Promoting Hotkey Use through Rehearsal with ExposeHK

Sylvain Malacria, Gilles Bailly, Joel Harrison, Andy Cockburn, Carl Gutwin

Promoting Hotkey Use through Rehearsal with ExposeHK

Sylvain Malacria[1] **Gilles Bailly**[2] **Joel Harrison**[1] **Andy Cockburn**[1] **Carl Gutwin**[3]
[1]University of Canterbury, Christchurch, New-Zealand
[2]Quality and Usability Lab, Telekom Innovation Laboratories, TU Berlin, Berlin, Germany
[3]Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, Canada
sylvain@malacria.fr, andy@cosc.canterbury.ac.nz, gillesbailly1@gmail.com, gutwin@cs.usask.ca

Figure 1. The user wants to open a new tab but is not sure of the hotkey. He visually locates the button in the toolbar (boxed on left), then presses the Command key (⌘) to activate ExposeHK, which overlays toolbar items with available hotkeys (right). He completes the command by pressing ⌘ T.

ABSTRACT
Keyboard shortcuts allow fast interaction, but they are known to be infrequently used, with most users relying heavily on traditional pointer-based selection for most commands. We describe the goals, design, and evaluation of ExposeHK, a new interface mechanism that aims to increase hotkey use. ExposeHK's four key design goals are: 1) enable users to browse hotkeys; 2) allow non-expert users to issue hotkey commands as a physical rehearsal of expert performance; 3) exploit spatial memory to assist non-expert users in identifying hotkeys; and 4) maximise expert performance by using consistent shortcuts in a flat command hierarchy. ExposeHK supports these objectives by displaying hotkeys overlaid on their associated commands when a modifier key is pressed. We evaluated ExposeHK in three empirical studies using toolbars, menus, and a tabbed 'ribbon' toolbar. Results show that participants used more hotkeys, and used them more often, with ExposeHK than with other techniques; they were faster with ExposeHK than with either pointing or other hotkey methods; and they strongly preferred ExposeHK. Our research shows that ExposeHK can substantially improve the user's transition from a 'beginner mode' of interaction to a higher level of expertise.

Author Keywords
Hotkeys; Keyboard Shortcuts; Rehearsal; Menus; Command Selection; Novice Mode; Expert Mode.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation: User Interfaces. – Graphical user interfaces (GUI).

INTRODUCTION
Hotkeys, also called keyboard shortcuts or accelerators, offer a shortcut alternative to pointer-based selection of commands from toolbars and menus. Their efficiency stems from three mechanical advantages over pointing: first, in many tasks such as word-processing, the hands rest on the keyboard, so hotkeys eliminate the need to move the hand to a pointing device and back; second, they eliminate the need for a pointing round-trip from the workspace to the control widgets and back; and third, they allow a wide range of commands to be selected with a single key combination, thus removing the need to traverse a menu or tab hierarchy. A variety of theoretical models (e.g., KLM [7]) and empirical studies (e.g, Odell et al. [24]) demonstrate that hotkeys improve user performance.

Despite this potential, hotkeys are under-used: several studies have demonstrated that few users employ any form of shortcut interface [1, 5, 9, 19, 20]. Carroll's 'paradox of the active user' [9] suggests that users are simply too engaged in their tasks to consider learning alternative strategies or methods, even if these methods may eventually improve performance. In addition, keyboard shortcuts require that users learn hotkeys beforehand, potentially resulting in errors due to incorrect hotkey/command memory associations.

The poor adoption of hotkeys and other high-performance interface techniques creates a substantial usability problem. While the performance difference between hotkeys and pointer-based commands may be small for some actions, it can be large when command activation involves hierarchical navigation (such as a menu-cascade) or when selecting widgets located far from the workspace. These small performance differences can compound into substantial effects when multiplied across

Authors' version.
CHI 2013, April 27–May 2, 2013, Paris, France.

thousands of daily repetitions. It is therefore important to find ways of helping users transition from the pointer to hotkeys.

Two recent studies show the state of the art in encouraging hotkey use. Grossman et al. [12] and Krisler and Alterman [17] investigated the use of feedback (emphasising the hotkey after pointer selection) and cost (imposing additional steps to complete pointer selections) as mechanisms for encouraging adoption and use of hotkeys, with positive results.

However, these approaches have three substantial limitations. First, they use pointer-based selection as the starting point for hotkey presentation (i.e., the hotkey is only shown after mouse-based selection). Consequently, users reinforce pointing even while trying to learn a faster non-pointer method. Second, cost-based techniques work by penalising pointer selections, rather than by making hotkeys more attractive. For example, one of Grossman's successful techniques imposed a delay after pointer-based selection: the hotkey 'incentive' was turning users away from the old technique, rather than actually improving performance. Third, users are unable to exploit the spatial-location knowledge developed through prior pointer-based selection. Users must switch entirely to the new approach, which usually implies a temporary but substantial reduction in performance (a 'performance dip' [31]) that is likely to deter hotkey use.

We have developed and evaluated a new method for encouraging and improving hotkey use – called ExposeHK (EHK) – that addresses these issues. When activated with a modifier key (e.g., the Control key), EHK displays the hotkeys in the application's toolbar (see Fig.1), menu, or ribbon. Importantly, EHK is compatible with existing toolbar and ribbon designs, and it can be readily adapted to menus.

This work makes three main contributions. First, it describes design principles for promoting hotkey use. Second, it presents ExposeHK (EHK) – a system instantiating the principles with toolbar, menu and ribbon designs. Third, it presents empirical results showing that EHK promotes earlier and higher levels of hotkey use, that participants are faster with it, and that they strongly prefer EHK.

EXPOSEHK: DESIGN GOALS AND RELATED WORK

EHK uses a simple interaction mechanism to promote hotkey use. While a modifier key is held down, all hotkeys are concurrently displayed on top of their graphical controls. Selections are completed using the hotkey or by pointing.

We implemented and evaluated three forms of the technique for toolbars (ExposeHK$_T$, see Fig.1), menus (ExposeHK$_M$, see Fig.6), and ribbons (ExposeHK$_R$, see Fig.9), with details presented later. This section describes the design goals for EHK, as well as key associated prior work.

Design Goals

The ultimate objective of EHK is to improve the rate at which users attain expertise with interfaces, by promoting hotkey use. Figure 2, adapted from [31], illustrates this idealised objective. It shows that switching to traditional shortcuts involves a temporary 'performance dip' that discourages their use. The dip occurs because users who are competent with pointer-based selection must pause, display the hotkey, learn it, and rehearse the key sequence. With EHK we aim instead to support a smooth and continual transition to expert performance. We have four goals to help EHK achieve this objective, as follows.

