INTRODUCTION

The need for better information visualization and navigation tools is widely recognized [1], [2], [7], [8], [10]. It is difficult to sort and resort tables or listings by more than one attribute and still maintain an understanding of the origin/position of items. The concept of "value bars" was created to help users visualize and navigate large information spaces that have characteristics of a line-oriented listing with multiple, quantifiable attributes. Value bars provide a graphical means of displaying quantifiable attribute values and can be thought of as a cross between a scroll bar and a vertically-oriented stacked bar chart (or one-dimensional tree-map [6], [9]). This unique combination provides navigation and information visualization techniques that may allow for a more comprehensive understanding of a large multi-attribute listing. The general idea of value bars was developed by Shneiderman as part of ongoing data visualization research at the Human-Computer Interaction Laboratory. The specific design, terminology, interaction techniques, implementation, and future research ideas for value bars are this paper's contributions. An implementation is written on Unix in the OpenWindows environment on a Sun SparcStation.

DESCRIPTION

A value bar is a thin stripe added to a text window (figure 1), placed next to the scroll bar if one exists; any number of value bars may be added as screen space allows. Each value bar maps one specific attribute shared by items. The attribute must be quantifiable through some mechanism because each item gets assigned a weight for its attribute value. The weight can be assigned linearly or other methods can be used if the attribute value range is especially small, large, or unevenly distributed. The partitioning algorithm equates the sum of the items' weights to the value bar height. Each item is given a height in the value bar proportional to its part of the total weight of all items. Each varying height region is placed vertically in the value bar from top to bottom in the same order that its item appears in the listing sequence. The end result is a graphic image that looks like a ladder with varyingly spaced rungs. Once partitioned the value bar regions are not changed, moved, or scrolled, all regions are always in view. A minimally useful height is set such that shorter rectangles are not given value bar regions, though their region heights are still accounted for in the partitioning algorithm. Items represented in the value bar provide a global view of the distribution of an attribute's values in the listings. This provides for another variant of the much heralded fisheye view [3], [5]. An item is represented by differing height regions in distinct value bars; it is possible an item may not be represented in every value bar.

When an item becomes selected either by a mouse click in the text window or in any value bar, all representations of that item must be highlighted to provide visual feedback as to everywhere that item is represented. There are many sensible ways to highlight a value bar rectangle, the current implementation inverts the rectangle from white to black. The item in the text window is also highlighted with inverse video to be in harmony with both value bar rectangle highlighting and routine text selection.

Another value bar artifact is the "visibility marker." Many scroll bars provide an indicator of how much of the entire listing is currently in view, this concept is a requirement of a good value bar implementation. The current implementation uses exactly the same graphic characteristics as the OpenWindows scrollbar: the solid black part of the gray stripe shows which part of the whole is currently in view. Whenever the text window is scrolled, visibility markers in each value bar are moved independently to show which rectangles represent the text items currently in view.

APPLICATION TO UNIX DIRECTORY LISTINGS

Consider a Unix directory listing with value bars for file size and file age (figure 1). Along the right side of the window are two value bars. The "S" value bar maps file size largeness (Size) and the "Y" value bar maps file modification recency (Youth). The rest of the window is a normal scrolling textpane devoted to the directory listing. Each value bar is partitioned into individual rectangular regions. A region's height represents that part of the attribute total for a particular listing item; both value bars use a linear weighting assignment. The partitioning concept should be easy to comprehend for the Size value bar, for the Youth value bar the concept is less clear. Each listing item gets assigned a weight that is the number of days from February 8, 1990, that the file was last modified; this date was chosen because that is the oldest modification date for any file. More recently modified files have a larger weight associated with them. The Size value bar has a bigger variance in region heights than the Youth value bar because its values vary more widely. The mapped size values vary from 60,000 - 1,000,000 (factor of 16), the mapped age values vary from 147 - 460 (factor of 3).

DISCUSSION

One of the most important value bar advantages is its ability to provide a global distribution overview of attribute values in a single view. Users are able to compare an item’s attribute value to other items in the value bar without the need to scroll the text items or rearrange the sort order many times. Users receive many attributes' distribution overviews at the same time in the same view. Noticing that one item has large regions for many attributes may be enlightening. Within one value bar, noticing clusters of large regions together may be an important insight to users. The fact that an attribute has values that are relatively equal throughout the items (e.g. all files are similar in size) can be recognized easily; this knowledge might allow users to realize that attribute is not of interest to them since the values do not vary significantly and will not be helped by a value bar. There are many ways the single view of multiple attribute distributions can help users distill trends or notice interesting clusters. This also may allow new discoveries that couldn't have been made due to the high cognitive load of resorting a list many times while retaining acquired knowledge about items. Sorting by one attribute at a time has several disadvantages that value bars overcome:

- a completely different view of items for each sort,
- no external memory aid for remembering interesting items from a previous sort, and
- no overview to gauge how one item's values for multiple attributes compare simultaneously to other items' values.

