LensBar – Visualization for Browsing and Filtering Large Lists of Data

Toshiyuki Masui
Sony Computer Science Laboratories, Inc.
3-14-13 Higashi-Gotanda, Shinagawa, Tokyo 141-0022, Japan.
masui@csl.sony.co.jp

Abstract

We propose a simple and powerful graphical interface tool called the LensBar *for filtering and visualizing large lists of data. Browsing and querying are the most important tasks in retrieving information and LensBar integrates the two techniques into a simple scroll window with slider. While it looks familiar to users of conventional graphical interface tools, its filtering and zooming features offer sophisticated handling of large lists of textual data.*

1 Introduction

Various visualization techniques for browsing large amount of data have been proposed recently. A number of information retrieval techniques for filtering huge chunks of data using query keywords have also been proposed. In some systems, visualization and filtering are integrated, allowing users to filter and browse data at the same time.

Although the combination of browsing and querying is more powerful than doing each separately, most existing systems suffer from several shortcomings. First, most of the visualization techniques are far from general-purpose, applicable only to special application domains. Second, special interaction techniques are usually required to effectively filter data and control the visualization result. For example, with the FilmFinder system[2], users can get information on movie titles by entering the names of actors or years of production, and they can control the visualized query result displayed as a 2-dimensional scattered plot. The filtering and visualization method is valid only for this particular application, and the overall design of the visualization and interaction techniques would need to be modified for different applications.

It is natural that the handling of special data requires special visualization and interaction techniques, but it seems strange that not many effective techniques for query and visualization have been proposed and used widely even for handling common data structures like texts or simple data lists. We propose a simple, powerful interface tool called *LensBar* for filtering and visualizing large data lists. LensBar works as an extension to a conventional scroll window or a substitute for a hierarchical menu, and it can be applied to wide range of applications where these tools are currently used.

2 Visualization and interaction techniques

Our technique is based on the following strategies:

- Browsing the whole list using a precisely-controllable slider and 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

The following sections contain a detailed description of the techniques.

Integrating browsing and querying To browse a large list, we use a slider (scroll bar) with accompanying scroll window, which is a common combination in current graphical user interfaces. An extra text input area is added for filtering list items, so that 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 that match the pattern. For example, if the first entry in the list matches the pattern, the top line of the slider background is highlighted. When no pattern is specified, all 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 moves only on the highlighted portion. The highlighted region within the slider knob always corresponds to the items displayed in the scroll window.

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

Dynamic approximate string matching When no entry in the list matches the specified pattern, dynamic approximate string matching[12] is done automatically, and entries closest to the pattern are treated as matched. This is usually more convenient than giving no query result when no match is found, and especially useful when users are not very sure of the spelling, e.g., when 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, just as with a conventional scrolling window. Using LensBar, a user can control the number 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 for display. For example, when the zooming level is set to a large value, items with small DOI values disappear to produce a zoom-out effect.

A zooming interface is essential in 3D interactive systems, and it has also been proved to be effective in 2D GUI[3, 15]. Although zooming interfaces for handling (1D) large lists are not as widely used as in handling 3D/2D objects, they are convenient for controlling the amount of displayed data.

Precise control of the slider Using a conventional slider, precise scrolling of large data chunks is difficult, because 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]; we adopted the technique described in [13]. Clicking the mouse on the slider knob allows the user to move the knob directly to any place on the bar, as with a conventional slider. Clicking on the scroll bar at a spot other than the knob moves the knob toward the cursor location. Speed is proportional to distance, giving the user an easy way to control speed.

3 Examples

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

3.1 Example 1: Searching words in a dictionary

The first example shows how LensBar works in a simple dictionary application.

Figure 1 shows the initial display of a dictionary application using LensBar. A slider with a transparent knob is displayed at left. In the background of the slider area, a highlighted rectangle is displayed, showing that all dictionary entries are selected for display. In Figure 1, only the ten words corresponding 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 entire list.

