KeyMap: Improving Keyboard Shortcut Vocabulary Using Norman's Mapping

Blaine Lewis*
University of Toronto
Toronto, Canada
blaine@dgp.toronto.edu

Greg d'Eon*
University of British Columbia
Vancouver, Canada
gregdeon@cs.ubc.ca

Andy Cockburn
University of Canterbury
Christchurch, New Zealand
andy@cosc.canterbury.ac.nz

Daniel Vogel
University of Waterloo
Waterloo, Canada
dvogel@uwaterloo.ca

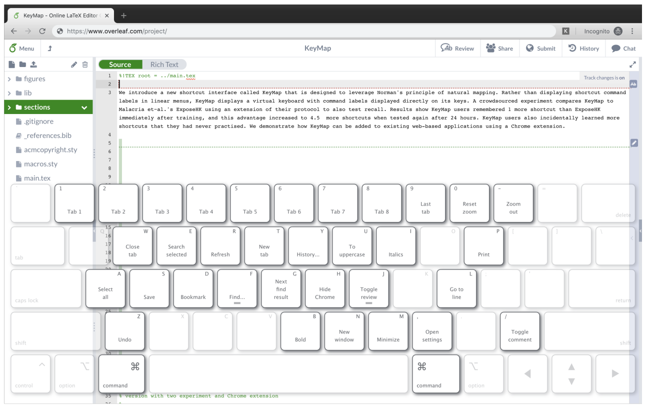
ABSTRACT
We introduce a new shortcut interface called KeyMap that is designed to leverage Norman's principle of natural mapping. Rather than displaying shortcut command labels in linear menus, KeyMap displays a virtual keyboard with command labels displayed directly on its keys. A crowdsourced experiment compares KeyMap to Malacria et al.'s ExposeHK using an extension of their protocol to also test recall. Results show KeyMap users remembered 1 more shortcut than ExposeHK immediately after training, and this advantage increased to 4.5 more shortcuts when tested again after 24 hours. KeyMap users also incidentally learned more shortcuts that they had never practised. We demonstrate how KeyMap can be added to existing web-based applications using a Chrome extension.

Author Keywords
interaction techniques; learning; keyboard shortcuts

CCS Concepts
•**Human-centered computing** → **Human computer interaction (HCI);** *Empirical studies in HCI; Interaction techniques;*

INTRODUCTION
Keyboard shortcuts (or "hotkeys") are an important part of desktop applications: they allow expert users to input commands quickly, with little motor action or visual distraction [12, 15]. However, despite these advantages, users fail to adopt keyboard shortcuts over slower graphical user interface (GUI) techniques. Most users have a small shortcut vocabulary [12] for common commands such as "copy" and "paste", and it is difficult to remember and use new shortcuts. An important reason for this difficulty is that shortcuts are typically hidden in GUI-activated tooltips, dropdown menus, or help screens, so they are hard to find, and using a shortcut for the

[*]The first and second authors contributed equally to this paper.



```
%!TEX root = ../main.tex

We introduce a new shortcut interface called KeyMap that is designed to leverage Norman's principle of natural mapping. Rather than displaying shortcut command
labels in linear menus, KeyMap displays a virtual keyboard with command labels displayed directly on its keys. A crowdsourced experiment compares KeyMap to
Malacria et-al.'s ExposeHK using an extension of their protocol to also test recall. Results show KeyMap users remembered 1 more shortcut than ExposeHK
immediately after training, and this advantage increased to 4.5  more shortcuts when tested again after 24 hours. KeyMap users also incidentally learned more
shortcuts that they had never practised. We demonstrate how KeyMap can be added to existing web-based applications using a Chrome extension.
% version with two experiment and Chrome extension
```

Figure 1. KeyMap is an interaction technique for improving keyboard shortcut vocabulary. It displays command labels on a virtual keyboard upon pressing and holding a modifier key. The figure shows KeyMap being used with Overleaf. For a closer view of KeyMap see Figure 2.

first time requires switching away from the GUI. This difficulty contributes to a production bias [5], where users sacrifice long-term productivity in favour of short-term productivity.

Several interaction techniques and design guidelines have been proposed to improve the adoption of keyboard shortcuts. Skillometers [14] motivate users to learn shortcuts by visualizing their potential benefits. Grossman et al. [10] explored design guidelines, such as adding auditory cues and disabling menu items to help users identify and memorize shortcuts. ExposeHK [13] displays all available commands at once when a user presses a modifier key, such as ctrl, helping them rehearse the action of inputting a keyboard shortcut. However, few previous techniques have focused on making shortcuts more memorable through their visualization and layout.

We introduce KeyMap, an interactive guidance technique that makes keyboard shortcuts more memorable (Figure 1). When a modifier key is held down, KeyMap displays an on-screen keyboard with command names labelled on the keys. The interface is inspired by Norman's principle of *natural mapping*, matching the display of the shortcuts to the physical layout of the keyboard. This layout makes shortcuts more memorable due to spatial memory, giving users memory cues with greater cue-target strength, while also helping users discover and practice keyboard shortcuts.

A crowdsourced study ($n = 98$) compares KeyMap's performance to the current state-of-the-art, Malacria et al.'s ExposeHK [13]. Using Malacria et al.'s 13-command protocol extended to measure recall, KeyMap participants remembered 1 more command immediately after training, and 4.5 more 24-hours later. Further, KeyMap promotes incidental learning: 14 KeyMap participants remembered at least one command that they had never previously entered. We provide an open source Chrome extension[1] to show KeyMap is easily added to existing applications.

We make two contributions: (1) a demonstration that using Norman's natural mapping also makes interfaces more memorable; (2) a new technique that improves shortcut memorization compared to the state-of-the-art.

RELATED WORK

We review why people fail to use keyboard shortcuts and previous techniques designed to encourage shortcut adoption.

Keyboard Shortcuts in Practice

Keyboard shortcuts are among the fastest interaction methods available to users [17, 12, 10, 19, 14]. This speed is due to low *psycho-motor demands* and *workflow disruption*. To illustrate these factors, consider a user typing a document with both hands. To select a command from a menu, they must home one hand to the mouse, "wiggle" the mouse to display and identify the cursor, move the cursor to the menu, select the command, move the cursor back to the document, and home the hand back to the keyboard. This process can disrupt their workflow, and many of these steps require hand-eye coordination. Both of these issues are greatly improved when using keyboard shortcuts.

