Eurographics/ IEEE-VGTC Symposium on Visualization (2006)
Thomas Ertl, Ken Joy, and Beatriz Santos (Editors)

PhylloTrees: Phyllotactic Patterns for Tree Layout

Petra Neumann[1], Sheelagh Carpendale[1], and Anand Agarawala[2]

[1] Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada T2N 1N4
[2] Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, ON, Canada M5S 3G4

Abstract

Motivations for drawing hierarchical structures are probably as diverse as datasets to visualize. This ubiquity of tree structures has lead to a manifold of tree layout algorithms and tree visualization systems. While many tree layouts exist, increasingly massive data sets, expanding computational power, and still relatively limited display space make tree layout algorithms a topic of ongoing interest. We explore the use of nature's phyllotactic patterns to inform the layout of hierarchical data. These naturally occurring patterns provide a non-overlapping, optimal packing when the total number of nodes is not known a priori. We present PhylloTrees*, a family of expandable tree layouts based on these patterns.*

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces – Graphical user interfaces (GUI) I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction techniques

1. Introduction

Hierarchically structured data sets occur with sufficient frequency that tree visualization approaches continue to be a reoccurring research topic. Trees arise naturally due to data characteristics as in the case of family trees, phylogenetic trees, and software structures, as well as from common methods of information organization. One example involves creating categories and subdividing these categories, thereby imposing a hierarchical structure that can be useful in information access and navigation. The resulting hierarchies lend themselves to be visualized as trees. While many tree visualizations exist, increasingly massive data sets, expanding computational power, and still relatively limited display space makes this a topic of ongoing interest.

We add a family of layout variations, called *PhylloTrees*, to the growing number of tree algorithms. The inspiration for the underlying layouts comes from phyllotactic patterns [Vog79]. Phyllotactic patterns are common in nature. The term *phyllotaxis* refers to the arrangement of lateral organs of plants that have been sequentially produced. It is derived from the Greek *phyllos* = leaf and *taxis* = order. Familiar examples include the placement of seeds in a sunflower, the organization of scales on a pineapple, or the arrangement of leaves on plants (see Figure 1). Phyllotactic patterns are an everyday phenomenon found in plant life around the globe but are at the same time highly complex and follow subtle mathematical relationships. The florets in the head of a sunflower, for example, form two oppositely directed spirals whose numbers are successors in the Fibonacci sequence.

Figure 1: *Phyllotactic patterns in plants. Images are copyright of Tillman Steinbrecher (left) and Tobias Isenberg (middle, right), used with permission.*

Phyllotactic patterns offer several advantages for tree layout. They are fractal and, therefore, self-repeating, making it easier to comprehend a large number of nodes, as one only needs to understand one principle shape. They support interactive manipulation yielding many layout possibilities through the adjustment of only two parameters. They also provide the ability to visualize large hierarchies. Phyl-

loTree layouts can handle growth in the number of nodes at any level without requiring significant restructuring. Such expandable layouts are of interest because information visualizations often have to contend with dynamic data that is not fully specified a priori. A common example is our personal computer data files where it may be necessary to add nodes on many levels during everyday use.

Our paper is structured as follows. First we introduce related work in Section 2 followed by a mathematical description of how to create phyllotactic patterns in Section 3. Section 4 presents the three-dimensional PhylloTree layout followed by a discussion and conclusion in Sections 5 and 6.

2. Related Work

The inspiration for our 3D layout algorithm comes from ConeTrees [RMC91], one of the first three-dimensional tree layouts. ConeTree nodes are laid out symmetrically around three-dimensional cones with the apex of the cone pointing to the parent node. Fractal Trees [KY93] extend ConeTrees using the same form factor. Similar to our approach Fractal Trees show a self-similar layout on each level. This approach displays a subset of nodes that are selected according to a level of interest and are thus able to show huge hierarchies. The H3 viewer uses a three-dimensional hyperbolic space for tree layout which creates an automatic focus-in-context display capable of displaying thousands of nodes [Mun97]. The PolyPlane layout uses subplanes defined by regular polytopes to lay out tree structures in 3D [HM04]. According to the authors this layout reduces visual complexity and eases navigation in the tree. While their algorithm is easy to implement, finding the best partitioning is NP hard.

On a semantic level the botanical tree visualization by KLEIBERG et al. is related to our approach [KvdWvW01]. Their visualization technique is also inspired by nature and visualizes hierarchical structures in 3D using the strand model which mimics the internal vascular structure of a botanical tree. The resulting visualization is very unlike ours giving the tree data structure an appearance close to the branch structure found in natural trees with the leaves being displayed as fruits on the tree.

In addition to these 3D tree layouts, many others have been developed in 2D. We refer to [HMM00] for an overview. Using a third dimension is often done with the goal of making more effective use of screen-space and for allowing the whole tree structure to be perceived at one glance. However, using the third dimension also introduces problems. Occlusion might render parts of the structure invisible and often times the depth of objects is hard to interpret. To circumvent these problems it is essential to use interaction or additional depth cues to understand the 3D structure.

