Using small screen space more efficiently

Tomonari Kamba[1], Shawn A. Elson[2], Terry Harpold[2]
Tim Stamper[2], Piyawadee "Noi" Sukaviriya[1]

(1) Graphics, Visualization, and Usability Center
College of Computing
Georgia Institute of Technology

(2) School of Literature, Communication, and Culture
Georgia Institute of Technology
Atlanta, GA 30332-0165

tomo@cc.gatech.edu, elson@cc.gatech.edu, terry.harpold@lcc.gatech.edu,
tstamp@mindspring.com, noi@cc.gatech.edu

ABSTRACT

This paper describes techniques for maximizing the efficient use of small screen space by combining delayed response with semi-transparency of control objects ("widgets") and on-screen text. Most research on the limitations of small display screens has focused on methods for optimizing concurrent display of text and widgets at the same level of transparency (that is, both are equally opaque). Prior research which proposes that widgets may be made semi-transparent is promising, but it does not, we feel, adequately address problems associated with user interaction with text that is partially obscured by the widgets. In this paper, we will propose that a variable delay in the response of overlapping widgets and text improves the effectiveness of the semi-transparent widget/text model. Our conclusions are based on usability studies of a prototype of an online newspaper that combined transparency and delayed-response techniques.

Keywords

PDAs, icons, transparency, usability study

INTRODUCTION AND PROBLEM STATEMENT

The dramatic growth in recent years of the personal digital assistant (PDA) market demonstrates that users are willing to put up with small, hard-to-read, displays, limited storage and battery life, slow CPU speeds and cumbersome data transfer, in the hope of achieving truly portable access to electronic data. It is probably safe to predict that devices available only a few years from now will be dramatically improved. Future generations of PDAs will have higher-contrast, easier-to-read displays, they will have greater storage capacities, they will be much faster and run for longer periods of time between charges, and they will be more flexible in how they communicate with each other and with other computing devices. These probable changes in PDAs are, however, limited by two constraining factors: the dimensions of displayed text and of the screens on which the text is displayed are unlikely to change very much. Reductions in the size of displayed text will be limited by the ability of users to discern small type sizes on any display device, especially one of relatively low resolution. Our informal observations of PDAs based on the Apple Newton and General Magic's Magic Cap operating systems suggests that a practical threshold of legibility for most users lies somewhere between 9 and 12 points (72 points = 2.5 cm.) The small screen size of PDAs is not a technical limitation, but a key factor in their usefulness. Users clearly want devices that can be easily carried about and held in one hand. It is not difficult to imagine even smaller PDAs, worn on the wrist or carried on a keychain.

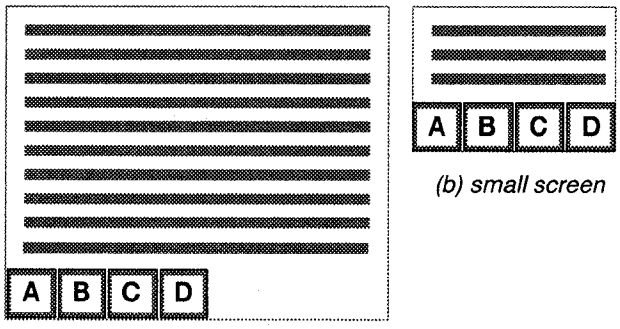
The small physical size of a PDA limits the maximum size of its screen, which can be no larger than the dimensions of the machine in which it is embedded. On the other hand, the need for displayed text to be legible defines another, more subtle boundary: if the size of text cannot be reduced below a threshold of legibility, then, as the screen shrinks in size, and less information may be shown on it, and the user will be required to increase the level of interaction with the device in order to get to desired information.

The design of user interfaces for PDAs must balance two opposing forces: the need to shrink the screen to a size that fits inside a very small box (we'll call this the "physical" limitation), and the need to keep the screen sufficiently large to show enough information that the device is actually useful (we'll call this the "functional" limitation.) This balance becomes particularly difficult in the case of navigational or functional controls, the widgets that must somehow be made available to the user to allow interaction with the information on-screen (switching between tasks, selecting information to be changed in some way, adding new information, etc.).