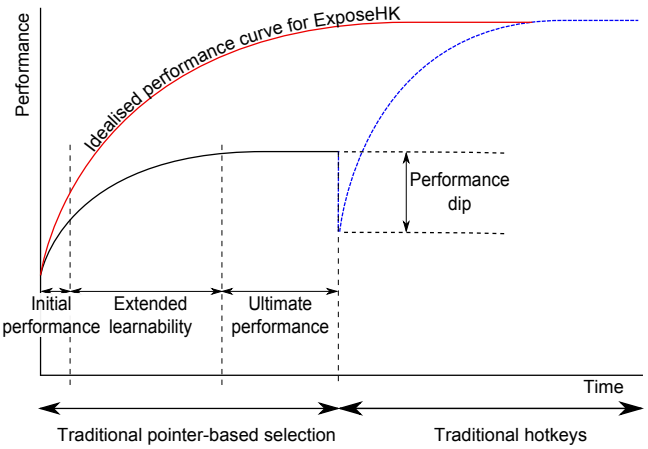

Figure 2. Intended learning curve for ExposeHK, compared with modality switching (adapted from [31]).

Goal 1: Enable hotkey browsing

In most contemporary interfaces hotkeys are not displayed until the user posts a menu item or dwells on a toolbar item (exceptions, such as Alt keys, are discussed with goal 4 below). Consequently, learning hotkeys involves moving the cursor to the point where a simple click would complete the selection – but then pausing, awaiting the shortcut feedback, and committing it to memory before proceeding (possibly by clicking rather than by using the shortcut). These actions can induce a performance dip (Fig. 2) that discourages hotkey use and traps the user in pointer-based 'beginner mode' [31]. Any memory errors while learning hotkeys will further inhibit performance.

To avoid this trap, systems should allow users to browse hotkeys *without requiring* a pointing action. ExposeHK meets this goal by showing the hotkeys when a modifier key is pressed, as shown in Figures 1 and 6. Users can therefore initiate their command actions using the *same modality* that they will ultimately use as experts. Novice users will still need to visually search for the target interface control, but once identified, they can press the EHK modifier key to display the hotkeys underlying the control, avoiding the need to move the hand to the mouse and point to the target. Furthermore, EHK does not require that users learn hotkeys beforehand, avoiding errors due to incorrect hotkey/command memory associations. By providing a single modality that removes the need to pre-learn the hotkeys we aim to minimise the magnitude and deterrent effect of the performance dip (Fig. 2).

Goal 2: Support physical rehearsal

In analysing factors influencing the development of expertise, Kurtenbach [19] proposed the principle of *rehearsal*: *'guidance should be a physical rehearsal of the way an expert would issue a command'*. He deployed this principle in marking menus, which allow novices to select items by moving the cursor into one segment of a visually displayed 'pie menu' [6]

(a cursor-centred circular arrangement of menu items). Experts select items using exactly the same physical action (a directional gesture), but without waiting for feedback.

The potential value of using the same physical action for novice and expert selections is supported by psychology literature on automaticity and the power law of practice (see [32] for a comprehensive review of motor control and learning). *Automaticity* [33] represents a high level of attainment in motor learning, allowing controls to be manipulated without substantial conscious deliberation. It is characterised by fast and parallel processing. The key determinant in the development of automatic performance is extreme repetition, although repetition alone is not sufficient for some tasks (e.g., Card et al. [8] studied one user repeating a text editing task 1100 times, without observing automaticity). These effects support Kurtenbach's principle of rehearsal, yet few shortcut interfaces (other than marking menus and their variants) exploit the principle. Traditional hotkey methods require users to discover hotkeys using a non-hotkey modality (pointing), and consequently users rehearse pointing, not hotkey use. EHK, in contrast, allows users to discover and rehearse hotkeys using only the hotkey modality. Furthermore, EHK is compatible with traditional interfaces, whereas systems like marking menus require interface changes (i.e., non-traditional circular menu layouts).

Goal 3: Rapid hotkey identification for intermediate users

When users have an intermediate level of skill, they are likely to have partial knowledge of their interaction environment. For example, users may know the spatial vicinity of controls, even if not their exact location; or they may suspect that a particular hotkey sequence triggers a needed command, but be unwilling to execute it without confirmation. EHK leverages human spatial memory by ensuring that hotkeys are displayed at the spatial location of the underlying visual control, which reduces visual search time and allows rapid confirmation of hotkey bindings.

Goal 4: Stable commands in a flat hierarchy

A key reason for hotkeys' efficiency is that most hotkey interfaces use a single key combination to select a command, instead of a series of several point-and-click activities to traverse the command hierarchy. For example, the hotkey CTRL-B may replace two pointing actions for the 'Font' menu and 'Bold' item. Importantly, not all hotkey methods support hierarchy flattening. For example, Alt-Key navigation (part of the accessibility interface in Microsoft Office) displays hotkeys associated with successive levels of the interface hierarchy when the 'Alt' key is pressed, allowing users to traverse the interface. Theoretical and empirical results tend to show that selection time increases with the number of menu levels for experts [10, 31]. In addition, it has been shown that it is difficult to chunk a multiple-keys sequence into one single cognitive unit when using Alt-Key navigation [23].

ExposeHK's hotkey bindings are also stable across invocations (within an application), which allow users to learn and rapidly reproduce key sequences. While this may seem an obvious feature of hotkey interfaces, there are interesting accessibility interfaces that support some of EHK's characteristics, but violate this goal. In particular, the 'show numbers' [11] interface in Windows 7 visually overlays each widget with a number that fades in and out (similar to EHK). Speech input then allows any widget to be selected by reading the number. While this is a powerful accessibility tool, it violates goal 4 because the activation numbers change across invocations.

Other related work

There is extensive literature on understanding and supporting expert performance with user interfaces. This includes work on user strategies (e.g., [5]), interface techniques that optimise expert performance (e.g., Flower Menus [3]), techniques supporting transitions from novice to expert performance (e.g., Blur [31], Marking Menus [18, 19], and Octopocus [4]), understanding collaborative skill sharing [25], and supporting community expertise [22].

Key prior work specifically on hotkeys includes empirical performance comparisons between shortcut techniques including hotkeys (e.g., [16, 20, 24, 15, 2]) and specialised hardware promoting hotkey use (e.g., a touch-sensitive keyboard that highlights on-screen widgets before a key is pressed [28]). Finally, the two studies most closely related to our objectives concern the use of interaction cost as a disincentive for pointer-based selection [12] and the use of feedback to help users learn hotkey bindings [12, 17]. As summarised in the introduction, we want to promote hotkeys without requiring pre-learning, without imposing an explicit cost on pointer-based techniques, and without the potential distraction of post-action feedback.

USER STUDIES RATIONALE

We conducted three studies to deploy and evaluate ExposeHK with traditional organizations of commands in graphical user interfaces: toolbars, menus, and ribbons. Studies 1 and 2, which respectively used toolbar and menu adaptations of ExposeHK, focus on goals 1 to 3, which concern promoting transitions to hotkey use. The primary measure of studies 1 and 2, therefore, is the proportion of commands completed using hotkeys – these studies answer the question, 'does ExposeHK promote earlier and increased hotkey use compare to other methods?' Study 3 then focuses on goal 4, examining the performance improvements that ExposeHK can provide when applied to the Microsoft Ribbon user interface, in comparison to existing methods.