Despite this potential, keyboard shortcuts are used relatively infrequently [12] and typical users only know a small vocabulary of common commands, such as "copy" and "paste" [2]. One important reason for this low adoption rate is the *paradox of the active user* [5], where users fail to seek higher performance due to cognitive biases. With shortcuts, there is a "production bias", where users sacrifice long-term productivity to preserve their short-term productivity. This is because learning and practising new shortcuts creates an initial productivity "dip" since using shortcuts is so different than GUI interaction [19, 9]. In general, rather than optimising absolute performance, users often *satisfice* [21] by choosing a technique like clicking on a menu or toolbar because it is "fast enough" [22].

Improving Shortcut Adoption

Adopting regular use of a keyboard shortcut involves four steps. First, a user becomes motivated to learn a new shortcut for a command. Second, they discover the relationship between a command and its shortcut: for instance, that "copy" is ctrl+c. Third, they rehearse the shortcut action and begin practising it. Fourth, they memorize the shortcut mapping and fully adopt using the shortcut for the command. With continued usage, the shortcut action ultimately becomes "muscle memory" and they attain *automaticity* [20]. Prior work has addressed one or more of these four steps.

To motivate users to learn keyboard shortcuts, Skillometers [14] display a user's overall command selection performance, and also show how much they could improve if they used shortcuts. Grossman et al. [10] evaluated more forced motivational approaches, including adding a 2-second delay after each command selected with a menu and disabling menu selections all together.

Other interfaces help users discover new keyboard shortcuts. For example, to make shortcuts more prominent in the GUI, IconHK [8] integrates visual hints about keyboard shortcuts into command icons. Other methods communicate associated shortcuts after menu commands. Blur [19] shows shortcut sequences for recent menu commands in a "calm notification" window, and Grossman et al. [10] communicate the associated shortcut with a visual cue card or using speech synthesis audio.

The HotKeyCoach [11] also uses a cue card to aid discovery, but it adds the ability to rehearse the shortcut in a special mode. Malacria et al. [13] note that shortcuts are often hidden in menus or tooltips that appear when using the GUI, not the keyboard. Instead, they propose ExposeHK, a techique that displays all associated shortcuts across all GUI menus when a modifier key is pressed. This helps users discover shortcuts while also rehearsing the initial action of entering a shortcut. Experiments showed this improved shortcut usage over IconHK [8] tooltips or Grossman et al.'s [10] audio feedback. We consider ExposeHK the state-of-the-art, and it forms the benchmark for our comparison.

Some of the previous techniques have indirectly addressed the fourth step of memorizing shortcut mappings. For example, Grossman et al. found that disabling menu items strongly motivated users to commit shortcuts to memory, and using audio for discovery also helped users to remember shortcuts. While previous works such as Grossman et al. and Malacria et al. draw conclusions about memorization from their experiments, these are based on rates of shortcut usage compared to GUI commands. No previous shortcut adoption techniques have explicitly tested for memorization performance using formal short-term and long-term recall tests [3]. Measuring shortcut memorization is crucial to understanding technique performance, and improving memorization rates is necessary to expand a user's vocabulary of shortcuts.

In summary, previous techniques focused on shortcut motivation, discovery, and rehearsal. Our aim is to make keyboard shortcuts more memorable as well.

KEYMAP

KeyMap (Figure 2) is an interface that aims to improve keyboard shortcut adoption in three ways. Like ExposeHK [13], KeyMap makes it easier to *discover* and *rehearse* shortcuts, but it is also designed to make shortcuts more *memorable*. We first describe the KeyMap interface, then provide theoretical grounding for hypothesized benefits.

KeyMap Interface and Interaction

KeyMap is partially inspired by the visual design of Hotkey Palette [1], a general purpose interaction technique for document and window retrieval. It uses an on-screen keyboard with

[1] Chrome extension: `https://chrome.google.com/webstore/detail/fpminkfnndfokkmoobbngdpnijbcajkl`, with source code at `https://github.com/blainelewis1/keymap`

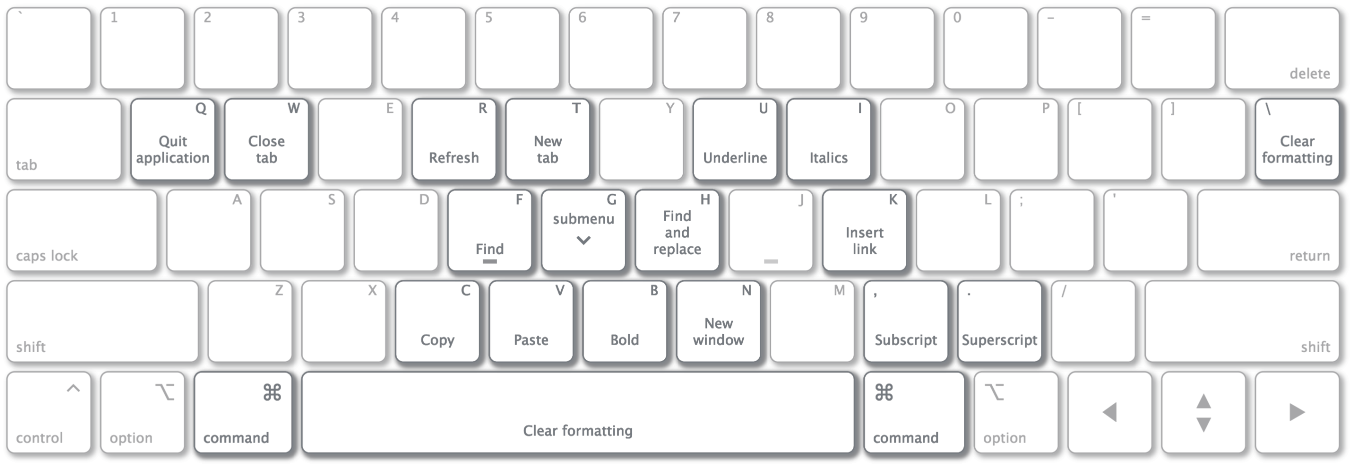


Figure 2. Example KeyMap interface shown while the Command modifier is held down, each shortcut key associated with Command is rendered with the name of the corresponding application command.

previews of resources displayed inside their associated keys. A resource is opened using Fn and the key, or by clicking on the rendered key. However, the technique does not support shortcuts for commands and no evaluation was presented.

