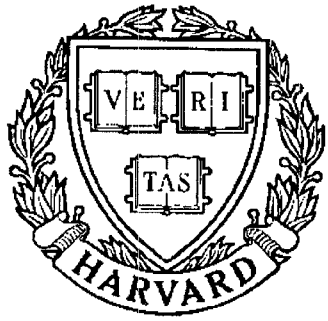
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Improving the Visualization of Hierarchies with Treemaps: Design Issues and Experimentation

by D. Turo and B. Johnson

TR 92-62

SRC-TR-92-62
CAR-TR-626
CS-TR-2901

May 1992

Improving the Visualization of Hierarchies with Treemaps: Design Issues and Experimentation

David Turo
turo@cs.umd.edu

Brian Johnson
brianj@cs.umd.edu

Department of Computer Science &
Human-Computer Interaction Laboratory
University of Maryland, College Park, MD 20742

Abstract

Controlled experiments with novice treemap users and real data highlight the strengths of treemaps and provide direction for improvement. Issues discussed include experimental results, layout algorithms, nesting offsets, labeling, animation and small multiple displays. Treemaps prove to be a potent tool for hierarchy display. The principles discussed are applicable to many information visualization situations.

For single copies
please write to:

Teresa Casey
Human-Computer
Interaction Laboratory
A.V. Williams Building
University of Maryland
College Park MD 20742

1 Introduction

Treemaps are a novel method for presenting large hierarchical information spaces on planar display areas of limited size [John91, Shne92]. A treemap is generated by recursively slicing the screen into rectangular bounding boxes to convey global structure (hierarchy); within each bounding box, individual node information is presented through display attributes such as size and color. Two families of treemap algorithms have been developed for tiling 2-D planes: *slice-and-dice*, which alternates between vertical and horizontal screen slices (Figure 2), and *top-down*, which slices in only one dimension, either horizontal or vertical (Figure 1). Treemaps combine features of multivariate coding and display layout to present hierarchies in a richly visual environment which fosters *relative* comparison of structures in the hierarchy.

Two experiments, one involving employees of GEnie, a consumer information service run by GE Information Services, and one with UNIX users, drive the discussion presented herein. Section 2 of this paper discusses the two primary treemap layout algorithms. Section 3 describes improvements to the slice-and-dice algorithm to further convey information about the hierarchy. Section 4 offers a sampling of current research directions. Section 5 describes the experiments conducted with treemaps. Section 6 is the conclusion.

2 Partitioning Algorithms

The partitioning of the rectangular screen region into a treemap can take one of two approaches: the slice-and-dice approach as previously described in [John91] and the top-down approach originally proposed by Shneiderman [Shne91], which is discussed below.

2.1 Top-Down

Development of the top-down algorithm was motivated by the desire to preserve the structure (and user familiarity) of traditional tree diagrams, which flow from one side of the screen to the other (usually from top to bottom). The algorithm slices up the rectangular region along one dimension and flows from the root (on one side of the screen) to the leaves (on the opposite side). The algorithm relies upon each node in the tree having a pre-determined *weight*, dependent upon a domain-specific attribute. An overview of the basic algorithm is given below using a top to bottom flow.

TOP-DOWN ALGORITHM

1. *The bounding box of the root node is the entire treemap display area. Make the root node the "current node".*
2. *Divide the bounding box of the current node proportionally along its vertical axis using its weight compared with the sum of the weights of its children. This produces an upper region for the current node and a lower region for its children .*
3. *Partition the lower region along the horizontal axis among the children based on their weights, creating a bounding box for each child.*
4. *Iteratively make each child the "current node" and goto Step 2.*

Essentially, the horizontal axis is recursively sliced up and divided among the children. All of the leaf nodes of the tree will eventually "touch" the bottom of the display. Figure 1 illustrates a traditional tree structure overlaid on its top-down representation. Offsets are used to emphasize the hierarchy structure.

The area in each bounding box is determined by the weight attribute. For example, if the hierarchy in Figure 1 was an organization chart with size representing salary and color representing years of service (the lighter grays representing the most years), large light gray boxes would indicate long-term employees who are well paid.

This concept of emphasizing importance through size is very similar to the fisheye concept [Furn86], though there are multiple points of interest in treemaps.

As the top-down approach to tiling planar areas limits recursive subdivision to one dimension, acceptable results are produced only for hierarchies of limited size. HCIL testing indicates hierarchies of around 100 to 200 nodes overwhelm the top-down algorithm on typical displays with 640 x 480 resolution. As an example, the GEnie hierarchy used in the treemap experiment contained 120 nodes (products) in two-levels (product manager and product type). This hierarchy could not be displayed using the general top-down algorithm due to limited horizontal resolution; a modified top-down approach solved this by partitioning the *vertical* axis at the final level. This modified algorithm (displayed in Figure 6) works well, but it is not generalizable to hierarchies that are not of a uniform, fixed depth.

The main benefit of the top-down design, therefore, is its ability to conform to traditional tree diagram conventions. With small hierarchies, traditional tree diagrams may be used in conjunction with top-down treemaps as in Figure 1, which fosters comparative analysis while preserving traditional diagrammatic notation.

