CSC 544
Data Visualization

Joshua Levine
josh@arizona.edu

Lecture 13
Hierarchies/Trees

Feb. 27, 2023

Today's Agenda

- Reminders:
  - A03 questions?, P02 questions?
- Goals for today:
  - Discuss visualization techniques for trees and hierarchies

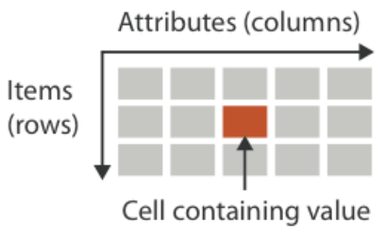
Terminology for Trees and Graphs

Data and Dataset Types

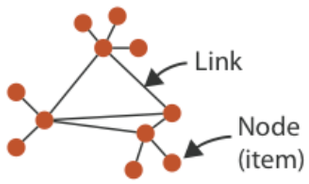
Tables	Networks & Trees	Fields	Geometry	Clusters, Sets, Lists
Items	Items (nodes)	Grids	Items	Items
Attributes	Links	Positions	Positions	
	Attributes	Attributes		

Dataset Types

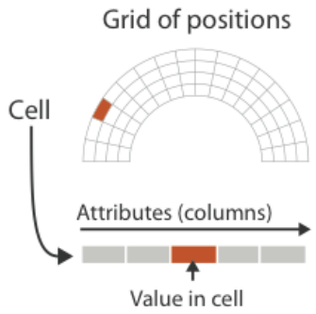
Tables



Networks

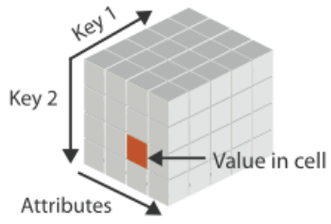


Fields (Continuous)



Geometry (Spatial)

Multidimensional Table



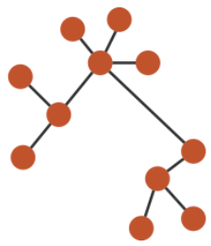
Trees

Design Choices for Trees and Graphs

Arrange Networks and Trees

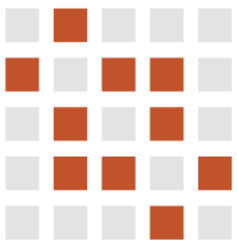
Node–Link Diagrams
Connections and Marks

✔ NETWORKS ✔ TREES



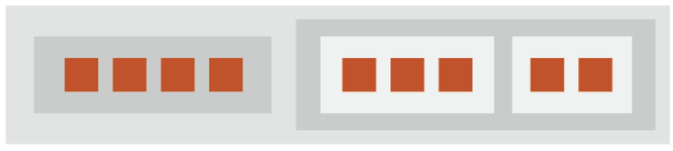
Adjacency Matrix
Derived Table

✔ NETWORKS ✔ TREES



Enclosure
Containment Marks

✘ NETWORKS ✔ TREES



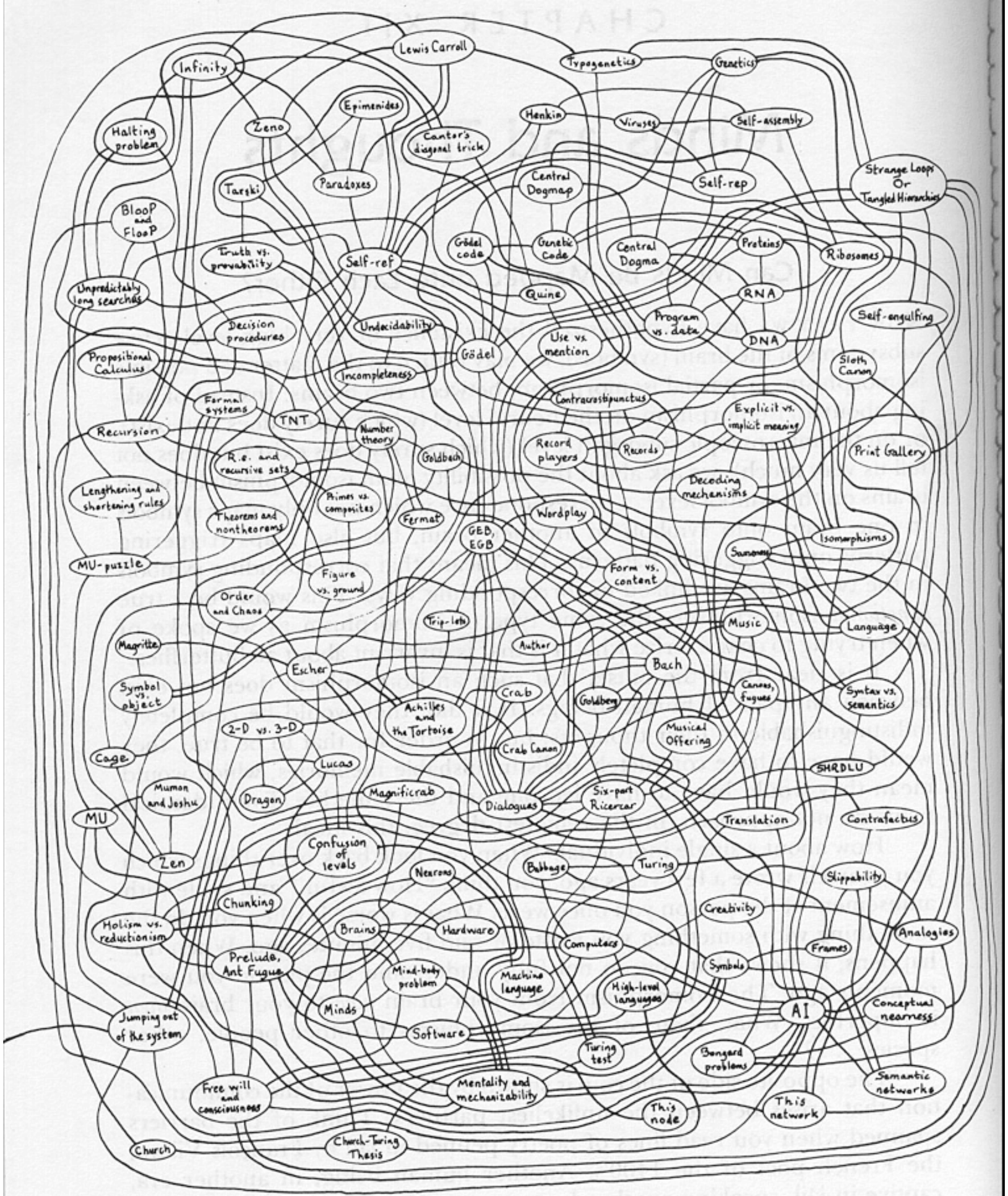
cnn.com

boingboing.net

wired.com

apple.com

Webpages as Graphs http://www.visualcomplexity.com/vc/project.cfm?id=333



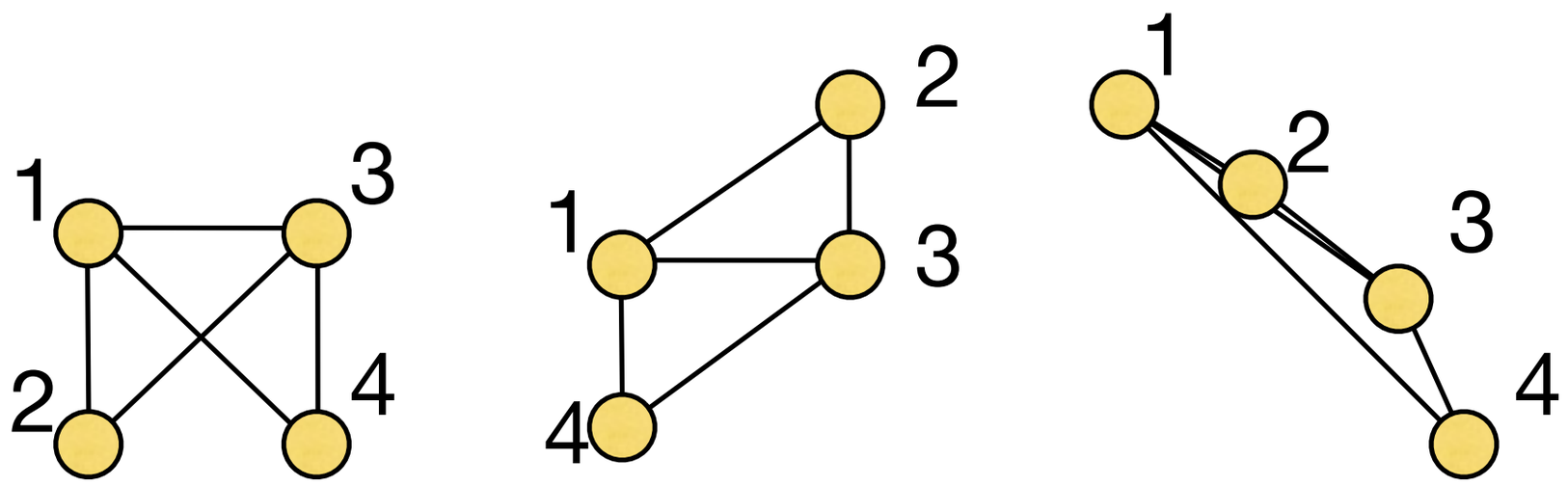
Hofstadter 1979

Defining Graphs

- A graph G consists of a collection of vertices (or nodes) V and a set of edges E, consisting of vertex pairs.

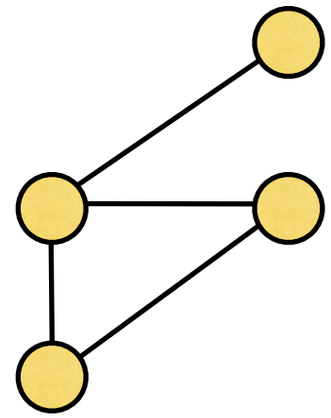
- An edge e_{xy} = (x,y) connects two vertices x and y.

- For example: V={1,2,3,4}, E={(1,2),(1,3),(2,3),(3,4),(4,1)}

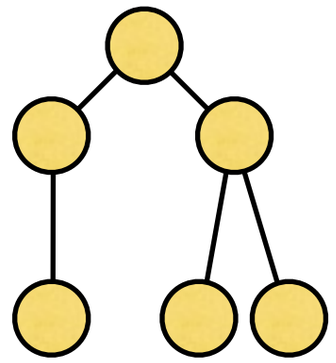


Graphs vs. Trees

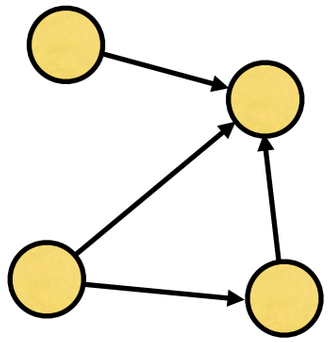
- Graphs
  - Model relations amount data
  - Have nodes and edges



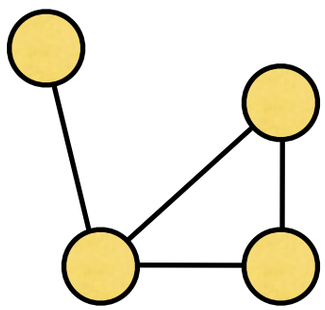
- Trees
  - Graphs with hierarchical structure
    - A connected graph with N-1 edges
  - Nodes referred to as parents and children



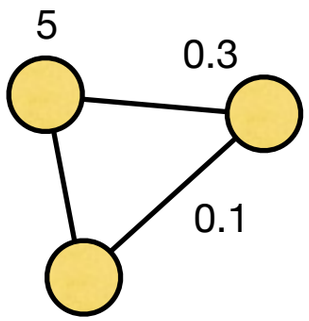
Graph Terminology



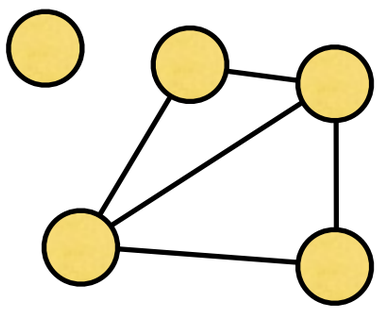
A directed graph



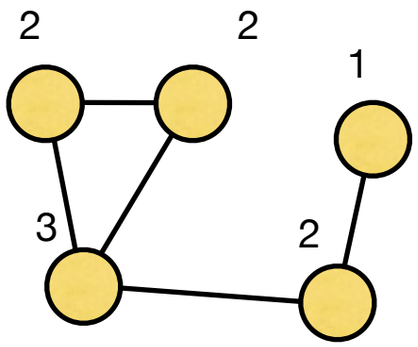
An undirected graph



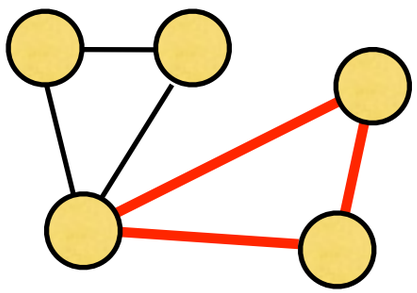
Weighted



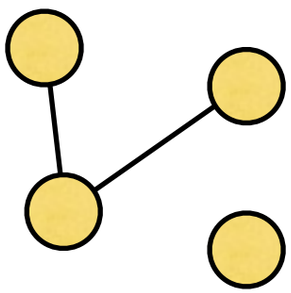
Unconnected



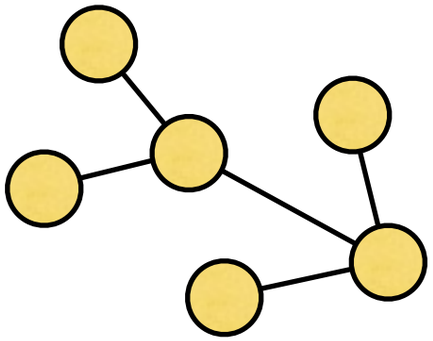
Node degrees



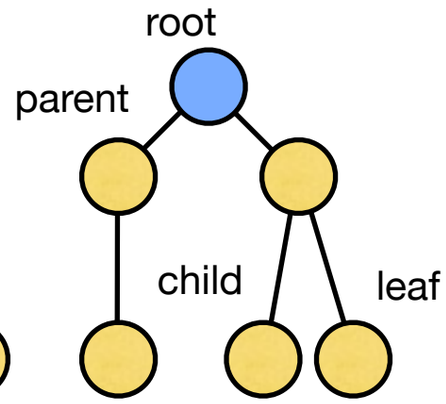
A **cycle**



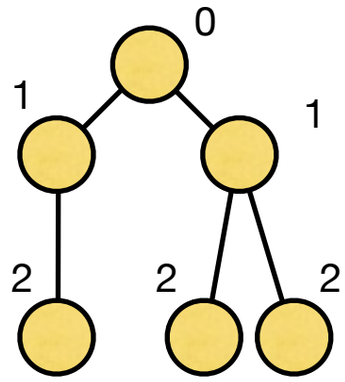
An acyclic graph



A connected acyclic graph, a.k.a. a **tree**



A rooted tree or hierarchy



Node depths

Graph Visualization and Navigation in Information Visualization: A Survey

Ivan Herman, *Member*, *IEEE Computer Society*, Guy Melançon, and M. Scott Marshall

Abstract—This is a survey on graph visualization and navigation techniques, as used in information visualization. Graphs appear in numerous applications such as web browsing, state-transition diagrams, and data structures. The ability to visualize and to navigate in these potentially large, abstract graphs is often a crucial part of an application. Information visualization has specific requirements, which means that this survey approaches the results of traditional graph drawing from a different perspective.

Index Terms—Information visualization, graph visualization, graph drawing, navigation, focus+context, fish-eye, clustering.

1 Introduction

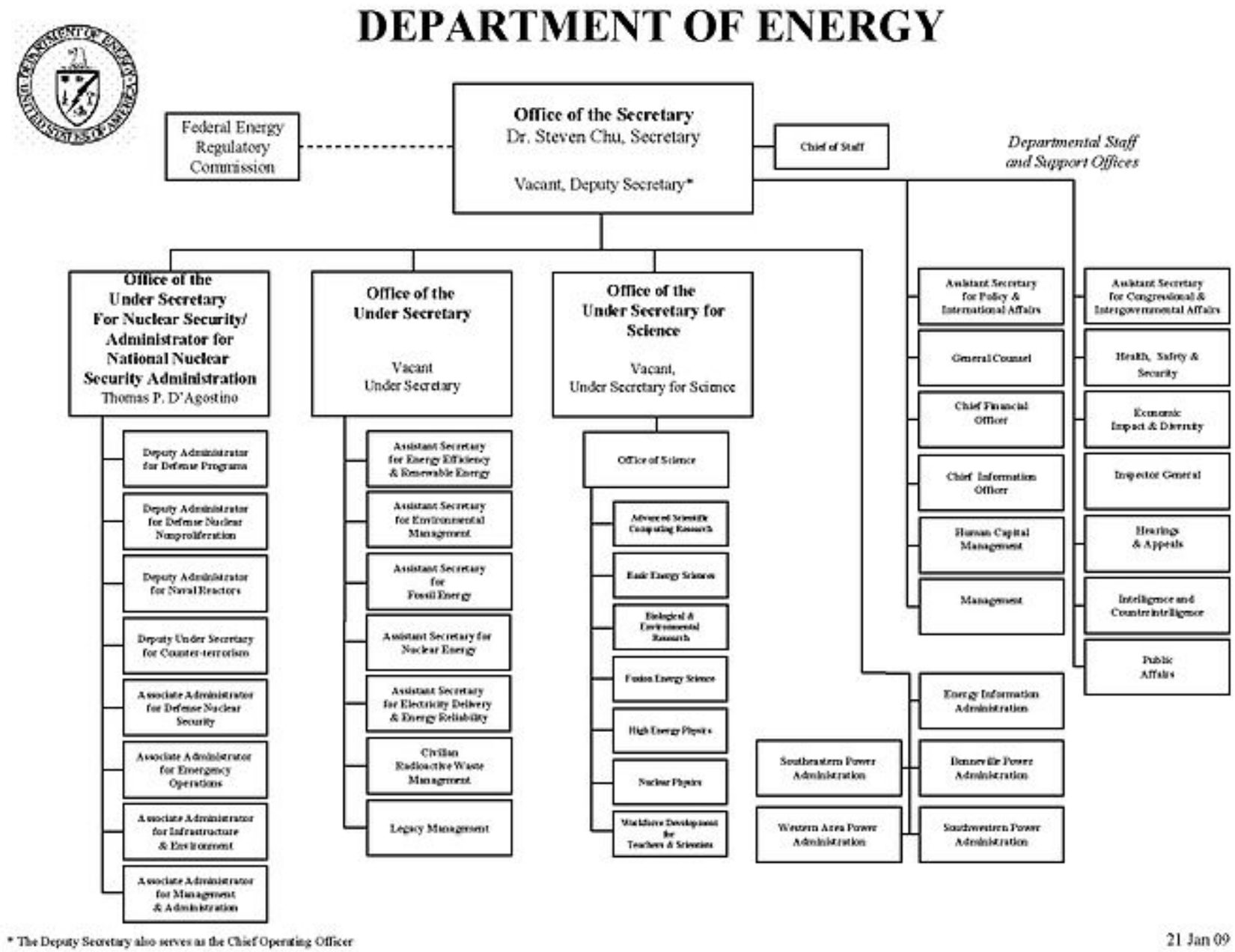
Although the visualization of graphs is the subject of this survey, it is *not* about graph drawing in general. Excellent bibliographic surveys [4], [34], books [5], or even on-line tutorials [26] exist for graph drawing. Instead, the handling of graphs is considered with respect to information visualization.

