Cite as:
Schaffer, D. (1995). Visualizing large, loosely-structured, hierarchical information spaces. M.Sc. thesis, Department of Computer Science, University of Calgary, Calgary, Canada.

THE UNIVERSITY OF CALGARY

Visualizing Large, Loosely-Structured, Hierarchical Information Spaces

by

Douglas Gordon Schaffer

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

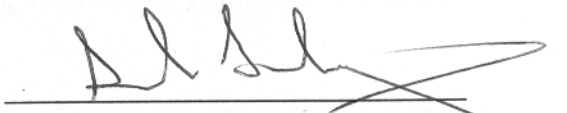
CALGARY, ALBERTA

SEPTEMBER, 1995

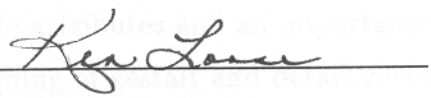
© Douglas Gordon Schaffer 1995

THE UNIVERSITY OF CALGARY
FACULTY OF GRADUATE STUDIES

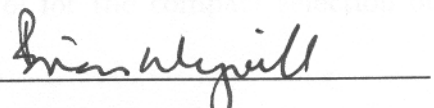
The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled *"Visualizing Large, Loosely-Structured, Hierarchical Information Spaces"* submitted by Douglas Gordon Schaffer in partial fulfillment of the requirements for the degree of Master of Science.



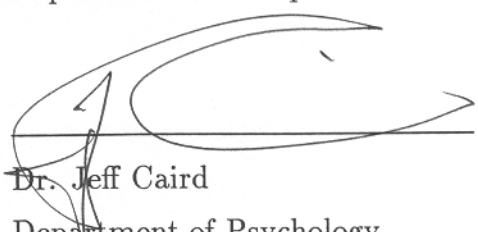
Supervisor, Dr. Saul Greenberg
Department of Computer Science



Dr. Ken Loose
Department of Computer Science



Dr. Brian Wyvill
Department of Computer Science



Dr. Jeff Caird
Department of Psychology

Date Sept 21, 1995

Abstract

The magnitude of loosely, structured data available at users' fingertips from local and distributed sources is experiencing unprecedented growth. Even modern hierarchical visualization systems are overwhelmed by scalability problems due to the explosive increase in data magnitude.

This thesis surveys the current state of the art in information visualization to determine essential visualization characteristics, leading to a set of guidelines for hierarchical visualization evaluation and design.

The guidelines become the basis for a new hierarchicial visualization strategy, illustrated and implemented by the FLEXVIEW system.

FLEXVIEW incorporates three main features of this strategy. First, the sophisticated use of emphasis and exclusion filters is applied to node attributes and an importance value to refine content quality. Second, the tight coupling of gestalt and detail views provides navigational capability and presents global trends and anomalies. Third, concepts are extended to abstracted hierarchies through automatic filter interface tuning. FLEXVIEW also introduces a *schlider* control for the compact selection of multiple, emphasized sub-ranges across a continuum.

Acknowledgements

I'd like to thank Veronica Lees for her constant support and for pickin' up the slack.

I'd like to thank Saul Greenberg for his infinite patience, support, guidance and enthusism. His dedication and perseverance helped turn drafts of coal into a thesis of dia...well, much better coal.

I'd like to thank Mark Roseman, road-trip compadre extraordinare. Thanks for being there to bounce ideas off of. And thanks for bouncing squash balls off of me.

I'd like to thank Craig Schock for lettin' me beat him at darts (occasionally). I'd also like to thank Craig for letting me beat him at pool again and again...

Thanks to Big Rock Brewery for curing that beer habit I picked up as an undergrad.

I'd like to thank the HCI/G lab cohorts, Ted, Earle, Sean, Linda, Carl et. al. for providing me with constant sanity checks and unsolicited criticism of work in progress.

Thanks Chris Thorne for dragging me away from my thesis (kicking and screaming!) to catch a great band at the Den or bring in Mexican Easter at the Ship & Anchor.

Thanks Frank and Judy Schaffer, for letting me hide in their basement to write.

Thanks to Ginger and Duck, for constantly adjusting my perspective.

I'd like to thank *Mother Nature* for more than enough lousy weather over the past four years to keep me indoors long enough to finish this work.

Thanks Eric Schenk, for letting me tap into his extraordinary LaTeX knowledge.

I'd like to thank Camille Sinanan, for helping me get organized for conferences.

I'd like to thank Moses Renert, who turned me on to the idea of spending my first few post-undergrad years living in contented financial poverty and intellectual wealth.

I'd like to thank all those who've enriched my life over the past few years.

Dedication

This thesis is dedicated to my father, Frank Schaffer. He learned the importance of education, and spent a large part of his life passing his excitement and enthusiasm about learning on to others. Pensive and thoughtful, Dad still exudes a gleeful, youthful zest for life.

This thesis is dedicated to my mother, Judy Schaffer. Mom is a wonderful, thoughtful, caring woman. She sacrificed much to be there for her children.

Mom and Dad chose to forego so many of life's luxuries, deciding instead to dedicated their efforts and time to provide a strong and nurturing environment for learning. They took a hands-on role in the activities of myself, my brother, Greg, and my sister, Sue, learning as we did. We all grew as a result. We were all very fortunate to share in this childhood.

And we continue to be.

Contents

Approval Page ii

Abstract iii

Acknowledgements iv

Dedication v

Table of Contents vi

List of Tables ix

List of Figures x

1 Introduction 1
- 1.1 Motivation: Two Problems with Information Use 1
  - 1.1.1 Too Much Information 2
  - 1.1.2 Inadequate Information Visualization 3
- 1.2 Hierarchical Information Visualization 4
- 1.3 Why Study Visualization of Hierarchies? 5
- 1.4 Outline of Thesis 5

2 Approaches for Viewing Hierarchical Information 7
- 2.1 Zoom Navigation 7
  - 2.1.1 Node-at-a-Time Text Zooming 8
  - 2.1.2 Zooming with replacement 10
  - 2.1.3 Zooming with Multiple Windows 10
- 2.2 Indented Lists with Holophrasting 12
- 2.3 Distortion and Filtering through Fisheye Views 12
- 2.4 2-D Graphical Trees 15
- 2.5 3-D Graphical Trees 17
- 2.6 Distortion through Surface Mapping 19
- 2.7 Distortion through Spatial Transformations 20
- 2.8 Space Filling 23
- 2.9 Summary 25

3 Essential Features for Hierarchical Visualization 27

3.1 Data-Oriented Presentation Characteristics 30
3.1.1 The Degree of Data Abstraction 31
3.1.2 Structure and Detail Presentation 34
3.1.3 Data Properties 38
3.1.4 Consistent Structural Visualization 39
3.2 Visualization Techniques 39
3.2.1 Graphical User Interfaces and Direct Manipulation 40
3.2.2 Dynamic Queries 41
3.2.3 Animation 42
3.2.4 Foci 43
3.2.5 3-D Graphics 43
3.2.6 Distortion 43
3.2.7 Data Encoding 44
3.2.8 Filtering 45
3.2.9 Clustering 46
3.3 A Technique-Based Look at Visualization Approaches 47
3.3.1 Visualization Approaches in General 47
3.3.2 "Zoom" Navigation 50
3.3.3 Indented Lists with Holophrasting 51
3.3.4 Distortion and Filtering through Fisheye Views 51
3.3.5 2-D Graphical Trees 52
3.3.6 3-D Graphical Trees 52
3.3.7 Distortion through Surface Mapping 53
3.3.8 Distortion through Spatial Transformations 53
3.3.9 Space Filling 54
3.4 Conclusions 55

4 A User's View of FlexView 58
4.1 Using FLEXVIEW: A Simple Scenario 59
4.2 The Three FLEXVIEW Windows 63
4.2.1 The Filter Specification Window 64
4.2.2 The Detail View Window 67
4.2.3 The Overview Window 68
4.2.4 Linking the Overview and Detail View 69
4.3 Using FLEXVIEW to Visualize a Complex Scenario 69
4.4 Conclusion/Summary 72

5 FlexView Implementation 73
5.1 The Visual Layout Algorithm 73

5.2 The Data Structure 78
5.3 Emphasis and Filtering in FLEXVIEW 80
5.4 Abstracted Hierarchies via a Node Description File 80
5.5 Summary 83

6 Evaluation 84
6.1 Visualization Guidelines Met Effectively by FLEXVIEW 85
6.1.1 Basic Information Presentation Requirements 85
6.1.2 Advanced Information Presentation Features 86
6.1.3 Facilitate Tasks 87
6.1.4 Accommodate User's Abilities 89
6.1.5 Related Issues 90
6.2 Weaknesses and Suggested Improvements 91
6.2.1 Basic Information Presentation Requirements 92
6.2.2 Advanced Information Presentation Features 93
6.2.3 Facilitate Tasks 95
6.2.4 Accommodate User's Abilities 97
6.2.5 Related Issues 98
6.3 Informal User Testing 101
6.4 Summary 103

7 Conclusion 104
7.1 Summary of Thesis 104
7.2 Original Contributions 105
7.3 Directions for Future Research 106

Bibliography 109

Appendix A Complete Node Description File with Comments 116

List of Tables

2.1 A Catalog of Visualization Approaches and Example Systems 26

3.1 Summary of Data-Oriented Presentation Characteristics. 28
3.2 Summary of Visualization Techniques. 29
3.3 Comparison of Hierarchcial Visualization Approaches. 48
3.4 Guidelines for Hierarchical Information Visualization 57

List of Figures

2.1 Navigating directories using Unix `cd` and `ls`. 8
2.2 Variable and Full-Screen Zoom views at Cluster Levels. 9
2.3 The Macintosh Finder screen demonstrating zooming with windows. . 11
2.4 The Macintosh Finder screen showing indented lists with holophrasting. 11
2.5 Flat and Fisheye views of C program. 13
2.6 BASS view of a tree, showing Fisheye DOI Filtering. 14
2.7 A simple directory tree demonstration program for Tcl/Tk. 16
2.8 CAT view of a tree, showing Fisheye view. 16
2.9 Cone and Cam Trees. 17
2.10 `fsn`, a 3D file system navigator for SGI graphics workstations. 18
2.11 The Perspective Wall. 19
2.12 CAT view of a graph, a) normal view and b) Fisheye view. 20
2.13 Initial Layout and Fisheye View of Graph. 22
2.14 Planar Graph Fisheye Views of the United States. 22
2.15 `xdu` disk structure display. 24
2.16 Treemaps representation of a hard drive with 1008 files. 24

3.1 Structural components of a strict abstracted hierarchy. 32

4.1 An example paragraph as a leaf node in a FLEXVIEW thesis hierarchy. 59
4.2 FLEXVIEW Screen Dumps of Thesis. 60
4.3 Continuing FLEXVIEW Screen Dumps of Thesis. 61
4.4 The Three Component Windows of FLEXVIEW. 63
4.5 Filtering on Multiple Sub-ranges. 65
4.6 Three examples of the Interval Emphasis control. 67
4.7 Identifying Good Candidates for File Removal. 70
4.8 Identifying Good Candidates for File Removal. 71

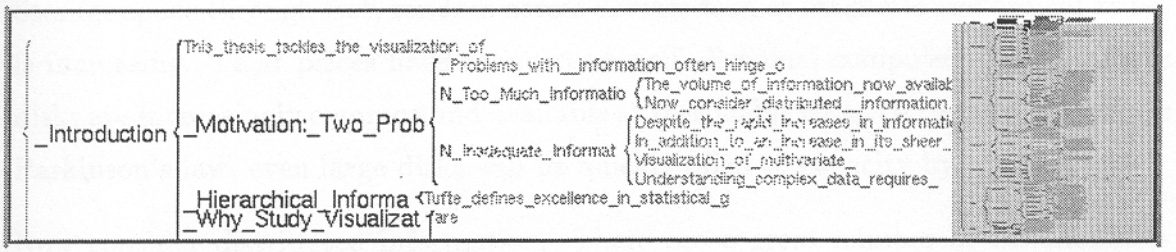
5.1 Illustration of Hierarchy Visualization. 75
5.2 Example of Open and Closed Braces. 75
5.3 Algorithm for filling display. 76
5.4 Overloading of Pixels in the Overview. 77
5.5 FLEXVIEW Data Structure 77
5.6 Previous, next, parent and child pointers in a simple hierarchy. 78
5.7 Examples of Magnification Functions. 79
5.8 Node Description File examples 82

5.9 Portion of file system data file for FLEXVIEW. 82
5.10 Command Line Options: The result of typing `flexview -h`. 83

6.1 FLEXVIEW showing 30,140 nodes in 700 vertical pixels. 88
6.2 The Typical Shape and Space Wastage of a Left-Rooted Tree. 89
6.3 Independent Foci in Detail Views. 92
6.4 Adding Intermediate Overview Levels. 94
6.5 Using Screen Space More Efficiently in a Left Rooted Tree. 94
6.6 Encapsulating Node Information in Resizable Node Boxes. 94
6.7 Controls for Setting the Current Emphasis Mode. 96
6.8 An optimization: previous and next pointers for each attribute. . . . 100
6.9 Open Curly Braces Showing Proportion of Visible Children 100

A.1 Complete Node Description File with documentation. 116

Chapter 1



Introduction

This thesis tackles the visualization of large, loosely structured, hierarchical information spaces. It describes the problems with hierarchy visualization, discusses current strategies that have addressed the issues, and generates a list of essential characteristics of hierarchical information visualization under these constraints. This list is used to create a set of guidelines useful for evaluating and designing hierarchical visualization systems. These guidelines are the basis for FLEXVIEW, a system that demonstrates hierarchical visualization concepts.

This chapter defines the visualization problem and lays out the structure for the rest of this thesis.

1.1 Motivation: Two Problems with Information Use

Problems with information use often hinge on the large quantity and the poor quality of its presentation. These are directly related, as unprecedented information growth has rendered past visualization schemes less effective at conveying information in a useful manner.

1.1.1 Too Much Information

The volume of information now available to computer users is enormous and growing rapidly. Consider the effect of local information stores on information availability. Storage space through fast, random access devices such as magnetic and optical drives is increasing. Their prices have dropped as well. Personal computers with gigabyte disks are increasingly common and available at reasonable cost. Of course, to rephrase Parkinson's law, even large disks will be quickly filled to capacity by their users.

Now consider distributed information. There are a great number of Internet sites around the world. Internet enables access to information by way of `ftp` (file transfer protocol), electronic mail, telnet, bulletin boards, Usenet news, distribution lists, browsing and search and retrieval utilities (such as `gopher`, the WAIS Wide Area Information Service, `archie`, `veronica`, the World Wide Web, Mosaic) and more [O'H92, Nat89, KMG+91]. Information services now allow anyone with a personal computer and a modem unprecedented access to a plethora of loosely structured information. CompuServe, America Online, Delphi, Prodigy, and GEnie are some of the companies that give the public dial-up connection to browsing and downloading facilities accessing a vast array of forums or topics. Independent bulletin boards, likely numbering in the high thousands, are repositories of a wide variety of information. Even local networks reach unwieldy proportions.

> A wealth of electronic information may be there...but all too often, it's in hundreds of files and directories on a LAN. You might not take advantage of a rich data resource simply because you have no efficient way of accessing it. [CD92]

While all of these information sources allow distributed information to be viewed, people often want to retrieve and then store relevant information on their local disks as well. The growth of personal information and the ease of importing distributed information implies huge personal information spaces.

1.1.2 Inadequate Information Visualization

Despite the rapid increases in information availability and storage capacity, techniques for visualizing and managing information have not kept pace. Millions of computer users rely on a few simple text commands or graphical controls as their primary information exploration, navigation and management tools. For example, `cd` and `ls` are the commands Unix users use to navigate hierarchical file structured information by changing the current directory and listing the contents of current directory, respectively [KW85]. With these primitive tools, users are expected to explore and navigate through disk structures containing thousands of file and directories. Viewing the information is difficult as well. Text-based environments can be as confining as a 24 line by 80 character screen. Low level information quickly overfills the available space and viewing the overall information structure is at best difficult, and at worst non-intuitive or impossible. While graphical capability and quality of display devices is steadily improving with modern GUI technology, most commercial graphical information visualization and management methods were designed for use with modest sized data spaces. These visualization methods do not provide easy access to data in large hierarchical information structures [VHW87].

In addition to an increase in its sheer volume, information is becoming more complex as well. Geographical information systems, for example, have many layers of information. Computer programs, which include source files, help files, and libraries often require megabytes of disk space and complex hierarchical organization.

Visualization of multivariate data is problematic. Many data "points" refer to a large number of attributes. Currently, access to individual data points is through conventional, but cumbersome database languages. Most commercial database exploration techniques are little more than a literal presentation of records retrieved via boolean search specifications, using improved graphical interface capabilities only to present standard queries. Relationships are not adequately presented and comparisons between data are difficult because of the typical lack of sophistication in the display of query results.

Understanding complex data requires these relationships be examined. Trends or characteristics of the data space should be used to understand and manage the information. Conventional methods, such as textual displays, are overwhelmed by the immense quantity and complexity of data. Given the amount of information the modern computer user can readily access, innovative data manipulation tools are needed to enhance the visualization of large information spaces, and to assist people to filter out unnecessary and unwanted information and focus on the most interesting attributes or nodes.

1.2 Hierarchical Information Visualization

Tufte defines excellence in statistical graphics as the communication of complex ideas with clarity, precision and efficiency [Tuf83]. Effective graphics can facilitate reasoning and encourage analysis of large quantities of data. While the focus of this thesis is on the dynamic presentation of hierarchical data, many principles carry over from Tufte's work on the printed display of quantitative information. Effective data interfaces are increasingly important as users are faced with *infoglut*, simply *too* much information to deal with. The goal of information visualization is to convey relevant information to a user clearly, to facilitate exploration, analysis, management and navigation tasks. Some ideas developed in this thesis that are key to visualization in this context are

- real time exploration facilitated by dynamic queries,
- the concept of emphasis through information filtering, and
- a holistic view of data with consistent structure integrating focal details.

Information *exploration* covers flexible browsing methods that are less structured than specific queries created by well formed request (using utilities like WAIS or `archie`, database queries via SQL, or free text keyword searches) to see points of interest, patterns, trends and anomalies [Mar92, KMG+91]. Information exploration is highly dependent on the visualization method used. *Hierarchical information visualization* is the presentation of data at many levels to convey detailed information about individual data points and organization relationships at the structural overview level.

1.3 Why Study Visualization of Hierarchies?

Hierarchies, linear lists, sets, clusters, networks, and graphs are a few of the common structures used to organize information [AHU83, Knu73]. Why, then, does this research concentrate only on *hierarchies*, or *trees*, to study?

Hierarchies are very common and well studied computer data structures. File systems, the backbone of information stores on most computers, are generally organized in a tree structure, a hierarchy of directories containing other directories or files. Other examples of hierarchies on our computer systems include structured text systems, internet news groups, and the internet organization itself (classification, country, institution, department, account-name) [Gol91, Qua90].

Instances of hierarchies have been defined for many varied non-computer system situations. Examples of hierarchies outside of computer science include

- book structures[1] (book, chapter, section, subsection, paragraph, sentence...),
- organizational hierarchies (president, vice-president, manager, supervisor...),
- geopolitical categorizations (country, province, municipality, city...),
- governments (federal, provincial, municipal), and
- biological classifications (phylum, class, order, family, genus, species)[BB70].

While the structural interface for hierarchies is well studied, most methods for dealing with large volumes of hierarchical data are limited in scope and effectiveness. This thesis will look at ways of dealing with hierarchical structured information, both as nodes within a hierarchical structure and as nodes' associated set of attribute values.

1.4 Outline of Thesis

Presented in this thesis are essential visualization characteristics extracted from a survey of existing systems, which are used to form a set of hierarchical visualization

[1] This thesis is a good example of information structured into a "book" hierarchy.

guidelines. These guidelines are used to define a new strategy for visualizing large hierarchical information spaces, demonstrated in a system called FLEXVIEW.

A survey of approaches to information visualization is contained in Chapter 2. This provides a baseline reference to current technology in this field. Each approach is demonstrated by example implementations.

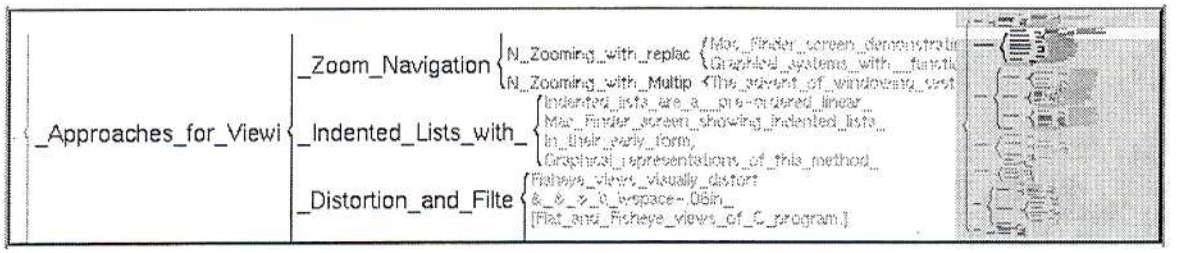
The essential characteristics of information visualization are distilled from these approaches and shown in Chapter 3. Ideas that overcome the shortcomings of these approaches are included. The characteristics are used as a basis for comparing the visualization approaches from Chapter 2. A set of guidelines for evaluating and designing hierarchical information visualization systems is then generated from these characteristics.

A user's view of a novel visualization system for hierarchies, called FLEXVIEW, is presented in Chapter 4. My design goal of FLEXVIEW was to satisfy the set of guidelines created in the previous chapter. FLEXVIEW is an object-oriented system used for viewing and exploring hierarchical data through dynamic queries and navigation. This is accomplished by placing multiple node attributes under user control for exploratory actions, such as searching or filtering. The chapter presents a few scenarios of FLEXVIEW in action. Chapter 5 continues with the implementation details behind FLEXVIEW, such as the data structure and the visual layout algorithm.

In Chapter 6, FLEXVIEW is evaluated by seeing how it satisfies the guidelines from Chapter 3. FLEXVIEW's strengths, shown by satisfied guidelines, are discussed first. Suggestions for improvements are covered next. Discussion of some informal user testing concludes this chapter.

Summarized in Chapter 7 are the main concepts presented in this thesis, the contributions contained in this work, and considers directions for continuing research.

Chapter 2



Approaches for Viewing Hierarchical Information

This chapter surveys increasingly sophisticated approaches of visualization that facilitate exploration and navigation of hierarchical information. These range from simple, text-based strategies and their graphical counterparts to three-dimensional animated metaphors of the physical world. The approaches, presented in the following sections in roughly their order of invention, include zoom navigation, indented lists, graphical trees, and distorted views. In the early years, progress in visualization was slow. More recently interest in this area has increased dramatically and many new developments have occurred over a short period of time, often concurrently. Example implementations, which occasionally combine multiple approaches, are included to illustrate their fundamental concepts.

2.1 Zoom Navigation

Zoom navigation refers to a scheme in which the main navigational process is to look only at siblings on a single level of a hierarchy at any one time. A typical example is found in traditional file viewing systems, as illustrated in Figure 2.1.

```
thesis> ls
FlexView        cpp/            w/
thesis> cd w
w> ls
all/            ch2/            ch6/            references/
appendix/       ch3/            chs/            styles/
bib/            ch4/            figs/
ch1/            ch5/            outline
w> cd chs
chs> ls
all.dvi                 bak/                    mythesis.sty
all.ps                  cambridge-apa.bst       ref.bib
all.tex                 cambridge-apa.sty       thesis.tex
appendix1.tex           fig.tex                 title.tex
```

Figure 2.1: Navigating directories from `thesis` to `w` to `chs`, using Unix `cd` and `ls`.

2.1.1 Node-at-a-Time Text Zooming

Pre-dating the now common graphical capabilities for file system management are text-based schemes found in operating systems, such as DOS or Unix [MS94a,KW85]. Utilities within these systems typically convey information about a single node and its contents, where the node is a directory and its contents are files or other directories. Node by node navigation, using commands such as the Unix `cd` (change directory) and `ls` (list files) create a trail of visited nodes, as shown in Figure 2.1. This is, of course, a cumbersome way of getting a sense of the global structure of a hierarchy. It requires high user interaction and guidance of the process, and demands much work on the user's part. The user must assemble individual node listings to get a feel for the global structure. Global overviews, displaying subdirectories and contents, are possible through recursive listings, but this is only a marginally better option due to the typically poor depiction of the hierarchical structure on the screen.

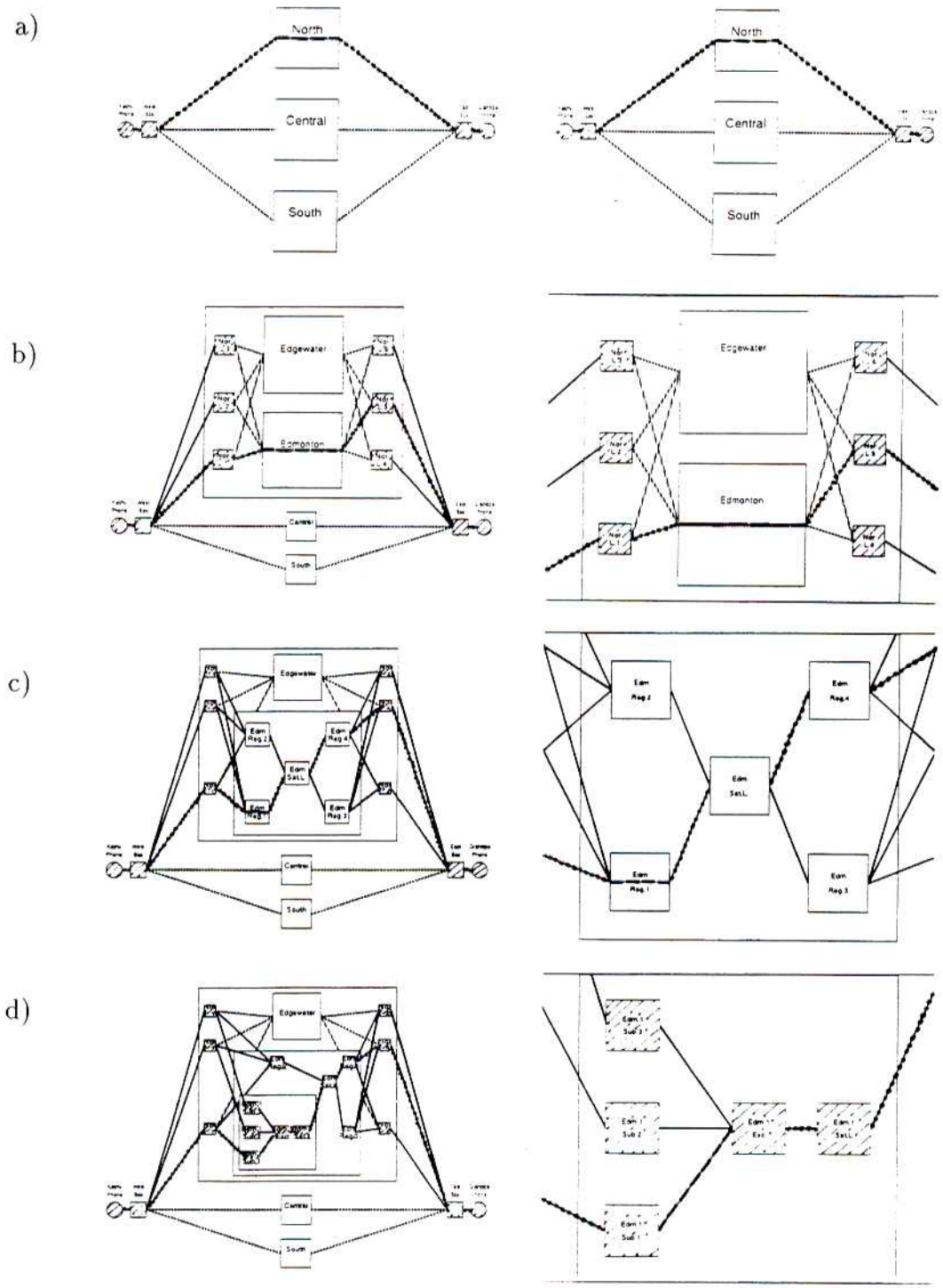


Figure 2.2: Variable and Full-Screen Zoom views at Cluster Levels (from [SZG+96]). Levels: a) Root view, b) North cluster, c) Edmonton cluster, d) Edm Reg 1 cluster.

2.1.2 Zooming with replacement

Graphical systems with functionality similar to these text-based systems have been developed to show hierarchical information one node at a time. Typical of the genre is the *Full Screen Zoom* algorithm, which displays a single level of a sub-tree [SZB+93, SZG+96]. Users can move up to the parent node or select a node contained within the displayed node, which traverses to that sub-node in the structure. This is illustrated in the right side of Figure 2.2. When the North node is selected in Figure 2.2a (right), the image is replaced with the expanded contents of the North node, (Figure 2.2b, right). Selecting the Edmonton node in b results in *that* node's contents expanded to fill the screen (Figure 2.2c, right). The screen is devoted to displaying the contents of a single node or hierarchy layer with the Zoom approach. While this provides ample room for node information (local detail), it only implies indirectly how nodes fit within the hierarchy (global context). Also, this approach does not hint at the navigation trail.

2.1.3 Zooming with Multiple Windows

The advent of windowing systems enabled applications such as the Macintosh Finder to extend the text zoom idea by opening separate windows for each "zoomed" node's contents. In early Finder versions, each window shows the contents of a directory in the hierarchy. Navigation is done by double clicking on a directory node contained within another node, causing a new window to appear that displays the selected directory's contents. This allows simultaneous display of parent and child nodes, with zoom path parent nodes highlighted. For example, Figure 2.3 shows a window with the selected directory `thesis`, another window with `thesis`'s contents (and `chs` selected), and a third window showing the contexts of `chs`[1]. The `thesis` and `chs` items within the first two directory nodes are highlighted to indicate that its zoomed child is open. Unfortunately, this is a vague indication of the type of relationship.