KeyMap also uses a visualization of a physical keyboard, rendered near the bottom of the screen. It appears after a short delay when the user presses and holds a modifier key (e.g. shift, command, ctrl). Keys are labelled with associated commands. These commands are typically from the focused application, but global commands can be included. To maximize available screen space, only the bottom left modifier keys are shown until activated.

Users can activate KeyMap and select commands in several ways. They can click a modifier key to display the full keyboard, then click the key for the command. More likely, they can hold down a modifier key on the physical keyboard, use the keyboard visualization to scan for a command, then press the corresponding physical key. They can also mix these strategies by typing a modifier, then clicking on a key. Finally, expert users can simply type shortcuts without waiting for the visualization to appear.

KeyMap uses visual embellishments to help users identify and remember shortcuts. One example is using colour-coding to represent semantic groupings, helping users quickly browse and scan commands. For instance, commands from an "edit" menu could be coloured purple. Showing recognizable command icons on the keys could help users quickly identify common commands. The on-screen keyboard also includes important visual details, such as the nubs on the f and j keys.

In the experiments that follow, we evaluate a rudimentary version of KeyMap with fixed command sets. However, a deployed version of the system could dynamically populate shortcut-to-command mappings using tools such as Microsoft's UI Automation API, which was used in Blur [19], to discover and display shortcuts automatically. In the Discussion section, we describe how our Chrome extension populates shortcut sets.

KeyMap Benefits

The core principle behind KeyMap is that a visual interface should match as closely as possible to the physical actions used to operate it. This tight correspondence between display and input is an instance of Norman's natural mapping [16], but we leverage this principle in a unique way. Normally, natural mapping is used to make interfaces intuitive; instead, we use it to encourage users to learn and remember shortcuts. Below we explain how leveraging Norman's mapping leads to this and other potential benefits.

Memorization

KeyMap helps users memorize shortcuts by leveraging spatial memory. Keyboard layouts have many recognizable "landmarks" [24], such as the distinctive shape of certain keys, nubs on the f and j keys, and a consistent and familiar layout. Together, these make a keyboard image well suited for spatial memory when commands are associated with keys. A keyboard layout also helps users learn spatial relationships *between* commands. For instance, if the user knows that ctrl+ c activates "copy" and notices that "copy" is beside "paste", they can infer that the shortcut for "paste" is next to c on their keyboard.

In particular, KeyMap's visualization helps users' *cued recall* process. When entering a keyboard shortcut, users think of a command (a *cue*), and they must *recall* the keyboard shortcut based on this cue. KeyMap improves this process by increasing the strength of the cue-target association [4]. For example, consider the "replace" command, with the shortcut ctrl+h. This command can be difficult to remember since the command's name has nothing in common with its keyboard shortcut, giving no cue-target association. KeyMap strengthens this cue-target association by matching the display of the command with its shortcut execution. KeyMap's layout also creates new potential information for cues. For instance, the similar commands "find" and "replace" use the nearby keys f and h, improving the cue-target association.

Discovery
Like ExposeHK, KeyMap supports browsing for shortcuts by displaying shortcuts when the user presses a modifier key or combination. Importantly, this browsing process can be done solely with the keyboard, and it does not require using the mouse. Additionally, the same features that support spatial memory also help users rapidly scan for commands, speeding up the visual search process [18].

Once the user has some knowledge of commands, KeyMap facilitates discovery of further shortcuts by using spatial and colourised groupings. Related commands appear in nearby locations and have similar colours. These visual details let KeyMap represent semantic relationships without a traditional menu structure, helping users learn commands that are similar to the ones they already know.

KeyMap also helps users learn new commands through *incidental learning*. This benefit is unique to KeyMap. In a regular menu, commands that are next to each other are typically semantically related, but their shortcuts might be completely different. In contrast, commands that appear near each other in the KeyMap visualization must have shortcuts with similar motor actions. Thus, the same spatial relationships that help with spatial memory can also help users discover and learn shortcuts for new commands, even without practicing them.

Rehearsal
When a user is rehearsing a shortcut, they hold down a modifier key, wait for the visualization to appear, find the correct command, and type the shortcut key. Skilled users simply repeat these actions without waiting for visual guidance. As in ExposeHK, users can browse commands without using a pointing device, making the novice action useful practice for the expert action and easing the novice-to-expert transition [19].

However, the KeyMap display provides further reinforcement for these actions. When the user selects a command with the pointer, ExposeHK has no element of rehearsal, but KeyMap still helps users learn the positions of commands due to its use of Norman's mapping. Although clicking a command is not a rehearsal of a keyboard shortcut motor action, it still helps users learn the location of the shortcut on their keyboard.

Faster Execution
Finally, by using Norman's mapping, KeyMap also has higher *stimulus-response* (SR) *compatibility* [7] compared to ExposeHK. This effect states that human response times are faster when the response action is *compatible* with its stimulus. For instance, a classic experiment tested participant reaction times when pressing buttons on their left or right in response to lights on their left or right, and found slower reaction times when the light-button relationship was crossed. These effects generalize to a wide range of activities [23].

In KeyMap, the *response* of typing a shortcut is compatible with the *stimulus*, which is laid out like a keyboard. Thus, when users are rehearsing shortcuts, they should enter shortcuts more quickly using KeyMap than with a standard menu or ExposeHK. Additionally, if KeyMap users do benefit from the improved memory, they can leverage memory guided visual search [25]. Thus, KeyMap should benefit from a slightly faster execution due to SR compatibility and memory guided visual search, but it is unclear whether this effect will be strong enough to observe.

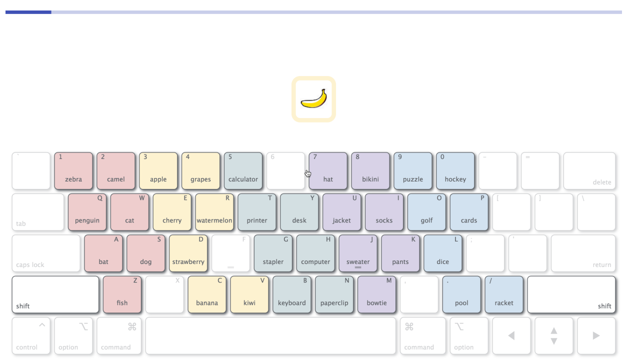


Figure 3. The selection task interface. The interface includes the shortcut technique (in this case, KeyMap), an image stimulus with a coloured border, and an experiment progress bar. The progress bar was at the top for KeyMap and the bottom for ExposeHK. Stimuli were placed such that the technique did not occlude them.

