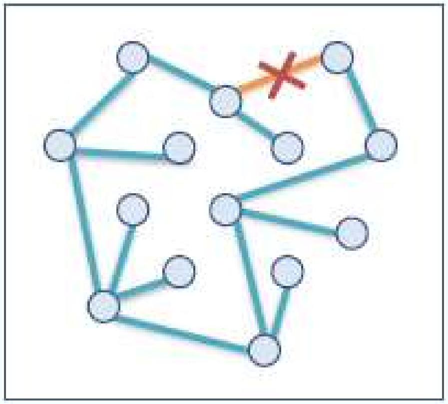
VISUALIZATION

Visualizing Trees/Hierarchies

Tree

- A **tree** is a hierarchical structure with a set of connected nodes.
  - Nodes have a parent and zero or more children
  - Root: Node with no parent (the top-most node in the tree hierarchy)
  - Leaves: Nodes with no children
- A Tree is a graph with no cycles
- Each child can be treated like the root node of its own subtree (recursive structure)
- Many occurrences in the real world:
  - File/directory systems on computers
  - Organization charts
  - Animal kingdom: Phylum,…, genus,…
  - Object-oriented software classes



Tasks

- Describe/understand structure
- Find items
- What are the parent/children/siblings of x?
- Where is this subtree?
- Where are nodes with particular values located?
- What kind of attributes does this subtree have?

Tree Visualization Approaches

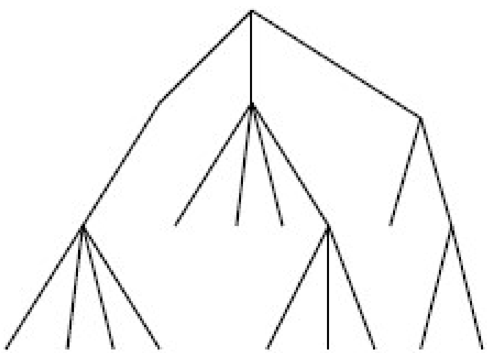
- Node and Link
- Space Filling

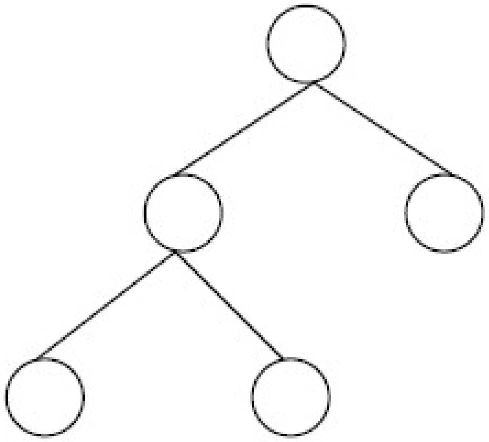
Tree Visualization Approaches

- Node and Link
- Space Filling

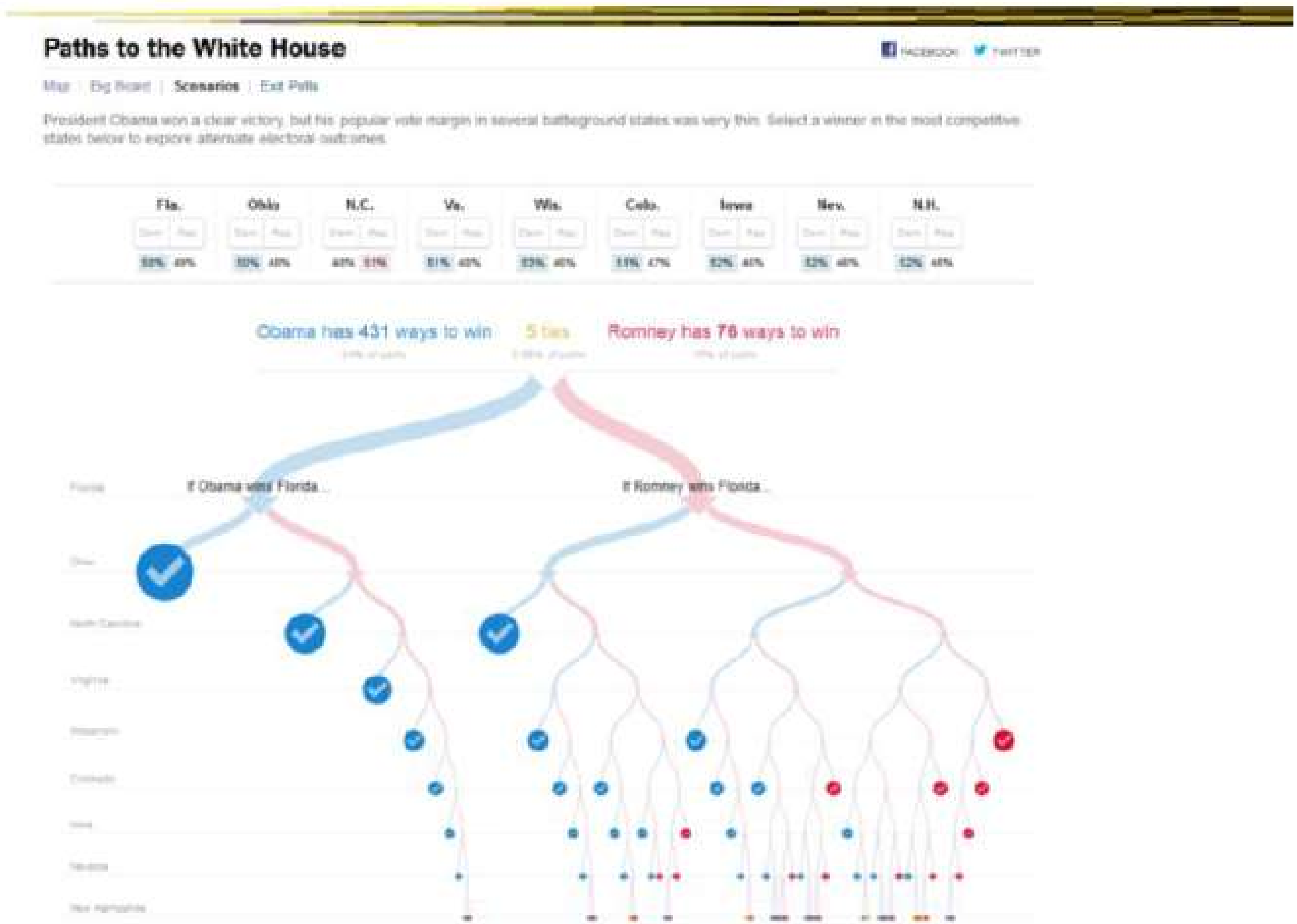
Node link diagram

- Root at top, leaves at bottom is very common



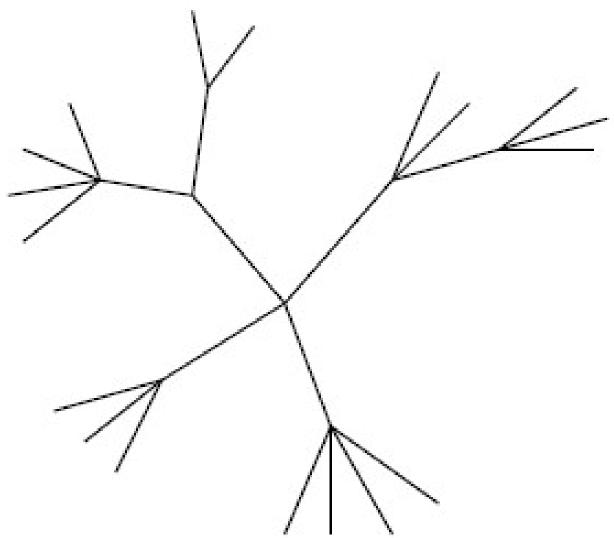


Election ‘12

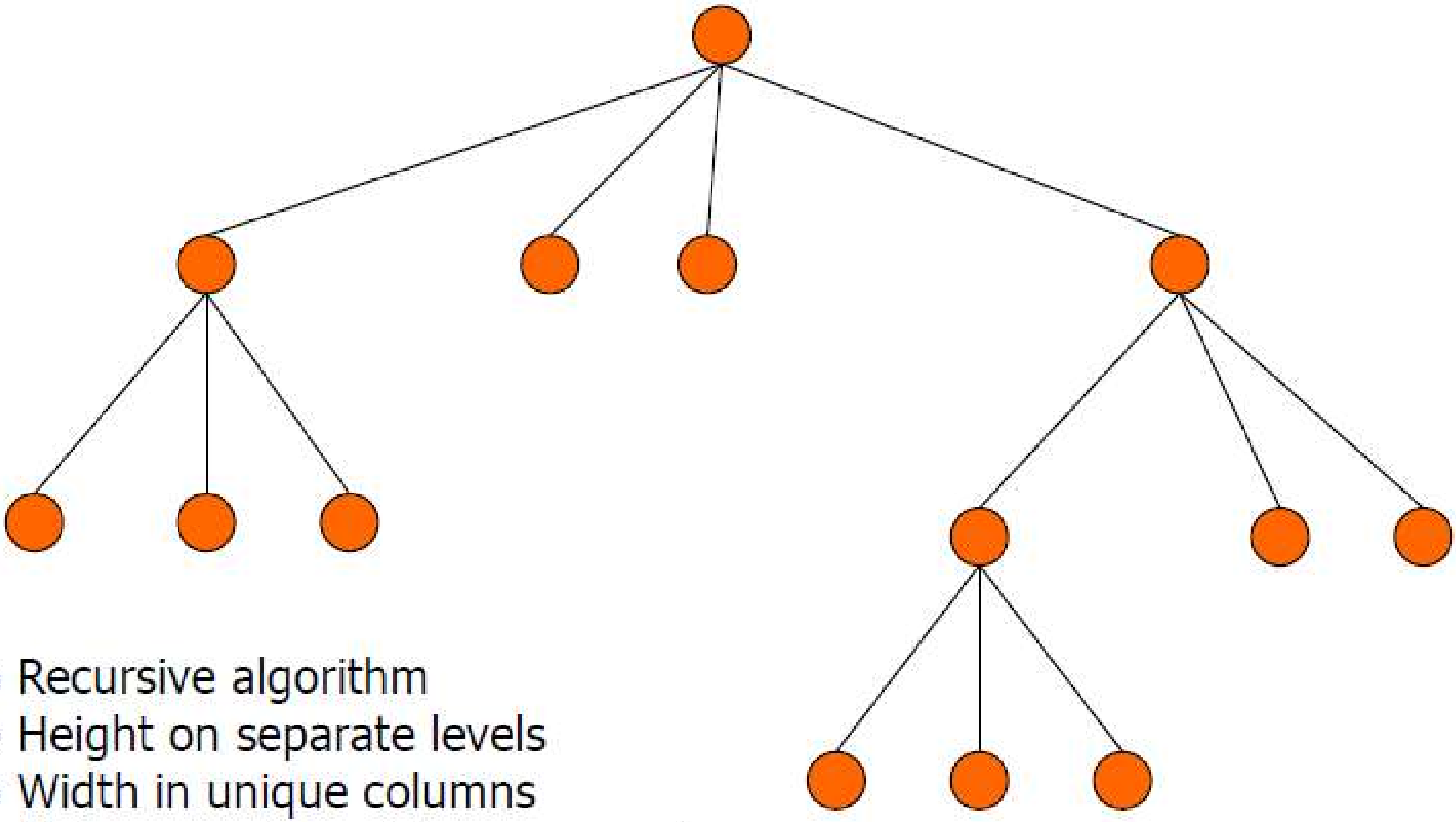

Paths to the White House
Scenarios
Fla.
Ohio
N.C.
Va.
Wis.
Colo.
Iowa
Nev.
N.H.
Obama has 431 ways to win
5 ties
Romney has 76 ways to win
If Obama wins Florida...
If Romney wins Florida...

