LensBar – Visualization for Browsing and Filtering Large Lists of Data

Toshiyuki Masui[1]

Sony Computer Science Laboratories Inc.

Abstract

We propose a simple and powerful graphical interface tool called the *LensBar*, for filtering and visualizing a large list of data. Browsing and querying are the most important techniques for information retrieval, and LensBar integrates the two techniques into a simple-looking scroll window with a slider. While it looks familiar to users of conventional graphical interface tools, its filtering and zooming mechanism offers sophisticated handling of large lists of text-oriented data.
CR Categories and Subject Descriptors: I.3.6 [Computer Graphics]: Methodology and Techniques - Interation Techniques; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval - Information Filtering; H.5.2 [Information Interfaces and Presentation]: User Interfaces - Graphical User Interfaces.
Additional Keywords: zooming interface, approximate string matching, slider, LensBar

1 INTRODUCTION

Recently, various visualization techniques for browsing large data have been proposed. Also, a number of information retrieval techniques for filtering huge data using query keywords are also proposed. In some systems, visualization and filtering are integrated and users can filter and browse the data at the same time.

Although the combination of browsing and querying is much more powerful than using each of the techniques separately, most of the existing systems suffer from the following shortcomings. First, most of the visualization techniques are far from general-purpose, and they are applicable only to special application domains. Second, special interaction techniques are usually required to effectively filter data and control the visualization result. For example, using the FilmFinder system[2], users can get the information of movie titles by providing information such as the names of actors and the year of the production, and they can control the visualized query result shown as a scattered plot on a 2D space. However, the filtering and visualization method is only valid to this particular application, and the whole design of the visualization and interaction techniques should be modified for different applications.

It is natural that handling special data requires special visualization and interaction techniques, but it seems strange that even for handling a common data structure like a text or a simple list of data, not many effective techniques for query and visualization have been proposed and used widely.

We propose a simple and powerful interface tool called the *LensBar* for filtering and visualizing a large list of data. LensBar works as an extension to a conventional scroll window or a substitute for a hierarchical menu, and it can be applicable to wide range of applications where these tools are currently used.

[1]3-14-13 Higashi-Gotanda, Shinagawa, Tokyo 141, Japan. masui@csl.sony.co.jp

2 VISUALIZATION AND INTERACTION TECHNIQUES

Our technique is based on the following strategies.

- Browsing the whole list using a precisely-controllable slider and a scroll window
- Controlling the amount of data to be displayed by keyword filtering and zooming
- Visualizing the distribution of filtered data in the background of the slider

We will describe the techniques in more detail in the following sections.

Integrating browsing and querying To browse a large list, we use a slider (scroll bar) with an accompanied scroll window, which is a very common combination in current graphical user interface systems. In addition to them, an extra text input area is attached for filtering the list of items, and only those entries that match the specified pattern are selected and displayed in the scroll window. At the same time, corresponding positions in the slider background are highlighted to show the locations of entries which match the pattern. For example, if the first entry in the list match the pattern, the top line of the slider background is highlighted. When no pattern is specified, all the lines in the slider background are highlighted, yielding a highlighted rectangle in the slider background. When a user tries to move the slider knob, it only moves on the highlighted portion. The highlighted region within the slider knob always corresponds to the displayed items in the scroll window.

When the user modifies the query pattern, string matching is immediately performed to all the entries in the list, and the display of the slider background and the scroll window also change immediately. With this *dynamic query*[18] feature, users can easily find the relation between query pattern and the distribution of data items.

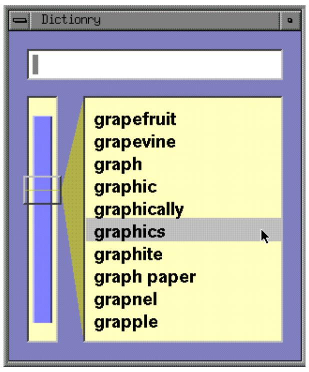


Figure 1: Initial display of the dictionary application

Dynamic approximate string matching When no entry in the list matches the specified pattern, dynamic approximate string matching[12] is performed automatically, and entries which are closest to the pattern are treated as being matched. This is usually much more convenient than giving no query result when no matching is found, and especially useful when users are not very sure of the spelling: e.g. searching for a word in a dictionary of a foreign language.

Zooming interface Users can control the amount of information to be displayed by zooming operations. Before zooming out, all the entries that match the query can be displayed by moving the slider knob, just like a conventional scrolling window. Using LensBar, a user can control the amount of items to be displayed by changing the zooming level. A DOI (degree of interest) value is dynamically assigned to each entry automatically, and only those entries whose DOI value is bigger than the zooming level are selected to be displayed. For example, when the zooming level is set to a large value, items with small DOI values disappear to produce a zooming out effect.

