10-6-2010

Perpetual exploration of relational information and enhanced star glyphs for multi-source data visualization

Andrew Crowell

PERPETUAL EXPLORATION OF RELATIONAL INFORMATION AND ENHANCED STAR GLYPHS FOR MULTI-SOURCE DATA VISUALIZATION

by
Andrew Crowell

A Thesis

Submitted in partial fulfillment of the requirements of the
Master of Science in Engineering Degree
of
The Graduate School
at
Rowan University
March 2010

Thesis Chair: Adrian Rusu, Ph.D.

In loving memory of my paternal grandparents, Tom and Betty.

ABSTRACT

Andrew Crowell
PERPETUAL EXPLORATION OF RELATIONAL INFORMATION AND ENHANCED STAR GLYPHS FOR MULTI-SOURCE DATA VISUALIZATION
2009/10
Adrian Rusu, Ph.D.
Master of Science in Engineering

Information Visualization is an area of research concerned with the presentation of abstract data in a visual format. It has many diverse applications in fields of software engineering, information sciences, biology, chemistry, and medical, among others.

This thesis is primarily concerned with the application of information visualization to the display of directed graphs and to the display of multivariate data for analysis. Two novel applications will be presented that are both advancements in the field of information visualization.

The first application applies to the visualization and navigation of large or infinite graphs and networks. The quality of a graph drawing algorithm is often measured by its edge crossings, angular resolution, aspect ratio, and node labeling. Algorithms for drawing trees in general are segregated from algorithms for drawing graphs. This thesis will present a graph visualization system that uses a novel interconnection between a tree drawing algorithm and graph drawing techniques.

First, the graph is transformed into a tree and nodes that have multiple parent connections within the graph are duplicated within the tree. While some of the connection information is lost during this transformation, the multiple connections can be regained by interactively displaying the details based on the degree of interest. The tree

drawing algorithm that is used also provides a way to display a connection without drawing an edge. This edgeless visualization allows edge crossings and angular resolution issues to be eliminated. The graph is represented using a rings visualization, so it naturally has an aspect ratio of 1. Finally, a circular labeling method is used that provides user-friendly labels that do not overlap and clearly show node affiliation. The result is an interactive, navigable graph visualization system that is a useful tool for the display of large or infinite hierarchical and network graphs.

The second application involves the display of multivariate, multi-set data. Generally, the purpose of graphically displaying large datasets is to find the general average of where most of the data lies and then to find the outliers (i.e. the data points that are most distant from the average). The visualization presented in this thesis will attempt to find both by using a multitude of common graphing techniques to expand upon the traditional star glyph, creating a three-dimensional visualization that allows comparisons among multiple datasets of multivariate data.

ACKNOWLEDGMENTS

First of all, I would like to thank God for the opportunity He has given me. All I have accomplished is through Jesus Christ my Lord, and to the glory of God. I would not be here, today, if not for His grace.

I would also like to express my gratitude and appreciation for my major advisor and friend, Dr. Adrian Rusu. His enthusiasm for research and concern for his students has touched my life, greatly. The guidance, encouragement, and motivation he provided on a daily basis were what made this thesis possible. Many thanks to him for supporting me in my endeavors as a Research Assistant. I will always be grateful to him for the life-altering opportunities he created for me.

Many thanks to Dr. Shreekanth Mandayam and Dr. Ganesh Baliga for being a part of my thesis committee.

I would also like to thank Mr. Andrew Fabian, Mr. Bryan Petzinger, Mr. Eric Thomas, and Ms. Linda Falconi for the assistance they provided as members of the Software Engineering, Graphics, and Visualization (SEGV) Research Group.

Also, many thanks to my close friend Confesor Santiago for his contributions to my research, his encouragement, and his advice. I would not be here without the love and support of my friends and family. Special thanks to my parents, who have supported me since the day I was born; to my mother for teaching me how to learn from an early age, and to my father for sparking my interest in technology: I could not have asked for a better education than the one you provided me. Also thanks to my maternal grandparents, Hal and Nancy, for helping to teach me the value of hard work and for their love and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Information Visualization

Information visualization is the general concept of taking abstract data and applying algorithms to it to allow it to be represented in a meaningful manner to the user. Many types of data have had information visualization concepts applied to them, but this thesis will discuss only two types of abstract data: relational data (graphs and trees) and multi-set, multivariate data.

When applying information visualization concepts to a set of abstract data, there are many properties that can be used to measure the quality of the final visualization. However, the most important part of any visualization is that it conveys to the user the meaning of the specific data being represented. In relational data, this meaning may be the relationships among a terrorist network or the pages connecting one Web page to another. Multivariate data is often visualized in order to search for outliers which may represent inconsistencies in financial data or a group of people that has not yet been marketed to.

The following sections will discuss in detail the two types of abstract data that will be referenced in this thesis.

1.2. Graph and Tree Drawing

Graph drawing algorithms and tree drawing algorithms are traditionally treated as two separate domains. Although a tree is a subset of a graph and they are structurally similar, when developing a drawing algorithm there is typically little that can overlap into both domains. Both areas of research are concerned with the automatic generation of aesthetically pleasing visualization of relational information.

Graph drawing is concerned with information that has no limitation on connections. Vertices represent entitities in whatever subject is being visualized and edges between the vertices represent some relationship between the connected entities. These entities can be connected by directed edges or non-directed edges. Directed edges would signify that the relationship has some type of order (e.g. order of precedence), while a non-directed edge would signify that the relationship is mutual between both entities (e.g. social connection). Entities can be connected in any way, and a single entity can have connections to many other entities. A graph can be complicated to draw since it has no limitations on how connections are formed between nodes.

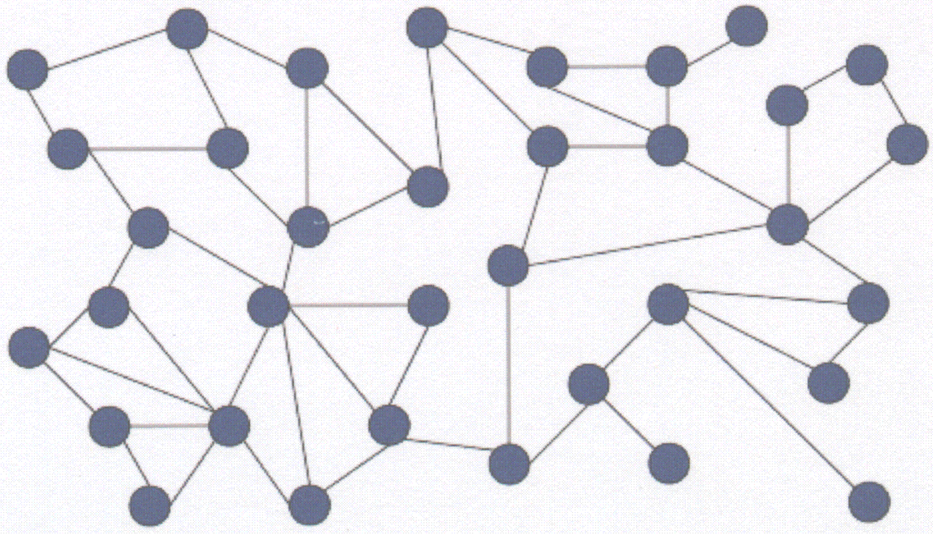

Figure 1.1 A planar graph. The graph can be drawn on a two-dimensional plane with no edge crossings.

A planar graph (Figure 1.1) would be the easiest type of graph to visualize. A graph is planar when the vertices of the graph can be arranged on a two-dimensional plane in such a way that all connections can be represented with no edge crossings within the drawing. This is a special type of graph and very few graphs fall into this category. All other graphs fall into the non-planar category and are much more complicated to visualize.

Non-planar graphs cannot be represented in two-dimensional space without edge crossings (Figure 1.2), and so there is no correct answer to visualize each graph. Many different graph drawing algorithms have been developed for visualizing non-planar graphs. Each algorithm has its own strengths and weaknesses and many of them choose just one or a few properties to optimize. These properties can include aspect ratio, edge crossings, angular resolution, area, and edge length, among others. These properties provide a quantitative measurement of the quality of a graph drawing, however, the ultimate goal of any graph drawing algorithm is to provide a graph visualization that is aesthetically pleasing and useful to its target crowd.

Tree drawing is concerned with information that is much more limited in its relationships than graph drawings. A tree is a graph that contains no loops and a single parent for each node. Each node in a tree can be assigned to a single level based on its depth from the root node of the tree. Each node can only have a single connection from a node above its own level (the parent) and can have many connections to nodes below its own level (the children).

There are many graphs that cannot be categorized as a tree due to the strict requirements. The most common type of relational information that can be visualized as a tree is an information hierarchy. An information hierarchy is a collection of relational information that is arranged in a ranking organization where each entity is subject to a single other entity, except for the top level element. Some common examples of information hierarchies are management structures, directory structures, and evolutionary trees.

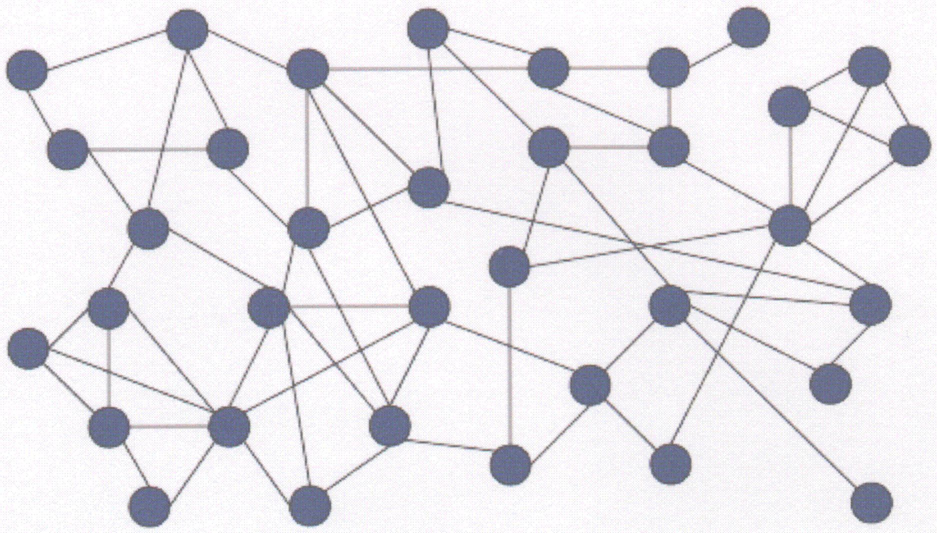

Figure 1.2 A non-planar graph. There is no way to draw this graph on a two-dimensional plane without having edge crossings.

1.3. Properties of Graph Drawings

There are many properties of graph drawings that are used to determine the quality of the drawing. Four of these properties are addressed in this thesis and are described in detail in the next four sections.

1.3.1. Edge Crossing

One of the most popular properties that graph drawings often are concerned with is edge crossing. Typically, it is best to have a drawing with a low number of edge crossings, since edge crossings in a drawing can confuse the user and make the connections between nodes not as visible. A non-planar graph cannot traditionally be represented in two-dimensional space without edge crossings.

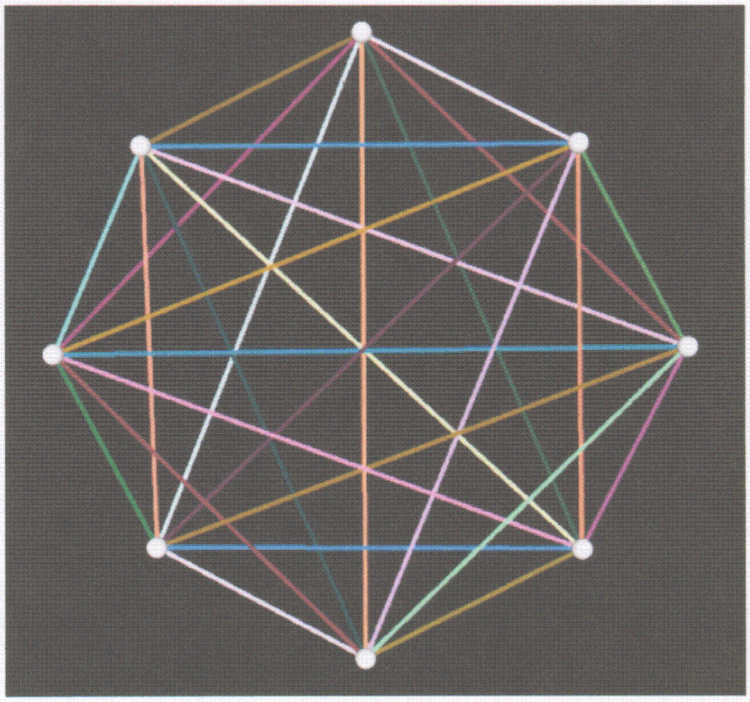

Figure 1.3 A graph using edge coloring to distinguish edge crossings as described in [1].

Some algorithms have been developed that attempt to determine the optimal drawing that would provide the least amount of edge crossings. The quality of these drawings can easily be measured by counting the number of edge crossings they produce.

Others have taken a less traditional approach to solving the edge crossing problem, such as coloring edges [1] (Figure 1.3) or displaying the graph in three-dimensional space.

1.3.2. Angular Resolution

Another property that has to do with how readable the graph drawing is, is angular resolution. Angular resolution is the minimum angle between all sets of adjacent edges in the drawing. Typically, a higher angular resolution is better, because as the angular resolution is reduced it becomes increasingly harder to distinguish edges.

As with edge crossing, a traditional approach to angular resolution is limited by the graph structure. The maximum angular resolution of a single node in a graph drawing is $2\pi / N$, where N is the number of connections the node has.

1.3.3. Aspect Ratio

Aspect ratio is a property describing the dimensions of a graph drawing. It is the numerical value of the ratio of the width to the height. This value is an important property of a drawing as it tells how well it can be displayed on the provided screen space. The general rule of thumb is that an aspect ratio close to a value of 1 is desirable, however this rule is somewhat flexible due to the different screen orientations available. In the past most monitors were commonly 4 : 5 ratio, but recently 5 : 8 has become a more popular screen dimension for computers and 16 : 9 for televisions. Due to this shift in screen dimensions, an aspect ratio of close to 2 is much more acceptable than it was in past years.

1.3.4. Labeling

Labeling is an important property of most graph drawings. Most often a vertex in a graph drawing represents some entity that is affiliated with textual information. For instance, in a social network each node represents a person. Every person has a name, and therefore, this name must be conveyed to the user in some way. Since names can only be conveyed textually, the graph drawing now requires text to be drawn on the graph.

There are generally two categories of labeling in graph drawing: edge labeling and node labeling. Edge labeling (Figure 1.4) places a label for each edge of the graph and usually describes the relationship of the two nodes connected by the respective edge. Node labeling places a label for each node of the graph and describes the nodes themselves, not the relationships.

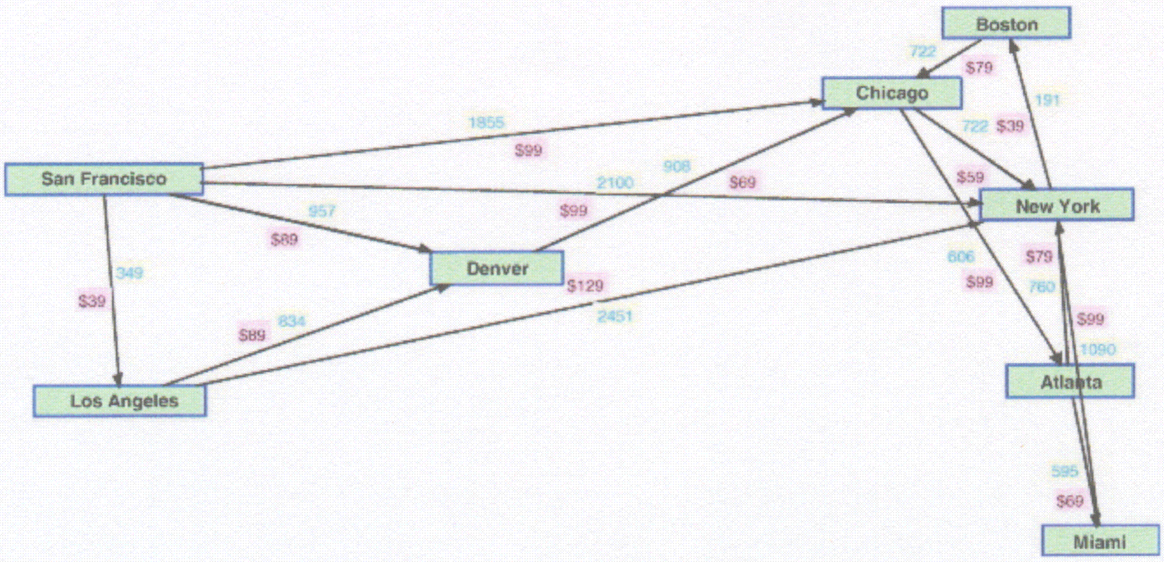

Figure 1.4 Edge labeling as described in [2].