2.2 Slice-and-Dice

The slice-and-dice algorithm avoids the problems of the top-down algorithm by recursively partitioning the planar display area along both dimensions. Much larger hierarchies, greater than 1,000 nodes, can thus be displayed, as is clearly seen here in Figure 9 and in [John91]. The slice-and-dice algorithm is presented and discussed in detail in [John91].

A slice-and-dice treemap presents the hierarchy as a series of recursively-drawn bounding boxes, sliced alternatively

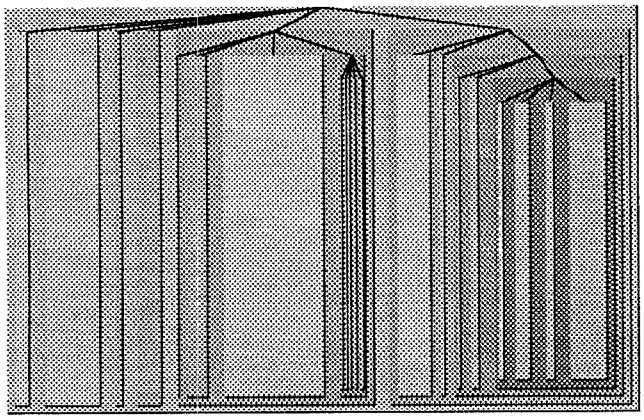


Figure 1: Top-Down, Size by Weight

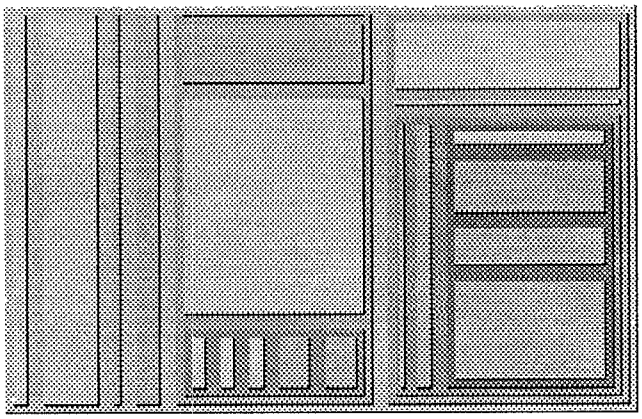


Figure 2: Slice-and-Dice, Size by Weight

vertically and horizontally. Figure 2 displays the same hierarchy as in Figure 1, only drawn using the slice-and-dice algorithm.

Because of the quantity of information that can be presented, large treemap diagrams should be thought of as powerful visualization tools requiring a degree of familiarity. Since the slice-and-dice treemap proves to be the most viable for display of large hierarchies, all further discussion of treemaps will assume partitioning along both dimensions.

2.3 Treemap Display Limitations

All static hierarchy presentations have limits as to the quantity of information they are capable of presenting on a finite screen display. When these limits are reached, navigational techniques such as scrolling or panning must be used, creating the potential for loss of context [Bear90]. Common character-based applications use a set number of lines to display the hierarchy. Graphical tree diagrams have more leeway: depending upon the drawing algorithm and the size of the display space, a hundred or so nodes can be adequately

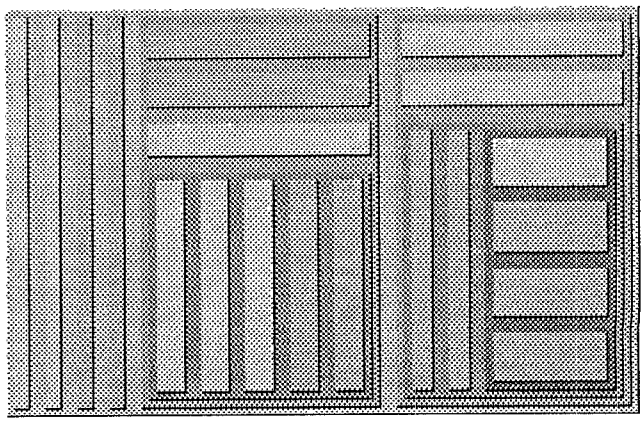


Figure 3: Slice-and-Dice, Size by Unit

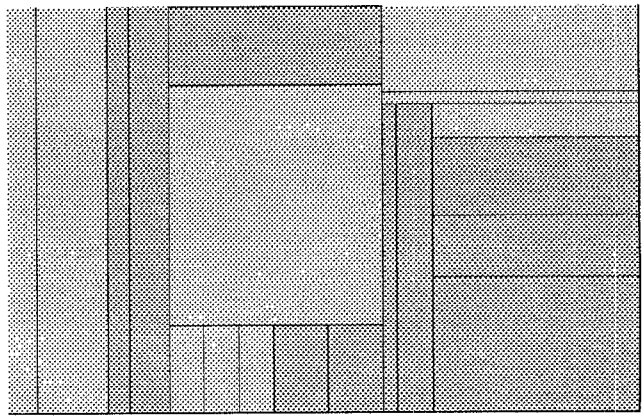


Figure 4: Slice-and-Dice, no offsets

represented on screen without the need for panning or zooming techniques.