EXPERIMENT
We conducted a crowdsourced experiment to compare KeyMap to ExposeHK for improving memory and shortcut use. Our protocol follows that of Malacria et al. [13], but we extend it with two tests to examine shortcut memory: a recall test at the end of the training session, and a retention test 24 hours later. The experiment tests three main hypotheses:

- **H1:** KeyMap users remember more shortcuts than ExposeHK users.
- **H2:** KeyMap users incidentally learn more shortcuts than ExposeHK users.
- **H3:** KeyMap users have faster practiced selection times than ExposeHK users. However, we also expected KeyMap to initially be slower because users need to identify the mapping between command groups and colours, which is more clearly labelled in ExposeHK.

Command Selection Techniques
We tested fundamental, feature-comparable versions of KeyMap and ExposeHK. To reduce potential confounds stemming from KeyMap's use of coloured command groupings, command categories were coloured in both ExposeHK and KeyMap (Figures 3 and 4). Both techniques support three methods of command selection: *pointer-based*, *guided*, and *shortcut*, explained below.

KeyMap
A set of buttons representing modifier keys is displayed near the bottom left of the screen. In a *pointer-based* selection, the user begins by clicking on a modifier button, which displays the KeyMap visualization across the bottom of the screen (Figure 3). Categories of similar commands appear as coloured groups of keys, but these categories are not labelled. When the user clicks a key on the onscreen keyboard, the associated command is selected, and the visualization disappears.

In a *guided* selection, the user presses and holds the shift modifier key on the physical keyboard. The KeyMap visualization appears after a 500 ms delay. Once the desired command is found, the user presses the associated physical shortcut key to select the command. Finally, in a *shortcut* selection, the user simply executes the associated keyboard shortcut (e.g. shift+A). In this case, there is no need to wait for the 500 ms delay and the visualization never appears.

ExposeHK

A linear menu bar is displayed across the top of the display (Figure 4). Each menu is labelled with its category name, and all items within each category are identically coloured. In a *pointer-based* selection, the user clicks the label of a menu to expand it, then completes a command selection by clicking on the intended menu item. In a *guided* selection, the user presses and holds the shift key for 500 ms to expand the menus, then searches for a command and presses the corresponding key. Lastly, *shortcut* selections are identical to KeyMap: the user presses shift and the corresponding letter.

Unlike the original ExposeHK, we added a 500 ms delay before displaying either technique visualization. This delay allows us to distinguish between *guided* and *shortcut* invocations. We also chose to use the shift modifier key for all shortcuts, maintaining consistency across operating systems (Windows ctrl vs. OS X command) and avoiding conflicts with existing web browser shortcuts.

Tasks

During the experiment, participants complete two types of tasks: *selection* and *recall*. The *selection* task is closely based on Malacria et al.'s experiment. First, the participant positions their cursor in a 70×70 px in the centre of the screen and presses the space key. This step emulates a real command selection, where the mouse begins on a document. Next, an image stimulus appears, representing a command to be entered, with a coloured border indicating the command's category. The image is positioned to avoid occluding the technique visualizations. The participant selects the target command with their interface using one of the three selection methods. If they select an incorrect command, the screen flashes red, and "Incorrect" appears on the screen with a 3 second timer before continuing to the next trial.

The *recall* task was added to test participants' memory of shortcuts. In a recall task, an image stimulus appears without a coloured border. Participants simply enter a keyboard shortcut to continue to the next trial. The KeyMap or ExposeHK visualizations do not appear, and the participant is not told whether the shortcut was correct.

We adapted command names and image stimuli from Grossman et al. [10]. We updated some command names and icons for clarity (e.g. the watermelon icon was unclear). We also removed the "vegetables" category from the main experiment to avoid semantic conflict with "fruits," leaving 60 commands in 5 categories. To avoid any effects if some commands are more memorable than others, we used a random subset of 35 commands for each participant, regardless of condition.

Procedure

The session began with a consent form and a pre-study questionnaire about demographics, computer information, and keyboard shortcut use. Participants were then randomly assigned to use KeyMap or ExposeHK. Next, they selected one of five keyboard layouts that best matched the physical layout of their keyboard. Then, they followed an interactive tutorial that explained how to use all three selection methods of the assigned technique and introduced them to the selection task. The tutorial used commands from the "vegetables" category, which did not appear during the main experiment.

After the tutorial, participants completed 6 blocks of selection tasks. Each block contained 25 trials comprising selections of 13 different commands (some repeated). Participants were encouraged to take a short rest between blocks. After the final block, they completed two sets of recall tasks. The first set tested all 13 commands that appeared during the selection tasks. The second tested five additional commands that did not appear in the selection tasks, with one command randomly chosen from each category. Finally, participants answered a post-study questionnaire asking how many shortcuts they thought they remembered and subjective questions about their overall experience.

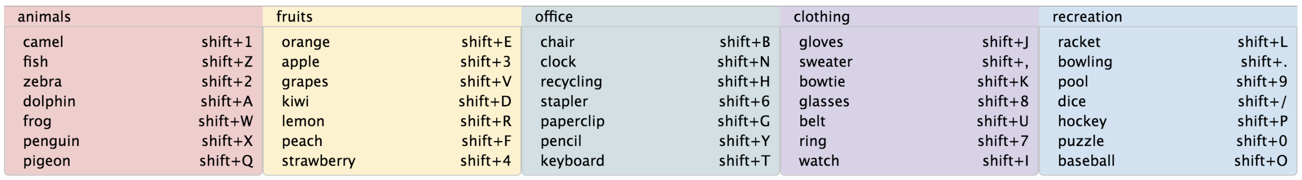
On the final screen, we reminded participants that there would be a short followup session. We contacted them 24 hours later with a link to the followup experiment. If they accepted, they completed two sets of recall tasks using the same commands as the main session. Then, they answered a followup questionnaire asking again how many shortcuts they thought they remembered.

We were worried that participants might take screenshots of the visualizations to aid their memory. We combatted this concern in two ways. First, participants were not aware that their recall would be tested until the end of the experiment, so they had little reason to record the menu until too late to do so. Second, at the end of the followup session, we asked participants if they recorded the menu. We worded this question so it was clear that there were no negative implications for their participation or monetary reward if they had recorded the screen.

