Botanical Visualization of Huge Hierarchies

Ernst Kleiberg,* Huub van de Wetering,† Jarke J. van Wijk‡
Department of Mathematics and Computer Science
Eindhoven University of Technology

Abstract

A new method for the visualization of huge hierarchical data structures is presented. The method is based on the observation that we can easily see the branches, leaves, and their arrangement in a botanical tree, despite of the large number of elements. The strand model of Holton is used to convert an abstract tree into a geometric model. Non-leaf nodes are mapped to branches and child nodes to sub-branches. A naive application of this model leads to unsatisfactory results, hence it is tailored to suit our purposes better. Continuing branches are emphasized, long branches are contracted, and sets of leaves are shown as fruit. The method is applied to the visualization of directory structures. The elements, directories and files, as well as their relations can easily be extracted, thereby showing that the use of methods from botanical modeling can be effective for information visualization.

Keywords: *botanical tree, logical tree, huge hierarchy, strands, tree visualization, directory tree, phyllotaxis*

1 Introduction

An effective approach to the organization of a large number of items, be it files, staff members or books, is to group them and to repeat this step recursively. As a result, hierarchical data structures are ubiquitous. Small hierarchical structures are very effective to locate information, but the content and organization of large structures is much harder to grasp. A typical use-case is the question 'Why is my disk full?'. To answer this question in an effective and efficient way, the support offered by interactive visualization, showing both the structure as well as the sizes of individual files and directories, is almost indispensable.

We present a new method for the visualization of huge hierarchical structures. The method is based on a simple intuition. The term *tree* is standard for hierarchical data. When we observe botanical trees, we find that the leaves, branches, and their arrangement can easily be extracted, in spite of the very large numbers. Hence, what would happen if we try to visualize hierarchical data as botanical trees?

We explore this intuition as follows. In section 2 we discuss related work, both for tree visualization and tree modeling. In section 3 the strand model of Holton [6] is described and adopted as a basis for our method. In section 4 this method is tuned to our particular aim. Especially, we show that files can be visualized more effectively as fruit than as leaves. Results are shown and discussed in section 5, followed by the conclusions in section 6.

2 Background

Many methods have been developed for the display of hierarchical information structures, or, for short, trees. File browsers are the best known example, but they are limited in the number of items that can be shown simultaneously. A second important category are node and link diagrams. Within the graph drawing community [13] many algorithms have been developed to generate such diagrams in 2D. However, these diagrams do not use display space effectively, hence their applicability is limited to small trees, with at most a few hundred of nodes. The treemap method of Shneiderman [12] uses a space-filling approach, and can cope with much larger hierarchies. Also, treemaps offer a natural way to visualize additional information via the area of the rectangles. However, the structural information may get lost here, although this can be remedied by adding extra cues [14].

Another escape is to display trees in 3D instead of 2D. We refer to [5] for an overview. The hope is that the extra dimension would give, literally, more space, and that this would ease the problem of displaying large structures. This is not without problems, however. Occlusion renders parts of the model invisible, the depth of lines is hard to grasp. Hence, real time rotation and extra depth cues are indispensable to understand such 3D visualizations of trees. The best known example for 3D visualization of trees is the

*e.a.m.kleiberg@stud.tue.nl
†wstahw@win.tue.nl
‡vanwijk@win.tue.nl

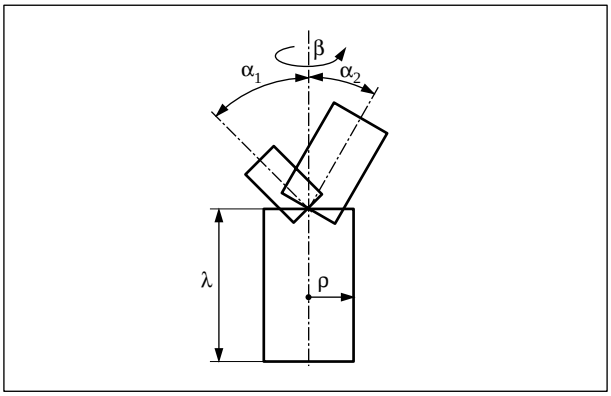

Figure 1. Stem and two sub-branches.

cone tree of Robertson [11], refined and extended by others to display larger structures [7, 2]. However, with these methods the number of elements that can be shown simultaneously in a comprehensible way still seems to be limited.