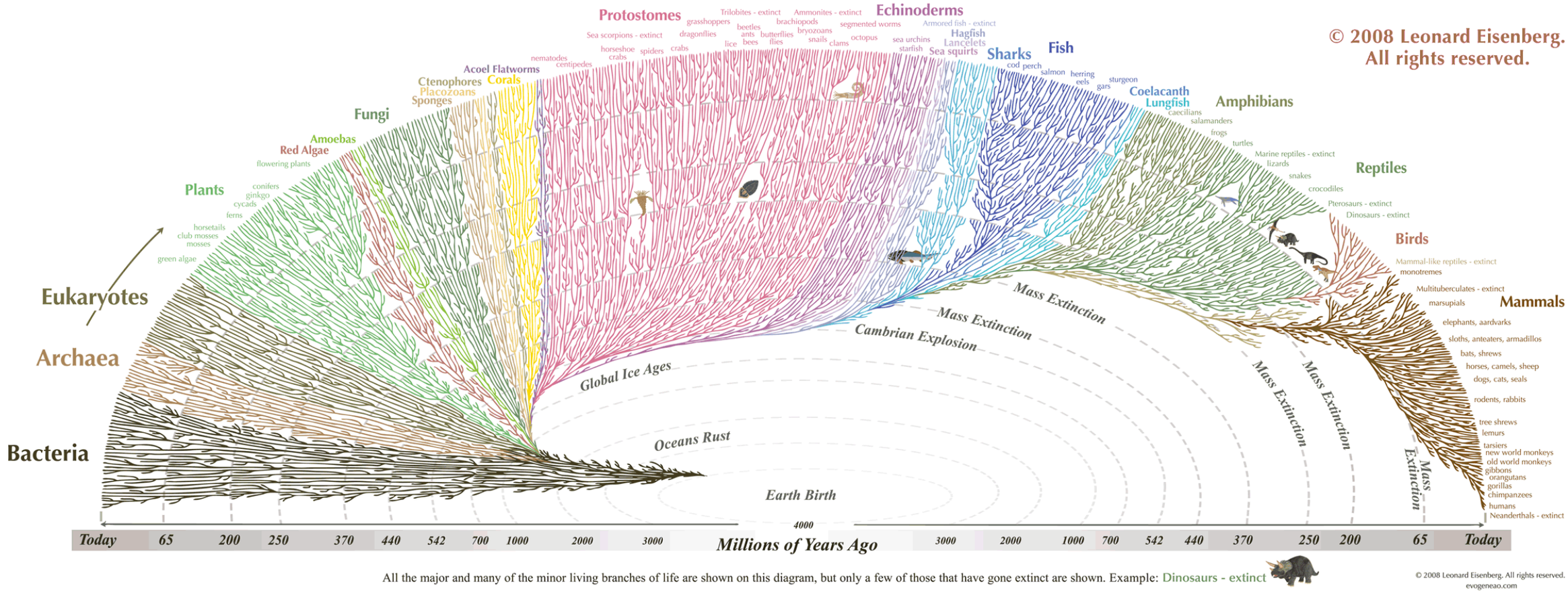
Information visualization has become a large field and "subfields" are beginning to emerge (see, for example, Card et al. [16] for a recent collection of papers from the last decade). A simple way to determine the applicability of graph visualization is to consider the following question: *Is there an inherent relation among the data elements to be visualized*? If the answer to the question is "no," then data elements are "unstructured" and the goal of the information visualization system might be to help discover relations among data through visual means. If, however, the answer to the question is "yes," then the data can be represented by the nodes of a graph, with the edges representing the relations.

I?" "Where is the file that I'm looking for?" Other familiar types of graphs include the hierarchy illustrated in an organizational chart and taxonomies that portray the relations between species. Web site maps are another application of graphs, as well as browsing history. In biology and chemistry, graphs are applied to evolutionary trees, phylogenetic trees, molecular maps, genetic maps, biochemical pathways, and protein functions. Other areas of application include object-oriented systems (class browsers), data structures (compiler data structures in particular), real-time systems (state-transition diagrams, Petri nets), data flow diagrams, subroutine-call graphs, entity relationship diagrams (e.g., UML and database structures), semantic networks and knowledge-representation diagrams, project management (PERT diagrams), logic programming (SLD-trees), VLSI (circuit schematics), virtual reality (scene graphs), and document management systems. Note that the information isn't always guaranteed to be in a purely hierarchical format—this necessitates techniques

In “relational data”, it’s the **relationship between data items** that matters

- The reports-to relationship in an organization



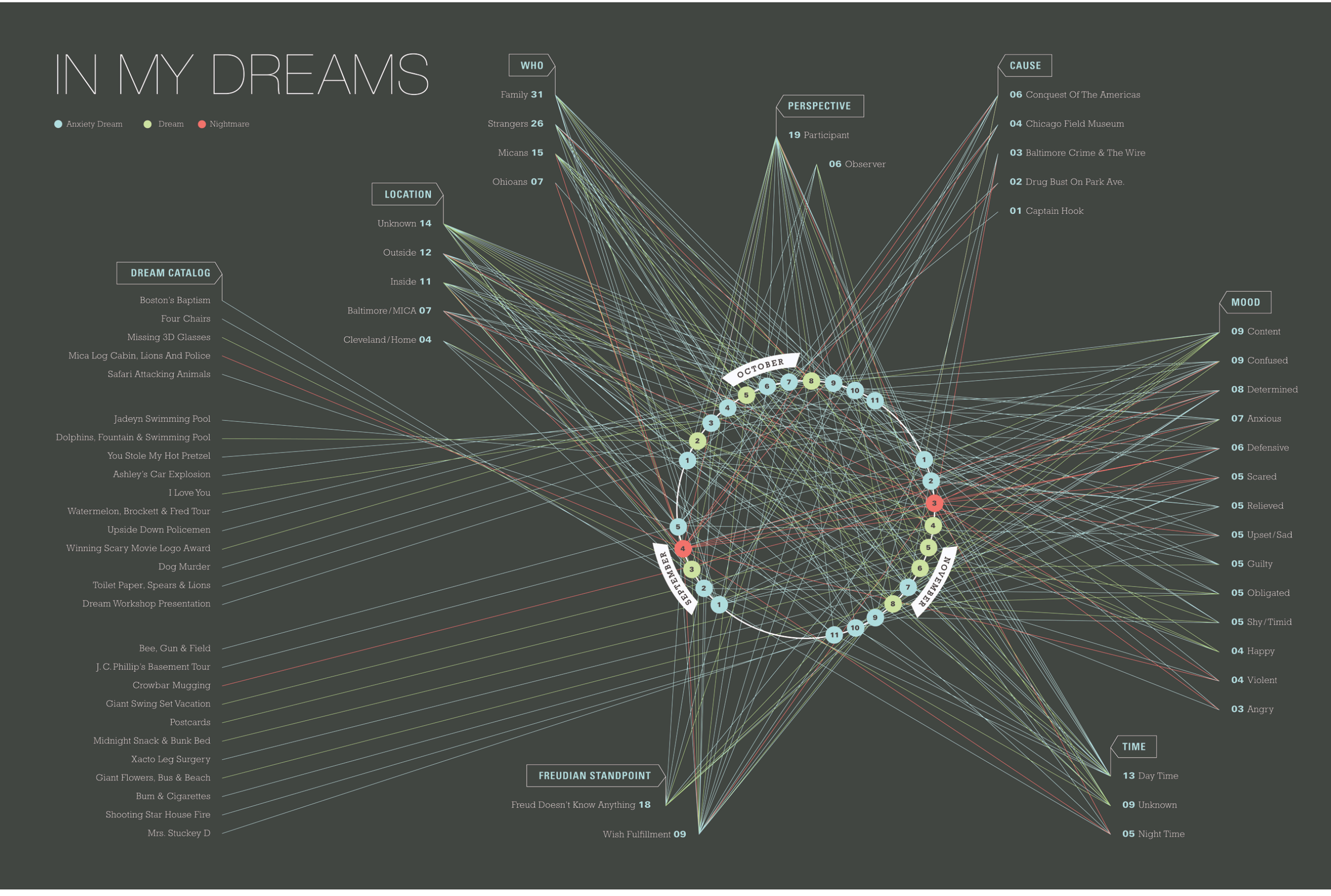


- The “tree of life”
  - evolution of species creates branching mechanism and “ancestor-of” relationship

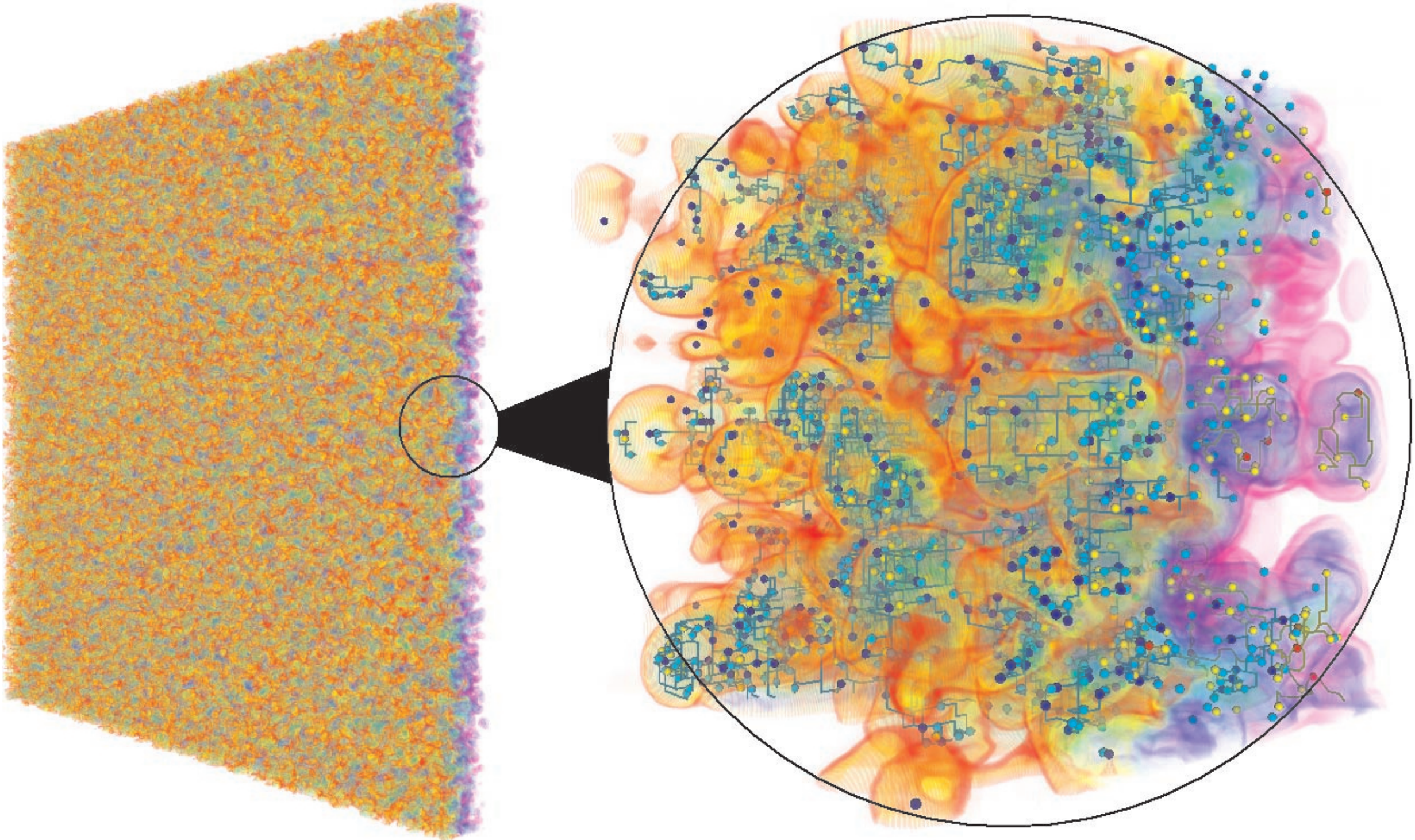
Spatial Layout

- Primary concern of graph drawing is the spatial layout of nodes and edges

- Often (but not always) the goal is to effectively depict the graph structure
  - Connectivity, path-following
  - Network distance
  - Clustering
  - Ordering (e.g., hierarchy level)



Kailie Parish, In My Dreams https://cargocollective.com/kailieparrish/In-My-Dreams



Gyulassy, Bremer, Hamann, Pascucci, 2008

Visualizing Trees, Part 1

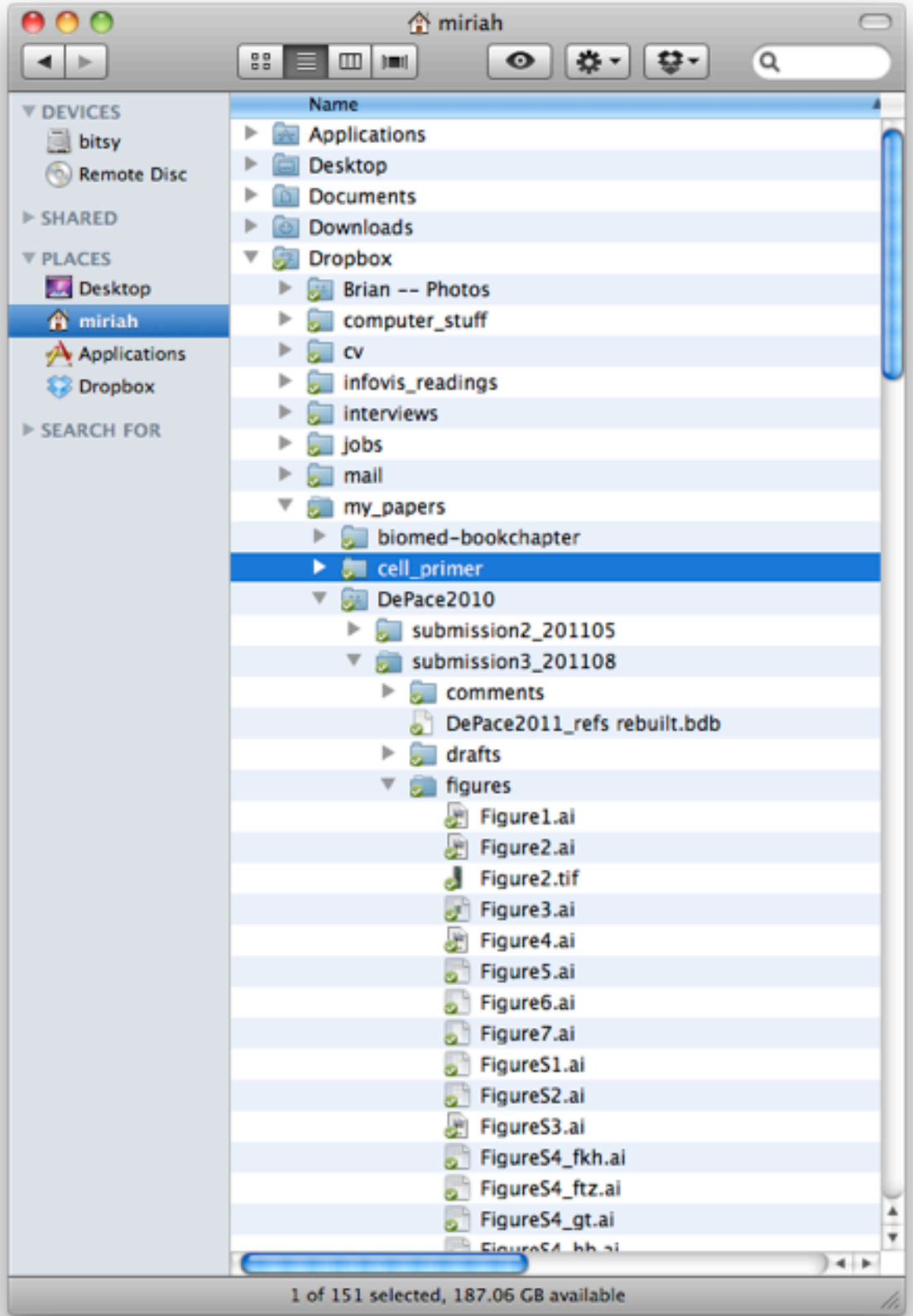
Tree Layout

- Recursion makes it elegant and fast to draw trees
- Approaches:

