CAR-TR-577
CS-TR-2748

August 1991

# Visual User Interfaces for Information Exploration

Ben Shneiderman
Human-Computer Interaction Laboratory & Department of Computer Science
University of Maryland, College Park, MD 20742

Abstract:
The next generation of database management, directory browsing, information retrieval, hypermedia, scientific data management, and library systems can enable convenient exploration of growing information spaces by a wider range of users. User interface designers can provide more powerful search techniques, more comprehensible query facilities, better presentation methods, and smoother integration of technology with task. This paper offers novel graphical and direct manipulation approaches to query formulation and information presentation/manipulation. These approaches include a graphical approach to restricted boolean query formulation based on generalization/aggregation hierarchies, a filter/flow metaphor for complete boolean expressions, dynamic query methods with continuous visual presentation of results as the query is changed (possibly employing parallel computation), and color-coded 2-dimensional space-filling tree-maps that present multiple-level hierarchies in a single display (hundreds of directories and more than a thousand files can be seen at once).

*Proceeding of the 54th Annual Meeting of The Amercian Society for Information Sciences, vol. 28 (Washington, DC, Oct. 27-31, 1991) 379-384.*

## 1. INTRODUCTION

Exploring information resources becomes increasingly difficult as the volume grows. A page of information is easy to explore, but when the information becomes the size of a book, library or larger, it may be difficult to locate desired information or to browse to gain an overview. The computer is a powerful tool for searching, but current user interfaces may be a hurdle for novice users and an inadequate tool for experts (Borgman, 1986). This paper suggests some novel possibilities for novice and expert users. There will certainly be other inventions and refinements, but this report presents some novel ideas to gain feedback and to enable other researchers and developers to create still more effective variants.

We will take the model of information-seeking (Marchionini & Shneiderman, 1988) over documents (containing text, graphics, images, etc.) organized into collections. The user's goals range from specific (fact-finding) to general (browsing):

- to locate the documents in the collection that are relevant to their needs, for example:
  - get the document on corn production in Iowa in 1977
  - find the document with the birthdate of Louis Daguerre
- to gain an overview of what is in the collection, for example:
  - are there more documents related to architecture than to oceanography?
  - what percentage of the microbiology documents deal with bacteria?
  - how many documents are there on Marco Polo?

The users might range from high school students wanting an introduction to French history, to expert historians wanting to find every primary source on Madame de Stael's salons and as many of her contemporaries as possible. Of course, readers could start at the beginning and simply work their way through the entire document collection, but the time required makes this approach unrealistic for large document collections.

Developing computerized filters to reduce the set of documents that need to be examined is a primary goal of researchers in computer and information science. The broad range of users and tasks makes the designers' role difficult, but computerized search interfaces do provide some useful and powerful tools:

**Full text string search**: users type a word and the system locates the next occurrence of the word in a document or the complete list of documents that contain the word. For example, (human OR person OR man) AND (computer OR machine). Many variants of string search exist:

- word stemming (plurals and other variants are retrieved),
- boolean combinations (AND, OR, NOT, and parentheses),
- online thesaurus, used to add synonyms, broader, or narrower terms,
- relevance feedback, indicating strength of relationship between the search terms and the documents,
- more-like-this document retrieval: users indicate which documents are relevant and the system provides more documents with similar terms.

String search is effective and widely used (Salton, 1989).

**Formatted field search:** users write queries in a database query language (Reisner, 1988; Jarke & Vassiliou, 1986) and specify matches on specific field values such as author, date of publication, language, publisher, etc. For example, (DATE > 1986 AND DATE < 1990) AND (LANGUAGE = ENGLISH OR FRENCH) AND (PUBLISHER = ASIS OR HFS OR ACM). Each document has values for the formatted fields and database management methods enable rapid retrieval even with millions of documents.

**Library index search:** rather than supplying a string for searching in the text, users explore a hierarchical index of subject terms that have been created by the document collection owner and can retrieve all the documents that have been indexed under that subject term (Soergel, 1985). With a rapid display it may be possible to browse productively through the subject index and learn its structure. Index terms may be alphabetically organized as in the Library of Congress Subject Headings or meaningfully organized as in the Dewey Decimal System. Other indexes may be organized geographically, chronologically (by publication date or by historical contents), etc.

Librarian or author defined indexes have the advantage that the terms are selected carefully, synonyms are cross referenced and hierarchic structures help by clustering related documents. By contrast, automatically generated keyword indexes with terms extracted from a document (except for a filter list of commonly used words) are easier to construct, but tend to be larger, more chaotic, and more ambiguous.

**Book index search:** A back-of-the-book index is usually author created, while a concordance is an automatically generated index on all words (except for a filter list of commonly used words). A table of contents is author generated and usually much shorter than an index, but sequential by topic (rather than alphabetical). Tables of contents or brief indexes have the advantage that they can be scanned to get an overall impression of scope of coverage.

