Visualization of Usenet Newsgroups through
Graphical Representations
by
Maik Flanagin

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical [Computer] Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

May 21, 2000
© Copyright 2000 Maik Flanagin. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Author
Department of Electrical Engineering and Computer Science
May 17, 1998

Certified
by
Walter Bender
Thesis Supervisor

Accepted
by
Arthur C. Smith
Chairman, Department Committee on Graduate Theses
Visualization of USENET Newsgroups through Graphical Representations
by
Maik Flanagan

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 2000

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer [Electrical] Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The purpose of this paper is to describe the design for a graphics-based reader for Usenet newsgroups that will allow users to search for specific information more efficiently in a given newsgroup and gain insight about a newsgroup that they would not otherwise have had. Newsgroups are complex and unorganized, making it difficult for people to find what they’re looking for. Graphics, if used correctly, can assist a user in finding the information in a newsgroup by drawing attention to what could be relevant to the user while simultaneously relegating less important information.

Thesis Supervisor: Walter Bender
Position: MIT Media Laboratory - News in the Future Group
Table of Contents

Abstract................................................................................................................. 2
1.0 Introduction .................................................................................................... 5
2.0 Problem Definition........................................................................................ 7
3.0 Related Work................................................................................................. 9
4.0 Design Criteria.............................................................................................. 11
5.0 Design Components..................................................................................... 14
6.0 Visualization Tools....................................................................................... 16
7.0 Comprehensive Design Overview.............................................................. 19
    7.1 Underlying System Implementation......................................................... 19
    7.2 Design #1............................................................................................ 23
    7.3 Design #2............................................................................................ 26
    7.4 Design #3............................................................................................ 33
    7.5 Design #4............................................................................................ 36
8.0 Conclusions.................................................................................................. 41
9.0 Bibliography.................................................................................................. 43
Appendix: Illustrated Example of Design #2 algorithm............................... 44
List of Figures

Figure 1: Graphical User Interface.................................................................22

Figure 2: Example of Design #1.................................................................24

Figure 3: Design #1 breaks down due to numerous selected criteria ..............26

Figure 4: Placement of Criteria around the center of the graph in Design #2........28

Figure 5: Sample thread in design #2..........................................................29

Figure 6: Example of Design #2.................................................................30

Figure 7: Design #2 with one criteria..........................................................32

Figure 8: Example of Design #3.................................................................33

Figure 9: Example of Design #4.................................................................36

Figure 10: Example of design #4 with a criteria selected...............................38
1.0 Introduction

Information. With the sweeping revolution computers and the Internet have brought us in the past several years, information is literally at our fingertips... Or is it? With the thousands of people writing email, building web pages, and using instant messaging systems, users are bombarded with so much information that finding exactly what they are looking for has become a challenge in itself.

In particular, the Usenet newsgroups are a wealth of information, developing ideas, enthralling debate, and useless nonsense. Status quo news browsers are great for leisurely Sunday afternoon reading, but the process of looking for a specific piece of information can be quite frustrating. The staggering number of posts “can take considerable time to survey and it is often difficult to keep up with the volume” (Hauben and Hauben 63).

Users need a newsreader specifically tailored to their needs. For this reason, I have designed and implemented a news browser that includes a graphical user interface, which allows users to utilize a system of restraints on authors and keywords in order to help isolate the information they are looking for.

The organization of this paper is as follows: Chapter two will discuss the problem of newsgroup visualization and information finding in more detail. Chapter three describes the system upon which the news browser I built is based. Chapter four lists some of the objectives I strive to accomplish in this project. Chapter five describes the components involved in building the visualization, while chapter six discusses the principles which I
considered in designing the visualization. Chapter seven describes the actual implementation of the news browser, going into detail about the underlying classes that allow it to function. After which, chapter seven covers each iteration of my visualization design and debates the advantages and disadvantages of each design. Chapter eight concludes the paper with an analysis of the design process and thoughts about future work in this area.
2.0 Problem Definition

The problem centers around Usenet, a world-wide distributed discussion system. The Usenet system is divided into newsgroups, which are arranged hierarchically by subject. Articles or posts are posted to these newsgroups and can be viewed by users in possession of the appropriate news browser software (Moreas 1). Current implementations of news browsers are hardly more than lists of messages with subject, author date, time, and other fields.

