Interactive Information Visualization of a Million Items

Jean-Daniel Fekete Catherine Plaisant
Human Computer Interaction Laboratory
University of Maryland
http://www.cs.umd.edu/hcil
fekete@lri.fr plaisant@cs.umd.edu

Abstract

Existing information visualization techniques are usually limited to the display of a few thousand items. This article describes new interactive techniques capable of handling a million items (effectively visible and manageable on screen). We evaluate the use of hardware-based techniques available with newer graphics cards, as well as new animation techniques and non-standard graphical features such as stereovision and overlap count.

These techniques have been applied to two popular information visualizations: treemaps and scatter plot diagrams; but are generic enough to be applied to other 2D representations as well.

1. Introduction

Information visualization is a research domain aiming at supporting discovery and analysis of data through visual exploration. Made popular by Edward R. Tufte's books [22], its principle is to map the attributes of an abstract data structure to visual attributes such as Cartesian position, color and size, and to display the mapping. This is in contrast with scientific visualization, which deals with data that usually has an intrinsic representation.

Popular mappings exist for a large range of data structures such as tables, trees, graphs as well as more specialized ones. During the last decade, dozens of new visualization techniques were invented, such as treemaps [13] or ConeTrees[18]. Most techniques have a strong interaction component allowing users to rapidly explore the data [1,15]. Commercial products are now available and successfully used in a wide array of applications domains such as oil production or drug discovery [7].

Yet, little is known about the limits of information visualization techniques in term of scalability. Current systems tend to avoid the problem by relying on aggregation or sampling techniques that limit the number of visible items to about 10^4 and occasionally 10^5 at the cost of interaction. Our goal is to display and manage a million items. By *item*, we mean *atomic object displayed as a distinguishable contiguous area using one visualization technique*. For systems displaying the same object several times according to different attributes, we count each instance as a different item. Management of the items involves maintaining continuous interaction and smooth animation.

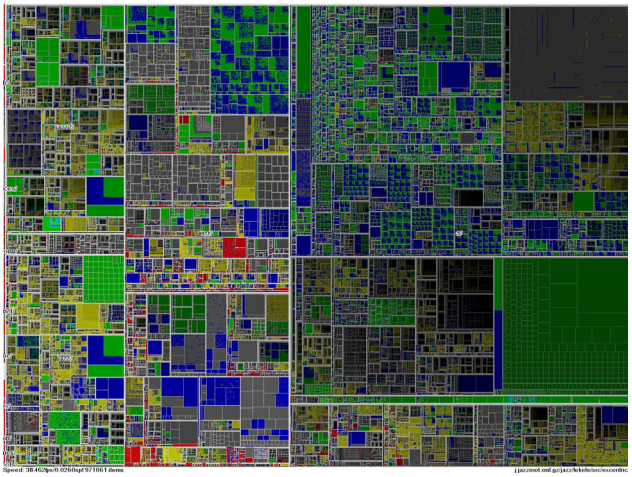


Figure 1: This treemap gives an overview of 970,000 files of a file system containing 1 million files (smaller files are smaller than one pixel and not counted), on a 1600x1200 display. The size of each rectangle is determined by the file's size; color represents file type. Deeply nested directories appear darker.

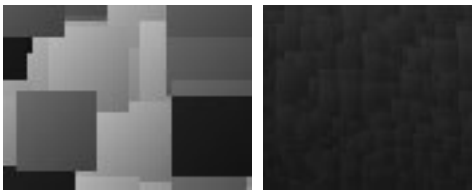


Figure 2: Smooth shaded rectangles help distinguish items in dense visualizations (details of scatter plots)

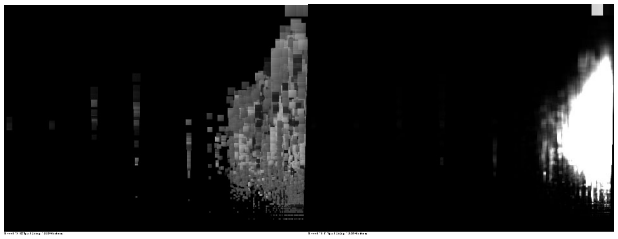


Figure 3: Scatter plot of 1 million items on the left and overlap counts on the right. The bright rectangle in the upper right corner reveal that many items are overlapping. They are aligned, and unnoticeable in the left view.

Scalability issues are important in information visualization because of the proliferation of large data sets requiring human-supervised analysis, but maximizing the number of items visible for a given display size is also a challenge for designers of visualizations on small devices like PDAs.

