

Dips and Ceilings: Understanding and Supporting Transitions to Expertise in User Interfaces

Joey Scarr¹, Andy Cockburn¹, Carl Gutwin², Philip Quinn¹

¹Computer Science, University of Canterbury, Christchurch, New Zealand,

{joey, andy}@cosc.canterbury.ac.nz, philip.quinn@canterbury.ac.nz

²Computer Science, University of Saskatchewan, Saskatoon, Canada, gutwin@cs.usask.ca

ABSTRACT

Interface guidelines encourage designers to include shortcut mechanisms that enable high levels of expert performance, but prior research has demonstrated that few users switch to using them. To help understand how interfaces can better support a transition to expert performance we develop a framework of the interface and human factors influencing expertise development. We then present a system called *Blur* that addresses three main problems in promoting the transition: prompting an initial switch to expert techniques, minimising the performance dip arising from the switch, and enabling a high performance ceiling. *Blur* observes the user's interaction with unaltered desktop applications and uses *calm notification* to support learning and promote awareness of an alternative *hot command* interface. An empirical study validates *Blur*'s design, showing that users make an early and sustained switch to hot commands, and that doing so improves their performance and satisfaction.

Author Keywords

Expertise, novice-to-expert transition, command interfaces.

ACM Classification Keywords

H5.2. [User Interfaces]: Interaction Styles.

General Terms

Human Factors, Experimentation.

INTRODUCTION

Windows, Icons, Menus, Pointer (WIMP) interfaces mediate most communication between humans and computers. Their success is partly due to their natural support for novice users – the phrase ‘see and point versus learn and remember’ [38] describes how novices benefit from using visual search for salient controls, rather than retrieving command names from memory or manuals.

However, the very mechanisms that make WIMPs effective for novices fail to support users as they become more experienced, and they can trap users in a ‘beginner mode’ of interaction that has a low performance ceiling. The

richness and power of human perception, cognition, and motor action is then constrained to relatively slow and laborious action. Conversely, interfaces designed for experts (e.g., keyboard shortcuts or command-lines) allow high performance ceilings, but only after extensive training.

Although it is clear that these expert interfaces *can* provide performance advantages [34], their success in practice has been limited, and several researchers have reported that users fail to switch to expert interface methods (e.g., [3, 7, 27]). Furthermore, while there has been considerable research into interfaces for either novices or experts, there has been relatively little on the transition to expertise.

We therefore form a framework encapsulating the factors influencing expertise development, with a focus on those affecting the switch to expert interface mechanisms. These factors include lack of knowledge about the availability or the performance benefits of the alternative UI; concern about the time or effort required to make the switch; the prevalence of satisficing [40], in which ‘good enough’ strategies are maintained; and fears about the drop in performance that can occur because the user must ‘start from scratch’ with the new interface. This drop in performance (called ‘the dip’) is particularly important, because it can deter switching in the first place, but also because it negatively affects the user's first impressions. The framework suggests that any system attempting to support a switch to expert interface mechanisms should have three goals: first, maximize the likelihood that the user will initiate a switch to the expert modality; second, minimize the cost of doing so; and third, enable a high performance ceiling to rapidly reward use.

To investigate supporting these goals, we have developed a new system (called *Blur*) that uses *calm notification* and *hot commands* to support a transition from WIMP interaction to a more efficient command-based interface. Through calm notifications, *Blur* provides an immediate and bidirectional translation between WIMP and command line (CLI) methods – WIMP inputs are immediately displayed as equivalent CLI outputs, promoting learning and awareness of the CLI. Through the hot commands mechanism, *Blur* enables CLI control of the interface without altering the underlying GUI, thereby supporting gradual exploration of the CLI without requiring users to completely abandon use of familiar WIMP interaction.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHI 2011, May 7–12, 2011, Vancouver, Canada.

Copyright 2011 ACM 978-1-4503-0267-8/11/05...\$10.00.

We carried out a study using repetitive tasks in PowerPoint to see how well Blur supports users' transition to the expert CLI. Results showed that Blur supported users' transition to the expert interface much better than standard shortcut keys – all 20 participants switched to Blur within the first three trials, whereas none switched to built-in Alt-shortcuts. The study also showed that Blur's performance dip was small and that its ultimate efficiency was faster than the WIMP and shortcut interfaces.

Blur demonstrates a general and reproducible strategy for supporting users' transition from a WIMP interface to a CLI-style interface. Blur provides both a mechanism for learning and promoting the expert interface while the user carries out WIMP actions, and a means for allowing gradual exploration and adoption of the higher-performance CLI interaction. Specifically, we make three contributions on expertise development:

- A framework of issues influencing expertise development within and across interface mechanisms.
- Design of Blur, which promotes the transition to expert interaction through calm notification and hot commands.
- Empirical evidence demonstrating Blur's success.

FRAMEWORK & REVIEW OF INTERFACE EXPERTISE

Learnability is consistently identified as a critical component of usability [13, 31, 37], but as Grossman *et al.* [18] observe, there is little agreement over how it can be defined or measured. Grossman *et al.* [18] provide a comprehensive review of definitions, metrics, and methodologies for assessing learnability, and they suggest a new protocol for measuring it. However, evaluation methodologies are applied after an interface has been designed, and current best-practice guidelines for supporting expertise are often high level generalisations, such as 'provide shortcuts' [31] that provide little direct guidance or insight into underlying design issues.

This section presents a framework of the interface and human factors influencing expertise development. Our focus is on issues of how interfaces can support users' transition to expertise, rather than on fundamental issues in human skill acquisition and strategic thinking; see [1] for a review of human skill acquisition and [3] for an analysis of the development of expert interaction strategies.

