INTERACTIVE VISUALIZATION OF LARGE GRAPHS AND NETWORKS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Tamara Munzner

June 2000

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Pat Hanrahan
(Principal Adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Marc Levoy

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Terry Winograd

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Stephen North
(AT&T Research)

Approved for the University Committee on Graduate Studies:

Abstract

Many real-world domains can be represented as large node-link graphs: backbone Internet routers connect with 70,000 other hosts, mid-sized Web servers handle between 20,000 and 200,000 hyperlinked documents, and dictionaries contain millions of words defined in terms of each other. Computational manipulation of such large graphs is common, but previous tools for graph visualization have been limited to datasets of a few thousand nodes.

Visual depictions of graphs and networks are external representations that exploit human visual processing to reduce the cognitive load of many tasks that require understanding of global or local structure. We assert that the two key advantages of computer-based systems for information visualization over traditional paper-based visual exposition are interactivity and scalability. We also argue that designing visualization software by taking the characteristics of a target user's task domain into account leads to systems that are more effective and scale to larger datasets than previous work.

This thesis contains a detailed analysis of three specialized systems for the interactive exploration of large graphs, relating the intended tasks to the spatial layout and visual encoding choices. We present two novel algorithms for specialized layout and drawing that use quite different visual metaphors. The H3 system for visualizing the hyperlink structures of web sites scales to datasets of over 100,000 nodes by using a carefully chosen spanning tree as the layout backbone, 3D hyperbolic geometry for a Focus+Context view, and provides a fluid interactive experience through guaranteed frame rate drawing. The Constellation system features a highly specialized 2D layout intended to spatially encode domain-specific information for computational linguists checking the plausibility of a large semantic network created from dictionaries. The Planet Multicast system for displaying the tunnel topology of the Internet's multicast backbone provides a literal 3D geographic layout of arcs on a globe to help MBone maintainers find misconfigured long-distance tunnels.

Each of these three systems provides a very different view of the graph structure, and we evaluate their efficacy for the intended task. We generalize these findings in our analysis of the importance of interactivity and specialization for graph visualization systems that are effective and scalable.

Acknowledgments

I would like to thank the many people who have helped me on the path towards this dissertation, both during and before my time at Stanford.

I am extremely grateful for the opportunity to have had Pat Hanrahan as an advisor for the past five years. Pat has been an inspiration to me since I first met him at The Geometry Center in 1991. When I was considering graduate schools a few years later, a major part of my decision-making procedure was to read through the previous decade of Siggraph proceedings. I ended up at Stanford because I found that the papers that most delighted me had Pat's name on them. What I value most about this past five years was the opportunity to absorb not only his insights into the specifics of my work, but his fundamental approach to research that emphasizes rigor and first principles.

I next thank the members of my reading committee for their time and energy in helping me improve this document: Marc Levoy, Terry Winograd, and Stephen North. I have had the pleasure of learning from them in other ways as well. Listening to a quarter of Marc's lectures on rendering as his teaching assistant provided a valuable lesson on how to simultaneously engage and challenge an audience. I was able to observe Terry's approach to project management through my involvement in the Interactive Mural project. Stephen's thorough knowledge of the field of graph drawing has been a valuable resource, and I appreciate his trip from the East Coast to attend my oral defense.

Several fellow graduate students share an interest in information visualization. My deepest gratitude is due to François Guimbretière for being a coauthor, a friend, and an intellectual sparring partner. His insightful commentary on drafts of this entire dissertation and many previous papers has been immensely useful in helping me clearly communicate my sometimes inchoate thoughts. Our innumerable hours spent discussing the issues of information visualization and related topics have been a major influence in my vision of the entire field. Maneesh Agrawala, Robert Bosch, Chris Stolte, and Diane Tang have been both intellectual and personal comrades, and have also spent many hours reading paper drafts.

I have enjoyed the company of my several officemates: thanks to Lucas Pereira for his stupendous enthusiasm and nearly irrepressible ability to take joy in life, Sean Anderson for many hours of illuminating conversations and the occasional piano serenade, Karen Butler for her level-headed companionship, and Larry Page for an uncounted number of jovial arguments.

I also thank many of the other people in the 3B wing who have contributed to its convivial atmosphere, including Andrew Beers, Cindy Chen, Greg Humphreys, Craig Kolb, David Koller, and Gordon Stoll. John Owens has gathered vast amounts of good karma by proofreading this entire document. Thanks to John Gerth for keeping the graphics lab machines running despite any and all fits of tempermentality on their part, and also for sharing his accumulated wisdom in the fields of visualization and graphics. Ada Glucksman's continual good cheer was contagious as she shielded me as much as possible from the bureaucratic maw of the university, as did Sarah Beizer. Thanks to James Davis for being my partner in adversity through the past four years of designing, building and supporting the video lab. Thanks also to Phil Lacroute, who helped me reverse-engineer and transport the previous video rack to the then-new Gates Building.

Thanks to many current and former people in the Stanford Computer Science department for their friendship over the years, including Guido Appenzeller, Edouard Bugnion, Stuart Cheshire, Tom Costello, Pavani Diwanji, Denis Leroy, Dave Ofelt, Anna Patterson, and Beth Seamans. Lise Getoor has in particular been my comrade in arms for the past five years.

I gratefully acknowledge the efforts of the rest of the Site Manager product team at Silicon Graphics: Ken Kershner, Greg Ferguson, Alan Braverman, Donna Scheele, Doug O'Morain, and Julie Brodeur. I also thank the following people and organizations for the use of the data used in Chapter 3: function call graph data from Anwar Ghuloum of the Stanford University Intermediate Format (SUIF) compilers group, Autonomous Systems data from Hans-Werner Braun of the National Laboratory for Applied Network Research (NLANR) and David M. Meyer of the University of Oregon Route Views Project, and Internet routing and Autonomous System data from Daniel W. McRobb of the Consortium for Applied Internet Data Analysis (CAIDA)

I am grateful for the time and ideas of the computational linguists from Microsoft Research who were the target users of the Constellation systems: Lucy Vanderwende, Bill Dolan, and Mo Corston-Oliver. Thanks also to Mike Barnett for providing MindNet support, Mary Czerwinski for her significant contributions to the user-centered design and evaluation process, and George Robertson of the User Interface group at Microsoft Research for hosting me as an intern during the summer of 1998.

This work was mainly supported by the National Science Foundation Graduate Research Fellowship Program and the Microsoft Research Graduate Fellowship Program. Additional support was also provided by the Advanced Research Projects Agency (grant 2DMA818) and Silicon Graphics. I am also grateful for the support of the SFB288 Differential Geometry and Quantum Physics group at the Technical University of Berlin, where I spent a refreshing and productive summer as a visiting researcher before starting the graduate program at Stanford.

I have had the great fortune of benefiting from the wisdom, encouragement, and friendship of many mentors. The somewhat roundabout causal chain that led me to this dissertation started with two junior high school teachers. In eighth grade my science teacher Dennis Searle encouraged me to apply for a formal industry mentorship program, which led me to spend the summer learning FORTRAN at Control Data's supercomputer division. I am extremely grateful that my mentor Dick Kachelmeyer gave his time unstintingly, both in person that summer and in many phone conversations over the years. He has been advising me to get a PhD since I was twelve years old, and my decision to both start grad school and stay here until finished is in no small part due to his unswerving advocacy. Dick was also instrumental in helping me find my first job in the computer industry: I spent three summers at Control Data's supercomputer spinoff company ETA Systems.

The other life thread started in sixth grade, when my math teacher Sally King allowed and encouraged me to start working independently. I ended up in the accelerated classes of the University of Minnesota Talented Youth Mathematics Project. This project not only gave me the opportunity to finish all the mathematics requirements for an engineering degree by the end of high school, but also led to my next summer job thanks to a follow-on program to help alumni find interesting internships. These two threads of my life intertwined when my resume was sent to the Geometry Supercomputer Project, in what appears to be a clear case of word-based pattern matching.

It was during my summer at the GSP that I fell in love with computer graphics, and I returned there as a technical staff member after I received my undergraduate degree from Stanford, by which time the GSP had become the NSF-funded Geometry Center. Charlie Gunn and Stuart Levy were both friends and mentors during my four years there, which were instrumental in shaping my career goals. I thank Charlie for sharing his experience through a broad range of lessons about graphics, mathematics, life in industry, and life in general. I am grateful that I was able to absorb a small part of Stuart's immense technical knowledge, and also to be exposed to his philosophy of life: he is the gentlest person that I have ever met, and one of the kindest. While at the Center, I also learned from Al Marden, Dick McGehee, Mark Phillips, and George Francis. I am also grateful for the continuing friendship of former colleagues Celeste Fowler and Nina Amenta.

I am thankful that many people in my field have offered both sound professional advice and welcome friendship, including Nancy Blachman, Paul Burchard, Andrew Hanson, Jan Hardenburgh, Mary Lou Jepsen, Bill Lorensen, Delle Maxwell, Theresa-Marie Rhyne, Maureen Stone, and Betsy Zeller. Many friends have helped me stay sane through these sometimes difficult years, including Ron Avitzur, Julian Cash, Karen Cassil, Thida Cornes, Cassidy Curtis, Craig DeForest, Collette Duplessis, John Gilmore, Ian Goldberg, Jed Hartman, Dikran Karagueuzian, Kevin Lahey, Sarah Liberman, Mark Lottor, Tsutomu Shimomura, Peter Shipley, Lynn Stewart, and Meg Worley.

Thanks to k claffy for being both a friend for the past dozen years and the impetus of the Planet Multicast project, introducing me to my coauthors Bill Fenner and Eric Hoffman. I am deeply appreciative of Eric's companionship, both intellectual and personal, over the past four years.

I acknowledge, appreciate, and return the love and support of my family, without whom I would be lost. My parents Joan and Aribert Munzner have been my emotional anchors through not only the vagaries of graduate school, but my entire life. My sister's partner Sheila Oehrlein has also become an important person in my world. My sister Naomi will always share a part of my soul, and I dedicate this thesis to her.

To Naomi Paisley Munzner

Contents

Abstract **iv**

Acknowledgments **v**

1 Motivation **1**

1.1 Three Design Studies 2

1.2 Information Visualization 3

1.2.1 Visual Encoding 4

1.2.2 Integral vs. Separable Dimensions 5

1.2.3 Preattentive Processing 5

1.3 Approach 6

1.3.1 Interactivity 6

1.3.2 Domain and Task Focus 7

1.3.2.1 Evaluation 7

1.3.3 Scalability 9

1.4 Contributions 10

1.5 Thesis Organization 11

2 Related Work **12**

2.1 Deliberate Distortions 12

2.2 Graph Drawing 13

2.2.1 Geographical Systems 13

2.2.2 Hierarchies 14

2.2.3 Distortion-Based Graph Drawing 15

2.2.4 Topological Force-Directed Systems 16

2.2.5 Online and Incremental Layouts . . . 17
2.2.6 Other Approaches . . . 17
2.3 Automatic Presentation Systems . . . 18

3 H3: 3D Hyperbolic Quasi-Hierarchical Graphs 19
3.1 Quasi-Hierarchical Graphs . . . 20
3.1.1 H3 Task Domains . . . 21
3.1.1.1 External Representation Functions . . . 21
3.1.1.2 Hyperlink Structure for Webmasters . . . 22
3.1.1.3 Function Call Graph Structure for Compiler Users . . . 24
3.2 Spatial Layout . . . 26
3.2.1 Nonlinear Distortion . . . 26
3.2.2 Hyperbolic Geometry . . . 26
3.2.2.1 Exponential Room . . . 27
3.2.2.2 Projection . . . 27
3.2.3 Tree Layout . . . 32
3.2.3.1 Bottom-up Estimation Pass . . . 34
3.2.3.2 Top-down Placement Pass . . . 35
3.2.3.3 Sphere Packing . . . 35
3.2.3.4 Derivation . . . 37
3.2.3.5 Single-pass Recursive Algorithm . . . 40
3.3 Drawing . . . 40
3.3.1 Adaptive Drawing . . . 40
3.3.1.1 Candidate Nodes for Drawing . . . 40
3.3.1.2 Active, Idle, and Pick Modes . . . 41
3.3.2 Drawing Implementation . . . 43
3.3.2.1 Active Mode . . . 43
3.3.2.2 Idle Mode . . . 44
3.3.2.3 Pick Mode . . . 44
3.3.3 Label Drawing . . . 45
3.4 Visual Encoding . . . 46
3.4.1 Visual Metaphor . . . 46
3.4.1.1 Effects of Distortion . . . 46

3.4.1.2 Dimensionality 47
3.4.1.3 Information Density 47
3.4.2 Grouping 49
3.5 Implementation 50
3.5.1 Linked Views 50
3.5.2 Precision 54
3.5.3 Availability 55
3.6 Interaction 55
3.6.1 Navigation 56
3.6.2 Non-tree links 57
3.7 Results 58
3.7.1 Visual Appearance 58
3.7.2 Speed and Size 61
3.7.3 User Study 62
3.7.4 Outcomes 65
3.7.5 Additional Task Domains 66

4 Planet Multicast: Geographic MBone Maintenance 69
4.1 The Multicast Backbone Maintenance Task 70
4.1.1 Multicast 70
4.1.2 Tunnels 71
4.1.3 Tunnel Placement 71
4.1.4 Target Users 72
4.1.5 Topology Data 72
4.2 Visual Metaphor 73
4.2.1 Geographic Distance 73
4.2.2 Implications of 3D Globe 74
4.2.3 Spherical Geodesics 77
4.3 Visual Encoding 78
4.4 Implementation 78
4.4.1 Geographical Determination 79
4.4.2 Browsers 80
4.4.3 Availability 80

4.5 Results . . . 81
4.5.1 Topology Insights . . . 81
4.5.2 Additional Task Domains . . . 82
4.5.3 Outcomes . . . 84
4.5.4 Barriers to Adoption . . . 85

5 Constellation: Linguistic Semantic Networks 87

5.1 The Linguistic Plausibility-Checking Task . . . 87
5.1.1 The MindNet Semantic Network . . . 88
5.1.2 Plausibility-Checking Task . . . 89
5.1.3 Visualization Requirements . . . 91
5.2 Spatial Layout . . . 92
5.2.1 Spatial Position . . . 92
5.2.2 Paths . . . 93
5.2.3 Curvilinear Grid . . . 96
5.2.4 Associating Definition Graphs with Pathwords . . . 98
5.2.5 Path Segment Layout . . . 98
5.2.6 Definition Graph Layout . . . 99
5.2.7 Increasing the Layout Density . . . 102
5.2.7.1 Path Segment Elision . . . 103
5.2.7.2 Horizontal Space . . . 103
5.2.7.3 Vertical Space . . . 103
5.2.7.4 Aspect Ratio . . . 105
5.2.7.5 Alternatives . . . 105
5.2.8 Graph-Theoretic Description . . . 105
5.2.9 Text Layout . . . 106
5.2.10 Adaptive Segment Division . . . 107
5.3 Visual Encoding . . . 109
5.3.1 Constellations . . . 109
5.3.2 Perceptual Channels . . . 112
5.4 Interaction . . . 113
5.4.1 Interactive Visual Emphasis . . . 114
5.4.1.1 Pie Flipper . . . 114

5.4.1.2 Hovering . . . 115
5.4.2 Navigation . . . 116
5.5 Implementation . . . 118
5.6 Results . . . 118
5.6.1 Discussion . . . 118
5.6.2 Layout Efficacy . . . 119
5.6.3 Outcomes . . . 121

6 Discussion **123**
6.1 General Discussion . . . 123
6.1.1 Visual Popout . . . 123
6.1.2 Hidden State . . . 126
6.1.3 Ordering Encoding . . . 126
6.1.4 Dissemination . . . 128
6.2 Future Work . . . 129
6.2.1 H3 . . . 129
6.2.1.1 Incremental Layout . . . 129
6.2.1.2 Web Visualization for End-users . . . 129
6.2.1.3 Visualizing the Entire Web . . . 130
6.2.1.4 Visualizing Changes over Time . . . 130
6.2.2 Planet Multicast . . . 131
6.2.3 Constellation . . . 131
6.2.4 New Directions . . . 132
6.2.4.1 Principled Design . . . 132
6.2.4.2 Information Visualization as New Interface Paradigm . . . 132
6.3 Conclusion . . . 133

Bibliography **135**

List of Tables

3.1 **Euclidean and hyperbolic formulas.** 37

3.2 **Maximum information density comparison for two hyperbolic browsers.** 48

3.3 **Codimension comparison for three graph drawing systems.** 48

4.1 **Raw tunnel data from `mwatch`.** 73

5.1 **Color scheme used for the visualization, in both HSB and RGB.** 113

List of Figures

1.1 **Range of specificity.** . . . 2

1.2 **System scalability and dataset size.** . . . 9

3.1 **Constructing a spanning tree for quasi-hierarchical web site.** . . . 23

3.2 **Constructing a spanning tree for quasi-hierarchical function call graph.** . . . 25

3.3 **Exponential volume of hyperbolic space.** . . . 28

3.4 **Euclidean equidistant parallels vs. hyperbolic divergent parallels.** . . . 29

3.5 **Models of hyperbolic space.** . . . 30

3.6 **1D hyperbolic projection.** . . . 31

3.7 **2D hyperbolic projection.** . . . 31

3.8 **Circumference vs. hemisphere.** . . . 33

3.9 **Discs vs. spherical caps.** . . . 34

3.10 **Circle packing.** . . . 35

3.11 **Band layout.** . . . 36

3.12 **Child hemisphere placement.** . . . 38

3.13 **Active vs. idle frames, obvious case.** . . . 41

3.14 **Active vs. idle frames, subtle case.** . . . 42

3.15 **Site Manager.** . . . 51

3.16 **Linked views.** . . . 52

3.17 **Traffic log playback.** . . . 53

3.18 **Hyperbolic motion over a 30,000 element Unix file system.** . . . 56

3.19 **Non-tree links.** . . . 57

3.20 **Part of the Stanford graphics group web site drawn as a graph in 3D hyperbolic space.** 59

3.21 **Call graph.** . . . 60

3.22 **Link structure of a web site laid out in 3D hyperbolic space.** 60
3.23 **XML3D interface.** 63
3.24 **XML3D vs. 2D interface study results.** 64
3.25 **Autonomous System paths analysis.** 67

4.1 **Transport protocols.** 70
4.2 **Tunnelling.** 71
4.3 **MBone shown as arcs on a globe.** 74
4.4 **Horizon view, arc height and grouping.** 76
4.5 **Textured globes.** 77
4.6 **Thresholding.** 79
4.7 **Two regional closeup views of the MBone.** 81
4.8 **MBone tunnels grouped by backbone status.** 83
4.9 **MBone tunnels of the major backbone networks, colored by provider.** 84
4.10 **MBone tunnel structure in Texas at two different times.** 85

5.1 **Parsed definition graph from MindNet.** 89
5.2 **Previously existing textual view in MindNet.** 90
5.3 **Traditional layouts avoid crossings to prevent false attachments.** 93
5.4 **Plausibility gradient encodes a domain-specific attribute.** 94
5.5 **Selective emphasis avoids perception of false attachments.** 94
5.6 **Paths in grids.** 95
5.7 **Bad approaches.** 96
5.8 **Curvilinear grid.** 97
5.9 **Attaching definitions to path segments.** 99
5.10 **Definition graph layout.** 100
5.11 **Very early layout with empty proxy slots.** 101
5.12 **Early sparse layout.** 102
5.13 **Adjusting grid for maximum information density.** 104
5.14 **Viewing levels.** 108
5.15 **Constellations.** 110
5.16 **Relation type constellations.** 111
5.17 **Pie flipper.** 115

5.18 **Hovering.** 116
5.19 **Zooming.** 117
5.20 **Three effective viewing levels.** 119
5.21 **Dissimilar query words.** 120
5.22 BIRD-FEATHER **10 path dataset.** 120
5.23 **Layouts of kangaroo-tail dataset using pre-existing systems.** 121

6.1 **Visual popout.** 124
6.2 **Avoiding visual artifacts from word sizing.** 125

Chapter 1

Motivation

Node-link graphs are simple, powerful, and elegant abstractions that have broad applicability in computer science and many other fields. Any domain that can be modelled as a collection of linked nodes can be represented as a graph. For example, in the domain of the World-Wide Web, nodes represent web pages and links represent hyperlinks. For a dictionary, nodes represent words and links represent word relationships such as `is-a`, `part-of`, `modifier`, and so on. Biological taxonomies are trees, which are a subset of general graphs: nodes represent species, and links represent evolutionary descent. In a graph of the Internet, nodes could represent routers and links would imply direct network connectivity.

The field of graph theory offers a powerful set of domain-independent algorithms for computationally manipulating graphs efficiently, even if they are very large. Graphs have a natural visual representation as nodes and connecting links arranged in space. Visual representations of small graphs are pervasive: people routinely sketch such a picture when thinking about a domain, or include pictures of graphs in explanatory documents.

An informal statement that explains the popularity of graph pictures is that people must find an explicit visual representation of the graph structure helpful for some tasks. A more formal analysis of their utility is that visual depictions of graphs and networks are **external representations** that exploit human visual processing to reduce the cognitive load of a task. Endeavors that require understanding global or local graph structure can be handled more easily when that structure is interpreted by the visual processing centers of the brain, often without conscious attention, than when that structure has to be cognitively inferred and kept in working memory. External representations change the nature of a task: an external memory aid anchors and structures cognitive behavior by providing information that can be directly perceived and used without being interpreted and formulated explicitly [Zha91]. A graph is a **topo-visual formalism**: that is, the important aspect of

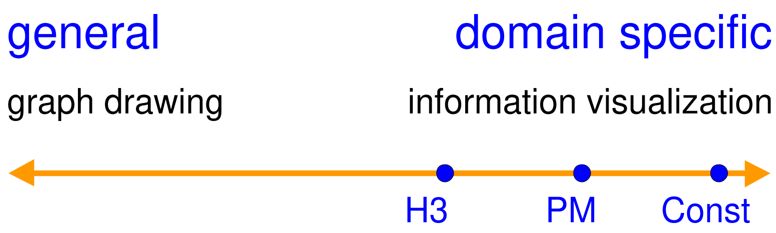

Figure 1.1: **Range of specificity.** All three systems in this thesis trade off generality for scalability, ranging from the relatively general H3 system suited for the class of quasi-hierarchical graphs, to the more focused geographic network visualization of the Planet Multicast system, to the highly targeted Constellation system.

drawn graphs is non-metric topological connectedness as opposed to pure geometric distance [Har88, Noi94].

1.1 Three Design Studies

In this thesis we extend the reach of graph drawing using ideas from information visualization, particularly by incorporating interactivity and domain-specific information. We present and analyze three specialized systems for interactively exploring large graphs. The common choice in all three design studies was to relax the constraint for total generality so as to achieve greater scalability and effectiveness. The amount of domain-specific specialization, as shown in Figure 1.1, ranges from a highly targeted system designed to fit the needs very small user community of computational linguists, to the middle ground of a geographic network layout, to a more general layout originally intended for the hyperlink structure of web sites that is suited for a entire class of graphs that we call *quasi-hierarchical*.

The goal of this thesis is two-fold: not only the creation of new algorithms for graph layout and drawing, but also an analysis that links a specific intended task with our choices of spatial layout and encoding. Our analysis provides an evaluation of these graph drawing systems from an information visualization perspective that distills the lessons that we learned from building these specific systems into a more general framework for designing and evaluating visualization systems.

The three software systems that we present are Planet Multicast, H3, and Constellation. The Planet Multicast system was developed in 1996 for displaying the tunnel topology of the Internet's multicast backbone, and provides a literal 3D geographic layout of arcs on a globe to help MBone maintainers find potentially misconfigured long-distance tunnels. The H3 system was developed between 1996 and 1998 for visualizing the hyperlink structures of web sites. It scales to datasets of over 100,000 nodes by using a carefully chosen spanning tree as the layout backbone, presents a Focus+Context view using 3D hyperbolic geometry, and provides a fluid interactive experience through guaranteed frame rate drawing. The Constellation system was developed between 1998 and 1999, and features a highly specialized 2D layout intended to spatially encode domain-specific information for computational linguists checking the plausibility of a large semantic network created from dictionaries.

1.2 Information Visualization

The field of computer-based information visualization draws on ideas from several intellectual traditions: computer science, psychology, semiotics, graphic design, cartography, and art. The two main threads of computer science relevant for visualization are computer graphics and human-computer interaction. The areas of cognitive and perceptual psychology offer important scientific guidance on how humans perceive visual information. A related conceptual framework from the humanities is semiotics, the study of symbols and how they convey meaning. Design, as the name suggests, is about the process of creating artifacts well-suited for their intended purpose. Cartographers have a long history of creating visual representations that are carefully chosen abstractions of the real world. Finally, artists have refined methods for conveying visual meaning in subdisciplines ranging from painting to cinematography.

Information visualization has gradually emerged over the past fifteen years as a distinct field with its own research agenda. The distillation of results from areas with other goals into specific prescriptive advice that can help us design and evaluate visualization systems is nontrivial. Although these traditions have much to offer, effective synthesis of such knowledge into a useful methodology of our own requires arduous gleaning.

The standard argument for visualization is that exploiting visual processing can help people explore or explain data. We have an active field of study because the design challenges are significant and not fully understood. Questions about visual encoding are even more central to information visualization than to scientific visualization. The subfield names grew out of an accident of history, and have some slightly unfortunate connotations when juxtaposed: information visualization is not unscientific, and scientific visualization is not uninformative. The distinction between the two still not agreed on by all, but the definition used here is that **information visualization** hinges on finding a spatial mapping of data that is not inherently spatial, whereas **scientific visualization** uses a spatial layout that is implicit in the data.

Many scientific datasets have naturally spatialized data as a substrate: for instance, airflow over an airplane wing is given as values of a three-dimensional vector field sampled at regular intervals that provides an implicit 3D spatial structure. Scientific visualization would use the same 3D spatialization in a visual representation of the dataset, perhaps by drawing small arrows at the spots where samples were taken, pointing in the direction of the fluid flow at that spot, with color coded according to velocity. Scientific visualization is often used as an augmentation of the human sensory system, by showing things that are on timescales too fast or slow for the eye to perceive, or structures much smaller or larger than human scale, or phenomena such as X-rays or infrared radiation that we cannot directly sense.

In contrast, a typical information visualization dataset would be a database of film information that is more abstract than the previous example, since it does not have an underlying spatial variable. One possible spatialization would be to show a 2D scatterplot with the year of production on one axis and the film length on the other, with the scatterplot dots colored according to genre [AS94]. That choice of spatialization was an explicit choice of visual metaphor by the visualization designer, and is appropriate for some tasks but not for others.

1.2.1 Visual Encoding

In all visualizations, graphical elements are used as a visual syntax to represent semantic meaning [RRGK96]. For instance, in the scientific fluid flow dataset mentioned earlier, the position of the arrow denotes the position of that sample data point, the direction of the fluid flow was mapped to the orientation of the graphical element of an arrow, and the color of that element represented fluid velocity. We call these mappings of information to display elements **visual encodings**, and the combination of several encodings in a single display results in a complete **visual metaphor**.

Several people have proposed visual encoding taxonomies, including Bertin [Ber83, Ber81], Cleveland [CM84] [Cle94, Chapter 4], Mackinlay [CM97a, Mac86a, Mac86b] [CMS99, Chapter 1], and Wilkinson [Wil99a]. The fundamental substrate of visualizations is spatial position. Marks such as points, lines, or area-covering elements can be placed on this substrate. These marks can carry additional information independent of their spatial position, such as size, greyscale luminance (brightness) value, surface texture density, color hue, color saturation, curvature, angle, and shape. The literature contains many different names for these kinds of visual encodings: retinal variables, retinal attributes, elementary graphical perception tasks, perceptual tasks, perceptual dimensions, perceptual channels, display channels, display dimensions, and so on.

The critical insight of Cleveland was that not all perceptual channels are created equal: some have provably more representational power than others because of the constraints of the human perceptual system [CM84]. Mackinlay extended Cleveland's analysis with another key insight that the efficacy of a perceptual channel depends on the characteristics of the data [Mac86a]. The levels of measurement originally proposed by Stevens [Ste46] classify data into types. **Nominal** data has items that are distinguishable but not ranked: for instance, the set of fruit contains apples and oranges. **Ordinal** data has an explicit ordering that allows ranking between items, for example mineral hardness. **Quantitative** data is numeric, such that not only ranking but also distances between items is computable.[1]

The efficacy of a retinal variable depends on the data type: for instance, hue coding is highly salient for nominal data but much less effective for quantitative data. Size or length coding is highly effective for quantitative data, but less useful for ordinal or nominal data. Shape coding is ill-suited for quantitative or ordinal data, but somewhat more appropriate for nominal data.

Spatial position is the most effective way to encode any kind of data: quantitative, ordinal, or nominal. The power and flexibility of spatial position makes it the most fundamental factor in the choice of a visual metaphor for information visualization. Its primacy is the reason that we devote entire sections to spatial layout in later chapters, separate from the section discussing all other visual encoding choices.

1.2.2 Integral vs. Separable Dimensions

Perceptual dimensions fall on a continuum ranging from almost completely separable to highly integrated. Separable dimensions are the most desirable for visualization, since we can treat them as orthogonal and combine them without any visual or perceptual "cross-talk". For example, position is highly separable from color. In contrast, red and green hue perceptions tend to interfere with each other because they are integrated into a holistic perception of yellow light.

