Improving Command Selection with CommandMaps

Joey Scarr[†], Andy Cockburn[†], Carl Gutwin[‡], Andrea Bunt[*]

[†]Computer Science
University of Canterbury
Christchurch, New Zealand
{joey,andy}@cosc.canterbury.ac.nz

[‡]Computer Science
University of Saskatchewan
Saskatoon, Canada
gutwin@cs.usask.ca

[*]Computer Science
University of Manitoba
Winnipeg, Canada
bunt@cs.umanitoba.ca

ABSTRACT

Designers of GUI applications typically arrange commands in hierarchical structures, such as menus, due to screen space limitations. However, hierarchical organisations are known to slow down expert users. This paper proposes the use of spatial memory in combination with hierarchy flattening as a means of improving GUI performance. We demonstrate these concepts through the design of a command selection interface, called CommandMaps, and analyse its theoretical performance characteristics. We then describe two studies evaluating CommandMaps against menus and Microsoft's Ribbon interface for both novice and experienced users. Results show that for novice users, there is no significant performance difference between CommandMaps and traditional interfaces – but for experienced users, CommandMaps are significantly faster than both menus and the Ribbon.

Author Keywords

Expertise; spatial memory; commands; hierarchies.

ACM Classification Keywords

H.5.2 [User Interfaces]: Interaction Styles.

INTRODUCTION

Most GUI applications provide access to commands using visual components such as menus, toolbars, or the Ribbon interface seen in Microsoft Office. When an application has a large number of commands, designers often use a hierarchical navigation structure to partition the components (e.g., with menus or Ribbons) – partly to save screen space, but also to provide semantic groupings of commands (e.g., "File," "Insert," or "View") that simplifies search for novice users. However, hierarchical structures have been shown to be less efficient for expert users (e.g., [7]) – experts already know which commands they want and where those commands are, but a hierarchical selection widget requires additional navigation actions that take more time and increase the chance of navigation errors.

This problem has been recognized by researchers, and alternative command-selection techniques have been studied that allow better performance for experts. For example, command languages, marking menus, and shortcut keys have all been shown to perform better than standard controls (e.g., [27, 30]). These alternative approaches gain their performance advantage through the use of flat (rather than hierarchical) organizations of commands, and rapid memory-based selection mechanisms. For example, when people become experienced with marking menus or shortcut keys, they begin to retrieve the correct command using muscle memory rather than visual search; similarly, experts with command languages use retrieval of the correct command from memory.

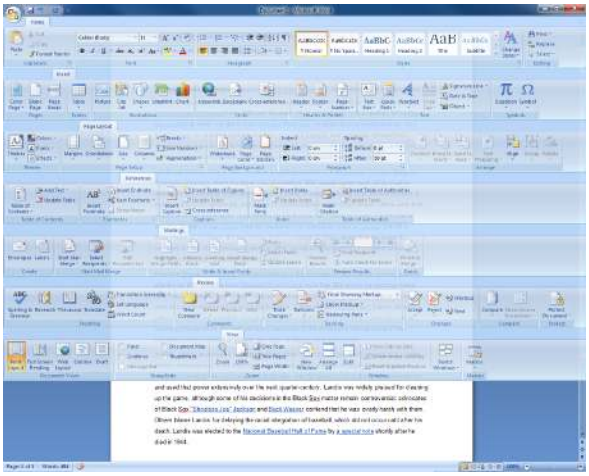


Figure 1. An example CommandMap for Microsoft Word.

Although these techniques have been shown to be effective, they have characteristics that may not fit well with existing GUI styles. Most WIMP (Windows, Icons, Menus and Pointer) based systems use a strongly visual presentation style because of its advantages for novices, and are heavily invested in existing widget types (like standard menus and Ribbons); this means that it may be difficult to ask users to switch to a radically different interaction paradigm such as a command language; in addition, these systems are most often used with a mouse, which can make gesturing (as used with marking menus) more difficult.

What other kind of fast retrieval could be used to improve expert performance in traditional GUI applications? In this paper, we explore the use of spatial memory as a fast retrieval mechanism that could replace hierarchical selection techniques, and that can fit the general appearance and presentation style of GUI systems. Previous research has

CHI'12, May 5–10, 2012, Austin, Texas, USA.

shown that spatial memory is a powerful and persistent mechanism for fast retrieval (e.g., [17, 32]), but this idea has not been studied in detail for command interfaces, other than in a few small experiments.

One inspiration that spatial memory can be used in this way comes from anecdotes about expert use of complex applications such as AutoDesk's Maya. Experienced users of these systems often arrange several visual toolbars in a stable spatial arrangement, and then hide and show the tools when needed. Following from these examples, and as a way to evaluate the effectiveness of spatial memory as a command-selection mechanism, we developed a technique called *CommandMaps (CMs)*. CMs have two main properties: they show all (or at least a substantial fraction) of an application's commands at once, and they do so in a spatially-stable fashion, allowing users to build up spatial memory of frequently-used commands (Figure 1).

We carried out studies to compare the performance of CMs to standard GUI command-selection techniques (menus and Ribbons) both for experts and novices. We found that for novices, there were no overall differences between CMs and the standard GUI techniques, showing that a spatial memory approach does not impose an extra burden when users are just starting out with an interface.

When users had more experience with the interface, there were much larger differences in favour of CMs. Selections with CMs needed significantly less time than both menus (34% faster) and Ribbons (25% faster); furthermore, the error rate with CMs was one-tenth of the other interfaces. CMs were also strongly preferred by participants.

These results show that spatial memory can be successfully used as a command-selection mechanism in GUI interfaces, and that the CommandMaps instantiation of this idea should be considered by UI designers as a way to dramatically improve performance ceilings for expert users.

RELATED WORK

Interfaces for Improved Performance with WIMP

User performance in WIMP interfaces is dominated by two operations. The first is the need to locate a desired command among those available, and the second is the time to select it using the mouse (or other similar device). Pointing time is commonly modelled using Fitts' Law [15], a logarithmic function of target width and distance from the cursor. The time to locate a target, on the other hand, has been shown to depend on the users' expertise or familiarity with the interface [9]. Novice users must rely on visual search (typically a linear function of target count), while experts can decide about their location (a log function [19, 20]).