Intramodal and Intermodal expertise development

The framework adapts Newell and Rosenbloom's [30] power law of practice, using it as a qualitative guideline for characterisation, rather than a mathematical model. We use the power curve shown on the left side of Figure 1 to characterise *intramodal* expertise development: how user performance improves over time with a *single* interaction modality, subdividing the curve into three segments for *initial performance*, *extended learnability*, and *ultimate performance*. These three stages are suggestive of Anderson's [1] model of skill acquisition, which identifies *cognitive*, *associative*, and *autonomous* stages in which initial models are formed, followed by establishment of

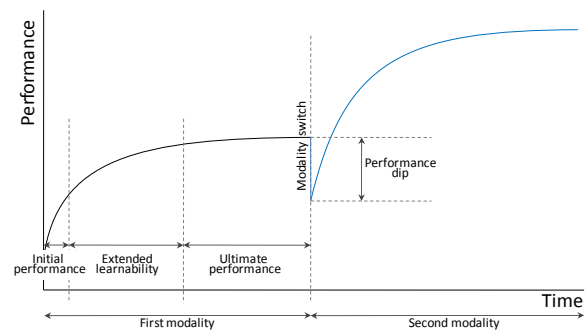


Figure 1. Performance curves characterising intra and intermodal performance improvement. Note the postulated performance dip when switching modalities.

associations between concepts, finally leading to the development of autonomous skills.

Many interfaces, however, support more than one interaction mode for the same task. For our purposes modes are distinguished by the interaction mechanics used to control the interface. For example, menu items can be selected by direct clicking with the mouse, or by activating their associated hotkeys using the keyboard; similarly, files and file hierarchies can be manipulated via direct manipulation (e.g. using Windows Explorer or the Desktop) or through command line alternatives (such as the file path entry or a command window).

Our framework characterises *intermodal* expertise development by combining two power law curves (shown on the left and right sides of Figure 1). This characterisation postulates that users are likely to suffer a performance dip when switching to a new modality, even if it offers a higher ultimate performance ceiling. For example, a user who frequently uses the 'bulleted lists' toolbar item may decide to learn the keyboard shortcut, spending time to determine, memorise, and make autonomous its key sequence.

While the curve characterises the user's *actual* performance when switching modalities, the user's *perception* of future performance critically influences whether the modality switch is made – if users perceive that an interface will be hard to learn, temporarily slow, or ultimately inefficient they will ignore it, and thus never attain the high performance ceiling it actually enables.

Figure 2 summarises the properties of the intra and intermodal performance curves, and the interface and human factors affecting them, which are described in the following sections. The right hand side of Figure 2 also shows the techniques that our system, called Blur, uses to influence expertise development. We describe Blur and its relationship to the framework later in the paper.

Review of intramodal expertise development

Initial performance (intercept)

Interface design for the initial stages of learning is strongly promoted in most usability guidelines [13, 31, 32, 37]. At this stage, users are unfamiliar with the interface, and must

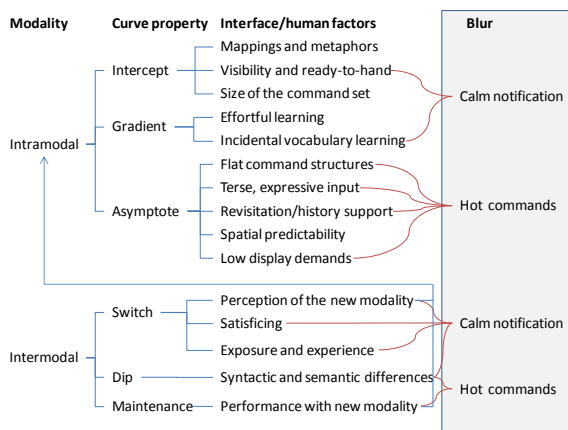


Figure 2. Summary facets of intra and intermodal expertise development, and the human and interface factors affecting them. Blur’s mapping to the framework is described later.

rely on their prior experiences, visual search, and recognition to find the commands they need.

Mappings and metaphors [31, 33] promote initial interface familiarity, but supporting such mappings in command languages is difficult because of the variability in words used to describe actions [26]. Furnas *et al.* [15] describe how this vocabulary problem can be reduced through aliasing. In hotkey assignment creating meaningful mappings is further complicated by the limited expressive capacity of the input language: once ‘P’ is assigned to ‘Print’, for example, ‘Paste’, ‘Previous’, etc., must be assigned additional modifier keys or less symbolic letters.

Visibility and ‘ready to hand’. The notion that controls should be visible to be learned is also well expressed in most usability guidelines, but the corollary of making novice things visible is that expert things are often suppressed, reducing the likelihood that they will be discovered. A related concept is that appropriate interface controls should be ‘ready to hand’ [23], where controls and feedback are available for use but not obstructing task completion. Dyck *et al.* [14] observe that many computer games achieve the dual objectives of availability without obstruction through ‘*calm messaging*’, using transient text, animation, and audio.

Size of the command set. Large command vocabularies are likely to take longer to learn than small ones. Carroll and Carrithers [6] exploited this effect with their ‘Training Wheels’ interfaces, which intentionally reduced the vocabulary size to aid learning. Related ideas were recently pursued with ‘multi-layer interfaces’ [39].

Extended learning (gradient)

Several factors influence the rate at which performance increases after initial familiarisation. Our review focuses on interface techniques that improve the efficacy of *recall* and on assisting users in establishing a good vocabulary.

Effortful learning. Psychology researchers have proposed that “deeper” cognitions, which take longer to process,

result in stronger memories [10, 35], and similar effects have been demonstrated with interfaces [8, 17]. However, explicitly manipulating the effort of interaction is risky as users are prone to frustration when training is too difficult and boredom when activities are mundane.

Incidental learning to extend vocabulary. Psychology literature also suggests that users should learn interface components as a side effect of their display while using other components. Shelton [36] showed that subjects’ memory in a paired-associate learning task was improved simply by prior exposure to the stimuli. Jones [22] therefore hypothesised that hypertext browsing would result in greater incidental learning than indexed search (due to exposure to content), but experiments failed to find significant differences.

