that only one level of the tree was even readable at a time (lower levels were "visible" but never readable). Readability and a good use of the screen space had not been optimized enough. The conclusion was that instead of continuous scaling, a step approach was needed: nodes should be either *readable* or not, and once they are not readable they could be seen as *individuals* or aggregated in an *abstract* representation. This was made possible by the semantic zooming afforded by Jazz. All scaling is therefore calculated on the fly. Figure 11 shows an example of alternative previews of a tree branches.

Maximize the number of levels opened at any time

Feedback from users made it clear that they resented having to open the tree "one level at a time" when there was room to open more levels at once. This is illustrated in Figure 6.

Decompose the tree animation

We experimented with several animations of the layout to reflect the change of focus and found that we received our most positive feedback with a decomposed animation following 3 main steps: trim, translate, and grow. When users select a new focus, SpaceTree evaluates how many levels of the new branch can be opened to fit in the window, then 1) trims the tree of the branches that would overlap the new branch to be opened; 2) centers the trimmed tree so that the new branch will fit on the window, 3) grows the branch out of the new focus point.

Maintain landmarks

As the tree is trimmed, expanded or translated it is crucial to maintain landmarks to help users remain oriented [Jul & Furnas, 1998]. The obvious candidates for landmarks are the *focus points* users selected, i.e. the current focus and the path up the tree, which usually matches the history of focus points as users traverse the tree. The ancestor path of the current focus is highlighted in blue. The node under the cursor is gold, and its ancestor path is shown in gold up until it meets the blue one. When users click on a node, their eyes are already on the gold node, which remains gold as the tree is animated to a new layout, and then turns blue to reflect the new focus.

The constant relative position of siblings and the overall shape of upper tree help maintain the larger context up the tree (Webbrain.com illustrates how changing the reorientation of siblings can be disorienting).

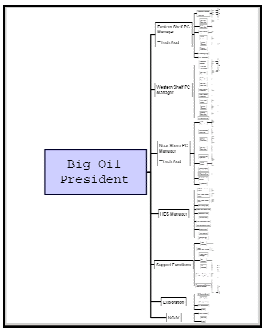

Figure 9: Early prototype: overview of the continuously scaled tree.

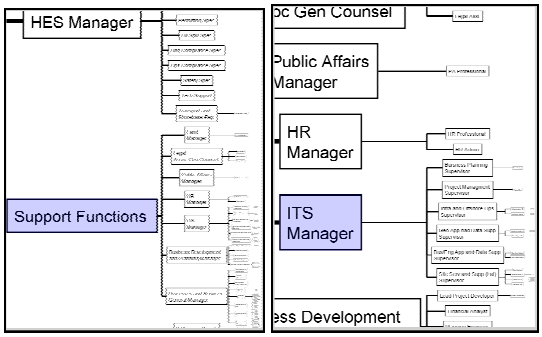

Figure 10: Early prototype: geometric zoom allowed users to fly through the tree but only made one new level readable at a time, and poorly used the screen space.

(a) (b) (c)

Figure 11: Current solution: semantic zooming on multiple representations of the tree. Previews can consist of a miniature of the branch (a) when the number of nodes is small or an abstract representation of the branch like the triangles of Figure 3 . (b) and (c) are alternatives to the triangle and provide more details on distribution of nodes in the next level branches.

Take advantage of overviews and dynamic filtering

Search and dynamic query techniques are not new, but SpaceTree offers a good demonstration of their application. One option we debated is whether to dynamically trim the tree of the nodes that would "fall off" with the query, or just gray them out and give "on demand pruning" after the query. We chose the later option that avoids constant and wild animation of the tree.

Use "data-aware" zooming controls

Another of the lessons we learned was the need to provide data-aware controls. Our initial browser permitted free zooming by clicking anywhere in the data space (on node or outside of nodes). This was the default control of Jazz but was only usable by expert zooming users, others being rapidly lost in the fog of empty information space. A second version gave users a preview of the area of the screen that would come to full view once they clicked (Figure 12). This helped users to avoid empty areas, but users complained that the area rarely matched the topology of the tree. Therefore, the best results were attained by only allowing users to zoom by clicking on nodes.