Improvements to traditional WIMP interfaces have sought to make accessing commands more efficient by reducing either pointing time or search time. One such line of work involves alternative command organizations. For example, *pie menus* [5] aim to reduce pointing time by having menu items centred around the cursor when the menu is invoked. *Marking menus* extend pie menus by allowing experts to leverage their spatial knowledge using gestural selections that pre-empt menu display [22]. While keyboard methods, such as shortcut keys, can also reduce pointing time [27], few users make the transition from mouse to keyboard [30].

Work on adaptive interfaces has examined using past user behaviour to either spatially promote likely commands [13, 16, 25] or to visually highlight them [14, 16]. Theoretically, spatial relocation has potential benefits in reducing pointing time and visual search time (if users perform a top-down linear search; see [4] for an analysis of visual search paths). However, empirical evaluations demonstrate that spatial relocation can harm performance [16, 25], and performance models attribute this to the increased reliance on visual search rather than rapid decision [9]. Adaptive visual highlighting aims to leverage visual pop-out effects to decrease visual search time by focussing the search space. For example, Findlater *et al.* [14] empirically demonstrate that 'ephemeral adaptation' improves menu selection performance. However, the benefits of the technique are likely to diminish as users gain expertise in target location.

Spatial Memory

Considerable research on human memory of object locations has been carried out, both in psychology (e.g., [2, 3, 28]) and in HCI (e.g., [10-12]). Much psychology work has been done on memory for navigation: for example, Thorndyke [33] divides spatial knowledge into three types: landmark knowledge, procedural or route knowledge, and survey knowledge (a global overview of the space). Survey knowledge is related to object location memory (the type of memory at issue in this research), and studies have shown that expert human retrieval of object locations is governed by the Hick-Hyman Law [19, 20] (which states that retrieval time is proportional to the log of the number of items in the set), and that spatial learning is governed by a power law of practice [26] (which states that performance improves quickly at first, and levels off with experience).

Several researchers in HCI have explored the use of spatial memory in computer interfaces, and studies have shown that although abilities can vary widely [31], people are capable of using spatial memory to remember large numbers of items, and retrieve them quickly. For example, retrieval of 100 web pages using the memory-based Data Mountain technique [29] was significantly faster than with a standard bookmarking system, and the spatial memory also persisted over several months [10]. Other research, however, suggests that the form of presentation is critical, and that when location is used as the only retrieval cue, spatial memory fares less well [21].

There are relatively few studies that investigate spatial memory as a command-selection mechanism for interfaces. One of these is the *ListMaps* interface [17], which showed that a 15x15 grid of buttons was faster for experts than a linear list of 225 alphabetical items, but considerably slower for novices. This work indicates that the potential value of

spatial memory as a fast retrieval technique must be balanced against the time it takes to learn item locations. Another study tested a spatially-stable arrangement of page thumbnails as a document-navigation interface, and showed that spatial memory outperformed scrolling (and that the difference increased dramatically with revisitation) [8].

Hierarchical Navigation

Three decades of research since Miller's [24] analysis of performance with different menu structures has produced extensive and apparently conflicting empirical evidence of the relative merits of 'broad and shallow' versus 'narrow and deep' hierarchical structures. Recent work, however, demonstrates that the apparent conflict between study results can be explained by differences in the experimental conditions [7] – specifically, performance improves with breadth (shallow hierarchies) when item selection performance is a logarithmic function of number of candidate items; but performance follows a 'U' shape with breadth when selection performance is a linear function of the number of items. Logarithmic performance is possible when users can both anticipate a target's location (e.g., by drawing on spatial memory or their knowledge of ordered data) and rapidly control the interface mechanics to acquire the item (e.g., by pointing). Linear performance results when the user either has to visually search for the item (e.g., an unknown target location, or a random data order) or when the interface mechanics constrain selection performance (e.g., stepping through a list one item at a time using an arrow key).

Combining prior findings on spatial memory and hierarchical navigation therefore suggests that expert performance can be enhanced by supporting spatially stable items in the shallowest possible hierarchy.

STUDY 1: USERS' SPATIAL KNOWLEDGE OF GUIS

Our overall hypothesis is that spatial memory can be the basis for command-selection interfaces. To test the basic premise of this hypothesis, we carried out a study to see whether experienced users of a real-world application (Microsoft Word 2010) have built up spatial knowledge of familiar commands in the Ribbon interface.

Method

Twelve participants were recruited from a local university; all considered themselves to be experienced Word 2010 users (7 male, 5 female, mean age 25.1). A study system (Figure 2) running on a Windows 7 PC with a 1600x1200 monitor prompted participants through three tasks.

Task 1: determine familiar commands. Participants were asked to inspect the study system's mock-up of the Word 2010 interface and to indicate which Ribbon-based commands they were familiar with (this was a subjective decision with no strict categories of use). These commands were then used in the remaining tasks.

Task 2: specify locations with Ribbon hidden. For each command determined in Task 1, the participant was shown the name and icon of the command, and asked to click on the location of the command with the Ribbon interface hidden. The participant then clicked on a blank space where they thought the Ribbon item would be (see Figure 2). The study system recorded these locations to determine the error in people's spatial memory of the command's location.

Task 3: select commands using the Ribbon. After specifying a location in Task 2, the participant was asked to find that command with the Ribbon interface. Participants clicked on a Ribbon tab to show that tab, and then on the command to complete the task. The system recorded the number of tab switches and clicks used to correctly complete the task.

Participants completed Task 1, then interleaved Tasks 2 and 3 for each of their selected commands. Commands were presented in a random order, and each command was shown twice overall.

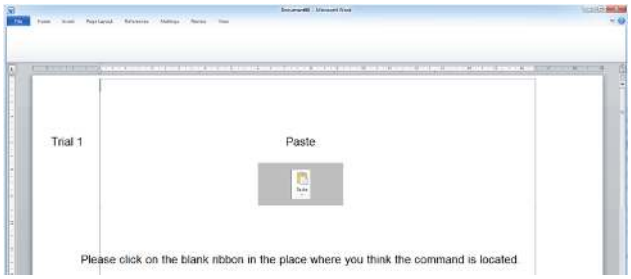


Figure 2. Study interface for Study 1.