Zooming interface is essential in 3D interaction systems, and it is also proved to be effective in 2D GUI[3, 15]. Although zooming interfaces for handling a (1D) large list is not as widely used as handling 3D/2D objects, it is convenient to control the amount of displayed data smoothly.

Precise control of the slider Using a conventional slider, precise scrolling of a very large data is difficult, since the resolution of the display and the pointing device is low. To enable precise control, conventional scroll bars sometimes have up and down arrows around the knob or at the top and bottom of the bar. Other techniques for fine control of the slider have also been proposed[1, 13], and we adopted the technique described in [13]. When a user clicks the mouse on the knob of the slider, he can move the knob directly to any place on the bar, just like a conventional slider. When the user clicks the mouse on the scroll bar other than on the knob, the knob will move gradually to the mouse cursor, in a speed proportional to the distance between the mouse position and the knob, and users can control the moving speed of the knob easily.

3 EXAMPLES

It would be best to use examples to show how each of the techniques described in the previous section actually works. Since LensBar is a general-purpose tool for visualizing and filtering a large list, we introduce various examples to show how the technique is applicable to different applications.

3.1 Example 1: Searching Words In a Dictionary

First, we show the behavior of LensBar using a simple dictionary application.

Figure 1 shows the initial display of a dictionary application using LensBar. In the left side, a slider with a transparent knob is displayed. In the background of the slider area, a highlighted rectangle is displayed, showing that all the entries in the dictionary are currently selected for display. In Figure 1, only ten words which correspond to the position of the knob are shown in the scrollable area at the right. The highlighted line at the center of the knob shows the position and the size of the ten words in the whole list.

The dictionary contains more than 30,000 words. The user can drag the knob to any position of the dictionary, and he can click other place in the slider to move the knob precisely (Figure 2). He can also drag the scroll window by clicking the mouse in the window and dragging it vertically.

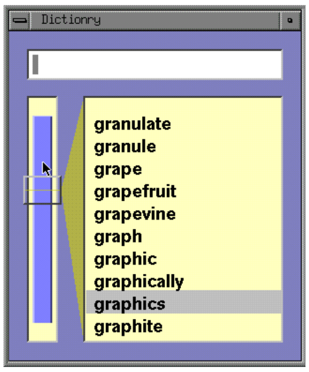


Figure 2: Scrolling the window

DOI value	Entry
...	...
1	granule
2	grape
1	grapefruit
3	grapevine
1	graph
2	graphic
1	graphically
20	graphics
1	graphite
2	graph paper
...	...

Table 1: Assignment of DOI values

In this dictionary application, DOI value is assigned to each line, as shown in Table 1. Since "`graphics`" is currently selected, it has the largest DOI value. When the user clicks the mouse on a word and drags it to the left, the zooming level changes accordingly, and the number of words to be displayed are gradually reduced and wider range of words appear in the scroll window. For example, when the zooming level is 2.0, every two word around "`graphics`" is displayed. In Figure 3, every 256 word in the dictionary is displayed (zooming level is around 9), and in Figure 4, every 4096 word is displayed. Hidden entries are shown as gray lines between words. At the same time, the background of the slider changes to show the distribution of the selected words. As the range of the displayed words becomes larger, the slider knob also becomes larger to cover the area.

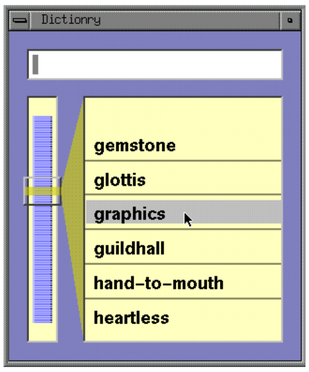


Figure 3: Zooming out to show every 256 word

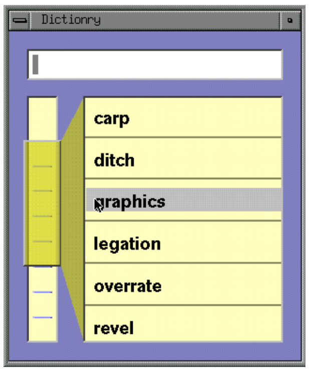


Figure 4: Zooming out more to show every 4096 word

The zooming level is determined by the horizontal movement of the mouse. Since the scrolling window follows the vertical movement of the mouse, the mouse cursor is always on the line of the same word ("**graphics**", in this case) during the zooming operation, and when the user moves the mouse cursor back to the previous position, the system goes exactly back to the previous state. This continuous and reversible characteristic of zooming interface[14] helps users recover from erroneous operations easily.

