Improving List Revisitation With ListMaps

ABSTRACT

Selecting items from lists is a common task in many applications. Alphabetically-sorted listboxes are the most common interface widget used to accomplish this selection, but although general they can be slow and frustrating to use, particularly when the lists are long. Also, when the user regularly revisits a small set of items, listboxes provide little support for increased performance through experience. To address these shortcomings, we developed a new list selection device called a ListMap, which organizes list items into a space-filling array of buttons. Items never move in a ListMap, which allows people to make use of spatial memory to find common items more quickly. We compared selection of font names from a set of 220 fonts using both ListMaps and standard listboxes. We found that although listboxes are faster for unknown items, revisitation leads to significant performance gains for the ListMap.

Author Keywords

List selection, listboxes, ListMaps, revisitation.

ACM Classification Keywords

H5.2 [User Interfaces]: Interaction styles.

INTRODUCTION

Selecting items from lists is a common task in current interactive systems, and the most common way to make the selection is through a scrolling list box. For example, people use listboxes to select currencies, languages, font names, or functions from a variety of programs (Figures 1 and 2). Selecting an item from a listbox involves using the scrollbar to perform an alphabetic search until the desired item is visible, and then clicking the item with the mouse.

Although listboxes are general, and will work in almost any task situation, they are not always optimal. One task that listboxes support poorly is selection of items that are well known. In many cases, people use only a small set of items from the list – for example, people normally use a small set of fonts, and select only a few countries or languages out of the possible alternatives.

In these situations, using listboxes can be frustrating. Their alphabetical arrangement makes it equally difficult to find any item in the list, irrespective of the user's dominant interest in a small set of recurring items. Even though the user knows exactly what they want to find, and they have found it many times before, they must scroll through the items in a similar manner to their first search. Although this approach is robust and can be used reliably in any situation where items have textual labels, it requires cognitive effort and focused attention. A listbox provides few opportunities for users to become experts – in particular, alphabetical lists poorly support using memory and pattern recognition as aids to task completion.

Some listboxes provide a shortcut to recently used items by adding a sub-list, which we term a 'recency cache', to the list (see Figure 1, right). This can help, but in cases where the desired item is no longer in the cache, the user is forced to carry out two searches instead of one. Prior research has also shown that adaptive split-menus such as these offer limited performance improvements [7].

In this paper we present and evaluate the ListMap – a new way to interact with list data that allows people to use spatial memory to speed up the selection of previously-seen items. ListMaps organize items into a space-filling array of buttons, with all items visible at once in static locations (see Figure 2). We compared ListMaps and listboxes in a user study where participants were asked to select fonts from a set of 220 names.

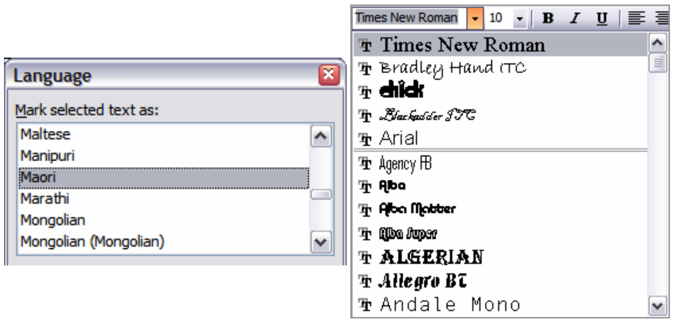

Figure 1. Left: language selection list. Right: font selection list with recency cache (both Microsoft Word XP).

LOCATION MEMORY IN INTERACTIVE SYSTEMS

Spatial object location memory is knowledge of where things are located in a space [8,16]. Spatial knowledge in two-dimensional spaces is built up primarily through interaction; that is, people remember locations after having had experience with that location [5]. People may remember particular items based on landmarks in the space, or with more experience, may be able to maintain a more complete 'mental map' in which they can remember and find many different objects very quickly [8,17].

For example, Robertson and colleagues tested a spatial memory technique (the Data Mountain) in which people placed thumbnails of web pages on a simulated inclined plane [4,17]. Once 100 pages were placed, participants carried out a number of find-and-select retrieval tasks. The study found that retrieval was significantly faster with the spatial technique than with a standard bookmarking system. In addition, the memory of where items were placed persisted over a long time: participants who returned six months later were able to retrieve items at the same level of performance, with only brief retraining [4].

The items in these studies contained symbolic information (thumbnails, colours, icons and names) as well as spatial position. An early study by Jones and Dumais [11] showed that spatial memory fares less well when location is the sole cue to retrieval. In their experiments retrieval of items that were only identified by location was slower and less accurate than when items were represented by name. There is also evidence that location learning is dependent on the amount effort used when interacting with objects [6].