More advanced graphical diagrams such as cone trees [Robe91] increase the display limit through the use of a virtual third dimension at the expense of increased navigation (in this case, rotation).

The number of nodes that can be displayed by a treemap can be an order of magnitude greater than traditional graphical tree diagrams. This is the result of the tiling approach which packs the display space. Treemaps, though, have limits as well; as with previous presentation methods, zooming, panning, and animation can extend these limits.

Table 1 indicates display limits for binary trees using the above approaches with non-overlapping nodes. The formulas for graphical trees assume no horizontal separation space for nodes on the leaf level and also assume enough vertical space to display all tree levels. The treemap figures assume that all leaf node weights are equal, which will generate square bounding boxes for this example. The italicized entries in Table 1 are smaller than the practical minimum node size; they are included for completeness.

Node Size	256	128	64	32	16	8	*4*	*2*	*1*	(squares, pixels/side)
Horizontal Resolution in Nodes	2	4	8	16	32	64	*128*	*256*	*512*	
Vertical Resolution in Nodes	2	4	8	16	32	64	*128*	*256*	*512*	
Tree Diagram #Levels	2	3	4	5	6	7	*8*	*9*	*10*	log2(#leaves)
Tree Diagram #Leaves	2	4	8	16	32	64	*128*	*256*	*512*	DW/NW
Tree Diagram #Nodes	3	7	15	31	63	127	*255*	*511*	*1,023*	2*DW/NW-1
Treemap #Levels	3	5	7	9	11	13	*15*	*17*	*19*	log2(#leaves)
Treemap #Leaves	4	16	64	256	1,024	4,096	*16,384*	*65,536*	*262,144*	DW/NW*DH/NH
Treemap #Nodes	7	31	127	511	2,047	8,191	*32,767*	*131,071*	*524,287*	2*DW/NW*DH/NH-1
512 Pixel by 512 Pixel Display Size 262,144 Total Pixels					DW: Display Width DH: Display Height			NW: Node Width NH: Node Height		

Table 1: Binary Tree Display Resolution

It can be seen that a treemaps (without offsets) are capable of displaying an order of magnitude more information. It should be noted that without offsets only the leaf nodes appear in the display. Figure 4 illustrates the same tree as Figure 2 with offsets removed. The maximum size of representable hierarchies decreases as offset size increases.

2.4 Degenerate Cases

Treemaps can display the largest hierarchies when the aspect ratios of the bounding boxes are approximately one. When this condition does not hold, information may begin to "drop" out of the display.

Rather than having nodes with small weights or extreme aspect ratios disappear from the display, it is possible to set minimum node dimensions. With this approach nodes whose display size would normally fall below the resolution of the display medium are assigned some small constant extent (width or height) at the expense of borrowing display space from sibling nodes.

Although this approach prevents nodes from dropping out in many cases, it has its own limitations. When the number of such nodes to be partitioned along a given axis exceeds the resolution of the display along that axis, information will still disappear. Regions where this occurs can be indicated by a special color and zooming facilities provided. A typical region with "drop outs" is illustrated in Figure 5; black areas indicate clusters of nodes that are not displayed.

Since the display size (bounding box) of a node is determined by its weight, nodes typically drop out of the display in order of their weights. This "graceful degradation" preserves relatively important nodes while indicating where collections of relatively less important nodes are located.

3 Conveying Information

User disorientation is a problem when presenting large bodies of information, especially when users are confronted with unconventional display methods like treemaps. The

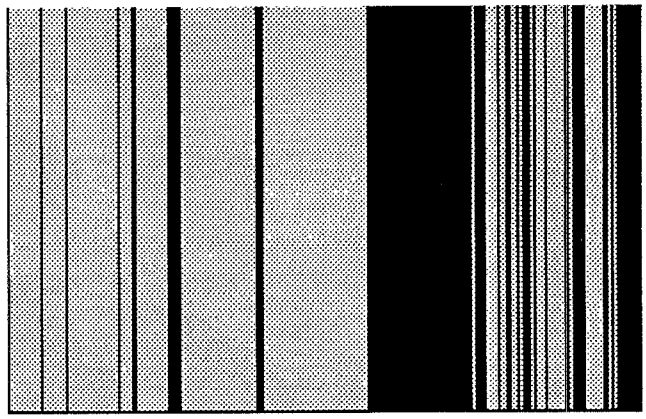


Figure 5: Node "Drop-Outs"

GEnie and Unix experiments along with general use of the treemap as a Macintosh directory manager called TreeViz (developed in the HCIL) have highlighted a number of usability issues and subsequent refinements. A few of the more important issues and refinements are:

- nesting offsets
- user control of attribute mapping, node filtering, and sibling node sort order
- animation
- small multiple display
- zooming
- textual "signposts"

3.1 Nesting Offsets

Treemaps convey structure via containment (nesting and grouping) in the same fashion as Venn diagrams [John91, Trev89]. Nesting offsets give users control over the allocation of display space between internal and leaf nodes. Larger offsets put greater emphasis on internal nodes and hence the structure of the hierarchy; smaller offsets emphasize leaf nodes. Without offsets only leaf nodes are directly visible; the internal structure of the hierarchy must be inferred from text labels and the grouping of leaf nodes.