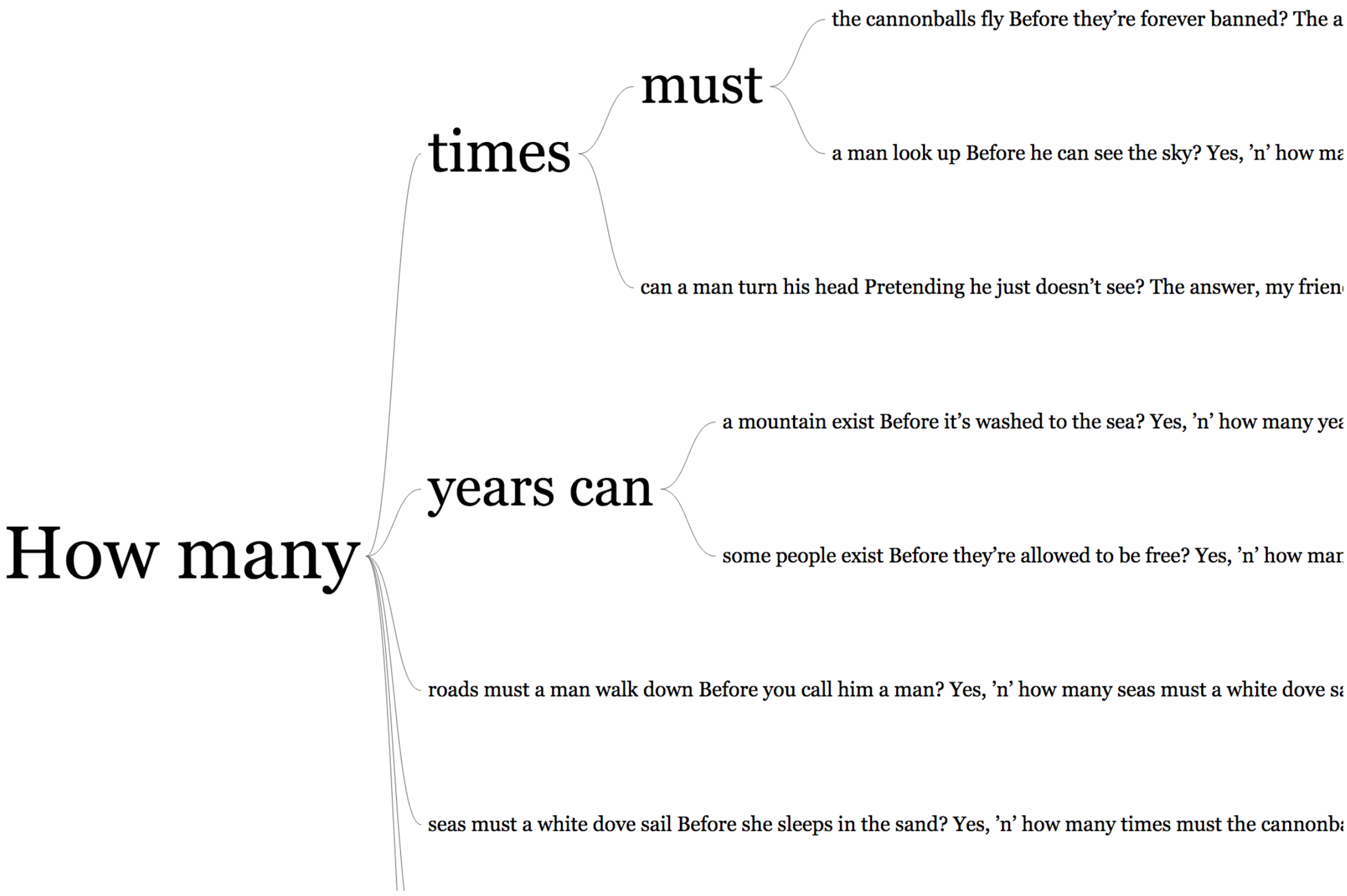
1. **Indentation**
2. **Node-link Diagrams**
3. Enclosure
4. Layering

Indentation

- Place all items along vertically spaced rows
- Indentation used to show parent/ child relationships
- Commonly used as a component in an interface
- Breadth and depth contend for space
- Often requires a great deal of scrolling



Word Tree



https://www.jasondavies.com/wordtree/

Node-Link Diagrams

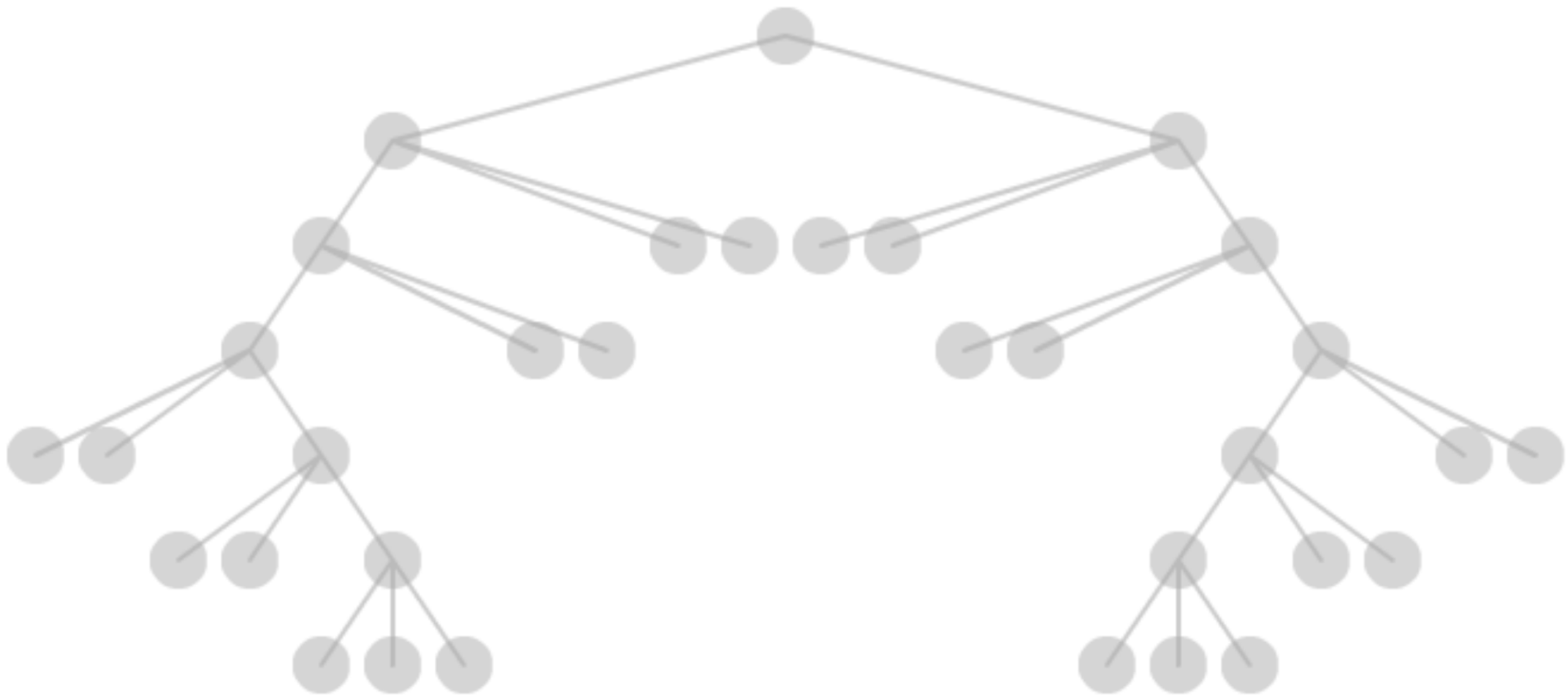
- Nodes are distributed in space, connected by straight or curved lines

- Typical approach is to use 2D space to break apart breadth and depth

- Often space is used to communicate hierarchical orientation

Basic Recursive Approach

- Repeatedly divide space for subtrees by leaf count
  - Breadth of tree along one dimension
  - Depth along the other dimension
- Problem: exponential growth of breadth

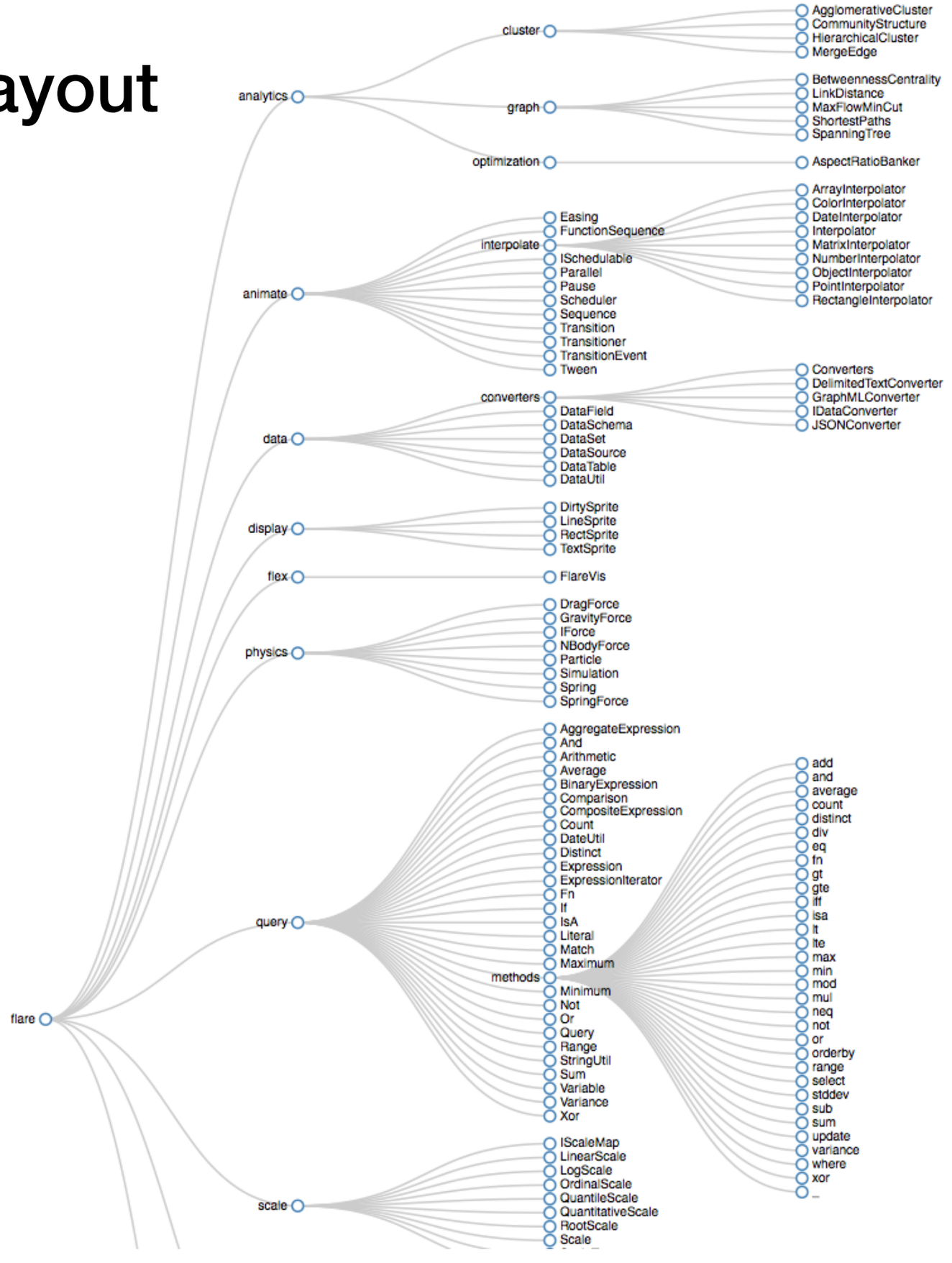


Reingold-Tilford “Tidy” Layout

- Goal: make smarter use of space, maximize density and symmetry

- Design concerns
  - Clearly encode depth level
  - No edge crossings
  - Ordering and symmetry preserved
  - Compact layout (don’t waste space)

- Approach
  - Bottom up recursive approach
  - For each parent make sure every subtree is drawn
  - Pack subtrees as closely as possible
  - Center parent over subtrees