Results

Number of familiar commands. Overall, participants chose a mean of 59.6 commands as "familiar" (median 62, standard error 6.72). Many participants appeared to select all of the commands that they had previously used in the interface, rather than just those they used frequently, so we expected a range of actual familiarity with the commands.

Error distance with blank Ribbon. Participants' clicks on the blank Ribbon were on average 147 pixels from the centre of the correct command. There were several outliers, however (see Figure 3), suggesting that some commands were not as well-known as the participant believed. The median error value (less sensitive to outliers) was 92 pixels, which represents approximately 2.5cm on the study monitor. Figure 3 shows the distribution of error distances.

Number of tab selections. When selecting commands with the (visible) Ribbon, participants most often found the command with a single tab selection (one selection was the minimum since the Ribbon was closed at the start of each trial). However, more than one tab selection was needed in 28% of trials; the overall average was 1.95 selections to find the correct command.

These results provide us with two main findings. First, for many commands, people do have a good spatial memory of the commands' locations in the GUI: 50% of commands (i.e., about 30 commands) were known to within 100 pixels. Second, people know the tab location of most of their familiar commands, but for a sizeable subset (28%), they needed more than one selection to find the command.

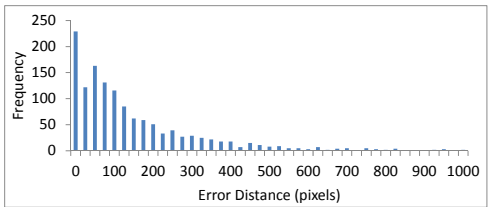


Figure 3. Histogram of error distances. Bins are 25 pixels.

COMMANDMAP DESIGN AND PERFORMANCE MODEL

CommandMap Overview

CommandMap interfaces (e.g., Figure 1) are intended to replace traditional command interfaces such as menus, Ribbons and toolbars. They provide multiple stacked Ribbons that are concurrently displayed when the user presses a dedicated mouse button or command key (e.g. CTRL). Command selections are then made by clicking on the appropriate icon in the CommandMap.

When activated, CommandMaps rapidly fade in to a configurable opacity level (allowing the underlying workspace to be viewed). They remain displayed until their activation key is released, allowing multiple commands to be issued in sequence without reposting.

CommandMap Objectives

Compatible with traditional interaction

Traditional WIMP interfaces have dominated desktop interaction for thirty years. Although faster command invocation mechanisms (such as shortcuts) are available for experts, it is known that these facilities are lightly used [6, 23] and that most users are content to 'make do' with mouse driven selections. CommandMaps therefore maintain the familiar 'point and click' style of interaction.

Improve performance for knowledgeable users

The primary objective for CommandMaps is to improve performance for knowledgeable users. Many office workers use the same computing tools for years or decades, and they are therefore likely to be knowledgeable much longer than they are novice. CommandMaps use two methods to improve knowledgeable user performance: *spatial stability* and *hierarchy flattening.*

Spatial stability. As discussed in Related Work there is extensive empirical evidence showing that consistent spatial placement facilitates location learning and improves selection performance by supporting rapid spatial decisions.

Hierarchy flattening. Traditional interfaces display only a small subset of commands at a time, so command hierarchies are used to partition command subgroups. The result is that even when users know the ultimate location of their targets (as shown by Study 1), they need to mechanically navigate the command hierarchy to satisfy interface requirements. Furthermore, each hierarchical level constitutes an interaction mode, introducing the risk of mode errors – e.g., "Zoom" is not displayed at its known location if the "Home" tab is selected. Scarr *et al*. [30] observed that interface expertise is best supported when interfaces provide a flat command structure. CommandMaps provide a graphical means for hierarchy flattening, maximising the proportion of commands immediately available and reducing the risk of mode errors.

Maintain performance by novice users

While CommandMaps are primarily intended to improve performance by knowledgeable users, it is important that they do not harm novice performance.

Maximise workspace display

When using a desktop application, the user's attention is likely to be on the workspace, such as their document or spreadsheet. Commands must be available on demand, but for much of the time they produce visual clutter and consume space that might be better reserved for the workspace. CommandMaps maximise the workspace by using a modal separation of workspace and commands.

Performance Models: CommandMaps, Menus, Ribbons

To formalise our analysis of the relative merits of CommandMaps, Ribbons, and menus we used the Search, Decision, and Pointing (SDP) model [1, 9] to make theoretical performance predictions. SDP was specifically designed to model performance with menu systems across hierarchical structures and levels of expertise. Our use of SDP also accounts for the proportion of selections requiring the previously selected parent item to be changed.

The SDP model [1, 9] calculates the time to select an item as the sum of time taken at each hierarchical level. The key component of the model is the time taken at each level, which is calculated as the "search/decision time" plus the pointing time (from Fitts' Law). Search/decision time depends on whether the user can decide about an item's location or must visually search for it, with experts being able to make spatial decisions, while novices must rely on visual search. Decision time uses the Hick-Hyman Law of choice reaction time [19, 20], which is a logarithmic function of the number of equally probable choices. Visual search time is a linear function of the number of candidates. The transition from novice visual search to expert decision is modelled using a power law of practice [26]. The reader should refer to Ahlström et al. [1] for a more detailed explanation of the SDP modelling process.

Model assumptions and theoretical performance issues

Using the model to compare CommandMaps, menus, and Ribbons exposes several important theoretical issues about their use. In particular the modelling process demonstrates that knowledgeable use of CommandMaps involves a single decision and pointing activity, while menu use involves two (one for selecting the right menu, and another for selecting the item). Ribbon use is more involved, depending on whether the Ribbon is minimized or not and on whether the target item is within the current tab (details below).