Design

We followed Malacria et al.'s protocol for examining command selection. We used a truncated Zipfian distribution of 13 commands, with each command selected the following number of times per block: [8, 4, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1]. This distribution has a heavy skew towards the most common commands, reflecting real command frequencies [6]. These commands were assigned to shortcut keys in the same way as Malacria et al. The most frequent command (8 times per block) used a key from the left side of the keyboard (`123 qwe asd zxc`), the second most frequent (4 times) used a key from the right side (`7890 uiop jkl; m,./`), and the third most frequent (2 times) used a key from the center (`456 rty fgh vbn`). The remaining 10 commands were distributed such that 2 were assigned to each category.

The experiment uses a $2 \times 6 \times 4$ mixed-factorial design. TECHNIQUE is a between-subjects factor with 2 levels:



animals		fruits		office		clothing		recreation	
camel	shift+1	orange	shift+E	chair	shift+B	gloves	shift+J	racket	shift+L
fish	shift+Z	apple	shift+3	clock	shift+N	sweater	shift+,	bowling	shift+.
zebra	shift+2	grapes	shift+V	recycling	shift+H	bowtie	shift+K	pool	shift+9
dolphin	shift+A	kiwi	shift+D	stapler	shift+6	glasses	shift+8	dice	shift+/
frog	shift+W	lemon	shift+R	paperclip	shift+G	belt	shift+U	hockey	shift+P
penguin	shift+X	peach	shift+F	pencil	shift+Y	ring	shift+7	puzzle	shift+0
pigeon	shift+Q	strawberry	shift+4	keyboard	shift+T	watch	shift+I	baseball	shift+O

Figure 4. The expanded version of ExposeHK.

KeyMap or *ExposeHK*. BLOCK is a numeric within-subjects factor ranging from 1 to 6. FREQUENCY is a within-subject factor with levels 8, 4, 2, and 1 based on the Zipfian command distribution. Each participant completed 25 trials $\times$ 6 blocks = 150 selections.

We recorded three dependent measures. *Selection time* was measured during the selection task. A selection began as soon as the previous trial was complete, and continued until the participant began the next trial. This time therefore includes the mouse centering task and command selection. *Recall* is the total number of correct selections made during the recall tasks in the main experiment session. *Retention* is the same as *Recall*, but for the followup session 24 hours later.

Participants and Apparatus

Participants were recruited from the United States using Amazon Mechanical Turk. They were informed that the task would take approximately 20 minutes and they would receive US\$2.50, which is approximately United States' minimum wage. We required participants to have 1000 approved HITs (tasks on Mechanical Turk) and a 95% approval rate. We granted qualifications to workers to ensure that they could only complete the experiment once. Participants were also paid a US\$1.00 bonus for completing the followup session. The pay rate for the followup session was higher than the main session to encourage participants to return.

The experiment was built as a web application using JavaScript and React. Participants used either a laptop or desktop to complete the experiment, and were limited to using either Chrome or Firefox to maximize compatibility.

Results

We recruited 118 participants. We used two criteria to filter out low-quality data: if a participant answered affirmatively to recording the menu visualization (3 participants), and if they answered 0 correct during the followup session (17 participants). The following analysis uses the data from the 98 participants remaining after filtering.

Participants were aged 20 to 61 (M=34.8, SD = 8.59); 61 were male, 36 were female, and 1 identified with another gender; 87 were right handed, and 11 were left handed. Participants reported that they used shortcuts moderately often on a range from 1 ("I never use shortcuts") to 5 ("I am an expert shortcut user") (M=3.07, SD = 1.08). Desktop users made up the majority of participants (56 used desktop and 42 laptops). 77 participants used a mouse, and 21 used a trackpad.

The main session was balanced with 49 participants in each technique condition. 28 participants from ExposeHK returned

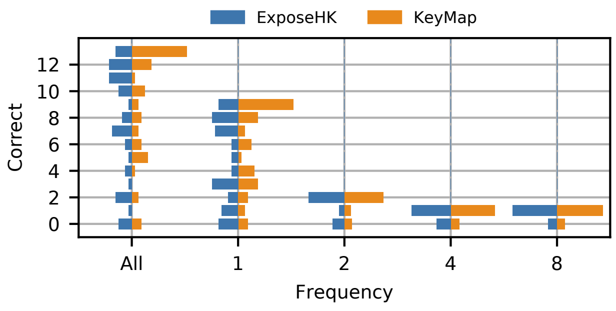


Figure 5. *Recall* scores in the main session, grouped by frequency. Overall, KeyMap participants recalled a median of 10 commands, and ExposeHK participants recalled 9.

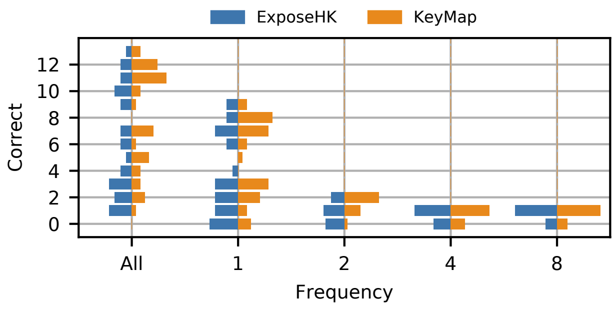


Figure 6. *Retention* scores in the followup session, grouped by frequency. Overall, KeyMap participants recalled a median of 10 commands, and ExposeHK participants recalled 5.5.

for the followup session (57%) and 37 participants returned for the KeyMap technique (76%). A test of proportions shows that this is nearly significant ($\chi^2 = 2.924$, $p = .087$, $\phi = .21$), which might indicate that KeyMap provided a more positive experience than ExposeHK.

H1: Shortcut Memory

Participants remembered more shortcuts with KeyMap than with ExposeHK. In the *recall* test (Figure 5), participants remembered more shortcuts using KeyMap (MEDIAN = 10) than ExposeHK (MEDIAN = 9). A Mann-Whitney test showed that this difference is significant: ($U = 354$, $p < 0.05$, $\eta^2 = 0.37$). 24 hours later, in the *retention* test (Figure 6), participants again remembered more shortcuts using KeyMap (MEDIAN = 10) than using ExposeHK (MEDIAN = 5.5): ($U = 903.5$, $p < 0.05$, $\eta^2 = 0.4$). In other words, KeyMap helped users memorize one more shortcut (7% of the command set) than ExposeHK, and KeyMap users tended to retain this memory advantage after 24 hours. We therefore accept H1: KeyMap helped users to memorize more shortcuts than ExposeHK.