There is a fundamental tension in visualizing a network of nodes connected by links because of the interference of two perceptual conflicts: proximity and size. The Gestalt proximity principle means that nodes drawn close together are perceived as related, whereas those drawn far apart are unrelated. However, the size channel has the opposite effect: a long edge is more visually salient than a short one. Almost all graph drawing systems solve this conflict by choosing proximity instead of size, which makes sense given the primacy of spatial positioning described in section 1.2.1.

1.2.3 Preattentive Processing

Another fundamental cognitive principle is whether processing of information is done deliberately or pre-consciously. Some low-level visual information is processed automatically by the human perceptual system without the conscious focus of attention. This type of processing is called **automatic**, **preattentive**, or **selective**. An example of preattentive processing is the **visual popout** effect that occurs when a single yellow object is instantly distinguishable from a sea of grey objects, or a single large object catches one's eye. Exploiting pre-cognitive processing is desirable in a visualization system so that cognitive resources can be freed up for other tasks. Many features can be preattentively processed, including length, orientation,

[1] Stevens also distinguished ratio from interval (his term from quantitative).

contrast, curvature, shape, and hue [TG88]. However, preattentive processing will work for only a single feature in all but a few exceptional cases, so most searches involving a conjunction of more than one feature are not pre-cognitive. For instance, a red square among red and green squares and circles will not pop out, and can be discovered only by a much slower conscious search process.

1.3 Approach

In addition to the design principles of the previous section, our information visualization approach differs from traditional graph drawing by our emphasis in three key areas: interactivity, domain specificity, and scalability.

1.3.1 Interactivity

The word **interaction** is often used in different contexts, sometimes interchangeably with **animation**. The following four meaning are often conflated, but in this thesis we only intend the first three:

Navigation Interactive navigation consists of changing either the viewpoint or the position of an object in a scene.

Making Choices Interactivity is also common in non-navigational settings, for example through radio buttons on a control panel or menu choices that affect the display.

Animated Transitions Viewers have a much easier time retaining their mental model of an object if changes to its structure or its position are shown as smooth transitions instead of discrete jumps [RCM93].

VCR-style Animation Many studies of multimedia applications have compared user performance between still imagery and prescripted animations where the user has start, pause, and stop controls [MTB00].[2]

Interactivity is the great challenge and opportunity of computer-based visualization. Visual exposition has a long and successful historical tradition, but until recently it was confined to static two-dimensional media such as paper. The invention of film and video led to new kinds of visual explanations that could take advantage of dynamic animation. The advent of computers sets the stage for designing interactive visualization systems of unprecedented power and flexibility.

The most straightforward kind of interactive visualization system mimics the real world. A two-dimensional interface can implement the semantics of paper by allowing panning and zooming. In three dimensions, virtual objects can be manipulated like real objects with controls for rotation, translation, and scaling. Literal

[2] The findings on the lack of benefit of many animations run counter to the assumptions of many people.

interaction is relatively well-understood: much of the cognitive psychology literature on graphical perception focuses on the kinds of displays that can be drawn equally well on a piece of paper as on a computer screen[CM84]. Most of the effective knowledge transfer from cognitive psychology to visualization has been the theory of perceptual dimensions and retinal variables for static 2D displays.

Computers offer the possibility of moving beyond simple imitations of reality, since we can tie user input to the visual display to get semantics impossible in the real world. For instance, distortion methods allow the user to see a large amount of context around a changeable area of focus, and multiscale methods result in the visual appearance of an object changing radically based on distance from the user's virtual viewpoint. However, only a few of the cognitive principles involving exotic semantics are understood: for instance, studies on environmental [Gol87, Section 5.4.6: Cognitive Distance] and spatial [Tve92] cognition provide some evidence that appropriate distortion is cognitively defensible. Moreover, perceptual channel taxonomies have been extended to nonstatic attributes such as velocity, direction, frequency, phase, and disparity [Gre98]. Although these studies provide some guidance, there is a huge parameter space of possible interaction techniques that has not yet been thoroughly analyzed. Two of the three design studies in this thesis are forays into the parameter space of nonliteral interaction techniques.

1.3.2 Domain and Task Focus

A hallmark of many information visualization systems is a focus on the tasks of a group of intended users in a particular domain. Methods from user-centered design [ND86] and ethnography can help the visualization practitioner understand the workflow of a user group to understand their high-level goals. For instance, the goals of webmasters would be to create and maintain a web site.

However, these goals are too high level to directly address with software: they must be broken down into tasks at a lower level [MT93]. In the webmaster example, a low-level task might be optimizing end-user navigation by minimizing the number of hops between the entry page and other pages deemed important by the site designer, or finding and fixing broken links. Such tasks are specific enough that a visualization designer can make decisions about appropriate visual encodings, so that perceptual inferences can be substituted for the more cognitively demanding logical inferences [Cas91]. Finally, the low-level task breakdown provides a handle for evaluating the effectiveness of the resulting visualization system.

1.3.2.1 Evaluation

Evaluating a visualization system is much more difficult than evaluating most graphics systems, because it is hard to judge whether some piece of software really helped somebody get something done more easily.

Graphics software such as a rendering system can be quantitatively evaluated on whether it is faster or more photorealistic than previous work. Some aspects of visualization are similarly quantitative: the implicit (or explicit) assumption that a previous technique is effective allows researchers to argue that a new algorithm is better because it is faster or scales to larger datasets.

However, the effectiveness criteria for a visualization system are far less understood than the low-level psychophysics of human vision. One way to document the effect of a visualization system is to mention the size of the user community that has chosen to adopt the software. User testing can be more rigorous, documenting not only whether people liked it, but whether performance for a particular task improved. User testing range from informal usability observations in an iterative design cycle to full formal studies designed to gather statistically significant results. User studies are championed by some as the path to scientific legitimacy, but are tricky to construct without confounding variables. Well-designed studies are a critical part of the evaluation arsenal, but it is sometimes difficult to convince others that the positive results of a study merit high-level conclusions about the validity of an approach. A less contentious use of user testing is for fine-tuning a visualization system by exposing the best choice from among similar alternatives. Anecdotal evidence of discoveries attributed by a user to insights from a visualization system is important in cases where user studies are infeasible because the target audience is small, or the task is something long-term and serendipitous such as scientific discovery.

Finally, an analysis which relates design choices to a conceptual framework is a powerful evaluation method. Such a conceptual analysis can be useful both as a means to evaluate the merits of a visualization system for a particular task, and to analyze what other tasks such a system might be well-suited to help with. Taxonomies and principles can help us go beyond simply asking *whether* something helps by offering tools to answer questions of *why* and *how* it helps [SR96]. Several authors have presented such frameworks, in addition to the taxonomies of Bertin, Cleveland, and Mackinlay discussed in Section 1.2.1. Shneiderman has also been active in taxonomizing the field [Shn96], with his "overview first, zoom and filter, then details-on-demand" mantra. Wilkinson offers an elaborate framework based on a design grammar that evolved out of his experience in designing statistical graphics [Wil99a]. Ware's recent textbook on information visualization provides a great deal of prescriptive advice based on a detailed analysis of the psychophysical and cognitive underpinnings of human perception [War00].

In this thesis we draw on ideas from several of these frameworks in our analytical evaluation of all three visualization systems. We also evaluate our software systems using a combination of the preceding methods. The H3 system evaluation includes algorithmic improvements over previous related techniques, a discussion of user adoption, and a formal user study. The Planet Multicast project evaluation consists mainly of anecdotal create highly aesthetic layouts of general graphs. A paper in 1994 declared graphs of more than 128 nodes to be "huge" [FLM94]. More than one-half of the existing graph drawing systems handle only very small input graphs of less than one hundred nodes [Döm94, FW94, Rei94, GT96]. A few exceptional systems such as Gem3D [BF95] and *dot* [GKNV93] can handle hundreds or even a few thousand nodes.

Although a few thousand or even a few hundred nodes is more than one would want to lay out by hand, Figure 1.2 shows that many real-world datasets are far larger. Backbone Internet routers have over 70,000 other hosts in their routing tables, and the number of hosts on the entire Internet is over 70 million and growing. Dictionaries contain millions of words defined in terms of each other. The Web consists of over a billion hyperlinked documents, and even moderately-sized single Web sites such as the Stanford graphics group site have over 100,000 documents.

1.4 Contributions

The contributions of this thesis fall into two areas: analysis and algorithms.

Our analytical contribution is a detailed analysis of three specialized systems for the interactive exploration of large graphs, relating the intended tasks to the spatial layout and visual encoding choices. Each of these three systems provides a completely different view of the graph structure, and we evaluate their efficacy for the intended task. We generalize these findings in our analysis of the importance of interactivity and specialization for graph visualization systems that are effective and scalable.

Our algorithmic contribution is two novel algorithms for specialized layout and drawing. The H3 algorithm trades off generality for scalability, whereas the Constellation algorithm trades off generality for effectiveness. The H3 system for visualizing quasi-hierarchical graphs has the following advantages over previous work:

- a highly scalable layout algorithm that handles very large datasets quickly (over 100,000 nodes in 12 seconds)
- a novel layout results in high information density without clutter, by exploiting mathematical advantages of 3D hyperbolic space for a Focus+Context view
- a guaranteed frame rate novel drawing algorithm that uses a combination of graph-theoretic and viewpoint dependent information for a fluid interactive experience

The Constellation system for visualizing paths through semantic networks has the following features:

- a novel specialized layout highly tuned for effectiveness in a carefully documented task by visually encoding domain-specific semantics
- a novel drawing algorithm tuned for maximum task-based label readability at multiple viewing levels
- a novel interaction technique of selectively highlighting sets of nodes and edges
- the effective use of multiple perceptual channels to visually distinguish interactively chosen foreground layer from unobtrusive background

The H3 project was a solo undertaking. The Planet Multicast project was joint work with Eric Hoffman, K. Claffy, and Bill Fenner. The Constellation project was joint work with François Guimbretière.

1.5 Thesis Organization

This thesis begins with motivation for the interactive visualization of graphs and background material on information visualization, followed by a summary of our original research contributions. We discuss related work in Chapter 2.

The next three chapters discuss the software systems at the core of this thesis. Each chapter begins with a task analysis, then covers spatial layout and visual encoding choices, and concludes with a discussion of the results.

The H3 system for visualizing large quasi-hierarchical graphs in 3D hyperbolic space is covered in Chapter 3. Parts of this chapter were described in series of publications. We have published the H3 layout algorithm [Mun97] and the H3Viewer guaranteed frame rate drawing algorithm [Mun98a]. We have also presented a brief overview of both the layout and drawing algorithms, augmented with a discussion of possible tasks that could benefit from a graph drawing system [Mun98b]. A paper that is in press describes the the user study that demonstrated statistically significant benefits of a browsing system incorporating the H3Viewer [RCMC00].

Chapter 4 describes Planet Multicast, a 3D geographic system that displays the tunnel topology of the Internet's multicast backbone as arcs on a globe to help MBone maintainers find potentially misconfigured long-distance tunnels. We have presented this system as a case study [MHCF96].

Chapter 5 is about the Constellation system for visualizing semantic networks using a custom 2D spatial layout. A brief paper described the key aspects of this visualization system [MGR99].

We finish with discussion, future work, and conclusions in chapter 6.

Chapter 2

Related Work

There are several relevant threads of related work. We have already discussed some of the core information visualization data- and task-based taxonomies in Chapter 1.

We begin with the previous work in the deliberate use of distortion to show as much context as possible around a focus point. The bulk of this chapter is a discussion of the many previous systems for drawing graphs and hierarchies, both topologically and geographically. Our main focus when reviewing previous systems for automatic graph drawing is their limited scalability. Of all the systems that we discuss in this chapter, only two of them handle very large datasets. In section 2.2.2 we cover the Cheops system, which has a highly compact display footprint for tree display that is more suited for an index than for exploration [BPV96]. On page 2.2.6 we cover the Nicheworks system for large graph exploration [Wil97, Wil99b], which was concurrent with our work on H3. Our discussion of effectiveness is more limited, since few of these systems attempt to specialize for a particular task to the degree that we pursue with our Constellation system. We end by justifying our choice to embark on design studies by discussing the limited relevance of automatic presentation systems, since their finite palette of visual encoding techniques does not extend to the domain of interactive presentations of large graphs.

2.1 Deliberate Distortions

One of the important challenges in a visualization system is how to present as much important information as possible given a finite display area. When the structure of interest is too big to see in detail all at once, the most straightforward solution is to allow the user to pan and zoom the visible area.[1] The disadvantage

[1] Or in a three-dimensional display, to rotate, translate, and zoom.

of simply providing navigation controls is that users often lose track of the position of their current viewport with respect to the global structure. Adding a smaller secondary window showing a global overview with the current viewport location marked can provide some guidance, but forcing users to continually switch their locus of attention from one window to another can still lead to disorientation.

A large class of visualization techniques have been developed to address this problem by attempting to smoothly integrate detail views with as much surrounding context as possible, so that users can see all relevant information in a single view. **Distortion** techniques of this sort have been given several more or less general names, including **Focus+Context** [RC94], nonlinear magnification [KR97], fisheye views [SB94, Fur86], and pliable surfaces [CCF95].[2] These categories are not completely interchangeable: the Magic Lens system [SFB94], which featured movable filters, falls into the Focus+Context category but is not a distortion technique. Multiscale views such as Pad++, where the visual appearance of an object changes radically based on the distance to the virtual viewpoint, share some of the ideas of distortion-based systems [BH94, PF93, FB95]. Leung and Apperly taxonomize distortion techniques that appeared in the literature before 1994 [LA94].

2.2 Graph Drawing

Early work on automatic graph layout and drawing is scattered through the computer science literature [FPF88, WS79, Moe90]. The first book devoted solely to graph drawing, by Battista and colleagues [BETT99], summarizes large areas of the field. The Graph Drawing conference series beginning in 1994 has resulted in proceedings that cover recent work in both systems and theory.[3] The focus of this thesis is systems, so we do not concentrate on the wealth of theoretical proofs about upper and lower algorithmic bounds: suffice it to say that most interesting computations on general graphs are NP-hard [Bra88].

2.2.1 Geographical Systems

A geographical view of a graph or network is appropriate for some tasks, particularly when showing telecommunication network topology or traffic information. The 1992 video by Cox and Patterson showed a "$2\frac{1}{2}$"-dimensional view of the NSFNet backbone rising above a flat map of the US from an oblique viewpoint [CP92]. The SeeNet [BEW95] system featured a totally flat 2D geographic layout of links on a map. The

[2] The Nonlinear Magnification Homepage maintained by Keahey links to downloadable versions of many of these papers: `http://www.cs.indiana.edu/hyplan/tkeahey/research/nlm/nlm.html`.

[3] `http://www.cs.virginia.edu/~gd2000/`

SeeNet3D system [CE95] presented two different visualization approaches. The first was a somewhat abstract 3D view of arcs lofted into the third dimension over a flat map, seen from an oblique viewpoint. The second layout approach, which showed links as arcs on a three-dimensional globe, inspired the similar visual encoding in the Planet Multicast system.

All the systems in the previous paragraph were highly literal, since each graph node had a geographic location attribute that was used for placement. Although the drawn links in some sense corresponded to physical cables in the real world, drawing them is an abstraction since those cables are not visible to the casual real-world observer.

The *fsn* system from Tesler and Strasnick of SGI [TS92] also places nodes on a ground plane, but has two major differences. First, the node locations are an abstract visual encoding of file system directory structure rather than inherent attributes of the data. Second, the geographic scale is that of a city rather than a country or the entire world, since the file size is encoded as the height of a building-like structure.[4] The MineSet system from SGI[5] includes an implementation of this algorithm. The Harmony system [And95] also used a similar visual metaphor.

Several systems for showing geographic traceroutes in both 2D and 3D [Too, Jon, Chr96, PN99, Aug98], have appeared either concurrently with or later than the Planet Multicast project, which was published in the fall of 1996. None of these newer systems have a more sophisticated interface from an information visualization point of view, and moreover many of them are more primitive.

It is worth noting that these systems discussed here fall into the realm of information visualization even though GIS (geographic information systems) and terrain rendering do not. In the latter two cases, no notion of an abstraction or a visual encoding choice spatializes data that is not inherently spatial.

2.2.2 Hierarchies

Strict hierarchies are a subset of general graphs, and aesthetic tree layout has been proved to be possible in polynomial time, making it a more tractable problem than general graph layout [SR83]. The literature on creating aesthetically pleasing 2D drawings of trees includes both rectilinear [WS79, RT81, Moe90] and radial [BETT99, pp. 52–55] methods, all of which are recursive. However, the example datasets are quite small.

A number of early Web visualization systems use either straightforward or previously presented tree layouts to show hyperlink structure [Döm94, AS95, PB94]. None of these approaches are suited for more than one hundred nodes.

[4] This system is known to many outside the field because of its appearance in the feature film *Jurassic Park*.

[5] `http://www.sgi.com/software/mineset`

Treemaps are a visualization method for hierarchies based on enclosure rather than connection [JS91]. Treemaps make it easy to spot outliers (for example, the few large files that are using up most of the space on a disk) as opposed to parent-child structure.

The Cheops system provides a highly compact interface to navigating very large hierarchies [BPV96]. They discuss an example hierarchy with a depth of 9 and a branching factor of 8 that can contain up to 20 million nodes. Cheops is compact to the point of being terse; it functions well as an index, but is not well-suited for browsing local areas or serving as the substrate for encoding auxiliary information in addition to the structure encoded in spatial layout.

Multitrees allow the depiction of several different link structures atop the same set of nodes [FZ94]. The H3 approach of distinguishing between links that belong to the spanning tree and non-tree links can be described as a 2-way multitree.

2.2.3 Distortion-Based Graph Drawing

Noik's taxonomy of distortion-based graph layouts [Noi94] summarizes the systems that appeared in the literature before 1994. Several systems have explored fisheye-style distortions, including Generalized Fisheye Views [Fur86], Graphical Fisheye Views and Rubber Sheets [SSTR93, SB94], and SemNet [FPF88]. In all instances the dataset size was quite limited, which remained the case in later systems such as the work of Kaugars [KRB94].

The SemNet system for visualizing semantic networks [FPF88] offered a choice of 3D layout algorithms, a few of which used deliberate distortion. SemNet was also one of only a few early systems to address navigation as well as layout. In addition to absolute and relative viewpoint positioning controls, the user could jump directly to a previously saved site, or navigate hop by hop through the graph structure. The SemNet visualization was bidirectionally linked to a knowledge management system, so that the knowledge base could be directly manipulated via interacting with the graph structure, and knowledge base operations could be reflected in the graph view.

The 3D Pliable Surface graph viewer [CCFS95] epitomizes the difference between the H3 algorithms and all of the previous work in distortion-based graph drawing: Carpendale uses a known algorithm from the GraphEd system [Him94] for layout, and uses distortion techniques only for navigation. In H3, both layout and navigation occur in the 3D hyperbolic space. The layout algorithm is carefully tuned for the characteristics of the distorted space, resulting in a more uniform information density.

The SHRiMP graph viewer is based on the multiscale Pad++ system [SWFM97, SM98]. The paper makes no explicit claims on scalability but the example datasets were limited to a few dozen nodes. One of the challenges of Pad-style multiscale views is the propensity for users to lose track of their whereabouts, a problem that is only partially addressed in recent work on multiscale navigation [JF98]. The earlier Continuous Zoom system allows multiple focal points [BHDH95], and has a similar multiscale feel and scalability limits.

The influential Cone Tree approach [RMC91] for recursive 3D tree layout is more similar to the radial 2D tree layouts than the rectilinear ones. The authors argue that Cone Trees fall into the Focus+Context domain, since the standard 3D Euclidean perspective projection emphasizes near objects at the expense of distant ones. There have been many extensions and refinements to Cone Trees, including the bottom-up algorithm of Carrière and Kazman that minimizes the chances of territory overlap [CK95]. Koike [KY93] presented the extension of fractal trees to visualize what he called “huge” hierarchies. His scalability claim is “hundreds or thousands” of nodes, and the example images show at least 1000 nodes.

The 2D Hyperbolic Tree browser from Lamping, Rao and Pirolli is one of the best known examples of a distortion-based layout for hierarchies [LR94, LRP95]. Their system was limited to strict hierarchies and used a two-dimensional layout algorithm. Section 3.4 contains a detailed analysis of the scalability and information density differences between this system and H3. Another difference between the two systems is covered in in Section 3.2: the PARC system uses the conformal projection from hyperbolic to Euclidean space, as opposed to the projective model used in H3.

2.2.4 Topological Force-Directed Systems

Force-directed layout [BETT99, Chapter 10] is a popular choice for general graph layout. One reason for its appeal is the intuitively clear analogy of a system with attracting springs along the links and magnetic-style repelling forces at the nodes. Nodes are seeded in an initial position and the system converges to an energy-minimizing state using methods such as gradient descent [KK89] or simulated annealing [DH96]. Although the graph drawing literature includes some quite sophisticated combinations of the best features of several previous algorithms [BHR95, Tun93], many straightforward force-directed implementations have appeared that do not seem to benefit from this previous work. Examples include the 3D Narcissus system for visualizing the hyperlink structure of the web [HDWB95], and some recent applets for browsing semantic and conceptual networks [Des, The98].

The Gem system [FLM94] has been recognized [BETT99, p. 323] as one of the most scalable 2D force-directed systems by virtue of handling datasets of more than 100 nodes. The Gem3D system [BF95] extends their algorithm to three dimensions and scales to a few hundred nodes. The Gem3D paper is one of the few in the Graph Drawing proceedings to explicitly discuss navigation separately from layout.

The literature on n-body simulation has a different but equivalent problem statement, and contains results which scale to large datasets where n is 10,000 by using a hierarchical $O(n \log n)$ algorithm instead of a naive $O(n^2)$ approach [App85].

Scalability is not the only problem with force-directed systems when considered from an information visualization perspective. The final visual appearance of a dataset is usually different on each invocation of the system, either because the initial node positions are random or because the minor tweaking of layout parameters results in major changes in the final layout. In contrast, systems that repeatably lay out a graph the same way on each invocation can help a user form a stable mental model of the graph structure, and are well-suited for incremental or online layouts.

2.2.5 Online and Incremental Layouts

A few systems try to address the problem of handling extremely large graphs by incrementally processing only a subset of the graph. These systems have an operational window of a fixed number of nodes, and can continually add nodes and vertices dynamically to the considered set, dropping older items when the window fills. However, these windows are extremely modest in size: both the DaTu [HE97] and OFDAV [CH97] systems have windows of only a few dozen nodes.

North's 1995 paper on an incremental layout which could handle graphs of modest size proposed online hierarchical layout for the exploration of large graphs as a fruitful area for possible future work [Nor95]. A incremental layout approach that features a resistive circuit distance model provides notable speedups, handling datasets of over one hundred nodes in a few seconds [CH97].

2.2.6 Other Approaches

Some 2D graph drawing systems attempt to highlight symmetry [LNS85], circular structure [TX94], or hierarchy [GKNV93]. The *dot* system [GKNV93] is one of the most popular 2D hierarchical systems, and scales to datasets of hundreds or even thousands of nodes. However, the noninteractive interface limits its applicability for very large datasets.

All attempts to use three dimensions have at least some form of interactivity. However, simply adding a third dimension does not solve scalability limits: the orthogonal Giotto3D system [GT96] uses computationally intensive display elements such as Bézier tubes, but the example figures feature less than two dozen nodes.

Nicheworks [Wil97, Wil99b] is the only graph drawing system to date besides our own H3 system that scales to large datasets. The Nicheworks papers are concurrent with the 1997-1998 H3 work. The layout algorithms easily scale to well over 100,000 nodes. (Although the system can theoretically handle one million nodes, layout would take one day or more.)[6] Nicheworks features interactive navigation and a divide-and-conquer layout approach, with a suite of 2D layout algorithms that separately handle each connected component. Both focus on scalability, but their presentation strategies are quite different: Nicheworks allows the user to zoom between an overview and a detail view, while H3 focuses on showing a very large neighborhood around a focus point. We discussed the tradeoffs of these two approaches earlier in this chapter, in Section 2.1.

2.3 Automatic Presentation Systems

Computer-based system for the automatic design of graphical presentations have been gradually expanding their coverage. Mackinlay's influential APT system used a data-oriented taxonomy and a palette of noninteractive 2D techniques including scatterplots, bar charts, and simple networks [Mac86a, Mac86b]. Casner's BOZ system introduced a task-based taxonomy [Cas91]. The Sage systems provided a slightly expanded palette of encoding techniques, but was still limited to two dimensions and no interactivity [RKMG94]. The ANDD system of Joe Marks brought automatic design to graph drawing [Mar91], but was limited to very small datasets of fewer than two dozen nodes.

None of the automated presentation systems exploit interactivity. Our three design studies on interactive navigation of large graphs can be thought of as forays into an unexplored part of the design space that could expand the vocabulary of future automatic presentation systems. A full characterization of this space will require both successful case studies and the kind of principled analysis of non-literal interaction techniques for which we argued in section 1.3.1.

[6] Personal communication, Graham Wills, April 2000.

Chapter 3

H3: 3D Hyperbolic Quasi-Hierarchical Graphs

The special-purpose MBone and Constellation visualization systems were developed in conjunction with users undertaking a particular focused task. In contrast, the H3 visualization system was not developed using a user-centered design process. It was instead intended as a somewhat more general solution to the problem of scalability in graph layout and drawing systems for an entire class of graphs. In Section 3.1 we define the class of quasi-hierarchical graphs and discuss the kinds of tasks that require viewing them.

Next, in section 3.2, we present a spatial layout algorithm for quasi-hierarchical graphs that uses a spanning tree constructed with the help of domain-specific information as a backbone. The H3 algorithm uses hyperbolic geometry for its computations, so we also present some basic background information about hyperbolic space and its relationship to general distortion-based techniques for visualizing information.

In section 3.3 we present the H3Viewer guaranteed frame rate drawing algorithm, which uses a combination of graph-theoretic and viewpoint information to ensure fluid interaction. Our visual encoding approach is documented in section 3.4, including an analysis of the information density of the H3 approach and the cognitive implications of using a radial layout. Implementation issues, including the desirability of linking the H3Viewer with other views of the dataset, are covered in section 3.5.

Section 3.6 covers interaction. The results of the project are discussed in section 3.7. We begin by documenting that the H3 system can quickly handle datasets more than one hundred times larger than most previous systems, then cover the project outcomes, which include incorporation into a commercial product. We then discuss several additional task domains where people have chosen to use the H3 system, ranging from simply creating data files in the H3 format for quick perusal to creating custom applications using our libraries. Finally, we present the results of a user study that found statistically significant benefits when using the H3Viewer for certain tasks.

3.1 Quasi-Hierarchical Graphs

Large graphs are extremely difficult to lay out in a way that helps people understand them. In the H3 project our main goal was to push the state of the art in dataset size far past previous limits. We achieved this scalability at the cost of sacrificing generality.

Most traditional taxonomies classify graphs based on purely graph-theoretic properties. For instance, determining whether a graph is acyclic or bipartite is completely independent of the domain of the data. We instead introduce a classification that depends on both graph structure and domain knowledge. We investigated a class of graphs that we call **quasi-hierarchical**, which can be effectively visualized using a spanning tree as the backbone of a layout algorithm. The visualization challenge is to clearly demarcate the situations in which a tree-based backbone drawing would enlighten rather than mislead the user.

A **spanning tree** is a connected acyclic subgraph that contains all the vertices of the original graph, but does not have to include all the links. A spanning tree (or a forest of spanning trees) can be computed for any graph. There are many well-known algorithms for building minimum spanning trees from general graphs, such as Kruskal's and Prim's [CLR93, Chapter 24]. The characterization of building a spanning tree that is most useful for our purposes is to cast it as a problem that must be solved at each node, by selecting which of the incoming links to a node would be the best one to use as the parent for that node in the spanning tree. A slightly more formal definition of a quasi-hierarchical graph is one for which there is a decision procedure for selecting the preferred parent link from among all the incoming links to a node. From a purely graph-theoretic point of view, the problem is simply finding a minimum-weight spanning tree through a graph with weighted edges, where domain-specific information is used to compute the weights.

Trees and directed acyclic graphs (DAGs) are trivially quasi-hierarchical, even without the use of domain knowledge. One might assume at first glance that only graphs that are quite sparse are quasi-hierarchical. For example, a Unix file system is nearly tree-like, since there are relatively few symbolic links. However, graphs that are considerably more dense than trees can also be quasi-hierarchical. The hyperlink structure of the web is a good example, and we discuss it further in the next section. The H3 algorithm can be used on graphs that are not quasi-hierarchical, by simply using a fallback algorithm such as breadth-first search to create a spanning tree, but the resulting visualization will probably not be useful.

Extremely dense graphs are unlikely to be quasi-hierarchical, although there may be some domains with counterexamples. The examples in this chapter range from $|E| = |V|$ in the case of trees to $|E| = 4|V|$ for the Autonomous Systems described in section 3.7.5. Although even the latter is sparse by the standard measurements of graph theory, since $|E| < |V|^2$, it is considerably more dense than graphs that have been traditionally regarded as well-suited for tree-based layout methods.