Radial Layout

- Root can be at center with levels growing outward too



Basic Algorithm



- Recursive algorithm
- Height on separate levels
- Width in unique columns
- Make room for subtrees upwards

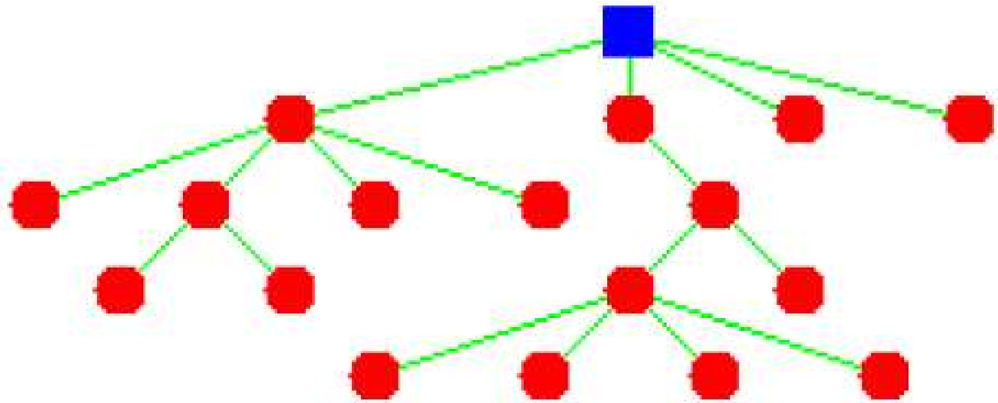
- Repeatedly divide space for subtrees by leaf count
- Breadth of tree along one dimension
- Depth along the other dimension

Potential Problems

- For top-down, width of fan-out uses up horizontal real estate very quickly
  - At level n, there are 2n nodes
- Tree might grow a lot along one particular branch
  - Hard to draw it well in view without knowing how it will branch

Reingold-Tilford Algorithm

Compact layout
Uses symmetry
Depth on levels



- ❖ Linear algorithm
- ❖ Starts with bottom-up pass of the tree
  - ▪ Y-coord by depth, arbitrary starting X-coord
  - ▪ Merge left and right subtrees
  - ▪ Shift right as close as possible to left
    - • Computed efficiently by maintaining subtree contours
  - ▪ "Shifts" in position saved for each node as visited
  - ▪ Parent nodes are centered above their children
- ❖ Top-down pass for assignment of final positions
  - • Sum of initial layout and aggregated shifts

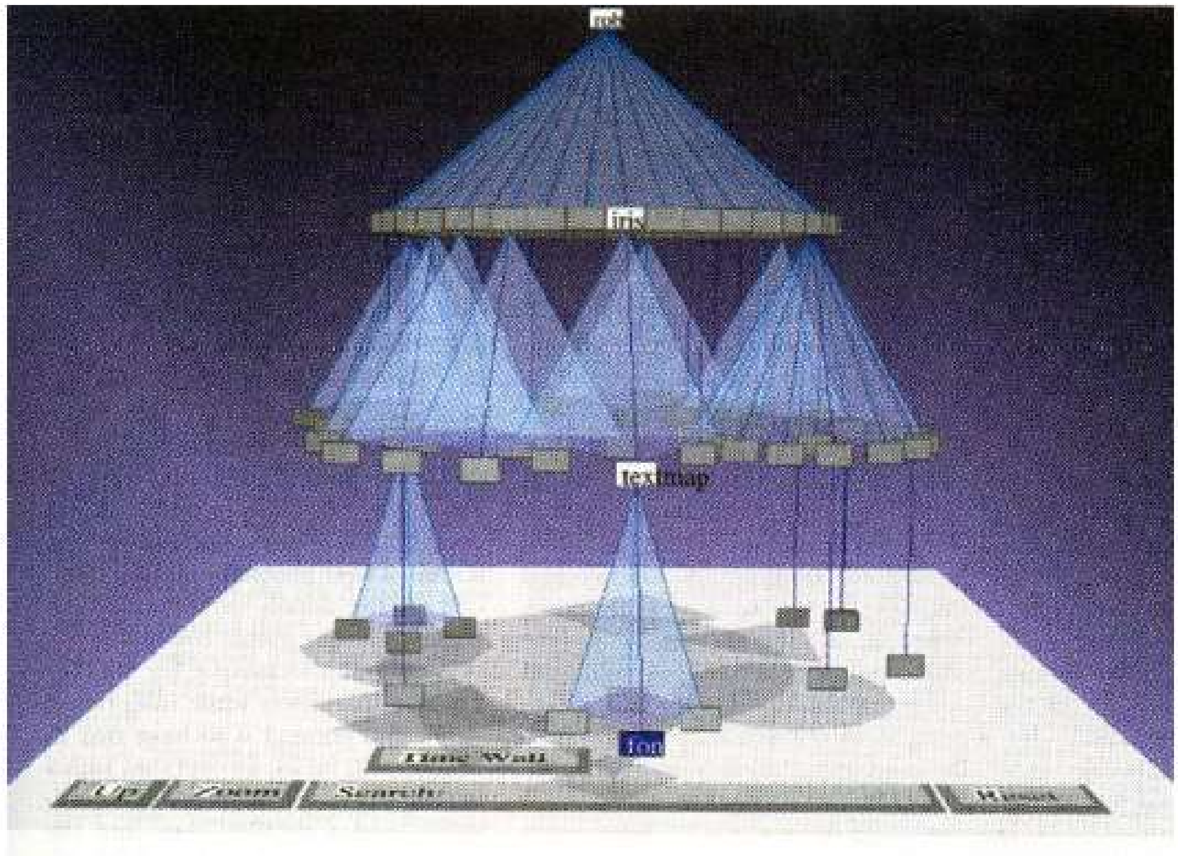
3D Techniques

- Add a third dimension into which layout can go
- Benefits
  - "Gaining more space"
  - Human familiarity with 3D

Cone Tree

Developed at Xerox PARC

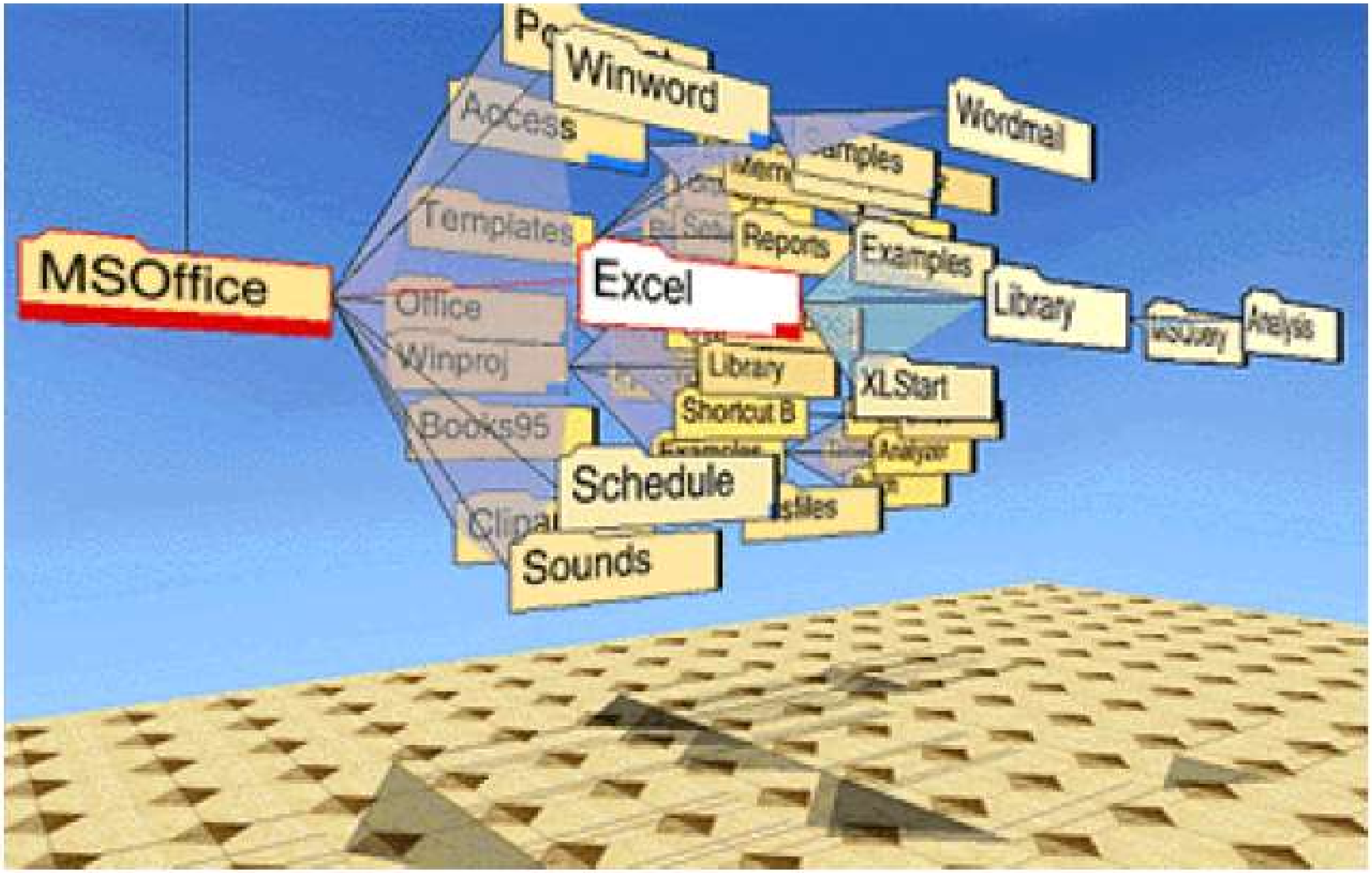
3D views of hierarchies such as file systems



Robertson, Mackinlay, Card

Children of a node are laid out in a cylinder “below” the parent
Siblings live in one of the 2D planes

