Published in *Human Factors in Computing Systems CHI '86 Conference Proceedings*, 16-23. 1986.

Generalized Fisheye Views

George W. Furnas

Bell Communications Research
435 South St.
Morristown, New Jersey
201-829-4289

Abstract

In many contexts, humans often represent their own "neighborhood" in great detail, yet only major landmarks further away. This suggests that such views ("fisheye views") might be useful for the computer display of large information structures like programs, data bases, online text, etc. This paper explores fisheye views presenting, in turn, naturalistic studies, a general formalism, a specific instantiation, a resulting computer program, example displays and an evaluation.

1. Introduction.

Computer programs, structured data bases, organizational charts, on-line text, menu access systems and maps -- users are forced to view all of these potentially huge structures through windows sometimes as small as a 24x80 character video display. The problem is that there is too much to show, ranging from local details to global structural information. Currently the most common viewing interface is simply a small window for looking into the structure, centered at some point. For example, a simple editor window might show a line in a program and a dozen consecutive lines before and after it. A menu based retrieval system might show the set of choices available at the current node. The user navigates through the structure by moving the window around (by scrolling, traversing arcs, etc). As a result it is easy to get lost, i.e., to find oneself in some incomprehensible wrong place with little idea how to get to the right one (e.g., [1]). Presumably this happens because such views have little information about the global structure, and where the current view fits in. Several techniques have arisen to try to deal with this problem, most notably variants on a Zoom Lens analogy -- making available both a global and detailed view of a structure, either side by side, as with paper road maps, or in sequence. (One of the earliest examples was in Englebart's Knowledge Augmentation Workshop [2].)

We have been exploring a different viewing strategy, based on an analogy to a very wide angle, or "fisheye", lens. Such a lens can show places nearby in great detail while still showing the whole world -- simply by showing the more remote regions in successively less detail. An instructive caricature of this appears in the "New Yorker's View of the United states", a poster by Steinberg and now much imitated for other cities. In the poster, midtown Manhattan is shown street by street. To the west, New Jersey is a patch of color on the other side of a blue-grey ribbon labeled "Hudson." The rest of the country is reduced to a few principal landmarks (Chicago, the Rocky Mountains, California, etc.) disappearing in the distance. While this representation is certainly a distorted view of the U.S., it is a manageable abbreviation in which the most important features of the New Yorker's world are preserved. The view allows the New Yorker to answer local questions like, "Where is the closest mail box?", but also more global questions like "To ski in the Rocky Mountains, does it make more sense to connect through LA or Chicago?". If New Yorkers' fisheye views allow them to answer such questions, perhaps analogous views would be useful in computer interfaces.

The fundamental motivation of a fisheye strategy is to provide a balance of local detail and global context. Local detail is needed for the local interactions with a structure, whether that means finding the nearest mailbox in midtown or editing a particular line of a large program. The global context is needed to tell the user what other parts of the structure exist and where they are (e.g., the Rockies are out west, beyond Chicago but before LA; there is an *if* construct above the *else* construct currently being edited). Global information may also be important even in the mere interpretation of local detail (e.g., the meaning of the *else* statement in fact depends on the content of the associated, but remote, *if*() statement).

By looking for an analogy to the New Yorker's abbreviated view, i.e., a trade-off of detail with distance, it is possible to consider fisheye views in a suprising number of domains. In this paper we look at naturally occurring fisheye views, and then turn to the question of creating them for computer interfaces.

2. Naturally occurring fisheye views.

We have undertaken studies of naturally occurring fisheye views for several reasons. At one level, as cognitive psychologists, we were simply interested in how humans represent large structures in their heads. More relevant

 ACM 0-89791-180-6/86/0400 - 0016

here, we thought that if fisheye views were ubiquitous it might be because they were "naturally" useful in human interactions, and might therefore make effective interfaces. In addition we hoped to learn more about what such views might look like, anticipating that findings might suggest features for fisheye interface design.