The easiest way to label the graph drawing would be to draw the text centered over top of its respective node or edge. However, this method can be messy in a large graph, since the text will begin to overlap. Overlapping text, generally, should be avoided when labeling a graph drawing. Many labeling algorithms have been developed for different graph drawings. Some solutions to labeling are to use collision avoidance algorithms to avoid overlaps and to use interactive means to display the labels the user is most likely interested in.

1.4. Multi-Set, Multivariate Data

Multivariate data is data that is represented by more than one variable. There are many different types of multivariate data, but this thesis will focus on one specific type, which is a single continuous variable that is categorized by one or two discrete variables and weighted by another continuous variable. Some data that may fall into this category of multivariate data are surveys and financial data.

Multi-set data refers to high level categorization of data. Often each set represents a separate survey or system. These different data sets become a blocking factor in the data, and so the data cannot be mixed, but it can be compared to see what effect this blocking factor has. This blocking factor could represent different sources of data or a different system that is tested. This thesis will present an example of multi-set, multivariate data that is a measure of the accuracy of multiple safety critical systems in the United States National Airspace System categorized by types of aircraft.

1.5. Contributions and Outline

The major contributions of this thesis are two novel visualization systems. The first system allows for the visualization of an infinite or large graph structure by using a tree drawing of a small section of the graph and user interaction to navigate and gain details of the graph. The second system that will be presented provides a visualization system to analyze and compare multiple sets of multivariate data.

The thesis will follow this outline:

- In Chapter 1 (this chapter) we give an overview of the concepts of graph drawing and information visualization.
- Chapter 2 introduces the interactive graph visualization system and discusses some applications.
- Chapter 3 discusses the multi-set, multivariate visualization system and some applications of it.
- Finally, Chapter 4 summarizes the work of this thesis and identifies some future work.

2. PIEVIS: INTERACTIVE GRAPH VISUALIZATION USING A RINGS-BASED TREE DRAWING ALGORITHM FOR CHILDREN AND CRUST DISPLAY FOR PARENTS

2.1. Introduction

Many different graph drawing algorithms have been developed attempting to create an aesthetically pleasing graph visualization. Much research has been devoted to dealing with edge crossings in a visualization [1] [3] [4], since these edge crossings make the graph hard for the human eye to follow. As the number of connections to and from a particular node increases, the angle between the edges (angular resolution) [5] [6] [7] rapidly decreases, making it hard to distinguish each line. Another property often given to research is aspect ratio [5], which is the ratio of the width of the graph drawing to the height. It is typically desirable to have an aspect ratio close to 1, since this allows the drawing to easily fit into the available screen space. The last property addressed in this chapter is labeling. Adding text to a graphical visualization can be difficult, so many different algorithms have been developed attempting to find the most efficient way to label a graph [2]. Labels typically should be displayed in a way that allows as much important information as possible to fit inside the label without having labels overlap each other. It is also important to ensure the user can easily distinguish which label is affiliated with which node. This chapter presents a novel graph algorithm that addresses these four major issues using an interactive visualization and a tree drawing algorithm.

Often, tree drawing and graph drawing, though similar areas of research, are treated as two separate domains. Our visualization interconnects these two domains, using a tree drawing algorithm and graph drawing techniques to create an interactive directed graph visualization system. The graph visualization provided is a navigable visualization that attempts to display a detailed representation of only the data that is most likely of interest, while data that is not as interesting to the user may not be displayed or may be displayed in a more abstract form that allows the visualization to remain clean and aesthetically pleasing. Several different means of interaction are provided to gain several different levels of detail about interesting areas of the graph.

The visualization uses a rings-based tree drawing algorithm that represents connections via enclosing smaller rings inside larger rings. This unique way of showing connections between nodes allows for the removal of directed edges within the visualization. As a result, edge crossings and angular resolution issues are eliminated. Another feature of the visualization is a display of parent nodes along the edge of the tree similar to the crust of a pie.

2.2. Previous Work

The visualization system presented here applies a tree drawing algorithm and user interaction to display a directed graph. Both of these ideas have had a considerable amount of research applied to them. [8] presents a graph drawing created by adding the set of curved edges to a Treemap drawing. [9] applies a fisheye view and several other interactive techniques in order to remove clutter from a small section of a large graph drawing.

[10] is closely related to the work presented here. The author introduces a force-directed graph visualization that provides interactive means to display a section of a

graph as a tree which can then be incrementally explored. A smooth animation between transitions is used to help the user remain oriented within the visualization.

Several concepts used by these previous systems are applied in our visualization including incremental exploration, using a tree drawing algorithm for node placement, and applying user interaction to gain more information about an area of the graph.

2.3. Tree Representation

When representing a graph as a tree, the first step is removing the loops. Loops are created within a graph when a single node has multiple parent nodes. In order to not lose information when these loops are removed, each node is duplicated for the number of parent connections it has (Figure 2.1). By doing this the loops can be removed without actually removing any connections, though at this point it is difficult for the user to see these many connections.

The data is stored as a graph using Object Oriented Design [11] principles that allow references to be made to the same node (the Object) multiple times, without creating multiple physical copies of the same node. The tree drawing algorithm can then be applied to the graph structure, without inefficiently creating many copies of a node.

Our visualization begins with a basic FastRings tree visualization [12] (Figure 2.2) algorithm that was extended from the RINGS algorithm [13]. Although many of the techniques introduced in this chapter can be applied to RINGS, FastRings was chosen because it provides a top-down algorithm that is useful for quickly displaying large graphs. A bottom-up algorithm requires the entire graph to be analyzed before it can be displayed. Our visualization is intended for displaying large graph structures, such as the

World Wide Web, so a top-down algorithm is most practical. The FastRings algorithm allows the removal of edges, since connections are represented by child nodes being drawn inside the parent node, and since FastRings draws the tree in a circular format, this naturally provides the desirable aspect ratio of 1.

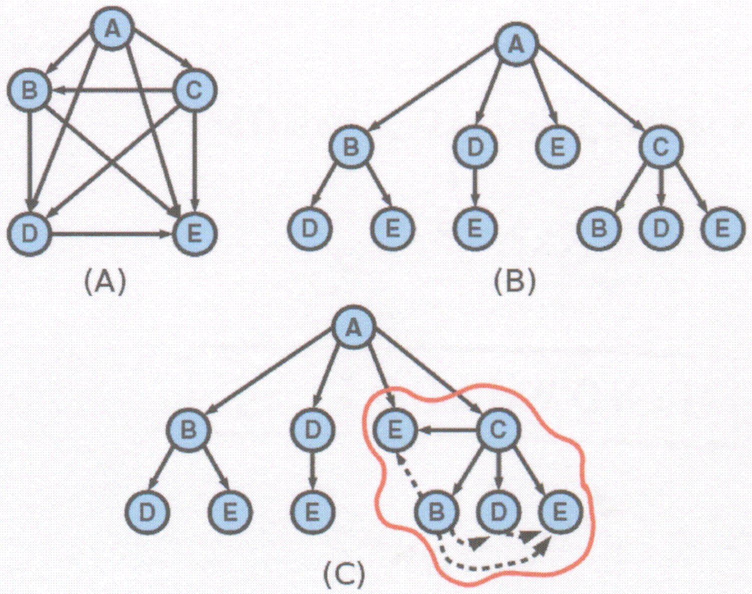

Figure 2.1 (A) The visualization begins with a directed graph, and (B) transforms it into a tree by removing loops and duplicating nodes. (C) Connections are reestablished when more detail is shown of sections of the drawing based on Degree of Interest interactions.

Another feature of the FastRings visualization is that the tree is visualized from the point of view (POV) of a particular node, called the focal node. Instead of displaying the entire tree structure at once, a node of interest is chosen as the focal node and then several levels down from that node are drawn. This results in a visualization of a subtree of interest rather than a visualization of the entire tree. The disadvantage of this is that it is harder for the user to get the overall picture of the tree. However, the advantage of this feature of FastRings is that, most often, the user is not interested in the entire tree, but instead, is interested in exploring a small section within the tree. Our visualization takes advantage of this feature to apply the visualization to large graph structures that would not be practical or even feasible to display the entire structure, such as the World Wide Web. This ability to incrementally explore infinite graph structures, along with the look of the visualization, inspired the naming of the visualization system as Perpetual Information Exploration Visualization, or PIEVis.

By allowing the user to navigate the focus to different nodes of interest, a way is provided to semantically zoom to the part of the graph he or she is most interested in bringing that part of the graph into focus and moving the rest of the graph out of focus. When a subtree is in focus, the root node of that subtree is drawn as the largest circle and each child of that root node is drawn as a circle inside of it. In this same manner, several levels of the tree are drawn recursively. The number of levels drawn can be specified by the user, but the visualization generally begins to get crowded at greater than three levels.

Once the graph has been transformed into a tree, and displayed using the FastRings algorithm, methods of recreating the graph must be applied to allow the user to retrieve the details he or she requires.

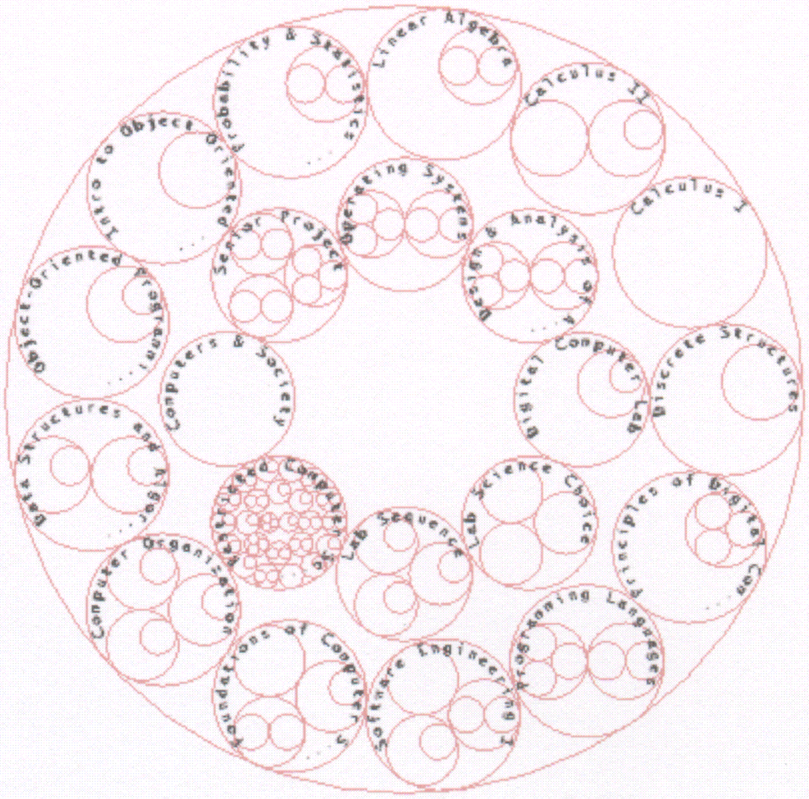

Figure 2.2 FastRings Tree Visualization of Rowan University's Computer Science Degree Requirements

2.3.1. Node and Subtree Highlighting

In order to reestablish the missing links within the graph drawn in PIEVis, a means of displaying the multiple connections that were lost, without introducing edges, is required. Highlighting common nodes when the user hovers provides a way to recreate multiple connections to a single node in an area of interest. When the user hovers the mouse over any node in the visualization, that node will be highlighted by filling it in with a transparent color. Likewise, every copy of that node within the current view will be highlighted (Figure 2.3). This highlighting recreates the lost connections of that node of interest without drawing any edges, and therefore introducing no edge crossings.

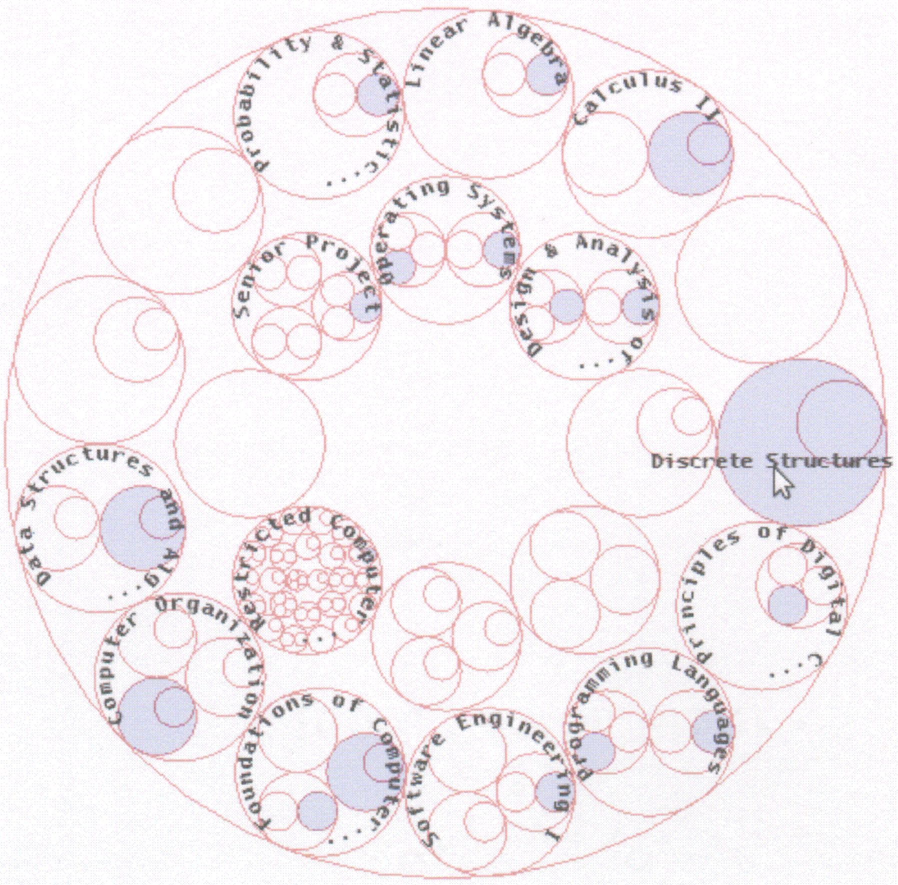

Figure 2.3 Node highlighting: when the user hovers over a node, each copy of that node is highlighted, and labels of any subtree not containing the node are removed.

Although highlighting can help the user to recreate the multiple connections to the node, it does not always allow the user to see where these multiple connections actually exist. When a user highlights a node that is several levels deep within the current visualization, although his primary interest would be the highlighted node, the immediate subtree of that node is also of interest. Consider the example that *B* is a child of *A* and *A* is a child of both *C* and *D* (Figure 2.4). When the graph is transformed into a tree, *A* would be duplicated to be a child of both *C* and *D*. Now, if the user hovers over the *B* node that is inside the *A_C* node, the user is likely interested in *A*, *B*, and *C*, though the primary interest is on *B*. The *B* node will be highlighted inside of *A_C* and *A_D*, but the user will have no way of telling that both *A_C* and *A_D* are actually the same *A* node.

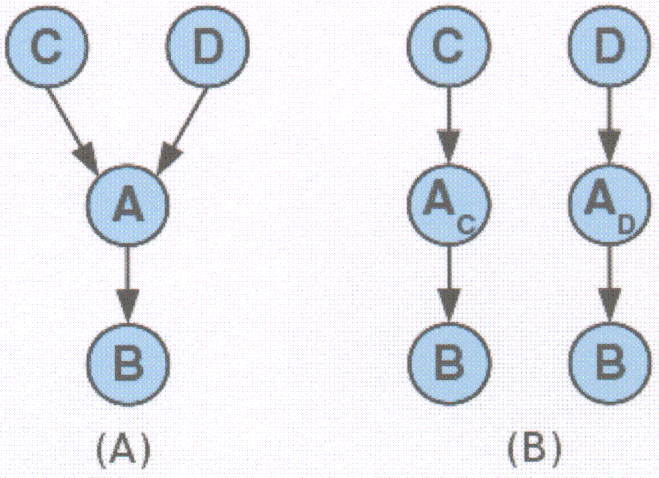

Figure 2.4 (A) The original graph as described. (B) The graph transformed into a tree: using only node highlighting, the connections of *C* and *D* to *B* will not be correctly portrayed as through a common node *A*.

The immediate subtree is defined as the entire subtree, beginning at level 1, of the currently highlighted node. If the highlighted node is at level 1 in the current view, then the immediate subtree is only the node itself. Otherwise it is the level 1 node that the highlighted node is inside, and recursively each node below level 1 that the highlighted node is inside. In the example in the previous paragraph, the immediate subtree of the *B* node inside the *A_C* node would be *C-A-B*. In order to show the connections of these interesting nodes each level is highlighted a darker color as the level gets closer to the hovered node (Figure 2.5). Each copy of the highlighted subtree nodes are also highlighted with the same color. Highlighted subtrees are drawn with a thick line of a distinguishable color, as opposed to highlighted nodes which are filled with a translucent color.