The general concept of the value bar with its unique combination of information visualization and immediate navigation makes the contribution, not the physical characteristics of this implementation. The highlighting characteristics of the selected region and of the visibility marker need only follow reasonable principles of user interface design; the black triangle and the thick lines are simply two choices. The width of the value bar is not relevant to its semantics. Being able to fit a horizontal text label above the value bar seems logical, but other methods could be as effective. An early user has suggested that a wider value bar would bring more attention to it, though the value bar's usefulness and the identifiability of the individual regions seem independent of width. Having a minimum region height is an issue of information complexity reduction and of interaction convenience. Only the bigger regions (more important values of a certain attribute) are of interest to users and not the small ones. If the small regions are of interest, then the inverse attribute should be mapped in its own value bar; file size smallness instead of largeness, for example. Value bars work well for hundreds of items that requires several screenfuls to view, but could really be helpful for very large information spaces on the order of thousands or hundreds of thousands of items when scrolling or even resorting become unwieldy.

The immediate navigation provided by value bars allows users to jump around and examine individual items as the need arises. There is no guess work as to how much scrolling needs to be done to view an item nor clumsy search specification. There is low likelihood of becoming lost; but should that happen, a single click will reposition the items with certainty. There is a natural blend of task with interface that allows users to continually concentrate on the task with minimal interface effort.

Some interesting issues arise when several value bars are attached to a multi-attribute listing. Some items won't be represented in all value bars. Users can compare an item's mix of different attribute values to other leading items for each attribute to decide how interesting is an item. The ability to compare multiple attributes' distributions in one view has great information analysis potential. Users can see that some attributes have values that are relatively equal among the items mapped, while other attributes' values have a wide range.

An item has different values for each attribute and therefore is represented by different size regions in different value bars. Specifically, those items in view in the text window are marked by a visibility marker in each value bar; each visibility marker will almost assuredly be of different height. As the text window is scrolled with the normal scroll bar, the visibility markers move along their respective value bars in varying size jumps. More importantly, each visibility marker changes size with each scroll because there is a new set of items visible in the text window with a different set of varyingly sized value bar regions representing them. The changing size visibility marker helps users understand where a certain set of contiguous value bar regions are in the listing. This concept and its usefulness may be difficult to visualize without seeing a value bar in action.

APPLICATION SUITABILITY

In general, value bars are useful for analyzing multi-attribute listings and tables where a particular sort order should be maintained and analysis of the top percentage of items for each attribute is beneficial. The main features of the value bar are:

- the ability to see in one view an attribute distribution overview for the "important" items (as defined by attribute values) in a fisheye view variant,
- very small screenspace footprint,
- the ability to see at once many attribute overviews,
- the ability to locate outliers and exceptions, and
- low cognitive load navigation.

Some of the task domains for which value bars are well suited include directory listings, stock market tables, inventory systems, medical information, and database search output. To expand on the last domain, consider the hierarchical text browsing system SuperBook [4]. Users of Superbook specify a query and the number of paragraphs that match the query are displayed next to table of contents entries. A good strategy for users to follow is to find the section with the highest number of hits and then browse that section for relevant information. Value bars could be used here beneficially when the table of contents (TOC) is large. Without the value bar, users may not know that an item(s) several screenfuls away in the TOC actually has more hits than a TOC item currently on the screen. Users seeing a high number of hits for a TOC item currently on the screen would likely begin to browse that item rather than scroll through the TOC (probably fruitlessly) to make sure no other items have more hits. However, a value bar mapping the number of hits for each TOC item would allow users to see in a glance how the distribution of hits occur within the TOC, navigate quickly to those locations, and use SuperBook conventionally from there.