We conducted several experiments using a simple production paradigm. Subjects were told to imagine that a young child of a newly immigrated neighbor family had asked to be told about X's (where X's are *States*, *Presidents*, *Events in History,...).* The subject's task was simply to list 10 examples of category X that they thought the child should know about. The empirical fisheye conjecture is that, to be cited, exemplars would have to be either of great *a priori* importance or "close to home". Such fisheye subsets were indeed listed. For *states*, subjects in both New Jersey and Texas mention states of major *a priori* importance (e.g., New York and California), and then show geographic bias (e.g, Texans listed Arkansas, New Jerseyans listed Connecticut). Similarly, subjects listed *presidents* that were either pre-eminent (e.g., Washington, Lincoln) or recent (e.g., Carter, Reagan).

Using other techniques, we have found that people in a large corporation know a fisheye subset of the management structure. Employees know local department heads, but only the Vice Presidents of remote divisions.

We have also looked at academicians' views of the academic world and found that in similarity ratings, the disciplines near one's own loom extra large: an experimental psychologist will judge the pair "management" and "marketing" more similar than "experimental psychology" and "psychiatry," but people in the business school will make the opposite evaluation.

By examining the patterns of stories in 12 newspapers from three geographic regions, we found news editors have evolved a fisheye editorial strategy. The papers will contain local news stories (e.g., a continuing local garbage strike) and only more distant ones that are compensatingly greater importance (e.g., the bombing of the U.S. embassy in Beirut).

While there may be many interesting processes behind these results, we draw the conclusion that many naturally occurring views of the world do exhibit a fisheye character. This suggests that apropriately generalized fisheye views might provide a good viewing interface for large structures.

3. Formalizing generalized fisheye views.

In order to apply the fisheye concept to interface design, the idea must be clarified formally. Fisheye views are an example of a more basic strategy for the display of a large structures. This basic strategy uses a "Degree of Interest" (DOI) function which assigns to each point in the structure, a number telling how interested the user is in seeing that point, given the current task. A display of any desired size, *n*, can then be made by simply showing the *n* "most interesting" points, as indicated by the DOI function.

At this general level, successful display would depend on discovering appropriate DOI functions. One might, for example, seek to understand and decompose them in terms of more primitive aspects of the structure. Generalized fisheye views arise by decomposing the DOI into two components: *a priori* importance and distance. In its simplest, additive form the generalized fisheye Degree of Interest function is,

$$DOI_{fisheye}(x|.=y) = API(x) - D(x,y)$$

where $DOI_{fisheye}$ is, according to the fisheye model, the user's Degree of Interest in a point, *x*, given that the current point of focus is *y*, *API(x)* is the global *A Priori* Importance of *x* and *D(x,y)* is the Distance between *x* and the current point *y*. That is, the interest increases with *a priori* importance and decreases with distance. (Presumably the usefulness of a DOI so defined will depend at least on the suitable definition of distance and *a priori* importance.)[1]

This simple formulation allows fisheye views to be defined in any sort of structure where the necessary components can be defined. Rooted tree structures will be illustrated as a straightforward example that is quite different from the New Yorker's map. They are of particular interest since many large structures on computers are trees: structured programming languages (e.g., like LISP, PASCAL and C), hierarchically organized text (e.g., manuals, legal codes), various highly structured scientific and technological knowledge domains (e.g., biological taxonomies), hierarchical file systems (e.g., UNIX), corporate management structures, hierarchical menu access systems, etc. The definition of fisheye DOI functions for trees would thus allow fisheye displays for these structures.

To define the necessary components for a tree, consider that *D(x,y)* has as a natural instantiation as $d_{tree}(x,y)$, the path length distance between *x* and *y* in the tree. Similarly *API(x)* can become $-d_{tree}(x,root)$, the distance of x from the root, under the approximating assumption that points at levels closer to the root are intrinisically more important. (The minus sign simply gives the correct "sense" to the arithmetic term -- *further* from the root means *less* importance.) This gives,

1 The strategy of using a DOI with a threshold to abbreviate a display requires only ordinal properties. Thus, in the current discussion, the DOI function is not required to be positive. In fact, the example given below has only negative values. Extensions of this simple DOI strategy can depend on more than ordinal relationships, but not discussed here.

