# The Design Space of Implicit Hierarchy Visualization: A Survey

Hans-Jörg Schulz, Steffen Hadlak, and Heidrun Schumann

**Abstract**—Apart from explicit node-link representations, implicit visualizations and especially the Treemap as their frontrunner have acquired a solid position among the available techniques to visualize hierarchies. Their advantage is a highly space-efficient graphical representation that does not require explicit drawing of edges. In this paper, we survey the design space for this class of visualization techniques. We establish the design space along the four axes of dimensionality, edge representation, node representation, and layout by examining existing implicit hierarchy visualization techniques. The survey is completed by casting some light into regions of the design space that have not yet been explored. Our design space is not a mere theoretical construct, but a practically usable tool for rapid visualization development. To that end, we discuss a software implementation of the introduced design space.

**Index Terms**—information visualization, hierarchy visualization, treemaps, visualization design space, rapid visualization prototyping.

## 1 Introduction

In times where typical data sizes grow much faster than the available screen sizes, space-efficient visualization techniques are receiving increased attention [1]. For hierarchical data, the parent-child relations of nodes in the hierarchy are an important aspect to be visualized. We call techniques that explicitly show these relations as straight lines, arcs, or curves explicit tree visualizations [2]. In contrast to that, implicit tree visualizations are potentially more space efficient because they resort to an implicit representation of parent-child relations by positional encoding of nodes: for instance by node overlap or containment [2]. In the last 25+ years, a wealth of implicit visualization techniques have been proposed. In fact, there are so many that it is impossible to include all of them in this survey. Therefore, instead of giving an overview of all existing implicit hierarchy visualizations, this survey establishes a systematic design space that includes all possible techniques – existing ones as well as (yet) non-existing ones.

A design space describes the universe of all possible design choices. The most prevalent strategy to define a design space is to extract common design principles from a set of existing visualization techniques. Once identified, these common principles can be used as axes to span the design space. Despite not being a space in the strict mathematical sense of a vector space, it serves as an established analogy in many fields. An extensive example is given by Bugajska's framework [3]. This approach addresses first and foremost the need for a theoretical classification of existing techniques. Beyond that, visualization design spaces enable us to capture some of the tacit knowledge of visualization designers. Once made explicit, design spaces can be used to access an otherwise diffuse and large set of visualization techniques:

- individual visualization techniques can be pinpointed to a distinct position, whereas classes of visualizations can be understood as regions or subspaces within the design space,
- subspaces can be mapped to problem/data domains [4], and design guidelines leading to expressive, effective, and aesthetically pleasing regions of the design space can be identified (as done by Bertin and Tufte [5], [6]),
- the design process can be seen as directed "navigation" or undirected "exploration" of the design space, where each displacement in the design space corresponds to a change of the visualization.

In this paper, we strive for a less theoretical definition, one that allows us to practically realize the entire design space, making it possible to actually browse the design space and providing *rapid visualization prototyping* for implicit hierarchy visualization. The ideas for this design space stem from a design space for 3-dimensional Treemap visualizations [7]. Treemaps are widely used in many application domains today. In fact, this visualization approach seems to be so convincing that people have even started to utilize it for non-hierarchical, multivariate data by applying an OLAP-like hierarchical structuring of the data, as discussed in [8], [9]. As Treemaps are a specific instance of implicit hierarchy visualizations, the Treemap design space is merely a subspace of the one presented here.

To establish the broader design space for implicit techniques, we examined implicit techniques from more than 25 years of visualization research. We extracted common visualization principles that span the design space covered in Section 2. To show that our proposed design space can be utilized not only for placing exist-

• *Hans-Jörg Schulz, Steffen Hadlak, and Heidrun Schumann are with the Department of Computer Graphics, University of Rostock, Germany E-mail: {hjschulz,hadlak,schumann}@informatik.uni-rostock.de*

*Manuscript received August 21, 2009; revised March 10, 2010.*

ing techniques, but also to explore new techniques, we give a few examples of novel implicit visualizations in Section 3. Our software toolkit implementing the design space is briefly presented in Section 4. Finally, Section 5 concludes the paper and points out future research directions. An appendix chronologically assembles examples of all implicit hierarchy visualizations considered in this paper as Figures 12 and 13.

## 2 Design Space Definition

The design space will mainly be discussed along the lines of the two most prevalent implicit visualization techniques: Treemaps (Figure 12c) and Icicle Plots (Figure 12b) [10]–[12]. Most of the implicit visualization techniques relate to one, if not both of them, other implicit visualizations are extremely rare. They are more of theoretical interest and can represent only certain types of trees, as it is the case, e.g., for *unit rectangle-visibility representations* [13]. Because of that, they are only of limited interest for the visualization community and we do not discuss them in detail.

**Treemaps** originate from the *Euler-Venn-Diagrams* that have been known and used for a long time – actually even before Euler [14]. They were used as graphical representations in formal logic and later in set theory. Since any hierarchy can be represented by nested sets, using and deriving these diagrams for hierarchy visualization seems just logical in retrospect. The parent-child relationship between two nodes is represented through nesting the child inside the parent. The first concrete ideas on how to do that (e.g., the Slice-and-Dice layout) were brought up by Shneiderman and Johnson in 1991. As early as 1993, Johnson outlines many fundamental ideas that were not followed up until much later [15]: elliptical Treemaps (Figures 12f and 13k), 3-dimensional Treemaps using nested cuboids (Figures 12h and 13c) and cylinders (Figures 12g and 13i), and hierarchical pie charts called polar Treemaps (Figures 12e and 12p). Since then, Treemaps have become the most prevalent implicit hierarchy visualization technique. Today, many commercial tools provide a Treemap view as a matter of course: Spotfire®, IBM ILOG Elixir, Hive Group's Honeycomb, and others [16, ch.4.7].

**Icicle Plots** allude to the *Castle Diagrams* (Figure 12a) from the early 1980's, yet similar representations have already been known before them [14], [17]. They are common in the statistical sciences – mainly to depict cluster dendrograms. Icicle Plots represent the parent-child relationship by vertical adjacency: the child nodes are placed right on top of their parent. A 3-dimensional variant using stacked boxes (Figure 12b) was also proposed in 1983 [12]. The 2D visualization is included in some commercial statistics software, such as SPSS.

Hence, Treemaps and Icicle Plots are prominent examples of implicit visualization and therefore uses to illustrate the following discussion of the design space.

### 2.1 Design Space Dimensions

A design space definition tries to identify independent design dimensions that subdivide and span the design space. Building upon the basic design space for Treemaps [7], we identified the following four axes of the design space for implicit tree visualization. Each axis encodes one degree of freedom in the design process:

- **Dimensionality:** 2D or 3D
- **Node Representation:** graphics primitives
- **Edge Representation:** inclusion, overlap, adjacency
- **Layout:** subdivision, packing

The latter three axes can have additional parameters that govern the actual realization in detail. These additional parameters are used to fine-tune a visualization. Examples are the surface properties of the primitives (e.g., color, texture, or transparency), edge representation details (e.g., amount of overlap), and layout constraints (e.g., ordering, or desired aspect ratio).

To understand how the four design dimensions control the visualization outcome, one has to realize the basic conflict that underlies the design of implicit hierarchy visualizations. They try to convey two things at the same time: their hierarchical relation as well as numerical and categorical node attributes. In principle, this can be done by mapping both onto independent visual attributes – the characteristics of the graphical primitives (size, shape, color,...) are used for node attributes, and the relative position of nodes encodes the parent-child relationship. Yet, it is often difficult to show attributes and structure at the same time. Some implicit tree visualizations are better suited for showing node attributes, while others emphasize more on structure.

Examples can be found among the Treemap variations: the *Squarified Treemap* encodes a node attribute by node size and facilitates value comparison by aiming for aspect ratios close to 1 [18]. Furthermore, by it is self-evident that a non-leaf's attribute value is the aggregate of its children's attribute values, because the children together occupy the exact same area as the parent. While here the hierarchical structure of the data can still be discerned, the focus of this technique lies clearly on representing the node attributes. On the other hand, there are for example *Cascaded Treemaps* (Figure 13p) that put more emphasis on the hierarchical structure [19]. This is achieved by leaving a small unoccupied boundary between a node and its children, and by using an overlap relation instead of nesting. This enhances the perception of the actual tree structure, as it conveys a 2.5D impression and maintains partial visibility of lower levels of the visualization (higher levels of the tree). Variations such as Squarified Treemap that enhance the perception of the tree structure are shown in Figure 1. Here, the original visualization (a) was altered along the design dimensions ((b) dimensionality, (c) edge representation, (d) node representation, and (e) layout).

The individual axes of the design space and their parametrizations are discussed in the following sections.

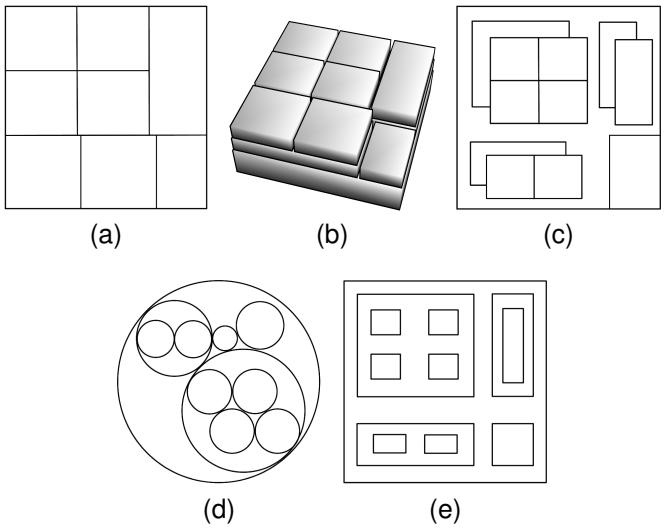



Fig. 1. (a) Original Squarified layout emphasizing the attribute values [18], (b) Steptree – emphasizing the structure by extrusion [20], (c) Cascaded layout – emphasizing the structure by using overlap [19], (d) Circular Treemap – emphasizing the structure by using a non-space-filling primitive [21], (e) Nested layout – emphasizing the structure by leaving white space around the primitives [10].

#### 2.1.1 Dimensionality

Dimensionality is highly controversial in information visualization. Most of the published implicit visualization techniques are 2-dimensional. This is only natural, as they borrow from the *map metaphor* (hence Tree**map**) and maps are in general 2D representations. This universal design metaphor allows the visualization user to think of the data in terms of the metaphor, effectively establishing a mental map of the hierarchy. 2D visualizations have several advantages over 3D representations, including:

- they are suitable for static media (e.g., printouts), as no occlusion can occur in 2D that would require interactive view manipulation,
- they are useful in time-critical situations, as the search space for a node/subtree of interest is only 2-dimensional and no lengthy interactive exploration is needed,
- they perform better than 3D techniques for comparison tasks on node attributes, as areas can perceptually be better compared than volumes.

Hence, many people tend to prefer 2-dimensional representations, as they provide higher information availability than 3D [22]. Commercial visualization tools and practical implementations on the Internet offer 2-dimensional views most of the time.

Yet, it can be seen already from the earliest publications on Icicle Plots and Treemaps that 3-dimensional representations were often considered, too [12], [15]. Many later examples such as Beamtrees or Circular Treemaps do not leave their 3D extensions unmentioned, either [21], [23]. This is again only natural, as 2D representations easily extend to 3D by use of the *cityscape metaphor* or the *container metaphor* [24], [25]. They provide an obvious transition path from the 2-dimensional, basic technique to its 3-dimensional extension. Hence, the existing 3-dimensional techniques are usually adaptations of a 2-dimensional technique.

3D techniques that borrow from the cityscape metaphor typically extrude a flat 2-dimensional visualization into the third dimension. This applies to Treemaps as well as to Icicle Plots. Examples for extruded 2D Treemaps are Information Pyramids (Figure 12k), 3D Nested Treemaps (Figure 12n), StepTrees (Figure 13d), and even nested Hemispheres for circular Treemaps (Figure 13e). Icicle Plots have been adapted likewise to 3D Icicle Plots (Figure 12b) [12], [20], [26]–[28]. They have in common that they stack node primitives on top of each other, giving the graphical representation a recognizable shape and allowing an intuitive exploration from a bird's-eye perspective.

In the second case, 3D-techniques rely on the container metaphor. They mostly extend a known 2-dimensional subdivision or packing algorithm to the third dimension. Here only Treemap examples are possible, as Icicle Plots and Sunbursts do not rely on nesting in the first place. The most prominent 3D Treemap examples of this kind are probably Information Cubes (Figure 12i) and Treecubes (Figure 13c) [29], [30]. It is apparent that this latter approach suffers from heavy occlusion, and therefore, can only be used with semi-transparent primitives and for rather shallow hierarchies.

The main argument supporting the use of 3D is that implicit visualizations rely heavily on spatialization, as they encode the edges of the hierarchy into relative positions: the position of one node with respect to another determines their relationship. The additional 3rd dimension opens up new perspectives for this relative positioning. A good example is the Beamtree technique (Figure 13a) [23]. Its 2-dimensional form encodes the parent-child relationship by overlapping rectangles. This leads to occlusion of most of the internal tree structure, whereas the leaves are fully visible. Adding a third dimension allows one to transform Beamtrees into a 3D technique that uses adjacent cylinders instead of overlapping rectangles. The orthogonal view from the front still shows a bottom-up view with the leaves up front, resembling the original 2D technique. Additionally the 3D visualization can be rotated and viewed from the back, yielding a top-down view with the root facing the user, or from the side, revealing the individual levels of the hierarchy.