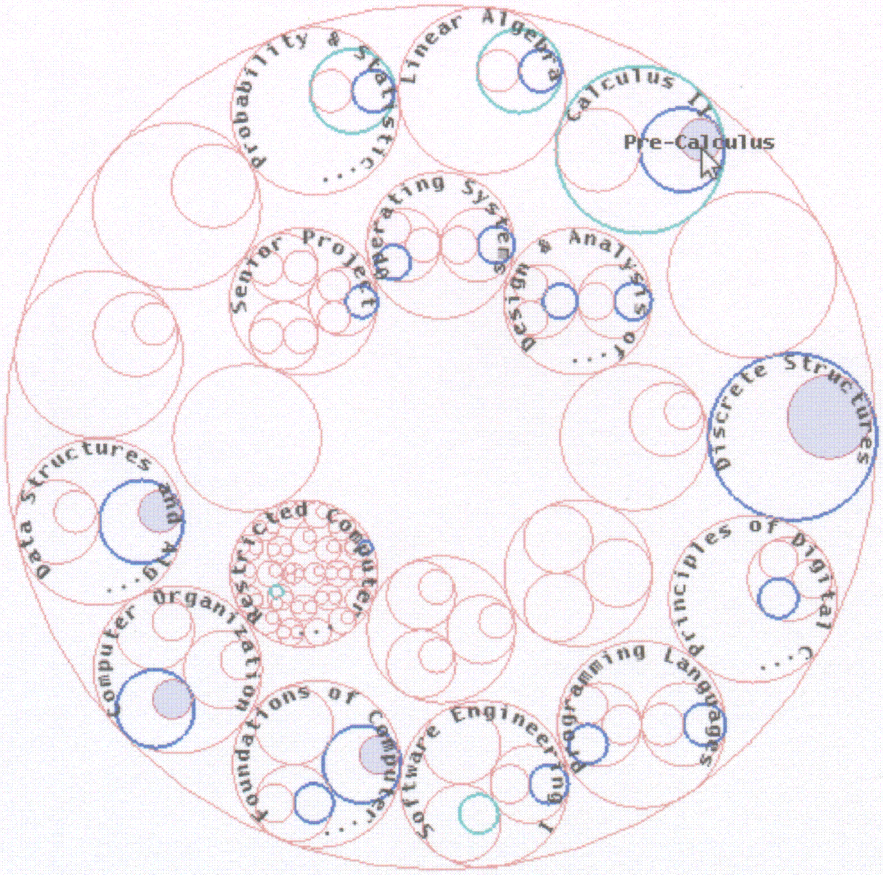

Figure 2.5 Subtree marking: when the user hovers over a deep node, the subtrees containing that node are marked.

Subtree highlighting along with node highlighting allows the user to see multiple connections and the level at which each connection is made, without introducing edges. In turn, much of the data that was lost is now regained. Although this does not recreate every connection to the node of interest, it attempts to recreate all of the connections that are interesting to the user. Any connections to the node that are from parents not currently being drawn in the visualization, will not be drawn when the user hovers. In most cases, a parent node not currently being drawn in the visualization means it is a connection the user is not currently interested.

2.4. Parent Display

The FastRings algorithm creates a drawing intended for incremental exploration, so there is some data that may be interesting to the user that is lost in the visualization. If the user navigates to a node of interest, with the traditional FastRings algorithm only the subtree of the focal node is visible. This means only the children of that node, as well as its children's children and so forth, are displayed. However, the user may also be interested in the parents of the focal node. In a traditional graph visualization, the user would be able to retrieve parent information by backtracking the directed edges coming into the node he is interested in. When the graph is converted into a tree this information is lost unless the entire tree is displayed. Since the FastRings algorithm does not draw the entire tree, parent information of the focal node is lost.

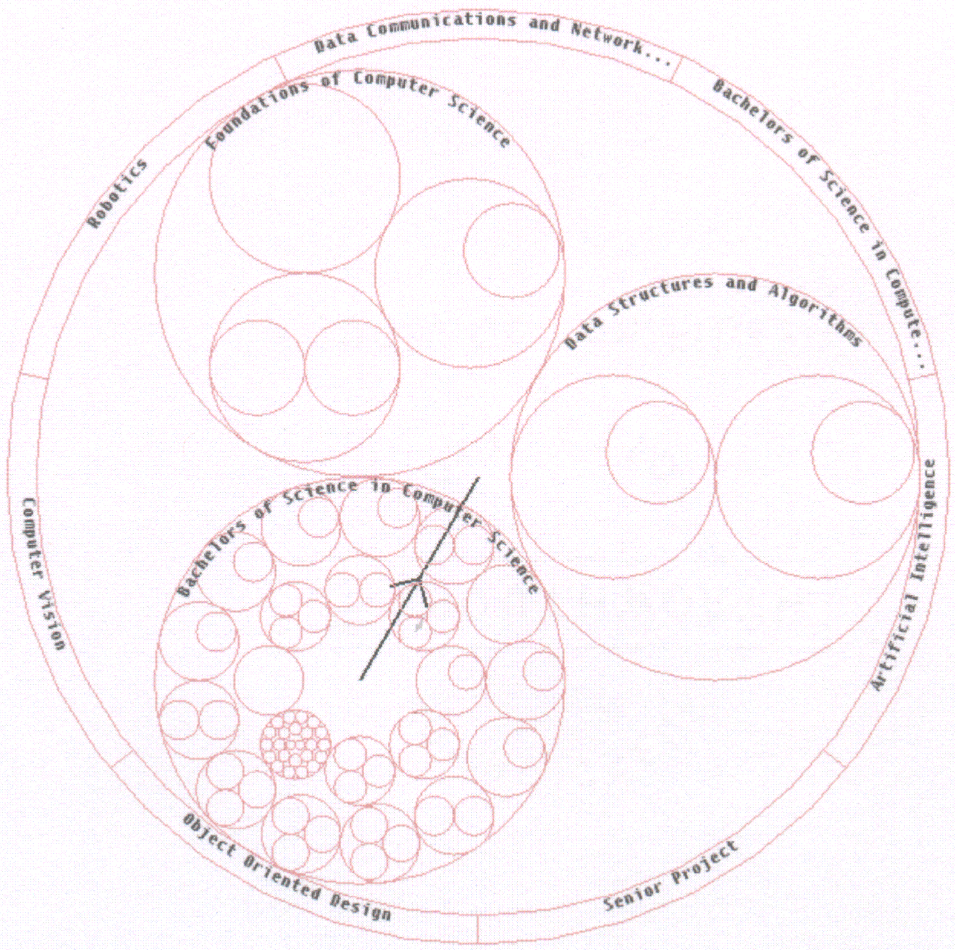
Data Communications and Network...
Bachelors of Science in Compute...
Foundations of Computer Science
Robotics
Data Structures and Algorithms
Bachelors of Science in Computer Science
Computer Vision
Artificial Intelligence
Object Oriented Design
Senior Project

Figure 2.6 Parents of the focal node are displayed around the outer edge of the node, in a format similar to a pie crust.

In order to address this issue, the FastRings algorithm is applied to the children and the resulting image is extended to display the parents of the focal node. The parents of the node are displayed in a crust around the outer edge of the largest circular node (Figure 2.6) similar to the crust of a pie. This provides a visualization of one level up in the graph from the current focal node. Combined with the traditional FastRings, the visualization now displays one level up and several levels down from a focal node helping to recreate the graph without introducing edges.

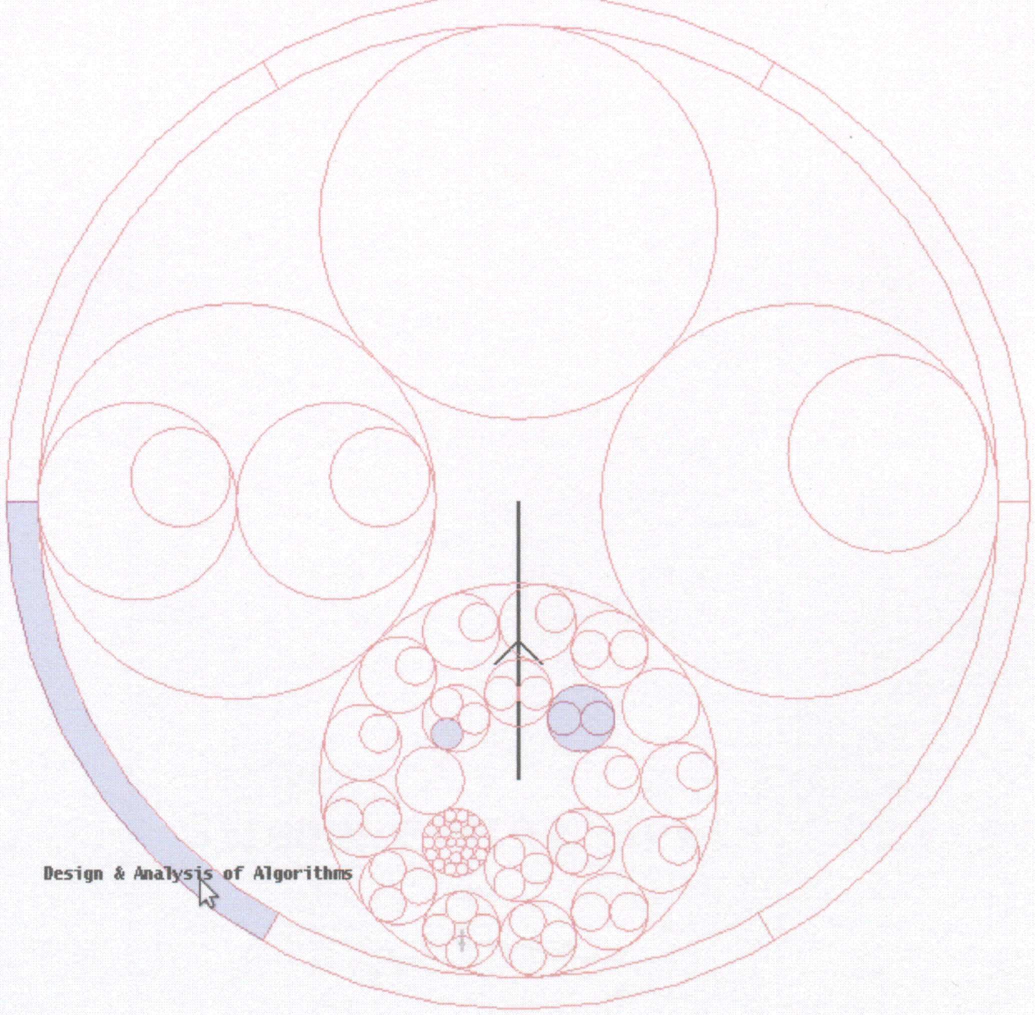

Figure 2.7 Parents can be interacted with just as children can. When a parent is hovered over, all duplicates of that node will be highlighted.

The user may interact with parent nodes in the same manner as child nodes, and node and subtree highlighting are also applied to parent nodes. Hovering over a parent will highlight the respective node in all other parts of the visualization (Figure 2.7), and hovering over a child will cause connected parent nodes to be highlighted appropriately (Figure 2.8).

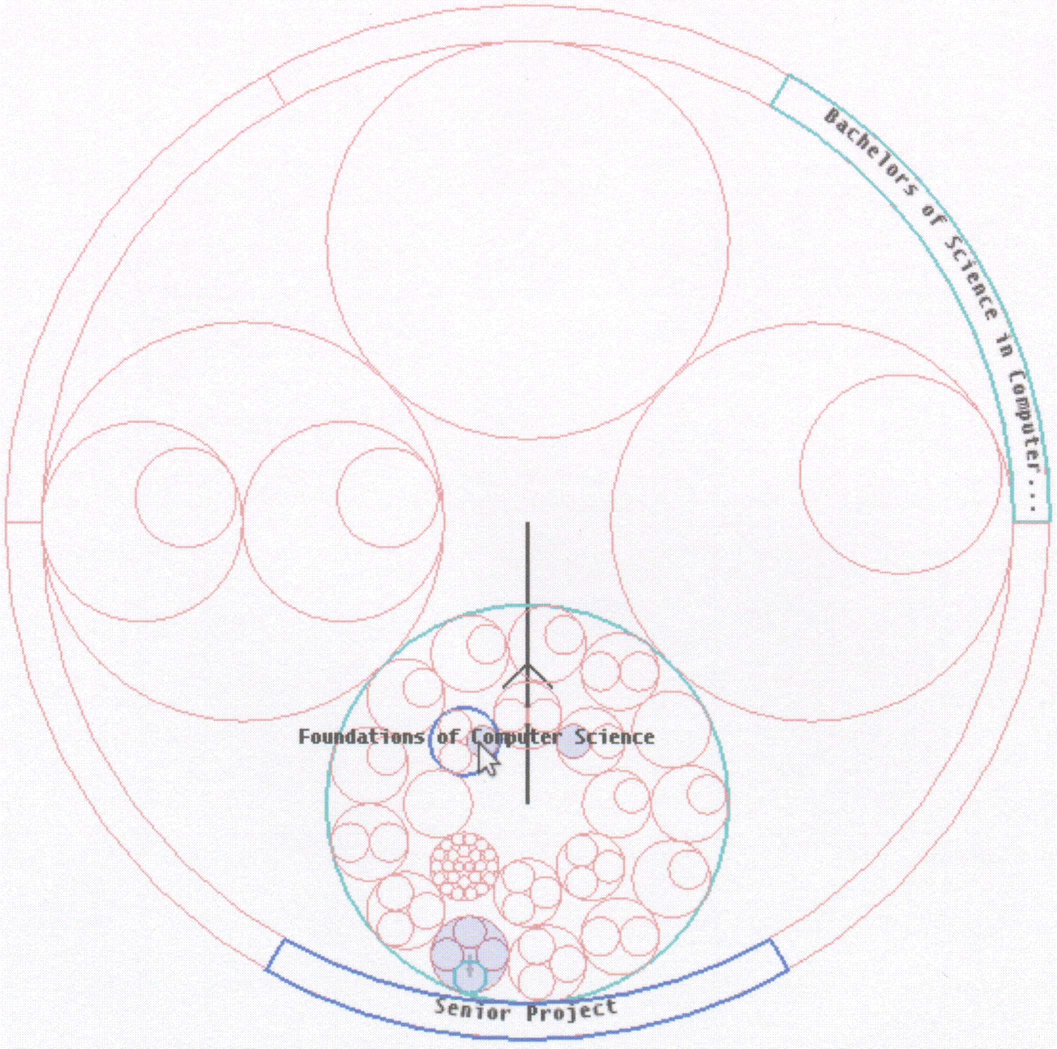
Bachelors of Science in Computer...
Foundations of Computer Science
Senior Project

Figure 2.8 Parents will be marked and highlighted when the user hovers over a child node, just as children are.

2.5. Degree of Interest

As discussed, the PIEVis system uses a feature called incremental exploration that allows the visualization to display a subset of the graph drawn from the focal point of a single node as chosen by the user. Much of the graph data is abstracted into tree data to avoid unwanted issues such as edge crossing and overcrowding. However, the graph data can be reconstructed using the interactive node and subtree highlighting and marking features that are provided. These features are an important part of the PIEVis system and combined create a feature called the Degree of Interest (DOI).

DOI is a method of suppression in which each node in the graph is applied a value of interest, and any node not over a threshold value is not displayed. In our visualization, several thresholds are used to decide the level of detail that is displayed. The first threshold is used to decide whether or not a child node should be displayed. This value can be set by the user, and decides how many levels down from the current focal node are displayed. The second threshold value tells whether a parent should be displayed or not and is currently hard-coded to one level above the focal node. Finally, there is a threshold that decides whether to display the details of the connections of a particular node. This threshold is concerned with the node highlighting and marking.

Conventionally, DOI is a numerical value assigned to each node in a graph, and there are several graph visualizations that use this DOI function to decide the level of detail of a node to display [14]. Although our visualization does not use a conventional DOI function, the results are similar.

2.6. Labeling

Since the nodes are represented as circles, a circular labeling method was implemented (Figure 2.9). These circular labels provide a way to fit as much text as possible inside the space provided by each node. Since only the first level of nodes and the parent nodes are labeled, and the label is contained completely inside the node, label overlap is impossible. The circular labels are easy to read and it can easily be determined which label is affiliated with which node.

Although the circular labels provide more room for text than straight labels would, with no overlap, and they provide an aesthetically pleasing means of labeling, it is preferable to limit the child node labels to using 270° of the circle. This is to limit the amount of the label that is displayed upside-down to a minimum and to help distinguish the start of a label from the end. Often, the label for a node is longer than that which can fit within the 270° limitation. In order to display entire labels, an interaction was implemented in which when the user hovers over a node the label of the hovered node is extended straight and the entire label is displayed. In this same way, labels for deeper level nodes can also be provided. The label shown when hovered is still connected to the node and follows the mouse cursor around so as to place the label directly where the user's attention is focused.