**Hypertext:** users follow links in documents and jump rapidly from document to document. This works well for following a set of related items, but it may be difficult to find all relevant documents. Hypertext links may be author generated or automatically produced. Hypertext browsing enables simple point-and-click access and is also applicable to indexes, tables of contents, keyword lists, etc.

Of course, combinations of these approaches exist and are widely used. String search can provide an initial set of documents from which hypertext traversals begin. Index terms may be combined with full text string search, or a table of contents may be used to limit the range of a string search. The most effective strategy or combination of strategies will depend on the tasks, document collection structure, system response time, and experience of the users.

Refinements of the user interface can make a powerful difference. For example the Library of Congress's SCORPIO system is an effective query language in use since 1975 by thousands of Capitol Hill staffers and library visitors. However, the users need at least some training and practical experience to gain even basic proficiency (Stevens & Shneiderman, 1981). The first time user will need the assistance of a librarian acting as an intermediary. SCORPIO supports index searching (for authors, titles, and subjects) and formatted field search (for dates of publication, national language, etc.). Boolean combination can be specified but only one operator at a time can be applied to a numbered set of already selected documents (for example COMBINE 3 AND 5). Full text search is not supported, and neither is hypertext traversal. The recent development of a touchscreen interface has made life easier for first time users, and shortcuts to permit easier link following are being implemented.

Continuing research on improving the user interface for these search methods is needed because novice and first time users have great difficulty in learning new systems and because even expert users fail to find relevant information efficiently. As the flood of available information increases in online databases, electronic mail, bulletin board systems, etc., improved interfaces can help users cope successfully.

## 2. ARGUMENTS FOR VISUAL APPROACHES

The success of direct manipulation interfaces is indicative of the power of using computers in a more visual or graphic manner. A picture is often cited to be worth a thousand words and for some tasks (not every task) it is clear that a visual presentation, such as a map or photo, is dramatically easier to use than a textual description or a spoken report. As computer speed and display resolution increase, graphical interfaces are likely to have an expanding role. If a map of the United States is displayed then it should be possible to rapidly point at one of a thousand cities to get tourist information. Of course, a foreigner who knows a city name (for example, New Orleans), but not its location may do better with a scrolling alphabetical list. Visual displays become even more attractive to provide orientation or context, to enable selection of regions, and to see dynamic feedback for identifying changes (for example, a weather map).

Overall, the bandwidth of information presentation seems potentially higher in the visual domain than with any of the other senses. Users can scan, recognize, and remember images rapidly and detect changes in size, color, shape, movement, or texture. They can point to a single pixel, even in a mega-pixel display, and can drag one object to another to perform an action. User interfaces have been largely text oriented, so it seems likely that as visual approaches are explored some new opportunities will emerge.

There have been many attempts at graphical query formulation (Wong & Kuo, 1982; Jarke & Vassiliou, 1986; Kim et al., 1988) but the focus has often been on specifying linkages across relations, between components of an entity-relationship diagram, or between components of a binary relationship diagram (Senko, 1977; Mark, 1989). Graphical selection of attribute values and graphical specification of boolean operations (Michard, 1982) is likely to be a worthy direction for expansion.

There have been fewer efforts at graphical displays of database search results, although the potential seems very strong (Roussopoulos & Leifker, 1984). There is a growing movement among researchers in user interfaces for Geographic Information Systems that have a graphic orientation (Egenhofer, 1990). The attraction of visual displays, when compared to textual displays, is that the representation may be closer to the more familiar 3-dimensional world in which we were raised and in which we live. Within visual displays there are opportunities for showing relationships by proximity, by containment, by connected lines, or by color coding. Highlighting techniques (for example, boldface text or brightening, inverse video, blinking, underscoring, or boxing) can be used to draw attention to certain items in a field of thousands of items. By pointing on a visual display selection can be rapid and feedback is apparent. The eye, the hand, and the mind seem to work smoothly and rapidly in performing actions on visual displays.

## 3. SPECIFYING COMPLEX BOOLEAN EXPRESSIONS

Commercial information retrieval systems, such as DIALOG or BRS, permit complex boolean expressions with parentheses, but widespread use has been inhibited by their difficulty for users. Numerous proposals have been put forward to reduce the burden of specifying complex boolean expressions (Reisner, 1988). Part of the confusion stems from informal English usage where a query such as “List all employees who live in New York and Boston” would result in no results because the “and” would be interpreted as an intersection. Similarly the English usage “I’d like Russian or Italian salad dressing” is meant to be mutual exclusion, not union.

This section describes two research projects to develop more graphical visualizations of boolean expressions. One of the advantages of direct manipulation interfaces is that users are reminded of attribute names and values and merely select from the list provided by the designer. This approach works very well for lists up to a few hundred values, because recognition is easier than recall, typographic errors are eliminated, keystrokes are reduced, and fast scrolling can be accomplished on modern displays.