3D extensions of implicit visualizations come into play when it is required to embed them into 3D applications (e.g., volume visualization or virtual collaborative spaces), or when the visualization designer wants to specifically address the user's spatial memory. Yet, the latter is still somewhat disputed, since there are reports about positive, neutral, as well as negative effects of 2D vs. 3D visualizations [31]–[33]. But as these studies were

conducted for visualization techniques in general, it is arguable whether they are applicable to the concrete case of implicit hierarchy visualizations.

Yet recently, notable attempts have been made to make 3-dimensional implicit tree visualizations (usually Treemaps) available to a wider audience. Examples come from the field of Software Visualization, e.g., the immersive software visualization "CodeCity", which uses a Steptree visualization for program code comprehension and analysis of code evolution and quality, or the Fraunhofer IESE Visual Software Analysis Tool, which likewise uses a 3D Treemap approach for visualizing code quality [34], [35].

#### 2.1.2 Node Representation

In most practical applications, implicit tree visualizations use rectangles for 2-dimensional representations and cuboids for 3-dimensional views. They are easy to stack, easy to nest, and easy to draw with interactive frame rates even with 10,000s of nodes. There are a number of reasons for using alternative shapes of the nodes to something else than rectangles/cuboids.

The first is to improve the aspect ratio of the overall visualization. A common approach is to switch to circles and circle sections, thus achieving an aspect ratio of 1: Treemaps become Circular Treemaps, Pebble Maps (Figure 13b), or Crop Circles (Figure 13h), and Icicle Plots become, e.g., PieTrees (Figure 12o), Sunbursts (Figure 12p), or InterRing visualizations (Figure 12s) [21], [36]–[42].

The second reason is to give each individual node a more recognizable shape by using irregular convex polygons. Hence, Treemaps have been altered into Voronoi Treemaps (Figure 13f) and Circular Partitions (Figure 13q), and the Sunburst technique into the Radial Edgeless Trees (Figure 13m) [43]–[47]. These techniques are not restricted to a certain type of primitive that is then packed as tightly as possible in the available screen space. Instead, the focus lies on the layout algorithm that carves the individual areas of the children out of the screen space occupied by the parent. This carving process usually yields odd-shaped, irregular polygons. The advantage is a distinct shape for each node that makes the whole visualization less uniform and aids in memorizing and recognizing parts of the structure. The obvious disadvantage concerns again the visualization of node attributes: varying shapes are harder to estimate and to compare in terms of their area.

Apart from these two shared reasons, Treemaps and Icicle Plots each have a third reason for switching to non-rectangular shapes. For Treemaps, it is the fact that tightly packed rectangles do not leave any of the underlying structure visible. However, packed circles or ellipses do, thus enhancing the perception of the tree structure [15], [21], [36], [48]. This way, lower levels are visible through gaps in the layout. On the one hand, this facilitates discerning a tree's depth, but on the other hand, less space is available for each level. As trees tend to grow wider with each level, this aggravates any spatial constraints.

For Icicle Plots, the third reason is a different one: as the space on top of the root limits the entire layout from the beginning, the available space for laying out a node's children stays constant, independent of how wide the hierarchy actually grows. Yet, this problem can be somewhat reduced by switching to a radial variant with circle sections as primitives as done for Sunbursts. Here the available space (in this case the circumference) grows with increased distance to the root. Different triangular variants follow the same design argument. Notable examples are the Triangular Aggregated Treemap (Figure 12l), which is a misnomer as it has little to do with the Treemap layout, and Cheops™ (Figure 12j) [49], [50].

Note that a 3D extension of a 2D node representation can usually not be determined unambiguously. For example, a circle can be extended into either a (hemi-)sphere, a cylinder, or a cone, whereas a square can be extended into either a cuboid or a pyramid. For the design space axes to be indeed orthogonal and not to influence one another, it is important to include the disambiguation in the node representation, e.g., by defining three different circle primitives: circle/sphere, circle/cylinder, and circle/cone.

Additional parametrizations may influence the surface properties or rendering style of nodes. Node surfaces can be used for coloring and texturing to either enhance the perception of structure (Cushion Treemaps – Figure 12m, Cushion Icicle Plots – Figure 13l) or the perception of attribute values and changes thereof (Contrast Spiral Treemaps – Figure 13o) [51]–[53]. For 3-dimensional techniques, different rendering styles may be chosen. They can enhance the visualization by providing additional insight, e.g., by applying a wire frame rendering to certain nodes or by adding so-called *Ghost Views* to ensure visibility of otherwise occluded nodes [54].

#### 2.1.3 Edge Representation

Implicit tree visualization techniques represent edges by *inclusion*, *overlap*, or *adjacency* of the graphical objects that represent the tree's nodes. Nodes that have a spatial extent instead of being just points in space may contain or overlap one another. By doing so, they implicitly represent the parent-child relation between nodes of a hierarchy as exemplified in Figure 2. For this relative positioning of the set of graphical objects $O$, a spatial relation $R$ is required that exhibits *acyclicity* (including *irreflexivity*) and *non-convergence*. These conditions make sense, as they directly reflect the structural properties of the tree to be represented:

- acyclicity: a node cannot be its own parent, neither directly (irreflexivity) nor indirectly (acyclicity)
  $\forall a \in O, \forall n \in \mathbb{N}^+ : \neg(aR^n a)$
- non-convergence: a node cannot have two parents, only one – and accordingly two nodes cannot have

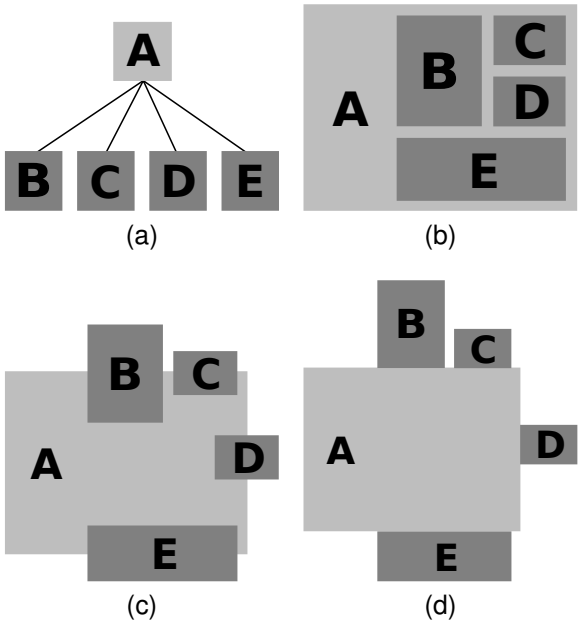



Fig. 2. (a) Explicit, node-link layout, (b) Implicit layout by inclusion, (c) Implicit Layout by overlap, (d) Implicit layout by adjacency.

the same child node
$\forall a \in O \nexists b, c \in O : \ bRa \wedge cRa$

In addition to these necessary conditions, in some cases it may be useful to have *asymmetry* or *transitivity*:

- asymmetry: $\forall a, b \in O : \ aRb \rightarrow \neg(bRa)$
- transitivity: $\forall a, b, c \in O : \ aRb \wedge bRc \rightarrow aRc$

Asymmetry is useful for giving an additional hint on the direction of the edges, pointing from higher levels towards the leaves. This results from the obvious distinction between parent and child, e.g., in the case of $a$ overlapping $b$ it is apparent which one is the parent and which one is the child. This is useful when one has zoomed into the visualization and can see only a portion of it: even though neither root nor leaves might be visible, it is made obvious by the asymmetry in which direction both lie and how the visualization has to be read. Transitivity on top of that enables finding out quickly whether one node is a (distant) ancestor of another node, as this becomes a simple look-up task. It also eases comparison of nodes, as with transitivity they cannot be located too far away, no matter how deep the tree is.

The three commonly used spatial relations fulfill these two conditions, though all differently:

- **Inclusion** guarantees asymmetry and transitivity: if $a$ is nested in $b$, it cannot be the other way around. Also, if $a$ is nested in $b$ and $c$ is nested in $b$, it is obvious that $c$ lies within $a$, too.
- **Overlap** guarantees only asymmetry: if $a$ overlaps $b$, $b$ cannot overlap $a$. Transitivity does not hold for overlap.
- **Adjacency** guarantees neither: if $a$ is adjacent to $b$, this relation is symmetrical. Also, it is not transitive.

The only missing combination is a spatial relation that is transitive and symmetrical at the same time. Yet, it can easily be shown that this case cannot occur, as asymmetry is a necessary condition for transitivity. If it was not $\exists a, b \in O$ with $aRb, bRa$, which (because of transitivity) leads to $aRa$, and thus violates the acyclicity/irreflexivity of $R$.

Adjacency reveals more of the hierarchical structure than inclusion as each individual level is visible and depth equals distance. Yet, inclusion has a very useful property which adjacency does not have: it does not grow outwards and one can be sure that it only uses the amount of space dedicated to display the root node. All other nodes will be placed within this space – and not attached to it. Overlap tries to find a good compromise between both by showing more of the structure, while not growing outwards too much.

As it is mainly the different edge representation that discerns Treemaps from Icicle Plots, there is not much diversity to be expected for this design dimension: Treemap variants use inclusion by default and in very few cases overlap – 2D Beamtrees and Cascaded Treemaps [19], [23]. Icicle Plots and related techniques use adjacency in all cases. Up to now, there seems to be not a single Icicle Plot variant that uses overlap – even though this would be a valid option. Overall, it can be stated that inclusion is the most common among the implicit techniques in use today.

As an interesting side note: In 2004, IBM has filed a patent application in the US for providing "an efficient way of presenting hierarchical data or information based on a container metaphor" [55]. It states further: "Nodes or information associated with the hierarchy are represented visually using a geometric shape, such as a rectangle, square, or circle. The hierarchical relationships of the data in the hierarchy can then be represented by displaying the shapes within one another to illustrate a container relationship or adjacent to each other to represent a different level of hierarchy." This would leave overlap as the only unpatented edge representation.

#### 2.1.4 Layout

In general, one can discern two major layout methodologies: *subdivision* and *packing*. Subdivision is applied starting from the root. It takes the space assigned to a node and subdivides it into regions for the children of this node. This method is applied recursively to the next level of the hierarchy. Packing goes the other way around: starting from the leaves, it determines the shapes and sizes of the nodes according to their attribute values and then attempts to pack sibling nodes tightly into their parent's space. Even though packing is known to be NP-complete by reduction to the Bin Packing Problem [56], this is only valid for the optimal solution. If a good approximation is enough, as it is usually the case for visualizations, fast heuristics can be applied for packing the objects. An example is given in [57]

which is used by the visualization method Data Jewelry Box (Figure 12r) [58]. Subdivision uses the space fully, whereas packing tends to leave gaps in between. The pros and cons for each are exactly the same as already discussed for node primitives that cover their ancestors completely (e.g., rectangles) or do not (e.g., circles): the latter allows for a glimpse at the structure through the gaps, but sacrifices at each level some of the available space.

All known layout methods fall into one of these two categories, even though it may not always be obvious. This is sometimes the case for extruded/stacked techniques. In 2D, these are for example Sunburst or Icicle Plots, which are in fact 1-dimensional subdivision techniques of the perimeter of circle(-segments) or of one side of a rectangle, respectively. In 3D, this can be observed for Steptrees (Figure 13d) and 3D Circular Treemaps (Figure 13i), which are actually a 2-dimensional subdivision and packing technique, respectively [20], [21]. This shows that the user always has more than one option if the tree structure needs to be emphasized: instead of changing the layout from subdivision to packing, one can still maintain the subdivision and extrude the layout into another dimension to achieve the same goal.

Additionally, it is usually the layout algorithm that has to ensure that certain external constraints are fulfilled. Wattenberg introduces four fundamental constraints to be met by a "perfect" layout [59]:

- *Stability*: small changes in the data should only result in small changes in the visualization,
- *Split Neutrality*: structural changes should only affect the parent-region they occur in,
- *Order Adjacency*: if defined, an ordering of the nodes should be maintained,
- *c-Locality*: the representation should be as compact as possible (aspect ratio $\approx 1$).

While variants of Icicle Plots for instance handle ordering by design, other constraints must be explicitly enforced in the layout. Different layouts put a different emphasis on these constraints. Some of them are mostly concerned with order adjacency (Spiral Treemap layout – Figure 13o, Ordered Treemap layout), others with $c$-Locality (Squarified Treemap) [18], [53], [60]. Wattenberg introduced the *Jigsaw Map* (Figure 13g) as one possible realization that fulfills all of the above constraints [59].

Apart from that, the layout may be required to fulfill additional application-specific demands. This is the case for instance for *2-dimensional order adjacency* as realized for cartographic contexts in Spatially Ordered Treemaps, or when dealing with *unusual aspect ratios* as in Treemap-Bars, which embed Treemaps into bar charts [61], [62].

Even though the layout is an independent dimension of the design space, we admit that there can be some interplay with the choice of the node representation. While packing layouts can (in theory) handle all kinds of primitives, it gets a little more complicated for subdivision layouts. In this case, the layout actually shapes the node representation by subdividing the space. Hence, an independent choice of node primitive and layout is hardly possible. If a shape is chosen other than the one that is generated by the layout, it must be scaled and packed inside the space generated by the layout.

## 2.2 Design Space Discussion

Bringing the discussed design dimensions together to form the actual design space raises several concrete questions that are worth investigating. First and foremost, the question is whether the overall design space is complete and consistent, two important properties a design space should fulfill. The required practical utility of the design space beyond pure classification of existing techniques imposes additional design issues.

