Visualization of Large Hierarchical Data by Circle Packing

Weixin Wang [1], Hui Wang [1], Guozhong Dai [1, 2], Hongan Wang [1, 2]
[1] Intelligence Engineering Lab, Institute of Software, Chinese Academy of Sciences
[2] State Key Lab. of Computer Science
4# South Fourth Street, Zhong Guan Cun, P.O. Box 8718, Beijing 100080, P.R. China
{weixin, wanghui}@ios.cn, {dgz, wha}@ iel.iscas.ac.cn

ABSTRACT
In this paper a novel approach is described for tree visualization using nested circles. The brother nodes at the same level are represented by externally tangent circles; the tree nodes at different levels are displayed by using 2D nested circles or 3D nested cylinders. A new layout algorithm for tree structure is described. It provides a good overview for large data sets. It is easy to see all the branches and leaves of the tree. The new method has been applied to the visualization of file systems.

Author Keywords
Tree visualization, nested circles, circle packing, file system

ACM Classification Keywords
H.5.2. [Information Interfaces]: User Interfaces. I.3.6 [Methodology and Techniques]: Interaction Techniques.

INTRODUCTION

The visualization of hierarchical information structures is an important topic in the visualization community [9]. Most of the work concentrated on the challenge to display large hierarchies in a comprehensible form. Various methods have been developed for this challenge, among which, node-link diagrams and space-filling visualization have been admittedly thought of as two classical methods. Node-link approach [4,6,7], which connects nodes with line segments in Euclidean space or hyperbolic space, represents the branches and leaves of a tree explicitly. However, the traditional node-link diagrams make poor use of the available display space [5]. It does not give an effective overview of large hierarchies for only those nodes that expanded manually are visible. Among the space-filling approaches, there are rectangular space-filling techniques, such as treemaps [5] and its variations [1,9,10], and radial space-filling techniques [2,8]. The treemaps partitions the display space into a collection of rectangular bounding boxes representing the tree structure. It provides a good overview of large data sets. However, one drawback of treemaps is that the hierarchical structure is hard to discern: Most of the space is used for the display of leaf nodes, and the branches are encoded implicitly [9]. The radial space-filling techniques work well in revealing hierarchical structures with fan-shaped slices. But the peripheral slices with low aspect ratio in the radial display are difficult to distinguish sometimes. Now the Grokker [3] visualization attracts much attention. It categorizes search results and displays them by circles containing subcategories in smaller circles or squares. It represents some categories well, but it does not give an overview of large data sets. The circles are not placed compact and many subcategories are invisible. It is necessary to click on a circle manually to expand and show its subcategories.

Inspired by treemaps and Grokker, we present a novel space-filling approach for tree visualization by using nested circles. It is similar to treemaps, which provides good overviews for large hierarchical data sets. But it use nested circles instead of rectangles. It makes it easier to see groupings and structural relationships. It differs from Grokker in that: it gives an effective overview of large hierarchies; all the branches and leaves are visible in this tree visualization; examples show the circles are placed compactly and thousands of circles can be displayed well in a screen space.

PACKING CIRCLES FOR TREE VISUALIZATION

In this section, the layout algorithm for tree visualization is described. It includes packing brother nodes at the same level and packing tree nodes at the different levels.

1. Packing brother nodes

The brother nodes at the same level and sharing the same parent are represented by a group of circles, that is $\{(C_i,\ r_i) \mid i=1,2,\cdots,N\}$, among, C_i is the center of the circle i, r_i is its radius related to the node size, and N is the number of nodes. The position of C_i is determined by r_i and N, according no overlapping and tangency. Firstly, we place three circles tangent to each other externally around a convenient point (e.g. origin). Connecting the centers of the three circles forms a closed loop. The loop of line segments (circles) is called **front-chain**. Then, a new circle is placed externally tangent to two of the circles on the front-chain. The front-chain will expand towards the exterior and has to be updated when a new circle is added as shown in Figure 1. The front-chain is represented by a double list. The center, the radius and the distance from the origin of each circle are stored.