Imaginative designers have discovered that the lists need not be limited to scrolling columns of words, 2-dimensional graphical selectors can be very attractive. For example, a graphical selector for states in the United States might be a map on which the users click on as many of the states as they wish. Numeric values are nicely represented with some form of a slider in which the user can specify:

- a point,
- a range that is less than a given value,
- a range that is greater than a given value, or
- upper and lower bounds for values.

For example, to search for books published between 1968 and 1975, the users simply mark the desired range on a time line. When the list of numbers or names for selection grows very long, either keyboard entry or some hierarchical menu approach is needed.

### 3.1 Boolean expressions based on aggregation/generalization hierarchies

Our hypertext version of the Guide to Opportunities in Volunteer Archaeology (GOVA) was a success with Smithsonian Institution visitors, because it allowed them to explore possible dig sites by following links (Shneiderman et al., 1989). Museum visitors could touch a region on a world map and then see a close-up map with dig sites labeled in blue letters that could be touched to get

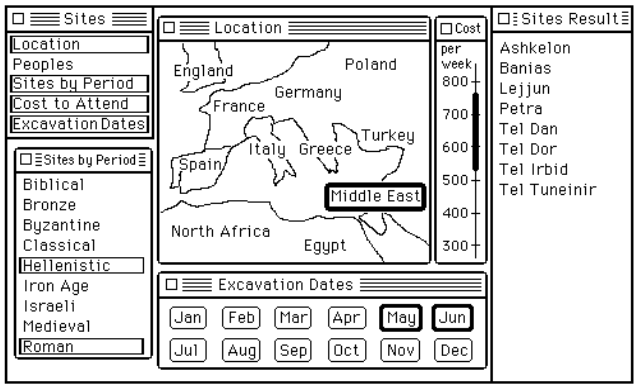


Figure 1: Graphical query for GOVA found 8 dig sites that were Hellenistic or Roman in the Middle East during May or June for $530 to $760 per week.

an article about the dig site. Other articles discussed periods of history or special projects such as underwater archaeological explorations. This low cognitive load approach coupled with complete reversibility of actions and the touchscreen interface made for a low-risk and easy-to-browse environment.

However, to satisfy queries such as: Give me all the Hellenistic or Roman dig sites in the Middle East that accept volunteers in May or June? would be difficult and time consuming since dozens of relevant articles would have to be located and read. Responding to this need, an alternate query facility was implemented.

A linear keyword-oriented search language is a typical solution, but it hardly seemed possible to teach museum visitors the use of such a language. Our approach was to turn the map items into set selectors (instead of merely links), so that the users could select several world regions and get the union of the possibilities, without ever thinking of the OR operator or typing field names and values (Figure 1). Similarly, the 12 months of the year were laid out as a sequence of buttons and the users could select as many as they wished. The months selected are ORed together and then ANDed with the world regions. Periods of history were specified by a scrolling list, and the cost was shown by a vertical bar on which users could select ranges (for example $400 to $600/week). As each selection was made, the list of matching dig sites was immediately shown on the display. This progressive refinement approach was very much appreciated by users, since they could immediately see the impact of adding or deleting a selected value. Since the permitted combinations were conjuncts of disjuncts (ANDing over ORed groups), some queries could not be constructed (for example, selecting dig sites in the Mediterranean region in July OR the Middle East in August).

Empirical testing was conducted with 16 subjects in a counterbalanced-ordering within-subjects design that compared this graphical approach to a linear keyword approach (Weiland & Shneiderman, 1991). The novice users had more problems than anticipated with the graphical approach because of inadequate experience with the mouse, scrolling menus, and an unwieldy window manager. Still, error rates with the graphical interface were approximately one-tenth of what they were with the linear keyword interface, although significant time differences did not emerge. Graphical selection of attribute values, progressive refinement of queries, and immediate re-computation and re-display of the results appears to contribute many benefits. Subjective comments favored the graphical interface. Improved window management should enable shorter training and more rapid performance.

### 3.2 Filter/flow representation of boolean expressions

The desire for full boolean expressions, including nested parentheses and NOT operators, led us towards novel metaphors for query specification.

Venn diagrams (Michard, 1982) and decision tables (Greene et al., 1990) have been used, but these both get clumsy as query complexity increases. We sought to support arbitrarily complex boolean expressions with a graphical specification. Our approach was to apply the metaphor of water flowing from left to right through a series of pipes and filters, where each filter lets through only the appropriate documents and the pipe layout indicates relationships of AND or OR.