3.1.1 H3 Task Domains

After the core H3 algorithms were developed, we sought task domains where visualizing quasi-hierarchical graphs would be useful. The most important task consideration is whether drawing a graph is at all useful. Many problems could conceivably be cast in graph-theoretic terms, but a visualization is useful only for tasks tied to explicit human understanding of link structure of that graph. If that link structure is indeed relevant and, moreover, there is task domain information available to choose a good parent for each child node, then the H3Viewer is an appropriate visualization tool for the job.

3.1.1.1 External Representation Functions

An interactive visual representation of the node-link structure of a graph is helpful when it partially transfers cognitive load to an external representation. We have identified three quite different functions that an external representation can serve for a user:

- **Structure discovery:** The most basic task that justifies graph drawing is helping a user form a mental model of the structure of an unfamiliar graph. Usually this information is communicated by the spatial visual encoding.

- **Contextual backdrop:** Showing additional information in the context of the graph structure can lead to insights even if the graph structure is already familiar. The spatialization of the graph can be used as the backdrop against which other perceptual channels (e.g. retinal variables) such as color or size can encode additional data. The combination of all the perceptual channels can be more effective looking at any individual dataset outside of that shared context.

- **Linked index:** In many situations an interactive graph-browsing viewer is most useful when tightly integrated with other ways to view the same data. Linking multiple views with different spatial arrangements of the same dataset so that data highlighted in one is also highlighted in the others is called **brushing**, and has been advocated in several of the information visualization systems from Bell Labs [BC87]. If a graph viewer is one of several views which all support linked selection and filtering, the graph structure becomes one way to index the information. Such an index can be useful for selecting items in a known graph in addition to discovering patterns in an unfamiliar one. With multiple linked views, the user can switch between the displays as appropriate for the task at hand.

An example domain where all three of these functions are useful is the creation and maintenance of a large web site, a task where understanding the hyperlink structure of the site is important.

The structure discovery function is relevant because the hyperlink structure of a site has a direct influence on the number of visitors to particular pages: the more clicks it takes a visitor to reach a page from the entry point, the less likely it is to be reached [Nie00]. Site designers usually want to tune sites so that important or popular pages are a minimal number of hops from the root instead of being relegated to obscurity by an overly deep hyperlink structure. The task of organizing a site so that information is categorized in a meaningful and easily traversable way is difficult, in part because it requires keeping track of the relationship of many individual pages and aggregate groups of pages. The size of most nontrivial web sites makes this hard to do without an external representation. For instance, the medium-sized academic web site of the Stanford graphics group has 70,000 pages.

An example of a contextual backdrop is showing traffic log information in the context of the hyperlink structure of the site (see Figure 3.17 in section 3.5.1). A simple ordered list of the most popular pages on a site can be computed from a site's traffic logs and presented to the designer without any need for visualization. However, that list alone is only the beginning of the job. The traffic logs record which outgoing hyperlink is chosen by a visitor to a web page. The designer can often make more informed decisions about site organization if the actual choices made by the visitors are shown in the context of the full range of possibilities – that is, if the traffic information is presented in relation to the hyperlink structure of the site instead of as a simple list. The graphical representation of the link structure of the site is helpful because it transfers some of the cognitive burden to an external representation, instead of forcing the site designer to reconstruct that structure mentally.

Linking a visualization window to other views gives the webmaster much more flexibility and power than would be possible a single standalone browser. For example, clicking on the node that represents a document in the visualization window would cause it to be rendered in a linked HTML viewer, or typing a search query in another window could trigger an animated transition in the visualization window that ends with the desired node framed in the view. An example is shown in Figure 3.15.

3.1.1.2 Hyperlink Structure for Webmasters

A web site is an example of a quasi-hierarchical graph where nodes represent web documents and the links represent hyperlinks between those documents. Even though web sites are highly interconnected, Web site

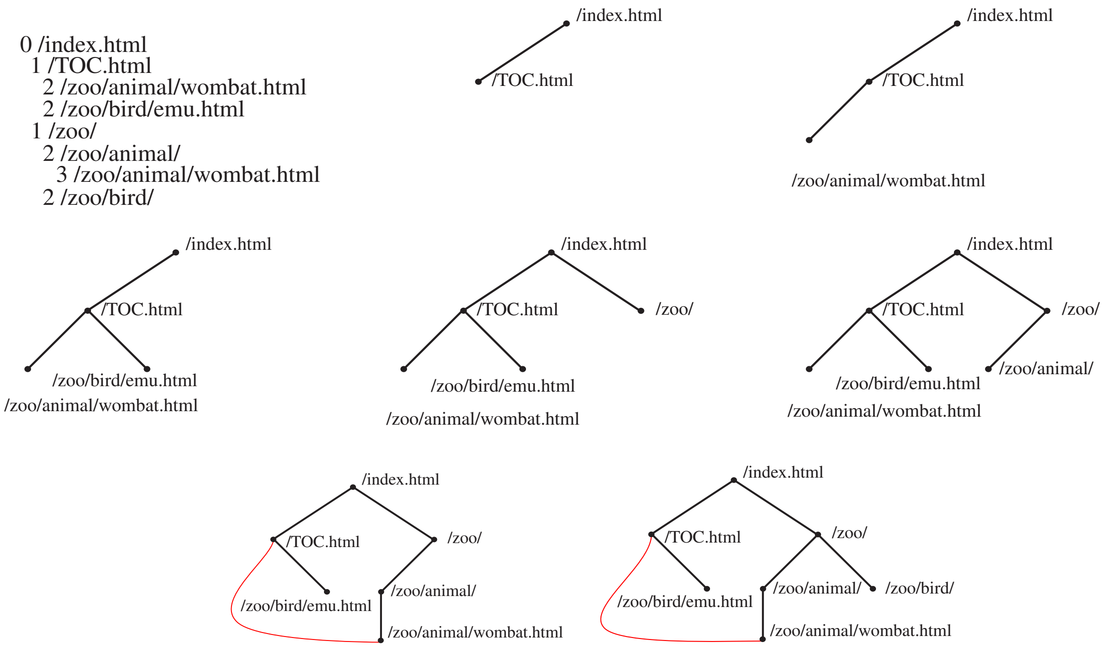

Figure 3.1: **Constructing a spanning tree for quasi-hierarchical web site. Top Left:** The hyperlink structure of a simple hypothetical site, as it would be reported by a web spider starting at the top page. Nodes represent web pages, and links represent hyperlinks. Although the graph structure itself is determined by hyperlinks, additional information about hierarchical directory structure of the site's files is encoded in the URLs. **Top Row:** We build up the graph incrementally, one link at a time. **Middle Row:** We continue adding nodes without moving any of the old ones around. **Bottom Row:** When the `animal/wombat.html` page is added, the label matching test shows that `animal` is a more appropriate parent than `/TOC.html`, so the node moves and the link between `animal/wombat.html` and `/TOC.html` becomes a non-tree link. In the final stage, note that `bird/emu.html` does not move when the `bird` is added, even though the labels match, because there is no hyperlink between them.

designers usually have a clear notion of hierarchy within the site: "Pages within sites are typically organized in hierarchical, tree-like, fashion" [HA99, p. 2]. A common mental model for a designer is that most content pages conceptually have a single main parent even though they might have multiple incoming links from related lateral pages. We rely on the pattern that server directories usually have a default document (often called `index.html`) that is a natural parent for every other file in that directory, and that the location of subdirectories is a deliberate choice reflecting the organization of the site content. This heuristic that the directory structure of a web site reflects authorial intent on the part of the site designer is often, but not always, true.

Information about directory structure of a site is easy to obtain, since it is encoded in the URL of the page. We can use this directory information to resolve parentage within the hyperlink graph structure by giving preference to URLs that have as many matching characters as possible from the root on the left towards the right. Figure 3.1.1.2 shows an illustrative example of how a spanning tree would be incrementally formed by parsing a linear file that contains a depth-first traversal of the graph. Nodes are initially directly attached to the first incoming link, but can be moved when a more appropriate match is found. In this case, the label matching has top priority, as shown in the figure. If none of the incoming nodes is a clear choice given the directory structure information, we fall back on a preference for nodes closer to the root so that the resulting spanning tree is closer to breadth-first than depth-first.

When the directory structure of the site does connote authorial intent, then our heuristic is usually a much closer match to the user's mental model than a simple breadth-first layout. For instance, many sites have a table of contents page one hop from the entrance page. Since that page contains a link to every other document at the site, it would end up being the parent of almost every document on the site. The resulting degenerate spanning tree would not encode any interesting structural information.

Although we use the directory structure as a heuristic for making decisions about the spanning tree, the underlying graph that we are laying out is the hyperlink structure of the tree. We are not simply laying out the file system structure of the site: although the nodes are the same in both cases, the links are different.

3.1.1.3 Function Call Graph Structure for Compiler Users

One task faced by the SUIF compiler research group at Stanford was optimizing performance by detecting which parts of a program can be parallelized [HAA+96]. Their methods worked fully automatically up to a point, but they then relied on user feedback to achieve full optimization. The SUIF optimizing compiler pinpointed brief sections of code that were shown to the user. If the user could assert that certain conditions were met using high-level knowledge not available to the compiler, then more of the program was successfully parallelized.

The difficulty faced by the user was understanding how a brief section of code (generally a few lines inside a tight loop) fit into the bigger picture of the entire program. Although an original author might be well aware of the interdependencies, software engineers who were modifying unfamiliar code or authors who had long since forgotten the details of their own code were faced with a challenge. The SUIF developers hoped that an interactive graphical representation of the function call graph would help SUIF users understand a complex program's structure more quickly. Such graphs are notoriously difficult to understand when all the links appear in a 2D nonplanarized layout, which is unfortunately the usual approach even in state-of-the-art performance analysis tools.

In the SUIF graph, nodes represent functions and links represent a call from one function to another. This

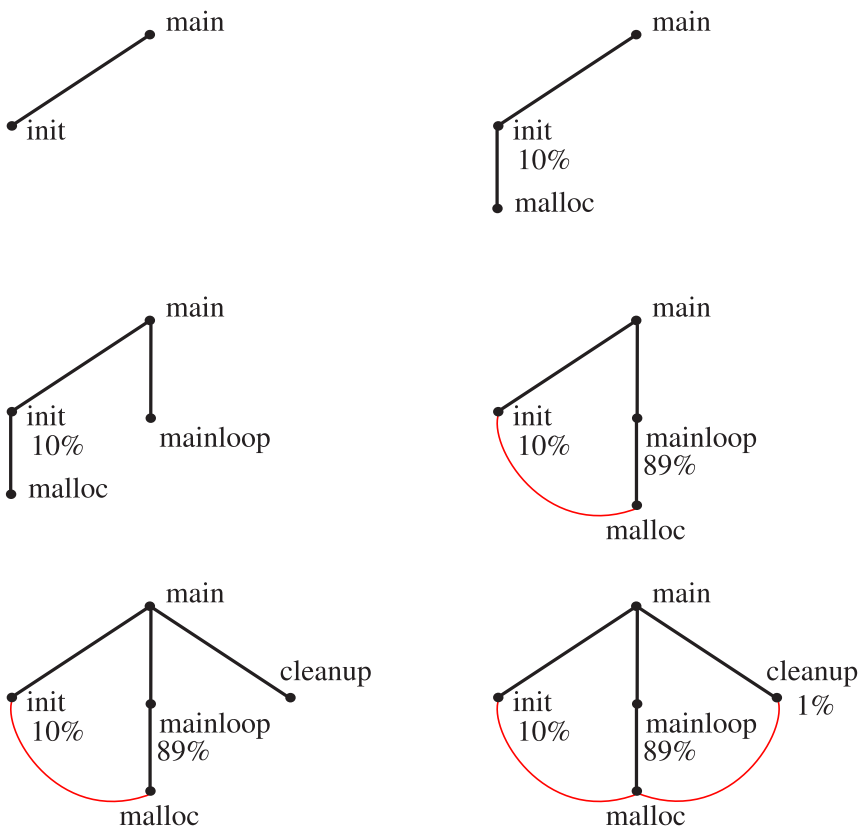

Figure 3.2: **Constructing a spanning tree for quasi-hierarchical function call graph.** In this simple hypothetical function call graph, nodes represent functions, and links represent calls from one function to another. The call graph is computed by a static analysis of the program text. The spanning tree is determined by run-time profiling of the code so that the calling procedure that is responsible for the most execution time in the called procedure is the parent. We show the layout incrementally as in Figure 3.1.1.2: the parent of a node can change when new information about a more appropriate candidate emerges, and the small multiples should be read row by row starting at the top left.

base graph is known as a static function call graph, where the existence of a call is determined by parsing the program code. The SUIF researchers decided that using execution time measurements to create a spanning tree would result in layouts of these quasi-hierarchical graphs that closely matched their mental model of the structures. They used their existing tools for extracting run-time dynamic profile information to recover the percentage breakdown of the time attributable to each calling procedure out of all the time spent executing the called function. The calling procedure responsible for the majority of a child's execution time was used as the main parent in the spanning tree. Figure 3.2 shows a simple example. (Figures 3.14 and 3.21 show call graphs displayed in our visualization system.)

3.2 Spatial Layout

The previous section covers the motivation for using an appropriate spanning tree as the backbone for the spatial layout of a quasi-hierarchical graph. In this section we discuss how to determine a position in space for each tree node. We first cover the idea of using nonlinear distortion for information visualization, then explain the utility of hyperbolic geometry for distortion-based layout and navigation. We end with the specifics of the H3 hyperbolic tree layout algorithm.

3.2.1 Nonlinear Distortion

The classic problem with tree layout in Euclidean space is that the number of child nodes to place grows exponentially at each level of the tree, but the available room in which to place them grows only polynomially. Specifically, the circumference of a circle or the area of a sphere increases as a polynomial function of its radius. The usual approach to avoiding collisions is to allocate less room to nodes deeper in the tree than to ones near the root. The disadvantage of this strategy is that only a small local neighborhood of any node can be examined at once because of the size differential between root and leaf nodes. Zooming back to see an overview of the entire tree results in leaf nodes too small to see. Examining nodes deep in the tree requires zooming in so far that surrounding context is lost.

There is a whole class of distortion-based visualization methods that strive to show detail within as much surrounding context as possible in a given amount of screen area, which we discuss in section 2.1. In the H3 system we achieve an elegant single-focus distortion effect by using hyperbolic geometry for both layout and navigation.

3.2.2 Hyperbolic Geometry

Hyperbolic geometry is one of the non-Euclidean geometries developed at the turn of the century. We use it for two reasons: first, there is an elegant way to draw a Focus+Context view using a known projection that maps the entire infinite space into a finite drawing region. Second, we can allocate the same amount of room for each of the nodes in a tree while still avoiding collisions because there is an exponential amount of room available in hyperbolic space.

Hyperbolic and spherical geometry are the only two non-Euclidean geometries that are homogeneous and have isotropic distance metrics: in other words, there is a uniform, meaningful, and continuous concept of the distance between two points.[1] These geometries are internally consistent despite the lack of Euclid's parallel

[1] There are five additional three-dimensional geometries that are homogeneous but nonisotropic, which together with the three isotropic geometries (spherical, Euclidean, and hyperbolic) form Thurston's Eight Model Geometries [Thu97, Section 3.8] [Wee85,

postulate. In the spherical case there are no parallel lines: all great circles intersect each other. (The Planet Multicast system described in Section 4.2 uses great circles, which are geodesics on the surface on a 2D sphere, as part of the geographic layout computation.) In the hyperbolic case there are many lines through a point that are parallel to another line. The ramifications of this property are extensive, and it is beyond the scope of this thesis to discuss their full implications. The following material on projection and exponential room is only a brief introduction to the surprising and fascinating mathematics of hyperbolic space, which is covered in far more detail in many mathematical textbooks [Mar75, Wol45, Mes64, Cox42, Coo09, Shi63, Ive92, Fen89, Kul61].

3.2.2.1 Exponential Room

In hyperbolic space, circumference and area increase exponentially with respect to radius. There is literally more room in hyperbolic space than in Euclidean space, where these measures increase polynomially.[2]

The circumference of a hyperbolic circle increases exponentially with respect to its radius; the equation is $2\pi \sinh r$, as opposed to the Euclidean equation $2\pi r$. Figure 3.3 shows a picture of the 2D hyperbolic plane embedded into 3D Euclidean space, intended to give an intuitive sense of how much more room there is on a hyperbolic plane than on a Euclidean plane. Most pictures of the 2D hyperbolic plane, for instance Escher-style tilings [Sch92, DLW81], show one of the traditional projective or conformal views where projected features appear smaller on the periphery. This picture instead shows a piece of the 2D hyperbolic plane where true sizes are not distorted, so the only way to display it in 3D Euclidean space is with overlapping folds.

Considering the distinction between parallel and equidistant lines in hyperbolic space can provide a more focused mathematical intuition for why this holds true. Two parallel straight lines (geodesics) are always the same distance apart in Euclidean space, but in hyperbolic space most parallel straight lines are not equidistant. The two hyperbolic geodesic lines in Figure 3.4 are parallel because they do not intersect, but they are clearly diverging.

3.2.2.2 Projection

Hyperbolic space, like Euclidean space, is infinite in extent. However, it is possible to map the entire infinite space into a finite portion of Euclidean space. That finite projection corresponds to a Euclidean camera taking a picture of an entire hyperbolic universe from the "outside", providing a Focus+Context view. Any projection from hyperbolic to Euclidean space will necessarily involve some distortion, since the distance metrics are

Chapter 18].

[2] An equivalent statement is that hyperbolic objects have negative Gaussian curvature.

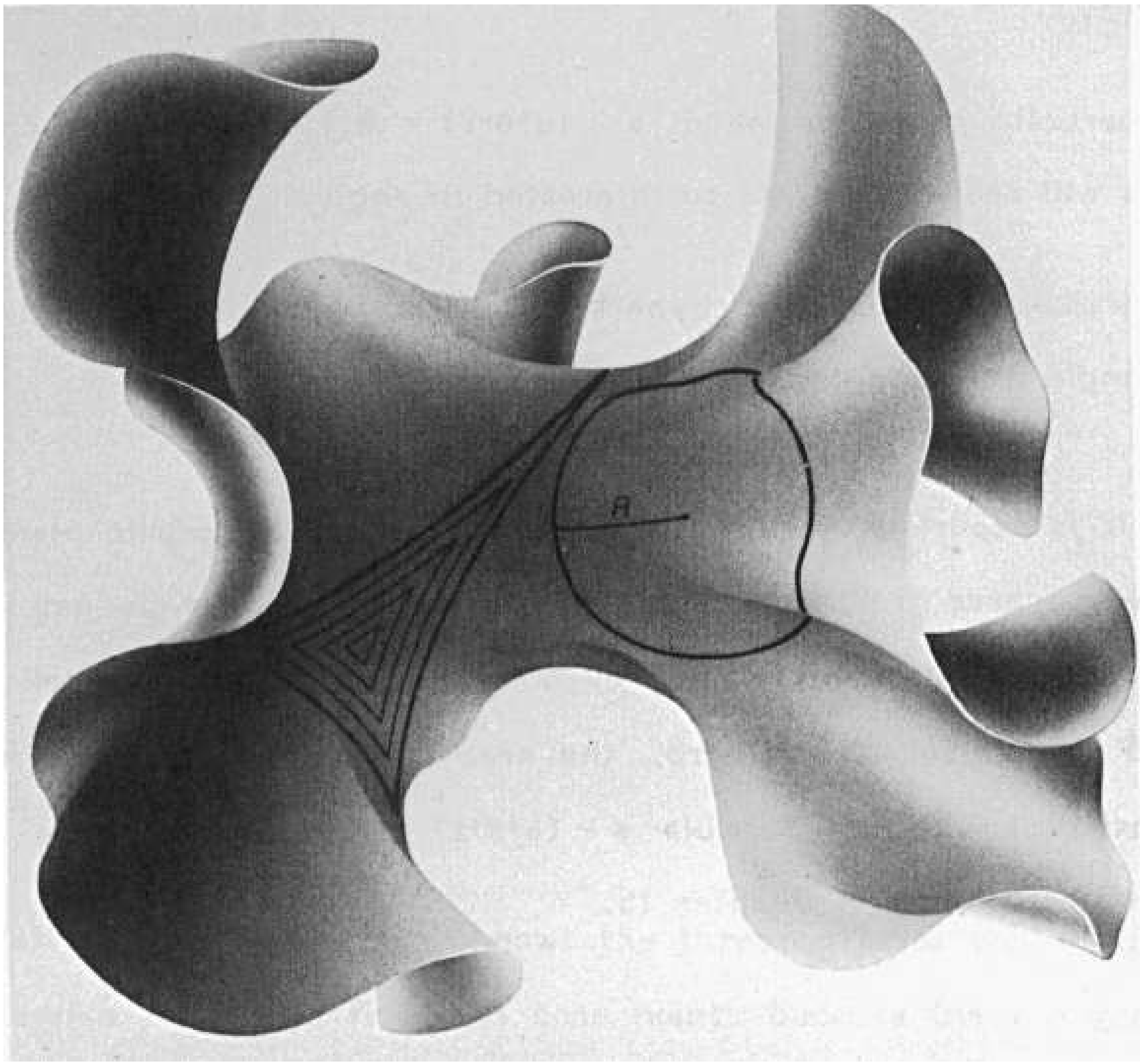

Figure 3.3: **Exponential volume of hyperbolic space.** There is literally more room in hyperbolic space than in Euclidean space, as shown in this embedding of the hyperbolic plane into Euclidean 3-space. A circle on a hyperbolic surface has an exponentially increasing circumference, as opposed to the familiar Euclidean situation where circle circumference grows quadratically with respect to its radius. Image from [TW84] used courtesy of Ian Warpole.

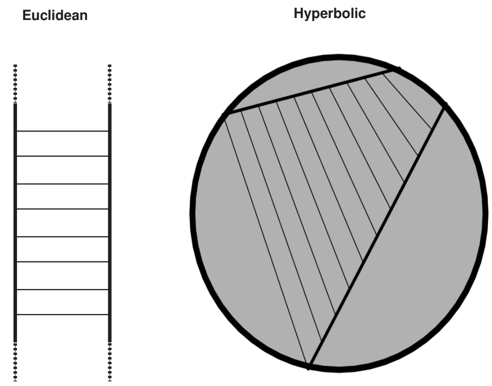

Figure 3.4: **Euclidean equidistant parallels vs. hyperbolic divergent parallels. Left:** Parallel lines in Euclidean space are always the same distance apart. **Right:** In hyperbolic space the distance between two lines that never meet does indeed change. Here we show two geodesics that are not equidistant, but extend to infinity without ever intersecting. Recall that the projective model is an open disc, so the boundary is not part of the space.

different. In our case we want to choose the distortion most advantageous for information visualization. Since our goal is a Focus+Context view, we ruled out an "insider" projection such as the one used in the pedagogical video Not Knot [GM91, Gun92, PG92]. We want to use a virtual camera that is not constrained by the same rules of hyperbolic motion as the geometric data, so that important information never falls outside of its field of view.

There are several standard projections from hyperbolic to Euclidean space, called **models** in the mathematical literature. The two that have finite projections are the **projective model** (sometimes known as Klein, or Klein-Beltrami) and the **conformal model** (sometimes known as the Poincaré disc or ball). There is a known mapping between these two models, and also between these and the models that have infinite projections, such as the upper half space or Minkowski models. The conformal model preserves Euclidean angle measurements but maps straight lines into arcs. The projective model preserves straight lines but distorts angles. Figure 3.5 shows the different visual appearance of a graph projected projectively and conformally using the *webviz* system we previously developed at the Geometry Center, which supported both models.

The conformal model is sometimes considered more aesthetically pleasing, at least in two dimensions: the famous Escher pictures of 2D hyperbolic tilings [Sch92, DLW81] are conformal, as is the 2D hyperbolic tree browser developed at Xerox PARC [LRP95]. However, in the 3D case the projective model is much faster for layout and drawing using standard graphics packages.

Most 3D graphics libraries have a highly optimized 4×4 matrix pipeline, often implemented in hardware, because they use homogeneous coordinates to efficiently describe transformations of Euclidean 3-space. We projects to a straight line in the disc at $z = 0$. Hyperbolic translation amounts to sliding 2D objects around on the hyperbola itself. Again, the size of the projected objects depends on their proximity to the pole, which projects to the origin of the disc.

The 3D case is hard to show in a diagram, since it involves a 3-hyperboloid projected into a ball – a finite section of Euclidean 3-space. The pole of the hyperboloid projects to the origin of the ball. In the previous figures we chose to diagram objects of dimension n embedded in a space of dimension $n+1$ only for clarity of exposition.[3] Although Figure 3.7 shows the 2-hyperboloid embedded in 3D space, projecting from it to the disc is a purely two-dimensional operation. Likewise, although drawing a diagram for the 3D case would require embedding these 3D objects in 4-space[4], the actual mathematics involved is purely three-dimensional.[5]

With a single still image, a projection from hyperbolic space looks similar to a Euclidean scene projected through a fisheye lens. The projection to hyperbolic space serves the purpose of a Degree of Interest function as required by Furnas [Fur86]. However, motion of an object constructed with hyperbolic geometry is qualitatively different from the motion of a Euclidean object. Although we could simply place Euclidean objects into hyperbolic 3-space and move them around according to the rules of hyperbolic geometry, we would not be exploiting the exponential amount of room available in hyperbolic space.

3.2.3 Tree Layout

The exponential amount of room in hyperbolic space is an elegant "impedance match" for a tree layout, since the number of nodes in a tree grows exponentially with its depth. To the best of our knowledge, the idea of using hyperbolic space for tree layout was first informally proposed by Thurston at least as early as the 1980's. Visualization systems for doing so were created independently and nearly simultaneously by both Lamping, Rao, and Pirolli at Xerox PARC [LR94, LRP95], and pre-dissertation work by myself and Paul Burchard at the Geometry Center [MB95].

In Euclidean space, trees can be laid out compactly only if the amount of space devoted to each node grows smaller towards the leaves. When inspecting detail at the leaf level, the structure near the root is difficult to understand because of the major differences in scale. Nodes can be allocated equal size only for a layout that is very sparse, which makes comparisons between levels equally difficult.

[3] Technically, a hyperboloid embedded in a space one dimension higher than itself is the Minkowski model, and our projection is the standard mapping from the Minkowski to the Klein-Beltrami model.

[4] The word "hyperbolic" should not be confused with the word "hyperspace", which is sometimes used in popular literature to mean 4-space.

[5] *The Shape of Space*, by Jeff Weeks, contains an eminently readable exposition of dimensionality and embedding [Wee85].

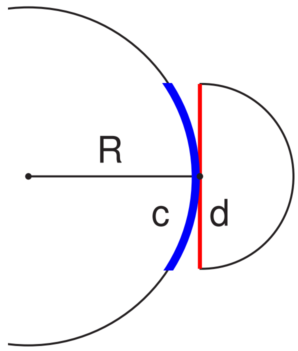

Figure 3.9: **Discs vs. spherical caps.** The area of a disc underneath a child hemisphere is an approximation for the area of the underlying spherical cap on the parent. This figure shows the 2D case, where the diameter d underneath the half-circle approximates the arc c. The approximation, which works well when there are many children, is necessary because the spherical cap area is not computable without the parental radius R.

cone body proper no longer takes up space but flattens out into a disc at the base of the hemisphere. The child hemispheres lie directly on the parental hemisphere's tangent plane, with no visible intervening cone body. Our hemispherical layout is both possible and effective because of the exponential amount of room in hyperbolic space. Specifically, the surface area of a hemisphere, $2\pi r^2$, increases polynomially with respect to its radius in Euclidean space. In hyperbolic space the formula for hemisphere area is $2\pi \sinh^2(r)$, and the hyperbolic sine and cosine functions ($\sinh$ and $\cosh$) are exponential.

The layout algorithm requires two passes: a bottom-up pass to estimate the radius needed for each hemisphere to accommodate all its children, and a top-down pass to place each child node on its parental hemisphere's surface. These steps cannot be combined because we need the radius of the parental hemisphere before we can compute the final position of the children.

3.2.3.1 Bottom-up Estimation Pass

We draw leaf nodes as tetrahedra of a fixed hyperbolic size; therefore, we trivially know the value of the radius of child hemispheres at the leaves. At each level we need to determine how large of a hemisphere to allocate for a parent hemisphere given the radii of all the child hemispheres. We do this by summing the area of the disc at the bottom of each child hemisphere.