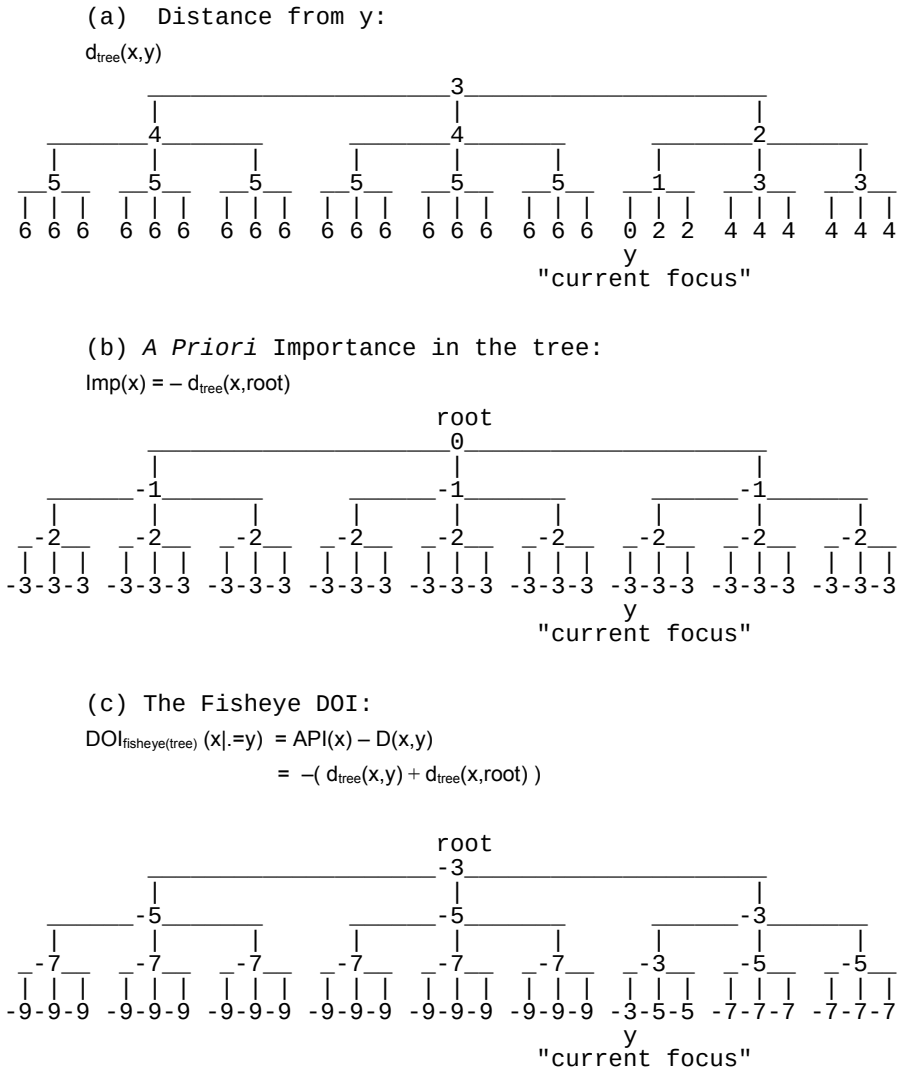

Figure 1. Distance, *A Priori* Importance and the Fisheye DOI for a rooted tree.

$DOI_{fisheye(tree)}(x|.=y) = -(d_{tree}(x,y) + d_{tree}(x,root))$

Figure 1 illustrates these two components, and how they add together point by point to form the fisheye DOI function for the tree. In the resulting DOI function, an arithmetically larger number means the corresponding point is more interesting for interactions focused at *y*. Thus, the points with *DOI*=–3 form the most "interesting" subset, those with *DOI*=–5 form the next most "interesting" subset, etc.