The dictionary contains more than 30,000 words. The knob can be dragged to any position in the dictionary, and the user can click another place on the slider to move the knob with more precision (Figure 2). The user can also drag the scroll window by clicking the mouse in the window and dragging it vertically.

In this dictionary application, a 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 is gradually reduced and a wider range of words appears in the scroll window. For example, when the zooming level is 2.0, every second word on each side of "`graphics`" is displayed. In Figure 3, every 256th word in the dictionary is displayed (zooming level is around 9), and in Figure 4, every 4096th word is displayed. Hidden entries appear as gray lines between words. At the same time, the background of the slider changes to show distribution of

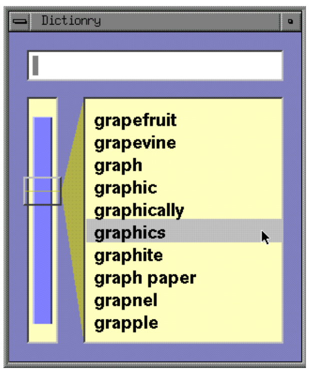


Figure 1. Initial display of the dictionary application

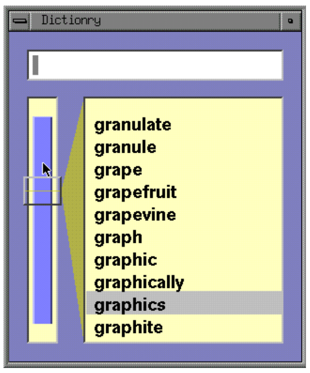


Figure 2. Scrolling the window

the selected words. As the range of the words displayed becomes larger, the slider knob also becomes larger to cover the area.

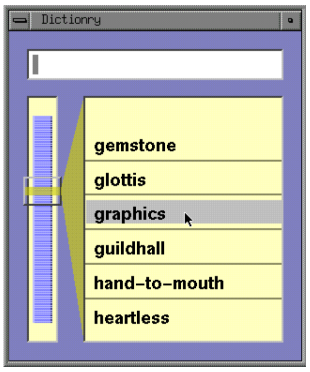


Figure 3. Zooming out to show every 256th word

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

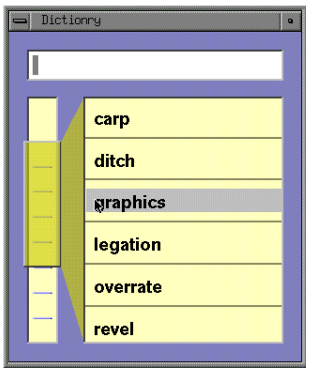


Figure 4. Zooming out more to show every 4096th word

The zoom level is determined by 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 zoom operation; when the user moves the mouse cursor back to the previous position, the system returns to the previous state. This continuous and reversible characteristic of the zooming interface[14] helps users easily recover from erroneous operations.

When the user enters a pattern string, dynamic approximate pattern matching is done and only those entries that match the pattern are selected for display. Figure 5 shows the display after the user specified “**`g`**” as the pattern string. The whole list is filtered and only those words beginning with “**`g`**” are selected and displayed.

A space (“␣”) is used as a wildcard, like the pattern “**`.*`**” used in regular expressions. So when “␣**`q`**” is specified as the pattern, all the words that include the letter “**`q`**” are selected (Figure 6). Since those words are scattered throughout the dictionary, the highlighted area in the slider resembles a comb.

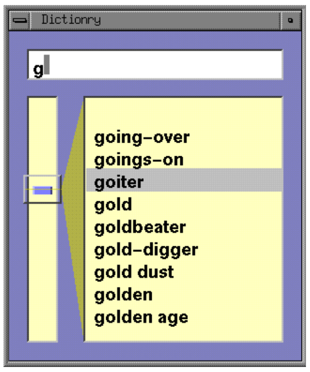


Figure 5. Filtering by “g”

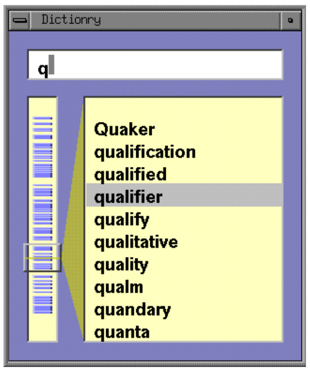


Figure 6. Filtering by "␣q"

Let's look for a word with a difficult spelling like "**`Pithecanthropus`**". Figure 7 shows the LensBar display after a user typed a wrong spelling "**`pite`**", revealing that only two words in the dictionary begin with "**`pite`**".