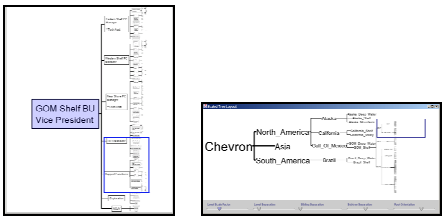

Figure 12: Early prototype: a rectangular cursor matching the window aspect ratio gave a preview of the area to be enlarged if users clicked, but didn't necessarily match a branch of the tree.

This matches our understanding of why the simple link following web interface is so successful: people can readily click on a link to see related information, while more complex interactions are difficult for users and typically require learning.

Controlled experiment

We conducted an experiment comparing 3 tree-browsing interfaces: Microsoft Explorer (Figure 13), a Hyperbolic tree browser[1] (Figure 14), and SpaceTree (Figure 15). Our goal was not to pit the interfaces against each other (as they are clearly at different stages of refinement and of different familiarity to users) but to understand what feature seemed to help users perform certain tasks. We used a 3x7 (3 interfaces by 7 tasks) repeated measure within subject design. To control learning effects, the order [illegible] sets were [illegible]

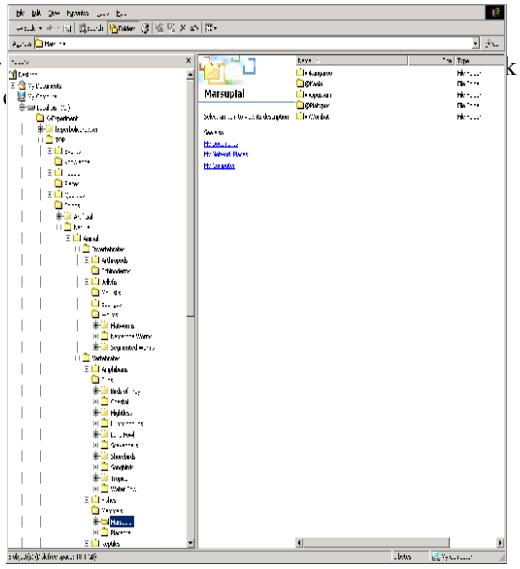

Figure 13: Microsoft Explorer, a classic expand and contract int[illegible]ces (1024[illegible]ntrol

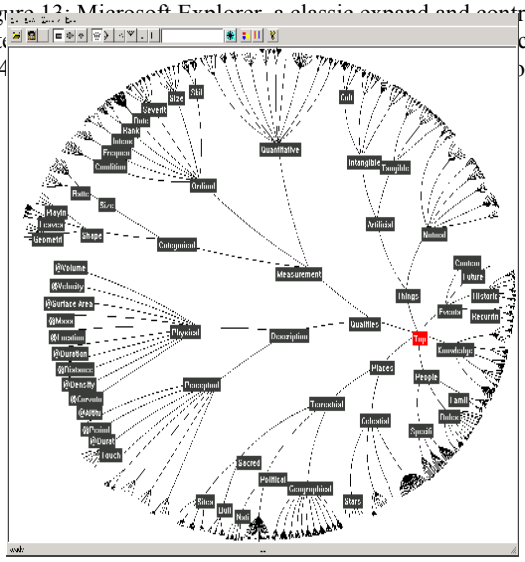

Figure 14: The hyperbolic viewer spreads the branches around the root making 2 or 3 levels of the tree visible. Users can click or drag a node to dynamically and continuously update the layout of the tree and quickly explore deeper levels of the tree.