CHI 96 Vancouver, BC Canada
© 1996 ACM 0-89791-777-4/96/04..$3.50

Most of the interface objects used in desktop computing environments – pull-down or popup menus, multiple windows, icons – consume a great deal of valuable screen space on the PDA screen. This accounts, we believe, for the efforts of the designers of the current crop of PDAs to minimize the size of these interface objects, or, in some cases, to eschew them altogether. It explains also the emphasis that many of these systems place on handwriting recognition technologies – the keyboard is perhaps the most cumbersome of control widgets. (The document-oriented, pen- and gesture-driven elements of the Newton interface are a good example of these approaches.)

Once again, however, the interface designer faces a version of the functional limitation: the user will expect not only to passively read content displayed on the device, but also to do something with it (to select it, modify it, or enter new content, etc.), and this means that he or she will need to interact with objects (widgets) that are different from the text, but equally available. These widgets must be large enough to be readily distinguished from the content and from each other, and to be practical targets for some kind of interaction (by means of a finger or pen, for example.) But if they are displayed all or most of the time, they will consume screen space that could otherwise be used to display content. Because the number of these widgets will depend on the functions supported by the active application, and not on the size of the screen, the portion of screen that must be surrendered to them will increase as the screen grows smaller, as illustrated in Figure 1.



(a) large screen

(b) small screen

FIGURE 1. Surrendering screen space to widgets

The price paid for showing control widgets can be substantial – in the Magic Cap interface, for example, nearly 25% of the screen is used to display icons along the bottom and right edge of the screen.

Our research focused on techniques for reducing the screen space that must be surrendered to widgets, thereby maximizing the available space for display of content. In the next section, we will review general solutions to this problem suggested by the work of other researchers and designers working with interfaces of both handheld devices and desktop computers. Among these solutions, we will single out one approach that seems especially promising to us: the use of semi-transparent widgets, laid over text, so that both are present on the screen at the same time, but the text is able to fill nearly the entire screen. We will, however, point out a serious limitation with this solution, related to user interaction with the text. In the remaining sections of the paper, we will propose a variation of the semi-transparent widget/text model that improves its responsiveness to user interaction, and describe the results of a series of usability studies of a prototype that applies this improved model.

RELATED WORK

Showing Information Structure More Efficiently

One approach to the problem of maximizing the display of content is to improve the efficiency by which its underlying structure is represented on-screen. Several tools in Information Visualizer [4] take this approach. Cone Tree [13] and Hyperbolic Geometry [8] display in lucid forms complicated data hierarchies that might be otherwise invisible to the user, and Perspective Wall [10] does much the same for complex linear data structures. Cone Tree represents data hierarchies in 3D cone-shaped graphics. Hyperbolic Geometry displays a focused point within a data hierarchy in a large bounded space, and its context in a smaller bounded space. Perspective Wall displays relations between different nodes within the same document on two adjacent planes ("walls"), with semantic or structural differences between the nodes represented by the relative positions of the nodes on these walls. These methods for representing underlying data structure are arguably more precise and efficient than, for example, the very concrete "desktop" paradigm of Magic Cap (in which, for example, the Datebook application is accessed by tapping on a miniature appointment book), but they do not address the functional limitation of simultaneous display of widgets and content on a small screen.

Showing Information Content More Efficiently

A second approach to the problem directly addresses the presentation of the displayed content. Magic Lens [3] is a filtering tool that can change or modify the presentation strategy of objects on the screen over which it is laid. For example, a portion of a geographical map can change into a weather map or a population map when a special lens is applied to it. Starfield [6] displays a two-dimensional scatterplot of a multidimensional database, applying a dynamic filtering mechanism which continuously controls the density of information shown on the screen, and panning/zooming techniques to focus on the portion of the content displayed. Table Lens [11] represents large databases in table format. The user can zoom in on an arbitrary part of a table, and view it alongside a global view of the table.

Pad++ applies another kind of zooming/panning technique to displayed content [2]. The user can focus on and zoom into any part of the content, and smooth animation during the zoom insures that the semantic link between the zoomed

information and its context are maintained. Galaxy of News [12] uses similar zooming techniques combined with dynamic restructuring of data hierarchies during the zooming/panning process.

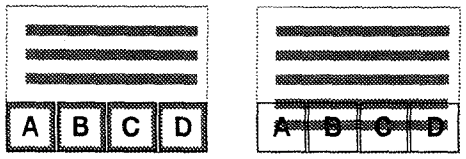
We feel that these methods suggest several improvements on the simple, scrolling displays of long lists or blocks of text that are commonly encountered when using PDAs. However, they suffer from a critical limitation: filtering information content so that it may be displayed in smaller chunks that require less screen space does not obviate the need for widgets required to apply the filters, or to enable manipulation of the filtered content.