In this article, we describe new techniques to visualize a million items using standard displays. These techniques rely on:

- hardware acceleration to achieve the speed required for interaction and animation;
- non-standard visual attributes such as stereovision or synthetic overlap count to enhance visualization;
- animation and interaction while replaying recorded visualization configurations (views) using time multiplexing techniques to analyze the data across several views and mappings without losing context.

We have applied these techniques to treemaps (Figure 1) and scatter plots. Treemaps are representative of space filling visualizations whereas scatter plots are representative of visualizations with overlapping, but the techniques we describe here are general enough to be applied to other kinds of 2D visualizations.

2. Previous Work

Visualizing one million items is a problem of visualization, perception and interaction.

2.1 Visualization of Large Data Sets

Existing information visualization techniques are usually limited to the display of a few thousand items or avoid the problem of visualizing large number of items by using aggregation, sampling and extracting, or by not managing occlusion and overlapping.

Among the popular techniques is the use of scatter plots connected to interactive controls such as in Dynamic Queries [1]. Scatter plots visualize multidimensional data by mapping two dimensions to the X and Y coordinates and mapping other dimensions to visual attributes such as color, width, height, orientation, intensity or shape. When augmenting the number of visible data in scatter plots, overlapping cannot be avoided and is not managed at all by current systems.

Avoiding overlaps, space-filling techniques such as treemaps [13], VisDB [14] and SeeSoft [9], offer a high density of information. They use special layout algorithms to fill up all the screen pixels, thus requiring redrawing all these pixels when a visualization parameter changes. Current implementations of these systems are limited mostly by the redraw time and to a smaller extent by the time to compute the layout. VisDB handles 50,000 items; SeeSoft 50,000 lines of code and Microsoft NetScan project can render thousands of newsgroups in a treemap.

Treemaps are visualization techniques for trees. They were introduced by Shneiderman [13] and several variations have been created which improve their readability. The initial "slice and dice" algorithm is simple: it uses the entire screen to represent the tree root and its children. Each child of the root is given a horizontal or vertical strip of size proportional to the cumulative size of its descendents. The process is repeated recursively, flipping horizontal and vertical strips at each hierarchy level. Some variants try to avoid long thin rectangles and allocate strips containing sub-trees as square as possible [4].

Hybrid techniques such as Mihalisin Associates System can visualize data sets ranging from 10^4 to 10^8 data points [14] along with several dimensions and "measures"; however, they rely on sampling, limiting the number of visible items to numbers not specified by the author. Jerding et al. "Information Mural" technique [12] is a good example of system displaying hundreds of thousand items by relying on aggregation.

Current visualization systems are limited to about 10,000 items partly because control panels, labels and margins waste too many pixels, the data structures are not optimized for speed, and they use slow graphics libraries, which brings the interaction to a very slow motion when dealing with more than 10,000 items (for example Spotfire can load more than 10,000 items but the interaction suffers enormously). Addressing those three issues is necessary – but not sufficient – to handle a million items.

2.2 Perception

To be effective, visualization techniques should rely as much as possible on preattentive graphical features [11,21]. These features are processed by the low level visual system and can be recognized "at a glance" without effort. An example of preattentive processing consists in spotting red dots among several blue dots. It does not take any effort to see whether there are one or several red dots and it can be done in less than 200 milliseconds if the region is small enough to be seen in one glimpse (more experiments can be found at http://www.csc.ncsu.edu/faculty/healey/PP/PP.html). Without pre-attentive processing, spotting a feature requires time linear to the number of features and will not scale well to displays of millions of items. An example of non-preattentive feature is text reading. Finding a name in a non-sorted list requires a time proportional to the number of labels (or more when the user loses track or gives up.)

The list of visual features available to visualize abstract information is long, but only a small set can be used in a preattentive way. Healey lists them in [11] as: line (blob) orientation, length, width, size, curvature, number, terminators, intersection, closure, color (hue), intensity, flicker, direction of motion, binocular luster, stereoscopic

depth, 3D depth cues and lighting direction. Furthermore, this list only means that in some controlled configuration, these features can be processed preattentively, not that they are always processed this way. For example, Healey has conducted experiments that show that only five to seven different well-chosen colors can be processed preattentively. When trying to use more colors, the error rate and time required to search colored items increased substantially and search time become linear. In addition, combining two or more preattentive features can create interferences so in practice only two or three features can be used together.