When the user enters a pattern string, dynamic approximate pattern matching is performed and only those entries that match the pattern are selected for display. Figure 5 shows the display after the

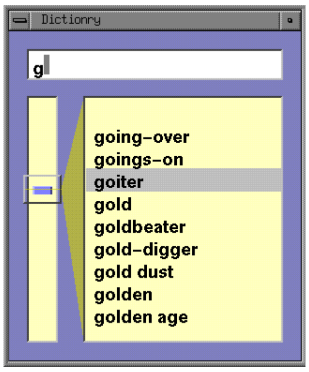


Figure 5: Filtering by "**g**"

user specified "**g**" as the pattern string. The whole list is filtered and only those words which begin with "**g**" are selected and displayed.

A space character ("␣") is used as a wildcard character, just like the pattern "**.***" used in regular expressions. So when "␣**q**" is specified as the pattern, all the words which include the letter "**q**" are selected (Figure 6). Since those words are scattered in the dictionary, the highlighted area in the slider looks like a comb.

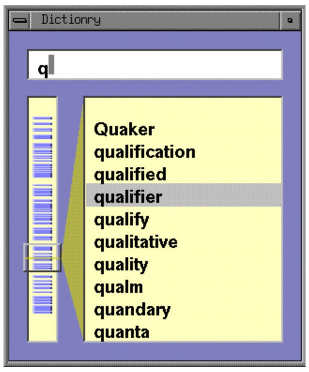


Figure 6: Filtering by "␣**q**"

Let's consider finding a word with difficult spelling like "**Pithecanthrop**
Figure 7 shows the display of LensBar after a user typed a wrong spelling "**pite**", showing that only two words in the dictionary begins with "**pite**".

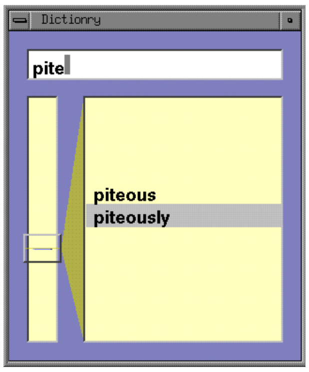


Figure 7: Typing "**pite**"

At this moment, the user can tell that "**pite**..." is not the right spelling for "**Pithecantropus**", but he can continue giving more letters as the pattern.

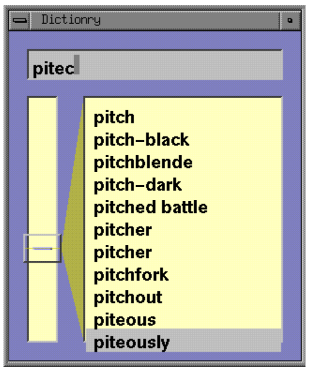


Figure 8: Typing "**pitec**"

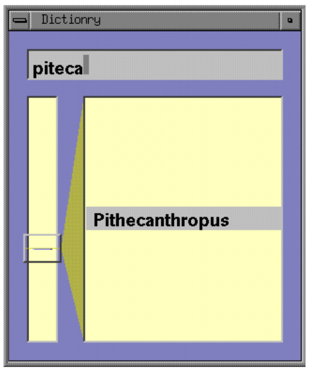


Figure 9: “**`Pithecanthropus`**” found from “**`piteca`**”

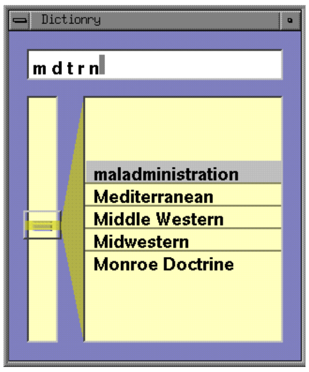
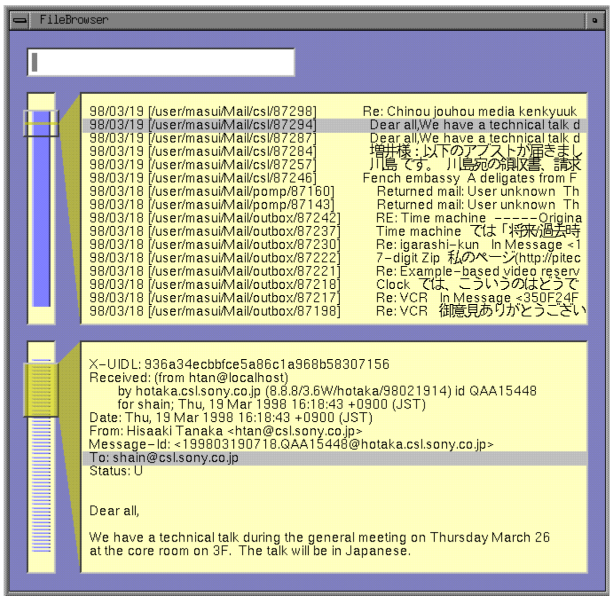


Figure 10: Typing “**`m␣d␣t␣r␣n`**”

When the user types “**`c`**” next, the string matcher notices that there is no word in the dictionary that begin with “**`pitec`**”, and all the words that match the pattern “**`pitec`**” with one error are searched and displayed. Many words including “**`pitch`**” and “**`piteous`**” match “**`pitec`**” with one error, and they are displayed in the scroll window. The text input area becomes darker, showing that no exact matching was found. When the user types “**`a`**” next, words like “**`pitch`**” disappear because they do not match “**`piteca`**” with one error, and only “**`Pithecanthropus`**” becomes visible. In this way, by dynamically changing the parameter of the approximate string matching algorithm and showing the words closest to the pattern, users can always see the candidates closest to the given pattern and have better chance of finding the desired information.