SELECTING FROM LISTS

List selection has been extensively researched, particularly in the form of menu selection. Sears and Shneiderman [18] described 'split menus' for enabling faster selection of frequently accessed menu items. The technique grouped a set of pre-determined menu items 'above the split' at the top of the menu, reducing the target acquisition distance. Their evaluations showed that static split menus, in which the frequent items do not adapt to user actions, allow faster selections than traditional menus. They also suggested further work on split menus that adapt to the user's menu selection patterns. Findlater and McGrenere [7] implemented and evaluated these suggestions, comparing split-menu performance across static, adaptive and adaptable variants. Their results showed that static split-menus are reliably faster than adaptive ones, and that adaptable split-menus are faster than adaptive ones when users are guided by examples.

Finally, scrolling has also been studied in detail, both in the context of lists and in larger text documents. Although scrolling is common, it has been found to have two main problems: it causes motion blur at high scroll speeds, and it causes problems for spatial memory [3,21]. A variety of techniques have been introduced to address the movement issues (e.g., [3]), and work has also looked at better ways of supporting spatial memory (e.g., [13]) in scrollbars.

REVISITATION AND RECENCY CACHING

Learning spatial locations is a function of experience with the items in the data space [5,6]. Therefore, the degree to which a user will be able to build a mental map is related to the amount of revisitation in the task. Different situations have different revisitation patterns, but in many information tasks, users repeatedly go back to a small set of items.

One type of revisitation is brought about by using only a limited set of items. For example, McGrenere [14] found in a survey of Microsoft Word users that people use only a small number of the available commands on a regular basis (a mean of 40 out of 265). Other revisitation arises through recency – in many tasks, a recently-used item is much more likely to be used in the near future than a randomly-chosen item. For example, repetitive patterns of use have been shown in operating system commands [9], and in navigation on the WWW: Tauscher and Greenberg [22] found that more than half of pages seen were revisits, and that revisitation occurs mainly to the last few pages visited – the last ten pages seen cover about 85% of revisits.

Revisitation can be supported with structures like split menus, as described above, or by visualizations of interaction history. Hill et al.'s idea of 'read wear' adds graphical information to computational objects to indicate the history of their use [10]. Depending on how the history is gathered and displayed, the visible marks can be used to determine which items have been visited more recently. A study of 'visit wear' (read wear to indicate which items have been visited) showed that visual recency information can improve revisitation in distorted spaces where memorability is difficult [20].

LISTMAPS: SPATIAL LAYOUT OF LIST DATA

A ListMap takes the items in a list and lays them out in a space-filling two-dimensional grid, ordered alphabetically by row (see Figure 2, right). Items are sized such that they all fit into the window, ensuring that they will all be visible and that no scrolling will be required. Clicking on an item's rectangle in the ListMap is equivalent to clicking the item in a listbox. The idea is therefore similar to the tool palettes used in graphics applications, although ListMaps do not use icons to represent items.

From this basic layout, several additional features are possible to improve selection and search.

- *Labels*. As much of the item's label as possible is written into the rectangle. The letters on each label give a reasonable indication of the item's name (Figure 2, right).
- *Tooltips*. Since the partial labels do not completely differentiate some items, the full name is shown in a tag that follows the user's mouse cursor (see Figure 2).
- *Selection highlight*. The border outline of the currently selected item is highlighted.
- *Colouring*. Items can be coloured to increase visual differentiation in the set. Each item in the ListMap is randomly assigned one of five colours, but colour could also represent other item attributes.
- *Marking*. Items can also be visually marked to indicate attributes such as recent selection (i.e. the recency cache used in the study below, shown in Figure 2), frequency of selection, or user-chosen bookmarks.

The main design principles in a ListMap are that all items are always visible, and that they do not move. This allows users to gradually change from using alphabetic search to spatial memory as their primary search strategy for frequently-used items. The spatial organization is intended to solve the problem described earlier: listboxes require approximately equal search effort regardless of revisitation, but retrieving an item in a ListMap should become easier