2.3 Interactive Techniques

In 1994, Ahlberg and Shneiderman [1] defined the steps of visual information seeking as: start with an overview of the data set, zoom in on items of interest and filter out uninteresting items, then details on demands. Increasing the number of visible items permits richer overviews to be presented. They also introduced the term "dynamic queries" to describe interactive methods for interactively specifying search queries in data sets. The definition is:

- visual presentation of the query's components (with buttons and range sliders);
- visual presentation of results;
- rapid [around 100ms], incremental, and reversible control of the query;
- selection by pointing, not typing; and immediate and continuous feedback.

Coupling visualization with dynamic queries has increased the effectiveness of visualization techniques by allowing user-controlled temporal research on the visualized data. Current systems dynamically filter visualized data through sliders and buttons up to 10,000 items. Above that, the refresh rate becomes unacceptable. An optimization technique developed by Tanin [20] demonstrates dynamic queries with 100,000 items by pre-computing the visible items for each reachable position of the slider bar but this technique was limited by the redisplay time.

To show more items or dimensions, all the visualization techniques can use space multiplexing, time multiplexing, overlapping, or space deformation techniques. Space multiplexing techniques display two or more visualization configurations on the same screen, using fewer pixels for each configuration. This is impractical when attempting to visualize a million items because space is already scarce to begin with. Time multiplexing techniques show each configuration successively, either at a regular pace (animation), or by using control panels, or by following interactive methods such as dynamic queries. Each configuration should appear in less than 100ms to maintain continuous interaction. Overlapping techniques such as Magic Lenses [5] and Excentric Labels [10] show transient information over the visualization and can be used effectively on dense data. Several interactive techniques have been designed to enhance the interaction for visualization and sparing screen real estate. See-through tools [5] are interactive enhancements to Magic lenses; they filter the visualization in-place using transparency or overlapping. Extensions of Pie-Menus [6] such as Control Menus [19] can be used to overlay controls without using permanent screen real estate.

Space-deformation techniques such as [15] are sampling or aggregation techniques that try to show details in the zones of interest and only show "important features" or samples elsewhere.

Aggregations provide powerful summaries but can sometime hide phenomena only visible at finer grain. For example, a US map of mortality data at the state level will hide local outliers and even errors in the data. This article focuses on techniques that push back further the need for aggregation and sampling.

3. Technical Constraints

To display one million of items, we need to address screen resolution and speed issues.

Screen definition and resolution: Current high end screens display around 2 million pixels (1600x1200) at a resolution of 150dpi, with the newest screens capable of displaying around ten million pixels at 200dpi. So displaying 1 million of items should not be a problem if each item fits in 2 to 10 pixels in average, not counting the overlaps. Screens or video projections can be tiled to increase the number of pixels available, virtually removing any limitation, but increasing the physical size of the display requires more head movements and slows down perception. An alternative is to increase screen resolution. The limit would be human perception: theoretically around 24 million pixels, practically around 10 million, and head or body movements can still be used to get closer to or farther away from the display. The only alternative to these movements is time multiplexing, through constant animation or interactive control such as a scrollbars for panning and zooming, which are even less effective than movements.

We have focused our research on 1600x1200 displays because they are widespread and well managed by current accelerated graphics boards.

Redisplay time: when time multiplexing is used, the redisplay rate should be maintained around 10 frames per second. If one million items have to be displayed at this rate, using accelerated graphics cards is the only option and opens the door to visualization enhancements. Common graphics cards can display around 15 million triangles per second at best so maintaining an acceptable refresh rate for one million items demands special techniques and more expensive cards or waiting for the next generation of cards.

For our work, we have used NVidia GeForce3 and 3Dlab Wildcat hardware accelerated boards with 2GHz and 1.7GHz Pentium PCs and the OpenGL API [23] with code written in C++.

4. Reaching One Million Items

To address the technical issues involved in visualizing and interacting with one million items, we have designed novel techniques relying on accelerated graphics hardware to provide high-density interactive visualization. The accelerated graphics hardware reduces the load of the main CPU (e.g. all rendering is done there) and offers many non-standard graphics attributes that we used to enhance the visualization of dense data sets.