To simplify modelling we make a series of assumptions. We model 210 commands that are evenly divided across seven groupings (approximately reflective of Microsoft Word), with all commands being equally probable. We assume that

command selections begin with the cursor located at the centre of the workspace, that tab/menu targets are 20 pixels wide, and that Ribbon items are 40 pixels wide. We also assume error-free performance. Predictions are calculated in a simple spreadsheet using previously published calibration parameters [9]. The spreadsheet is accessible at *removed for anonymity*.

CommandMap. We model novice selections as requiring a two level search process: first searching for the appropriate tab marker in the CommandMap, then searching for the desired command within that group. While two levels of searching are required, only a single pointing activity is necessary in the flat display. Experts are modelled using a single-level decision between all commands, followed by a single pointing activity. The mean pointing amplitude with CommandMaps is assumed to be 250 pixels.

Figure 4 shows expert performance predictions with the three interfaces as the proportion of selections involving a switch between parent items increases. CommandMaps are predicted to have constant fast performance of approximately 1.5s. Their speed is due to the single decision/pointing activity regardless of the need to switch from the previously selected parent.

Menu. All selections, regardless of expertise, involve a two level acquisition process. Users first search for (novice) or decide about (expert) the menu and point to it. They then search/decide and point to the item in the menu. We assume mean amplitude of 500 pixels from the screen centre to the top level menu, and amplitude of 300 pixels for second level selections (half way through a 30 item menu).

Figure 4 shows a constant expert menu prediction of approximately 3s. This slow performance is due to the two decisions and pointing actions for every selection.

Ribbon. The Ribbon can be minimised, causing it to disappear after each selection, which requires a tab to be clicked before it reappears. In this case Ribbon interaction (and model) is nearly identical to menus, involving a two-level search/decision and pointing process.

Modelling performance with the non-minimised Ribbon is theoretically interesting because it is *sometimes* necessary to switch the parent tab and sometimes unnecessary. For novices we use a two level searching process (as for CommandMaps and menus); however, time for first level pointing is only included when a tab-switch is necessary.

For experts, it is unclear whether acquisitions involve a single decision for a 'global' target (e.g., the user thinks "Bold" and recalls its spatial location) or two decisions (e.g., the user thinks "Home tab", "Bold"). If two decisions are involved, then selections within the currently selected tab involve a superfluous decision, wasting a small amount of time. However, if only a single decision is made then users are likely to encounter mode errors when tab changes are required – for example, the user thinks "Bold", recalls its location from memory, and encounters a mode error when the target is not where expected because the 'View' tab is selected. Anecdotal reports suggest that Ribbon users do make frequent mode errors, lending support to the one-level decision model.

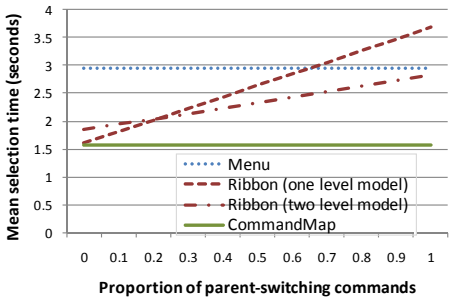


Figure 4. Predicted expert performance across proportion of commands requiring a tab change.

Figure 4 shows expert predictions for both one- and two-level Ribbon models (using the same pointing distances as menus). Ribbons are predicted to match CommandMaps only when no selections involve switching parents, and to gradually deteriorate as the proportion of parent switching increases. Note that the one-level model predicts that Ribbons will be worse than menus when most selections involve a tab switch.

STUDY 2: KNOWLEDGEABLE USE OF COMMANDMAPS

Studies 2 and 3 compare user performance with CommandMaps, Ribbons, and menus when knowledgeable and when novice. Study 4 then compares performance with two variant CommandMap designs for allowing window geometry manipulation. All participants completed Studies 2-4 in a single one hour session.

The primary aim of CommandMaps is to improve performance by knowledgeable users who have developed spatial awareness of command locations. Study 2 therefore tests the following hypotheses:

H1: Knowledgeable users can select commands faster using CommandMaps than when using Ribbons and menus.

H2: There is no performance difference between CommandMaps and Ribbons when selecting commands contained in the most recently used tab, but CommandMaps are faster than the Ribbon for tasks requiring switching between different parent tabs.

H3: Subjectively, users will prefer CommandMaps.

Hypotheses 1 and 3 are important but straightforward performance and preference comparisons. Hypothesis 2 is more nuanced, examining the theoretical performance model's assumptions. As the one-level model of Figure 4 shows, we predict no difference between CommandMaps and Ribbons for *non-switching* tasks. However, the model also predicts that CommandMaps will perform much better than Ribbons and menus when switching is required.

Procedure

To achieve the interface familiarity necessary to examine knowledgeable user performance, we based the experiment on a widely used desktop application: Microsoft Word 2007. All participants completed tasks using three interfaces: a Ribbon replicating the actual Word Ribbon, a menu, and a CommandMap. The menu design used seven top-level menus matching the Ribbon's tabs, with underlying menus containing all of the items in each tab, and similar group separation. The CommandMap, shown in Figure 1, presented all of the Ribbon tabs laid out from top to bottom within the window. None of the interfaces implemented third level pop-up/drop-down items – for example, clicking on the colour swatch drop-down arrow  did not post the associated dialog.

As participants may not have encountered the Word commands used in the experiment, and because no participant could have had prior experience with our tailor-made menu or CommandMap interfaces, they were required to complete two blocks of tasks with each interface: *familiarisation* and *performance*. The *familiarisation* block was used to assure familiarity with the location of commands in each interface condition, while the *performance* block was used for experimental analysis.

Tasks were initiated by clicking a 'Next' button in the centre of the window, which displayed a sidebar prompt containing the name and icon for a target. Task timing began when the prompt was displayed and ran until the correct item was selected. Incorrect selections produced an audible beep. Participants were instructed to complete tasks "as quickly and accurately as possible".

Three sets of command targets were generated, with each set consisting of a total of six commands located in three different tabs: three in the Home tab, two in the Insert tab, and one in the View tab. Each participant used the same command set for *familiarisation* and *performance* with one interface, and then different command sets for subsequent interfaces. The order of command set and interface was counterbalanced using a Latin square.