Alternate View

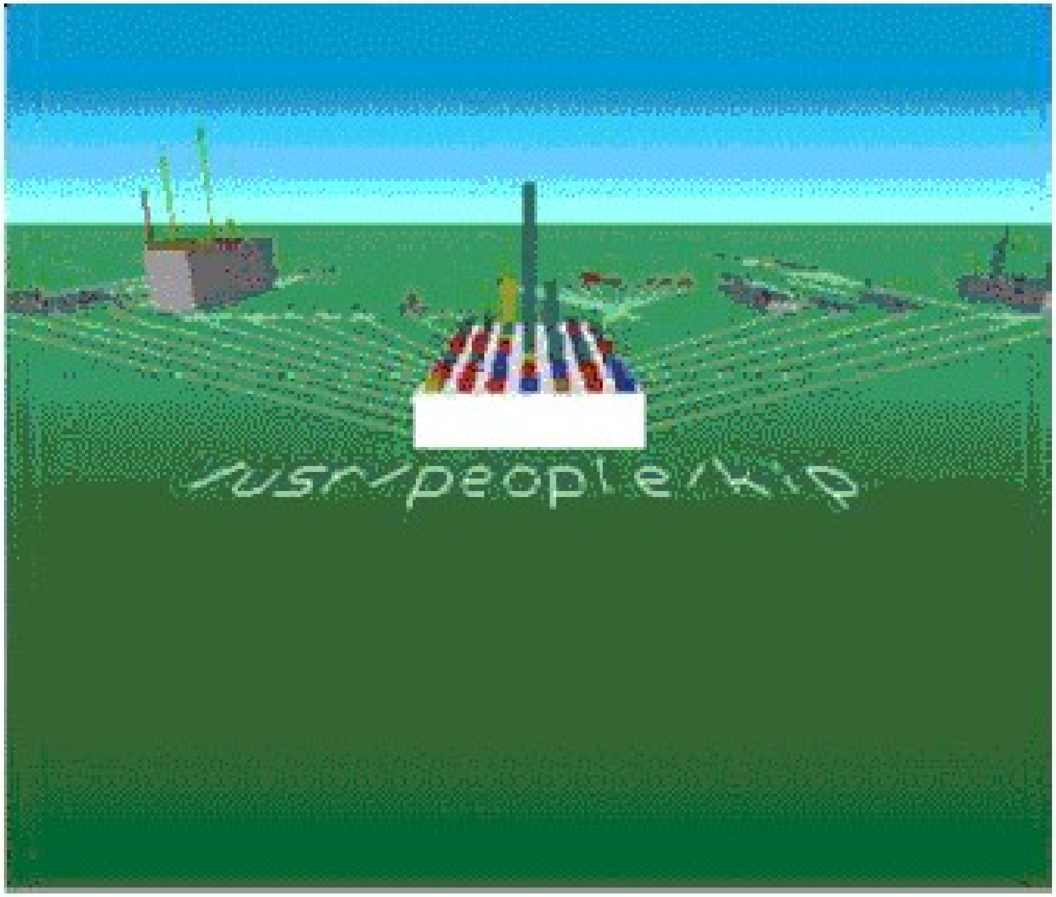


Cone Tree - Evaluation

- Pros
  - More effective area to lay out tree
  - Use of smooth animation to help person track updates
  - Aesthetically pleasing
- Cons
  - As in all 3D, occlusion obscures some nodes
  - Non-trivial to implement and requires some graphics horsepower

Fly-Through of 2D Representation

- SGI File System Navigator
- Size represents file size



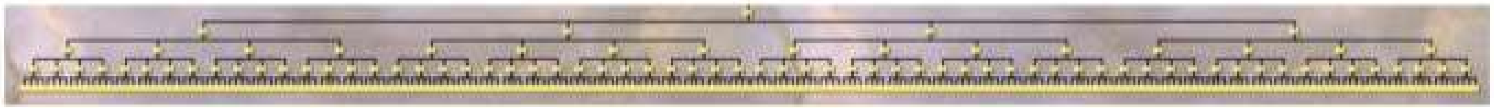
Visualizing Large Hierarchies

- Challenge: hundreds or thousands of nodes
- Approaches?
  - Interaction
  - Only show some items or attributes
  - Clustering & aggregation
  - Smart layout

- Focus+Context techniques

Degree of Interest Trees

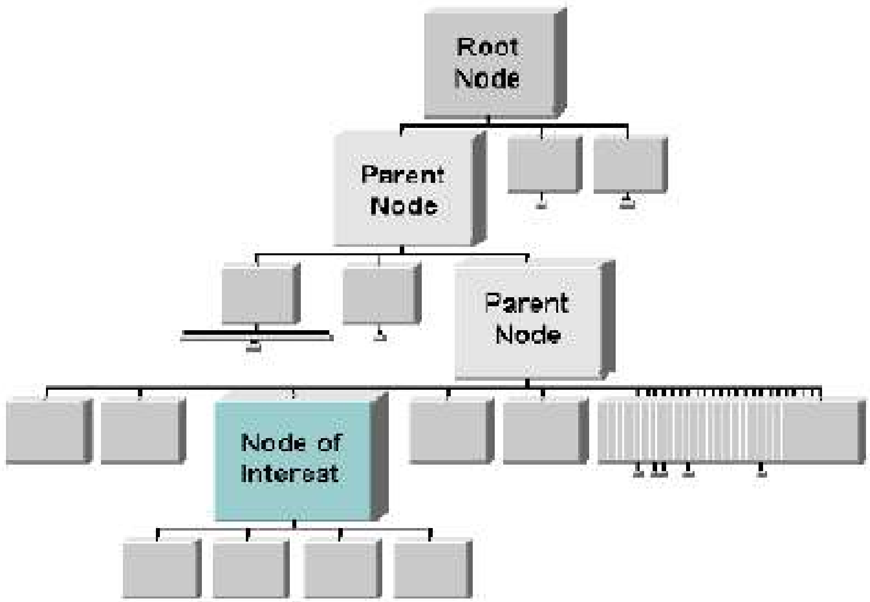
- Problem: Trees quickly degrade into line



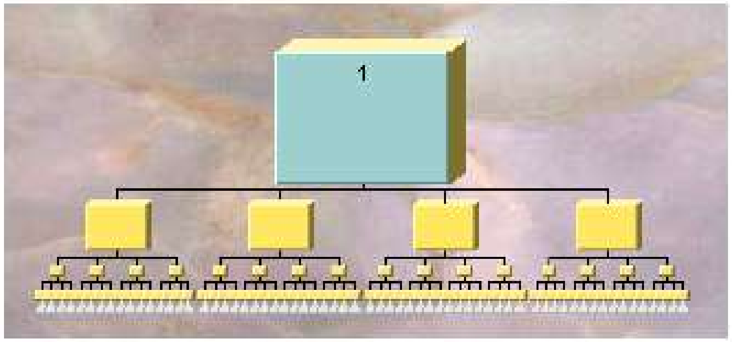
- Approach: Use fisheye-like focus & context ideas to control how a tree is drawn

Approach

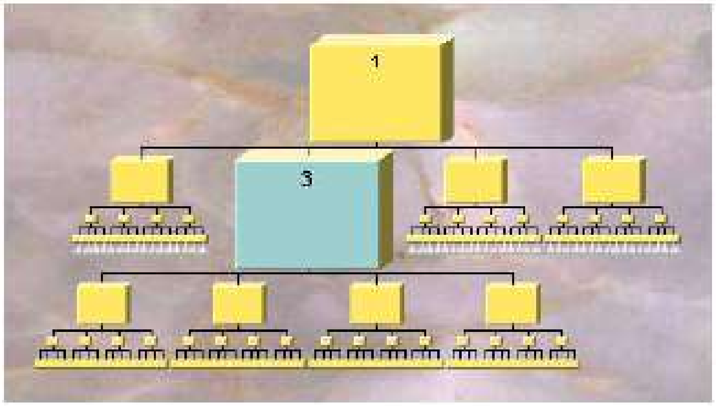
- Combine multiple ideas:
  - Expanded DOI computation
  - Logical filtering to elide nodes
  - Geometric scaling
  - Semantic scaling
  - Clustered representation of large unexpended branches
  - Animated transition



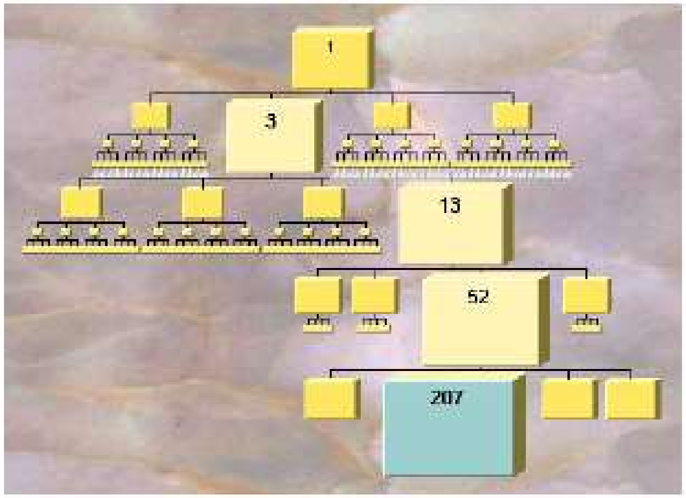
Example



1. Display of a uniform tree of 4 levels



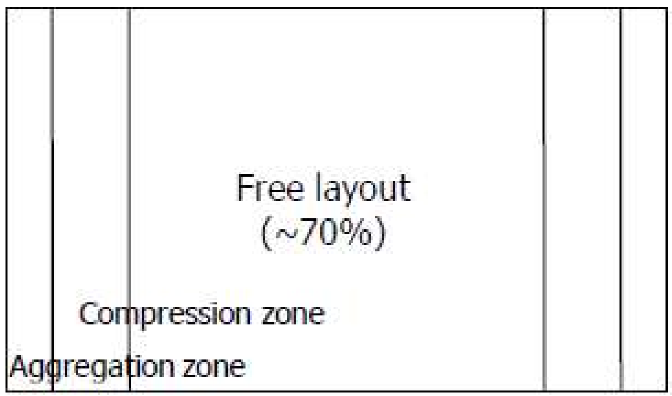
2. Same table with focus on Node 3



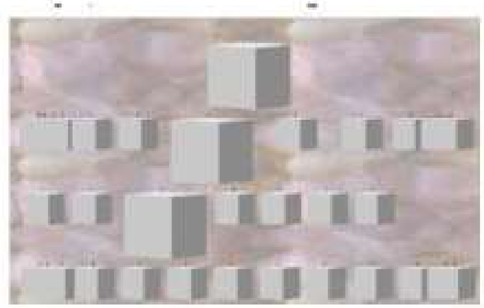
3. Same tree expanded down to a leaf node