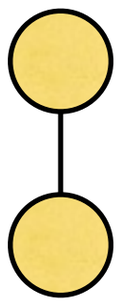


https://blocks.roadtolarissa.com/mbostock/4339184

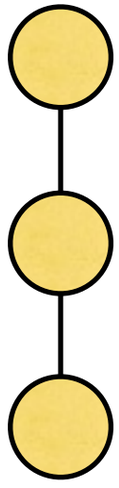
Reingold-Tilford

Reingold-Tilford

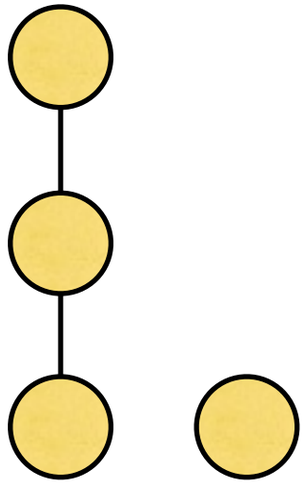
Reingold-Tilford



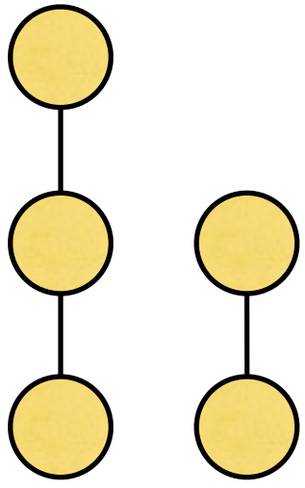
Reingold-Tilford



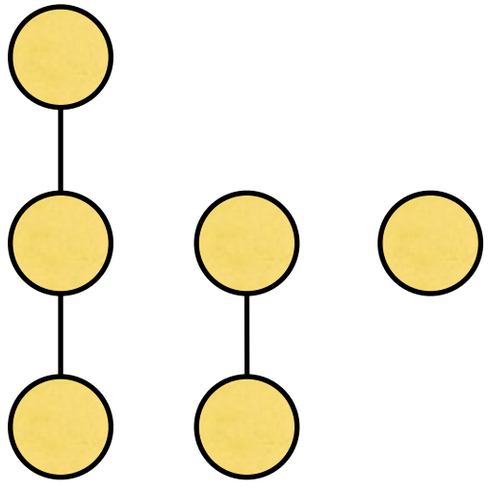
Reingold-Tilford



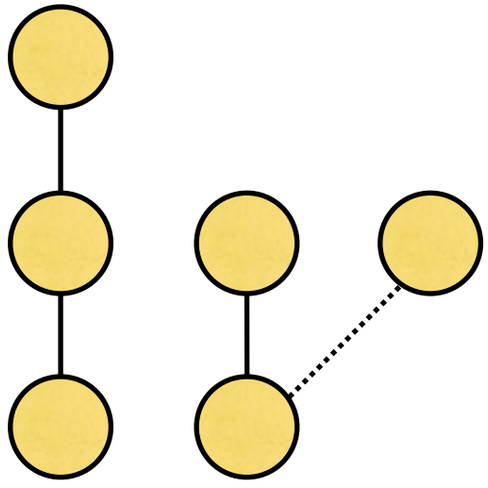
Reingold-Tilford



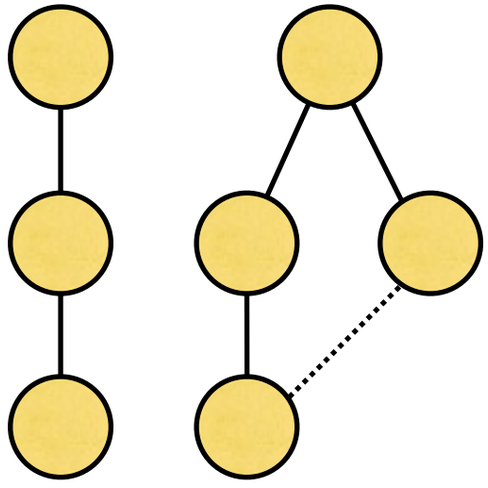
Reingold-Tilford



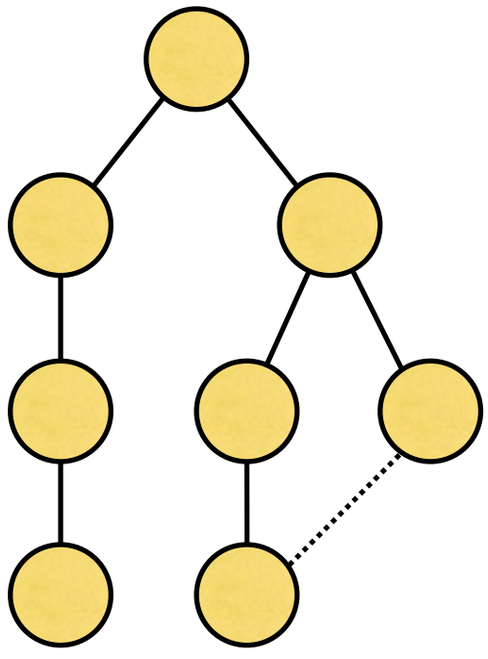
Reingold-Tilford



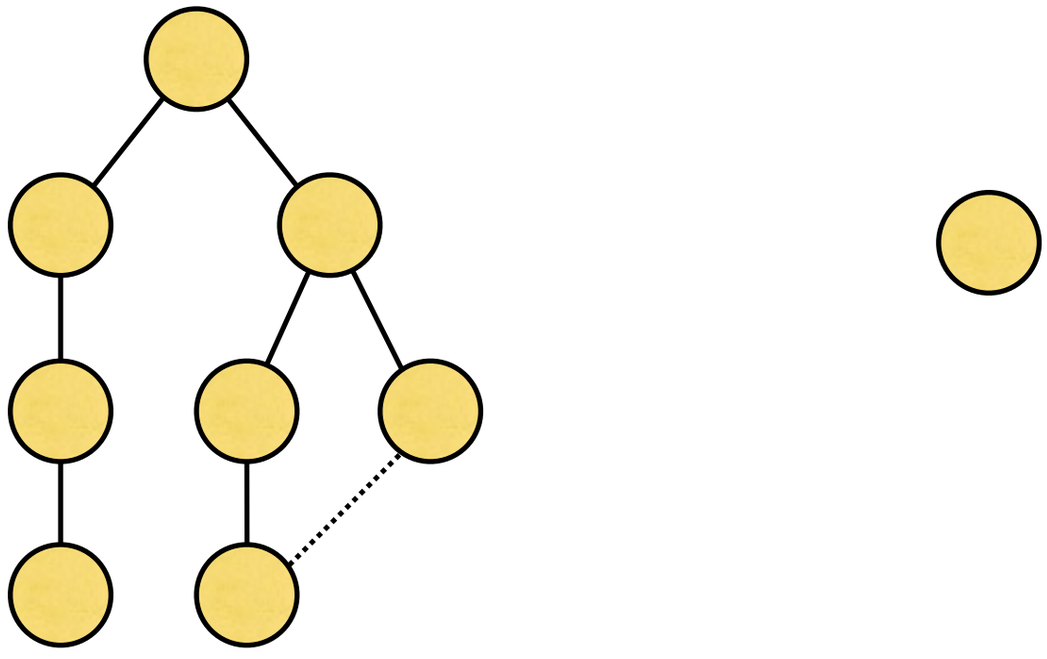
Reingold-Tilford



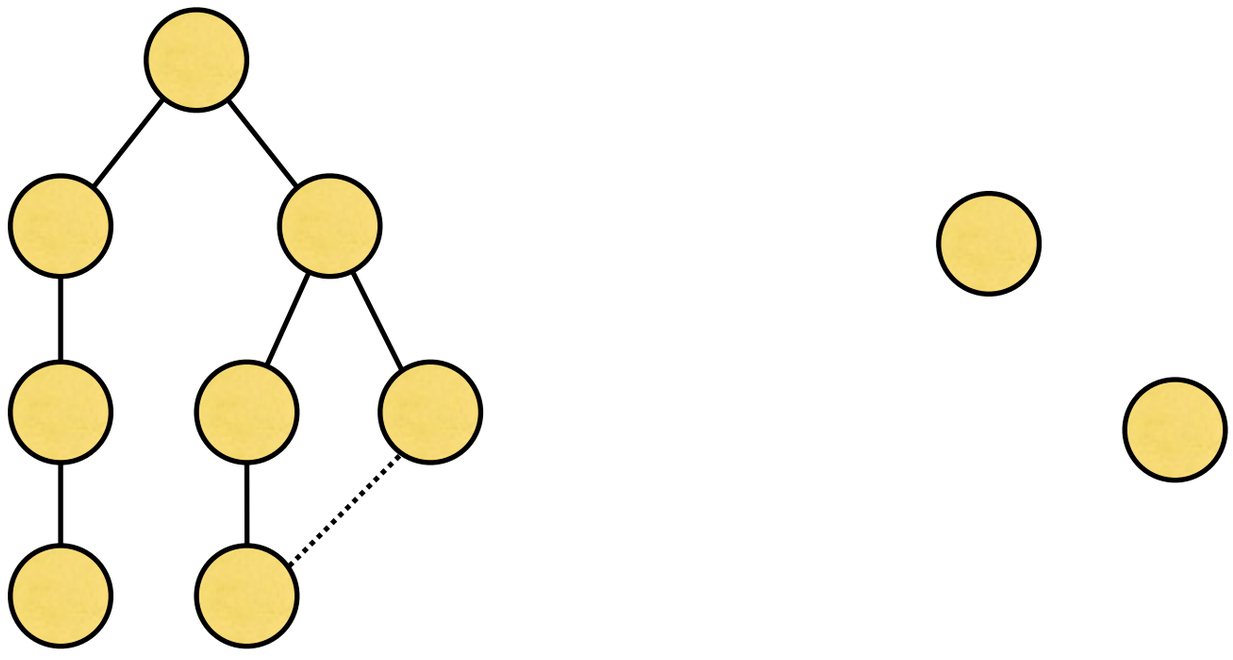
Reingold-Tilford



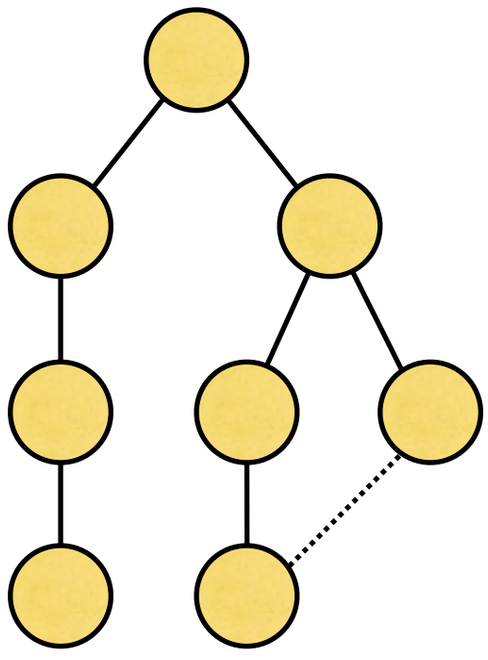
Reingold-Tilford



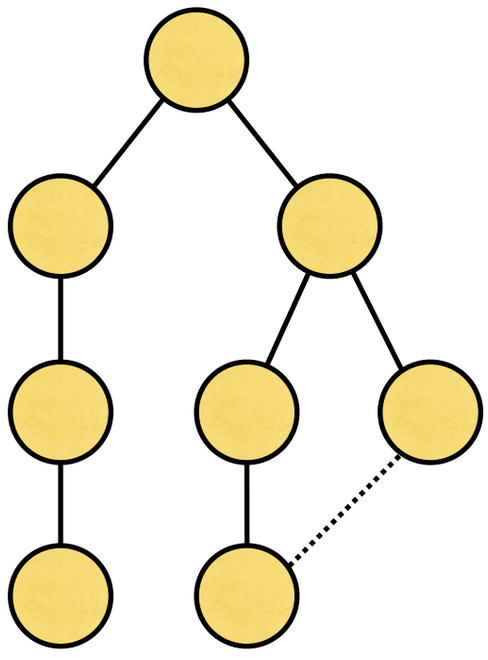
Reingold-Tilford

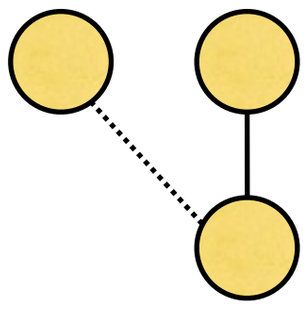


Reingold-Tilford

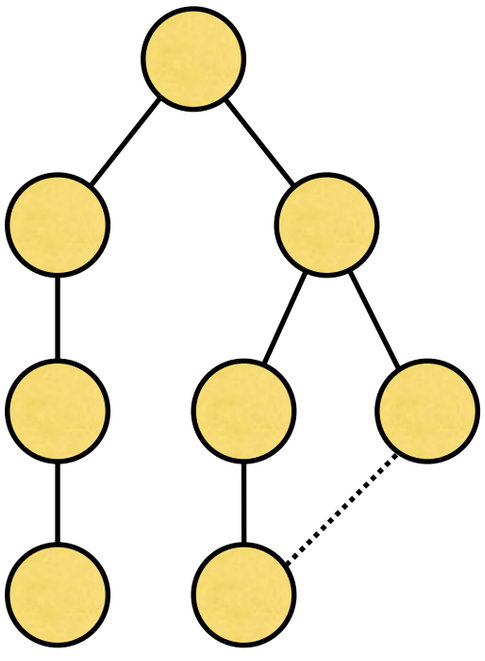


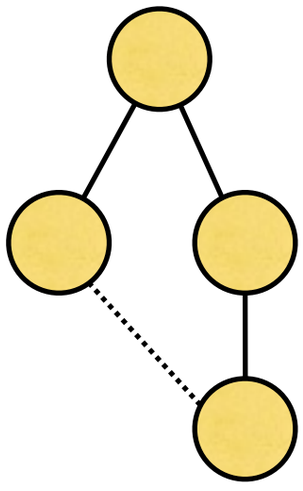
Reingold-Tilford



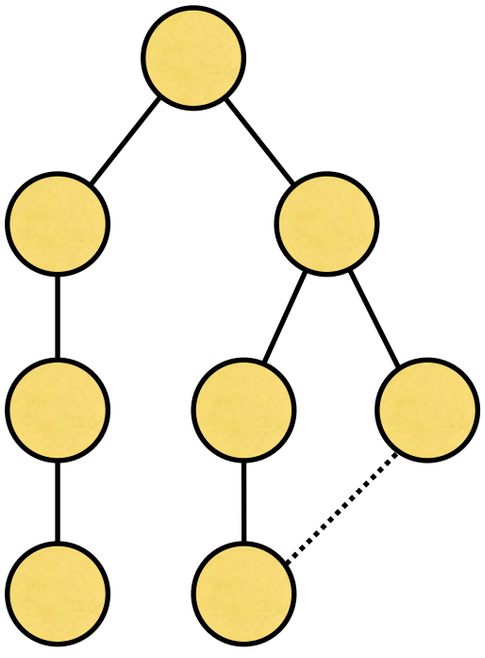


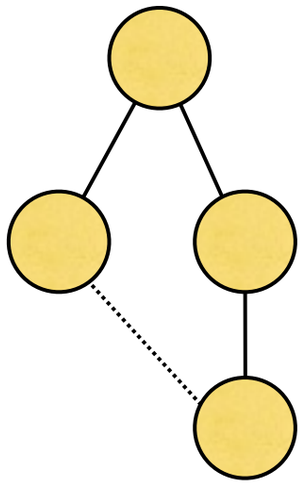
Reingold-Tilford



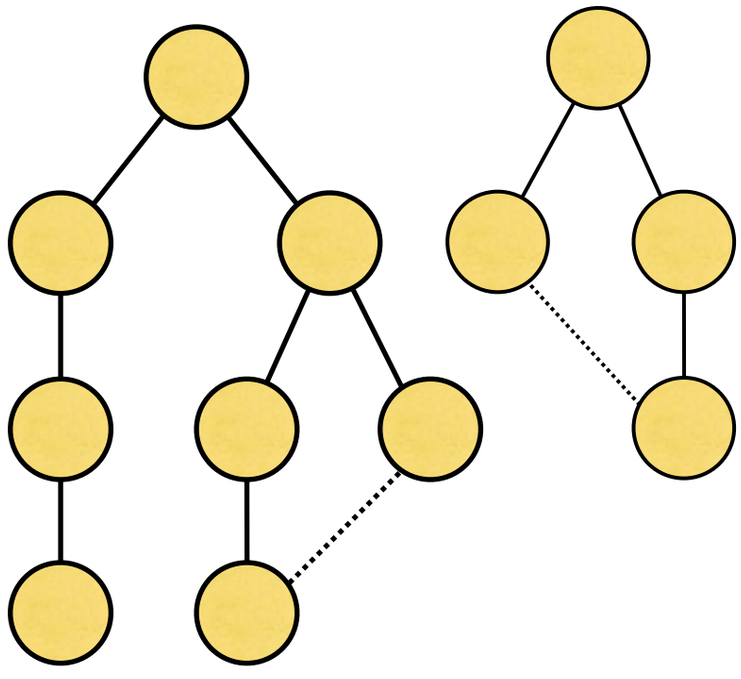


Reingold-Tilford

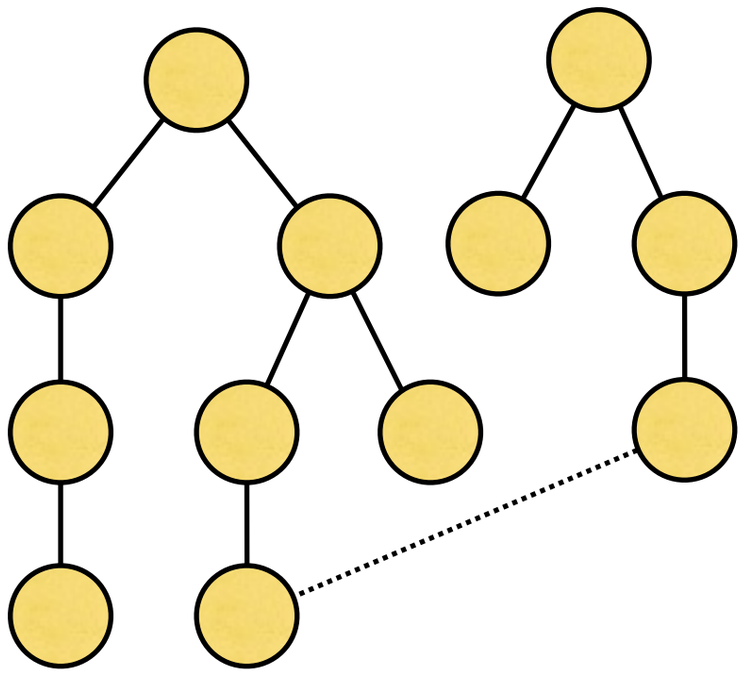




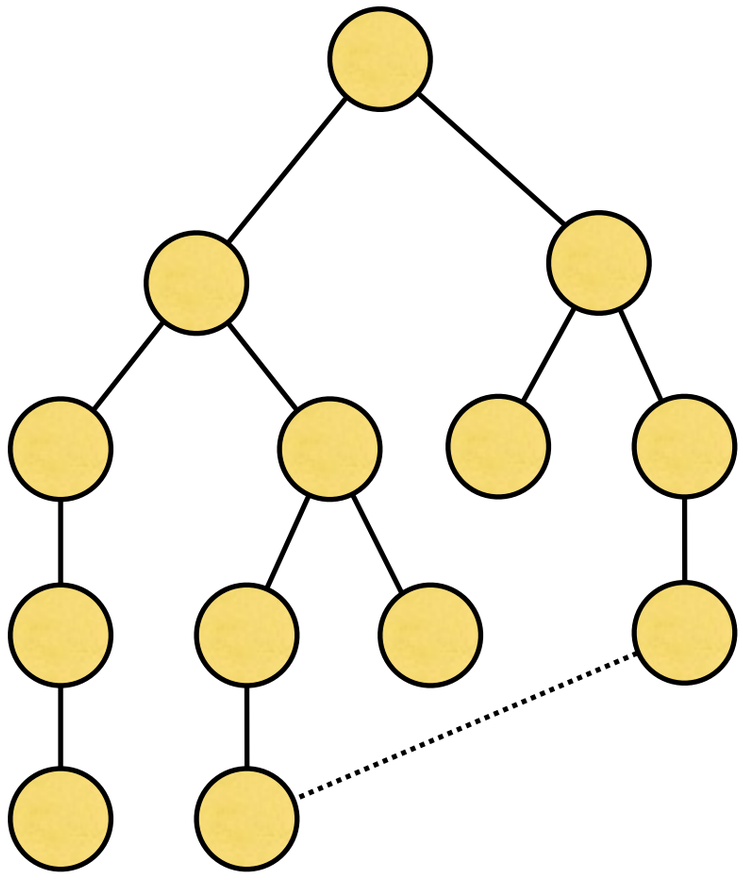
Reingold-Tilford



Reingold-Tilford



Reingold-Tilford

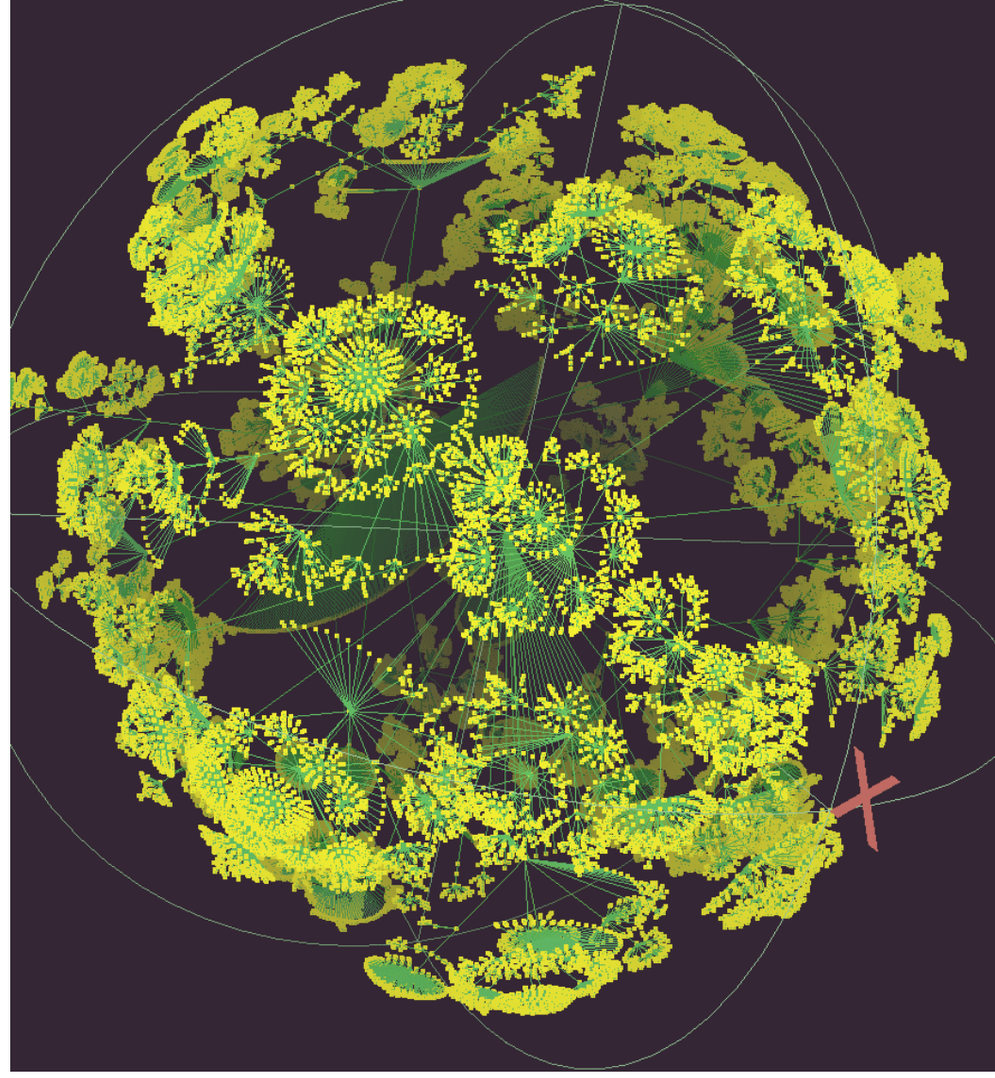


Reingold-Tilford Algorithm

- Bottom-up tree traversal
- y-coord is the depth of the node, x-coords are “locally defined” (so merge trees first is arbitrary)
- Merge trees:
  - push right tree as close as possible to left tree (this is where the contour comes in)
  - position **shifts** saved at each node
  - parent nodes are centered above direct children
- Final top-down pass to convert shifts to positions