STUDY 1: TOOLBARS (EXPOSEHK$_{\mathrm{T}}$)

The toolbar version of ExposeHK (ExposeHK$_{\mathrm{T}}$), shown in Figure 1, overlays the hotkey associated with each toolbar item when a modifier key is pressed. Study 1 focuses on hotkey adoption with ExposeHK$_{\mathrm{T}}$ compared with traditional tooltips and audio feedback. Traditional tooltips provide baseline comparison with a commercial standard, and audio feedback was used because Grossman et al. [12] showed it to be an effective method for promoting hotkey use. The audio feedback condition used voice synthesis to read the hotkey binding when the pointer clicked the item. Participants in the study were free to complete selections by pointing to toolbar items or by invoking hotkeys.

The study was designed to answer several questions about ExposeHK$_{\mathrm{T}}$ in comparison to traditional tooltips and audio-supplemented hotkey feedback:

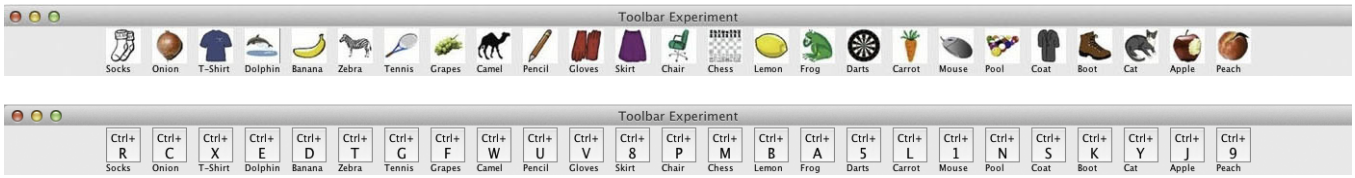

Figure 3. Example of a toolbar used in the study (top), and with the ExposeHK$_T$ mode on (bottom)

- does ExposeHK$_T$ result in earlier and higher levels of hotkey use?
- how does hotkey use with the three interfaces (ExposeHK$_T$, audio, tooltips) change with experience and frequency of item selection?
- does ExposeHK$_T$ support faster command selection?
- are errors affected by interface or selection mechanism?
- how are the systems subjectively perceived?

Experimental method

The method was based on Grossman et al.'s [12] hotkey study with some modifications, as described below.

Task and stimulus. Each task involved selecting a toolbar item in response to a voice-synthesised audible stimulus that read the name of one of the targets – Figure 3 shows a toolbar used in the study. Voice-synthesis was used to ease risks of confounds stemming from factors such as visual pop-out effects due to exact matches between the stimulus (e.g., a cat icon) and its representation in the interface.

The audio stimulus was produced, and task timing began, when the cursor was inside a 70×70 pixel box in the center of the screen and the space bar was pressed. The participant then selected the corresponding item, either by clicking on the target or by selecting the correct hotkey. As in Grossman et al.'s study, timing stopped when the user moved the cursor back to the box and pressed the spacebar, which simulates a pointing round-trip from the workspace to a control and back to the workspace. Completing one trial immediately initiated the next. All user events were logged, including mouse movement and incorrect selections.

Procedure. Participants completed tasks with all three interfaces: *ExposeHK$_T$*, *audio feedback*, and *tooltips*. In the *ExposeHK$_T$* condition, the hotkey binding replaced the corresponding toolbar icon when the control key was pressed. In the *audio feedback* condition, voice-synthesis read the hotkey binding when the user clicked on the item (i.e, after selection). In the *tooltip* condition, a tooltip showed the hotkey immediately when the cursor entered a toolbar item. Immediate display was used (rather than the traditional method of awaiting a dwell timeout) to maximise hotkey exposure and minimise the performance penalty for awaiting the tooltip.

Before using each interface participants received brief training (approximately 30 seconds; data discarded) in which they were prompted to select one item from a toolbar containing four items, first using the pointer, and then selecting the same item using a hotkey. During training, a single sentence was shown describing how to use the hotkeys: either 'Hover over the toolbar item to display the hotkey', 'Voice synthesis will read the name of the hotkey when you select the item', or 'Press the Control key to show the hotkeys'. The potential benefits of using hotkeys were not mentioned to the participants.

The toolbar used in the study consisted of 25 buttons, with a different set of buttons for each interface (e.g., Figure 3). Participants completed six blocks of trials with each interface. Each block consisted of 24 selections, comprising 12 different targets that were selected 8, 4, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1 times each, giving a near-Zipfian distribution (commonly exhibited with command selections, and also used by Grossman et al. [12]). The same 12 targets with the same frequencies were repeated in each block. Targets within this distribution were controlled so that the target appearing 8 times was activated by holding the control key and pressing a hotkey in the left-hand side of the keyboard (Control plus one character among q,w,e,a,s,d,z,x,c), the target appearing 4 times was selected by a key in the right set (u,i,o,p,j,k,l,m), and a target selected twice was in the middle-keyboard (r,t,y,f,g,h,v,b,n). The remaining targets were evenly distributed across the left, middle, and right keyboard sides. Finally, the physical location of targets in the toolbar was controlled to balance the pointing distance for each target of a particular frequency across interface conditions.

Participants were instructed to complete selections as quickly and accurately as possible. To discourage 'racing through' the study irrespective of errors, each erroneous selection incurred a 3 second delay prior to initiating the next task. Once all blocks with an interface were complete, participants completed NASA-TLX worksheets [13] for the technique and gave comments. At the end of the experiment, which lasted approximately 30 minutes, participants ranked their preference for the interfaces and gave final comments.

Design. The experiment used a 3×6×4 analysis of variance for within-subjects factors *technique* (*ExposeHK$_T$*, *audio*, and *tooltips*), *block* (levels 1-6), and *frequency* (whether the item was selected 1, 2, 4, or 8 times in the block). Order of *technique* and the dataset used with the toolbar were counterbalanced across the participants using a Latin Square. The primary dependent measure was the proportion of trials completed using hotkeys. Additional dependent measures included selection time (with error trials removed) and error rate.

Although the within-subjects factor *technique* was counterbalanced, we anticipated the possibility of asymmetric skill transfer, with particular technique orders differently influencing performance in subsequent conditions (see [26]). We therefore analysed factors *order* and *technique*, and found a significant interaction. The results analysis therefore reports

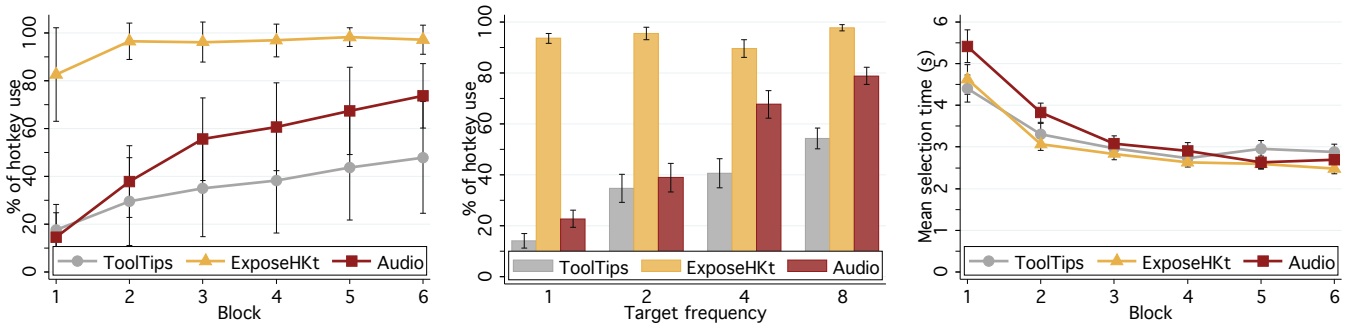

Figure 4. Results of Study 1. Left: % hotkey use by block; Center: % hotkey use by frequency; Right: mean time by block. Error bars $\pm 1 s.e.$

between-subjects treatment of *technique*, discarding data from all but the first technique for each participant. For significant ANOVA effects, we include partial eta-squared (η^2) as a measure of effect size [29].