USABILITY STUDY

A usability study was conducted with seven participants from the computer science department at the University of Maryland familiar with UNIX. A within-subjects design had them perform the exact same set of tasks on two different directories, /usr/lib and /usr/include as can be found on any UNIX system. The first iteration of the task set was performed with the UNIX command line in a scrolling text window, the second iteration was performed using a value bars interface (figure 1). Seven tasks were performed; they were tasks for finding the largest file, the newest file, identifying file names directly before and after (alphabetically) the largest and newest files, deciding if there were "clusters" of "large" and "new" files (quotes appeared this way in the task descriptions allowing participants to define these terms), and for deciding if any of the top five large files were also in the top five newest files. The use of the word file implies both file entries and directory entries; participants were directed to interpret the size of directories to be the number shown in the `ls -l` listing and not to consider the sum of file sizes within the directory. The domain of directory listings and tasks performed therein may not be rich enough to show adequately the power of value bars. However, with the resources available this domain was the only feasible choice. Future formal experiments will use a richer domain, more participants, and better high-level tasks.

These tasks were chosen to investigate a number of hypotheses about using a value bars interface:

- seeing an attribute value distribution overview while maintaining the original sort order is helpful,
- there is greater analysis potential seeing many attribute value distributions than in seeing one at a time, and
- immediate navigation to interesting items is useful.

Again, this domain is not very rich and system administrators would be the users most likely to benefit from the application of a value bars interface in this domain. But imagine a sales organization domain where there exists a listing of sales people with attributes for sales, number of customers, phone bills, hotel bills, etc. A manager could view these attributes and determine, without even thinking about the fact that a database of attribute values is being "queried," interesting trends and exceptions to trends in combinations of attribute values. The manager could discover that a certain salesperson has high sales volume but a low number of customers and hotel bills (by mapping in value bars the attributes sales largeness, customer number lowness, and hotel bill lowness, respectively). This could be an exception to a trend that no one else has high sales and low bills, therefore sparking motivation to spread this combination of attributes. In this way, value bars act as a database engine that has precomputed all the single attribute queries which result in the topmost valued items to be represented (as relatively sized box heights). The combined view of all these single attribute queries allows for serendipitous discovery of interesting items by virtue of their combinations of attribute values.

General Interaction Observations

Most participants during the UNIX phase performed ls -l commands (-l lists in long format providing permissions, number of links, owner, size in bytes, and file modification time), some adding the options -s (-s precedes the file name with size in number of blocks) and -1 ("one", which forces output into one column), with only two using the -t option (-t sorts by file modification time). Most used the scrollbar to review the same directory listing for many tasks, while one participant refused to use the scrollbar. Each directory contains on the order of 150 entries, most participants felt comfortable scrolling and scanning rather than feeling compelled to using the most efficient methods which likely involve many (somewhat obscure) commands and pipes.

During the value bar phase, participants used the value bars as expected for a first time encounter. Most tasks were biased towards finding files that were represented by the tallest boxes in the value bars, participants realized this relationship and successfully took advantage of it. Task completion times did not vary from the range of 10 to 60 seconds, regardless of the complexity of the task. Generally, they thought the value bars showed attribute value information uniquely, concisely, and interestingly, but weren't sure of their utility. One of the higher level information visualization aspects of a value bars interface rarely was used effectively; namely, the attribute value distribution overview. By the time participants reached the tasks for determining if "clusters" of files existed for the two attributes, they should have had enough experience to realize they could study the overview and clearly state that no clusters existed (in one of the conditions) by noticing there were never two or more tall boxes stacked upon each other. However, one participant did realize the converse--that the presence of tall boxes stacked on each other did not guarantee that they were clustered in the full listing because some items may not have boxes representing them in the value bar.

Observation by Task

The most startling fact discovered is that most of the participants were not aware of the -t option for the UNIX command ls, the option which sorts the directory listing by file modification date automatically. When asked to find the newest file, those participants simply scrolled an ls -l listing keeping track of the newest file. However, this is not so striking when one considers that ls has the most options of any UNIX command and that most people don't look at directories other than their own in everyday situations. Participants took about 60 seconds solving this task with UNIX. One participant issued a plethora of convoluted pipes from ls to sort in an attempt to solve this task, but fell back to the certainty of redirection into files and using diff. In contrast, when using the value bars interface, all participants easily identified the newest file by locating the tallest box in the Youth value bar in about 10 seconds on average. Participants had absolutely no comprehension problem with realizing the tallest box represented the newest file. The only problem encountered was determining which box was the tallest, this is elaborated later.