4.1 Appropriate Visual Attributes

We only use quadrilaterals to represent items because, in a dense configuration, the perception of shapes is subject to interference. As item density increases shapes overlap and appear to merge. Also, contrary to several existing systems, we don't outline items using a one-pixel black line, which wastes two lines and two columns of pixels and requires sending the coordinates twice. Instead, we use slightly shaded quadrilaterals so that they remain distinguishable when tiled or stacked (see figure 2). The other visual attributes we use are saturated colors (for categorical or numeral attributes), intensity (for numeral attributes), quadrilateral sizes and position in scatter plots.

We describe each quad vertex with four values: X, Y, Z and S. X and Y are positions. S is a texture coordinate index. The Z coordinate is mainly used for fog and stereovision: we tilt the quads so that the upper left corner is closer to the camera and the lower right corner farther away; the fog function changes their intensity and smooth shading interpolates it across the rectangle.

Instead of sending RGBA colors for each quad, we use one-dimensional texture indices. The texture can contain a set of colors for categorical attributes: one color per category. It can also contain starting and ending color for continuous valued attributes. We then rely on hardware linear filtering to generate in-between colors as shown in figure 2. We also use texture transforms to map from abstract attribute values to color values, avoiding all color computation on the CPU side.

More attributes can be used if required. For instance, for treemaps, we also use the fog function to fade the colors to black when items are deeper in the tree by assigning the tree depth attribute to the Z coordinate of the quadrilaterals (see Figure 1). More control could be obtained by using a two dimensional texture and assigning the U texture coordinate (also easily available via the accelerated graphics hardware) to one abstract attribute and the V coordinate to another one. However, the only coloring scheme we have found to be effective using pre-attentive features with two attributes was assigning a saturated set of colors to one attribute and varying the brightness with the other.

4.1.1 Synthetic overlap attribute

When sending data to the graphics hardware, the count of overlapping items can be calculated using the stencil buffer. Displaying the content of the stencil buffer as intensity shows the overlapping counts (see figure 3.) This synthetic attribute is very important for visualization techniques that cannot avoid overlaps such as scatter plots and parallel coordinates. Even with hundreds of items, the distribution tends to be sparse with areas of high density that are hard to see. Transparency is useful when up to five items overlap, but with one million items, hundreds of overlapping items are common.

The overlap count can also be mapped to color, or used to filter the display and reveal a specific layer of items. This technique is similar to dynamic queries on standard attributes but instead relies on the stencil buffer for rejecting fragments above or below a specified number of overlaps.

4.1.2 Transparency and Stereovision

Using an accelerated graphics card provides graphic attributes not available or usable with traditional graphics APIs. We have experimented with transparency and stereovision. Transparency is beneficial with overlapped items but is not sufficient by itself to understand the number of overlaps. Furthermore, by blending colors, it interferes with its preattentive processing. Therefore, transparency is only useful when it can be varied interactively to reveal overlaps and density of overlapping items.

Stereovision hardware is now available for all the standard graphics cards for less than $100 through shutter glasses. Stereovision is preattentive, but there again, overlaps interfere with it and cannot be avoided at all, even with space-filling visualization techniques, since stereovision requires a perspective projection that introduces occlusion and therefore overlaps. Like transparency, we have found stereovision mostly useful for transient inspections.

4.2 Animation and Interaction

Exploring a large data set without a-priori knowledge typically requires trying several mappings of data attributes to visual attributes. A special problem happens when changing views using time multiplexing: the whole layout of the screen changes and the user cannot tell where sets of regions of interest in the original view have gone on the new one. A long time may be required to understand the relationship between items visualized in the new view and in the previous one. This problem is not addressed at all by current systems. When the number of items is small and space multiplexing possible, two or more views could be

displayed together using "snap together" techniques [17] and brushing and linking techniques [8] can be used to explore the tightly coupled views by highlighting items in multiple views; but this is not an option in our case.

We first describe the simple case of non-geometrical changes and then the general case.

4.2.1 View Flipping

When the positions of the data items are preserved – i.e. when only changing colors or stacking order – flipping between views allows quick comparisons to be made using retina persistency. This technique is widely used in astronomy to track variations over time (called *blinking.*) By flipping back and forth and moving the focus of attention, two views can be compared in seconds.

More views can be flipped through, similar to flipping techniques of traditional animators, to help discover trends and outliers across multiple views. This technique only requires a redisplay time below 300ms and a refresh time below 50ms, only achievable with double buffering.

4.2.2 Interpolation of geometrical attributes

When the geometry is modified between two views, it becomes difficult to impossible to understand relationships between views with flipping [16].