### 2.2.1 Completeness and Consistency

Telling whether a design space definition is *complete* is a complicated task. There may very well be undiscovered or unpublished implicit hierarchy visualizations that are consistent with the design space definition, but that are so awkward that our proposed design space definition does not include them (yet). Hence, we assume a design space definition to be complete, if it contains the known implicit techniques as of today. If in the future an implicit visualization is found that is not yet covered by the design space, it needs to be expanded. While completeness is hard to prove, in Table 1 we show that it is possible to position the majority of today's implicit hierarchy representations. Even though these are of course not all existing implicit techniques, being able to place them in the design space is already a solid indication for the design space's completeness.

While an incomplete design space would be the result of a design space definition that is too rigid, an inconsistent design space usually results from a definition that is too broad and may go well beyond the design space originally intended. We consider a design space definition for implicit techniques as *consistent*, if it does not violate the basic design principle of implicit visualizations: representing the hierarchy without drawing edges. This excludes two kinds of visualizations:

- any detached arrangement of graphics primitives that does not represent the hierarchical structure at all, as there is no apparent spatial relation between them. An example is the Graph Signature Visualization shown in Figure 3(a) [63]. This visualization uses a scatterplot to show different node classes. Even though they do not include any links between the nodes, these visualizations cannot be considered implicit, because the hierarchical structure is completely lost in the mapping step and cannot be discerned from the representation any more.
- any visualization that contains explicit links. An example for a hierarchy visualization of this category are the Information Slices depicted in Figure 3(b) [64]. Even though they are an implicit visualization techniques at their core (a semi-circular

| | 2D | 3D | Adjacency | Overlap | Inclusion | Rectangles/Cuboids | Squares/Pyramid (Frustums) | Triangles/Trapezoids | Polygons | Circle(-segments)/Cylinders | Circle(-segments)/Spheres | Ellipses/Ellipsoids | 1D Subdivision | 2D Subdivision | 3D Subdivision | 1D Packing | 2D Packing | 3D Packing | Year Published | Figure in the Appendix |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Treemap [10],[11] | ■ | | | | ■ | ■ | | | | | | | | ■ | | | | | 1991 | 12(c) |
| Contrast Spiral Treemap [53] | ■ | | | | ■ | ■ | | | | | | | ■ | | | | | | 2007 | 13(o) |
| 2 1/2D Treemap [65] | ■ | ■ | | | ■ | ■ | ■ | | | | | | | ■ | | | | | 1992 | 12(d) |
| Polar Treemap [15] | ■ | | | | ■ | | | | | ■ | ■ | | | ■ | | | | | 1993 | 12(e) |
| Jigsaw Map [59] | ■ | | | | ■ | | | | ■ | | | | ■ | | | | | | 2005 | 13(g) |
| Generalized Treemap (Pie) [8] | ■ | | | | ■ | | | | | ■ | ■ | | | ■ | | | | | 2006 | 13(j) - left |
| Generalized Treemap (Pyramid) [8] | ■ | | | | ■ | | | ■ | | | | | | ■ | | | | | 2006 | 13(j) - middle |
| Generalized Treemap (Pie+Pyramid) [8] | ■ | | | | ■ | | | | | ■ | ■ | | | ■ | | | | | 2006 | 13(j) - right |
| Tree Cube [30] | | ■ | | | ■ | ■ | | | | | | | | | ■ | | | | 2003 | 13(c) |
| 3D Treemap [15] | | ■ | | | ■ | ■ | | | | | | | | | ■ | | | | 1993 | 12(h) |
| Cushion Treemap [51] | ■ | | | | ■ | ■ | | | | | | | | ■ | | | | | 1999 | 12(m) |
| Voronoi Treemap [44] | ■ | | | | ■ | | | | ■ | | | | | ■ | | | | | 2005 | 13(f) |
| Circular Partitions [45] | ■ | | | | ■ | | | | ■ | | | | | ■ | | | | | 2008 | 13(q) |
| Quantum Treemap [66] | ■ | | | | ■ | ■ | | | | | | | | ■ | | | ■ | | 2001 | 12(q) |
| Data Jewelry Box [58] | ■ | | | | ■ | ■ | | | | | | | | | | | ■ | | 2002 | 12(r) |
| Cascaded Treemap [19] | ■ | | | ■ | | ■ | | | | | | | | ■ | | | | | 2008 | 13(p) |
| Ellimap [48] | ■ | | | | ■ | | | | | | | ■ | | ■ | | | | | 2007 | 13(k) |
| Treemaps with Ovals [15] | ■ | | | | ■ | | | | | | | ■ | | ■ | | | | | 1993 | 12(f) |
| Pebble Map [36] | ■ | | | | ■ | | | | | ■ | ■ | | | | | | ■ | | 2003 | 13(b) |
| CropCircles [37] | ■ | | | | ■ | | | | | ■ | ■ | | | | | | ■ | | 2006 | 13(h) |
| Lifted Treemap [68] | | ■ | | | ■ | ■ | | | | | | | | ■ | | | | | 2009 | 13(r) |
| 3D Nested Treemap [25] | | ■ | | | ■ | ■ | | | | | | | | ■ | | | | | 1999 | 12(n) |
| Information Cube [29] | | ■ | | | ■ | ■ | | | | | | | | | | | | ■ | 1993 | 12(i) |
| Nested Columns [15] | | ■ | | | ■ | | | | | ■ | | | | | | | ■ | | 1993 | 12(g) |
| 2D Icicle Plot [12] | ■ | | ■ | | | ■ | | | | | | | ■ | | | | | | 1983 | 12(b) - left |
| Castles [17] | ■ | | ■ | | | | | ■ | | | | | ■ | | | | | | 1981 | 12(a) |
| Cushioned Icicle Plot [52] | ■ | | ■ | | | ■ | | | | | | | ■ | | | | | | 2007 | 13(l) |
| Triangular Aggregated Treemap [48] | ■ | | ■ | | | | | ■ | | | | | ■ | | | | | | 1998 | 12(l) |
| Sunburst [40] | ■ | | ■ | | | | | | | ■ | ■ | | ■ | | | | | | 2000 | 12(p) |
| Interring [42] | ■ | | ■ | | | | | | | ■ | ■ | | ■ | | | | | | 2002 | 12(s) |
| PieTree [38] | ■ | | ■ | | | | | | | ■ | ■ | | ■ | | | | | | 2000 | 12(o) |
| Radial Edgeless Tree [46],[47] | ■ | | ■ | | | | | ■ | | | | | ■ | | | | | | 2007 | 13(m) |
| Steptree [20] | | ■ | ■ | | | ■ | | | | | | | | ■ | | | | | 2004 | 13(d) |
| 3D Icicle Plot [12] | | ■ | ■ | | | ■ | | | | | | | ■ | | | | | | 1983 | 12(b) - right |
| Nested Hemispheres [26] | | ■ | | | ■ | | | | | | ■ | | | | | | ■ | | 2004 | 13(e) |
| 3D Nested Cylinders and Spheres [21] | | ■ | ■ | | | | | | | ■ | ■ | | | | | | ■ | | 2006 | 13(i) |
| 3D Sunburst [70] | | ■ | ■ | | | | | | | ■ | | | | ■ | | | | | 2007 | 13(n) |
| 3D Beamtree [23] | | ■ | ■ | | ■ | | | | | ■ | | | ■ | | | | | | 2002 | 13(a) |
| Information Pyramids™ [28] | | ■ | ■ | | | | ■ | | | | | | | ■ | | | | | 1997 | 12(k) |
| Cheops™ [50] | ■ | | ■ | | | | | ■ | | | | | | | | ■ | | | 1996 | 12(j) |

TABLE 1
Classification of some existing implicit tree visualization techniques, approximately ordered from the most attribute-centric technique at the top to the most structure-centric one at the bottom.

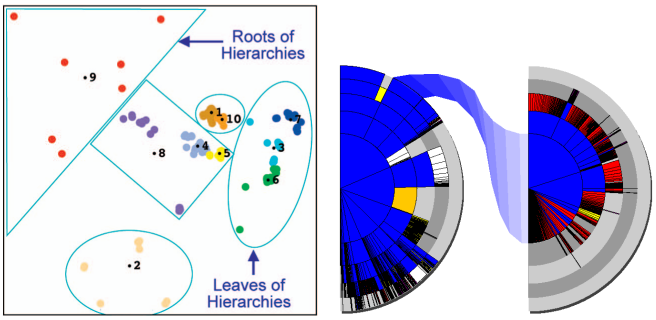



Fig. 3. Examples of hierarchy visualizations that are not part of the design space: (a) Graph Signature Visualizations and (b) Information Slices [63], [64].

Sunburst variation), they use explicit links to connect multiple, otherwise separate parts (slices) of their implicit hierarchy visualization and are thus not a part of our design space.

The consistency is basically ensured by the way the design space is constructed as it does not provide any explicit edge representations (fulfilling the second condition) but forces the layout to use at least one of the available edge representations (adjacency, overlap, or inclusion) and to fulfill acyclicity and non-convergence. This effectively prevents any detached arrangement and arrangements that do not represent a tree (fulfilling the first condition).

Having a complete and consistent design space does not prevent the definition of mostly "useless" visualization techniques. Figure 4 shows an example of a visualization design that is so heavily occluded by its own layout that it obscures more of the data than it actually shows. This is a problem faced by all vast design spaces: it is easy to get lost in them and hard to find the subspaces of useful techniques. Yet, if the designer wants to explore only the known design decisions, a list of the existing techniques as given in Table 1 would be sufficient. We do not see the ability and flexibility to roam freely in the design space as a burden, but rather as a chance to discover completely new techniques. Also, we found it very educational to actually see when and why well-known techniques break down when altering them along certain design dimensions. It may even be the case, for special input data (e.g., binary trees of bounded height), that useless techniques "degenerate" into good visualizations. This can be taken further to actually tailor a visualization to a concrete data set and/or task by incremental refinement of its design.

Another implication of a complete design space is that it allows the definition of impossible combinations of design parameters, for which no reasonable practical realization exists. This occurs when at least two of the design decisions made are incompatible – for instance when the area provided by the layout method and the (base) area of the child do not match. This is always the case when one does not respect the dependencies between layout and node representation mentioned at the end of Section 2.1.4. An example would be the use of a Voronoi Treemap layout with circles instead of the convex polygons provided by the layout. This example is not far fetched, as similar, incompatible combinations of graphics primitives are frequently used under the term "Circle Maps" in cartography. Since our flexible design space definition does allow such incompatible combinations, a way must be found to handle these. Because of the sheer number of possible but incompatible combinations, this problem is far too complex to be solved elegantly in the conceptual design space. Instead, a practical solution would be to require each node primitive to provide mechanisms to adapt to a possibly incompatible area.

In our concrete case, each node primitive must provide methods that compute an inscribed as well as a circumscribed circle/sphere and rectangle/cuboid. They are used to interface between a node primitive and an incompatible primitive provided by a layout either by inscribing into the primitive generated by the layout or by circumscribing the node primitive – both with the same interface: either circle/sphere or rectangle/cuboid. They can then be put together easily. The inscription mode to use (circle/sphere or rectangle/cuboid) is the one that maximizes the resulting area/volume occupied by the inscribed primitive. This pragmatic approach does not guarantee an optimal usage of the space and may not in all cases lead to aesthetically pleasing results. Yet, it effectively bridges the gap between the general design space and a prototypical realization that allows one to put together new visualization techniques.

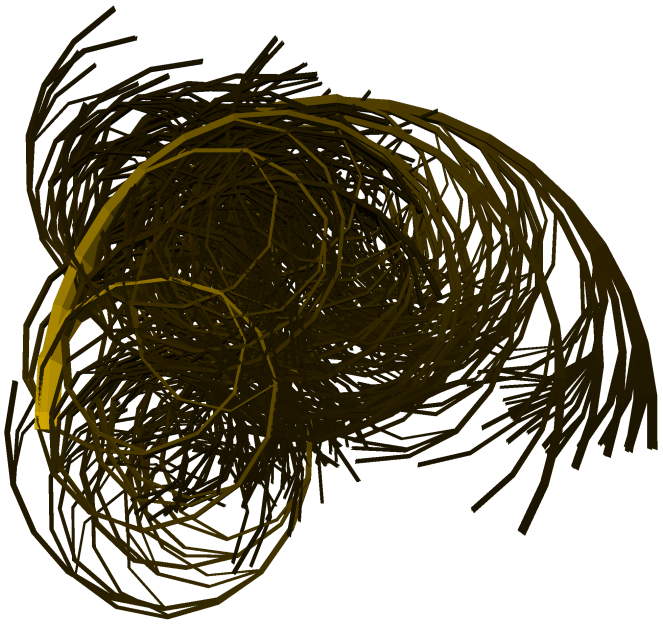


Fig. 4. The Spiral Steptree from Section 3.1.3 (cp. Figure 7) where the layout was changed to a Strip Treemap Layout, rendering it practically useless.