Ultimate performance (asymptote)

The final characteristic of the intramodal curve is the asymptote, or performance ceiling. There is extensive literature on supporting and understanding expert interface performance (although it is largely independent of the processes enabling its attainment). In particular, the seminal work of Card, Moran and Newell [5] provides strong predictive models and empirical evidence of ‘expert performance of routine tasks’, including analysis of one user who repeated the same editing task thousands of times to study progression to automaticity. Five interface characteristics for high performance ceilings follow:

Flat command structures. GUIs typically contain more controls than can be easily displayed at once, necessitating interface partitions such as windows, tabs, and menu hierarchies. Navigating through these partitions takes time, and consequently there are potential performance benefits in flattening the command structure to make more items accessible at once. Commands issued by CLIs and hotkeys are exemplars as they have global interface scope (e.g. <Ctrl>-C executes ‘copy’ regardless of the interface state). Several research and commercial systems have used CLIs to improve interface performance: e.g., Quicksilver¹, Spotlight², Enso³, and GEKA [21]. Although empirical results for CLI benefits over GUIs have been mixed (e.g. [42]), it is widely accepted that CLIs enable higher efficiency, and power users are strong advocates (e.g. [2]).

Terse and expressive. Powerful interfaces communicate a lot of meaning in rapidly expressed actions. For example, a single alphabetic character can discriminate 26 commands, or 52 with case sensitivity; increasing to 2704 with two case-sensitive characters. However, there is often a tension between supporting terse, expressive power and meaningful mappings: for example, Alt-shortcuts in Office 2007 allow access to most controls, but they are abstract and hard to remember (e.g. ‘<Alt> n, nu, t’ inserts a page number).

¹ <http://www.blacktree.com>

² <http://support.apple.com/kb/HT2531>

³ <http://humanized.com>

Revisitation/history support. Users' interactive behaviour is often repetitive (e.g., command use [16] and web navigation [41]), and interfaces can aid efficiency by explicitly supporting repetition. For example, web browser URL address bars and the Google search box memorise previous activities and offer type-ahead shortcuts for them: e.g., the keystrokes 'cn↓<Ret>' become a shortcut for a user who frequently visits CNN's website.

Spatial predictability. Studies have demonstrated that spatial stability allows users to make rapid decisions about items' locations rather than relying on comparatively slow visual search (e.g. [19]). Despite the desirability of spatial stability it is often compromised due to display space constraints – interface controls are often elided and repositioned as window geometry is manipulated, and this is necessary because widgets typically do not scale. There are interesting design opportunities in spatially stable interfaces that dynamically scale widgets.

Low display demands. A final property of high performance interfaces is that they help the user focus on their primary activity, which typically involves their data. WIMPs rely on visual presentation, which consumes screen real-estate that might otherwise be used for data. This is critical on small devices, such as Netbook computers. For example, the window border, Ribbon, ruler and foot controls in Office 2007 applications consume approximately 195 vertical pixels, and the default Windows 7 Taskbar consumes another 30, for a sum of 225 pixels, which is 38% of a 600 pixel Netbook.

Review of intermodal expertise development

Compared to the extensive literature on intramodal expertise development, there has been much less on the factors influencing whether, how, and when users switch from novice to expert mechanisms. Note, our analysis does not review end-user programming (e.g., [11]) or interface customisation (see [4] for a good summary), which raise their own challenges (reviewed in [24] and [28]); although some of the factors identified below are applicable.

In the following analysis, we address three critical points on the *intermodal* performance curve shown in Figure 1: first, factors influencing the initial switch to a new interface modality; second, the performance dip that a user is likely to experience when switching from a familiar interface to an unfamiliar one; and third, factors influencing the maintenance of the new modality.

Making an initial switch

Perception of the new modality. As Figure 2 shows, the probability of switching to a new modality is likely to be influenced by how the user perceives any future interaction with the new modality, so all of the intramodal factors described above play a role. Importantly, though, several studies have demonstrated that perceived experience differs from actual (e.g. [12]), and that users can mistrust their abilities, leading to false assumptions of poor performance

(e.g., [9] showed users predicted poor performance in a spatial task, but performed well).

Satisficing and optimising for immediate needs. The notion that users have a tendency to maintain existing strategies and use what is known and ready to hand in preference to new and improved ways of working is encapsulated by several theories, including Simon's 'satisficing' [40] and 'Maslow's hammer' [29] ('to a man with a hammer, everything looks like a nail'). In HCI, Carroll and Rossen [7] named the effect 'the paradox of the active user', in which users "are likely to stick with the procedures they already know, regardless of their efficacy".

Exposure and experience. Incidental learning, as described with the intramodal factors, can also be used to promote learning across modalities by exposing users to alternative ways of achieving their tasks as a side effect of their interactions. More forcefully, interfaces can demand that users experience the new modality by requiring that actions are completed through it.

Grossman [17] experimented with a variety of schemes for assisting hotkey learning. These included visual and audio schemes to expose users to the hotkeys, a delay-based technique to deter use of the GUI (i.e., making the system unresponsive for 2 seconds after each selection), and a technique that forced hotkey use following each menu selection. Their results showed that forced use and audio feedback worked well, with 72.8% and 66.6% of experimental selections being made with hotkeys. Subjective data showed no significant adverse response to the audio and forced use.

One concern for hotkey strategies, though, is that Grossman's results suggest that users may only be able to learn a small hotkey vocabulary. In their experiment 73% of selections could be completed using six hotkeys, and 83% with eight, yet the participants' mean use was less than 73%. Current desktop applications support hundreds of commands, and it is unclear how well audio feedback or enforced use can work in practice.

Performance dip after switching

Semantic and syntactic differences. The size of the performance dip that occurs after switching to a new interface modality will be influenced by the magnitude of the semantic and syntactic differences between the pre- and post-switch interfaces.

Interface semantics determine the interface and data states that can be attained with the interface. Frequently, these states differ across modalities, with one modality supporting a subset of the other. For example, it is common for only a subset of controls to be accessible via hotkeys. Semantic differences are likely to discourage users from investigating secondary modalities as the effort invested in seeking new facilities may go unrewarded (when it supports a subset) or require formation of a new model (when it supports a superset).

Interface syntax is determined by the mechanics of control and the manner in which control elements are combined. Marking menus [25] are an excellent example of promoting expertise by minimising the syntactic differences between novice and expert interaction modes. Their commands are arranged radially around the cursor, like segments of a pie. Novices attend the visual feedback and learn the drag-release movement directions for specific selections, such as ‘print is East’. Experts, however, can use precisely the same interface syntax (a rapid directional drag and release) to select items without need for visual feedback.