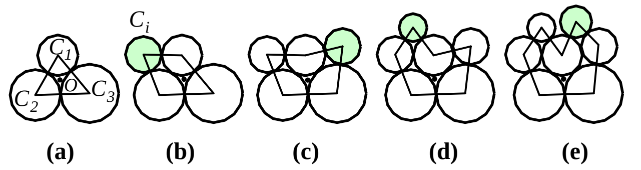


Figure 1. Update front-chain after a new circle is added

To simplify, the first three circles, denoted by C_1, C_2 and C_3, are placed tangent to each other around the origin $O(0,0)$ as shown in Figure 1(a). The initial front-chain is recorded in a double directed linked list $\{C_1 \leftrightarrow C_2 \leftrightarrow C_3 \leftrightarrow C_1\}$. Let C_m be the nearest circle to the origin on the front-chain and C_n ($n=m+1$) be the next circle of C_m on the front-chain as shown in Figure 2(a). The algorithm of circle packing is described in detail as follows.

(i) Calculate the center of C_i according to the radius r_i and the tangency of circle C_i to C_m and C_n.

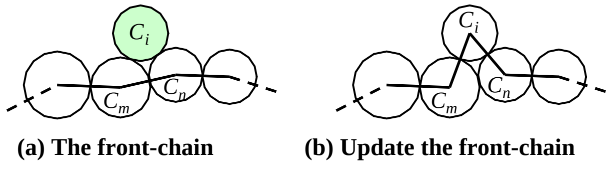
(ii) Search the front-chain and find out the circle C_j ($j \neq m$ and $j \neq n$) intersecting with C_i.

(iii) If C_i does not intersect with any circle on the front-chain, C_i is added to the front-chain directly, and packing C_i is terminated. The front-chain is updated to include C_i, such that $\{\cdots C_m \leftrightarrow C_n \cdots\}$ becomes $\{\cdots C_m \leftrightarrow C_i \leftrightarrow C_n \cdots\}$ as shown in Figure 2.

(iv) If C_i intersects with C_j, and C_j is a circle after C_n on the front-chain as shown in Figure 3(a), then the segments from C_m to C_j have to be deleted from the front-chain and the circle C_n is set by C_j. Go to step (i). A new position of C_i tangent to the two circles C_m and C_n (C_j) will be calculated as shown in Figure 3(b).

(v) If C_i intersects with C_j, and C_j is a circle before C_m on the front-chain as shown in Figure 3(c), then the segments from C_j to C_n have to be deleted from the front-chain and the circle C_m is set by C_j. Go to step (i). A new position of C_i tangent to the two circles C_m (C_j) and C_n will be calculated as shown in Figure 3(d).

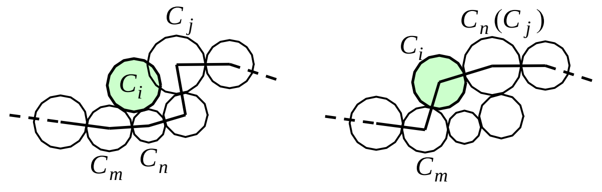
For $i = 4$ to N, repeat the process above.



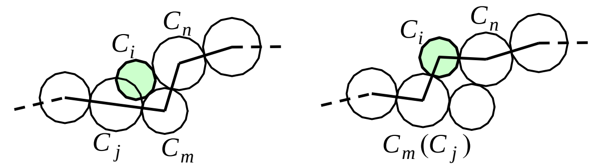
(a) The front-chain **(b) Update the front-chain**

Figure 2. Add C_i to the front-chain and update it

The circle packing algorithm is implemented on a PC with CPU speed of 2.41 GHz and 512 MB RAM using a VC++ 6.0 compiler. Figure 4(a) shows an example of packing 1000 circles with random radii from 1 to 10000. Figure 4(b) is a magnified view of part of Figure 4(a). The circle packing is robust and fast. The CPU time is only 0.005 seconds. The number of circles on the front-chain is 115.