The flat disc at the bottom of a child hemisphere is only an approximation of the area that will be used when the child is placed on the parental hemisphere. Figure 3.9 shows that the correct area would be the spherical cap that lies underneath the child hemisphere when it covers part of the parental hemisphere, as opposed to a flat disc on the tangent plane. However, computing the area of that spherical cap requires

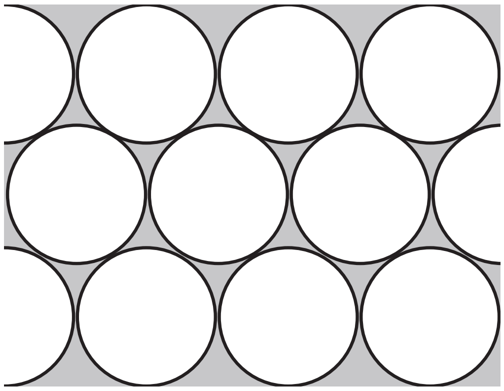

Figure 3.10: **Circle packing.** Packing uniformly sized circles on a plane is one of the most common circle packing cases. Note that even in this optimal packing of circles on the plane, the area of the circles sums to less than the area of the underlying plane, because of the grey interstitial gaps.

knowledge about the radius of the parental hemisphere, which we do not know since that is exactly what we are trying to discover. The area of the flat disc is a reasonable approximation when the child hemisphere subtends a small angle; that is, when a parent has many child nodes. The approximation breaks down when the number of children is small, but the special cases for properly handling a small number of children (1, 2, or 3 through 5) are easy to handle in the code.

3.2.3.2 Top-down Placement Pass

The bottom-up estimation pass starts at the leaves and ends with an estimate of the radius of the root hemisphere. We then use these radius estimates in the top-down pass to compute the actual point in space so as to place child hemispheres on the surface of their parent.

The problem of placing circles contiguously without overlaps has received extensive attention from mathematicians, under the name **circle packing** or **sphere packing** [CS88]. A related problem is that of distributing points evenly on a sphere [SK97]. Our particular instance is that of packing circles (1-spheres) on the surface of an ordinary sphere (2-sphere).

3.2.3.3 Sphere Packing

The particular requirements of our situation are somewhat different than the usual cases addressed in the literature, such as packing circles on a 2D plane as in Figure 3.10. Our circles are of variable size, and we are interested in a hemisphere as opposed to a sphere. More importantly, our solution must be fast and repeatable. Our solution cannot involve randomness: given the same input, we must generate the same output. We care

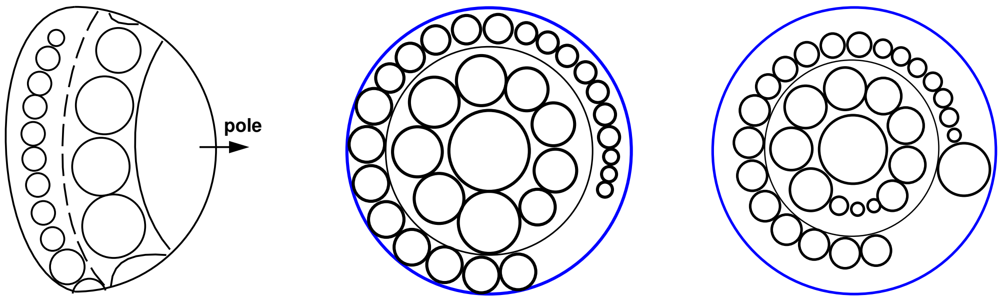

Figure 3.11: **Band layout. Left:** Discs at the bottom of child hemispheres are laid out in bands on the parent in sorted order according to the number of their descendants, with the most prolific at the pole. **Middle:** Sorting the children by total progeny results in a list of hemispheres sorted by radius size. **Right:** Unsorted hemispheres potentially result in a large amount of wasted space within the bands.

far more about speed than precision. An approximate layout is fine for our purposes, whereas a perfect but slow iterative solution would be inappropriate.

Our solution is to lay out the discs in concentric bands centered along the pole normal to the sphere at infinity. The left side of Figure 3.11 shows a view of a parent hemisphere from the side, with only the footprints of the child hemispheres drawn. We sort the child discs by the size of their hemispheres. This number, which is recursively calculated in the first bottom-up pass, depends on the total number of their descendants, not just their first-generation children. The ones that require the most area (i.e. the ones with the most progeny) are closest to this pole. The middle and right of Figure 3.11 compare the amount of space used by this sorted packing with the amount potentially wasted by an unordered packing. The equatorial (bottom-most) band is usually only partially complete.

Even if our circle packing were optimal, the area of the hemisphere required to accommodate the circles would be less than than the sum of the spherical caps subtended by those circles, since we do not take into account the uncovered gaps between the circles (as shown in Figure 3.9). Moreover, our banding scheme is easy to implement but is far from an optimal circle packing. We waste the leftover space in the equatorial band, which in the worst case contains only a single disc. If we did not sort the child discs by size the discrepancy would be even greater, in the worst cases by a factor of 2.

In summary, our estimated target surface area is only an approximation for three reasons. First, the surface area of a spherical cap is greater than the area of a flat disc on a tangent plane to the sphere. Second, even an optimal circle packing leaves uncovered gaps between the discs. Third, our circle packing is known to be suboptimal: circles may use less vertical space than their allotted band allows, and the packing may allow unused horizontal space in the outermost band.

All three reasons lead to an estimate that is less than the necessary area. We use an empirically derived area-scaling factor to increase the radius by an amount proportional to the estimate. In rare cases our initial adjusted estimate of the hemisphere area falls short, when at the end of the placement pass for a hemisphere we find that the incrementally placed child discs extend beyond the allocated hemisphere into the other half of the sphere. We then run a second iteration for placement using the exact size needed for the parental hemisphere radius, which is now directly computable.

The radius estimate scaling factor is one of the two parameters that controls the density of the layout. The other parameter in our implementation is the surface area allotted for hemispheres of leaf nodes. The layout would be too dense if the leaf nodes touch, so we generally specify a larger area than a hemisphere that would exactly fit around the geometric representation of a node.

3.2.3.4 Derivation

The H3 layout method operates in two passes. In the bottom-up pass we find an approximate radius for each hemisphere, and in the top-down pass we place children on the surface of their parent hemisphere. Here we present a detailed derivation of the radius r_p of a parental hemisphere and the spherical coordinate triple (r_p, ϕ, θ) needed to place a child hemisphere on the surface of that parental hemisphere. For each step of the derivation, we show first the Euclidean then the hyperbolic result. The Euclidean and hyperbolic formulas used are collected in Table 3.1.

Formula	Euclidean	Hyperbolic
right-angle triangle	$\tan\theta = \frac{opp}{adj}$	$\tan\theta = \frac{\tanh(opp)}{\sinh(adj)}$
right-angle triangle	$\sin\theta = \frac{opp}{hyp}$	$\sin\theta = \frac{\sinh(opp)}{\sinh(hyp)}$
circle area	πr^2	$2\pi(\cosh(r) - 1)$
hemisphere area	$2\pi r^2$	$2\pi \sinh^2(r)$
spherical cap area[6]	$2\pi r^2(1 - \cos\phi)$	$2\pi \sinh^2 r(1 - \cos\phi)$

Table 3.1: **Euclidean and hyperbolic formulas.**

Bottom-up estimation pass We compute a target surface area H_p of a hemisphere at level p by summing the areas of the discs at the bottom of the child hemispheres at level $p+1$. The Euclidean derivation is straightforward. The Euclidean hemisphere area formula is $H_p = 2\pi(r_p)^2$, so the Euclidean radius r_p would be $\sqrt{H_p/2\pi}$. The area of a Euclidean circle is πr^2, so the relationship between the parent and child

[6]The typographical error of having r instead of r^2 in both the Euclidean and hyperbolic equations in [Mun97] is corrected here.

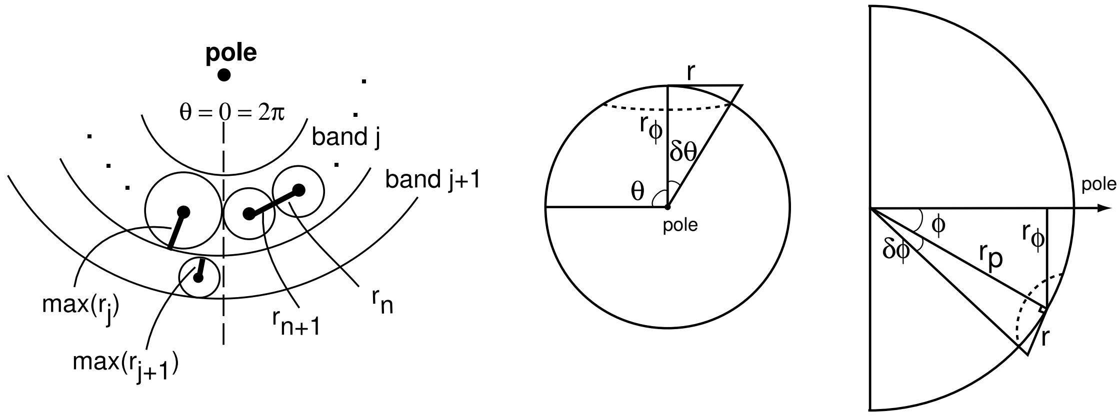

Figure 3.12: **Child hemisphere placement. Left:** Within a band, we increment θ to move from the center of one child with radius r_n to the center of the next with radius r_{n+1}. When band j is filled at $\theta = 2\pi$, we increment ϕ to move to a new band $j+1$ further away from the pole. The sorting guarantees that first child placed has maximum radius in the band. **Middle:** View from above the pole shows the trigonometry needed to find the incremental $\delta\theta$ to move within a band. r_ϕ is the radius of a spherical cap at ϕ. **Right:** View from the side shows finding the incremental $\delta\phi$ to move between bands. The radius of the parent hemisphere is r_p.

hemispheres is

$$H_p = \sum_{k=1}^{n} D_k = \sum_{k=1}^{n} \pi (r_k)^2,$$

where D_k is the area of a disc at the bottom of a child hemisphere at level $p+1$ and n is the number of children at that level. When we use the hyperbolic hemisphere area formula $H_p = 2\pi \sinh^2(r_p)$, the hyperbolic radius of the parental hemisphere is $r_p = \sinh^{-1}\sqrt{\frac{H_p}{2\pi}}$. The hyperbolic circle area formula is considerably different from the Euclidean case: $2\pi(\cosh(r) - 1)$. The parent-child relationship becomes

$$H_p = \sum_{k=1}^{n} D_k = \sum_{k=1}^{n} 2\pi(\cosh(r_k) - 1).$$

Top-down placement pass We use the radius r_p of the parent hemisphere to compute the remaining two spherical coordinates ϕ and θ, since the triple (r_p, ϕ, θ) completely specifies the position of the child hemisphere at level $p+1$. We compute ϕ and θ cumulatively, starting from the pole at the top of the hemisphere. The children are laid out in bands that are concentric around the pole. The band at the pole itself is the degenerate case of a spherical cap, therefore no value of θ needs to be computed. The total incremental angle $\delta\theta_{incr}$ between child n and child $n+1$ on a typical band j is the sum of the angles $\delta\theta_n$ and $\delta\theta_{n+1}$, which depend on the radii r_n and r_{n+1} of the children, as shown in Figure 3.12 on the left. We thus need to derive the angle $\delta\theta$

given some r, as in the middle of Figure 3.12. That angle depends on r_ϕ, the radius of the spherical cap at ϕ. We can use a Euclidean right triangle formula to find the radius $r_\phi = r_p \sin\phi$, and with the other Euclidean right-angle triangle formula $\tan\delta\theta = \frac{r_n}{r_\phi}$ we find the Euclidean angle

$$\delta\theta = \arctan\left(\frac{r}{r_p \sin\phi}\right).$$

If we instead use the hyperbolic right triangle formula to find the radius $r_\phi = \sinh^{-1}(\sinh r_p \sin\phi)$ and the other hyperbolic right-angle triangle formula $\tan\delta\theta = \frac{tanh(r_n)}{sinh(r_\phi)}$, we find the hyperbolic angle

$$\delta\theta = \arctan\left(\frac{\tanh r}{\sinh r_p \sin\phi}\right).$$

We substitute r_n and r_{n+1} to obtain our total incremental angle for placing a circle within a band:

$$\delta\theta_{incr} = \arctan\left(\frac{\tanh(r_n)}{\sinh r_p \sin\phi}\right) + \arctan\left(\frac{\tanh(r_{n+1})}{\sinh r_p \sin\phi_j}\right).$$

If the total cumulative angle $\theta + \delta\theta_{n+1}$ is greater than 2π, we drop down to the next band $j+1$ and reset θ_{n+1} to 0. The $\delta\phi$ angle between a child on one band and a child on the next band depends on the radius r_j of the largest child in band j and the radius r_{j+1} of the largest child in band $j+1$. In our current circle packing approach, we know that the first child in each band will have the largest radius, since we lay out the children in descending sorted order. We thus need to derive the angle $\delta\phi$ given some r, as in the right of Figure 3.12. The Euclidean angle ϕ corresponding to r is simply $\arctan(r/r_p)$ because of the right angle formula. We substitute the hyperbolic formula for right-angle triangles to find

$$\delta\phi = \arctan\left(\frac{\tanh r}{\sinh r_p}\right).$$

We substitute r_j and r_{j+1} to obtain our total incremental angle for placing a circle on the next band:

$$\delta\phi_{incr} = \arctan\left(\frac{\tanh r_j}{\sinh r_p}\right) + \arctan\left(\frac{\tanh r_{j+1}}{\sinh r_p}\right).$$

Armed with the triple (r, ϕ, θ), we can center the child hemisphere in the appropriate spot on the parent hemisphere. We discuss the tradeoffs of our layout technique further in section 6.1.3 (page 126).

3.2.3.5 Single-pass Recursive Algorithm

Although we have discussed our algorithm in terms of a bottom-up pass followed by a top-down pass for expository purposes, the necessary work can be done with single recursive function. The recursion starts with the root node, and the base cases are the leaf nodes of known radius. The function begins by looping through the children and calling the layout function recursively on each. Popping the stack after the recursive call yields the child radius, which we can use to compute the parental hemisphere radius estimate, and then immediately place the child discs on the surface of the parent.

3.3 Drawing

Our adaptive drawing algorithm is linear in the number of *visible* nodes and edges. The projection from hyperbolic to Euclidean space guarantees that nodes sufficiently far from the center will project to less than a single pixel. Although there are a potentially unlimited total number of nodes and edges, our algorithm terminates when features project to subpixel areas. Thus the visual and computational complexity of the scene has guaranteed bound – only a local neighborhood of nodes in the graph will be drawn.

3.3.1 Adaptive Drawing

A guaranteed frame rate is extremely important for a fluid interactive user experience. The H3Viewer adaptive drawing algorithm is designed to always maintain a target frame rate even on low-end graphics systems. A high frame rate is maintained by drawing only as much of the neighborhood around a center point as is possible in the allotted time.

3.3.1.1 Candidate Nodes for Drawing

The drawing algorithm incorporates knowledge of both the graph structure and the current viewing position. We use the link structure of the spanning tree to guide additions to a pool of candidate nodes and the projected screen area of the nodes to sort the candidates.

The link structure of the spanning tree provides us with a graph-theoretic measure of the distance between two nodes: the number of **hops** from one node to another is the integer number of links between them. When we draw a node, reasonable candidates for drawing next are the nodes one hop away, namely its parent and children in the spanning tree. Nodes that are a short number of hops from the center will usually be more visible than those that are a large number of hops away. However, if we simply rely on graph structure, we may waste time filling in sections that are less visible while neglecting those that are more prominent.

of those other views is the current focus of the user's attention. In the usual case, the algorithm halts on its own because all undrawn nodes will be smaller than one pixel in the hyperbolic projection.

Figure 3.13 also shows that we deliberately draw all links into and out of a node, even if we do not have time to draw the node at the other end. The presence of an unterminated link during motion hints to the user of something interesting in that direction. This situation occurs frequently when drawing nontree links, whose other end often lies far enough away from the center that the drawing loop ends before the terminating node can be drawn.

3.3.2 Drawing Implementation

Our drawing algorithm is built around two queues: the *ActionQueue* and the *DrawnQueue*, both of which contain nodes that are kept sorted by projected screen area. These queues are manipulated differently based on the current mode: active, idle, or pick. In all three cases the heart of the algorithm is a loop that manipulates these queues and continues until the time budget for a frame is exhausted.

3.3.2.1 Active Mode

The active mode is engaged when the user is active dragging the mouse, or during animated transitions.

1. **Initialization**

   At the beginning of an active frame, the DrawnQueue and ActionQueue are cleared. We initialize the ActionQueue with a single node: the one that had the largest projected screen area in the previous frame. We know that this node is close to the ball's center because of frame-to-frame coherency.

2. **Draw Loop**

   (a) Current node is popped off the ActionQueue.

   (b) Handle the current node.

   (c) Handle the nodes one hop away from the current one in the spanning tree: the parent node and the children nodes.

   (d) If the projected screen area of any of these neighboring nodes is at least one pixel, insert it into the ActionQueue, maintaining sorted order.

   (e) Terminate if:

      i. No more time left, or

      ii. ActionQueue is empty.

3. **Handle Node**

   We handle a node in the active frame by drawing it if necessary: if the node is not already marked,

   (a) Draw the node and mark as drawn.

   (b) Record the node's projected screen area.

   (c) Draw all incoming and outgoing links.

   (d) Insert node into DrawnQueue, maintaining sorted order.

3.3.2.2 Idle Mode

The idle mode is entered when the user stops dragging the mouse, or an animated transition ends. The idle and active modes are identical except for initialization and termination:

1. **Initialization**

   The ActionQueue and DrawnQueue from the previous frame are left untouched.

2. **Draw Loop Termination**

   Terminate if either:

   (a) No more time left, or

   (b) ActionQueue is empty. In this case unset the idle callback before termination.

A single idle frame is triggered by the idle callback only when there is no user input found in the main event loop. Several idle frames can be drawn back to back if no input is found, so that more and more of the scene fringe is filled in. The benefit of splitting the work into several bounded frames is that we guarantee a quick response to user action, which would not be the case if we attempted to draw all the remaining nodes in the scene as soon as the user temporarily stopped moving the mouse. When the ActionQueue is finally completely emptied, the callback is cleared so that the entire application program yields the CPU until further user or program activity warrants new drawing activity.

3.3.2.3 Pick Mode

The pick mode allows the system to identify the drawn object at the current location of the user's cursor.

1. **Initialization**

   We do no explicit initialization work in pick mode since the DrawnQueue created in the previous drawing loop passes is already sorted by projected screen area.

2. **Pick Loop**
    (a) Current node is popped off the DrawnQueue.
    (b) Handle the current node.
    (c) Terminate if:
        i. No more time left, or
        ii. DrawnQueue is empty.
3. **Handle Node**

    We handle a node in the pick frame by testing its screen area against the location of a pick ray cast into the scene using the standard OpenGL support for picking.

    (a) Test the node against the pick ray.
    (b) Record if there is a collision.

A pick could occur after several back-to-back idle frames, which may result in far more onscreen nodes than could be tested for intersection in the allotted time. The right subset to check is the large ones, since they are more likely to be the target of a user click. This is important not only for fast response time, but also to ensure correct behavior on the fringe: clicking on a blob should result in bringing largest node in the blob to the center, not a tiny leaf node that happened to be the exact pixel underneath the cursor.

The H3 library API supports instant visual feedback for a picked node by redrawing it in a highlight color in the front buffer. The picking cycle of the H3Viewer is fast enough to allow locate highlighting whenever the user simply moves the mouse over an object, and of course can also be used to determine whether the user has selected a node in response to an explicit click.

3.3.3 Label Drawing

Text labels are drawn for nodes whose projected screen areas are greater than a user-specified number of pixels. Although the label position is determined by the location of a node in the hyperbolic ball, labels are drawn as Euclidean objects whose size does not depend on the distance from the origin. We position labels in the 3D scene and draw them using any standard screen font, with a backing rectangle to ensure legibility. The labels are Z-buffered, not billboarded. Labels drawn on the right begin just in front of the node's center, which is the ending point for labels drawn on the left. The label left-right positioning is a global option. The color of the backing rectangle always matches the window background, and labels are drawn in reverse video when a node is highlighted.

3.4 Visual Encoding

The visual encoding choice of drawing spanning trees in the projective model of 3D hyperbolic space has many visual and cognitive implications. We discuss the choice of visual metaphor and analyze both the information density of the resulting view and the implications of a radial layout. We also cover the grouping mechanisms and the methods of link drawing.

3.4.1 Visual Metaphor

The two most salient visual features of H3 are that a spanning tree is used as the backbone for graph layout, and that it is drawn inside a ball of space with a pronounced 3D fisheye distortion. The distortion provides a Focus+Context view: a large neighborhood of the graph is visible around a focus node. We discussed the implications of using a spanning tree as the layout backbone at length in section 3.1, so here we focus on the effects of the hyperbolic distortion and the choice of dimensionality.

3.4.1.1 Effects of Distortion

The 3D fisheye effect results from using 3D hyperbolic space for both layout and navigation, and projecting from hyperbolic to Euclidean space as a subsequent step in the drawing process. Although most people are not familiar with the mathematics of 3D hyperbolic space, even novice users can easily navigate by clicking wherever looks interesting. The resulting animated transition that brings that feature to the central focus point provides visual continuity so that the graph is perceived as an single entity despite the distortion.

The hyperbolic distortion is the underlying reason that navigation is quick and fluid. Of course, without the constant frame rate drawing algorithm interaction might not be fluid, but the mathematics of hyperbolic geometry is the true reason that H3 works well for browsing large local neighborhoods.

Although many novel distortion-based visualization interfaces have been proposed (as discussed in Section 2.1), it is still an open problem to precisely characterize what kinds of tasks are *not* effectively served by distortion views. Well-chosen distortions elide extraneous detail, but ill-chosen distortions mislead or hide important information. For example, a distortion method would not be the right choice for tasks which demand making precise geometric or distance judgements. We can consider the effect of the H3 distortion on the three task categories presented in Section 3.1.1.1: structure discovery, linked index, and contextual backdrop.

Since structure discovery is about discovering topological structure, the exact geometric details are not important and the hyperbolic distortion is not misleading. Likewise, geometric distortion in the H3 view does not interfere with linking it to other views. The geometric distortion does affect search, which was not one of our intended task categories. Search is usually more difficult in Focus+Context views than in undistorted ones: serial search through all the objects in a scene is harder if more objects are simultaneously visible. Even undistorted graph layouts are usually not suited for supporting visual search; therefore, the combination of the graph layout and the very large visible neighborhood in H3 makes a linked view optimized for search almost mandatory (as discussed in Section 3.5.1).

The contextual backdrop function is somewhat affected by the hyperbolic distortion, but it is still reasonably well-supported. The main design choice that has an effect on backdrop function is that we do not provide the capability for user control of node shapes. Shapes in projected hyperbolic spaces are distorted everywhere except for the origin of the enclosing ball, such that the shape perceptual channel provides much less distinguishable information than in Euclidean space. Although the area devoted to a node shrinks rapidly as its position recedes from the origin, hyperbolic perspective distortion does not affect color, leaving it an appropriate perceptual channel for conveying nominal distinctions.

3.4.1.2 Dimensionality

Our choice to lay out the graph in 3D rather than in 2D requires justification, since two-dimensional graph layout algorithms which minimize occlusion exist, but occlusion from any single viewpoint is inevitable in most 3D layout approaches. If our goal was just to visualize trees, we might not have chosen to use 3D. Although we could lay out a tree in only two dimensions, most of the quasi-hierarchical graphs that we use as examples are nonplanar. In two dimensions the non-tree edges of the graph would unavoidably cause edge crossings, but our spanning tree layout in 3D space allows these non-tree edges to be drawn without intersection.[9] Although occlusion will occur in any single snapshot, the interactive rotation and translation capabilities can help the user understand the structure.[10]

3.4.1.3 Information Density

By carefully tuning the spatial layout algorithm to exploit many of the properties of 3D hyperbolic geometry, we succeed in providing a display with the right amount of information density: neither too cluttered nor too sparse. The H3 layout results in many more visible nodes than the 2D hyperbolic tree layout from PARC [LRP95] in any individual frame. We can categorize the drawn nodes into three classes: main, peripheral, and fringe. Main nodes are large enough to have labels drawn, and possibly also visible color or shape coding.

[9] Intersections could theoretically occur, since we do not explicitly check for them, but our layout algorithm is designed to make them highly unlikely. We have never observed non-tree link intersection in practice.

[10] Two studies have found beneficial effects from 3D kinetic depth cues for both trees [SM93] and general graphs [WF96], where larger structures could be understood with kinetic 3D than with flat 2D layouts.

What we call peripheral nodes are small, but still distinguishable as individual entities on close inspection. Fringe nodes are not individually distinguishable, but their aggregate presence or absence shows significant structure of far-away parts of the graph. Each class can fit roughly an order of magnitude more than the preceding one. The H3 and PARC tree browsers can both show up to a few dozen main nodes with visible labels. The PARC layout doesn't have peripheral nodes as such, since nodes are not drawn as discrete entities. The H3 layout can show up to hundreds of peripheral nodes. The H3 fringe can show aggregate information about thousands of nodes, whereas the PARC fringe aggregates information about hundreds of nodes. Table 3.2 summarizes the browser density comparison.

System	Main	Peripheral	Fringe
H3	dozens	hundreds	thousands
PARC 2D Hyperbolic Tree	dozens	-	hundreds

Table 3.2: **Maximum information density comparison for two hyperbolic browsers.** Main nodes are labelled, peripheral nodes are individually distinguishable, and fringe nodes are significant because of their aggregate presence or absence. See the Figure 3.20 caption (page 59) for a similar breakdown of an H3 screen snapshot.

Information density is a tradeoff between clutter and void. We present a different analysis of information density by relating it to the the topological idea of **codimension**, which is the difference in dimension between a structure and the space surrounding it. Table 3.3 shows a comparison between three systems for graph layout that have each have a different codimension. In all cases, the surrounding space is three-dimensional, therefore the layout strategies differ only in terms of the dimension of the structure used for layout.

	Space Dimension		Structure Dimension		Codimension
webviz	3	-	1	=	2
H3	3	-	2	=	1
(Carpendale)	3	-	3	=	0

Table 3.3: **Codimension comparison for three graph drawing systems.**

The first example is the *webviz* [MB95] system, our first attempt to bring cone trees to 3D hyperbolic space (which predated the work in this thesis). The *webviz* layout strategy shown in Figure 3.5 (page 30) was identical to the classical cone tree approach, and involved placing nodes on the circumference of a circle at the mouth of a cone. That circumference is a linear one-dimensional structure, so the codimension was 2, the difference between three and one. The codimension 2 *webviz* system was quite sparse.

The H3 layout strikes a reasonable balance between information density and clutter in part because it has codimension 1: that is, the nodes are arranged in a space one dimension less than that in which they are embedded. We assert that codimension arguments are particularly important for 3D graph layout systems, since minimizing occlusion is the major challenge. In all 3D graph layout systems, graphs are embedded in a 3D volume of space. The H3 algorithm places nodes on the 2D surface of a hemisphere at the mouth of each cone, instead of the 1D circumference. The 2D structure results in a happy medium between the sparseness of a 1D line and the density of a 3D volume. Since our embedding volume is a ball that we regard from the outside, if our node layout were too dense, the leaf nodes near the ball's surface would block our view of the rest of the structure.

We argue that our codimension 1 approach of surfaces in a volume is the right choice in most 3D graph layout situations. Some researchers have chosen a codimension 0 approach: Carpendale and colleagues [CCF97] placed graph nodes in a 3D grid, such that the dimension of both the laid-out graph and the space in which it was embedded was a 3D volume. Their interactive viewing system allows a view of an inner grid node by introducing a line-of-sight distortion probe to move the occluding nodes out of the way. Although the distortion does allow a small number of inner nodes to be seen, most nodes are not visible at any single time. We argue that codimension 0 approaches should be approached with caution because of their inherent occlusion problems, unless the semantics of the data or task specifically require a volumetric layout.

3.4.2 Grouping

Creating groups is explicitly supported in the H3 system. Groups can be drawn or hidden, and can also be nominally distinguished using color.[11]

A node can have multiple attributes, and each attribute can be set to a value. These values are the groups. One attribute could be used to control whether a node is drawn, while a different attribute could be used for coloring. For instance, in Figure 3.22 each of the nodes (which represent web documents) has been assigned a value for two distinct attributes. The document type attribute is used for coloring: the value HTML is colored cyan, the image value is colored purple, the VRML value is blue, and so on. Another attribute is location, which is used to filter node drawing. Local nodes are drawn, whereas external nodes are hidden.

The attribute-value mechanism gives the application user explicit control over grouping nodes. In contrast, links fall into one of two more fundamental groups: spanning tree links and non-tree links. These groupings are not explicitly specified by the users, although they can be indirectly controlled by the user's

[11] We use the term "color" advisedly: although we explicitly distinguish between hue, saturation, and brightness in the Constellation system (Section 5.3), in H3 colors are assigned as RGB triplets. Although sophisticated developers could of course choose to distinguish between hue, saturation, and brightness at the application layer, it would be misleading to imply that H3 supports those three perceptual channels instead of a single channel of color.

choice of spanning tree algorithms. Spanning tree links are always drawn. The non-tree links are drawn on demand, either for a particular node or for an entire subtree, for a fine-grained control of the display. The directionality of all links can be indicated by gradually interpolating between one color at the parent end and another at the child end. The default coding (changeable through the API) uses a subtle red to blue coding. This color coding is non-intrusive when only the spanning tree is shown, but allows the user to distinguish between incoming links and outgoing links when non-tree link drawing is enabled.

3.5 Implementation

The first implementation of the H3 layout algorithm was deployed before we developed the H3Viewer drawing algorithm. The application was written in C++ and used a highly modified version of Geomview [PLM93] for the drawing engine. The user needed to explicitly expand or contract subtrees, since this gardening was not automatically invoked by navigation. The original PARC cone tree system [RMC91] also used explicit gardening controls.