Maintenance of the new modality

Whether users continue to use a new modality after making an initial switch depends on the magnitude of the performance dip they encounter and on their perception of their future performance with the modality, including how quickly they expect to outperform the original modality, and their estimation of their ultimate performance ceiling. These issues primarily depend on intramodal issues, described earlier, as shown in Figure 2.

BLUR: USAGE, RATIONALE, AND IMPLEMENTATION

The framework highlights three main challenges that Blur aims to address in assisting users to make a transition to expert interaction: *promote an initial switch, minimise the dip in performance, and enable a high performance ceiling*. Figure 2 shows how Blur’s two main features of *calm notification* and *hot commands* (described below) are designed to map onto these components of the framework.

Overview of the user experience of Blur

Blur observes and controls interactions with unaltered desktop applications on Microsoft Windows platforms.

Blur’s calm notification feedback during WIMP use. Figure 3 shows Blur’s normal display state, with only a small translucent tab at the top of the screen displaying the text ‘Press <Esc>’. When the user carries out an interface action by clicking on an interface control using the mouse, Blur provides *calm notification* of an alternative syntax for achieving the same action – the translucent tab expands




Figure 3. Blur’s default state: a tab at the screen top.



Figure 4. Blur’s calm notification: command name associated with a GUI action.



Figure 5. Blur’s hot command suggestions.

displaying a command name. Figure 4 shows Blur’s feedback (‘Align Left’) after clicking the  button in Microsoft Word. After one second, the transparent window gradually contracts back to its tab state. Blur’s window can be clicked through, allowing continued manipulation of the underlying GUI while Blur is visible.

Controlling interfaces with Blur’s hot commands. Blur allows users to control the focal application, launch new applications, and manipulate windows using typed CLI *hot commands*. The user presses the Escape key to display Blur’s translucent window, and command recommendations are shown in response to each successive typed letter. Figure 5 shows Blur’s recommendations after typing ‘<Esc>al’ when Microsoft Word is the focal application: the ‘align left’ command is recommended first and can be selected by pressing the Return key, but the arrow keys can be used to move through the recommendations (e.g. ‘calc’).

Design rationale

Calm notification

Blur’s calm notification provides transitory feedback revealing the command name that is equivalent to each mouse initiated action. Calm notification is primarily intended to reduce the tendency to satisfice and to promote an initial intermodal switch to Blur’s *hot commands*. However as Figure 2 shows we also intend that it will provide a visually salient and continual reminder of the availability of the hot command alternative, as well as supporting incidental learning of the hot commands.

Importantly, we also intend that calm notification will help users identify that there is a one-to-one correspondence between each WIMP action and each *hot command*. This means that the user’s mental model of interaction is largely unaffected by the transition between interface modalities. In other words, Blur’s hot command interface does not change the structural decomposition of tasks into interface actions, and calm notification is the mechanism to communicate this absence of change to the user. This consistency is intended to minimise the performance dip associated with switching to Blur’s hot commands.

Hot commands

As Figure 2 shows, Blur’s hot commands are primarily designed to support a high performance asymptote, but as mentioned above they are also designed to provide a one-to-one mapping to WIMP commands to minimise the performance dip. This one-to-one relationship is unusual for CLIs, which normally require a different style of working typified by ‘action-object’ syntax, where data is identified through parameters following the command.

Blur’s hot commands are symbolic (provided the original application designer has assigned meaningful names to controls), which should aid learning through appropriate mappings and metaphors. They support a flat command structure, allowing interface controls to be accessed with a single command, and eliminating the need to navigate

through tab or menu hierarchies. They are terse and expressive, allowing unambiguous access to thousands of controls in a few keypresses. Hot commands also adapt to the user's interaction history by ordering command recommendations by frequency of use. Finally, Blur's interface consumes very little screen space.

Implementation⁴

Blur uses Microsoft's UI Automation API⁵ to discover the GUI control elements that users interact with. All applications implementing this interface can therefore be observed and controlled by Blur. When the user clicks an interface widget Blur intercepts the event to determine the control under the cursor, retrieving the command name and the shortcut key sequence that can be used to activate the control. The action can then be performed by typing the command into Blur, which automatically translates the command into corresponding keyboard shortcuts and sends them to the application, controlling the GUI.

Blur orders command recommendations according to three candidate classes: exact matches, prefix matches and substring matches. Within each class, commands are sorted by frequency of use, and are displayed to the user. Typo correction is provided by remembering the most likely candidate at each keypress. If, at any point, the user's input matches an empty set of candidates, the most likely candidate from the previous keypress is suggested.

Blur's command recommendations are also context sensitive to the focal application. Consequently, the characters "se" may match "Send" while using an email system and "select all" when using a word processor. Context sensitivity has several advantages including fewer matching commands for any typed string, and reduced probability of illegal commands (e.g., "Send" is illegal unless an email composition window is open). Blur also supports 'global' interface controls such as window management and application launching. For example, window focus can be manipulated by typing any substring of the window title, (e.g., "in" for "email inbox"). Global commands are not context sensitive.

Summary of Blur

Blur is primarily designed to encourage an early and sustained switch to a command line interface. Blur's main mechanisms for supporting this are *calm notification* and *hot commands*, and importantly the hot command interface provides a one-to-one mapping to the WIMP interface. Although many previous research and commercial systems have supported CLIs (e.g., [21]), none have explicitly investigated the mechanisms used to initiate the modality switch. Grossman *et al.* [17] did investigate transitional mechanisms, but the end modality was hotkeys, and their results suggest that hotkey vocabularies may be limited.

EXPERIMENT: BLUR'S IMPACT ON USERS

We conducted an experiment to answer four key questions about Blur's performance, focusing on the point of modality transition rather than long term maintenance.

- Q1. Do users switch to Blur's expert modality?
- Q2. How does performance with Blur compare to performance without it and with other methods?
- Q3. How large is the performance dip with Blur?
- Q4. What is the subjective response to Blur?