Users viewing new hierarchies often need offsets in order to become familiar with the global structure of the hierarchy. After a short period of use users generally prefer smaller offsets (0, 2, or 4 pixels), as noted in the two experiments and TreeViz. Small offsets provide a degree of global context while still maximizing the display space available for the display of leaf nodes.

3.2 Attribute Mapping

User confidence in the treemap application may be improved by providing the user more control over attributes which determine node size (weight) and color [Hene90]. Figure 3, for example, displays the same treemap as in Figure 2, except that all leaf nodes have the same weight (the areas of the leaf nodes' bounding boxes are all the same).

Modifying these attributes through functions allows users to emphasize features in the data of varying importance. Inversion of attributes is a simple function that flips the order of importance: instead of salespeople with the largest profits having the largest bounding boxes, salespeople with the *smallest* profits would have the largest bounding boxes. Logarithmic or power functions have proven useful for removing large discrepancies in the areas of the boxes, providing a more "balanced" view of the hierarchy.

Domain attributes may also be mapped to color. Quantifiable attributes (placed on a numeric scale) worked well with different luminosity levels of the same hue (maintaining constant saturation) in the GEnie experiment to represent profitability levels.

If a non-quantifiable attribute is to be displayed, the approach of assigning distinct hues to each attribute is effective. The Macintosh implementation of treemaps utilizes evenly separated hues while maintaining constant saturation and luminosity to convey different file types.

User control over the color is of primary concern as color preference varies by task and individual. Aesthetically pleasing color schemes should be preconfigured for the user and accompanied by a color key or chart for user reference [Cox90]. One area of color control addresses the problem of color deficiencies and monochrome monitors: providing a transformation to a gray scale or patterns alleviates the problem. Gray scale diagrams [Feen91] show that even when distinct hues are eliminated, information can be conveyed via gray scale.

3.3 Node Filtering

Filtering nodes allows users to concentrate on features of interest. In hierarchies users may wish to see only those nodes satisfying certain properties. Examples include: internal nodes, leaf nodes, specified branches of the hierarchy, nodes of certain depths or nodes with a particular attribute (for example, all text files).

Two functions are useful for initial orientation: expanding *a level at a time* allows the user to gradually step down the hierarchy and view each level before proceeding further; expanding *particular nodes* allows users to view the detail of node(s) in the context of the complete hierarchy.

3.4 Sibling Node Sort Order

The order in which sibling nodes are displayed within a parent can be used to further orient the user or provide additional information about these nodes (node type, rank, alphabetic order, etc.).

A concern specific to grouping sibling nodes in treemaps is that of display size. A very thin node between much larger nodes tends to become lost As it is commonly the case that among siblings, leaf nodes are far smaller than internal nodes (which contain other leaf nodes), it is thus often useful to group internal and leaf nodes separately. It is also advantageous to group leaf nodes together as a nested block (when offsets have been specified) instead of nesting each individual leaf node. This nested block saves display space and also provides further distinction between leaf and internal nodes.

3.5 Animation: Relative vs. Absolute

Treemap animation was used in the GEnie experiment with positive results: growing and shrinking bounding boxes reflected changes in the underlying data and conveyed these changes in a powerful way. Six months worth of GEnie product data were available, displayed one month at a time. Users were able to step the diagram through the months with a slider positioned beneath the treemap.

Animating a *full* treemap (one that fills up the entire display area) over a time sequence presents *relative changes* to the user. For example, suppose a hierarchy consisted of only two bounding boxes, each taking up 1/2 of the screen. If during the next animation sequence, the first box grows to occupy 3/4 of the screen, the diagram should be interpreted to mean "Relatively, the first node's weight is three times the value of the second node's weight."

From an absolute perspective, though, the weight of the two nodes may have changed in a number of ways: increase of the first node's weight with a smaller increase, no change, or a decrease in the second's; no change in the first node's weight with a decrease in the second's; or a decrease in the first node's weight and a larger decrease in the second's.

Treemaps can, however, be utilized for animation of absolute data if local maximums are known for the attribute which determines the weight. For example, in an application which displays daily stock market information with stock value as the size attribute, the day with the total highest stock values would be reflected in a treemap which occupies the full display area. The other days, with lower total values, would use treemaps which occupy a percentage of the total display

area based upon their value divided by the maximum value. Using this method, the entire treemap would grow and shrink, and the area of each bounding box would reflect absolute data over the time period being examined.

3.6 Small Multiples

Treemaps promote relative comparisons and are particularly suited to the presentation of small multiple views or animation when relative comparisons are desired.

The GEnie experiment used this technique to display six months' worth of data side by side. Users were asked to interpret changes by following particular squares across the diagram and noting changes in size, position, and color. Figure 6 is a small multiples view of the six months in the GEnie experiment using a subset of the product data. (The small multiple technique also applies to the concept of multivariate data display, discussed in Section 4.4.)

As the products are always displayed vertically in the same order, a single product may be followed across the months to gauge performance. In this case the first product's revenue (size attribute) is expanding relative to the other products in the product category; the darkening gray indicates the profit of the product is changing. The widths of the sections indicate overall growth in this 6 month period.