Compression

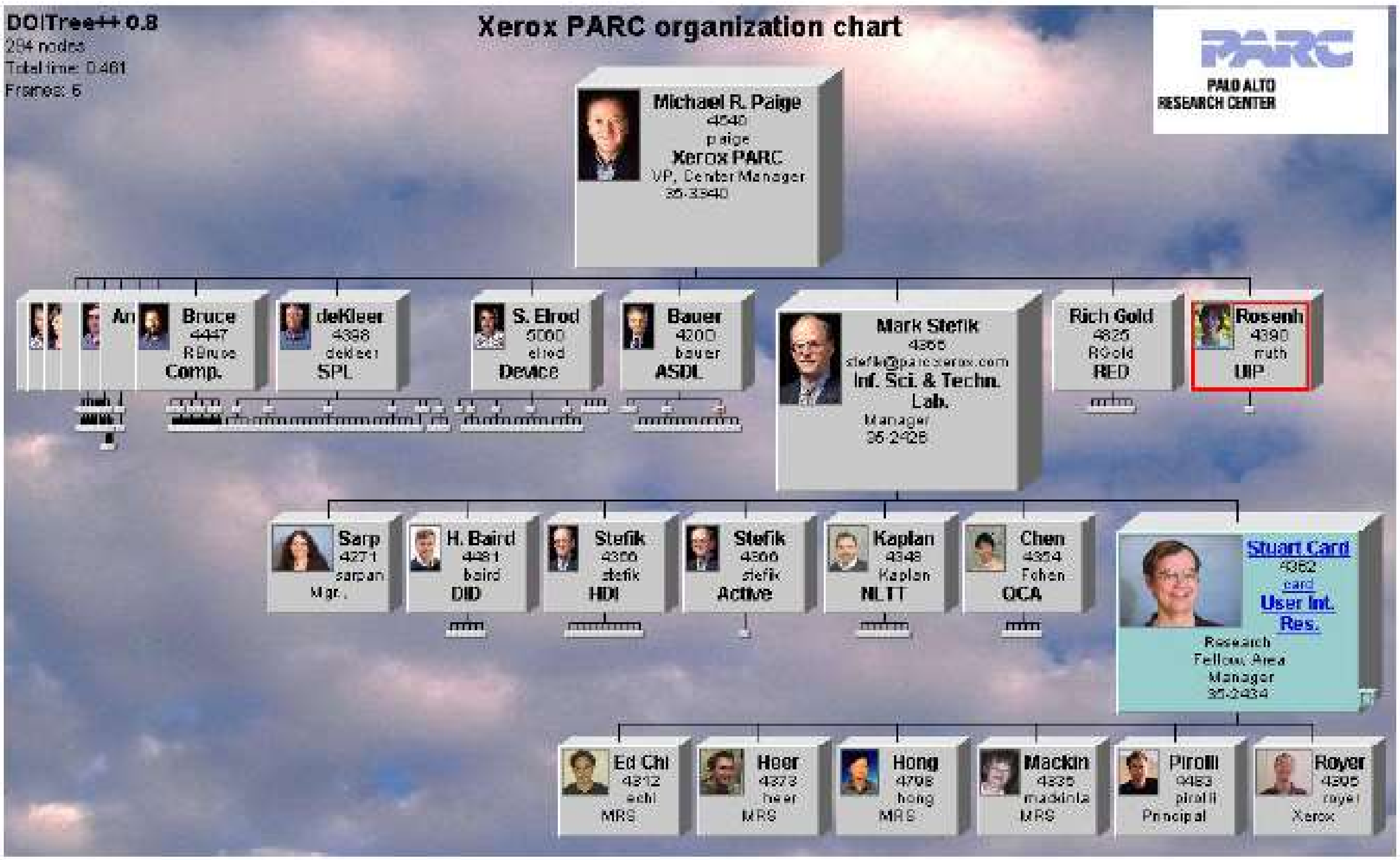
- For nodes: compress to fit (compress in X or in Y)



- Within-node compression
  - Data deletion
  - Word abbreviation
  - Node rotation



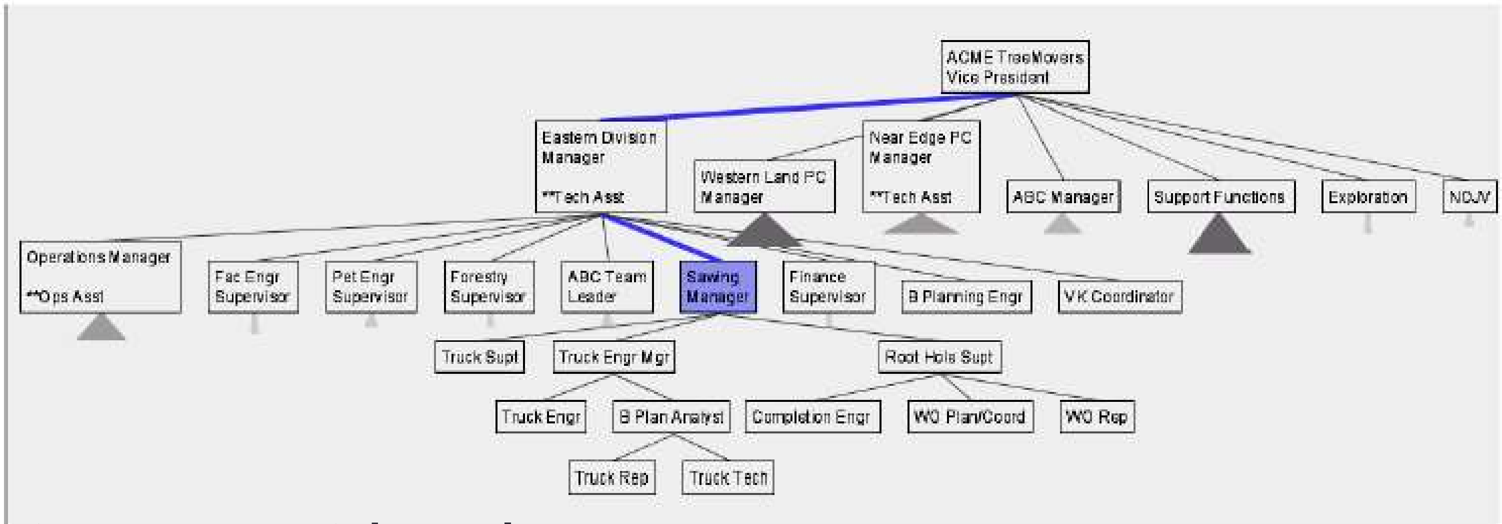
Better View of Org Chart



Organization chart with over 400 nodes accessible over WWW through Web browser

Space Tree

- Uses conventional 2D layout techniques with some clever additions



- Subtrees are triangles
  - Size indicates depth
  - Shading indicates number of nodes inside
- Navigate by clicking on nodes
- Plaisant, C., Grosjean, J., Bederson, B., *InfoVis 2002*

Euclidean Geometry - Postulates

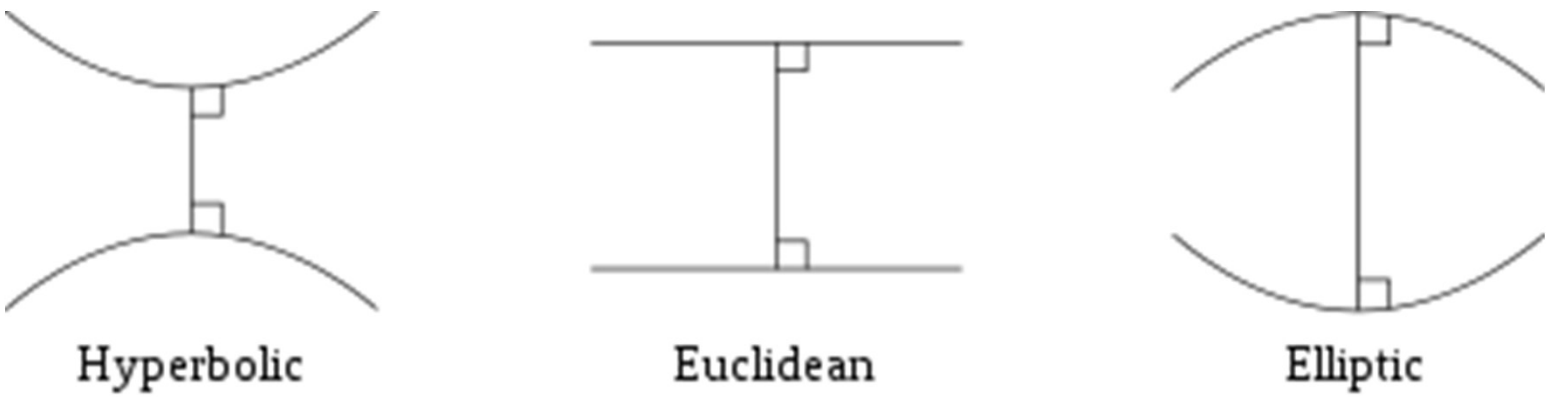
1. A straight line segment may be drawn from any given point to any other.
2. A straight line may be extended to any finite length.
3. A circle may be described with any given point as its center and any distance as its radius.
4. All right angles are congruent.
5. If a straight line intersects two other straight lines, and so makes the two interior angles on one side of it together less than two right angles, then the other straight lines will meet at a point if extended far enough on the side on which the angles are less than two right angles.

Parallel Postulate (Postulate 5)

- Cannot be proven as theorem
- Some equivalent properties
  - There is at most one line that can be drawn parallel to another given one through an external point. (Playfair's axiom)
  - The sum of the angles in every triangle is 180°
  - Two lines that are parallel to the same line are also parallel to each other.

Non-Euclidean Geometry

- Replacing Parallel Postulate gives rise to Hyperbolic and Elliptic Geometry



Behavior of lines with a common perpendicular in each of the three types of geometry

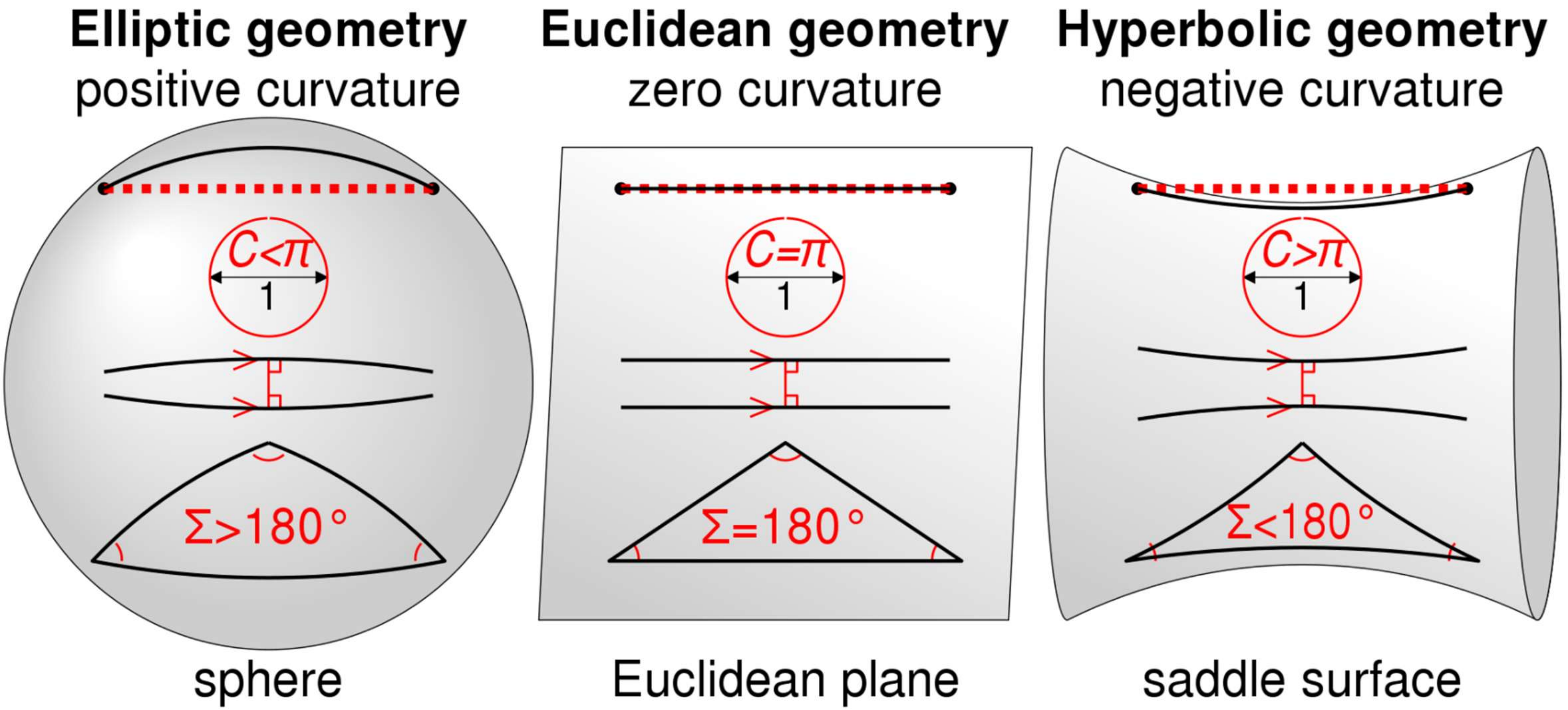
Hyperbolic and Elliptic Geometry

Hyperbolic Geometry

- For any given line *l* and point *P* not on *l*, in the plane containing both line *l* and point *P* there are at least two distinct lines through *P* that do not intersect *l*.

