Improving Window Switching Interfaces

Susanne Tak[1], Andy Cockburn[1], Keith Humm[1], David Ahlström[2], Carl Gutwin[3], and Joey Scarr[1]

[1] Computer Science and Software Engineering, University of Canterbury, Private Bag 4800, Christchurch 8140, New Zealand
susanne.tak@pg.canterbury.ac.nz, andy@cosc.canterbury.ac.nz, {khu23,jls129}@student.canterbury.ac.nz
[2] Department of Informatics Systems, Klagenfurt University, Universitätsstrasse 65-67, A-9020 Klagenfurt, Austria
david@isys.uni-klu.ac.at
[3] Department of Computer Science, University of Saskatchewan, 110 Science Place, Saskatoon, Saskatchewan, S7N 5C9, Canada
gutwin@cs.usask.ca

Abstract. Switching between windows on a computer is a frequent activity, but current switching mechanisms make it difficult to find items. We carried out a longitudinal study that recorded actual window switching behaviour. We found that window revisitation is very common, and that people spend most time working with a small set of windows and applications. We identify two design principles from these observations. First, spatial constancy in the layout of items in a switching interface can aid memorability and support revisitation. Second, gradually adjusting the size of application and window zones in a switcher can improve visibility and targeting for frequently-used items. We carried out two studies to confirm the value of these design ideas. The first showed that spatially stable layouts are significantly faster than the commonly-used recency layout. The second showed that gradual adjustments to accommodate new applications and windows do not reduce performance.

Keywords: window switching, revisitation patterns, spatial constancy.

1 Introduction

Switching between windows like email applications, word processors, and Web browsers is a very common task. A previous study [1] found that the mean time between window switches is only 20.9 seconds and that users have more than eight windows open more than 78% of the time. It has also been reported that the average number of simultaneously opened windows increases with available display space: from four for single monitor users to up to 18 for users with multiple monitors [2].

Current interface methods for window switching have changed relatively little since early graphical user interfaces – clicking on a window brings it into focus, as does selecting the window from a spatial iconic representation (e.g., the Windows Taskbar) or from a recency list (e.g., the Windows Alt+Tab display). Recent research and commercial systems demonstrate alternatives to these mechanisms, but problems

T. Gross et al. (Eds.): INTERACT 2009, Part II, LNCS 5727, pp. 187–200, 2009.

still exist – with more than a few windows in the display, users must often carry out a laborious search to find the desired window, even if that window is used frequently.

Despite the different designs that have been proposed, no window switching tools are based on empirical evidence about how people revisit windows in actual desktop work. To address this limitation, and to identify new design principles for window switching tools, we carried out three studies. We first conducted a longitudinal study where window switching behaviour was recorded; this study showed that revisitation is frequent, both to applications and to specific windows, and that most switches are to a small number of applications. From this study, we identified the design principles of spatial constancy and morphing target sizes. Spatial constancy supports window revisitation by keeping thumbnails for activating windows and applications in the same place in the switching display, allowing users to build up spatial memory of frequently-used items. Morphing target sizes allocates more space to frequently-used applications and windows, improving Fitts' Law targeting time for the most frequent items, and allowing for the addition of new items.

Our second and third studies tested the value of these two principles. The second showed that spatial constancy is effective: stable layouts are significantly faster than recency layouts (similar to Windows Alt+Tab). The third study showed that gradual adjustments in the display's layout do not reduce (or improve) performance. We discuss how these successful results can be deployed in new window switching tools.

2 Related Work

Two areas of related work inform our investigation: research on task and window switching interfaces; and studies of user behaviour with desktops and windows.

2.1 Window Switching Interfaces

The importance and frequency of window switching has led to extensive research and development into improved interfaces for the task. Two interface properties help distinguish window switching approaches: first, the type of information used to form and display relationships between windows, such as temporal, spatial, or semantic data about the windows; and second, the degree to which systems try to automatically establish these relationships, with some being entirely manual while others use sophisticated predictive methods to automatically establish window relationships.

Henderson and Card's seminal work with the *Rooms* [3] virtual desktop manager was almost entirely manual, with the user assuming all responsibility for the spatial placement of windows within a room based metaphor. *Scalable Fabric* [4] also uses manual relationship controls, including extensive support for zooming, but unlike *Rooms* it provides little explicit structure for grouping windows. The default desktop behaviour of Microsoft Windows XP/Vista is also largely manual (users place windows where they wish), although it automatically groups windows belonging to each application in the Taskbar. Microsoft's *GroupBar* [2] maintains manual control, but replaces the Taskbar's application-based grouping with user-defined task groups.

The primary limitation with manual control of window relationships is that users must carry out explicit actions to gain potential benefits. To remove the dependence

on manual actions many systems automatically form window relationships. The most basic and widely deployed automatic relationship system is the familiar Alt+Tab key binding in Microsoft Windows operating systems. Alt+Tab allows users to rapidly flip through a temporally based 'z-ordering' of windows on the display. Kumar *et al.* [5] observe that Alt+Tab is very efficient when the number of windows is low, but researchers have also labelled the method 'tedious' [6]. Mac OS X's Exposé also uses an automatic layout, displaying thumbnails of all windows at once; but the layout is not spatially constant, so users cannot accurately predict where specific windows will be located, demanding visual search to find them.