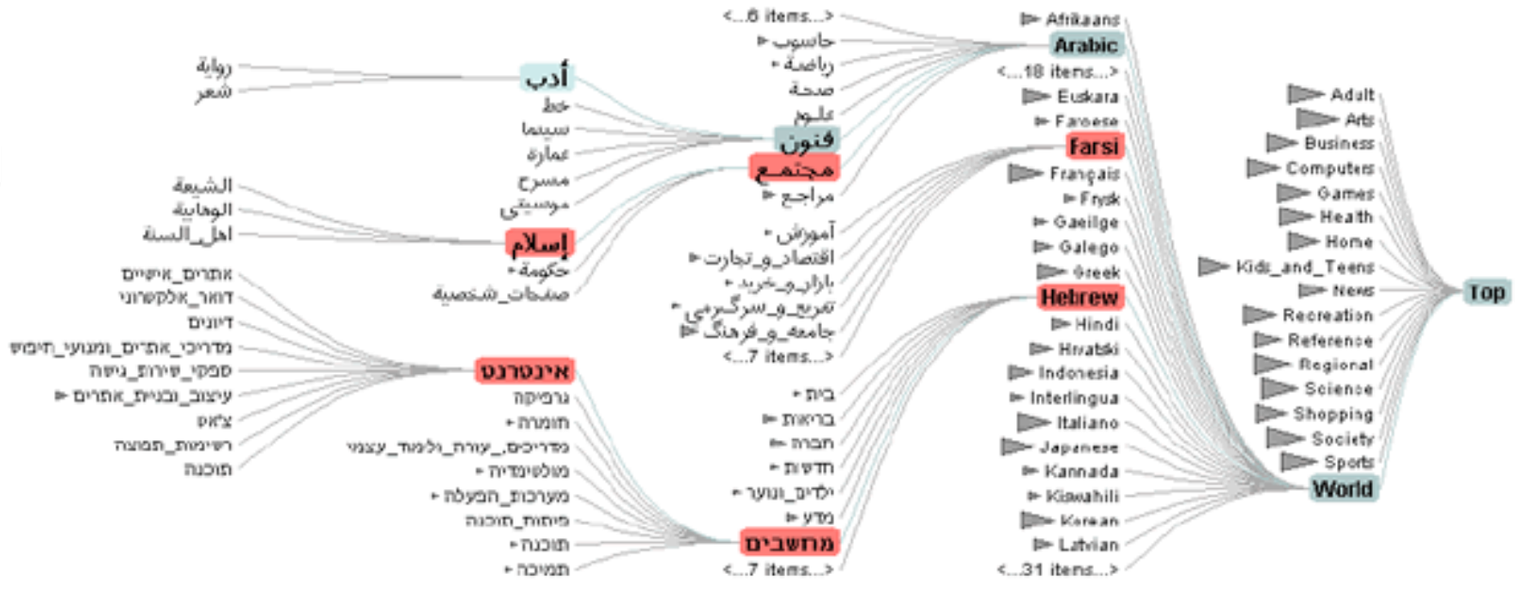
Problems with Node-Link

- Scale
  - Tree breadth often grows exponentially
  - Even with tidier layout, quickly run out of space



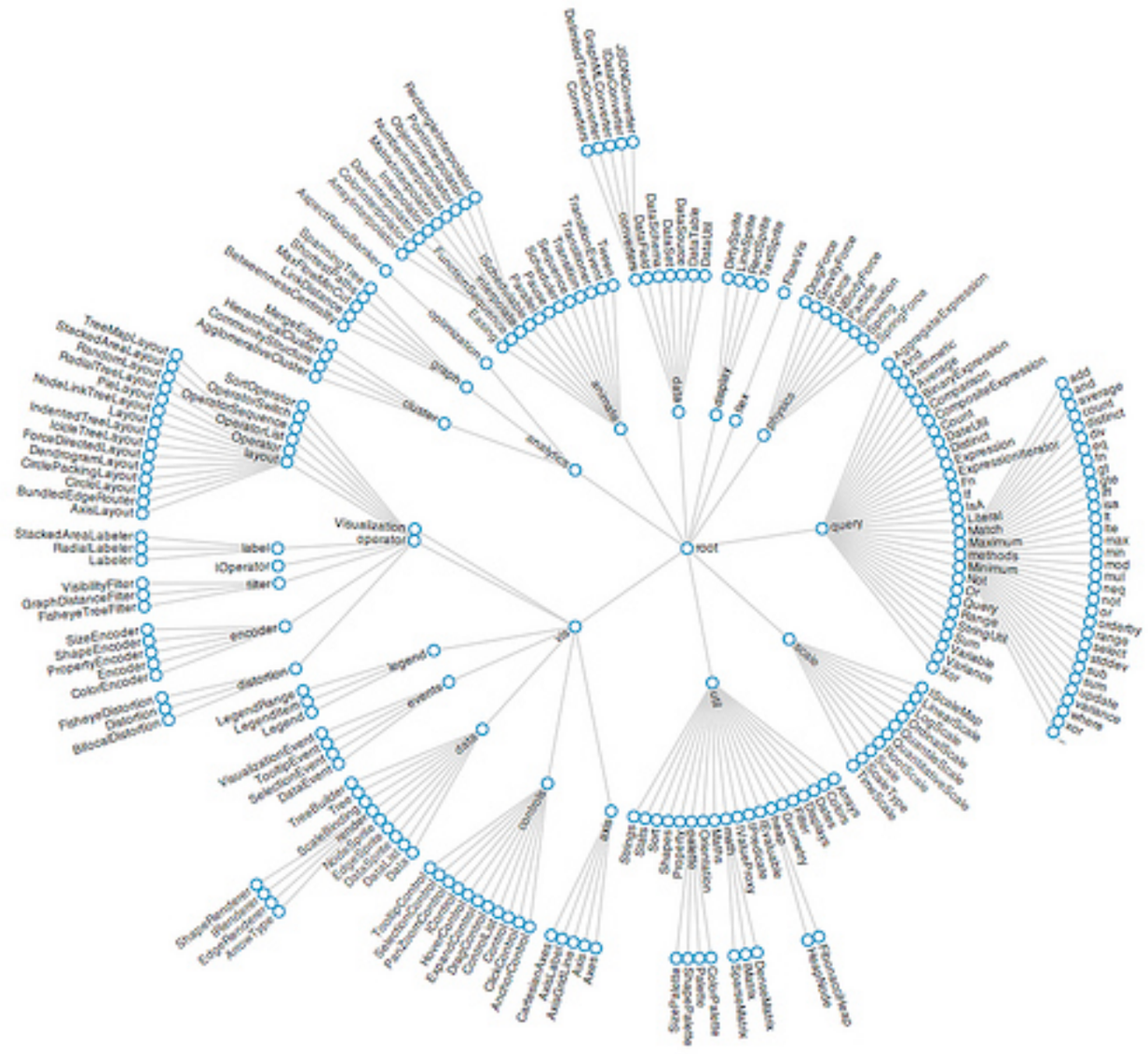
http://www.caida.org/tools/visualization/walrus/

- Possible solutions
  - Filtering
  - Scrolling or Panning
  - Zooming
  - Aggregation
  - Alternative Layouts



http://vis.stanford.edu/papers/doitrees-revisited

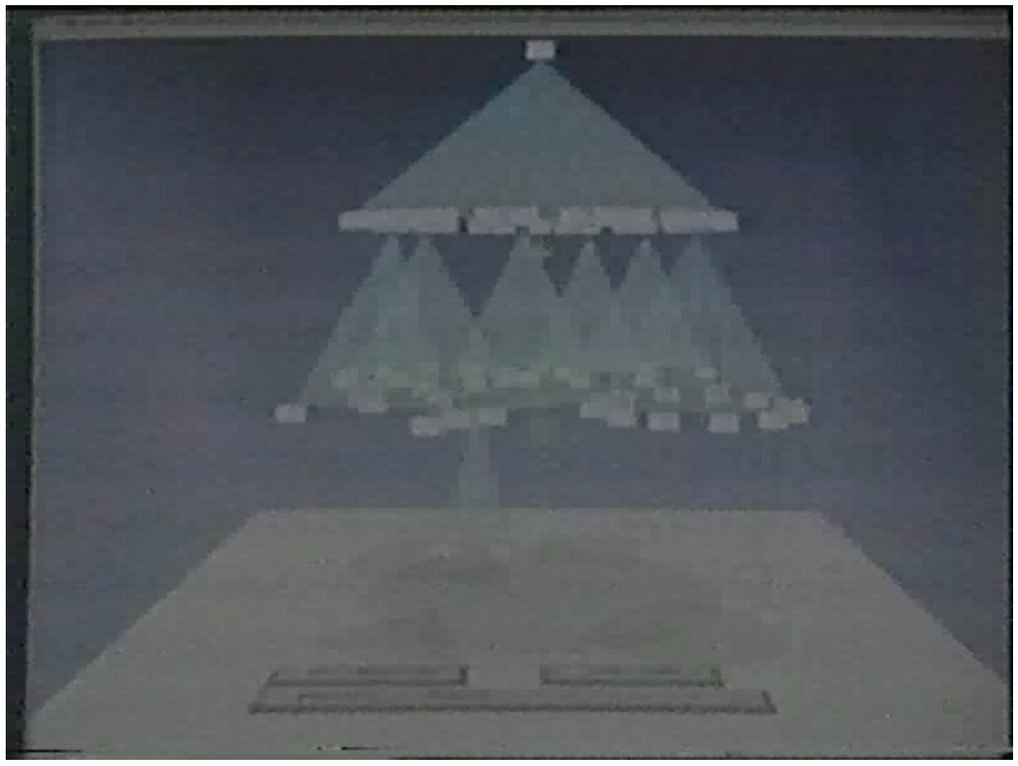
Radial Layout

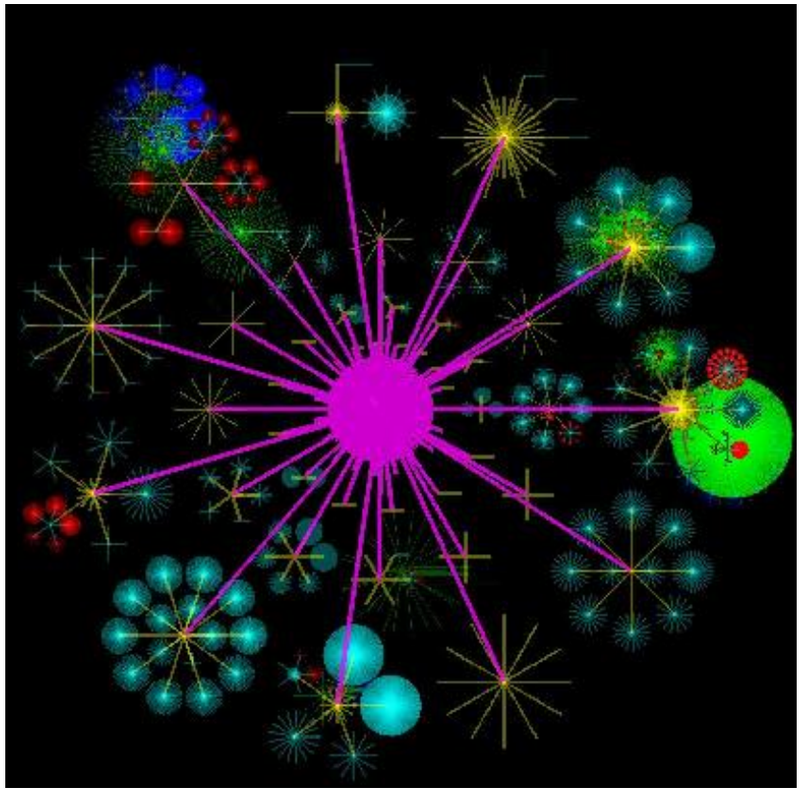


- Node-link diagram in polar coordinates
- Radius encodes depth with root in center
- Angular sectors assigned to subtrees
- Reingold-Tilford style rules can be applied

http://seeingcomplexity.wordpress.com/2011/02/05/hierarchical-edge-bundles/

Circular Drawing of Trees

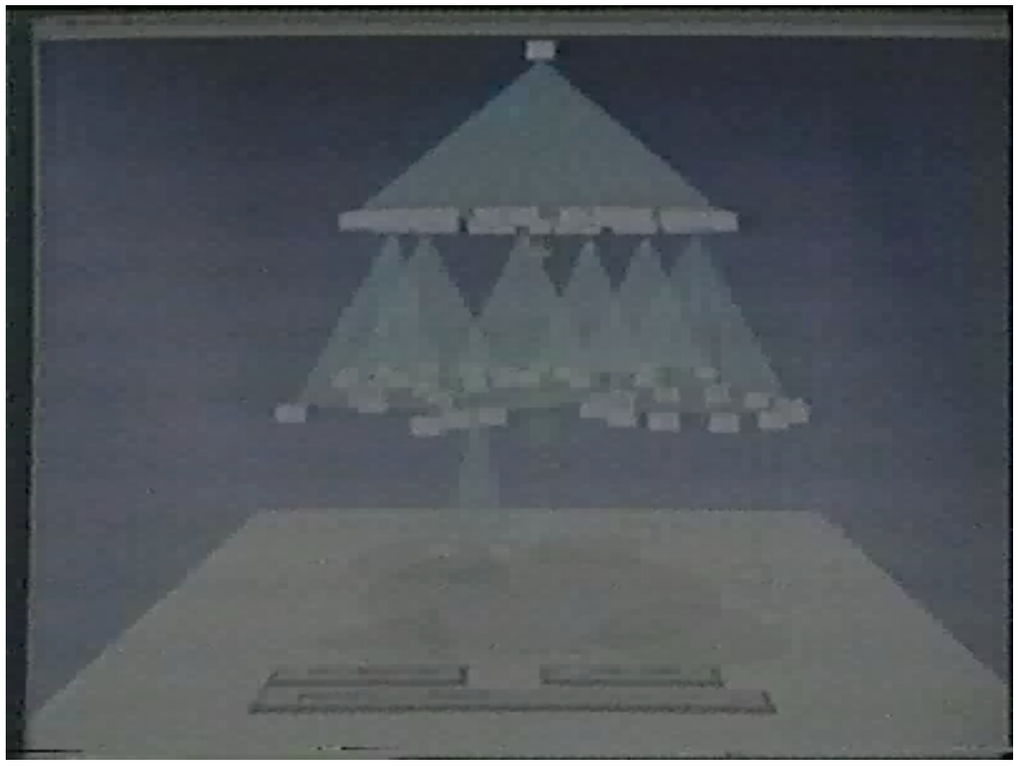


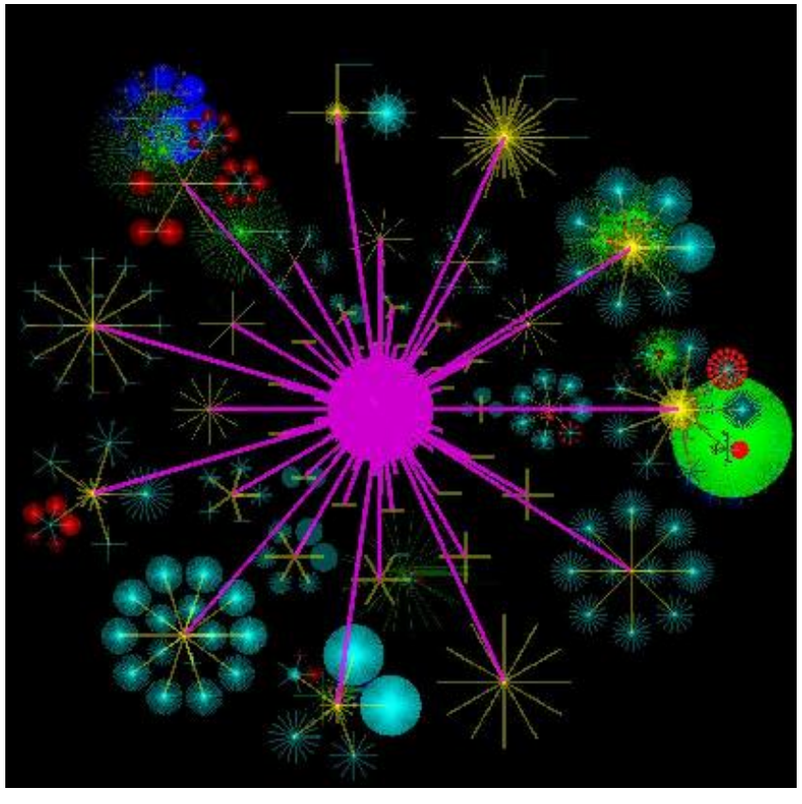


- Drawing in 3D forms
  Cone Trees

- **Balloon Trees** can be described as a 2D variant of a Cone Tree. Not just a flattening process, as circles must not overlap