Phyllotactic patterns found in nature offer exciting inspiration for information visualization and graph drawing. Modeling plants using these patterns has been an active research topic in mathematics, biology, and computer graphics. For a literature overview refer to [Eri83] and [Jea94]. We chose a model developed by VOGEL to describe the sunflower head which will be discussed in more detail in the following section [Vog79]. This research extends our earlier work described in [CA04]. So far, phyllotactic patterns have been used in computer graphics for realistic modeling of plants and plant structures. We know of no approach that uses these patterns in graph drawing or information visualization.

3. Creating Phyllotactic Patterns

We chose to create phyllotactic patterns according to VOGEL's [Vog79] model. VOGEL's model places equally sized organs on a flat disk, offering several layout variations through the manipulation of just two factors.

He gave the following mathematical description for phyllotactic patterns:

$$\phi = n * \alpha, \quad r = c * \sqrt{n}, \quad n = 0, 1, 2 \ldots, n_{max} \qquad (1)$$

where,

- ϕ: is the angle between a reference direction and the position vector of the n^{th} node in polar coordinates. The origin of the polar coordinate system is at the center of the phyllotactic pattern.
- n: is the ordered number of nodes counting outward from the center. For PhylloTrees the first child of a node has $n = 0$, the second child $n = 1$ and so on.
- α: describes the angular constant for the phyllotactic pattern and is constant between successive nodes.
- r: is the distance between the origin of the polar coordinate system and the n^{th} node.
- c: is a spacing constant describing the packing of nodes.

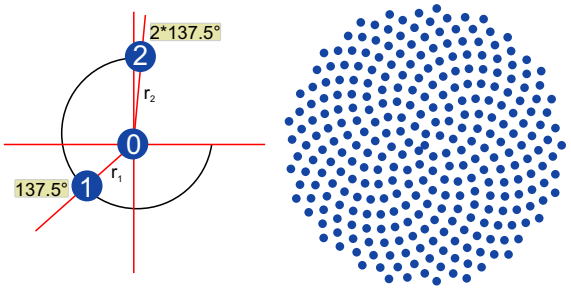
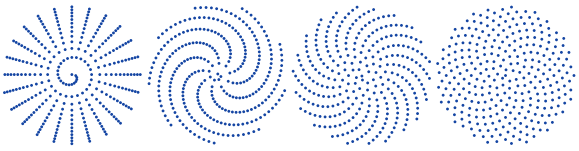


Figure 2: *Development of the phyllotactic pattern (a) and a phyllotactic pattern with 300 nodes using* $\alpha = 137.5^\circ$ *(b).*

Figure 2(a) shows the positioning of three nodes $n = 0$, $n = 1$, and $n = 2$, while the right figure shows the algorithmically generated pattern with 300 nodes. The angular constant for sunflowers, 137.5°, is used in 2(b) and has been

shown to provide optimal packing without introducing overlapping [Vog79]. Adjustments to this angular constant provide a rich palette of layout possibilities. Figure 3 gives an overview of four patterns created with a changing angular constant α.



(a) $\alpha = 10.0°$ (b) $\alpha = 39.5°$ (c) $\alpha = 82.0°$ (d) $\alpha = 137.5°$

Figure 3: *Different patterns created by changing the angular constant* α.

4. Creating PhylloTrees

Our layout algorithm uses a uniform node placement to visualize relationships between parent and child nodes. The layout process recursively arranges nodes for every subtree on every level of the tree. Starting from a parent node the children are laid out in a circular space bounded by $radius_{max} = c * \sqrt{n_{max}}$. The whole circular space is filled with a node's children using a chosen phyllotactic pattern. The children are then placed in 3D space and connected to their parent with edges creating a three-dimensional node-link diagram. The tree layout can be adjusted by interactively choosing different layout parameters as is discussed below.

4.1. Adjusting child node layout

Since our formula for calculating phyllotactic patterns only provides coordinates in two dimensions we are free to choose where to place child nodes along the third dimension. To provide for differing visualization needs we developed several ways in which child nodes can be laid out in 3D.

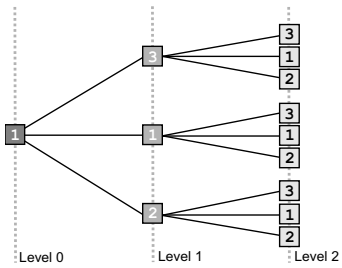
Figure 4 gives an overview of the four mappings currently implemented in our system. These four mappings were chosen for their algorithmic simplicity and for the intuitive layouts created, which make them easy to interpret. Based on these layouts many others can be developed, for example, another obvious layout would be a complete mapping of all nodes to cascading spheres. The main difference between the four presented layouts is the chosen *angular direction* and the node layout space. The *angular direction* of a node is defined as the angle created by the edge connecting the node and its parent. Next, we will briefly explain our four different mappings.

