PygmyBrowse: A Small Screen Tree Browser

Zvi Band
Human-Computer Interaction Laboratory
University of Maryland
College Park, MD 20742, USA
skeevis@skeevis.com

Ryen W. White
Human-Computer Interaction Laboratory
University of Maryland
College Park, MD 20742, USA
ryen@umd.edu

CHI 2006, April 22–27, 2006, Montreal, Canada.
ACM 1-xxxxxxxxxxxxxxxxx.

Abstract
We present PygmyBrowse, a browser that allows users to navigate a tree data structure in a limited amount of display space. A pilot evaluation of PygmyBrowse was conducted, and results suggest that it reduces task completion times and increases user satisfaction over two alternative node-link tree browsers.

Keywords
Navigation, trees, browsing

ACM Classification Keywords
H.5.2 – User Interfaces (Specifically, interaction styles)

Introduction
There is a long history of storing and visualizing data in trees, including thesauri, organizational charts, indices, and more recently Extensible Markup Languages (XML). Storing information in this way creates a need to provide interface support to navigate within such structures. Two main types of solutions have been proposed to display and manipulate trees interactively: space-filling and node-link techniques.

Space-filling techniques (e.g., TreeMaps [5]) have been successful at visualizing trees with attribute values at the node level. These techniques perform best when users care mostly about leaf nodes and their attribute

values but not the tree topology, or tree topology is trivial. Node-filling techniques leave the root side of the tree empty and overcrowd the opposite side, wasting display space. Optimized layout techniques produce more compact displays by shifting branches or nodes [4], but those techniques only partially alleviate the problem and can be inappropriate for interactive applications [5]. Other techniques, coupling overview + detail with pan and zoom [1], three-dimensional node-link diagrams [6], and circular layouts [2] have also been proposed.

An issue with all of these techniques is that they require a substantial amount of display space to be used effectively. For software applications and interfaces that either cannot dedicate a large amount of display space, or are compact applications themselves (e.g., PDAs, portable music players, toolbars), a new interface method needs to be developed to facilitate tree browsing.

PygmyBrowse

PygmyBrowse is our attempt to provide a method for browsing trees in a confined display space. To navigate a tree, users are first presented with one panel representing the root node, and a scrolling list of candidate child nodes. Two types of node are displayed in this list: *interior nodes* (i.e., nodes that contain child nodes), and *leaf nodes* (i.e., nodes that do not contain child nodes). Interior nodes are distinguished from leaf nodes by element color and contain in their label a count of the number of items categorized or classified into that node (shown as "choices"). To facilitate more effective navigation decisions, interior nodes containing less than five children are shown in red and those containing five or more children are shown in blue. As illustrated in Figure 1, as a user navigates the tree, panels are added to the top of the PygmyBrowse to depict their browse path.

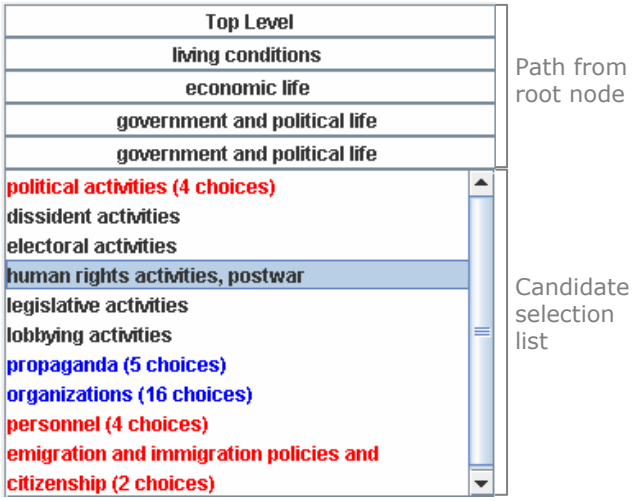

Figure 1: PygmyBrowse interface. In this example user is five levels down in the tree and has selected a leaf node.

At any point the user has three interaction options:

1. **Go deeper in the tree.** When the user clicks on an interior node, two things happen: (i) the selected choice is added to the browse path displayed at the top of the PygmyBrowse interface, (ii) the selection list is updated to display all children of the selected node that are one level down in the tree.
2. **Return to a higher level.** Selecting a panel in the browse path returns the user to that level. The selection list is then refreshed, showing the nodes contained within the selected node.
3. **Select a leaf node.** As mentioned earlier, leaf nodes are distinguishable from interior nodes by color (and lack a "choices" label). When a leaf node

is selected, no change occurs in the component other than the leaf node being highlighted. This typically marks the end-point of a browse path.

Pilot Evaluation

We conducted a pilot evaluation of PygmyBrowse with a sub-tree of a thesaurus developed by the Survivors of the Shoah Visual History Foundation to classify video-taped interviews with Holocaust survivors, witnesses, and rescuers. The sub-tree consisted of approximately 6000 nodes with an average depth of 6 nodes. The use of this data structure was in line with the aims of our research project and provided a reasonable data source for this pilot evaluation.