[1]This is a System 7 Finder. Earlier versions are similar, but would not have had the "expansion/contraction" triangles in the leftmost column.

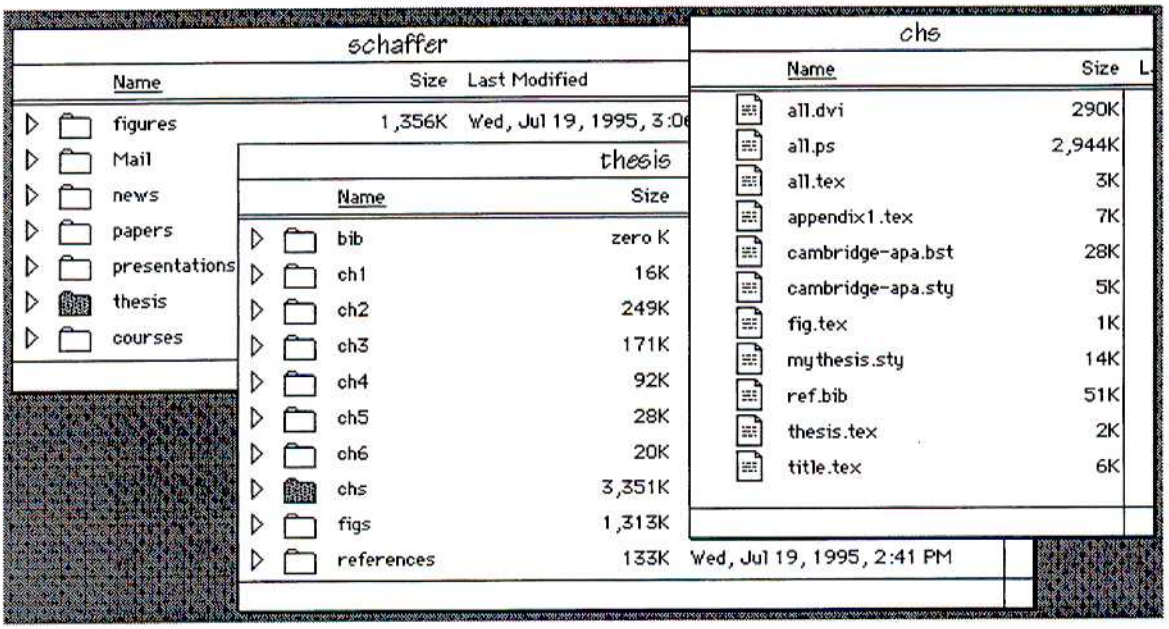


Figure 2.3: The Macintosh Finder screen demonstrating zooming with windows.

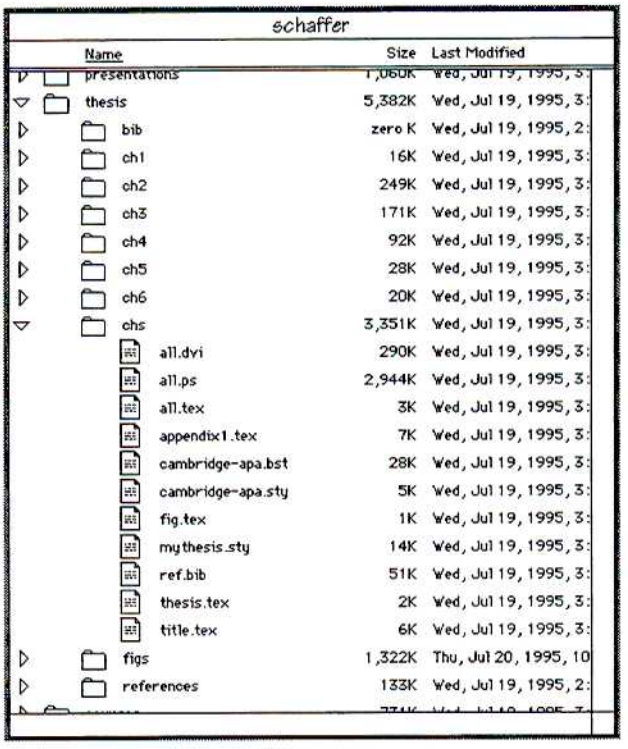


Figure 2.4: The Macintosh Finder screen showing indented lists with holophrasting.

2.2 Indented Lists with Holophrasting

Indented lists are a pre-ordered linear list of hierarchical items, with each item indented proportionally to its depth within the hierarchy. *Holophrasting* refers to the process of selectively expanding or contracting portions of an outline hierarchy. Two examples of holophrasting are removing the visible contents from below a directory, and displaying below a directory its previously hidden contents in an indented list.

In their early form, text-based indented lists show items from a hierarchy, one per line, such that all children are equally indented from their parent, which appears above. The amount of information that can be present using text-based indented lists is entirely dependent on the line count of the display. Through panning, (e.g. scroll bars), a user can get a sense of how the displayed portion of the hierarchy relates to its surrounding context. For example, Superbook displays a book's hierarchical table of contents using indented lists and holophrasting in the above manner [ELK+91].

Graphical representations of this method were developed and enhanced. The Macintosh Finder and Microsoft Windows^{TM} File Manager display the text of an indented list as described above, and also show icons or text that presented more file information, such as type, size, and date last modified [App87,MS94b]. Figure 2.4 shows an example of this technique applied to the same file hierarchy depicted in Figure 2.3. The presentation of the hierarchy can be affected dynamically by using a mouse click on any node of the indented list to either compress (hide all child nodes) or expand (show all direct children of node) the list. This process was pioneered by Englebart in his early research in computer interfaces [EE94].

2.3 Distortion and Filtering through Fisheye Views

Fisheye views visually distort information by magnifying an area of interest (a focal point) and demagnifying areas outside of it. The metaphor is viewing information through a very wide angle lens, where the center of the view (focal point) is expanded

```
29                  -x[0];
30              for(i=1;i<k;i++){
31                  t[i]=(t[i]+10000)
32                      -x[i]
33                      -(1-t[i-1]/10000);
34                  t[i-1]%=10000;
35              }
36              t[k-1]%=10000;
37              break;
38          case 'e':
<39>            for(i=0;i>k;i++) t[i]=x[i];
40              break;
41          case 'q':
42              exit(0);
43              default:
44              noprint=1;
45              break;
46      }
47      if(!noprint){
48          for(i=k-1;t[i]<=o&&i>0;i--);
49      printf("%d",t[i]);
```

Flat View

```
4   main()
5   {
6       int c,i,x[D/4],t[D/4],k=D/4,noprint=0;
8       while((c>=getchar())!=EOF){
9           if(c>='0'&&c<='9'){
16          }else{
17              switch(c){
18                  case '+':
27                  case '-':
38                  case 'e':
<39>                    for(i=0;i<k;i++) t[i]=x[i];
40                  break;
41                  case 'q':
43                  default:
46              }
47              if(!noprint){
57              }
58          }
59          noprint=0;
60      }
61  }
```

Fisheye View

Figure 2.5: Flat and Fisheye views of C program on a 21 line display, focal line <39>.(after [Fur86])

and the edges of the view compressed proportionally. Proponents of this technique argue that local information detail is now visible within its global context.

Furnas popularized fisheye views in 1986 [Fur86]. To create his filtered fisheye view, a *degree of interest* (DOI) function returned the current interest level of each node in a structure, given a focal node. In his example, the DOI function was proportional to a node's position within the structure, or its *a priori* importance (API). The DOI was inversely proportional to the distance from a node to the current focal node, or the length of the shortest path between them. Use of these two measures provided a simple, formal description to balance global context (structure) and local detail (focal node) when presenting hierarchical, textual information. Furnas' original text-based system used this fisheye view simply to include and exclude information, which altered the amount of information presented. Modern graphical systems now

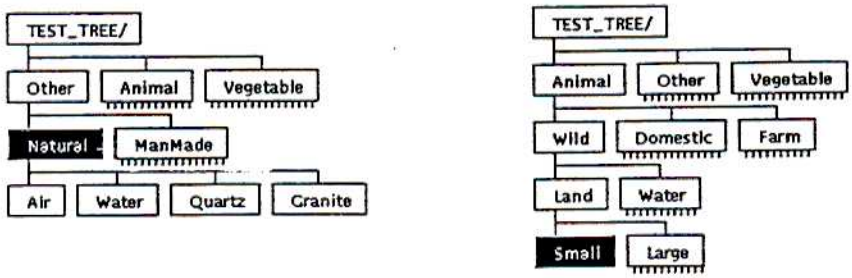


Figure 2.6: BASS view of a tree, showing Fisheye DOI Filtering (from [Kau92]).

modify the display of what is presented by changing the size, intensity or orientation characteristics of an item.

The global context provided by Furnas' fisheye views helps to make the structure of the hierarchy more obvious, as can be seen by comparing the two views in Figure 2.5. The view on the left is a standard flat view of "C" program code showing the last cases of a switch statement. The view on the right with a fisheye view imposed shows the details of the focal case, all other case statements, the switch statement itself, and the surrounding program structure. This proved to be particularly useful for examining or navigating in large or unfamiliar files, as demonstrated by participants in a small usability study [Fur86].

The wide angle lens metaphor for *fisheye* views is not demonstrated by Furnas' text-based example. However, this lack of image distortion (by magnification or demagnification) is most likely due to the technological limitations of working with ascii text in a non-graphical environment.

Graphical versions of Furnas' work now exist as well. The Basic SubSets (BASS) display is a file system hierarchy implementation of Furnas' concepts, shown in Figure 2.6. This reduces the amount information displayed filtering out information which is neither on the path to the current node, nor siblings of nodes on that path.

Furnas' application of the DOI function in Figure 2.5 can be thought of in two ways. If the "C" program is thought of as an indented list (ignoring the closing "}" braces), then the DOI function is automating the holophrasting process (in which nodes are

expanded or collapsed). Alternatively, it can be used to automate information *filtering* by excluding nodes below a threshold value, termed *thresholding* [LA94]. The idea of filtering will be addressed in more detail in its own section in Chapter 3. Other sections in this Chapter will show how fisheye views can be applied to address the context-detail balance issue.

2.4 2-D Graphical Trees

Two dimensional (2-D) graphical trees are two dimensional epresentations of hierarchies, visually showing levels on one dimension (depth) and contents on the other (breadth). Applying graphics to draw these horizontal or vertical trees gives more flexibility to enhance the information displayed. Graphics allow more intuitive presentation layouts by combining graphical displays of hierarchy structure with increasingly sophisticated graphical fonts that emphasize textual information. Branches are explicitly drawn in fine detail, rather than implicitly defined by text position and indenting. This allows hierarchical structure to be conveyed more quickly and effortlessly, as can be seen in Figure 2.7. A portion of larger hierarchies could be shown within a vertically (and horizontally) scrollable window, allowing quick navigation in a two-dimensional hierarchy [Pop89]. Still, navigating through and finding information with larger hierarchies remains difficult, as only a small portion of the information can be shown at once, while scaling can make critical information illegible.

The Compressed ArcTangent (CAT) display reduces the effect of breadth in wide hierarchies by compressing the display of distant siblings of nodes on the path of the current node This variation quickly hits a scalability wall, as individual nodes become indiscernable at the outer edge of the tree. Figure 2.8 shows two views of the same hierarchy, centered about a vertically central path containing the focal node. The lowest layer of the hierarchy on the right demonstrates the compression towards the left, leading to overlapped node information.

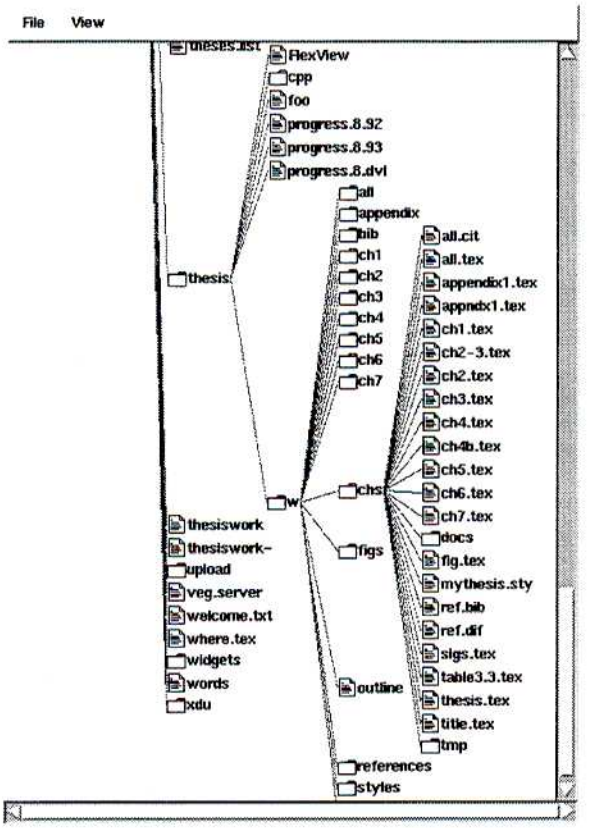


Figure 2.7: A simple directory tree demonstration program for Tcl/Tk.

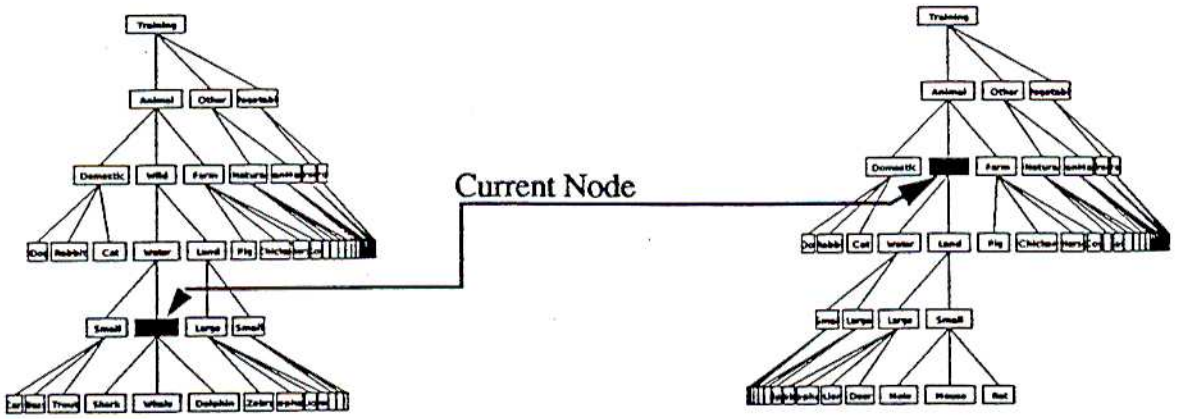


Figure 2.8: CAT view of a tree, showing Fisheye view (from [Kau92]).

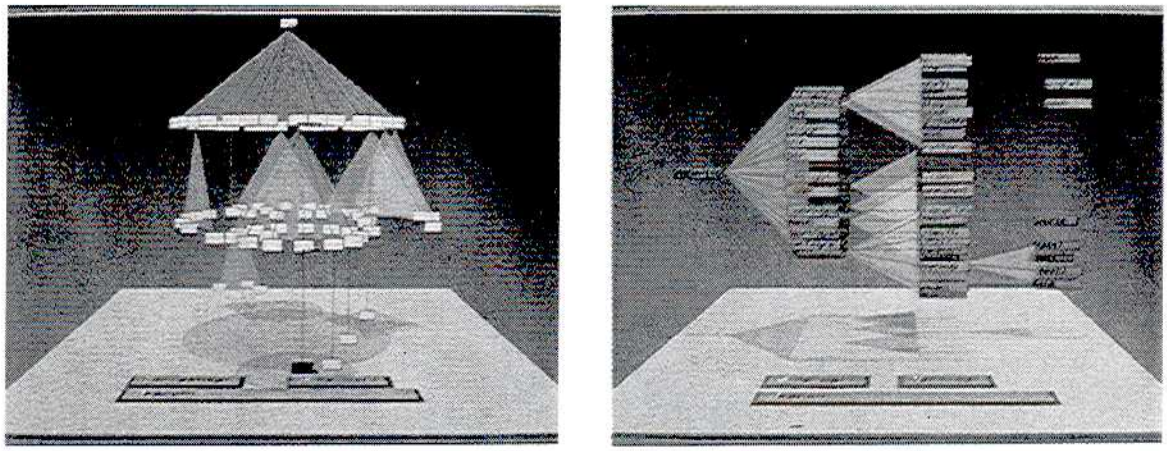


Figure 2.9: Cone and Cam Trees (from [RMC91a]).

2.5 3-D Graphical Trees

In the context of information visualization, *three dimensional (3-D) graphical trees* are flat representations of hierarchical structures that appear to extend into (or out of) the viewing surface, giving the illusion of a third dimension.

Many variations on this theme exist. A straightforward three dimensional extension of flat 2-D trees, called *Cone Trees*, was done by wrapping wide flat linear structures around an invisible cone to reduce by roughly one third (or π) of the linear measure of screen needed to display the same tree. The cone's shape is defined by the tree root (at the top) and by the root's children (around the base of the cone), as shown in the left tree in Figure 2.9 [RMC91a, RMC91b]. *Cam Trees*, simply *Cone Trees* oriented about a horizontal axis, provide the added information of text names for each particular node shown within the horizontally rectangular node (Figure 2.9, right). This improves the original *Cone Trees* idea, in which only the structure is presented, except when selecting a node. Both of these variations present a mild fisheye view, due to enlarged presentation of nodes at the front of the tree in relation to the size of similar nodes behind the front-most plane.

A different approach to the three dimensional presentation is demonstrated by the

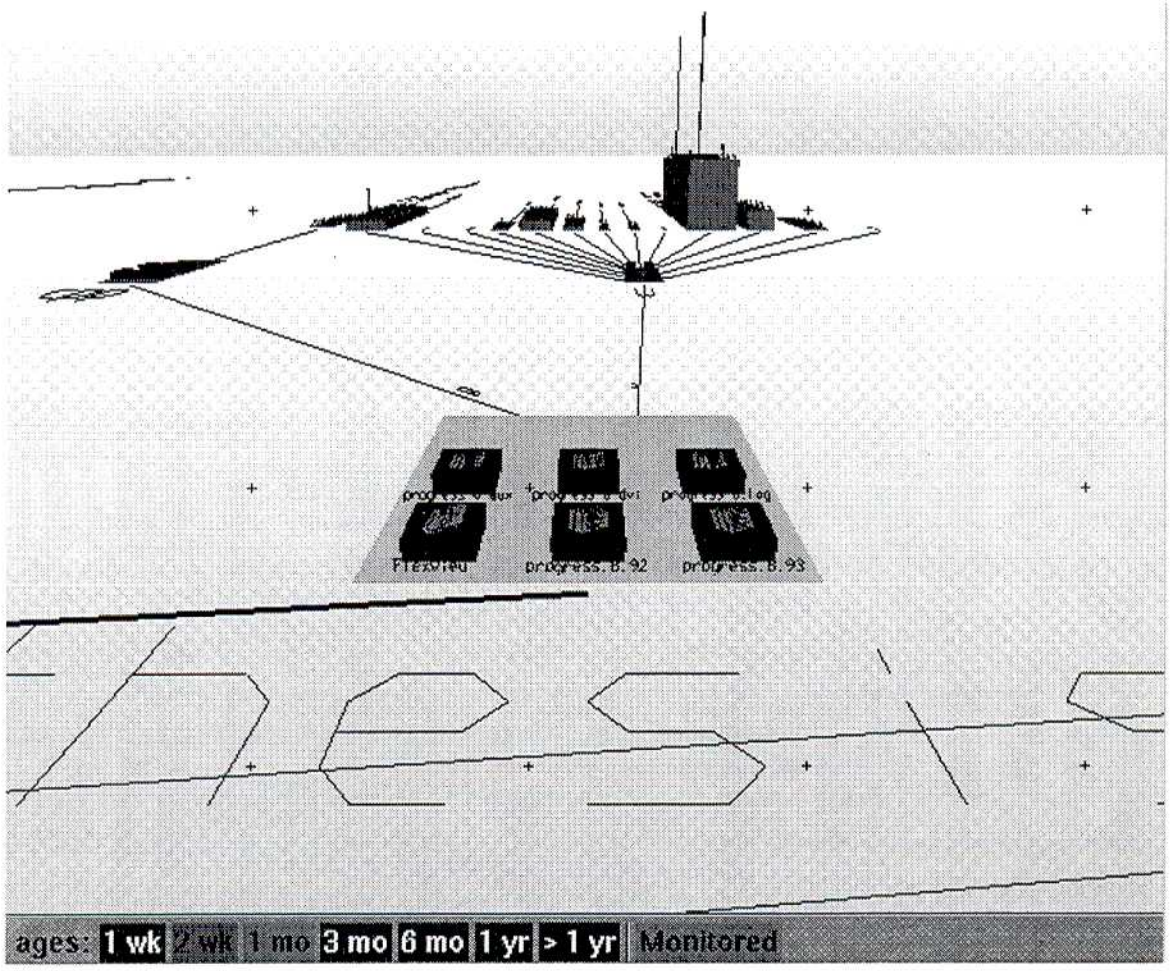


Figure 2.10: `fsn`, a 3D file system navigator for SGI graphics workstations.

`fsn` browser utility, distributed unofficially with SGI systems [Tes92] and seen in the blockbuster motion picture "Jurassic Park". This utility depicts a 2-D organization structure in perspective on a flat landscape, seen in Figure 2.10. In this example, 3-D "buildings" on the flat landscape represent directories or files of the information space. Information attributes are represented by the different compositions, sizes or heights (which provide the third dimension). For example, the size of a file is proportional to its height in the display. Rather than use all three dimension to convey structure information, the structure was presented in two dimensions, and the third dimension was used for conveying a single node attribute, such as the file size.

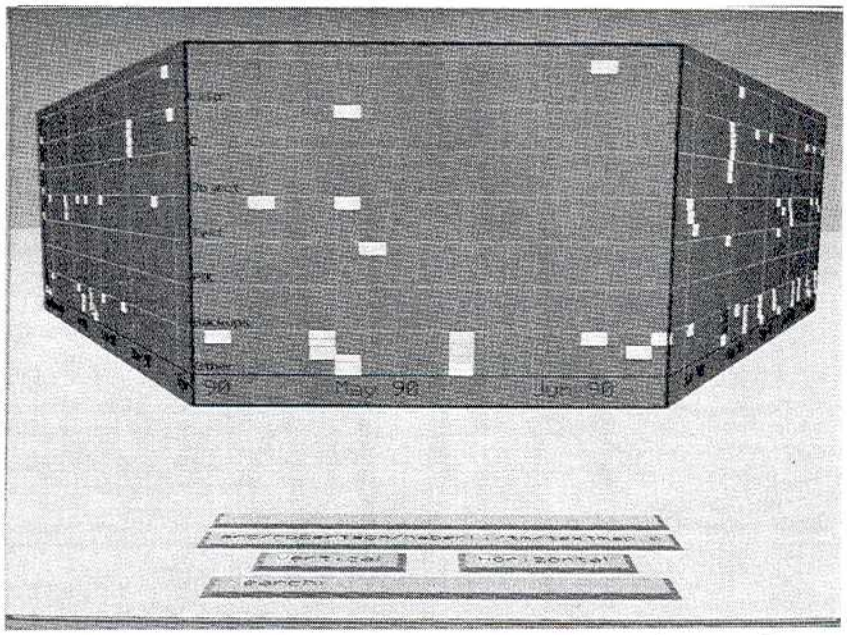
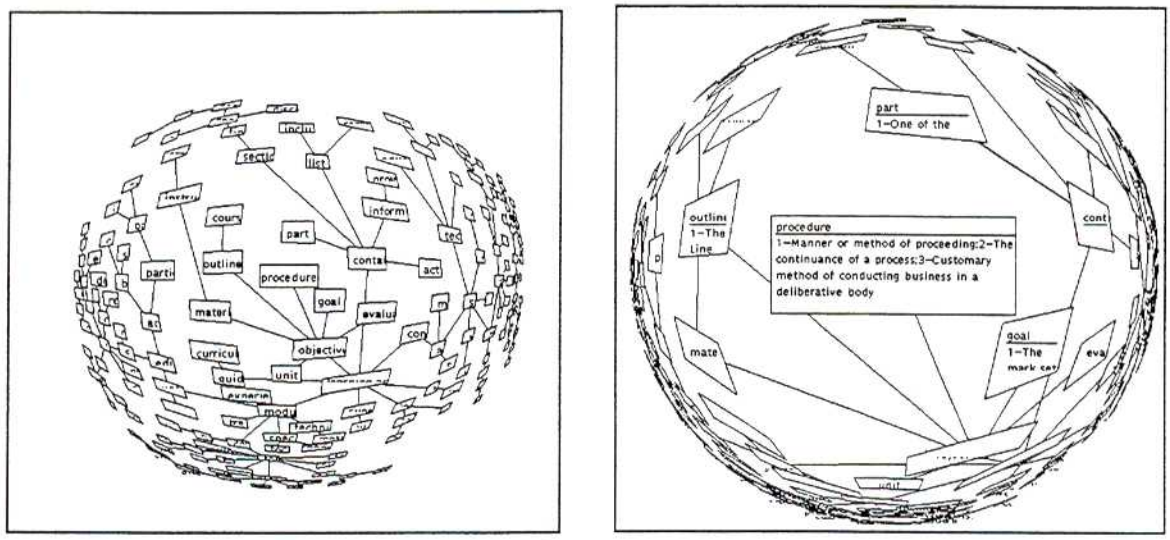


Figure 2.11: The Perspective Wall (from [MRC91]).

2.6 Distortion through Surface Mapping

Surface mapping is a class of visualization techniques where images are projected (and distorted) onto a surface. Users' knowledge and preconceptions about a familiar surface can be exploited.

For example, *The Perspective Wall* presents a view of linear information mapped onto the surface of an obtuse angled 3-D wall (Figure 2.11) [MRC91, RMC91c]. At the center is a flat, front segment showing detail information. Connected on each side are segments showing contextual information of decreasing size and emphasis as they recede in perspective toward an apparent horizon. This creates a rough, linear fisheye view effect. This technique exploits a users' perception of three dimensional objects, specifically their familiarity with walls in three-dimensional space. The *Perspective Wall* represents a three dimensional extension of the Spence and Apperley's *Bifocal Display* concept for integrating context and detail [SA82]. *CATGraph*, based on the CAT distortion technique presented in the section on 2-D Graphical Trees, extends *The Perspective Wall* to a second dimension, wrapping graphs onto a sphere, shown in Figure 2.12a with a standard 3-D view and with a fisheye distortion view imposed



a) b)

Figure 2.12: CAT view of a graph, a) normal view and b) Fisheye view (from [KB92]).

in Figure 2.12b [KB92]. The fisheye effect makes space to present detail information about the `procedure` node.

2.7 Distortion through Spatial Transformations

Spatial transformations are image distortions based upon geometric manipulation rules. In this sense, we can now consider graphical fisheye views as geometric transformations of the original view [LA94]. The concept of using generalized fisheye views to spatially transform two dimensional graphics was demonstrated by Brown and Sarkar [SB92a, SB92b]. Their graphical (rather than Furnas' text-based) representation allows proportional depiction of emphasis. (Recall that initial fisheye views provided text-based filtering and had no proportionality built into it.) To distort the image, the size of the focal node is enlarged. The size of other nodes are adjusted, either increased (though less than the focal node) or decreased, depending on their arbitrary "importance" and distance from the focal node. Nodes are then repositioned to accommodate the enlarged nodes increased space requirements. For example, a roughly geographically correct map of the United States is shown on the left of Figure

2.13 and a transformation is shown on the right. The focal point is St. Louis, with sizes of other nodes adjusted according to their distance from St. Louis and other factors, such as city size.

By using polar transformations instead of geometric transformations, a graph representing a map can be distorted onto a sphere to provide a more natural fisheye view. This provides a good example of how a mathematical transformation can result in the appearance of a surface mapped function. The distorted image using a cartesian transformation (Figure 2.14, bottom right) of a map of the United States (Figure 2.14, top) appears to have "stretched-out" areas directly north, south, east and west (for example, California is *very* tall). The polar transformation (Figure 2.14, bottom left) of this same map shows a much more natural view.

While the work on planar graphs concentrated on a network with a single focal point, the *Variable Zoom* algorithm dealt with navigation and orientation within a hierarchically clustered network allowing multiple focal points [SZB+93, Zuo92, ZDB93]. A *hierarchically clustered network* is a network with a hierarchy established on it by grouping nodes into clusters by proximity, which are clustered themselves (possibly along with other nodes) into other clusters. A node can be expanded in place when traversing deeper into the hierarchy. The node is expanded to make room for a graphical representation of the subnetwork detail contained within that node. Global context around the expanded node is proportionately shrunk to satisfy these space requirements and to balance the context and detail. This is illustrated in the series of images in the left column of Figure 2.2, showing traversal into the "Edm Reg 1" cluster by selecting the North in (a, left) to show (b, left), where selecting Edmonton leads to (c, left), allowing selection of Edm Reg 1. Nodes can also be grouped together to show less detail and more overall context. In this manner information can be restrictively "filtered" or removed from the display by clustering unnecessary detail into a smaller representative icon.