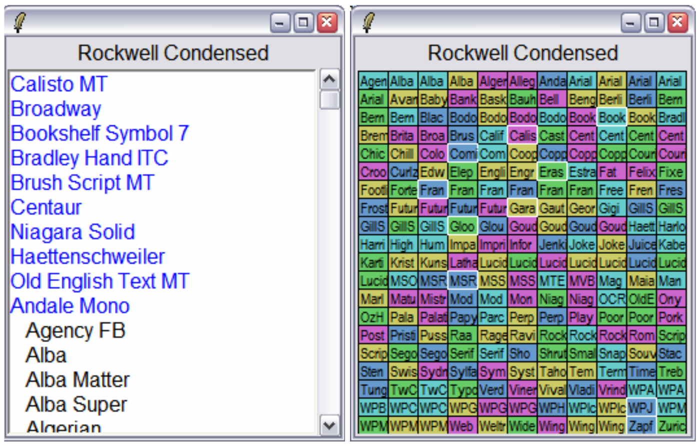
Rockwell Condensed
Calisto MT
Broadway
Bookshelf Symbol 7
Bradley Hand ITC
Brush Script MT
Centaur
Niagara Solid
Haettenschweiler
Old English Text MT
Andale Mono
Agency FB
Alba
Alba Matter
Alba Super
Algerian
Rockwell Condensed
Agen Alba Alba Alba Alger Alleg Anda Arial Arial Arial Arial
Arial Avan Baby Bank Bask Bauh Bell Beng Berli Berli Bern
Bern Bern Blac Bodo Bodo Bodo Bodo Book Book Book Bradl
Brem Brita Broa Brus Calif Calis Cast Cent Cent Cent Cent
Chic Chill Colo Comi Com Coop Copp Copp Copp Couri Couri
Croo Curlz Edw Elep Engli Engr Eras Estra Fat Felix Fixe
Footl Forte Fran Fran Fran Fran Fran Fran Free Fren Fres
Frost Futur Futur Futur Futur Gara Gaut Geor Gigi GillS GillS
GillS GillS GillS Gloo Glou Goud Goud Goud Goud Haett Harlo
Harri High Hum Impa Impri Infor Jenki Joke Joke Juice Kabe
Karti Krist Kuns Latha Lucid Lucid Lucid Lucid Lucid Lucid Lucid
Lucid MSO MSR MSR MSS MSS MTE MVB Mag Maia Man
Marl Matu Mistr Mod Mod Mon Niag Niag OCR OldE Ony
OzH Pala Palat Papy Parc Perp Perp Play Poor Poor Pork
Post Pristi Puss Raa Rage Ravi Rock Rock Rock Rom Scrip
Scrip Sego Sego Serif Serif Sho Shrut Smal Snap Souv Stac
Sten Swis Sydn Sylfa Sym Syst Taho Tem Term Time Treb
Tung TwC TwC Typo Verd Viner Vival Vladi Vrind WPA WPA
WPB WPC WPC WPG WPG WPG WPH WPIc WPIc WPJ WPM
WPM WPM WPM Web Weltr Wide Wing Wing Wing Zapf Zuric

Figure 2. Listbox interface (left) and ListMap (right), both showing recency cache. Cue for next selection is shown above each interface. Only one system was on screen at a time.

and faster with repeated retrieval as spatial memory improves. This theoretical advantage is tested in the experiments described later.

The graphical layout of a ListMap limits the number of items that can be shown in a given space, whereas a listbox can show any number of items in the same area. Based on the sizes of lists and listboxes in current interfaces, however, it appears that ListMaps would not require major interface changes. The minimum size of an item rectangle that contains one letter of a label is approximately 10x10 pixels; this allows more than 750 items to be shown in a 240x320 space.

COMPARISON STUDY: LISTBOXES AND LISTMAPS

To evaluate the idea of organizing list items spatially, we carried out a study in which we compared retrieval performance with standard listboxes and ListMaps.

Participants

Twelve participants (7 women and 5 men) were recruited from a local university. Participants ranged in age from 20 to 35 years and averaged 25 years. All were familiar with mouse-and-windows applications (i.e., more than 8 hours per week) and all used word processing applications regularly (at least 1 hour per week).

Apparatus

A custom system was built in Tcl/Tk for the experiment. The system presented either a listbox or a ListMap and prompted users to select sequences of items in different experimental conditions. The study was conducted on a P4 Windows system with a standard optical mouse (including a mouse wheel) and a 1024x768 display.

Interfaces used in the study

Two versions of both the listbox and the ListMap were used, one with a recency cache, and one without. The basic version of the listbox displayed an alphabetical list of items, shown in 12-point Arial (see Figure 2). The user could navigate the list in four ways: by clicking the up and down arrow buttons on the scrollbar; by dragging the scroll thumb; by clicking in the trough above or below the scroll thumb; and by using the scroll wheel on the mouse. We did not include keyboard bindings because many situations (e.g., pen-based computers) do not allow keyboard input. The recency-cache version of the listbox added a split-menu of items to the top of the list, duplicating items that had been recently selected. The recency set could hold at most ten items. Each new selection copied the item to the top of the split-menu, moving all others down by one position, and causing the removal of the 10th item from the recency cache. The items in the recency cache were displayed in blue with less indentation than others (see Figure 2) to clearly set them apart from regular list items.

The basic version of the ListMap worked as described earlier: items were arranged alphabetically in rows, randomly tinted with one of five colours (pilot studies showed that the random colouring helped people to remember locations), and annotated with the first few letters of the item's name (see Figure 2). The full name of the item under the mouse cursor was shown in a floating box beside the cursor, and the item currently under the cursor was highlighted with a white border. The only difference between the recency-based ListMap and the basic version was that in the recency version, the last ten items selected were highlighted with a white border (see Figure 2).

In all versions of both interfaces, the size of the display window was 255 x 280 pixels, which is also the size of the font menu of MSWord, as seen in Figure 2.

Tasks, Experimental Conditions and Dataset

The study used a set of 220 font names as the data items for both interfaces. This dataset is directly taken from a real-world task – selecting font names from a listbox in a word processing program. The list was taken from the set of fonts included with a standard distribution of Microsoft Office.

Each selection trial was cued within the user interface by showing the name of the next target font below the window title-bar (Figure 2).