Thus by choosing a threshold, *k*, and only displaying those points with $DOI(x) \geq k$, one can obtain fisheye views of different sizes. For example, letting *k*=–3 selects only the most interesting subset which, by the fisheye DOI, turns out to be the direct ancestral lineage between *y* and the root of the tree ("Zero-order fisheye view", see figure 2a, and figure 1). This subset is "most interesting" basically because points on that lineage increase in *a priori* importance in exact compensation for their corresponding increase in distance. If the display threshold is lowered to include the next most interesting subset ("First-order fisheye view", at *k*=–5, Figure 2b), the ancestral line and its "siblings" are included. At the next threshold value ("Second-order fisheye view", at *k*=–7, Figure 2c) "cousins" would be added. Consistent with the original fisheye inspiration, at any choice of threshold, only higher level points (i.e., by assumption, more major features) are shown for further regions of the tree.

These views have a number of interesting properties. In a regular tree, (1) the fisheye view achieves a logarithmically compressed display of the original tree. (2) Because of the convex, nested structure of the DOI sets, there exist fast algorithms for computing such views, in time proportional to the size of the view, and not the size of the tree. (3) As the point of focus changes from *y* to some new *y'*, the change in view is easily calculated, since the whole DOI function above their common ancestor is unchanged. (4) Users may move through the structure using such fisheye views in a number of steps proportional to the log of the number of intervening leaves of the tree. These formal

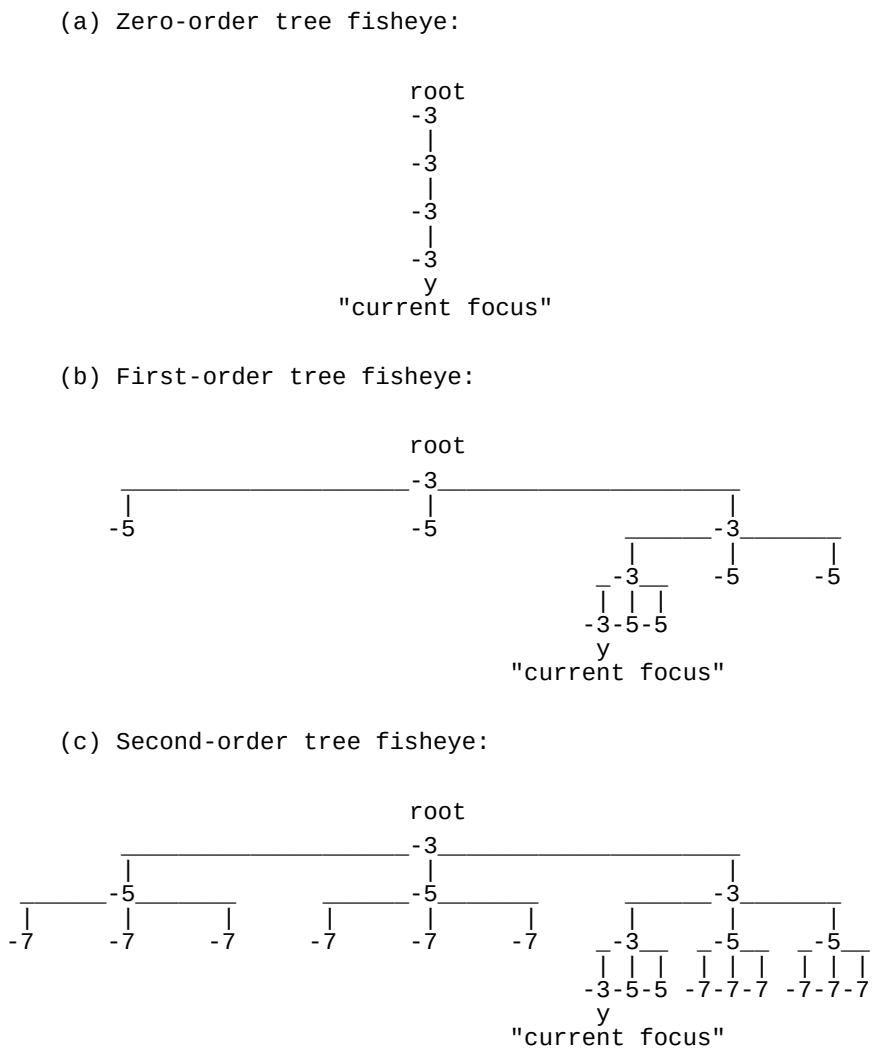

Figure 2. Zero-order, first-order and second-order fisheye views for a tree.

properties, among others, underscore the computational and interaction efficency possible with fisheye views.