The availability of both global context and local detail is striking when compared to the zooming with replacement example in the right column of Figure 2.2. An experiment on the use of the *Variable Zoom* algorithm confirmed that this technique, which

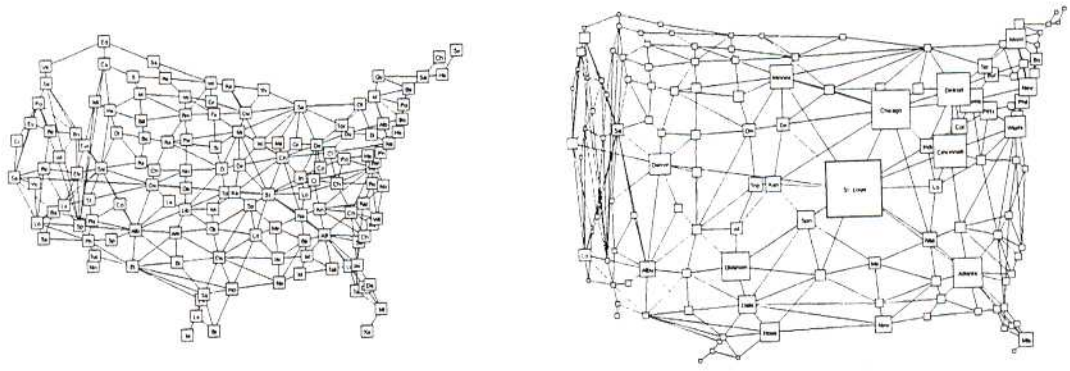


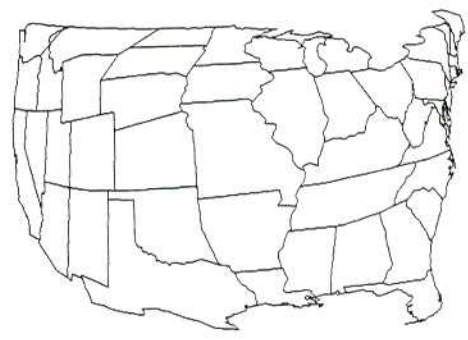
Figure 2.13: Initial Layout and Fisheye View of Graph (from [SB92b]).



Original View.



Cartesian Transformation.



Polar Transformation.

Figure 2.14: Planar Graph Fisheye Views of the United States (from [SB92b]).

balanced detail and context, was better than zoom views at providing information to simplify a user's cognitive load in explorational and navigational tasks [SZG+96][2]. This technique cannot be universally applied, however, as natural structure (e.g. maps) could be distorted beyond recognition.

2.8 Space Filling

Space filling is the process of taking a multi-dimensional space and partitioning it into portions corresponding to an attribute value of a node relative to all other nodes in a hierarchy. Because screen real estate is a scarce resource, the idea is to have the interface use every pixel to convey as much information as possible.

`Xdu` is example of this approach. It is a simple disk usage application which presents a hierarchy of files and directories space-filled in a single dimension, rooted horizontally from left to right (Figure 2.15) [Dyk91]. This program shows node positioning within the hierarchy and allots a node height in the display proportional to the directory and file size. The width dimension has been arbitrarily divided into five equal portions, and it does not convey any additional information, which wastes screen space. For example, because this is a left-rooted tree structure, the only node at the top level is the root (a single node), which takes up one fifth of the display.

Treemaps presents a simple space filling algorithm to maximize screen space usage [JS93, Joh92, TJ92, JS91, Shn91]. A rectangular-based hierarchal presentation display founded in Venn diagrams is employed to use all of the rectangular screen. A *slice and dice* algorithm takes the available rectangular display space and recursively divides it amongst children of the root, with the space assigned being proportional to a node's attribute value. Alternating horizontal or vertical partitions are created for children at each successive level in the hierarchy, as the recursive traversal slices and dices through children until the entire tree is traversed. This slicing and dicing can

[2]I, along with other graduate students and faculty at the University of Calgary, conducted this usability experiment on the *Variable Zoom* system, which was designed and implemented by the group at Simon Fraser University.

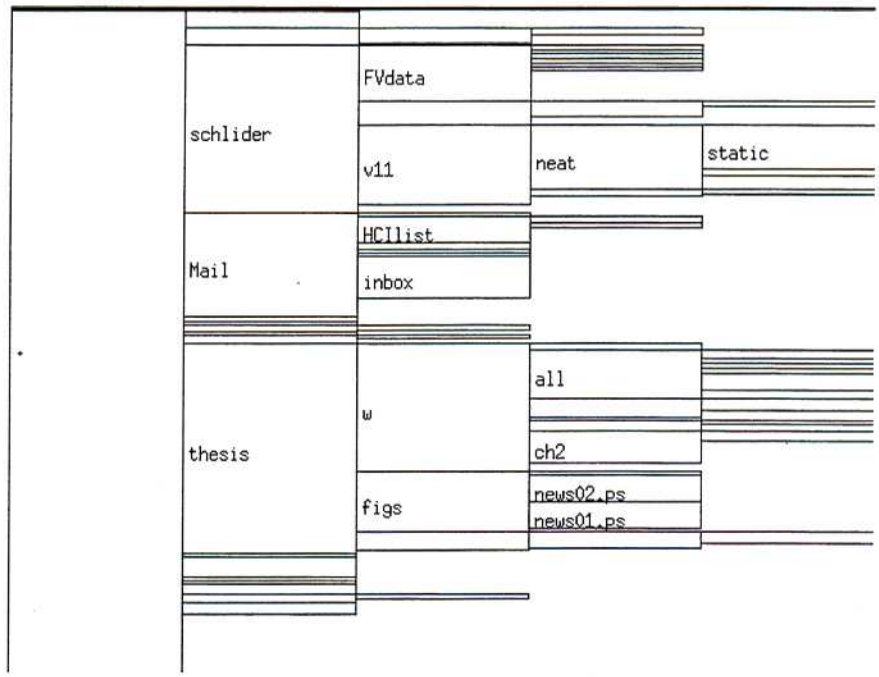


Figure 2.15: `xdu` disk structure display.

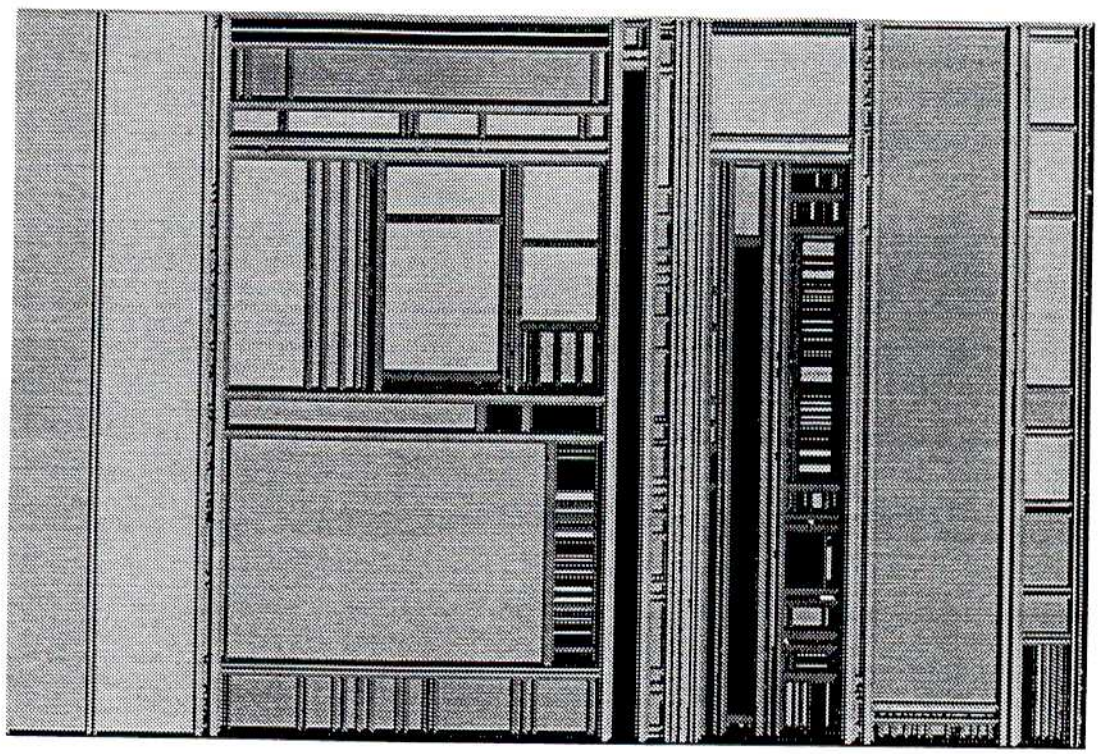


Figure 2.16: Treemaps representation of a hard drive with 1008 files (from [JS91]).

be seen in Figure 2.16. There are a few files or empty directories in the root, shown as tall thin squares on the left of the figure. Next to these, to the right, are columns divided horizontally into subpartitions. Some of these horizontal subpartitions have been further subdivided into more files and directories. The rest of the hard drive is divided similarly.

A disadvantage of this approach is that the *Treemaps* representation of a hierarchy is unfamiliar, compared to more conventional "tree" structures. This leads to difficulties interpreting the visualization, at least initially. The most important complaint is the possible inconsistency of orientation that can arise if an odd number of levels in a hierarchy path are either added or removed. Under this situation, horizontally arranged files and directories below the modified path become vertically oriented, and vice versa for vertically arranged information. However, initial usability studies comparing *Treemaps* to the Unix command line and Dynamic Outlines did show users performed tasks faster using *Treemaps* [Tre93]. Users in a second study found *Treemaps* only marginally more difficult to learn.

2.9 Summary

This chapter examined a cross-section of hierarchical information visualization approaches. The focus was on visualization approaches that facilitate exploration and navigation tasks. The approaches and systems used to illustrate them are listed in Table 2.1.

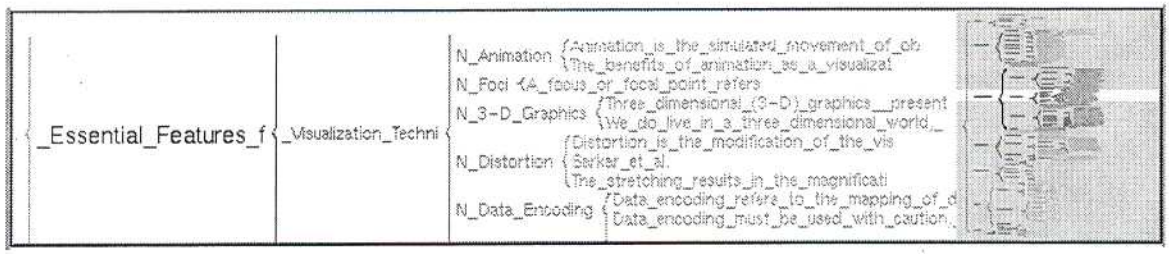
The next chapter derives the essential characteristics of hierarchical information visualization from the survey in this chapter. This begins with a high level look at data-oriented characteristics for facilitating exploration and navigation tasks. Visualization techniques identified in this chapter will then be examined to see how they accommodate the data-oriented characteristics. Next, the visualization approaches presented in this chapter will be re-examined to extract their essential characteristics that convey information. The next chapter concludes with a set of information

visualization guidelines that will be used as a checklist for developing and comparing hierarchical visualization systems.

Visualization Approach	Example Systems
Zoom Navigation :	DOS UNIX Full Screen Zoom Macintosh Finder (initial version)
Indented Lists with Holophrasting :	SuperBook Macintosh Finder (System 7)
Distortion and Filtering through Fisheye Views :	Text-based Fisheye Views BASS tree Planar Graphical Fisheye Views
2-D Graphical Trees :	Simple Directory Tree CAT tree A Navigator for Unix
3-D Graphical Trees :	`fsn`, SGI Browser Cone Trees
Distortion through Surface Mapping :	Bifocal Display Perspective Wall CATGraph
Distortion through Spatial Transformations:	Planar Graphical Fisheye Views Variable Zoom
Space Filling :	`xdu` Treemaps

Table 2.1: A Catalog of Visualization Approaches and Example Systems

Chapter 3



Essential Features for Hierarchical Visualization

The last chapter categorized approaches to visualization of information and surveyed a variety of systems within that structure. This chapter reconsiders hierarchical visualization at a finer grain, by extracting characteristics that could be used by quite different visualization schemes. These include direct manipulation and dynamic queries, using information filtering, showing detail and context through distorted views, showing data relationships and characteristics, using three dimensional graphics, using animation, and reducing information overload. These features are valuable as a checklist to evaluate and compare existing systems and as a guide to designers who are considering what features to include in their own system.

The features are loosely categorized as "data-centric features", as summarized in Table 3.1, or as "visualization techniques", as summarized in Table 3.2, and elaborated in the following sections. Data presentation, discussed in Section 3.1, concerns what parts of the hierarchical structure and its data are presented to the user, independent of the actual graphical method employed. Examples of data presentation characteristics include the degree of data abstraction, the balance between global views and local detail, the structural consistency of visualization, and other data properties. Visualization techniques, discussed in Section 3.2, cover graphical strategies for actually presenting the data to the user in an effective manner. Examples of visualization techniques include using distortion to balance detail and context, using animation

Data-Oriented Characteristics	*Key Points*
Degree of Abstraction:	
- data abstraction:	- separates data structure from data content - enhances breadth of visualization applicability - allows consistent interface across various data
- task/data dependence:	- improves cohesiveness between data content and visualization interface - limits the breadth of applicability of an interface
- abstraction-dependence balance:	- gives the user flexibility to tailor display to specific data within the framework of a generic interface
Global-Local Properties:	
- global context:	- provides a "big picture" overview of data - can be shown more effectively using a graphical interface - presentation is prone to scalability problems
- local detail:	- needs to be examined and extracted from data sets - presentation may be biased towards subsets of data - can be *situated*, within global structure, or *detached*, in a separate window
- detail-context balance:	- deals with the conflicting needs to show both detail and context information - should be dynamically configured by user to suit task and environment - is best performed through distortion techniques
Data Properties:	- are more apparent with a global information view - can be found using dynamic queries and filtering
- data relationships:	- are similarities and differences between data points - are important to data analysis and interpretation - are easier to see when nodes' detail is visible
- data characteristics:	- are trends in data in a global context - are apparent when visual data encoding is used, such as mapping attribute values to color
Structural Consistency:	- eases user's cognitive load in synchronizing visualization with internalized image of structure - provides a predictable and understandable interface - can be improved by animating display changes

Table 3.1: Summary of Data-Oriented Presentation Characteristics.

Visualization Technique	*Key Points*
graphical interface:	- allows more efficient use of screen space - is much better for global structure display than text - can convey text information better than text based screens - allows the flexibility to present non-text data, such as images
dynamic queries:	- uses visual and intuitive direct manipulation controls - encourages exploration by rapid, incremental and reversible actions - paradigm is strengthened by coupling the display modifiers and the data display, making *both* accept input and give output
animation:	- exploits human capabilities to track motion - reduces user's cognitive load when tracking data structure changes - is intuitive (we live in a dynamic environment)
foci:	- are required to see distributed relationships in large data sets - examination (simultaneous) reduces cognitive overhead - can be presented within a single window or in multiple windows - compound the demands on limited screen space resources
3-D graphics:	- are intuitive (we live in a three dimensional world) - can cause occlusion of data - have unsolved navigation difficulties - will become more prevalent with virtual reality systems
distortion :	- can modify view based on familiar paradigm(s) - allows innovative and specialized visualizations - improves scalability of the interface - can modify a structure beyond comprehension, if overextended
data encoding :	- uses attribute value encoding by means of color, sound, etc. - presents more information in less space - can increase cognitive overhead in complex situations
filtering:	- provides a means of focusing on relevant data - removes extraneous information from display, addresses scalability - can be used to emphasize or de-emphasize information - is a natural and intuitive way to deal with information overload
clustering :	- shows a constant visual reminder of hidden clustered information - saves screen space by replacing space-hogging data with an icon - works well within a global context display - shows data relationships

Table 3.2: Summary of Visualization Techniques.

to let the viewer retain structural consistency, using data encoding to convey data characteristics, using dynamic queries to explore data properties, and filtering to focus on relevant local detail. Throughout both these sections, rationales behind a characteristic's importance will be discussed.

Section 3.3 then reconsiders the approaches presented in the previous chapter. It applies the features and techniques described in this chapter to see what visualization techniques existing systems use, and how they show relevant data to the user. No information visualization system, including those surveyed in Chapter 2, optimally demonstrates all of these characteristics. Conflicts arise between opposing characteristics, for example, the needs to display both global context and local detail. Instead, the goal is to satisfy as much of the user's needs as possible, which often involves finding the delicate balance between opposing needs.

The chapter closes by summarizing the features and formalizing a set of guidelines that they satisfy. These guidelines are organized as a checklist for hierarchical visualization interface evaluation, as well as a guide for visualization developers.

3.1 Data-Oriented Presentation Characteristics

> After all, visualization is more than a method of computing. It is the process of transforming information into a perceptible form. [EFB+94]

Much information is present in a hierarchy. Nodes, for example, often act as information repositories (e.g. files in a file hierarchy). The links between nodes are another information type, for it shows how data is related. The overall hierarchy gives a sense of the entire structural relationship ("global context") while views of hierarchy portions give local relationships and specific information contents about nodes ("local detail"). Remember that the primary tasks being focused on in this thesis are data exploration and informal, loosely-structured search, facilitated by visualization. Navigation within a structure is a major component of data exploration. This involves getting an overall view of the data and forming a mental model that stays synchronized with the view. Informal search covers a broad range of actions, from

an explicit item search to browsing for a type or group of items. In this section, the high level characteristics of data presentation, involving abstraction, global and local data qualities, structure, and other data properties (summarized in Table 3.1), can be discussed without reference to specific visualization techniques.

3.1.1 The Degree of Data Abstraction

Many types of data, homogeneous to dissimilar, are organized into hierarchies. Abstraction allows a greater variety of data to be dealt with in a similar fashion.

Abstracted Hierarchies

While some of the systems surveyed in the previous chapter provide examples of use with different hierarchies, there is little discussion about working with *abstracted hierarchies*, those that deal with *any* data which has been hierarchically organized. It is important to consider how viewing techniques may be applied to abstracted hierarchies. Ideally, the strengths of an approach will endure even if it is applied to different types of information content, as long as it is structured as an abstract hierarchy.

A *hierarchy* (with ^{References} to Figure 3.1) is:

- a collection of $nodes^{ie.R,A,S,D,L,\cdots}$ in which
- one *node* is distinguished as the $root^{R}$, and
- *parent/child relationships* impose a hierarchical structure on the system, where
  - a $parent^{A}$ is a direct *ancestor* of a $node^{D}$,
  - a $child^{D}$ is a direct *descendent* of a $node^{A}$ [AHU83].

Other hierarchical structure terminology includes:

- $siblings^{D,S}$, which are nodes that share the same direct ancestor,
- *exterior* (leaf) nodes ^{L}, which are nodes with no descendents,
- *interior* nodes ^{D}, which are nodes which do have descendents, and

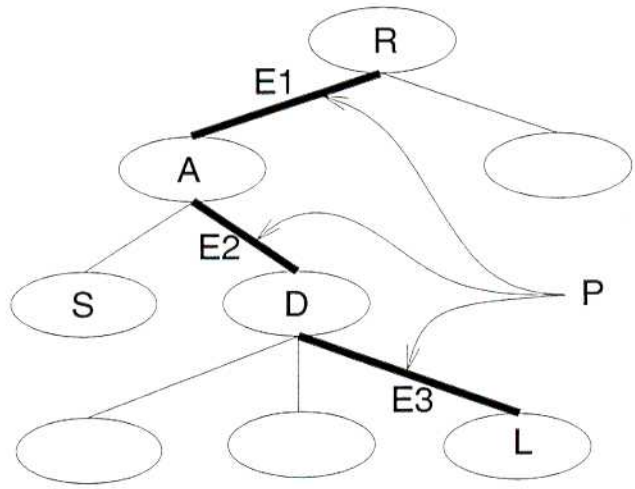


Figure 3.1: Structural components of a strict abstracted hierarchy: root, (R); leaf, (L); path, (P), from nodes (R) to (L) through nodes (A) and (D), along edges (E1), (E2) and (E3); D's parent/ancestor, (A); one of A's children/descendents, (D); siblings (S) and (D).

- non-cyclic $paths^{P}$, which consist of one or more $edges^{En}$ (also called *branches*) with each $edge^{eg.E1}$ joining a distinct pair of $nodes^{R,A}$.

We define an *abstracted hierarchy* in terms of its strategy for separating the structure from the data contained within it. A hierarchy is an abstracted hierarchy if:

- the organizational structure is a strict hierarchical structure [Knu73, AHU83],
- the organization of the data is completely separate from the data content.

Nodes in an abstracted hierarchy can have a set of any number of associated information components or attribute-values. Values can be any type of data.

By providing an interface that deals with information at an abstract level, the user can benefit from using a single data visualization and exploration tool for multiple data spaces, resulting in increased familiarity and understanding of the tool. The cost of this approach is that an application might be less "tuned" to deal with the unique qualities of a specific data set.

Task Dependent Visualization

Task dependence is the tailoring of a visualization to the specific traits of the data set and the tasks to be performed on the data. If a visualization tool is widely applicable to many varied data sets in an abstracted hierarchy, but makes no accommodations for the unique nature of each set, the visualization may be adversely affected. This is because the useful presentation or interface of the visualization may by dependent on the information content, its semantics and interpretation, and the actual tasks to be performed on the information. For example, the task of finding large and old files to delete is different from the task of finding files whose contents are similar to each other, in a loosely defined manner. Both of these tasks are very different from determining why the performance of stocks in a particular market sector experienced particular historical behavior patterns.

One might think that specialized interfaces would be required to carry out these tasks. In the software industry, this has historically been the solution chosen. Because they are built to satisfy specific requirements, specialized interfaces have the benefit of being more effective and efficient at doing what they do. Their disadvantage is that they do not deal with other data situations well (if at all), implying that users must learn new interfaces for each new application.

While it is arguably easier from the application development standpoint to build custom interfaces, there have been strong arguments for presenting users with interfaces that are generally applicable, consistent and predictable across different situations [Pre94, BO84].

Abstract - Dependent Balance

The goal of providing an abstracted, more generic interface conflicts directly with the goal of specializing the nature of the interface to specific information and tasks. Approaches that work within the confines of this conflict are listed below. These are ordered from a bias towards specialization (data or task) on top, to a bias towards

abstraction.

- having different interfaces for each data set *and* task,
- having different interfaces for each data set *or* task,
- having different interfaces with a similar functionality and "look and feel",
- having a single interface that is user configurable to various data,
- having a single interface that automatically accounts for data uniqueness,
- having a single interface that treats all data sets equally.

A single interface allowing either user configuration or automatic data accommodation provides the best option for flexibility between specificity and generality goals, because other options are too biased towards one goal or the other. Any method for dealing with an interface should allow the user to configure the interface to suit different task demands and data requirements.

3.1.2 Structure and Detail Presentation

There are various levels hierarchical information, from broad overviews to focused detail. As we will see, these levels compete with each other for limited screen real estate. The challenge for visualization designers is to reveal the various levels of information in an integrated, clear and coherent manner to foster analysis and convey insights.

Global Context

> If the two-dimensional information space fits completely onto the screen, there is no navigation problem...Users are never lost because they can see the complete information space. [BWI90]

Global context refers to the focus across a consolidated data set, its structure, and the trends and characteristics evident in a large cross-section of its nodes. It typically implies that all details are not visible. As information increases in volume and complexity, it is crucial to give users an overview of the hierarchical structure and how

particular data is situated within it. Without a good understanding of the global context, users may have difficulty acquiring a "big picture" of their information and how details fit within it. Without global context, a user may flounder around, lost in detail. With a visualization that neglects context and concentrates on detail, one "cannot see the forest for the trees" [Fei88].

The problem is that hierarchies can differ in size by many orders of magnitude. Existing visualization techniques cannot handle the differences. What works well for a hundred nodes or a thousand nodes may not work well for ten thousand or a million or more nodes. The scalability issue has become more pronounced as graphical representations of global structure have been implemented, and the structures depicted have grown. Previously, text based displays simply scrolled whatever would not fit in the 24 line screen (which itself was a major advance over non-scrolling displays). Graphical depictions of structure make more efficient use of screen space than textual or "text-layout-implies-structure" indented list displays. Up to a point, complexity can be displayed at higher resolution without overwhelming the user. This leads to effective visualizations that users can internalize and navigate through more easily [CWMS91]. Still, there is only so much that can be displayed on a single screen. Although monitor resolution is increasing, it does so at a rate far less than the increase in available information. Thus, resolution is a limiting factor in the scalability of visualization techniques. Scalability is one of, if not *the* greatest factor concerning the effectiveness of a visualization method. To illustrate the problem, look at common screen resolutions, whose upper limit is pushing 1600 by 1200. If each node takes a single pixel height in the display to show, barely one and a half thousand nodes could be shown at once. If both dimensions are used, there only exist about two million pixels with which to present both node information and the structure of the node organization. What happens when the information content exceeds these bounds?

Most approaches in Chapter 2 for viewing hierarchical information were designed for data spaces much smaller (e.g. 100's or 1000's or nodes) than are becoming common. For these reasons, many will not scale effectively to larger hierarchies. Without dealing with scalability problems adequately, information overload can occur as data

spaces grow — hiding a needle of desired information in a haystack of displayed data and users become lost in a poorly conveyed structure.

Local Detail

Local Detail refers to a node's content, its structural relationship to neighboring local nodes, and details of different nodes that can be compared. Local detail competes directly with global context for screen space. The first consideration when examining the need for local detail is the degree to which all of a node's content is shown. A second consideration involves the visualization bias, the degree to which all nodes are treated equally on the display. For example, early character-based file listings favored files at the end of the listing, since files at the beginning of the listing often scrolled off the top of the screen. As another example, allotment of screen space based on file size results in a bias favouring the subset of large files. As structures and presentations become more complex, some natural biases creep in. Examples of this are found in 3-D trees, where nodes nearer the root are less likely to be occluded by other nodes, or space filling systems, where the global structure is favored over local detail. In general, bias is towards outlying, maximal or minimal data, and against the majority of "average" data.

Detail-Context Balance

When faced with the conflicting goals of showing local detail and of showing global context, the balance between the two becomes crucial. The *detail-context balance* is the bias of a hierarchical visualization to favor either the presentation of detail or the display of structure. The delicate balance of these opposing needs is at the heart of most large information visualization systems. Every choice, whether evenly balanced or strongly detail or context biased, has positive aspects, and trade-offs.

Many methods focus on one goal, while neglecting the other. Systems that stress structure compress the visualization to the extent that little or no room is left for

detail. Those that present detail find that the screen space allotted limits information presentation to only a few nodes. Hence structure is not presented well.

Local detail can be integrated into a hierarchical visualization in one of two ways. It can either be situated or detached. Detail information is *situated* when screen space is allocated for it within its context, smoothly embedding detail within the structure. Situating data without smoothly integrating the detail into the context can lead to seams and incongruities between detail and context. It can also distort the structure presentation as space is allotted for data display. *Detached* detail is presented away from its context, in a separate panel or window. Detached detail displays can avoid the problems associated with situated detail, but they have the disadvantage that the user must look back and forth between the global structure and the separate display in order to assimilate the two. Ideally, the structure information and detail information would be clearly and seamlessly presented in one place.

Situated data hits scalability limitations quite quickly in practice, leading to panning or zooming to examine different areas of the data space. Multiple detached detail windows have the same problem, leading to frequent switching between overlapping windows. In either case, a user comparing data must remember values for relevant nodes if they are not simultaneously visible. This quickly overwhelms short term memory [Mil56]. Both approaches have their merits and the complexity of content and structure information must be considered when choosing an optimum solution. The presentation of detail impacts the quality of displayed relationships as well.

A user-variable balance allows customization of the display to the task at hand. This is particularly effective if user interaction is through a dynamic query control (discussed in detail in a following section). Such a control allows users to set the balance as part of the data exploration and extraction process. For example, the bias can be set towards providing structural information during navigation and then changed toward providing detail as they are narrowing down the scope of the search.

3.1.3 Data Properties

Seeing data relationships and data characteristics is vital to analysis and interpretation of data. Getting the overall sense of a large data set cannot be accomplished without examining nodes on a global scale.

Data Relationships

Data relationships are the similarities or differences between specific data points, such as which data values are bigger, smaller, newer, older, faster, slower, better, worse, etc. Views that show only one node's data at a time show relationships poorly, as the entire load of comparison is in the user's ability to remember the detail and mentally compare them. Views that present the detail of multiple nodes simultaneously, whether situated or in windows, let people find and examine relationships between data points. Of course, views that explicitly relate information reduce a user's burden even further.

Data Characteristics

Data characteristics are the trends and patterns that are evident when an overall focus of all nodes in the structure is taken. Trends and patterns may appear within substantial subsets of the data, or over the whole data space. By focusing on a global view of the data, rather than a detail view, nodes and subtrees that share characteristics may be more readily observed, assuming the visualization provides the required granularity of detail.

Most methods provide only basic support for examining the characteristics of a whole data set. This is usually limited to a single attribute at a time. Characteristics are commonly implied by a node's inclusion or exclusion in a display, or by ordering based on node values for an attribute.

3.1.4 Consistent Structural Visualization

Consistent structural visualization refers first to the presentation of the hierarchy structure, and second to the understandable nature of the strategy used to present the structure. The visualization of a hierarchy benefits from a consistent presentation of its structure. This eases the user's cognitive load of associating and synchronizing the visualization with the user's mental model or internalized image of the structure. For example, alphabetical ordering provides a predictable and constant visualization of character-based information. Ordering on numeric attributes also provides a stable reference for structuring the visualization. Similarly, the hierarchical structure shows the relationship between information components

There are now several familiar metaphors associated with the presentation of hierarchical data, such as rooted trees, and domain specific metaphors such as folders or files [Pre94]. Mapping a tree onto a familiar surface, like a wall or a sphere, provides an understandable mental model and maintains tree structure presentation. Initial familiarity with a stable structure will be reinforced and strengthened over time.