PhylloTree Layout I
Our basic tree layout is similar to the original ConeTree layout. Figure 4(a) shows a diagrammatic description of this layout. The first child nodes on a subtree level all have the same angular direction, thus the edges connecting the first child with its parent are parallel. For example, in our implementation the angular direction for the first child node is always parallel to the z-axis. The other child nodes are then laid out in the space around the first child node on a plane perpendicular to the first child node's angular direction and bounded by $radius_{max}$. The advantage of this simple layout is that all tree nodes that reside on the same level in the hierarchy are displayed on the same plane in 3D making it easier to find nodes on a per-level-basis. The layout is also very compact usually not extending much beyond the first level's layout radius, depending on the number of nodes at higher levels. However, due to the compact layout it becomes difficult to discern a large number of nodes at higher levels. Figure 4(b) and 4(c) show a bottom and side view of this layout.

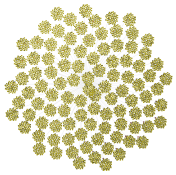
PhylloTree Layout II
To provide more space for the layout of nodes on higher levels this layout stretches subtrees on subsequent levels further apart (Figure 4(d)). In this layout the angular constant is inherited from the parent. All nodes on one level in the hierarchy remain on one plane in the visualization but the layout might require a large display space if a high number of subtrees needs to be displayed. Look at Figure 4(b) and 4(e) for a direct comparison of Layout I and II's spatial requirements for trees with the same number of nodes.

PhylloTree Layout III
Layout III combines some advantages of Layout I and II. First child nodes inherit their parent's angular direction, as in Layout II but all other child nodes on a subtree level are laid out on a plane perpendicular to this angular direction. Figure 4(g) again gives an overview of this layout. This layout is more compact compared to Layout II but leaves more room for larger node display compared to Layout I. Figures 4(b), (e), and (h) provide a visual comparison between these first three layout patterns. In this Layout III nodes at one level in the hierarchy are not placed on the same plane in the layout. The user has to inspect the tree closer or use other means of associating nodes to levels by, for example, coloring nodes according to their level.

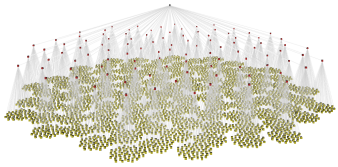
PhylloTree Layout IV
Our last layout is derived from Layout III. Again, the first child node inherits its parents angular direction (cf. Figure 4(j)). For this fourth layout we wanted to emphasize parent-child relationships by making all parent-child links in a subtree of equal length. One can imagine the final layout to look similar to a traditional soccer ball where the nodes would be laid out on curved patches. As the mapping from the two-dimensional polar coordinates to three-dimensional layout coordinates is not trivial we will describe our mapping in more detail. A simple mapping would use the 2D polar coordinates from Equation 1, convert these to Cartesian coordinates (x,y) and find a point on the sphere with a radius r equal to a chosen level distance by calculating $z^2 = r^2 - x^2 - y^2$. This, however, would map most 2D points



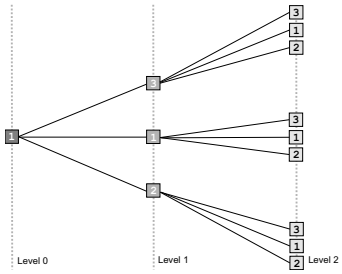
(a) PhylloTree Layout I.



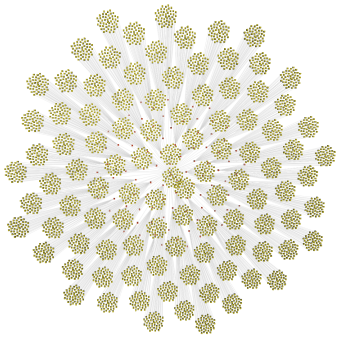
(b) Layout I seen from the bottom.



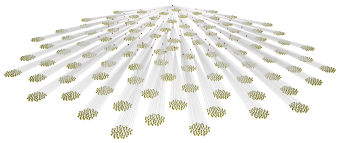
(c) Layout I seen from the side.



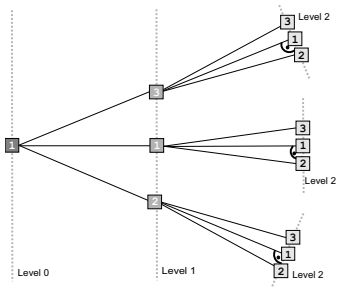
(d) PhylloTree Layout II.



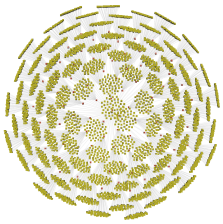
(e) Layout II seen from the bottom.