When the user is not sure of the spelling, he can explicitly use wildcard characters instead of specifying uncertain letters. Figure 10 shows the situation where a user tries to find “**`Mediterranean`**”, and specifies only small number of characters that he thinks must exist in that word. Since there are only five words in the dictionary which have “**`m`**”, “**`d`**”, “**`t`**”, “**`r`**” and “**`n`**” in its spelling, the user can easily find the desired word in the list.



```
98/03/19 [/user/masui/Mail/csl/87298]       Re: Chinou jouhou media kenkyuuk
98/03/19 [/user/masui/Mail/csl/87294]        Dear all,We have a technical talk d
98/03/19 [/user/masui/Mail/csl/87287]        Dear all,We have a technical talk d
98/03/19 [/user/masui/Mail/csl/87284]        増井様：以下のアブストが届きまし
98/03/19 [/user/masui/Mail/csl/87257]        川島です。 川島宛の領収書、請求
98/03/19 [/user/masui/Mail/csl/87246]       Fench embassy  A deligates from F
98/03/18 [/user/masui/Mail/pomp/87160]       Returned mail: User unknown  Th
98/03/18 [/user/masui/Mail/pomp/87143]       Returned mail: User unknown  Th
98/03/18 [/user/masui/Mail/outbox/87242]     RE: Time machine  -----Origina
98/03/18 [/user/masui/Mail/outbox/87237]     Time machine  では「将来/過去時
98/03/18 [/user/masui/Mail/outbox/87230]     Re: igarashi-kun   In Message <1
98/03/18 [/user/masui/Mail/outbox/87222]     7-digit Zip  私のページ(http://pitec
98/03/18 [/user/masui/Mail/outbox/87221]     Re: Example-based video reserv
98/03/18 [/user/masui/Mail/outbox/87218]     Clock  では、こういうのはどうで
98/03/18 [/user/masui/Mail/outbox/87217]     Re: VCR   In Message <350F24F
98/03/18 [/user/masui/Mail/outbox/87198]     Re: VCR   御意見ありがとうござい
```

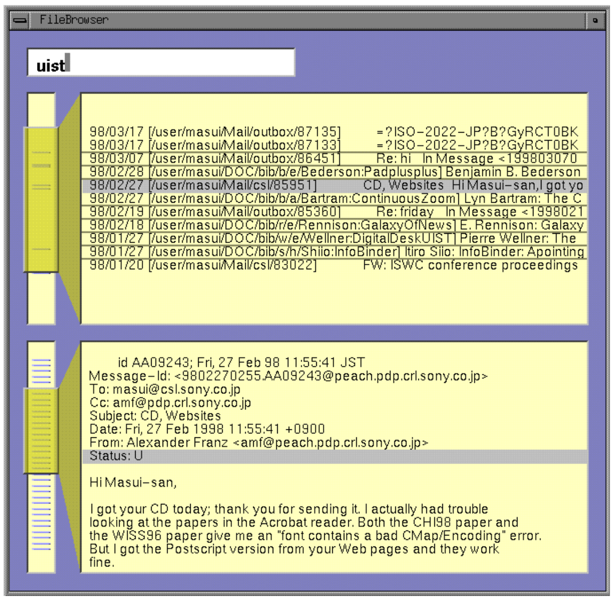
```
X-UIDL: 936a34ecbbfce5a86c1a968b58307156
Received: (from htan@localhost)
        by hotaka.csl.sony.co.jp (8.8.8/3.6W/hotaka/98021914) id QAA15448
        for shain; Thu, 19 Mar 1998 16:18:43 +0900 (JST)
Date: Thu, 19 Mar 1998 16:18:43 +0900 (JST)
From: Hisaaki Tanaka <htan@csl.sony.co.jp>
Message-Id: <199803190718.QAA15448@hotaka.csl.sony.co.jp>
To: shain@csl.sony.co.jp
Status: U


Dear all,

We have a technical talk during the general meeting on Thursday March 26
at the core room on 3F.  The talk will be in Japanese.
```

Figure 11: File/Message Browser

3.2 Example 2: File/Message Browser

Handling huge number of files and e-mail messages is not a trivial task, especially when a user sends and receives a number of e-mail messages every day. LensBar is very useful for file/message browsers, since integration of browsing and querying helps the user finding desired document or message easily.