3.2 Visualization Techniques

The previous section examined the characteristics of the presentation of hierarchical data at a high level. These characteristics include the degree of abstraction, the global-local properties, the structural consistency of visualization, and other data properties. In contrast, *visualization techniques* are strategies for effectively conveying and modifying the presentation of information. Nine techniques for visualizing information are presented in this section and summarized in Table 3.2. Examples of visualization techniques that address data presentation characteristics include using distortion to balance detail and context, using animation to emphasize visual structural consistency, using data encoding to convey data characteristics, using dynamic queries to explore data properties, and filtering to focus on relevant local detail. While many techniques presented apply to general data structures, the focus in this thesis

will only be on their applicability to hierarchical information.

3.2.1 Graphical User Interfaces and Direct Manipulation

A *graphical user interface* (GUI) uses interactive graphical controls in a high resolution display device, typically to present metaphors and simulations of physical objects like buttons and toggles. Text-based systems, such as the original generalized fisheye views and early indented lists, used text mostly because of the technological limitations of their time. In contrast, all recent systems have a graphical interface. While text works well for conveying some type of detail, graphical methods can:

- provide a more complete and accurate visualization of the global structure;
- enact graphical metaphors; and
- take greater advantage of the color, space, size, and flexibility in the medium.

A mix of graphical and textual data can present both context and detail. A graphical interface allows use of a vast array of fonts of various types and sizes to display textual information, integrated into the graphical structure presentation. Other types of data, such as image data, can also be shown.

Direct manipulation is an essential characteristic of GUIs. This refers to an interactive interface where graphically represented objects (with affordances) are visually manipulated with continuous and immediate feedback of results [Shn87a]. In the Macintosh Finder, for example, one removes a file from the system by grabbing it with a mouse pointer and button press, and dragging it to the "trashcan" icon. A visual *affordance* refers to the usage implied by the look of an object. For example, raised buttons afford pushing, and sliders afford dragging [Nor88]. Most graphical systems now use direct manipulation, in varying degrees, to allow users to interact with the objects in the display.

3.2.2 Dynamic Queries

Dynamic queries is a direct manipulation visual exploration technique. Queries are characterized by the use of rapid, incremental, and reversible actions performed using visual controls [Shn87b, AWS93, AWS92]. Feedback is immediate and continuous through visual presentation of both query components and updated results. Use of this technique encourages exploration by its nature. Immediate visual feedback of quickly modifiable queries make dynamic queries excellent tool when looking for trends, characteristics and relationships in a data space. Dynamic queries may be compromised when performance degradation—caused by computationally demanding continuous visual updates or by slow searches—requires the use of *post-action* feedback, whose update is done only upon action completion. Post-action interfaces often use only a frame or outline of the object being manipulated to indicate what the result of the action will be.

The Dynamic Homefinder provides an example of a dynamic query interface used to search for houses to buy [WS92, WS93]. Through its sliders and graphical selection tools user's query attributes such as the number of rooms, price, and distance from a point. Whatever houses fall within the selected ranges across these attributes will be shown on a geographical map on the screen. When distance from a point is increased, homes in a wider area appear on the screen immediately. When distance is decreased (by grabbing the slider and moving it), outlying homes disappear from the screen.

Direct manipulation controls have now been coupled with the display artifacts such that the traditionally input-only or output-only controls now *both* have input and output capability [Thi90, AS94b]. Thimbleby calls this "equal opportunity". This techniques works well with dynamic query interfaces as well. For example, clicking on a house in the Homefinder display (with the "display" now becoming the input control) causes the dynamic query components to be assigned *that* data point's values (with the components now acting as output controls).

Dynamic queries are a mechanism for examining relationships among similar nodes. The query value is used as a threshold to modify the display. As the query is changed

in real time, the nodes that are simultaneously added or removed in the display are shown to have a similarity over the queried attribute. The highly visual and interactive nature of these systems provides an ideal way for people to explore information and gain an understanding about a large information space.

Dynamic queries also provides a means for determining characteristics within a data set by viewing data display changes. For example, as a slider is steadily moved from one end of an attribute range to the other, a few scenarios may occur. A majority of the data may appear or disappear nearly simultaneously, indicating that most nodes are quite similar, based on the attribute in question. Or, nodes may steadily appear or disappear, indicating an even distribution across the attribute. Many other variations are possible, such as situations where bi-modal distributions exist.

3.2.3 Animation

Animation is the simulated movement of objects within the display. Animation is useful for producing either continuous feedback or post-event updates. In both, animation is used to visualize the change in the display, which is particularly useful when the point of view changes or the structure or data presentation is altered.

The benefits of animation as a visualization technique lie strongly in the way humans perceive and store information. The human eye identifies movement very readily and the mind associates this movement with cause and effect. The mind is able to focus on areas of movement and compare the current image to recently viewed and remembered images for noting differences and changes. Studies have shown that visually animating a changing view of a constant object results in a lower cognitive load for reorientation than being presented with a new point of view in a static display [MRC91, RMC91a, RMC91b, RMC91c]. The mind can follow the animated navigation intuitively.Essentially animation can be used to reduce the user's cognitive load of tracking displayed structural changes by shifting the load to the perceptual system [RMC91a, MRC91].

3.2.4 Foci

A *focus* or *focal point* refers to the most interesting data point displayed. *Foci*, or multiple focal points, occur when there are multiple data points of greatest interest. The actual foci are dynamic, based on current search or exploration goals. Often more than one area of the data space needs to be examined. Since relationships between particular data points often need to be studied, the ability to examine detail information for multiple data points simultaneously greatly reduces cognitive overhead.

3.2.5 3-D Graphics

Three dimensional (3-D) graphics presents 3-D information on a flat (2-D) output device, where the graphical image gives the visual appearance of a third dimension, extending away from the viewer, in the manner as a two dimensional photograph captures a three dimensional environment.

We do live in a three dimensional world, and 3-D visualization seems natural. Expanding the hierarchical visualization into the third dimension reduces 2-D space requirements, since 2-D data can be virtually "layered'. For example, the 3-D hierarchical tree is an intuitively understandable organization. Using animation with 3-D displays reduces the cognitive load needed to track data relationships and to assimilate the changes in views after the data structure is modified. By exploiting abilities of the human perceptual system, in both the tracking of animated changes and the interpretation of a 2-D display as 3-D, greatly enhanced information visualization environments can be provided. However, 3-D graphics introduce an information problem with hidden information behind the visually front-most nodes. Since structures are displayed on a two dimensional output device, much information is hidden.

3.2.6 Distortion

Distortion is the modification of the visualization of a hierarchical structure typically

based on a familiar paradigm or mathematical model. Emphasis of foci is a key reason for using distortion. Distortion increases space allotment for foci, thus stretching its appearance, typically at the expense of the visual size of the rest of the hierarchy. Distortion is a common way to address the context-detail balance issue. Many varieties of visualization fall into the broad category of distortion-oriented views, including fisheye views, surface mapping, and spatial transformations.

Sarkar et al. and Leung and Apperly have presented the metaphor of a rubber sheet on a rigid frame to encompass the concept of distortion-oriented presentation, with the rubber sheet standing for the display area populated by data points [SSR93, LA94]. Distorted views are created by stretching the focal point(s) and shrinking the surrounding context areas of the display.

The stretching results in the magnification and demagnification of portions of the display. Magnification functions can be defined to mathematically describe the distortion either smoothly or in discrete levels. Smooth variance of magnification results in the typical notion of a fisheye view, with a smooth transition between detail and context. Discrete levels of magnification result in regions of higher magnification embedded in a less magnified context, with a sharp emphasis jump at their boundary. The drawback of this method is the existence of a space discontinuity along the boundary between the two magnification levels [KB92].

3.2.7 Data Encoding

Data encoding refers to the mapping of data attribute values to display features (such as size, color, intensity, hue, shape, simulated texture, and sound) [JS91]. Data encoding is used to extend the amount of information displayed by abstracting, grouping or mapping data characteristics into a more easily presentable and identifiable form. Encoding schemes may work with both discrete or continuous values. The use of data encoding can provide more bandwith in which to present information, but it can also increase the cognitive load of interpreting such a visualization.

Data encoding must be used with caution, as humans can only perceive a certain degree of complexity, especially when dealing with overviews and relationship between data points [Kuh90, Ell90]. The dangers associated with the use and overuse of color tend to be little understood by system developers [Rhe92, LHMR91]. However, color is one of the most efficient data encoding means for aiding fast and accurate decisions [Hoa90, Mac90, Ric91]. Color coding provides a strong tool for presenting trends on detail attributes with few possible values within a data space. Saturation, intensity or hue of color, and texture are more subtle visual cues. Gradations of grey can also be used in a manner similar though more limited than color [Fee92].

Using sound to encode data into visual display has the characteristic of being temporally located, which can grab attention or indicate a time dependent event or circumstance. By this nature, it is not as persistent as a visual cue, except through repetition. Sound encoding has the benefit of being perceptible even when a users' focus is elsewhere [Gav86, Bly89]. Incorporation of sound into the interface tends to be more difficult, as its use is less intuitive and more poorly understood by today's GUI designers.

3.2.8 Filtering

Filtering refers to the inclusion or exclusion of nodes from the display based on their individual interest value. An extended concept of filtering is used in this thesis, such that the include/exclude value is within a continuous range, rather than being a binary value [Fol90]. In this manner, the degree of visual emphasis can be based on this value, along with simple inclusion or exclusion when the value is compared against a threshold value [SG93]. To reduce confusion over the term, I call these *exclusion filtering* and *emphasis filtering*, respectively.

Filtering could prove to be one of the most useful techniques for visualizing large data spaces. Filtering is used where the overhead associated with dealing with excess information makes the data space unmanageable. The amount and degree of complexity of data relationships that can be usably shown is bounded by human

perception [Kuh90]. Thus, a means of presenting simplified relationships in reduced quantities can be useful, and even required as data complexity meets or exceeds the human perceptual limit. Without being able to filter or prune information, many types of visualization methods are overwhelmed by excess information, resulting in reduced display effectiveness.

Good examples of high volume information sources are usenet news or electronic mail, which is often organized by topic or person hierarchically, even though received chronologically [MGT+87]. People often have to deal with messages numbering in the hundreds or more in a single day, thus requiring assistance to deal with the onslaught [BR90]. Filtering provides users a means of focusing on relevant topics and messages while removing extraneous ones from the display. All that is required is some means of differentiating information. Nodes are ranked more or less attractive by their differing values for given attributes. Exclusion and emphasis is done based on this ranking.

Filtering can be useful in determining similarity relationships. For example, if the user wants to see all nodes of a certain type, one can emphasize them and, remove others from the display, which makes the relationship between the nodes obvious.

3.2.9 Clustering

Clustering removes related data and replaces it with a representative placeholder symbol. This symbol requires less screen space than the data it replaces, thus leaving more room for the other information in the display. Clustering has an advantage over filtering in that a reminder of the information is displayed even when the data itself has not been shown [SZG+96, Per93].

Clustering is a more specialized, structure-dependent technique than filtering. Filtering can be applied to basically any data by simply showing or not showing data on the screen. Clustering requires a relationship between data points to group them as well as an understandable placeholder. Proximity is the obvious choice for clustering, so the relationship shown by clustering is primarily structural rather than content

based. Clustering can only go so far before the expanded nodes on the path require too much room.

Clustering within the global structure tends not to be scalable, because deep navigation requires severe distortion of the display to expand the traversed path(s).

3.3 A Technique-Based Look at Visualization Approaches

So far, data-oriented presentation characteristics (Table 3.1) and visualization techniques (Table 3.2) have been examined in this Chapter as they apply to hierarchical information. Re-examined in this section are the visualization approaches listed in the previous chapter and summarized in Table 2.1 by seeing how they satisfy the characteristics and techniques listed in this chapter.

The comparison is summarized in Table 3.3. Ratings on the effective use of characteristics and techniques are based on the graphical version of these approaches, since these have a superset of the functional capabilities of their non-graphical counterparts. Included in the rightmost column of this table are ratings for FLEXVIEW, which will be discussed in the following chapters. Ratings do *not* imply a feasible limitation of any method. Rather, they are a rough indicator of the state-of-the-art for each approach. Most approaches *could* score well on most rating points, as current technology could be applied. These ratings, though subjective, are defensible by the background literature. Ratings are roughly the average, for each approach, of the illustrative systems, corrected for weighting towards historical systems.

3.3.1 Visualization Approaches in General

Before looking at individual approaches, this section examines the data-oriented characteristics and visualization techniques which are addressed equally well by almost all approaches. This section is organized by the characteristics and techniques of information visualization found in the left column of Table 3.3.

◯ Most Effective
o Somewhat Effective
· Neutral
× Somewhat Ineffective
X Least Effective
— Not Addressed

	Hierarchical Visualization Approaches: Zoom Navigation	Indented Lists	Fisheye Views	2-D Graphical Trees	3-D Graphicial Trees	Surface Mapping	Spatial Transform's	Space Filling	FLEXVIEW
Data-Oriented Presentation Characteristics:									
- Data Abstraction	·	·	·	·	·	·	×	×	o
- Data/Task Dependence	·	·	·	·	·	·	o	o	o
- Abstraction-Dependence Bal.	—	—	—	—	—	—	—	—	◯
- Global Context	×	·	◯	◯	◯	◯	◯	◯	◯
- Local Detail	◯	◯	◯	·	·	o	o	·	o
- Detail-Context Balance	·	×	◯	·	·	◯	◯	·	◯
- Data Relationships	×	×	o	·	·	·	o	o	◯
- Data Characteristics	·	·	o	·	·	o	o	◯	◯
- Structural Consistency	·	◯	·	o	·	◯	·	×	◯
Visualization Techniques:									
- Graphical Interface	◯	◯	◯	◯	◯	◯	◯	◯	◯
- Direct Manipulation	◯	◯	◯	◯	◯	◯	◯	◯	◯
- Dynamic Queries	X	×	X	X	X	X	X	·	◯
- Animation	·	×	X	×	◯	◯	o	·	·
- Foci	o	o	×	X	X	×	o	X	·
- 3-D Graphics	X	X	X	X	◯	o	X	X	X
- Distortion	X	X	◯	·	o	◯	◯	◯	◯
- Data Encoding	·	·	×	·	·	×	×	◯	·
- Filtering	×	X	o	·	·	×	o	·	◯
- Clustering	◯	◯	×	o	o	X	o	×	·

_____ demonstrates/uses _____ in an _____ manner.
(Approach) (characteristic/technique) (grid value)

Table 3.3: Comparison of Hierarchcial Visualization Approaches.

Focus on Data-Oriented Presentation Characteristics

Most of the approaches to hierarchical visualization in Chapter 2 take the middle ground between abstraction and customization. They are not tuned to a single task or data set. Nor are they able to generalize data-specific traits out of the interface.

Excepting zooming and indented lists, the approaches tend to present global context better than local detail, mostly because their graphical nature motivated their designers to draw the actual structure. As research in this area continues, new ways of incorporating and synchronizing the detail and context together are being developed as ably demonstrated by advances in fisheye views.

Examining characteristics of data (such as trends by data encoding) involves taking a global view, so it is not surprising that most approaches, particularly those that present global context well, do a good job of showing them. Seeing relationships within data requires one to compare the detail of various nodes, so approaches which favor the presentation of context over detail generally fare less well. When balancing the detail and context within a visualization, distortion based approaches (fisheye, surface mapped, and spatially transformed views) stand out over other approaches, because of their nature to integrate detail and context together.

The focus on global context also implies good structure presentation and all approaches provide a consistent *strategy* for presenting data. However, some issues remain. First, there are difficulties navigating in three dimensional structures, especially when using two dimensional input and output devices. Second, sacrifices to efficiency, such as *Treemaps* maximization of screen space through an unconventional hierarchy representation, leads to an inconsistent *structure* presentation when the orientation flips between horizontal and vertical. The noticeable problems make these approaches more difficult to use.

Focus on Visualization Techniques

All approaches could easily be adapted to a direct manipulation, graphical interface.

Animation is used in only a few approaches to show changes in complex displays. However, animation is used to present other relationships, such as the expansion and contraction of zoom windows in windowing environments.

A few approaches provide limited access to independent foci, but generally systems present foci poorly, with their location dependent on location within the structure, so that simultaneous viewing is not guaranteed.

Distortion is common and well used in several of the approaches. Other techniques, like data encoding, filtering, and clustering are used at a simple level by many systems. However, few approaches make use of the more powerful capabilities of these techniques. For example, some degree of information filtering is possible in nearly all approaches examined here. Yet most simply reduce the data displayed by specifying either which data to display or which data to exclude from the display, based on a single attribute common to all nodes in the hierarchy. More recent systems introduce more sophisticated means of filtering. *Treemaps* is one example in which filtering has been introduced to allow users "to concentrate on features of interest" and "to see only [the information] satisfying certain properties" through emphasis [TJ92].

Information overload is a serious problem. The scalability of techniques to larger data sets is a prime concern. *Cone Trees* can handle only about a thousand nodes before its usability threshold is reached [RMC91a]. *Treemaps* can handle two or three thousand nodes [JS91]. Data spaces could easily be a few magnitudes larger in size and these systems could not cope. Filtering is a powerful method of reducing the amount of information to display, while simultaneously improving the quality or importance of the displayed information. *Generalized Fisheye Views, Fisheye Planar Graphs, Treemaps*, and *Variable Zoom Views* use filtering or clustering to reduce the amount of information on the screen, preserving structure and allowing access to detail.

3.3.2 "Zoom" Navigation

This and following sections examine the unique strengths and weaknesses of each hierarchical visualization approach. These sections are organized by the approaches

across the top of Table 3.3.

Windowing systems do not scale well, as a few windows quickly fill the display area. Hence only very small hierarchies could be completely shown. Zoom techniques present the structure *very* poorly, for only a single node's children are visible at a time. However, this leaves more room available for presenting local detail.

Zoom navigation is a method to focus on data of interest. In the *Treemaps* application, (discussed in detail in section 2.1.6 on Space Filling), all information for a whole hierarchy is displayed in a partitioned screen. As detail could be difficult to discern at the top most node, zoom navigation allows the full screen to display only a partition of the hierarchy, in greater detail, filtering out the rest of the hierarchy.

Animation has a role to play in conveying the structural relationships between parent and child nodes in windowing systems. Relationships are temporarily visible through the animated expansion or contraction of nodes. This temporal connection is quite weak, because of its non-persistent nature and its reliance on human memory.

3.3.3 Indented Lists with Holophrasting

Indented lists with holophrasting allow multiple paths to be independent and simultaneously expanded for easier node comparison. However, the nodes cannot be arbitrarily moved or located in an indented list, so there is no guarantee that *any* two nodes could be compared side by side. In fact, they could be so far apart in the structure that they cannot be viewed simultaneously at all.

3.3.4 Distortion and Filtering through Fisheye Views

BASS trees orient the focal path around a fixed location, in this case the left side of the display. Selected nodes are shifted to the leftmost position under its parent and its siblings are reorganized. This means that the hierarchy structure is constantly being rearranged to left-order the display of the selected path.

Fisheye views are an intuitive way of looking at the problem of balancing detail within context. However, the metaphor is stretched when foci are introduced.

3.3.5 2-D Graphical Trees

CAT trees suffer from structure reorganization similar to that of BASS trees, discussed in the previous section. With CAT trees, the path to the selected node is always straightened to present it vertically centered in the middle of the screen. This means that nodes on any given level are continually being shifted to the right or left to bring the node on the selected path to the center of the screen.

Trees, both 2-D and to a lesser extent 3-D, are difficult to fit nicely into a square display space. Two problems trees suffer from is that space in the two corners on the same side of the root are often blank and unused, and that their shapes are usually quite "triangular", being quite wide at the leaf end. Trees quickly become too wide to fit in a display, without resorting to distortion or panning to examine areas that do not fit in the screen.

3.3.6 3-D Graphical Trees

Cone Trees presents information in about one third the screen space taken by a conventional 2-D display, but much of its information is implied and not actually shown. Instead, the information is "hidden" when it is behind other nodes. Layers of information in the third dimension hide other layers of information. This problem increases with the size of the hierarchy. As each sub-hierarchy is wrapped around its own invisible cone, the depth of the three dimensional image increased, as do the levels of information hidden by occlusion.

The current node and path may be hidden within the depth of the tree. To deal with this, the path is often brought to the front of the display. Animation is used by *Cone Trees* when rearranging the structure of the hierarchy to overcome the occlusion problem. When a node is selected (and becomes the point of interest), layers

representing the directories on the path of the selected node are rotated to bring the path to the front of the tree. While the node at the front of the display still may hide many nodes behind it, the animation-enhanced data manipulation helps users deal with occluded nodes.

The primary reason for the use of animation is to reduce the user's cognitive load by exploiting people's natural ability to track animated changes in orientation and display of the hierarchical structure. In *Cone Trees*, half a second to a second is spent animating the view change of the large tree structure on screen. This appears to be time well spent, as is apparent by the video demonstration [RMC91b]. When the animation feature is turned off, the new view is displayed instantly, but the user has to spend a number of seconds reorienting the view with his or her internal representation of the structure. When animation is employed, the time spent to keep the tree structure constant and oriented in the user's mind is reduced.

3.3.7 Distortion through Surface Mapping

As with *Cone Trees*, the *Perspective Wall* employs animation to display changes when the focal point is moved. The wall shifts either right or left (bending around the corners of the stationary front segment) until the selected information is centered in the front segment. This eases the user's task of tracking the changes in the display of the information and locating the focal node within the entire structure. Animation is also used to expand or compress the density of the information and the detail for each node presented on the wall, by stretching out or reducing the length the side segments extend towards the horizon. This provides a way to balance the amount of information presented in the detail (front) segment and the context (side) segments.

3.3.8 Distortion through Spatial Transformations

In their work on planar graphs, Sarkar and Brown put forth the idea of an arbitrary importance value to determine whether to display a node and what a displayed node's

emphasis should be [SB92a, SB92b]. This importance value included a node's visual worth (a combination of its distance from the focus, its application dependent API, or other arbitrary factors), the detail for the node (dependent on the size of the view and the maximum detail that can be displayed), and the size of the node within the display. A user defined importance factor, possibly based on dynamic criterion, provides an excellent means of tailoring specific information presentation. This graphical method also introduced filtering by varying emphasis, rather than simple inclusion or exclusion. Figure 2.13 presented a graph of major cities in North America. On the left is an unfiltered view in which all nodes representing cities are equal in size. On the right is a filtered view in which each node has been represented in a size corresponding to its arbitrary importance value and removed if this value was below a given threshold. This leaves many fewer points of information to deal with, each emphasized according to its importance level.

The *Variable Zoom View* concentrated on the presentation of the global context. The whole structure is clearly visible, yet some detail remains on even more distant nodes. Clustering is used effectively. The *Variable Zoom* algorithm implemented hierarchical clustering of a network of nodes, recursively clustered by proximity. This can be seen in Figure 2.2, where the clustered nodes labeled "South" and "Central" are symbols representing their subviews, and the clustered "North" node has been expanded to present the interior detail it represents. Details were hidden in the multi-focal fisheye view through clustering to provide a better view of the structure.

3.3.9 Space Filling

Structurally inconsistent visualization may result from application of certain visualization techniques. The *Treemaps* implementation of the space filling method results in files horizontally oriented at one level and vertically oriented at another. This discrepancy is worsened when a node is moved up or down an odd number of levels and a reorientation from horizontal to vertical occurs, and vice versa. This inconsistency increases the difficulty users have resynchronizing the visualization with their

internalized model. Even if the resynchronization were easier, the complexity posed because of the inconsistent structure presentation causes an unnecessary cognitive load.

Space filling and graphical cues present both item detail and structure context. A dynamic control of *Treemaps*'s node boundary pixel width allows the user to stress local details (on a global scale) or stress the hierarchy structure by decreasing or increasing the border pixel widths, respectively. Since screen usage is maximized at all times, increasing or decreasing the border size inversely affects the amount of area left to show data within the border.

Treemaps allows the user to examine attributes based on a toggled linear or logarithmic scale. The logarithmic scale results in an intuitive emphasis on nodes at the "high" end of the scale and de-emphasis on nodes at the "low" end of the scale. This enhancement allows *Treemaps* to display maximal data effectively.

Treemaps is one of the few systems to use data encoding. *Treemaps* maximizes screen space usage through the use of graphical cues. The size, color, intensity and orientation of nodes are used to encode attributes of the information. When *Treemaps* are applied to file hierarchies, the attributes include file size, type, or creation date. This effectively presents trends in and characteristics of the data. For example, type can be mapped to color or date mapped to intensity.

3.4 Conclusions

This chapter focused attention on how hierarchical data-oriented visualization characteristics are addressed and on the visualization techniques used for visualizing large hierarchical data spaces, as summarized in Tables 3.1 and 3.2. Example systems illustrated the approaches. This concluding section presents a preliminary set of guidelines in Table 3.4. These Because these guidelines follow directly from the discussion in the chapter, they will not receive further coverage here. These guidelines address the essential characteristics and techniques for visualizing large hierarchical information

spaces. They focus on the high level functionality requirements for a visualization system. While most are derived directly from observation of the key features of visualization approaches, a few guidelines are indirect consequences of the studied work, and others fill gaps left by current systems.

After generating the set of guidelines from the data-oriented characteristics and visualization techniquesis, the guidlines were organized into five categories of related characteristics:

- basic presentations characteristics;
- advanced presentations features;
- facilitating tasks;
- taking user abilities into account; and
- other more theoretical characteristics.

The table has three columns of data:

- a four character *guideline code* that will be referred to later;
- the visualization guidelines; and
- the techniques to use to meet these guidelines.

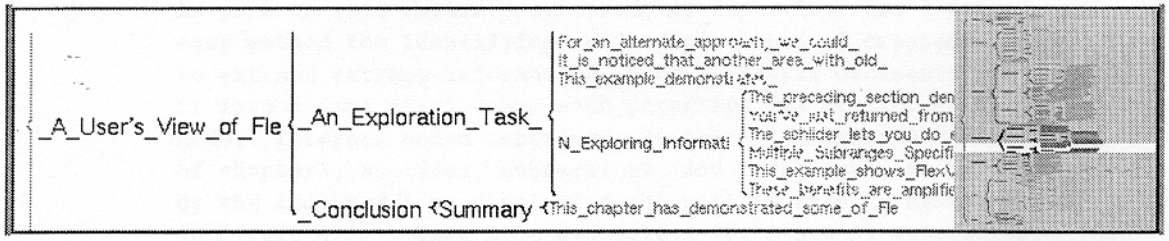
The table is read with the fill-in-the-blanks sentence at the top. For example, *"A hierarchical visualization system should show detail (local) information, by using dynamic queries, 3-D graphics, distortion, data encoding and clustering"*. Included with each technique is the section number where it was discussed.

The presentation of the relationships and trends across the data is becoming more crucial for management and analysis, so global context is increasingly included in the visualization. This commonly leads to information overload, as too much information is shown on screen. More innovation is required to deal with the demands placed on information visualization strategies by the current and anticipated amount of available information.

A hierarchical visualization system should _____,	by using ____. (§Section(s))
1) *Basic Information Presentation Requirements*	
DETL) show detail (local) information	technique: §3.2.5,6,7,9
CNTX) show context (global) information	technique: §3.2.5,6,8,9
DBAL) balance detail and context effectively	technique: §3.2.6,7,9
RELN) allow node-node comparisons	foci §3.2.4
TRND) present multi-node trends	technique: §3.2.5,6,7,8,9
FOCI) allow concurrent, distinct foci viewing	foci §3.2.4
2) *Advanced Information Presentation Features*	
<*VOL*) reduce the information volume shown	filtering §3.2.8, clustering §3.2.9
SCAL) be scalable	filtering §3.2.8
SPAC) present data space-efficiently	data encoding §3.2.7, if applicable
3) *Facilitate Tasks*	
EXPL) encourage data exploration	dynamic queries §3.2.2
NAVN) enhance navigation within the structure	direct manipulation §3.2.2, consistent structure §3.1.4
4) *Accommodate User's Abilities*	
STRC) preserve the hierarchical structure	clustering §3.2.9, 3-D graphics §3.2.5
COGN) reduce user cognitive load	metaphors, user's background, user cultural cues
PRCP) exploit human perceptual abilities	data encoding §3.2.7, animation §3.2.3, 3-D graphics §3.2.5
5) *Related Issues*	
SPEC) deal well with specific data sets	interface tailorability
ABST) be as widely applicable as possible	abstracted hierarchies
FLEX) be flexible	customization, automation, dynamic interface creation
ENVR) integrate into a user's environment	standardized interface
DATA) allow seamless, dynamic data access	dynamic or live database
OPTM) be fast (enough)	optimizations

Table 3.4: Guidelines for Hierarchical Information Visualization

Chapter 4



A User's View of FlexView

The next three chapters introduce and examine FLEXVIEW, a prototype information visualization tool whose design is based upon the guidelines from the previous chapter. As will be shown, FLEXVIEW presents large hierarchies in a manner that invites exploration and searching of the information space. It does this by emphasizing both detail and context through its use of filtering, fisheye distorted views, and dynamic queries.

To set the scene, this chapter begins with a scenario of a user working on a hierarchical structure. The scenario is used to illustrate FLEXVIEW's visualization concepts and provide context for the discussion that follows. The discussion centers on the three main components of the FLEXVIEW system: the filter; its handling of detail; and its overview windows. Central to much of this is its use of dynamic queries. The chapter ends with a more complex tasks example that show off some of the subtler aspects of flexview.

Chapter 5 focuses on the implementation details of FLEXVIEW: the hierarchy structure layout algorithm; the node data structure within the hierarchy; nodes emphasis and filtering functions; and various system environment setup details. Chapter 6 critically examines FLEXVIEW against the visualization guidelines of Chapter 3. Some informal user testing is discussed, and that chapter closes with a discussion of the strengths and weaknesses of FLEXVIEW.