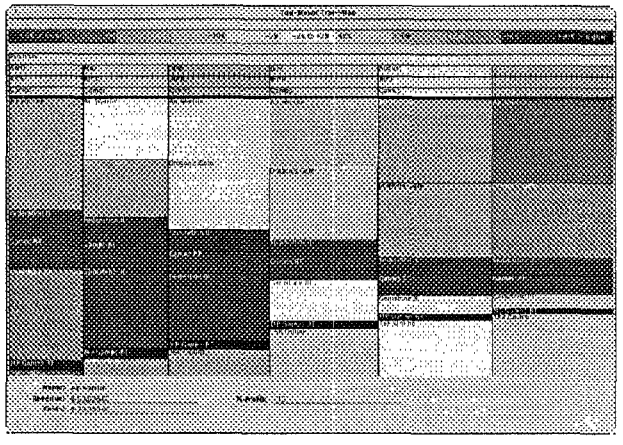


Figure 6: Top-Down with Small Multiples

3.7 Zooming

Zooming allows the promotion of any node to full display size (the zoomed node becoming the new root of the displayed hierarchy), providing space for the presentation of small cluttered regions. Navigational tools are a double-edged sword, for while they allow users to hone in on regions of interest, they also cut off users from previous contextual features.

Zooming was successfully implemented in the GEnie experiment. Users had the option of zooming in on any node in the display, thus allowing quick navigation through the hierarchy. Options to zoom back one level and zoom back to the original root were also provided.

Care should be taken to avoid disorienting the user. The zoom effect, therefore, should incorporate some traditional visual cues (such as zoom lines or increasing or diminishing rectangle outlines) to identify what exactly is being zoomed in or away from.

3.8 Textual Signposts

Text labels can be used to further orient the user to the hierarchy if space is available. Small nodes will not be able to include a textual string due to space limitations; however, nodes that can provide textual signposts are useful as landmarks in a sea of boxes. Figure 6 gives an indication of how text labels may be utilized in a treemap application. Each node in the diagram has a text label in the upper left corner.

4 Current Research

Current research directions have been greatly influenced by the usability studies and experiments that have been conducted thus far. We have found users to be generally receptive to the idea of treemaps but wishing to use them to display more familiar hierarchies. Tools for visualization are no more interesting than the data they present. Can treemap users mine the wealth in rich hierarchical data sets? We think there is great potential.

4.1 Queries

A capability that would allow users to specify queries and have the results shown by highlighting or blinking matching bounding boxes is the most prevalent request. Issues here relate to appropriate highlighting mechanisms and feedback to users. An offshoot of this involves implementation of dynamic queries [Will92, Buja91], which allow users to generate a large number of queries in a short period of time via direct manipulation with sliders or other widgets.

4.2 Aspect Ratio Perception Problem

Treemaps use a single numeric weight to determine the display area of a node in the hierarchy. Perceptual difficulties arise when area comparisons are made between nodes of differing heights and widths, as users cannot accurately gauge fine area differences between rectangles differing in both dimensions. Figure 3 illustrates this problem — all of the leaf nodes have the same weight (area), but their heights and widths differ.

Two-D representations are poor for comparing linear values that are similar, but they can show greater ranges, a benefit in the case of file sizes which can range over six orders of magnitude. Users may use display area to rapidly hone in on nodes of interest, which can then be compared in detail via mouse tracking and dialog boxes.

4.3 Dynamic Algorithms

In a dynamic environment it is useful to isolate global recomputation from local perturbations such as node insertion, deletion, or size changes. Treemaps allocate space in a relative manner, and as such are inherently susceptible to global recomputation. Current algorithmic research is concentrating on minimizing recomputation in a dynamic environment.

4.4 Multivariate Comparison

Treemaps have potential as a multivariate exploratory data analysis tool. Hierarchies can be created based on the degree of interest in a set of categorical variables [Miha91]. The display space is partitioned amongst the categorical levels of each variable relative to their proportionate values. Treemaps can be generated either singly or as a series of small multiples. Figure 7 shows only a small multiples presentation of subject performance in the treemap vs. UNIX directory browsing experiment.

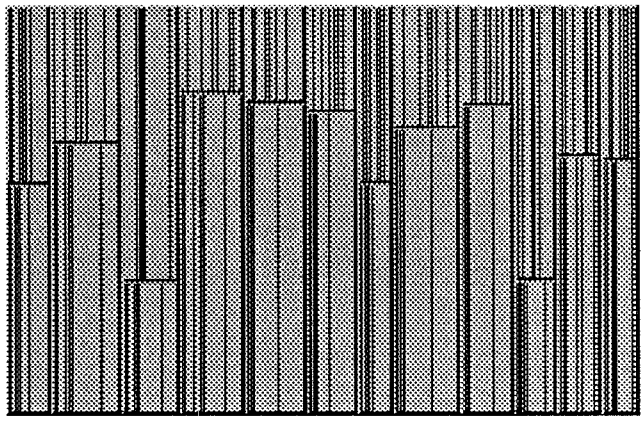


Figure 7: Treemap UNIX Experiment Results

The 12 largest vertical boxes represent the total time required by each subject to answer all questions. The two largest boxes within each subject represent the treemap questions on the top and the UNIX questions on the bottom; within each interface condition are the 7 individual questions. It is readily apparent that the global questions took substantially more time to answer and that, in general, subjects performed faster using the treemap interface. The same information is also presented in Table 2.