Elliptic Geometry

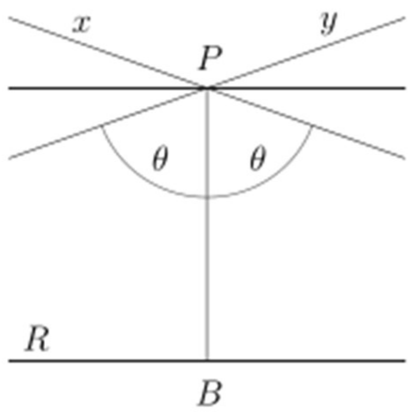
- For any given line *l* and a point *P*, which is not on *l*, all lines through *P* will intersect *l*.



Elliptic geometry
positive curvature
C<π
1
Σ>180°
sphere
Euclidean geometry
zero curvature
C=π
1
Σ=180°
Euclidean plane
Hyperbolic geometry
negative curvature
C>π
1
Σ<180°
saddle surface

Hyperbolic Geometry

- **Lines**
  - For any line *R* and any point *P* here are through *P* an infinite number of coplanar lines that do not intersect *R*.



Lines through a given point *P* and asymptotic to line *R*.

- **Circle**

In hyperbolic geometry, the circumference of a circle of radius *r* is greater than $2\pi r$.

Let $R = \frac{1}{\sqrt{-K}}$, where K is the Gaussian curvature of the plane. In hyperbolic geometry, K is negative, so the square root is of a positive number.

Then the circumference of a circle of radius *r* is equal to:

$$2\pi R \sinh \frac{r}{R}.$$

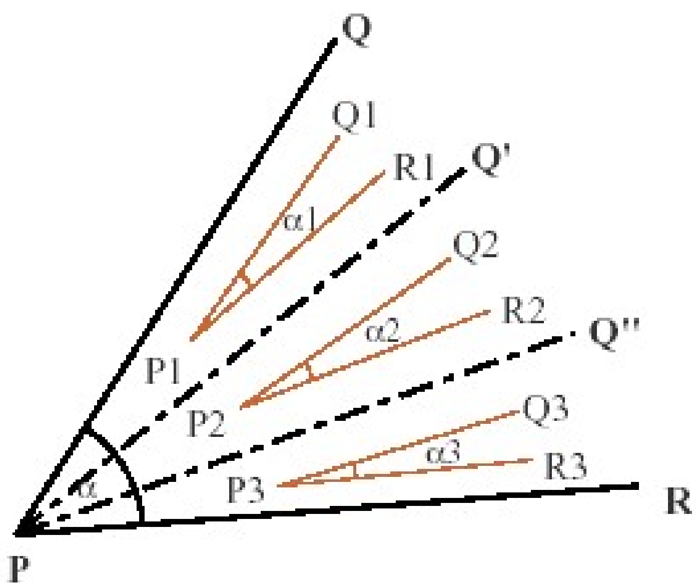
Hyperbolic Browser

- Focus + Context Technique
- Detailed view blended with a global view
- First lay out the hierarchy on the hyperbolic plane
- Then map this plane to a disk
- Start with the tree's root at the center
- Use animation to navigate along this representation of the plane

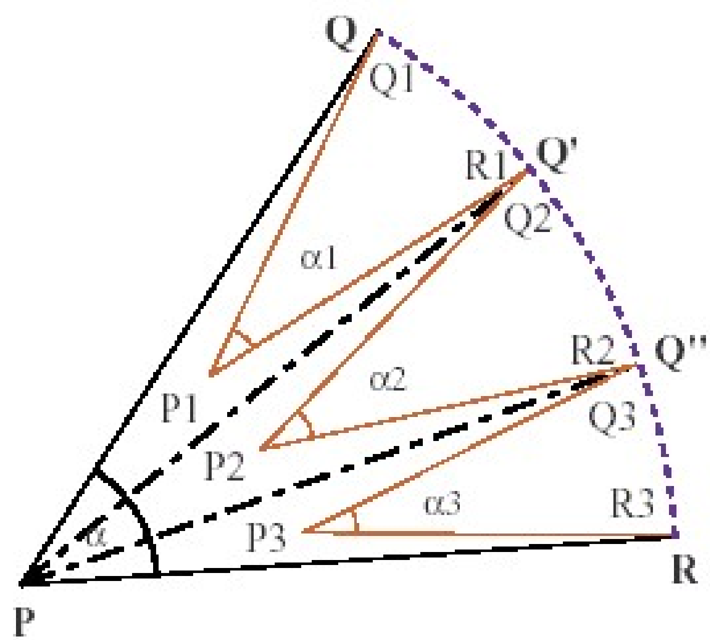
Simple Tree Construction Algorithm

- Node P with wedge QPR
- Subtrees start at P_1, P_2, and P_3
- The circumference of a circle grow exponentially with its radius

Euclidean

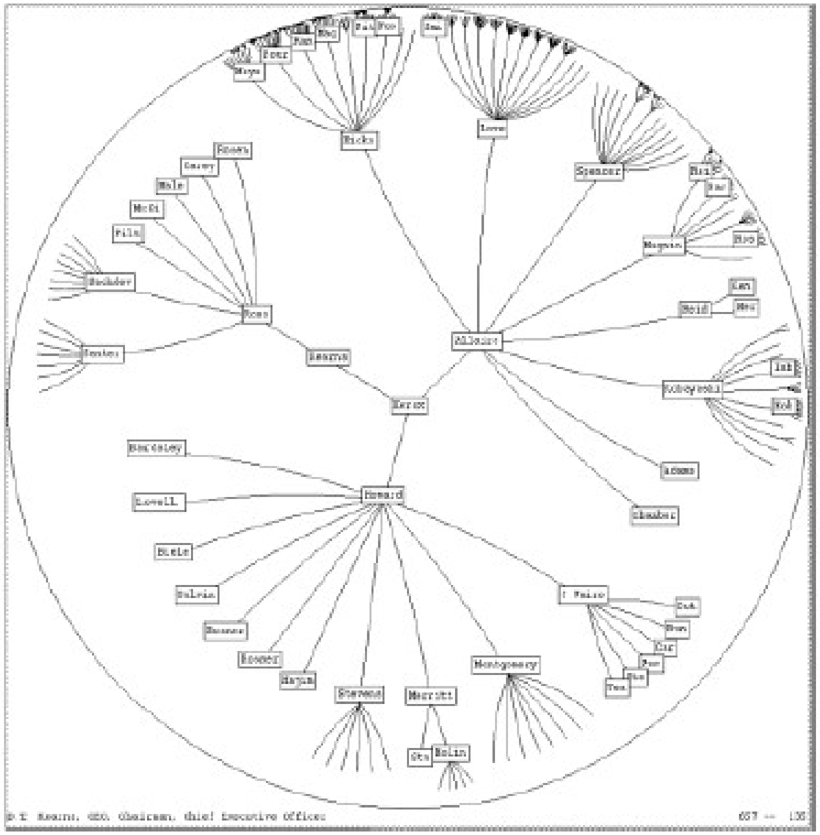


Hyperbolic



2D Hyperbolic Browser

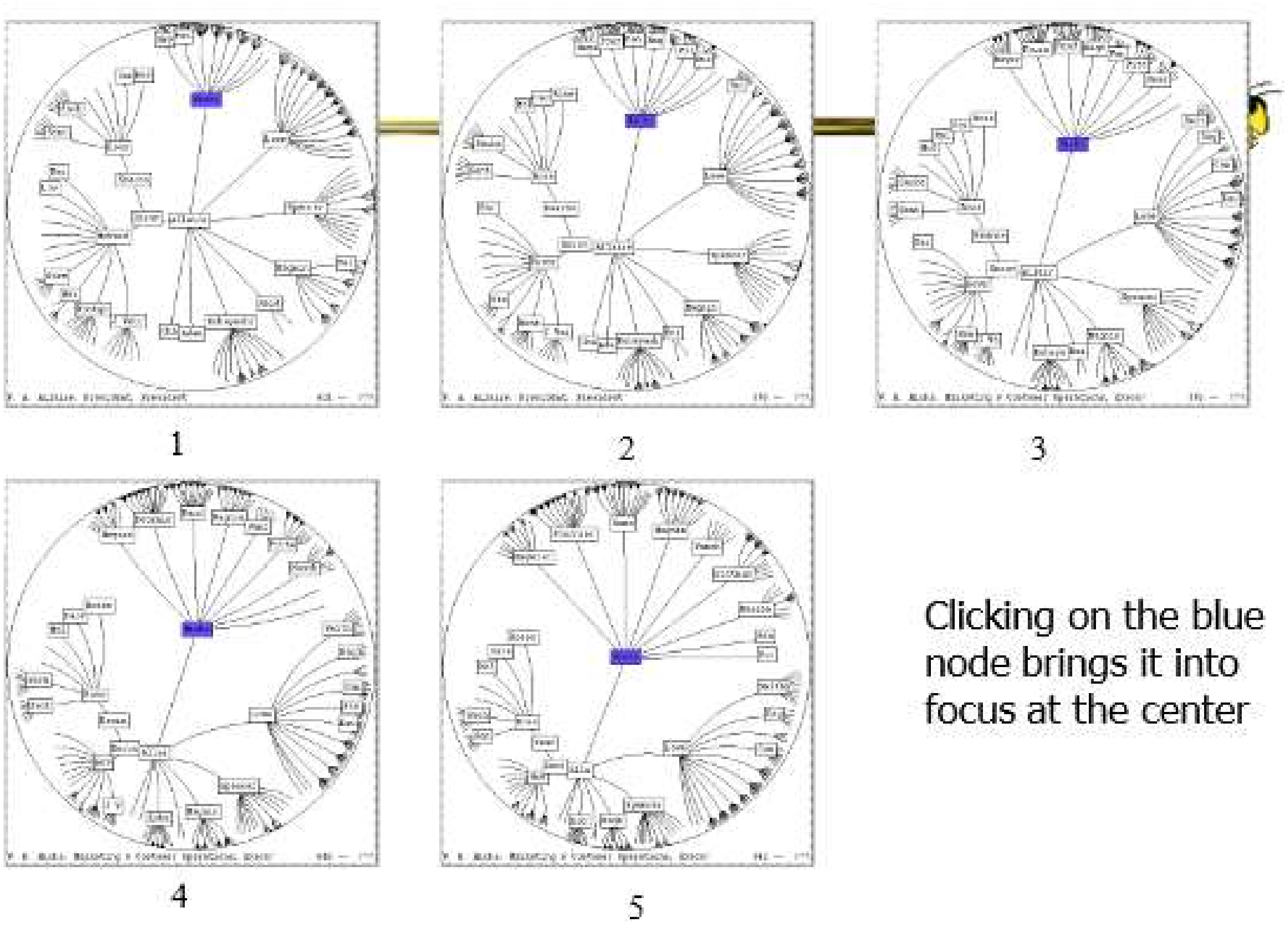
- **Approach:** Lay out the hierarchy on the hyperbolic plane and map this plane onto a display region.
- Comparison
  - A standard 2D browser: 100 nodes (w/3 character text strings)
  - Hyperbolic browser: 1000 nodes, about 50 nearest the focus can show from 3 to dozens of characters