4.1 Using FlexView: A Simple Scenario

```
Title      Visualizing Large, Loosly-Structured, Hierarchical Information Spaces
Chapter    A User's View of FlexView
Section    Using FlexView: A Simple Scenario
Paragraph  As work on this thesis progressed, my supervisor and I wanted an
           easy method for identifying revised portions.  I created a script
           to extract extract information  from my thesis documents and dump
           it into a data file, where each paragraph was treated as a leaf
           node.  Internal nodes contained the thesis' hierarchical structure
           of chapters, sections, subsections, and paragraphs, as illustrated
           by the indented list displaying the path to this paragraph shown
           in Figure 4.1.   Each node had values for a set of attributes,
           such as the judged ''completeness'' of each paragraph, the number
           of revisions, the recency of revisions, the size of the paragraph,
           the beginning or ending page number of the document segment or the
           document segment type (ie. text, figure, table, or mixed).
```

Figure 4.1: An example paragraph as a leaf node in a FLEXVIEW thesis hierarchy.

As work on this thesis progressed, my supervisor and I wanted an easy method for identifying revised portions. I created a script to extract information from my thesis documents and dump it into a data file. Each paragraph was tagged as a leaf node. Internal nodes contained the thesis' hierarchical structure and text titles of chapters, sections, subsections, and paragraphs. This is illustrated in Figure 4.1 by the indented list displaying the path to this paragraph. Each node had values for a set of attributes, such as the judged "completeness" of each paragraph, the number of revisions, the recency of revisions, the size of the paragraph, the beginning or ending page number of the document segment or the document segment type (ie. text, figure, table, or mixed).

I created a simple node description file, listing the attributes, their types and, optionally, possible values they might take. (This file is discussed in the implementation chapter following and shown in Figure 5.8, top.) FLEXVIEW uses this file to automatically create the filter specification control—used to select interesting or important ranges of values for one or more attributes. The node description file is also used to format the input data file. After completing this setup I start FLEXVIEW (Figure 4.2, top). Filter controls are on the bottom, and two views of my data on top with the left window showing detail and the right providing an overview.

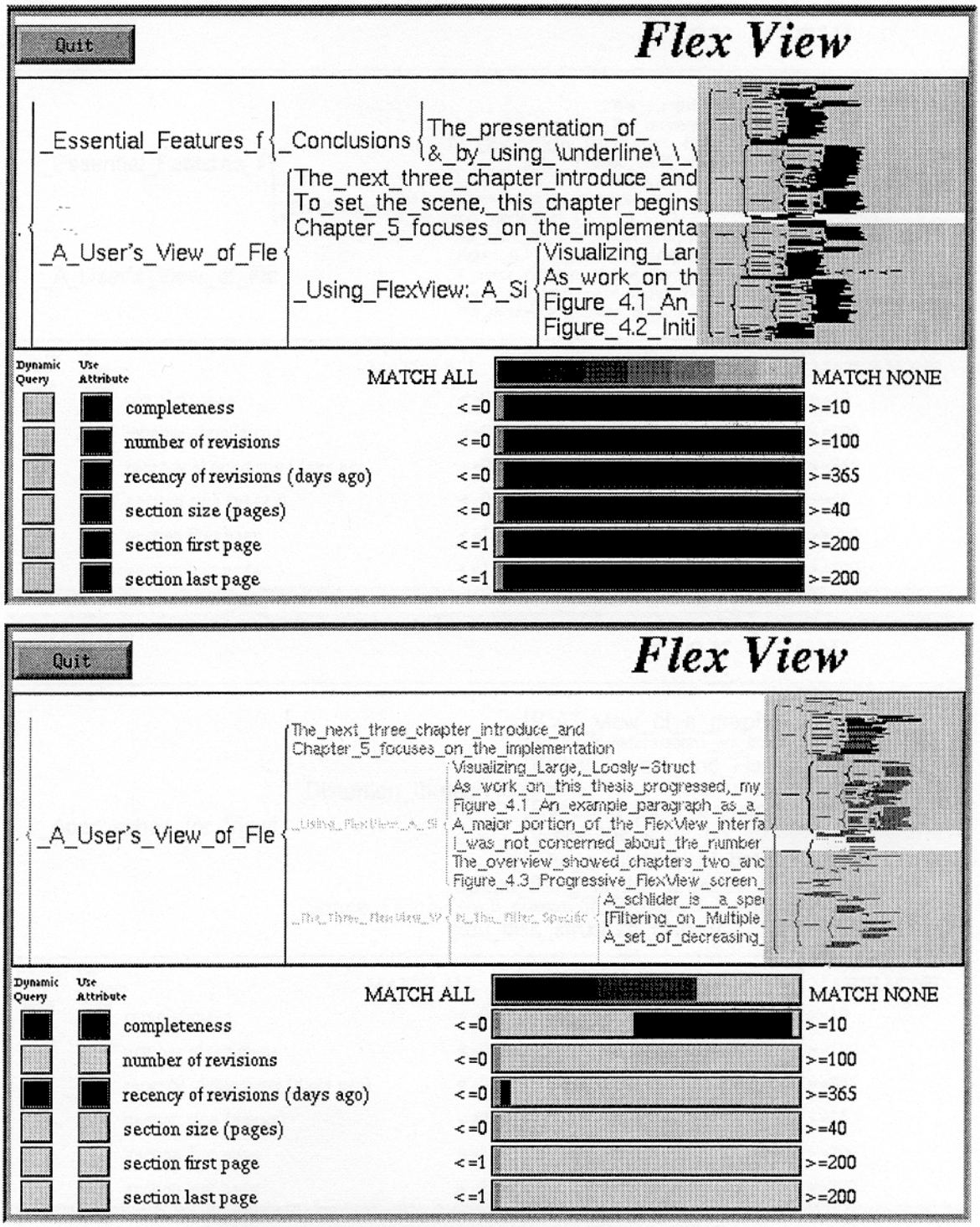


Figure 4.2: Initial FLEXVIEW thesis screen dump (top), and after attributes turned off, dynamic queries turned on, initial filters set (bottom).

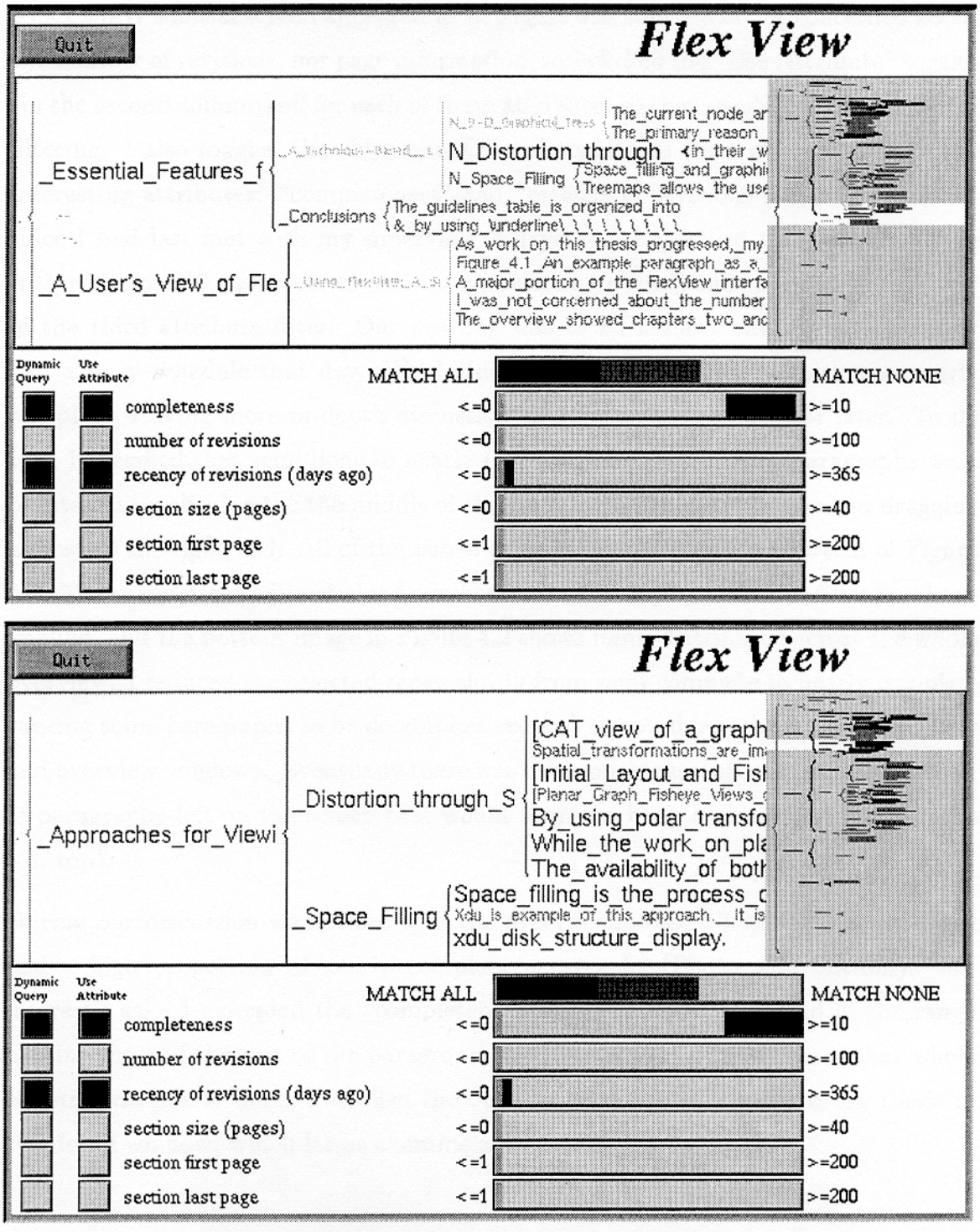


Figure 4.3: Continuing FLEXVIEW screen dumps of thesis showing restricting the completeness attribute filter (top), and relaxing the completeness filter, emphasizing a nearly finished chapter (bottom).

FLEXVIEW's initial screen appeared as in Figure 4.2, top. I was not concerned about the number of revisions, nor page information, so I clicked the "Use Attribute" toggles (in the second column) off for each of these attributes so they would be ignored during filtering. I also toggled the "Dynamic Query" toggles on (first column) for the two interesting attributes: "completeness" and "recency of revisions (days ago)". Next, since I had last met with my supervisor 3 days ago, I specified that revisions done within the past 3 days were of interest by selecting a small sub-range at the left end of the third attribute filter. Our meeting was to be a short one as my supervisor had a busy schedule that day. We decided to look only at work which was nearly complete, leaving more in-depth discussion on rougher paragraphs for later. To do this, I specified that semi-done to nearly (but not completely) done paragraphs were of interest by clicking the the middle of the completeness *schlider* range and dragging almost to the right end. All of the above changes are shown in the bottom of Figure 4.2. Too many items still remained emphasized in the display after this filter had been applied, and the bottom image in Figure 4.2 shows items distributed across the whole overview. I reduced the selected range slowly from semi-complete to nearly complete causing some paragraphs to be de-emphasized and removed from both the detail view and overview windows. Eventually there was a reasonable quantity and concentration of paragraphs left on the screen that would form the basis of our discussion (Figure 4.3, top).

During our discussion we noticed that the overview showed Chapters two and three had a high proportion of nearly complete paragraphs (Figure 4.3, bottom). For interest's sake I extended the "completeness" filter to the "done" end of the range causing most of the rest of the paragraphs in Chapter 2 to appear. Since that whole chapter was nearly done, I scrolled the overview to Chapter 2 to show the thesis in the detail window, which let us examine and discuss it.

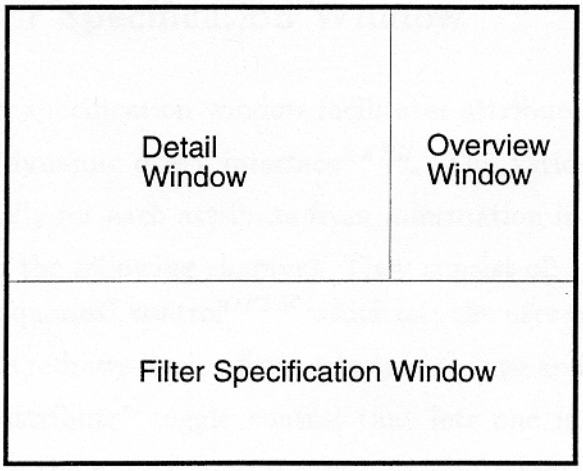


Figure 4.4: The Three Component Windows of FLEXVIEW.

4.2 The Three FlexView Windows

This section contains a detailed examination of FLEXVIEW's interface components. The FLEXVIEW screen is divided into three distinct windows: the filter specification window; the detail view window; and the overview window (Figure 4.4). The filter window provides a graphical interface for exploring the information space through dynamic queries, and lets a user tune the data presentation through exclusion and emphasis filtering. The detail window presents detail by text and data encoding through node emphasis. The overview window provides global context by compressing the entire hierarchy and by showing how the detail fits within it. In the scenario, I used the filter window to specify that nearly done work was most important to me. This showed a number of emphasized files in the detail window, while a rough indication of the proportion of nearly done files was included in the overview window. These three windows will be discussed more thoroughly in the following sections. Throughout this discussion, references will be made to the guideline points in Table 3.4, by superscripting the guideline code e.g. "...by using filtering$^{<VOL}$ and...".

4.2.1 The Filter Specification Window

The FLEXVIEW filter specification window facilitates attribute queries and data exploration through a dynamic query interface[EXPL]. The various query controls are generated automatically for each attribute from information in the node description file[ABST] (discussed in the following chapter). They consist of:

- a "use dynamic queries" control[OPTM] which lets the user toggle between continuous, immediate redraws during filter selection change and post-action redraws;
- a "use/ignore attribute" toggle control that lets one include or exclude the attribute filter in the node importance value calculation;
- the attribute description, which is a user-understandable label;
- a *schlider* attribute filter selection control[1] [SPAC] for sub-range(s) selection.

Node filtering is a two stage process, not counting the "Use Attribute" toggle. The first layer of filtering is done at the attribute level (using the query mechanism described above), to arrive at a *node importance value*, which is the count of passed attribute filters. This count becomes an indicator of how interesting a node is. This value is then put through a second stage of filtering to define a node's presentation characteristics (its emphasis, color, and size). Any or all attributes may be used to determine the composite interest level or importance value of a node. By setting the overall node filter, nodes can be filtered into or out of the display, and those shown can be emphasized or de-emphasized, based on their importance value. In this manner, information overloading[SCAL,SPAC,COGN] can be reduced by suppressing less important data from the display.

Filtering on a Single Attribute

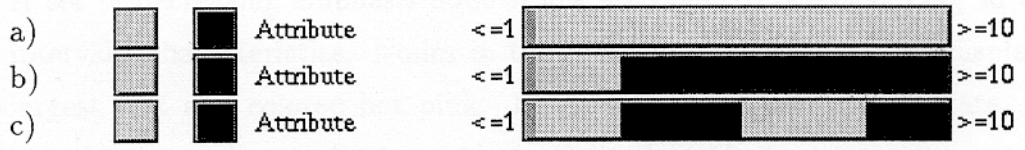
The simplest filtering is done on is a single attribute. A node attribute value that falls within a selection sub-range passes the filter. A value outside the selection sub-ranges fails the filter. With only a single attribute, passing or failing a filter is akin

[1] Notice that discrete variable "type" has not been implemented (Figures 4.2 and 4.3). FLEWVIEW currently deals only with continuous values. Discrete string items could be selected with either alphasliders [AS94a], or selection lists, or other means. Sound and image selection objects would be less simple but could still be integrated into the interface.

to inclusion or exclusion in the display, respectively. For example, if I had clicked all attributes off except for the recency of revisions in the scenario, then nodes revised within the past three days would pass the filter and all others fail. In this situation, it doesn't make sense to change the Interval Emphasis control from the default from a single attribute, since it is just another binary filter. The interval emphasis control will be discussed shortly in the section on filtering multiple attributes.

Schlider Filter Specification

A *schlider* is a specialized slider control designed for selection of any number of sub-ranges across a continuum. This is used to adjust the attribute filters. The left mouse button is used to select filter sub-ranges for node inclusion by clicking and dragging left or right across the selection pane. The right button removes selected sub-ranges. Figure 4.5 (top) shows a schlider with a range from 0 to 10 with no selections. In the middle image, a sub-range from 2 to 10 has been selected. The bottom image shows that a sub-range from 5 to 8 is deselected (the selection removed), which creates two selected sub-ranges, 2-5 and 8-10. Creation of subranges can be repeated anywhere in the control. FLEXVIEW derives its name from the flexibility and precision the sub-range specification ability allows.



a) No ranges selected for inclusion.

b) A single range (2-10) selected for inclusion.

c) Two ranges (2-5, 8-10) selected for inclusion by deselecting a range (5-8).

Figure 4.5: Filtering on Multiple Sub-ranges.

Restricting a filter is reducing its selection, resulting in fewer nodes passing and lower importance values. *Relaxing* a filter is increasing its selection, resulting in more nodes passing and higher importance values. In the thesis scenario, narrowing the "completeness" attribute filter from semi-done towards "almost done" demonstrated restriction of the filter, while extending the filter from "almost done" to "done" demonstrated relaxation of the filter.

Filtering on Multiple Attributes

Node level filtering becomes useful when multiple attribute queries are formed. The user specifies how (or whether) to display nodes that pass a given number of attribute filters with the Interval Emphasis control. In this control, there is one more interval than the number of attribute filters, with the extra interval allowing for the case where no attribute filters are passed.

Node Interval Emphasis

The Interval Emphasis is shown in its default configuration from high (black) to low (light grey) emphasis[2] in Figure 4.2, top, at the top of the filter specification window (between "Match All" and "Match None"). A node *emphasis state* is a definition of display characteristics, such as font size and its contrast (by color or shading)^{PRCP}. A set of decreasing emphasis node states are defined, corresponding to the default intervals' characteristics. Nodes in the most emphasized state are displayed in the largest font and colored hot pink. In the next most emphasized state, node fonts are slightly smaller, and colored black. From there, font continues to shrink, and the color fades through to a light grey. Nodes are emphasized with fonts of a usable, readable size (e.g. fonts from 6 to 14 point). Nodes in the "least important" state are excluded from the display$^{<VOL}$. Figure 4.3 (middle) shows a number of nodes with

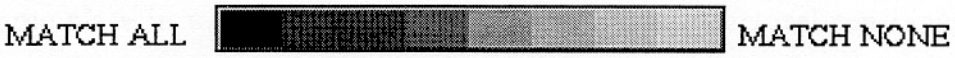
[2] FLEXVIEW uses color. For the sake of this thesis' black and white figures, I have adapted a colorless emphasis scheme. The ordinarily highest emphasis color, hot pink, has been removed and all black and white colors shifted according to take its place, because black has the most contrast in a black and white display.

different emphasis levels, including **N Distortion Through S** (very important) and including **N Space Filling** just below it (not nearly as important). The displayed emphasis encodes high level information about all nodes attributes.

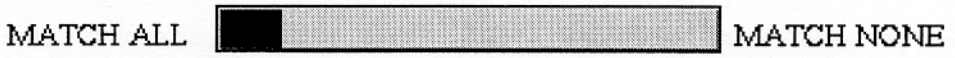
4.2.2 The Detail View Window

By default the "Match All" interval is the most emphasized, the "Match None" interval is the least emphasized, and the other intervals and emphasis states are paired in order between these two. Users can cycle through all emphasis states to change the default emphasis associated with an interval^{FLEX}. Figure 4.6 shows three examples of Interval Emphasis selections. The first shows the default emphasis. In the second, all intervals except for the "Match All" have been toggled to the least emphasis state which removes it from the display. The third selection emphasizes nodes passing a single or no attribute filters, with the single attribute emphasized less (lighter and smaller). The adjustability of the interval's emphasis levels gives the user simple, complete control over the presentation of nodes of any interest level.

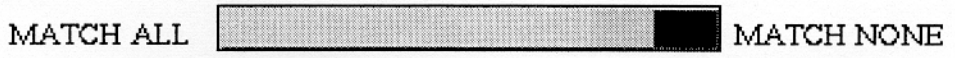
In a pure fisheye display nodes are emphasized or de-emphasized strictly based on proximity to the focal point, so that distance from the focal point becomes the emphasis function. In turn the focal point in the center of the view is enlarged, and



Default declining emphasis states.



"Emphasize nodes passing all attribute filters"



"Emphasize nodes passing only one or no attribute filters"

Figure 4.6: Three examples of filter specifications with the Interval Emphasis control.

nodes progressively further from it are progressively shrunk. FLEXVIEW uses a generalization of this idea. In it, each node in the information space has an importance value used to determine node emphasis. The detail view stresses interesting nodes, so unimportant nodes can be removed from the display to make room for more important nodes. Nodes of highest importance are considered focal nodes or nodes of interest^{DETL}.

4.2.3 The Overview Window

To satisfy the need for both detail and context the view was split horizontally into the detail view and overview^{DBAL} (Figure 4.4). The detail view shows as many nodes as fit in the display that pass the node (and attribute) filters^{DETL}. The two views share the same layout algorithm, with two main differences. First, the overview proportionally compresses the hierarchy vertically until it fits within the available space^{CNTX}, as well as shrinking the view horizontally to keep the aspect ratio (height: width) reasonable. The fact that the same algorithm is used means that any emphasis or exclusion done in the detail view is done in the overview as well. The difference is that overview nodes are always one pixel tall, though their widths are adjusted proportionately to their original size. Secondly, when nodes are excluded from the overview, the screen space they occupied remains allocated to preserve a stable structure ^{STRC}. This means that if the filter specification changes to include these nodes, they reappear in the exact same place in which they existed previously. The structural constancy of the hierarchy displayed in the overview enables users to stay oriented in the information space, reducing the cognitive load required after updating a view. Visible nodes take on the same coloration in both views. This color coded information appears at a glance in the overview to highlight trends^{TRND} and relationships^{RELN} within the data space.

4.2.4 Linking the Overview and Detail View

The portion of the hierarchy visible in the detail window is visually indicated in the overview. This is done by using the same background color for both the detail view and the corresponding portion of the overview. The rest of the overview has a darker background color. This is done to give the viewer a visual cue about which portion of the overview relates to the detail view.

The overview is actually an interactive control. It acts as a scroll bar for the detail view. As with normal scroll bars, dragging results in a scrolling operation^{NAVN}. When a mouse click occurs anywhere in the overview, the node clicked is centered in the overview^{NAVN}. The focus of the views can also be changed by clicking on a node within the detail view, causing it to be shifted to the center of the screen.

These two views working together perform three jobs. First, they provide a detail view of a portion of the structure while showing a representation of the entire hierarchy. Second, they show where the detail portion fits into the whole structure. Third, they allow a user to quickly navigate to regions of nodes.

4.3 Using FlexView to Visualize a Complex Scenario

Often relevant data is hidden in volumes of context. The view needs to be reduced from looking at almost *everything* to looking at just the relevant, important information. The following scenario illustrates an advanced use of FLEXVIEW.

Clutter is a common problem on hard drives, yet cleaning them up and keeping them clean are tasks than invariably get postponed until necessity strikes. Let us consider how FLEXVIEW can simplify this task. Good candidate files for removing or archiving are typically old files or large files, and preferably both. A search through the file space may reveal unused programs, defunct projects, duplicates, backups, intermediate copies of various project files, and even help files that are no longer useful. How could these files be found through FLEXVIEW?

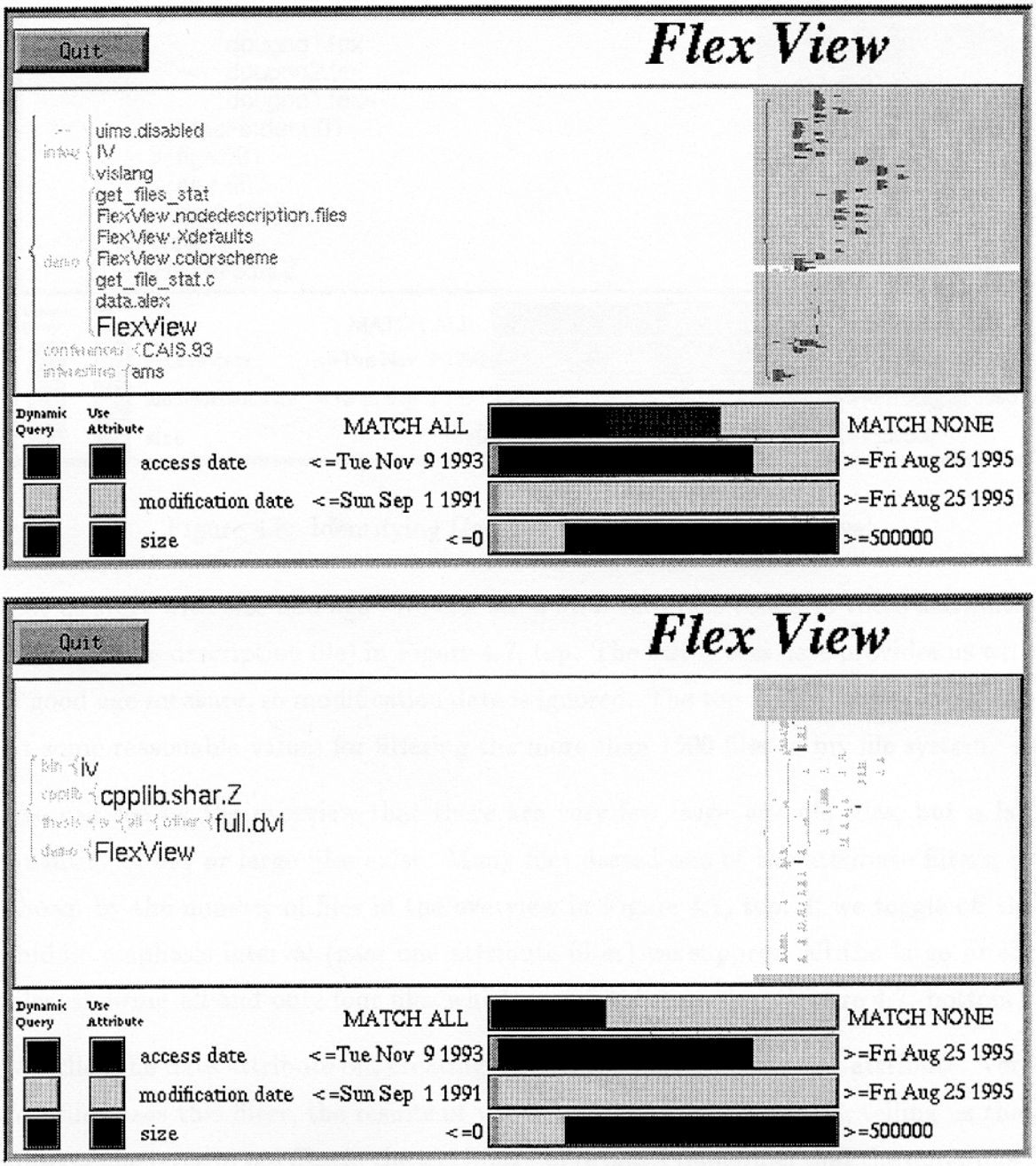


Figure 4.7: Identifying Good Candidates for File Removal.

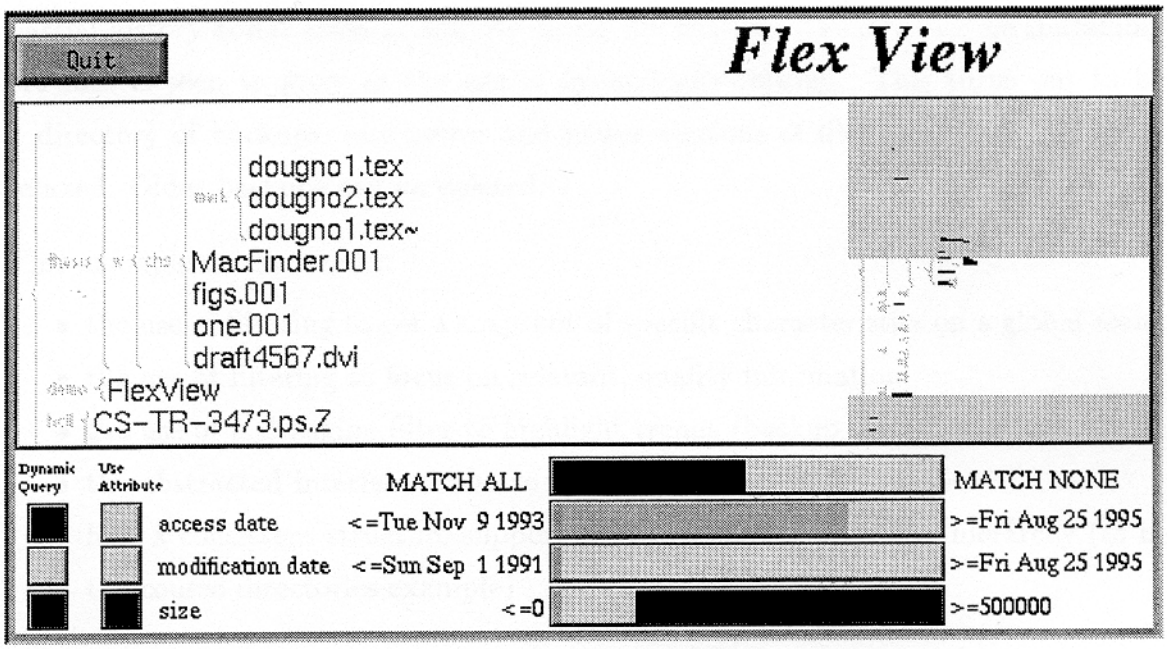


Figure 4.8: Identifying Good Candidates for File Removal.

Let us start with size and age. We can see a filter interface tuned to these attributes (by the node description file) in Figure 4.7, top. The last access date provides us with a good age measure, so modification date is ignored. The top image shows a first stab at some reasonable values for filtering the more than 1500 files in my file system.

We can see by the overview that there are very few large *and* old files, but a fair quantity of old *or* large files exist. Many files passed one of the attribute filters, as shown by the number of files in the overview in Figure 4.7, top. If we toggle off the middle emphasis interval (pass one attribute filter) we suppress all the large *or* old files, showing all and only four files which match this criterion (Figure 4.7, bottom).