4. Fisheye interfaces.

The fisheye DOI function derived for trees in the previous section was used to develop a program for making first-order fisheye views of tree structured text files. An example, showing views of a C-program, is presented in the picture, for example. Very little orienting information is available.

On the other hand, the fisheye view, seen in figure 4, shows that the programmer is at a short *for* loop, within the *e* case of a *switch* in which there are also four other cases +, -, *q*, and *default*. This switch is in the *else* block of the indicated *if* statement, within a *while* loop, in program *main()*, etc. It is conjectured that being able to see their work focus together with such contextual information will be of use to programmers working with structured code. Figure 5 compares the content of these two views. The box indicates the standard window view figures 3, 4 and 5. (This is a short calculator program which does reverse-Polish-notation integer addition and subtraction.) The simple flat window view of figure 3 shows lots of detail, some of which is not likely to be very useful when working on the indicated line (marked by ">>"). The arithmetic details of the previous *case* intrude in the top of

of figure 3 and the underlining shows the lines in the fisheye view of figure 4. The main difference is that, while both show detail at the center, some superfluous detail at the edges of the flat view has been traded for some more remote but higher-level, contextual information. Related program viewing schemes have been proposed recently for syntax-driven program editors [3] [4] [5]. These have made use almost exclusively of the distance component, whereas we also emphasize *a priori* importance. Views that are effectively first order tree fisheye views have arisen in the browsers of the SMALLTALK [6] and INTERLISP-D environments.

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

Figure 3. Standard 'flat-window' view of a C program. Line numbers are in the left margin.

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

Figure 4. A fisheye view of the C program. Line numbers are in the left margin. "..." indicates missing lines.

We conjectured that such fisheye views should be more useful, at least in for the tasks of navigating around or examining unfamiliar parts of a large file. To test this we ran an experiment in which 20 subjects were asked to perform a navigation-related task in a large unfamiliar hierarchical structure. The task was meant to compare various views' ability to support a basic cognitive operation for moving from one (undesired) location in a file to another (target) location. Specifically subjects were asked to determine the relative positions ("Which comes first?") for two different parts of the hierarchical structure, given various views of those parts. One sort of view was a 22-line standard "flat" view of the file, centered at a randomly chosen line of focus. The other sort of view was a first-order fisheye view centered at the line. Subjects received either two flat views, two fisheye views, or one of each on which to base their decision, and saw a total of 16 pairs in all. In order to prevent subjects from answering on the basis of prior knowledge, a very unfamiliar structure was used -- a botanical taxonomy of the Class of Dicotyledons, classified down to families.

We found that people were only 52% correct with two flat views, 64% correct with one fisheye and one flat view, and 75% correct with two fisheye views. That is, as expected, fisheye views are far superior. This result is most certainly simply because the fisheye shows the necessary structural information, and the fact is not lost on the subjects.

In addition to implementing fisheye views for indent structured programs of figures 3-5 and the botanical taxonomies of our experiment, we have an interactive fisheye veiwer for part of the Texas Legal Codes, text outlines,[2] a decision tree (identification key) for types of trees, a directory of telephone area codes, our corporate directory, and UNIX file hierarchy listings. All of these applications are based on the tree fisheye DOI function derived above.