Hyperbolic Browser - Concepts

- Natural magnification(fisheye) in center
- Layout depends only on 2-3 generations from current node
- Smooth animation for change in focus
- Don’t draw objects when far enough from root (simplify rendering)

Hyperbolic Browser - Interactions



Clicking on the blue node brings it into focus at the center

Hyperbolic Browser - Problems

- Orientation
  - Watching the view can be disorienting
  - When a node is moved, its children don't keep their relative orientation to it as in Euclidean plane, they rotate
  - Not as symmetric and regular as Euclidean techniques, two important attributes in aesthetics

Tree Visualization Approaches

- Node and Link
- Space Filling

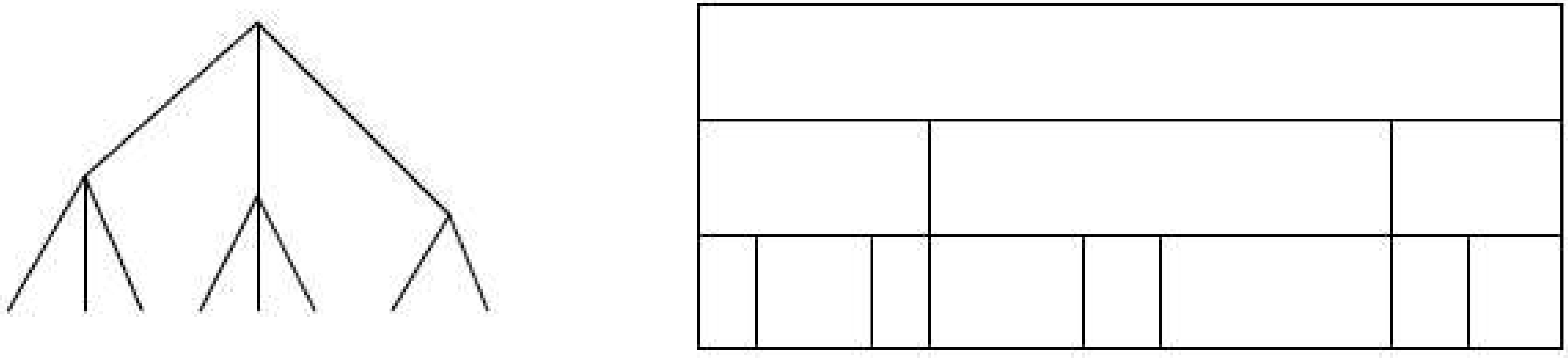
Node-Link Shortcoming

- Difficult to encode more variables of data cases (nodes)
  - Shape
  - Color
  - Size
- …but all quickly clash with basic node-link structure

Space Filling Representation

Each item occupies an area

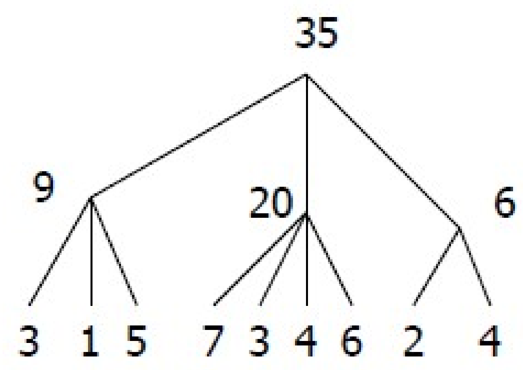
Children are "contained" under parent



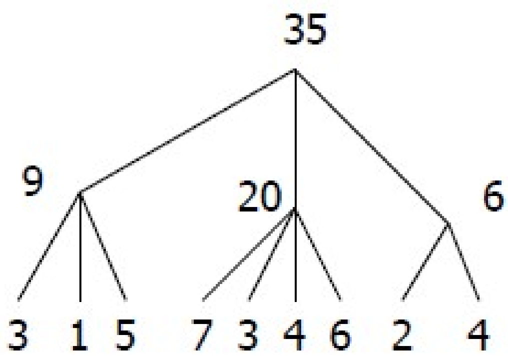
Treemap

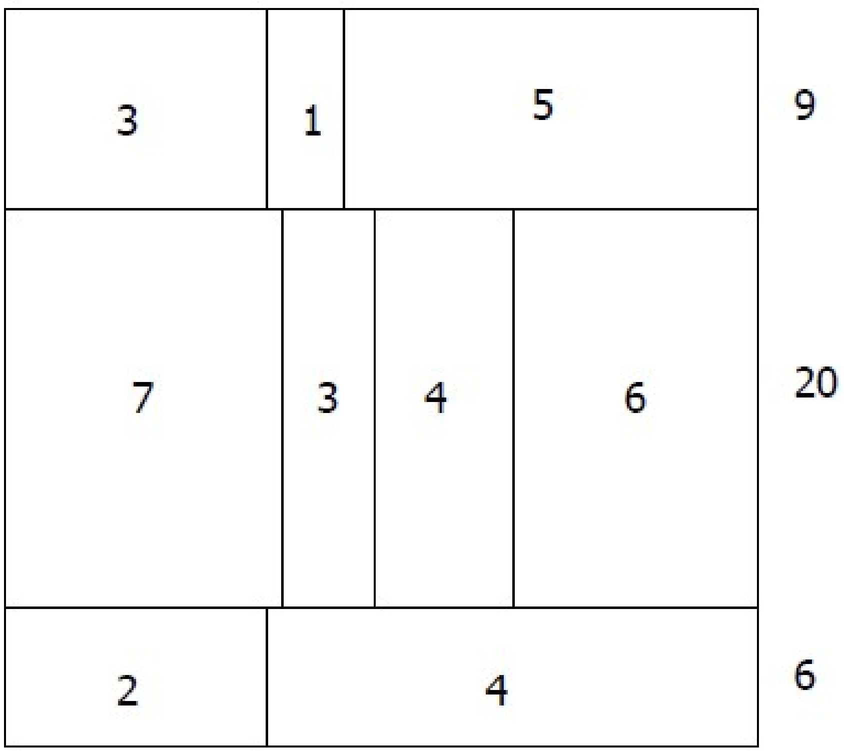
- Space-filling representation developed by Shneiderman and Johnson, Vis ‘91
- Children are drawn inside their parent
- Alternate horizontal and vertical slicing at each successive level
- Use area to encode other variable of data items