The next version of the system featured an OpenGL implementation of the H3Viewer guaranteed frame rate drawing algorithm. The adaptive drawing routines incorporated automatic subtree expansion and collapse, obviating the need for manual gardening by the user. We incorporated both the H3Viewer drawing and the H3 layout code into a callable library with a documented API.

The H3 layout and H3Viewer drawing libraries have been integrated into Site Manager, a commercial product from Silicon Graphics that was bundled with Irix.[12] Site Manager was designed to aid webmasters and content creators in web site creation and maintenance. The visualization component was integrated with components for version control, editing, previewing, and log file analysis.

3.5.1 Linked Views

In section 3.1.1.1 we discussed the value of linking an interactive graph browser to other views of the data. The Site Manager system mentioned earlier is an example that features a tight integration between the 3D hyperbolic graph browser and several other views of the web site dataset. The screen snapshot in Figure 3.15 shows part of the Stanford Graphics Group web site on display during a Site Manager interactive session. The 3D hyperbolic graph view shows the hyperlink structure of the site, which is linked with the 2D browser view that shows the directory tree structure in traditional outline format. When a node is selected in one view, it is highlighted and moved to the center in both. If several nodes are selected at once, by rubberbanding in

[12] `http://www.sgi.com/software/sitemgr.html`

hit playback of the traffic logs. Instead of being colored by MIME type, the documents are being gradually colored according to their popularity on a color scale with a saturation ramp ranging from grey to red, as shown in the key in the bottom left of the H3 window. At the beginning of the playback, all nodes are initialized to grey. Each hit recorded in the logs triggers a series of color changes: first the source node, then the link between the source and destination nodes, and finally the destination node turns yellow. Then the source returns to its original color, and the destination permanently changes color to a slightly more saturated red. Since the color changes occur in quick succession, the visual effect is a source shooting a laser beam at a destination, making it hotter on each hit. The user can choose to have the destination mode always animate to the center so that the user's activity is always visible instead of being potentially lost in the fringe, or retain navigational control if the goal is to understand the traffic in a local area. The user can also choose to have the affected links stay yellow to leave trails that persistently show hits, even those that were not on the spanning tree. As we discussed in section 3.1.1.1, showing the choice actually made by the user in the context of all available hyperlink possibilities provides a detailed picture of user activity that is hidden in the simple flat list of aggregate popularity.

Linking an H3 window to additional windows or user interface elements can also make better use of screen real estate than a standalone radial layout – a solo H3 window subtends a square instead of a rectangle with the dimensions of the standard 4:3 monitor aspect ratio.[13]

3.5.2 Precision

Although the layout described in section 3.2 uses hyperbolic distances, we must eventually project from hyperbolic to Euclidean coordinates so as to actually draw a picture with standard low-level graphics libraries.[14] The stage in the algorithm at which this conversion takes place has a major effect on the size of the static structure that can be displayed without encountering precision problems. Distinct hyperbolic coordinates that are too far from the origin will be projected so close to the surface of the unit ball that there are not enough bits to distinguish between their Euclidean coordinates. Such nodes are far from the root and project to an area much smaller than a single pixel. The limit comes into play only if we store a static structure in Euclidean coordinates that is much larger than the part that is actively being drawn.

In our current implementation we do store a static structure: we project immediately after the layout phase, and then change the focus from one part of the structure to another by applying a transformation to the static structure. Any single mapping from hyperbolic to Euclidean space permits drawing only a limited

[13] The library API also allows the trivial solution of nonsquare windows with a distorted aspect ratio so that the sphere at infinity is drawn as an ellipsoid.

[14] All standard libraries support the additional projection step from 3D Euclidean space to 2D screen space.

number of nodes before the drawing system succumbs to precision problems.

In the first Geomview implementation, nodes that were too far from the current root of the tree were simply truncated, and could be seen only if by reloading a new input file where the root used to compute the spanning tree was changed. Moving from single to double precision extended the viewable range, but was only a stopgap improvement.

The current version of the H3Viewer drawing algorithm partially addresses the precision problem. The automatic expansion and contraction tied to the navigation allowed users to productively view datasets large enough that the precision horizon was actually reachable. The H3Viewer system will compute a remapping from hyperbolic to Euclidean space as necessary when the cumulative error becomes too great. This remapping is a global operation that depends on the total number of nodes in the scene instead of only the visible nodes. When drawing large graphs, this remapping will cause a temporary interruption in the otherwise smooth frame rate or a jump instead of an animated transition. This remapping is the only exception to our frame rate guarantee.

A more sophisticated implementation could defer the projection until render time and dynamically determine the appropriate Euclidean coordinates for only objects in the neighborhood around the focus that are large enough to see. An incremental mapping algorithm along these lines is possible future work.

3.5.3 Availability

The Site Manager application is bundled with Irix 6.2-6.5, and can also be downloaded for free.[15] It runs only on SGIs.

The H3 and H3Viewer library source is available for free noncommercial use.[16] The software was written in C++ and OpenGL, so it will run on any machine that has OpenGL (or Mesa), including Windows and most Unixes. The library distribution includes an example application of a simple standalone graph viewer.

3.6 Interaction

Static pictures such as figures in this paper could be misinterpreted as showing 2D objects on the surface of a hemisphere instead of as 3D objects inside the volume of a ball. In the interactive system the dimensionality is obvious when the objects rotate inside the ball.

The main way for users to navigate is by simply clicking on a node or edge. When the user clicks on a node, it is selected and undergoes an animated transition to the center of the sphere. The frame rate during

[15] `http://www.sgi.com/software/sitemgr.html`
[16] `http://graphics.stanford.edu/~munzner/h3`

these controls. In the Site Manager application, we explicitly distinguish between entering and leaving links categories in the user interface, but other application programmers might choose to hide this API distinction if such fine-grained control is unnecessary for their target task.

3.7 Results

Our implementation can handle graphs two orders of magnitude larger than most previous systems by manipulating a backbone spanning tree instead of the full graph. Carrying out both layout and drawing in 3D hyperbolic space lets us see a large amount of context around a focus point. Our layout is tuned for a good balance between information density and clutter, and our adaptive drawing algorithm provides a fluid interactive experience for the user by maintaining a guaranteed frame rate.

We begin by discussing the visual appearance of several datasets under the H3 layout, followed by an analysis of the layout algorithm speed and scalability that is both quantitative and qualitative. We next discuss a user study conducted at Microsoft Research that found a statistically significant improvement in task time when a novel web browser that included the H3Viewer was compared to more traditional browsers. We then present several additional task domains where H3Viewer was used, and conclude by summarizing the outcomes of the project.

3.7.1 Visual Appearance

The H3 layout technique can easily handle tens of thousands of nodes and has been tested on graphs of over 100,000 nodes. It is highly effective at presenting a large neighborhood around a focus node of a huge graph in a small amount of screen space. For instance, in Figure 3.20 the user can see enough of the distant subtrees to identify dense and sparse ones. The destinations of nontree links are distorted, but the rough sense of their destination helps the user construct and maintain a mental model of the larger graph structure. Figure 3.18 shows how the details become clear in a smooth transition when an area of the structure moves towards the center. The context shows up on several levels: the local parent-child relationships of the spanning tree, the nontree links between disparate nodes in the graph, and the rough structure far away from the current focus of interest.

Figure 3.21 shows our layout techniques at work on one of the function call graphs described in section 3.1.1.3 instead of the web sites shown in many of the other figures.

Figure 3.22 shows a medium sized web site that contains 5000 total nodes. The top image shows a selected node with outgoing nontree links drawn. Although distant subtrees are quite distorted, we can see enough context that the destinations of the non-tree links can be roughly distinguished. In the other two images we bring the cluster of nodes that contains the destination of most of the nontree links closer to the focus. Note that we can still see the originating node, although it is quite far away in the tree structure.

3.7.2 Speed and Size

The H3 layout algorithm is linear in the number of spanning tree edges. Adding an edge to the spanning tree is a constant amount of work, and every edge addition requires one comparison, so the spanning tree creation is order $|E|$ (the number of edges in the graph). In the current implementation the required input file format is a depth-first traversal of a graph with an explicitly declared root.[17] The node layout algorithm depends on only the spanning tree structure, not the underlying graph structure. The top-down placement pass is order $|V|$, the number of vertices in the graph, since placing the child hemispheres on the parent requires only one computation per node. The bottom-up estimation pass where we sum over the child nodes is also order $|V|$. On an SGI Onyx2 a large graph with 110,000 edges was laid out in 12 seconds, a medium sized graph of 31,000 edges was laid out in 4 seconds, and a small graph of 4000 edges took a fraction of a second.

The drawing time is constant - the H3Viewer drawing algorithm always maintains the target guaranteed frame rate. A slow graphics system will simply show less of the context surrounding the node of interest during interactive manipulation. Figures 3.13 and 3.14 show two views of the same scene - one corresponding to what the user would see during interaction, and one after the fringe has been filled in when the user was idle. On startup, the initial loading phase includes both file I/O and data structure building. For the graphs mentioned in the previous paragraph, this time was approximately 2 minutes, 20 seconds, and 2 seconds, respectively.

The layout and drawing algorithms presented here work well up to the limits of main memory, but not beyond: if the entire graph does not fit into main memory, the system is unusable. The graph with 110,000 edges could be manipulated interactively on an SGI with 1 GB of main memory but could not be loaded on a smaller 128 MB machine.[18] Thus, the obvious computational limit on H3 scalability is main memory. Although the drawing algorithm uses only information about a local area, the layout algorithm is global. The precision problem discussed in section 3.5.2 is also a limit on the current implementation, albeit not on the algorithm itself.

However, we have also found cognitive factors that limit its scalability, and these factors cannot be solved by simply adding more memory to a machine or tightening the memory footprint of the software. Depending on the characteristics of the dataset, the local visible neighborhood around a focus point in H3 is between

[17] If we relaxed that requirement, spanning tree creation would instead require the $n \log n$ time of Kruskal's algorithm.

[18] The most recent version of the software has a somewhat smaller memory footprint, but the fundamental point remains.

five and twelve hops. Although this is a much wider local view than previous systems, it is nevertheless not a global overview. The distinction is blurred in datasets of moderate size, but becomes clearer as the size of the dataset increases. People can easily become disoriented when the H3Viewer displays huge datasets of over 100,000 nodes. The important factor in retaining orientation is the ability to relate the currently visible neighborhood to the entirety of the graph, which is related to the diameter and branching factor of the graph as opposed to the sheer number of nodes. We conjecture that disorientation sets in when the diameter is over an order of magnitude larger than the visible neighborhood.

The combination of a carefully tuned layout and interaction allowed us to push the limits of large graph browsing far beyond their previous place of a few thousand nodes. We conjecture that the reach of H3 could be extended by adding landmarks to the scene, perhaps so that the direction of a path back the root node was always obvious. Another possibility would be to link an H3 window with a true global overview window, perhaps based on the skeletonization work of Herman and colleagues [HMM$^+$99]. One way to extend the existing H3 implementation to much larger datasets would be by using multiple linked H3 windows to explore the dataset at more than one discrete levels of detail: clicking on a node in a higher-level window would cause its subgraph to be displayed in another window. This scheme could probably scale to a few levels of recursion before disorientation set in.

3.7.3 User Study

Kirsten Risden and Mary Czerwinski of Microsoft Research ran an empirical study [RCMC00] comparing three web browsers, one of which incorporated the H3Viewer libraries. The user study demonstrated the strengths and weaknesses of three browsers. Two were conventional 2D browsers: a standard collapse-expand tree browser of the form found in many Windows applications, and a web-based hierarchical categorization from the `snap.com` portal (which is similar to Yahoo!). The third browser was XML3D, a novel browser that integrates an H3-based interactive 3D hyperbolic graph view with a more traditional 2D list view of the data. The focus of the study was browser designs that could help organize, make sense of, and manage large collections of documents available on the Web. We were were particularly interested in collections that are not strictly hierarchical in structure. Our target user population deals with non-hierarchical datasets, where categories may have one or multiple parents, on a daily basis in the course of Web site content generation and maintenance.

XML3D is capable of graphically representing the information contained within schema-like, XML text files. The interface consists of two types of linked components: an instance of the H3Viewer and several 2D lists – of parents, children, and siblings – which dynamically update based on the selected graph node.

opposed to a new category. The main effect of category was qualified by a significant interaction with the parent variable, $F(1, 11) = 60.84, p < .001$. Follow-up analyses showed that the difference between existing and new category tasks was larger when a single parent was involved, $t(14) = -9.96$, $p < .001$, than when multiple parents were involved, $t(13) = -1.07$, $p > .05$. Neither the main effect of parent nor any of the other interactions were significant.

There were no reliable differences in overall user satisfaction or consistency across the three user interface designs. In other words, the benefit in performance in the XML3D case was not offset by a lack of agreement across users as to where web content should be placed. It was informally observed that integrating the ability to view the overall structure of the information space with the ability to easily assess local and global relationships was key to successful search performance. XML3D was the only tool of the three that efficiently showed the overall structure within one visualization.

Future studies should explore optimal ways of integrating the use of novel 3D graph layouts and 2D lists for effective information retrieval, so that other kinds of tasks might benefit from the new designs. Nevertheless, we have seen that giving users the option to use a novel visualization in addition to a traditional 2D list view appears to be a powerful design solution, at least until alternative user interfaces become more pervasive.

3.7.4 Outcomes

The H3 system was the most successful of the three projects by several of the evaluation metrics discussed in section 1.3.2.1: user studies, tangible algorithmic improvements compared to previous systems, and size of user community. The preceding section contains the details of the performance advantage for an H3-based system found in the user study. We have presented a layout algorithm that scales linearly to datasets 100 times larger than almost all previous systems and a drawing algorithm that guarantees a constant frame rate. The potential user community is large because H3 was incorporated into a commercial product.

One of our motivations for integrating the research software with a product was the hope of getting feedback from a large user community. Our previous experience with large software projects was as one of the core developers of Geomview [PLM93], a public domain 3D interactive graphics package that explicitly solicited user response in its documentation, which generated considerable feedback. Site Manager instead followed the pattern of a commercial product release, which does not establish channels of communication between the developers and the users. Thus, we have no data on the size and satisfaction level of the commercial users.

However, we have had communication from noncommercial H3Viewer users. SGI funded the development of the H3 and H3Viewer code base, but agreed to allow us to release the libraries for free noncommercial use. The software has been publicly downloadable[19] since 1999, and was also privately released to a limited number of interested researchers between 1997 and 1999. The following section covers these additional task domains.

3.7.5 Additional Task Domains

The H3 system has been used for many kinds of data in addition to web hyperlink structure. Our collaboration with the SUIF compiler group to incorporate an H3 view of function call graphs into their development environment had a promising start, but was cut short when the key contact person left their group. Several researchers have converted their datasets into the H3Viewer format to use the simple standalone viewer to browse through their data. A representative example is Dave Vieglais of the Natural History Museum and Biodiversity Research Center at the University of Kansas, who wanted to see species taxonomy data. Such a project involves a minimal investment of researcher time, on the order of hours or days.

A more significant investment of time, on the order of weeks or more, was required from the researcher who incorporated the H3 libraries into his own software system. Daniel W. McRobb of the Cooperative Association for Internet Data Analysis (CAIDA) did so for two problem domains in networking: Internet tomography and BGP peering relationships.

Skitter: Internet Router Connectivity Skitter[20] is a project from CAIDA to develop and deploy active measurement tools for the dynamic discovery and depiction of global Internet topology and for measuring performance across specific paths. The project goals are acquiring infrastructure-wide connectivity information, collecting round trip time and path data for up to 60,000 destinations, and analyzing the visibility and frequency of routing changes. The Skitter team tried several methods of visualizing their large datasets, including Cheswick and Burch's software[21], and have settled on the H3Viewer as the more effective approach for their needs as of yet. They built a custom software tool for browsing this dataset that incorporates the H3Viewer libraries.

Autonomous Systems McRobb has also used H3 to explore the peering relationships between the Autonomous Systems (ASes) that constitute the backbone of the Internet. The routing relationships between Autonomous Systems are notoriously complex because the policies that can be set via the Border

[19] `http://graphics.stanford.edu/~munzner/h3`
[20] `http://www.caida.org/Tools/Skitter/`
[21] `http://www.cs.bell-labs.edu/~ches/map`

Gateway Protocol. Visualizing these graphs is particularly challenging for many systems because of the large branching factor for many of the nodes. The H3 layout is able to handle these large datasets, and the guaranteed frame rate drawing algorithm is critical for exploring them in real time. Figure 3.25 shows McRobb's analysis of Autonomous System data gathered using the `mrtd` tool.[22] (Figure 3.13 on page 41 also shows Autonomous System data, from a different source.)

[22] `http://www.mrtd.net`

Chapter 4

Planet Multicast: Geographic MBone Maintenance

Our first case study is a geographic visualization of the MBone, the Internet's multicast backbone topology. It allows efficient transmission of real-time video and audio streams such as conferences, meetings, congressional sessions, and NASA shuttle launches. The MBone is an interesting domain that shares many characteristics with the global Internet, but provides a much more manageable testbed because it is several orders of magnitude smaller. They both have experimental origins, exponential growth rates, and bottom-up growth without planning by a central authority. The latter property has led to a tunnel structure that is not optimal and hard to decipher. We built a visualization system to help maintainers more accurately understand the large-scale geographic structure of the MBone.

This chapter begins with a discussion of the task that we addressed with the Planet Multicast system in section 4.1. We next discuss our choice of spatial layout in section 4.2: we created a geographic representation of the tunnel structure as arcs on a globe by resolving the latitude and longitude of MBone routers. In section 4.3 we move on to visual encoding and interaction techniques such as grouping and thresholding. Section 4.4 is devoted to more specifics about the implementation. The chapter concludes with a discussion of the results and outcomes of the project in section 4.5.

of them. Since tunnels are opaque to the multicast protocol, more than one tunnel overlaid on a single unicast link will result in multiple multicast packets of identical data passing through that link. The middle and right of Figure 4.2 show a comparison of good and bad tunnel placement.

However, there is no central arbiter of tunnel placement: the only coordination required in setting up a new tunnel is finding a willing network administrator at the other endpoint. The result is that some new tunnels are placed haphazardly, and in the worst case may even use unicast links that already have a tunnel overlay. The rapidly changing nature of Internet complicates the tunnel placement task. A tunnel that connected two multicast-enabled islands via an uncongested high bandwidth unicast path six months ago might now be routed through a different, more congested unicast path. Moreover, some of the intervening routers may have been upgraded to support multicast, so the single long-distance link should ideally be split into two shorter tunnels.

4.1.4 Target Users

The primary intended audience for the Planet Multicast visualization system was the core MBone developers and maintainers. Our goal was to help with the maintenance task of finding badly placed tunnels that might be causing distribution inefficiencies by wasting bandwidth. As the size of the MBone grew and Internet congestion became increasingly problematic, there was greater incentive to tune tunnel placement to provide the most efficient distribution topology and minimize the imposed Internet workload.

Although the MBone maintainers have no formal authority, they can and do make suggestions, which are usually followed. A secondary goal was help people joining the MBone who need to find a good place to connect a new tunnel.

4.1.5 Topology Data

In the beginning the MBone was a new and small enough system that maintaining detailed maps of its topology did not require a sophisticated infrastructure. The early mapmaking projects, which culminated in a four-page PostScript printout meant to be hand-tiled, were abandoned in mid-1993.[1]

Since the MBone is now too large to be manually indexed, the only way to discover its topology is through active detection. The `mwatch` program developed by Atanu Ghosh at University College London lists the IP addresses (and often hostnames) of multicast routers at tunnel endpoints. Piete Brooks at the University of Cambridge used this tool to traverse the topology of the MBone every night.[2][3] We were able to augment

[1] `ftp://parcftp.xerox.com/pub/net-research/mbone/maps`
[2] `http://www.mbone.cl.cam.ac.uk/mbone`
[3] Although this resource was available at the time of the project in 1996, in 1998 Brooks stopped keeping this list because of an

> multe.uib.no(129.177.13.20)	lilja.uib.no(129.177.11.224)	[1/6/tunnel/down/leaf]
> multe.uib.no(129.177.13.20)	gwaihir.zi.uib.no(129.177.64.13)	[1/6/tunnel/down/leaf]
> multe.uib.no(129.177.13.20)	munin.hf.uib.no(129.177.207.20)	[1/6/tunnel/leaf]
: cs2.slu.se(130.238.118.1)	11.0,mtrace	-4885
> cs2.slu.se(130.238.118.1)	ultrouter5.slu.se(130.238.118.2)	[1/0/pim]
: lab-1-rtr-S0.InterNex.Net(205.158.1.18)	11.1,prune,mtrace,snmp	-1061037
= lab-1-rtr-S0.InterNex.Net(205.158.1.18)	205.158.3.209(205.158.3.209)	
> 205.158.3.209(205.158.3.209)	INADDR_ANY(0.0.0.0)	[1/0/pim/querier/leaf]

Table 4.1: **Raw tunnel data from** `mwatch`**.** The MBone topology data from June 1996 consisted of over 75 pages of textual data in this form.

the Cambridge list with information about MBone routers in a few firewalled private networks thanks to help from one of the MBone maintainers. Hidden within this large dataset were highly suboptimal tunnel placements, misconfigurations, and outdated parts of the MBone topology that should be removed. As of June 1996, the MBone topology dataset consisted of over 75 pages of textual data of the form shown in Table 4.1.

4.2 Visual Metaphor

Planet Multicast showed the MBone tunnel topology using the visual metaphor of arcs rising from a 3D globe, with detail-on-demand hypertext associated with each arc. Figure 4.3 shows this straightforward visualization scheme, which deliberately emphasizes the salience of geographic distance. This simple approach was quite effective, despite being neither novel nor algorithmically complex. Immediate comprehension is one of the great advantages of this highly literal representation: users need little or no explanation of the visual encoding.

4.2.1 Geographic Distance

Badly placed long-distance tunnels are more likely to cause problems than misplaced tunnels that span only a short geographic distance. Long-distance tunnels are likely to contain more unicast hops, and at least one of those unicast hops must itself span a long distance. The longest-distance unicast links are often between continents. Such links are a scarce resource since they are limited in number and often congested. Moreover, if multiple MBone tunnels exist between two widely disparate geographic locations, the chance that more than one tunnel is routed through some single unicast link is higher than if the tunnels are local.

Since geographic distance is weakly correlated with resource usage, we chose to emphasize it in our visualization. We might have chosen a different approach if we had access to resource usage data, but gathering information about the underlying unicast links in the global Internet would have been prohibitively difficult, so no such data was available.

upstream provider charging policy change.

Thus, our choice of visual metaphor acts as an implicit visual filter that reduces the dataset considerably before rendering: in Figure 4.3, 3200 of the roughly 4400 tunnels are not drawn because they are within the same city. (700 tunnels are drawn, and we were unable to determine a geographic position for the remaining 500.)

Both of these related properties of the geographic metaphor fit with our assumption that distance is correlated to unicast resource usage. However, we also know that this correlation is weak, which means that this visual metaphor is only partially matched with the underlying task of ensuring multicast distribution efficiency.

One advantage of using a 3D globe instead of a 2D birdseye map is that it reduces clutter by occluding one hemisphere. Put another way, rotating the globe around its own axis is an explicit visual filter that controls which hemisphere is visible at any one time. Although the occlusion could be problematic since it precludes an overview of the entire dataset, such an overview is usually unnecessary for the more local task of finding long-distance tunnels. Tunnels that pass between the front and occluded hemispheres are visually salient despite having only one visible endpoint. Since the interactive interface provides hypertext details on demand in response to a mouse click on a tunnel arc, there is no need to navigate to the other endpoint.

When we tried mapping the tunnels on a standard 2D world map, the lines representing the tunnels were far too dense to allow for interpretation. The clutter was particularly severe because long-distance tunnels between North America and the Pacific stretched across the entire map, occluding important information about European tunnels. Thus, the hemispherical occlusion of the globe is actively useful.

Another important advantage of using a globe is the ability to rotate around a point on its surface for a horizon view, as in Figure 4.4. The oblique viewpoint allows the user to see the varying arc heights clearly at a local level, which is useful for improving the visibility of shorter tunnels.

We use a globe constructed by Stuart Levy using the CIA World Map database. Most of the figures show outlines of the continents drawn on a solid-colored sphere. Figure 4.5 shows two uses of texture maps, which lead to increased computational requirements on machines that do not have hardware texturing support.[4] Photorealistic texture maps that include geographic features introduce visual clutter that is extraneous to the visualization task. However, the more abstract texture that distinguishes land from water can serve a useful purpose.

[4] In 1996 hardware texturing support in low-end machines was less common than it is in 2000.

4.3 Visual Encoding

Although the spatialization choice of arcs on a globe is the most powerful perceptual cue, we can encode additional visual information using **groups** of tunnels. The purpose of these additional visual encodings is twofold: to make nominal distinctions between logical sets, and to help the user understand complicated tunnel structure in densely connected areas.

We can partition the dataset into groups of tunnels using the batch pipeline discussed in section 4.4. Those groups can then be manipulated in the browser by interactively choosing different colors and linewidths, or explicitly choosing to draw or hide the group. Figure 4.4 shows a scene where one group of tunnels is distinguished from the rest via color and linewidth. More color and linewidth changes are shown in Figures 4.8 and 4.9, in order to help separate the coast-to-coast tunnels that were likely to be legitimate from those that were potentially problematic.

Groups can also be interactively moved with respect to the others, which can help the user understand complicated structures. The Gestalt principle of common fate is a cognitive explanation for why moving one set of objects against the static background of the rest of the scene is visually comprehensible. Although such motion of course dislocates the tunnels from their geographic reference points on the globe, which would be disorienting for a single tunnel, it was useful when comparing tunnel groupings complex enough that the tunnels alone connote geographic structure. The bottom left of Figure 4.9 shows an example where it was useful to move the BBN tunnels away from the cluttered United States, so that they could be quickly inspected without explicitly eliding all the others.

Another way to help the user understand the details of dense tunnel structures is by **thresholding**, where only part of the tunnel arc is drawn. The mode that we found the most useful is to elide the middle of the tunnel, leaving partial arcs ascending from each endpoint as in Figure 4.6. The thresholding technique for reducing clutter was used in previous network visualization systems [BEW95], thus it is not novel. It is a simple and obvious first choice that was effective, echoing the entire arcs-on-globe visual metaphor. We needed the thresholding capability to check for problematic areas in the American Midwest, which was hidden by the large number of tunnels between the East and West coasts.

4.4 Implementation

We leveraged existing software whenever possible, using existing browsers that supported linked 3D data and hypertext for display. We created the geometric data for tunnel arcs and their associated hypertext using a billing addresses for these domains are useless for individual host location data. Hosts of large administrative entities such as backbone providers are often the most critical in building a true picture of network usage.

We constructed our own database using a collection of manually intensive methods. The InterNIC database was only one source of information. Our collaborators used Web searches, network maps, traceroutes, personal communication, and personal knowledge about organizations and network structure. Despite our best efforts, the database contained inaccuracies and omissions.

4.4.2 Browsers

During most of the development of Planet Multicast we used the 3D viewer Geomview [PLM93] in conjunction with a slightly modified version of the WebOOGL [MMBL95] external module for handling hyperlinks. The 3D tunnel geometry contained a link to hypertext containing the IP address, hostname, location (city and state or country), and latitude and longitude of the tunnel endpoints. Providing detail on demand is a standard information visualization technique [Shn96], but doing so via hypertext was novel at the time.

Geomview has the navigational capability necessary for the oblique horizon view, and supports interactive changes of color, linewidth, and position for named groups. We also made the interactive 3D maps available in VRML [CM97b] using the existing Geomview-to-VRML converter. Although most VRML browsers were less capable than Geomview in that they did not support the preceding features, they did at least allow users to spin the globe around and see the hyperlinked information by clicking on the tunnel arcs. (In 1996, ubiquitous multiplatform support for VRML seemed near, but as of 2000 it has not lived up to that promise.)

We used Geomview for exploratory interactive analysis of tunnel groups, usually using the high-end SGI version for display of high resolution data at interactive frame rates. Useful configurations of color and linewidth coding were then exported via VRML by rerunning the pipeline and hardcoding in the color and linewidth values that were found during a previous interactive session. The VRML files that we exported for wide dissemination usually contained lower resolution data, with fewer segments for both the continental outlines and the arcs. We assumed that most of the VRML users would have lower-end systems could not maintain reasonable frame rates with a large number of short line segments.

4.4.3 Availability

The Planet Multicast software is publicly available.[7]

[7] `http://ipn.caida.org/Tools/pipeline`

coding: first, some of the redundancy may be legitimate, in the cases where the long-distance tunnels belong to major national backbones. Second, many of the long-distance tunnels are not directly under the control of the backbones, and these are good candidates for further investigation by the MBone maintainers.

We then drilled down to further understand the status of the first group, namely the major backbones colored black in the previous figure. Figure 4.9 shows the dataset partitioned further, by color-coding the groups corresponding to each backbone ISP.[8] The combination of color and linewidth coding, interactive navigation, and moving groups away from the central mass to quickly see hidden structure allowed us to confirm that the each individual backbone had a quite reasonable topology with little redundancy. It would have been much harder to make sense of the structures with only a 2D birdseye view, or tunnels that could not be quickly moved in and out of position.