We aim at a 3D visualization method that enables the user to understand the structure of large tree structures as well as attributes of data. Our inspiration comes from nature: Botanical trees are large, but the elements and their arrangements can still be extracted. How can we tailor methods from botanical modeling to our purposes?

The modeling of botanical trees has intensively been studied in the graphics community. Besides for scientific curiosity, it also has direct practical applications. Modeling trees by entering polygons or other geometric objects directly is a tedious job, hence higher level methods are indispensable to generate trees and forests automatically, for instance for animation purposes. As a result of this research, many methods are available to generate images and models of botanical trees, such as fractals [9], texture mapping [1], and L-systems [10]. Especially the L-systems and all their extensions are an impressive machinery. L-systems enable the user to model the structure of a wide variety of plants and trees in a compact way. However, for our application this is not so relevant. The structure of the tree to be generated is determined by the data to be visualized, whereas in graphics the task is to generate the structure.

For our purposes the strand model of Holton [6], or a similar model used by Devroye and Pruszewski [3], is very convenient. It is particularly suitable because it can be tuned easily to follow the structure of the input hierarchy. Also, the strands enable a simple mapping of the size of elements into the radii of branches.

In the following sections we elaborate on this. As an example of a large hierarchical structure we use directory trees. The aim is the visualization of both the structural information as well as the content, i.e. file type and size.

3 Method

We first explain in detail the part of the strand model we use. Next we adapt it to suit the needs for visualizing directory trees.

3.1 Strand model of botanical trees

Strands are used to mimic the internal vascular structure of a botanical tree. The smallest branch is assigned one strand and each other branch has a number of strands that equals the sum of the strands of the sub-branches. In Holton's model [6] strands are used to determine the tree's structure, branch thickness and length, as well as branching angles between a branch and a sub-branch. Below we give a recursive function for generating a tree model as a set of cylinders. The model is a simplified version of Holton's model, leaving out, for instance, gravity and phototropism.

tree(s)
  stem :=*cylinder ((*0,0,0*), (*0,λ,0*),ρ);*
  if s_0=1 **then return** *stem;*
  s_1 :=℘*(s-2) +1; s_2 :=s-s_1;
  α_1 :=α*s_2/s; α_2 :=α*s_1/s;
  r_1 :=$\sqrt{s_1/s}$; r_2 :=$\sqrt{s_2/s}$;
  return *stem*
    $\cup\, T(0,\lambda,0) \circ R_y(\beta) \circ R_z(\alpha_1) \circ S(r_1)$ *tree*(s_1)
    $\cup\, T(0,\lambda,0) \circ R_y(\beta) \circ R_z(-\alpha_2) \circ S(r_2)$ *tree*(s_2);

The function *tree(s)* results in a model with as stem a cylinder with radius ρ and length λ and containing $s > 0$ strands. If s is larger than one the returned model contains the stem plus the transformed model of two sub-trees. The number of strands for each sub-tree is computed using some probability function ℘ which depends, for instance, on the recursion level and is chosen such that the total result looks like a real tree. One step of the recursive function *tree* is shown in figure 1. The transformation of the first sub-tree consists of a scaling (S) with a factor r_1 equal to the square root of the ratio of the corresponding strands. Hence, the area of the cross-section of the cylinder is proportional to the number of strands. After scaling, this sub-tree is rotated with an angle α_1 about the z-axis (R_z); the angles α_1 and α_2 are chosen such that their sum, the angle between the two sub-trees, equals a given angle α; furthermore, they are chosen such that the heaviest of the two sub-trees diverts the least from the stem. Both sub-trees are rotated with an angle β about the y-axis (R_y). For β often the angle $360/\varphi$ is used where $\varphi = (1+\sqrt{5})/2$ is the golden section. This is a result of the study of phyllotaxis, the arrangement of leaves on a stem, and is as such also applied in [4, 8] for modeling plants and trees.