The task to find if any of the top five largest files was also among the top five newest files proved to be the most interesting, as expected. Proficient UNIX participants were able to complete this task successfully in about 10 seconds with the help of the -t option to ls. The rest took about 60 seconds, scanning up and down an ls -l listing while trying to determine and remember the cutoff points for the top five of each attribute. One participant took over five minutes to accomplish the task using UNIX. With the value bars interface, all participants quickly formulated a plan and carried it out straightforwardly without problems in about 45 seconds on average. The plan (apparently) used by all was to locate the five tallest boxes in a value bar, click in them, and determine if any (items represented by that box in that value bar) had any boxes in the other value bar that were among the top five tallest. This is how it was thought participants would put a value bars interface to work. The task was biased to be solved more straightforwardly by the value bars interface, but the plan and interaction success thereof was independently carried out by all participants. Upon reading the task description during the UNIX command line phase, most participants (those not knowing the -t option to ls) grunted or moaned probably because they felt it may be a difficult or tedious task. However, quite oppositely, during the value bars interface phase most participants made an audible or visual cue that suggested they suddenly realized the potential power of a value bars interface.

It is interesting to discover that only one participant attempted to sort a directory listing by size to solve the task of finding the largest file using the UNIX command line. The others were content to scan.

That same participant failed for 60 seconds trying many advanced uses of commands to name the two files alphabetically before and after the largest file just found. That task was created exactly to see how re-sorting in UNIX might effect sequential tasks involving different attributes. This may be a strong indication for the value of a value bars interface which preserves the primary sort, alphabetical for this domain. One last curiosity is to note that this singled out participant using advanced commands and pipes did not know about the `-t` option to `ls`. This can only strengthen the case for applying simple to use value bars to complex interfaces on multiple attribute domains.

Subjective Data

When asked to identify the best feature about value bars, most participants decided it was the ability to locate the tall boxes representing the topmost valued items for an attribute. One participant went further to say it's not just locating the tall box but the immediate navigation to viewing the item (file) in context (of the full directory listing) that makes value bars useful. One participant mentioned the best feature is the ability to see multiple attributes for a single item in comparison to other items' values and the whole.

When asked to identify the worst feature about value bars, three participants mentioned that it is hard to distinguish the difference in values for similarly sized boxes in the value bar. In this situation, users are forced to click in the box to navigate to the file in the listing to look up the attribute value. This is a known difficulty with the information presentation but only affects certain categories of tasks. There are several ways to assist users in this regard, one that could be quite helpful would be to display temporarily the actual value directly above the mouse pointer upon a different input event than normal. In this situation, users are interested first in the absolute value and are interested second (if at all) in navigating to the item to see the other attribute values. Another solution to this problem is discussed two paragraphs below. One participant thought having more than one value bar was not useful; this is interesting since a different participant found that having more than one value bar was the best feature. One participant thought that items not given boxes is confusing and suggested the "zoom" feature elaborated in the future research section below.

Participants were also asked to describe an additional feature that could be added to value bars to make them more useful. Only one suggestion was not previously thought of and elaborated in the future research section below. The novel suggestion was to help with determining the difference in similar height boxes by marking the next N taller and next N shorter boxes on demand, perhaps by placing the numbers 1..N and -1..-N in the appropriate taller and shorter boxes, respectively. In this way, it is easy to locate and rank the N similarly sized boxes which are indeed shorter and taller than the current box.

FUTURE RESEARCH

There are a large number of exciting avenues to explore further with the value bar concept. One is to employ smarter algorithms for various aspects of value bar graphic representations. Determining the nature of value ranges and distributions may lead to automatic selection of weight assignment mappings other than linear. One could imagine various data transformations, logarithmic mappings, step functions, etc. Another aspect is how to determine the initial number of items to be given regions in the value bar.

Perhaps users may not always be interested in the topmost valued items, the "middle" items may be of concern to users. Items can be mapped that would have been left out of value bars mapping only the topmost items. Also, "zoom" regions could be introduced for items that would normally not be mapped in the value bar due to a combination of their value, the mapping algorithm for assigning region heights, and the minimum value bar region height. When users select a zoom region, any of several actions may occur with the common characteristic that it is expanded into value bar regions for its constituent items. The scale for the new value bar regions is different, but since relative comparison is the value bar paradigm the scale factor should not cause a problem.

Additional graphic features could help users understand where and which items are being mapped in a value bar. A thin line could be drawn between all value bars intersecting each where the selected item would (or does) appear, removing cognitive load users may expend toward such a determination. Another graphic feature could be to allow users to mark uniquely items' value bar regions so that as selections occur users can stay focused on those particular marked items' value bar regions.