The experiment involved a repetitive series of tasks using an unaltered version of Microsoft PowerPoint 2007, with and without Blur running. Participants also completed the tasks using the Office 2007 <Alt> shortcuts, which allows comparison between Blur and existing shortcut facilities. Repetitive tasks were used to compress long term interaction experiences into the short duration of a lab study, similar to Grossman's [17] study of hotkey learning.

Participants and Apparatus

The 20 participants aged 21 to 36 (mean 25, s.d. 4.4) were recruited from a local university. They reported using computers for a mean of 46.7 (s.d. 18.1) hours per week.

The experiment ran on a Windows 7 computer with a 1680×1050, 21" display. The target state for each task was shown on a sheet of paper placed alongside the computer.

Tasks

Five different PowerPoint slides were created, each containing five drawn objects of varying sizes and shading as shown in Table 1. The target state was shown on a sheet of paper at the side of the keyboard. All target states could be attained in a minimum of four commands, such as 'Select All', 'Align Left', 'Group', 'Flip Vertical'. Participants were instructed to complete tasks as quickly and accurately as possible. Automatic logs recorded all user interactions, including task time data.

Procedure

At the start of the experiment participants were given a two minute introduction to Blur using a PowerPoint slide containing a single text field. They were instructed to make the text Bold, Italic, Bulleted and Numbered using the GUI and to observe Blur's feedback, and then repeat the same actions by typing Blur commands.

The experiment proceeded through five stages, with each stage using one of Tasks 1-5 shown in Table 1, as follows:

1. familiarisation and training with PowerPoint
2. optional use of Blur (*blur*)
3. normal PowerPoint (*wimp*)
4. instruction to maximise use of Blur (*blur-max*)
5. instruction to use <Alt> shortcuts (*alt*)

Participants repeated the manipulations to move from the initial to target states five times with Task 1 and twelve times each in Tasks 2-5. Participants completed NASA-TLX worksheets [20] and gave comments after Tasks 2-5.

⁴ Blur is available at: www.cosc.canterbury.ac.nz/blur

⁵ <http://msdn.microsoft.com/en-us/library/ms747327.aspx>

Task 1 was used to familiarise participants with PowerPoint’s alignment, rotation, grouping and flipping controls, with instruction as necessary. Participants then repeatedly moved from initial to target states five times.

Tasks 2 and 3 were completed with the WIMP and with Blur (counterbalanced). Before using Blur, participants were instructed to “complete the tasks as you please.”

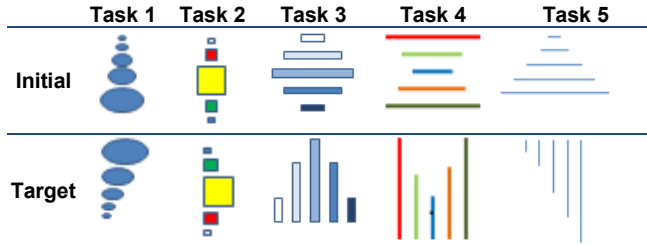


Table 1: Initial and target states for the five tasks.

Task 4 was completed following explicit instruction to “use Blur’s facilities as much as possible”. This condition was included because we wanted to be certain to obtain measurements of Blur’s performance. We could not rely on Tasks 2/3 to produce this data because participants had the option to ignore Blur entirely.

Task 5 was used to analyse user performance with Microsoft Office’s built in <Alt> shortcut navigation controls. This task was included because it was possible that the existing shortcut facilities would outperform Blur. Prior to completing these tasks participants received two minutes instruction on <Alt> shortcuts and practiced with the same text tasks used for familiarisation with Blur.

Results

Q1. Do users switch to Blur’s expert modality?

In Task 2/3, where participants were instructed to complete tasks however they liked, all chose to use Blur. The mean trial number at which participants switched to Blur was 1.65 (s.d. 1.0), with 55% of participants using Blur in the first trial, increasing to 90% in the second, 95% in the third, and 100% by the fifth. Two of the participants briefly returned to using the WIMP after their first use of Blur, but switched back and continued using Blur through the final trials (we suspect that these participants were confirming that Blur was faster than the WIMP). In contrast to the successful switch to Blur, none of the participants tried to use <Alt> shortcuts. Several used prior knowledge of Ctrl-a for ‘select all’, and nearly half used Ctrl-g for ‘group’, but many expressed surprise similar to the statement “there’s no shortcut for group” after visually inspecting the menu (there is, but it’s not shown in the UI).

Q2. How does performance with Blur compare to performance without it and with other methods?

Trial times in Tasks 2-5 were analysed in a 4×3 repeated measures ANOVA for within-subjects factors *interface* (*wimp*, *blur*, *blur-max*, and *alt*) and *block* (repetitions 1-3, 4-7, and 8-11, with trial 0 discarded as preparation).

Results show a significant effect of *interface* ($F_{3,57}=97.9$, $p<.001$), with means of 9.72, 9.92, 7.54, and 19.97 seconds for *wimp*, *blur*, *blur-max* and *alt* respectively (see Figure 6). There is also a significant effect of *block* ($F_{2,38}=77.9$, $p<.001$), with participants’ mean performance improving from 15.2 seconds in the first three trials to 10.7 seconds in trials 4-7 and 9.5 seconds in trials 8-11. Finally, as Figure 6 suggests, there is a significant *interface* × *block* interaction ($F_{6,114}=15.3$, $p<.001$), which is best explained by the relatively small cross block performance improvement with *blur-max* (due to prior experience to Blur) contrasting with the large improvement with *alt*.

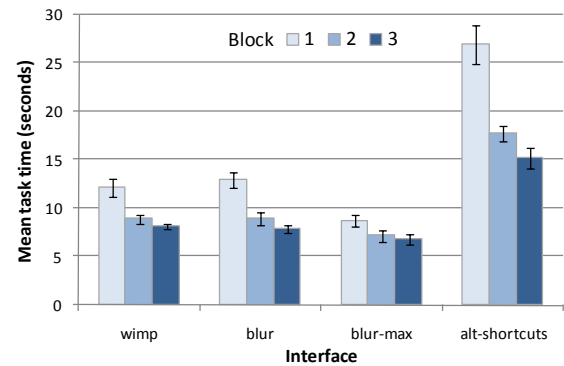


Figure 6. Mean task times with the four interfaces across the three blocks (1, 2, 3). Error bars ±1 s.e.m.