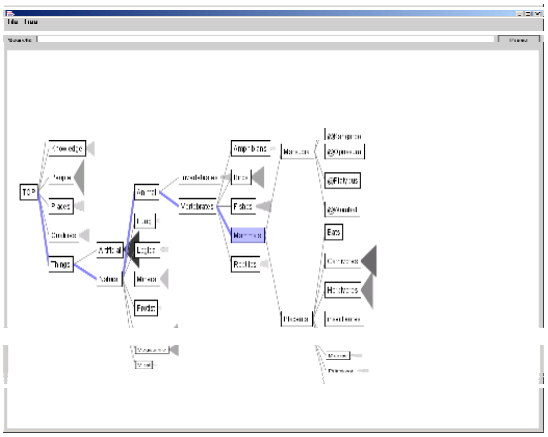

Figure 15: The SpaceTree opened to "mammals" and showing nodes seven levels down the tree.

[1] We attempted to use the downloadable version from inxight.com but could not transform the test data into the required format. Instead we used an older prototype, and asked three colleagues to compare the 2 versions. The old version was found similar to the current version in term of the features used in the experiment (e.g. we didn't use color, attribute values, graphics or database access in the test tree). Obviously the current commercial version has many more features that make it a useful product but that we were not comparing here.

Eighteen subjects participated, and each session lasted a maximum of 40 minutes. Subjects each received $10 for their participation. To provide the motivation to perform the tasks quickly and accurately, an additional $5 was given to the fastest user within each interface (with no errors). We chose to use computer science students that could be assumed to have a homogeneous level of comfort with computers and tree structures. Subjects were given a maximum of 2 minutes of training with each interface. In order to see what problems users would encounter without any training, the experimenter gave no initial demonstration, but after about 30 seconds of self-exploration, the experimenter made sure that users had discovered everything properly. Hyperbolic users were told that they could continuously drag nodes, and the meaning of the triangle icons was explained to SpaceTree users (misunderstanding were first recorded, and then clarified).

We used a tree of more than 7,000 nodes from the CHI'97 BrowseOff [Mullet, 97]. The three task sets used different branches of the tree and were carefully chosen to be equivalent in terms of number of levels traversed and semantic complexity of the data explored. Three types of tasks were used. Node searches (e.g. find kangaroo, find planaria), search of previously visited nodes (return to kangaroo) and typology questions (e.g. read the path up the tree, find this branch 3 nodes with more than 10 direct descendants, and which of the three branches of this node contain more nodes). To avoid measuring users' knowledge about the nodes they were asked to find (e.g. kangaroos) we provided hints to users (e.g. kangaroos are mammals and marsupials) without giving them the entire path to follow (e.g. we didn't give out the well known steps such as animals). Those hints were also kept similar in the three sets of tasks. The terminology of the questions was explained in the initial training.

The size of the window was the same for each interface (1024x768 pixels for the usable display area). The focus of the tree layout was initialized at the top of the tree at the beginning of tasks but was not reset between tasks to match a normal work session. The entire explorer hierarchy was re-contracted in between users. After the short training, users were asked to conduct 7 tasks with each interface, after which they filled a questionnaire and gave open-ended feedback about the 3 interfaces. The dependant variables were the time to complete each task, the presence of errors (only relevant for 2 questions), and subjective ratings on a 9-point Likert-type scale.

Results

For each speed and preference dependant variable we performed a one-way ANOVA followed by a post hoc Bonferroni analysis. The confidence interval is set at 95% for all ANOVA and post-hoc analysis.

For conciseness our hypotheses are described for each type of task, followed by a brief summary of the results. We report mean times in seconds in the following order: (E) for Explorer, (H) for Hyperbolic and (S) for SpaceTree.

A) First-time node finding

For finding nodes that had never been seen before, we hypothesized that SpaceTree and Hyperbolic would be similar in term of speed and faster than Explorer because they both provide access to more than one level at a time, which enables users to select categories further down the tree. Explorer uses smaller fonts and the size of the targets is smaller than the 2 other interfaces, but the distances to travel are also smaller and users are extremely familiar with the interface. An advantage might be seen for the SpaceTree because of the alignment of the labels, allowing faster scanning of the items, but this advantage may not compensate for the advantage of the fast continuous update of the tree layout in Hyperbolic, which allows rapid exploration of neighborhoods.