5. Conclusions.

This paper has described generalized fisheye views. Fisheye views provide a balance of local detail and global context by trading off *a priori* importance against distance. They appear naturally in many human contexts and can be implemented for a wide variety of computer information structures. The formal definition presented here allows interfaces to be defined and explored in any structure where distance and some display-relevant notion of *a priori* importance can be defined. This is possible for lists, trees, acyclic directed graphs (DAG's, such as ISA networks in knowledge representations), general graphs and Euclidean spaces, among other structures.[3] It is important to remember that, unlike the geographic example which inspired the metaphor (the New Yorker's

---

[2] Fisheye views of outlines and structured text like legal codes have much in common with views generated by "outline processors", now coming onto the market place, and the early hypertext ideas of Nelson [7]

[3] We note that "*A Priori* Importance" need not be structurally defined, like "level" in a tree. It may be independently specified for each point, though often less efficient algorithms may result.

View), the underlying stuctures need not be spatial nor the "output" even graphic. For example, the stucture might be a semantic net and the output be a fisheye-structured exposition in natural language text.

Even without formal treatment, fisheye-type views can be invented simply by analogy -- trading off distance and detail. One such example, with a rather different flavor, is presented in figure 6. It is a "fisheye calendar", showing the current day in "day-at-a-time" detail, the current week in "week-at-a-time" detail and the rest of the month in "month-at-a-time" detail. The goal is to give the user needed hour-by-hour information about today, but some sense of the appointment structure for the rest of the week and month. We are currently implementing an interactive version of this calendar.[4] A number of results from our studies of natural fisheye representations suggested future work in creating views. In particular some effects were not consistent with a simple fisheye model: (1) In some cases, the sphere of local interest was somewhat

```
 1 #define DIG 40
 2 #include <stdio.h>
 3
 4 main()
 5 {
 6      int c, i, x[DIG/4], t[DIG/4], k = DIG/4, noprint = 0;
 7
 8      while((c=getchar()) != EOF){
 9           if(c >= '0' && c <= '9'){
10                x[0] = 10 * x[0] + (c-'0');
11                for(i=1;i<k;i++){
12                     x[i] =  10 * x[i]
13                          + x[i-1]/10000;
14                     x[i-1] %= 10000;
15                }
16           } else {
17                switch(c){
18                     case '+':
19                          t[0] = t[0] + x[0];
20                          for(i=1;i<k;i++){
21                               t[i] = t[i] + x[i]
22                                    + t[i-1]/10000;
23                               t[i-1] %= 10000;
24                          }
25                          t[k-1] %= 10000;
26                          break;
27                     case '-':
28                          t[0] = (t[0] + 10000)
29                               - x[0];
30                          for(i=1;i<k;i++){
31                               t[i] = (t[i] + 10000)
32                                    - x[i]
33                                    - (1 - t[i-1]/10000);
34                               t[i-1] %= 10000;
35                          }
36                          t[k-1] %= 10000;
37                          break;
38                     case 'e':
39                          for(i=0;i<k;i++) t[i] = x[i];
40                          break;
41                     case 'q':
42                          exit(0);
43                     default:
44                          noprint = 1;
45                          break;
46                }
47                if(!noprint){
48                     for(i=k - 1;t[i] <= 0 && i > 0;i--);
49                     printf("%d",t[i]);
50                     if(i > 0) {
51                          for(i-- ; i >= 0; i--){
52                               printf("%04d",t[i]);
53                          }
54                     }
55                     putchar('\n');
56                     for(i=0; i > k;i++) x[i] = 0;
57                }
58           }
59           noprint = 0;
60      }
61 }
```

[4] The layout of this calendar is very similar to some graphics work by Farrand [8].

Figure 5. Full view of the C program. Box shows lines in "flat" view. Underlines show lines in the fisheye view.

exaggerated when compared to a simple immediate fisheye tradeoff -- suggesting a similar need in display design. For example one might include just a few more local lines around the *for* loop line in figure 4. (2) Often there were cases of "multi-focus" fisheye views, as in the geographic study when the subject had lived in more than one state. In this case detail occurred at both foci and fell off at points far from either. This observation serves to remind that users might need to see detail in more than one place at a time, with a fisheye context around each. The fisheye calendar we are currently developing will explore this capability -- showing two days at higher detail, when desired. (3) Finally, there were typically additional, non-fisheye effects (e.g., human-interest newspaper stories could have almost any geographic origin). This is a good reminder that while perhaps useful, fisheye views do not capture everything. There may also have to be *ad hoc*, domain and task dependent components of any display of a large structure.