Participants and Apparatus. 36 university staff and students participated, aged 20-45, 6 female. Experimental software, implemented in Java Swing, ran on Windows XP, using a 1680×1050 22" display, optical mouse, standard QWERTY keyboard, and earphones.

Results

The overall mean time for a correct item selection was 3.37s (σ 1.5), with a low overall error rate of 2.8%. Across all conditions, the mean time to select an item using a hotkey was 2.74s (σ 0.19), which was 35% faster than pointer-based selections (4.16s, σ 0.27). The following paragraphs describe analysis of the dependent measures. Subjective responses for all studies are reported after Study 3.

Proportion of hotkey use. Figure 4 (left) shows the proportion of command selections completed using hotkeys with the three *techniques* across blocks. With ExposeHK$_T$, 94% of correct selections were completed using hotkeys, compared with 50% with *audio* and only *35%* with *tooltips* ($F_{2,33} = 21.7, p < .001, \eta^2 = .57$). As expected, hotkey use progressively increased across *blocks* ($F_{5,165} = 39.6, p < .001, \eta^2 = .54$) and across all *item frequencies* ($F_{3,99} = 31.7, p < .001, \eta^2 = .49$), with infrequent items having lower hotkey selection rates.

Importantly, there was a significant *technique*×*block* interaction ($F_{10,165} = 6.3, p < .001, \eta^2 = .27$). Figure 4 (left) shows that ExposeHK$_T$ users made an early and sustained switch to hotkeys, contrasting with other techniques' gradual increase. This suggests that ExposeHK$_T$ fulfills goal 1 (supporting hotkey browsing by novices), as over 81% of first block selections were completed using hotkeys.

Figure 4 (center) shows a significant *technique*×*frequency* interaction ($F_{6,99} = 9.2, p < .001, \eta^2 = .36$). While hotkeys were consistently employed regardless of item frequency with ExposeHK$_T$, low-frequency items were seldom selected using hotkeys with *tooltip* and *audio* conditions.

Finally, our study replicates Grossman's finding [12] that *audio* feedback outperforms standard tooltips.

Item selection time. Figure 4 (right) shows mean selection times with the different *techniques* across *block*. While there was no significant main effect of *technique* ($F_{2,33} = 1.09, p = .34$), there was a significant *technique*×*block* interaction ($F_{10,165} = 1.99, p < .05, \eta^2 = .10$), stemming from *audio*'s slower selection times initial blocks. As expected, *block* ($F_{5,165} = 82.8, p < .001, \eta^2 = .71$) and *frequency* ($F_{3,99} = 89.7, p < .001, \eta^2 = .73$) showed significant effects. Finally, the mean time for pointer-based selections in the final block (for those who continued to use the mouse with any technique) was 3.53s, compared to 2.38s with hotkeys.

Error rate. *Tooltips*, *audio*, and *ExposeHK$_T$* had similar error rates (for both mouse and hotkey selection) of 2.6%, 3.3% and 2.7% respectively, showing no significant main effect ($F_{2,70} = 1.3, p = .28$). There was no effect of *block* ($p = 0.24$) .

STUDY 2: MENUS (EXPOSEHK$_M$)

Deploying ExposeHK with toolbars is simple because each icon has a unique spatial location, giving an unambiguous position for each hotkey when ExposeHK is activated. However, extending ExposeHK to the large command sets offered by menus raises design challenges because items can share the same display location. For example, Figure 5 (left) shows that the 'Safari' menu extends horizontally across 'File', 'Edit', 'View' and 'History' menus.

Characterising menu hotkeys

To understand how hotkeys are used in existing menus, and the degree of overlapping, we wrote a program using the Mac OS accessibility API to inspect the structure and location of menus in 30 mainstream Mac applications such as Microsoft Word and Adobe Photoshop.

The mean number of top-level menus (excluding the global Apple menu) was 8.97 (σ 1.77), and the mean number of items in one top-level menus was 10.48 (σ 5.9). The total number of hotkeys offered by applications ranged from 15 to 112 (mean 55.31, σ 20.53). Across applications, a mean of 77.8% of hotkeys were located in top-level menus, and 22.1% in second-level menus. The proportion of menu items with associated hotkeys is 47.6% in top-level menus and 15.4% in second-level menus.

We calculated the degree of spatial overlapping between items with hotkeys using heatmap-style data similar to that shown

in Figure 5 (right) for each application. The mean number of spatial collisions (two overlapping items) between hotkey items per application was substantial at 23.93 (σ 10.35), and 44.9 (σ 18.78) between a hotkey item and a normal item.

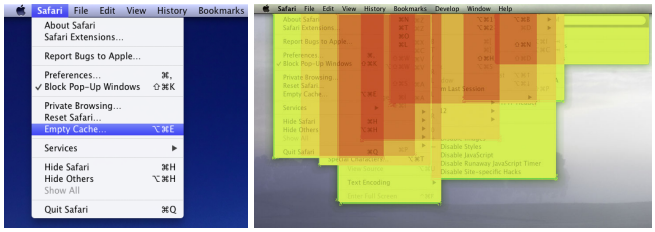

Figure 5. Left: The Safari menu would overlap with File, Edit, View and History menus. Right: heat map of Safari menus overlapping (red/dark means more overlap).

Adapting ExposeHK to menus

The primary design concept of ExposeHK$_M$ is similar to ExposeHK$_T$ – pressing a modifier key exposes all of the menu items with associated hotkeys overlaying items, as shown in Figure 6. Command selections can be completed by pressing the hotkey or by using the pointer. Second-level menu cascades are not displayed, so some degree of pointer-based selection is still necessary.

ExposeHK$_M$ modifies the visual layout of menus to remove spatial overloading. This is necessary to support hotkey browsing (goal 1) and to help intermediate users exploit their spatial memory (goal 3). With ExposeHK$_M$, the width of the gap between top-level menu items is determined by the width of the longest item with a hotkey in each menu. If a longer item without a hotkey exists within the menu, the name is truncated until the cursor hovers over it.

An obvious consequence of this design is that the menubar width increases. Using data from our menu structure analysis, we calculated a mean menubar width of 1204 px for the 30 applications (using the default system font), ranging from 722 px to 1736 px. This raises interesting issues for deployment – ExposeHK$_M$ can be readily adapted to Mac OS X menus because they extend across the width of the primary screen; however, it would be more difficult to deploy in small windows where the menubar resides within the window. We return to issues of deployment in the discussion.