Example

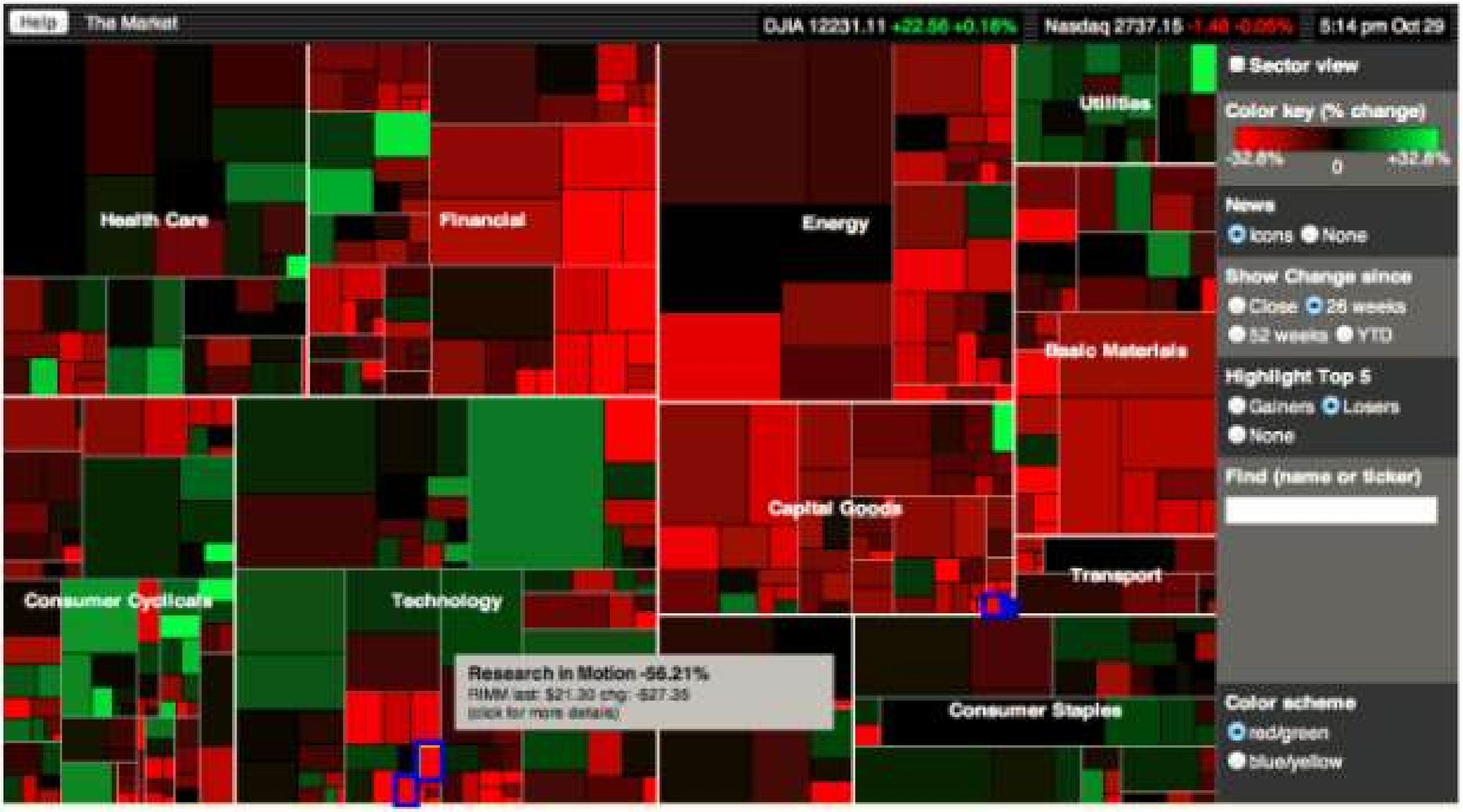


Example



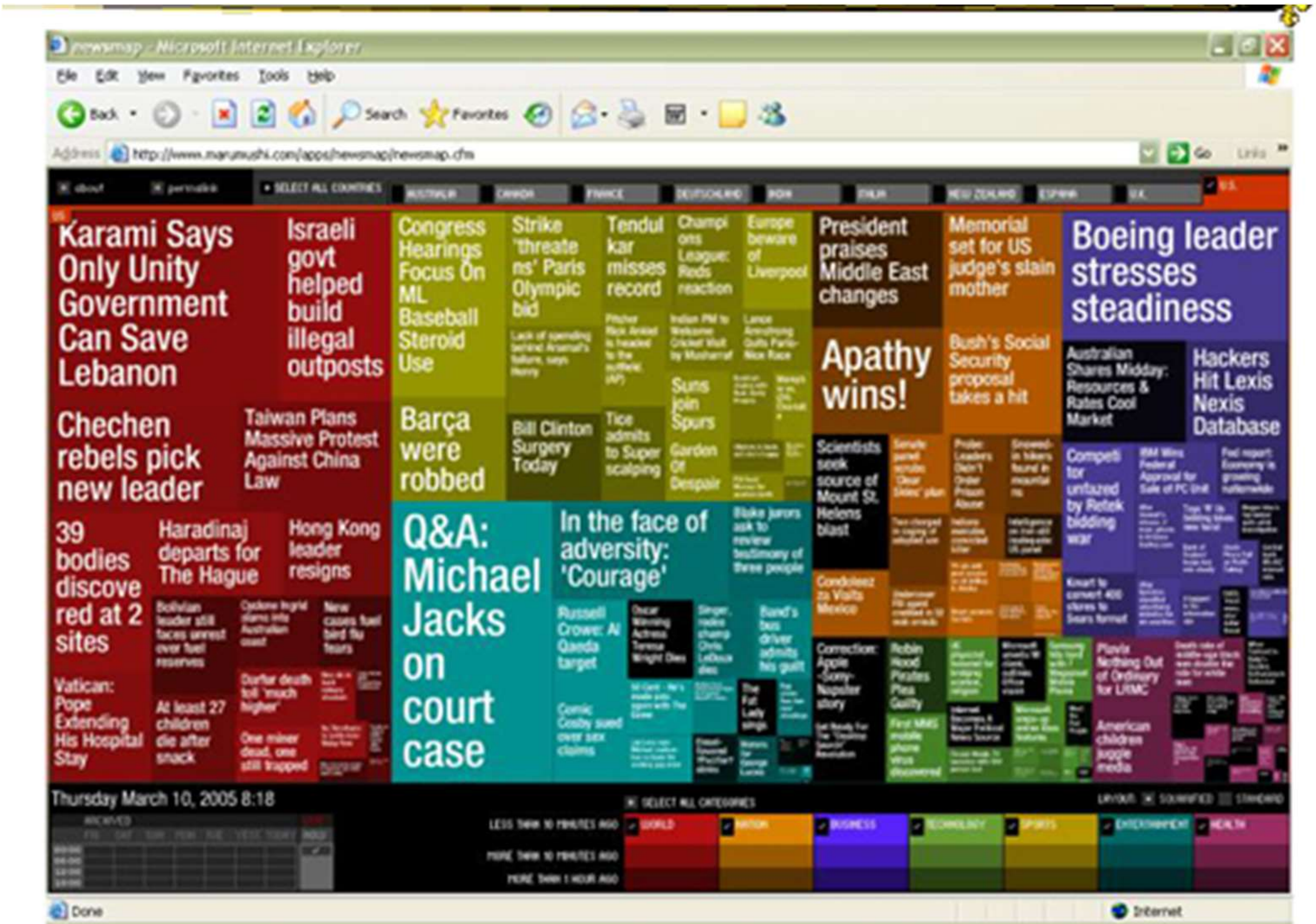


Example - Finance



Map Of The Market (Source: SmartMoney website)

Example - News



Treemap - Algorithm

Draw() {

  Change orientation from parent (horiz/vert)

  Read all nodes at this level

  Make rectangle for each, scaled to size

  Draw rectangles using appropriate size and color

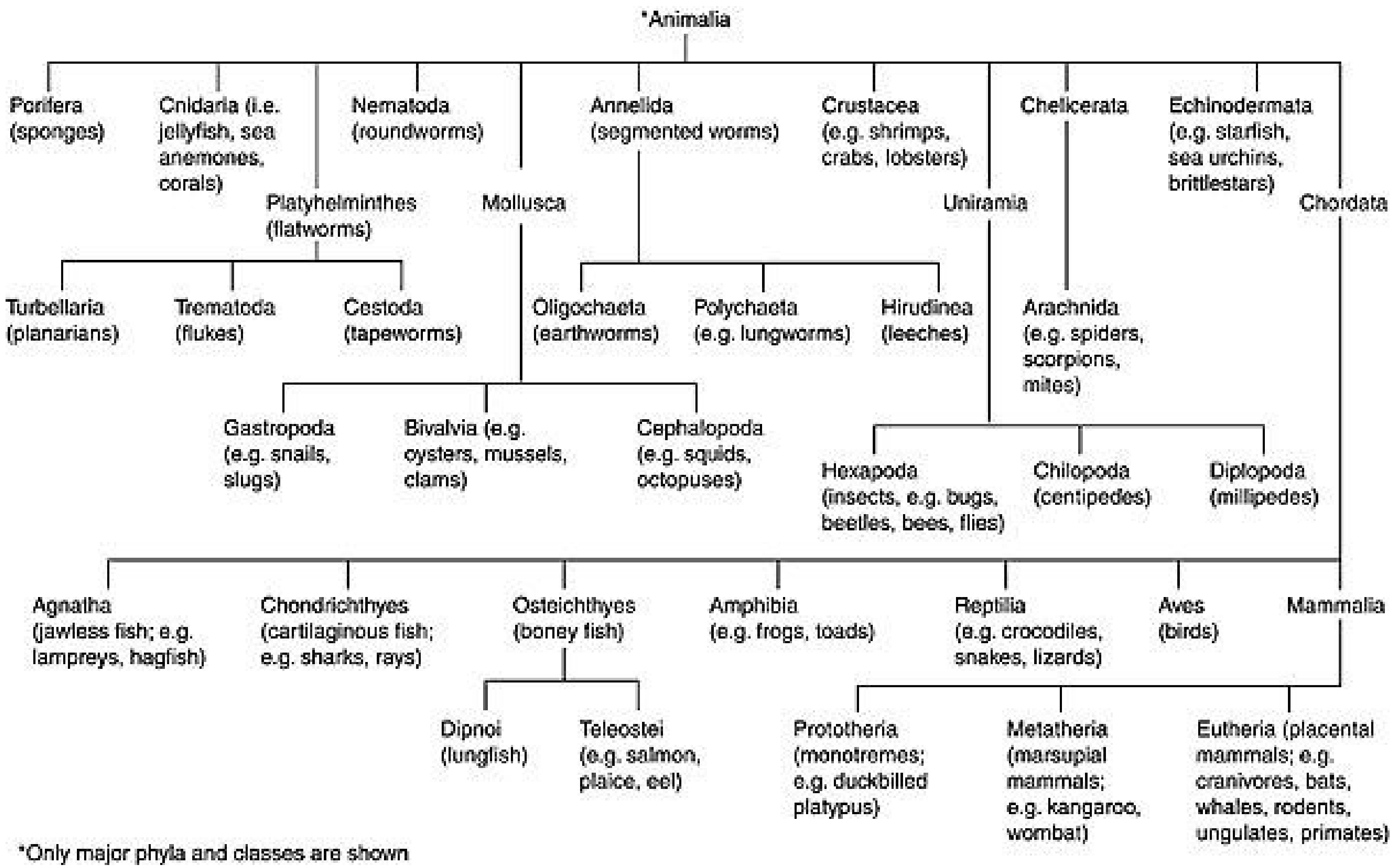
  For each directory

    Make recursive call using its rectangle as focus

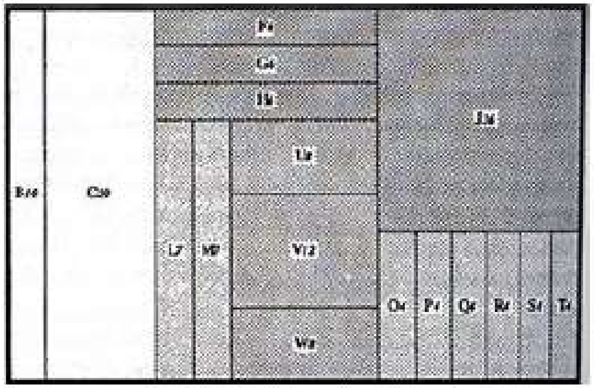
}

Exercise

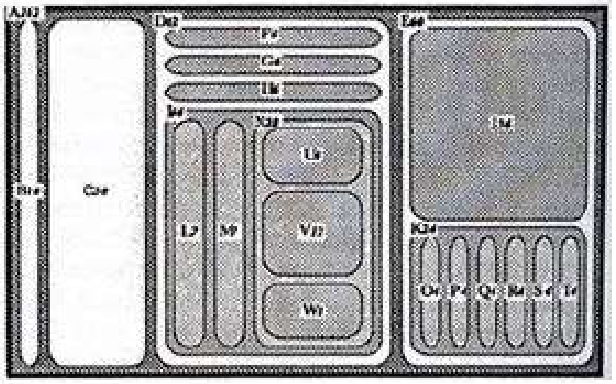
- Draw a treemap view of the following:



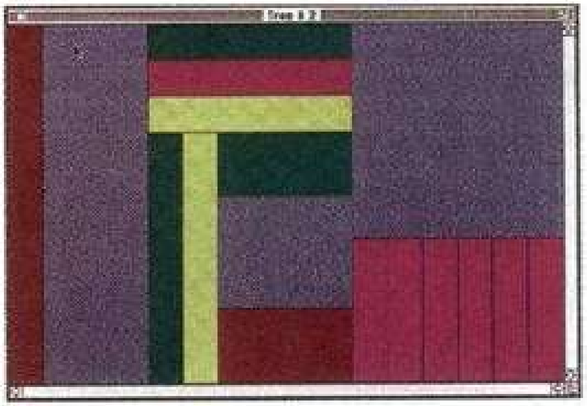
Nested vs Non-nested



Non-nested Tree-Map



Nested Tree-Map



Treemap - Evaluation

- Good representation of two attributes beyond node-link: color and area
  - Not as good at representing structure
- What happens if it’s a perfectly balanced tree of items all the same size?
- Also can get long-thin aspect ratios
- Borders help on smaller trees, but take up too much area on large, deep ones