Circular Drawing of Trees

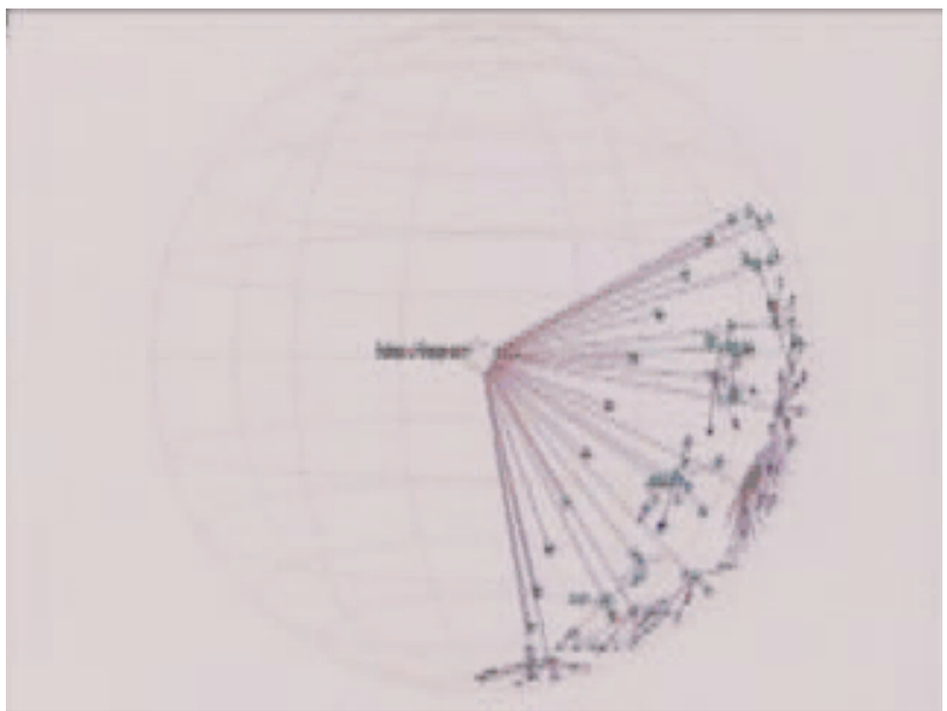




- Drawing in 3D forms
  Cone Trees

- **Balloon Trees** can be described as a 2D variant of a Cone Tree. Not just a flattening process, as circles must not overlap

Hyperbolic Layout

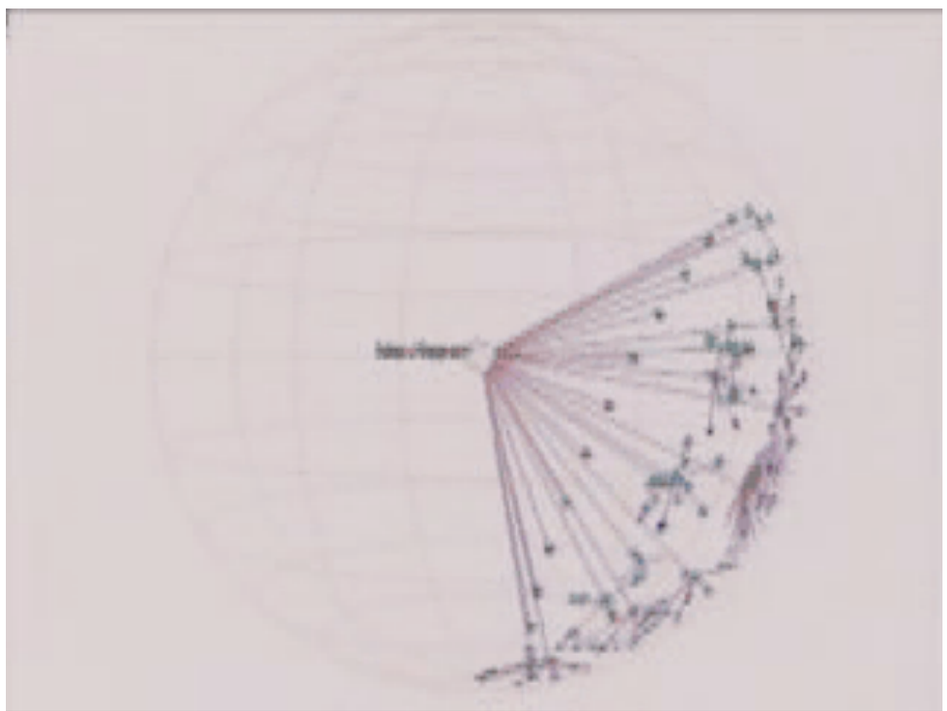


- Perform tree layout in hyperbolic geometry, then project the result on to the Euclidean plane.

- Why? Like tree breadth, the hyperbolic plane expands exponentially!

- Also computable in 3D, projected into a sphere.

Hyperbolic Layout



- Perform tree layout in hyperbolic geometry, then project the result on to the Euclidean plane.

- Why? Like tree breadth, the hyperbolic plane expands exponentially!

- Also computable in 3D, projected into a sphere.

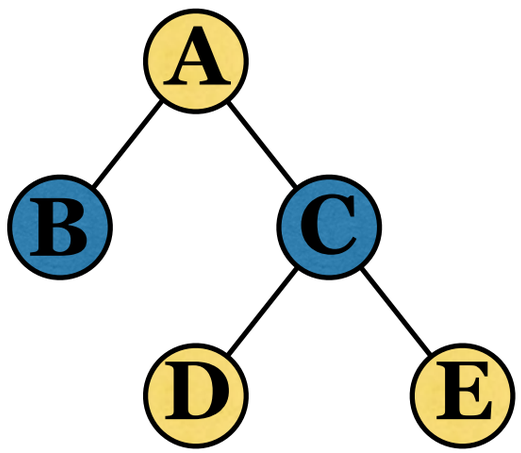
Visualizing Trees, Part 2

Tree Layout

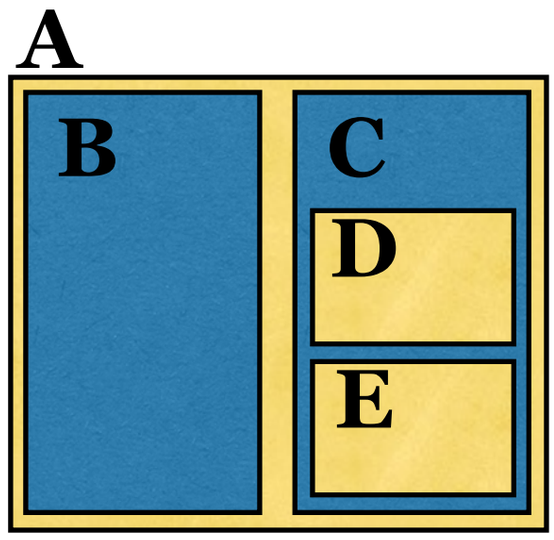
- Recursion makes it elegant and fast to draw trees
- Approaches:
  1. Indentation
  2. Node-link Diagrams
  3. **Enclosure**
  4. **Layering**

Enclosure Diagrams

- Encode structure using spatial enclosure
  - Often referred to as **treemaps**

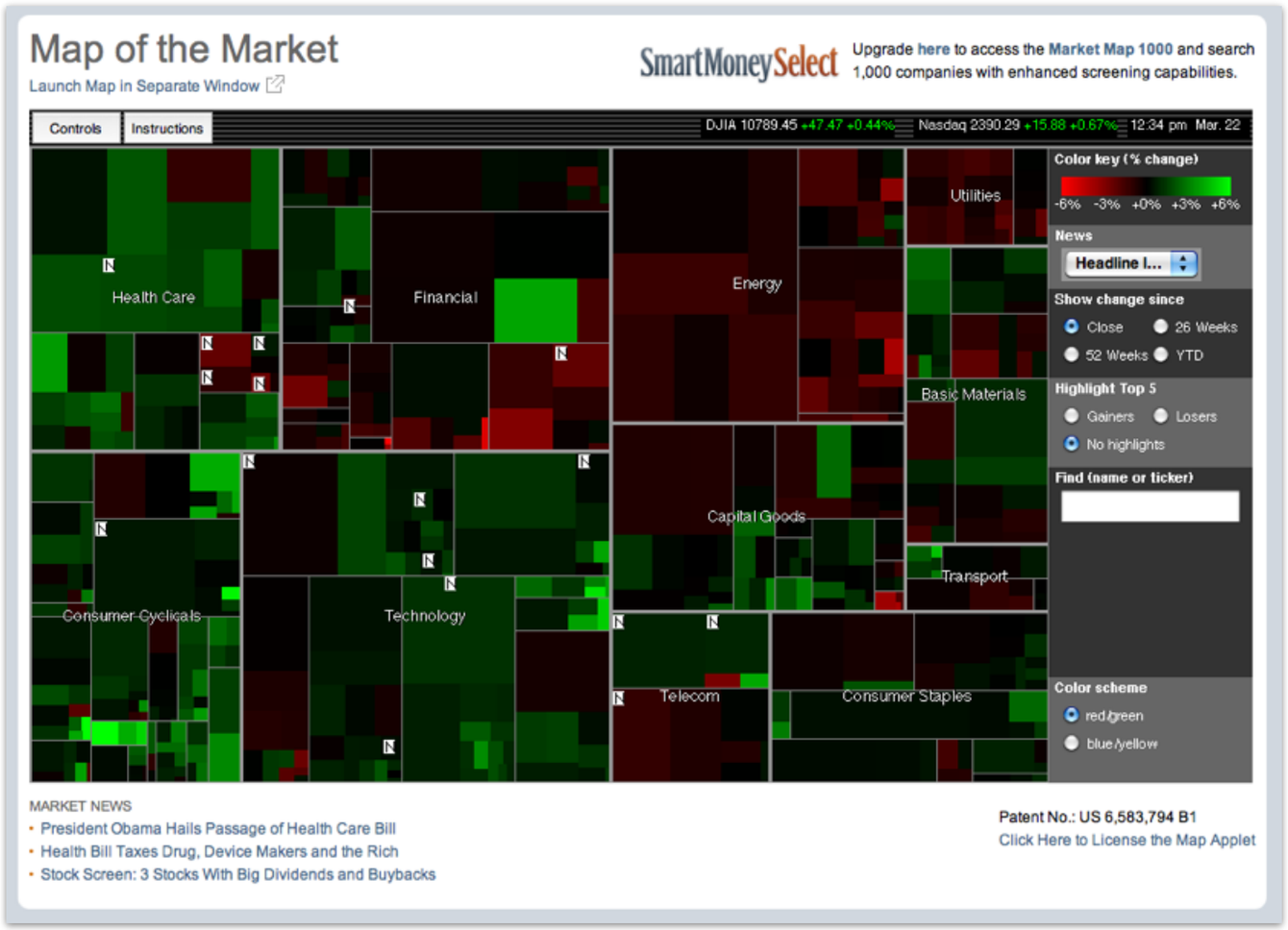


- Benefits
  - Provides single view of entire tree
  - Easier to spot small / large nodes



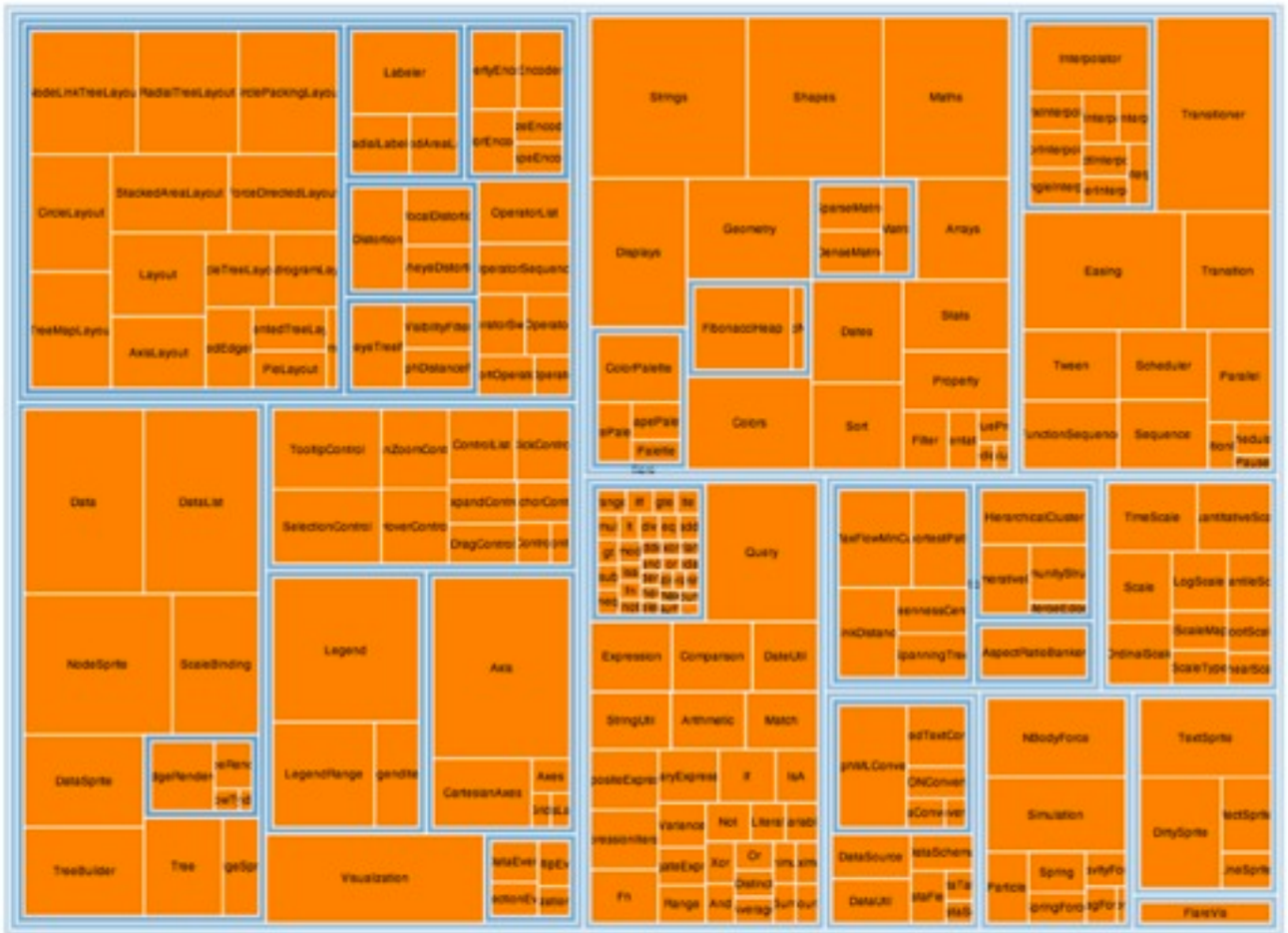
- Problems
  - Difficult to accurately read depth

Treemaps



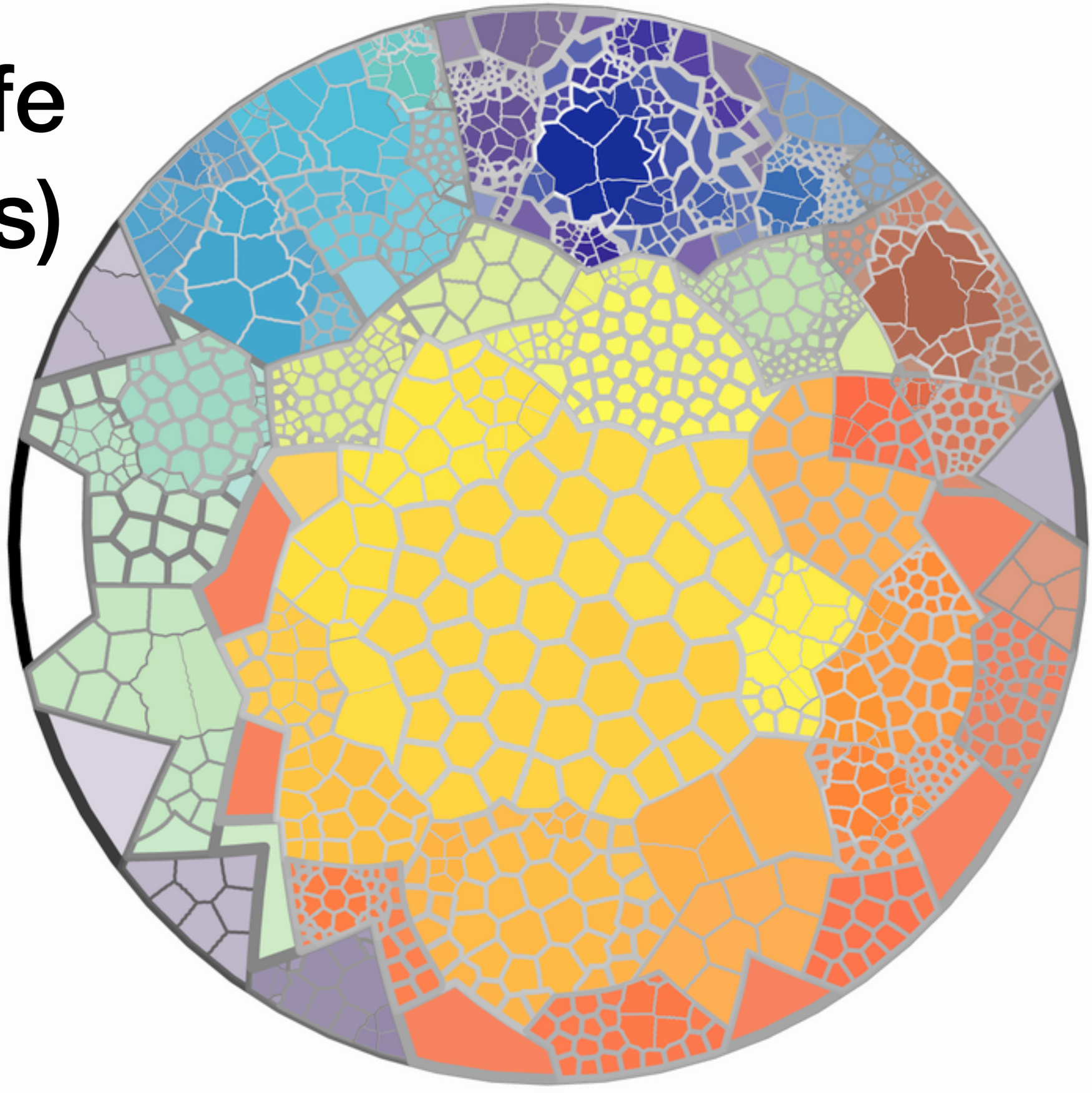
- Recursively fill space based on a size metric for nodes.
- Enclosure signifies hierarchy.
- Additional measures can be taken to control aspect ratio of cells.
- Often uses rectangles, but other shapes are possible, e.g., squares, circles, Voronoi tesselations.