Results: Only two of the 3 node finding tasks showed significant differences, Explorer being faster than Hyperbolic in the 1st task where learning may have been a factor (in seconds: E=10.5, H=13.2, S=11.1), and SpaceTree being faster than explorer in the third task (E=11.3, H=5.6, S=4.7). Observations confirmed that most users took advantage of the ability of Hyperbolic and SpaceTree to show multiple levels of the tree by clicking down often more than one level at a time. The faster users did continuously drag nodes to reveal details with Hyperbolic, while with SpaceTree they still had to select and animate the tree in steps when going deep in the tree. Explorer users showed their experience by avoiding using the small ⊞ icon and clicked on the labels to expand the hierarchy in the folder view.

B) Returning to previously visited nodes

We had predicted that the SpaceTree would be faster than the hyperbolic tree because the layout remains more consistent, allowing users to remember where the nodes they had already clicked on were going to appear, while in the hyperbolic browser, a node could appear anywhere, depending on the location of the focus point. Figure 16 shows 2 examples of different locations for kangaroo. We predicted that Explorer would be faster than both TreeBrowser and Hyperbolic when the start and end point were next to each other because Explorer allows multiple branches to remain open therefore making it very easy to go back and forth between 2 neighboring branches. On the other hand, if the start and end point are separated by many other branches that remained opened (resulting from other tasks), scrolling will be required and finding the beginning and end points will be much more difficult

and frustrating, overweighing the advantage of seeing multiple open branches.

Results: One of the two tasks (the longer one involving a return trip between 2 known locations) showed significant differences. SpaceTree was significantly faster than Hyperbolic, and Explorer was significantly faster than the two other interfaces (E=6.5, H=22.7, S=15). Explorer was favorably helped by the ability to keep several branches opened. The other very short returning task did not show any significant differences. Explorer lost its advantage because other open branches now separated the target nodes.

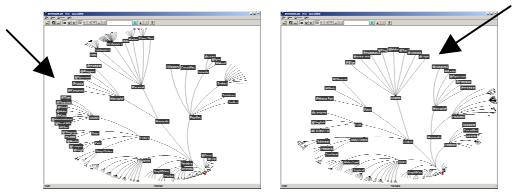

Figure 16: with Hyperbolic the layout changes between visits. Kangaroo was on the right of screen (a), now on the left (b). With SpaceTree the relative location of nodes is more consistent.

For topology tasks:

C) Listing all the ancestors of a node

We had predicted that the SpaceTree would perform better than both Explorer and Hyperbolic as all ancestors are clearly visible and highlighted. Hyperbolic gives more screen real estate to the local lower levels therefore often hiding the ancestors, while Explorer keeps the path visible but the small offset makes it hard to separate siblings from parents.

Results: SpaceTree was significantly faster than Explorer (E=11.4, H=9.3; S=6.8). Two users made errors with Explorer (alignment problems) and one user made an error with Hyperbolic (skipped a level). Two users commented that they liked the clear highlight of SpaceTree along the path, in this path task as well as during other tasks.

D) Local topology (task: find 3 nodes that have more than 10 direct descendants):

We predicted that Hyperbolic would be faster that the SpaceTree, which would be itself faster than Explorer. With Hyperbolic users would be able to estimate the number of children by looking at the number of rods radiating from a node, and navigate through the leaf nodes by continuously fanning the tree at a varying depth level.

Results: Hyperbolic was significantly faster than the SpaceTree, but not significantly faster than Explorer (E=61.4, H=46.8, S=98.3). Hyperbolic users interpreted correctly the fans of lines, and Explorer users mostly chance. This task showed that SpaceTree users had not understood the width coding of the triangles (or didn't trust their understanding). Users could be seen intuitively following wider and darker triangles, but would give up after following 2 or 3 level down, even though the answer was often one click away because large fans were usually at leaf level. A wide base triangle only suggests that "somewhere" down the tree there are large fans. Obviously better coding is needed. The experiment was run with the icon size being relative to the parent, making it more usable for local comparisons, but also more confusing as its meaning appeared to change with the depth in the tree. Icons relative to the root would probably be more easily understood.