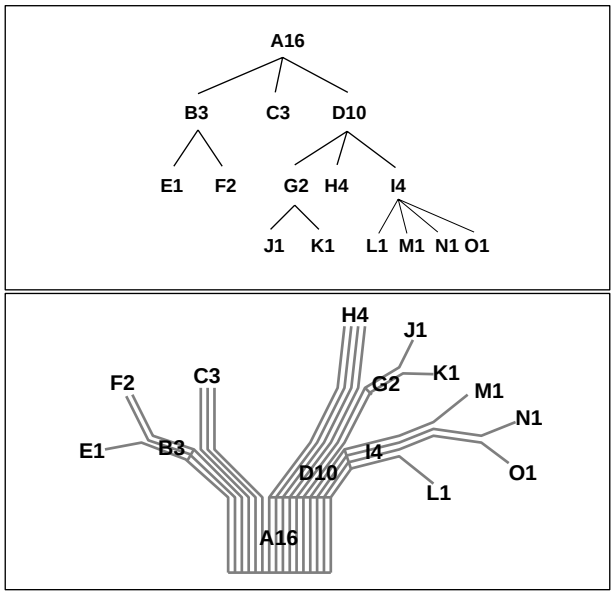

Figure 2. Node and link diagram (t) and corresponding strands model (d).

3.2 Strand model of directory trees

In this section we describe how we can use Holton's model to create a three-dimensional model of a directory tree. The original model generates a binary tree whereas in general a directory tree is not. However, we again obtain a binary tree by visualizing the directories as continuing branches: At each branching point a sub-directory is split off and the other sub-branch is now a continuation of the parent branch. The order in which the sub-directories are split off, is given by their sizes : The largest sub-directories are split off first. The top figure of figure 2 shows a node and link diagram of a directory tree where the numbers indicate the size of files and directories. In the bottom figure the corresponding strand model is shown.

As in the original model the geometrical properties of a tree are based on numbers of strands, here we define a number of strands for a directory, say dir, as follows:

$$strands(dir) = \sum_{d \in dl} strands(d) + \sum_{f \in fl} f.size$$

where we use the following data-structures for the directory tree.

```
Directory = record dl : list of Directory;
                   fl : list of File;
File      = record size : integer;
                   type : String;
```

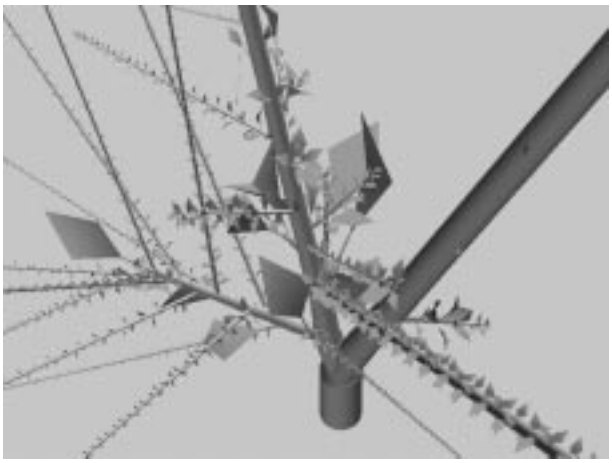

Figure 3. Result of *dtree1*.

Hence, a continuing branch of a large directory has many strands and is represented by a thick and long first segment. This visual cue will help us when browsing a tree looking for space consuming files and/or directories.

Below we give the recursive function *dtree1(d)* for the visualization of a directory tree d. On the deepest level the function *leaves* is called which returns some geometry representing the file list it gets as a parameter. To navigate through the directory tree we use the *pop* function on a list which removes the first element of a list and returns this element as its result.

```
dtree1(d)
    stem :=cylinder ((0,0,0),(0,λ,0),ρ);
    if d.dl=∅ then return leaves(d.fl);
    s :=strands(d); d1 :=d.dl.pop(); d2 :=d;
    s1 :=strands(d1); s2 :=strands(d2);
    G1:=Ry(β) ∘ Rz( αs2/s) ∘ S(√(s1/s)) dtree1(d1);
    G2:=Ry(β) ∘ Rz(-αs1/s) ∘ S(√(s2/s)) dtree1(d2);
    return stem ∪ T(0, λ, 0) (G1 ∪ G2);
```

A first implementation of the function *leaves* is similar to *dtree1*: At each recursion level a leaf is split off and a stem and a few triangles depicting a stalk and a blade are generated with the same transformations applied as for the sub-branches in *dtree1*. Figure 3 shows a result of the above algorithm. From the figure three visualization problems come forward : (1) the continuing branches representing a directory can not be followed easily at a branching point and as a result it becomes unclear which sub-branch represents the parent; (2) directories with many sub-directories and/or files lead to thin and long branches; (3) the leaves tend to clutter and do not give insight. In the following section we tackle these problems.