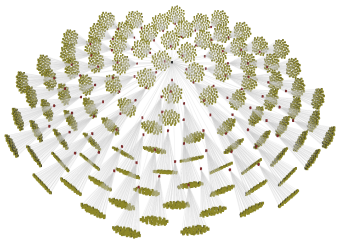
(f) Layout II seen from the side.



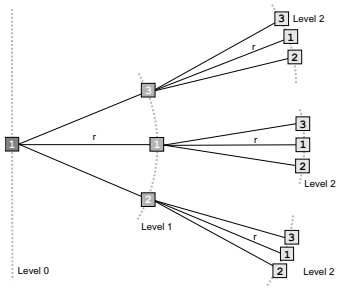
(g) PhylloTree Layout III.



(h) Layout III seen from the bottom.

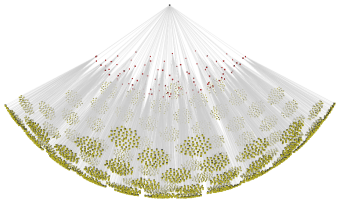


(i) Layout III seen from the side.



(j) PhylloTree Layout IV.

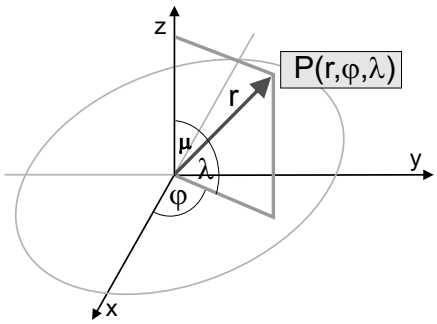
(k) Layout IV seen from the bottom.



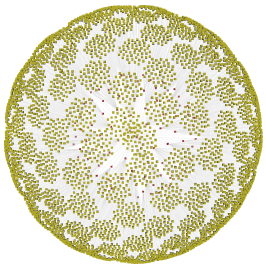
(l) Layout IV seen from the side.

Figure 4: *Different possible mappings of the 2D pyllotactic patterns to 3D. The second column shows the respective layouts applied to a 5.000 node tree as seen from the same distance in 3D. The third column shows a close-up side view of each layout.*

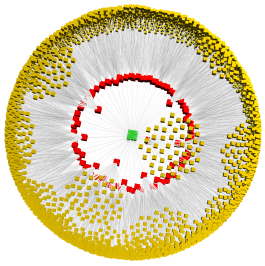
to the outer edges of the sphere and leave a large hole in the center (c.f. Figure 5(c)).



(a) Spherical Coordinates used for Layout IV.



(b) A 5.000 node tree using PhylloTree Layout IV.



(c) The same tree using a less effective spherical mapping.

Figure 5: *Spherical mapping for PhylloTree Layout IV. The first image (a) shows the spherical coordinates used for the mapping shown in (b). A less effective mapping is shown in (c). How to avoid the apparent node overlap in (b) will be discussed in the following.*

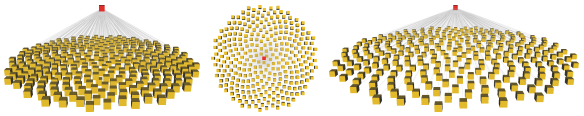
For our mapping we calculated the 2D polar coordinates relative to spherical coordinates. We first choose a maximal angle μ_{max} (cf. Figure 5(a)) to determine the maximal size of the layout cone. Then we get the maximal layout radius r_{max} for our phyllotactic pattern in 2D from Equation 1. The ratio of the radius for each node r_i to r_{max} is used to calculate μ_i for each node in relation to μ_{max}:

$$\mu_i = (r_i / r_{max}) * \mu_{max} \quad (2)$$

This leads to a much more equal spread of nodes across the sphere as can be seen in Figure 5(b). This layout is the most compact among the ones introduced here.

4.2. Adjusting the spacing constant

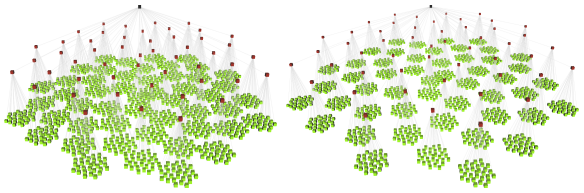
Figure 6(a) shows a simple tree with a root and 300 children that have been laid out using the angular constant $\alpha = 137.5^\circ$. The view from the top (Figure 6(b)) shows the emerging spirals with closely packed nodes. In Figure 6(c) the spacing constant has been adjusted to allow more space for the addition of second level child nodes.