E) *Topology overview task* (example: Which of the 3 branches of "measurements" contains a larger number of nodes). We hypothesized that SpaceTree would lead to fewer errors in the estimation of size because of the icon representation of the branches. We had first measured the time to complete the task, but pilot test users spent so much time with Explorer and Hyperbolic trying to open every branch of the tree – without great success – that we gave a time limit and compared error rates.

Results: Users made 12 errors with Explorer (out of 18), 10 with Hyperbolic and only 2 with SpaceTree. Explorer users mostly made wild guesses or used "properties". Hyperbolic users were able to review the tree quickly but still made many errors, often deciding for a branch that was less than half the size of the correct answer (150 nodes versus 300). SpaceTree users seemed to have made errors when the small differences in the shading of the icons were confounded by size differences.

F) User preferences

Our hypotheses were that users would find the Hyperbolic Browser more "cool" than Explorer and SpaceTree, but would prefer to use the SpaceTree.

Results: Users significantly found Explorer less "cool" than the other interfaces, and no significant difference were found between SpaceTree and Hyperbolic (mean ratings on the 9 point scale with 9 being "very cool" were E=3.9. H=7.7, S=6.6.) There were no significant differences between interfaces in term of future use preference (E=5.9, H=5.1; S=6.2 with 9 being "much prefer to use").

Summary of results

Our hypotheses were only partly supported, but the careful observation of users during the experiment was very helpful to understand differences in user behavior. There were wide differences between subjects in terms of speed, leading to only a limited number of statistically significant results. There were also wide differences in preferences, confirming the general need for providing interface options to users. During training, we observed that users did not guess the 3-attribute-coding of the triangle that always had to be clarified. Users could guess that the icon represented the branch below and was linked to the number of nodes in the branch, but often misinterpreted the width of the triangle to be proportional to the number of direct descendants. This

miscomprehension of the meaning of the icons had a particularly strong effect on the task that asked users to find nodes with more than ten descendants. Future research will focus on the design of a simpler preview for novice users, as well as a set of options for expert users who should be able to adapt the icon to their tasks.

Conclusions

SpaceTree illustrates that interactive visualization of node link diagrams can still be improved. It was found more attractive than Explorer, and performed relatively well for both navigation and topology tasks, even though no extreme performance differences were found between the interfaces. SpaceTree's consistent layout allowed users to quickly return to nodes they had visited before, making it more appropriate for trees that are used regularly. An example of this would be an organization chart used by a personal staff. SpaceTree preview icons are unique in helping users estimate the topology of the tree, and we will continue improving their design.

For more information see:
www.cs.umd.edu/hcil/spacetree

Acknowledgements

We appreciate the feedback and suggestions to improve SpaceTree from Cheryl Lukehart and Don Schiro from Chevron-Texaco and from Jean-Daniel Fekete and Ben Shneiderman from HCIL. Partial support for this research was provided by Chevron-Texaco and DARPA.

References

[1] Andrews, K., Heidegger, H. (1998) Information Slices: Visualising and exploring large hierarchies using cascading, semicircular disks. *Proc of IEEE Infovis'98 late breaking Hot Topics* IEEE, 9-11. ftp://ftp.iicm.edu/pub/papers/ivis98.pdf

[2] Beaudoin, L. , Parent, M-A, Vroomen, L. (1996) Cheops: a compact explorer for complex hierarchies, *Symposium on Volume Visualization - Proc. of the conference on Visualization '96*, 87-92 + color p. 471, ACM, New York

[3] Beard, D. V., Walker II, J. Q. (1990). Navigational Techniques to Improve the Display of Large Two-Dimensional Spaces. *Behavior & Information Technology*. **9** (6), 451-466.

[4] Bederson, B., Shneiderman, B., Wattenberg, M. (2002). Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies, *To appear in ACM Transactions on Computer Graphics*.