In order to not overcrowd the visualization when hovering over a node the circular labels of the nodes that are not connected to the hovered node are removed (Figure 2.10). Any node that does not contain a visible connection to the hovered node, either directly or through subtrees, will have its label temporarily removed. This causes the maximum amount of information about its connected nodes to be kept, but the visualization remains clean and aesthetically pleasing while the user's attention is drawn to the areas he or she is likely most interested in.

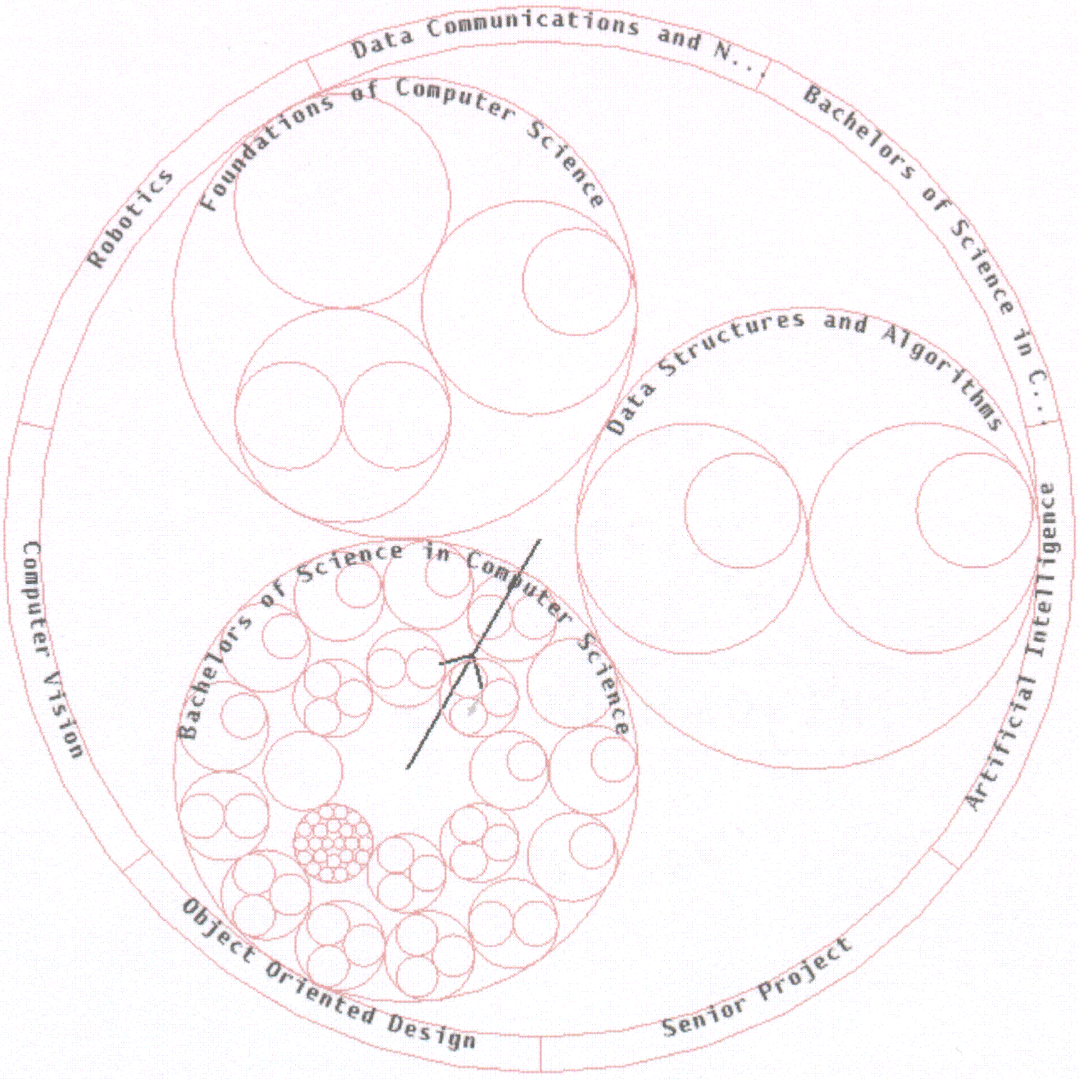
Data Communications and N...
Bachelors of Science in C...
Foundations of Computer Science
Robotics
Data Structures and Algorithms
Artificial Intelligence
Computer Vision
Bachelors of Science in Computer Science
Object Oriented Design
Senior Project

Figure 2.9 Circular labels are drawn inside each level 1 node, as well as inside each parent node along the crust.

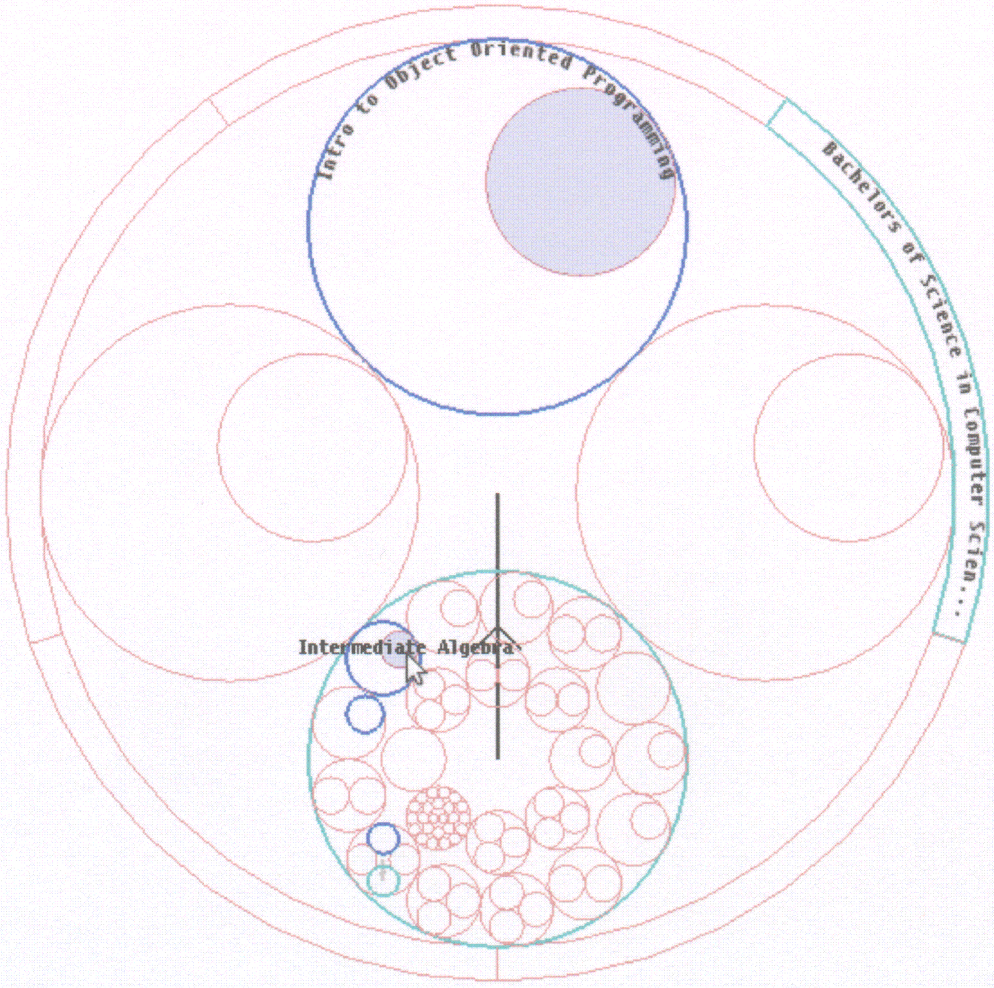
Intro to Object Oriented Programming
Bachelors of Science in Computer Scien...
Intermediate Algebra

Figure 2.10 When the user hovers over a node, labels of the level 1 nodes, and parent nodes, that are connected in some way to the hovered node, are not removed.

2.7. Navigation

As mentioned in previous sections, this graph visualization relies heavily on user interaction to display useful information. The user is provided several different forms of interaction with the visualization which can help to retrieve more data about an area of interest. One of the most important forms of interaction is the ability to navigate through the levels and nodes of the graph.

The visualization only displays a subtree and rarely fits the entire tree into the visualization, so navigation is required in order to gather information about the graph. The user can navigate to a node of interest by clicking it. When this action is performed, the visualization begins a smooth animation to move the focus to the clicked node (Figure 2.11 and Figure 2.12). The previous focal node scales down, and the new focal node scales up, as they move to their new respective locations. Once the animation is completed, the previous focal node is displayed within the new focal node connected by an arrow to help the user to maintain a navigational history.

This process is presented as a smooth animation in order to help the user remain oriented within the graph. The transition of the focus from one node to another can easily be followed through the animation. However, the animation is quick, lasting just over one second. This quickness allows the user to continue using the visualization with little delay while shifting the focus, but it is slow enough for the user to follow and retain a mental map.

Navigation, along with the hover interaction, allows the user to gather information, locate areas of interest, and semantically zoom to gather more information about an interesting area. PIEVis supports events that may be triggered by actions such as navigation. These events can cause more detailed information to be displayed separate from the visualization and can differ depending on the content of the visualization. A few examples of events that may occur are opening a file, displaying course information, or browsing to a Web page. A Web example will be discussed in a following section.

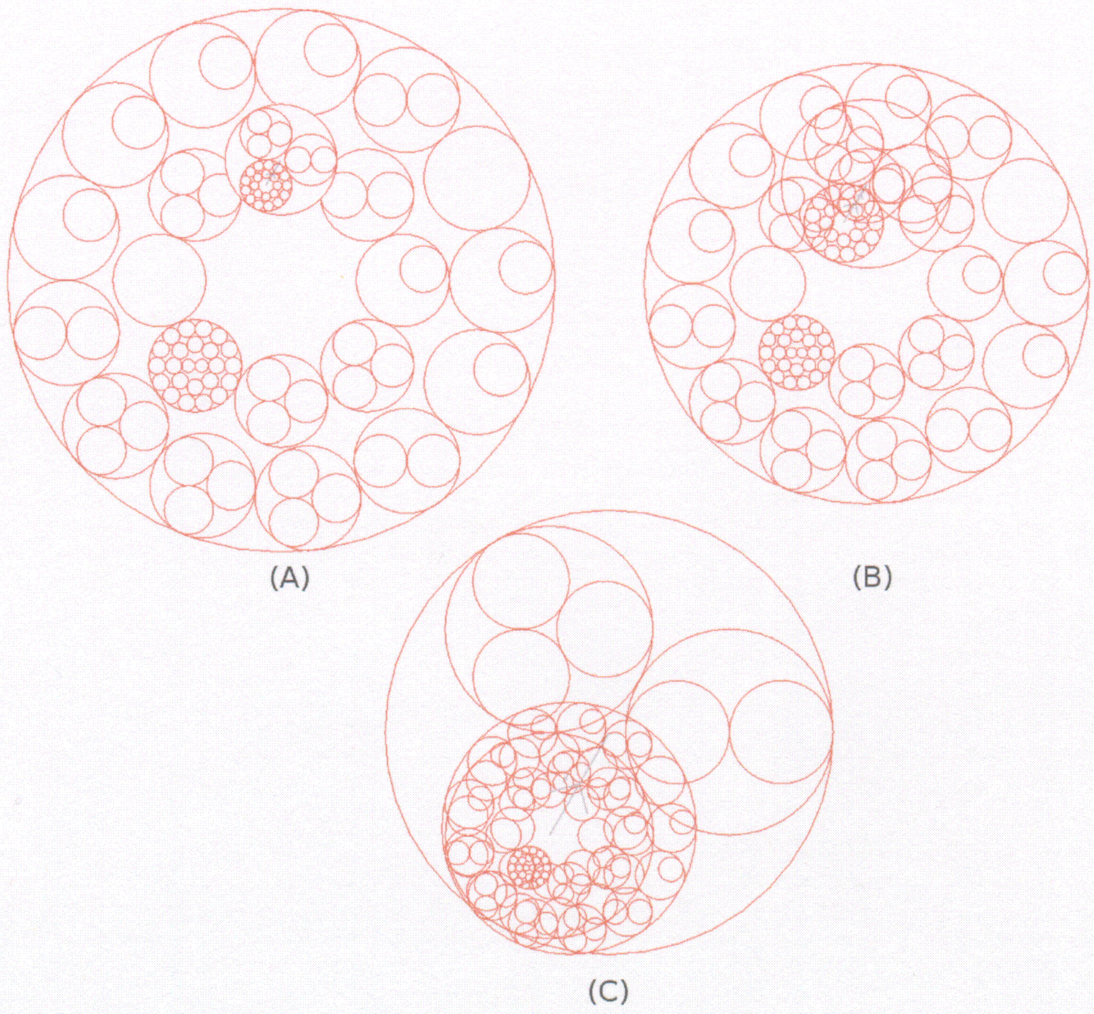

Figure 2.11 (A) As a node is clicked, it quickly pulses to draw the user's attention. (B) The clicked node is gradually scaled up and moved toward the center, while the previous node is scaled down. (C) Once resized, the previous node begins to move toward its new location within the new focal node.

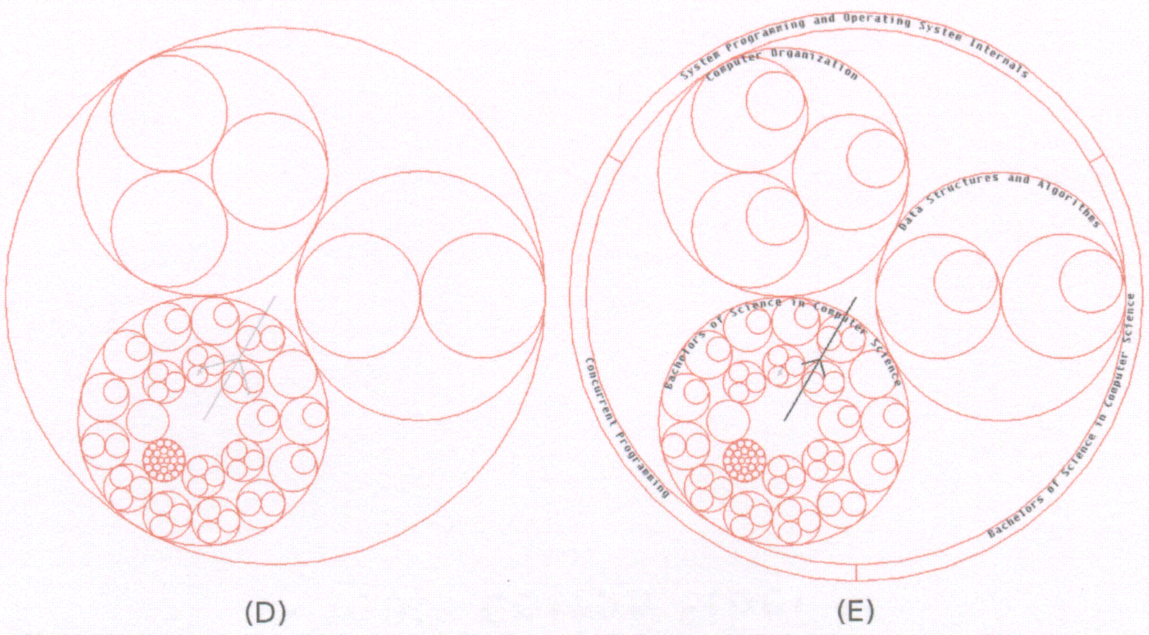
System Programming and Operating System Internals
Computer Organization
Data Structures and Algorithms
Bachelors of Science in Computer Science
Concurrent Programming
Bachelors of Science in Computer Science
(D)
(E)

Figure 2.12 (D) An arrow is drawn from the previous node showing the direction the user navigated. (E) Finally, the animation ends, and the correct level of detail is displayed.

2.8. Program Re-Design

An important part of any information visualization application is the way the program is designed. It is a well accepted practice to attempt to separate the data from the visualization so each can change separately without affecting the other. The original visualization application was called WebGPS. This application was designed to display web data using the FastRings visualization. When the program and the source code was inherited, the initial idea was to extend this application to meet our needs. However, the source code was not designed to be extended. WebGPS successfully performed what it was implemented to do, but adding more features and extending it proved to be difficult.

In order to accomplish our goals, we decided it would be more efficient to redesign the entire system from scratch, only taking bits and pieces of code from the original application. Several months were spent fleshing out a robust, extendable design that would allow the accomplishment of all of our goals, as well as provide an organized, easily extendable interface, so we could add more features as needed with as little change as possible, and without causing other features to break. Another requirement of this new design was to clearly document everything.

Object-Oriented Design patterns were used extensively throughout the new design. There is no single design pattern that stands out as the main pattern, but many of them were combined to create a very robust design. Some of the patterns included are Singleton, Observer, Blackboard, Factory, Flyweight, Bridge, and Strategy. The design was split into five different modules for data gathering, data storage, visualization, interaction, and application execution. The modules were designed to have high cohesion (meaning each module has a very specific functionality and does not cross functionality with other modules) and very low coupling among other modules. In Figure 2.13, the design of the entire application is shown. We can see from the figure that there is very low coupling between classes as well as very low coupling between modules. In the following subsections we will discuss, in detail, the purpose and design of each module.