We have implemented a set of interpolation techniques to animate the transformations from one view to the other so that the eye can follow sets of items and understand patterns of change between views. Because the changes could be complex, users can select a subset of items and we only animate those items. The animation lasts one or two second and can be replayed back and forth at will. A slider can also be used to manually control the interpolation between views to facilitate understanding of item movements.

The simplest technique for animation is linear interpolation. However, when several visual attributes change, linear interpolation is confusing [25] and only allow users to track position changes at best. To help users understanding changes, we found that animating in two stages was beneficial: changing positions, then changing dimensions.

For scatter-plot-like visualization, position and size are independent so the middle configuration is easy to compute. Linear interpolation of positions is used to reach the middle configuration from the initial and linear interpolation of sizes is used to reach the final configuration. Both show trends when they exist.

Space-filling visualizations compute positions according to the size of items. Most of them [3, 9, 12] can be animated by linearly changing the size attributes since they use a layout that remains stable. Only some treemaps described in [4] use layout algorithms that are not geometrically stable. However, we found a general way to stabilize all these layouts for stage 2 (i.e. size changes) by computing the final layout using the final sizes values and linearly changing the sizes of all areas according to their initial values. This is shown in figure 4 where the final layout is computed by the "squarified treemap" algorithm, and used by previous frames with the sizes interpolated. The final layout tries to have items as "square" as possible but not the previous ones. Still, trends are visible.

For stage 1 (i.e. position changes), linear interpolation is usually the only option. Layout changes may be due to the underlying algorithm or to the change of an axis on a scatter plot, or the sorting order of a dimension linked to an axis. However, by only changing positions and not sizes, items are easier to track and trends are noticeable. For example, when changing the attribute to be mapped on the X axis of a scatter plot, items only move along the Y axis.

With treemaps, texture mapping [2] is very effective at speeding-up both stage 1 and 2. Most space-filling techniques use hierarchical containment, preserved by attribute change so one set of items is always inside a rectangle that can be warped and turned during the animation and there is no need to interpolate each item separately. With this technique, smooth interpolation can be achieved with one million item treemaps.

Logging/playback: Interactively specifying mappings from data attributes to visual attributes can become tedious and diverts the user from the visualization analysis. To avoid this distraction from occurring too often, we developed a tool allowing users to record sets of configurations and view them in turn using the left and right arrow keys, with animated transitions as described above. These views and configurations can also be saved to a file and applied to other data sets (e.g. to simplify routine examination of similar datasets.)

4.2.3 Dynamic Queries

Dynamic queries implies interactively filtering and redisplaying of the dataset through continuous interaction. Current systems use "range-sliders" to filter one attribute at a time, either changing the lower or upper value, or sweeping a given range of values between the smallest and the largest. The dataset needs to be loaded into main memory. To achieve the redisplay speed required for smooth interaction, we have designed a technique that relies on hardware acceleration. When the user activates a slider to perform the series of queries, all the items are sent to the graphics processor (GPU) and stored in a display list. The Z coordinate is calculated according to the attribute being filtered by the slider (e.g. if the user is filtering a film database on the size of the film, the size is assigned to the Z-axis.) Each time the slider moves, a new near or far plane value is computed and sent to the GPU and the list is redisplayed, leaving the visibility computation to the hardware.