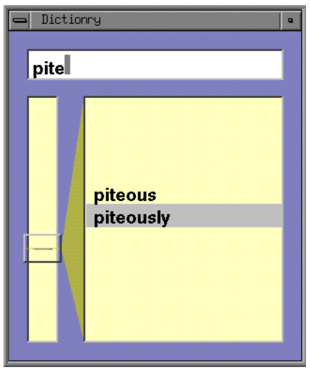


Figure 7. Typing "`pite`"

Here, the user knows that "**`pite`**..." is not the right spelling for "**`Pithecantropus`**", but doesn't know what other letters to add.

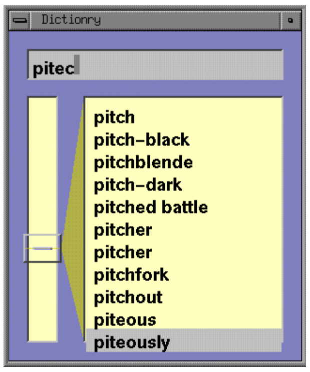


Figure 8. Typing "`pitec`"

When the user types "**`c`**" next, the string matcher notices that no words in the dictionary begin with "**`pitec`**", so it searches and displays all words that match the pattern "**`pitec`**" with one error. 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 match was found. When the user types "**`a`**" next, words like "**`pitch`**" disappear because they do not match "**`piteca`**" with one error, and only "**`Pithecanthropus`**" displays. By dynamically changing the parameter of the approximate string matching algorithm and showing the words closest to the pattern, the system allows users to see the candidates closest to the given pattern and have better chance of finding the desired information.

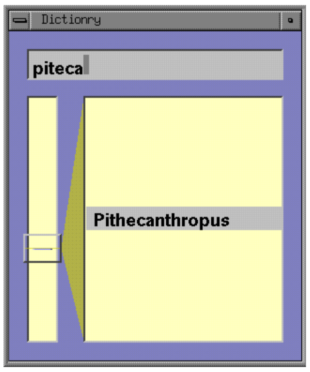


Figure 9. "`Pithecanthropus`" found from "`piteca`"

When not sure of the spelling, the user can explicitly use wildcard characters instead of specifying uncertain letters. Figure 10 shows a situation where a user tries to find "**`Mediterranean`**", and specifies only a few of the letters thought to exist in that word. Since there are only five words in the dictionary containing "**`m`**", "**`d`**", "**`t`**", "**`r`**" and "**`n`**" in their spelling, the user can easily find the desired word in the list.