#### 2.2.2 Design Issues

To be applicable as a theoretical framework for the classification of visualization techniques and as a practical tool that aids in the design of new visualization, a number of issues are worth mentioning:

**Mixing Design Choices.** The design space has so far been applied to the tree as a whole. Yet, a global scope for design decisions is neither necessary nor useful. In fact, there are a number of *mixed implicit visualizations* that cannot be expressed without a more refined, local way to specify the visualization design. The term *mixed* was first used for *mixed Treemaps* by Vliegen et al. for their Generalized Treemaps (Figure 13j) in which the authors use a Slice-and-Dice layout for the upper levels of a hierarchy and the squarified layout for the lower levels [8]. The concept of mixing different design choices, be it layouts or node primitives, has been known (but not been named) well before that. It can also be observed for:

- 2 1/2D Treemaps [65] (Figure 12d) mixing a 2D representation for the inner nodes with 3D pyramid-like graphics objects for the leaves
- Quantum Treemaps [66] (Figure 12q) mixing a subdivision layout for the inner nodes with a packing layout for the leaves
- 3D circular packing [21] (Figure 13i) mixing cylinders for the inner nodes with hemispheres for the leaves
- 2D Beamtrees [23] mixing overlap for the inner nodes with inclusion for the leaves
- 3D Beamtrees [23] (Figure 13a) mixing adjacency for the inner nodes with inclusion for the leaves

Altering design choices for leaf- and non-leaf-nodes is the most common form of mixing. This is either because they are of different types (e.g., directories vs. files) or just to help discerning them. As mentioned in [8], [9], design choices can be made on a per-node basis. This allows us to map not only structural node properties (e.g., is_leaf, or depth) to design parameters, but also derived properties, such as Strahler numbers or other numerical or categorical attributes [67]. Mixed representations can also be used to emphasize entire subtrees, as it is done in [68] (Figure 13r) by adding orthogonal sub-Treemaps on top of a 2-dimensional base Treemap.

**Positioning in the Design Space.** The design dimensions are sufficient to be used for general discussions. For a practical application their specification needs to be refined. The reason is that they are not concrete enough to define each and every aspect of a technique to clearly position it in the design space. This is only natural, as a design space should first and foremost serve as a clear mental map for a class of visualizations. Parameters that describe every tiny facet of a visualization's appearance would only obfuscate this map. Yet, for a concrete realization of the design space these additional clarifications are needed to resolve otherwise unspecific design decisions. An example is that the choice of a node primitive does not decide on where exactly to attach the child nodes in case of adjacency – for a cylinder it can be its flat top (3D Circular Treemap) or its curved coat (3D Beamtree). Consequently, before the design space can be realized, missing details like this must be added to its definition. In case of attaching 3D primitives, one needs to specify three additional attributes for the primitives: the surface onto which to attach the children, the surface normal to indicate which side to attach to, and the anchor point for connecting the parent.

**Displacement in the Design Space.** The notion of individual visualization techniques being concrete positions within a design space allows us to establish a notion of similarity between techniques, even though it is hardly possible to define a distance metric in the mathematical sense. Yet, each change in design decisions can be understood as a displacement in the design space, which in turn provides a measure of similarity. The number of changes needed to convert one technique into another one gives a sense of how much they have in common and thus how close they are conceptually – very much like edit distances. This is quite useful for devising user studies for a newly developed technique, as it allows finding established techniques that are close to the new technique as candidates for comparison. As an example, the conversion path between 2D Polar Treemaps and Information Pyramids shows that all four design axes are needed to transform one into the other. While this might not be the shortest or most direct path within the design space, it is one that only uses known techniques as intermediary steps:

1) 2D Polar Treemaps: change relationship from nesting to adjacency – yields: 2D Sunburst
2) 2D Sunburst: change primitive from circle(-section) to rectangle – yields: 2D Icicle Plot
3) 2D Icicle Plot: change dimensionality from 2D to 3D – yields: 3D Icicle Plot
4) 3D Icicle Plot: change layout from horizontal slicing to squarified – yields: Steptree
5) Steptree: change primitive from cuboid to pyramid frustum – yields: Information Pyramids

**Ranges in the Design Space.** The design space can be used to trace different design alternatives across the entire design space. One example for such a design alternative is the mentioned conflict between attributes and structure. As it is often impractical to decide for either one, the resulting visualization should rather be a good compromise between both. Techniques that focus solely on either one can be understood as the endpoints of this particular range. An implicit visualization can be moved in this design rage to achieve certain goals:

- focus on attributes: Squarified Treemaps – a 2D inclusion technique using rectangles of aspect ratio close to 1 and a space-filling subdivision layout
- focus on structure: Cheops$^{TM}$– a 2D adjacency technique that uses overlapping triangles of uniform size and a nested packing technique with lots of white space to visualize paths and subtrees

A prominent application example for a visualization that focuses almost entirely on attributes rather than on hierarchical structure is Wattenberg's *Map of the Stock Market* [69]. In this case, the focus is only natural, as the hierarchy in question has only 3 levels, which are easy to overview: the whole stock market (root), the industrial sectors (internal nodes), and the individual companies (leaves).

Within the range between these endpoints lie all the intermediary compromises between showing values and showing structure. Table 1 is approximately sorted along this spectrum with the attribute-oriented (squarified) Treemap technique at the top and the structure-oriented Cheops™ at the bottom. Another example would be the range of how siblings are spatially related to each other: all the way from no apparent spatial relation (as in the original Treemap), through a certain degree of co-location (as in Wattenberg's Stock Market Treemap layout, later called Cluster Treemap), to a linear order (as in Jigsaw Maps or Spiral Treemaps) and finally to a direct overlap (as in Cheops™).

## 3 Generating New Techniques

Having surveyed the existing techniques in Section 2, we now fill in some of the gaps that have been left unexplored and unpublished so far. As the number of unexplored implicit visualization techniques is of course infinite, because there are endless ways of putting together and mixing numerous graphics primitives in 2D and 3D, we can only discuss a few examples of not yet published techniques. However, we want to use these examples to highlight three important design strategies that in many cases lead to useful and aesthetically pleasing visualization results:

- seek unexplored regions within the existing design space, e.g., novel combinations of visualization parameters,
- establish new connections between different regions of the design space, e.g., by mixing design choices,
- find a novel parametrization of an otherwise fixed design, e.g., by using derived measures.

All three approaches target a different part of the design space: the first uses the design axes to construct a new technique, the second utilizes the possibility to combine multiple designs into a new one, and finally the third explores the parameter space of a visualization design to the same end. Since these three strategies are independent of each other, they can also be used in conjunction if needed. Each of them is illustrated by selected examples in the following.

In these examples, we focus on different node primitives, because they largely determine a visualization's appearance. In many cases techniques are even named after the primitive used: e.g., Information **Pyramids**, Information **Cube**, Tree **Cube**, Crop**Circles**, **Elli**maps. As pointed out in Section 2.1.2, a number of design considerations can be addressed just by the choice of an appropriate primitive. This section discusses aspects of the node representation along the lines of the three points listed above: discussing novel node primitives in Section 3.1, mixing different node primitives in Section 3.2, and parametrizing node primitives according to derived measures in Section 3.3.

### 3.1 Novel Combinations of Visualization Parameters

Seeking out unexplored regions of the design space to generate new techniques is at first nothing else than deciding for (a combination of) values of the individual design axes that no one had thought about before. These visualization parameters can be, for instance, a certain layout or a specific node primitive. Once identified, they span a subspace within the design space that contains novel visualization techniques. Yet chances are that the found visualizations will be not very useful. Careful parametrization, testing with different data sets, and subsequent evaluation are necessary before a ready-to-use technique will emerge.

In this section, we pick a few noteworthy techniques from the so far largely unexplored design subspace of spherical and cylindrical node primitives. As a prerequisite for combining these primitives with all other visualization parameters (e.g., edge representation or layout), it is necessary to make all possible cut sections of these primitives available. Otherwise, for instance, a Slice-and-Dice layout will not work as expected. The 3-dimensional sections for our two examples are shown schematically in Figure 5, which lists all the different polar cuttings of a cylinder and a sphere. Sphere and cylinder sections are not new, but apart from the 3D cylindrical Sunburst they are relatively unexplored in the context of implicit tree visualizations [70]. Once made explicit, these sections are no longer just parts of a standard primitive (cylinder, sphere), but they are primitives in their own right and can be used as such, as illustrated in the following Sections 3.1.1 and 3.1.2. One can of course also generate a new primitive as an entirely new parameter and explore the design subspace it opens up. An example of this approach is given in Section 3.1.3.

#### *3.1.1 3D Polar Treemaps / Radial Treecube*

This technique uses the sphere section and the cylinder section for a subdivision layout together with an inclusion relation for edge representation. It uses the radial Slice-and-Dice layout that does not cut along the $x$-, $y$-, and $z$-axis, but along the two angles and the radius instead. It is combined with a sphere/sphere-section primitive yielding a new technique that can with equal justification be described either as a 3D Polar Treemap (being a 3D extension of Johnson's Polar Treemap) or as a Radial Treecube (being a radial variant of the Treecube technique) [15], [30].

There are more than one possible 3-dimensional extensions of a circle: sphere, cylinder, or cone. Hence, we have also derived a cylindrical version – shown

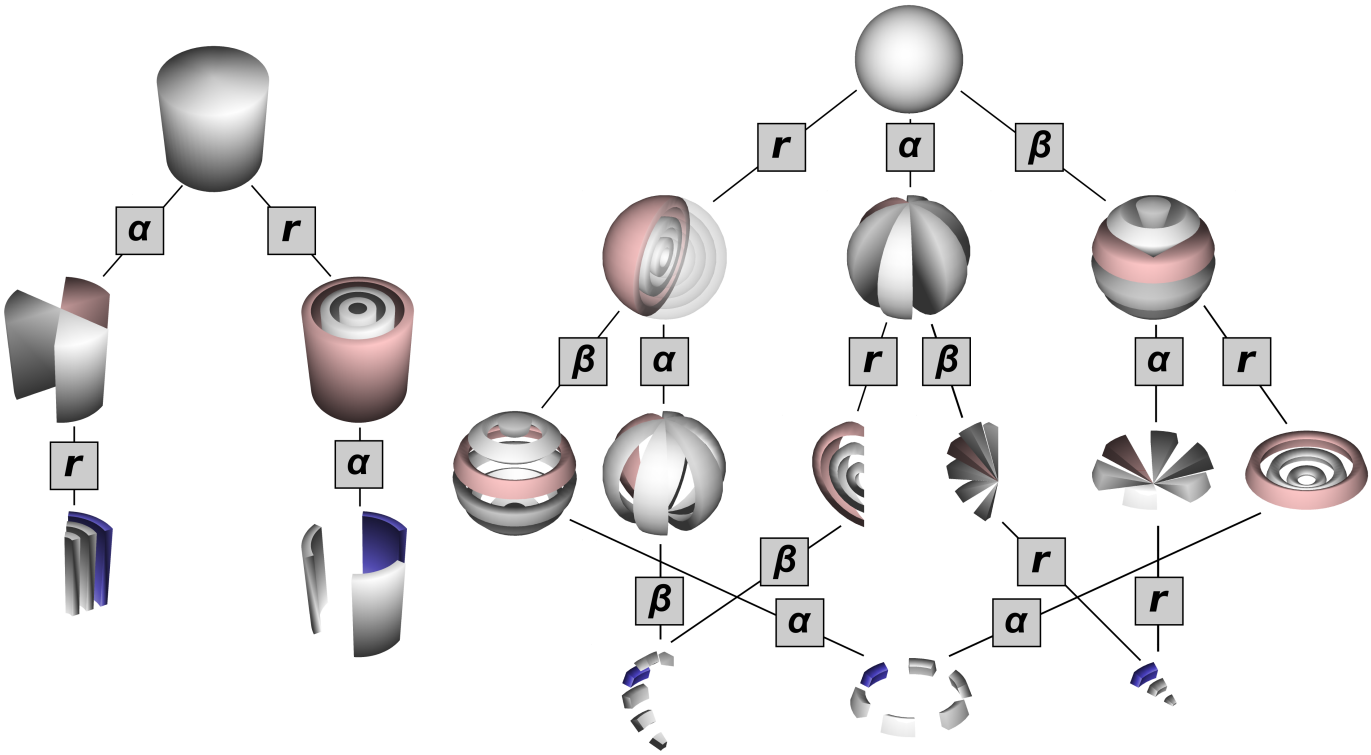



Fig. 5. All possible polar cuttings of a cylinder (a) and a sphere (b). The red colored sections are cut in the next step. The blue colored sections are the resulting least elements – the building blocks of all the shapes above them.

alongside the spherical one in Figure 6(a-c) together with the original 2-dimensional Polar Treemap. While being aesthetically very pleasing, 3D Polar Treemaps suffer from the same amount of occlusion as Treecubes and cannot be recommended for deep hierarchies. Yet, changing the inclusion relation into an adjacency relation leads to the extruded 3D Polar Treemap shown in Figure 10. Likewise, it could also be called a radial cylindrical StepTree, if the squarified layout of the StepTree is altered into a polar coordinate Slice-and-Dice. It still has the Polar Treemap feel to it, but it is less occluded than the ones discussed here.

#### 3.1.2 Spherical 3D Sunbursts

While in [70] a 3-dimensional version of the well known Sunburst technique has already been presented, it is based on a cylindrical extension of the circular Sunburst. Our spherical 3D Sunburst is an extruded technique that makes use of adjacency to represent the parent-child relation. It is an example of how the sphere section can be used as a primitive by itself and (correctly parametrized) stacked on top of others of its kind.

Again, as there is more than one possible 3-dimensional extension for circles, we introduce the so far missing spherical 3D Sunburst in two variants. Both variants use a sphere and certain spherical cross sections as primitives – one allows the entire root to be covered (variant "Sphere"), whereas the other only allows attaching primitives to a narrow, horizontal belt (variant "Wheel") – in part mimicking the original 2D layout. "Sphere" and "Wheel" are shown in Figure 6(d-f) alongside the original 2D Sunburst. 3D Sunbursts tend to accentuate the leaves of a hierarchy, while occluding most of their interior nodes – quite alike to the original 2D Treemap layout without borders. At the same time, it is a very space saving way of displaying a rather wide hierarchy that is not too deep. The variant "Wheel" lessens this effect a little by allowing viewers to peek inside the visualization from above and below. Yet, to do so it sacrifices some of the space that the variant "Sphere" uses so efficiently.