It may not always be desirable to scroll the text window to find out about an item the interest in which was triggered by its value bar region height. Other styles of information presentation could be used to display information about an item. Immediate navigation is not a necessary component of interacting with value bars, just a convenient and powerful component in many situations.

Color could be introduced as an additional coding factor for value bar regions. An item can have attributes displayed within value bar regions that are not naturally quantified. Such attributes could be type (e.g. file type for a directory listing), ownership, etc.

Control panels can be used for interactive control of value bars. Control panel functionality could include the ability to choose which attributes are mapped; which function to use to map item values to region heights (linear, logarithmic, etc.); the exclusive choice among:

- percentage of items given regions (e.g. top 20%),
- absolute number of items (approximately) to be given regions (e.g. 40), or
- item values given regions (e.g. items with size > 50);

switching to different input data; and there are surely more.

Value bars could be applied to many domains if there existed an application programmer's interface. Options that could be parametrized include vertical or horizontal orientation, size dimensions, graphic characteristics for selection, etc. More importantly, there must be facilities for linking value bar regions with the actual data items and specifying when and how interaction events invoke value bar updates.

CONCLUSION

The value bar represents a unique combination of information visualization and navigation for multi-attribute listings and tables. The information visualization is powerful providing items' local detail in a global context for attribute values and their distributions. Navigation and interaction are simple, clean, and builds on generic GUI concepts. Value bars can be applied in many diverse domains that have data with characteristics similar to multi-attribute listings or tables. Value bars are a useful tool which can be integrated unobtrusively into existing application environments with ease.

ACKNOWLEDGEMENTS

This research was supported by Sun Microsystems, Inc. I'd also like to thank the ongoing contributions by my colleagues at the Human-Computer Interaction Laboratory at the University of Maryland, especially Ben Shneiderman and Brian Johnson. Discussions among us three about visualizing large hierarchical information spaces using two-dimensional graphic displays ([6], [9]) laid the groundwork for research on value bars.

REFERENCES

1. Beard, D., and Walker, J. Navigational Techniques to Improve the Display of Large Two-dimensional Spaces. Behaviour & Information Technology 9, 6 (June 1990), 451-466.

2. Card, S., Robertson, G., and Mackinlay, J. The Information Visualizer, an Information Workspace. In Proceedings ACM CHI'91 Human Factors in Computing Systems Conference (New Orleans, LA, April 27 - May 2). ACM, New York, 1991, pp. 181-188.

3. Chimera, R., Wolman, K., Mark, S., and Shneiderman, B. Evaluation of Three Interfaces for Browsing Hierarchical Tables of Contents. University of Maryland Human-Computer Interaction Laboratory technical report CAR-TR-539, CS-TR-2620.

4. Egan, D., Remde, J., Gomez, L., Landauer, T., Eberhardt, J., and Lochbaum, C. Formative Design Evaluation of SuperBook. ACM Transactions on Information Systems, 7, 1 (January 1989), 30-57.

5. Furnas, G. Generalized Fisheye Views. In Proceedings ACM CHI'86 Human Factors in Computing Systems Conference (Boston, MA, April 13 - 17). ACM, New York, pp. 16-23.

6. Johnson, B., and Shneiderman, B. Tree-Maps: A Space-filling Approach to the Visualization of Hierarchical Information Structures. To appear in Proceedings of ACM Visualization '91 Conference (San Diego, CA, October 22 - 25). ACM, New York.

7. Mackinlay, J., Robertson, G., and Card, S. The Perspective Wall: Detail and Context Smoothly Intergrated In Proceedings ACM CHI'91 Human Factors in Computing Systems Conference (New Orleans, LA, April 27 - May 2). ACM, New York, 1991, pp. 173-179.

8. Robertson, G., Mackinlay, J., and Card, S. Cone Trees: Animated 3D Visualizations of Hierarchical Information. In Proceedings ACM CHI'91 Human Factors in Computing Systems Conference (New Orleans, LA, April 27 - May 2). ACM, New York, 1991, pp. 189-194.

9. Shneiderman, B. Tree Visualization with Tree-maps: A 2-d Space-filling Approach. To appear in ACM Transactions on Graphics.

10. Spence, R., and Apperley, M. Database Navigation: an Office Environment for the Professional. Behaviour & Information Technology, 1, 1 (January 1982), 43-54.