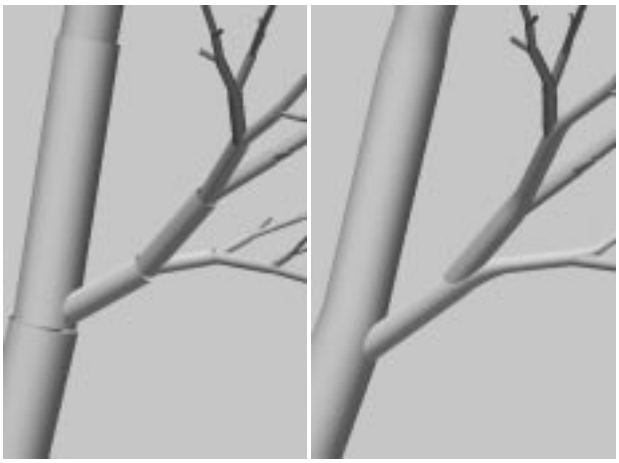

Figure 4. Continuation without (l) and with (r) extrusion.

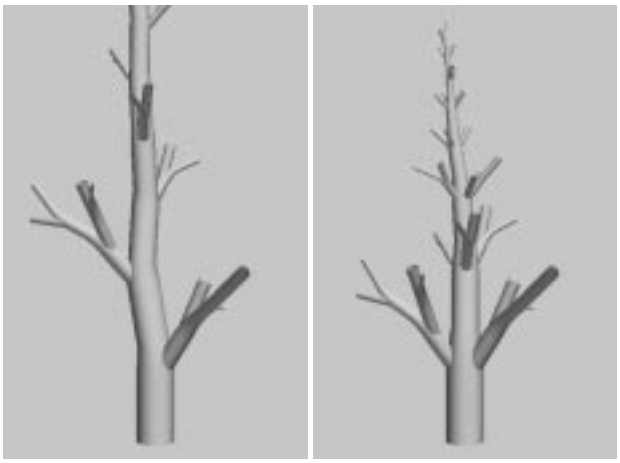

Figure 5. Model without (l) and with (r) contraction.

4 Refinements

4.1 Continuing branches

The visualization of a directory by a continuing branch should be clearly visible in the three-dimensional model. In [6, 1] sophisticated solutions are given, here we use a simple solution since we are not interested in modeling a tree but in visualizing directory data. We refine our model by adding a smooth transition between two cylinders. This geometric transition is simply created by adding an extrusion, along a suitably chosen spine, of the top circle of the parent cylinder to the bottom circle of the next cylinder in the continuing branch (see figure 4). The difference between the smooth and the abrupt transitions makes clear what the status of each branch is. As an extra cue, branches at different levels in the hierarchy are assigned different colors.

4.2 Contraction of long branches

When for each sub-directory or file a separate stem and sub-branch are used, a large number of those leads to a long sequence of stems, visible as a long and thin continuing branch. We improve the behavior of our model in this respect by conditionally removing the stem in the sub-branch of the continuing branch. Effectively, this replaces the binary tree with a general tree. The function *dtree2(d,e)* results in a three-dimensional model of a directory tree d where only a stem is introduced if $e \leq 0$. At the highest level *dtree2* is called with $e = 0$ indicating that the stem must be drawn, in the continuation of a branch e is set to $\varepsilon strands(d)$ indicating that a stem (in the continuing branch) is only drawn after at least a fraction $\varepsilon \in [0, 1]$ of the strands of d is split off. As a result the length of a continuing branch is limited to at most λ/ε.