Although ExposeHK$_M$ is intended to promote hotkey use, it is compatible with pointer-based interaction and users can select items using the pointer if preferred. Doing so should be more efficient than traditional menus because it removes the requirement to first point to a menu to post it (pressing the modifier key posts all menus at once), thus replacing two pointing actions with a single one. Although this design risks giving an initial impression of visual clutter, studies suggest that similar methods of parallel presentation can improve pointer-based selection performance [30] and reduce visual search times because rapid eye saccades can replace comparatively slow pointer-based manipulation [14, 27]. Finally, users can still browse the menus and select the items using the pointer only as with a regular menubar if preferred.

Experimental method

Study 2 examined hotkey use with ExposeHK$_M$ in comparison to traditional menus supplemented with audio feedback (preference results are deferred until after Study 3). Audio feedback was used because it succeeded in promoting hotkey use in [12] and in Study 1. The method was based on Study 1 (in turn based on [12]), with modifications described below.

Task and stimulus. Tasks involved selecting a menu item in response to an iconic target stimulus (e.g., a hat icon for the target 'Hat'). Iconic stimuli were used instead of text because pilot studies suggested that long and short word stimuli were easier to visually identify in the menu; iconic stimuli removed this confound: Figure 6 shows a menu and stimulus used in the study. The mechanism for presenting stimuli and completing selections was otherwise identical to Study 1.

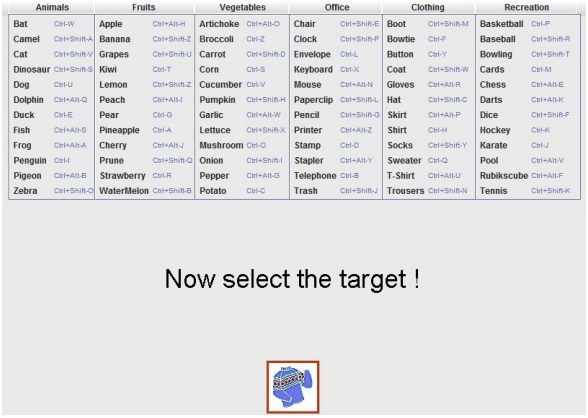

Figure 6. Example menu and stimulus used in Study 2.

Procedure. All participants completed the tasks using *ExposeHK$_M$* and *audio feedback*. When using *ExposeHK$_M$*, all the menus concurrently appeared when the Control key was pressed (Fig. 6). With both interfaces selections could be completed by pointing or by using a hotkey. In the *audio feedback* condition voice-synthesis read the hotkey binding (e.g., 'Control-D') whenever a selection was made using the pointer. The hotkeys associated with target items all consisted of a single letter plus the Control key. The menus were populated using Grossman et al.'s dataset, consisting of 6 menu categories, each containing 12 items (Fig. 6).

Participants received brief training prior to using each interface, which involved selecting an item 4 or more times using the pointer and hotkey (data discarded). They then completed six blocks of trials, with each block consisting of 24 selections, using the same distribution as Study 1.

Design, Participants and Apparatus. The experiment was designed as a 2×6×4 repeated-measures design for within-subjects factors *technique* (levels *ExposeHK$_M$* and *audio*), *block* and *frequency* (as for Study 1). Order of *Technique* was counterbalanced. The target sets used in the menus were also counterbalanced. All participants from Study 1 proceeded immediately to Study 2 using the same apparatus.

We inspected the data for evidence of asymmetric skill transfer, and proceeded with within-subjects ANOVA having found

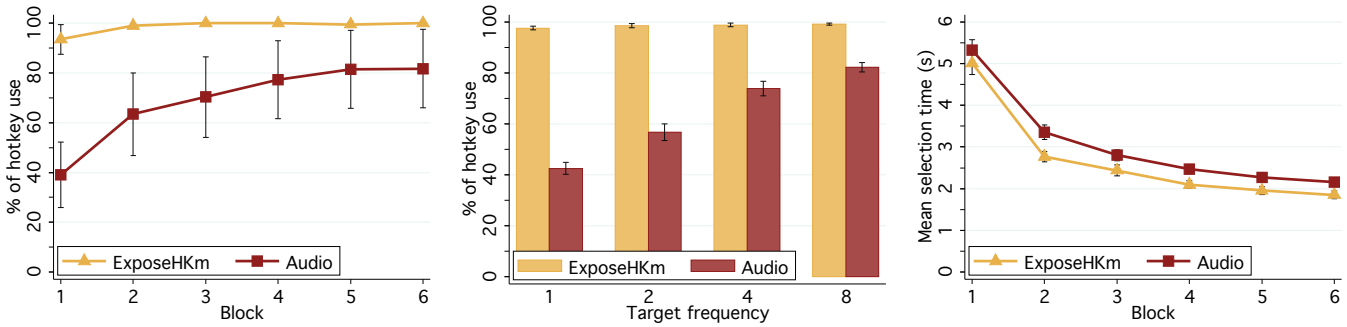

Figure 7. Results of Study 2. Left: % hotkey use by block; Center: % hotkey use by frequency; Right: mean time by block.

none (between-subjects analysis does not substantively change the results of significance tests).

Results

Proportion of hotkey use. There were strong main effects for *technique* ($F_{1,35} = 45.2, p < .001, \eta^2 = .56$), *frequency* ($p < .001, \eta^2 = .51$), and *block* ($p < .001, \eta^2 = .64$), as shown in Figure 7 (left and center). With ExposeHK$_M$, 99% of selections were completed using hotkeys, ranging from 93% in block 1 to 100% in block 6, compared with 64% with *audio*, ranging from 37% in block 1 to 77% in block 6. The continuous high hotkey use with ExposeHK$_M$, compared to the gradual increase with *audio* caused a significant *technique*×*block* interaction ($F_{5,95} = 20.3, p < .001, \eta^2 = .46$). There was also a significant *technique*×*frequency* interaction ($F_{3,105} = 32.6, p < .001, \eta^2 = .48$) caused by ExposeHK$_M$'s consistently high hotkey use across frequencies, but only 42% hotkey use for infrequent targets with *audio*.

Item selection time and errors. Figure 7 (right) shows the mean selection time (errors removed) with the different *techniques* across *block* (including both pointer-based and hotkey-based selections). Mean selection times with ExposeHK$_M$ (2.78s σ 1.8) were faster than *audio* (3.16s σ 2.02): $F_{1,35} = 15.06, p < .001, \eta^2 = .30$. As anticipated there were significant main effects of *block*, and *frequency* ($p < .001$), but *technique* did not interact with either.

Error rates were marginally lower with ExposeHK$_M$ (2.4%) than *audio* (3.0%): $F_{1,35} = 3.32, p = .08$.

Regardless of *technique*, the mean hotkey selection time was 2.53s, which was less than half that of pointer selections (5.9s). Hotkey error rates (2.4%) were also lower than pointer selections (5.9%), which we attribute to the narrow height of menu items causing relatively frequent 'off by one' errors.