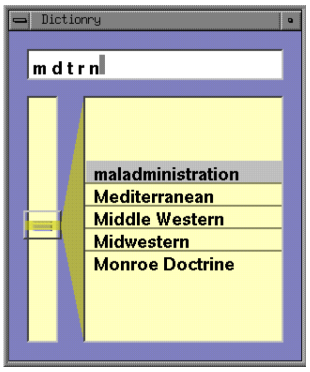


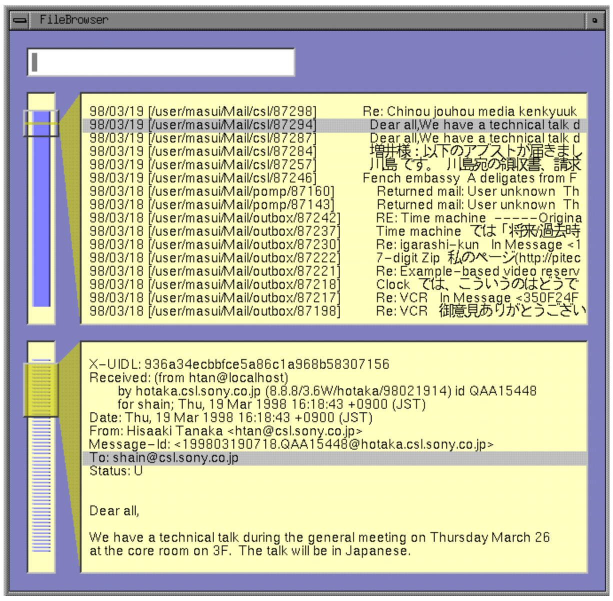
Figure 10. Typing "`m␣d␣t␣r␣n`"

3.2 Example 2: File/message browser

Handling a huge number of files and e-mail messages is not trivial, 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 find the 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 is used as a substitute for a conventional scrollable text window (with a scrollbar with uniform background.) When the user enters a keyword, messages containing the keyword are selected, and the content of the selected message is shown in the scroll window at the bottom.

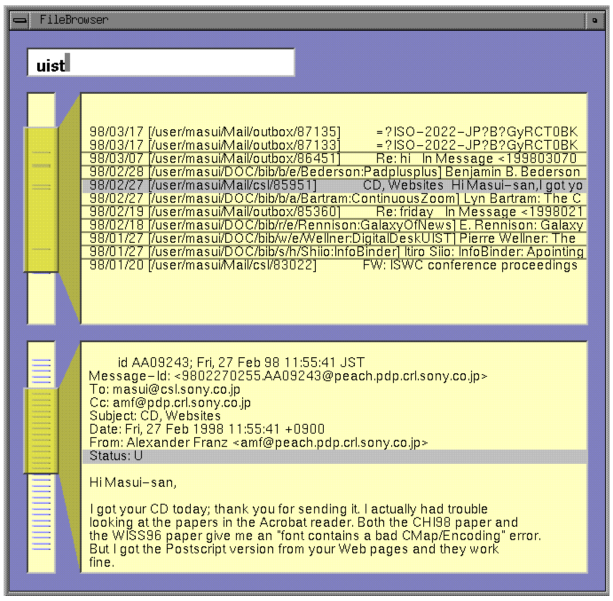
When a user enters the pattern "`uist`", messages and files related to the UIST conference are selected and listed. Since the messages and files are listed in chronological order, the user can see the activity related to the keyword. Here, the user knows he was active on the subject around 3/17 and 1/27,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



```
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



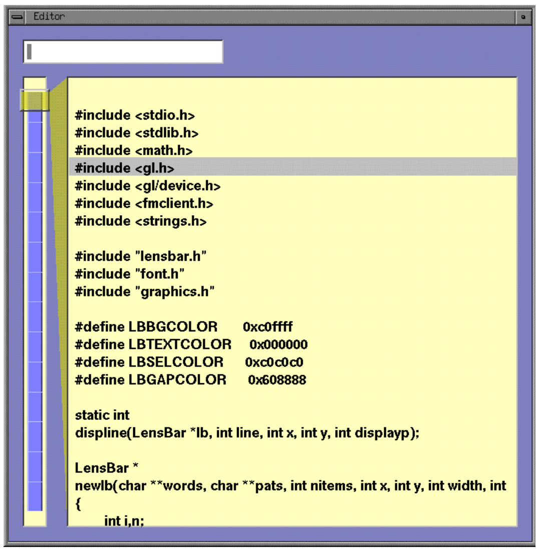
```
uist

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`"