Messages are posted to a particular newsgroup in a post/reply model, where a user can post his own original message or link his message to an existing one. Such organization allows freedom for those posting messages to a given newsgroup, but results in chaos for those reading newsgroups. Specifically, a user must face the two following problems:

2.1 User can’t find specific information in a given group.

Prospecting for information in newsgroups is largely unprofitable (Kellogg and Richards 8-9). The user has no guarantee that the information he’s looking for even exists. Even if it did, the time spent engrossed in fruitless search outweighs the potential value of the information found. Furthermore, the user does not know whether or not a more relevant source of information exists for his purposes.
2.2 User has no intuition about a given group.

The ability to search for information in a newsgroup may not be enough to find the information a user is looking for. Any news browser whose functionality is limited to raw searching lacks in that it “requires a user to bring knowledge of what they are looking for ...in order to access the information” (Rennison 10). Other than the title of the newsgroup (alt.rodney-king or alt.cloned-sheep.bah.bah.bah), the user does not know anything about the newsgroup until he starts shuffling through the posts. He has to prospect, not only to find specific information in a newsgroup, but also to find out general information about the newsgroup.

For instance, alt.rodney-king could have discussions centering on racism, the trial, the riot, the L.A.P.D, or jokes about Rodney King. Surely a user needing to do a research paper on the L.A. riots would not want to waste time reading jokes. If he knows that the newsgroup is not primarily focused on the Los Angeles riots, he can find a different newsgroup like alt.riots.
3.0 Related Work

Warren Sack’s thesis proposes a solution to the chaos of Usenet newsgroups. His system, called a conversation mapper, divides the newsgroup metaphor into two aspects, a social network and a semantic network, and builds separate models of the newsgroup using these two principles. The goal of this interface is to provide a very logical means of exploring and relating the many posts in a very complex newsgroup environment.

There are two types of users in the world of newsgroups. One type is defined as the scientific user, while the other is defined as the layman. The conversation mapper is useful for helping the laymen decide if a particular group or a particular thread within a group is relevant to what he’s looking for and simultaneously the conversation mapper is useful for helping scientists gain a better understanding of VLSC.

The conversation mapper produces a representation of the social network within a newsgroup. The social network is the network of relationships between different users within a group as they post. Relationships are built when User A either responds to or quotes User B. Through the social network, Spam is easier to ignore, since users who post frequently and elicit no interaction with other users are easier to find. Conversely, the social network also shows which users generate the most interactions with in the newsgroup and act as unofficial moderators.

The conversation mapper also provides a representation for the semantic network within a newsgroup. This semantic network is built from the most frequently reoccurring ideas
in a newsgroup. Through the use of a specialized language parser, the conversation mapper can build networks of interrelated ideas. For instance, consider the case of two nouns, which are used frequently in different postings, that are subjects in a sentence. In such a case, if the nouns share the same verb, then a relationship can be derived between the two nouns. A user can then exploit this graph to find the keywords that can help him find what he is looking for.

However, Sack’s Conversation mapper suffers from profound complexity issues. Even though it builds linguistic relationships between different subjects in a newsgroup, a user wouldn’t understand the graph unless he understood the linguistic principles behind it. Furthermore, even though the system can show which users generate the most discussion in the newsgroup and which users are largely ignored, a user wouldn’t easily understand this information for two reasons: 1) he has to coordinate information from four different output screens at once, and 2) given a sufficiently large enough newsgroup, the graph gets too complex to derive any sort of insight into the interface, since information starts to overlap in the display.
4.0 Design Criteria

My objective is to create a graphical user interface for a Usenet newsgroups browser that requires a minimal amount of training to use and simultaneously presents graphics in the most efficient way possible in order to aid the user in finding the sort of information he’s looking for. To achieve this objective, the system should, as much as possible, fulfill the following criteria:

4.1 Easy Integration

Integration is the extent to which components of knowledge in one domain are tied to other components of a person’s knowledge (Riley 167). A user has an easier time learning how to use a new interface if he can apply knowledge he already has.