In summary, the results provide similar validation to that of Study 1. With ExposeHK$_M$, participants chose to use hotkeys for nearly all commands almost immediately. In contrast, with audio feedback the transition to hotkeys was more gradual: hotkey use reached a lower maximum level, and hotkeys were comparatively rarely used with infrequent items.

STUDY 3: RIBBONS (EXPOSEHK$_R$)

Microsoft's Office 2007 'fluent' user interface merges toolbar and menu designs, using a tabbed 'ribbon' to separate multiple toolbars. Adapting EHK to the ribbon is challenging because each of several tabs places different items in the same spatial location. Also, the ribbon supports an Alt-key method for navigating its ribbon that is superficially similar to EHK. This raises two questions: can EHK be adapted to the ribbon, and does it improve performance over the Alt-key technique?

The ribbon adaptation, called ExposeHK$_R$, is visually similar to ExposeHK$_T$, but it additionally allows users to move between tabbed toolbars using the scrollwheel (as does the current Microsoft Ribbon interface) or arrowkeys. To remain consistent with the Ribbon's tabbed subsets, ExposeHK$_R$ relaxes goal 1, which motivated the concurrent presentation of all items with toolbars and menus.

There are two key differences between Microsoft's Alt-key interface and ExposeHK$_R$, both arising with respect to goal 4 (stable commands in a flat hierarchy). First, every selection with Alt-keys is necessarily hierarchical, involving a hotkey specification of the target tab and then the target item within the tab. Multi-level selections are necessary even when the correct target tab is pre-selected. With ExposeHK$_R$, in contrast, hotkeys are globally available in a flat hierarchy. Furthermore, selecting a hotkey with ExposeHK$_R$ has the side effect of switching to the tab that contains the target item, which is intended to assist with browsing hotkeys when users make a series of selections within the same tab. Second, Alt-keys are unstable, with the same letter representing different meanings in different modes (e.g., in Microsoft Word 'H' is used for Home, Shading, Header, and more), whereas ExposeHK$_R$'s commands are stable and mode-insensitive. These design differences have implications for the designer's choice of hotkey bindings, as discussed later.

Experimental method

Studies 1 and 2 focused on the proportion of hotkey use resulting from techniques that aim to promote them. In Study 3, the primary concern is the ultimate objective of hotkey interaction – improved performance – compared across three ribbon methods: ExposeHK$_R$, mouse selection, and Alt-keys.

Task and stimulus. Each task involved selecting an item within a ribbon-like interface (Figure 8) in response to a text stimulus representing the target. The mechanism for presenting stimuli and handling errors was otherwise identical to Study 2. The key-bindings used with Alt-keys all used Alt, then a

Figure 8. Example of a ribbon used in our experiment, with the ExposeHK$_R$ mode on.

single keypress for each of two levels (tab, then item). The key-bindings for ExposeHK$_R$ all used Control then a single keypress.

Procedure. All participants completed the tasks using *ExposeHK$_R$*, *Alt Keys* and *pointer* (mouse selection). With each condition they were required to complete selections using the primary interaction modality in order to get them to a performance maximum with that modality. Therefore, with ExposeHK$_R$ and *Alt Keys* they were required to use hotkeys and with *pointer* hotkeys were disabled. Before using each technique, participants received brief instruction and then completed eight random selections from a ribbon populated with a training set of items (data discarded). They were instructed to select targets as quickly and accurately as possible.

The ribbon used in the experiment consisted of 7 tabs with 18 items each. Participants completed 8 blocks of 12 selections with each interface. Each block comprised 2 selections of the same 6 targets, allowing participants to gain expertise across blocks. Half of the selections in each block involved switching to a different tab, allowing us to inspect the impact on performance of tab-switching with the different interfaces.

Design, Participants and Apparatus. Selection time data was analysed using a 3×8×2 analysis of variance for within-subject factors *technique* (*ExposeHK$_R$*, *Alt Keys* and *pointer*), *block* (levels 1-8), and *tab switch* (yes or no). Order of *technique* and the dataset used with each technique were counterbalanced across participants. 18 volunteers (aged 21-45, 1 female) participated. Experimental software, written in C#, ran on Windows 7 with a 1280×1024 22" display, optical mouse, and QWERTY keyboard.

Results

Figure 9 summarises the selection time results for the three *techniques* across *block* (left) and *tab switch* (right). Analysis of variance showed significant main effects for all factors: *technique* ($F_{2,34} = 34.5, p < .001, \eta^2 = .67$), *block* ($F_{7,119} = 111.2, p < .001, \eta^2 = .87$), and *tab switch* ($F_{1,17} = 282.5, p < .001, \eta^2 = .94$).

The overall mean selection time with ExposeHK$_R$ (2.86 s, σ 1.6) was 8.6% faster than pointer selections (3.13 s, σ 0.97), and 36% faster than Alt Keys. However, Figure 9 (left) shows a significant *technique*×*block* interaction, with ExposeHK$_R$ slightly slower than pointer in the first two blocks, but substantially faster (30%) than pointer by the final block (posthoc Bonferroni, $p < .05$). Pairwise comparison of performance in the final block shows significant differences between all three interfaces. The early asymptote of the pointer performance curve suggests that pointer selection rapidly reaches a relatively low performance ceiling.

Figure 9 (right) shows the significant *technique*×*tab switch* interaction ($F_{2,34} = 10.6, p < .001, \eta^2 = .67$), which is best attributed to the contrast between the pointer and ExposeHK$_R$ being similarly affected by the need to switch tabs, while performance with Alt Keys was more consistent across the tab switch levels. This is explained by the Alt Key mechanism requiring users to carry out identical hierarchical actions regardless tab switch state, whereas pointer and ExposeHK$_R$ actions are different depending on the need to switch tabs (they need to carry out additional actions to click on the item or to view the hotkey). With experience, however, once hotkeys are well learned, ExposeHK$_R$ users do not need to switch tabs, which best explains the three-way significant *technique*×*block*×*tab switch* interaction ($F_{14,238} = 2.81, p < .001, \eta^2 = .14$).

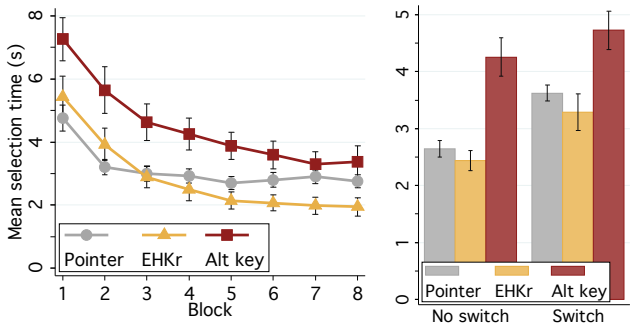

Figure 9. Mean selection time (s) per block (left) and condition (right)

SUBJECTIVE RESULTS (ALL STUDIES)

After each study participants completed NASA-TLX worksheets to assess subjective workload with each interface, they ranked the interfaces for overall preference, and they provided comments. In all cases, the participants' subjective responses strongly favoured EHK – 66.7% ranked it first for overall preference in Study 1, 77.8% preferred it over *audio* in Study 2, and 88.9% ranked it first in Study 3. Mean responses are shown in Table 1, with significant differences shown in bold (Friedman and Wilcoxon tests).