Showing Widgets More Efficiently

We were surprised to find little research that directly addresses the efficient display of the widgets themselves. We suspect that this may be because the physical/functional opposition outlined above is both obvious and intractable – designers know that widgets can only be so small before they cease to be useful, so they have concentrated their efforts on increasing the intelligence or functionality of the widgets, rather than how much screen space must be sacrificed because of them.

Gestural interfaces, like that used in Apple's Newton operating system, mark an important departure from the paradigm of on-screen control objects. Tying functionality to pen-based gestures frees up valuable screen space, because the user can directly interact with displayed content by simply drawing over it with the pen. This technique works well in many applications, but it also has limitations. First, the user must learn the gestures, and be able to reproduce them accurately. (The list of gestures on the inside of the screen covers of the Newton MessagePad 110 and 120 suggests that users are unlikely to be able to recall all the possible gestures without a mnemonic aid.) Second, not all functions supported by current PDAs can be mapped to intuitive gestures. There is a row of unchanging iconic buttons below the active screen space of the MessagePads. These buttons are used to switch between applications, and to undo previous tasks. By moving these buttons out of the active screen space, Apple has reclaimed area for displaying information, but in so doing has also underscored the difficulty of the problem illustrated in Figure 1.

Kurtenbach and Buxton's Marking Menus [7] offer an interesting variation on the gestural interface. In their system, when the user holds a pen to the screen for a predefined period of time, a pie-shaped menu is displayed below the pen, and the user can select an item in the menu by stroking toward it. Users who are familiar with the structure of the menu can select menu items without waiting for the pie to be displayed, by simply stroking in the direction of that slice of the pie. This approach appears to combine strengths of a gestural interface and a menu- or icon-driven interface, as it hides widgets until they are needed, and allows the content to fill the entire screen. For these reasons, Marking Menus may be a good solution for reclaiming space on small screens. It does not, however, permit simultaneous



(a) *opaque widgets* (b) *semi-transparent widgets*

Figure 2. Selecting content beneath semi-transparent widgets

display of widgets and content. Because the pie menu is opaque, it will at least temporarily obscure underlying content.

Semi-transparent Widgets And Their Limitations

An approach that begins to address this objection makes use of semi-transparent popup menus that do not completely obscure underlying content [5]. (Lieberman, et al. have used a similar semi-transparency method to show data context in a zooming interface [9].) Harrison, et al. proposed allowing users to define the level of menu transparency. They showed that the transparency levels selected by users will vary, depending on the tasks to be completed, and each user's level of expertise.

We believe that the use of control widgets of variable transparency is a promising method for maximizing usable screen space. If the transparency of widgets is adjusted so that the content with which they intersect on-screen is nonetheless legible, then it should be possible to display both the widgets and a full screen of content without sacrificing any screen space reserved for the latter. There is, however, a serious obstacle that must be overcome before this method will support user interaction with the content.

For the purposes of our argument, let us assume that the widget layer is on top of the content layer. If opaque widgets obscure otherwise selectable content, and no mechanism is provided for passing through the widget layer to the content, then the user will be able to interact with only those parts of the content which are not obscured. If, however, the widget layer is semi-transparent – that is, if the underlying content and the widgets are legible at the same time – then it is not immediately clear which of the two layers is selectable (for copying, editing, etc.), *even if there is no formal mechanism for selecting the bottom layer.* This is illustrated in Figure 2.

One solution to this problem is to turn off the selectability of underlying objects, even when they are visible. However, this is likely to confuse and frustrate users, who will reasonably expect that visible objects will sustain some degree of interaction, simply because they are visible. It is, moreover, fairly easy to build a scenario where this solution will effectively prevent underlying objects from ever being selected – if, for example, the upper layer cannot be moved, and the lower layer cannot be scrolled so as to move the partially-obscured content into a selectable region of the screen. The only course of action in this case would be a

hardware toggle of some kind to force selectability – an awkward exception to the normal behavior of the interface and an unnecessary extra task for the user.

Instead, we propose a technique for determining the layer receiving user interaction that does not require an additional modifying step, as it based upon variations in the duration of the interaction. In this variation on the transparent widget/content model, the length of time during which the user engages with a region of the physical screen (wherein widgets and content overlap) would determine which virtual layer of the screen receives the interaction.