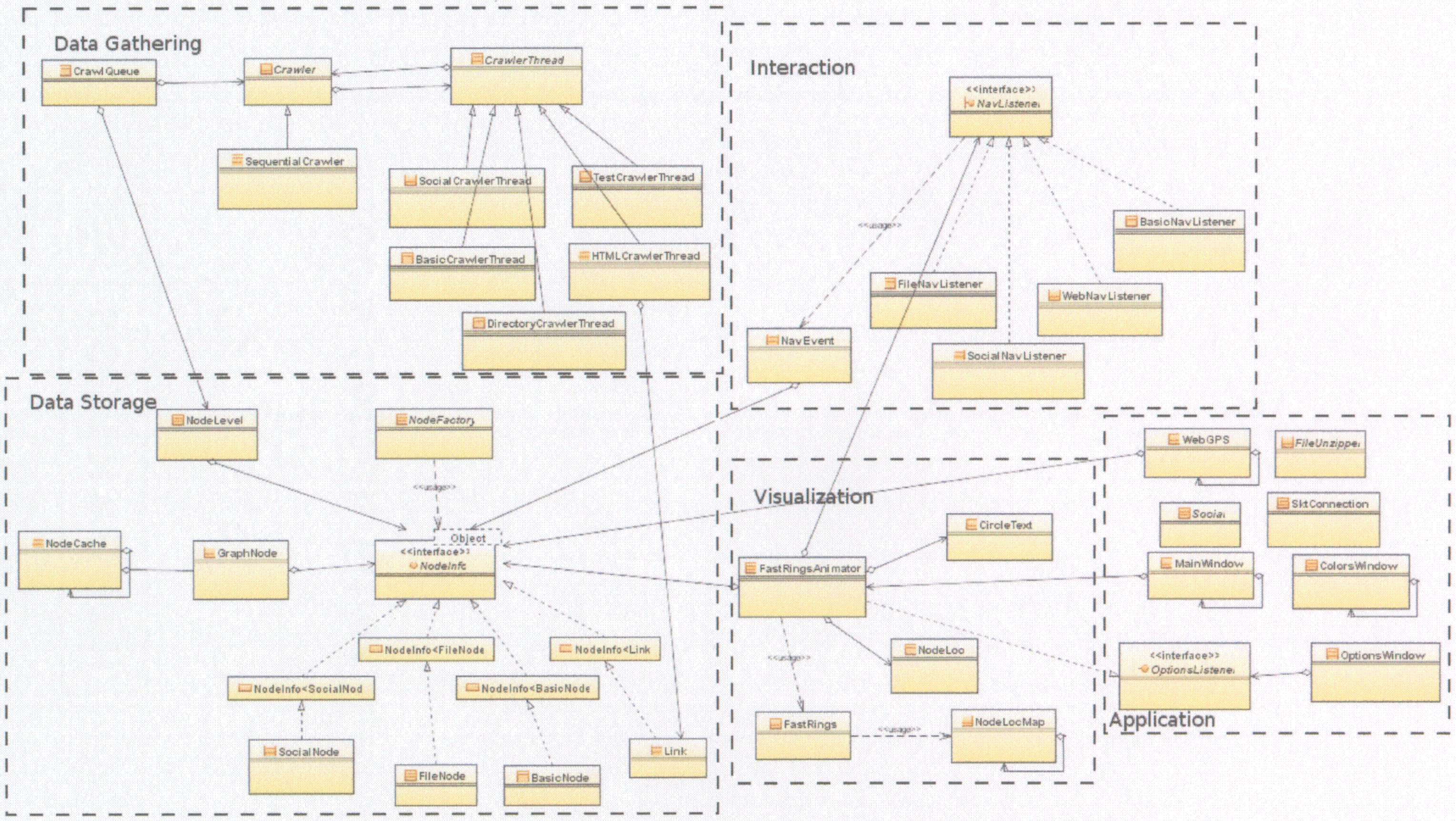

Figure 2.13 The overall design of the application. There is high cohesion and low coupling between modules.

2.8.1. Data Gathering

Data gathering is performed by an object we call a Crawler. Crawlers are agent-based [15] [16] objects, meaning that a single Crawler is independent of all other Crawlers, but there can be many Crawlers performing their jobs at the same time, without interfering with one another. Because of the way these Crawlers work, a faster computer can create many of them to gather data quicker, while slower computers can create only a few to conserve resources. All Crawlers work in this agent-based format, but each specific type of data requires its own type of Crawler. For instance, a Crawler designed for finding links on web pages works differently than a Crawler that is designed for finding files within a directory.

The Data Gathering module follows a design pattern called Bridge [17]. This pattern separates the abstraction (Crawler) from its implementation (CrawlerThread). By using this pattern the abstraction can vary independently from the implementation. In our design, the Crawler is the algorithm that decides where to go next within the graph. Once it decides where to go next, the CrawlerThread is used to read the data of the node and find all of its child nodes. Our Crawler's decision algorithm can vary without making any modifications to each of the CrawlerThreads. A new CrawlerThread is needed for each different type of data that is being crawled (web data, directory data, social network data, etc.). At the time this was written, only a sequential crawling algorithm existed. The sequential crawling algorithm is a first in, first out algorithm. It is our intent, in the future, to implement more powerful crawlers that take many parameters into consideration before deciding where to crawl next.

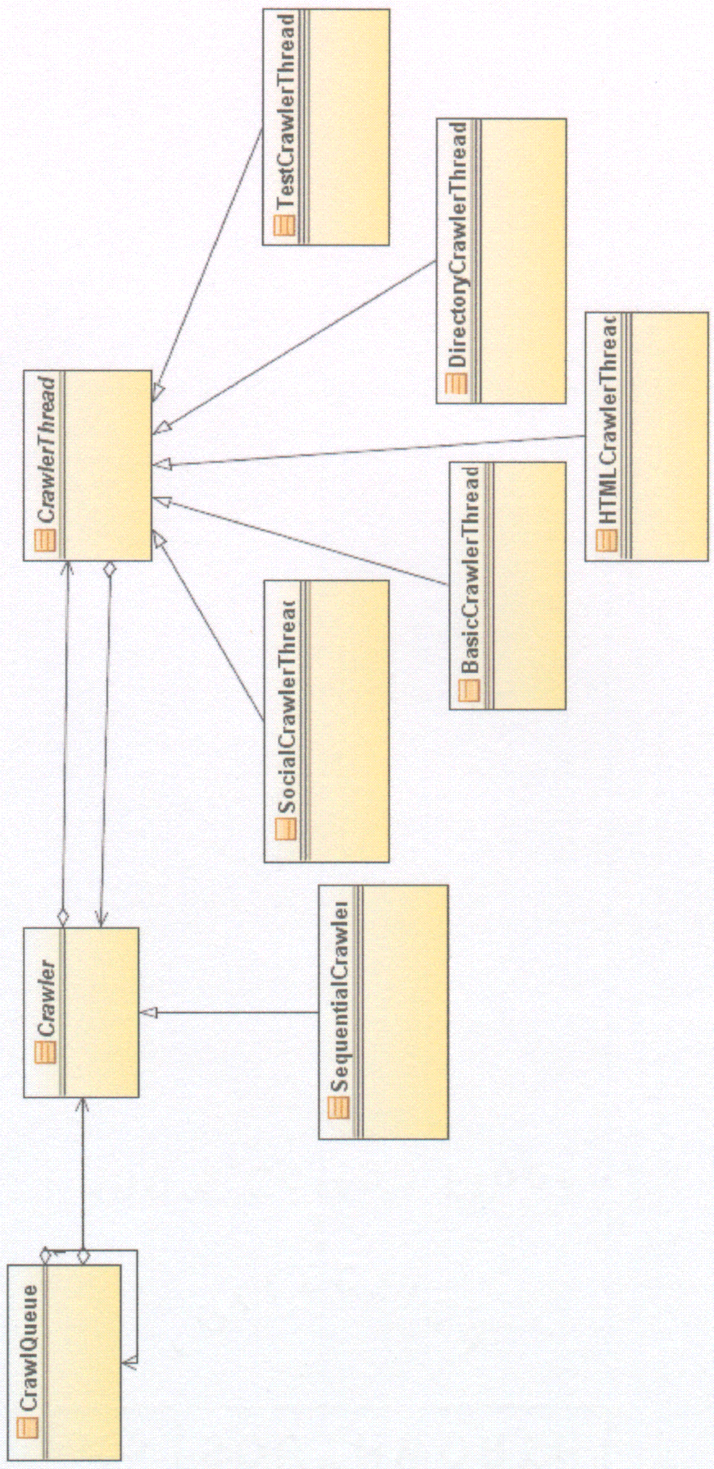

Figure 2.14 The class diagram of the Data Gathering Module. As in the overall design, there is low coupling between classes.

2.8.2. Data Storage

As the Crawlers gather the data, they store it in a cache. This cache stores a set of unique node objects. Each node stores links to its children and its parents, in order to allow the visualization to navigate through the graph quickly in any direction. As with Crawlers, each different type of data requires a type of Node object specifically for storing that data. For instance, a Node object for a degree requirements graph may require course descriptions within each node, while a Node object for a web visualization graph will require a hyperlink within the node. However, there are many similarities among all different types of Nodes (i.e. list of parents, list of children, title) so it is important that these items are abstracted to the general Node class.

This module combines several design patterns to accomplish its purpose. The Factory [18] pattern, a pattern used to manage the creation of objects, is used for creating each type of Node. Once the nodes are created, they are stored in an object called the NodeCache that uses the Blackboard [19] and Singleton [11] patterns. The NodeCache contains a single static reference to itself that all other objects can use to access the "blackboard." Using the static reference, the agent-based crawlers can add to the cache without affecting one another while the visualization retrieves the data from the cache.

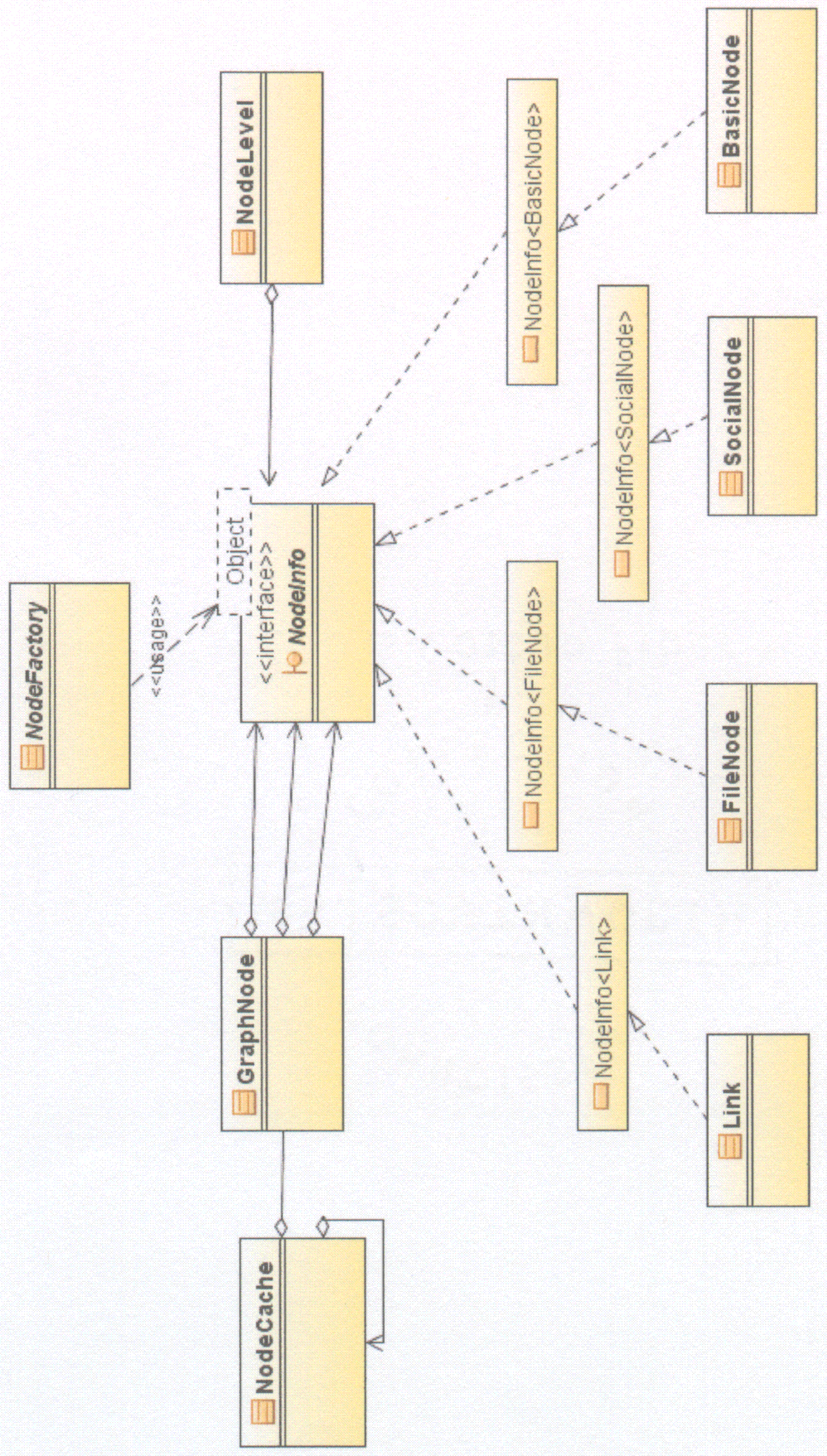

Figure 2.15 The class diagram of the Data Storage module. Once again, the design is characterized by high cohesion and low coupling.

2.8.3. Visualization

It is common practice to separate the visualization from the data in an information visualization application. Our visualization simply accesses the cache of Nodes: first retrieving the focus node, and then retrieving each child and parent as needed. Interactions that are involved solely with the visualization such as highlighting and focus navigation, are handled by the visualization itself. However, the visualization also carries event handlers that can cause more complex processes to occur through interacting with the visualization.

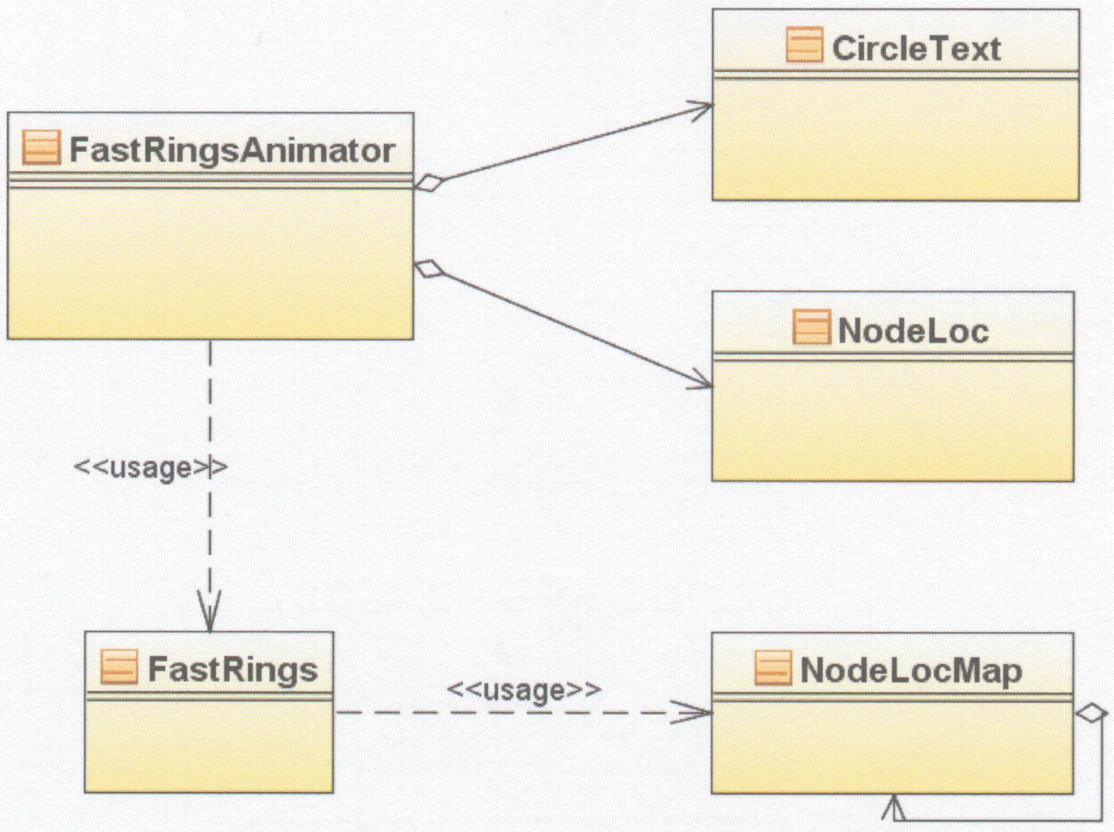

Figure 2.16 The class diagram of the visualization module. Visualization algorithms are separated from the actual display class.