(a) (b) (c)

Figure 6: *Single level tree with $\alpha = 137.5^\circ$; varying the spacing constant adjusts the space between nodes.*

Figure 7(a) includes 30 secondary child nodes for each of the 50 primary children. One can see that the secondary children are becoming crowded. This crowding can be addressed by increasing the spacing constant. In Figure 7(b) the primary children's spacing constant has been increased and now allows for additional secondary children.



(a) (b)

Figure 7: *Using the primary children's spacing constant to provide more space for secondary children.*

The spacing within each subtree on a given level can be calculated adaptively and independently. With this individual spacing algorithm subtrees will be assigned a spacing constant on each level depending on the distance to its closest neighbor, thus ensuring a maximal size for the subtree on that level without introducing overlap. Figure 8 gives an overview of a small balanced and unbalanced tree with a constant spacing level for each subtree on each level (left column) and an adaptive spacing constant chosen for each subtree to maximize its display space (right column).

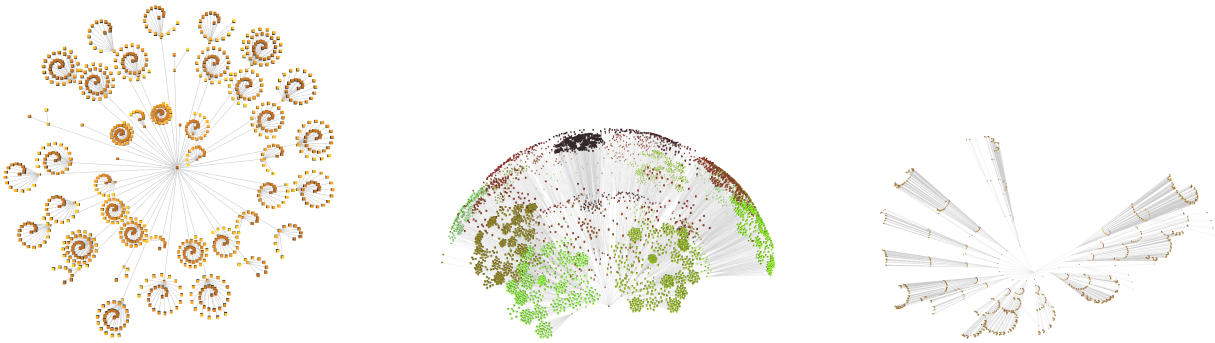
4.3. Adjusting the angular constant

While the angular constant $\alpha = 137.5^\circ$ provides optimal packing when the total number of nodes is not known apriori, many other angles also provide interesting layouts, though the node spacing is not as dense. Figure 9 gives an overview of several different layouts of balanced trees using varying angular constants. It can be seen that varying the angular constant influences the grouping of nodes and also the view we get on lower level nodes.

5. Implementation and Experimental Results

We implemented the PhylloTree algorithm using C# and OpenGL as a standard 3D graphics API. We provide lighting and shading as well as interactive zoom, rotation, and

(a) A 5.000 node tree using $\alpha = 93°$. (b) A 11.111 node tree using $\alpha = 12°$. (c) A 105.100 node tree using $\alpha = 9°$.



(d) A 1.000 node tree using $\alpha = 21.5°$. (e) A 5.000 node tree using $\alpha = 57.5°$. (f) A 10.000 node tree using $\alpha = 9°$.

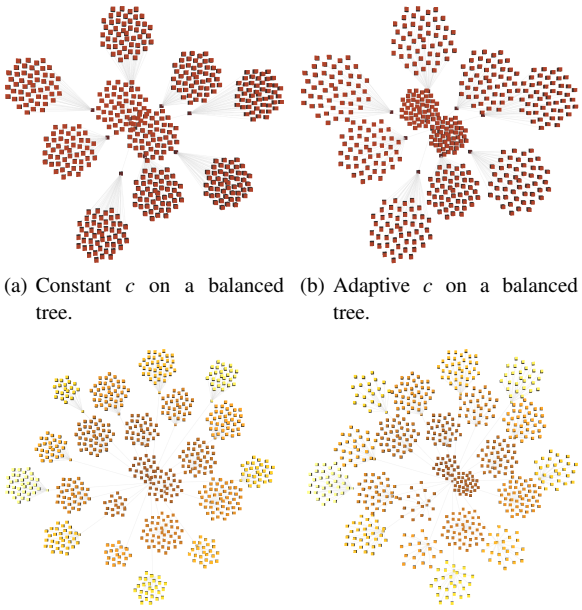
Figure 9: *Several large balanced and unbalanced trees using different angular constants.*

translation for exploration and to help understanding of the displayed tree structure. On a desktop computer (Intel P4 3Ghz) layout generation is instantaneous (<0.1s) for trees of approx. 60.000 nodes with pre-determined node spacing. The layout of a tree with 198.622 nodes took 0.3s to compute with pre-defined spacing. This performance allows responsive relayout and interactive adjustment of the spacing and angle parameters in real time. Individual, adaptive node spacing increases the layout generation time depending on the number of levels in the tree. The bottleneck for interactive rotation, translation, and zooming operations lies in the drawing of the tree. Graphics card properties, display size, as well as options such as lighting and antialiasing influence the possible framerates that can be achieved.