#### 3.1.3 Spiral Steptree

At the core of this new technique stands a new primitive which we call a "leaning box". This is a cuboid with its top being cut off at a certain angle. Stacking a number of them on top of each other in a Steptree manner results in them leaning towards one side of the representation. If the tree is deep enough, such a visualization starts to curl around itself, forming a Spiral Steptree as shown in Figure 7. This proves to be an extremely space efficient way to display very deep, but narrow hierarchies. How tightly the representation curls up can be specified by the cut-off angle of the leaning boxes.

It is crucial to use a layout method that works well with this kind of primitive. Otherwise, visualizations

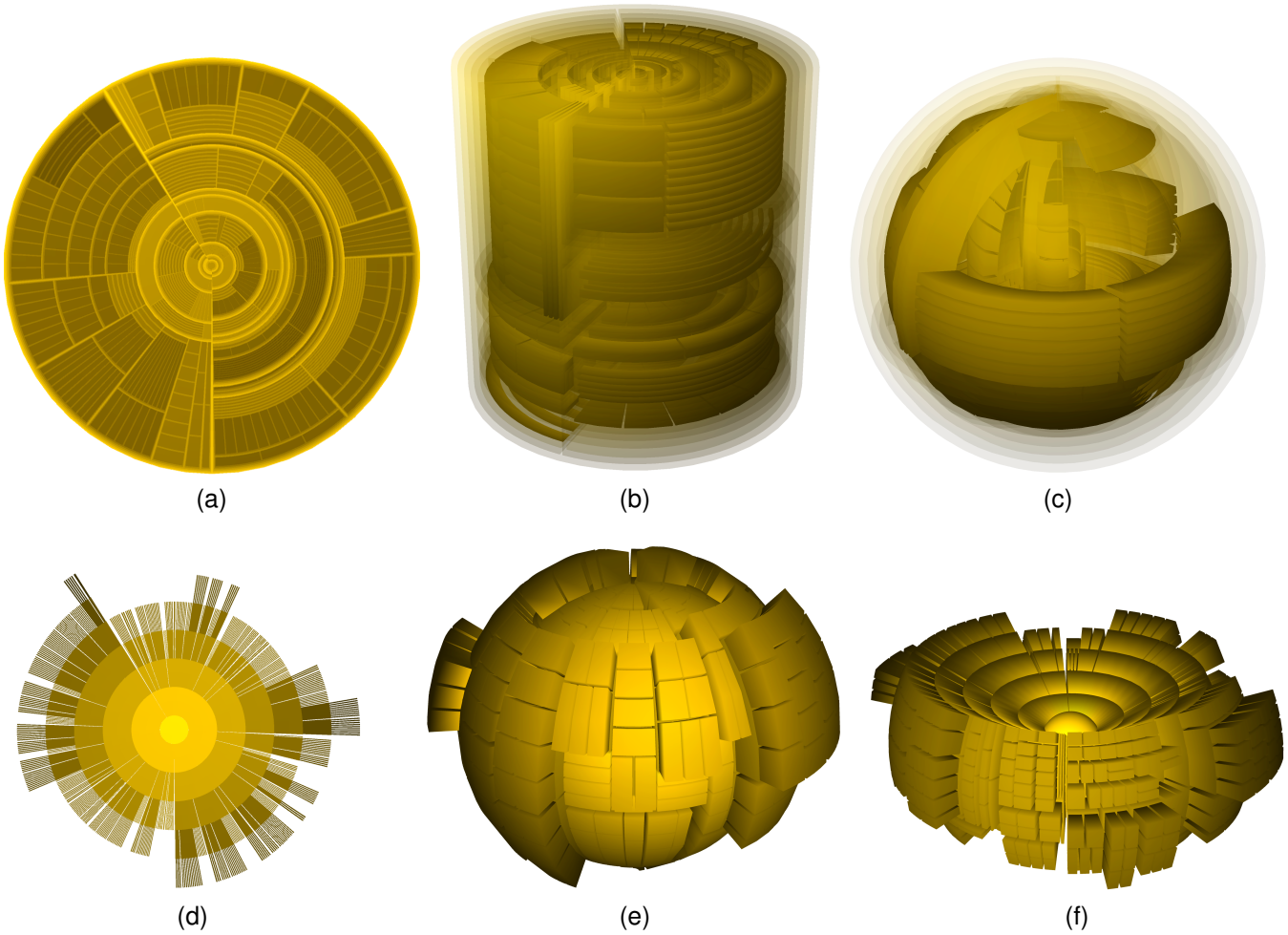



Fig. 6. (a) The original 2D Polar Treemap (b) The 3D Polar Treemap variant "Cylinder" (c) The 3D Polar Treemap variant "Sphere" (d) The original 2D Sunburst (e) The 3D Sunburst variant "Sphere" (f) The 3D Sunburst variant "Wheel".

like the one shown in Figure 4 will be the outcome. The simple layout we have chosen here subdivides the available space only along one dimension. This results in something that can be perceived as an Icicle Plot where the branches are not shown side by side, but are instead stacked behind one another. Because it is so compact, this technique is of no use to inspect individual nodes. It rather provides an overview for deep trees with expected uniform height. Any deviations from the uniform height will be instantly visible: shorter branches will show as gaps in the spiral, longer branches will peek out of the end of the spiral. To easily determine height differences between branches, an alternating color scheme has been chosen. Furthermore, the layout can be reordered to show a branch of interest up front.

### 3.2 Mixing Design Choices

The above means of generating new techniques already allow us to obtain numerous hierarchical representations. Mixing two or more of them together into a new technique extends these possibilities even further. It is interesting that mixing is not used much besides the few examples given in Section 2.2.2, despite its simple concept and its unobtrusive way to accentuate certain nodes and subtrees within the shown hierarchy. On top of that, it also allows one to use specifically tailored design choices depending on the characteristics of a subtree.

Our first example mixes node primitives and parent-child relationships at the same time as shown in Figure 8. Adjacency is used to stack non-leaf nodes and inclusion is applied to nest the leaves (shown as red spheres) into their parents. The level of transparency of the non-leaf nodes can be adjusted to emphasize either the overall structure (if made opaque) or to give an impression of the leaf-level (if made transparent).

The second example in Figure 9 depicts a mixed Treemap with 2D and 3D node representations: (a) shows a mixed 2D/3D tree where all internal nodes are mapped onto stacked boxes in a Steptree-fashion, and all leaves are displayed as 2D Treemap on top of the boxes. (b) emphasizes the leaves, by adding transparency to the 3-

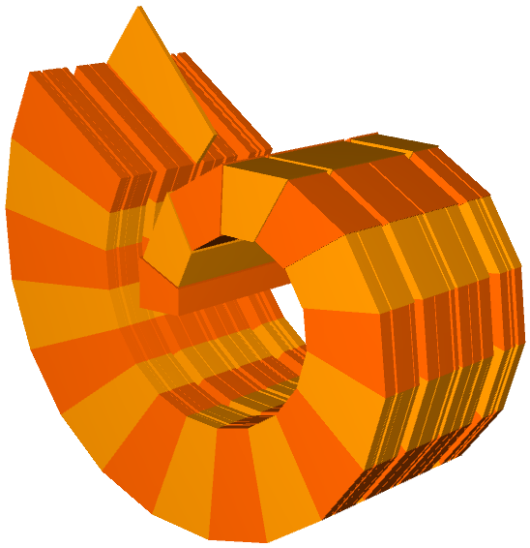


Fig. 7. Spiral Steptree that generates dense visualizations of hierarchies.

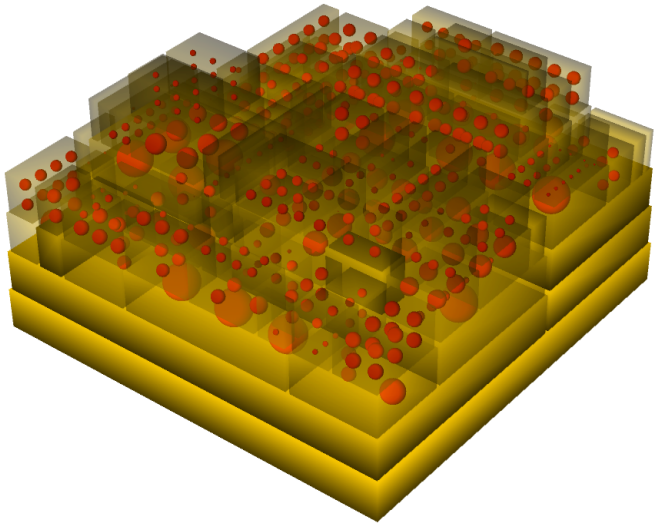


Fig. 8. An example of mixing adjacency for internal nodes and inclusion for leaves.

dimensional boxes, leaving only the 2D-Treemap formed by the leaves clearly visible, but conveying their depth within the tree by the different heights of the 2D patches. (c) shows a bird's eye view, orthogonal to the top-faces of the 3D boxes, effectively yielding a 2-dimensional Treemap view of the mixed representation.

### 3.3 Integrating Derived Measures

In addition to the aforementioned design strategies, it is always possible to combine analytical means with visual cues defined by the design space. Two examples illustrate how such analytical measures can be used to integrate further structural information into an implicit hierarchy visualization – a 3D Polar Treemap.

#### *3.3.1 Implicit Tree Skeletons*

So far, tree skeletons have been investigated for node-link-visualizations only. They are defined as "the set of nodes and edges that are determined to be significant by a given metric" and are usually encoded in visual attributes of the edges [71]. While implicit visualizations lack the edges, the idea of showing just an overview of the most important main branches of a hierarchy is equally intriguing for implicit techniques. What the main branches are and how far down to display them is usually determined by a complexity measure such as the Strahler numbers. An example for a 3D realization of an implicit tree skeleton using Strahler numbers is shown in Figure 10(b). Here, only nodes having a Strahler number greater than a certain cutoff value are shown opaque, all others are rendered semi-transparent.

#### *3.3.2 Eccentricity Values*

Eccentricity quantifies the distance between a node and the tree's graph theoretical center. It plays an important role in rebalancing search trees, and in Operations Research, where a negative correlation exists between eccentricity and cost efficiency. Often it is desired that the root of a tree is also its most central node. Therefore, mapping the eccentricity value onto a tree representation reveals much about its balance. Figure 10(c) depicts such an unbalanced hierarchy in which the root is not the most central node.

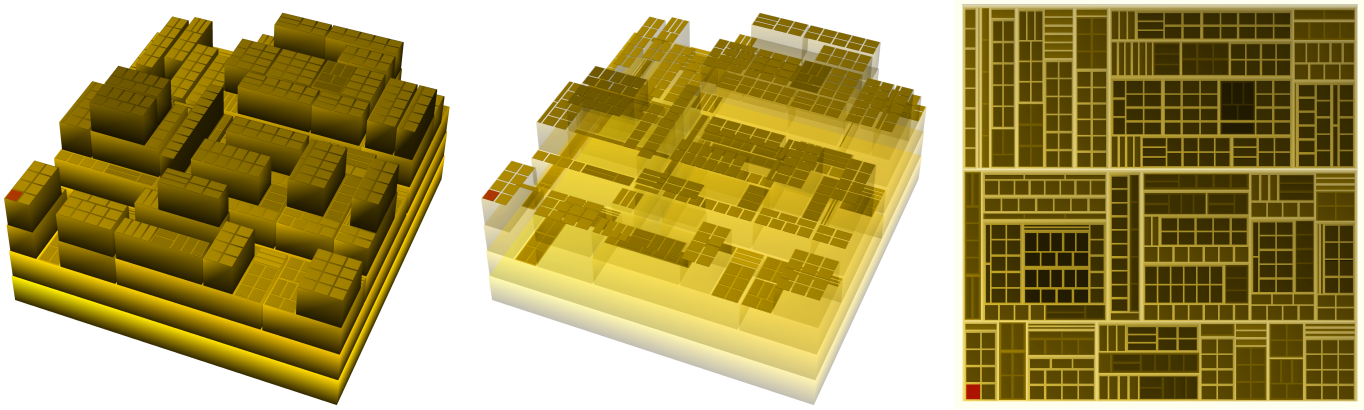
As the design space allows us to map eccentricity and the tree skeleton to orthogonal visual cues, it is now possible to bring both of them together in one visualization. Figure 10(d) shows this combination. It gives an overview of the main branches and their balancing without being occluded by the large number of leaves.

## 4 Implementation of the Design Space

This section goes beyond the pure survey presented so far by briefly describing a practical realization of the introduced design space. As interesting as the design space may be by itself, it was mainly devised as a tool to define new visualizations in the spirit of *rapid visualization prototyping* [72], [73]. It was our aim from the beginning to actually access the design space and to try out new design combinations.