For example, Microsoft applications have similar graphical user interfaces since they have similar functionality. If a user is familiar with Word, then he is also familiar with the features of word that appear in programs like Excel such as cut-and-paste, file save, and spell check. Microsoft applications share many of the same menu bar items such as file, edit, window, etc. Due to their similar structures, if a user knows how to load and save files in one application, he can also do so in another without having to spend any time learning how to do it. The news browser should employ a similar concept and hopefully be simple enough to use without any outside assistance.
4.2 Simplicity

Sometimes less is more and such is the case when dealing with representing Usenet with graphics. Trying to represent something as complex as a Usenet newsgroup can result in a graph that is just as complex and reveals nothing except that graphs can get really messy. We already knew that! So the problem to solve is how to represent something potentially complicated in a very simple manner without losing any vital information.

The ideal browser would show the user exactly what he needs to know, no more and no less. Graphics is useful to express information when it “expresses exactly the input information, that is all the information and only the information” (Mackinlay 17). Any additional information has the potential to distract the user and undermine the effectiveness of a simple graphical representation.

4.3 Scalability

The browser must scale properly. A newsgroup can have 5 or 500 entries. A message can have 0 or 50 replies. To make matters worse, visual display space is limited by the size of the user’s computer screen. Any interface must take such enormous scaling issues under consideration. Not only can a poorly scaled system crowd the display area, making a presentation hard to see, a poorly scaled system might also be plagued by the complexity issues described above.
Therefore, the system should use screen space as efficiently as possible and avoid representations that only seem insightful with a limited data set, but overwhelming with a slightly larger set.

4.4 Insight

A news browser ought to provide implicit information about a given newsgroup. When a user reads messages from a newsgroup everyday he gains a certain intuition about that particular newsgroup: what he expects to find, what the "hot" topics are, and who the major players are in some of these hot topics. On the other hand, when a user looks at a newsgroup for the first time, he lacks this aforementioned intuition. The ideal news browser should make the first time use of a given newsgroup feel like any other time.

Such insight can improve a user's experience with Usenet. Knowing about the usage patterns of certain individuals and which topics generate discussion can help a user differentiate spam from relevant topics and decide if the post he wants to make would fit in the given newsgroup or if he would be better off finding another newsgroup to post in.
5.0 Design Components

The very heart of this project is the visualization of a newsgroup in such a way that the user can find specific information. On one side is the user who is trying to find some particular information that interests him. This user must face off against the garbled and vast vault of knowledge, the Usenet newsgroup. The visualization must be able to translate the users wants and the newsgroup’s information into something graphical, so that the pair may interface in some way. By using threads to represent newsgroup content and criteria to represent the user’s desires, the system will have the concepts it needs to build graphics.

5.1 Threads

A newsgroup can be thought of simply as a set of individual articles. However, articles can be related to other articles by taking the form of a reply. Replies to an article indicate an extension of the ideas presented in the source article. An article can have any number of replies and replies can have their own replies, thus creating treelike structures representing a developing discussion, often referred to as a thread. Thinking of a newsgroup as a set of threads seems preferable to thinking of a newsgroup as merely a set of articles, since the latter ignores the relationships between these intertwined articles.

5.2 Criteria

Criteria are those objects that the user hopes to find in the newsgroup. For the purposes of this project, criteria have been restricted to two categories: author and
keyword. The author is the name used in the ‘from’ fields of the posted articles. If the user seeks an article written by a specific person, then he could select the user’s name from a list of names and view all the different threads that the user has posted in. A keyword is simply an occurrence of a specific word in a given thread. Using keywords helps the user search for particular topics of discussion.
6.0 Visualization Tools

This section examines the concepts that can be employed in order to fulfill the stated design criteria. According to Andrienko's paper on map visualization, "each presentation method is based on the use of one or more of the visual variables such as size, shape, shade, and orientation" (par. 3). Some of these types of variables are ideal for showing quantitative data, while some are better suited for qualitative data.

6.1 Size

Size does matter. In fact, making one graph object bigger than another show the relevant importance of the larger object to the smaller. Size is easy for users to notice and should be the primary method by which to show qualitative differences between objects.

6.2 Shape

Differences in shape are even easier to notice than differences in size. However there is no real way to compare shapes against each other, nor is there a way to easily understand relationships between two different shapes without explicit knowledge of what each shape means.