Future work will include the incorporation of this layout into a tree visualization system that supports further operations on trees like searching, labeling, highlighting, or visual comparison of two trees. Also many more three-dimensional mappings and layout variations can be explored. Our current focus is the development of our layout algorithm and the exploration of the applicability of this family of layout variations to different types and different sized trees as follows.

We used different types of datasets to explore our different layout variations. First, we used several balanced and unbalanced automatically generated datasets that ranged in size from small to large. For balanced trees we found aesthetically pleasing results with trees up to approx. 100.000 nodes (cf. Figure 9(c)). Unbalanced trees were also tested. For these trees it is particularly important to be able to spot outliers and trends in the dataset. Figure 9 (d)–(f) shows three medium to large unbalanced tree drawings. It can be seen that clusters and outliers can be readily identified.

We also used real-world data sets to test even larger unbalanced tree structures. Figure 10 shows four datasets from the Infovis 2003 Contest [Inf05]. Figure 10(a) and 10(b) show the file structure extracted from two log files of the University of Maryland's web server. In the left image we can see that the first two levels of the hierarchy contain approximately the same number of files or folders while a small number of second level folders contains a high number of third level files and folders which each again contain ap-



(a) Constant c on a balanced tree.

(b) Adaptive c on a balanced tree.

(c) Constant c on an unbalanced tree.

(d) Adaptive c on an unbalanced tree.

Figure 8: *Constant vs. adaptively chosen spacing constants.*

proximately two files on the fourth level. We also see that a small number of files are nested deeper than 9 levels from the root. Zooming and rotating enable a closer examination of the tree in 3D leading to further discoveries in the data. In Figure 10(b) we chose a different colour scheme and layout to show different aspects of the data. Each first-level subtree has its own colour. From this and the circular layout we can identify large file clusters and further view manipulation supports the identification of the roots of these subtrees.

Figure 10(c) and 10(d) show two different versions of a scientific classification of living organisms in the Animal Kingdom. The nodes in this tree represent animals and animal families, each with a scientific name and rank. These are nested to show which organisms fall into successively larger named groups. A child node is interpreted as belonging to the group named in its parent node. In the left image we see interesting patterns emerging. It seems that an axis goes through from the root node to the highest levels. At the lower levels most families have few descendants while at one point in the middle and closer to the end we see a high number of leaf nodes emerge, identifying different species. The right images uses a different tree layout algorithm and a different version of scientific classification. Still, groupings of tree families are visible and could be identified with closer examination.

In summary, interesting questions that can be answered about these data sets are questions about subtree size and general patterns emerging in the tree. Through interactive manipulation of the view parameters more knowledge about the dataset can be gathered than from looking at a static image.

6. Conclusion

We introduce the use of phyllotactic patterns to create tree layouts. PhylloTrees offer a readily available family of layouts through adjustments of two parameters: the angular constant and the spacing constant. The phyllotactic angular constant of 137.5° offers optimal packing with an expandable layout but other angular constants offer different but equally interesting layouts. Other parameters include the choice of mapping from a two-dimensional phyllotactic pattern to the three-dimensional tree layout. Since in information visualization the focus is often on creating visualizations for data where the exact number of nodes is not known a priori, algorithms that can handle increases and decreases in the number of nodes gracefully are an asset.

Acknowledgments

We thank Tobias Isenberg for fruitful discussions about the layout and graphics implementation. We also gratefully thank our funding providers: Alberta Ingenuity, Alberta's Informatics Circle of Research Excellence (iCORE), Natural Sciences and Engineering Research Council (NSERC).

References

[CA04] CARPENDALE S., AGARAWALA A.: Interactive Poster: PhylloTrees: Harnessing Nature's Phyllotactic Patterns for Tree Layout. In *IEEE Infovis, Poster Compendium* (2004), IEEE Press, pp. 7–8.

[Eri83] ERICKSON R. O.: *The Growth and Functioning of Leaves*. University Press, Cambridge, 1983, ch. The Geometry of Phyllotaxis, pp. 53–88.