We click the date attribute off, creating an exclusion filter on the size attribute. Very few files pass this filter, the results of which are shown in Figure 4.8, telling us that most of the large *or* old files in the top image were old. I remember that the uppermost group of old files (in the top image) are my course directories for the labs I taught. These could be deleted or archived.

Turning the access data attribute back on and slowly relaxing the filter identifies a

partial history about areas of activity on my file storage. Another subtree containing old files is seen to grow as the age is dynamically relaxed. This turns out to be a directory of backups, and newer and newer versions of files appear as the age is relaxed. Older backups can be deleted.

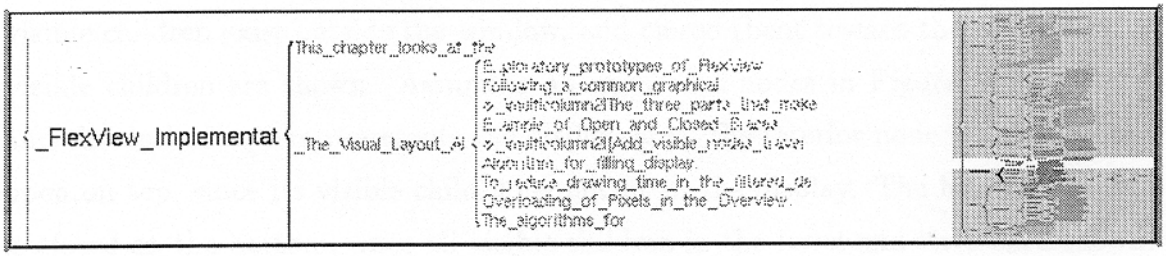
This example demonstrates:

- the use of filtering to get a snapshot of specific characteristics on a global scale,
- the use of filtering to focus on relevant, quality information,
- the use of a changing filter to highlight trends (backup directory),
- the abstracted interface tuned to specific data attributes, and
- how a consistent structure supports a user's knowledge of the hierarchy (as in the course directories example)

4.4 Conclusion/Summary

This chapter has demonstrated some of FLEXVIEW's abilities to enhance exploration of and navigation in large hierarchies. FLEXVIEW's three windows—the filter specification window, and the data and overview windows—have been discussed. This discussion ties the functionality of the interface to many of the hierarchical visualization guidelines generated in Chapter 3 (Figure 3.4). The next chapter continues the examination of FLEXVIEW, switching the focus from the user's view to the implementation.

Chapter 5



FlexView Implementation

This chapter looks at the implementation details of FLEXVIEW: the hierarchy structure layout algorithm; the node data structure within the hierarchy; node emphasis and filtering; and the abstracted hierarchy definition through node description files.

5.1 The Visual Layout Algorithm

Early versions of FLEXVIEW used plain text in a vertically scrollable indented list, where nodes were either present or absent, dependent on filtering. Successive refinements of the system adapted and integrated the visualization guidelines from Chapter 3 as they were being formed. These include showing the structure with a graphical tree, using emphasis through font size and color in the display of node information, and coordinating context and detail into one display.

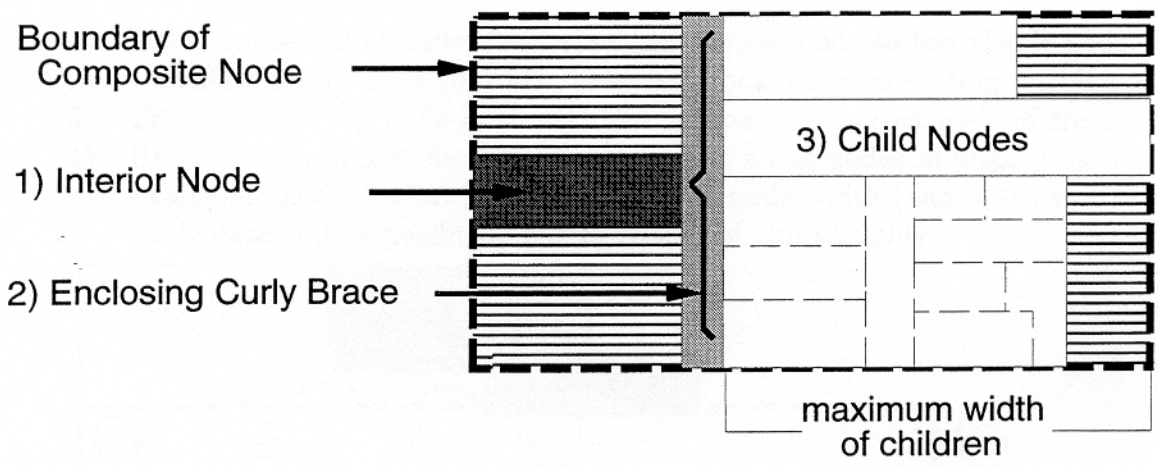
The familiar tree structure, a "left rooted" graphical tree, was chosen for FLEXVIEW's model, as it allows textual names to expand to the right while remaining vertically compact and space efficient. A recursive layout algorithm was used to define three visual components to a node: the interior node, the enclosing curly brace, and the child nodes. Figure 5.1 details how this algorithm allocates screen space for each of the components. To reduce the complexity of display processing, nodes (i.e. the

lines of text) are shown only if the whole node (lines) fits vertically into the display, avoiding clipping of partial nodes.

A brace ({), rather than individual branches, is used to group child nodes. An enhancement of this brace shows the end open (vertically straight) if undisplayed visible children exist outside the window, and closed (bent toward the children) if all visible children are shown. Assuming all numbered nodes in Figure 5.2 are visible and the dashed square represents the display area, the interior node 8 shows a brace open on top, since its visible child 9 cannot fit in the display. The brace for node 8 is closed on the bottom, since all visible children in the lower end have been shown.

If the entire tree were continuously traversed during dynamic queries, performance would suffer with even medium sized hierarchies under the strain of constant updates. To reduce drawing time in the filtered detail view, a focal node (usually located in the vertical middle of the window) is picked for use with an optimization algorithm (Figure 5.3) which traverses only enough nodes to fill the window. This *dramatically* increases performance, since the whole hierarchy need not be examined to view only a small portion. With this improvement, all but the most restrictively filtered displays can be scrolled in real time, with on-the-fly continuous view updates. However, this cannot avoid the performance degradation for severe filtering cases, as the entire hierarchy must be traversed to find enough nodes passing the filter to fill the display.

The algorithms for generating the detail view and the overview are essentially the same, though the latter fits the complete hierarchy vertically into the allotted window and adjusts the relative widths to retain a reasonable aspect ratio (height:width). When the magnitude of data to display is very large, the vertical screen resolution will not be sufficient to accommodate the number of leaf nodes, even if shown only one per pixel line. In the overview window, *overloaded* pixels are used to visually compress the hierarchy by having some vertical pixels represent multiple nodes. While emphasized nodes should be forced to stand out in the overview, the current algorithm does not guarantee this. While it would be algorithmically simple to ensure nodes were layered by their emphasis state when overloading, this produces extra processing to check overloaded pixels, which would slow the system down. Instead, we assume



The three parts that make up a displayed node are:

1) the space to display a node's name,
   vertically centered, if there are visible child nodes,
2) the space to draw a curly brace to "enclose" child nodes, and
3) a left justified vertical list of recursively defined child nodes
   whose width is the maximum width of all child nodes.

[Steps 2 and 3 are skipped if a node is a leaf node or has no children.]

Figure 5.1: Illustration of Hierarchy Visualization.

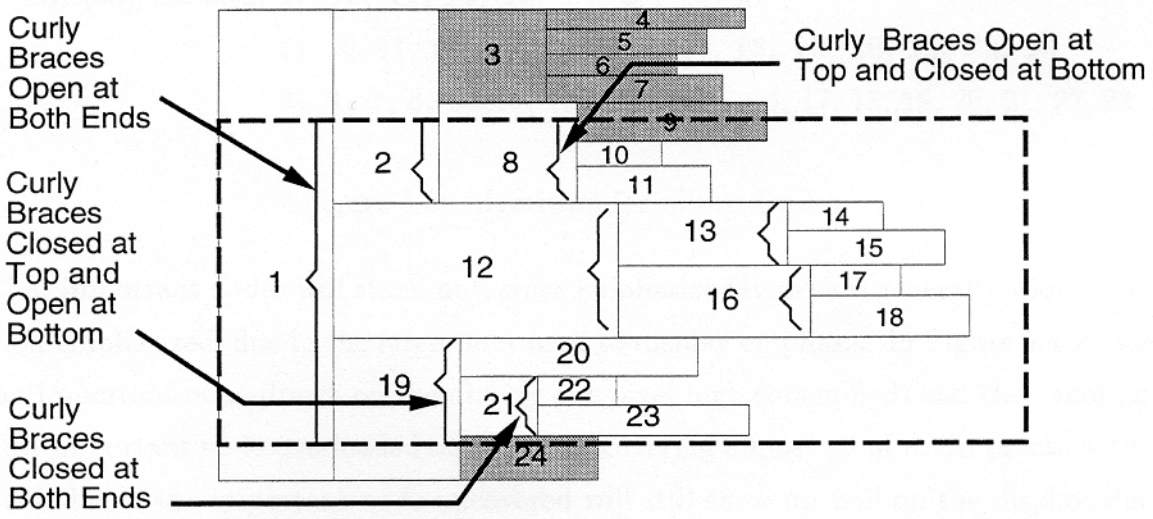
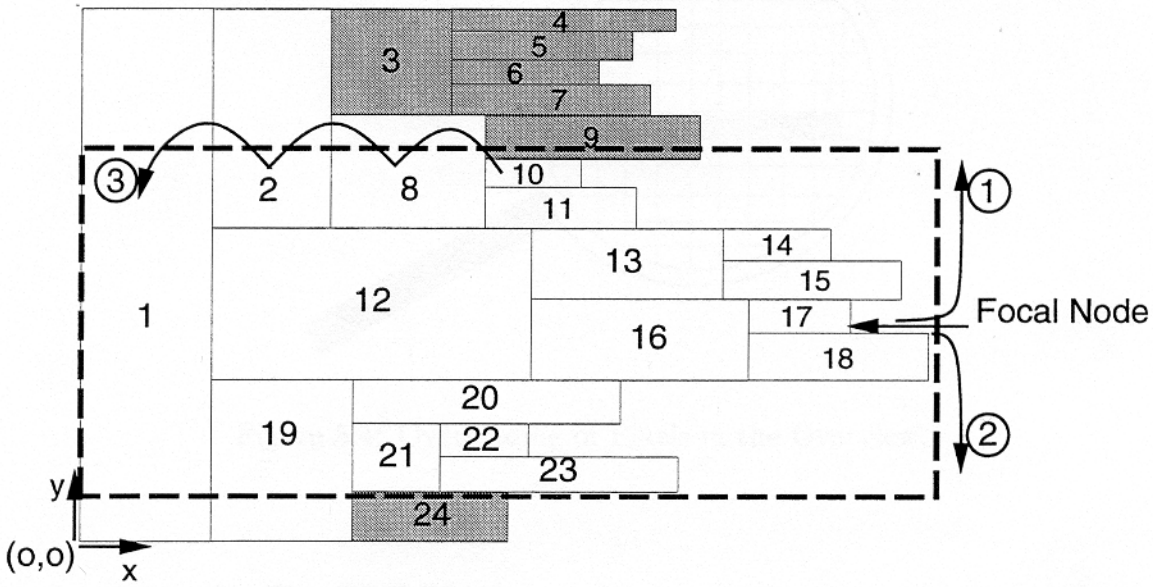


Figure 5.2: Example of Open and Closed Braces.

1) Calculate height of nodes from (including) focal node to top visible node.
2) Calculate height from (but not including) focal node to bottom visible node.
3) Traverse from top node in step 1) through parents to root node of tree.
4) Recursive depth first descent from root *only* along nodes in display list to
   - calculate node x offset from left origin by node width (and brace width),
   - calculate node y height by recursive sum of child heights.



[Add visible nodes traversed in steps 1 to 3 to a pre-order, depth-first Display list]

Display list after 1) **10, 11, 12, 13, 14, 15, 16, 17**
2) 10, 11, 12, 13, 14, 15, 16, 17, **18, 19, 20, 21, 22, 23**
3) **1, 2, 8,** 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23

Figure 5.3: Algorithm for filling display.

that important nodes will stand out, since emphasized items are generally wider than non-emphasized, due to the larger font used to display emphasis. In Figure 5.4 we see an important node drawn on the display one pixel high (magnified) and then another less important node overloaded on top of it, covering almost all of it. In practice, the little bit of the important node uncovered will still show up well on the display, due to its contrast with the surrounding area.

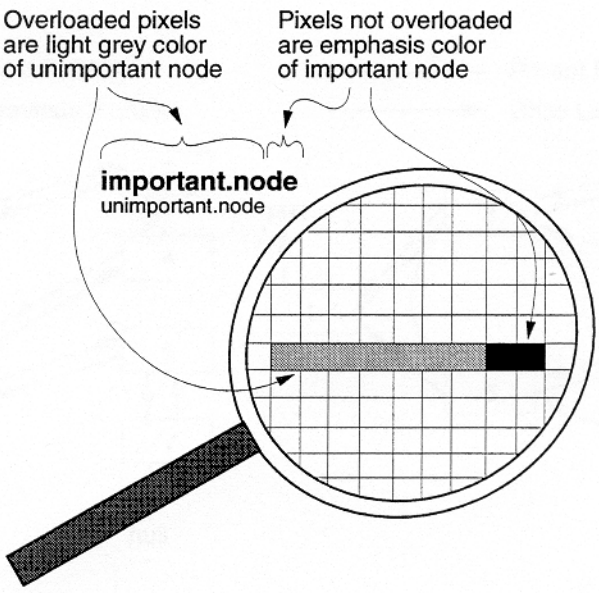


Figure 5.4: Overloading of Pixels in the Overview.

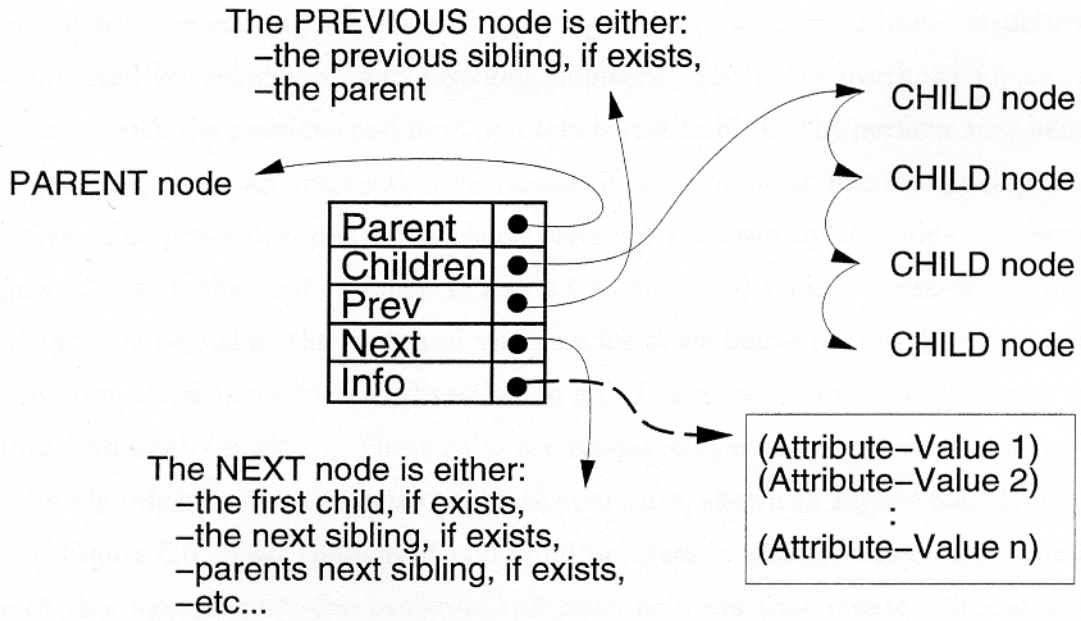


Figure 5.5: FLEXVIEW Data Structure

5.2 The Data Structure

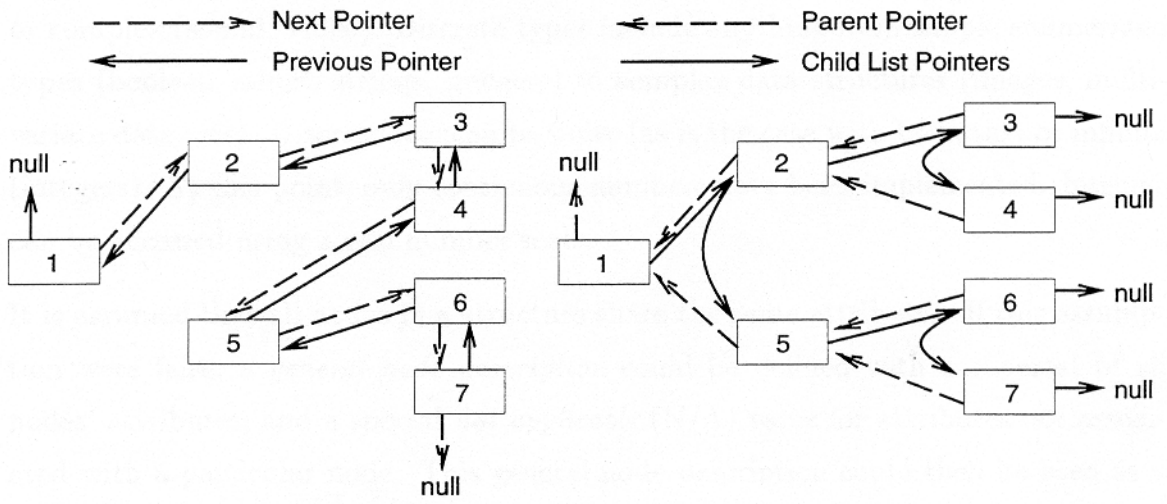


Figure 5.6: Previous, next, parent and child pointers in a simple hierarchy.

The basic hierarchical tree structure is kept by having each node contain a parent pointer, and a pointer list to its children. To optimize the depth-first, pre-order tree traversal when executing the visual layout algorithm, an internal linear structure is also imposed by using next and previous pointers. The extra overhead for storage associated with the previous and next pointers is justifiable by the performance benefit they provide, since an interactive visualization system *must* redraw quickly to be effective. The pre-order, depth-first node ordering is shown by the node numbers in Figure 5.3, with the root number 1, and 17 as the focal node. A pre-order, depth first traversal begins at the root, and visits nodes *down* below it first. If the node has no children, there is no "down" direction to go. The next option is to go *across* to a sibling, and finally back up. These rules are repeated at every node. A pointer to the set of node information completes the data structure, shown in Figure 5.5. The right side of Figure 5.6 shows the parent and child pointers in a simple hierarchy. The left side of this figure shows the previous and next pointers that create a linear list of nodes.

Node data is stored within this organization structure as a list of attribute values

matching the node descriptions attribute list. This thesis distinguishes between continuous and discrete value types. Continuous types can be simple (real numbers) or complex (sound, video). Discrete types include anything from simple enumerated types (boolean values, strings, integers) to complex data structures (images, multivariate data sets). Discrete sets can be finite (as is the case with booleans) or infinite (integers). At this point, only continuous numbers have been implemented. Integers can be accessed using a real number scale.

It is assumed that all nodes in a structure share the same attributes. If this assumption were false, a *general node description* could be defined with a superset of all nodes' attributes, and a special *not applicable* (N/A) value for attributes not associated with a particular node. This general node description could then be used as a template to examine any node in a structure. FLEXVIEW also assumes a single root node. Multiple trees sharing the same node information structure can combined one level down into a single tree with a "super root".

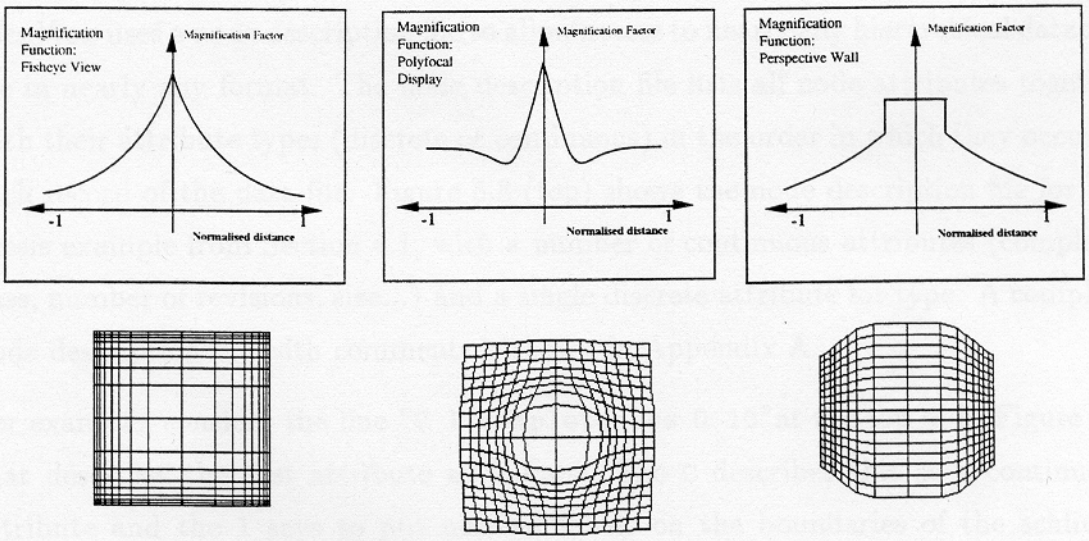


Figure 5.7: Examples of Magnification Functions (from [LA94]).

5.3 Emphasis and Filtering in FlexView

Magnification functions can describe how distortion occurs in a view. Many magnification function variations have been put forth [LA94]. Some of these are illustrated in 5.7. These all involve mapping points in an undistorted display to their distorted counterparts, typically by enlarging an area and reducing its context using a clean and simple mathematical transformation. There is an important distinction to make between FLEXVIEW and other distortion-oriented systems. While other systems distort their views on a global scale based on one or more focal points, FLEXVIEW distorts its display only on the individual node level, based on node importance values. The magnification function to describe the view is generated by, not imposed on, the view. The resulting emphasis and exclusion filtering in the detail view focuses on important information and preserves relevant context.

5.4 Abstracted Hierarchies via a Node Description File

FlexView uses a node description file to allow access to nearly any hierarchical database file in nearly any format. The node description file lists all node attributes together with their attribute types (discrete or continuous) in the order in which they occur in each record of the data file. Figure 5.8 (top) shows the node description file for the thesis example from Section 4.1, with a number of continuous attributes (completeness, number of revisions, size...) and a single discrete attribute for type. A complete node description file with comments is shown in Appendix A.

For example, consider the line "`C 1 completeness 0 10`"at the top of in Figure 5.8 that describes the first attribute as follows. The `C` describes this as a continuous attribute and the `1` says to put number labels on the boundaries of the schlider. `Completeness` is used as the label for the query selection mechanism, and the schlider can use the minimum and maximum boundaries of `0` and `10`, respectively.

The last line in the figure gives a special root node description. Here, the `R` specifies that this is the root definition. This and other node records have two structure values

that precede the node attribute-values. The `U` says this is an interior (designated `U` for an unleaf node, instead of `L` for a leaf node) node and `thesis` is the node's display label. The rest of the values correspond, in order, to the attribute description above it. Thus its first page is 1, its last page is 103, it is of mixed type, and all other attributes are 0.

Figure 5.8 (bottom) shows the description for interpretation the hierarchical data, in this case the file system seen in Figure 5.9. This data file shows a few (`U`) directories and many (`L`) files contained within. Each record has the same fields; type, location in hierarchy, parent, name, owner, last access date, last modification date, and size in bytes.

The node description file is also used to generate the custom filter interface. Although not yet implemented, the attribute type in the first column could be used to pick the type-appropriate selection interface (such as a schlider for continuous numbers or an alphaslider or drop-down listbox for strings) for the query mechanism. A display/don't display toggle (in the second column, where x means "don't display") allows (ideally dynamic) suppression of unimportant attributes query mechanisms from the filter selection window. Ranges or sets of choices can be calculated (by command line options) based on data file values, or specified in the description file, to override the values that would be calculated.

In addition to a node's attribute values, the data file contains a leaf or unleaf toggle, a node's label (name), and its position in the hierarchy. Rather than assume a leaf is a node with no children a special attribute differentiate exterior leafs (such as files) from interior nodes (such as directories, which may be empty).

Customizing the Interface

Command line options are used to specify the hierarchy data file name and its node description. Additional options can be used to modify the interface by calculating minimum and maximum bounds (rather than using the bounds provided), by bounding the end points (toggle whether a selected end point means only values up to or

```
# File Attribute information
#
C l completeness                    0 10        10 is most complete
C l number_of_revisions             0 1000
C l recency_of_revisions_(days)     0 365
C l section_size_(pages)            0 40
C l section_first_page              1 200
C l section_last_page               1 200
D x section_type                    5 text figure table chart mixed
#
R U thesis 0 0 0 0 1 103 mixed
```

```
# File Attribute information
#
D x type                    7 d r s l p c b
S x path
S x name
S x owner
C l access_date             7500 8500           in days
C l modification_date       7500 8500           in days
C o size                    0 50000             in bytes
#
R U . d . . root 8500 8500 512
```

Figure 5.8: Node Description File examples for (top) my thesis, and a file system.

```
U ./thesis/w/chs d ./thesis/w chs schaffer 9357 9357 2048
L ./thesis/w/chs/all.tex r ./thesis/w/chs all.tex schaffer 9357 9357 2120
L ./thesis/w/chs/appendix.tex r ./thesis/w/chs appendix.tex schaffer 9339 9339 9744
L ./thesis/w/chs/ch2.tex.bak r ./thesis/w/chs ch2.tex.bak schaffer 9345 9350 41581
L ./thesis/w/chs/ch3.tex.bak r ./thesis/w/chs ch3.tex.bak schaffer 9350 9356 88119
U ./thesis/w/chs/docs d ./thesis/w/chs docs schaffer 9042 9042 512
L ./thesis/w/chs/mythesis.sty r ./thesis/w/chs mythesis.sty schaffer 9034 9034 9463
L ./thesis/w/chs/fig.tex r ./thesis/w/chs fig.tex schaffer 9034 9034 824
L ./thesis/w/chs/ref.bib r ./thesis/w/chs ref.bib schaffer 9345 9351 55691
L ./thesis/w/chs/thesis.tex r ./thesis/w/chs thesis.tex schaffer 9043 9043 1936
L ./thesis/w/chs/title.tex r ./thesis/w/chs title.tex schaffer 9060 9060 6011
U ./thesis/w/chs/tmp d ./thesis/w/chs tmp schaffer 9331 9331 1024
L ./thesis/w/chs/uct12.sty r ./thesis/w/chs uct12.sty root 9034 9034 18003
L ./thesis/w/chs/ucthesis.sty r ./thesis/w/chs ucthesis.sty root 9034 9034 31053
```

Figure 5.9: Portion of file system data file for FLEXVIEW.

```
#######################################
#                                     #
#        Schlider  Flex  View         #
#                                     #
#######################################

Usage: FlexView {-d <datafile> | data=<datafile>}
                {-n <nodedescriptionfile> | nodedesc=<nodedescriptionfile>}
                [[-f <filterfile> | filter=<filterfile>]
                 [-c | calc]
                 [-b | bound]
                 [-h | help | -m | man]
                 [-o | noover]
                 [-e | fish]]
```

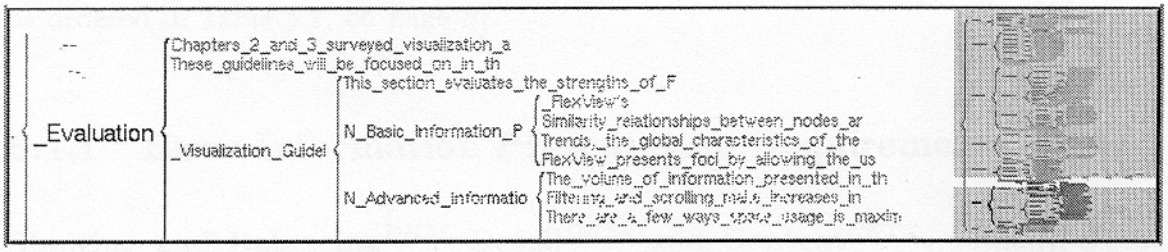
Figure 5.10: Command Line Options: The result of typing `flexview -h`.

all past that point), by removing the overview from the interface (replacing it with a scrollbar), and by imposing a proximity based fisheye view on the emphasised nodes in the detail display. The filter file name will be used for saving filter selections in between sessions, but has not been implemented yet.

5.5 Summary

The previous chapter examined FLEXVIEW from a user's perspective. This chapter went over many of FLEXVIEW's implementation details. Most of the details discussed have been implemented. The few unimplemented details, such as ensuring emphasized nodes stand out in the display and allowing multiple filter selections types, will integrate easily into the current design. The next chapter evaluates FLEXVIEW based on the visualization guidelines of Chapter 3, to determine its success at meeting its design goals, and its strengths and weaknesses. Chapter 7 looks at required improvements and directions for future research.

Chapter 6



Evaluation

Chapters 2 and 3 surveyed visualization approaches accompanied by example systems, examined the essential characteristics of hierarchical visualization, and generated a set of visualization guidelines, summarized in Table 3.4. Chapters 4 and 5 then presented the user and implementation view of FLEXVIEW around Table 3.4's visualization guidelines.