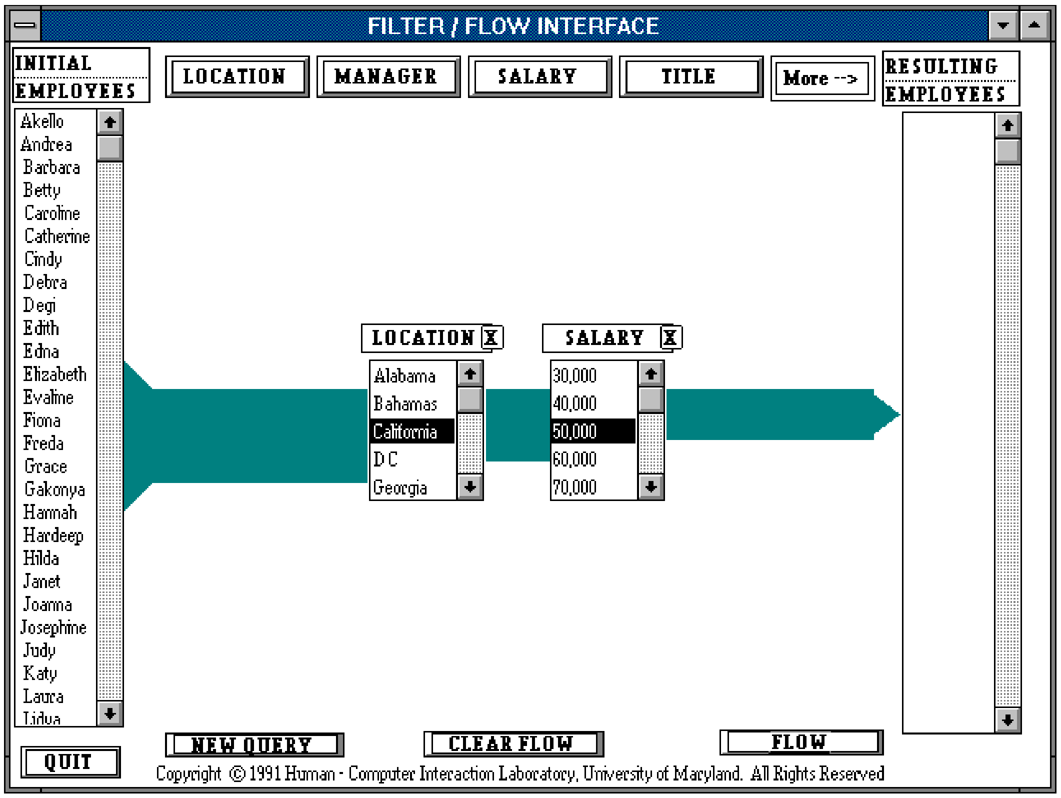
ANDs are shown as a linear sequence of filters, suggesting the successive application of required criteria (Figure 2). As the flow passes through each filter the flow is reduced and the visual feedback shows a narrower bluish stream of water. ORs are shown two ways: within an attribute, multiple values can be selected in a single filter; and across multiple attributes, filters are arranged in parallel paths (Figure 3). When the parallel paths converge the width of the flow reflects the size of the union of the document sets.

Negation was handled by a NOT operator that, when selected, inverts all currently selected items in a filter. For example, if California and Georgia were selected and then the NOT operator was chosen, those two states would become deselected and all the other states would become selected. Finally, clusters of filters and pipes can be made into a single labeled filter. This ensures that the full query can be shown on the display at once and allows clusters to be saved in a library for later reuse.

We believe that this approach can help novices and intermittent users in specifying complex boolean expressions and learning boolean concepts. A usability study is being conducted.