dtree2(d,e)
  stem :=*cylinder* ((0,0,0),(0,λ,0),ρ);
  if $d.dl=\emptyset$ **then return** *leaves(d.fl);*
  s :=*strands(d);* d_1 :=*d.dl.pop();* d_2 :=*d;*
  s_1 :=*strands*(d_1); s_2 :=*strands*(d_2);
  if $e \leq 0$ **then** f :=ε*s **else** f :=e;
  G_1:=$R_y(\beta) \circ Rz(\alpha s_2/s) \circ S(\sqrt{s_1/s})$ *dtree2*(d_1, 0));
  G_2:=$R_y(\beta) \circ Rz(-\alpha s_1/s) \circ S(\sqrt{s_2/s})$ *dtree2*(d_2, f-s_1));
  if $e > 0$ **then return** $G_1 \cup G_2$
  else return *stem* $\cup\, T(0, \lambda, 0)(G_1 \cup G_2)$;

Figure 5 shows an example: Long branches are eliminated without sacrificing the clarity of the structure.

4.3 Files as fruit

To prevent cluttering of leaves we introduce an icon to represent a list of files and their sizes. We model this icon as a fruit consisting of a sphere with spots for each file. The positioning of the, possibly different sized, spots on the sphere is done again with a method from botanical modeling. In [4] such a method is given, here we use a less involved method given in [8]. In this last method items are placed on a sphere using a so-called phi-ball, a sphere that is divided in as many horizontal slices as there are items (here files) to be placed. The area of the slices (and equivalently their height) is proportional to the size of the corresponding files. At each of the slices a spot is placed and each slice is rotated β degrees with respect to the slice above it. The spots may be represented by disks with an area in proportion

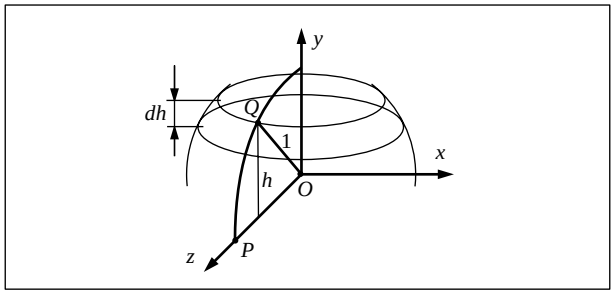

Figure 6. Phi-ball construction.

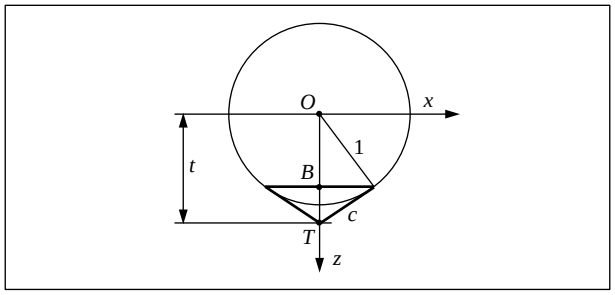

Figure 7. Computation of cone parameters.

to the size of the corresponding files. However, planar disks pasted on the sphere look irregular. Mapping of disks to the surface of the sphere requires many polygons or an involved texture generation step. The use of cones is more efficient and leads to an attractive result, especially if the dimensions and the position of the cones are chosen such that they are tangent to the sphere. Below a function *phiball(fl,bot,top)* is given that produces a sphere and cones for each file in the list *fl*. The *bot* and *top* parameters, both in $[-1, 1]$ indicate which part of the sphere is sliced for positioning spots.

phiball(fl,bot,top)
  $size := \sum_{f \in fl} f.size$;
  $P := (0,0,1)$; $O := (0,0,0)$; $h := top$; $G := \emptyset$;
  for each $f \in fl$
    $Q := (0, h, \sqrt{1-h^2})$;
    $c := \sqrt{f.size/size}$; $t := \sqrt{c^2+1}$;
    $B := (0,0,1/t)$; $T := (0,0,t)$;
    $G := R_x(-\angle POQ)(cone(B, T, c/t)) \cup R_y(\beta)G$;
    $dh := (top - bot)\, f.size/size$; $h := h - dh$;
  return $G \cup sphere(O, 1)$;

The sphere is positioned in the origin and has radius 1 (see figure 6). Each file's cone has its axis along the z-axis and a hypotenuse with length c equal to the square root of the ratio of the file size and the total size of the file list. From this the top (T) and base point (B) as well as the cone's radius can be computed (see figure 7). A cone is positioned by rotating it along the x-axis over the angle $\angle POQ$. In the subsequent iterations of the loop a cone is rotated along the y-axis over β degrees. For β the use of $360/\varphi$ gives good results.