A planned comparison between *wimp* and *blur-max* in the final block (repetitions 8-11, where participants are reaching their maximum level of performance) shows a significant difference ($F_{1,19}=7.6$, $p<.05$) with *blur-max* 17% faster (6.77 seconds, s.d. 2.4) than the normal GUI controls (*wimp*, 8.14 seconds, s.d. 1.45). A final data point indicating that Blur enabled higher levels of performance is that 90% of the participants had their fastest task completion time when using Blur.

Performance with *Alt-shortcuts* was particularly slow, and participants commented that they were ‘painful’ and ‘awful’. Participants also commented about the difficulty of learning the shortcuts due to their lack of symbolism (e.g. ‘<alt> h, g, a, l’ for ‘align left’). 35% of the participants stated that they were aware of Alt-shortcuts before the experiment, but none attempted to use them except during their enforced use in Task 5.

Q3. How large is the performance dip with Blur?

Our framework postulates that changing modality causes a performance dip, so we analysed performance data at the point that users switched to using Blur. Mean trial completion times are shown in Figure 7, with the solid line showing times immediately preceding and following the switch in the *blur* condition, and the dashed line showing data for the first four trials in the *wimp* condition for comparison. The figure shows that Blur caused a small performance dip in the first post-switch trial (from 20.5 sec to 20.9 sec), but that this loss was quickly recovered with

performance matching that of the *wimp* condition by the third post-switch trial and eventually outperforming it.

Importantly, Blur’s performance dip was relatively small, and consequently the participants were not discouraged from continuing to use it. The performance dip with *alt*, in contrast, was large (Figure 6), and participants would have immediately discontinued use if not required to do so.

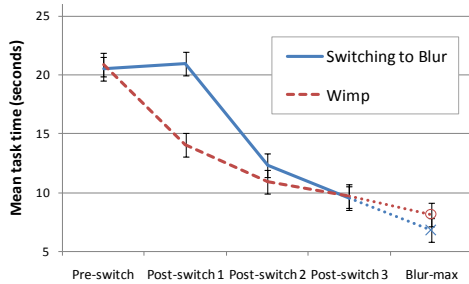


Figure 7. Evidence of Blur’s small ‘performance dip’, shown by the slight increase in task time in trials preceding and following the switch to using Blur (including trial 0).

Q4. What is the subjective response to Blur?

Participants responded positively to Blur, and provided many suggestions for improvements. At the end of the experiment they ranked the three interfaces (*blur*, *wimp* and *alt*) in order of preference, with 1 most preferred, and 3 least. Twelve ranked *blur* first, 6 *wimp* first, and 1 *Alt-shortcuts*, giving a significant rankings difference (Friedman $\chi^2=18.9, p<.001$), with means of 1.37 (s.d. 0.5), 1.84 (0.69) and 2.79 (0.54) for *blur*, *wimp* and *alt*.

We measured how intrusive participants found Blur’s calm notification. One participant stated that it was “*annoying*”, and another that it was “*distracting, but easily ignored*”. The mean response to the question “Blur’s feedback was distracting” (1 disagree, 5 agree) was 2.45 (1.4).

	Wimp	Blur	Alt	χ^2	Sig
Mental Demand	2.45 (1.1)	2.85 (0.9)	4.45 (0.8)	23.5	<.001
Physical Demand	3.15 (1.3)	2.5 (1.1)	3.1 (1.1)	3.8	=0.15
Hurried	2.85 (1.0)	2.8 (1.1)	3.5 (1.1)	3.7	=0.16
Successful	3.95 (0.7)	4.1 (0.4)	2.9 (0.9)	14.8	<.001
Hard work	2.85 (1.3)	3.2 (0.8)	4.3 (0.7)	16.1	<.001
Insecure	2.3 (1.4)	2.2 (0.7)	4.2 (1.0)	19.2	<.001
Efficient	2.9 (1.3)	4.3 (0.7)	2.1 (1.2)	20.1	<.001
Easy to learn	4.1 (1.1)	4.1 (0.8)	2.6 (1.2)	13.7	<.005
Error prone	2.5 (1.1)	2.8 (1.0)	4.35 (0.9)	19.1	<.001

Table 2. Mean responses (sd) to 5 point Likert-scale questions.

NASA-TLX worksheet responses, summarised in Table 2, show significant interface differences for mental demand, perceived success, amount of work, and insecurity, but the main cause is the poor performance of Alt-shortcuts, rather than differences between *wimp* and *blur* conditions. Table 2 also shows higher efficiency ratings for Blur’s (mean 4.3) than *wimp* (2.9) or *alt* (2.1). Mean ratings for Blur’s ease of learning were the same as the normal interface (4.1), despite their brief exposure to it; their rating of Alt-shortcuts learnability was much worse (2.6).

The final stage of the experiment involved asking participants to switch between windows and launch applications, performing minor operations in each: conducting an image search in Firefox, copying one of the resultant images to Paint, cropping the image, pasting it into a Microsoft Word document, checking their email inbox, then repeating the process with a different image search. They were instructed to complete the tasks in any way they liked, and that Blur was available.

The participants made extensive use of Blur throughout the task. We noted a tendency for users to persist with the mouse after using it to complete a task (e.g. cropping an image), and one user commented that “*it’s easy to forget that Blur’s available, but the fading window reminds you*” (referring to calm notification). Participants commented that “*I especially like it opening and running programs instead of the start menu*” and that “*I liked fast switching*”.

Finally, two participants stated a desire for recency ordering in command recommendations: “*The ranking system should give higher precedence to the last used command*” and “*Items should be promoted faster*”.

DISCUSSION

Blur’s *calm notification* and *hot commands* interface was designed to promote an initial modality switch, to minimise the performance dip associated with doing so, and to offer a high performance ceiling (perceived and actual). The experiment validated the design, showing an early switch to hot commands, that the performance dip was small, that users continued to use Blur, that it enabled higher levels of performance than the normal UI and Alt-shortcuts, and that users preferred it.