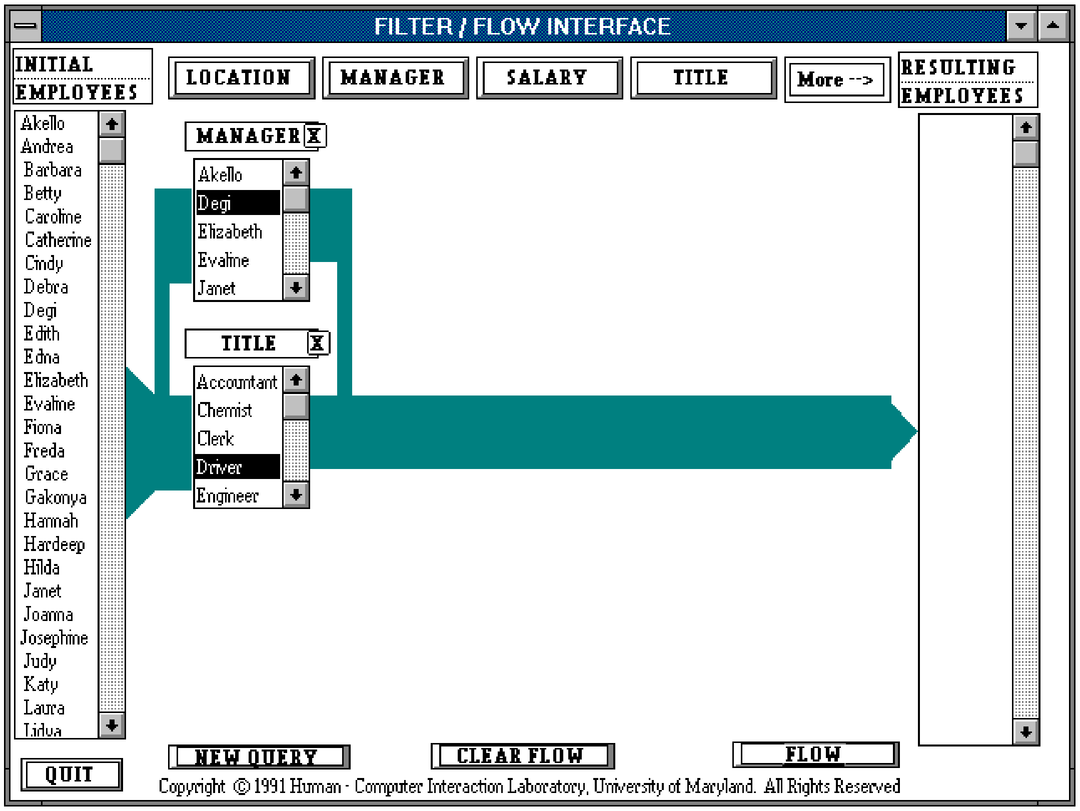
## 4. DYNAMIC QUERIES

The results of the graphical boolean query methods described in Section 3 are merely a list of items. This traditional approach is appropriate in many problem solving tasks, but we found that displaying the results in a graphical manner was an advantage in some situations. For example, in GOVA, if the user had selected July or August and costs of less than $600 per week, it would be nice to show the set of dig sites by bright yellow spots on the world map. Then users could click on the yellow spots to get the full information on the dig site. If there were only a few dig sites that satisfied this query, users would discover this immediately and could move the slider to a slightly higher cost producing more yellow spots in the desired geographic regions.
Other geographic applications emerge naturally. A college selection tool could be built based on sliders for location, size, cost, and male/female ratios. A system for real estate brokers and their clients could locate homes by price, number of bedrooms, maintenance costs, quality of schools, etc. (Figure 4). Another



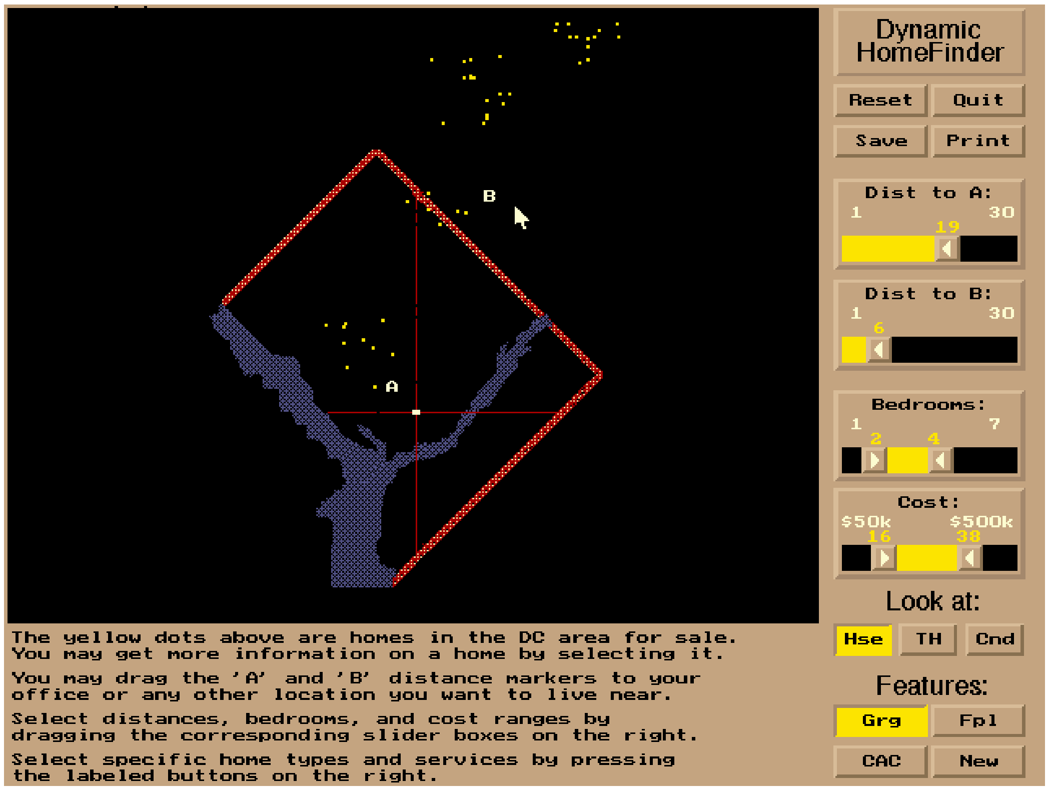
Copyright © 1991 Human - Computer Interaction Laboratory, University of Maryland. All Rights Reserved

Figure 2: Filter/flow representation of boolean AND query showing (Location = California) AND (Salary = 50,000). Implemented in Toolbook by Degi Young



Copyright © 1991 Human - Computer Interaction Laboratory, University of Maryland. All Rights Reserved

Figure 3: Filter/flow representation of boolean OR query showing (Manager = Deji) OR (Title = Driver). Implemented in Toolbook by Degi Young.