In the case of the newsgroup browser visualization, shapes can be used to distinguish between the two types of objects it deals with. Criteria and Threads ought to assume different shapes in any sort of visualization.
6.3 Color

Color can also be used to show qualitative differences. However, the problem that plagues color is that it does not have a linear value. Color is broken down in different magnitudes of red, blue, and yellow. Because of this, there is no way to know that a green object is more important than a black object. In this way, color bears some similarity to shape in that the user has to know explicitly what each color means. But unlike shape, color can have subtle differences that the user may not easily detect.

6.4 Position

Placement in a visualization can focus the user's attention on important objects, while distracting away from less important objects. According to Andrienko's paper, position "has the highest expressiveness, it can encode any type of data" (Andrienko, par. 3). The browser's visualization should try to exploit the versatility of position. By placing objects in the front and the center of the scene, will obviously capture a user's attention before something off-center or in the background.

6.5 Spacing

Relationships between different objects can be demonstrated by their placement. Objects placed close together imply a 'close' relationship, while distant objects imply the opposite. Such a relationship can be made even more explicit using a linking graphical object as such a line connecting two node objects.
6.6 Labels

Some pictures are worth a thousand words, while others need a thousand words of explanation. When precision is important, placing a label on or next to an object may be easier than attempting some sort of abstract representation. The unfortunate drawback to labeling is that it may run into scaling issues; As the size of the data set the system deals with increases, so does the amount of representations needed. Labels may have to be shrunk or face collision problems, since the size of the display is fixed.
7.0 Comprehensive Design Overview

This section describes thoroughly the system built to solve the problem of finding information that the user is interested in. It starts with section 7.1, a description of the low-level, back end portion of the system, which fetches the newsgroup and its articles. This section also describes the middle layer system which parses the newsgroup into information that the browser can use to create its display and, finally, describes the user interface elements common to all of the visualization implementations. The following sections describe different iterations of the newsgroup visualization design along with commentary about the positives and negatives of each design.

7.1 Underlying System Implementation

The back end of the USENET news browser is general enough to allow for a number of different user interfaces. This underlying system is written in Java, since the simplicity of its net package allows for quick coding of a protocol.

7.1.1 NNTP Interface Module

USENET uses the Network News Transport Protocol (NNTP) to facilitate communication between remote sources. The protocol class simply opens a connection at port 114, sends text messages, and waits for a response. The process of retrieving an article from a server is tedious and, by far, the largest time bottleneck in the entire system. Most conventional browsers only retrieve the
headers of articles when they load a newsgroup and retrieve the full article when a user wants to read it. On the other hand, the news browser relies on full retrieval of all messages in order to gather the information needed to generate a visualization of the newsgroup. Various methods could be employed to improve system performance such as caching old articles. However, they are out of the scope of this project and should be saved for later.

7.1.2 NewsArticle Module

Once an article is retrieved from the server, it is instantiated as a NewsArticle by parsing the retrieved data. NewsArticle stores all of the relevant information such as whom the message is from, what the subject of the message is, and so forth. NewsArticle has an array of NewsArticles, which represent the replies to a given message, allowing for the building of thread structures.

7.1.3 Newsgroup Module

The top level of the back end models the structure of a newsgroup as a list of articles. These articles are retrieved from a server using the NNTP protocol. The system iterates over all articles, retrieves the ones that have not yet expired, and enters them into a hash table data structure. If the reference field of the given article is the same as the identification field of an article already stored in the hash table, then the article is put in the table and in the replies array of the referring article. By doing so, any higher level interface can either retrieve all of the articles
by getting an enumeration of the hash table or retrieve all of the threads, or groups of articles, by getting an enumeration of all articles that have no reference ID.

7.1.4 WordPool Module

In order to extract information from the newsgroup to build search criteria, the Newsgroup must retrieve a list of keywords from each article and keep a running total of the number of instances of each word. The newsgroup must also keep a running total of how many articles a given author posts. This is done through the use of a WordPool data structure.

The WordPool must first retrieve the information from the articles. When gathering information about keywords, the WordPool calls a method in NewsArticle that returns an array of all of the words in the article. The NewsArticle class filters out unimportant words like articles, prepositions, and conjunctivas. Obviously the word “a” might be frequently mentioned in every news group, but it might not be of any importance.