PygmyBrowse was compared against two alternative node-link based tree browsing interfaces in a restricted display space 100 pixels wide by 150 pixels high. A within-subjects experimental design was employed, and systems and tasks were counterbalanced to counteract learning effects. Participants were assigned search tasks, and we measured the time to complete each task (timed by an observer using a stopwatch), and the number of correct answers to questions (assessed *a posteriori* by experimenters). Questionnaires used Likert scales, semantic differentials and open-ended questions to elicit participant opinions [3]. We now describe the two comparator systems, tasks, research questions, and other experimental issues.

Systems

We compare the PygmyBrowse system against an expandable outliner (as a Java JTree), and SpaceTree [5], an interactive tree browser based on dynamic rescaling. Since the topology of the tree was complex, and the leaf nodes had no attributes other than their label we chose to compare PygmyBrowse against two node-link interfaces rather than space-filling versions.

EXPANDABLE OUTLINER

Interfaces such as the "Explorer" facility in Microsoft Windows generally display data as an indented list of nodes. The baseline interface in our evaluation displays the underlying data structure in this way (Figure 2).

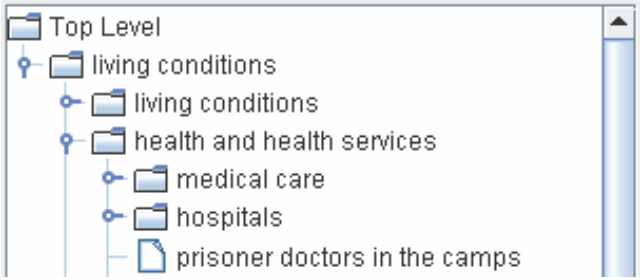

Figure 2: Tree presentation in the Expandable Outliner.

Interior nodes and leaf nodes are distinguished by icons displayed next to them. Presenting trees in this format facilitates determination of relative levels, and the viewing of alternative nodes at higher levels. However, it can be difficult to distinguish between deeper levels in the tree, scrolling down can hide ancestor nodes, and preservation of state can disorient users. All such problems are more acute given restricted display space.

SPACETREE

Perhaps the most understandable way of navigating a tree is to actually view it as a tree, and allow users to dynamically rescale it as they browse. Currently, one of the best examples of this is SpaceTree [5], shown in Figure 3.

Figure 3: Tree presentation in SpaceTree.

Like the Expandable Outliner, there are advantages and disadvantages to this method. Understanding the structure of the underlying data using SpaceTree is easy, users find it fun to use, and they can zoom in and out to resize the tree. However, when browsing deeply in the tree or viewing the tree in small screens users may lose track of ancestors, expanding a node with many children can unbalance the tree, and moving around the tree can be disorienting.

Although other interfaces such as those described in the "Introduction" could have been compared to PygmyBrowse, the Expandable Outliner and SpaceTree gave reasonable scope for comparison.

Tasks

Participants attempted five tasks on each of the three interfaces, with a two minute maximum per task, imposed to improve experimental consistency. There were three types of task, based on the classification used in [5]:

1. Node search (e.g., find node entitled "cannibalism")
2. Navigate from previously visited nodes (e.g., from "cannibalism" to "field hospitals")
3. Topology questions (e.g., How many steps would you have to make to get between node "labor affairs organizations" and "field hospitals"?)

Tasks were rotated according to a Graeco-Latin square based on their classification. The resultant experimental design is shown in Table 1.

Table 1: System/Task type assignment for all six participants.

Participant		System,Task type		
		1	2	3
1	4	SA,T1	SB,T2	SC,T3
2	5	SB,T3	SC,T1	SA,T2
3	6	SC,T2	SA,T3	SB,T1

As can be seen from the table, the order of the five tasks within each task type does not change between participants, only the type order is counterbalanced.

Research Questions

Three research questions drove our investigation:

1. Which system performs best for tasks involving specific node searches?
2. Which system performs best for tasks involving navigation from previously visited nodes?
3. Which system performs best for tasks about the topology of the data structure?

The dependent variables (task time and correctness) are used to answer these questions.

Participants

Six participants were recruited from within our research project and beyond. To provide motivation to perform well a $10 bonus was offered to the fastest participant (with no incorrect answers) on each system. The median age of participants was 25.5 (range: 23—49), they used computers daily, and tree browsers occasionally.

Experimental Procedure

The experiment lasted 45 minutes. Each participant followed this procedure:

1. welcomed, introduced to the experimental goals, and completed introductory questionnaire;
2. attempted five tasks on the first system (assigned according to experimental design);
3. completed short post-system questionnaire;
4. repeated Steps 2 and 3 for other systems, and;
5. completed final questionnaire.

Minimal training was given to participants since part of our evaluation was intended to measure how usable the systems were when participants were given no training. The trees displayed in the interfaces were re-contracted between each participant, but not between each task, to simulate how the tool would generally be used.