2.8.4. Interaction

Events occur whenever a user performs specific actions within the visualization, such as hovering over a node or clicking on a node. These events are passed to each event handler for it to deal with them as it needs. Multiple event handlers can be added into the visualization at once, but typically only one event handler is used that is specific to the type of data being displayed.

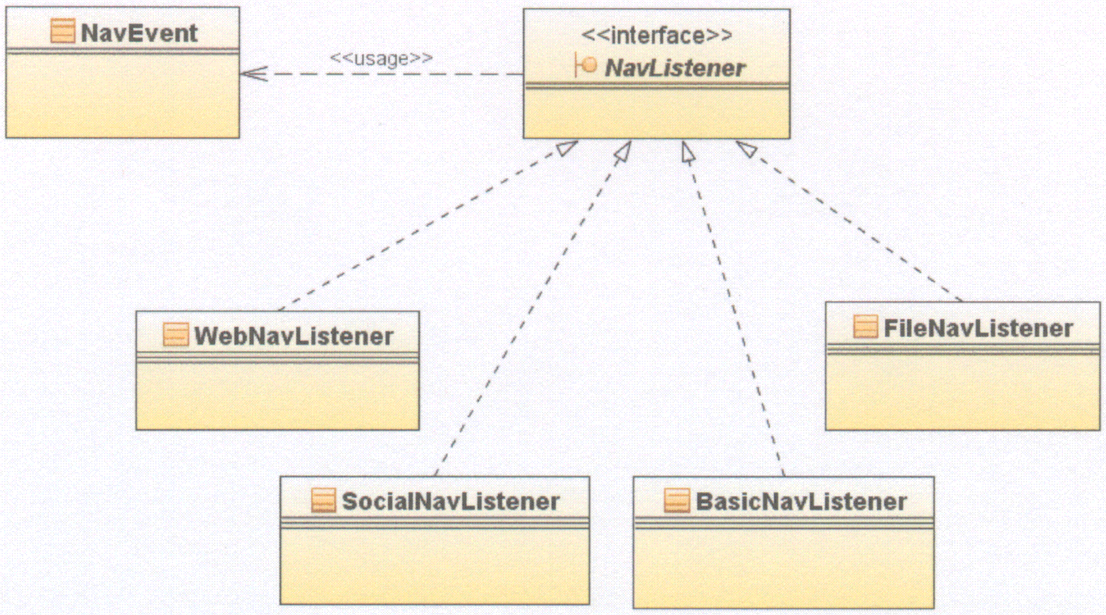

Figure 2.17 The class diagram of the interaction module. This module follows the popular Observer pattern.

PIEVis' Interaction module uses the popular Observer design pattern, similar to the design used by the Java API. The way this pattern works in the case of our application is that the visualization object is an "observable" and any number of "observers" can be added to it. The observers are watching the observable for certain events to occur. When one of these events occurs, such as a node is clicked, the observable notifies every observer of the event and passes along information about the event. For instance, in the case of a web visualization, where each node represents a hyperlink on a web page, when the user clicks a node, the observable would notify an observer of which node was clicked. The observer would then retrieve the URL from the node and connect to that URL in a browser window. Meanwhile, the visualization would navigate to the new focus node, and another observer would trigger the crawler agents to begin crawling the new focus node.

2.9. Example Usage: Web Visualization

The PIEVis system has been applied to an interactive World Wide Web (WWW) visualization called WebGPS [12]. WebGPS used the traditional FastRings algorithm to display the Web data in a hierarchical tree visualization. FastRings was chosen, instead of the traditional RINGS, because as a top-down algorithm it allowed the visualization to display the data for a particular Web page as soon as the Web page was crawled. RINGS is a bottom up algorithm, and so would require the entire Web structure to be crawled before any data can be displayed, which would be slow and impractical.

However, one drawback of displaying Web data using FastRings was that the algorithm was developed for displaying tree data. The WWW is a directed graph, and therefore, cannot be represented as a tree without loss of information. This drawback triggered review of other graph drawing algorithms for use in displaying the WWW. Many graph visualizations, however, are designed for displaying the entire graph at once and do not work well with information being added on the fly. Although a few graph visualizations do apply incremental exploration methods [20] [21] for visualizing the Web, the advantage of using FastRings is that more information can be added to the graph as it is needed and as it is found, without changing the part of the visualization that is already displayed. The result of our research was an extension to the FastRings visualization that keeps all of FastRings' advantages while overcoming many of its disadvantages.

The improved WebGPS system can be used to navigate and visualize the WWW in ways that are not possible with a traditional web browser. However, by connecting it to a traditional Web browser, through the Interaction Module as discussed in section 2.8.4, the user can be provided all the familiar capabilities of a traditional browser while more capabilities are added through the WebGPS visualization (Figure 2.18).

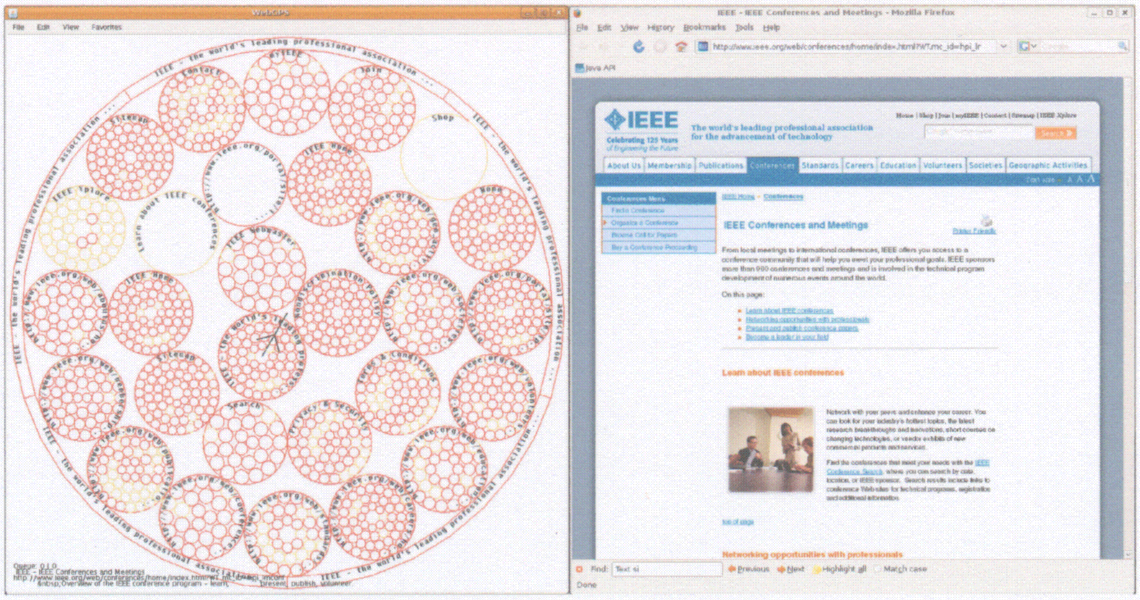
WebGPS
IEEE - IEEE Conferences and Meetings - Mozilla Firefox
IEEE
IEEE Conferences and Meetings

Figure 2.18 The WebGPS application using PIEVis to display IEEE's webpage.

Each node in the WebGPS visualization represents a link to a Web page. The focal node is the Web page the user is currently browsing, and is the same Web page that is displayed in the browser window. Each level 1 node in the visualization would then represent a hyperlink that is on the Web page the user is currently browsing. Each level 2 node represents a hyperlink that is on the Web page that the level 1 node represents, and so forth. In this way, the user is able to look forward in the Web and skip over Web pages if an interesting page is found several levels deep within the visualization.

Another feature of WebGPS, that greatly surpasses the capabilities of a traditional browser, is to display the parent nodes of a Web page. In a traditional Web browser, the user is able to see all of the children of a Web page represented as hyperlinks within the body of the page. However, there is no way, using any traditional Web browser, to locate the parents of a Web page. The parents of a Web page are the pages that contain links to the current page. This information can be important and useful to a user. Many search engines, including Google [22], use the number of links to a Web page in their search rating. There are several tools available for finding this information, however none package this information into a visual, interactive, browsable display like WebGPS.

When a user clicks a node within the visualization, along with the visualization navigating focus to that new node, the browser window will load the Web page that is represented by that node. The user can perform this action with a node at any level, navigating one or many levels, and even navigating up to a parent node.

2.10. User Proposed Improvements

PIEVis was originally developed as an improvement and expansion of the

WebGPS application. When performing any upgrade to an existing system, or in this case a complete redesign, it is important to ensure that the new version has indeed improved upon the original and has not degraded in performance.

In the user study performed in [12], there were four major concerns raised by the users. All of these concerns have been addressed in PIEVis.

The first concern was about the edges connecting the nodes. These edges were found to be confusing and cluttering and did not really add anything to the visualization, since the relationships between nodes are conveyed by the placement of the nodes in the visualization. PIEVis has removed these edges altogether.

The second concern was the color and positioning of the labels. The coloring of the labels was not appealing and the positions of them did not always immediately convey to the user the node it was affiliated with. The previous labeling method was replaced with a new circular labeling method that allows the entire label to be placed inside the node, causing no confusion as to which node it is affiliated with. Furthermore, colors of all objects in the visualization, including the labels, are fully customizable by the user, so they can be set according to a user's preference.

The next concern was about the speed of the animation. WebGPS' navigation animation took approximately five seconds to complete from when the user initially clicks on a node to when the visualization is ready to be navigated again. Another part of this concern was the fact that the browser would not load the web page until the animation had completed. So the user had to wait at least five seconds before he or she could even begin looking at the information of interest. PIEVis has replaced the old animation with a much smoother and faster one, taking just over a second to when the user can navigate again as described in section 2.7. Furthermore, the web page is loaded as soon as the user clicks a node so the user can immediately begin viewing the information while the animation completes.

The final concern was once again regarding speed. The users commented that the web crawling performed by WebGPS could use an improvement as could the speed that the visualization updated. In order to achieve this goal, the web crawler had to be completely redesigned, similar to the rest of the application, taking efficiency into consideration. To ensure this goal had been achieved, several experiments were performed, detailed in the following subsection.

2.10.1. Speed Efficiency Experiment

The web crawler designed for PIEVis was put through a number of tests against the original WebGPS crawler in order to examine its speed and efficiency. For consistency, the tests were all performed on the same computer. The computer is a custom built PC with an Intel Core 2 Duo 6420 at 2.13GHz with 4 GB of RAM and a NVIDIA GeForce 8800 GTS 512 MB display adapter running Windows XP SP3 32-bit.

The testing consisted of starting the crawler on a single Web page, which it then would recursively crawl to each link on that page. Several pages were chosen ranging from 17 to 60 links on the first page and 297 to 1240 links total among all child nodes.

The results of the speed test had a very wide range. The minimum speed difference was when crawling Rowan University's home page, which consisted of 59 links on the home page and 1240 links among children. The total crawl time for PIEVis was an average of 43.06 seconds out of 20 runs. WebGPS averaged 99.17 seconds across

20 runs. The maximum speed difference was also in favor of PIEVis. The Graphviz home page (http://www.graphviz.org) consists of 40 links on the home page and 525 links total. PIEVis resulted in an average crawl time of 4.86 seconds while WebGPS averaged a time of 52.88 seconds. Although there was a wide range of speed differences between the two systems, PIEVis always had a result at least twice as fast as WebGPS, proving that there was indeed a significant speed improvement in the new web crawler.

Another test was performed to determine the responsiveness of the visualization. An important part of the visualization is the highlighting that takes place when the user hovers over a node. This was found to be slow and unappealing in the original WebGPS and it needed to be improved for PIEVis. In order to test this, a large graph, consisting of 1240 total nodes being displayed, was loaded into each system. The mouse was then dragged across twenty-five nodes in a span of five seconds. The number of nodes highlighted when this action was performed was then recorded. PIEVis highlighted all twenty-five nodes immediately as the mouse was hovered over them no noticeable delay. WebGPS was able to highlight an average of six nodes that were hovered over, with a very noticeable delay. It is important to note that this delay was only noticed in the visualization itself. The system was able to recognize every node hovered over and displayed the name of the node in a text window immediately as it was hovered over.

The final results of the experiment were that PIEVis did indeed improve over the original WebGPS in its speed and responsiveness of both the visualization and the underlying algorithms.

2.11. Future Work: Application to Three Dimensions

One area that was given a lot of thought while this research was being performed, was extending the concepts into the third dimension. There are several graph visualization systems that use three dimensions in order to avoid edge crossings and provide more drawing area. These are often very powerful tools for visualizing the overall structure of a large graph, but due to the limited functionality of traditional input devices, many of these 3D visualizations can be difficult to navigate.

One solution to this, is to use natural head movement to navigate the graph. When looking at a three-dimensional image, even when drawn on a two-dimensional plane, it is natural for a user to tilt his or her head to try to get a better view from another angle, even if he or she cognitively knows it is simply a 2D projection of a 3D image. However, a virtual reality (VR) head tracking system can take these subtle head movements and translate them into rotation, zooming, and other navigations within the 3D visualization.

While most head tracking systems can be very expensive, an inexpensive alternative is being used to begin this research: the Wii remote head tracking system[1]. Using this system, any monitor can be used, and no expensive glasses are needed that create strain on the user's eyes. The browser window of WebGPS can be displayed just as it would traditionally while the visualization tracks the movement of the user's head and moves accordingly. Using this head tracking VR system and extending the concepts introduced in this chapter into three dimensions can create an interactive 3D map of the World Wide Web that immerses the user in a way that no traditional browser ever could.

---

[1] http://johnnylee.net/projects/wii/

2.12. Conclusion and Other Future Work

PIEVis is a novel, interactive graph visualization that has many useful applications. It is an incremental exploration system that uses a tree drawing algorithm and user interaction to display details of interesting areas of the graph. The tree drawing algorithm used allows for the removal of edges and, therefore, the elimination of edge crossing and angular resolution issues. Its circular format provides a desirable aspect ratio of 1, and the circular labeling method presented allows aesthetically pleasing labels to be displayed based on Degree of Interest without overlap.

One application of PIEVis to WWW visualization was presented. PIEVis has potential for many other applications, including social network visualization, database analysis, and network structure displays.

The PIEVis system is not without its drawbacks, however. Although the PIEVis system's incremental exploration feature allows large graphs, such as the WWW, to be displayed quickly and efficiently, it is limited in how many levels from the focal node it can display. The level of children does not have a hard limit, but most graphs become too crowded at more than three levels. Some users may desire to see much more of the graph structure than PIEVis allows for. For some large graphs, it may even be desirable to display the entire graph structure before focusing on a small area of interest.

Another weakness PIEVis currently has is the assumptions it makes in the data visualization. All crawlers currently implemented in PIEVis assume that the data they retrieve is correct. No error checking is performed currently, and although this may not be an issue in the Web Visualization example used in this thesis, errors in a visualization of a power grid or terrorist social network could be consequential. A useful visualization of networks of such high importance would require error testing implemented into PIEVis' crawlers to ensure the data being visualized is correct. This weakness will be part of some future work performed on PIEVis that will include displaying visualizations of networks in which accuracy is critical.

Along with error checking and the 3D visualization concepts, research will be devoted to connecting PIEVis with another graph visualization, such as Walrus [23], that provides a good view of the overall graph structure. Providing a multi-windowed visualization using focus and context should eliminate many of the drawbacks of the PIEVis system and create a powerful tool for graph visualization and analysis.

3. ENHANCED THREE-DIMENSIONAL STAR GLYPHS FOR MULTIVARIATE, MULTI-SET DATA ANALYSIS

3.1. Introduction

Representing data in different contexts can present a whole new meaning to the way it is interpreted. Many graphs have successfully been displayed in a two-dimensional environment. However, these graphs are limited to the amount of data that can be displayed at one time because of the constraints of physical viewing space a two-dimensional graph has. By adding in an extra axis, data is not as limited and there are virtually infinitely many new ways to display data.

A star glyph is a multivariate graphing technique in which each variable is represented by a ray, or "spoke," each of which extends out via a connecting line from a common origin with equal angular distance between each spoke. The length of each line is proportional to the magnitude of the variable compared to the maximum value of all the variables [24]. Star glyphs are effective at determining both where data begins to cluster and where the outliers are. However, the downfall occurs when the data set becomes increasingly large as the graph becomes overwhelmingly cluttered.

Many statistical graphing techniques are available to help overcome the clutter of large data sets. Scatter plots, parallel coordinates, and star plots all have done a good job at creating more modern techniques such as GeoTime [25], Jigsaw [26], NetLens [27] Substrate Designer [28], and [29].

This chapter[2] discusses a new way of graphing to visualize multiple sets of quantitative data in a three dimensional environment. The basic concept of a star glyph, a figure with *n* rays, where *n* is the dimensionality of the data set [30], is used and a second attribute is added to the end of the spoke represented in the form of a sphere. By adding a third axis the visualization can compare multiple sets of similar data. The result is a four dimensional multivariate plot, not limited to a two dimensional plane, useful for comparing similar grouped attributes across multiple data sets. As an example, these properties of an enhanced star glyph are applied to a visualization showing the accuracy of aircraft trajectory predictions across multiple systems used in the U.S. National Airspace System.