The WordPool data structure must handle large amounts of data fairly quickly. Therefore, it is implemented as a red black tree of (word, count) pairs. The tree is sorted alphabetically by the words to allow additions to be made in logarithmic time. When a word that already exists in the tree is added, the count of the word increments by one and nothing new is inserted into the tree.
Once the WordPool finishes adding every word, the list is sorted using a quicksort algorithm and the most popular choices are returned to the Newsgroup.

WordPool allows for the arbitrary choice of the number of top keywords, so the program could see the best two or twenty keywords!

Building a word pool from the authors is hardly any different. The process of retrieving authors from the articles is simple, since the WordPool only has to look at the ‘from’ field of the news article.

7.1.5 Common Graphical User Interface Module

![Graphical User Interface](image)

**Figure 3: Graphical User Interface**

All designs essentially use the same base graphical user interface. The graphical user interface breaks down into three components; a panel that displays what is in the newsgroup. (With the exception of the last design, this information is shown simply as a list of the subjects of all posts in the newsgroup.) The purpose of this
panel is to match the functionality of a conventional newsreader. The intuition behind displaying a list of post subjects is that any given user would already be familiar with such a device since it is what's in a typical newsreader, thus tying into the objective of achieving easy integration.

Opposite of the newsreader panel is the visualization panel, which displays a graphical representation that aids the user in finding some specific information in a newsgroup. The visualization includes, in some way, the set of user defined criteria and demonstrates the relation between these criteria and relevant threads in the given newsgroup.

A selection panel is situated above the visualization panel, which allows the user to select his desired criteria. The user can make a selection in the top choice area to choose to select either authors or keywords. The choice area underneath it contains the list of top authors or keywords.

7.2 Visualization Design #1: Sweet and Simple

The first incarnation of the browser system is an attempt to simplify the graphical visualization as much as possible. The display consists of only three elements: a set of criteria selected by the user represented as boxes, threads in the given newsgroup,
represented by circles, and links in between the two representing the relevance of criteria to each of the threads.

The criteria are represented by boxes presented as a column on the left-hand side of the display. Choosing a keyword or an author from the lists of selectable criteria adds another box at the bottom of the column.

![Figure 4: Example of Design #1](image)

The threads are represented by circles of varying sizes, representing thread size, in a column on the right side of the display. Each circle has the name of the root article in the thread and the threads are in chronological order based on the date of this root article, but
other than that, the user isn’t provided with any other information about the specifics of a particular thread.

Connecting the criteria to the threads are weighted lines which show the relevance of the criteria to the posts. The weight of each thread is simply computed as the number of instance of the given word in all of the articles in the thread if the criteria is a keyword, or the weight is the number of articles in the thread posted by a particular author in the case the criteria is an author.

7.2.1 Advantages

As stated before, this design is simple and easy to explain. If a user is looking for information about two or three subjects, he can simply select them and find the threads that have edges to all of the criteria he selected. Showing the relative size of the thread can help the user find relevant information even better, since larger threads indicate more interesting discussion.

7.2.1 Disadvantages

The disadvantages of the system lie in its simplicity. In fact, this design oversimplifies the problem of searching. What if the user wanted to select a large number of criteria? In such a case, it’s difficult for the user to find the sort of thread he’s looking for since each thread has some subset of all of the criteria. The user must sort out the information himself.
The system also fails to provide much insightful information about the newsgroup itself. The display only shows the number of threads and their relative size. A better system could sacrifice some of this simplicity and provide the user with a lot more information.

7.3 Design #2: The Vector Approach

The second incarnation of the browser system takes advantage of spatial relationships to show the relevance of criteria to the various threads. Instead of using two monotonous
columns of graph nodes, the second design places the graphical representations for threads at a distance away from the criteria that represents the applicability of the criteria to the thread.

Generating meaningful graphs is a very difficult problem. The best one could hope for is an algorithm that generates graphs in polynomial time (Juenger 1-2), which is not at all desirable considering the vast number of nodes the system could potentially be dealing with. Consider one selected criteria: The visualization could consist of a criteria node and a thread node for each thread that relates to the criteria at a distance proportional to the relevance of the criteria. Such a system works fine with only one criteria node, but falls apart with two or more. When two criteria are selected and share the same three or four criteria, the nodes must be positioned in such a way that the lengths of the edges are consistent. Sometimes this is just not possible to do.