4.5 2 1/2-D Treemaps

A "third" dimension can be added by using the 2-D rectangular area as the base of a 3-D solid. Increasing the visual vocabulary can provide for richer information resolution [Ding90]. Simply extruding the rectangles produces a Manhattan-like scene, where rectangular solids obscure one another. Using a single point in the third dimension creates pyramids, which do not obscure each other as much. The height dimension can code one more variable, and the location of the apex, apex skew, and the four sides can code additional properties in a manner similar to datajacks [Cox90, Ells90]. Leaf nodes become pyramids and internal nodes become flat top "plateau" pyramids. Figure 8 below illustrates 2 1/2-D treemaps. True 3-D treemaps would be volumes partitioned on all 3 dimensions.

Free movement of the perspective point can provide natural zooming and perspective. More interesting nodes, defined by nearness to the perspective point, receive more display space, a natural fisheye view. As is always the case with treemaps, nodes with greater weights (more interesting data points) also receive greater display space, as their base dimensions are greater.

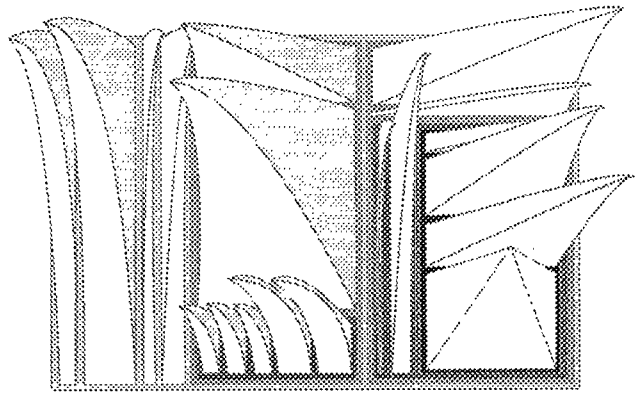


Figure 8: 2 1/2-D Treemaps

5 Experimentation

The utility of treemaps was tested during the past year in two experiments. Both experiments were designed to reflect "real-world" situations and needs.

5.1 GEnie Experiment

An experiment was conducted using a top-down treemap to display financial data for the product hierarchy of GE Information Services' consumer information service, the General Electric Network for Information Exchange (GEnie). Six months of GEnie data was used — for each product, revenue and profit figures were noted.

The experiment utilized a between-group design with 18 GEnie employees and tested two versions of the treemap against hard-copy financial reports currently in use at GEnie. The treemap versions included an "animated" version, in which the data was stepped through one month at a time via a slider, and a "small multiples" version, in which all six months were placed side-by-side. Size, color and animation were each tested for via twelve questions on revenue, profits

		Treemap Interface								Unix Interface							
Subject	Group	1	2	3	4	5	6	7	Total	1	2	3	4	5	6	7	Tota
1	TA	21	15	61	21	19	38	114	289	13	12	48	21	16	95	167	371
5	TA	13	57	55	25	32	93	62	337	23	33	79	48	24	300	169	676
9	TA	160	23	32	15	12	277	67	586	15	14	35	9	17	117	71	277
3	TB	23	47	37	16	40	12	24	199	15	27	120	94	27	265	197	745
7	TB	26	12	36	16	25	37	60	211	7	7	41	13	20	300	300	688
11	TB	26	34	32	20	16	22	42	191	14	9	50	10	17	152	300	552
2	UA	42	16	29	55	16	29	41	227	9	15	43	14	11	39	168	298
6	UA	48	73	37	18	21	50	40	288	9	22	34	27	14	300	275	680
10	UA	44	13	41	18	23	21	29	41	19	18	20	17	10	300	215	600
4	UB	42	54	92	13	11	61	148	420	6	14	15	14	9	102	42	202
8	UB	30	16	52	64	10	26	40	235	35	24	28	109	51	117	48	412
12	UB	46	16	50	23	25	25	22	206	26	20	57	14	25	171	31	345
	Ave	43	31	46	25	21	58	57	269	16	18	48	32	20	188	165	487
	StDev	38	21	18	16	9	73	38	136	9	8	29	34	11	98	99	190
	Median	36	19	39	19	20	33	42	231	14	17	42	16	17	162	168	482
	Min	13	12	29	13	10	12	22	41	6	7	15	9	9	39	31	202
	Max	160	73	92	64	40	277	148	586	35	33	120	109	51	300	300	745

Table 2: UNIX Experiment Results

and data trends. Users were timed for each question with a maximum limit of five minutes per question. The results were analyzed at the p = 0.05 level using a 1-way ANOVA.