The *familiarisation* block comprised 30 trials, with 5 selections for each of the six targets. The *performance* block contained 90 trials, with 15 selections for each of the same six targets. The order of target selection within each condition was established with a one-off random process, where the selection sequence was repeatedly regenerated until it met our constraint that 50% of selections would involve a tab switch when using the Ribbon.

Participants completed NASA-TLX [18] worksheets after each interface, and at the end of the experiment they ranked the three interfaces for preference.

Participants and Apparatus (for studies 2-4)

18 participants were recruited from a local university (16 male, 2 female). The experiment was performed on a Windows 7 desktop with a 2.66 GHz Intel Core 2 Quad and 8GB of RAM. A 22″ screen was used, running at a resolution of 1680×1050.

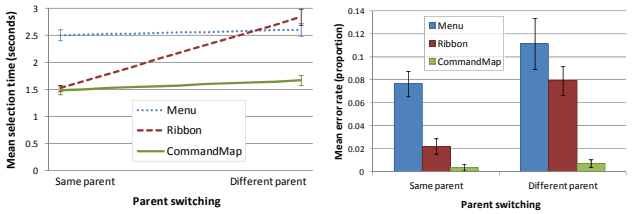
Design

The experiment is designed as a 3×2 analysis of variance for within-subjects factors *interface {ribbon, menu, commandmap}* and *parent {same, different}*. The factor *parent* allows analysis of the impact of moving between different interface structures – tasks are *same* when the current selection occurs in the same menu or Ribbon tab as the last one; otherwise they are *different*. The dependent measures are task time and error rate.

Results

We analysed task time data with and without trials containing incorrect selections, with both analyses producing the same statistical outcomes.

Mean acquisition times (errors removed) were fastest with *commandmap* (1.57 s, s.d. 0.4), followed by *ribbon* (2.11 s, s.d. 0.8) and *menu* (2.40 s, s.d. 0.4), giving a significant main effect of *interface*: $F_{2, 34} = 114.0$, $p < .001$. Bonferroni corrected pairwise comparisons (total $\alpha = .05$) confirm that *commandmaps* were faster than *ribbon* (by 25%) and *menu* (by 34%). We therefore find support for **H1**.



(a) Task time **(b) Errors**

Figure 5. Results for Study 2, with (a) shown as a line chart for consistency with Figure 4. Error bars show standard error.

As expected, there was a significant effect of *parent* ($F_{1, 17} = 155.5$, $p < .001$), with *same* selections faster than *different*. Importantly, though, there was a strong *interface* × *parent* interaction ($F_{2, 34} = 187.4$, $p < .001$). This is shown in Figure 5a: *commandmaps* and *ribbon* performed similarly for *same* tasks, but *commandmap* was relatively faster in *different* tasks (the lines in the figure show linear interpolation between data for *same* and *different* tasks with each interface). We therefore find support for **H2**. The model predictions shown in Figure 4 are confirmed by Figure 5a, including the crossover effect of *ribbon* performance becoming worse than menus in *different* tasks.

The proportion of trials containing an error was much lower with *commandmaps* (0.6%) than either *ribbon* (5%) or *menu* (9%): $F_{2, 34} = 21.6$, $p < .001$. A significant *interface* × *parent* interaction ($F_{2, 34} = 5.26$, $p < .05$), evident in Figure 5b, is caused by *commandmap* error rates being relatively unaffected by *parent*, while *ribbon* and *menu* have much higher errors in *different* parent tasks (suggestive of the hypothesised mode errors).

The combination of time and error data is important, as it shows that *commandmaps* do not increase errors to achieve their improved temporal performance – they are both faster and more accurate than menus and Ribbons.

User response to CommandMaps was positive, with 14 participants ranking it as their first preferred interface, two rating *ribbons* first, and two *menus*: $\chi^2=16.0,\ p < .001$. CommandMaps were also rated as having the lowest workload on all significant NASA-TLX measures (Table 1). We therefore find support for **H3**.

	Menu		Ribbon		CM		χ_r^2	Sig
Mental demand	3.1	(1.1)	3.4	(0.9)	2.5	(1.2)	11.9	< .005
Physical demand	3.7	(1.1)	3.5	(0.9)	2.4	(1.0)	11.6	< .005
Temporal demand	2.9	(1.1)	3.2	(0.9)	2.4	(1.2)	9.3	< .01
Hard work	3.1	(0.9)	3.1	(1.0)	2.0	(1.1)	10.5	< .01
Frustration	3.3	(1.0)	2.9	(1.0)	1.9	(1.1)	13.4	< .005

Table 1. Mean (st. dev.) NASA-TLX values (1= low, 5=high).

STUDY 3: NOVICE USE OF COMMANDMAPS

CommandMaps are primarily intended to enhance knowledgeable users' performance, but novice performance is also important. Study 3 therefore compares novice performance with CommandMaps, Ribbons, and menus. Since CommandMaps display all commands at once, there is a risk that visual search performance will be impaired by the need to visually scan many concurrent candidates.

Procedure

The experiment involved acquiring randomly located targets in logical groupings using CommandMap, menu, and Ribbon interfaces. Five groups of 24 items each were created to populate the interfaces (*animals*, *cartoon characters*, *food*, *office items*, and *sports*). Only items from *animals*, *food*, and *sports* were used as targets. The groups were intentionally unconnected with computing to avoid transfer effects from traditional interface experience.

Tasks were presented to participants using an identical prompting interface to Study 2. Participants completed twenty-four tasks with each interface before proceeding to the next interface (interface order counterbalanced using a Latin square). The tasks with each interface comprised selecting eight unique targets in each of three different groups (e.g., eight different animals). The order of task presentation was manipulated such that half of the tasks involved switching parent group and half did not (to test the impact of searching within and across tabs). To reduce learning effects across tasks (and hence emulate novice visual search) no target item was reused throughout the experiment, and the location of all items (parents and items within groups) was randomised for every trial. Participants provided comments and rated the ease of finding targets at the conclusion of each interface condition, and at the end of the experiment they ranked the three interfaces for perceived performance and preference.

Participants, apparatus, and design are identical to Study 2.