Why did Blur succeed?

Blur is a realistic system that works with unaltered desktop applications. We designed calm notification and hot commands as generalisable and scalable transitional mechanisms that could work in real work settings. Furthermore, our framework suggests that modality switches depend on many factors, including perception of the efficiency of expert modalities. Consequently, it is difficult to isolate the independent contribution of Blur’s design elements in supporting the modality switch – while the gestalt design succeeds, we do not know whether this is due to the perceived efficiency of hot commands, the ready-to-hand reminder provided by calm notification, the one-to-one semantic relationship of hot commands, and so on. The participants’ comments similarly highlight different aspects of Blur’s perceived and actual utility. We believe that calm notification and hot commands are useful and generalisable approaches for interfaces wishing to support a transition to expertise, but further work is needed to tease out their independent and interacting value.

Does Blur work in the real world?

The experiment used repetitive tasks to compress long term command use into the short duration of a lab study. This is a common strategy for examining interface learning (e.g. [5,

17]), but it raises concerns that the findings may not generalise to real use, discussed below.

Limited vocabulary. The study used a small command vocabulary, so there are risks that Blur may not generalise to larger command sets (in the same way that Grossman's results suggest that hotkey vocabularies may be limited [17]). Two issues encourage us to believe that Blur's approach is robust to large vocabularies. First, the final stage of the experiment involved relatively unconstrained interaction within and across applications. The participants continued to use Blur throughout these activities, and were enthusiastic about its support for window switching and application launching, which were not heavily repeated. Second, we believe that the strategies of one-to-one Blur/GUI command relationship and of populating command names from the GUI create a strong and symbolic mapping that helps users anticipate commands. For example, Blur's command for  is 'subscript' (or 'sub'), and the Zoom control is 'zoom' (or 'zo'), but the built-in hotkeys are much less symbolic and (we believe) harder to remember: 'Ctrl+=', (or 'Alt+h5') and 'Alt+wq'.

Novelty bias. Participants in the experiment will have inferred that we were interested in their performance with Blur, which may have drawn them to use it. We counter this concern in two ways. First, Alt-shortcuts were also novel to most of our participants, but their response was strongly negative. Second, any user who installs a system like Blur is also likely to be curious about its behavior.

Not real work. Real work has different engagement and time pressure than experimental tasks. This is true of nearly all controlled experiments, but is particularly important in our experiment where we are concerned about reducing the tendency to satisfice. We believe that our participants were genuinely trying to optimize their performance, but understand that this may have artificially eliminated some of the tendency to satisfice. We will examine Blur's logs of real use in the coming year.

Our own experiences. Our concerns about real-world use are eased by our own experiences in using Blur over the last few months. Two of the authors are enthusiastic users (the other two use Macs and cannot run it), particularly for two activities and settings. First, application launching and window switching is extremely rapid (e.g., 'Esc+in' to check the email inbox and 'Esc+fi' to launch Firefox). Second, Blur's control of applications is invaluable when using a laptop computer without a mouse (e.g., on planes, in waiting rooms, etc.) We have found that the threshold for using Blur is influenced by the pointing device. When using a mouse, the threshold often favours Blur, but not when tasks predominantly involve direct manipulation (e.g., drawing). However, when using a less precise device, like a trackpad, Blur's benefits are substantial.

How can Blur be improved and generalised?

Improvements to Blur's CLI trigger. Four participants stated that using the Escape key was awkward, and that they

would prefer to assign their own key, such as Alt. One also stated that Blur's hot command window should remain open until explicitly dismissed, allowing multiple commands with a single Escape keypress.

Alternative implementations of calm notification and hot commands. Many tasks are predominantly mouse driven (e.g. CAD drawing), so hot commands would require a homing action away from the mouse. Blur could be adapted to support other expert modalities such as a ListMap [19], which provides a spatially stable miniaturized representation of UI controls (flattening the interface hierarchy). Calm notification could alert users to the location of selected controls within the ListMap. We are currently implementing this approach.

Removing limitations of the platform. Blur's support is limited by the capabilities of the UI Automation API and by how applications implement it. Many applications do not fully or properly implement the API, which requires Blur to implement work-arounds, such as parsing the control tree when an application is launched and dynamically detecting shortcuts for applications that do not properly connect the control tree. These issues increase implementation complexity and constrain functionality, but software vendors like Microsoft and Apple could ease the limitations by refining their scripting and automation technologies, and by promoting their use in applications.

Despite these limitations, Blur is a useful tool in its current implementation, and we have been using it successfully in unaltered Windows environments for several months.

CONCLUSIONS

Many office workers use their desktop applications for hundreds of hours each year, yet there is tendency to resist making the transition to expert strategies that could dramatically improve their performance.

This paper provided a framework for understanding the human and interface factors influencing the transition to expert interface modalities. We also described the design and evaluation of Blur, which uses *calm notification of hot commands* to prompt an initial switch to its expert modality, to minimise the performance dip associated with doing so, and to enable a high performance ceiling. Experimental participants made an early and sustained switch to Blur's hot commands, they benefited from doing so, and they preferred it to the normal interface.

There are several directions for further work. We will experiment with ListMap strategies for mouse-driven expert interaction, and with associated spatial means for calm notification. We will also refine and extend Blur's capabilities to assure that it is robust and functionally rich both within and across a wide range of applications. We will continue to empirically assess how its design features contribute to the initial modality switch, and finally, we will deploy Blur and assess its real-world use through field studies and log analyses.