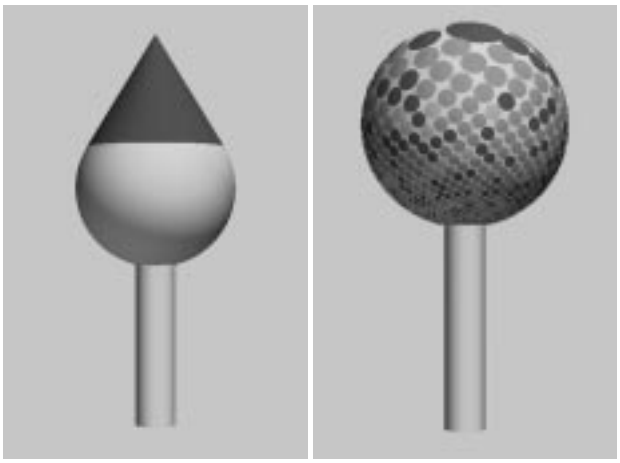

Figure 8. Phi-ball with one (l) and many (r) files.

With the function *phiball* the function *leaves* can be re-implemented by adding a stem to a phi-ball resulting in the models in figure 8. The use of cones and their parameterization lead to the effect that large files emerge as large, though bounded cones, whereas small files appear as small disks. Color is used to indicate the file type. The parameter *bot* is set slightly higher than -1 in these examples to prevent the stem to obscure file cones.

Showing a list of files by means of fruit as a phi-ball enhanced with cones, turned out to structure the massive information in the visualization of directories. The use of icons for showing the information was advised in [2] in the context of cone trees. Our solution with phi-balls is also viable in that context.

5 Results

Figure 9 shows the results of the refined model and can be compared to figure 3. The model represents the `C:\Winnt` directory on a windows NT machine, the prominently visible ball shows the `system32` sub-directory. Each cone with a yellow color on this ball represents a dll file in this folder. The model in this figure is comprehensive and clean compared to the model in figure 3. It is hence feasible to show even larger directory trees in one model. This is shown in figure 10 where the complete `C:\` folder of the same machine is shown. The largest and first sub-directory that is split of in this model is the

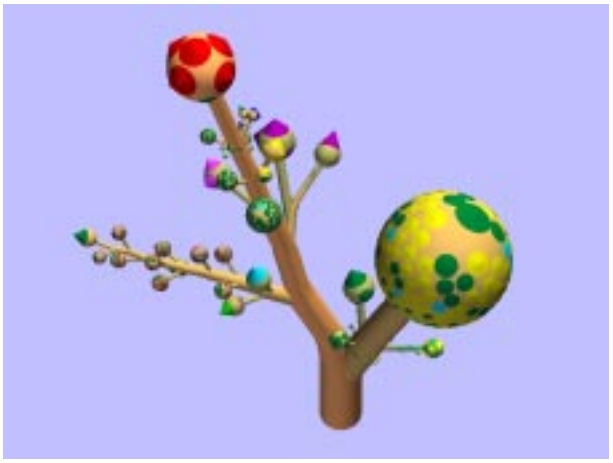

Figure 9. Final model with contraction, extrusion, and phi-balls.

`C:\Program Files` folder. The large green cone on top of the tree represents a file of more than 500 megabytes in `C:\`. The total tree consists of 47.551 files in 2.265 folders. Figure 11 shows a model of the same directory but with a different setting for α and β. The figures 12 and 13 show the unix directory of one of the authors. The first directory that is split off contains a big sub-branch with mostly blue cones representing postscript and pdf files. The phi-balls with mostly red cones represent directories with image files.

Three-dimensional models can only be judged fairly in an interactive environment where an inherent problem as occlusion can be overcome by interaction on the model. All images were generated with an interactive system we have implemented to generate and visualize these models. Real-time performance could be achieved on a modern PC with a 3D-graphics card using some straightforward level of detail methods, such as adaptive accuracy and removal of very small elements.

Hands-on experience with this system indicated that users are intrigued and stimulated by the visualization. Features such as large files (big cones), directories with many and/or large files (big spheres), and directories with a large overall content (thick branches) could easily be spotted. The pattern and color of the cones on the phi-balls provide a direct cue for the distribution and type of files. Finally, the resulting trees effectively show the files, the directories and their arrangements; in contrast to tree maps where this information is harder to extract.