All figures thus far have shown the MBone on the same day, in June 1996. Figure 4.10 compares the changes in the MBone across a four-month period of time. In both figures we highlight the Sprintlink tunnels and focus on Texas. We saw that in February Texas A&M University (TAMU) had configured a tunnel to a Sprint hub in Washington DC. We noticed that by June Sprint had extended their tunnel support to a major hub in Fort Worth, but the university had not leveraged the new topology and still tunnelled all the way to the East Coast instead of using the nearby hub. In a situation where TAMU and some other Sprintlink customer in Texas were both subscribed to the same multicast channel, it is likely that identical multicast packets would be routed through both tunnels. In the worst case, those packets might even traverse the same underlying unicast link. If the channel were high-bandwidth video, for instance a space shuttle launch, the result would be congestion. This situation is an example of a tunnel placement problem that had gone unnoticed in the text input data, but was easily found with the geographic visualization.

4.5.2 Additional Task Domains

Both the pipeline software and the geographic databases that we developed have been used for the visualization of other network topologies.

We used the toolkit ourselves to visualize the traffic load on the Squid[9] global web caching hierarchy [Wes96], visually encoding the traffic information by coloring arcs between parent and child caches. We used the toolkit for over a year as part of a nightly batch process to create a web page with images and VRML files showing the previous day's cache usage.[10] However, this page is no longer active because the cache usage information is now being gathered differently.

[8] This particular backbone segmentation was suggested by collaborators who were familiar with the multicast community.
[9] `http://ircache.nlanr.net/Cache/cacheviz.html`
[10] `http://ircache.nlanr.net/Cache/daily.html`

self-interest. We anticipated that a bootstrap database, despite its imperfections, would gain support from ISPs who would find enough benefits that they would contribute their internal (presumably accurate and up-to-date) data to maintain and improve it. Although our hopes proved to be somewhat optimistic, K. Claffy of CAIDA has continued to add to and maintain the database, and has indeed received some ISP data. Although we have not personally continued with this line of research, her group has built on our 1996 results in more recent projects [PN99, Net99, CH].

Geographic techniques will probably have only limited applicability for showing global MBone or Internet network infrastructure as long as geographic locations for routers are hard to discover. However, these techniques may work for corporate or otherwise proprietary networks where internal documentation of router locations is available. This approach is probably also appropriate for showing final destinations such as Web servers, since geographic information for Web servers is much more likely to be correct than for routers inside backbone networks.

Another obstacle to widespread adoption was that VRML viewer deployment was not as ubiquitous as we had expected. Although somewhat stable viewers exist for most platforms now, most network maintainers do not have them installed on their machines. The VRML movement has definitely lost momentum compared to its heyday, because of many shortfalls during a critical window of time: browsers were insufficiently stable, end-user machines lacked sufficient bandwidth and graphics power, and no compelling application appeared to drive a non-gaming consumer market for 3D.

Chapter 5

Constellation: Linguistic Semantic Networks

Constellation is a visualization system for the results of queries from the MindNet natural language semantic network. Constellation is targeted at helping MindNet's creators and users refine their algorithms through plausibility checking, as opposed to understanding the structure of language. Section 5.1 contains a full explanation of our chosen task.

We designed a special-purpose graph layout algorithm that exploits higher-level structure in addition to the basic node and edge connectivity. Our spatial layout prioritizes the creation of a semantic space to encode plausibility instead of traditional graph drawing metrics such as minimizing edge crossings, as covered in section 5.2.

Section 5.3 discusses our use of several perceptual channels both to minimize the visual influence of edge crossings and to emphasize highlighted constellations of nodes and edges. Section 5.4 outlines our navigation and interaction approaches, including a new *pie flipper* interaction technique that exploits a scrolling mouse for selecting instances of a constellation category. We cover implementation in Section 5.4, and the chapter concludes with a discussion of the results and outcomes of the project in section 5.6.

5.1 The Linguistic Plausibility-Checking Task

An explicit goal of the Constellation project was help a target group of people carry out a particular task more effectively, as opposed to finding a group of people with a problem that was a good match with a particular preconceived visualization solution. We followed a user-centered design approach as much as possible given the time constraints of our user community, who were quite available at the beginning and middle of the process. We conducted several preliminary interviews to determine the main design goals, and obtained detailed feedback that guided the evolution of several paper and software prototypes. Involvement by our target users dropped off towards the end of the design process when their attention was diverted to a new phase of their project. This situation is relatively common in user-centered design, and we relied on information gleaned from previous interactions. Our target user community was extremely small: a few computational linguists working on the MindNet system in the Natural Language Processing group at Microsoft Research.

5.1.1 The MindNet Semantic Network

MindNet is a system that constructs a large semantic network by parsing the text of machine-readable dictionaries and encyclopedias [DVR93, RDV98]. Its possible applications include grammar checking, intelligent agent help systems, machine translation, and common-sense reasoning.

The MindNet parsing process turns a dictionary or encyclopedia entry sentence into a small **definition graph** of roughly one dozen nodes. The nodes represent **word senses**: a natural language word may have several meanings depending on context, for instance “bank” as “financial institution” or “side of a river”. MindNet distinguishes between these word senses by adding a numerical suffix and treats them as separate nodes. Figure 5.1 shows the parsed definition graph for one of the senses of the word KANGAROO. The original English sentence is:

> KANGAROO: Any of various herbivorous marsupials of the family Macropodidae of Australia and adjacent islands, having short forelimbs, large hind limbs adapted for leaping, and a long, tapered tail.

The links represent directed labelled relations between words, such as `is-a`, `part-of`, or `modifier`. Parts of the sentence are parsed correctly, for instance the `is-a` relation between KANGAROO and MARSUPIAL, and the `modifier` relation between MARSUPIAL and HERBIVOROUS. However, AUSTRALIA is attached to MACROPODIDAE, the Latin name for the species, instead of the phrase ADJACENT ISLANDS. This kind of errors is one example of how the current MindNet algorithms could use refinement.

Two definition graphs that share a node can be combined into a larger graph, which can be further enlarged by incorporating other definition graphs with shared nodes. The unification process ultimately results in a huge semantic network that can contain millions of nodes.

The previously existing MindNet software interface to a path query shown in Figure 5.2 provided a textual view of both the paths and the headwords of the definition graphs used in the computation. The investigating linguists could click on any of those headwords to open a separate popup window to see a definition graph, as shown in Figure 5.1. This approach is problematic because several relationships were hard to understand: the relationship between one path and another, between a path and its constituent definitions, and between the definitions of different paths. For instance, since each path is listed separately, the only way to tell that two paths have shared subpaths is to read the individual words and cognitively compare them. To judge whether a path was plausible, the users had to click on several of the definitions to bring up individual definition windows, and then manually flip between many windows. The resulting cognitive load was extremely high since the task entailed comparison of currently seen items to previously seen items, which is much more difficult than side-by-side comparison.

5.1.3 Visualization Requirements

The MindNet developers expressed a desire to see an integrated view of the query results where paths were shown in the context of all definition graph words used in the path computation. The Constellation system was designed to be a special-purpose algorithm debugging tool. Although the input dataset consists of the relationships between English words, Constellation is not intended to shed light on the structure of the English language per se, since the linguists are quite familiar with that. Some of the other characteristics that we discovered in our preliminary interviews provided guidance for our iterative design process:

- **Ordering:** The returned paths have an explicit importance ranking, their plausibility weight. Since definition graphs are associated with paths, they too can be ranked. Moreover, after the first one or two highly ranked paths there is usually a sharp dropoff in the path weights. Long paths are often, but not always, low-ranking.

- **Dataset size:** Although each path usually contains fewer than ten words, the number of associated definition graphs can range from one to dozens. The users typically request the best ten or fifty paths. Thus, the total number of words returned in a query result ranges from a few hundred to a few thousand. That the entire MindNet semantic network has millions of words is irrelevant, since we are interested in only the set of words returned as the result of the query.

- **Sublinear growth:** The number of relevant items does not increase linearly with the number of requested paths, since additional low-weighted paths tend to share subpaths and definitions with higher-weighted paths.

- **Source and sink clusters:** Although the path query sent to MindNet consists of two *words*, the returned paths have *word senses* as starting and ending points, so there are multiple sources and multiple sinks. For instance, the top 10 paths between KANGAROO and TAIL use two different word senses of KANGAROO as sources and three word senses of TAIL as sinks.

- **English conventions:** Our target users are used to reading definitions in the format shown in Figure 5.1, which reflects the convention that English is typically read from left to right and from top to bottom. Many of the more exotic layouts, such as the radial displays of the Visual Thesaurus [Des], would not only violate this convention but also be unfamiliar to our target users.

- **Relation types:** MindNet uses two dozen kinds of labelled relations between words. Although this is a somewhat unwieldy number, only a small number of relation types are used frequently. With the help of the linguists, we binned them into eight categories: seven specific relation types, and an eighth generic category for all others.

- **Label legibility:** The plausibility-checking task is extremely reading-intensive: the linguists can make little progress without reading the word sense information in the graph node labels.

5.2 Spatial Layout

The query results returned from MindNet can be interpreted as a single medium-sized directed graph, ranging from a few hundred to a few thousand nodes. We created a spatial layout algorithm that visually encoded domain-specific features, as opposed to the usual approach of using spatial position to minimize false attachments. Our novel layout algorithm uses a curvilinear grid as the backbone for path layout, and attaches definition graphs to words on a path. Our final layout resulted from several iterations of working software prototypes, since we sought to balance the goal of visually communicating the maximum semantic content with that of providing reasonable information density.

5.2.1 Spatial Position

In most traditional graph drawing systems, spatial position bears most of the perceptual burden, and interaction is used simply for basic navigation. There are several standard constraints on spatial positioning, including crossing avoidance, bend minimization, and edge length minimization. The various classes of layout methods are rarely optimized for all three: for instance, force-directed layouts focus on edge length minimization and pay short shrift to crossing avoidance. In Constellation, we take a novel approach to the

Figure 5.3: **Traditional layouts avoid crossings to prevent false attachments. Left:** Node-edge crossings lead to ambiguity, where it is not clear if A is connected to C and B is merely in the way, or if A and B are connected as are B and C. **Right:** Edge-edge crossings create visually salient "X" artifacts that draw the viewer's attention from the important aspects of the graph structure.

edge crossing problem. Many of the traditional methods have the need to minimize edge crossings as one of the major constraints on spatial positioning, so as to avoid the visual impression of attachments that do not reflect the true structure of the dataset. Figure 5.3 shows two perceptual problems caused by false attachment with crossings: ambiguity when links may pass underneath nodes, and distracting visual artifacts at edge-edge crossings that divert the viewer's attention from the important graph structure.

However, a key insight from the information visualization literature is that spatial position is the strongest perceptual channel. Instead of "wasting" the power of spatial position by simply avoiding false attachments, we use position to visually encode the domain-specific attribute of plausibility, as shown in Figure 5.4. Our custom layout algorithm uses the high-level structures of paths and definition graphs to make node placement decisions. Even though both the paths and the definition graphs are a subset of the global semantic network, they play extremely different roles in analyzing the behavior of the MindNet program. We thus wanted them to be easily distinguishable in our visualization.

Our algorithm results in many crossings for the long-distance edges between all instances of a shared word. We avoid false attachments by using several other perceptual channels in concert to create dynamically changeable foreground and background visual layers. The user can interactively explore highlighted subsets (constellations) of the graph while retaining the context of the entire dataset. Figure 5.5 shows that we can avoid the *perception* of false attachments with a combination of interaction and additional perceptual channels, instead of relying on spatial position to bear the entire perceptual burden.

We chose to lay out the graph in 2D space instead of 3D space for two main reasons: the relatively modest size of the graph (under five thousand nodes), and the extremely strong task dependence on label reading.

5.2.2 Paths

Our graph layout algorithm depends on the domain-specific elements of paths and definition graphs. The paths returned by MindNet are used by our layout algorithm to create a skeleton framework, around which

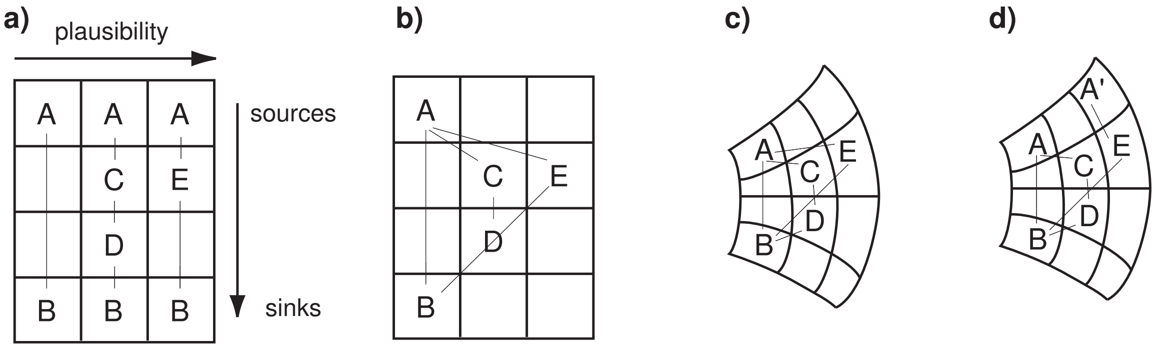

Figure 5.6: **Paths in grids. (a)** The base grid has a left to right plausibility flow between paths and a vertical flow within each path. **(b)** Consolidating shared items leads to the false attachment between D and the B-E link. **(c)** A curvilinear grid solves the colinearity problem. **(d)** Multiple endpoints are also accommodated by the curvilinear grid.

the definition graphs are inserted.

The broad layout parameters are based on the two orderings returned directly by MindNet: horizontal flow is derived from the plausibility ordering between the paths, and vertical flow is based on the internal ordering of words within a path, with the source on top and the sink on the bottom. Figure 5.6 (a) shows a rectangular grid with this ordering, using a simple example dataset. The number of grid segments horizontally is simply the number of paths. The number of vertical segments is the hop count N of the longest path in the dataset: N = 4 in the figure. Paths with fewer than N nodes will have some empty internal vertical segments, but the source and sink nodes are always laid out in the first and final spots.

However, this layout draws the same node in more than one place, which hides important connectivity information. Figure 5.6 (b) shows the same grid after we have consolidated shared nodes. If a **pathword** appears in more than one path, it is drawn in only the band of the most plausible path.

The problem with this layout is that the line between E and B is colinear with the line between D and B, such that it is hard to tell whether E is connected to D or B. These false attachments are unavoidable if nodes are laid out on rectangular grids and connected by links drawn as straight lines. The solution is to either draw links as curved lines or lay out nodes along curves instead of along straight lines. We chose the latter approach for the two reasons of perceptual simplicity and computational efficiency. People can more easily perceive a connection between two items connected by a straight line than two items connected by a curve, in keeping with the Gestalt principle of good continuation. Also, drawing curved lines is much more computationally expensive than drawing straight lines, since graphics libraries must draw a single curved line as a piecewise-linear approximation using many short straight segments. Line drawing happens much more frequently than layout in our system. Figure 5.6 (c) shows that a curvilinear grid eliminates the colinearity

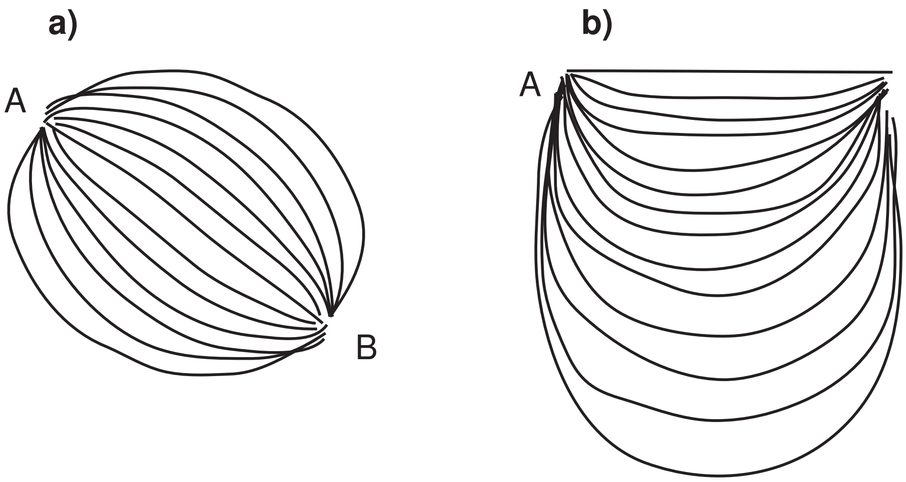

Figure 5.7: **Bad approaches.** **(a)** The middle path is the shortest in a diagonal layout. **(b)** In the top-top layout the display area has the opposite aspect ratio than a standard monitor.

problem.

The most important path will still be vertical, and highly ranked paths will be only slightly curved. The least important paths are both the most distorted and the longest. There are both perceptual and practical reasons for this choice. The shortest and straightest path will appear most perceptually important in accordance with its high weight by the Gestalt principle of proximity. Practically, the less plausible paths often have more hops than highly ranked ones, so it can be useful to have a greater distance along which to place them.

The example dataset previously shown is simpler than a real query result, which has multiple sources and sinks. (Recall that although the submitted query has a single source word and single destination word, the query results have word *senses* as endpoints.) Figure 5.6 (d) shows an example where the final path has a different source than the others, which is easily accommodated by the curvilinear grid.

5.2.3 Curvilinear Grid

There are many possible ways to construct a grid of bands and segments. We considered other possible choices for the locations of the path endpoints, but ruled them out because of undesirable perceptual or practical considerations. The diagonal layout in 5.7 (a) must either have a mid-ranked path as the shortest one or introduce some nonobvious ordering of the paths. The top-top approach of 5.7 (b) leads to a display area taller than it is wide when there are a large number of paths, which is the opposite of the standard aspect ratio of a computer monitor.

Figure 5.8 shows the particular curvilinear coordinate system that we chose, created by intersecting a

More experimentation led to radius settings of $r_j = 3\log(j+4)+2$. Substituting the value of $x^2 = r^2 - y^2$ into the parabola equation from above leads to

$$y = f + g(r^2 - y^2)$$
$$y = f + gr^2 - gy^2$$
$$gy^2 + y - f - gr^2 = 0$$

We found a simple analytical solution using Mathematica [Wol91]:

$$y = \frac{-1 + \sqrt{1 + 4fg + 4g^2r^2}}{2g} \tag{5.4}$$

5.2.4 Associating Definition Graphs with Pathwords

Paths are the backbone of our layout algorithm: each unique pathword is laid out in its own curvilinear grid cell. We fill in the definition graphs by attaching each of them to one of the pathwords. MindNet provides an explicit association in its returned query result between a given path and all the definition graphs used in its computation. For instance, the fifth path in Figure 5.2 (page 90) is explicitly associated with 23 definition graphs. We assign definition graphs that appear in multiple paths to the most plausible one: for instance, KANGAROO100 appears in paths 1, 2, 4, and 5, but is assigned to path 1.

In our layout algorithm we further assign each definition graph to a single word in that path. When the headword for the definition graph is the same as the pathword, the assignment is obvious. The more common case is that some leafword in the definition graph appears on the path. For instance, the definition of TAPIR contains the word SHORT, which is a pathword in the fifth path.

Pathwords that appear on more than one path can have definition graphs from all of them associated with it. We call the combination of a pathword and all its associated definition graphs a **path segment**.

5.2.5 Path Segment Layout

Although the placement grid is curvilinear, our drawing algorithms uses rectilinear boxes anchored at the upper left corner of the grid cell. The box allocated to each path segment is drawn in tan, as shown in Figure 5.9. The pathword itself is drawn at the top of the tan box. If a pathword has been assigned its own definition graph, as in the left side of the figure, that is also drawn in the tan section of the box. If there are other definition graphs assigned to that path segment, each of them is enclosed in a green box nested within the tan pathword box. The right of Figure 5.9 shows a path segment where the pathword is not itself defined, but has

5.2.7.1 Path Segment Elision

The first optimization involves avoiding unnecessary path segments. Some path segments have no associated definition graphs at all, containing only a single pathword. That word is guaranteed to occur in one of the definition graphs in either the previous or the next path segment. In this case we elide the singleton path segment, so that the grid cell can be used more productively. The drawn links between the words in the path will still visually indicate a path, which is more comprehensible without the extra visual step of following links to the singleton pathword. Comparing the top left of Figure 5.13 to Figure 5.12 shows the information density improvement of 40 words due to this optimization.

5.2.7.2 Horizontal Space

An empty horizontal band is a visual indication that every pathword in that path is shared with previous paths. The resulting gaps are not critical for the plausibility-checking task, and removing these horizontal gaps is a straightforward algorithmic improvement. The top right of Figure 5.13 shows the 20-word improvement obtained from removing the gap between bands 7 and 10. The new outermost band is 8, which has more horizontal room to draw words than the former outermost band 10 did. Moreover, the grid is more compact so the global overview viewpoint can be zoomed in slightly more, up to the cyan frame.

5.2.7.3 Vertical Space

The algorithm for using vertical space more effectively is somewhat more complicated. Some grid boxes are totally empty, whereas others are devoted to path segments packed with so many associated words that they have to be drawn in a very small font. The second pass of our layout algorithm allocates spare vertical space to overfull boxes so that they can expand. A box is a candidate for expansion if any word in it is drawn at less than the maximum font size. Boxes can only expand into empty grid cells directly above or below their horizontal extent. Because the grid is curvilinear, this is not as simple as checking in the band of the candidate box: non-empty boxes from other bands might also impede its growth. The expansion check is carried out band by band, starting with the most plausible one, so that more plausible boxes have priority. If two boxes from the same band could both potentially expand into an empty grid cell, the space is split between them. The bottom of Figure 5.13 shows the resulting grid of boxes after the vertical expansion pass. Over 90 more words are now readable, yet the semantically important path bands are preserved. The total information density difference between the base and optimized layouts is considerable: from 20 legible words to over 170.

Although the segments no longer necessarily lie on parabolas, we have not observed a problem with false attachments between pathwords. The fact that the layout was originally based on a curvilinear grid is not particularly visible after the expansion phase, nor should it be. Our goal is that the user perceive a structured space that reflects the left-to-right importance gradient and the connectivity between the paths. The curvilinear grid is simply a mechanism for avoiding false attachments within that structured space.

5.2.7.4 Aspect Ratio

Another information density improvement involves the aspect ratio of the display window. The example curvilinear grid of intersecting parabolas and circles shown in Figure 5.8 is somewhat distorted. Figure 5.13 shows even more distortion, since the horizontal and vertical bounds were set so that the original grid exactly fills the display window. We maintained the same aspect ratio of the pre-compression limits among all three rows of the image for easy comparison. Subsequent figures such as 5.14 (page 108) have even greater information density because the vertical and horizontal bounds were set after the compression pass.

5.2.7.5 Alternatives

We also considered the alternative of a global optimization by dynamically adapting the parameters of the parabola and circle families on a per-query basis. We rejected that choice because it would result in a highly irregular grid, which is not compatible with our goal of a structured space that shows semantic information via the concentric band spacing.

5.2.8 Graph-Theoretic Description

Our description of the layout algorithm has thus far been in terms of the domain-specific structures of paths and definition graphs. We can also describe the problem in more abstract graph-theoretic terms. The query results from MindNet form a directed graph of between a few hundred and a few thousand nodes. The observed edge density has been $|E| <= 3|V|$. Each path is a linear directed graph: i nodes connected by links, where each node has indegree and outdegree 1, except that the source node has no incoming link and the sink node has no outgoing link. Observed values for i range between 2 and about 8. The number of paths is explicitly requested by the user, and typical values are 10 and 50. These linear graphs can be combined based on shared nodes, but there is no guarantee that the result will be a single connected component.

The nodes in this combined graph are the **path segments**, which are drawn as variably-sized tan boxes with the pathword label in the upper left corner. These level 0 nodes can contain two levels of nested subgraphs. The first nesting level has a simple structure: there are j level 1 subnodes, where j varies between 1 and about 20. Each level 1 subnode corresponds to a single definition graph and is drawn as a green box of nonuniform size, with a headword label in the upper left corner. These nodes are all drawn nested within the boundaries of the tan level 0 box. Each of them has indegree 1 and outdegree 0, connected to the level 0 node but not to each other.

Finally, each of these level 1 nodes contains an entire nested subgraph, the definition graph itself, where each of the k level 2 nodes corresponds to a word in the definition. Observed values for k range between 2 and 20. These level 2 nodes are drawn nested inside the green box of their parent level 1 node. The level 2 nodes of a definition graph subgraph are a single connected component, and the node placement within the rectangular allotted space depends on the link structure of that subgraph. At least one level 2 (word) node per level 1 (definition) node is connected to the enclosing level 0 (pathword) node.

The base graph layout algorithm involves only the level 0 path segment nodes and the links between them. The second pass for maximizing information density tries to optimize the size of the level 0 enclosing node based on the size of the combined level 1 and 2 subgraphs.

In the Constellation layout algorithm, we completely ignore the set of links between a level 2 word node and all other nodes that have the same label. However, these links are always drawn. This approach is similar to the H3 layout algorithm, where the set of links that do not appear in the computed spanning tree does not affect the node layout decision. The difference lies in the drawing: in H3, these links are drawn only on demand, and usually only a subset of them are drawn at once.

5.2.9 Text Layout

The boxes allocated for words determine the font size that can be used to draw their labels. We always use a canonical stand-in word for the font size computation instead of the actual character string. (We discuss this design choice in detail in Section 6.1.1 on page 123.) Our stand-in word "Etaoinshrd100" is made from the first ten most commonly occurring characters in the English language combined with a word sense number. The correct distribution is important since we use a variable width font. The length of the stand-in was chosen empirically.

When the real label requires more horizontal room than is available in the box, we elide it to fit. For instance, the top left of Figure 5.14 shows an entire dataset at the global overview level, and the label "Old_English_Sheepdog100" in the lower left path segment is drawn as "Old_English_sh.100". The traditional three dots used to signal ellipsis takes more horizontal space than most single characters, so we instead use a single dot.

5.2.10 Adaptive Segment Division

Constellation is optimized for three viewing levels: a global view for inter-path relationships, a local view for reading individual definition graphs, and an intermediate view for associations within path segments. These differing emphases are all accomplished by making the amount of vertical space devoted to classes of words depend on the current zoom level. This adaptive word layout is a form of continuous multiscale navigation, albeit a less drastic one than the extreme approach taken in the Pad++ system [BH94].

The overview level is optimized for showing global path structure, so that pathwords and headwords are emphasized at the expense of leafwords. That is, a relatively large part of the available vertical box space is devoted to the pathword, then much of the remaining goes to the headwords, and finally all the leafwords are then fit into a small amount of vertical space. In cases of extreme vertical crowding, leafword boxes are still drawn but the text is omitted completely, as shown in the inset of the top left of Figure 5.14.

An earlier version of the system would suddenly switch from omitting leafwords to drawing them at the smallest font level possible, which resulted in a visually obtrusive jump during zooming because of the sudden appearance of large amounts of dark pixels. The top right of Figure 5.14 shows the intermediate greeking[1] state in the inset that is built into the final version of Constellation, where we draw one-pixel high black lines to visually smooth the transition between total omission and text drawing with the smallest font. We always draw the rectilinear ladder links, since when they are not highlighted they are visually unobtrusive and when they are highlighted they are explicitly intended to stand out.

The local viewing level was optimized for easy reading of definition graphs when zoomed in, and the allocation of vertical space is more equal between headwords and leafwords. The bottom left of Figure 5.14 shows that at high enough zoom levels, enough horizontal room is available to draw the full label for every word. The aspect ratio of the window is accordingly quite different from the equivalent area in the inset above it.

The path segment viewing level is optimized for showing the attachments between definition graphs and pathwords. In this intermediate stage between the global and local view, the size of headwords is close to that of pathwords. The bottom right of Figure 5.14 illustrates that framing an entire path segment with many associated definitions in the window results in much of the green box space being devoted to the definition graph headwords.

The zoom level is taken into account in the layout of words in a path segment. Since the layout inside a path segment changes when the zoom level does, it must be quickly computable to retain the reactive fluidity that is an important component of the visualization system. The mechanism for this proportional allocation

[1] "Greeking" is a term from the publishing world for this type of placeholder.

of space is to simply put a maximum cap on the font size but allow differential scaling in the horizontal and vertical directions. The visual result is that as the zoom level increases, the proportion of room allocated to the pathwords and headwords decreases while that of the leafwords increases. When the user is looking at a definition close up, every word in the definition graph is large enough to read in almost all cases.

5.3 Visual Encoding

The previous section describes the use of quantitative spatial position to encode plausibility and proximity to encode association. We also allow exploration of the dataset through the selective highlighting of **constellations** of boxes and edges. We carefully chose perceptual channels so that information is never hidden but highlighted constellations are easily discernible.

5.3.1 Constellations

There are four constellation types: paths, definition graphs, words, and relation types.