The tasks with each interface were administered in blocks of ten trials, with six blocks for each of three experimental conditions. Each condition is designed to compare how well the listbox and ListMap interfaces support a particular style of interaction, as follows:

- *Random selection*. This condition was administered without the recency-cache features of the listbox or ListMap interfaces. It was designed to provide baseline values for selecting items that the user does not use on a regular basis (i.e., no revisitation). Target items were chosen randomly from the full set of 220 font names.
- *Revisitation*. This condition was administered without the recency-cache features of the listbox or ListMap

interfaces. It was designed to examine whether, and how quickly, the participants' performance improves on repeated iterations with both interfaces in the absence of explicit recency support. In this condition, the participants repeatedly selected items from a working set of ten items. Each of the six blocks contained one trial for each of the ten times, but in a random order.

- *Revisitation+recency*. In this condition, the recency-cache features of the listbox and ListMap interfaces were enabled. It is designed to compare the effectiveness of the recency features across the two interface types. This condition also used a more realistic notion of revisitation than the *revisitation* condition by adding some non-revisited elements. For each item presented to the user, there was an 80% chance that the item would be drawn from the working set of ten fonts (the same ten used above), and a 20% chance that the item would be chosen randomly. Therefore, there was an 80% chance that the target would be in the recency cache (i.e., in the recency list for the listbox, or highlighted in white for the ListMap). This manipulation of revisitation probability is intended to provide insights into whether 'noisy' tasks (outside the routine set) interfere with the benefits of the recency cache with either interface.

Procedure

Participants were first introduced to the two different list interfaces, but populated with a different dataset to the experimental conditions. People carried out ten practice trials with each interface. They were then introduced to the study system and font dataset, and were randomly put into one of two order groups (listbox first or ListMap first).

Completing each trial within a block caused the next font target to be immediately displayed in the title bar. Software automatically logged task completion time and all item selections, including incorrect selections. Incorrect selections had no effect on the interface state (the same target remained displayed), and the task time continued to accumulate regardless of errors.

All of the participants completed all of the blocks with one interface before proceeding to the other. With each interface, the six blocks within each of the three experimental conditions were always completed in the order *random-selection*, *revisitation*, *revisitation+recency.*

After completing the six blocks of ten trials with each interface/condition combination, the participants completed a short preference questionnaire; the questionnaire was also given at the end of the study to capture overall preferences.

Study Design

Data from each of the three conditions are separately analysed in 2×6 repeated measures designs for factors *interface-type* (listbox or ListMap) and *block-number* (first to sixth block). The primary dependent measure in all analyses is task time. Error data and questionnaire responses are also analyzed and reported.

RESULTS

We organize the results below by the three experimental conditions: random selection, revisitation and revistation+recency. Across all 4320 trials (12 participants, 60 trials, 3 conditions, 2 interfaces) the tasks were completed quickly (overall mean 4.9s, sd 1.9), with few errors (< 6%).

Random selection

The ListMap interface was slower than the listbox interface in random tasks ($F_{1,11}$=12.4, p<.01), with means of 6.6s (sd 2.3) and 5.3s (sd 1.4). There was a reliable main-effect for trial block ($F_{5,55}$=7.7, p<.01), with task times reducing across the first and second blocks, but relatively stable performance thereafter (see Figure 3).

There was no interface×block interaction ($F_{5,55}$<0, p=.4), suggesting that neither interface provided a marked experience-based advantage over the other with randomly selected targets.

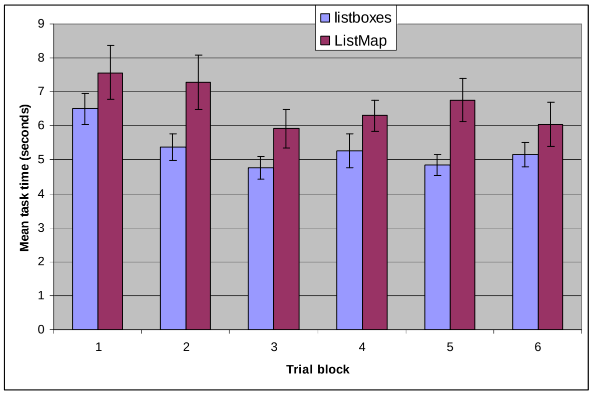

Figure 3. Mean completion times by trial block, random-selection task. Error bars show standard error.

Revisitation

The ListMap interface provided a significant performance advantage over the listbox interface in the resivitation condition ($F_{1,11}$=9.9, p<.01), with means task times of 3.8s (sd 1.4) for the ListMap, and 4.6s (sd 1.3) for the listbox. There was also a significant main effect of trial block ($F_{5,55}$=7.7, p<.01), with more gradual mean time improvement through blocks one to four than observed in the random selection condition (see Figure 4).

Unlike the random condition, there was a significant interface×block interaction ($F_{5,55}$=2.8, p<.05). The interaction, apparent in Figure 4, is explained by the steeper and more continual performance improvement across blocks when using the ListMap interface. Across the six blocks, mean performance with listboxes improved by only 0.28s (6%), compared to 1.74s (35%) with ListMaps.