THE EXPERIMENT

Overview

To evaluate the merits of this technique in a practical example, we decided upon a prototype emulating a text-based online newspaper. PDAs are widely used to read text downloaded from digital news sources. These applications involve moderate quantities of relatively unstructured text, and usually support limited interactivity – simple navigation between stories, or between issues, searching for text strings, transferring information from a story to another application (for example, a scrapbook), and hypertext linking between stories or issues. The emulation of hypertext linking offered a focused example of object selection through layers – how, we wondered, would users attempting to select a passage of "hot" text (that is, a hypertext link) that was partially covered by a semi-transparent icon (or vice-versa) manage the transition between layers?

The prototype was implemented on a Macintosh in Allegiant SuperCard. Although the mouse-driven interface of the Macintosh differs from the pen-based interfaces of many PDAs, we believed that using a desktop computer for initial evaluation would nonetheless generate valuable data. Though we have emphasized in this paper the importance of the issue of screen size for PDAs, the balancing of window size with maximum content display in small windows is of value on desktop computers, where multiple windows may be open at any one time, and the user may wish to reduce the size of any given window to a practical minimum, while still being able to display as much content as possible. Interaction techniques developed for small screens would in this case apply equally well to larger screens displaying many small windows. Moreover, this initial run of the experiment was designed to test the merits of varying the responsiveness of semi-transparent objects (widgets and text), and those kinds of objects need not exist only on PDAs.

The overall design of the prototype strongly resembled that of AT&T's PersonaLink news reader for Magic Cap devices (see Figure 3). The active screen region in the prototype was 320 x 240 pixels. The top portion of most screens in the prototype contained labels identifying the current story and icons used to move between stories or to return to an index for that issue of the newspaper. The text of the story filled the center of the screen. Hypertext links in the story were underlined. Along the bottom of the screen were seven icons (shown left to right in Figure 3): Desk (to quit the newspaper and return to a hypothetical desktop), Search, Archive (to display previous issues of the newspaper), Append (to copy the current story to a Scrapbook), Scrapbook (to open a hypothetical Scrapbook application), and Trash. Along the right edge of the screen were icons for moving "up" or "down," to the next or previous page of the current story.

Most of the icons, and all of the hypertext links were only partially functional – when the user successfully highlighted a link or icon, a status message was briefly displayed in a floating palette on the screen. (This palette is not shown in Figure 3.)