(a) C_i intersects with C_j after C_n on front-chain **(b) Calculate C_i tangent to C_m and $C_n(C_j)$**



(c) C_i intersects with C_j before C_m on front-chain **(d) Calculate C_i tangent to C_m (C_j) and C_n**

Figure 3. C_i intersects with C_j on the front-chain then calculate new C_i tangent to C_j

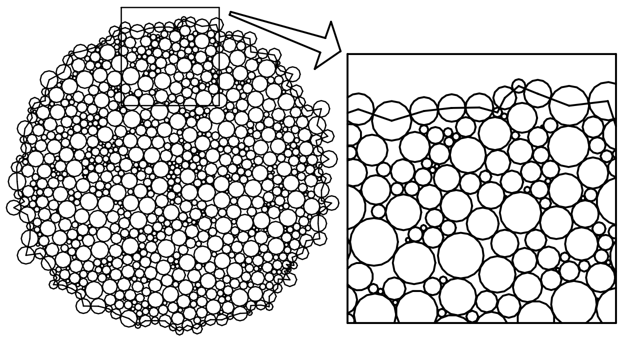
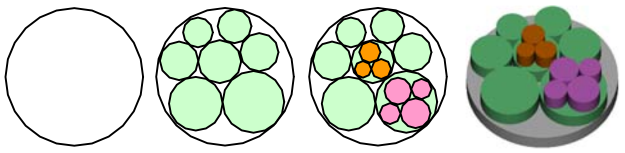


Figure 4. Packing 1000 circles with random radii

The circles of different radii are suitable for representing large numbers of brother nodes. There is no overlapping between any two circles, which ensures each circle is visible. The circles are to be packed as close as possible to save display space. A new circle is always placed outside the front-chain and beside the circle whose center is nearest to the origin, which ensures that the shape of front-chain is basically convex like a circle as shown in Figure 4(a). The circle-like shape is suitable for using a smaller circle to contain it.

2. Packing tree nodes

The tree nodes at different levels are represented by nested circles. The root node at level 0 is represented by a big circle as shown in Figure 5(a). All brother nodes (e.g. Figure 1e) at level 1 are packed into the root node as shown in Figure 5(b). If a node at level 1 has children, then its children at level 2 are packed into it as shown in Figure 5(c). The recursive procedure will return till a node has zero child nodes. The recursive procedure of packing nested circles is described as follows.



(a) Level 0 (b) Level 1 (c) Level 2 (d) 3D view

Figure 5. Pack circles into a circle recursively

(i) Determine the radius of each circle to represent node size. The size of an internal node is a sum of all its leaf node sizes.

(ii) Construct a circle to represent the root node (the current node).

(iii) Pack all child nodes at the next level of the current node according to the algorithm of circle packing.

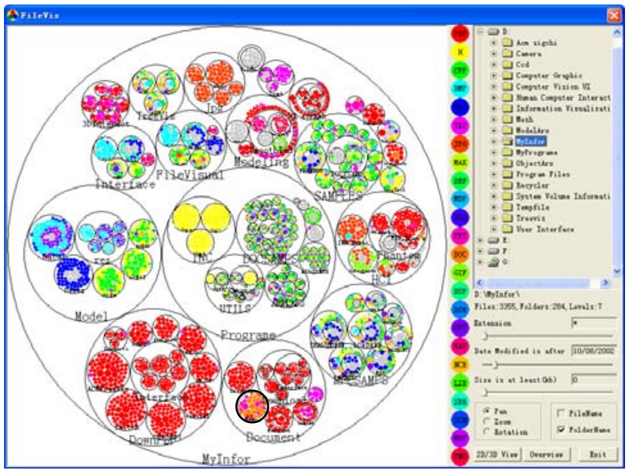
(iv) Put the child nodes into the current node. The child nodes need to be zoomed and moved suitably to make sure they are contained by the current node.

(v) For each child node, if it has children, then the current node is updated by it and go to step (iii); otherwise return.

It is easy to see the groupings and structural relationships from the two-dimension nested circles. In order to represent the hierarchical information structures more explicitly, the 2D nested circles can be turned into 3D nested cylinders. The radius of each cylinder is same to the radius of associated circle. The X-coordinate and Y-coordinate of each cylinder center are same to the coordinate of the associated circle center. Z-coordinate of each cylinder center is the product of the level of the associated node and the length of the cylinder. Figure 5(d) is 3D nested cylinders turned from Figure 5(c).