3.3 Example 3: Program browser

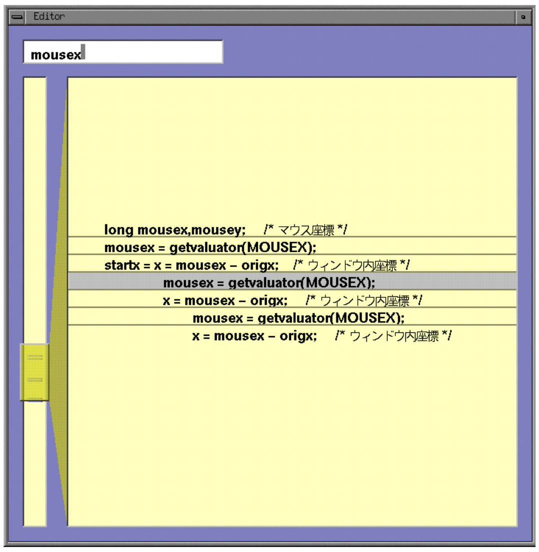
LensBar is also useful for browsing program text. Figure 13 shows how the source program **`lensbar.c`** can be viewed using LensBar.



```
#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;
```

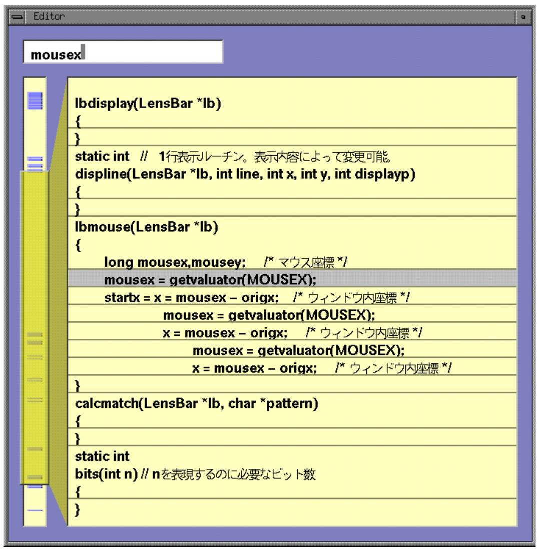
Figure 13. Program browser

When a variable name "**`mousex`**" is specified as the filter, 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 slider background, showing where in the program those lines exist.



```
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”



```
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

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	`}`

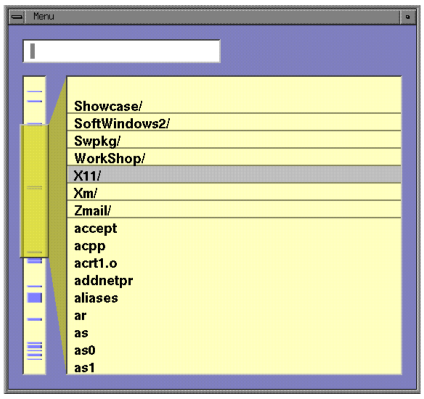
Table 2. Assignment of DOI values

Since lines with less indentation are usually more important in C program listings, large DOI values are assigned to lines with small indentation, as shown in Table 2. Lines that 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 emerge. Here, function definitions can be seen in addition to the lines containing “`mousex`” (Figure 15). In this way, users can focus on a variable and examine a long program listing, while seeing the global context, just as with the FractalView editor[10] or other editors that supports focus+context editing based on the generalized fisheye views technique[8].

Since this program browser is line-oriented, extending it to work as a text editor is simple.

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, they suffer many shortcomings. First, users can’t see how large and deep a menu is. Second, users can’t tell whether the menu contains the entry they want, until they intensively search the hierarchy. Third, the operation is usually not reversible. That is, when a user does something wrong (e.g., selecting a wrong menu entry), it is not possible to return to the previous state by moving the mouse in the reverse direction; the entire selection operation must be completed.