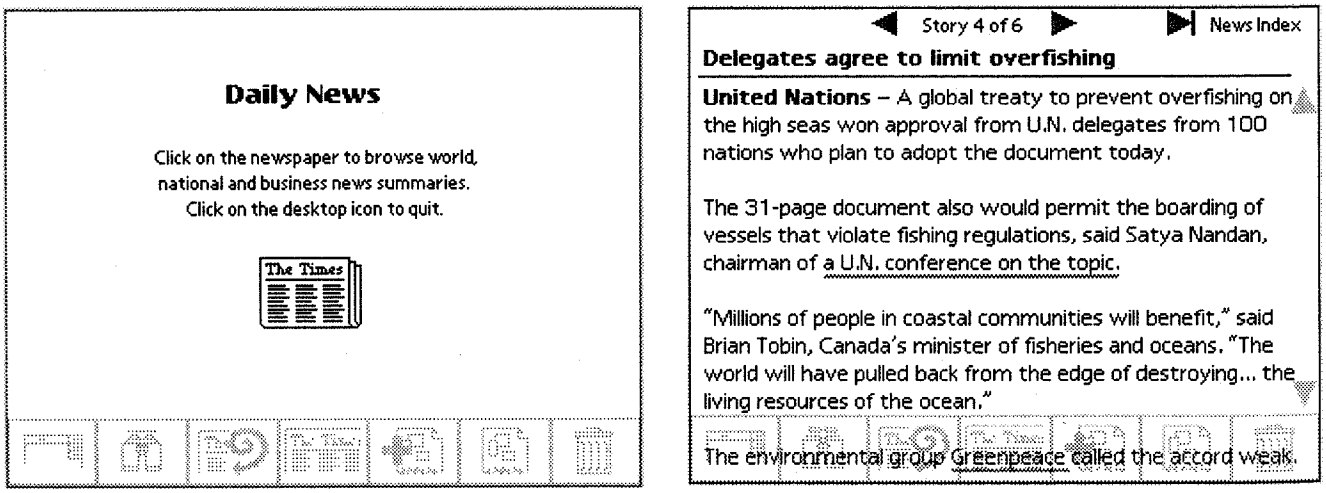


Figure 3. Typical screens in the prototype

The layered icons and text at the bottom of the screen – comprising nearly 20% of the total screen space – were of variable translucency. As the icon layer was made more translucent, the text layer became more opaque, and vice-versa. After a small series of pilot tests, we decided to fix the translucency settings at 80% opacity for the text, and 20% for the icons, to insure good legibility of both icons and text. As we were principally interested in recording the test subjects' reactions to changes in the responsiveness of links and icons, we did not allow the subjects to change the transparency settings, nor did we run the test at other transparency levels.

Sixteen volunteer subjects were selected from among the faculty, graduate students and staff of the Graphics, Visualization and Usability Center, and the School of Literature, Communication and Culture of the Georgia Institute of Technology. Most of the subjects were expert users of the Macintosh or another mouse-driven graphical user interface. Most were familiar with hypertext concepts. Fewer than half had any experience with PDAs.

After a brief training session with the prototype to demonstrate the functions of the semi-transparent icons and the delayed response behavior of the software, each subject was asked to perform eight tasks:

1) From the story index (not shown in Figure 3), select a story title overlapping an icon. (Selecting a story title jumps the user to the first page of that story.)

2) From the story index, select a story title that does not overlap an icon, scroll two pages into the story, and select a link overlapping an icon.

3) From the story index, select a story title that does not overlap an icon, and select an icon that does not overlap with any linked text.

4) On the same page, select a link that does not overlap with any linked text.

5) From the story index, select a story title that does not overlap with an icon, scroll three pages into the story, and select a link overlapping an icon.

6) From the story index, select a story title that does not overlap with an icon, and select a link overlapping an icon. (The "Greenpeace" link shown in Figure 3.)

7) From the story index, select a story title that does not overlap with an icon, And select an icon that does not overlap with any linked text.

8) On the same page, select an icon that does not overlap with any linked text.

(The actual instructions given to subjects were more specific than those listed above.)

Each subject repeated the eight tasks six times: three times with the prototype configured to favor link selection (that is, to initially highlight the link when the mouse was clicked where a link and an icon overlapped, as if the link were on top of the icon), and three times with the prototype configured to favor icon selection. In both selection methods, if the mousebutton were held down for longer than a predefined period of time, the click would appear to pass through the object on top, highlighting the object under it. The association between response delays and layers was rigorously enforced: if selection of links was favored, the subject was required to hold down the mousebutton for the stipulated period of time before any icon would be selected, even if the icon did not overlap with any links.

We further divided the link-first and icon-first selection groups according to the response delay that determined the switching between layers. Each subject attempted all eight tasks for each method of selection with preset delays of 2/5 second and 4/5 second. In the third set of tasks for each selection method, the subjects were allowed to adjust the response delay, between a range of 1/5 of a second and a full second. The sequence in which the subjects performed the series of tasks was shuffled to compensate for learning effects.

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	*Total*
Links-first, 2/5 sec	9	3	5	3	4	0	2	1	27
Links-first, 4/5 sec	11	2	9	1	7	1	3	1	35
Icons-first, 2/5 sec	7	16	8	0	5	8	6	0	50
Icons-first, 4/5 sec	2	11	6	0	7	3	1	0	30
Total	**29**	**32**	**28**	**4**	**23**	**12**	**12**	**2**	**142**

TABLE 1. Raw error counts

("Links-first 2/5 sec" = Links selected first, with a 2/5 second delay between icon and link layers)

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	*Total*
Links-first, 2/5 sec	1.053	0.388	0.793	0.355	0.513	0.000	0.148	0.077	**3.327**
Links-first, 4/5 sec	1.323	0.424	0.833	0.250	0.641	0.077	0.234	0.125	**3.907**
Icons-first, 2/5 sec	0.731	1.469	0.794	0.000	0.504	1.089	0.609	0.000	**5.196**
Icons-first, 4/5 sec	0.361	1.354	0.800	0.000	0.664	0.302	0.167	0.000	**3.648**
Total	**3.468**	**3.635**	**3.220**	**0.605**	**2.322**	**1.468**	**1.158**	**0.202**	**16.078**

Table 2. Normalized error counts

("Links-first, 2/5 sec" = Links selected first, with a 2/5 second delay between icon and link layers)

At the conclusion of the experiment, each subject was given a brief questionnaire that reviewed the test, and included questions asking which of the selection methods was preferred (links first or icons first), and which switching delay (between 1/5 and 1 second) was preferred for each method.