In the file/message browser shown in Figure 11, two LensBars are used. The one at the top shows the titles of files and messages listed in chronological order. The LensBar at the bottom shows the contents of the selected message. It does not have a text input area for filtering, but it is used as a substitute for conventional scrollable text window (with a scrollbar with uniform background.) When the user enters a keyword, messages which contain the keyword are selected, and the content of the selected message is shown in the scroll window at the bottom.



```
98/03/17 [/user/masui/Mail/outbox/87135]        =?ISO-2022-JP?B?GyRCT0BK
98/03/17 [/user/masui/Mail/outbox/87133]        =?ISO-2022-JP?B?GyRCT0BK
98/03/07 [/user/masui/Mail/outbox/86451]        Re: hi   In Message <199803070
98/02/28 [/user/masui/DOC/bib/b/e/Bederson:Padplusplus] Benjamin B. Bederson
98/02/27 [/user/masui/Mail/csl/85951]        CD, Websites  Hi Masui-san,I got yo
98/02/27 [/user/masui/DOC/bib/b/a/Bartram:ContinuousZoom] Lyn Bartram: The C
98/02/19 [/user/masui/Mail/outbox/85360]        Re: friday   In Message <1998021
98/02/18 [/user/masui/DOC/bib/r/e/Rennison:GalaxyOfNews] E. Rennison: Galaxy
98/01/27 [/user/masui/DOC/bib/w/e/Wellner:DigitalDeskUIST] Pierre Wellner: The
98/01/27 [/user/masui/DOC/bib/s/h/Shiio:InfoBinder] Itiro Siio: InfoBinder: Apointing
98/01/20 [/user/masui/Mail/csl/83022]        FW: ISWC conference proceedings
```

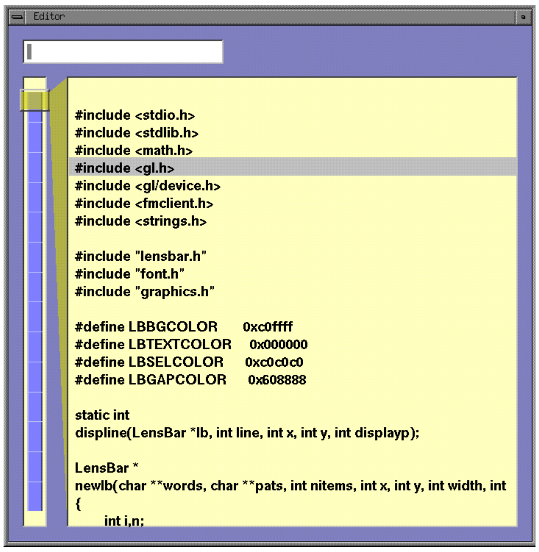
```
        id AA09243; Fri, 27 Feb 98 11:55:41 JST
Message-Id: <9802270255.AA09243@peach.pdp.crl.sony.co.jp>
To: masui@csl.sony.co.jp
Cc: amf@pdp.crl.sony.co.jp
Subject: CD, Websites
Date: Fri, 27 Feb 1998 11:55:41 +0900
From: Alexander Franz <amf@peach.pdp.crl.sony.co.jp>
Status: U

Hi Masui-san,

I got your CD today; thank you for sending it. I actually had trouble
looking at the papers in the Acrobat reader. Both the CHI98 paper and
the WISS96 paper give me an "font contains a bad CMap/Encoding" error.
But I got the Postscript version from your Web pages and they work
fine.
```

Figure 12: Filtering messages by “**`uist`**”

When a user enters a pattern “**`uist`**”, messages and files related to the UIST conference are selected and listed. Since the messages and files are listed in chronological order, he can see his activity related to the keyword. In this case, he can tell that he was active on the subject around 3/17 and 1/27, and he can check what he was doing around that time by checking other files and messages around it. It is useful for remembering what the user was doing when he received a message. In this sense, this browser can be viewed as another implementation of the Lifestreams system[7], with more powerful searching and browsing mechanism.