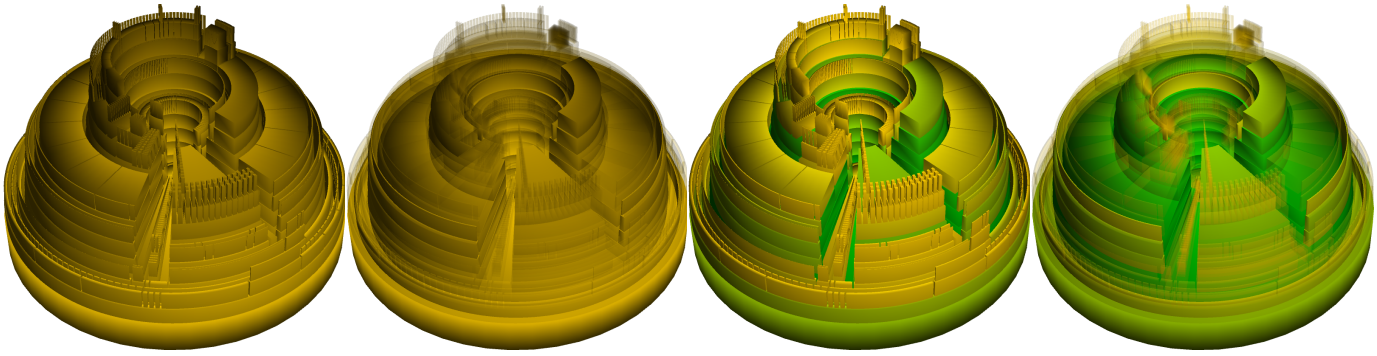
Our software implementation enables users to interactively explore the design space. All examples from the previous section were actually generated using our tool. This is made possible by an approach consisting of five main components:

- a software architecture that is modular and expandable to reflect the changing and ever expanding design space on the design axes of node primitives, layouts, and mixing,
- a number of data importers that allow to load custom hierarchies and real data, and thus to design visualizations tailored to certain data sets or a certain kinds of data (e.g., binary trees)



(a) (b) (c)

Fig. 9. These three visualizations show the same hierarchy with the leaf in the lower left corner being rendered in red to aid in comparing them: (a) mixed 2D/3D representation with the branches rendered in 3D and the leaves laid out in 2D on top of the 3D representation, (b) accentuating the leaves of the 2D Treemap by rendering the 3-dimensional primitives semi-transparent, (c) bird's eye view from above hiding the 3D part of the representation, leaving a 2-dimensional impression.



(a) (b) (c) (d)

Fig. 10. (a) A plain extruded 3D Polar Treemap, which is basically a Steptree-like technique, but uses the Polar Treemap subdivision layout (b) All nodes in the 3D Polar Treemap having a Strahler number smaller than a certain threshold are faded out (c) The eccentricity value mapped onto the color with green being mapped on smaller values, showing that the root is not the graph theoretical center of the hierarchy (d) A combination of (b) and (c)

- a flexible scripting capability to implement and run custom-made preprocessing algorithms (e.g., normalizing numerical node attributes or computing Strahler numbers)
- an interactive user interface that supports the back and forth of the exploratory process and makes visualization parameters directly available in a point-and-click fashion,
- a history mechanism that is able to capture this process in order to make the design decisions reproducible within a design session (undo/redo) as well as across multiple sessions (load/save)

The navigation of the design space is mostly interactive in the sense of point and click. In cases where a more elaborate parametrization is needed (e.g., making the height of 3D node primitives inversely proportional to their depth level within the hierarchy), input fields allow users to enter custom functions. In contrast to other specification mechanisms, such as the HiVE notation, which addresses concrete research questions for specific data sets [9], our operator-based description of visualization techniques is independent of the actual data set. All nodes automatically decorated with certain properties like `is_leaf`, `has_leaves`, `number_of_siblings`, or `strahler_number`, which can be used to exactly specify the subset of nodes that is affected by a design decision. A scripting interface allows users to define additional properties.

A prototype of our software implementation is accessible online as a Java applet at http://vcg.informatik.uni-rostock.de/~hs162/itvtk/start.html. It utilizes JOGL for 3D graphics and Groovy for the scripting support. We have equipped our software with a number of the most common primitives, layouts, and preprocessing

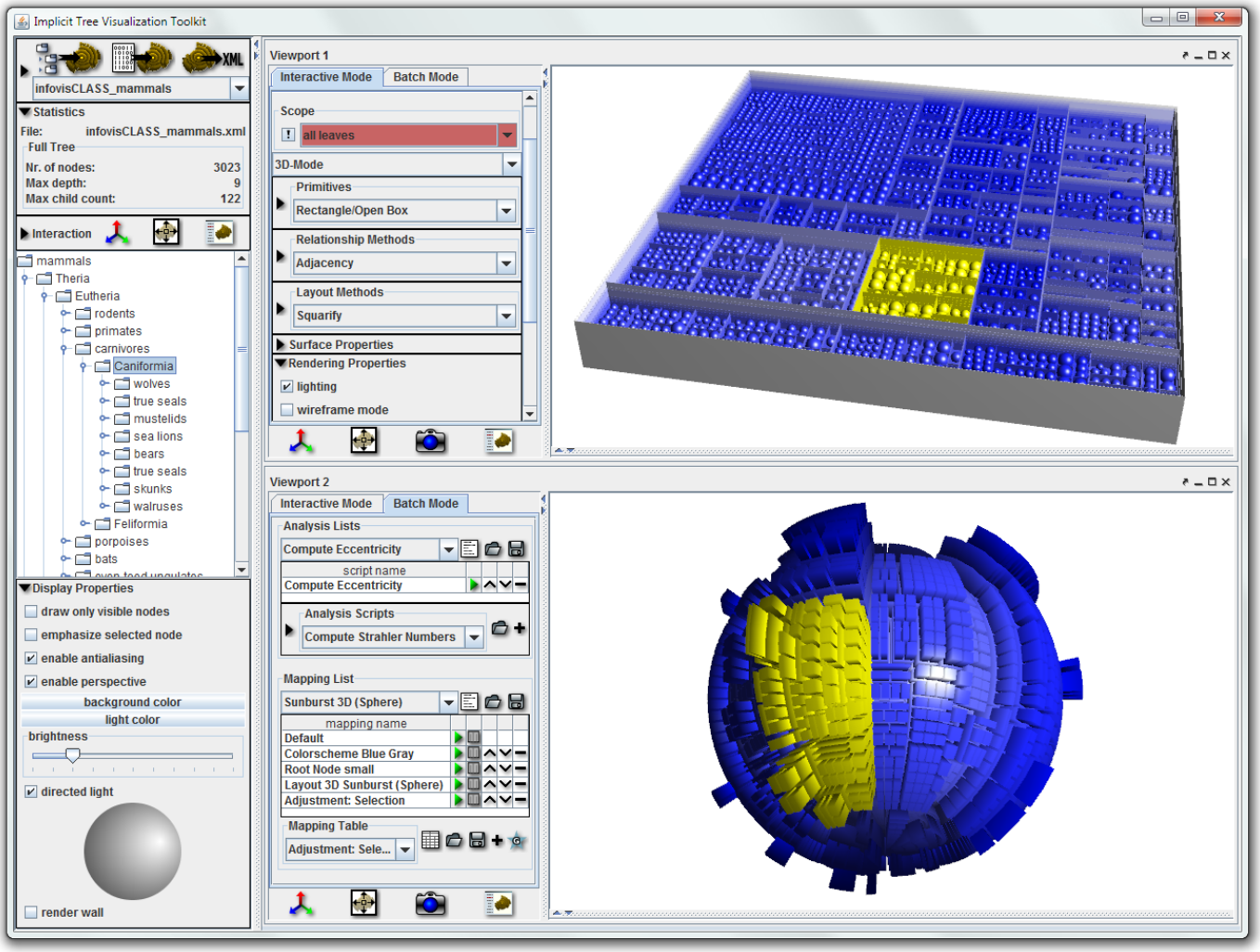



Fig. 11. The Implicit Tree Visualization Toolkit – The figure shows the mammals of the InfoVis 2003 contest classification hierarchy "A" with the subtree "caniforms" (dog-like animals) highlighted in all views. On the left-hand side, our software shows statistics of the data set, a regular tree view, and controls for visualization parameters that have a global effect. Multiple viewports can be opened to create implicit tree visualization prototypes. Each viewport has a set of local controls to alter visualization parameters on a per-viewport basis either by interactive adjustment or via scripting. Parameter settings and scripts can be stored for later reuse. The top view ("Viewport 1") shows a mixed 3D Treemap technique that stacks boxes inside one another for internal nodes and represents leaves as spheres inside these boxes. This unobtrusively accentuates the leaves and makes this prototype look very much like a carefully arranged collection of items, in this case all sorts of mammals. The bottom view ("Viewport 2") shows a spherical Sunburst variant as discussed in Section 3.1.2.

algorithms in this field. The largest part of the existing implicit hierarchy visualization techniques can be built out of the box using the prototype. Many new techniques can be generated just by re-combining and mixing these pre-configured modules in new and unexpected ways.

The prototype shown in Figure 11 displays different implicit visualization prototypes in multiple linked views. The linked views have proven to be extremely helpful for comparative testing of different techniques. A view can easily be generated by "cloning" the current view and then altering the new view's specification. The linking between the views is such that a selection in one visualization is reflected in all others. Optionally, it is also possible to sync zoom, pan, and rotate operations across all views – for example to have a second view that always shows the otherwise invisible back side of a 3D visualization. Applications for the linked views are for example:

- co-designing overview and detail visualization side by side,
- branching the design process by cloning the current viewport to establish two individual specification histories,
- comparing one's own visualization with one of the built-in standards, e.g., a standard Treemap.

This makes the software a versatile and powerful tool

for rapid visualization prototyping of implicit hierarchy visualization techniques.

## 5 Conclusion

Unraveling, defining, and surveying the design space for the class of implicit tree visualizations has shown to be useful in many aspects. The definition of a conceptually complete and consistent design space and its practical applicability stand as a basis for a holistic classification and a rapid prototyping of known and yet unknown visualization techniques in this class. The systematic discussion of the entire design space helps the sporadic visualization user to get acquainted with the general design concept behind a whole class of visualizations, putting the known examples into a broader context. Taking this step back and looking at the big picture enabled us to break down visualization techniques into the recurring visualization design patterns that they are made of. It is this externalization of design knowledge that provides a more generative perspective on implicit tree visualizations and by that to get a grip on its rather vast design space by means of rapid prototyping. It is the very same design space that allows comparing different techniques: conceptually through the number of steps needed in the design space to transform one into the other, and practically through direct interaction with multiple linked prototypes in the design space implementation.

Of course, this survey could not depict and discuss all existing implicit techniques. Even beyond the numerous examples that we evaluated in Table 1 and Figures 12 and 13, there are many more existing implicit hierarchy visualizations out there, e.g., Bubblemaps, Grid Treemaps, Context Treemaps, Treemaps with Textures and Bump Mapping, Filled Sunbursts and the Hybrid Sunburst/Treemap (basically a 2-sided Icicle Plot), Contour Maps, and the Blob Hierarchy Layout to name just a few [66], [74]–[79]. This abundance shows clearly that a survey was long overdue. But all of the techniques not considered in this survey can equally be placed in the proposed design space. As many more new implicit visualizations will surely follow, it is not the incomplete list of techniques in Table 1 that is the main contribution of this survey, but rather the complete and consistent design space definition that captures them all – existing ones, as well as new ones.

The design space implies multiple promising directions for future work. As the overall design dimensions have been identified, subtler design aspects such as parametrization and mixing of design choices come into focus. Especially their specification according to the characteristics of a hierarchy seems largely unexplored and highly promising to yield more effective visualizations. For instance, mixing different layout techniques or parametrizing a layout according to a subtree's width/depth would result in unique visualizations that by themselves are characteristic for the hierarchy they display.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the DFG graduate school *dIEM oSiRiS* – http://www.diemosiris.de.

## References

[1] M. J. McGuffin and J.-M. Robert, "Quantifying the space-efficiency of 2D graphical representations of trees," *Information Visualization*, 2009, to appear.
[2] H.-J. Schulz and H. Schumann, "Visualizing graphs: A generalized view," in *IV'06: Proceedings of the International Conference Information Visualisation*, 2006, pp. 166–173.
[3] M. Bugajska, "Framework for spatial visual design of abstract information," in *IV'05: Proceedings of the International Conference Information Visualisation*, 2005, pp. 713–723.
[4] P. R. Keller and M. M. Keller, *Visual Cues: Practical Data Visualization*. IEEE Computer Society Press, 1994.
[5] J. Bertin, *Graphics and Graphic Information Processing*. Walter de Gruyter, 1981.
[6] E. R. Tufte, *Envisioning Information*. Graphics Press, 1990.
[7] H.-J. Schulz, M. Luboschik, and H. Schumann, "Exploration of the 3D treemap design space," in *InfoVis'07: Poster Compendium of the IEEE Conference on Information Visualization*, 2007, pp. 78–79.
[8] R. Vliegen, J. J. van Wijk, and E.-J. van der Linden, "Visualizing business data with generalized treemaps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 789–796, 2006.
[9] A. Slingsby, J. Dykes, and J. Wood, "Configuring hierarchical layouts to address research questions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 977–984, 2009.
[10] B. Johnson and B. Shneiderman, "Tree-maps: A space-filling approach to the visualization of hierarchical information structures," in *Visualization'91: Proceedings of the IEEE Conference on Visualization*, 1991, pp. 284–291.
[11] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," *ACM Transactions on Graphics*, vol. 11, no. 1, pp. 92–99, 1992.
[12] J. B. Kruskal and J. M. Landwehr, "Icicle plot: Better displays for hierarchical clustering," *The American Statistician*, vol. 37, no. 2, pp. 162–168, 1983.
[13] A. M. Dean, J. A. Ellis-Monaghan, S. Hamilton, and G. Pangborn, "Unit rectangle visibility graphs," *The Electronic Journal of Combinatorics*, vol. 15, no. 1 - #R79, 2007.
[14] M. E. Baron, "A note on the historical development of logic diagrams: Leibniz, Euler and Venn," *The Mathematical Gazette*, vol. 53, no. 284, pp. 113–125, 1969.
[15] B. S. Johnson, "Treemaps: Visualizing hierarchical and categorical data," Ph.D. dissertation, University of Maryland, 1993.
[16] C. Chen, *Information Visualization. Beyond the Horizon*, 2nd ed. Springer, 2006.
[17] B. Kleiner and J. A. Hartigan, "Representing points in many dimensions by trees and castles," *Journal of the American Statistical Association*, vol. 76, no. 374, pp. 260–269, June 1981.
[18] M. Bruls, K. Huizing, and J. van Wijk, "Squarified treemaps," in *Data Visualization'00: Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, 2000, pp. 33–42.
[19] H. R. Lu and J. Fogarty, "Cascaded treemaps: Examining the visibility and stability of structure in treemaps," in *GI'08: Proceedings of the Graphics Interface Conference*, 2008, pp. 259–266.
[20] T. Bladh, D. A. Carr, and J. Scholl, "Extending tree-maps to three dimensions: A comparative study," in *APCHI'04: Proceedings of the Asia-Pacific Conference on Computer-Human Interaction*, 2004, pp. 50–59.
[21] W. Wang, H. Wang, G. Dai, and H. Wang, "Visualization of large hierarchical data by circle packing," in *CHI'06: Proceedings of the SIGCHI conference on Human factors in computing systems*, 2006, pp. 517–520.
[22] H. S. Smallman, M. S. John, H. M. Oonk, and M. B. Cowen, "Information availability in 2d and 3d displays," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 51–57, September-October 2001.