The Results

As the subjects attempted the eight tasks, the prototype recorded every mouseclick, tabulating successful and unsuccessful hits on targets (links or icons). Gross errors (unsuccessful hits) for all subjects for the task series during which we controlled the delay times are summarized in Table 1.

The error counts shown in Table 2 have been normalized so that each subject's contribution to the total errors is (Standard deviations were calculated for each subject to insure that there were no outliers included in the data.) Figure 4 shows a simple line graph of the data in Table 2.

DISCUSSION

After analyzing the results of the experiment, we arrived at the following conclusions:

- Despite an overwhelming preference expressed for the links-first method of selection (15 of the 16 subjects), the overall error rate for the two methods of selection (links-first and icons-first) were very similar.
- The subjects' error rates decreased as they became more familiar with a method of interaction.

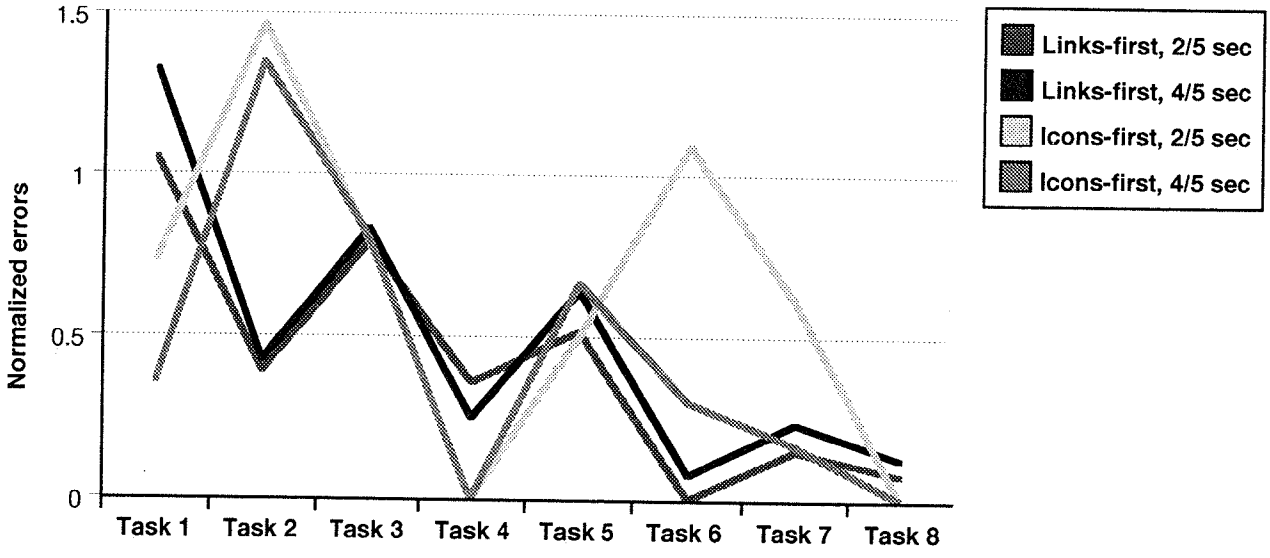


Figure 4. Line graph of data from Table 2

- Subjects had greater difficulty selecting links in icons-first mode, and vice versa. This accounts for the alternating peaks and troughs of Figure 4.
- In the questionnaire, we asked subjects how they would change the behavior of the delayed response. Many expressed the same desire: that all objects in one layer that did not overlap with objects in the other layer should be immediately responsive – in other words, that the subject should not have to wait for the mouseclick to pass through to the lower level if there is nothing on top of it in the upper level. This result surprised us, as we expected that a rigorous and consistent distinction between the layers would make it easier for subjects to conceptualize the two layers.
- The request for immediate responsiveness was given added weight by the subjects' answer to another question. When asked which delay time they preferred, most (12 of 16) responded that they preferred a delay time of 1/5 second – the minimum allowed by the prototype, regardless of which selection method they used. What the subjects appeared to be asking for was a way to simulate immediate responsiveness of non-intersecting objects, within the constraints of the prototype.
- The request was further supported by the difficulties that all subjects had with selection of non-obscured text while using the icons-first selection method. This may account for the high error rate in Task 6, when performed in icons-first, 2/5 second mode (see Figure 4). Users performing the same task in icons-first, 4/5 second mode had less trouble, possibly because they were by Task 6 accustomed to holding down the mousebutton for the extended duration required to select a text link.