3.3 Example 3: Program Browser

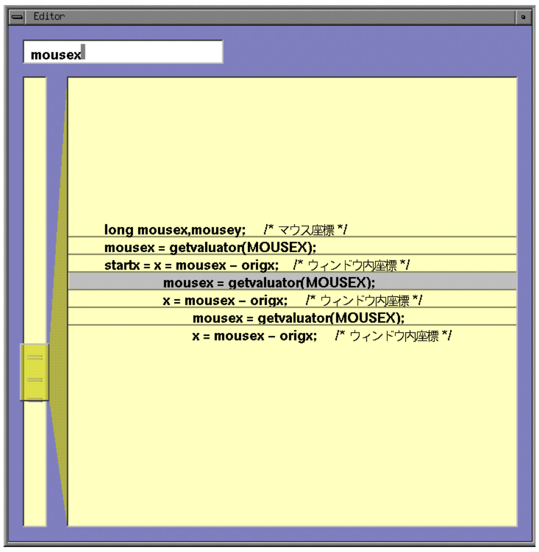
LensBar is also useful for browsing program texts. Figure 13 shows how the program `lensbar.c` can be viewed using LensBar.



```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <gl.h>
#include <gl/device.h>
#include <fmclient.h>
#include <strings.h>

#include "lensbar.h"
#include "font.h"
#include "graphics.h"

#define LBBGCOLOR      0xc0ffff
#define LBTEXTCOLOR    0x000000
#define LBSELCOLOR     0xc0c0c0
#define LBGAPCOLOR     0x608888

static int
displine(LensBar *lb, int line, int x, int y, int displayp);

LensBar *
newlb(char **words, char **pats, int nitems, int x, int y, int width, int
{
    int i,n;
```

Figure 13: Program browser

When a variable name "`mousex`" is specified as the filtering pattern, declaration statements and assignment statements related to the variable are selected and listed in the scroll window (Figure 14). Positions corresponding to the lines are also shown in the background of the slider, showing where in the program those lines exist.



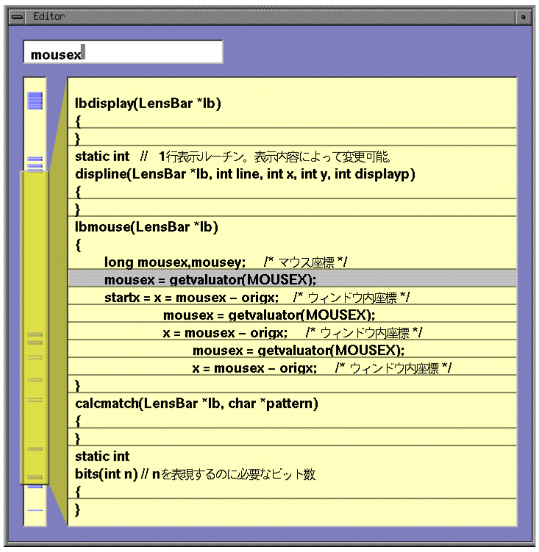
```
mousex

    long mousex,mousey;    /* マウス座標 */
    mousex = getvaluator(MOUSEX);
    startx = x = mousex – origx;   /* ウィンドウ内座標 */
            mousex = getvaluator(MOUSEX);
            x = mousex – origx;    /* ウィンドウ内座標 */
                mousex = getvaluator(MOUSEX);
                x = mousex – origx;    /* ウィンドウ内座標 */
```

Figure 14: Selecting lines with "`mousex`"

DOI value	Entry
0	`displine()`
0	`{`
-1	`    if(displayp){`
	`...`
0	`}`
0	`lbmouse(LensBar *lb)`
0	`{`
5	`    long mousex,mousey;`
-1	`    long origx,origy;`
	`...`
5	`    mousex = getvaluator(MOUSEX);`
5	`    startx = x = mousex - origx;`
	`...`
3	`            mousex = getvaluator(MOUSEX);`
	`...`
2	`                mousex = getvaluator(MOUSEX);`
-4	`                display();`
-1	`    }`
0	`}`

Table 2: Assignment of DOI values

Since lines with less indentation is usually more important in C program texts, large DOI values are assigned to lines with small indentation, as shown in Table 2. Lines which match the pattern string have positive DOI values, and other lines have DOI values smaller than or equal to zero. When the user drags the mouse to the right to expand the list, lines with large DOI values will emerge. In this case, function definitions can be seen in addition to the lines which contain "`mousex`" (Figure 15). In this way, users can focus on a variable and examine a long program text, while seeing the global context, just like using the FractalView editor[10] or other editors which supports focus+context editing based on the generalized fisheye views technique[8].



```
mousex

lbdisplay(LensBar *lb)
{
}
static int   //   1行表示ルーチン。表示内容によって変更可能。
displine(LensBar *lb, int line, int x, int y, int displayp)
{
}
lbmouse(LensBar *lb)
{
    long mousex,mousey;    /* マウス座標 */
    mousex = getvaluator(MOUSEX);
    startx = x = mousex – origx;   /* ウィンドウ内座標 */
            mousex = getvaluator(MOUSEX);
            x = mousex – origx;    /* ウィンドウ内座標 */
                mousex = getvaluator(MOUSEX);
                x = mousex – origx;    /* ウィンドウ内座標 */
}
calcmatch(LensBar *lb, char *pattern)
{
}
static int
bits(int n) // nを表現するのに必要なビット数
{
}
```

Figure 15: Showing important lines in addition to selected lines

Since this program browser is line-oriented, extending it to work as a text editor can easily be done.