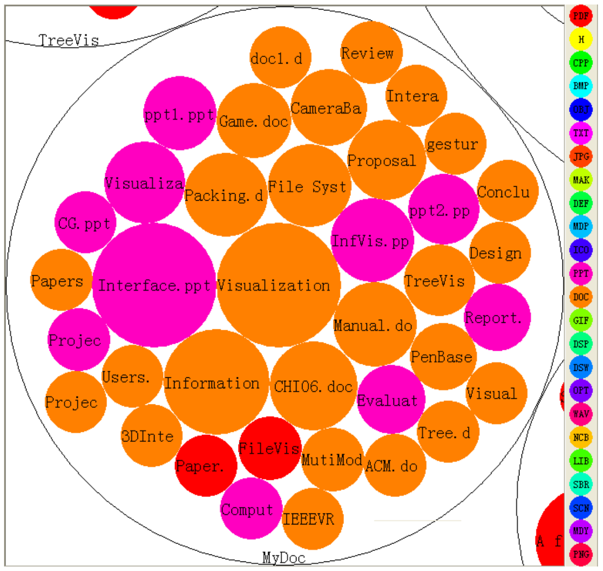
APPLICATION AND USER INTERFACE

The tree visualization using nested circles is applied to the visualization of file systems. The visualization tool for file systems has been developed by VC++6.0 and OpenGL. Figure 6(a) shows the user interface of the file system visualization. The main view left of the interface shows the overview of the directory "D:\MyInfor" using nested circles. There are 284 folders (internal nodes) and 3355 files (leaf nodes) in it. It contains 7 levels. The directories are represented by the white circles labeled at the bottom of them. The files are represented by the colorful circles labeled in the middle of them. The radii of colorful circles represent the file sizes. The right of the interface is the interaction controlling. The column of colorful circles right of the main view denotes twenty-four file types in common use, which is labeled by filename extensions. By clicking on one of them, others are filtered out. The tree view control top-right of the interface connects to a file system on a disk. By clicking on an item in it, the user can see the associated subitems represented by the nested circles in the left view. The dynamic query sliders (such as filename extension, file date and file size) are used to minimize the range of searching. The user can select a radio button to set the operation state (such as pan, zoom and rotation). The check box allows the user to set if the circles are labeled by the file names or folder names. The "2D/3D view" button allows the user to see the nested circles or 3D nested cylinders in the left view.

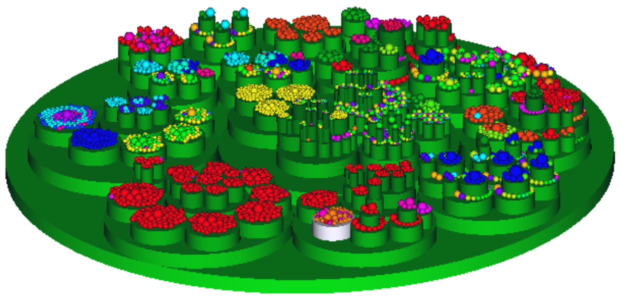


D:\MyInfor\
Files:3355, Folders:284, Levels:7
Extension *
Date Modified is after 10/06/2002
Size is at least(kb) 0
(• Pan) Zoom Rotation
FileName [✓ FolderName]

(a) User interface and the overview of "D:\MyInfor"



(b) The details of the focus "MyInfor\Document\MyDoc"



(c) 3D nested cylinders and spheres

Figure 6. The visualization of a file system

By clicking on any one folder (internal node) in the left view in Figure 6(a), the user can see the magnified or minified view freely. If the clicking point is in a smaller circle, the circle will zoom in; if the clicking point is outside of a circle bigger than the window left of the interface, the circle will zoom out. It is easy to determine if a clicking point is in a circle or not. For example, the clicking point is in the circle (MyInfor/Document/MyDoc) at the bottom of the overview in Figure 6(a), the circle becomes the focus and its details are displayed as shown in Figure 6(b). Figure 6(c) shows 3D overview turned from Figure 6(a). Among, the cylinders represent the internal nodes (directories) and the spheres represent the leaf nodes (files).