In studies 1 and 2, NASA-TLX measures showed that EHK had the lowest or equal-to-lowest workload measure in all categories. Several participants also commented that the audio feedback method was "annoying", and that they intentionally decided ignore its feedback. In study 3, EHK was assessed as resulting in lower physical demand, less frustration, and higher perceived success than pointing, but it had higher mental demand. We suspect that the low mental demand of pointing is a key factor in users failing to migrate to hotkeys, but we also suspect that once practiced, hotkeys require similar mental demand to pointing (e.g., using Control-C

for copy no longer requires much thought for most users). We return to this issue in the discussion.

	Study 1			Study 2		Study 3		
	eHK	T	A	eHK	A	eHK	Alt	P
Mental demand	**2.7**	**3.2**	**3.4**	2.6	2.9	**2.9**	**4.0**	**1.9**
Physical demand	**2.4**	**3.0**	**2.7**	**2.1**	**2.7**	**1.8**	**2.5**	**3.1**
Temporal demand	2.6	2.7	2.6	2.5	2.7	2.3	3.0	2.1
Hard work	**2.6**	**3.2**	**3.1**	**2.4**	**3.0**	**2.9**	**3.8**	**2.9**
Frustration	**2.1**	**2.6**	**3.0**	**2.0**	**2.7**	**1.9**	**3.2**	**2.7**
Successful	3.8	3.4	3.5	**4.0**	**3.7**	**4.2**	**3.0**	**3.2**

Table 1. Mean NASA-TLX and other values (1=low, 5=high) per experiment (Toolbars, Menus or Ribbon) and techniques (eHK: ExposeHK; T: Tooltips; A: Audio; Alt: Alt keys; P: Pointer). Significant effects for ExposeHK in bold.

DISCUSSION

Design goals 1-3 address issues in promoting users' transition to using hotkeys: helping novices browse hotkeys (goal 1), using physical rehearsal (goal 2), and drawing on spatial memory to assist hotkey identification and confirmation for intermediate users (goal 3).

Studies 1 and 2 showed that toolbar and menu adaptations of EHK improved on the state-of-the-art systems for promoting hotkey use. Participants made an early, comprehensive, and sustained switch to hotkeys when using EHK, whereas standard tooltips and post-selection audio feedback resulted in slower hotkey adoption and lower levels of use. Participants, mostly university staff and students, were almost certainly aware of keyboard shortcuts as an interface mechanism, possibly increasing hotkey use for all conditions. Finally, participants also strongly preferred EHK to the other conditions tested (audio feedback and contemporary designs).

Design goal 4 addresses ultimate performance with hotkeys by advocating stable bindings that are globally applicable in a flat interface hierarchy. Study 3 therefore examined user performance with EHK when adapted to a contemporary 'ribbon' design of tabbed toolbars, comparing performance with standard pointer-based selection and with a commercial hierarchical 'Alt' Keys interface. Results showed that EHK quickly outperformed the pointer by up to 30% on selection time. EHK substantially outperformed the Alt keys interface throughout.

Together the studies suggest that user performance in command selection can be improved by incorporating EHK in standard toolbar, menu, and ribbon interfaces. With EHK novice users have a mechanism to browse hotkeys without using the pointer, and they can issue hotkey commands as a rehearsal of the expert mechanism. Intermediate users can draw on their spatial knowledge of commands to rapidly find hotkey bindings. Finally, the ultimate performance of experts is improved through stable bindings and flat hierarchies.

The following subsections discuss issues of deployment and identify promising areas for further work.

Discoverability and pointer compulsion

The studies show that when participants are aware of EHK they use it to their advantage and are enthusiastic about it. However, all three studies included explicit instruction and practice with EHK prior to experimental tasks, which raises questions of how it will be discovered and used in practice. We see two challenges for practical deployments, related to its *discoverability* and to the need to break users' inertia in continuing with pointer-based selections (or *pointer compulsion*).

Discoverability. Interactions such as EHK have no visual representation to aid their discovery until the user accidentally or deliberately presses its modifier-key trigger. Similar problems of weak affordances are present in many forms of interaction, such as touchscreens, where the number of taps, the number of fingers, the tap location, or press duration can be used to control different interactive effects, without any visual representation. In such cases designers often use techniques such as on- and off-line tutorials, 'tip of the day' utilities, and general marketing to promote discovery and awareness.

Social interactions around computing (e.g., over-the-shoulder observation) also play a major role is disseminating useful system capabilities [25]. Consequently, useful facilities can become widely known even when their presentation is subtle or relatively cryptic.

Pointer compulsion. Studies 1 and 2 explicitly instructed users to make a series of pointer-based and hotkey-based selections with each interface prior to beginning experimental tasks. However, in pilot studies, some participants completed selections using the pointer despite clearly displayed instructions to use hotkeys, and they continued to do so until verbally instructed to use the hotkeys. Breaking this inertia of pointer-based interaction is challenging, but doing so is a primary objective of goals 1-3.

Regardless of these challenges to adoption, it is important to note that EHK is largely compatible with existing designs – it supplements their functionality without changing their basic behavior, allowing users to maintain existing interaction strategies without performance detriment, but also offering a higher performance ceiling if used.

In on-going work we are developing an application that will incorporate ExposeHK$_M$ into Mac OS X menus. This will allow us to gain insights into longitudinal use of EHK in real-world deployments.

Modifier keys and vocabulary

The hotkeys in our experiments used only the Control key as a modifier. However, some applications use multiple modifiers and modifier combinations to increase hotkey vocabulary (e.g. Control, Alt, and Control+Alt). Although we have not yet evaluated multiple modifiers, we suspect that the results will generalise to modifier key combinations. Importantly, design guidelines advocate using single modifier keys (like those evaluated) for important commands: for example, Apple recommends that developers "use the Command key as the main modifier key in a keyboard shortcut" [21]. Furthermore, our analysis of how hotkeys are used in 30 mainstream applications showed that 93.4% (sd 12.7%) of commands use the Command key, suggesting that our method is representative of contemporary hotkey deployments.

When multiple modifiers are required, EHK could be further adapted to help users learn the modifier key bindings. For example, the display mechanism for exposing hotkeys could

highlight the items accessed with the currently depressed modifier key (e.g., using a different color or opacity), while also showing all other hotkey bindings to assist users who are unsure of the required modifier for their target.

By advocating for stable hotkey bindings in a globally accessible flat hierarchy, goal 4 limits the number of commands accessible through a single hotkey character with each single modifier key to ≈ 45 (alphanumerical plus special characters). In contrast, a two-level Alt hierarchy that uses a single character at each level gives access to up to $t \times 45$ (plus numbers and special characters) commands (where t is the number of top-level tabs). Pragmatically, though, we suspect that very few users know as many as 45 hotkeys in any existing system.