Results

Mean acquisition times were similar with *commandmap* (4.45 s, s.d. 1.73) and *ribbon* (4.38, s.d. 1.4), but slower with *menu* (5.74, s.d. 1.6), giving a significant main effect of *interface* ($F_{2,34} = 110.9,\ p < .001$). In pairwise posthoc comparisons (Bonferroni adjusted T-Tests), menus were slower than both *ribbon* and *commandmap,* but there was no difference between *commandmap* and *ribbon* ($T_{17}<1$).

There was a significant *interface*×*parent* interaction ($F_{2,34} = 12.3,\ p < .001$; Figure 6), with *ribbon* slightly faster than *commandmap* for *same* tasks, but *commandmap* slightly faster than *ribbon* for *different* tasks. Pairwise comparisons between *commandmap* and *ribbon* in each of these conditions (*same* and *different*) show no significant difference ($p > .05$).

Error analysis showed a 2.8% error rate with *commandmap,* 5.1% with *ribbon,* and 16% with *menu*: $F_{2,34} = 35.2,\ p < .001$. There were marginally more errors with *different* parent (9.2%) than with *same* (6.6%): $F_{1,17} = 4.1,\ p = .06$. There was no *interface*×*parent* interaction ($F_{1,17} < 1$).

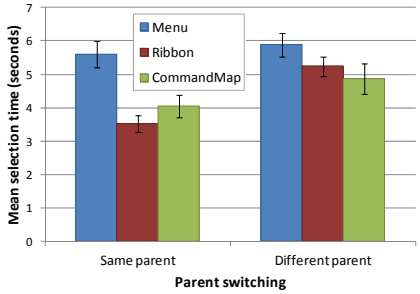


Figure 6. Mean selection times in Study 3. Error bars show standard error.

Subjective responses to the question "It was easy to find targets" (1 disagree, 5 agree) indicated greatest ease with *commandmap* (mean 3.5, s.d., 1.0), followed by *ribbon* (3.2, 1.0) and *menu* (2.4, 0.9): Friedman $\chi^2=10.0,\ p < .005$. Eleven participants ranked *commandmap* as their preferred interface for the task, four preferred the *ribbon*, and two preferred *menus*: $\chi^2=7.9,\ p < .05$. Comments on the *commandmap* presentation were mixed, with one participant stating *"Too much to see at once"*, and another saying *"I like how you can see all the buttons at once."*

The key finding is that novice performance is similar when using CommandMap and Ribbon designs; both are substantially better than menus.

STUDY 4: COMMANDMAPS AND WINDOW GEOMETRY

Studies 2 and 3 used large, static windows, but any practical deployment will need to accommodate variable window sizes and positions. This raises questions of how CommandMaps should respond to window geometry manipulation, and how this affects their performance. The following sections describe and test two CommandMap designs for responding to window geometry manipulation – one based on scaling within the window boundary, and another using a pop-up window.

Scaling and Pointing Lens CommandMap

Scaling CommandMaps are dynamically resized in response to window size manipulations so that items maintain relative spatial location. To avoid distortion when windows are resized on only one dimension, they maintain a 1:1 aspect ratio using the smaller window dimension. They are anchored to the top-left corner of the window. To assure that targets remain discernable at small scales a pointing lens is used to magnify the area under the cursor.

Pop-up CommandMaps

Pop-up CommandMaps are displayed in a pop-up window of constant (full) size. Like menus, the location of the CommandMap is anchored in the top-left window corner by default, but it is repositioned outside the window boundary when necessary for the entire CommandMap to appear within the display. Therefore, when the window is small, or when the window intersects a screen edge, the CommandMap extends outside the window boundary.

Evaluating the Designs

We compared knowledgeable user performance with *scaling* and *pop-up* CommandMaps at three different window *sizes*: full size (1280×1024), 50% (640×512), and 25% (320×256). The 50% size represents a realistic lower bound for window size with a standard desktop application. The 25% size represents an extreme limit of interaction.

Procedure

Experimental tasks involved selecting the same six targets used for the *commandmap* condition in Study 2. Participants initially performed a block of 'refresher' trials, selecting each of the six targets twice (data discarded). They then made 36 selections with *scaling* and 36 with *pop-up* interface (order counterbalanced). The 36 selections comprised 12 at each *size* (full, 50% and 25%), consisting of two repetitions of each of the six targets. The targets were ordered such that each selection used a different window size to the preceding one (e.g., a participant might select target 1 at full size, then target 2 at 25%, then target 3 at 50%, and so on) in order to maximise abrupt transitions between window sizes. Tasks were presented to users using the same prompting interface as Studies 2 and 3.

Participants, Apparatus, and Design

Participants and apparatus are identical to Studies 2 and 3. The design is a 2×3 RM-ANOVA for within-subjects factors *interface {scaling, popup}* and *size {full, 50, 25}*. The main dependent measure is task time.

Results

The error rate was low (a total of 10 across 1296 trials), so error analysis was not conducted. *Popup* (mean 1.54, s.d. 0.33) was much faster than *scaling* (2.65, 1.1), giving a significant effect of *interface* ($F_{1, 17} = 82.1$, $p < .001$). There was also a significant main effect of *size* ($F_{2, 34} = 81.5$, $p < .001$), but this was due to *scaling* performance deteriorating as *size* decreased, while *popup*'s performance remained stable, leading to a significant *interface* × *size* interaction ($F_{2, 34} = 77.7$, $p < .001$). *Popup* outperformed *scaling* even at *full* size, where the two conditions were identical. This suggests that the abrupt transitions between sizes were a significant detriment to performance with *scaling* – one participant commented *"I found I lost my sense of where things were as the scale changed."* All participants preferred the popup interface.

Popup's performance stability across window size is important. In Study 2, the *commandmap* mean of 1.57 s was 25% faster than *ribbon*, and *popup*'s mean in Study 4 was nearly identical at 1.54s. We did not include *ribbon* in Study 4, but it would clearly have performed worse than it did in Study 2 due to its progressive elision of items into additional hierarchical levels (see Figure 7). The results of Study 4 therefore suggest that the advantage for popup CommandMaps over the Ribbon will exceed 25% with small windows.