USER FEEDBACK

The file visualization tool (FVT) has been used by twenty-one students in our laboratory. The students were very familiar with Windows Explorer before. When we introduced FVT to them, they all felt interested in it. After they used it two weeks, eighteen students gave us anonymous feedbacks. In all there were a 67 percent positive rating, a 22 percent neutral rating and an 11 percent negative rating. Their comments are generalized as follows: It is robust and efficient even for thousands of files; The interface is friendly and it is easy to use; The user interface needs less operation than the traditional files management system to reach some goals as finding a target file, so this new UI has a relatively high efficiency; The visualization could give the users a very clear bird view of the whole files; It provided more information including not only the directory structures but also the file attributes; Because of your visualization, I can quickly find a file with an exact directory; Your zooming is convenient, I can quickly zoom to a very deep level by clicking only one or two times; The dynamic query sliders are useful to find files. I felt very comfortable when using the visualization tool and hope to use it again; It is better to use icons to represent file types; It is difficult to display a long file name fully; There is not file name as shown in 3D view; It is difficult to do a copy/cut and paste operation. These comments encourage us to improve it further.

CONCLUSION

This paper presents a new technique for the visualization of large hierarchy via the use of nested circles. The layout algorithm for tree visualization is described. The results of the algorithm lead to a simultaneously clearer and more compact visualization of a hierarchy than rectangular views and node-link diagrams. The visualization has the advantage over rectangular layouts of allowing even very small (leaf) nodes to remain visible without contorted aspect ratios. Though the use of display space is not more efficient than rectangular layouts, the space is helpful to see structural relationships and is easy to select (click) a node. The technique has been applied to the visualization of file system hierarchies. Most users in test give a positive rating. The nested circles technique has many future researches, such as introducing fisheye lens to visualization by the nested circles, improving 3D visualization, enhancing the interactive controls, evaluating the usability of the nested circles visualization.

ACKNOWLEDGMENTS

We thank the financial support from National Key Basic Research and Development Program, No. 2002CB312103 and National Natural Science Foundation of China, No. 60373056 and 60303019. We thank Professor S. H. Lo for revising our paper. We thank the anonymous reviewers for valuable guidance. We also thank the students who gave us feedback.

REFERENCES

1. Bederson, B.B., Shneiderman, B. Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies. *ACM Transactions on Graphics*, 21, 4 (2002), 833–854.
2. Chi, E.H., Pitkow, J., Mackinlay, J., Pirolli, P., Gossweiler, R., Card, S.K. Visualizing the evolution of Web ecologies. In *Proc. CHI 1998*, ACM Press (1998), 400-407.
3. Grokker. http://www.grokker.com
4. Heer, J., Card, S.K. DOITrees Revisited: Scalable, Space-Constrained Visualization of Hierarchical Data. In *Proc. AVI2004*, ACM Press (2004), 421-424.
5. Johnson, B., Shneiderman, B. Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information. In *Proc. Visualization 1991*, IEEE (1991), 284–291.
6. Lamping, J., Rao, R., Pirolli, P. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proc. CHI 1995*, ACM Press (1995), 1-8.
7. Robertson, G.G., Mackinlay, J.D., Card, S.K. Cone trees: Animated 3d visualization of hierarchical information. In *Proc. CHI 1991*, ACM Press (1991), 189-194.
8. Stasko, J., Zhang, E. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualization. In *Proc. Information Visualization 2000*, IEEE (2000), 57-65.
9. Van Ham, F., Van Wijk, J.J. Beamtrees: Compact visualization of large hierarchies, In *Proc. Information Visualization 2002*, IEEE (2002), 31–39.
10. Van Wijk, J.J., Van de Wetering, H. Cushion Treemaps: Visualization of Hierarchical Information. In *Proc. Information Visualization 1999*, IEEE (1999), 73-78.