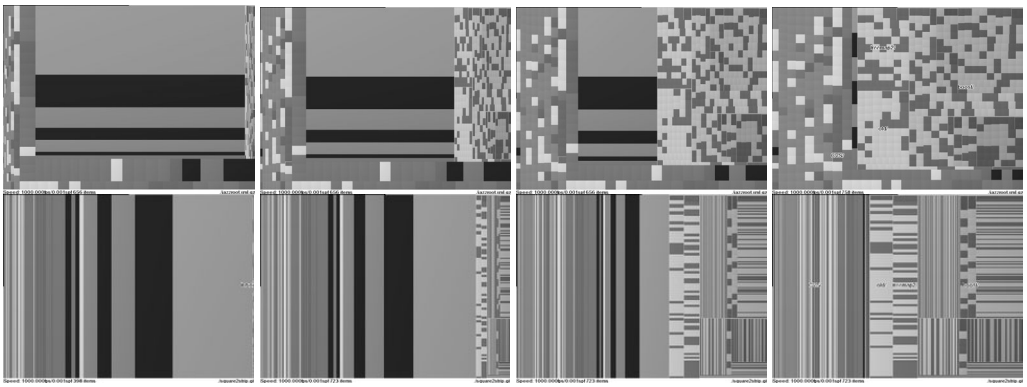
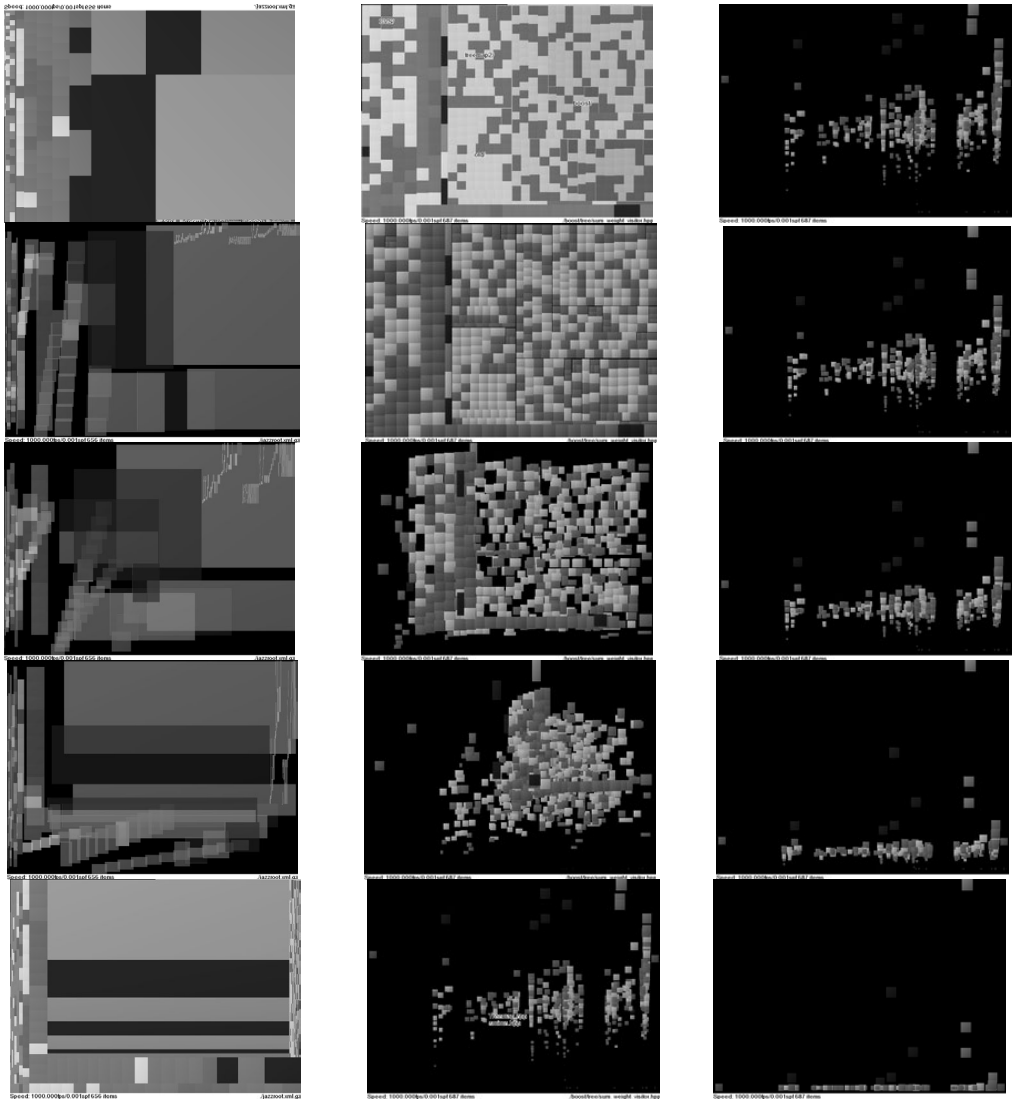


Figure 4: The animation of the treemap (with a squarified layout on the left or slice&dice layout on the right) allows users to follow specific items and observe patterns of change between two configurations using a different size attribute . A stabilized layout is used to avoid jumps.



a **b** **c**

Figure 5: Animation of visualizations using linear interpolation: a) interpolation between squarified and slice and dicelayouts, b) interpolation between a treemap and a scatter plot, c) interpolation when changing the Y axis on a scatter plot.