Favorable statistical differences were achieved by the treemap groups in tasks requiring users to identify global trends in revenue and profitability across the six months. Size allowed the users to concentrate on products of interest (products that generated the most revenue) [Furn86] and quickly answer questions related to high revenue; a hindrance to this was the aspect ratio problem mentioned earlier where products of similar revenue had to be compared manually.

Color was a potent tool for analyzing profit changes, and the low-saturated blues and reds (representing positive and negative profit levels, respectively) showed up on the screen in sharp contrast. Users found these colors satisfactory, but several seemed to be overwhelmed by the blend at times. Animation and zooming were used to a great extent, and users subjectively gave these features high marks; the problems mentioned above with relative animation surfaced here, however, and contributed to some high response times for treemaps.

Subjectively, there was a strong preference for the animated treemaps in terms of screen layout, information conveyed and capabilities. We did discover a tendency to not "trust" the diagram in the initial questions, given users' lack of experience with treemaps (even with training) and the aspect ratio problem mentioned above. This caused the treemap users' times to be skewed upward in those questions. The tendency disappeared in the latter half of the experiment.

5.2 Directory Browsing Experiment (treemaps vs. UNIX)

A within-subject counterbalanced experiment with twelve subjects was used to compare treemaps with UNIX (tcsh shell) for directory browsing tasks. A directory hierarchy with approximately 500 files and 50 directories was used. Subjects answered 7 questions with each interface. Figure 7 presents a relative view of users performance and Table 2 presents a tabular view user performance measured in seconds per question.

The first 5 questions were local in scope, dealing with particular files or directories. All local questions were correctly answered within the allotted time (5 minutes per question). On the local questions, statistically significant performance time differences were found for the first two questions, which favored UNIX. Since subjects all had at least one year of UNIX experience and no previous experience with treemaps other than the 15 minute training period, it is possible that this effect may have been largely due to learning effects. Subjects performed comparably using either interface on the remaining questions that were of local scope.

Since treemaps present the entire hierarchy at once it was hypothesized that treemaps would be faster for questions that are global in scope. Global questions dealt with portions of the hierarchy larger that single directories. Error rate analysis has been used instead of performance rate analysis on the global questions as a significant proportion of the subjects were unable to answer these questions correctly within the allotted time. Five subjects were unable to complete either one or both of the global questions correctly. A total of six errors were made as one subject could answer neither of the questions. All of the errors were made by subjects using UNIX. UNIX users made statistically significantly more errors at the p=0.05. All users successfully completed the global questions using the treemap interface, demonstrating the effectiveness of treemaps for global comparisons.

A few subjects remained after the experiment to use the treemap technique to visualize their own personal UNIX

directories. Visualizing their own information from a new perspective was both interesting and exciting.

Treemaps can significantly aid such tasks as locating large old files or clusters of files with similar attributes. Treemaps proved to be an effective visualization tool for file hierarchies.

6 Conclusion

Treemaps represent a unique approach to effectively communicating information about large hierarchies, which contain information that users may previously have gleaned over a long period of overlooked entirely. Providing users with the capability to display the entire hierarchy allows information to be drawn from it that may not have been obvious via traditional means. Applications include file hierarchies, organizational charts, medical clinical trials, sales figures, stock portfolio analysis, budget allocations — there are many possibilities. As information spaces grow in size, the need for treemaps and other similar data visualization tools will only increase.

Acknowledgments

Supported in part by the University of Maryland Systems Research Center under NSF Grant CDR-88-03012.

We would like to acknowledge the support of the members of the Human-Computer Interaction Lab, whose suggestions and criticisms have been greatly appreciated.

The TreeViz application, developed at the HCIL for the Macintosh, is distributed by the University of Maryland's Office of Technology Liaison, College Park, MD, 20742.

References

[Bear90] David V. Beard and John Q. Walker II. Navigational techniques to improve the display of large two-dimensional spaces. Behavior & Information Technology, 9(6):451-466, 1990.

[Buja91] Abdreas Buja, John Alan McDonald, John Michalak, and Werner Stuetzle. Interactive Data Visualization using Focusing and Linking. In Proceedings of IEEE Visualization '91 Conference, pages 156-163, 1991.

[Cox90] Donna J. Cox. The art of scientific visualization. Academic Computing, page 20, March 1990.

[Ding90] Chen Ding and Prabhaker Mateti. A framework for the automated drawing of data structure diagrams. IEEE Transactions on Software Engineering, 16(5):543-557, May 1990.

[Ells90] Richard Ellson. Visualization at work. Academic Computing, page 26, March 1990.

[Feen91] William R. Feeney. Gray Scale Diagrams as Business Charts. In Proceedings of IEEE Visualization '91 Conference, pages 140-147, 1991.

[Furn86] George W. Furnas. Generalized fisheye views. In Proceedings of ACM CHI'86 Conference on Human Factors in Computing Systems, Visualizing Complex Information Spaces, pages 16-23. 1986.

[Hene90] Tyson R. Henry and Scott E. Hudson. Viewing large graphs. Technical Report 90-13, University of Arizona, May 1990.

[John91] Brian Johnson and Ben Shneiderman. Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures. In Proceedings of IEEE Visualization '91 Conference, pages 284-291, 1991.