H2: Incidental Learning

KeyMap users recalled more unpracticed commands than ExposeHK users. During the *recall* test, we showed 5 commands that did not appear in the selection tasks. In the KeyMap condition, participants entered a total of 19 of these commands correctly, which is a 7.8% correct selection rate from the 245 cued items (49 participants $\times$ 5 commands). In particular, 11 participants entered one command correctly, 1 participant entered two, and 2 participants entered three. In ExposeHK, 6 participants entered one correct command (2.4% correct). A test of proportions shows that there is a significant difference between the total numbers of correct commands ($\chi^2 = 6.07$, $p < .05$, $\phi = .25$). However, when comparing the number of participants making at least one correct selection, the effect is marginal ($\chi^2 = 3.08$, $p = .079$, $\phi = .18$).

To contextualise these results, suppose that a participant remembered that the shift key was the required modifier. Then, there are approximately 50 keys to choose from, resulting in a correct selection rate of 2%. However, with KeyMap, users might remember the location of the category on the keyboard, leading to a more informed guess. If users perfectly remembered the 7 keys associated with each category, a random guess within this group would result in 14% accuracy, almost twice as many as observed.

These results suggest that KeyMap users retained more memory associated with unpracticed items than ExposeHK users, supporting H2. However, correct selections were sufficiently rare to suggest that this memory was primarily related to the general area of the keyboard, rather than the actual key binding.

H3: Selection Time

Participants completed a total of 14,700 selection tasks. To analyze this data, we removed 183 outlier trials that were more than 3 standard deviations from the mean of their respective technique (80 with ExposeHK; 103 with KeyMap). The selection times were log transformed due to a significant deviation from normality. The following analyses use the Greenhouse-Geisser correction where sphericity tests fail, resulting in fractional values for degrees of freedom.

We found that KeyMap users improved their selection times with practice more than ExposeHK users (Figure 7). As expected, there was a significant main effect of BLOCK, with mean selection times decreasing from 5.4 s in Block 1 to 2.6 s in Block 6 ($F_{3.35,321.7} = 486.3$, $p < .001$, $\eta^2_G = .43$). There was no significant main effect of TECHNIQUE on selection time ($F_{1,96} = .968$, $p = .33$, $\eta^2_G = 0.008$), with similar overall means of 3480 ms with ExposeHK and 3281 ms with KeyMap. There was a significant BLOCK $\times$ TECHNIQUE interaction ($F_{3.35,321.7} = 5.78$, $p < .0001$, $\eta^2_G = .009$). This interaction can be attributed to the cross-over effect shown in Figure 7: KeyMap selections were marginally slower than ExposeHK in Block 1, but became and remained slightly faster than ExposeHK by Block 3. However, Holm-Bonferroni-corrected Tukey post-hoc tests revealed no significant TECHNIQUE differences in any BLOCK. Although these differences are not significant, this trend matches our suspicion that KeyMap would be slower than ExposeHK during initial selections (while users learn command groupings), but slightly faster once these are known (due to improved memorisation).

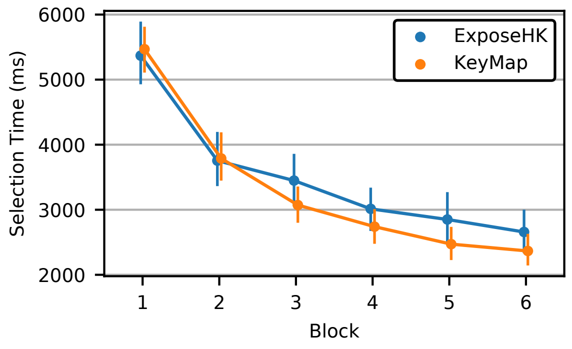


Figure 7. Mean selection time per block by technique. Error bars are 95% confidence intervals.

Shortcut Use and Error Rate

The proportion of commands selected using the keyboard was almost identical for both techniques. KeyMap participants selected 96.2% with the keyboard and ExposeHK 96.3%, and a Mann-Whitney test confirmed that this difference was not significant ($U = 1109$, $p = .34$, $\eta^2 = 0.004$). Mean error rates were also very similar between the two techniques, at 5.42% with KeyMap and 5.37% with ExposeHK ($U = 1200.5$, $p = 1$, $\eta^2 = 0$).

Subjective Ratings

Participants were asked whether they found the shortcut system useful or not. Only three KeyMap users did not find it useful (6%), compared to 10 with ExposeHK (20%). A chi-squared test found that this difference was nearly significant ($\chi^2 = 3.1928$, $p = .074$, $\phi = 0.18$), suggesting that participants might have a subjective preference for KeyMap.

After each session, participants were asked to estimate the number of shortcuts that they remembered as well as the number that they did not remember. These responses were similar for both interfaces, with identical medians of 6 remembered and 10 not remembered after the main session, and 5 remembered and 10 not remembered after the follow-up session.

DISCUSSION

Our experiment tested three hypotheses about the memorization and execution of shortcuts with KeyMap in comparison to ExposeHK, a state-of-the-art method for improving shortcut use. The results show that KeyMap increased shortcut vocabulary immediately after a training session and maintained this benefit 24 hours later. KeyMap also assisted with incidental learning of shortcuts that users had not previously invoked. Finally, selection times provide some support for our hypothesis that KeyMap would initially be slower than ExposeHK (due to the need to identify the locations of command groupings), but ultimately leads to better performance due to KeyMap's better support for shortcut memorization.

Why Did KeyMap Improve Shortcut Memory?

KeyMap's improvement in the number of shortcuts memorized is most likely due to participants' improved ability to leverage their spatial memory. Participants' comments support this inference. In the post-questionnaire, in response to the question "What features of the shortcut system were most helpful in memorizing shortcuts?", 14 KeyMap users referenced the ability to use "location". One participant noted "Being able to quickly see the shortcut when pressing shift in the location it really was on my keyboard was helpful", and another stated "having items group together in specific areas on the keyboard... made it easier to remember where things would be."

These comments lend support to the use of natural mapping to improve memory. In KeyMap, these benefits apply to both expert and novice users. On one hand, skilled users are familiar with a keyboard layout, so showing shortcuts in this layout gives them helpful cues for spatial memory. On the other, novice users find it useful to see each command at its position on the keyboard, making it easier for them to find and rehearse shortcuts.

We believe that other interfaces could achieve better memorization by adopting Norman's mapping. Norman's goal is to externalize knowledge into the world by representing it in a natural, intutitive way. In order for this mapping to be natural, the representation must be strongly related to the knowledge it represents, and this mapping will always create strong cue-target associations. It stands to reason that applying Norman's mapping should improve memorability when applied to other interfaces.