The yellow dots above are homes in the DC area for sale. You may get more information on a home by selecting it.

You may drag the 'A' and 'B' distance markers to your office or any other location you want to live near.

Select distances, bedrooms, and cost ranges by dragging the corresponding slider boxes on the right.

Select specific home types and services by pressing the labeled buttons on the right.

Figure 4: Dynamic query of D. C. HomeFinder showing sliders to select house by distance, cost, and number of bedrooms. Implemented on an IBM PC by Christopher Williamson.

geographic application would be to highlight states of the United States that satisfied values such as per capita income, air quality, employment, housing costs, etc. to help users choose a potential state to live in (Figure 5).

Other applications also seem attractive when there is natural 2-dimensional background to show search results: calendars, building layouts, circuit diagrams, or airplane seating. Imagine a chemical table of elements with a set of sliders for melting point, specific heat, ionization energy, or other properties (Figure 6). As the sliders are moved, the appropriate chemicals are high-lighted and students can refine their intuitions about the relation-ships among these properties and the atomic number or position in the table. Concert seat selection might by dynamic query: after indicating the number of seats you need together, you could move the price slider to see how spending more would bring you closer to the stage. Scientific applications seem abundant: imagine a star map with sliders for attributes of the stars, or DNA chains with selectors for specific sequences. Sociological data exploration would highlight individuals satisfying a range of attributes such as economic status, family size, education level, age, etc.

In addition to the sense of power and fun in dragging the sliders, the dynamic queries offer a unique capacity for finding cutpoints in the data when the number of satisfying records moves from a few to many. For example, in the real estate database it is useful to discover that moving from $80,000 to $85,000 might double the number of available homes but that moving up to $90,000 hardly makes any difference. Another cutpoint exploration benefit is to find that the minimum price for 3 bedroom houses is $65,000. Outliers are also located easily with dynamic queries; users can see that all low-priced houses are located in suburban neighborhoods but then discover that there are one or two bargains in the downtown sections.

Dynamic queries might be called direct manipulation queries, since they share the same concepts of visual display of actions (the sliders or buttons) and objects (the query results in the task domain display); the use of rapid, incremental, and reversible

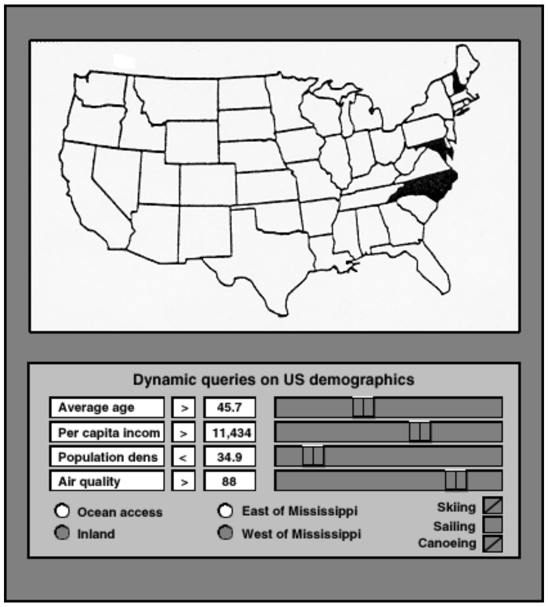
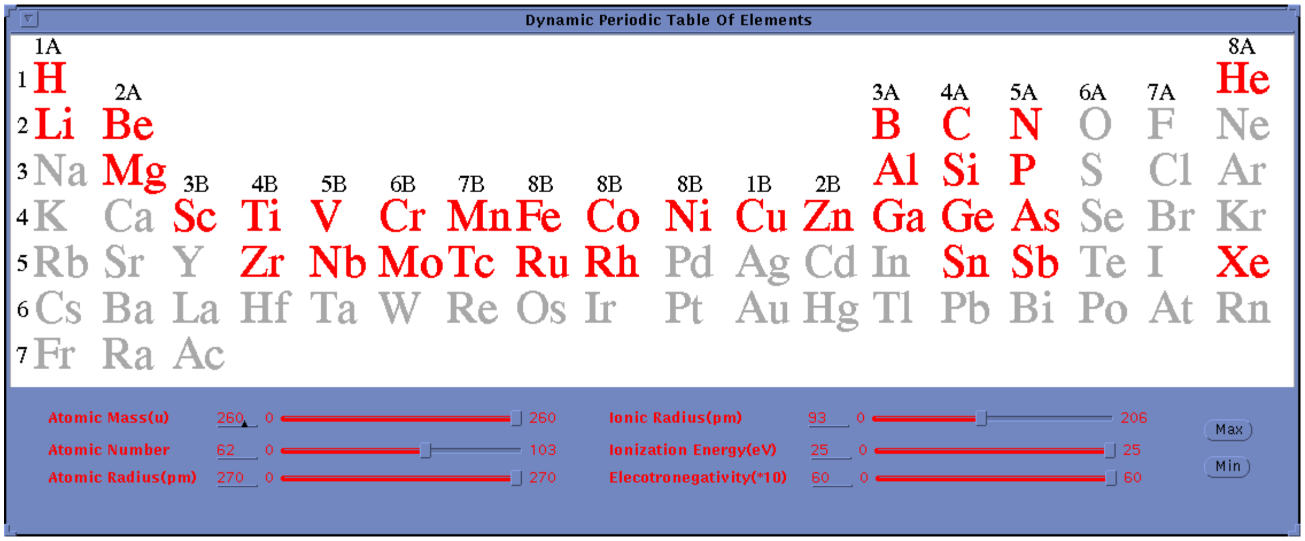