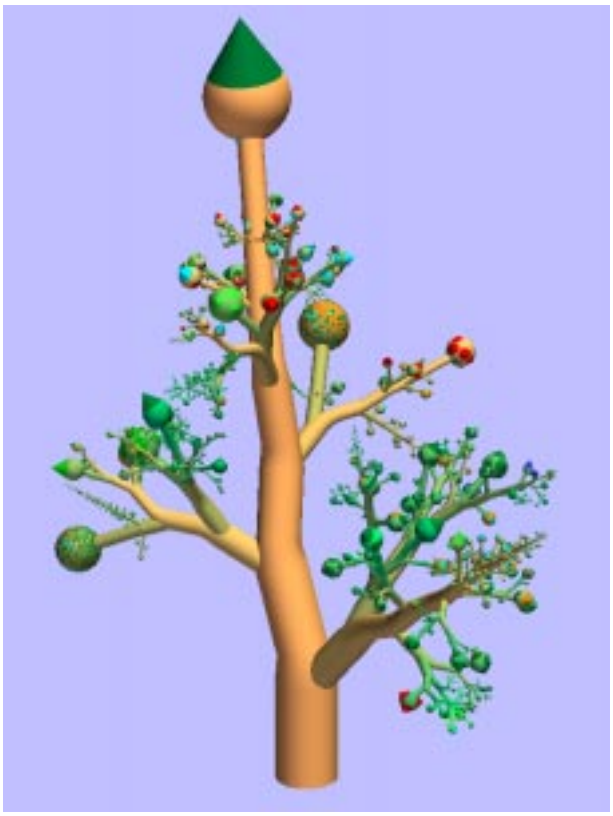

Figure 10. Complete hard disk with $\alpha = 45$ and $\beta = 360/\varphi$.

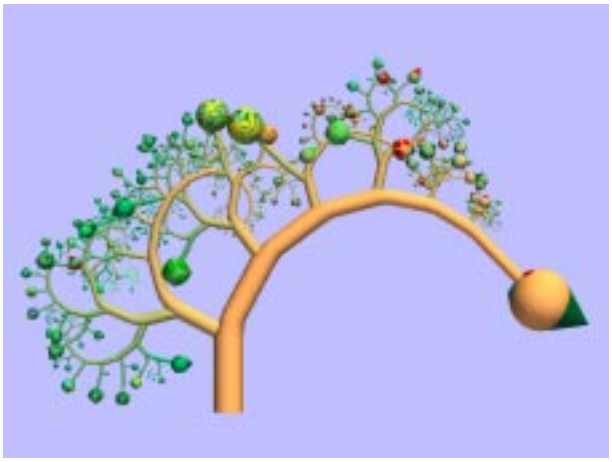

Figure 11. Complete harddisk with $\alpha = 90$ and $\beta = 0$.

6 Conclusions

We have presented a new method for the visualization of huge hierarchical data structures, based on methods from

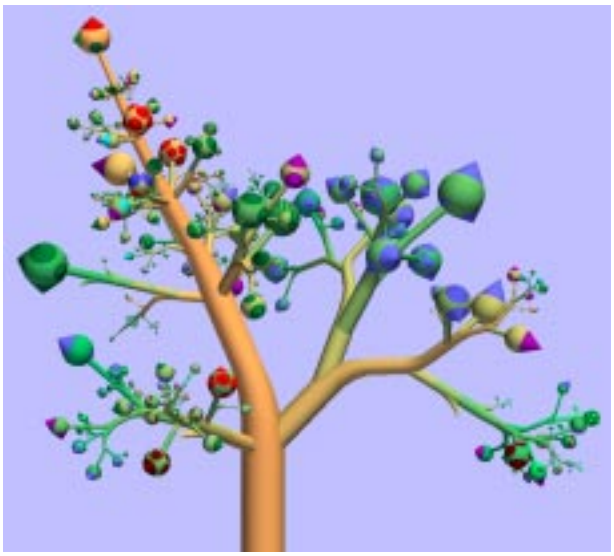

Figure 12. Unix home-directory.