[Miha91] T. Mihalisin, J. Timlin, and J. Schwegler. Visualization and Analysis of Multi-variate Data: A Technique for All Fields. In Proceedings of IEEE Visualization '91 Conference, pages 171-178, 1991.

[Robe91] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees: Animated 3d visualizations of hierarchical information. In Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems, Information Visualization, pages 189-194. 1991.

[Shne91] Ben Shneiderman. Personal Communication, 1991.

[Shne92] Ben Shneiderman. Tree visualization with treemaps: A 2-D space-filling approach. ACM Transactions on Graphics, January 1992.

[Trav89] Michael Travers. A visual representation for knowledge structures. In ACM Hypertext'89 Proceedings, Implementations and Interfaces, pages 147-158. 1989.

[Tuft83] E. R. Tufte. The Visual Display of Quantitative Information. Graphics Press, Cheshire, CT, 1983.

[Will92] Christopher Williamson and Ben Shneiderman. The Dynamic Homefinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System. In proc. of ACM SIGAR '92. ACM press, 1992.

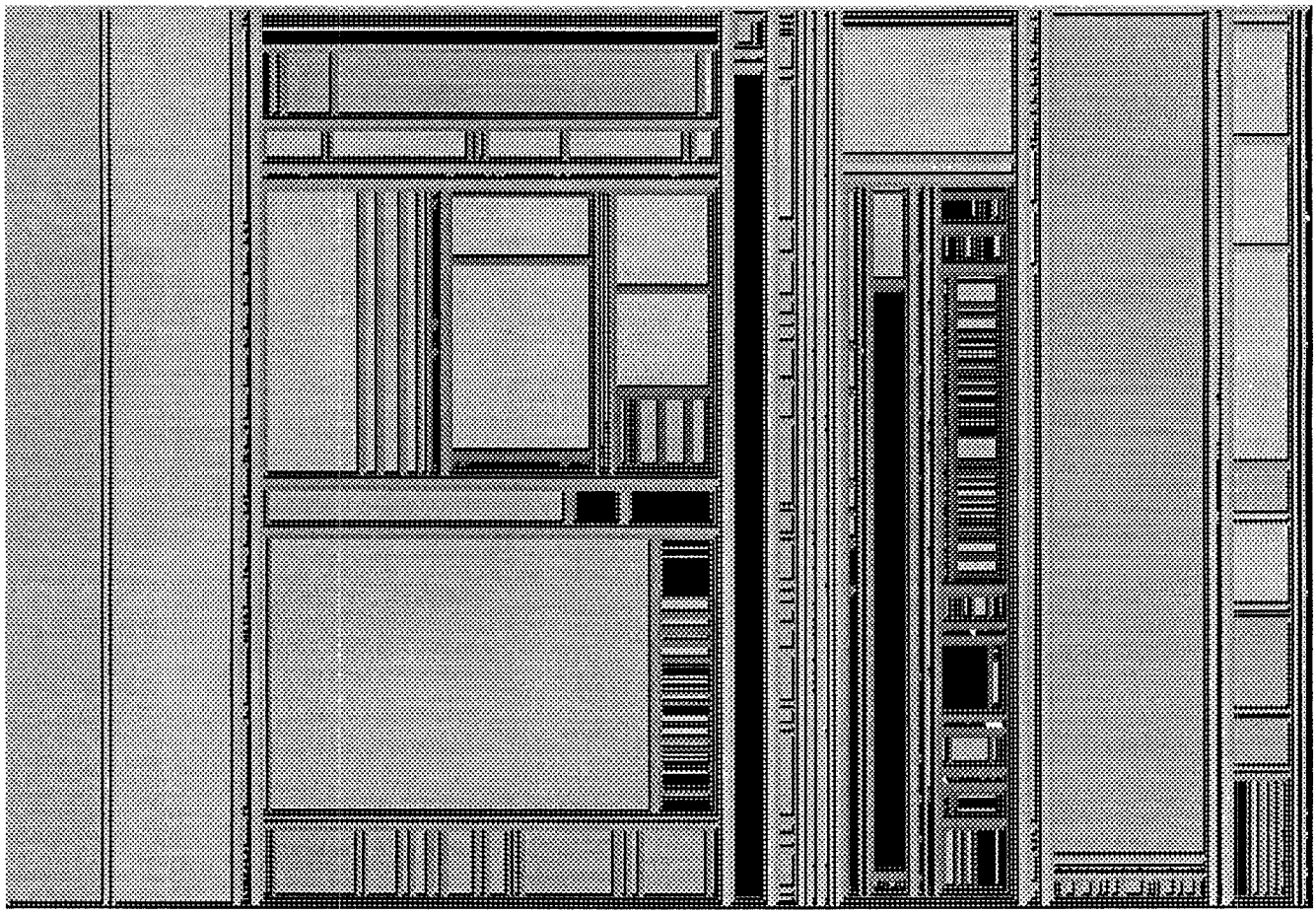


Figure 9: Slice-and-Dice Treemap
This Macintosh file hierarchy contains1000 files and 100 directories with a 2 pixel offset.
Black "drop-out" areas contain collections of many small programming source files.