Examples: http://www.bewitched.com/marketmap.html and https://finviz.com/map.ashx



http://hci.stanford.edu/jheer/files/zoo/

Tree of Life (mammals)



https://www.flickr.com/photos/arenamontanus/2037614308/in/set-72157594387083580/

Cushion Treemaps: Visualization of Hierarchical Information

Jarke J. van Wijk Huub van de Wetering
Eindhoven University of Technology
Dept. of Mathematics and Computing Science
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{*vanwijk, wstahw*}*@win.tue.nl*

Abstract

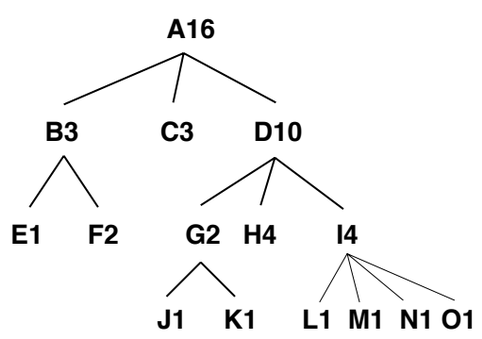
A new method is presented for the visualization of hierarchical information, such as directory structures and organization structures. Cushion treemaps inherit the elegance of standard treemaps: compact, space-filling displays of hierarchical information, based on recursive subdivision of a rectangular image space. Intuitive shading is used to provide insight in the hierarchical structure. During the subdivision ridges are added per rectangle, which are rendered with a simple shading model. The result is a surface that consists of recursive cushions. The method is efficient, effective, easy to use and implement, and has a wide applicability.

1 Introduction

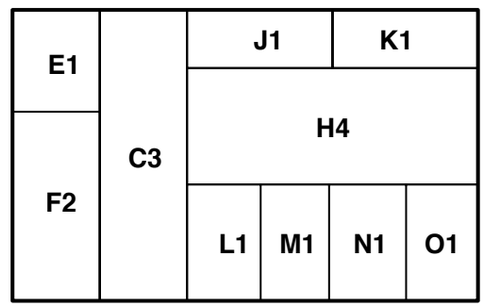
Hierarchical structures of information are ubiquitous: family trees, directory structures, organization structures, catalogues, computer programs, etcetera. Small hierarchical structures are very effective to locate information, but the content and organization of large structures is much harder

2 Background

Many methods exist to display and browse through hierarchical information structures, or, for short, trees. File browsers are the best known example. Usually a listing of the files and directories is used, where the levels in the hierarchy are shown by means of indentation. The number of files and directories that can be shown simultaneously is limited, which is no problem if one knows what to search for. However, if we want to get an overview, or want to answer a more global question, such as: "Why is my disk full?", scrolling, and opening and closing of subdirectories have to be used intensively. During this process it is hard to form a mental image of the overall structure [3].



Node and link diagram

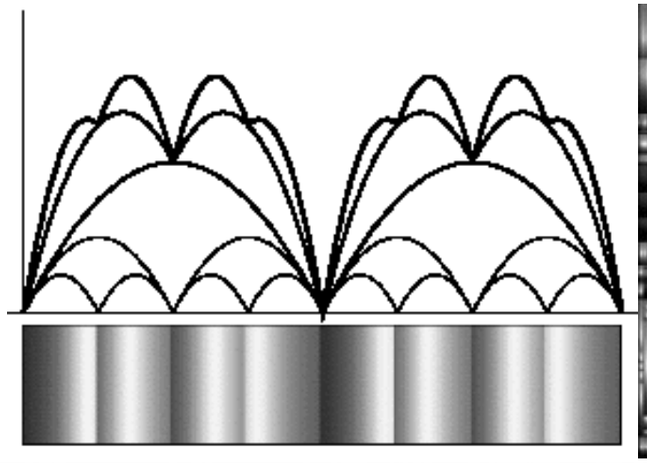


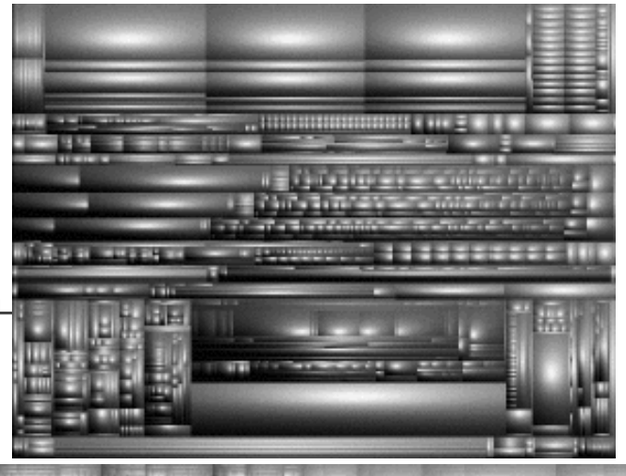
Treemap

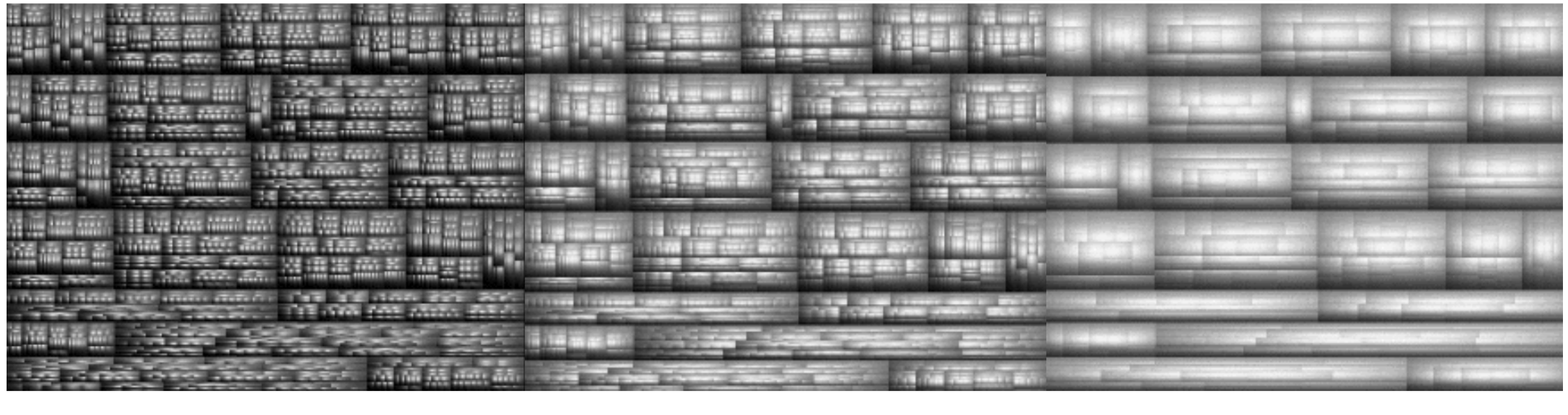
Cushion Treemaps

- Visual encoding: show nesting/topological structure more clearly with shading cues

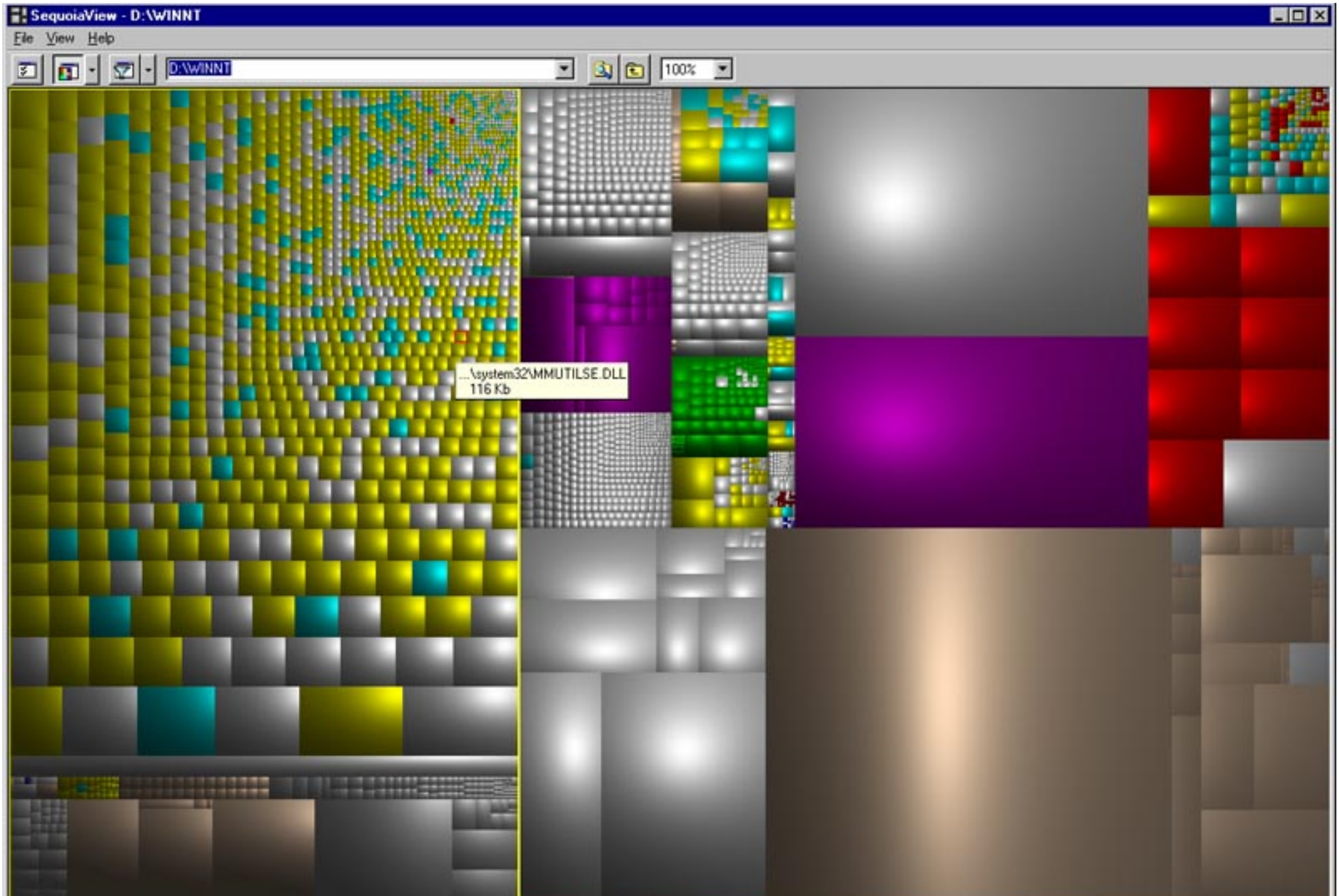
- Interaction: scale parameter controls global vs. local







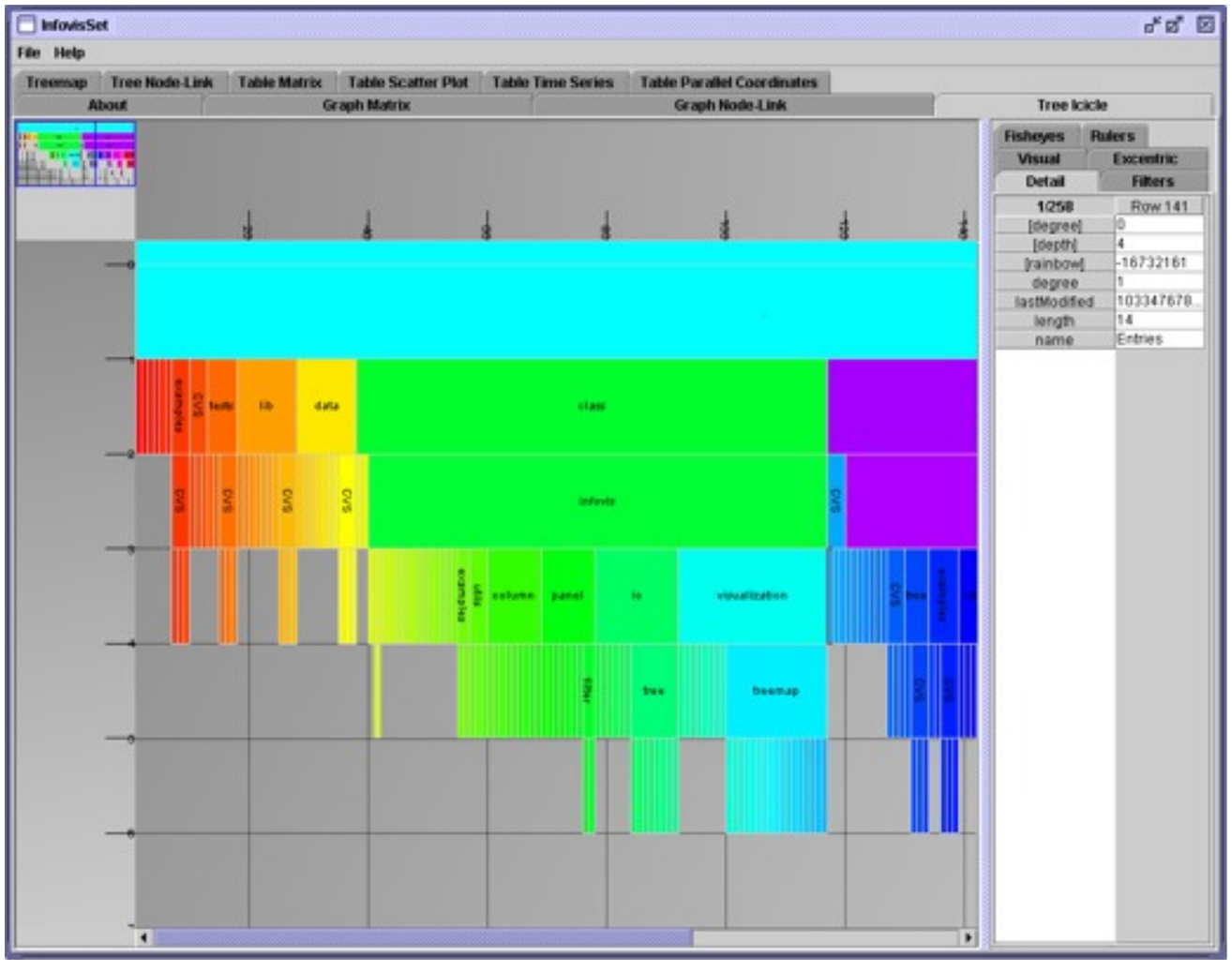
Critique


SequoiaView - D:\WINNT
File View Help
D:\WINNT
100%
...\system32\MMUTILSE.DLL
116 Kb

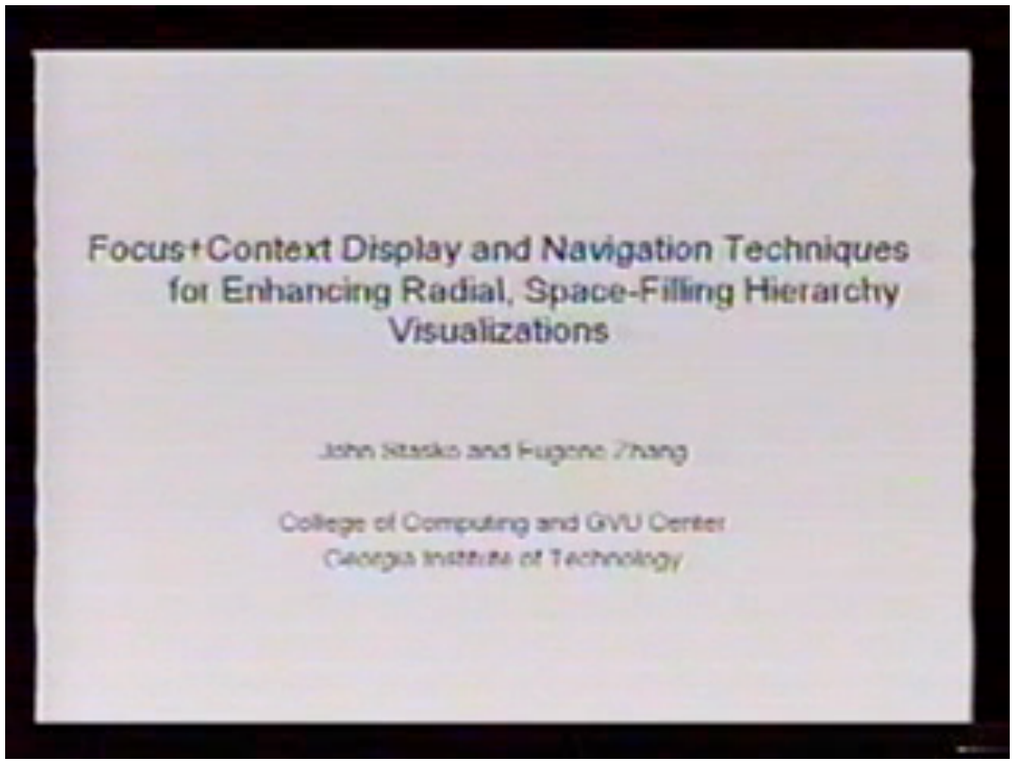
Layered Diagrams

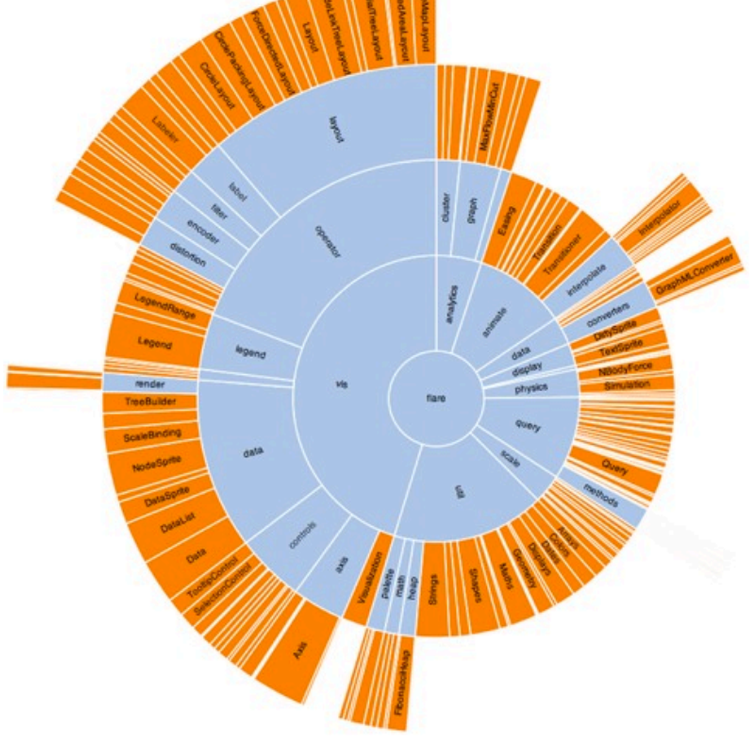
- Similar to node-link layouts without edges
    - Structured encoded using: layering, adjacency, and alignment
- Recursive subdivision of space
- Apply same set of approaches as in node-link layout
- Higher-level nodes get a larger layer area, whether that is horizontal or angular extent.
- Child levels are layered, constrained to parent’s extent