3.2. Enhanced Star Glyph

This section describes the concept of how an enhanced star glyph is extended from the basic two-dimensional star glyph into three-dimensional space.

3.2.1. Basic Star Glyph

A very basic star glyph extending out from a common origin in equal angular distances is presented in Figure 3.1. A star glyph has two dimensions: a quantifiable dimension and a categorical dimension. The quantifiable dimension is a single value and the magnitude of this value is represented by the length of a spoke. The quantifiable value is grouped by categorical data. For instance, suppose the star glyph in Figure 3.1 is showing the average amount of food a dog in a particular kennel will eat daily. Assume

[2] This chapter was published at the 13th International Conference on Information Visualization in collaboration with Adrian Rusu, Confesor Santiago, and Eric Thomas [37]

the data is grouped by the breed of the dog and *s4* represents all beagles, while *s2* represents all St. Bernards. It can be seen that, on average, a St. Bernard will consume much more food than a beagle and all other types of dogs, while a beagle will consume close to the least amount of food. This graph allows us to recognize an outlier either for exclusion, or for further analysis.

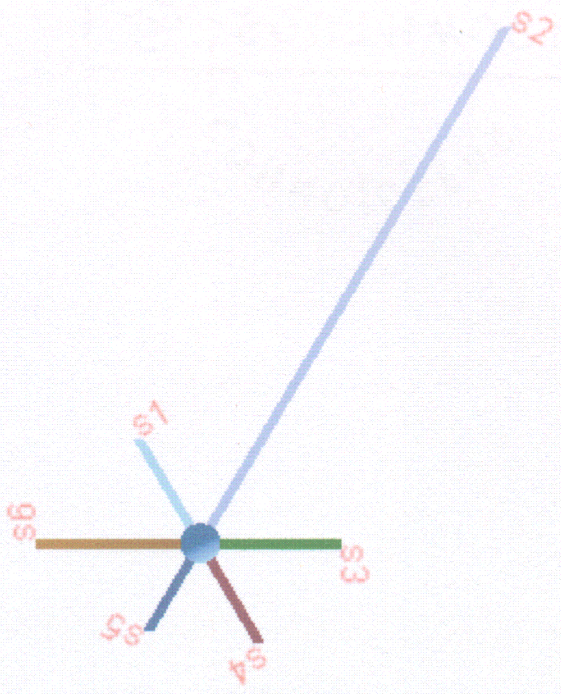

Figure 3.1 A basic star glyph

3.2.2. An Additional Attribute

Our visualization has the feature of adding another attribute to compare more data. This attribute can be any arbitrary value represented in the form of a sphere where the radius of the sphere is proportional to the magnitude of the attribute's value. Often the variable chosen for this sphere would be a weight so that the size of the sphere represents the significance of the data in that category.

Consider the graph in Figure 3.2 representing the average number of miles per day an athlete runs grouped by the sport he or she plays. Let the size of the sphere represent the average weight of the athletes. Now, assume that the large green sphere to the bottom right represents American football players while the bottom purple sphere represents cross country runners. It can be seen that the cross country runners, on average, run much longer than football players, however, football players are generally much heavier.

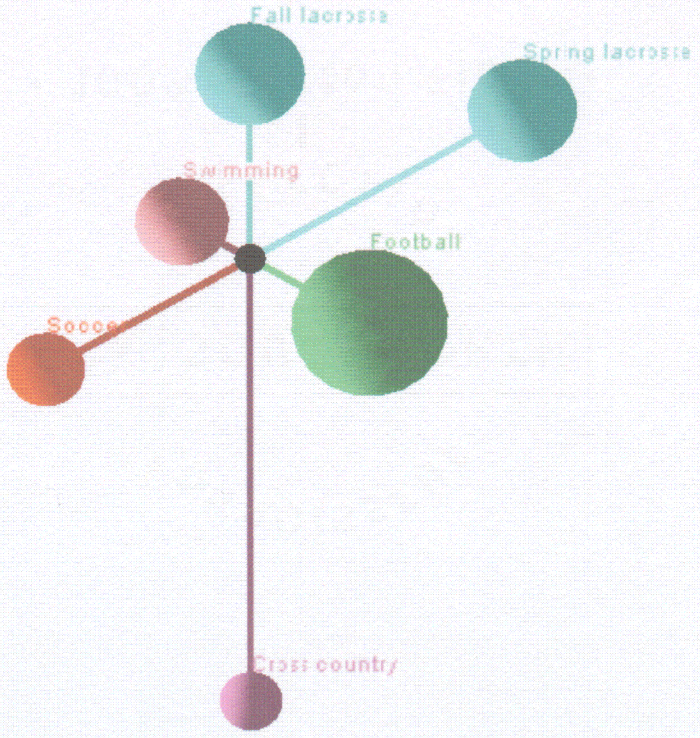

Figure 3.2 A star glyph with spheres

The addition of the sphere allows for an extra dimension to be added to the graph without physically requiring three-dimensions. Furthermore, the color of the spheres can put the data in a particular category. For example, in Figure 3.2 the two blue spheres on top and top-right represent lacrosse players. The sphere at the top represents lacrosse players in the fall, whereas the sphere on the top-right represents lacrosse players in the spring. From this graph it can be seen that lacrosse players run more in the spring since they are presumably more in shape by that time. Table 3.1 shows what a sample data set for the previous graph might look like.

Table 3.1 Data Chart for the Previous Graph

Sport	Miles	Weight	Color
American football	1.5	250	Green
Cross country	8	130	Purple
American soccer	4	160	Red
Swimming	0.5	140	Orange
Fall lacrosse	3	150	Blue
Spring lacrosse	5	150	Blue

3.2.3. Clustered Mean Graph

By adding in the spheres to represent another measurement, the clutter of a graph that has many points is increased. This creates a cluster graph as shown in Figure 3.3, making it difficult to analyze the groups inside the cluster, but easily identifying and analyzing most of the outliers of the graph. In order to view these outliers an option is given of graphing a clustered mean star glyph.

Wherever data is closely similar, the visualization clusters that data into a "blob" of spheres. It removes the labels for clustered data and leaves the labels for the outliers. The average data are always clustered in the center and the outliers extend out in a position that represents their absolute value from the average cluster. The following algorithm is used to determine whether or not data should be included in the cluster.

```
Algorithm Clustering
    Input: An input file containing the data points of the node.
    Output: A graph with nodes. Nodes in the graph are clustered if
    they are close to the average of the data points. Nodes that are not
    clustered are the outliers and their positions are the absolute values
    from the average of the nodes.
    for each node in list_of_nodes
        calculate avg of all nodes
    for each node in list_of_nodes
        if node is close to avg then
            add node to cluster
end Algorithm
```

Clustering data is increasingly more important in analysis especially in a topic such as data mining. By abstracting the details of the average, an analyst can focus only on the outliers. In data mining, an outlier could be a group of people that has not been marketed to or fraudulent data in a company's budget.

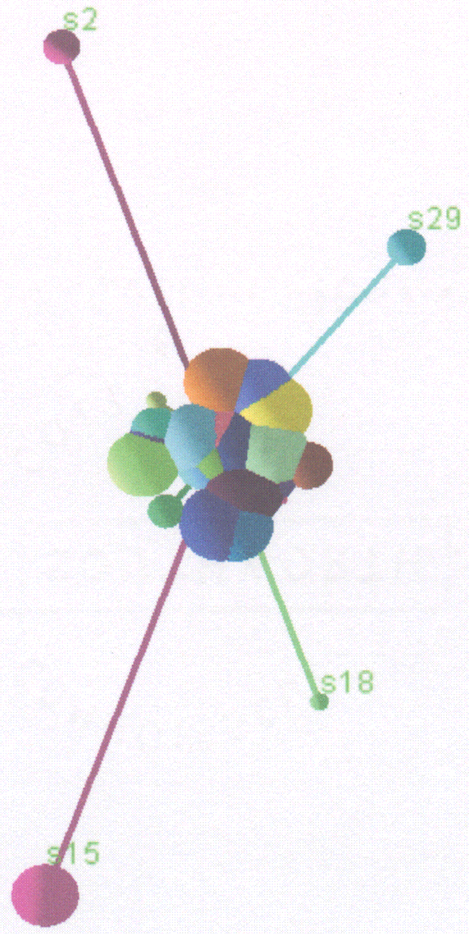

Figure 3.3 A clustered star glyph

3.2.4. Comparison of Multiple Star Glyphs

Finally, the last dimension is added into the graph. A single star glyph exists on a single two dimensional plane. Since this is a three dimensional graph, the third dimension can be used to display multiple star glyphs, allowing the comparison of several results to each other.

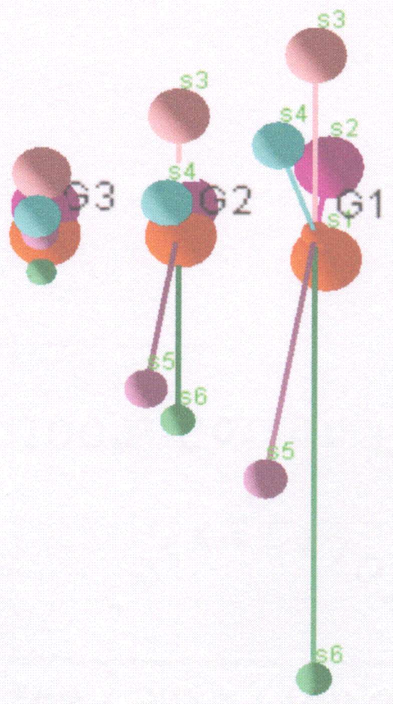

Figure 3.4 Multiple star glyphs

Consider that Figure 3.4 represents surveys taken by several different news agencies on which of six different companies produces the best ice cream. The distance of the spokes represents the average score of the ice cream producer (from 0 to 5, 5 being best), while the size of the sphere represents the amount of sales that producer had last year. Each separate glyph represents a separate survey. By displaying the three different surveys in the same visualization, it can be determined, among other things, which survey was more biased toward a particular producer.

3.2.5. User Interaction

Allowing the user to easily navigate the scene to focus on points of interest gives him invaluable insight into the information presented. One basic interaction is the ability to rotate the graph in any direction, allowing the user to get the perspective he or she desires (Figure 3.5), and to move about the graph to find the exact camera position required for analysis of a focal point.

Another interaction detail is allowing the user to manipulate the scales of the plot along each axis. This includes moving the planar star glyphs closer together for a more clustered appearance, or spreading them out more (Figure 3.6). Grouping the data in this way visually displays where data begins to cluster across the multiple data sets. This helps analysts discover which of the outliers of the data set are more distant from the other outliers.

Another necessity is the ability to change the scales of both the spokes and the spheres. This can make a point of interest less cluttered or easier to distinguish.

In many graphs, labeling of nodes can cause cluttering and often labels become indistinguishable. Giving the user the ability to display and hide labels of his choosing is a necessary provision. Each spoke, sphere, and planar glyph has its own label and, often many more details than can be displayed in the graph. For instance, each sphere may be labeled with what category it represents, but the user may be interested in the exact value the size of the sphere represents. By clicking on a particular sphere, detailed information about that category will be displayed, separate from the visualization. Using this feature, the user can view specific details for focal points to compare and analyze.

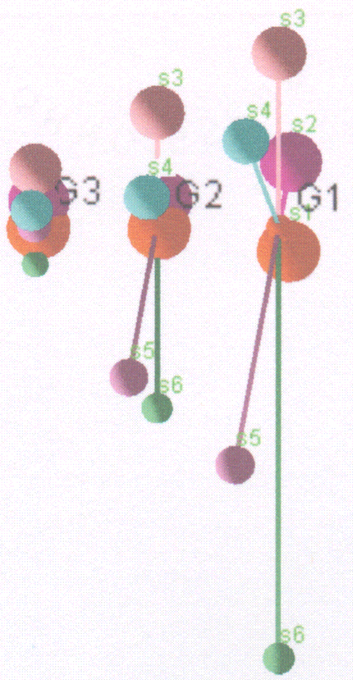

Figure 3.5 Multiple star glyphs displayed from a side angle.

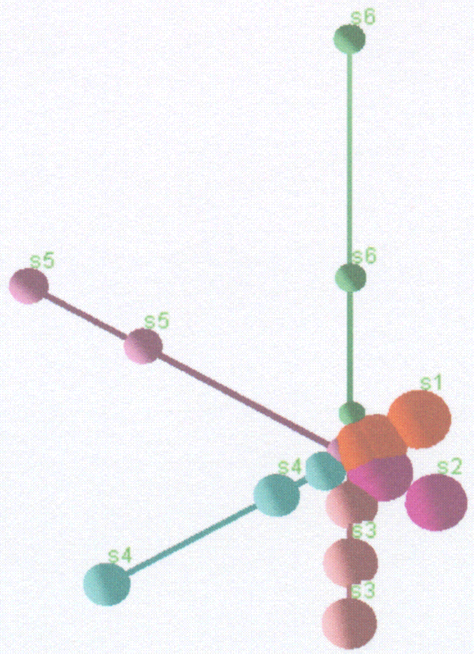

Figure 3.6 Multiple star glyphs displayed from the front angle, showing grouped data.

3.3. Application

The visualization techniques have been implemented in Java using JOGL, OpenGL bindings, to generate the 3D graphics. This section presents an application of the visualization.

3.3.1. Background

The U.S. Federal Aviation Administration (FAA) constantly strives for a safer, more efficient National Airspace System. It is predicted that by the year 2025, the current air traffic in the National Airspace System will have tripled what it is today. In order to handle this massive increase in air traffic density, many projects are under development as part of the FAA's Next Generation Air Transportation System. In this system, the air traffic controllers will use many Decision Support Tools to assist them in managing air traffic. An important element of these tools is a Trajectory Predictor (TP).

A TP assists the air traffic controller, by generating trajectories: four dimensional (latitude, longitude, altitude, time) predictions of where the aircraft will fly some time into the future. The trajectories are then processed by a Conflict Probe to predict when aircraft will come into conflict (i.e. loss of minimum separation standard; aircraft become too close together). A TP is not always accurate in predicting the future flight paths based on an array of factors (e.g. lack of aircraft intent, weather, etc.) Furthermore, the accuracy of trajectories generated by a TP determines its overall performance [31].

There are currently two major Air Traffic Control systems in which a TP is a defining element. The User Request Evaluation Tool (URET) is currently deployed in all air traffic control facilities across the United States. Extensive research has been performed on URET in order to validate its algorithms and evaluate its accuracy [31] [32]. This air traffic control system is sufficient for now, but will quickly become out of date as air traffic increases. The solution is a new system called En Route Automation Modernization (ERAM). Currently being developed by the Lockheed Martin Corporation, this Air Traffic Control system will be the replacement for URET and will be more powerful and more efficient. The accuracy of the TP in ERAM has recently been analyzed and results are present in [33]. Furthermore, the FAA has several in-house TPs used for regression analysis and evaluation of ERAM as it is being developed.

One main part of regression analysis is the comparison of a dataset to another reference dataset. The comparison of TPs with an enhanced star glyph allows analysts to quickly visualize strengths and weaknesses in each. This allows analysts and the ERAM developers to investigate where ERAM needs improvement, where other TPs are accurate, and how the current version of ERAM compares to a previous version.

3.3.2. Trajectory Predictor Comparison

For the example, six different TPs are compared. The two major ones (ERAM and URET) as well as four different FAA in-house TPs, those being Linear Predictor, Flight Plan, Hybrid, and Hybrid Merge [34]. The TPs have a selection of algorithms that make predictions of aircraft based on a number of factors that can include weather, air traffic, aircraft type, flight plans, etc. Our visualization graphically displays four different variable attributes for each TP. Each variable will be described so the reader is fully aware of how the strengths and weaknesses of a TP [35] are determined.

- *Look Ahead Times* are the time intervals in the future that a TP makes a prediction at. Generally, time in the U.S. National Airspace System is measured in seconds UTC that have past so far in a given day. So 12:00AM (midnight) is considered 0 seconds, whereas 12:00PM (noon) is considered 43,200 seconds. A look ahead time is how many seconds into the future a TP will predict the position of the aircraft. There are six look ahead times, used in these particular datasets: 0, 60, 300, 600, 900, and 1200 seconds. What this means is there is a total look ahead time of 1200 seconds (20 minutes) and the accuracy of that 1200 second look ahead time is analyzed at each listed time.
- *Engine Types* are large subsets that an aircraft falls under. The groups are based on the aircraft's engine which determines many things concerning the trajectory, including the top of ascent, speed, and distance an aircraft can fly. There are three main aircraft types: Piston(P), Turboprop(T), and Jet(J) [36].
- *Number of Measurements* determine the amount of data that is included in a subset. This can give an idea of how many flights are in the subset, how often a trajectory prediction was made, and even how long the flights in that subset were in the air. Table 3.2 displays eight rows of data. The second column refers to what engine type it is. Here it is shown that J stands for Jet, P for Piston, and not shown is T for Turboprop. The last column is a list of look ahead times for URET pertaining to each of those

aircraft types. The first column represents how many measurements were taken for a certain aircraft type and a certain look ahead time. For example, the first row indicates that in the database there are 4,737 measurements for jet aircraft types at a 0 seconds look ahead time. Row eight shows that there are 102 measurements for piston aircraft types at 60 seconds look ahead time. There are fewer measurements because there are less piston planes than jet planes in operation.