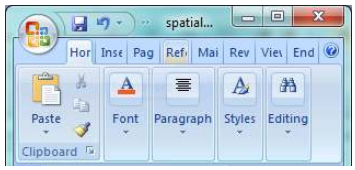


Figure 7. The Word Ribbon at 320 px width, necessitating additional hierarchical traversal to reach targets.

DISCUSSION

To summarise the results, Study 1 confirmed that users have a good memory for the spatial location of commands, but that their memory for the parent item containing commands is relatively weak. Studies 2 to 4 then tested CommandMaps. Study 2 demonstrated that CommandMaps provide substantial performance benefits for knowledgeable users – they were 34% faster than menus and 25% faster than the Ribbon. The results confirmed the predictive performance model, including a cross-over effect with Ribbon performance being worse than menus for selections involving a parent switch. CommandMaps were also much less error prone, with 0.6% errors compared to 5% and 9% with Ribbons and menus respectively. Study 3 showed that novice visual search for randomly located items in CommandMaps is faster than menus, but not significantly different to Ribbons. The study also showed that the relative performance of CommandMaps and Ribbons depends on whether selections involve switching from the previous parent item. Study 4 demonstrated that popup CommandMaps remain efficient regardless of window size.

Why did CommandMaps succeed?

The empirical results closely matched the theoretical predictions generated by the performance model (Figures 4 and 5a). Furthermore, the preferred 'one-level model' of Ribbon use anticipated frequent mode errors when parent switches are required, as observed with the Ribbon's 5% error rate (as compared to 0.6% with CommandMaps).

The theoretical model mechanically implements predictions using previously reported parameters [9] (eliminating any chance of calibration 'bias'), and the model's formulae for

expert performance attend only to the number of interface levels, the timing associated with location decisions at each level, and pointing requirements. Therefore, we attribute CommandMaps' success to their two defining properties – stability of item location (allowing spatial decisions), and maximally flattened hierarchy (allowing acquisition with a single decision and pointing action).

CommandMaps in the real world

The experiment focused on command selection performance, with tasks involving repeated selection of a small set of serially presented targets. While real work sometimes involves executing a series of commands (e.g., changing the zoom level, inserting a symbol, and formatting it) it normally interleaves activities on the workspace with command selections. This raises concerns about whether the experimental findings will generalise to real use, discussed below.

Impact of the small target set on spatial memory. Study 3 involved repeated selections of six target items. The small set was used to assure participants had a good spatial knowledge of target location (emulating expertise), but it is possible that the method induced spatial location memory that is artificially refined. We are confident that the results will generalise to larger active command sets for two reasons. First, Study 1 shows that participants have a good spatial knowledge of approximately 30 items (50% of a mean 59.6 "familiar items"). Second, prior studies have demonstrated that users can efficiently draw on spatial memory for large item sets (e.g. [29]).

Activating control. Our experimental interface used the CTRL key to activate the CommandMap, but this requires bimanual operation with one hand on the key and another on the mouse. Our experimental participants issued an intense series of command selections, so it was natural for them to keep one hand on or near the control key. However, during real work the non-dominant hand might be otherwise occupied, demanding a homing action to the activating key. Two solutions to this concern are first, the CommandMap could be posted by clicking in a designated area (e.g., window title); similar to how the current Ribbon can be posted once 'minimized'; second, a dedicated mouse button could be used to activate the CommandMap mode, allowing unimanual selection. Similarly, on a touchscreen device, the CommandMap could be activated with a specific gesture (e.g., four finger touch).

Workspace overlay. To display the full set of commands simultaneously, the CommandMap covers the user's work or content area with a configurable transparent overlay. While this overlay allows the underlying area to remain visible, it is possible users may respond less favourably to having their content somewhat obscured when invoking commands that allow previews prior to final selection (e.g., font size). We hope, however, that the substantial performance benefits of the CommandMap design outweigh this potential downside, which would be present for only a subset of commands.

Initial user reaction. Study 2 shows that novice visual search performance is similar between Ribbons and CommandMaps. However, there are two concerns on initial user reaction. First, three participants indicated that the number of controls was 'overwhelming' when first viewing the CommandMap, but this impression quickly dissipated on use. Second, there is an absence of control affordance due to the omission of obvious controls at their familiar location. Both of these concerns are short-term effects that might be eased with a help display after installation.

Limit of number of commands. While CommandMaps utilise screen real estate to a much higher degree than conventional techniques, there is still a limit to the number of commands that can be displayed at once. In situations where the available command set is too large, a hierarchical structure must still be employed. However, we still anticipate a performance increase over contemporary interfaces if the hierarchy is as shallow as possible. Furthermore, CommandMaps in their current form are unable to support certain features of the Ribbon, such as contextual tabs, due to a lack of screen space. Anyone designing a practical implementation of CommandMaps will therefore have to keep screen size limitations in mind when choosing control arrangements.

CONCLUSIONS

In modern user interfaces, hierarchical command organisations are common. However, we showed that users can remember the spatial locations of controls without the need for hierarchy, implying that hierarchy traversal is inefficient for experienced users.

We presented the notion of combining spatial memory and flat hierarchies to support efficient command access and instantiated these ideas within CommandMaps. We generated performance models supporting our design and empirically validated them through two studies: one demonstrating a speed increase for expert users of 34% over menus and 25% over Microsoft's Ribbon, and the other showing no significant performance difference for novices. Subjective responses indicated that CommandMaps was preferred across both experiments. Finally, we evaluated two alternative designs allowing CommandMaps to remain effective at smaller window geometries, with a "pop-up" design performing significantly better than one that scaled widgets according to the window dimensions.

There are a number of directions for future work. Our experiments used menus and Ribbons as baseline comparators due to their dominance in contemporary interfaces. However, comparisons with other command invocation techniques are needed, particularly with those that have been shown to support expert use, such as marking menus [22]. A second area of future work involves exploring ways to combine CommandMaps with other performance optimizations, particularly for systems that have a predictive capacity. For example, ephemeral adaptation [14] or a related scheme could be used to emphasize likely commands.