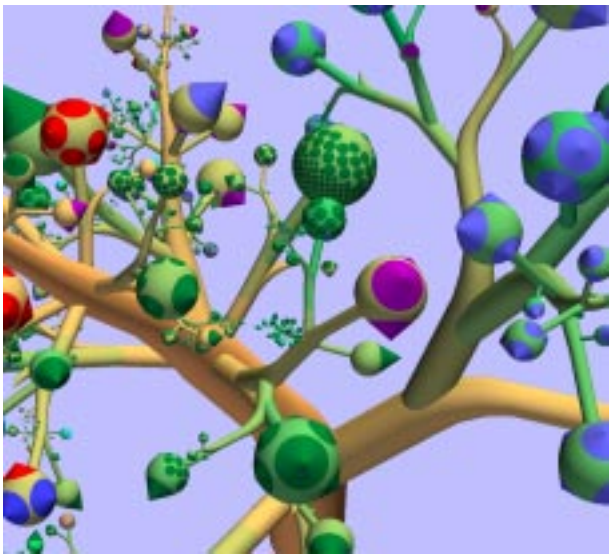

Figure 13. Detail of figure 12.

botanical modeling, specifically the strand model of Holton [6] and the phi-ball of Lintermann *et al.* [8]. Naive application of such botanical models gives unsatisfactory results, but we have shown they can be customized to visualize large hierarchical structures in a clear and compact way.

Two aspects deserve special attention. Firstly, the cone covered phi-ball is an effective icon for the visualization of the properties of a list of items, which we expect to be useful for other applications as well. Secondly, the branches of the tree and the cones on the phi-ball hardly ever collide, despite the fact that no special measures were taken to prevent this. Efficient use of space is an important topic in information visualization. To this end, nowadays mathematical methods (such as hyperbolic scaling), algorithmic methods (such as the tree map algorithm), and physically based models (such as mass-spring models) are used. The methods of botanical modeling and especially the concept of phyllotaxis provide a surprisingly simple and very effective alternative solution.

We have only just started to apply methods from botanical modeling for information visualization, and expect that many other applications can take advantage of this rich source of inspiration. Using our own work as an example, only a part of Holton's original strand model has been used. Texture and gravity could for instance be used to visualize other aspects of the data, such as the creation date of files and directories; for other types of hierarchies the parametrization of for instance branching angles to the level of hierarchy could be beneficial. In summary, *natura artis magistra*.

References

[1] Jules Bloomenthal. Modeling the mighty maple. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 305–311, 1985.

[2] Jeromy Carriere and Rick Kazman. Interacting with huge hierarchies: Beyond cone trees. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 74–81, oct 1995.

[3] Luc Devroye and Paul Kruszewski. The botanical beauty of random binary trees. In F. Brandenburg, editor, *Graph Drawing 95*, pages 166–177. Springer-Verlag, 1995.

[4] Deborah R. Fowler, Przemyslaw Prusinkiewicz, and Johannes Battjes. A collision-based model of spiral phyllotaxis. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 361–368, July 1992.

[5] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualisation and navigation in information visualisation. In B. Falcidieno and J. Rossignac, editors, *Eurographics*, 1999.

[6] Matthew Holton. Strands, gravity and botanical tree imagery. *Computer Graphics Forum*, 13(1):57–67, March 1994.

[7] Hideki Koike and Hirotaka Yoshihara. Fractal approaches for visualizing huge hierarchies. In *Pro-*

ceedings of the 1993 IEEE Symposium on Visual Languages, pages 55–60, 1993.

[8] Bernd Lintermann and Oliver Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19(1):56–65, January/February 1999.

[9] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W.H. Freeman & Company, New York, NY, USA, 1977.

[10] Przemyslaw Prusinkiewicz, Aristid Lindenmayer, James S. Hanan, et al. *The algorithmic beauty of plants*. Springer-Verlag, New York, 1990.

[11] 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.

[12] Ben Shneiderman. Tree visualization with tree-maps: A 2-D space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, January 1992.

[13] Ioannis G. Tollis, Giuseppe Di Battista, Peter Eades, and Roberto Tamassia. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[14] Jarke J. van Wijk and Huub van de Wetering. Cushion treemaps - visualization of hierarchical information. In G. Wills and D. Keim, editors, *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis'99)*, pages 73–78, Oct 1999.