Icicle and Sunburst Trees



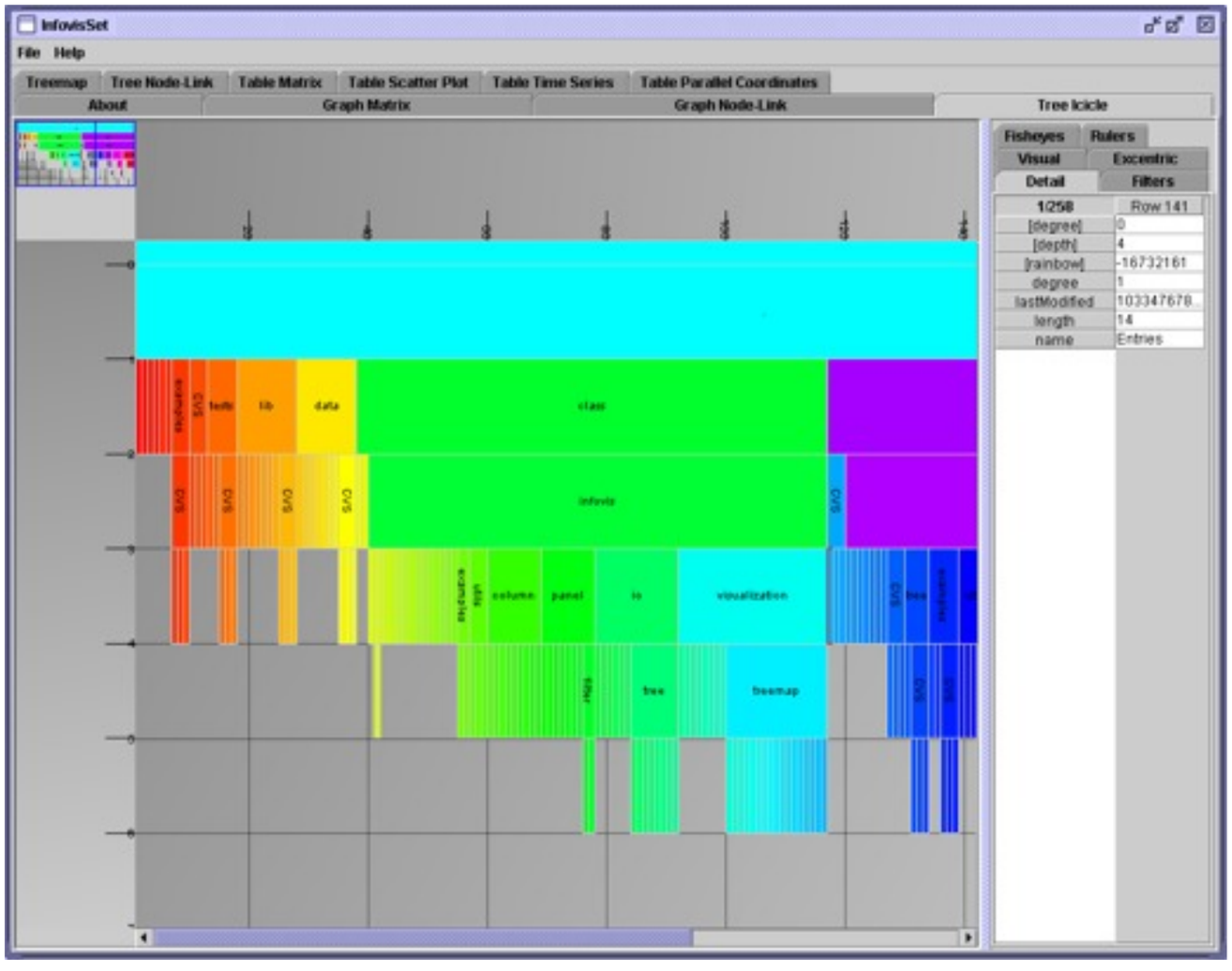
http://ivtk.sourceforge.net



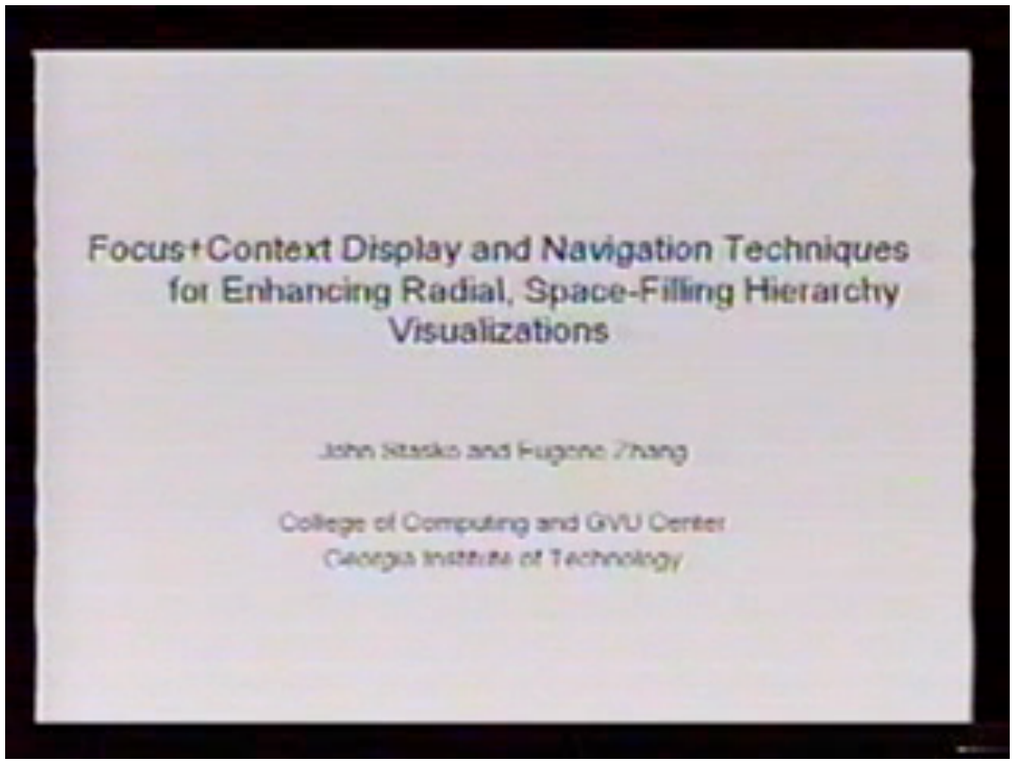


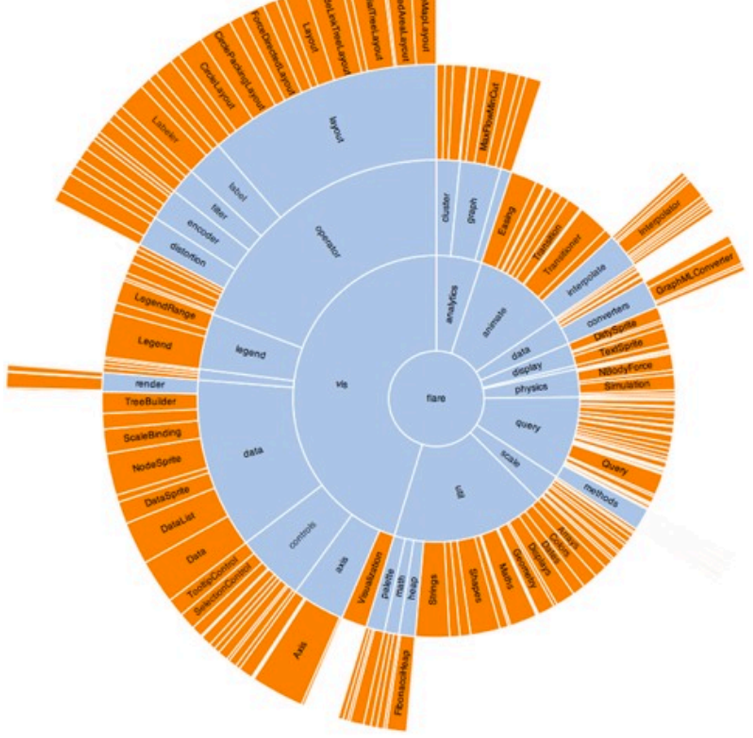
http://hci.stanford.edu/jheer/files/zoo/

Icicle and Sunburst Trees



http://ivtk.sourceforge.net





http://hci.stanford.edu/jheer/files/zoo/

Tree Visualization

http://www.randelshofer.ch/treeviz/index.html

Google

Apple Yahoo! Google Maps YouTube Wikipedia News (1,897) Popular Google Scholar

Home Fun **Science** Software www.randelshofer.ch

› Overview › **Tree Visualization**

Visualization of large tree structures

This project is about fast interactive visualization of large data structures organized in a tree.

This is an experimental software. If you have a feature request, or if you want to honour my work, send me an Amazon gift card or a donation.

Project Goals

- Visualize the data structure in a way which allows to get an overview of the data structure within a short time.
- Provide guidance which allows to quickly drill down into points of interest in the data structure.
- Render the data structure fast enough so that real-time navigation is possible.

Implemented Diagrams

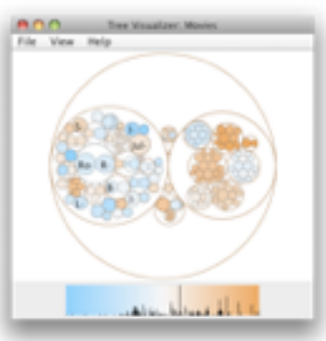
The project currently consists of a file browser demo, which visualizes the file system with the following tree diagrams:

- Hyperbolic Tree
- Circular Treemap
- Rectangular Treemap
- Sunburst Tree
- Icicle Tree
- Sunray Tree
- Iceray Tree

Screenshots

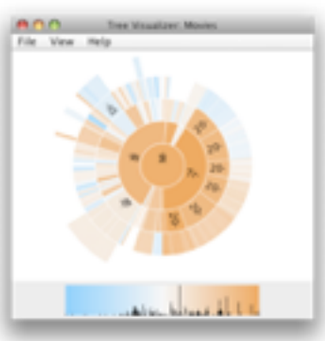
The following pictures show all the same data set:

Circular Treemap:



Rectangular Treemap:

Sunburst Tree:



Icicle Tree:

Try it out

Launch

Installs and launches Treeviz on all platforms. Needs a 64-bit JVM and at least 4 GB of RAM.

Launch with 32-bit JVM.

Downloads

1.0.4 (590 K)

Source code included.

The executable .jar file is located in the dist directory.

Examples

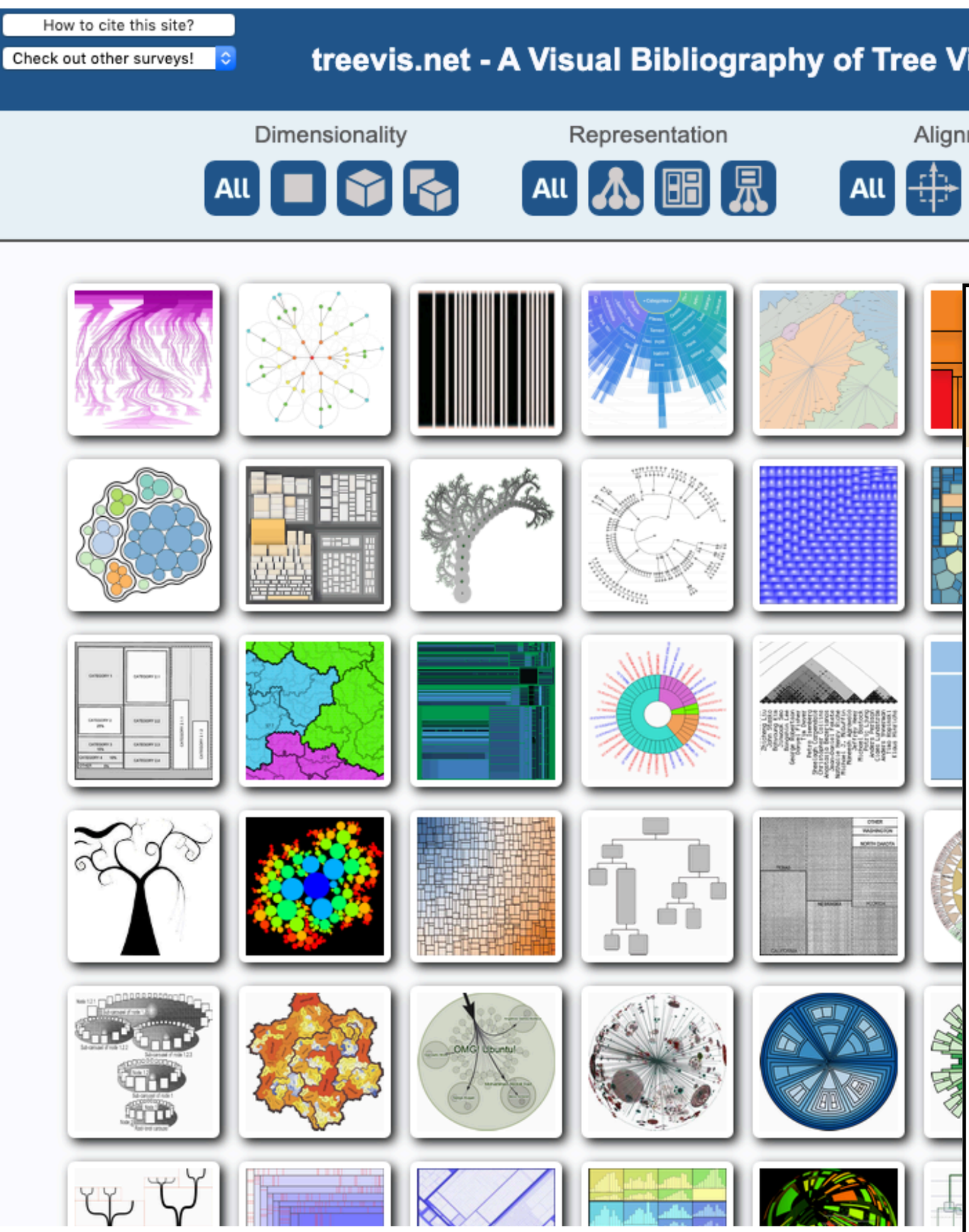
The demo application can visualize a directory structure or an XML file as illustrated in the following example files:

Treeviz Example.xml

business.xml

Please don't rely on the functionaity of the demo application. Since this is an ongoing research project, the functionality and the format supported by the demo application is

http://www.randelshofer.ch/treeviz/index.html



Graphically Speaking

Editor:
Miguel Encarnação

Treevis.net: A Tree Visualization Reference

Hans-Jörg Schulz
University of Rostock

Many people in the information visualization and graph-drawing communities consider tree visualization (see the sidebar) a solved problem. Although Kim Marriott and Peter Stuckey have shown that finding an optimal tree layout can be an NP-complete problem,[1] reasonably good tree layouts can nevertheless be computed efficiently in terms of runtime and screen space utilization. In the course of the search for heuristics to generate ever-tidier tree layouts, the comparatively simple problem of transforming parent-child relationships into graphical representations has been solved many times and is still the subject of information visualization research. Researchers have explored and published almost every way of arranging a tree's nodes in 2D and 3D; encoding them in different shapes or forms; and folding, unfolding, or otherwise interactively manipulating them.

The plethora of tree visualization techniques poses challenges to researchers and developers. Researchers, especially those new to the field, have no way of knowing every tree visualization that has been published, even over just the last two decades. So, they often reinvent existing techniques. Without pointing fingers—my colleagues and I have done our fair share of unwittingly reinventing visualizations—I've noticed that the published tree visualizations include a number of such reinventions. This is hardly surprising because it's almost impossible for peer reviewers as well to have a complete overview of prior research.

The same holds true for developers who implement tree visualizations for their customers, but with potentially direr consequences. Developing something that already exists could lead to ugly intellectual-property issues. And even though it seems like a good starting point to assume that something similar to your own idea has already been done, finding that similar technique can be extremely difficult.

However, opportunities also exist. The long history and remarkable coverage of the design space offer the opportunity to step back, take a look at the bigger picture, and learn from it. For example, we can identify recurring design patterns. Moreover, we can trace back the evolution of our modern visualization techniques to the visual archetypes that might have inspired them.

To address the challenges and exploit the opportunities, we must make a laborious but important first step: we must collect existing tree visualization techniques and form a reference for them that's as complete as possible. This is where the treevis.net project comes into the picture.

Hunting and Gathering Tree Visualizations

In early 2010, I set out to ramble through the available tree visualization literature and websites. Most tree visualizations could readily be excerpted from conference proceedings and journals. From these, I slowly built a "convex hull" by seeking those papers cited by the ones I found and those that cited the found ones. But this covered only the scholarly publications. Much harder to hunt down were the visualizations that appear on Flickr

Tree Visualization

Tree visualization (sometimes called *hierarchy visualization*) is a branch of information visualization dedicated to the graphical representation of connected, acyclic graphs—*trees*. Tree structures are common in many aspects of everyday life, such as ancestry (family trees) or file system organization (directory trees). Most tree visualizations are developed for *rooted trees*, which contain a selected top element, the *root node*; intermediate elements, the *internal nodes*; and bottom elements, the *leaves*. Drawing on the family tree metaphor, nodes standing in direct relation are called the *parent* node (the node closer to the root) and *child* node (the node further from the root).

Lec14 Reading

- Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. Danny Holten, Proc InfoVis 06, published as IEEE TVCG 12(5), p 741-748, 2006.

Reminder
Assignment 03

Assigned: Monday, February 20
Due: Monday, March 13, 4:59:59 pm

Reminder
Project Milestone 02

Assigned: Wednesday, February 22
Due: Wednesday, March 29, 4:59:59 pm