[5] Bederson, B. B., Meyer, J., & Good, L. (2000). Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java. *UIST 2000, ACM Symposium on User Interface Software and Technology, CHI Letters*, *2*(2), 171-180.

[6] Bertin, J. (1983) *Semiology of Graphics, Diagrams, Networks, Maps*, University of Wisconsin Press, Madison, WI.

[7] Card, S. K., MacKinlay, J. D., Shneiderman, B., (1999) *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann Publishers.

[8] Chignell, M, Poblete F., Zuberec, S. (1993) Exploration in the Design Space of Three-Dimensional Hierarchies *Proceedings of the Human Factors and Ergonomics Society 37th Annual Meeting,* v.1, 333-337

[9] G. W. Furnas (1981) The FISHEYE view: a new look at structured files, *1981 Bell Lab. Tech. Report*, reproduced in *Readings in Information Visualization: Using Vision to Think*, S. K. Card, J. D. Mackinlay, and B. Shneiderman, Eds. San Francisco: Morgan Kaufmann Publishers, Inc., 1998, 312-330.

[10] Hopkins, D. (1989), The Shape of PSIBER Space: PostScript Interactive Bug Eradication Routines.. *Proc. 1989 Usenix Graphics Conference*, Monterey California. www.catalog.com/hopkins/psiber/psiber.html

[11] Hightower, R. R., Ring, L., Helfman, J., Bederson, B. B., & Hollan, J. D. (1998). Graphical Multiscale Web Histories: A Study of PadPrints. In *Proceedings of ACM Conference on Hypertext (Hypertext 98)* ACM Press, 58-65.

[12] Johnson, B. and Shneiderman, B. (1991) Tree-maps: A space-filling approach to the visualization of hierarchical information structures, *Proc. IEEE Visualization' 91* (1991), 284 – 291, IEEE, Piscataway, NJ.

[13] Jul, S., & Furnas, G. W. (1998). Critical Zones in Desert Fog: Aids to Multiscale Navigation. *In Proceedings of* User Interface and Software Technology (*UIST 98*) ACM Press, 97-106.

[14] Kumar, H.P., Plaisant, C., Shneiderman, B. (1995) Browsing hierarchical data with multi-level dynamic queries and pruning *International Journal of Human-Computer Studies,* Volume 46, No. 1, 103-124 (January 1997).

[15] Lamping, J., Rao, R., Pirolli; P. (1995) A focus+context technique based on hyperbolic geometry for visualizing large hierarchies *Conference proceedings on Human factors in computing systems*, 1995, 401-408

[16] Mullet, K., Fry, C., Schiano, D. (1997) On your marks, get set, browse! (the great CHI'97 Browse Off), Panel description in *ACM CHI'97 extended abstracts*, ACM, New York, 113-114

[17] Nation, D.A., Plaisant, C., Marchionini, G., Komlodi, A. (1997) Visualizing websites using a hierarchical table of contents browser: WebTOC, *Proc. of 3rd Conference on Human Factors and the Web*, 1997, Denver, CO, June 12.

[18] Ellson, J., Gansner, E., Koutsofios, E., Mocenigo, J., North, S. , Woodhull, G., Graphviz, open source graph drawing software, http://www.research.att.com/sw/tools/graphviz/

[19] Noik; E. (1993) Exploring large hyperdocuments: fisheye views of nested networks, *Proceedings of the fifth ACM conference on Hypertext*, 192-205.

[20] Robertson, G. G. Mackinlay, J. D. Card, S. K. Cone Trees: animated 3D visualizations of hierarchical information, *Proc. Human factors in computing systems conference,* March 1991, 189-194.

[21] Shneiderman, B. (1994). Dynamic queries for visual information seeking. *IEEE Software*, **11,** (6), 70-77.

[21] Walker II., J. Q. (1990) A node-positioning algorithm for general trees. *Softw. Pract. Exp*., 20(7): 685-705, 21