Figure 5: Dynamic query mock-up of US demographic search using sliders for age, income, air quality, etc. with results shown by highlighting of states.

actions (sliders); and the immediate display of feedback.  The


Dynamic Periodic Table Of Elements
1A
2A
3B
4B
5B
6B
7B
8B
8B
8B
1B
2B
3A
4A
5A
6A
7A
8A
1 H He
2 Li Be B C N O F Ne
3 Na Mg Al Si P S Cl Ar
4 K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
5 Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe
6 Cs Ba La Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn
7 Fr Ra Ac
Max
Min

Atomic Mass(u) 260 0 260
Atomic Number 62 0 103
Atomic Radius(pm) 270 0 270
Ionic Radius(pm) 93 0 206
Ionization Energy(eV) 25 0 25
Elecotronegativity(*10) 60 0 60

Figure 6: Dynamic query of periodical table of elements with color coding (shown here by gray shades) and sliders for atomic mass, ionization energy, electronegativity, etc.  Implemented on the Sun SparcStation 1+ in OPENLOOK by Christopher Ahlberg.

benefits of no error messages and the encouragement of exploration are common to both concepts.

These dynamic queries can reveal global properties as well as assist users in answering specific questions.  As the database grows it will be more difficult to update the display fast enough and specialized data structures or parallel computation may be required.  These dynamic queries have been attracting much attention in our lab, although manyuser interface problems remain, for example how to:

- select a set of sliders from a large set of attributes,
- specify ranges such as greater than, less than, or greater than and less than,
- deal with boolean combinations of slider settings
- choose among highlighting by color, points or light, regions, blinking, etc.
- cope with thousands of points by zooming

Usability studies are being conducted.

## 5. TREE-MAPS

In exploring visual presentations for common information, designers often have to deal with hierarchical structures such as tables of contents, menu structures, organization charts, the Dewey Decimal System, stock portfolios, or computer programs.  A variety of tree diagram formats have been developed and there are numerous algorithms for displaying partial diagrams that have traversal mechanisms to show remaining portions of the diagram.  Our goal was to show the entire tree structure on the display at one time (no scrolling) by using every pixel as part of a space-filling representation of trees (Shneiderman, 1991; Johnson & Shneiderman, 1991).

We were confronted with the problem of an 80 megabyte computer disk filling up and having to find large files as candidates for deletion.  With 14 user folders at the root directory and a total of 3200 files in 400 folders at 6 levels, this was a challenging task.  This problem led to the development of a novel 2-dimensional space filling representation of tree structures that shows files with an area proportional to their size (Figure 7).  Thus with a few moments of observation it is possible to find several large candidate files, and by moving the cursor on to these files, the users can get a pop-up window that reveals the file and path name plus other attributes.  Area can be used to show size, and color can be used to show other file properties such as date of creation, security status, file type, etc.

Tree-maps are attractive for applications such as stock portfolio analysis where industry groups are decomposed into specific stocks, and then specific purchases.  Area could indicate amount of money invested and then color might indicate whether the owner was in the black (making a profit) or in the red (suffering a loss).  Similarly, tree-maps might be used to show library holdings organized by Dewey Decimal numbers to cluster biology, chemistry, physics, etc. books.  Area could indicate the number of books on each topic and color might indicate frequency of use.

It takes a few minutes for new users to understand the tree-map

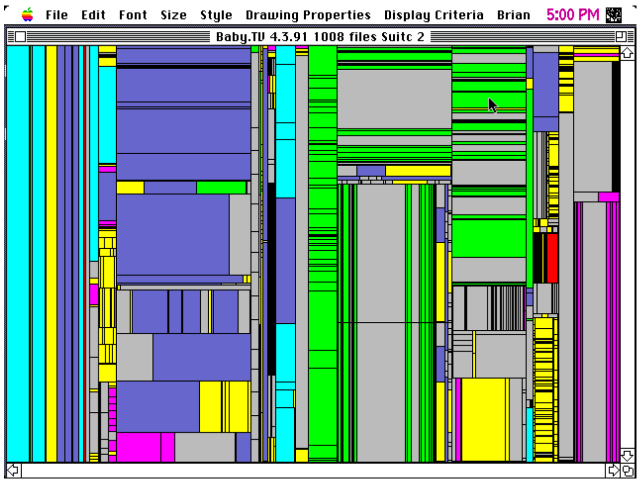


Figure 7: Tree-map showing 1008 files where each file size is represented by the proportional area and color coding (shown here by gray shades) to indicate file type.  Implemented on an Apple Macintosh by Brian Johnson.

layout, but once they do, they have unusual powers to cruise rapidly from one subtree to another, crossing many levels of the hierarchy. Our first usability test will be with experienced UNIX users browsing a large multi-level hierarchy of files.