The incidental learning results suggest that KeyMap improved users' ability to remember the approximate location of a keyboard shortcut within a region of the keyboard. Importantly, this finding shows that KeyMap provides a mechanism to help users construct memories for shortcut command invocation methods, including those that they have not previously selected. While users are unlikely to issue shortcut commands until they are confident that their selections are correct, KeyMap's guidance allows them to quickly find the correct key by conducting a visual search in the correct area of its visualization. Additionally, while completing this search, they can prepare to make the correct selection by moving a finger to the corresponding region on the keyboard.

Deploying KeyMap

To show how KeyMap can be practically deployed, we implemented it as a Chrome extension[2]. We chose a Chrome extension because it is cross-platform, it integrates with a variety of web applications, and it is easy for users to try. As an initial demonstration, KeyMap supports native Chrome shortcuts, as well as commands for Slack and Overleaf.

While our implementation comes with an initial set of commands, the extension has a simple API for adding new shortcuts from JSON configuration files. This choice makes it simple to add shortcuts to support new websites. A more sophisticated system could allow users to easily install "plugins" with packs of commands. We hope that including a working implementation of KeyMap helps users become more productive.

[2]Download the extension: `https://chrome.google.com/webstore/detail/fpminkfnndfokkmoobbngdpnijbcajkl`

To make KeyMap useful in practice, it is important to group related commands together. In our experiment, we used coloured keys for this purpose, but this grouping may not be practical: it is difficult to distinguish a large number of colors, and this issue is worse for colour blind users. In real-world applications, designers might use several alternative grouping methods. One example is physically locating commands together, as with "cut", "copy" and "paste". KeyMap could even make these physical groupings explicit by adding separators around the groups. Other methods that still leverage the principles of KeyMap might be changing the line style surrounding a shortcut to create a group, or creating similar iconography within each key.

Larger Command Vocabularies

Like Malacria et al., our experiment tested a small set of 13 shortcuts, and we used shift as the only modifier. However, real applications typically have more shortcuts, using several modifier keys to achieve this larger vocabulary. Would KeyMap's benefits persist with these larger command sets?

As a first step in this direction, we tried scaling our experiment up to a larger set of 35 commands, each appearing once per block. We tested three conditions: KeyMap, ExposeHK, and a baseline condition using standard linear menus. In a between-subjects experiment with 118 participants, we found that participants using KeyMap remembered a median of 6 commands, more than ExposeHK (2 commands) or the baseline (3 commands). However, we were suspicious of participants cheating. In ExposeHK and the baseline, we noticed participants making many shortcut selections for commands they had never seen before. This command set was much more difficult to learn than the smaller Zipfian distribution from the main experiment, and we suspect that the increased difficulty drove participants to cheat. Future work investigating larger command sets should carefully control the difficulty of the experiment.

We believe that KeyMap's benefits would also be maintained with multiple modifier keys. There are several ways to support these. The simplest solution is to only display the shortcuts that can be triggered with the currently held modifier key, but this would require users to cycle through modifiers to search for shortcuts. This search process could be eased if each key in the visualization indicated when another modifier was available for that key. In general, we suspect that KeyMap's memory advantages would be maintained if multiple modifier keys were used, resulting in long-term performance advantages due to an improved vocabulary of shortcuts. However, we also suspect that initial performance would be worse than ExposeHK until users learned the command groupings. Further work is needed to validate this suspicion.

Limitations and Future Work

Overall, our results indicate potential for KeyMap and, more broadly, its use of natural mapping to improve memory. However, beyond of scaling to larger command sets, future work is required to understand if KeyMap would still increase shortcut adoption and improve memory in practical use.

Our experiment was conducted using a standalone interface and participants from Mechanical Turk. This was beneficial, as it helped to achieve the large sample sizes that are required when using a between-subjects treatment. While we believe this was an appropriate choice for a first empirical investigation, more work is required to understand how KeyMap can help other groups of users in a wider variety of settings. Methodological triangulation including lab and field studies will help broaden the ecological validity of the interface.

Our evaluation also treated KeyMap as a replacement of traditional linear menus. However, it could be used as an augmentation to existing interface layouts (menus or toolbars), so users could choose whether to invoke KeyMap or not. We suspect that users' uptake of shortcut mechanisms in this setting would be much lower than indicated in our experiment. Again, further study is required to examine this variation.

We used colour to group related commands in our experiment. This choice was necessary because ExposeHK uses menus to display the categories, but KeyMap does not. Adding colour allowed us to make a meaningful comparison by looking at the relative differences in selection times. However, colouring the interfaces might have decreased visual search time for both menus, making it difficult to understand exactly how quickly users could select commands using KeyMap. For future work to measure this absolute selection time, it is important to study KeyMap's effect in a real task setting.

KeyMap also opens avenues for future interfaces. For example, KeyMap could be naturally extended for feedback by flashing the keys used in a shortcut after invoking the command, even if the command was invoked using a linear method. The ideas behind KeyMap might also work beyond 2D interfaces. Some menus in virtual reality interfaces are arranged in the shape of the controller; it would be interesting to see whether the benefits of Norman's natural mapping persist when the display is only a 2D projection of the input device.

CONCLUSION

Keyboard shortcuts allow fast command execution, but they are seldom used because most users have a small shortcut vocabulary. We designed and evaluated a system called KeyMap to improve support for shortcut memorization and use. KeyMap achieves these benefits in a unique way: it leverages Norman's principle of *natural mapping*, which makes shortcuts more memorable by giving users more cues for spatial memory. We evaluated KeyMap in comparison to ExposeHK, a state-of-the-art interaction technique, and found that KeyMap users memorized more commands and showed more incidental learning without sacrificing selection time. Designers can incorporate natural mapping into their interfaces to help their users become better users.

ACKNOWLEDGEMENTS

We would like to thank to Johann Wentzel for helping with the video figure. This work was made possible by NSERC Discovery Grant 2018-05187, the Canada Foundation for Innovation Infrastructure Fund 33151 "Facility for Fully Interactive Physio-digital Spaces," and Ontario Early Researcher Award ER16-12-184.