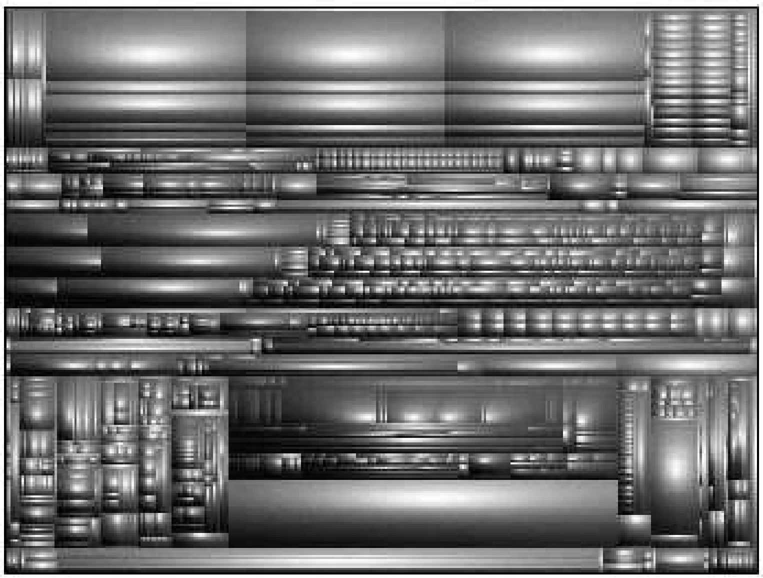
Showing Structure

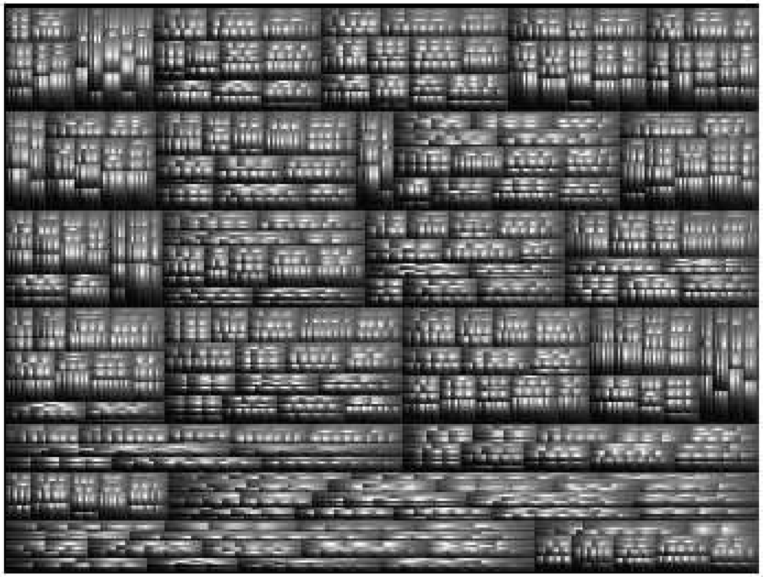
- Regular borderless treemap makes it challenging to discern structure of hierarchy, particularly large ones
  - Supplement Treemap view
  - Change rectangles to other forms

Cushion Treemap

Add shading and texture to help convey structure of hierarchy

Van Wijk & van de Wetering InfoVis '99



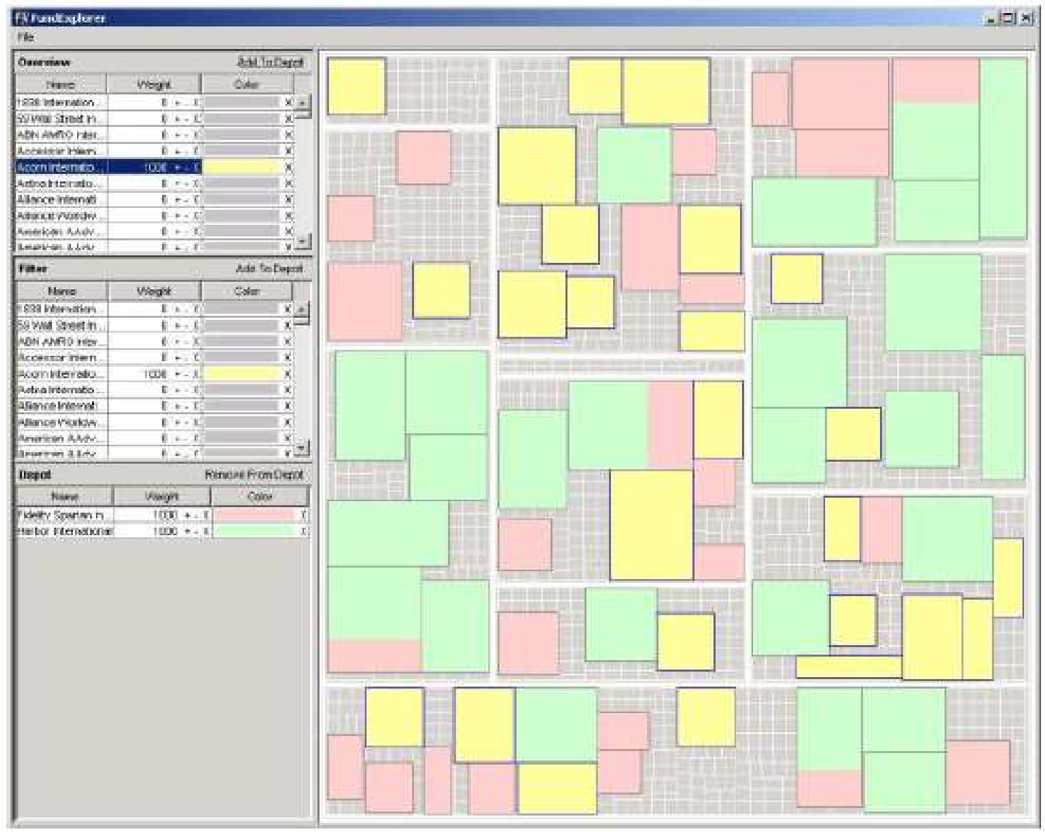


Context Treemaps

- What if nodes with zero value (mapped to area) are very important?
- Example: Stock or mutual fund portfolios: Funds you don't currently hold have zero value in your portfolio, but you want to see them to potentially buy them
- Context Treemap: Treemap with small distortion
- Give zero-valued items (all together) some constant proportion of screen area
- Provide dynamic query capabilities to enhance exploration leading to portfolio diversification

Fund Explorer

- Show mutual fund portfolios, including funds not currently held
  - Area maps to your relative investment in fund
- Want to help the user with portfolio diversification as well
  - If I add fund X, how does that overlap with my current fund holdings?



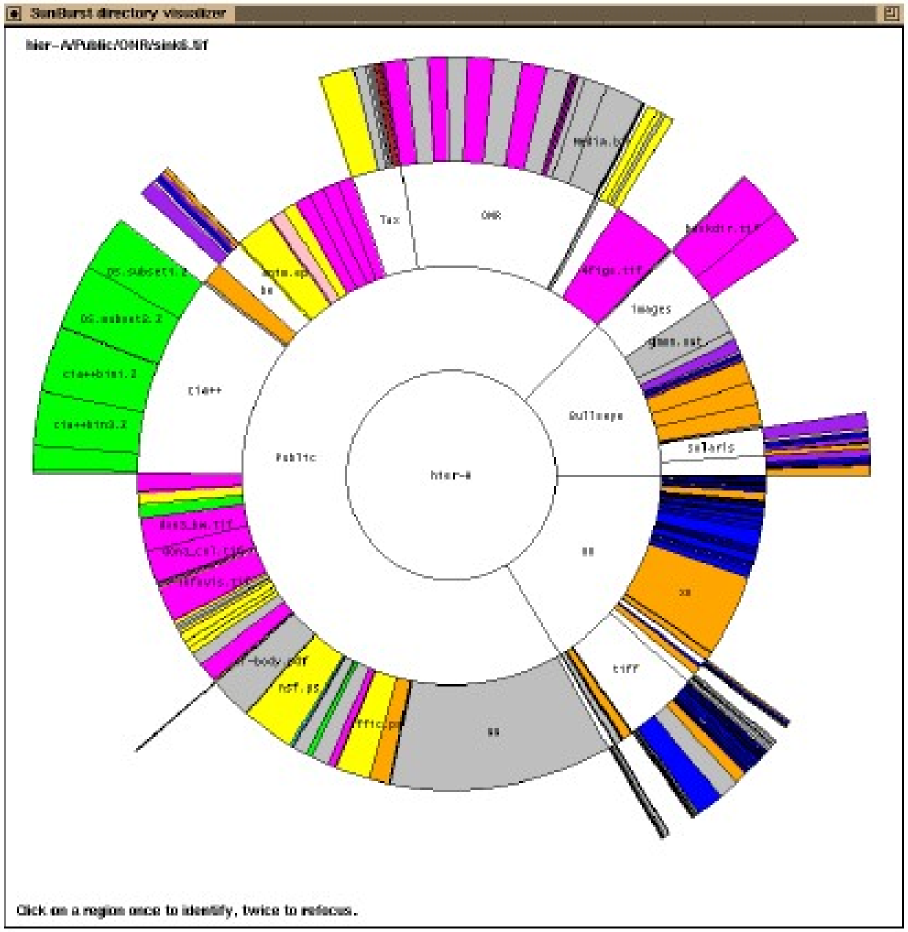
- Csallner, Handte, Lehmann & Stasko, InfoVis 2003

Another Technique

- What if we used a radial rather than a rectangular space-filling technique?
  - We saw node-link trees with root in center and growing outward already...
- Make pie-tree with root in center and children growing outward
  - Radial angle now corresponds to a variables rather than area

Sunburst

- Root directory at center, each successive level drawn farther out from center
- Sweep angle of item corresponds to size
- Color maps to file type or age
- Interactive controls for moving deeper in hierarchy, changing the root, etc.
- Double-click on directory makes it new root



Stasko and Zhang, Infovis 2000

Node-link or Space filling?

- Node-link typically better at exposing structure of information structure
- Space-filling good for focusing on one or two additional variables of cases

Reading

1. Reingold and Tilford: **Tidier Drawings of Trees.** *In IEEE Transactions on Software Engineering, March 1981* Available at: http://reingold.co/tidier-drawings.pdf

2. Robertson, Mackinlay and Card: **Cone Trees: Animated 3D visualizations of Hierarchical Information** *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems ACM CHI 1991.* Available at: https://www.researchgate.net/publication/221515543_

3. Card and Nation. **Degree-of-interest trees: A component of an attention-reactive user interface**. *Proceedings of the Working Conference on Advanced Visual Interfaces.* May 2002. Available at: https://www.researchgate.net/publication/228685532

4. Lamping and Rao. **The Hyperbolic Browser: A Focus + Context Technique for Visualizing Large Hierarchies**. *Journal of Visual Languages & Computing, March 1996*. Available at: https://www.researchgate.net/publication/220578919

5. Johnson and Shneiderman. **Tree Visualization with Tree-maps: 2-d space-filling approach.** *ACM Transactions on Graphics, Jan.1992*. Available at: https://www.cs.umd.edu/users/ben/papers/Johnson1991Tree.pdf

6. Stasko and Zhang. Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations. Infovis 2000. Available at: https://faculty.cc.gatech.edu/~john.stasko/papers/infovis00.pdf