[23] F. van Ham and J. J. van Wijk, "Beamtrees: Compact visualization of large hierarchies," in *InfoVis'02: Proceedings of the IEEE Symposium on Information Visualization*, 2002, pp. 93–100.

[24] C. Keskin and V. Vogelmann, "Effective visualization of hierarchical graphs with the cityscape metaphor," in *Proceedings of the Workshop on New Paradigms in Information Visualization and Manipulation 1997*, 1997, pp. 52–57.

[25] J. S. Risch, "On the role of metaphor in information visualization," *Computing Research Repository*, vol. abs/0809.0884, 2008.

[26] N. Churcher, L. Keown, and W. Irwin, "Virtual worlds for software visualisation," in *SoftVis'99: Proceedings of the Software Visualisation Workshop*, 1999, pp. 9–16.

[27] M. Balzer and O. Deussen, "Hierarchy based 3D visualization of large software structures," in *Visualization'04: Poster Compendium of IEEE Conference on Visualization*, 2004, pp. 81–82.

[28] K. Andrews, J. Wolte, and M. Pichler, "Information pyramids$^{TM}$: A new approach to visualising large hierarchies," in *Visualization'97: Proceedings of the IEEE Conference on Visualization*, 1997, pp. 49–52.

[29] J. Rekimoto and M. Green, "The information cube: Using transparency in 3D information visualization," in *Proceedings of the Workshop on Information Technologies and Systems 1993*, 1993, pp. 125–132.

[30] Y. Tanaka, Y. Okada, and K. Niijima, "Treecube: Visualization tool for browsing 3D multimedia data," in *IV'03: Proceedings of the International Conference Information Visualisation*, 2003, pp. 427–432.

[31] M. Tavanti and M. Lind, "2D vs. 3D, implications on spatial memory," in *InfoVis'01: Proceedings of the IEEE Symposium on Information Visualization*, 2001, pp. 139–146.

[32] A. Cockburn and B. McKenzie, "Evaluating spatial memory in two and three dimensions," *International Journal of Human-Computer Studies*, vol. 61, no. 3, pp. 359–373, 2004.

[33] M. Hicks, C. O'Malley, S. Nichols, and B. Anderson, "Comparison of 2D and 3D representations for visualising telecommunication usage," *Behavior and Information Technology*, vol. 22, no. 3, pp. 185–201, 2003.

[34] R. Wettel and M. Lanza, "Visualizing software systems as cities," in *VisSoft'07: Proceedings of the IEEE International Workshop on Visualizing Software for Understanding and Analysis*, 2007, pp. 92–99.

[35] P. Liggesmeyer, J. Heidrich, J. Münch, R. Kalcklösch, H. Barthel, and D. Zeckzer, "Visualization of software and systems as support mechanism for integrated software project control," in *HCI International'09: Proceedings of the International Conference on Human-Computer Interaction*, 2009, pp. 846–855.

[36] K. Wetzel, "Pebbles – using circular treemaps to visualize disk usage," 2003, http://lip.sourceforge.net/ctreemap.html.

[37] T. D. Wang and B. Parsia, "Cropcircles: Topology sensitive visualization of OWL class hierarchies," in *ISWC'06: Proceedings of the International Semantic Web Conference*, 2006, pp. 695–708.

[38] A. Dix, R. Beale, and A. Wood, "Architectures to make simple visualisations using simple systems," in *AVI'00: Proceedings of the International Working Conference on Advanced Visual Interfaces*, 2000, pp. 51–60.

[39] R. O'Donnell, A. Dix, and L. J. Ball, "Exploring the PieTree for representing numerical hierarchical data," in *Proceedings of the HCI'06 Conference on People and Computers XX*, 2006, pp. 239–254.

[40] J. Stasko and E. Zhang, "Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations," in *InfoVis'00: Proceedings of the IEEE Symposium on Information Visualization*, 2000, pp. 57–65.

[41] J. Stasko, R. Catrambone, M. Guzdial, and K. McDonald, "An evaluation of space-filling information visualizations for depiction hierarchical structures," *International Journal of Human-Computer Studies*, vol. 53, no. 5, pp. 663–694, 2000.

[42] J. Yang, M. O. Ward, and E. A. Rundensteiner, "Interring: An interactive tool for visually navigating and manipulating hierarchical structures," in *InfoVis'02: Proceedings of the IEEE Symposium on Information Visualization*, 2002, pp. 77–84.

[43] K. Andrews, W. Kienreich, V. Sabol, J. Becker, G. Droschl, F. Kappe, M. Granitzer, P. Auer, and K. Tochtermann, "The InfoSky visual explorer: Exploiting hierarchical structure and document similarities," *Information Visualization*, vol. 1, no. 3/4, pp. 166–181, 2002.

[44] M. Balzer and O. Deussen, "Voronoi treemaps," in *InfoVis'05: Proceedings of the IEEE Symposium on Information Visualization*, 2005, pp. 49–56.

[45] K. Onak and A. Sidiropoulos, "Circular partitions with applications to visualization and embeddings," in *SCG'08: Proceedings of the ACM Symposium on Computational Geometry*, 2008, pp. 28–37.

[46] J. Hao, K. Zhang, and M. L. Huang, "RELT – visualizing trees on mobile devices," in *VISUAL'07: Proceedings of the International Conference on Visual Information Systems*, 2007, pp. 344–357.

[47] J. Hao, K. Zhang, C. A. Gabrysch, and Q. Zhu, "Managing hierarchical information on small screens," in *APWeb/WAIM'09: Proceedings of the Joint International Conferences on Advances in Data and Web Management*, 2009, pp. 429–441.

[48] B. Otjacques, M. Noirhomme, X. Gobert, P. Collin, and F. Feltz, "Visualizing the activity of a web-based collaborative platform," in *IV'07: Proceedings of the International Conference Information Visualisation*, 2007, pp. 251–256.

[49] M. C. Chuah, "Dynamic aggregation with circular visual designs," in *InfoVis'98: Proceedings of the IEEE Symposium on Information Visualization*, 1998, pp. 35–43.

[50] L. Beaudoin, M.-A. Parent, and L. C. Vroomen, "Cheops: A compact explorer for complex hierarchies," in *Visualization'96: Proceedings of the IEEE Conference on Visualization*, 1996, pp. 87–92.

[51] J. J. van Wijk and H. van de Wetering, "Cushion treemaps: Visualization of hierarchical information," in *InfoVis'99: Proceedings of the IEEE Symposium on Information Visualization*, 1999, pp. 73–78.

[52] F. Chevalier, D. Auber, and A. Telea, "Structural analysis and visualization of C++ code evolution using syntax trees," in *IWPSE'07: Proceedings of the International Workshop on Principles of Software Evolution*, 2007, pp. 90–97.

[53] Y. Tu and H.-W. Shen, "Visualizing changes of hierarchical data using treemaps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1286–1293, 2007.

[54] M. Luboschik and H. Schumann, "Discovering the covered: Ghost-views in information visualization," in *WSCG'08: Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2008, pp. 113–118.

[55] A. F. Stuart, "Container metaphor for visualization of complex hierarchical data types," United States Patent Application US 2006/0080622 A1, filed Oct. 12, 2004, published Apr. 13, 2006.

[56] M. Garey and D. Johnson, *Computers and Intractability – A Guide to the Theory of NP-completeness*. W.H.Freeman, 1979.

[57] T. Itoh, Y. Yamaguchi, Y. Ikehata, and Y. Kajinaga, "Hierarchical data visualization using a fast rectangle-packing algorithm," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 3, pp. 302–313, 2004.

[58] T. Itoh, Y. Kajinaga, Y. Ikehata, and Y. Yamaguchi, "Data jewelry box: A graphics showcase for large-scale hierarchical data visualization," IBM Research, Tech. Rep. RT0427, 2002.

[59] M. Wattenberg, "A note on space-filling visualizations and space-filling curves," in *InfoVis'05: Proceedings of the IEEE Symposium on Information Visualization*, 2005, pp. 181–185.

[60] B. B. Bederson, B. Shneiderman, and M. Wattenberg, "Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies," *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 833–854, 2002.

[61] J. Wood and J. Dykes, "Spatially ordered treemaps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1348–1355, 2008.

[62] M. L. Huang, T.-H. Huang, and J. Zhang, "TreemapBar: Visualizing additional dimensions of data in bar chart," in *IV'09: Proceedings of the International Conference Information Visualisation*, 2009, pp. 98–103.

[63] P. C. Wong, H. Foote, G. C. Jr., P. Mackey, and K. Perrine, "Graph signatures for visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 6, pp. 1399–1413, 2006.

[64] K. Andrews and H. Heidegger, "Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs," in *InfoVis'98: Proceedings of the IEEE Symposium on Information Visualization*, 1998, pp. 9–12.

[65] D. Turo and B. Johnson, "Improving the visualization of hierarchies with treemaps: Design issues and experimentation," in *Visualization'92: Proceedings of the IEEE Conference on Visualization*, 1992, pp. 124–131.

[66] B. B. Bederson, "PhotoMesa: A zoomable image browser using quantum treemaps and bubblemaps," in *UIST'01: Proceedings of the ACM Symposium on User Interface Software and Technology*, 2001, pp. 71–80.

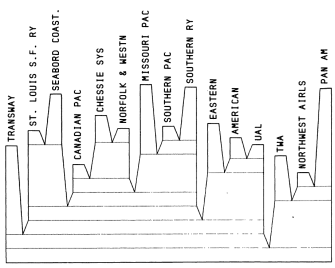
[67] D. Auber, M. Delest, J.-P. Domenger, P. Duchon, and J.-M. Fédou, "New Strahler numbers for rooted plane trees," in *Proceedings of the 3rd Colloquium on Mathematics and Computer Science Algorithms*, 2004, pp. 203–215.
[68] A. Chaudhuri and H.-W. Shen, "A self-adaptive treemap-based technique for visualizing hierarchical data in 3D," in *PacificVis'09: Proceedings of the IEEE Pacific Visualization Symposium*, 2009, pp. 105–112.
[69] M. Wattenberg, "Visualizing the stock market," in *CHI'99: Extended abstracts on Human factors in computing systems*. ACM, 1999, pp. 188–189.
[70] M. Schedl, P. Knees, G. Widmer, K. Seyerlehner, and T. Pohle, "Browsing the web using stacked three-dimensional sunbursts to visualize term co-occurrences and multimedia content," in *InfoVis'07: Poster Compendium of the IEEE Conference on Information Visualization*, 2007, pp. 2–3.
[71] I. Herman, S. Marshall, G. Melançon, D. J. Duke, M. Delest, and J.-P. Domenger, "Skeletal images as visual cues in graph visualization," in *Data Visualization'99: Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, 1999, pp. 13–22.
[72] R. Kazman and J. Carriére, "Rapid prototyping of information visualizations using VANISH," in *InfoVis'96: Proceedings of the IEEE Symposium on Information Visualization*, 1996, pp. 21–28.
[73] A. Lienhard, A. Kuhn, and O. Greevy, "Rapid prototyping of visualizations using Mondrian," in *Vissoft'07: Proceedings of the IEEE Workshop on Visualizing Software for Understanding and Analysis*, 2007, pp. 67–70.
[74] D. Harel and G. Yashchin, "An algorithm for blob hierarchy layout," in *AVI'00: Proceedings of the International Working Conference on Advanced Visual Interfaces*, 2000, pp. 29–40.
[75] T. Schreck, D. Keim, and F. Mansmann, "Regular treemap layouts for visual analysis of hierarchical data," in *SCCG'06: Proceedings of the Spring Conference on Computer Graphics*, 2006.
[76] C. Csallner, M. Handte, O. Lehmann, and J. Stasko, "FundExplorer: Supporting the diversification of mutual fund portfolios using Context Treemaps," in *InfoVis'03: Proceedings of the IEEE Symposium on Information Visualization*, 2003, pp. 203–208.
[77] D. Holten, R. Vliegen, and J. J. van Wijk, "Visual realism for the visualization of software metrics," in *VisSoft'05: Proceedings of the IEEE International Workshop on Visualizing Software for Understanding and Analysis*, 2005, pp. 27–32.
[78] R. D. Herbert, R. Webber, and W. Jiang, "Space-filling techniques in visualizing output from computer based economic models," Society for Computational Economics, Computing in Economics and Finance 2006 – #67, July 2006.
[79] H. Kubota, T. Nishida, and Y. Sumi, "Visualization of contents archive by contour map representation," in *JSAI'06: Proceeding of the Conferences of the Japanese Society for Artificial Intelligence*, 2006, pp. 19–32.