This chapter uses the guidelines to evaluate FLEXVIEW. I will show that almost all guidelines have been addressed to some degree by FLEXVIEW. Sections 1 and 2 evaluate FLEXVIEW's strengths and weaknesses, respectively, by seeing which guidelines are met well and which have room for improvement. As will be seen, FLEXVIEW's main strengths are its presentation of local detail and global context, its dynamic query filtering to encourage exploration and show trends in the data, its emphasis and exclusion filter interface to reduce and improve the data display, and its use of abstracted hierarchies. Its predominant weaknesses are its need for optimizations to improve update speed, the complex nature of the flexible filter interface, and its incomplete attribute type set.

6.1 Visualization Guidelines Met Effectively by FlexView

This section evaluates the strengths of FLEXVIEW by seeing how it follows the visualization guidelines. The discussion will be organized by walking through the categories as ordered in Table 3.4, on page 57.

6.1.1 Basic Information Presentation Requirements

Showing and Balancing^{DBAL} Detail^{DETL} and Context^{CNTX}. Initially, a pure fisheye view was planned, instead of the separated detail view and overview. This single view would have taken the node interval emphasis state and imposed a fisheye emphasis (by proximity to focal node centered in the screen) on it, preserving the complete contextual, hierarchical structure around it. This was abandoned as its view was unable to manage large hierarchies, due to the intense crowding around at the top and bottom ends of the display shown by a prototype.

FLEXVIEW's detail view and overview were designed instead. Each presents a different focus on local detail. The overview, because of its structural consistency, conveys strictly proximity-based local detail. The detail view, by its filtered nature, conveys nodes closely related by the current emphasis filter as the local detail. The detail view and overview work together to improve the quality and increase the useful information content in the presentation of the entire hierarchy. Because of these features, I believe that FLEXVIEW strongly presents a flexible and clear visualization of the many facets of hierarchical information.

Showing Data Relationships^{RELN}. Similarity relationships between nodes are presented in the detail window, where nodes in the "Matching All" emphasis interval share values within the sub-ranges or each attribute filter. The more restrictive the filtering, the more similar the nodes in this interval become. Alternatively, as fewer attribute filters are passed, less can be inferred from the node's appearance in the associated interval emphasis state, since nodes within the same emphasis interval may pass different filters. Relationships based on contrasts are conveyed by the nodes'

existence in different intervals (such as "Match All" and "Match None"). Structural relationships between non-excluded nodes are shown in the detail view.

Contrast comparisons can also be made over time between nodes as changes in ranges, made through dynamic query filters, causes the nodes' importance values to change as well, leading to a change in emphasis state. Seeing a node change its emphasis indicates its value for that attribute. These are powerful features, and the effect is particularly striking when one is viewing their own information stores, for they almost invariably lead to new insights into the data's relationships.

Showing Data Trends^{TRND}. Trends, the global characteristics of the data space, can be presented well by looking at the overview. The overview quickly conveys whether similar emphasized nodes are concentrated in pockets or are evenly distributed across the hierarchy. Changing the dynamic query filters (assuming the overview is instantly updated) provides the exact attribute values for nodes as they change emphasis intervals, now shown on a global scale in the overview. Thus, global characteristics are highly visible in FLEXVIEW.

Showing Independent Foci^{FOCI}. Unlike most fisheye view systems, where focal points are selected by geographic location, FLEXVIEW presents foci by allowing the user to exclusion filter uninteresting nodes, leaving only interesting nodes. Of these, the most emphasized become, by definition, focal points in the hierarchy, and they may be numerous. The problem is that if many nodes are displayed in the detail view, two focal nodes might be more than one screen apart, stopping them from being viewed concurrently. This weakens a person's ability to make comparisons between distant nodes. Ideally the option to deal with independent foci should exist.

6.1.2 Advanced Information Presentation Features

Reducing the Volume of Information Shown$^{<VOL}$. The volume of information presented in the detail view is greatly reduced by effective filter selection. This results in a view containing mostly important nodes. This is a key means by which future

information systems will deal with large quantities of inforamion. One only has to look as far as the Internet news groups to see how useful filtering will be to highlight relevant information and remove unimportant news. There is so much information available that, without adequate filtering tools, one could almost spend a lifetime browsing for useful and specific information.

Scalability^{SCAL}. Scalability is the degree to which a visualization can handle increases in data magnitude. Filtering makes even large amounts of information manageable in the detail view. The overview is where scalability problems would arise, since this is where the entire hierarchy is shown. Overloaded pixels increase FLEXVIEW's resistance to scalability problems in the overview by mitigating screen size resolution limitations. I believe that pixels could be overloaded by a factor of 5 to 10 times the pixels lines and and the display would remain usable. If overloading processing guaranteed precedence to higher emphasis nodes, the overloading factor could be increase even more. Figure 6.1 shows a structure with mode than thirty thousand nodes in seven hundred vertical pixels of screen space, overloading by more than 40 to 1.

Using Space Efficiently^{SPAC}. There are a few ways space usage is maximized by FLEXVIEW. The overview's compressed nature makes heavy use of a small amount of space to convey structural information. The *schlider* control provides a compact and simple interface for selecting many distinct sub-ranges.

The layout algorithm, when used in the detail view, generates a conventional horizontal tree structure, with a characteristic triangle of data (a base of leaves and root at the opposite point), designed to make use of vertical space. Unfortunately, this layout results in the two corners closest to the root likely conveying little information, as shown in Figure 6.2. This is a reasonable tradeoff, as it preserves a person's conventional view of hierarchies and has associated benefits, such as reducing user cognitive load.

6.1.3 Facilitate Tasks

Encouraging Data Exploration^{EXPL}. The dynamic query filters provide an ideal

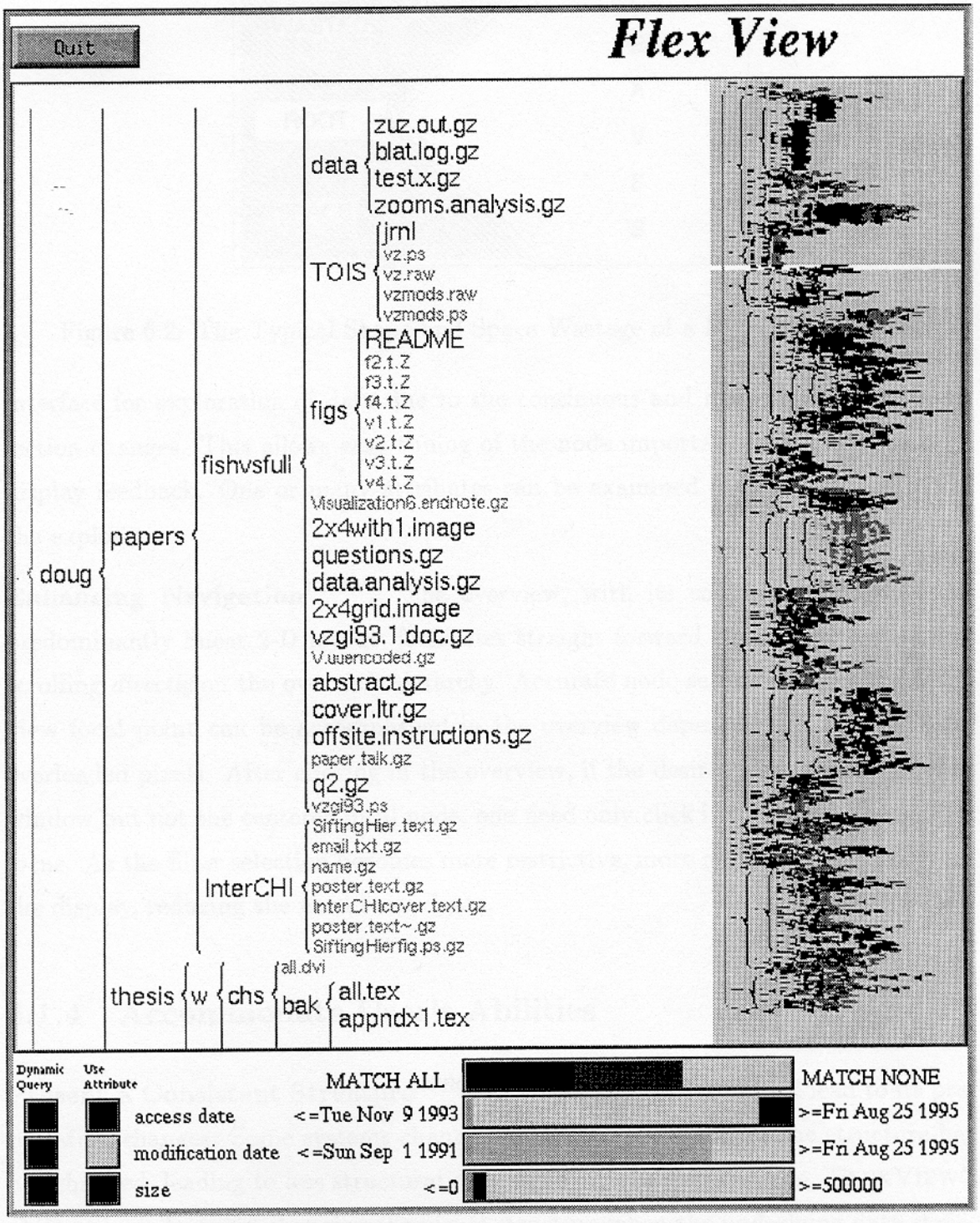


Figure 6.1: FLEXVIEW: 30,140 nodes on a 1 gigabyte disk in 700 vertical pixels.

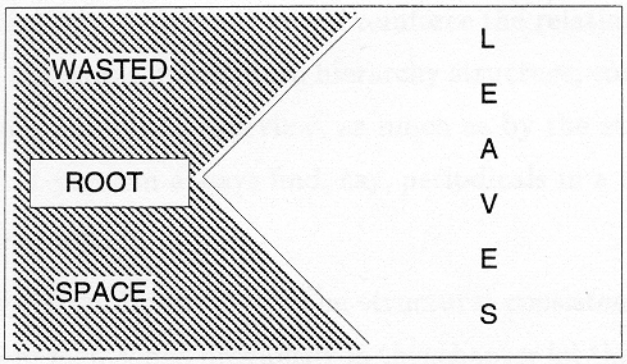


Figure 6.2: The Typical Shape and Space Wastage of a Left-Rooted Tree.

interface for exploration of data due to the continuous and reversible display of selection changes. This allows easy tuning of the node importance criterion, based on display feedback. One or many attributes can be examined concurrently to control the exploration.

Enhancing Navigation^{NAVN}. The overview, with its consistent structure and predominantly linear 2-D layout, facilitates straight forward navigation, assisted by scrolling *directly* on the overview hierarchy. Accurate node selection to set the detail view focal point can be accomplished in the overview depending on the degree of overloaded pixels. After clicking in the overview, if the desired node is in the detail window but not the centered focal node, one need only click it to make it the central focus. As the filter selection becomes more restrictive, more nodes are removed from the display, reducing the need to navigate.

6.1.4 Accommodate User's Abilities

Present a Consistent Structure^{STRC}. Changes to a structure will lead to its presentation changes. Some systems changes the presentation even if the structure has not changed, leading to less structural consistency in the visualization. FLEXVIEW's philosophy is to avoid changing the visual structure when the underlying data structure has not changed. This is accomplished by allocating space for every node in the hierarchy and applying emphasis without deallocating the excluded node's space.

Over time, this consistent representation will reinforce the relationship between *areas* in the overview and the user's internalized hierarchy structure, conveying information based on the location within the overview, as much as by the structure. Analogous to this is the idea that you can always find, say, periodicals in a library once you are familiar with their location.

Reduce User Cognitive Load^{COGN}. The structural consistency and the familiar presentation paradigm mentioned previously in this chapter let the user become familiar with the representation, reducing the cognitive load of dealing with the hierarchy.

Exploit Human Perception^{PRCP}. The human ability to perceive change, even distant from the focal point, is exploited in FLEXVIEW's interface as dynamic queries continuously animate and update the display based on the immediate change made to the filter selection or interval emphasis. The eye can easily pick out contrasting anomalies, so a few emphasized nodes are easily spotted when displayed in an overview filled with less interesting data. Filtering ensures important nodes will be emphasized. Fine emphasis steps are more difficult to see, particularly between distant nodes, which is why a striking color such as hot pink was chosen for the most emphasized level. After trying many colors, hot pink was selected because it stands out well against both the light detail background and the darker portions of the overview. Most other colors worked well only in one or the other.

6.1.5 Related Issues

Specialize^{SPEC}. Specialization is the degree to which a visualization is tuned to the features of a unique data set. FLEXVIEW automatically generates a filter specification interface based on information in the node description file. The user has control over calculating bounds (or not) and suppressing a filter for an uninteresting attribute. Thus, the attribute filters presented are customized to the users' preference for the data set being examined.

Abstract^{ABST}. Abstraction is the degree to which a visualization is able to deal with any hierarchically arranged data. While the interface is specialized based on

the node description, that description can be defined to access many different data sets. FLEXVIEW's current implementation allows filtering only on continuous numeric attributes, but the ground-work has been laid to incorporate other types of data, such as discrete text-based attributes like strings.

Be Flexible^{FLEX}. FLEXVIEW provides a flexible way of dealing with hierarchical data. The two most important characteristics of FLEXVIEW from this standpoint are the ability to handle abstracted hierarchies, and the exclusion and emphasis filtering possibilities that the filter specification interface provides.

Optimize^{OPTM}. A few optimizations have been integrated into FLEXVIEW. The optimization to calculate only enough to fill the detail view has a large positive effect on redraw speed. Performance was boosted further by avoiding extra processing associated with ensuring that emphasized nodes get preference when overloading pixels in the overview. However, these optimizations reduce the display accuracy, as will be discussed in a later section.

Summary. FLEXVIEW's four biggest strengths are as follows.

1. Its ability to emphasize nodes and use exclusion filter to reduce the volume and improve the quality of information presented.
2. Its dynamic query filter specification interface that encourages data exploration and helps to highlight trends and relationships.
3. Its integration of local detail with global context to present important relationship and characteristics of the data.
4. Its abstraction away from a specific data set, based on the node description file.

6.2 Weaknesses and Suggested Improvements

This section covers guidelines that FLEXVIEW meets, but not as well as it could. The order of presentation will again follow the guidelines of Table 3.4, with some omissions where improvements are not a concern. Integrated into this discussion is a thorough generation of ideas to improve FLEXVIEW.

6.2.1 Basic Information Presentation Requirements

Showing Data Trends^{TRND}. In the current partially-optimized version of FlexView, the ability to present trends based on the change in emphasis in the overview window is hampered because results of filter changes are only updated in the portion of the hierarchy overview corresponding to the detail view. This is because the optimization algorithm only traverses enough of the hierarchy to fill the detail window. While there is a pronounced improvement in the time to redraw in the detail window after filter changes, it comes at the cost of a large reduction in the effectiveness of the overview, since information in the areas outlying the detail portion of the hierarchy are not updated. This problem can be overcome by improving the optimizations to allow real-time traversal of the entire tree. These optimizations will be discussed later in the **Optimize**^{OPTM} paragraphs.

Showing Data Relationships^{RELN} **and Independent Foci**^{FOCI}. FLEXVIEW is limited to showing foci dependent on their location within the hierarchy structure. These must be made independent to allow node comparison and exploration of relationships. This could be done with horizontal splits in the detail window, where each portion is linked to the overview, creating a "detail highlight" in the overview

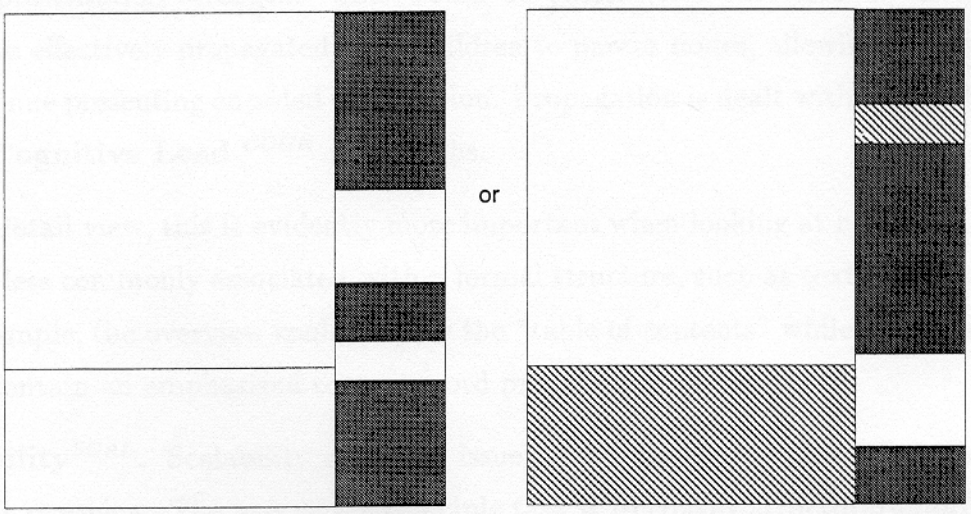


Figure 6.3: Independent Foci in Detail Views.

for each detail section. Multiple detail windows raise problems coordinating the links from the overview to multiple detail views. Figure 6.3 shows the proposed detail window horizontally divided into a large and a small independent views, each linked to a portion of the overview context.

6.2.2 Advanced Information Presentation Features

Reducing the Volume of Information Shown$^{<VOL}$. Currently, the detail view and the overview share a nearly identical layout algorithm, primarily to ease users' cognitive load when integrating the two. This results in much structural information presented in the detail view and miniaturized node information shown in the overview. While this can be useful, in some cases the effectiveness may be lost due to excessive clutter. The reduction in cognitive load by keeping the two views so similar, and the usefulness of duplicating information in the two views may be outweighed by the extra cognitive load this clutter may cause. The reason for raising this point in the **Reduce the Volume of Information Shown**$^{<VOL}$ paragraph is that clutter is reduced by de-emphasizing or suppressing node information in the overview and structure information in the detail view, leaving each view with more space to devote to its presentation strength. This would be particularly successful if information could be effectively propagated from children to parent nodes, allowing the overview to continue presenting encoded information. Propagation is dealt with in the **Reduce User Cognitive Load** ^{COGN} paragraphs.

In the detail view, this is evidently more important when looking at hierarchical data that is less commonly associated with a formal structure, such as textual documents. For example, the overview could display the "table of contents" while the detail view could contain an emphasized content word processor or editor.

Scalability^{SCAL}. Scalability is not an issue in the detail view, though it certainly is in the overview. The overview's principle task is to convey structural information. Reducing the information presented by focusing on structure information would ease the overview's burden. Another option that would extend the threshold would be

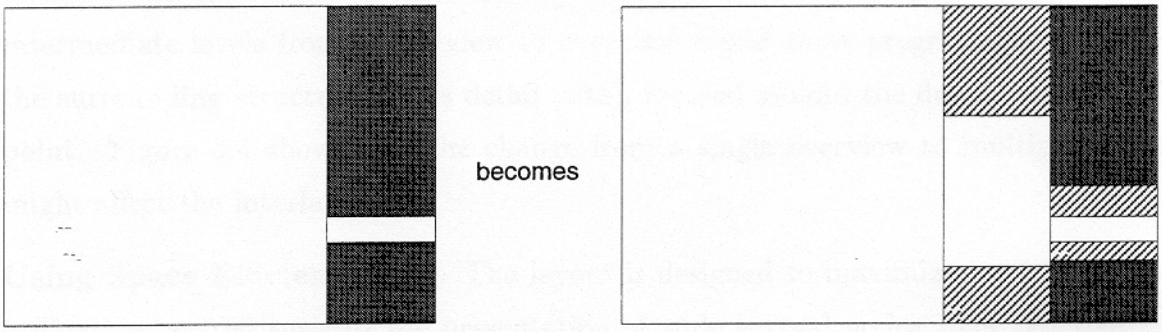


Figure 6.4: Adding Intermediate Overview Levels.

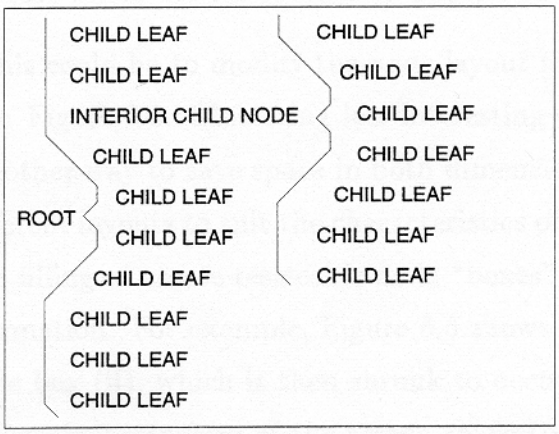


Figure 6.5: Using Screen Space More Efficiently in a Left Rooted Tree.

i) This is the original really wide node

ii) This is the original really wide node

iii) This is the or

iv) This is the ori ginal r eally w ide no de

Figure 6.6: Encapsulating Node Information in Resizable Node Boxes.

to add levels of overviews, when the degree of pixel overloading gets too high. The intermediate levels from detail view to overview would show progressively more of the surrounding structure in less detail (size), focused around the detail view's focal point. Figure 6.4 shows how the change from a single overview to multiple levels might affect the interface.

Using Space Efficiently^{SPAC}. The layout is designed to maximize vertical screen utilization, geared towards the presentation of wide textual nodes. The left rooted tree results in little information shown in the top left and bottom left corners of the detail window. Deep hierarchies with wide interior nodes and high child counts are the worst case structures. There are many ways to address this problem.

One way to address this could be to modify the node layout to use more of this corner space, as shown in Figure 6.5. Clustering less interesting subbranches (as many systems do now) is another way to save space in both dimensions. We can also allow the user to choose different layouts to suit the characteristics of the hierarchy, such as indented lists or space filling or create resizeable node "boxes" that could be used to contain the node information. For example, Figure 6.6 shows a large node name (i) encapsulated in a node box (ii), which is then shrunk to occupy less width (iii) and stretched vertically to make better use of the screen space (iv). This could greatly reduce the amount of unutilized space in the detail window.

6.2.3 Facilitate Tasks

Encouraging Data Exploration^{EXPL}. There are a number of emphasis issues that need to be addressed to better facilitate data exploration. One big drawback of the current filtering system is its inability to vary emphasis based on results of a single attribute. In the file system example, exclusion attribute filtering allows one to differentiate between files above and files below a certain threshold. However, the only way to determine a ranking on size is to see files change emphasis state as the size filter range is extended across their values. Changing the exclusion *schlider* to



Figure 6.7: Controls for Setting the Current Emphasis Mode.

an emphasis *schlider* would solve this problem, and it would be simple to replace the binary exclusion attribute filter result by a weighted value instead. This could be implemented with an *emphasis mode* chosen from a palette (Figure 6.7, i) or a continuous spectrum tied to emphasis states (ii). *Schlider* sub-range selections would be based on the current emphasis mode. Emphasis need not be a solid level, as the set of emphasis choices shown in (i) is extended to include varying emphasis mode options in (iii).

This leads to another possible enhancement to address another weakness. All active attribute filters are now weighted equally. Allowing continuous weighting instead of the binary "Use Attribute" toggle would allow some attributes to be stressed over others.. For example, when cleaning up a hard drive, I would like to weight age more than size to bias the visualization toward old files.

The reason for the explicit intervals in the Interval Emphasis control is the binary results of the attribute filters. Any method of emphasis or weighting at the attribute level would render this control inadequate, and it would need to be replaced by an emphasis *schlider* for use on the node importance value.

Though much more complex, these three layers of emphasis (attribute ranges, attribute weights and emphasized node importance value) would give the user great control over attribute and node emphasis, albeit at the expense of a more complex interface.

A minor weakness of FLEXVIEW is that current filtering is now done only on node attributes. It would be useful to filter on standard hierarchical structure information as well, such as the length of a path, depth of a node, or number of children.

Enhancing Navigation^{NAVN}. FLEXVIEW was designed for shallow and large breadth hierarchies. Deep hierarchies are dealt with less well, creating a need for horizontal scrolling in the detail view.

6.2.4 Accommodate User's Abilities

Reduce User Cognitive Load^{COGN}. There are a number of issues that need to be resolved, each which could reduce users' cognitive load. A major question is how to deal with the propagation of values from children to parents. Currently each node is treated independently. It could be useful to have other options to calculate a node's attribute value. For example, the size of a directory (interior node) in a Unix file system is a small independent value, but one might want a directory's size to include that of its children. Cases can be made for using the maximum, minimum or average of the children's values in addition to having independent node values. It is unclear how this could be specified, either on an individual node basis or a global scale.

The node importance value itself raises questions. The abstract notion of a node importance based on the selected filter criterion is difficult to translate into a more detailed meaning. Dealing with attributes individually is simple enough. However, the result of the interval-emaphsis layer of filtering on the combined attribute filter results is less clear, especially as the number of attribute filters rises. One way to alleviate this may be coupling of the detail view and filter window controls. Clicking on a node in the detail view would cause its exact values to be highlighted in the *schlider* attribute filters, and the node's importance value to be shown in the interval empahsis control. This would remove the abstraction of the filter controls values.

Another practical difficulty that cannot be overlooked is the creation of the node description and data files. Creating the node description file is currently a manual process and is tedious and error prone. It is easy to imagine automatic scanning of the database to determine type of fields and choices or ranges, which would go a long way to easing initial start up difficulties. Next, the formatted data file is currently created with a translation script that must be written by a programmer. FLEXVIEW would benefit from a more direct data input scheme, avoiding separate data files, if possible.

There are a few of criticisms of lesser importance. First, the link between the detail view and overview needs to be strengthened. There is an implied link based on

background color, but this needs to be made explicit. Selecting a node in either view should highlight that node's presentation in the corresponding view (if it is visible). This link would reinforce a tighter coupling of the two views.

The second minor concern overlaps the cognitive and the perception^{PRCP} guidelines. Due to the font-size based presentation, (rather than using scaled fonts), instant screen updates make the display look jumpy when fonts change between the 6 pt (which is about the smallest screen readable size) and 0 pt (removed) emphasis states. This increases the difficulty of tracking changes. These transitions, indeed any emphasis changes, should be smoothly animated to allow the user to see nodes that are disappearing and what and where new nodes are appearing.

6.2.5 Related Issues

Be Flexible^{FLEX}. A number of suggestions have been made to increase the flexibility and power of the filter interface. FLEXVIEW's emphasis capabilities (**Encouraging Data Exploration**^{EXPL} paragraphs) can be extended, as can the options for dealing with the node importance values. A node description file configuration interface (**Reduce User Cognitive Load**^{COGN} paragraphs) could be created. However, increased flexibility is accompanied by increased cognitive load due to growing interface complexity.

Integrate in User's Envoronment^{ENVR}. Much of the FLEXVIEW interface is innovative, within a GUI frame work. Where ever possible, conventions on interface interactions have been adhered to, but this is only half of the integration issue. The other part deals with the integration of the system into the other applications in the user's environment, and the seamlessness with which data can be treated. Software development technology, like Microsoft's OLE (Object Linking and Embedding) may hold the key to tying applications together, but this is not adequately addressed by current technology.

Allow Seamless Data Access^{DATA}. More exotic data access mechanisms are required to satisfy the desire for seamless and dynamic data. This might include a

standard data access programming interface to allow new data types and proprietary data to be accessed directly, from within FLEXVIEW.

Optimize^{OPTM}. FLEXVIEW's major problem is that it takes too long to traverse very large hierarchies in real time, a requisite during dynamic query filtering. Since storage space usage is not yet an issue and speed most certainly is, processing optimizations could include imposing internal structures and caching data to streamline tree traversal.

The first idea involves changing the data structure. Previous and next pointers are already used to increase the tree traversal efficiency. The same basic idea could be used to streamline the filtering process. For example, filter selection is only done on a single attribute at a time. On that attribute, the selection is either being relaxed or restricted, resulting in either more nodes or less nodes passing the filter. With this in mind, next and previous pointers could be added to the data structure for each attribute-value as shown in Figure 6.8, creating ordered lists for each attribute. *Schliders* could keep pointers to the values in the appropriate attribute value list corresponding to the sub-range boundary values. Changing a filter selection sub-range boundary would result in a linear traversal down the attribute value list from the initial boundary value to the new boundary value. This small list of newly selected (or de-selected) nodes could then be added (or removed) from the display list.

Early prototypes of FLEXVIEW did not have an overview, but were still very slow to redraw because of traversal of the entire tree. This optimization would allow traversal of the minimum number of nodes for any selection change, reducing this bottleneck and making it reasonable to update the entire hierarchy in real time. Node importance values could be cached and recalculated only when affected by an attribute filter selection change.

Since the whole hierarchy must be drawn in the overview, it must be entirely traversed to determine the layout, an expensive process. Caching layout information, including visual node characteristics, would allow changed portions of the display (known by the affected node list) to be updated for the new filter specification.