3.4 Example 4: Hierarchical Menu

LensBar can also be used as a substitute for a hierarchical menu. Although hierarchical menus are widely used in current graphical user interfaces, it suffers from many shortcomings. First, users can't see how large and deep the menu is. Second, users can't tell whether the menu contains the entry they want, until they search the hierarchy intensively. Third, the operation is usually not reversible. That is, when a user does something wrong (e.g. selecting a wrong menu entry), he sometimes cannot go back to the previous state by moving the mouse in the reverse direction, but he has to do the entire selection operation from the start.

Using a LensBar in place of a conventional hierarchical menu, all of the above problems are solved. We show this by using the directory structure of Unix as an example of a very large hierarchical menu.

Figure 16 shows the initial display of the menu. Files and directories under `/usr/lib/` is listed. In this application, large DOI value is assigned to higher-level menu entries.

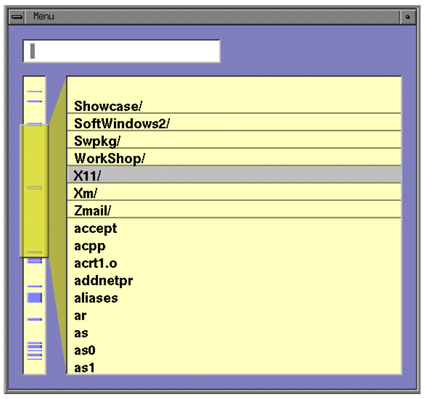


Figure 16: Showing the top menu

When the user clicks the mouse on `X11` and drags to the right, more files and directories with smaller DOI values become visible, and subdirectories of the `X11` directory are displayed as if the directories is expanded like a hierarchical menu. (Figure 17)

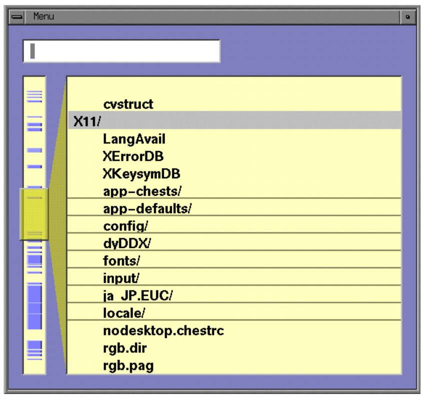


Figure 17: Expanding a sub menu

The user can further expand the subdirectories and find an entry `k14.pcf.Z` (a font file) in the `X11/fonts/misc/` directory.

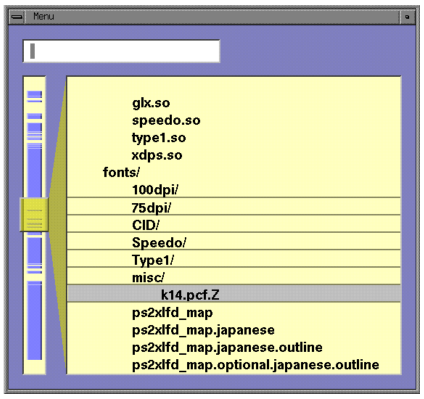


Figure 18: Finding an entry

When the user drags the mouse to the left from the position, the menu will be shrunk to look like Figure 19. This is similar to Figure 17, but all the operation is continuous and reversible, and unlike conventional hierarchical menu, the user can easily go back and forth between Figure 18 and Figure 19, just by dragging the mouse horizontally.

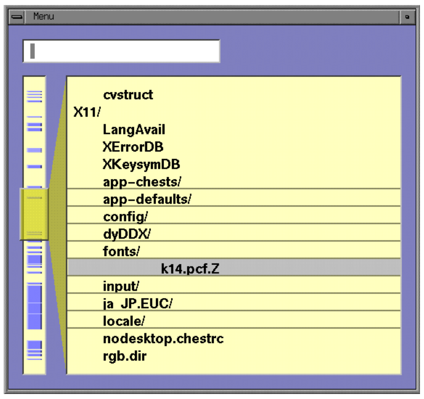


Figure 19: Shrinking the menu

4 RELATED WORK

Many visualization and interaction techniques have been proposed for browsing and filtering large data. Examples include Xerox PARC's Information Visualizer[4] and TableLens[16], nonlinear magnification systems like Generalized Fisheye Views[8], Graphical Fisheye Views[17] and Hyperbolic Visualizer[11], visualization systems using fractal[10], 2D zooming systems like Pad[15] and Pad++[3], 2D/3D zooming information retrieval system like the WING system[14], various information visualization systems developed at University of Maryland including FilmFinder[2], etc. and most of the systems supports methods for browsing the whole data, methods for filtering the data, and methods for showing the focal point while retaining the context.