We analysed the fit of the data to the power law of practice [15], a robust model of human skill acquisition. It states that performance time improves across trials according to the following formula: $\log(T_n) = C - \alpha \log(n)$, where T_n is the

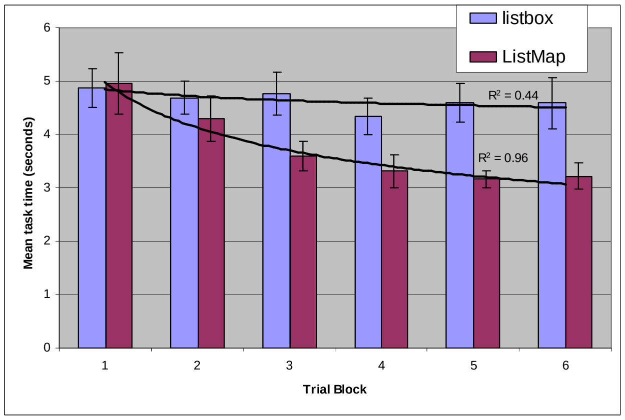

Figure 4. Mean completion times for revisitation task. Power-law regression line of best fit is overlaid.

time to complete trial *n*, *C* is the time on the first trial, and *α* is the steepness of the learning curve. Regression analysis of performance across blocks with ListMaps shows an almost perfect fit with the power law of practice formula, with R^2=.96, p<.05, and α=3.58. The listbox data, however, poorly fits the model, with R^2=.44, p=.11 (Figure 4). This suggests that listboxes poorly support traditional models of skill acquisition.

Revisitation+Recency

To recap, trials in the revisitation+recency condition were generated with an 80% probability of being within the revisitation set and a 20% probability of being randomly selected. In addition, the revisitation set was the same as that used in the previous task, to simulate a situation where the user knows a number of items well. This section examines overall performance in this condition; the next section examines the effectiveness of the recency-cache facilities, which were only present in this condition.

Mean performance with ListMaps (mean 4.1s, sd 1.7) was approximately 17% faster than listboxes (5.0s, sd 1.6), giving a significant main effect for interface: $F_{1,11}$=9.8, p<.05. The performance advantage of the ListMap over the listbox (0.9 seconds) is approximately equal to that in the previous task (0.8 seconds); this means that a small amount of randomness in the items to be retrieved does not disrupt the overall advantage of the ListMap.

There was less of a performance improvement across trial blocks than in the previous task, likely because participants were already familiar with the revisitation set. There was also no significant interface×block interaction ($F_{5,55}$=0.3, p<.9). The slower performance in the first block may be due to the fact that the participants also had to get used to the new recency-based interface during these trials; after this block, performance improved only marginally. Figure 6 shows that performance continues roughly on the line established in the revisitation task (if we ignore block one).

We also looked at whether the origin of the item (revisited or random) affected performance in similar ways to that

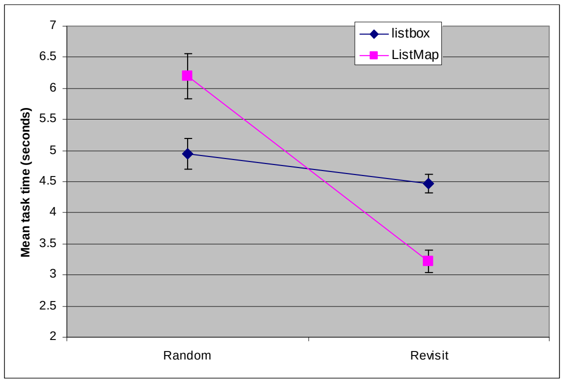

Figure 5. Crossover between random and revisited items for the revisitation+recency task

found in the first two tasks. In a post-hoc analysis, there was a significant interface×origin interaction: $F_{1,11}$=26.2, p<.01. The interaction is apparent in the cross-over effect (Figure 5), caused by listboxes outperforming ListMaps with random targets, and the inverse for revisited ones.

Effectiveness of the recency caches

The recency cache facilities were only present in the revisitation+recency condition. To investigate the effectiveness of the cache facilities in the listbox and ListMap interfaces, we re-analysed the data from the revisitation condition together with the data from the revisitation trials in the revisitation+recency condition. The analysis used a 2×2×6 design for factors *interface-type* (listbox or ListMap), *caching* (absent in revisitation, present in revisitation+recency), and *block*.

As expected from the prior analyses, there was a significant main effect for interface type ($F_{1,11}$=23.7, p<.01), with ListMaps (3.5s, sd 1.3) outperforming listboxes (4.6s, sd 1.2) on revisited data. There was a marginal main effect for caching ($F_{1,11}$=4.2, p=0.065), with a no-caching mean of 4.2s (sd 1.4) versus caching 3.8s (sd 1.3). Performance across blocks improved significantly: $F_{5,55}$=5.1, p<.01.

There was a significant caching×block interaction ($F_{5,55}$=3.6, p<.01), which is probably caused by relatively stable performance across blocks with ListMaps compared to variable performance with listboxes (see Figure 6).