Therefore it is necessary to exploit the fact that the graph uses two different types of nodes: criteria and threads. The way this design accomplishes this is by placing thread nodes as close to all of the relevant criteria nodes as possible.
Figure 4: Placement of Criteria around the center of the graph in Design #2

As shown above in figure 4, the criteria are set up in a ring around the middle of a display. If there is just one criterion node selected, then that node is placed in the center. If there are two criteria nodes selected, then two nodes are placed equidistant on opposite sides of the center of the display. As more are added, they are placed the same distance away from the center are equals angles of 2*PI divided by the number of selected criteria.

The relevance weights are a value between 0.0 and 1.0 and computed by dividing the number of occurrences of a criterion in a thread by the highest number of occurrences of any criterion for that thread. For example, if the user were looking for the keyword “brick” and there were 10 instances of the word brick in a thread about bricklaying and is
the most common word in the bricklaying thread is “stuff” with 20 instances, then the weight of the edge between the bricklaying thread and brick is $10/20 = 0.5$. Ideally, it is necessary to limit the size of the edges to build a reasonable graph.

Figure 5: Sample thread in design #2; arrows indicate relation between criteria and thread

The placement of the thread nodes is computed through the following process. First, an initial point is selected by choosing the midpoint between all of the criteria nodes that are relevant to the thread. If all criteria pertain to this thread, then by this definition, this initial point would be in the center of the display. Next, for each criterion, a vector is added to this point. The direction of this vector is the direction of the criterion from the initial point and the magnitude of this vector is the relative weight of the criteria’s relevance multiplied by the distance from the criteria to the center. The result is that
criteria that heavily pertain to certain threads have those threads relatively nearby, while threads that are equally relevant to all criteria are positioned in the center of the display. A fully illustrated example of the vector placement system can be found in Appendix A.

### 7.3.1 Advantages

The advantages to this system are that it can handle a large number of selected criteria and still allow the user to immediately see what he wants to. As long as the user has some notion of how the positioning of the nodes work, he can make good use of the system.
7.3.2 Disadvantages

The first of two moderate drawbacks to this style of display is that the frequency of criteria occurrences takes a back seat. With more than one criteria selected, whatever sits closest to the center of the display is the most relevant thread to all selected criteria. A one-post thread with one occurrence of each criteria would, by the definition of the design, be the most relevant thread and sit in the middle of a display, while a thread with more posts and more occurrences, say 10 of one criteria and 9 of another would be located off-center in the display, even though it is obviously more relevant than the first thread.

However, even if really important threads are off-center, the user will not be thrown into utter confusion. The system attempts to differentiate between threads not only with position, but also with size. Larger threads will have larger sized representations and will draw just as much attention to the user as a centralized location. Besides, if one criteria is more important to the user than another, then he can focus more on thread nodes that are biased in the direction of the criteria he’s more interested in.

But the problem of handling the one-criterion case is more troublesome. When one criterion is selected, the criterion is placed in the center of the display and the vectors will push relevant threads as far away as possible. So, in this case, more
relevant threads are far away from the criteria node.

Finally, the design suffers the problem of overlapping representation in the display. Naturally, it is possible for many nodes to hold the same position in the graph. But it is not desirable for the user to have to deal with such a mess. Design #2 attempts to detect overlap and rotate colliding thread nodes about one of their criteria, corrupting the meaning of the display slightly, but at the same time make all nodes accessible to the user.
7.4 Design #3: The Third Dimension

The objective of the third design is to rectify the overlap problem of design #2 by expanding out into the third dimension! Three-dimensional representation cures the ills of overlap by maintaining the correct representation and simultaneously making the system a lot more aesthetically pleasing.

Figure 8: Example of Design #3

The design of the system is essentially the same as the second design, except that spheres are used to represent thread nodes instead of two dimensional circles and the text to show
criteria nodes are three-dimensional. The key similarity is the placement of the thread
nodes, which is similar to the placement in the previous design. The criteria nodes and
thread nodes are initially placed in the z=0 plane. As each additional thread is rendered,
the system checks whether the thread node’s representation will overlap another thread
node. In such a case, the thread will be translated by an arbitrary distance into or out from
the Z-plane so that no overlap can occur.