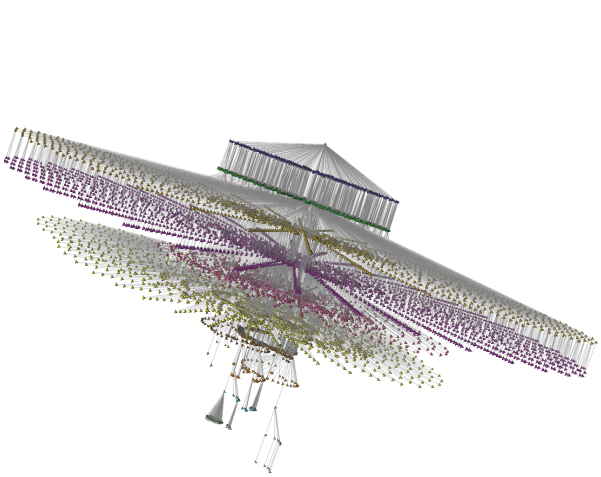
[HM04] HONG S.-H., MURTAGH T.: Visualization of Large and Complex Networks Usin PolyPlane. In *Graph Drawing* (2004), Springer Verlag, Berlin, pp. 471–481.

[HMM00] HERMAN I., MELANÇON G., MARSHALL M. S.: Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Trans. on Vis. and Computer Graphics 6*, 1 (Jan.–Mar. 2000), 24–43.

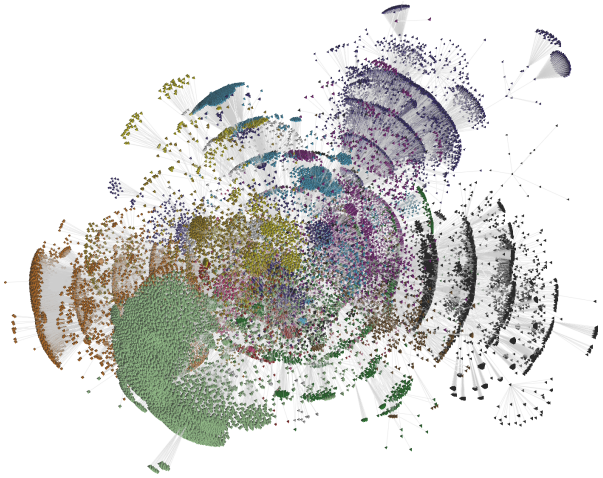
[Inf05] Infovis 2003 contest. Website (`http://www.cs.umd.edu/hcil/iv03contest/datasets.html`), Download: 08/2005.

[Jea94] JEAN R. V.: *Phyllotaxis: A Systematic Study in Plant Morphogenesis*. Cambridge Studies in Mathematical Biology. Cambridge University Press, 1994.

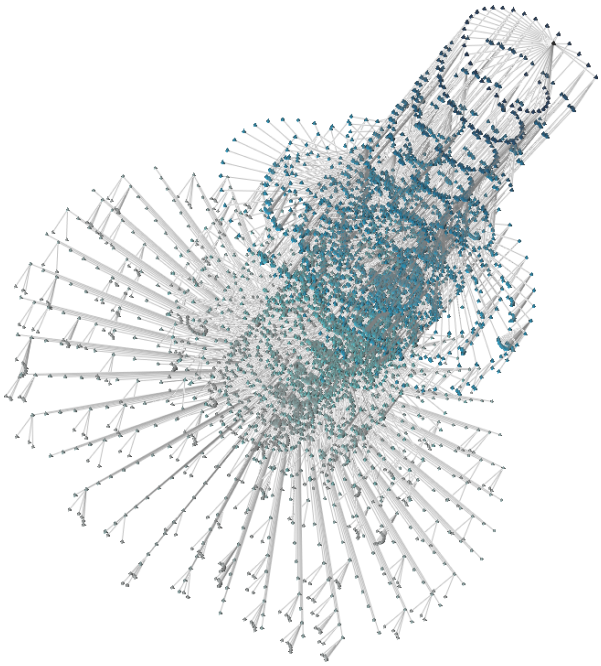
[KvdWvW01] KLEIBERG E., VAN DE WETERING H., VAN WIJK J. J.: Botanical Visualization of Huge Hierarchies. In *IEEE Infovis* (Los Alamitos, CA, 2001), IEEE Press, pp. 87–94.



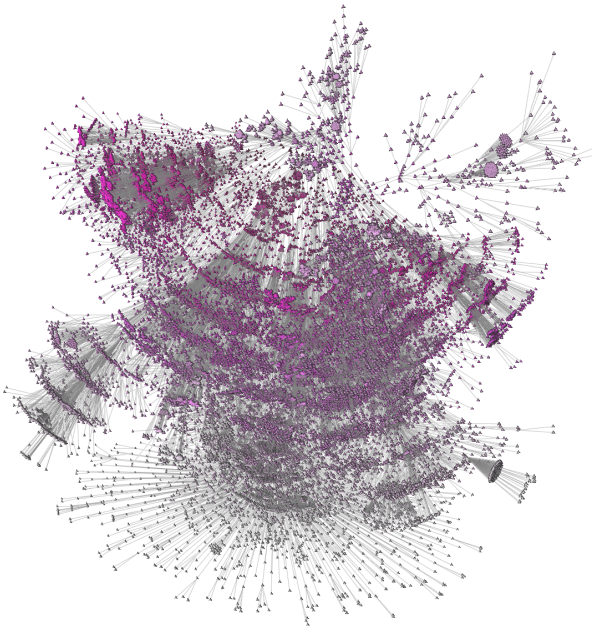
(a) Files on the University of Maryland's webserver; 76.551 nodes, 12 levels.