We predicted an interface×caching interaction, because we believed that the ListMap's static highlighting of recent items would help users more than the listbox's adaptive contents. However, this prediction was not supported by the data, with no significant interaction: $F_{1,11}$=1.6, p=0.2

Errors

Errors were measured as the total number of incorrect selections per block divided by the number of targets per block. The overall error rate was low (mean 0.04, sd 0.09), ranging from zero to 0.6.

Analysing errors in each of the three conditions (random, revisitation, and revisitation+recency) using 2×6 ANOVAs for *interface-type* and *block* showed no significant main

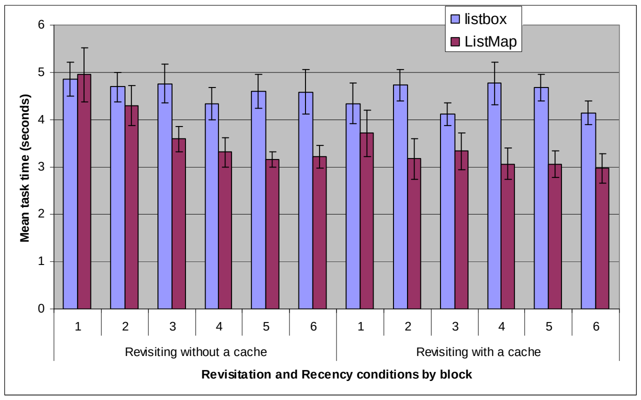

Figure 6. Mean completion times for revisited items only, for both revisitation task and revisitation+recency task.

effects or interactions. Figure 7 summarizes error rates across the three conditions for both interface types. Although not significant ($F_{1,11}$=4.2, p=0.07), the error rate for ListMaps when caches are present was more than double that of the listbox interface. Despite this high error rate, participants completed their selections faster with the ListMaps, suggesting that the ListMap recency cache may have promoted hasty commitment to selections. This behaviour may have arisen because participants knew that there was little cost for guessing incorrectly. Further study of this result is needed, since in the real world the cost of an error could be higher (e.g., reposting a dialog).

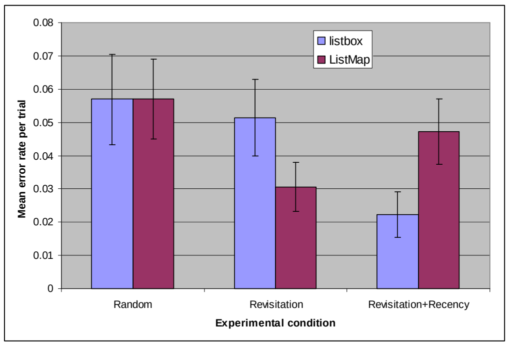

Figure 7. Mean error rates for all tasks.

Preferences
After each condition, we asked participants to state which of the two interfaces they thought was easier to use, which they thought was faster, and which they preferred overall. As shown in Table 1, preference was strongly in favour of the listbox after the random condition: only one person preferred the ListMap, and only two thought that it was the faster interface. After the Revisitation condition, half of the participants thought that the ListMap was faster, but still only three preferred it overall. After the revisitation+ recency condition, a majority (of seven) thought the listbox was faster, although people still thought that the listbox was easier, and preferred it overall.

At the end of the session, we also asked participants whether they would choose to use a ListMap if the widget were available in the real applications that they used on a regular basis. Even though the ListMap was not people's preferred interface during the tasks, nine of the twelve participants stated that they would use the ListMap in a real-world application.

DISCUSSION
We draw several main conclusions from the user study:

- For random selection, listboxes outperform ListMaps;
- However, real world selection is seldom random, and when users revisit items, ListMaps outperform listboxes;
- Revisitation performance with ListMaps fits models of skill acquisition extremely well, but listboxes do not – suggesting that listboxes trap users in 'beginner mode';
- Recency caches appear to have little effect with listboxes. This supports prior work indicating that adaptive split menus do not aid menu selections [7]. With ListMaps, however, the recency cache appears to assist revisitation, although possibly at the cost of higher errors;
- The speed advantage of the ListMap becomes apparent after participants had revisited items twice;
- Users preferred the listbox, although by the last task, a majority felt that they were faster with the ListMap.

In the next sections, we deal with several issues raised by the study and by our experiences with the ListMap. We consider reasons for the performance differences between the ListMap and the listbox, we address several issues in the design and use of the technique, and we look at ways to address the ListMap's poor preference scores.

Why was the listbox faster for random retrieval?
The listbox had three advantages in the random-selection condition. First, participants were far more experienced with listboxes than they were with ListMaps, and they were all extremely well-practiced at finding items using the traditional method. Second, the ListMap provides less visual search information than does the listbox – that is, only three or four letters of the font's name were visible in the map rectangle. As a result, it is more difficult to visually pick out a particular item, and users often had to carry out a horizontal scan with the mouse, watching the pop-up text to find the correct item.

Finally, the two-dimensional alphabetic arrangement of the ListMap appeared to be more difficult for unknown items than the one-dimensional arrangement of the listbox. On a few occasions, we observed participants going the wrong direction in the ListMap; one participant also stated that they found it more difficult to search for things in rows compared to looking in the vertical list.