REFERENCES

[1] Jonathan Aceituno and Nicolas Roussel. 2014. The Hotkey Palette: Flexible Contextual Retrieval of Chosen Documents and Windows. In *Proceedings of the 26th Conference on L'Interaction Homme-Machine (IHM '14)*. ACM, New York, NY, USA, 55–59. DOI: http://dx.doi.org/10.1145/2670444.2670452

[2] Jason Alexander and Andy Cockburn. 2008. An Empirical Characterisation of Electronic Document Navigation. In *Proceedings of Graphics Interface 2008 (GI '08)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 123–130. http://dl.acm.org/citation.cfm?id=1375714.1375736

[3] Fraser Anderson and Walter F. Bischof. 2013. Learning and Performance with Gesture Guides. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1109–1118. DOI: http://dx.doi.org/10.1145/2470654.2466143

[4] Alan D. Baddeley, Michael W. Eysenck, and Mike Anderson. 2009. *Memory*. Psychology Press.

[5] John M. Carroll and Mary Beth Rosson. 1987. Interfacing Thought: Cognitive Aspects of Human-computer Interaction. MIT Press, Cambridge, MA, USA, Chapter Paradox of the Active User, 80–111. http://dl.acm.org/citation.cfm?id=28446.28451

[6] Leah Findlater and Joanna McGrenere. 2004. A Comparison of Static, Adaptive, and Adaptable Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 89–96. DOI: http://dx.doi.org/10.1145/985692.985704

[7] Paul M. Fitts and Charles M. Seeger. 1953. S-R compatibility: spatial characteristics of stimulus and response codes. *Journal of Experimental Psychology* 46, 3 (1953), 199–210.

[8] Emmanouil Giannisakis, Gilles Bailly, Sylvain Malacria, and Fanny Chevalier. 2017. IconHK: Using Toolbar Button Icons to Communicate Keyboard Shortcuts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4715–4726. DOI: http://dx.doi.org/10.1145/3025453.3025595

[9] Wayne D. Gray and John K. Lindstedt. 2017. Plateaus, Dips, and Leaps: Where to Look for Inventions and Discoveries During Skilled Performance. *Cognitive Science* 41, 7 (2017), 1838–1870.

[10] Tovi Grossman, Pierre Dragicevic, and Ravin Balakrishnan. 2007. Strategies for Accelerating On-line Learning of Hotkeys. In *Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems (CHI '07). ACM, New York, NY, USA, 1591–1600. DOI:http://dx.doi.org/10.1145/1240624.1240865

[11] Brian Krisler and Richard Alterman. 2008. Training Towards Mastery: Overcoming the Active User Paradox. In *Proceedings of the 5th Nordic Conference on Human-computer Interaction: Building Bridges (NordiCHI '08)*. ACM, New York, NY, USA, 239–248. DOI:http://dx.doi.org/10.1145/1463160.1463186

[12] David M. Lane, H. Albert Napier, S. Camille Peres, and Aniko Sandor. 2005. Hidden Costs of Graphical User Interfaces: Failure to Make the Transition from Menus and Icon Toolbars to Keyboard Shortcuts. *International Journal of Human Computer Interaction* 18, 2 (2005), 133–144. DOI: http://dx.doi.org/10.1207/s15327590ijhc1802_1

[13] Sylvain Malacria, Gilles Bailly, Joel Harrison, Andy Cockburn, and Carl Gutwin. 2013a. Promoting Hotkey Use Through Rehearsal with ExposeHK. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 573–582. DOI: http://dx.doi.org/10.1145/2470654.2470735

[14] Sylvain Malacria, Joey Scarr, Andy Cockburn, Carl Gutwin, and Tovi Grossman. 2013b. Skillometers: Reflective Widgets That Motivate and Help Users to Improve Performance. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 321–330. DOI: http://dx.doi.org/10.1145/2501988.2501996

[15] Hugh McLoone, Ken Hinckley, and Ed Cutrell. 2003. Bimanual Interaction on the Microsoft Office Keyboard. (September 2003), 49–56. https://www.microsoft.com/en-us/research/publication/bimanual-interaction-on-the-microsoft-office-keyboard/

[16] Donald A. Norman. 2002. *The Design of Everyday Things*. Basic Books, Inc., New York, NY, USA.

[17] Daniel L. Odell, Richard C. Davis, Andrew Smith, and Paul K. Wright. 2004. Toolglasses, Marking Menus, and Hotkeys: A Comparison of One and Two-handed Command Selection Techniques. In *Proceedings of Graphics Interface 2004 (GI '04)*. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 17–24. http://dl.acm.org/citation.cfm?id=1006058.1006061

[18] Joey Scarr, Andy Cockburn, and Carl Gutwin. 2013. Supporting and Exploiting Spatial Memory in User Interfaces. *Foundations and Trends in Human-Computer Interaction* 6, 1 (2013), 1–84. DOI: http://dx.doi.org/10.1561/1100000046

[19] Joey Scarr, Andy Cockburn, Carl Gutwin, and Philip Quinn. 2011. Dips and Ceilings: Understanding and Supporting Transitions to Expertise in User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2741–2750. DOI: http://dx.doi.org/10.1145/1978942.1979348

[20] Richard M. Shiffrin and Walter Schneider. 1977. Controlled and automatic human information processing: II. Perceptual learning, automatic attending and a general theory. *Psychological Review* 84, 2 (1977), 127–190.

[21] H A Simon. 1956. Rational choice and the structure of the environment. (1956). DOI: http://dx.doi.org/10.1037/h0042769

[22] S. Tak, P. Westendorp, and I. van Rooij. 2013. Satisficing and the Use of Keyboard Shortcuts: Being Good Enough Is Enough? *Interacting with Computers* 25, 5 (Sept 2013), 404–416. DOI: http://dx.doi.org/10.1093/iwc/iwt016

[23] Mike Tucker and Rob Ellis. 1998. On the Relations Between Seen Objects and Components of Potential Actions. *Journal of Experimental Psychology: Human Perception and Performance* 24, 3 (1998), 830–846.

[24] Md. Sami Uddin, Carl Gutwin, and Andy Cockburn. 2017. The Effects of Artificial Landmarks on Learning and Performance in Spatial-Memory Interfaces. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3843–3855. DOI: http://dx.doi.org/10.1145/3025453.3025497

[25] Jordana S. Wynn, Michael B. Bone, Michelle C. Dragan, Kari L. Hoffman, Bradley R. Buchsbaum, and Jennifer D. Ryan. 2016. Selective scanpath repetition during memory-guided visual search. *Visual cognition* 24, 1 (02 Jan 2016), 15–37. DOI: http://dx.doi.org/10.1080/13506285.2016.1175531 27570471[pmid].