The system also adds an additional feature that shows the relative age of a thread by
adjusting the nodes transparency. Transparency shows the metaphor of a thread “slowly
fading out of existence.” An age weight is computed by dividing a thread’s most recent
post’s date, by the newest thread’s most recent post’s date. The resulting weight, which is
a value between 0.0 and 1.0, is multiplied by the thread node’s transparency.

3D graphics requires the writing of additional modules in order to interact with it. First of
all, the user needed to see the scene from different vantage points, since the system is
three dimensional and, obviously, some objects are going to block other objects.

Therefore, a module was written to rotate the scene along any axis by dragging the
mouse. Furthermore, since the display is no longer coded using Java’s AWT, the mouse
can’t directly interact with the graph in the same way and, thus, can’t open the thread for
reading. Therefore the system incorporates a module that takes the position of a mouse
click and sends a pick ray through the scene. The first object this pick ray hits is returned
and if this object is a thread node, the information is retrieved and the thread is opened.
7.4.1 Advantages

The biggest advantage of this design is that the representation is maintained with minimal tradeoff. Aside from that, by adding point light sources, and specularity to the scene, the display looks more impressive: a far cry from design one and its boxes and circles. The system can also use appearance attributes to show more information about particular threads. For example, design three added transparency to show relative age of threads. Other thread attributes can be added to the display in a similar fashion.

7.4.2 Disadvantages

Aside from sharing the latter disadvantage and the lack of simplicity of design #2, the third design suffers from the limitations of the 3D package used to write it. As of this writing, the Java3D package, used to code the third design is still in its beta version. The most frustrating deficiency is the lack of two-dimensional text. By themselves, spheres in a scene graph are meaningless. It is essential to know something about a thread’s content just from looking at the display. Ideally the system would make a two dimensional text representation of the subject of a thread and wrap it around the thread sphere providing the nodes with a pleasant, Christmas ornament-like label. But 2-d text is yet to be implemented.

Additionally, the picking mechanism used to open thread for reading suffers when the view of the scene is altered through manipulation such as rotation.
7.5 Design #4: Treemaps

The fourth and final design in the thesis attempts to painlessly integrate insight about the newsgroup into the system described in design #3 by replacing the left panel, which simply lists the posts much like an inbox would, with a graphical structure that represents the entire newsgroup.

Figure 9: Example of Design #4

The right panel is converted into a treemap, a graphical structure designed by Dr. Ben Shneiderman from Maryland in order to show large amounts of hierarchical data in a
situation where graphical space is limited. Instead of using the traditional nodes and edges found in most hierarchical graphs, the treemap uses rectangles to represent nodes and embeds these rectangles into a large rectangle to represent a subordinate relationship to another node. The treemap splits a rectangular display area into an arbitrary number of smaller rectangular areas, each of which can be further subdivided. Using color and size allows the treemap to display information about its nodes that a conventional hierarchical graph would have difficulty efficiently displaying.

Design number 4 uses the rectangular nodes to represent newsgroup threads. The treemap is always one level deep and the rectangle size reflects the relative size of the threads. As different criteria are selected and displayed in the right panel, rectangles that represent the nodes displayed in the criteria graph are colored in the right panel.

Treemaps are built in the following way: A treemap takes as input a mapping of objects to weights. In the case of the news browser, the objects represent different threads in a newsgroup and the weights represent the number of posts in a given thread. The list of objects is arbitrarily divided in half. The weights of one half of the objects are summed, as is the other half of the objects. The rectangle is divided into two smaller rectangles by a line segment perpendicular to one of the sides. The area of these smaller rectangles are equal to the total weight of their corresponding half of all objects. These smaller rectangles are subdivided recursively in a similar fashion, by breaking up a half list of objects into halves.
As shown above in figure 10, when a user selects a criteria, the threads that appear in the visualization panel are now highlighted in the reader panel.