## 6. CONCLUSIONS

Our research has led us to some novel approaches to presenting information in a more visual manner. These ideas are attractive because they present more information rapidly and allow for more rapid exploration. To be fully effective some of these approaches require novel data structures, high resolution color displays, fast data retrieval, specialized data structures, parallel computation, and some training. We are in the process of collecting the empirical data to refine our intuitions and our designs, so these approaches will certainly be modified, but the preliminary studies and our excitement compels me to share these ideas and seek feedback from the community of information scientists. However, we believe that these ideas represent only the beginning steps towards more effective visual interfaces to support information exploration by novices and experts.

## ACKNOWLEDGEMENTS

I gratefully acknowledge support for research from Apple, GE Information Services, NCR, Sun Microsystems. I appreciate the creative work of researchers at the Human-Computer Interaction Laboratory at the University of Maryland, especially the students whose work I have described: William Weiland, Degi Young, Christopher Ahlberg, Chris Williamson, and Brian Johnson. Helpful comments for this manuscript were provided by Rick Chimera.

This article is an early draft of part of a new chapter for Designing the User Interface: Second Edition, to be published by Addison-Wesley Publishing Company.

## NOTES

Christine L.Borgman, 1986. “Why are online catalogs hard to Use? Lessons learned from information-retrieval studies”, Journal of the American Society for Information Science 37, 6, 387-400.

S.L. Greene, S.J. Devlin, P.E. Cannata and L.M.Gomez, 1990. “No IFs, ANDs, or ORs: A study of database querying”, International Journal of Man-Machine Studies 32, (March 1990), 303-326.

M. Jarke and Y.Vassiliou, 1986. “A framework for choosing a database query language”, ACM Computing Surveys 11, 3, 313-340.

Brian Johnson, and Ben Shneiderman, 1991. “Tree-maps: a space-filling approach to the visualization of hierarchical information structures”, Proc. IEEE Visualization’91, IEEE, Piscataway, NJ.

H. J. Kim, H. F. Korth and A. Silberschatz, 1988. “PICASSO: A graphical query language”, Software: Practice and Experience 18, 3, 169-203.

Gary Marchionini and Ben Shneiderman, 1988. “Finding facts and browsing knowledge in hypertext systems”, IEEE Computer 21, 1, (January 1988), 70-80.

Leo Mark, 1989. “A graphical query language for the binary relationship model”, Information Systems 14, 3, 231-246.

A. Michard, 1982. “A new database query language for non-professional users: Design principles and ergonomic evaluation”, Behavioral and Information Technology 1, 3, (July-September 1982), 279-288.

Phyllis Reisner, 1988. “Query languages, In Helander”, Martin (Editor), Handbook of Human-Computer Interaction, North-Holland, Amsterdam, The Netherlands, 257-280.

N. Roussopoulos and D. Leifker,, “An introduction to PSQL: A Pictorial Structured Query Language”, 1984 IEEE Workshop on Visual Languages, IEEE Computer Society Press, Washington, DC.

G. Salton, 1989. Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer, Addison-Wesley, Reading, MA.

M. E. Senko, 1977. “DIAM II and FORAL LP: Making pointed queries with light pen”, Proc. IFIP Congress 77, North-Holland Publishers, Amsterdam, The Netherlands.

Ben Shneiderman, 1991. “Tree visualization with tree-maps: A 2-d space-filling approach”, ACM Transactions on Graphics, (To appear, 1991).

Ben Shneiderman, Dorothy Brethauer, Catherine Plaisant and Richard Potter, 1989. “Three evaluations of museum installations of a hypertext system”, Journal of the American Society for Information Science, (May 1989), 172-182.

Ben Shneiderman and Greg Kearsley, 1989. Hypertext Hands-On!, Addison-Wesley Publ., Reading, MA.

Dagobert Soergel, 1985. Organizing Information: Principles of Data Base and Retrieval Systems, Academic Press, Inc., Orlando, FL.

Pat Stevens and Ben Shneiderman, 1981. “Exploratory Research on Training Aids for Naive Users of Interactive Systems”, Proc. American Society for Information Science 1981, 65-67.

William J. Weiland and Ben Shneiderman, 1991. “A graphical query interface based on aggregation/generalization hierarchies”, Department of Computer Science Technical Report CS-TR-2702, University of Maryland, College Park, MD.

H.K.T. Wong and I. Kuo, 1982. “GUIDE: Graphical User Interface for Database Exploration”, Proceedings of the 8th Very Large Databases Conference.