DIRECTIONS FOR FUTURE WORK

This expressed preference for immediate responsiveness of non-intersecting objects merits additional research. How, for example, would this change to the delayed-response model affect complex text selection, such as the drag-select gesture supported by many PDAs? How would it affect selection of objects in PDA-based drawing applications? Would novices find this inconsistency in the method of object selection confusing? Would further experiments support the subjects' intuition that immediate responsiveness is a more efficient way to select non-intersecting objects, and might the data from those experiments reveal differences between the links-first and icons-first methods?

CONCLUSION

The very small screens of PDAs severely limit the space available for the display of both text and the widgets required to navigate within or modify that text. A promising method for maximizing the usable screen space of PDAs is to vary the transparency of overlapping objects on the screen, so that objects beneath other objects are still visible, and valuable screen space is not sacrificed in order to display both widgets and information at the same time. In this paper, we have proposed that a variable delay in the response of overlapping widget and text improves the effectiveness of the semi-transparent widget/text model. We assumed that varying the delay by which objects in different virtual layers of the screen responded to users' attempts to select those objects would make it possible for users to select partially-obscured objects without having to resort to a toggle to switch between layers.

Our experiment with a prototype for an online newspaper that uses this selection technique bears out this assumption, and raises additional questions. After an initial learning period, the test subjects were able to select underlying screen objects, regardless of whether those objects were text or icons. Subjects did, however, have greater difficulty successfully selecting objects of one kind when the other kind of object was on the top layer. Our results suggest that it may be easier for subjects to use a variant of the semi-transparency/delayed response model, in which the delay in responsiveness does not apply to objects in one layer which do not appear to intersect with objects in the other layer.

Nothing in our experiment calls into question the merits of techniques for maximizing usable screen space that rely on alternative views of information content. Indeed, it should be possible to combine this semi-transparency/delayed response model with content-based methods for reclaiming space on small screens.

ACKNOWLEDGEMENTS

We show great thanks to the News and Observers who allowed us to user their articles for our internal experimental newspaper.

REFERENCES

1. Ahlberg, C., Shneiderman, B., Visual Information Seeking: Tight Coupling of Dynamic Query Filters With Starfield Displays, CHI'94 Proceedings, pp. 313–17, 1994.

2. Benderson, B., Hollan, J., Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics, UIST'94, pp. 17–26, 1994.

3. Bier, E., Stone, M., Pier, K., Buxton, W., DeRose, T., Toolglass and Magic Lenses: The See-through Interface. SIGGRAPH'93 Conference Proceedings, pp. 73–80, 1993.

4. Card, S., Robertson, G., Mackinlay, J., The Information Visualizer, an Information Workspace, CHI'91 Proceedings, pp. 181–88, 1991.

5. Harrison, B. L., Ishii, H., Vicente, K. J., Buxton, W., Transparent Layered User Interfaces: An Evaluation of a Display Design to Enhance Focused and Divided Attention, CHI'95 Proceedings, 1995.

6. Jog, N., Shneiderman, B., Interactive Smooth Zooming

in a Starfield Information Visualizer, University of Maryland Technical Reports, CS-TR-3286, 1994.

7. Kurtenbach, G., Buxton, W., User Learning and Performance with Marking Menus. CHI'94 Proceedings, pp. 258–64, 1994.

8. Lamping, J., Rao, R., Piroli, P., A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies, CHI'95 Proceedings, 1995.

9. Lieberman H., Powers of Ten Thousand: Navigating in Large Information Spaces, UIST'94 Proceedings, pp. 15–16, 1994

10. Mackinlay, J., Robertson, G., Card, S., The Perspective Wall: Detail and Context Smoothly Integrated, CHI '91 Proceedings, pp. 173–79, 1991.

11. Rao, R., Card, S. K., The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information, CHI'94 Proceedings, pp. 318-322, 1994.

12. Rennison, E., Galaxy of News, UIST'94 Proceedings, pp. 3–12, 1994.

13. Robertson, G. G., Mackinlay, J. D., The Document Lens, UIST'93 Proceedings, 1993.