A **path constellation** highlights every word on a path and the links between them, as in the top left of Figure 5.15. In path constellations, all other versions of a highlighted word have subtle highlighting on their label boxes, but we do not explicitly emphasize the connecting links between master and proxy versions of the word. The relationship between a path and its constituent definition graphs can be quite complex. The top left of Figure 5.14 shows the constellation for the path $\{$kangaroo103 $\rightarrow$ animal109 $\rightarrow$ tail101$\}$, which visually reflects that the connection between ANIMAL109 and TAIL101 is due to a derivation by MindNet involving several definition graphs.

A **definition graph constellation** highlights every word in a definition graph, the local axis-aligned links between those words, the long-distance slanted proxy links between every instance of those words, and the enclosing box. The top right of Figure 5.15 shows an example. A **word constellation** highlights a single word, all words directly connected to it via a relation, and the links in between, as in the bottom of Figure 5.15. If any of the highlighted words in a word or definition graph constellation appears in more than one place, every instance of it is highlighted, as are the links between all versions of that word. The final constellation category, **relation type**, highlights only the lines representing the relations of a particular category. Figure 5.16 shows several examples. Also, many additional figures throughout this chapter show constellations.

Figures 5.11 and 6.2 (page 125) show previous versions of our system that implemented constellation emphasis by simply showing and hiding sets of nodes and edges. The danger of such filtering is that it introduces hidden state: users can easily forget the exact choices they made in the past that affect the current display. They might then reach a conclusion that is unwarranted given the true characteristics of the data, by drawing inferences from a subset rather than all of the data. In the final version of our system, user navigation is the only reason that all the information might not visible at all times. We solved the problem of visual clutter by the careful use of several perceptual channels to distinguish between emphasized and unemphasized information.

5.3.2 Perceptual Channels

Although no other perceptual channel alone is as salient as spatial position, combining several of them has proved to be highly effective at creating visual popout to distinguish a foreground from a background visual layer [Tuf91]. The background layer with its many edge crossings is visible at all times for context, but is unobtrusive since the background boxes and lines have low saturation and their brightness is quite similar to that of the background color. We emphasize the foreground layer by increasing both saturation and brightness. In the case of lines, we also increase the size because hue differences in wide lines are much more discriminable than in the unhighlighted narrow ones.

The colored text background boxes inside the path segment boxes (as in Figure 5.9) use grouping and enclosure to encode the hierarchical relationship between pathwords and definition graphs. These boxes also provide a colored area large enough for effective hue discrimination and maximize the legibility of the black label text. The long slanted lines between master and proxy instances of the same word sense encode association with a connection cue. The orientation of a line is an additional perceptual cue for an additional orthogonal layer, since all local definition links are axis-aligned and only long-distance proxy edges between shared words are slanted.

Finally, we use hue as a nominal variable, to distinguish between the types of enclosure boxes and the types of relation lines. Each of the eight relation types is color coded with with hues 45 degrees apart on the HSB color wheel, whereas the three hues for the enclosure boxes (tan, green, and blue) were empirically chosen to complement them.

The Constellation color scheme was designed by Guimbretière. He drew heavily on ideas from Reynolds [Rey94], who presented a set of color palettes to improve the legibility of air traffic control displays. We summarize the guidelines used to construct the color palette in Table 5.1:

- **Conspicuity:** The luminance contrast between an object and its background is the most important factor in conspicuity, which is further reinforced if the color used for an object is more saturated than its background. The similarity of brightness levels between the background and the unhighlighted information helps attenuate the visual clutter generated by edges. Proxy words are rendered at a lower

5.4.1 Interactive Visual Emphasis

The previous section discusses the use of multiple perceptual channels to bring a particular subset of the data to the emphasized foreground visual layer. This interactive visual emphasis is similar in spirit to the dynamic queries of previous information visualization systems such as FilmFinder [AS94].

5.4.1.1 Pie Flipper

An early prototype of our system allowed the user to flip through instances of constellation categories by hitting keyboard keys. Our users found that it was difficult to keep track of which key would do what. Guimbretière designed an interface that would allow them to choose between possibilities without the cognitive burden of remembering. Some kind of menu is the obvious solution to this problem.

He chose to use the hardware affordances of the now-common scrolling mouse: a mouse with a small wheel between the main buttons that offers an additional linear control channel independent of the x-y positioning. Our user community is small enough that we can simply assume that they all have such a mouse.

In the current version, the user brings up his new **pie flipper** interface by holding down the right mouse button, which triggers the translucent radial popup display shown in Figure 5.17.

Holding a mouse button down and dragging the cursor into a slice picks a category type, and then scrolling the wheel (with the button still held down) selects instances in that category. If the user drags the mouse over to a different radial slice, the constellation category changes accordingly. When the user releases the mouse button, the radial display disappears and the flipping operation is terminated, leaving the last-chosen constellation highlighted. The scrollwheel can be used both to quickly spin past many choices and to flip between constellations one by one using the subtle detents on the wheel. Visual feedback is provided by both the selective highlighting visible in the main window through the translucent popup, and auxiliary information in a lower status bar. If the initial mouse click is over a word instead of the background, then the word and definition constellation slices act as toggles for that word instead of flipping through all possibilities.

The pie flipper is similar to a pie menu [CHWS88] in that it has a pie chart layout on a popup menu, but it does not directly trigger an action. Instead, it allows the user to temporarily enter a mode that controls their category choice. Mode errors are minimized because of the sensory feedback of actively holding down the mouse button [SKB92]. Radial pie menus allow faster selection than linear ones [CHWS88], and the translucent popup provides a minimal screenspace footprint. Another reason that he chose to use a popup menu instead of a fixed menu location at the top of the window was so that constellation flipping would be extremely lightweight and not require any distracting mouse motion.

5.5 Implementation

Constellation is implemented in C++, using OpenGL for drawing. Since MindNet runs only under Microsoft Windows, Constellation was also developed under Windows.

We implemented several optimizations to speed up the interaction. We avoid the slower immediate-mode drawing in favor of a using single cached large display lists whenever possible, for example when the user pans or triggers the transparent popup pie flipper. When user actions change the appearance of the window, for instance through selective highlighting, then we must create a new display list. The most expensive operations are zooming or changing the window size: these operations require recomputing the path segment spatial layout in addition to a redraw. The segment layout recomputation is necessary since the relative allocation of space on the screen inside the boxes depends on the zoom level. However, the curvilinear grid layout and vertical cell expansion happens only once per dataset.

With a sufficiently fast graphics card and PC, the frame rate remains interactive even in the worst case of redraw plus replace, but the optimizations result in smoother interaction in the other cases. We did not spend a great deal of time optimizing the drawing, instead focusing our attention on the iterative design of the rest of the system. This allocation of resources seemed reasonable given that our very small user community of researchers is quite well funded, and graphics card price and performance continue to improve rapidly.

5.6 Results

We first cover the design tradeoffs of the project, then present the visual appearance of example datasets. Finally, we discuss the reasons for the project outcome, since Constellation is not in active use by our small target audience of computational linguists.

5.6.1 Discussion

Our final layout algorithm is the result of many iterations as we explored the tradeoff between legibility and the semantic use of space on a finite resolution display. At the former extreme, we could tile the window with a rectangular grid containing 300 words, but there would be no spatial encoding whatsoever. At the latter extreme, a very strict spatial encoding (as in Figures 5.12 or 5.13) would allow an exact encoding of the desired attributes, but vast amounts of navigation would be required to do much reading because of low information density. Our horizontal plausibility gradient is a middle ground where more important words are allocated more room in the overview position. The transformations from the sparse to the dense grid work well because they preserve the ordinality of the bands and pathwords, which is our main concern for this task.

many information visualization systems. The interactive visual emphasis capability is as fundamental to the dataset exploration as the interactive navigation, and these interactions are as important as the base spatial layout for understanding the full dataset. In this chapter we have presented a very detailed analysis of our design choices by justifying them against the task requirements. We have also documented some of the ways that the current version of the software differs from previous iterations as a result of informal usability observations of the linguists working with the earlier prototypes.

Chapter 6

Discussion

In the previous chapters we have covered the design of three software systems. In this one we discuss some general points that pertain to all three. We then present some ideas for future work: first, specifics that are more tied to each individual system or problem domain, and then, higher-level thoughts about the field of information visualization. We finish with concluding remarks.

6.1 General Discussion

Four discussion points that pertain to more than one of the systems in this thesis are visual popout, hidden state, ordering encoding, and dissemination.

6.1.1 Visual Popout

In section 1.2.3 we introduced the idea of preattentive visual processing and visual popout. We now discuss the ways in which the visually salient features of all three systems match our original design goals, shown in Figure 6.1. In the H3 system, our spatial layout was primarily aimed at maximizing the size of the local neighborhood visible at any given moment. The distortion-based layout provides a smooth visual gradient between a focus area with visible detail and the areas near the fringe of the ball that show only aggregate information about the existence or nonexistence of structures. These distant points of complexity are visually salient as blobs on the fringe, which is useful when the user is navigating through an unfamiliar graph structure seeking places of possible interest. In the Planet Multicast system, we explicitly tie tunnel height to geographic distance, so that the lofted arcs of the long distance tunnels visually pop out. The salience match is good when our assumption that geographic distance is correlated with resource usage holds true. In the

6.1.2 Hidden State

In the previous section we discussed the possibility that visually salient artifacts could lead to false *positive* conclusions. The inverse problem is also possible: leading people to draw false *negative* conclusions. People have a tendency to assume that only objects that are visible exist, a phenomenon called the closed-world assumption [Lev88].[1] Researchers investigating human perception of drawn graphs have also noted this phenomenon: "The absence of any of the above features is interpreted semantically as the absence of the interpreted quality." [DC98, p. 442] The lesson that we could apply to the design of visualization systems is that hidden state can mislead the user.

Despite the dangers, many visualization systems have hidden state, including Planet Multicast and H3. In H3, all non-tree links are explicitly filtered by default, and only a subset of them are usually drawn at once. In Planet Multicast, hidden state is implicit since tunnels within the same city are never drawn. As usual, visualization system designers are faced with a tradeoff: in many cases the benefits of visual filtering are arguably worth the risks of hidden state.

In the final version of Constellation, we completely avoid hidden state. We respond to user choices by toggling the visual salience of objects between emphasized and unobtrusive, but nothing is ever completely hidden. In earlier versions we simply toggled between drawing and hiding the objects in response to user interaction. We noticed the difficulties presented by this hidden state when observing the user of the prototypes by our target users. The linguists had a tendency to forget previous mode-changing actions, and so would draw false negative conclusions since parts of the dataset had been temporarily hidden. Our more sophisticated selective emphasis scheme in the final version successfully avoided this source of confusion.

6.1.3 Ordering Encoding

The three systems each fall into a different class of semantic power with respect to the visual encoding of ordering semantics. The MBone tunnel dataset does not contain any sort of explicit visual ordering, so the geographic layout of the Planet Multicast system makes no attempt to visually communicate ordering.

In contrast, the Constellation dataset has several orderings: the high-level path structures have an explicit ordering, as do the low-level structures inside each definition graph. The path ordering is linear, and is visually encoded by horizontal band placement. The internal definition graph subgraph structure is hierarchical, and both parent-child and child-child ordering is important. To expand on the latter point, the children of a node have a specific sibling order. The related domain-specific fact is that reading the definition graph text is an important subtask, and English has very strong top-to-bottom, left-to-right reading conventions. Our design

[1] Interestingly, this assumption is much stronger for visible graphical scenes than for textual descriptions of scenes.

goal for the project was to weight most tradeoffs in favor of effective spatial semantics. The combination of these factors almost mandated a rectilinear layout.

The H3 situation falls in between these two extremes. There is one type of ordering that is part of the dataset, by the definition of a quasi-hierarchical graph: the recursive hierarchical parent-child ordering which is encoded via the spanning tree. However, that definition is agnostic with respect to sibling order: we do not have any information about a preferred order between the children of a parent node. Either a radial or a rectilinear layout could be used for this task: in graph layouts, people ascribe more importance to centrality than to periphery in a graph layout, and likewise top is considered more important than bottom [DC98]. When a dataset has both sibling and parent-child orderings, then a radial layout might not be the best choice. However, many datasets do not have an inherent sibling ordering, or the focus for a given task is the relationships *between* generations instead of *within* generations. In such cases, a rectilinear layout would show the siblings with an explicit visual order that would be arbitrarily chosen, as opposed to being implicit to the task at hand, so misleading visual artifacts could be a problem. One common default in such a case is to show siblings in alphabetical order, which is usually not semantically meaningful.

The design goal for H3 was scalability, and our radial layout is very effective at achieving high information density by exploiting the mathematics of hyperbolic geometry. Our layout technique thus fits with the goal of scaling to very large datasets, since it resulted in greater information density than previous methods by taking full advantage of the characteristics of the projected space. Although the standard finite mathematical projections (conformal and hyperbolic) of hyperbolic space are both fundamentally radial, it would also be possible to construct a rectilinear hyperbolic projection where each axis is transformed separately [HGD95, KR96]. The H3 layout algorithm that we present here is inherently radial, but possible future work includes designing a different layout for a rectilinear projection.

The H3 radial layout was designed to be well-suited for browsing large neighborhoods. One advantage of our descendant-based H3 radial ordering is that it can aid in finding the complex regions of unfamiliar structures. Our layout algorithm always places the node with the most descendants at the “pole” of the hemisphere along the same axis as the incoming link from the parent node. A simple and effective navigation strategy for finding potentially interesting complexity is to always click on the child node at the pole after a parent moves into its canonical orientation. This strategy for finding complexity fits well with another aspect of the H3 layout, namely the visual salience of aggregate information about distant points of possible interest.

Neither the H3 or the Constellation layout currently takes advantage of the cognitive principle that people infer much stronger connotations about vertical flow than horizontal flow: top is perceived to be more important than bottom, but left and right are more neutrally equal [Tve97, LJ80, DC98]. We chose a horizontal instead of a vertical flow for Constellation deliberately, in order to fit well with the standard monitor aspect ratio: because there are a much larger number of paths than of hops in a path, we draw the former horizontally. However, no such problem bars us from a future adaptation of the H3 layout so that it grows from top to bottom instead of from left to right. A top to bottom orientation would be more cognitively defensible, and moreover potential users have explicitly requested this item of future work. In the terminology of the original PARC paper [RMC91], a vertical *cone* tree orientation might be superior to our original horizontal *cam* tree orientation.

6.1.4 Dissemination

Both the H3 system and the Constellation systems feature highly non-literal visual metaphors and interaction techniques. In contrast, the Planet Multicast system is quite literal, both in terms of visual metaphor and interaction. It is by far the least sophisticated of the three systems. Perhaps the most surprising aspect of that project is the wide appeal of the resulting still images. We have received many image reprint requests, including such mass-market magazines as Wired [Wir97] and National Geographic [Car00]. Although in some cases the images were accompanied by a technical explanation of the project goals or methods, in many cases the images were intended simply to be evocative illustrations. We conjecture that literal pictures are evocative because of their immediate comprehensibility, with minimal or no explanation of the visual encoding needed. Moreover, the literal visual metaphor is matched by the literal interaction semantics, so that a still picture can tell most of the story.

We argued in Chapter 1 that much of the great promise of computer-based information visualization lies in its freedom from the shackles of real-world literality. However, one challenge with more sophisticated interaction techniques that go beyond mimicking real-world navigation is that they are often "videogenic" instead of photogenic. Documenting the look and feel of the on-screen interaction through video is an important part of the publishing process, particularly for nonliteral interaction techniques. We have done so for all three of systems described here.[2] H3 is the most extreme case: still pictures convey only a small fraction of the interactive experience. Reprint requests for H3 material have included both still pictures and video footage. Constellation is between the other two cases: a large part of its functionality can be communicated through carefully chosen individual still pictures, but understanding of its subtle multiscale interaction is difficult without seeing the video.

[2]The three accompanying videos are available in online digitized form from `http://graphics.stanford.edu/~munzner/videos.html`.

6.2 Future Work

There are many possible future directions for this work. We first discuss ideas that pertain to the three specific systems, and then step back to take a broader view of the information visualization field.

6.2.1 H3

H3 is the most general of the three systems, and has provoked many thoughts about interesting future possibilities.

6.2.1.1 Incremental Layout

H3 is scalable and highly interactive, but not incremental. An algorithm for stable incremental layout that scaled to large datasets would be very useful. It is likely that a scalable incremental layout would be hierarchical, albeit probably a somewhat different type than the spanning tree of H3.

6.2.1.2 Web Visualization for End-users

H3 might be useful as part of a system aimed at end-users browsing the web, for example as a more powerful version of a history list that shows the the connectivity of documents recently traversed by a web surfer. The breakdown of one-dimensional history lists is that only one branch is visible, so backtracking up to a previous choice and then choosing a different page results in a loss of the previously displayed context. A full graph view would solve this problem.

Another possibility is to show surfers a visual model of the web structure that will keep them from getting lost. However, the right approach to this problem is not clear. Although the hyperlink structure of the web is easy to obtain, either with web robots or by instrumenting traditional browsers, the hyperlink relationships between individual web documents is not necessarily the best substrate for a robust user mental model of the web. Surfers might benefit more from a different representation, perhaps one that more faithfully reflected the semantic information contained in those documents, or one that used collaborative filtering to rank documents according to traffic analysis. Our goal for the H3 project was not to create semantic representation of the web, but to push the scalability of methods for visualizing large quasi-hierarchical graphs. If and when such representations become available, and they can be modelled as quasi-hierarchical graphs with fewer than a few hundred thousand links, then H3 could be used to display them.

A less ambitious but still useful improvement would be to find improved heuristics for computing Web site spanning trees that were more effective at capturing authorial intent.

6.2.1.3 Visualizing the Entire Web

Visualizing a large part of the web is a daunting but appealing prospect because of its sheer size.[3] The entire web has over a billion pages, according to one estimate in early 2000.[4] In contrast, the H3 libraries were used in Site Manager for visualizing the hyperlinks between documents on medium-sized web sites with fewer than 100,000 pages. H3 will not scale to a dataset of that size because of the computational and cognitive limits discussed in Section 3.7.2. Our intention with H3 was to push the possible scale of a detail view, which is complementary to the efforts in abstraction and level of detail that will be necessary to visualize such an enormous dataset.

Some interesting attempts have been made to create overviews through automatic abstraction of graphs [KLRZ94] and trees [HMM$^+$99]. Bray proposed the abstraction of a web site as a higher level unit for web visualization [Bra96], as opposed to the lower level approach of visualizing the hyperlinks between individual pages. We conjecture that the web is large enough now that an even higher-level abstraction would be useful: a global overview might be possible by showing a core "backbone" of the most popular sites. Visualizing the entire web might be feasible with three levels of detail: backbone, sites, and individual web pages. It would be interesting to compare such a visualization built from three recursively linked H3 views with one built using a different visualization approach.

6.2.1.4 Visualizing Changes over Time

Although the H3 layout was designed to show static graphs, it is possible to show dynamic graphs by the brute force approach of simply reloading the entire dataset. Systems specifically designed to support incremental layout have algorithms that try to minimize the differences between one layout and the next, and usually show a morph from one version to the next via an animated transition to help the user understand the changes. We conjecture that the H3 algorithm is somewhat robust in terms of generating a similar layout: adding nodes will result only in minor visual changes as hemisphere sizes grow slightly, but adding links could result in major changes if they caused a node to swap from one parent to another. Such a change would be highly disorienting without an animated morphing transition, which is not currently supported.

However, the H3 layout was designed to show structure. A completely different visual metaphor might be more effective if the goal is showing differences.

Finally, the combination of this and the previous problem leads to showing the evolution of the entire web over time, which would be a major challenge.

[3] Another difficulty with large-scale web visualization is that more and more of the web is being converted to dynamically programmed content instead of indexable static HTML pages.

[4] `http://www.inktomi.com/webmap`

6.2.2 Planet Multicast

The Planet Multicast system was a first step at the rich problem domain of visualizing network data. Linking a geographic view with other views, as in the SWIFT-3D system [KNTK99], is an obvious productive direction.

An interesting direction for future work would be carefully designing an interactive two-dimensional system that could retain some of the benefits of the 3D globe view while avoiding occlusion. Although the straightforward 2D maps used in systems such as MapNet [CH] suffer from visual clutter because of the tunnels across the Pacific, allowing the user to have interactive control over the placement of the longitudinal split on the map might sufficiently ameliorate the problem. If the groups working on the geographic determination problem make some headway [Net99], such a project could be useful for geographic network visualization.

However, the task of reducing wasted bandwidth is only partially solvable by tunnel topology information alone. We would be able to show a more complete picture if we also had information about the full unicast path underlying each tunnel. The unicast topology information would be particularly useful if annotated with capacity information for each link, and real-time congestion information about those unicast links would be even more valuable. Although none of this data is available now, if it does become feasible to to collect it in the future there would be a interesting opportunity to design a new visualization system to present this data effectively.

The challenges would include dealing with an order of magnitude more data if unicast topology information were available, and finding a good visual metaphor for traffic flows. One way to begin would be to extend the current visual metaphor by adding smaller arcs underneath the multicast tunnels arcs to represent the unicast topology and using color, linewidth, or motion coding of the arcs to show traffic. However, it could be that the combination of both more and quantitatively different data would benefit from an entirely different non-geographic metaphor.

6.2.3 Constellation

The Constellation system was the most highly focused of the three design studies. Our current implementation is a strong foundation, but further polishing could make it a more productive tool for our target users. Possibilities include adding incremental visual search capability, increased support for finding high-connectivity "hotspot" word constellations, and tighter integration with the main MindNet text views. By tackling the specialized plausibility checking task we addressed only a subset of the potential visualization needs of our target linguist users. It would be interesting to build additional visualization tools that support them in other tasks.

6.2.4 New Directions

The three design studies that we have presented are specific points in the parameter space of possible designs of interactive visualization systems for large graphs. We have thus far focused our attention on the intersection of information visualization and graph drawing. We now step back to consider the direction of the field of information visualization as a whole.

6.2.4.1 Principled Design

One of the main themes underlying this dissertation is the value of discovering and using design principles. In Chapter 1, we discussed the difficulty and importance of distilling previous experience into prescriptive advice to further the state of the art in this field. Our analysis of the three visualization systems presented here is a step in that direction. Perhaps an even more useful purpose is served by treating this work as an existence proof of the rewards of using design principles. It is no coincidence that the chronological order in which the three systems were built is directly correlated with the amount of rigor in their design. The earliest system, Planet Multicast, was quite ad hoc. The next system grew out of an idea for a nifty technique that predated this thesis. Its subsequent evolution into the final H3 system was driven by attempts to design effective solutions to the major limitations encountered in use. Finally, the most recent Constellation system was intended from the beginning to be an exercise in targeted design. The lesson that we learned from this progression is that design principles do indeed deliver — they are not merely abstract formalisms for the sake of academic hair-splitting. Rather, principles are a practical way to quickly find some of the more interesting paths through the parameter space of possible visualization designs. Although this parameter space is huge, only a tiny fraction of the possibilities are a good cognitive match to the problem at hand. A random walk usually results in a long journey through the wilderness, whereas the principles of design can act as a guide to help system builders discover, or return to, the islands of function.

6.2.4.2 Information Visualization as New Interface Paradigm

We assert that information visualization has the potential to be a major part of the future of computing. The history of computing shows a progression where a larger and larger percentage of a computer's processing power is devoted to the interface with a human. The falling price of CPU cycles due to the skyrocketing power of computer processors has financed this change in priorities. In the earliest days, CPU cycles were a rare and costly commodity, and humans devoted large amount of time to exceedingly careful preparation of the input and interpretation of the output. Eventually the more interactive command-line interface replaced the batch mode, buying human productivity at the expense of using part of the computer's processing power to support this real-time interaction.[5] The leap to the current interface paradigm of bitmapped windows and mice for 2D cursor positioning required dedicating a much larger portion of the processor to maintain this interactive two-dimensional workspace, delivering a concomitant increase in user productivity. We conjecture that the next-generation interface paradigm will be an even more extreme shift toward using processor cycles to augment human understanding. Information visualization currently occupies a somewhat specialized niche, but in the future it could become the main modality for interacting with a computer.

The amount of data to process is increasing at a rate even greater than the impressive processor speed advances described by Moore's Law. This explosion of data comes from many sources: processors with the ability to log events have become interwoven with the fabric of daily and business life; sensors have become small, cheap, and networked; and the growing feasibility of simulation allows the gathering of data about virtual rather than real-world events. Data collection is not an end of itself, but a means to the end of helping humans deal with the world. Computer-based visualization allows humans to wend their way through these mountains of data, making decisions based on understanding.

6.3 Conclusion

We have presented three software systems for the interactive exploration of large graphs.[6] We discuss these as design studies, with a detailed analysis relating the intended tasks to our spatial positioning and visual encoding choices. These three systems inhabit deliberately disparate parts of the parameter space of possible designs for graph drawing systems. Our systems can handle larger datasets than previous graph drawing systems by incorporating interaction as an essential part of the system design and by narrowing the problem scope on a domain-specific basis. We have also presented two novel algorithms for the layout and interactive navigation of large graphs. One is focused on scalability, the other on effectiveness for a highly targeted task.

Chronologically, our first foray into this problem was the 1996 Planet Multicast system, which featured the extremely literal geographic visual metaphor of arcs on a 3D globe. In several months we built a lightweight system that has proved interesting as a straightforward baseline against which we could consider the two other more ambitious projects.

The H3 system has was the project with the most visibility and the longest duration, of over two years. Our first system for visualizing hyperlink graph structure in 3D hyperbolic space predated this dissertation [MB95]. The novel H3 layout was developed in a second generation software system with the goals of achieving greater information density and handling datasets much larger than any previous graph drawing

[5] On a personal note, in the 1980's I was personally horrified at the idea of wasting expensive supercomputer CPU cycles by running a Unix shell instead of using a job control language for batch processing. I have, of course, expanded my world view since then.

[6] Videos demonstrating the systems in action are available at `http://graphics.stanford.edu/~munzner/videos.html`.

system. We have succeeded in scaling to datasets over 100 times larger than the previous work. Our methods are appropriate for the class of quasi-hierarchical graphs, where a reasonable spanning tree can be used as the backbone for layout and drawing by incorporating domain-specific information into the creation procedure. The novel H3Viewer drawing algorithm was implemented in a third-generation software system developed in response to the possibilities opened up by a layout system that could handle very large graphs. It features an adaptive drawing algorithm with a guaranteed frame rate. H3 has been incorporated into a commercial product, and has been shown to have statistically significant performance advantage for a particular web site maintenance task.

Finally, the Constellation project, which spanned eighteen months, featured the most principled process, with an emphasis on iterative user-centered design. Our main goal was to create a highly effective design for a highly targeted task that involved comprehension of a complex graph structure. Since our target audience was extremely small, the intended contribution of this project to information visualization was methodology and analysis, as opposed to widespread adoption. The novel Constellation layout algorithm focused on communicating high level domain-specific semantics instead of the traditional graph drawing goal of avoiding crossings. We avoided the perception of false attachments through a new interaction method of selective emphasis using multiple perceptual channels. We explored the tradeoffs between information density and the semantic use of spatial position in several working software prototypes before arriving at a final layout that uses subtle multiscale techniques to maximize legibility at several viewing levels.

Our analysis had several aims: to justify our particular design choices in the context of the problem, to help us distill or further elucidate design principles, and to serve as a model for subsequent work by relating new visualization techniques to a conceptual framework as an integral part of the presentation. Our methodology is relevant not only to the particular problem domain of graph drawing, but to the field of information visualization as a whole.

Bibliography

[And95] Keith Andrews. Visualizing Cyberspace: Information Visualization in the Harmony Internet Browser. In *Proceedings of Information Visualization '95 Symposium (Atlanta, GA, October 30-31, 1995)*, pages 97–104. IEEE, 1995.

[App85] Andrew W. Appel. An Efficient Program for Many-Body Simulations. *SIAM Journal on Scientific and Statistical Computing*, 6(1):85–103, 1985.

[AS94] Christopher Ahlberg and Ben Shneiderman. Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '94)*, pages 313–317, 1994.

[AS95] Eric Z. Ayers and John T. Stasko. Using Graphic History in Browsing the World Wide Web. In *Proceedings of the Fourth International World-Wide Web Conference, Boston*, 1995.

[Aug98] Björn Augustsson. XTraceroute, 1998. `http://www.dtek.chalmers.se/~d3august/xt/`.

[BC87] Richard A. Becker and William S. Cleveland. Brushing Scatterplots. *Technometrics*, 29:127–142, 1987. Reprinted in Dynamic Graphics for Data Analysis, edited by W. S. Cleveland and M. E. McGill, Chapman and Hall, New York, (1988).

[Ber81] Jacques Bertin. *Graphics and Graphic Information Processing*. Walter de Gruyter, 1981.

[Ber83] Jacques Bertin. *The Semiology of Graphics*. University of Wisconsin Press, 1983. (First edition 1967).

[BETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[BEW95] Richard A. Becker, Stephen G. Eick, and Allan R. Wilks. Visualizing Network Data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):16–28, March 1995.

[BF95] Ingo Bruß and Arne Frick. Fast Interactive 3-D Graph Visualization. In *Proceedings of Graph Drawing '95, Lecture Notes in Computer Science 1027*, pages 99–110. Springer-Verlag, 1995.

[BH94] Benjamin B. Bederson and James D. Hollan. Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. In *Proceedings of UIST '94*, pages 17–26, 1994.