Adaptation to other interfaces

Several design solutions could be used to adapt EHK to menu and toolbar combinations. Most toolbar items are replicated in menus, so the simplest solution would be to allow the menus to occlude the toolbar. However, this would impair intermediate users' ability to use their spatial memory of items in the toolbar (opposing goal 3). Alternatively, different modifier keys, or different actions on the same modifier key, could be used to trigger toolbar and menu hotkey exposure: for example, a single Control key press might show ExposeHK$_{\mathrm{T}}$, and a second one ExposeHK$_{\mathrm{M}}$.

By displaying all of the available menu commands at once, ExposeHK$_{\mathrm{M}}$ effectively turns the command-access interface into a CommandMap [30]. Scarr et al. demonstrated that pointer-based interaction with CommandMaps substantially outperforms ribbon and menu designs because it replaces two actions (each involving search, decision, and pointing) with a single one. Consequently, even when items within ExposeHK$_{\mathrm{M}}$ have no associated hotkey, ExposeHK$_{\mathrm{M}}$ should improve performance over traditional schemes.

CONCLUSION

Keyboard shortcuts such as hotkeys are widely deployed in current applications but they are known to be underused. We presented ExposeHK, a simple interactive technique that overlays hotkeys onto existing widgets when a modifier key is pressed. Its design promotes hotkey use and enhances efficiency through four goals: enable hotkey browsing; support physical rehearsal; support rapid hotkey identification; and support stable commands in a flat hierarchy. We presented three ExposeHK exemplars that work with the main methods for organising commands in graphical user interfaces: toolbars, menus, and tabbed 'ribbon' toolbars. Three studies, one with each of the designs, demonstrated that the adaptations succeed in promoting earlier and higher levels of hotkey use than other methods, that they improve user performance, and that they are subjectively preferred to alternatives.

REFERENCES

1. Alexander, J. *Understanding and Improving Navigation Within Electronic Documents*. PhD thesis, University of Canterbury, Christchurch, New-Zealand, 2009.
2. Appert, C., and Zhai, S. Using strokes as command shortcuts: cognitive benefits and toolkit support. *CHI '09*, ACM, 2289–2298.
3. Bailly, G., Lecolinet, E., and Nigay, L. Flower menus: a new type of marking menu with large menu breadth, within groups and efficient expert mode memorization. *AVI '08*, ACM, 15–22.
4. Bau, O., and Mackay, W. Octopocus: a dynamic guide for learning gesture-based command sets. *UIST '08*, ACM, 37–46.
5. Bhavnani, S. K., and John, B. E. The strategic use of complex computer systems. *Hum.-Comput. Interact. 15*, 2 (Sept. 2000), 107–137.
6. Callahan, J., Hopkins, D., Weiser, M., and Shneiderman, B. An empirical comparison of pie vs. linear menus. *CHI '88*, ACM, 95–100.
7. Card, S., Moran, T., and Newell, A. The keystroke-level model for user performance time with interactive systems. *Commun. ACM 23* (July 1980), 396–410.
8. Card, S., Newell, A., and Moran, T. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., 1983.
9. Carroll, J. M., and Rosson, M. B. *Paradox of the active user*. MIT Press, Cambridge, MA, USA, 1987, 80–111.
10. Cockburn, A., and Gutwin, C. A predictive model of human performance with scrolling and hierarchical lists. *HumanComputer Interaction 24*, 3 (2009), 273–314.
11. Common commands in speech recognition. http://windows.microsoft.com/en-NZ/windows-vista/Common-commands-in-Speech-Recognition.
12. Grossman, T., Dragicevic, P., and Balakrishnan, R. Strategies for accelerating on-line learning of hotkeys. *CHI '07*, ACM, 1591–1600.
13. Hart, S., and Staveland, L. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Human mental workload*, Elsevier, 139–183.
14. Hornof, A. Visual search and mouse-pointing in labeled versus unlabeled two-dimensional visual hierarchies. *ACM Trans. Comput.-Hum. Interact. 8*, 3 (Sept. 2001), 171–197.
15. Howes, A., Payne, S. J., and Woodward, A. The trouble with shortcuts. *CHI '00 Extended Abstracts*, CHI EA '00, ACM, 267–268.
16. Jorgensen, A., Garde, A., Laursen, B., and Jensen, B. Using mouse and keyboard under time pressure: Preference, strategies and learning. *Behaviour & Information Technology 21*, 5 (2002), 317–319.
17. Krisler, B., and Alterman, R. Training towards mastery: overcoming the active user paradox. *NordiCHI '08*, ACM, 239–248.
18. Kurtenbach, G., and Buxton, W. Issues in combining marking and direct manipulation techniques. *UIST '91*, ACM, 137–144.
19. Kurtenbach, G. P. *The design and evaluation of marking menus*. PhD thesis, University of Toronto, Ontario, Canada, 1993.
20. Lane, D., Napier, A., Peres, C., and Sandor, A. The Hidden Costs of Graphical User Interfaces: The Failure to Make the Transition from Menus and Icon Tool Bars to Keyboard Shortcuts. *International Journal of Human-Computer Interaction 18* (2005), 133–144.
21. Mac os x human interfaces guidelines. https://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/Intro/Intro.html.
22. Matejka, J., Li, W., Grossman, T., and Fitzmaurice, G. Communitycommands: command recommendations for software applications. *UIST '09*, ACM, 193–202.
23. Miller, C. S., Denkov, S., and Omanson, R. C. Categorization costs for hierarchical keyboard commands. *CHI '11*, ACM, 2765–2768.
24. Odell, D., Davis, R., Smith, A., and Wright, P. Toolglasses, marking menus, and hotkeys: a comparison of one and two-handed command selection techniques. *GI '04*, 17–24.
25. Peres, C., Tamborello, F., Fleetwood, M., Chung, P., and Paige-smith, D. Keyboard shortcut usage: The roles of social factors and computer experience. *HFES '04*, 803–807.
26. Poulton, E., and Freeman, P. Unwanted asymmetrical transfer effects with balanced experimental designs. *Psychological Bulletin 66*, 1 (1966).
27. Quinn, P., and Cockburn, A. The effects of menu parallelism on visual search and selection. *AUIC '08*, Australian Computer Society, 79–84.
28. Rekimoto, J., Ishizawa, T., Schwesig, C., and Oba, H. Presense: interaction techniques for finger sensing input devices. *UIST '03*, ACM, 203–212.
29. Richardson, J. T. Eta squared and partial eta squared as measures of effect size in educational research. *Educational Research Review 6*, 2 (2011), 135–147.
30. Scarr, J., Cockburn, A., Gutwin, C., and Bunt, A. Improving command selection with commandmaps. *CHI '12*, ACM, 257–266.
31. Scarr, J., Cockburn, A., Gutwin, C., and Quinn, P. Dips and ceilings: understanding and supporting transitions to expertise in user interfaces. *CHI' 11*, ACM, 2741–2750.
32. Schmidt, R. A., and Lee, T. D. *Motor control and learning: A behavioral emphasis*, vol. 3. Human Kinetics, 2005.
33. Schneider, W., and Chein, J. Controlled & automatic processing: behavior, theory, and biological mechanisms. *Cognitive Science 27*, 3 (2003), 525–559.