Although these techniques are useful in various application domains, they tend to be special to particular applications, or their interaction method is completely different from existing GUI tools, as we have shown in Section 1. Also, most of the techniques are not applicable to visualizing simple list of many number of items.

On the other hand, LensBar is a simple extension to conventional slider and scrolling window, and its appearance and interaction method are not very different from conventional GUI tools. In spite of that, LensBar provides many powerful mechanisms for browsing and filtering a large list of items.

Various extensions to conventional sliders have been proposed. With AlphaSlider[1], users can control the sensitivity of the slider knob by clicking different portion of the knob. FineSlider[13] also enables users to move the slider knob precisely, and LensBar is taking the same approach.

Furnas[9] argued that appropriate data structure is necessary for navigating in a large data set, and as an example, he proposed adding a tree structure for the efficient view traversal of a large list. LensBar's assignment of DOI values yields the same effect, without using an additional tree structure.

LensBar's method of using the background of the slider is somewhat similar to Eick's "data visualization slider"[6] in that both of them are trying to make more use of the screen real estate. Although data visualization slider is useful for visualizing a large list, supported user operations are very different from conventional sliders and should be considered as a completely different visualization tool. Filtering and zooming are not supported in the data visualization slider. Chimera's ValueBars[5] is an approach to display additional information of the list items in adjacent to a conventional slider. This technique is useful in browsing various attributes of listed items, and it would be interesting to add similar features to LensBar.

5 CONCLUSION

We introduced a new powerful visualization and filtering technique called the LensBar. Although LensBar looks and acts very much like a conventional scroll window with a slider, its filtering and zooming mechanism makes various sophisticated interaction possible. LensBar can be used in wide range of applications which handle any kind of large lists, ranging from text browsers to hierarchical menus.

References

[1] Christopher Ahlberg and Ben Shneiderman. AlphaSlider: A compact and rapid selector. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94)*, pages 365–371. Addison-Wesley, April 1994.

[2] Christopher Ahlberg and Ben Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94)*, pages 313–317. Addison-Wesley, April 1994.

[3] Benjamin B. Bederson and James D. Hollan. Pad++: A zooming graphical interface for exploring alternate interface physics. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'94)*, pages 17–26. ACM Press, November 1994.

[4] S. K. Card, G. G. Robertson, and J. D. Mackinlay. The Information Visualizer, an information workspace. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)*, pages 181–188. Addison-Wesley, April 1991.

[5] Richard Chimera. Value Bars: An information visualization and navigation tool for multi-attribute listings. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'92)*, pages 293–294. Addison-Wesley, May 1992.

[6] Stephen G. Eick. Data visualization sliders. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'94)*, pages 119–120. ACM Press, November 1994.

[7] Eric Freeman and Scott Fertig. Lifestreams: Organizing your electronic life. In *AAAI Fall Symposium: AI Applications in Knowledge Navigation and Retrieval*, Cambridge, MA, November 1995.

[8] G. W. Furnas. Generalized fisheye views. In *Proceedings of the CHI'86 Conference on Human Factors in Computing Systems and Graphic Interfaces*, pages 16–23, Boston, May 1986. Addison-Wesley.

[9] George W. Furnas. Effective view navigation. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'97)*, pages 367–374. Addison-Wesley, April 1997.

[10] Hideki Koike and H. Yoshihara. Fractal approaches for visualizing huge hierarchies. In *Proceedings of 1993 IEEE Symposium on Visual Languages (VL'93)*, pages 55–60. IEEE Computer Society, IEEE Computer Society Press, 1993.

[11] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'95)*. Addison-Wesley, May 1995.

[12] Toshiyuki Masui. An efficient text input method for pen-based computers. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'98)*, pages 328–335. Addison-Wesley, April 1998.

[13] Toshiyuki Masui, Kouichi Kashiwagi, and George R. Borden. Elastic graphical interfaces for precise data manipulation. In *CHI'95 Conference Companion*, pages 143–144. Addison-Wesley, May 1995.

[14] Toshiyuki Masui, Mitsuru Minakuchi, George R. Borden IV, and Kouichi Kashiwagi. Multiple-view approach for smooth information retrieval. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'95)*, pages 199–206. ACM Press, November 1995.

[15] Ken Perlin and David Fox. Pad: An alternative approach to the computer interface. In *ACM SIGGRAPH'93 Conference Proceedings*, pages 57–64, August 1993.

[16] Ramana Rao and Stuart K. Card. The Table Lens: Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94)*, pages 318–322. Addison-Wesley, April 1994.

[17] Manojit Sarkar and Mark H. Brown. Graphical fisheye views. *Communications of the ACM*, 37(12):73–83, December 1994.

[18] Ben Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77, November 1994.