Why was the ListMap faster for revisitation?
It seems clear that the speed advantage of the ListMap comes from the better support for spatial memory that exists in the map representation. As items are revisited,

people start to remember where they are, and it becomes easier to get back to them in future. Although a certain amount of spatial memory could be used in the listbox (by remembering the location of the scroll thumb), the cue is much less specific than it was in the ListMap.

Several participants stated that by the end of the study, they had memorized the locations of many of items in the working set. They stated that some items were easier to remember than others: for example, 'Arial Black' was easier since it was in the top row of items; 'Goudy Stout' was more difficult since it was in the middle.

The advantages of using spatial memory as the recency cache become clear in situations where an item is no longer in the widget's recency set. In the ListMap, even when items are no longer highlighted, people remember roughly where they are – that is, people use the highlight primarily to refine their targeting action, and they are likely to be close (based on spatial memory) regardless of the highlight. In contrast, when an item moves out of the listbox's cache, the recency support simply fails: there is no way to be 'close' if the item is not in the sublist, and a whole new search is required.

These principles mean that there is a natural correlation in the ListMap between amount of use and retrieval performance – a desirable state in any interface. Items that are used often, get remembered better, and so become easier to find and faster to retrieve. The persistence of spatial memory (as shown by [4]) also suggests that the 'timeout' period for spatial memory is much longer than what could be used in an interface-based recency cache.

A final issue here involves the amount of revisitation that is needed before a ListMap will be more effective than a listbox. Based on the data of the revisitation+recency task, a crossover point can be estimated; if roughly 50% or more of the selections are revisits, a ListMap should be the better display. It should be noted, however, that selections in the real world are rarely random: if not from a frequently-used working set, they are often from a 'familiar' set [14], which should improve the overall performance of the ListMap.

Questions about ListMap Design and Use

What if two applications use different ListMaps for fonts?
The underlying principles of the ListMap are that all items should be visible, and that items should not move. Therefore, any change to the list data or to the arrangement of the map could have negative effects on performance with a ListMap. Although undesirable, people can learn multiple mappings for the same information — for example two different keyboards, or two different key-bindings for cut-copy-paste. The problems are that learning a second mapping is difficult after you already know one, and that even once the two mappings are both well learned, people will make mode errors when they forget which mapping is current. We believe that the best way to deal with this is to have standardized ListMaps for common lists (like fonts) that are available at the widget level, ensuring that the maps are consistent across applications.

What if you use a computer with a different set of fonts?
Spatial knowledge from one ListMap will rarely transfer to a different map of the same data type. However, there are many examples of specializations that improve performance for the local user but that are not transferable to other systems: for example, the way that one person organizes their menus, toolbars, desktop icons, or system preferences often means that they are less effective at someone else's workstation. Nevertheless, these specializations are valuable, since the majority of most people's work is done on a single computer that they can tailor to their personal context. If ListMaps can save a second or more from every list selection, and save the frustration of dealing with listboxes, many users will be willing to accept the narrowness of the solution.

What if you add a new font to your system?
The third type of change that can happen to a ListMap involves gradual additions over time. We plan to test the effects of small changes on ListMap retrieval time, but we believe that people will be able to adapt quickly. It is also possible to design the map to be more resilient to change: for example, leaving one column of blank rectangles as 'expansion slots' would allow several items to be added without the need to shift any items to the next row.

What if you want to see the actual appearance of the fonts?
The space-filling representation of the ListMap does not leave enough room to show details such as the appearance of a font. Although it would be simple to use the actual font in the pop-up box, ListMaps are not designed for browsing.

How many ListMaps can people learn?
Our informal tests suggest that people can learn to use more than one ListMap, and some test users have successfully learned working sets in three different maps in about one hour. These intensive trials are very unlike the longer-term and less-frequent use that characterizes real-world interaction with list data, and so further study is needed on this question. However, the range and persistence of spatial memory in everyday life suggests that people will be able to learn multiple maps over time.

What about keyboard selection from lists?
Some lists let users type a key to step through the items starting with that letter. We did not include this capability in our study, since we are interested in situations where there is no keyboard, such as pen-based systems, or situations where the user works primarily with the mouse. Nevertheless, keyboard input does provide an alternate list-selection mechanism to scrolling the listbox, and it allows a shortcut to a particular region of the list. We believe that keyboard bindings would benefit the ListMap as much as they would the listbox – that is, it would be possible to add this functionality to the ListMap using a 'current selection' highlight on the map to give feedback as the user stepped through items starting with a particular letter.

Improving preferences for ListMaps

Even though the ListMap was significantly faster (and participants' subjective responses recognized this), they often still preferred the listbox. We believe that the ListMap's lack of popularity is due to the difficulty that people have in finding things for the first time.