Showcase/
SoftWindows2/
Swpkg/
WorkShop/
X11/
Xm/
Zmail/
accept
acpp
acrt1.o
addnetpr
aliases
ar
as
as0
as1

Figure 16. Showing the top menu

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

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

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 were expanded like a hierarchical menu. (Figure 17)

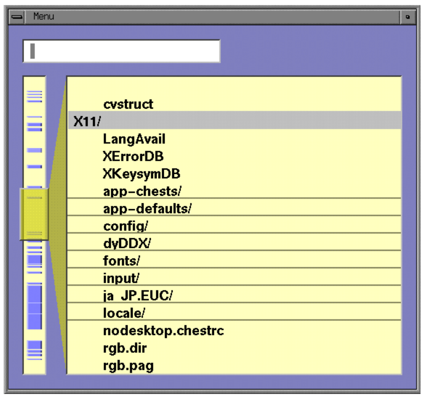


Figure 17. Expanding a sub menu

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

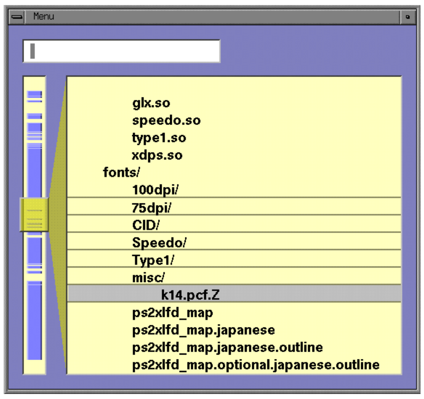


Figure 18. Finding an entry

Dragging the mouse to the left shrinks the menu to look like Figure 19. This is similar to Figure 17, but all operations are 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.

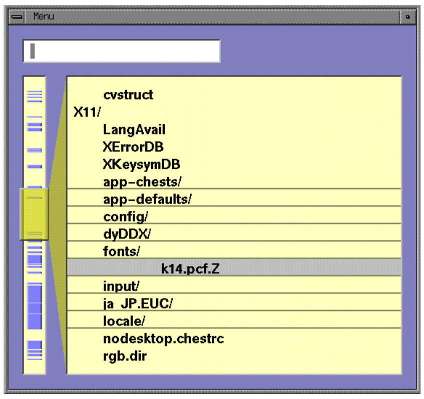


Figure 19. Shrinking the menu

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 systems like the WING system[14], and various information visualization systems developed at University of Maryland including FilmFinder[2]. Most of the systems support methods for browsing all of the 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 focus on particular applications, or their method of interaction is completely different from existing GUI tools, as we have shown in Section 1. Beyond that, most of the techniques are not applicable to visualizing simple lists containing many items. LensBar is a simple extension to the conventional slider and scrolling window, and its appearance and interaction differ only slightly from conventional GUI tools. Nevertheless, LensBar provides many powerful features for browsing and filtering large item list.

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

Furnas[9] argued that an appropriate data structure is needed to navigate a large data set; 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 slider background is somewhat similar to Eick's "data visualization slider"[6] in that both try to make better use of screen real estate. Although the data visualization slider is useful for visualizing a large list, supported user operations differ greatly 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 displaying additional information of the list items 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 have introduced a powerful new visualization and filtering technique called LensBar. Although LensBar looks and acts very much like a conventional scroll window with a slider, its filtering and zooming mechanism makes sophisticated interaction possible. LensBar can be used in wide range of applications that handle large lists, ranging from text browsers to hierarchical menus.

References

[1] C. Ahlberg and B. 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] C. Ahlberg and B. 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] B. B. Bederson and J. 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] R. 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] S. 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] E. Freeman and S. 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] G. 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] H. 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] J. Lamping, R. Rao, and P. 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] T. 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] T. Masui, K. Kashiwagi, and G. R. Borden. Elastic graphical interfaces for precise data manipulation. In *CHI'95 Conference Companion*, pages 143–144. Addison-Wesley, May 1995.

[14] T. Masui, M. Minakuchi, G. R. B. IV, and K. 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] K. Perlin and D. Fox. Pad: An alternative approach to the computer interface. In *ACM SIGGRAPH'93 Conference Proceedings*, pages 57–64, August 1993.

[16] R. Rao and S. 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] M. Sarkar and M. H. Brown. Graphical fisheye views. *Communications of the ACM*, 37(12):73–83, December 1994.

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