REFERENCES

[1] Robertson, G., D. McCracken and A. Newell The ZOG approach to man- machine communication, Technical Report CMU-CS-97-148, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1979.

[2] D. C. Englebart and W. K. English, A research center for augmenting human intellect, *AFIPS Conference Proceedings*, *Vol. 33*, 1968, 15ff. Also SRI-ARC Catalog item 3954.

[3] Alberga, C. N., A. L. Brown, G. B. Leeman, M. Mikelsons and M. N. Wegman, A program development tool, IBM Research Report, Computer Science Department, IBM Thomas J. Watson Research Center, Yorktown Heights, New York,1979.

[4] Horton, M. *Design* of *a Multi Language Editor*, Doctoral Thesis, U. C. Berkeley Computer Science, 1981.

[5] Mikelsons, M., IBM Research Report, Computer Science Department, IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

[6] Teslar, L, The Smalltalk Environment, *BYTE*, 6, 1981, 90-147.

[7] Nelson, T. *Computer Lib* Hugo's Book Source: Chicago, IL, 1974.

[8] Farrand, W. A. *Information Display in Interactive Design*, Doctoral Thesis, Department of Engineering, University of California at Los Angeles, 1973.

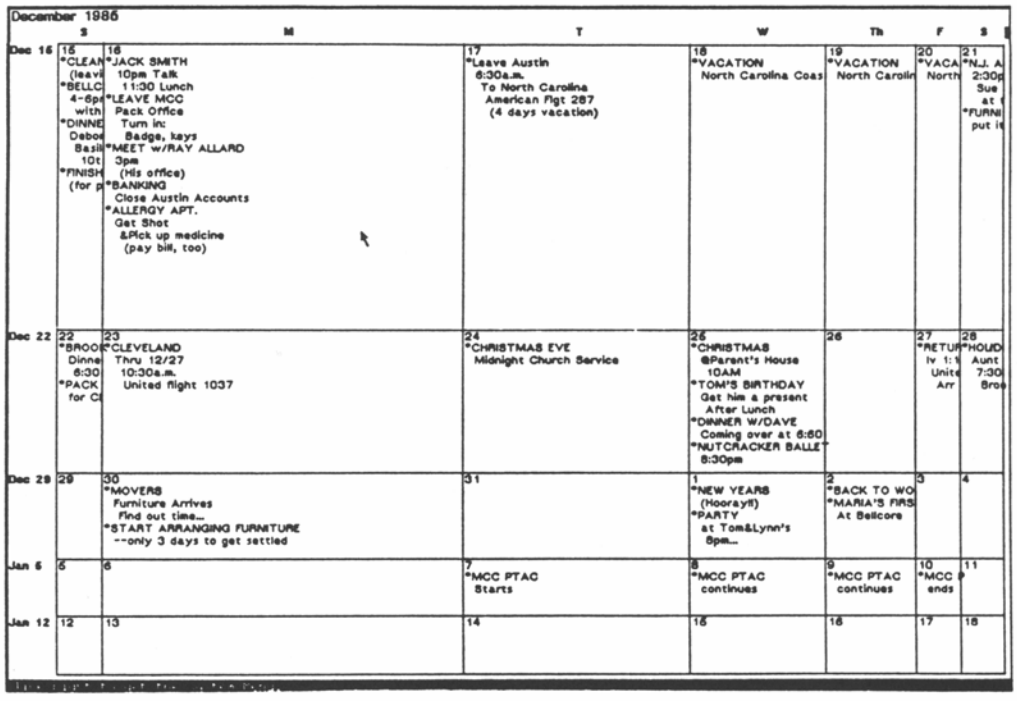

Figure 6. A Fisheye Calendar.