Findings

Running six participants in our chosen experimental design gave us 10 task times and potential answer keys for each task-system pair. We set the level of significance is to $p < .05$, and given the small sample size and abnormality of the distribution across task-system pairs and systems alone[1] we use non-parametric statistical testing [7].

Task times and Answer correctness

The average task times and answer correctness for each of the three task types is shown in Table 2. Trends in the results suggest that PygmyBrowse leads to the lowest task completion times for all three search task types. We applied non-parametric Kruskal-Wallis Tests to the data gathered for each task type.[2] The results of the tests revealed no significant differences between the systems for each task type (see Table 2). We compared task times for the three systems across all tasks using a Friedman Rank Sum Test; differences between systems were not significant.[3] Although there were insufficient data to perform sound statistical testing on the number of tasks answered correctly on each system, the results are suggestive of statistically insignificant differences had a larger sample been used.

Participant perceptions

Participants were asked for their opinions of the three systems immediately following each set of tasks. Although they completed Likert scales and semantic differentials in these questionnaires the small sample size restricts our analysis of participant responses to something less rigorous that full-blown statistical significance testing. Tables 3 and 4 present some findings from the post-experiment questionnaire where participants selected preferred systems and answered open questions. Results indicate that subjects preferred the PygmyBrowse system and generally found it easier to learn, easier to use and overall.

[1] The sample did not conform to a normal distribution using a Shapiro-Wilk Test ($ps > .386$).

[2] Since this analysis involved many tasks, we use a Bonferroni correction to control the experiment-wise error rate and set the *alpha level* (α) to .0167, i.e., .05 divided by the number of task types. This correction reduces the number of Type I errors i.e., rejecting null hypotheses that are true.

[3] $\chi^2(2) = 2.56$, $p = .287$, where $N = 26$. Tasks lasting longer than the two minute threshold were terminated by the experimenter and marked as incomplete.

Table 2: Task completion (time and number).

Task type	ExpOutliner		SpaceTree		PygmyBrowse		Significance (Kruskal-Wallis Test)
	Time (secs)	Number	Time (secs)	Number	Time (secs)	Number	
1	31.11	9	22.75	10	15.29	7	$\chi^2 = 2.05$, df = 2, $p = .359$
2	44.78	10	56.50	10	20.00	10	$\chi^2 = 3.48$, df = 2, $p = .176$
3	43.11	10	61.22	10	34.44	10	$\chi^2 = 5.25$, df = 2, $p = .074$

Table 3: Participant preferences

Participant	Easiest to Learn	Easiest to Use	Overall
1	PygmyBrowse	PygmyBrowse	SpaceTree
2	SpaceTree	SpaceTree	SpaceTree
3	PygmyBrowse	PygmyBrowse	PygmyBrowse
4	ExpOutliner	PygmyBrowse	PygmyBrowse
5	SpaceTree	SpaceTree	ExpOutliner
6	PygmyBrowse	PygmyBrowse	PygmyBrowse

Table 4: Participant comments.

ExpOutliner
+ "Preserves state" [P2], "Easy to Learn" [P3], "Familiar" [P6]
– "Loss of parent node when scrolling" [P1], "disoriented after a while" [P3], "a lot on screen at once" [P4]

SpaceTree
+ "Fun to use, understandable structure" [P6]
– "Too slow to actually use" [P5], "Limited visibility" [P6]

PygmyBrowse
+ "Easy to manipulate" [P3], "Compactness" [P1]
– "Random ordering" [P4], "Color coding not helpful" [P1]

Conclusions

In this paper we have presented an approach to tree browsing in a confined display space. The results of our pilot evaluation were promising. They suggest that PygmyBrowse may lead to lower task completion times, especially for tasks where the target is known, or where users need to move between two nodes on separate branches of a tree. Browsing trees in confined display spaces in this way is an area worthy of further investigation. In future work we will enhance PygmyBrowse based on participant feedback, and run a larger sample of participants with other data structures.

References

[1] 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.

[2] Bertin, J. (1983). *Semiology of graphics, diagrams, networks, maps*. University of Wisconsin Press.

[3] Busha, C.H., Harter, S.P., (1980). *Research methods in librarianship: Techniques and interpretation*. New York: Academic Press.

[4] Ellson, J., Gansner, E., Koutsofios, E., Mocenigo, J., North, S., Woodhull, G. (2005). *Graphviz, open source graph drawing software*. http://www.research.att.com/sw/tools/graphviz/

[5] Plaisant, C., Grosjean, J., Bederson, B.B (2002). SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proc. of Infovis*, pp. 57-64.

[6] Robertson, G. G. Mackinlay, J. D., Card, S. K. (1991). Cone Trees: Animated 3D visualizations of hierarchical information. In *Proc. of SIGCHI*, pp. 189-194.

[7] Siegel, S., Castellan, N.J. (1988). *Nonparametric statistics for the behavioural sciences.* 2nd ed. Singapore: McGraw-Hill.