7.5.1 Advantages

Design #4 successfully synthesizes two seemingly conflicting objectives: 1) showing what the user wants to see and 2) showing the user information about the entire newsgroup that he might like to know. The solution involves using two separate panels.
The spirit of treemaps can be used to extend the system’s scope beyond a single newsgroup. The entire system presented in this paper has assumed a single newsgroup as input. However, since newsgroups are hierarchical and the virtue of the treemap is to display hierarchical relationships, it is easy to see how the treemap can extend the system to multiple newsgroups and even subgroups. The treemap design for use is only one level deep. If the user wants to look at more than one newsgroup at once, the treemap can first allocate different rectangles for each newsgroup and, then, subdivide each rectangle into threads as described above, producing a two layer treemap. Of course more levels could be added if multiple newsgroups were considered under the same hierarchy such as alt.pro-wrestling which has the subgroups alt.pro-wrestling.wwf and alt.pro-wrestling.wcw. Unfortunately such a concept is infeasible, since working with only newsgroup is already computationally frustrating.

Whereas the visualization can only show which threads are more pertinent to given criterion relative to other threads, the highlighting of threads in the reader panel gives the user a good idea of how much of the newsgroup is pertinent to a given criteria or set of criterion. The user can simply look and judge how much of the treemap’s area is highlighted. Using such information can help a user decide if the group he is viewing is what he is looking for, exactly the desired insight described in section 4.4.
7.5.2 Disadvantages

The cost of this compromise is that the system potentially loses its easy integration, the quality described in section 4.1. The whole point of the right Panel is to provide the user with an interface that he already recognizes. The treemap approach eliminates the need for such a list, but introduces a new way of thinking to the user that he may initially reject due to its foreign appearance.

Design #5 also suffers a similar affliction as the second design, in that textual information becomes unreadable. In design two, threads located next to each other in the visualization might overlap the text in the graph describing the thread subject. In this design, the smaller the rectangle, the smaller and harder to read the text becomes and lengthy subject lines get cut.
8.0 Conclusions

The criteria listed in section 4 guided the design process over four iterations. The first design included a listing of articles that the user could select and read. Since users are already familiar with such a model, the system succeeds in providing easy integration. The first design also tries a simplistic approach to data representation as prescribed by the second criteria. Unfortunately, the benefits of simplicity are rendered ineffective as the number of selected criteria increases. The third design attempts to include some insight by using position and spacing to draw attention to more relevant information. But the system still could not handle scaling, nor could it provide insight about the newsgroup as a whole. The third design addressed the scaling criteria by using a 3D display that could let the user see everything by moving around in a 3D space. The final design tried fulfill the insight criteria by changing the representation of the newsgroup from a list of posts to a graphical representation of threads. This final design satisfies each criterion, with the exception of the first one: easy integration.

The future for the news browser includes adding a feature to fix the scaling problems of the treemaps. The second design fixes its scaling problems by adding another dimension. Instead of text overlap, the text is written on a spherical object that is not in contact with any other object in the graph. A solution to this problem could take a similar form by using 3D text placed at different depths to show the subject of each thread. Alternatively, the problem could be simply fixed by just adding a mouse motion listener that displays the full text of a thread only when the mouse is over a particular thread.
The problem of designing user interfaces for any system is subjective in nature; What may work for one user may not work for another. The fact is that there are several directions this design can take, but in order to really test the effectiveness of the browser, field testing needs to be done in order to find out what the user really wants.
9.0 Bibliography


Appendix: Illustrated Example of Algorithm used in Design #2

Figure 11: Criteria are placed at an equal distance away from the center of the display.
Figure 12: Thread is initially placed at the origin. Each thread has a set of weighted vectors that represent the relationship between the vector and a criteria. The length of the vector reflects the strength of the relationship. (Longer arrows mean stronger relationships.)
Figure 13: Thread is moved to the center of all of the criteria it has any relationship to. This point is called the vector origin and is the point at which the relationship vectors are applied.
Figure 14: Once the vector origin is found, the thread representation is moved a distance equivalent to the sum of the vectors applied to it. The thick arrow represents the three vectors applied to this example summed together.
Figure 15: Here’s the whole picture from start to finish. The sample thread has a relationship with criteria A, B, and C. The thread representation is moved to the midpoint of the shape created by the criteria A, B, and C. Finally, the vectors are summed and the sample thread is moved to that position.