(b) Files on the University of Maryland's webserver; 76.388 nodes, 12 levels.



(c) Classification of the animal kingdom; 190.264 nodes, 15 levels.



(d) Classification of the animal kingdom; 198.622 nodes, 15 levels.

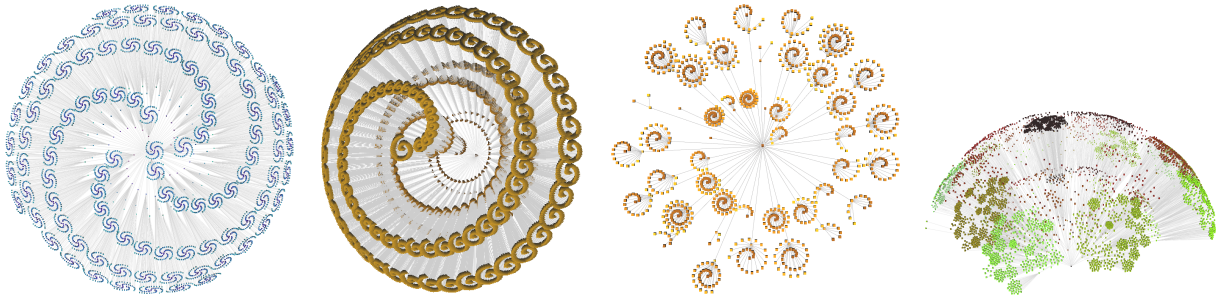
Figure 10: *Several real world examples for large unbalanced trees.*

[KY93] KOIKE H., YOSHIHARA H.: Fractal Approaches for Visualizing Huge Hierarchies. In *IEEE Visual Languages* (August 1993), IEEE Press, pp. 55–60.

[Mun97] MUNZNER T.: H3: Laying Out Large Directed Graphs in 3d Hyperbolic Space. In *IEEE Infovis* (Los Alamitos, CA, 1997), IEEE Press, pp. 2–10.

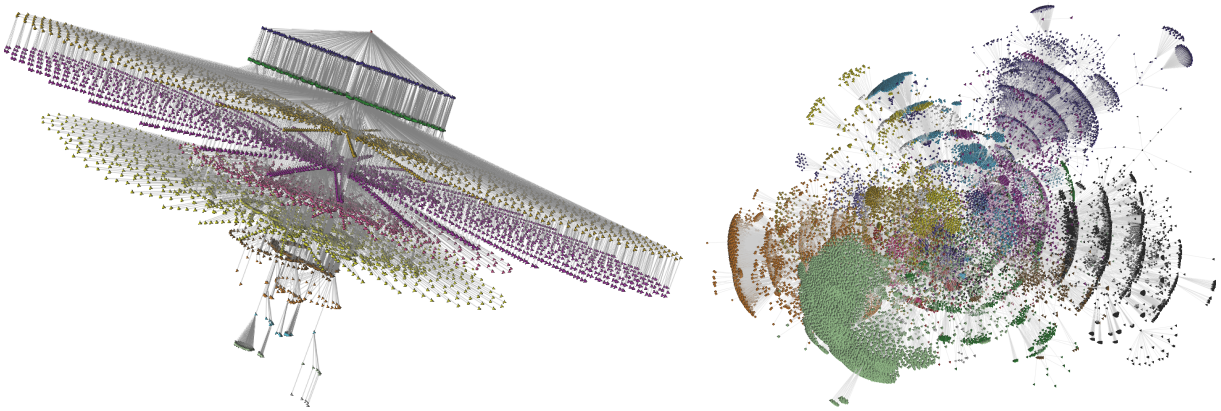
[RMC91] ROBERTSON G. G., MACKINLAY J. D., CARD S. K.: Cone Trees: Animated 3D Visualizations of Hierarchical Information. In *Proc. of CHI'91* (New York, NY, USA, 1991), ACM Press, pp. 189–194.

[Vog79] VOGEL H.: A Better Way to Construct the Sunflower Head. *Mathematical Biosciences 44*, 3–4 (June 1979), 179–189.



(a) 5.000 nodes, $\alpha = 93°$. (c) 105.100 nodes, $\alpha = 9°$. (d) 1.000 nodes, $\alpha = 21.5°$. (e) 5.000 nodes, $\alpha = 57.5°$.

Figure 9: *Several large balanced and unbalanced trees using different angular constants.*



(a) 76.551 nodes, 12 levels. (b) 76.388 nodes, 12 levels.

Figure 10: *Real world examples for large unbalanced trees: files on the University of Maryland's webserver.*