REFERENCES

1. Anderson, J. Learning and Memory. Wiley, NY, 1995.
2. Barrett, R., Kandogan, E., Maglio, P., Haber, E., Takayama, L. and Prabaker, M. Field studies of computer system administrators. in *Proc. CSCW'04*, ACM, (2004), 388-395.
3. Bhavnani, S. and John, B. The Strategic Use of Complex Computer Systems. *HCIJ 15* (2000), 107-137.
4. Bunt, A., Conati, C. and McGrenere, J. Supporting Interface Customization using a Mixed-Initiative Approach. in *Proc. IUI '07*, ACM, (2007), 92-101.
5. Card, S.K., Moran, T.P. and Newell, A. *The Psychology of HCI*. Lawrence Erlbaum, 1983.
6. Carroll, J. and Carrithers, C. Training Wheels in a User Interface. *Comms. ACM 27*, 8 (1984), 800-806.
7. Carroll, J. and Rossen, M. Paradox of the active user. in Carroll, J. ed. *Interfacing Thought: Cognitive Aspects of HCI*, MIT Press, 1987, 80-111.
8. Cockburn, A., Kristensson, P., Alexander, J. and Zhai, S. Hard Lessons: Effort-Inducing Interfaces Benefit Spatial Learning. in *Proc. CHI*, (2007), 1571-1580.
9. Cockburn, A. and McKenzie, B. Evaluating the Effectiveness of Spatial Memory in 2D and 3D Physical and Virtual Environments. in *Proc. CHI'02*, ACM, (2002), 203-210.
10. Craik, F. and Lockhart, R. Levels of processing: A framework for memory research. *Journal of Verbal Learning and Verbal Behavior 11* (1972), 671-684.
11. Cypher, A., Dontcheva, M., Lau, T. and Nichols, J. *No Code Required*. Morgan Kaufmann, 2010.
12. Czerwinski, M., Horvitz, E. and Cuttrel, E. Subjective Duration Assessment: An Implicit Probe for Software Usability. in *Proc. IHM-HCI*, (2001).
13. Dix, A., Finlay, J., Abowd, G. and Beale, R. *Human-Computer Interaction*, Prentice Hall. (1993).
14. Dyck, J., Pienelle, D., Brown, B. and Gutwin, C. Learning from Games: HCI Innovations in Entertainment Software. in *Proc. GI*, (2003).
15. Furnas, G.W., Landauer, T.K., Gomez, L.M. and Dumais, S.T. The vocabulary problem in human-system communication. *CACM 30*, 11 (1987), 964-971.
16. Greenberg, S. and Witten, I. Supporting Command Reuse. *IJMMS. 39* (1993), 353-390.
17. Grossman, T., Dragicevic, P. and Balakrishnan, R. Strategies for Accelerating On-line Learning of Hotkeys. in *Proc. CHI'07*, ACM, (2007). 1591-1600.
18. Grossman, T., Fitzmaurice, G. and Attar, R. A survey of software learnability. in *Proc. CHI*, (2009). 649-658.
19. Gutwin, C. and Cockburn, A. Improving List Revisitation with ListMaps. in *Proc. AVI'06*, ACM, (2006), 396-403.
20. Hart, S. and Staveland, L. Development of NASA-TLX. in Hancock, P. and Meshkati, N. eds. *Human Mental Workload*, 1988, 139-183.
21. Hendy, J., Booth, K. and McGrenere, J. Graphically Enhanced Keyboard Accelerators for GUIs. in *Proc. Graphics Interface*, (2010).
22. Jones, T. Incidental learning during information retrieval: a hypertext experiment. in Maurer, H. ed. *Computer Assisted Learning*, Springer, 1989, 235-251.
23. Karat, J., Karat, C. and Ukelson, J. Affordances, motivation and the design of user interfaces. *CACM 43*, 8 (2000), 49-51.
24. Ko, A., Myers, B. and Aung, H. Six Learning Barriers in End-User Programming Systems. in *Proc. VL HCC'04*, IEEE, (2004), 199-206.
25. Kurtenbach, G. and Buxton, B. The Limits of Expert Performance Using Hierarchic Marking Menus. in *Proc. InterCHI'93*, (1993), 482-487.
26. Landauer, T.K., Galotti, K.M. and Hartwell, S. Natural command names and initial learning: a study of text-editing terms. *Comms. ACM 26*, 7 (1983), 495-503.
27. Lane, D.M., Napier, H.A., Peres, S.C. and Sandor, A. Hidden costs of graphical user interfaces. *I. J. HCI 18*, 2 (2005), 133-144.
28. Mackay, W. Triggers and barriers to customizing software. in *Proc. CHI'91*, ACM, (1991), 153-160.
29. Maslow, A. *The Psychology of Science: A Reconnaissance*. Harper & Row, New York, 1966.
30. Newell, A. and Rosenbloom, P.S. Mechanisms of Skill Acquisition and the Law of Practice. in Anderson, J. ed. *Cog. Skills & Acquisition*, Erlbaum, 1981, 1-55.
31. Nielsen, J. *Usability Engineering*. Morgan Kaufmann, San Francisco, 1993.
32. Norman, D. Design principles for Human-Computer Interfaces. in *Proc. CHI 83*, 1983, 1-10.
33. Norman, D. *The Psych. of Everyday Things* (1988).
34. Odell, D., L., Davis, R., C., Smith, A. and Wright, P., K. Toolglasses, marking menus, and hotkeys: a comparison of one and two-handed command selection techniques. in *Proc. Graphics Interface*, (2004), 17-24.
35. Schmidt, R. and Bjork, R. The Conceptualizations of Practice. *Psychological Science 3*, 4 (1992), 207-217.
36. Shelton, D. and Newhouse, R.C. Incidental Learning in a Paired-Associate Task. *Journal of Experimental Education 50*, 1 (1981), 36-38.
37. Shneiderman, B. *Designing the User Interface*, Addison Wesley, 1992.
38. Shneiderman, B. Direct Manipulation: A Step Beyond Programming Languages (excerpt). in Baecker, *et al.*. *Readings in HCI*, 1987, 461-467.
39. Shneiderman, B. Promoting universal usability with multi-layer interface design. in *Proc. Universal Usability*, ACM, (2003), 1-8.
40. Simon, H. Theories of Decision-Making in Economics and Behavioral Science. *American Economic Review 49*, 3 (1959), 252-283.
41. Tauscher, L. and Greenberg, S. How People Revisit Web Pages. *IJHCS. 47*, 1 (1997), 97-138.
42. Whiteside, J., Jones, S., Levy, P. and Wixon, D. User Performance with Command, Menu, and Iconic Interfaces. in *Proc. CHI'85*, ACM, (1985), 185-191.