This technique works very well as long as the GPU has enough memory to hold the items and the shape of the items doesn't change during the dynamic queries. Using quadrilaterals requires 64MB of free memory (4 vertices per item, 4 data per vertex with 4 bytes per data) for one million items and only fits on special machines such as SGI O2. When applying dynamic queries on scatter plots with PC graphics cards, we use points instead of quads. Currently, OpenGL doesn't allow sending one array of points with varying sizes, although it can be done with NVidia Vertex programs extension. We send arrays of points sorted by decreasing point-size, dividing by 4 the amount of data sent to the card and making dynamic queries possible with one million items on a PC.

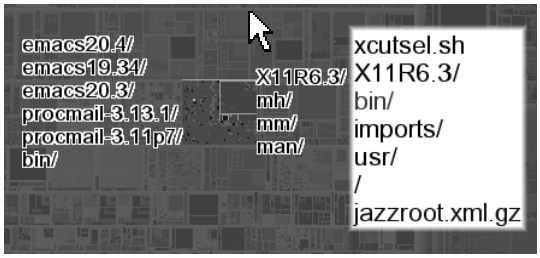


Figure 6: Dynamic Labeling of a Treemap (detail). When clicking on the treemap, labels are displayed dynamically on the outside of the region, which is grayed out. The popup menu allows users to adjust the depth of the region of interest.

4.2.4 Labeling

Text labels are not preattentive but are nevertheless important to understand the context in which visualized data appear. Labeling each item cannot be done statically on a dense visualization so we used the Excentric Labeling [10] dynamic technique for the scatter plot and extended its design for the Treemap, as shown in figure 6.

5. Performance

Our system reads data encoded in XML or a directory structure as input formats. It is made of 23,000 lines of C++, using high-performance techniques such as template metaprogramming [23] to achieve the required speed. We have used it with an NVidia GeForce3 board on a 2 GHz Pentium and a 3Dlab Wildcat 5110 on a dual 1.7GHz Pentium. To scale to a million items, the computation of layouts should be done in time linear with the number of items. This is the case with some treemaps and scatter plots but not with VisDB for example. Even using the fastest techniques, layout computation takes about 50% of the redisplay time.

Despite the high theoretical performance of the boards, we have not been able to go beyond 6 million quads per second on any of the boards we tried. The theoretical speed of 15 million triangles per second is only achievable for triangle strips, which is of no use for scatter plots and would require expensive computation for treemaps.

Combining software and hardware techniques provides a sustained performance around 2.5 million quads per second. By using texture mapping for animating treemaps, we achieve 10 frames per second for animating across any family of treemap. For scatter plots we have only reached 3 frames per second for animations on 1 million items, and 6 frames per second for dynamic queries. Finding techniques for improving that speed would be useful but the next generation of graphics cards and computers will solve the problem.

Our estimate is that these results correspond to a 20 to 100 time improvement on the available systems.

6. Conclusion and Future Work

Using a set of novel techniques we were able to visualize and explore for the first time 1 million items on a 1600x1200 display without aggregation. Our techniques rely heavily on commonly available accelerated hardware for displaying items in a dense yet manageable manner. We designed new animation techniques to help understand view changes and show trends and outliers when they exist. We developed a method to perform dynamic queries using the Z-buffer of a graphics card and achieved the speed required to interact with a million items. Finally, we experimented with non-standard visual attributes such as transparency and stereovision and found them effective for temporary inspections.

This work shows that the technical limits of information visualization are well beyond the typical 10^4 items handled by most existing systems, and opens the door to new possibilities for users: a library manager can use a treemap to review usage patterns of a million individual books organized with the Dewey decimal system; large databases of highway incidents or juvenile justice cases can be first examined without sampling or aggregation.

Our early user testing has been limited to collecting feedback from colleagues about the large treemap displaying our shared drives of one million files. It seems to confirm that users' experience analyzing data transfers to large visualizations, allowing them to make local and global comments on the data presented. Users appreciated the fine patterns, e.g. the distinctive pattern of web page directories that combine text and graphics, and seemed to actively engage their visual skills to compare and make sense of patterns. We have now identified two applications for user testing: 1) the visualization of the University of Maryland catalog and years of circulation data, and 2) fine grain analysis of Census Bureau data. The user testing will involve domain expert users, who are more apt to make sense of such large datasets and make suggestions for improvements.

More experiments are needed to understand how humans can cope with a large number of items and the possible limits of visualization, e.g. how can we best use visualization to gain an understanding of the organization and content of the 7 million items of the Library of Congress American Memory collection?

We are confident that human visual skills and the evolution of hardware will push information visualization much further than the million of items.