This drawback of the ListMap could be reduced by combining the two widgets, allowing users to switch back and forth between them. Listboxes currently do not map the right mouse button; we are building a prototype widget that lets users switch from ListMap to listbox with a single right click. Users can choose the interface that they want for the particular task at hand. In order to help build the spatial map of revisited items, our prototype switches back to the ListMap after a listbox selection, and highlights the item that was chosen. We plan to test whether this dual representation will allow people to gradually switch from the general-but-limited listbox, to the higher-performance ListMap. (The prototype can be downloaded from web.address.domain/ListMap/).

CONCLUSION

Selection from lists is a frequent task in most user interfaces, and the most common mechanism for this task is the scrolling listbox. Although listboxes are general, they do not allow people to capitalize on revisitation, and they limit the amount that performance can improve. We introduced and evaluated an alternate interface for list selection – the ListMap – that lets people use spatial memory to improve performance with increased experience. A user study showed that ListMaps allow significantly faster retrieval when users revisit items, and we estimate that ListMaps will be faster overall when fifty percent or more of a user's selections are revisits.

In future, we plan to carry out further studies of the ListMap and the idea of spatial organization. As mentioned above, we will test people's ability to learn and use multiple maps, and will examine the effects of gradual change to the list data. We plan to extend ListMaps for use on space-constrained devices such as mobile phones and PDAs, and we will also explore the idea of spatial shortcuts in other forms: for example, a row of buttons above a traditional listbox that can be set by the user to stand for high-frequency items in the list. Finally, we will test the composite listbox/ListMap widget in a realistic interface setting, and track people's usage patterns in a longer-term task setting.

REFERENCES

1. Baddeley, A. (1990). *Human Memory*. Hove: Lawrence Erlbaum Associates.
2. Bederson, B. (2000). Fisheye Menus, *Proc. ACM UIST 2000*, 217-225.
3. Cockburn, A., Savage, J., and Wallace, A. (2005). Tuning and Testing Scrolling Interfaces that Automatically Zoom, *Proc. ACM CHI 2005*, 71-80.
4. Czerwinski, M., van Dantzich, M., Robertson, G., and Hoffman, H. (1999) The Contribution of Thumbnail Image, Mouse-over Text and Spatial Location Memory to Web Page Retrieval in 3D Viewing, *Proc. IFIP INTERACT 1999*, 163-170.
5. Darken, R., and Sibert, J., Wayfinding in Large-scale Virtual Environments, *Proc. ACM CHI 1996*, 142-150.
6. Ehret, B. (2002). Learning where to look: location learning in graphical user interfaces, *Proc. ACM CHI 2002*, 211-218.
7. Findlater, L., and McGrenere, J. (2004). A comparison of static, adaptive, and adaptable menus, *Proc. ACM CHI 2004*, 89-96.
8. Golledge, R., and Stimson, R., *Spatial Behaviour*, New York: Guilford Press, 1997.
9. Greenberg, S., and Witten, I. H. (1993). Supporting command reuse: Mechanisms for reuse. *IJMMS*, 39(3), 353-390.
10. Hill, W., Hollan, J. Wroblewski, D. and McCandless, T., Edit Wear and Read Wear. *Proc. CHI 1992*, 3-9.
11. Jones, W., and Dumais, S., The Spatial Metaphor for User Interfaces: Experimental Tests of References by Location versus Name. *ACM TOIS*, 4, 1986, 42-63.
12. Lynch, K., *The Image of the City*. Cambridge: MIT Press, 1960.
13. McCrickard, S., and Catrambone, R., (1999). Beyond the Scrollbar: An Evolution and Evaluation of Alternative Navigation Techniques, *Proc. IEEE Visual Languages 1999*, 270–277
14. McGrenere, J., and Moore, G. (2000). Are we all in the same "bloat"?, *Proc. Graphics Interface'00*, 187-196.
15. Newell, A., and Rosenbloom, P. (1981). Mechanisms of skill acquisition and the law of practice. In J. Anderson, ed., *Cognitive skills and their acquisition*. Hillsdale, NJ: Lawrence Erlbaum, 81-135.
16. Postma, A., & De Haan, E. (1996). What Was Where? Memory for Object Locations, *Quarterly Journal of Experimental Psychology*, 49A (1), 178-199.
17. Robertson, G., Czerwinski, M., Larson, K., Robbins, D., Thiel, D., and van Dantzich, M., Data Mountain: Using Spatial Memory for Document Management, *Proc. ACM UIST 1998*, 153-162.
18. Sears, A., & Shneiderman, B. (1994). Split Menus: Effectively Using Selection Frequency to Organize Menus. *ACM ToCHI*, 1(1), 27-51.
19. Shneiderman, B. Tree Visualization with Tree-maps: A 2-D space-filling approach. *ACM Transactions on Graphics*, 11, 1, 1992, 92-99.
20. Skopik, A. and Gutwin, C. (2005). Improving Revisitation in Fisheye Views with Visit Wear, *Proc. ACM CHI 2005*, 771-780.
21. Smith, R., and Taivalsaari, A. (1999). Generalized and Stationary Scrolling, *Proc. ACM UIST 1999*, 1-9.
22. Tauscher, L. and Greenberg, S., Revisitation Patterns in World Wide Web Navigation. In *Proc. ACM CHI 1997*, 398-406.