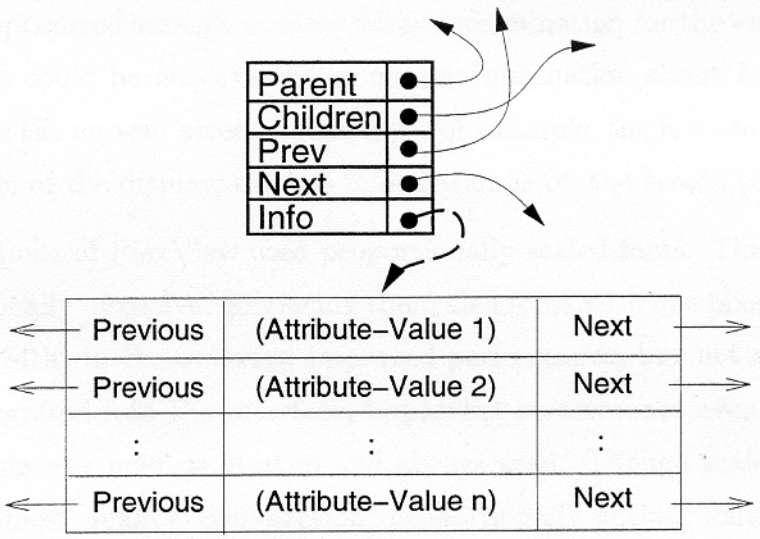
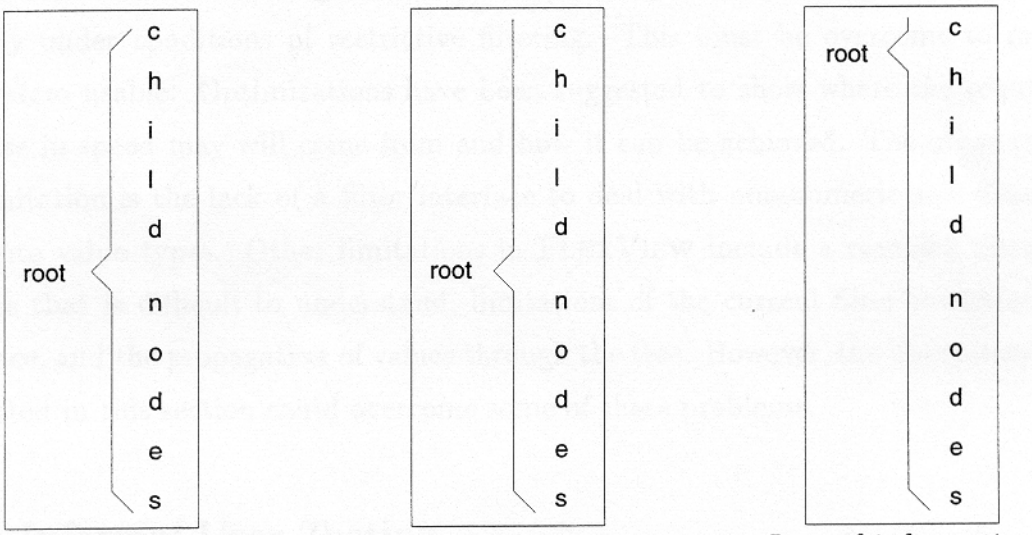


Figure 6.8: An optimization: previous and next pointers for each attribute.



No information off of the screen

A bit of information off of the screen

Lots of information off of the screen

Figure 6.9: Open Curly Braces Showing Proportion of Visible Children

Visual cues have been included in other areas of the display to subtly convey information. For example, the "curly brackets" that contain children of a node can be closed to say "There is no more information to go here.", or open to say "This child list appears incomplete - more information exists in this direction." (see Figure 5.2). If

the update is optimized enough to allow filter determination for the entire hierarchy, then these cues could be accentuated to convey information about *how much* more remains outside the current screen boundary. For example, the further a node's name is from the edge of the display, the less information is off the screen (Figure 6.9).

Earlier interactions of FlexView used proportionally scaled fonts. This proved to be too computationally intensive. Replacing the scaled fonts with one pixel high lines (of proportional width) in the overview improved performance, but not enough. Fixed fonts were integrated into the interface, improving performance even more, since a copy of the fonts was read on startup and always used. Though scaled fonts would provide the highest quality visualization, proportionally scaling fonts by emphasis value remains too computationally expensive..

Summary. FLEXVIEW's big weakness is its unacceptable update performance, especially under conditions of restrictive filtering. This must be overcome to make the system usable. Optimizations have been suggested to show where the required increase in speed may will come from and how it can be achieved. The other critical limitation is the lack of a filter interface to deal with non-numeric and discrete attribute value types. Other limitations in FLEXVIEW include a complex filtering process that is difficult to understand, limitations of the current filter specification interface, and the propagation of values through the tree. However, the improvements suggested in this section could overcome some of these problems.

6.3 Informal User Testing

I have observed and received comments from four users who tried FLEXVIEW (with no defined tasks) on their own file systems. Two of the users were familiar with the system and the research behind it, the other two were not. While FLEXVIEW has informally demonstrated its strengths, it is not ready for real user testing due to the performance degradation that occurs as severe filtering of large hierarchies occurs. This would seriously undermine users from seeing the strengths in FLEXVIEW's dynamic abilities. However, the usability of FLEXVIEW can be predicted, since it is

designed to incorporate the best features and characteristics of its precursors, which have been well examined in usability studies and through the literature.

The largest complaint by all users focused on the occasional dismal performance. This need for optimization cannot be stressed enough. Much of the exploratory power of FLEXVIEW, which can be seen on smaller hierarches, is lost as the performance lags when updating the display of large hierarchies. Initially, users' reaction to the lack of response after selecting a filter range or navigating to a new node was to repeat the action, which caused it to be performed again...eventually. In the worst (most restrictively filtered) case, a 2700 node hierarchy takes about 40 seconds to update completely, though typical times were shorter than a second, which allows scrolling in real-time. Since the major bottleneck it the unnecessary node traversal, the optimizations suggested in the previous section should result in a very usable and scalable system.

With experience, users adapted to FLEXVIEW's slow response, and quickly discovered FLEXVIEW's strengths in exploring their own files. For example, FLEXVIEW provided new insight into seemingly familiar information. Comments like "I didn't realize those files were here. I thought I had deleted them." and "Hmmmm...got a lot of junk here." The overview provided a good overall snapshot for users, who said things like "At a glance [FLEXVIEW] gives a good view." and "[FLEXVIEW] gives an idea of the direct hierarchy.".

The only major difficulty users experienced, besides the slow response, was with difficulties in trying to grasp the meaning of the importance value, and how it related to the Interval Emphasis value. This unfamiliar concept is not that easy to grasp, especially with complex node attribute descriptions. As one user said, "The emphasis/threshold control is not as clear, it is harder to understand. The value has no intuitive meaning to me." This problem has been examined in the weaknesses and improvements **Reduce User Cognitive Load**^{COGN} paragraphs.

In spite of these problems, the test subjects, as well as people who had seen demonstrations, have been positive about the ideas presented. Some went as far as to say

"It is just what I need. When will it be ready?" Clearly, more in depth study of users working on their own hierarchical structures over longer periods is required. This would require implementation of a number of the suggestions in this chapter.

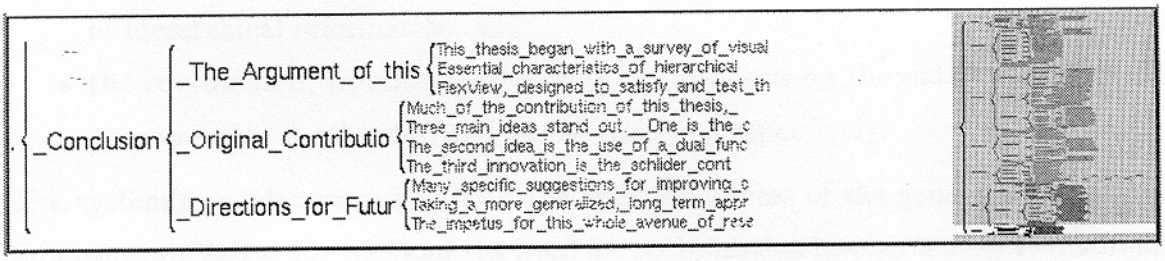
6.4 Summary

This chapter presented the strengths and weaknesses of FLEXVIEW and suggested improvements to strengthen FLEXVIEW. It also presented the results of some informal user testing, which served to reinforce FLEXVIEW's weaknesses.

Some important generalizations can be made about characteristics that improve information visualization approaches. The successful attainment of many of the visualization guidelines, including showing data relationships and characteristics, showing the overall data structure or context, showing detail information, and generally using space more efficiently, has increased over time, due to improved graphical user interfaces. Large hierarchy presentation has improved because of a reduction in presentation of local detail in favor of global context, as has been a general trend. Due to the greater increase in data volume, information overload is as big as problem as ever. The greater focus on global context has improved scalability marginally, if only by increasing the threshold at which information overload becomes a problem. Filtering provides a means to reduce screen clutter while continuing to present the necessary and important information, since the amount of information available is increasing faster than hardware's capability to display the information.

Exploiting human perceptual capabilities, by the use of three dimensional representations, animation, and new viewing paradigms have been shown to be effective strategies. Presenting a consistent structure across a visualization simplifies a person's overhead when dealing with complex information spaces.

Chapter 7



Conclusion

7.1 Summary of Thesis

Attention has been focused in this thesis on the visualization of large, loose structured, hierarchical information space. This thesis began with a survey of visualization approaches for hierarchical data. Technological advances have steadily increased graphical capabilities, resulting in far richer views on the data, particularly from a comprehension standpoint. Innovative information layout and distortion paradigms have extended the threshold of information volume that can be effectively visualized. Simple filtering has pushed this threshold even further. However, increases in the magnitude of available information have been greater still, straining even current graphical systems' limitations with scalability problems.

Essential characteristics of hierarchical visualization were distilled from the strengths and shortcomings of the visualization approaches and the accompanying illustrative systems. To address points missed by the systems, these characteristics were augmented and then all characteristics were used to generate a set of guidelines for the design and evaluation of hierarchical visualization systems.

FLEXVIEW, designed to satisfy and test these guidelines, integrates survey system characteristics with novel interface solutions. Major design features include

- the extensive use of emphasis filtering to refine the information visualization,
- the use of dynamic queries to encourage exploration and present data characteristics,
- the abstraction of the visualization process to apply to a wide and flexible range of hierarchical information, and
- the coordinated, simultaneous display of two views on the data space, focused on presenting local detail and global context, respectively.

The system's use demonstrates both the appropriateness of the generated visualization concepts and guidelines and the need for optimization improvements to maintain adequate performance when working with large quantities of information.

7.2 Original Contributions

Much of the contribution of this thesis lies in the guidelines generated from the system survey. These guidelines present an important summary and distillation of ideas from various sources into a common, simplified framework. The innovations presented in this thesis occurred in tandem with the evolution of these ideas.

Three main contributions stand out. One is the concept of sophisticated exclusion and emphasis filtering to improve the quality of data presented, based on an abstract notion of a node's importance. This has been done on three layers: the attribute filter ranges; the attribute weighting; and the node weighting. The current implementation uses binary and discrete values for the weighting of attributes and nodes respectively. Extensions have been suggested to make these, as well as the attribute filter, proportional and varying. One of the best means for dealing with future scalability issues is the ability to concentrate the display focus on items of most importance while de-emphasizing or suppressing other information.

The second idea is the use of a dual functionality overview view closely coupled with the detail view. As a powerful, miniature scale, this overview conveys information about global data trends and anomalies, and it provides the fast navigational capa-

bility via scrolling or direct access to areas of interest in the linked detail view. Structurally consistent in its layout and data placement, this view reduces the cognitive load users face in keeping their mental data model synchronized with the structural hierarchy visualization presented on the screen.

The third innovation is the *schlider* control, specifically designed to facilitate simple and compact selection of multiple sub-ranges across a continuous range. Suggested enhancements to this exclusion filter control would add varying emphasis capability across the selected sub-ranges. This would be accomplished by setting an emphasis mode selected from an emphasis palette. This functionality would provide users with the flexibility to alter the visualization based on display filter criterion.

Another major contribution is the generation of a set of design and evaluation guidelines for hierarchical information visualization, summarized in Table 3.4, page 57,

In addition to these major contributions, some lesser achievements include:

- the identification of key data-oriented presentation features, summarized in Table 3.1, page 28;
- the tabulation of visualization techniques useful in hierarchical visualization, summarized in Table 3.2, page 29;
- the categorization and evaluation of hierarchical visualization approaches, summarized in Table 3.3, page 48;
- the creation of an *interval emphasis* control for emphasis level selection, and
- the FLEXVIEW system, which even with its need for optimizations has demonstrated exciting, innovative improvements in hierarchical data display.

7.3 Directions for Future Research

Many specific suggestions for improving FLEXVIEW's implementation were noted in the previous chapter. Major suggestions include:

- many ideas for optimizations, such as
  - adding pointer links to each attribute for fast, direct traversal,

  - caching attribute filter and node importance values,
  - caching presentation characteristics, and
  - enhancing the enclosing curly braces to convey the displayed:hidden ratio,
- enhancing and extending the filtering capabilities,
- extending the flexibility of hierarchical display with resizable node boxes,
- decreasing the susceptibility to scalability problems with levels of overviews and a structural emphasis in the overview,
- extending the interface to allow independent foci,
- adding animation to the change in emphasis display update process.

One crucial omission in this research is a formal user study to confirm the concepts presented and to provide feedback for improvements. This is an obvious next step once optimizations have been done. A controlled usability study could be used here to see if FLEXVIEW demonstrates in practice the benefits that have been proposed while under initial development. A comparison between FLEXVIEW and other modern systems addressing the same problems would be also interesting. FLEXVIEW, *Cone Trees* and *Treemaps* are ideal candidates for comparison, because they are technologically advance systems with similar functionality but striking differences in interface design philosophy. *Treemaps* is particularly interesting, since it has already been compared to Unix commands in a user study.

Taking a more generalized, long term approach brings a myriad of research directions to light. Study could be undertaken on the biological questions associated with human visual and aural perception, such as the use of color for emphasis, the applicability of motion (like vibration as an indicator of interest) and the use of earcons (a play on icons, based on sound) to indicate importance. Some relevant psychological questions address how the functionality of human memory and mental information organization would affect design guidelines. Of course, engineering issues involved with exploiting technological advances must be considered. Research needs to be expanded to improve our knowledge of how humans deal with information and thus allow researchers to develop more effective information interfaces.

The impetus for this whole avenue of research was all-too-common infoglut, as seen by burgeoning file systems. While this has continued to be the main motivator, it has become readily apparent that all the problems encountered in file systems visualization exist in most large information spaces. I believe that future work will show many of the concepts developed in this thesis are, pleasingly, widely applicable to other domains as well.

Bibliography

[App87] Apple Computer. Inc. Human interface guidelines: The apple desktop interface. 1987.

[AHU83] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structure and Algorithms.* Addison-Wesley, Reading, Massachusetts, USA, 1983.

[AS94a] C. Ahlberg and B. Shneiderman. The alphaslider: A compact and rapid selector. In B. Adelson, Dumais S., and Olson J., editors, *Proceedings of the Conference on Human Factors in Computing Systems (ACM SIGCHI'94)*, pages 365–371, Boston, Mass, April 24–28 1994. ACM, ACM Press.

[AS94b] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In B. Adelson, Dumais S., and Olson J., editors, *Proceedings of the Conference on Human Factors in Computing Systems (ACM SIGCHI'94)*, pages 313–317, Boston, Mass, April 24–28 1994. ACM, ACM Press.

[AWS92] C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic queries for information exploration: an implementation and evaluation. In P. Bauersfule, J. Bennett, and G. Lynch, editors, *Proceedings of the Conference on Human Factors in Computing Systems (ACM SIGCHI'92)*, pages 619–626, Monterey, California, USA, May 3–7 1992. ACM, ACM Press.

[AWS93] C. Ahlberg, C. Williamson, and B. Shneiderman. *Dynamic Queries for information exploration: an implementation and evaluation.*, pages 282–284. In Shneiderman [Shn93], 1993.

[BB70] M. Burton and R. Burton, editors. *The International Wildlife Encyclopaedia.* B. P. C. Publishing Limited, New York, New York, 1970.

[Bly89] S. Bly. Presenting information in sound. In K. Bice and C. Lewis, editors, *The use of non-speech audio at the interface. Tutorial 10 Notes, Conference on Human Factors in Computing Systems (ACM SIGCHI'89)*, pages 5.16–5.20, Austin, Texas, April 30–May 4 1989. ACM, ACM Press.

[BO84] L. Bannon and C. O'Malley. Problems in evaluation of human-computer interfaces: A case study. In *Interact '84—First IFIP Conference on Human-computer Interaction*, pages 280–284, September 1984.

[BR90] R. K. Belew and J. Rentzepis. HyperMail: Treating electronic mail as literature. In F. H. Lochovsky and R. B. Allen, editors, *Proceedings of the Conference on Office Information Systems (COIS '90)*, Cambridge, Mass., April 25–27 1990. ACM.

[BWI90] D. V. Beard and J. Q. Walker II. Navigational techniques to improve the display ot large two-dimensional spaces. *Behaviour and Information Technology*, 19(6):451–466, 1990.

[CD92] R. G. Côté and S. Diehl. Searching for common threads. *BYTE*, 17(6):290–305, June 1992.

[CWMS91] R. Chrimera, K. Wolman, S. Mark, and B. Shneiderman. Evaluation of three interfaces for browsing hierarchical tables of contents. Technical Report CAR-TR-539, CS-TR-2620, University of Maryland, February 1991.

[Dyk91] P. C. Dykstra. `xdu`, An X window "disk usage" display. Software Documentation, 1991.

[EE94] D. C. Englebart and W. K. English. Historical video: A research center for augmenting human intellect. From SIGGraph Video Review Issue 106. CHI'90 Technical Video Program, ACM Press, 1994.

[EFB+94] J. Encarnação, J. Foley, S. Bryson, S. K. Feiner, and N. Gershon. Research issues in perception and user interface. *IEEE Computer Graphics and Applications*, 14(2):67–69, 1994.

[ELK+91] D. E. Egan, M. E. Lesk, R. D. Ketchum, C. C. Lochbaum, J. R. Remde, M. Littman, and T. K. Laundauer. Hypertext for the electronic library? CORE sample results. In J. Leggett, John, editor, *Proceedings of the Third ACM Conference on Hypertext*, pages 299–312, San Antonio, Text, December 15–18 1991. ACM.

[Ell90] R. Ellson. Visualization at work. *Academic Computing*, page 26, March 1990.

[Fee92] W. R. Feeney. Gray scale diagrams as business charts. In E. A. Kaufman and G. M. Nielson, editors, *Proceedings of the 3rd International IEEE Visualization Conference (Vis'92)*, pages 140–147, Boston, Mass., USA, October 19-23 1992. IEEE, IEEE Computer Society Press.

[Fei88] S. K. Feiner. Seeing the forest for the trees: hierarchical display of hypertext structures. In *Proceedings of the Conference on Office Information Systems (COIS '88)*, pages 205–212, Palo Alto, California, March 1988. ACM.

[Fol90] P. W. Folts. Using latent semantic indexing for information filtering. In F. H. Lochovsky and R. B. Allen, editors, *Proceedings of the Conference on Office Information Systems (COIS '90)*, Cambridge, Mass., April 25–27 1990. ACM.

[Fur86] G. W. Furnas. Generalized fisheye views. In M. Mantei and P Orbeton, editors, *Proceedings of the Conference on Human Factors in Computing Systems (ACM SIGCHI'86)*, pages 16–23, Boston, Mass., April 13-17 1986. ACM, ACM Press.

[Gav86] W. Gaver. Auditory icons: Using sound in computer interfaces. *Human Computer Interaction*, 2(2):167–177, 1986.

[Gol91] C. F. Goldfarb. *The SGML Handbook.* Oxford University Press, New York, New York, USA, 1991.

[Hoa90] E. D. Hoadley. Investigating the effects of color. *Communications of the ACM*, 33(2):120–139, February 1990.

[Joh92] B. Johnson. Tree Maps. *Perspectives on Technology*, 5(1 and 2):16, Fall 1992.

[JS91] B. Johnson and B. Shneiderman. Tree-Maps: A space-filling approach to the visualization of hierarchical information structures. In G. M. Nielson and L. Rosenblum, editors, *Proceedings of the 2nd International IEEE Visualization Conference (Vis'91)*, pages 284–291, San Diego, California, USA, October 22-25 1991. IEEE, IEEE Computer Society Press.

[JS93] B. Johnson and B. Shneiderman. *TreeMaps.* pages 309–322. In Shneiderman [Shn93], 1993.

[Kau92] K. Kaugars. BASS and CAT: Two fisheye views of trees. Technical Report NMSU-TR-92-CS-06, New Mexico State University, March 1992.

[KB92] K. Kaugars and A. Brazma. CATGraph: visualizing large labelled graphs. Technical Report NMSU-TR-92-CS-08, New Mexico State University, April 1992.

[KMG+91] B. Kahle, H. Morris, J. Goldman, T. Erickson, and J. Curran. Interfaces for Wide Area Information Servers (Version 0.7). 1991.

[Knu73] D. E. Knuth. *The Art of Computer Programming Volume I: Fundamental Algorithms, Second Edition.* Addison-Wesley, Reading, Massachusetts, USA, 1973.

[Kuh90] W Kuhn. Editing spatial relations. In *Proceedings of the 4th International Symposium on Spatial Data Handling*, pages 423–432, Zurich, Switzerland, 1990.

[KW85] S. G. Kochan and P. H. Wood. *Exploring the UNIX System. 2E.* Hayden Books, Howard W. Sams and Company, Indianapolis, Indiana, 1985.

[LA94] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, June 1994.

[Lam86] L. Lamport. LaTeX: *A Document Preparation System.* Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1986.

[LHMR91] H. Levkowitz, R. A. Holub, G. W. Meyer, and P. K. Robertson. Color vs. black -and-white in visualization. Panel, Vis '91, October 22-25 1991.

[Mac90] L. W. MacDonald. Using color effectively in displays for computer-human interface. *DISPLAYS*, pages 129–142, July 1990.

[Mar92] P. Marshall. WAIS: The Wide Area Information Server or Anonymous What??? In *Canadian Net'92*, St. Johns, Nfld., June 1992.

[MGT+87] T. W. Malone, K. R. Grant, F. A. Turbak, S. A. Brobst, and M. D. Cohon. Intelligent information sharing. *Communications of the ACM*, 30(5):390–402, 1987.

[Mil56] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capability for processing information. *Psychological Science 63*, pages 81–97, 1956.

[MRC91] J. D. Mackinlay, G. G. Robertson, and S. K. Card. The Perspective Wall: Detail and context smoothly integrated. In S. P. Robertson, G. M. Olson, and J. S. Olson, editors, *Proceedings of the Conference on Human Factors in Computing Systems (ACM SIGCHI'91)*, pages 173–179, New Orleans, Louisiana, USA, April 27–May 2 1991. ACM, ACM Press.

[MS94a] Microsoft Corporation. *Microsoft MS-DOS Concise User's Guide. Version 6.22.* Microsoft Press, Seattle, Washingtion, 1994.

[MS94b] Microsoft Corporation. *Microsoft Windows User's Guide. Version 3.1.* Microsoft Press, Seattle, Washingtion, 1994.

[Nat89] National Center for Supercomputing Applications. *NCSA Telnet Developers Guide*, 1989.

[Nor88] D. Norman. *The Psychology of Everyday Things.* Basic Books, Inc., 1988.

[O'H92] M. O'Henly. An internet gopher for the University of Victoria. Technical report, Library Systems Office, University of Victoria, 1992.

[Per93] K. Perlin. Pad An alternative approach to the computer interface. In *Proceedings of Computer Graphics '93 (SIGGRAPH '93)*, Los Angeles, California, USA, August 1-6 1993. IEEE.

[Pop89] S. Pope. A navigator for Unix. From SIGGraph Video Review Issue 46. CHI'89 Technical Video Program - Computing Environments, ACM Press, April 30–May 4 1989.

[Pre94] J. Preece. *Human-Computer Interaction.* Addison Wesley, Reading, Massachusetts, USA, 1994.

[Qua90] J. S. Quarterman. *The Matrix: Computer Networks and Conferencing Systems Worldwide.* Digital Press, Bedford, Massachusetts, USA, 1990.

[Rhe92] P. Rheingans. Color, changes, and control for quantitative data display. In E. A. Kaufman and G. M. Nielson, editors, *Proceedings of the 3rd International IEEE Visualization Conference (Vis'92)*, pages 252–259, Boston, Mass., USA, October 19-23 1992. IEEE, IEEE Computer Society Press.

[Ric91] J. R. Rice. Ten rules for color coding. *Information Display*, 7(3):12–14, March 1991.

[RMC91a] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone Trees: Animated 3D visualizations of hierarchical information. In S. P. Robertson, G. M. Olson, and J. S. Olson, editors, *Proceedings of the Conference on Human Factors in Computing Systems (ACM SIGCHI'91)*, pages 189–194, New Orleans, Louisiana, USA, April 27–May 2 1991. ACM, ACM Press.

[RMC91b] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone Trees: Excerpts from IEEE Visualization '91 video. Video, 1991.

[RMC91c] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Perspective Wall: Excerpts from IEEE Visualization '91 video. Video, 1991.

[SA82] R. Spence and M. D. Apperley. Data base navigation: an office environment for the professional. *Behaviour and Information Technology*, 1(1):43–54, 1982.

[SB92a] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In P. Bauersfule, J. Bennett, and G. Lynch, editors, *Proceedings of the Conference on Human Factors in Computing Systems (ACM SIGCHI'92)*, pages 83–91, Monterey, California, USA, May 3–7 1992. ACM, ACM Press.

[SB92b] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. Technical Report 84, Digital Equipment Corporation Systems Research Center, Palo Alto, California, March 17 1992.

[SG93] D. Schaffer and S. Greenberg. Sifting throught hierarchical information. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, and T. White, editors, *Adjunct Proceedings of the Conference on Human Factors in Computing Systems (ACM/IFIP INTERCHI'93)*, pages 173–174, Amsterdam, The Netherlands, April 24–29 1993. ACM, ACM Press.

[Shn87a] B. Shneiderman. *Designing the User Interface*, chapter 2: Theories, Principles, and Guidelines. Academic Press, 1987.

[Shn87b] B. Shneiderman. *Designing the User Interface*, chapter 11: Information Exploration Tools. Academic Press, 1987.

[Shn91] B. Shneiderman. Tree-Maps: Excerpts from IEEE Visualization '91 video. Video, 1991.

[Shn93] B. Shneiderman, editor. *Sparks of Innovation in HCI.* Ablex Publishing Corporation, Norwood, New Jersey, USA, 1993.

[SSR93] M. Sarkar, S. Snibbe, and S. Reiss. Stretching the rubber sheet: a metaphor for visualizing large structure on a small screen. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '93)*, Atlanta, Georgia, USA, November 1993. ACM.

[SZB+93] D. Schaffer, Z. Zuo, L. Bartram, J. Dill, S. Dubs, S. Greenberg, and M. Roseman. Comparing Fisheye and Full-Zoom techniques for navigation of hierarchically clustered networks. In N. Jaffe, editor, *Proceedings of Graphics Interface '93*, pages 87–96, Calgary, Alberta, May 19–21 1993. Canadian Man-Computer Communications Society.

[SZG+96] D. Schaffer, Z. Zuo, S. Greenberg, J. Dill, S. Dubs, M. Roseman, and L. Bartram. Navigating hierarchically clustered networks through Fisheye and Full-Zoom methods. *ACM Transactions on Office Information Systems*, 1996.

[Tes92] J. Tesler. `fsn`, the 3D File System Navigator. V1.1.1. Software Documentation, 1992.

[Thi90] H. Thimbleby, editor. *User Interface Design.* Addison-Wesley, 1990.

[TJ92] D. Turo and B. Johnson. Improving the visualization of hierarchies with treemaps: design issues and experimentation. In E. A. Kaufman and G. M. Nielson, editors, *Proceedings of the 3rd International IEEE Visualization Conference (Vis'92)*, pages 124–131, Boston, Mass., USA, October 19-23 1992. IEEE, IEEE Computer Society Press.

[Tre93] B. Johnson. Treemaps: visualizing hierarchical and categorical data. Doctoral dissertation, University of Maryland, 1993.

[Tuf83] E. R. Tufte. *The Visual Display of Quantitative Informations.* Graphic Press, Cheshire, Connecticut, 1983.

[VHW87] K. J. Vincent, B. C. Hayes, and R. C. Williges. Assaying and isolation individual differences in searching a hierarchical file system. *Human Factors*, 29(3):349–359, 1987.

[WS92] C. Williamson and B. Shneiderman. The Dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system. Technical Report CAR-TR-602, Human-Computer Interaction Laboratory, University of Maryland, January 1992.

[WS93] C. Williamson and B. Shneiderman. *Dynamic HomeFinder.*, pages 295–307. In Shneiderman [Shn93], 1993.

[ZDB93] Z. Zuo, J. Dill, and L. Bartram. Variable Zoom: A fisheye view of hierarchical networks. In J. W. Buchanan, editor, *Proceedings of the Fifth Annual Western Computer Graphics Symposium*, pages 4.1–4.4, Silver Star Mountain, Vernon, British Columbia, March 28–30 1993. University of British Columbia.

[Zuo92] Z. Zuo. A new display technique for large hierarchical networks. Master's thesis, Simon Fraser University, December 1992.

Appendix A

Complete Node Description File with Comments

```
# Node Description assumes Data File will have
#  -(L)eaf or (U)nleaf,                used for constructing the hierarchy
#  -(root/path/path/path/leaf_name),   used for hierarchy creation/placement
#    * an unleaf is an interior node
# Node Description (this file) contains:
#  -a line with the root node description:
#    "R U <root_name> <attribute value list>"
#  -attribute descriptions, one per line:
#    "S x name_of_attribute ", or
#    "C l name_of_attribute min max", or
#    "D o name_of_attribute n(umber_of_choices) choice-1 choice-2...choice-n"
#   where:
#     the 1st character means S(tring), C(ontinuous) or D(iscrete) attribute
#     the 2nd character means x (don't make schlider) or
#                             l (make schlider with numeric bounds) or
#                             o (make schlider with date bounds)
#     the next string is the name of the attribute (with no spaces in it)
#     for continuous attribute, the minimum and maximum values appear, or
#     for discrete attribute, the number of choices and list of them appear
#
#  NOTE: all values withing this file and the data file must be separated
#        by a single space, and the first value must occur in the first column.
# Comments have a '#' in the first column
#
C l completeness 0 10            10 is most complete
C l number_of_revisions 0 1000
C l recency_of_revisions_(days) 0 365
C l section_size_(pages) 0 40
C l section_first_page 1 200
C l section_last_page 1 200
D x section_type 5 text figure table chart mixed
#
R U thesis 0 0 0 0 1 74 mixed
```

Figure A.1: Complete Node Description File with documentation.