7. Acknowledgments

This work was supported in part by Chevron Texaco. We want to thank Ben Bederson and Ben Shneiderman for their helpful feedback on the paper.

For more information and demonstration of animations see: http://www.cs.umd.edu/hcil/millionvis/

8. References

[1] Ahlberg, C. and Shneiderman, B. Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Display. *Conference proceedings on Human factors in computing systems*, April 1994, 313–318, ACM New York.

[2] Aliaga, D. Visualization of Complex Models Using Dynamic Texture-Based Simplification, *Proceedings of IEEE Visualization'96*, Oct 27-Nov 1, pp. 101—106, 1996, IEEE, Piscataway, NJ.

[3] Baker, M. J. Eick, S. G. Space Filling Software Visualization. *Journal of Visual Languages and Computing*, Vol. 6, 119—133, 1995.

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

[5] Bier, E. A. Stone, M. C. Fishkin, K. Buxton, W. Baudel, T. A Taxonomy of See-Through Tools, *Conference proceedings on Human factors in computing systems*, April 1994, 358–364, ACM New York.

[6] Callahan, J., Hopkins, D., Weiser, M. Shneiderman, B. An Empirical Comparison of Pie vs. Linear Menus, *Proceedings on Human factors in computing systems May 1988*. 95–100, ACM New York.

[7] Card, S. K., MacKinlay, J. D., Shneiderman, B., *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann Publishers, 1998.

[8] Cleveland, W.S., McGill, M.E., Eds., *Dynamic Graphics for Statistics*. 1988

[9] Eick, S. G. Steffen, J. L. and Sumner, E. E. Jr. SeeSoft—A Tool for Visualizing Line Oriented Software Statistics. *IEEE Transactions on Software Engineering*, 18 (11) 957–968, November 1992

[10] Fekete, J.-D. Plaisant, C. Excentric Labeling: Dynamic Neighborhood Labeling for Data Visualization, *Proceeding of the CHI 99 conference on Human factors in computing systems*, May 1999, 79–90.

[11] Healey, C. G., Booth, K. S., and Enns, J, Visualizing Real-Time Multivariate Data Using Preattentive Processing, ACM *Transactions on Modeling and Computer Simulation* 5, 3, 1995, 190-221.

[12] Jerding D. F., Stasko J. T., The Information Mural: A Technique for Displaying and Navigating Large Information Spaces, *IEEE Transactions on Visualization and Computer Graphics*, 4(3), July/September 1998, 257—271.

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

[14] Keim, D.A., Visual Exploration of Large Data Sets *Communications of the ACM*, August 2001, Volume 44, Issue 8, 38–44.

[15] Lamping J., Rao, R., Pirolli, P., A focus+context technique based on hyperbolic geometry for visualizing large hierarchies, In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. ACM, May 1995, 401—408.

[16] Nowell, L., Hetzler, E., Tanasse, T., Change blindness in information visualization: a case study. *Proceeding of IEEE Symposium on Information Visualization 2001*, IEEE Press, 200, 11—15.

[17] North, C., Shneiderman, B., Snap-Together Visualization: Evaluating Coordination Usage and Construction, *Int'l Journal of Human-Computer Studies special issue on Empirical Studies of Information Visualization*, Volume 53, 5 (November 2000), 715-739.

[18] 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.

[19] Pook, S. Lecolinet, E. Vaysseix, G. Barillot, E. Context and Interaction in Zoomable User Interfaces, *Proceedings of the Working Conference on Advanced Visual Interfaces* May 2000, 227 – 231.

[20] Tanin, E., Beigel, R., and Shneiderman, B., Incremental Data Structures and Algorithms for Dynamic Query Interfaces, *SIGMOD Record* (25) 4, 21-24, December 1996.

[21] Triesman, A. Preattentive Processing in Vision. *Computer Vision, Graphics, and Image Processing*, 31 (1985) 156-177.

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

[23] Veldhuizen, T. "Using C++ template metaprograms," *C++ Report* (7) 4, May 1995, 36-43.

[24] Woo, M. Neider, J. Davis, T. and Shreiner, D. *OpenGL Programming Guide*, Third Edition, Addison-Wesley, 2001.

[25] Yee, K.-P., Fisher, D.; Dhamija, R.; Hearst, M., Animated exploration of dynamic graphs with radial layout. *Proceeding of IEEE Symposium on Information Visualization 2001*, IEEE Press, 2001, 43 – 50.