[BHDH95] Lyn Bartram, Albert Ho, John Dill, and Frank Henigman. The Continuous Zoom: A Constrained Technique for Viewing and Navigating Information Spaces. In *Proceedings of UIST '95*, pages 207–215, 1995.

[BHR95] Franz J. Brandenburg, Michael Himsolt, and Christoph Rohrer. An Experimental Comparison of Force-Directed and Randomized Graph Drawing Algorithms. In *Proceedings of Graph Drawing '95, Lecture Notes in Computer Science 1027*, pages 76–87. Springer-Verlag, 1995.

[BMK95] Jim Blythe, Cathleen McGrath, and David Krackhardt. The Effect of Graph Layout on Inference from Social Network Data. In *Proceedings of Graph Drawing '95, Lecture Notes in Computer Science 1027*, pages 40–51. Springer-Verlag, 1995.

[BPV96] Luc Beaudoin, Marc-Antoine Parent, and Louis C. Vroomon. Cheops: A Compact Explorer For Complex Hierarchies. In *Proceedings of Visualization '96*, pages 87–92, 1996.

[Bra88] Franz J. Brandenburg. Nice Drawing of Graphs are Computationally Hard. In P. Gorney and M. J. Tauber, editors, *Visualization in Human-Computer Interaction, Lecture Notes in Computer Science 439*, pages 1–15. Springer-Verlag, 1988.

[Bra96] Tim Bray. Measuring the Web. In *Proceedings of the Fifth International World-Wide Web Conference, Paris*, 1996.
`http://www5conf.inria.fr/fich_html/papers/P9/Overview.html`.

[BT98] Stina Bridgeman and Roberto Tamassia. Difference Metrics for Interactive Orthogonal Graph Drawing Algorithms. In *Proceedings of Graph Drawing '98, Lecture Notes in Computer Science 1547*, pages 57–71. Springer-Verlag, 1998.

[Car00] Allen Carroll. CartoGraphic. *National Geographic*, 197(1):140, January 2000.

[Cas91] Stephen M. Casner. A Task-Analytic Approach to the Automated Design of Graphic Presentations. *ACM Transactions on Graphics*, 10(2):111–151, 1991.
`http://www.acm.org/pubs/articles/journals/tog/1991-10-2/p111-casner/p111-casner.pdf`.

[CCF95] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Three-Dimensional Pliable Surfaces: For effective presentation of visual information. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '95)*, pages 217–226, 1995.

[CCF97] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Extending Distortion Viewing from 2D to 3D. *Computer Graphics and Applications*, pages 42–51, 1997.

[CCFS95] M. Sheelagh T. Carpendale, David J. Cowperthwaite, F. David Fracchia, and Thomas Shermer. Graph Folding: Extending Detail and Context Viewing into a Tool for Subgraph Comparisons. In *Proceedings of Graph Drawing '95, Lecture Notes in Computer Science 1027*, pages 127–139. Springer-Verlag, 1995.

[CD85] D. R. Cheriton and S. E. Deering. Host Groups: A Multicast Extension for Datagram Internetworks. In *Proceedings of the Ninth Data Communications Symposium*. ACM/IEEE, September 1985.

[CE95] Kenneth C. Cox and Stephen G. Eick. 3D Displays of Internet Traffic. In *Proceedings of Information Visualization '95 Symposium (InfoVis '95)*, pages 129–131. IEEE, 1995.

[CH] K. Claffy and Brad Huffaker. MapNet.
`http://www.caida.org/tools/visualization/mapnet`.

[CH97] Robert F. Cohen and Mao Lin Huang. Online Information Visualization of Very Large Data. In *Late Breaking Hot Topics Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*, pages 53–56, 1997.

[Chr96] Bryan Christenson. WhatRoute, 1996.
`http://crash.ihug.co.nz/~bryanc`.

[CHWS88] Jack Callahan, Don Hopkins, Mark Weiser, and Ben Shneiderman. An Empirical Comparison of Pie vs. Linear Menus. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '88)*, pages 95–100, 1988.

[CK95] Jeromy Carrière and Rick Kazman. Interacting with Huge Hierarchies: Beyond Cone Trees. In *Proceedings of Information Visualization '95 Symposium (Atlanta, GA, October 30-31, 1995)*, pages 90–96. IEEE, 1995.

[Cle94] William S. Cleveland. *The Elements of Graphing Data*. Hobart Press, Summit, New Jersey, 1994. Revised edition.

[CLR93] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, McGraw-Hill, 1993.

[CM84] William S. Cleveland and Robert McGill. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *Journal of the American Statistical Association*, 79(387):531–554, September 1984.

[CM97a] Stuart K. Card and Jock Mackinlay. The Structure of the Information Visualization Design Space. In *Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*, pages 92–99, 1997.

[CM97b] Rikk Carey and Gavin Bell Chris Marrin. The Virtual Reality Modeling Language. International Standard ISO/IEC 14772-1:1997, 1997.
`http://www.web3d.org/Specifications/VRML97`.

[CMS99] Stuart Card, Jock Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999.

[Coo09] Julian Lowell Coolidge. *The Elements of Non-Euclidean Geometry*. Oxford at the Clarendon Press, 1909.

[Cox42] H.S.M. Coxeter. *Non-Euclidean Geometry*. University of Toronto Press, 1942.

[CP92] Donna Cox and Robert Patterson. Visualization Study of the NSFNET, 1992. Video.
`http://www.ncsa.uiuc.edu/SCMS/DigLib/text/technology/Visualization-Study-NSFNET-Cox.html`.

[CS88] John Horton Conway and Neil J.A. Sloane. *Sphere Packings, Lattices, and Groups*. Springer-Verlag, 1988.

[DC98] Edmund Dengler and William Cowan. Human Perception of Laid-Out Graphs. In *Proceedings of Graph Drawing '98, Lecture Notes in Computer Science 1547*, pages 441–443. Springer-Verlag, 1998. Poster.

[Des] Plumb Design. Thinkmap Visual Thesaurus.
`http://www.plumbdesign.com/thesaurus`.

[DH96] R. Davidson and D. Harel. Drawing Graphics Nicely Using Simulated Annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.

[DH98] Steve Deering and Robert Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, December 1998.

[DLW81] Douglas Dunham, John Lindgren, and David White. Creating Repeating Hyperbolic Patterns. In *Proceedings of SIGGRAPH 81*, pages 215–223, 1981.

[Döm94] Peter Dömel. Webmap – A Graphical Hypertext Navigation Tool. In *Proceedings of the Second International World-Wide Web Conference, Chicago, Illinois*, 1994.

[DVR93] William B. Dolan, Lucy Vanderwende, and Stephen Richardson. Automatically Deriving Structured Knowledge Bases from On-line Dictionaries. In *Proceedings of the Pacific Association for Computational Linguistics*, April 21–24 1993. Vancouver, British Columbia.

[FB95] George W. Furnas and Benjamin B. Bederson. Space-Scale Diagrams: Understanding Multiscale Interfaces. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '95)*, 1995.
`http://www.acm.org:82/sigs/sigchi/chi95/Electronic/documnts/papers/gwf_bdy.htm`.

[Fen89] Werner Fenchel. *Elementary Geometry in Hyperbolic Space*. Walter de Gruyter, 1989.

[FLM94] Arne Frick, Andreas Ludwig, and Heiko Mehldau. A Fast Adaptive Layout Algorithm for Undirected Graphs. In *Proceedings of Graph Drawing '94, Lecture Notes in Computer Science 894*, pages 388–403. Springer-Verlag, 1994.

[FPF88] K. Fairchild, S. Poltrok, and G. Furnas. SemNet: Three-Dimensional Graphic Representations of Large Knowledge Bases. In R. Guindon, editor, *Cognitive Science and its Applications for Human-Computer Interaction*, pages 201–233. Lawrence Erlbaum, 1988.

[Fur86] George W. Furnas. Generalized Fisheye Views. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '86)*, pages 18–23, 1986.

[FW94] Michael Frölich and Mattias Werner. Demonstration of the Interactive Graph Visualization System *da Vinci*. In *Proceedings of Graph Drawing '94, Lecture Notes in Computer Science 894*, pages 266–269. Springer-Verlag, 1994.

[FZ94] George W. Furnas and Jeff Zacks. Multitrees: Enriching and Reusing Hierarchical Structures. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '94)*, pages 330–336, 1994.

[GKNV93] Emden R. Gansner, Eleftherios Koutsofois, Stephen C. North, and Kiem-Phong Vo. A Technique for Drawing Directed Graphs. *IEEE Transactions on Software Engineering*, 19(3):214–229, March 1993.

[GM91] Charlie Gunn and Delle Maxwell. *Not Knot*. AK Peters, Ltd., Wellesley, MA, 1991. (Video).

[Gol87] Reginald G. Golledge. Environmental Cognition. In Daniel Stokols and Irwin Altman, editors, *Handbook of Environmental Psychology*, volume 1, chapter 5, pages 131–174. John Wiley & Sons, 1987.

[Gre98] Marc Green. Toward a Perceptual Science of Multidimensional Data Visualization: Bertin and Beyond, 1998.
`http://www.ergogero.com/dataviz/dviz0.html`.

[GT96] Ashima Garg and Roberto Tamassia. GIOTTO3D: A system for visualizing hierarchical structures in 3D. In Stephen North, editor, *Proceedings of Graph Drawing '96), Lecture Notes in Computer Science 1190*. Springer-Verlag, 1996.
`http://www.cs.brown.edu/cgc/papers/gt-gsvhs-97.ps.gz`.

[Gun92] Charlie Gunn. Visualizing Hyperbolic Geometry. In *Computer Graphics and Mathematics*, pages 299–313. Eurographics, Springer Verlag, 1992.

[HA99] Bernardo A. Huberman and Lada A. Adamic. Evolutionary Dynamics of the World Wide Web. *Nature*, 9 September 1999. Scientific correspondence.

[HAA+96] Mary W. Hall, Jennifer M. Anderson, Saman P. Amarasinghe, Brian R. Murphy, Shih-Wei Liao, Edouard Bugnion, and Monica S. Lam. Maximizing Multiprocessor Performance with the SUIF Compiler. *IEEE Computer*, December 1996.

[Har88] David Harel. On Visual Formalisms. *Communications of the ACM*, 31:514–530, May 1988.

[HDWB95] R.J. Hendley, N.S. Drew, A.M. Wood, and R. Beale. Narcissus: Visualizing Information. In *Proceedings of Information Visualization '95 Symposium*, pages 90–96. IEEE, 1995.

[HE97] Mao Lin Huang and Peter Eades. A Fully Animated Interactive System for Clustering and Navigating Huge Graphs. In *Proceedings of Graph Drawing '98, Lecture Notes in Computer Science 1547*, pages 374–383, 1997.

[HGD95] Volkmar Hovestadt, Oliver Gramberg, and Oliver Deussen. Hyperbolic User Interfaces for Computer Aided Architectural Design. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '95)*, 1995. short paper.
`http://www.acm.org/sigchi/chi95/Electronic/documnts/shortppr`.

[Him94] Michael Himsolt. GraphEd: A Graphical Platform for the Implementation of Graph Algorithms. In *Proceedings of Graph Drawing '94, Lecture Notes in Computer Science 894*, pages 182–193. Springer-Verlag, 1994.

[HMM+99] Ivan Herman, M. Scott Marshall, Guy Melanon, David J. Duke, Maylis Delest, and J.-P. Domenger. Skeletal Images as Visual Cues in Graph Visualization. In *Proceedings of the Joint Eurographics IEEE TCVG Symposium on Visualization (VisSym '99)*. Springer Verlag, 1999. Also available as a Report of the Centre for Mathematics and Computer Sciences, INS-9813.
`http://www.cwi.nl/~ivan/AboutMe/Publications/INS-R9813.pdf`.

[Hop97] Hugues Hoppe. View-Dependent Refinement of Progressive Meshes. In *SIGGRAPH 97 Conference Proceedings*, pages 189–198. ACM SIGGRAPH, Addison Wesley, 1997.

[Ive92] Birger Iversen. *Hyperbolic Geometry*. Cambridge University Press, 1992.

[JF98] Susanne Jul and George W. Furnas. Critical Zones in Desert Fog: Aids to Multiscale Navigation. In *Proceedings of UIST '98*, pages 97–106, 1998.

[Jon] Jerry Jongerius. VisualRoute.
`http://www.visualroute.com`.

[JS91] Brian Johnson and Ben Shneiderman. Treemaps: A Space-filling Approach to the Visualization of Hierarchical Information. In *Proceedings of IEEE Visualization '91 Conference*, pages 284–291, 1991.

[KK89] Tomihisa Kamada and Satoru Kawai. An Algorithm for Drawing General Undirected Graphs. *Information Processing Letters*, 31(1):7–15, 12 April 1989.

[KLRZ94] Doug Kimelman, Bruce Leban, Tova Roth, and Dror Zernik. Reduction of Visual Complexity in Dynamic Graphs. In *Proceedings of Graph Drawing '94, Lecture Notes in Computer Science 894*, pages 218–225. Springer-Verlag, 1994.

[KNTK99] Eleftherios E. Koutsofios, Stephen C. North, Russell Truscott, and Daniel A. Keim. Visualizing Large-Scale Telecommunication Networks and Services. In *Proceedings of Visualization '99*, pages 457–461, 1999.

[KR96] T. Alan Keahey and Reid Rivenburgh. HyperLINK: A Program for Visualizing Traversal of the WWW. In *Proceedings of the ACM Conference on Hypertext (HyperText '96)*, March 1996. Demonstration.
`http://www.acl.lanl.gov/~keahey/c3/navigate/hyperbolic.html`.

[KR97] T. Alan Keahey and Edward L. Robertson. Nonlinear magnification fields. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 51–58, 1997.

[KRB94] Karlis Kaugars, Juris Reinfelds, and Alvis Brazma. A Simple Algorithm for Drawing Large Graphs on Small Screens. In *Proceedings of Graph Drawing '94, Lecture Notes in Computer Science 894*, pages 278–281. Springer-Verlag, 1994.

[Kul61] Stefan Kulczycki. *Non-Euclidean Geometry*. Pergamon Press, 1961.

[KY93] Hideki Koike and Hirotaka Yoshihara. Fractal Approaches for Visualizing Huge Hierarchies. In *Proceedings of the 1993 IEEE Symposium on Visual Languages*, pages 55–60, 1993.

[LA94] Y.K. Leung and M.D. Apperley. A Review and Taxonomy of Distortion-Oriented Presentation Techniques. In *ACM Transactions on Computer-Human Interaction*, volume 1, pages 126–160, June 1994.

[Lev88] H.J. Levesque. Logic and the Complexity of Reasoning. *Journal of Philosophical Logic*, 17:355–389, 1988.

[LJ80] George Lakoff and Mark Johnson. *Metaphors We Live By*. University of Chicago Press, 1980.

[LNS85] Richard J. Lipton, Stephen C. North, and J.S. Sandberg. A Method for Drawing Graphs. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 153–160, June 1985.

[LR94] John Lamping and Ramana Rao. Laying Out and Visualizing Large Trees Using a Hyperbolic Space. In *Proceedings of UIST '94*, pages 13–14, 1994.

[LRP95] John Lamping, Ramana Rao, and Peter Pirolli. A Focus+Content Technique Based on Hyperbolic Geometry for Viewing Large Hierarchies. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '95)*, pages 401–408, 1995.
`http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/jl_bdy.htm`.

[Mac86a] Jock D. Mackinlay. *Automatic Design of Graphical Presentations*. PhD thesis, Stanford University, 1986.

[Mac86b] Jock D. Mackinlay. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, 5(2):111–141, 1986.

[Mar75] George E. Martin. *The Foundations of Geometry and the Non-Euclidean Plane*. Springer-Verlag, 1975.

[Mar91] Joe Marks. A Formal Specification Scheme for Network Diagrams That Facilitates Automated Design. *Journal of Visual Languages and Computing*, 2:395–414, 1991.

[MB95] Tamara Munzner and Paul Burchard. Visualizing the Structure of the World Wide Web in 3D Hyperbolic Space. In *Proceedings of the VRML '95 Symposium*, pages 33–38. ACM SIGGRAPH, 1995.

[Mes64] Herbert Meschkowski. *Noneuclidean Geometry*. Academic Press, 1964.

[MGR99] Tamara Munzner, François Guimbretière, and George Robertson. Constellation: A Visualization Tool For Linguistic Queries from MindNet. In *Proceedings of the 1999 IEEE Symposium on Information Visualization*, pages 132–135, 1999.

[MHCF96] Tamara Munzner, Eric Hoffman, K. Claffy, and Bill Fenner. Visualizing the Global Topology of the MBone. In *Proceedings of the 1996 IEEE Symposium on Information Visualization*, pages 85–92, 1996.

[MMBL95] Tamara Munzner, Daeron Meyer, Paul Burchard, and Stuart Levy. WebOOGL: Integrating 3D Graphics and the Web, 1995.
`http://www.geom.umn.edu/locate/weboogl`.

[Moe90] Sven Moen. Drawing Dynamic Trees. *IEEE Software*, pages 21–28, July 1990.

[MT93] Paul M. Mullins and Siegfried Treu. A Task-Based Cognitive Model for User-Network Interaction: Defining a Task Taxonomy to Guide the Interface Designer. *Interacting with Computers*, 5(2):139–166, 1993.

[MTB00] Julie Bauer Morrison, Barbara Tversky, and Mireille Betrancourt. Animation: Does It Facilitate Learning? In *Proceedings of Smart Graphics AAAI Spring Symposium*, pages 53–60. AAAI Press Technical Report SS-00-04, 2000.

[Mun97] Tamara Munzner. H3: Laying Out Large Directed Graphs in 3D Hyperbolic Space. In *Proceedings of the 1997 IEEE Symposium on Information Visualization*, pages 2–10, 1997.

[Mun98a] Tamara Munzner. Drawing Large Graphs with H3Viewer and Site Manager. In *Proceedings of Graph Drawing '98, Lecture Notes in Computer Science 1547*, pages 384–393. Springer-Verlag, 1998.

[Mun98b] Tamara Munzner. Exploring Large Graphs in Hyperbolic Space. *IEEE Computer Graphics and its Applications*, 18(4):18–23, July/August 1998.

[ND86] Donald A. Norman and Stephen W. Draper, editors. *User Centered System Design: New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates, 1986.

[Net99] NetGeo, 1999.
`http://www.caida.org/tools/utilities/netgeo`.

[Nie00] Jakob Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, 2000.

[Noi94] Emmanuel G. Noik. A Space of Presentation Emphasis Techniques for Visualizing Graphs. In *Proceedings of Graphics Interface '94*, pages 225–233, 1994.

[Nor95] Stephen C. North. Incremental Layout in DynaDAG. In *Proceedings of Graph Drawing '95, Lecture Notes in Computer Science 1027*, pages 409–418. Springer-Verlag, 1995. `ftp://ftp.research.att.com/dist/drawdag/dynadag.ps.gz`.

[PB94] James Pitkow and Krishna Bharat. WEBVIZ: A Tool for World-Wide Web Access Log Visualization. In *Proceedings of the First International World-Wide Web Conference, Geneva, Switzerland, May 1994*, 1994.

[PCJ95] Helen C. Purchase, Robert F. Cohen, and Murray James. Validating Graph Drawing Aesthetics. In *Proceedings of Graph Drawing '95, Lecture Notes in Computer Science 1027*, pages 435–446. Springer-Verlag, 1995.

[PF93] Ken Perlin and David Fox. Pad: An Alternative Approach to the Computer Interface. In *Proceedings of SIGGRAPH 93 (Anaheim, CA, August 1–6, 1993)*, pages 57–64, 1993. `http://galt.cs.nyu.edu/students/fox/pad.html`.

[PG92] Mark Phillips and Charlie Gunn. Visualizing Hyperbolic Space: Unusual Uses of 4x4 Matrices. In *1992 Symposium on Interactive 3D Graphics (Boston, MA, March 29 - April 1 1992)*, volume 25, pages 209–214, New York, 1992. ACM SIGGRAPH. Special issue of *Computer Graphics*.

[PLM93] Mark Phillips, Silvio Levy, and Tamara Munzner. Geomview: An Interactive Geometry Viewer. *Notices of the American Mathematical Society*, 40(8):985–988, October 1993. Computers and Mathematics Column.

[PN99] Ram Periakaruppan and Evi Nemeth. GTrace - A Graphical Traceroute Tool. In *LISA '99*, 1999.

[Pos81] Jon Postel. Internet Protocol – DARPA Internet Program Protocol Specification. RFC 791, September 1981.

[Pur97] Helen C. Purchase. Which Aesthetic has the Greatest Effect on Human Understanding? In *Proceedings of Graph Drawing '97, Lecture Notes in Computer Science 1353*, pages 248–261. Springer-Verlag, 1997.

[RC94] Ramana Rao and Stuart K. Card. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '94)*, pages 318–322, 1994.

[RCM93] George G. Robertson, Stuart K. Card, and Jock D. Mackinlay. Information Visualization using 3D Interactive Animation. *Communications of the ACM*, 36(4):57–71, April 1993.

[RCMC00] Kirsten Risden, Mary P. Czerwinski, Tamara Munzner, and Dan Cook. An Initial Examination of Ease of Use for 2D and 3D Information Visualizations of Web Content. *International Journal of Human Computer Studies*, 2000. To appear: special issue on Empirical Studies of Information Visualization.

[RDV98] Stephen D. Richardson, William B. Dolan, and Lucy Vanderwende. MindNet: Acquiring and Structuring Semantic Information from Text. In *Proceedings of COLING '98*, pages 1098–1102, 1998.

[Rei94] Steven P. Reiss. 3-D Visualization of Program Information. In *Proceedings of Graph Drawing '94, Lecture Notes in Computer Science 894*, pages 12–24. Springer-Verlag, 1994. (Extended Abstract and System Demonstration).

[Rey94] Linda Reynolds. Colour for Air Traffic Control Displays. *Displays*, 15(4):215–225, 1994.

[RKMG94] Steven F. Roth, John Kolojejchick, Joe Mattis, and Jade Goldstein. Interactive Graphics Design Using Automatic Presentation Knowledge. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '94)*, 1994. `http://www.cs.cmu.edu/Groups/sage/InteractiveDesign/InteractiveDesign.html`.

[RMC91] George Robertson, Jock Mackinlay, and Stuart Card. Cone Trees: Animated 3D Visualizations of Hierarchical Information. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '91)*, pages 189–194, 1991.

[RRGK96] Bernice E. Rogowitz, David A. Rabenhorst, John A. Gerth, and Edward B. Kalin. Visual Cues for Data Mining. In *Proceedings of the SPIE/SPSE Symposium on Electronic Imaging*, volume 2657, pages 275–301, February 1996.

[RT81] Edward S. Reingold and John S. Tilford. Tidier Drawings of Trees. *IEEE Transactions on Software Engineering*, 7(2):223–228, March 1981.

[RT96] Bernice E. Rogowitz and Lloyd A. Treinish. How Not to Lie with Visualization. *Computers In Physics*, 10(3):268–273, May/June 1996. `http://www.research.ibm.com/dx/proceedings/pravda/truevis.htm`.

[SB94] Manojit Sarkar and Marc H. Brown. Graphical Fisheye Views. *Communications of the ACM*, 37(12):73–84, December 1994.

[Sch92] Doris Schattschneider. *Visions of Symmetry: Notebooks, Periodic Drawings, and Related Work of M.C. Escher*. W.H. Freeman, 1992.

[SFB94] Maureen C. Stone, Ken Fishkin, and Eric A. Bier. The Movable Filter as a User Interface Tool. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '94)*, pages 306–312, 1994.

[Shi63] P.A. Shirokov. *A Sketch of the Fundamentals of Hyperbolic Geometry*. P. Noordhoff Ltd, Groningen, the Netherlands, 1963.

[Shn96] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualization. Technical Report CS-TR-3665,ISR-TR-96-66, University of Maryland, Department of Computer Science, 1996.

[SK97] E. B. Saff and A.B.J. Kuijlaars. Distributing Many Points on a Sphere. *The Mathematical Intelligencer*, 19(1), 1997.

[SKB92] A. Sellen, G. Kurtenbach, and W. Buxton. The Prevention of Mode Errors through Sensory Feedback. *Journal of Human Computer Interactions*, 7(2):141–164, 1992.

[SM93] R.L. Sollenberger and P. Milgram. The Effects of Stereoscopoic and Rotational Displays in the Three-Dimensional Path Tracing Task. *Human Factors*, 35(3):483–500, 1993.

[SM98] Margaret-Anne D. Storey and Hausi A. Müller. Graph Layout Adjustment Strategies. In *Proceedings of Graph Drawing '98, Lecture Notes in Computer Science 1547*, pages 487–499. Springer-Verlag, 1998.

[SR83] Kenneth J. Supowit and Edward M. Reingold. The Complexity of Drawing Trees Nicely. *Acta Informatica*, 18:377–392, 1983.

[SR96] Mike Scaife and Yvonne Rogers. External Cognition: How Do Graphical Representations Work? *J. Human-Computer Interaction Studies*, 45:185–213, 1996.

[SSTR93] Manojit Sarkar, Scott S. Snibbe, Oren J. Tversky, and Steven P. Reiss. Stretching the Rubber Sheet: A Metaphor for Viewing Large Layouts on Small Screens. In *Proceedings of UIST '93*, pages 81–91, 1993.

[Ste46] S. S. Stevens. On the Theory of Scales of Measurement. *Science*, 103(2684):677–680, 1946.

[SWFM97] Margaret-Anne D. Storey, K. Wong, F. David. Fracchia, and Hausi A. Müller. On Integrating Visualization Techniques for Effective Software Exploration. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '97)*, pages 38–45, 1997.

[TG88] Anne Triesman and S. Gormican. Feature Analysis in Early Vision: Evidence from Search Asymmetries. *Psychological Review*, 95(1):15–48, 1988.

[The98] TheBrain.
`http://www.thebrain.com`, 1998.

[Thu97] William P. Thurston. *Three-Dimensional Geometry and Topology*, volume 1. Princeton University Press, 1997.

[Too] Clever Tools. GeoBoy. `http://www.clevertools.com/products/netboys/gb.html`.

[TS92] Joel Tesler and Steve Strasnick. FSN: The 3D File System Navigator, 1992. ftp://sgi.sgi.com/sgi/fsn.

[Tuf91] Edward Tufte. *Envisioning Information*. Graphics Press, 1991.

[Tun93] Daniel Tunkelang. A Layout Algorithm for Undirected Graphs. In *Proceedings of Graph Drawing '93, ALCOM International Workshop PARIS 1993 on Graph Drawing and Topological Graph Algorithms*, 1993. `http://reports-archive.adm.cs.cmu.edu/anon/1994/CMU-CS-94-161.ps`.

[Tve92] Barbara Tversky. Distortions in Cognitive Maps. *Geoforum*, 23(2):131–138, 1992.

[Tve97] Barbara Tversky. Cognitive Principles of Graphic Displays. Technical Report FS-97-03, Reasoning with Diagrammatic Representations II (1997 AAAI Fall Symposium), November 1997.

[TW84] William P. Thurston and Jeffrey R. Weeks. The Mathematics of Three-dimensional Manifolds. *Scientific American*, pages 108–120, July 1984.

[TX94] Ioannis G. Tollis and Chunliang Xia. Drawing Telecommunication Networks. In *Proceedings of Graph Drawing '94, Lecture Notes in Computer Science 894*, pages 206–217. Springer-Verlag, 1994.

[War00] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann/Academic Press, 2000.

[Wee85] Jeffrey R. Weeks. *The Shape of Space*. Marcel Dekker, 1985.

[Wes96] Duane Wessels. Squid Web Proxy Cache, 1996. `http://www.squid-cache.org/`.

[WF96] Colin Ware and G. Frank. Evaluating Stereo and Motion Cues for Visualizing Information Nets in Three Dimensions. *ACM Transactions on Graphics*, 15(2):121–140, 1996.

[Wil97] Graham J. Wills. NicheWorks – Interactive Visualization of Very Large Graphs. In *Proceedings of Graph Drawing '97, Lecture Notes in Computer Science 1353*, pages 403–414. Springer-Verlag, 1997.

[Wil99a] Leland Wilkinson. *The Grammar of Graphics*. Springer-Verlag, 1999.

[Wil99b] Graham J. Wills. NicheWorks – Interactive Visualization of Very Large Graphs. *Journal of Computational and Graphical Statistics*, 8(3):190–212, June 1999.

[Wir97] *Wired*, 5(11):194, November 1997. Idées Fortes section.

[Wol45] Harold E. Wolfe. *Introduction to Non-Euclidean Geometry*. Dryden Press, NY, 1945.

[Wol91] Stephen Wolfram. *Mathematica: A System for Doing Mathematics by Computer*. Addison-Wesley, second edition, 1991.

[WS79] Wetherell and Shannon. Tidy Drawing of Trees. *IEEE Transactions on Software Engineering*, 5(5):514–520, 1979.

[Zha91] Jiajie Zhang. The Interaction of Internal and External Representations in a Problem Solving Task. *Proceedings of the Thirteenth Annual Conference of Cognitive Science Society*, 1991. `http://acad88.sahs.uth.tmc.edu/research/publications/cs91_IntExt.pdf`.