Alternatively, a subset of frequently used commands could remain visible in workspace mode (similar to Gajos' Split Interface [16]). Finally, studies with more complex tasks would provide insight into the strengths and limitations of the CommandMap design when command invocation is intermixed with content manipulation.

ACKNOWLEDGEMENTS

We would like to thank our study participants for the use of their valuable time. This work was partially funded by Royal Society of New Zealand Marsden Grant 10-UOC-020.

REFERENCES

1. Ahlström, D., Cockburn, A., Gutwin, C. and Irani, P. Why it's Quick to be Square. in *Proc. CHI'10*, (2010).
2. Andrade, J. and Meudell, P. Short report: Is spatial information encoded automatically? *Quarterly Journal of Experimental Psychology 46A* (1993), 365-375.
3. Baddeley, A.D. *Human Memory*. Erlbaum, (1990).
4. Byrne, M., Anderson, J., Douglass, S. and Matessa, M. Eye Tracking the Visual Search of Click-Down Menus. in *Proc. CHI'99*, ACM, (1999), 402-409.
5. Callahan, J., Hopkins, D., Weiser, M. and Shneiderman, B. An Empirical comparison of Pie Versus Linear Menus. in *Proc. CHI*, (1988), 95-100.
6. Carroll, J. and Rossen, M. Paradox of the active user. in Carroll, J. ed. *Interfacing Thought: Cognitive Aspects of HCI*, MIT Press, 1987, 80-111.
7. Cockburn, A. and Gutwin, C. A Predictive Model of Human Performance with Scrolling and Hierarchical Lists. *HCI 24*, 3 (2009), 273-314.
8. Cockburn, A., Gutwin, C. and Alexander, J. Faster Document Navigation with Space-Filling Thumbnails. in *Proc. CHI'06*, ACM Press, (2006), 1-10.
9. Cockburn, A., Gutwin, C. and Greenberg, S. A Predictive Model of Menu Performance. in *Proc. CHI'07*, ACM Press, (2007), 627-636.
10. Czerwinski, M., van Dantzich, M., Robertson, G. and Hoffman, H. The Contribution of Thumbnail Image, Mouse-Over Text and Spatial Location Memory to Web Page Retrieval. in *Proc. INTERACT*, (1999), 163-170.
11. Darken, R.P. and Sibert, J.L. Wayfinding strategies and behaviors in large virtual worlds. in *Proc. CHI '96*, ACM, (1996), 142-149.
12. Ehret, B. Learning Where to Look: Location Learning in Graphical User Interfaces. in *Proc. CHI'02*, 211-218.
13. Findlater, L. and McGrenere, J. A comparison of static, adaptive, and adaptable menus. in *Proc. CHI'04*, ACM, (2004), 89-96.
14. Findlater, L., Moffatt, K., McGrenere, J. and Dawson, J. Ephemeral adaptation. in *Proc. CHI'09*, ACM Press, (2009), 1655-1664.
15. Fitts, P.M. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *J. Experimental Psych. 47* (1954), 381-391.
16. Gajos, K., Czerwinski, M., Tan, D. and Weld, D. Exploring the Design Space for Adaptive Graphical User Interfaces. in *Proc. AVI'06*, (2006), 201-208.
17. Gutwin, C. and Cockburn, A. Improving List Revistation with ListMaps. in *Proc. AVI'06*, 396-403.
18. Hart, S. and Staveland, L. Development of NASA-TLX. in *Human Mental Workload*, 1988, 139-183.
19. Hick, W.E. On the rate of gain of information. *Quarterly Journal of Experimental Psychology 4* (1952), 11-26.
20. Hyman, R. Stimulus information as a determinant of reaction time. *Experimental Psych. 45* (1953), 188-196.
21. Jones, W. and Dumais, S. The Spatial Metaphor for User Interfaces: Experimental Tests of Reference by Location versus Name. *ACM TOIS 4*, 1 (1986), 42-63.
22. Kurtenbach, G. and Buxton, W. User Learning and Performance with Marking Menus. in *Proc. CHI'94*, (1994), 258-264.
23. Lane, D.M., Napier, H.A., Peres, S.C. and Sandor, A. Hidden costs of graphical user interfaces. *I.J. HCI 18*, 2 (2005), 133-144.
24. Miller, D. The depth/breadth tradeoff in hierarchical computer menus. in *Proc. HFES*, (1981), 296-300.
25. Mitchell, J. and Shneiderman, B. Dynamic versus Static Menus. *ACM SIGCHI Bulletin 20*, 4 (1989), 33--36.
26. Newell, A. and Rosenbloom, P.S. Mechanisms of Skill Acquisition and the Law of Practice. in Anderson, J. ed. *Cog. Skills & Acquisition*, Erlbaum, 1981, 1-55.
27. Odell, D., L. , Davis, R., C., Smith, A. and Wright, P., K. Toolglasses, marking menus, and hotkeys: a comparison of one and two-handed command selection techniques. in *Proc. Graphics Interface*, (2004), 17-24.
28. Postma, A. and De Haan, E. What Was Where? Memory for Object Locations. *Quarterly Journal of Experimental Psychology 49A*, 1 (1996), 178-199.
29. Robertson, G., Czerwinski, M., Larson, K., Robbins, D., Thiel, D. and van Dantzich, M. Data Mountain. in *Proc. UIST'98*, (1998), 153-162.
30. Scarr, J., Cockburn, A., Gutwin, C. and Quinn, P. Dips and Ceilings: Understanding and Supporting Transitions to Expertise in User Interfaces. in *Proc. CHI'11*, ACM, (2011), 2741-2750.
31. Silverman, I. and Eals, M. Sex differences in spatial abilities: Evolutionary theory and data. in *The Adapted Mind*, Oxford University Press, (1992).
32. Tak, S., Cockburn, A., Humm, K., Ahlstroem, D., Gutwin, C. and Scarr, J. Improving Window Switching Interfaces. in *Proc. INTERACT'09*, (2009), 187-200.
33. Thorndyke, P.W. and Goldin, S.E. Spatial learning and reasoning skill. *Spatial orientation: Theory, research, and application* (1983), 195-217.