Table 3.2 Sample data set for URET

URET		
Count	**Engine Type**	**Look Ahead Time**
4737	J	0
4640	J	60
4258	J	300
3779	J	600
3297	J	900
2819	J	1200
100	P	0
102	P	60

- *Average Horizontal Error* is one of the most important values in comparing TPs. Horizontal error is the horizontal distance in nautical miles from the predicted location of the aircraft to the actual position of the aircraft at that predicted time. We take the average of all these values grouped by a single look ahead time and single engine type to generate one value in nautical miles for each grouping. Table 3.3 shows a selection of data for ERAM. The second column shows how close, on average, the

TP's prediction was to the actual value. A lower number indicates a higher level of accuracy. The first row shows that ERAM has made 202 predictions for aircrafts of piston type 600 seconds into the future. When those predictions are averaged together, the final result is 10.33. The rest of the data shows that ERAM is better at predicting turboprop aircraft than piston type aircraft. Also, as one might suspect, as the look ahead time grows larger, the average prediction becomes less accurate.

Table 3.3 Sample data set for ERAM

ERAM			
Count	**Avg. Horz. Error**	**Engine Type**	**Look Ahead Time**
202	10.3290	P	600
182	14.0511	P	900
162	17.3152	P	1200
354	0.8882	T	0
347	1.1405	T	60
311	2.2392	T	300
264	3.4741	T	600
221	4.4782	T	900
177	5.5882	T	1200

3.3.3. The Visualization

Finally, all the data gathered for the six TPs are graphed visually using the enhanced star glyph. The visualization presented in Figure 3.7 shows a side view of the TPs predictions at selected look ahead times for all engine types. Each spoke represents the horizontal error of a TP's prediction at a particular look ahead time. The length of the spoke is the average horizontal error of the prediction. Spokes that extend further from the center indicate that the TP did a poor job at making its prediction.

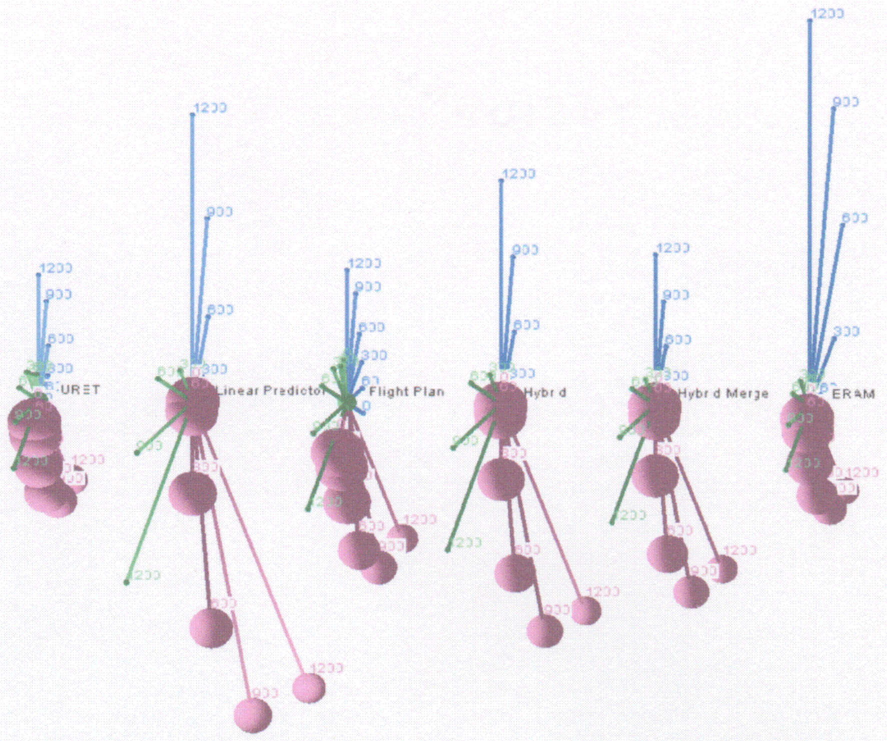

Figure 3.7 Multiple enhanced star glyphs of six trajectory predictors

The color of the spokes tells what engine type a spoke is associated with. For this example, the pink color is for jet planes, green is for turboprop, and blue is for piston.

Finally, the size of the spheres on the ends of the spokes relate to the number of measurements that have been taken. Larger spheres signify more measurements. The pink spokes at the bottom of the graph have much larger spheres since jet planes are much more common than turboprop or piston aircraft.

In Figure 3.7, the spokes at the top of the graph correspond to piston aircraft. This is the current focus of the graph (the graph can easily be manipulated to focus on a different engine type). We find that ERAM (star glyph on the far right) still needs work in predicting the positions of piston aircraft, as shown by the length of the spokes at the top which are much more distant from the center for ERAM. The visualization is confirmed when looking at the compiled version of piston aircraft at a 1200 second look ahead time in the following data set.

Table 3.4 shows a set of data for all the TPs for piston aircraft at a 1200 second look ahead time. ERAM's average horizontal error is 17.32 away from the actual value which is much worse than the other TPs. Being able to predict the position of aircraft early is crucial in preventing aircraft from entering conflict.

Table 3.4 Compiled data set

TP Type	Count	Avg. Horz. Error
URET	74	5.6446
Linear Predictor	172	12.9068
Flight Plan	171	5.9393
Hybrid	172	10.0111
Hybrid Merge	171	6.7264
ERAM	162	17.3152

Analysts and developers can now use this information to determine where in ERAM the algorithm for piston aircraft is causing inaccurate predictions.

ERAM, however, is very accurate in its predictions of jet aircraft, as seen by the pink spheres at the bottom of Figure 3.7. This extreme difference in accuracy is likely due to the developers focusing mainly on jet aircraft for now, since jets are much more prevalent in the NAS.

3.4. Conclusion

In this chapter we presented a way to improve the basic star glyph. By adding in a sphere at the end of each spoke on the star glyph another attribute could be graphed. The color of the sphere also indicated a category that a piece of data belongs to. Our clustering effect reduces the clutter problem that the star glyph suffers from. Finally, by arranging a set of star glyphs along a certain axis, a multitude of data sets can be compared.

This was tested on six trajectory predictors currently being analyzed by the Federal Aviation Administration. It was found that ERAM, the Air Traffic Control system set to be deployed within the next few years, performs well on horizontal prediction accuracy for jet and turboprop aircraft, but still needs work in predicting the horizontal locations of piston aircraft. This finding may indicate that ERAM needs to improve the performance model of this engine type.

The data needs to be analyzed deeper in order to find the particular problem that is causing this. For this purpose a planar plot of just the points of interest would be useful. Our future work includes researching new ways to display the detailed information of the particular points the user is interested in investigating further.

4. CONCLUSION AND FUTURE WORK

The relatively young field of information visualization is an ever-changing area of research. New concepts and theories are being introduced every day as new discoveries are made. This thesis presented two visualization systems that each introduce several new concepts to the field of information visualization.

Our PIEVis system introduced the concept of using a tree drawing to display directed graphs and using interactive highlighting to represent edges with no crossings. The interactive, circular labeling method is also a novel concept of creating aesthetically pleasing labels with no overlapping, for only the nodes of the most interest. The PIEVis system has many applications to visualizing large or infinite graph structures, and an example of this was discussed in World Wide Web visualization. PIEVis uses many ideas from the original WebGPS, but expands upon the capabilities and functionality by allowing users to view the full graph structure with no relationship information lost. All of the drawbacks that were discovered in the WebGPS user study have been addressed in PIEVis, creating a faster, more efficient, and cleaner visualization system, though PIEVis is not without its own drawbacks.

With the introduction of 3D televisions and other 3D consumer products recently, it seems only natural that the future of information visualization involves using the advantages of 3D technology. Many different types of 3D technology exist, but one of the most promising for information visualization is the use of head tracking. Head tracking 3D visualizations give the illusion of peering through a window at a visualization. As the user moves his or her head around to view different areas of interest, the visualization is updated accordingly using the natural human reaction as a means of interacting with the visualization. Some research has begun involving extending the PIEVis graph into three dimensions using head tracking for navigation. Though the idea is in a very early stage of development, early research seems promising as the extent of capabilities the head tracking system would provide could be extremely valuable to graph visualization.

In the near future, some work will be done in integrating PIEVis into a focus/context visualization, with PIEVis being the focus portion, and a graph system such as Walrus being the context portion of the visualization. This will help to overcome a few of the weaknesses of PIEVis, one of which is its inability to display the full context of a very large graph.

Going hand in hand with this weakness is PIEVis' limits in displaying levels. Although no hard limit is given, a large graph can get crowded quickly, and most graphs will become too crowded attempting to display any more than three levels of children.

Another weakness that could be potentially dangerous is the lack of error checking. With any visualization, a certain level of accuracy is assumed by the user. In displaying things such as the World Wide Web, this is not important as the crawler is real time and not displaying critical data, but if PIEVis is used to display data such as electrical graphs or terrorist networks, accuracy could be very critical. In these cases, it is possible to add error checking into the crawlers that are developed for those particular cases, but as of now, there is no error checking performed by PIEVis, and all input is assumed to be correct by the visualization.

Our Enhanced Star Glyphs expanded a popular data visualization method into three dimensions, allowing much more data to be visualized in a comparative plot. The visualization was extended further adding a fourth dimension of weight represented by a sphere on the ends of each glyph. By adding these two extra dimensions, Enhanced Star Glyphs can help determine overall and local outliers among blocked multivariate data sets, allowing the user to locate the areas that need further investigation. In the example case discussed, the visualization system proved to be useful in comparing multiple sets of similar data and even helped to discover an issue in a real Air Traffic Control system.

Though the system is good for displaying large amounts of data, with no ability to zoom in to view information in more detail, the user may miss some important details. A semantic zoom capability is currently being researched which will allow the user to choose a spoke of the graph to zoom into and expand. In this expanded view, the user will be able to interact with the more detailed information, discovering outliers in each data set, and giving the option to investigate those outliers further or remove them from the analysis causing the visualization to update accordingly.

The two visualization systems discussed in this thesis present many capabilities in several different areas of research and PIEVis may even hold some promise as a consumer product in the future.

REFERENCES

[1] R. Jianu, A. Rusu, A. J. Fabian, and D. H. Laidlaw. A Coloring Solution to the Edge Crossing Problem. In *Proceedings of 13th International Conference Information Visualisation*, pages 691-696, July 2009.

[2] U. Dogrusoz, K. G. Kakoulis, B. Madden, and I. G. Tollis. On Labeling in Graph Visualization. *Information Sciences: An International Journal*, 177:2459-2472, June 2007.

[3] G. Wills. Visualizing Network Data. *Encyclopedia of Database Systems*, 2009.

[4] A. Yamaguchi and H. Toh. Two-Layered Genetic Network Drawings with Minimum Edge Crossings. *Genome Informatics 12*, pages 456-457, 2001.

[5] G. DiBattista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[6] S. Malitz. On the Angular Resolution of Planar Graphs. *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, 1992.

[7] A. Rusu, C. Yao, and A. Crowell. A Planar Straight-Line Grid Drawing Algorithm for High Degree General Trees with User-Specified Angular Coefficient. *Proceedings of the 12th International Conference Information Visualisation*, July 2007.

[8] J. Fekete, D. Wang, N. Dang, A. Aris, and C. Plaisant. Overlaying Graph Links on Treemaps. *Proceedings of the 2003 IEEE Symposium on InfoVis*, pages 82-83, 2003.

[9] C. Tominski, J. Abello, F. van Ham, and H. Schumann. Fisheye Tree Views and Lenses for Graph Visualisation. *Proceedings of the 10th International Conference Information Visualisation*, pages 17-24, 2006.

[10] A. Pavlo. Interactive, Tree-Based Graph Visualization. Master's thesis, Rochester Institute of Technology, May 2006.

[11] E. Freeman, E. Freeman, B. Bates, and K. Sierra. *Head First Design Patterns*. O'Reilly Media, October 2004.

[12] Dr. A. Rusu, C. Santiago, and R. Jianu. Real-Time Interative Visualization of Information Hierarchies. *Proceedings of 11th International Conference Information Visualisation*, July 2007.

[13] S. T. Teoh and K. L. Ma. RINGS: A Technique for Visualizing Large Hierarchies. *Proceedings 10th International Symposium on Graph Drawing*, 2528:268-275, 2002.

[14] F. van Ham and A. Perer. "Search, Show Context, Expand on Demand": Supporting Large Graph Exploration with Degree-of-Interest. *IEEE Transactions on Visualization and Computer Graphics*, 15:953-960, December 2009.

[15] C. M. Macal and M. J. North. Tutorial on Agent-Based Modeling and Simulation. *Proceedings of the 2005 Winter Simulation Conference*, 2005.

[16] C. M. Macal and M. J. North. Tutorial on Agent-Based Modeling and Simulation Part 2: How to Model with Agents. *Proceedings of the 2006 Winter Simulation Conference*, 2006.

[17] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.

[18] T. Cohen and J. Gil. Better Construction with Factories. *Journal of Object Technology*, 2007.

[19] Y. Aridor and D. Lange. Agent Design Patterns: Elements of Application Design. *The Second International Conference on Autonomous Agents, IEEE,* 1998.

[20] J. Eklund, J. Sawer, and R. Zeiliger. Nestor Navigator: A Tool for the Collaborative Construction of Knowledge Through Constructive Navigation. *Proceedings of Ausweb '99: The Fifth Australian World Wide Web Conference,* 1999.

[21] M. L. Huang, P. Eades, and R. F. Cogen. Webofdav – Navigating and Visualizing the Web Online with Animated Context Swapping. *Proceedings of the 7th World Wide Web Conference,* pages 636-638, 1998.

[22] A. N. Langville and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings.* Princeton University Press, 2006.

[23] Y. Hyun. Walrus – A Graph Visualization Tool. http://www.caida.ord/tools/visualization/walrus/index.xml, 2002.

[24] J. Chambers, W. Cleveland, B. Kleiner, and P. Tukey. *Graphical Methods for Data Analysis.* Wadsworth, 1983.

[25] T. Kapler and W. Wright. Geotime Information Visualization. *Information Visualization 2005*, July 2005.

[26] J. Stasko, C Görg, and Z. Liu. Jigsaw: Supporting Investigative Analysis Through Interactive Visualization. *Information Visualization 2005*, July 2005.

[27] H. Kang, C. Plaisant, B. Lee, B. B. Bederson. Netlens: Iterative Exploration of Content-Actor Network Data. *Information Visualization 2005*, July 2005.

[28] A. Aris and B. Shneiderman. Designing Semantic Substrates for Visual Network Exploration. *Information Visualization*, 6, December 2007.

[29] J. S. Yi, R. Melton, J. Stasko, and J. A. Jacko. Dust & Magnet: Multivariate Information Visualization Using a Magnet Metaphor. *Information Visualization 2005*, July 2005.

[30] M. Ward. Xmdvtool: Integrating Multiple Methods for Visualizing Multivariate Data. *Proceedings of Visualization*, 1994.

[31] M. M. Paglione, Dr. H. F. Ryan, R. D. Oaks, J. S. Summerill, and M. L. Cale. Trajectory Prediction Accuracy Report User Request Evaluation Tool (URET)/ Center-Tracon Automation System (CTAS). *DOT/FAA/CT-TN99/10*, May 1999.

[32] M. L. Cale, S. Kazunas, M. M. Paglione, and Dr. H. F. Ryan. User Request Evaluation Tool (URET) Algorithm Assessment Report. *DOT/FAA/CT-97/4*, 1997.

[33] Dr. H. F. Ryan, G. Chandler, C. Santiago, M. M. Paglione, and S. Liu. Evaluation of En Route Host Computer System's Trajectory Generation and Strategic Alert Processing: Analysis of ERAM Performance. *DOT/FAA/TC-TN08/10*, November 2008.

[34] Dr. H. F. Ryan and M. M. Paglione. State Vector Based Near Term Trajectory Prediction. *American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control Conference*, 2008.

[35] M. M. Paglione and R. D. Oaks. Implementation and Metrics for a Trajectory Prediction Validation Methodology. *American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control Conference*, August 2007.

[36] US Sky Link. http://www.avchart.com/resource/charter-aircraft-types.asp.

[37] A. Rusu, C. Santiago, A. Crowell, and E. Thomas. Enhanced Star-Glyphs for Multiple-Source Data Analysis. *Proceedings of the 13th International Conference Information Visualisation*, July 2009.