**Hans-Jörg Schulz** received his diploma (MCS) from the University of Rostock in 2004. At present, he is a doctoral candidate of the interdisciplinary graduate school "dIEM oSiRiS" at the University of Rostock. There, he is involved in developing graph visualizations for applications in computational systems biology. His main interests concern the visualization of special graph classes, like bipartite or interval graphs, as well as graph representations aside node-link layouts.

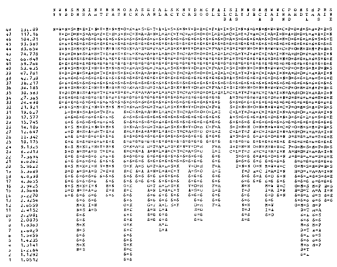
**Steffen Hadlak** received his diploma (MCS) from the University of Rostock in 2009. Currently, he is a PhD candidate in the interdisciplinary graduate school "dIEM oSiRiS" at the University of Rostock. There, he is working on multi-level visualization of dynamic graphs for biomedical and bioinformatics applications. His research interests include GPU-based rendering as well as its application to 3D Information Visualization.

**Heidrun Schumann** graduated at the University of Rostock (1977 Master degree, 1981 PhD, 1989 Habilitation). Since 1992 she is heading the Computer Graphics Research Group at the Institute for Computer Science at the University of Rostock. Her research profile covers Information Visualization, Visual Analytics, and Rendering. Her current research projects, supported by research institutions and industry, include development of scalable frameworks for information visualization and adaptive visual interfaces.

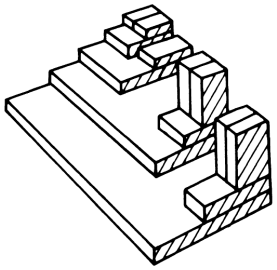
# APPENDIX: GALLERY OF IMPLICIT HIERARCHY VISUALIZATION TECHNIQUES

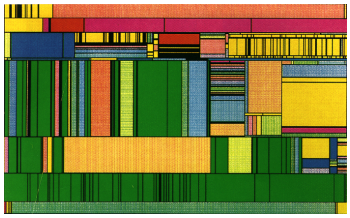



(a) Castles 1981 [17]

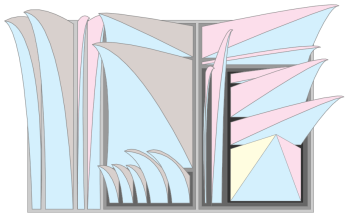


(b) 2D+3D Icicle Plots 1983 [12]

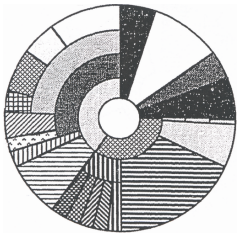




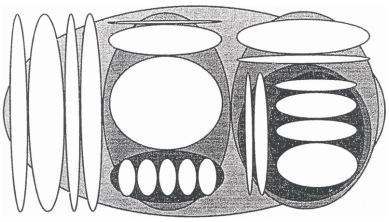
(c) Treemap 1991 [10], [11]



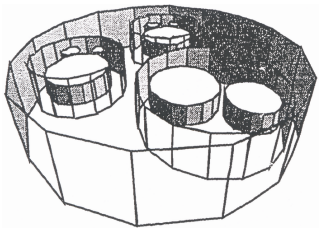
(d) 2 1/2-D Treemap 1992 [65]



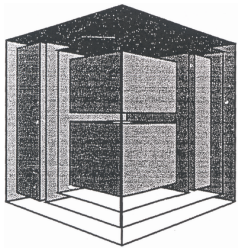
(e) Polar Treemap 1993 [15]



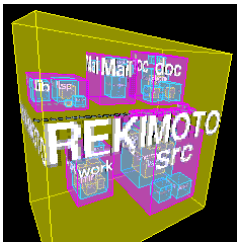
(f) Treemap with Ovals 1993 [15]



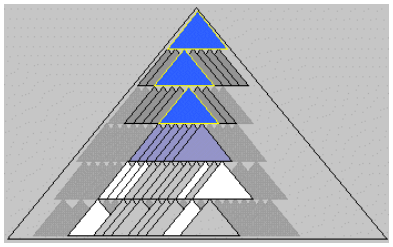
(g) Nested Columns 1993 [15]



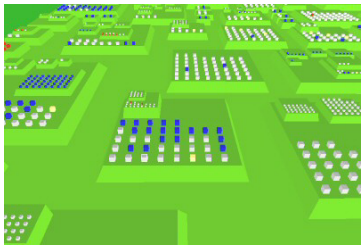
(h) 3D Treemap 1993 [15]



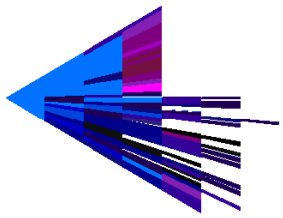

(i) Information Cube 1993 [29]



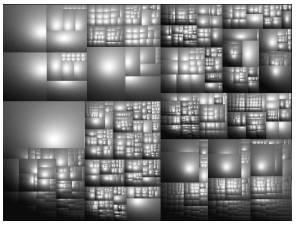
(j) Cheops$^{TM}$ 1996 [50]



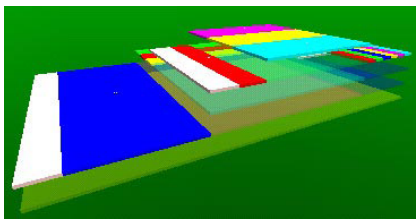
(k) Information Pyramids$^{TM}$ 1997 [28]



(l) Triangular Aggregated Treemap 1998 [49]

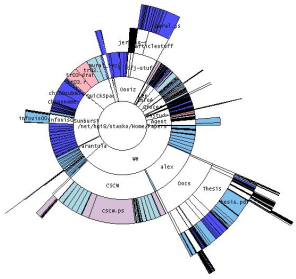


(m) Cushion Treemap 1999 [51]

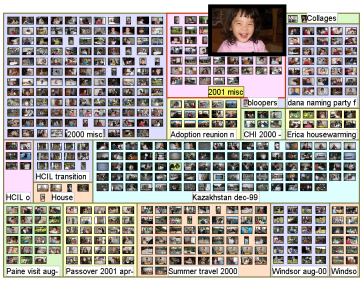


(n) 3D Nested Treemap 1999 [26]

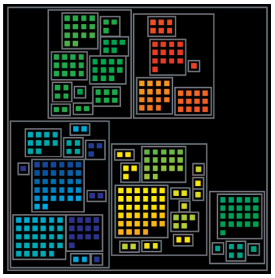
(o) PieTree 2000 [38]



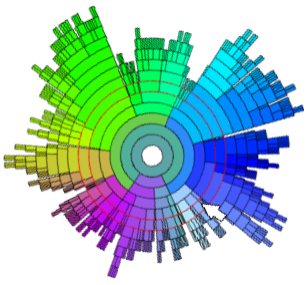
(p) Sunburst 2000 [40]




(q) Quantum Treemap 2001 [66]

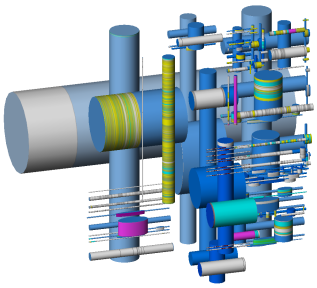


(r) Data Jewelry Box 2002 [58]

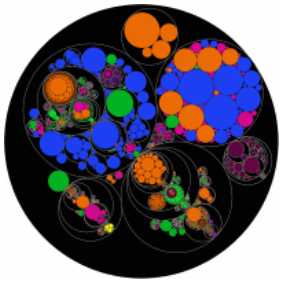


(s) InterRing 2002 [42]

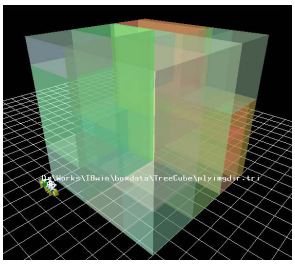
Fig. 12. A selection of prominent and lesser known examples of implicit tree visualizations. Part A: 1981–2002



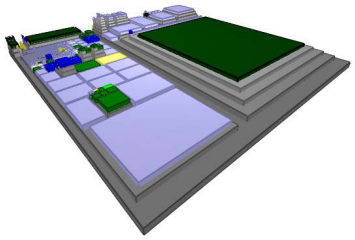
(a) 3D Beamtree 2002 [23]



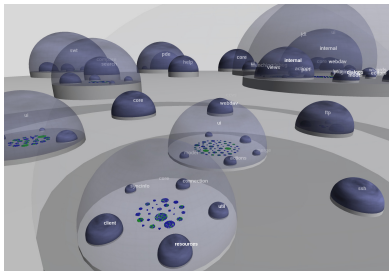
(b) Pebble Map 2003 [36]



(c) Tree Cube 2003 [30]

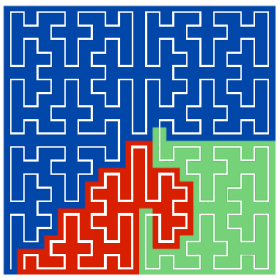


(d) StepTree 2004 [20]

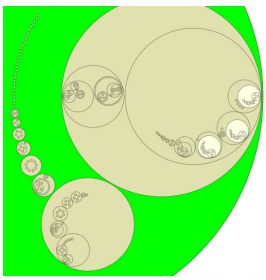


(e) Nested Hemispheres 2004 [27]

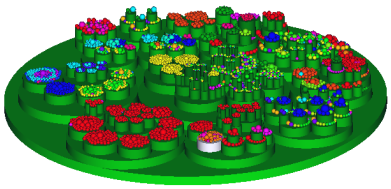
(f) Voronoi Treemap 2005 [44]



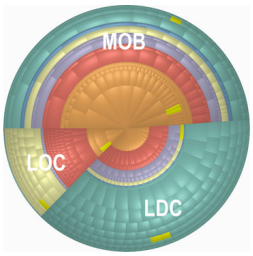
(g) Jigsaw Map 2005 [59]

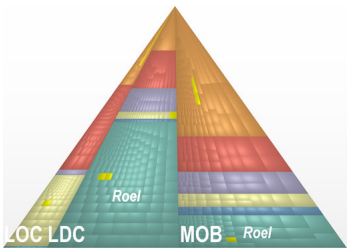


(h) CropCircles 2006 [37]

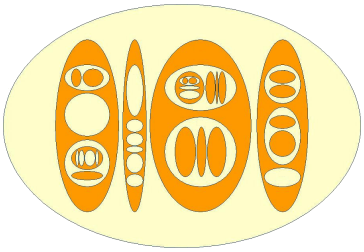


(i) 3D Nested Cylinders and Spheres 2006 [21]

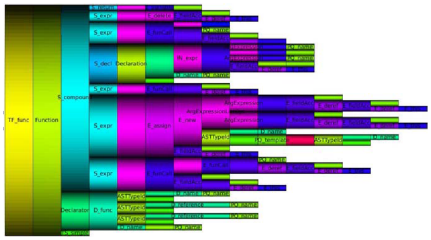


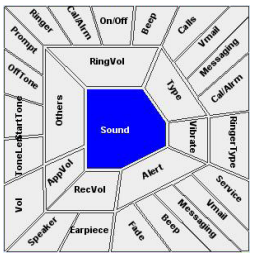

(j) Generalized Treemaps 2006 [8] – pie, pyramid, and combination of both



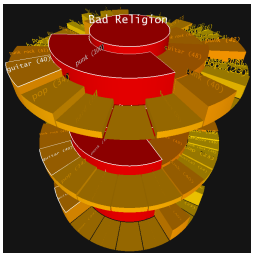
(k) Ellimap 2007 [48]



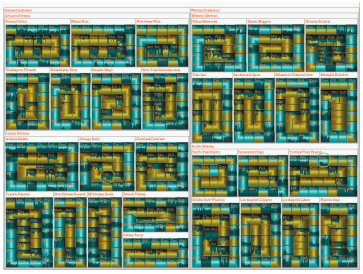
(l) Cushioned Icicle Plot 2007 [52]




(m) Radial Edgeless Tree 2007 [46], [47]

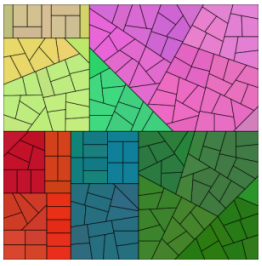



(n) 3D Sunburst 2007 [70]

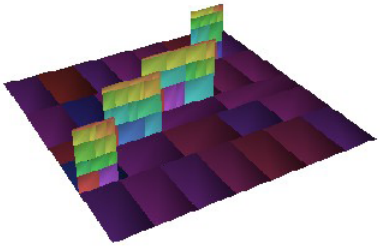


(o) Contrast Spiral Treemap 2007 [53]


(p) Cascaded Treemap 2008 [19]



(q) Circular Partitions 2008 [45]



(r) Lifted Treemap 2009 [68]

Fig. 13. A selection of prominent and lesser known examples of implicit tree visualizations. Part B: 2002–2009