SWISH [7], *UMEA* [8], *TaskTracer* [9], *RelAltTab* [10] and *Taskposé* [11] all use sophisticated methods to automatically adapt to user activities. *SWISH* uses temporal relationships between window focus events as well as window titles to establish semantic relationships. Their evaluations suggested 70% accuracy rates in assigning windows to task groups. Similarly, *TaskTracer* uses machine learning to modify the Microsoft Start menu, Taskbar and Windows Explorer. *RelAltTab* uses similar methods to modify the Alt+Tab window order. *Taskposé* provides an overview in which windows drift towards each other dependent on their temporal relationships, based on the results of a 'WindowRank' algorithm. *Taskposé* windows also gradually enlarge to reflect their relative importance as calculated by the algorithm.

The primary limitations of automatically adaptive systems are that they can incorrectly predict the user's intention and that users can fail to understand or anticipate the system's adaptation [12]. When this happens users must resort to time-consuming visual search of candidate targets. Of all the previous designs, however, only two – *WindowScape* [13] and *Elastic Windows* [14] – used stable layouts to help improve memorability of previously-used windows.

2.2 Studies of Window Use

Gaylin [15] provides an analysis of window activities from the early days of graphical user interfaces, based on 22 minute observations of nine participants. Observations show that window switching activities were far more frequent than window creation, deletion, or geometry management. Hutchings *et al.* [1] update these findings using automatic logs of 39 participants over a 3-week period. They report on how window management activities differ across single- and multi-monitor display use (in common with Grudin's earlier field study [6]) but their data also highlights important general characteristics of window management. This includes the finding that window switching is extremely frequent, with a mean window activation time of 20.9 seconds, and a median of only 3.77 seconds. This frenetic frequency of window switching is confirmed by Mackinlay and Royer [16], who also conducted a log analysis of window switching. Hutching *et al.'s* data also shows that users normally have many windows open, with eight or more windows open 78.1% of the time.

Many types of human behaviour are highly repetitive, as observed by Zipf's Law [17] and the Pareto Principle [18] (also called the "80-20" rule), where a large portion of effects comes from a small portion of causes. Zipfian distributions have been observed in many areas of computer use [19], such as frequency of command use and menu use. Although window and application switching activities are among the most frequent in computer use, with one study showing a mean time of 21 seconds between

actions [1], we are unaware of any empirical studies of users' patterns of revisitation to windows and applications. To address this limitation, we carried out the longitudinal study described in the following section.

3 Log Study of Application and Window Switching

To find out how users switch between applications and windows, we carried out a longitudinal study that recorded windowing behaviour as users went about their everyday tasks. We developed logging software for Windows XP that unobtrusively monitors window switches, window creation/destruction events, changes in window geometry, and the method used to switch windows (e.g., Alt+Tab, Taskbar, and direct mouse click). Nine frequent computer users (18 to 55 years old) took part in a study during which we recorded between 8 and 117 days of data per participant. Four participants used dual monitors; five, single monitors (see Table 1).

Overall, we obtained 241 person-days of data. Only manual window switches were included in the analysis: automatic switches (such as window/dialog pop-ups) were removed from the data. This left a total of 45,377 switch events.

3.1 Results

Results are divided into three categories. First, we present data describing revisitation patterns for the windows used each day. This characterises the users' main activities in window switching. Second, we describe long term revisitation patterns to applications. While many windows are transient, existing only for immediate work requirements (such as a window containing an email message during its composition) applications are relatively stable. For many tasks, such as "search on the Web for topic X" or "check my email inbox" the user's target is likely to be the application (e.g., Firefox or Outlook) rather than a specific window, so an application based analysis is potentially informative. Third, we analyse the interface mechanisms used to carry out window switching activities to gain insights to how current interfaces are used and to determine whether users adopt similar or divergent patterns of behaviour.

Daily Window Revisitation. For each participant on each day, we analysed how frequently each window was revisited. This was conducted by forming a ranked order of windows according to their percentage of total daily switches.

The number of window switches per day for participants ranged from 5 to 807, with a cross participant mean of 219 per day (*s.d.* 91). The number of distinct windows switched to per day ranged from 3 to 177, with a cross participant mean of 39 (*s.d.* 19). In their study of Web revisitation, Tauscher and Greenberg [20] define the *recurrence rate R* as the probability that any URL visit is a repeat of a previous visit, giving $R=(\textit{total URLs visited} - \textit{distinct URLs visited})/\textit{total URLs visited} \times 100$. Adapting this formula for window revisitation gives a mean recurrence rate of 82%; much higher than the 58-61% rate reported for the Web. This data shows that window revisitation is a very common activity.

Analysing the same data with respect to the Pareto principle shows that 80% of window switches were triggered by between 24 and 40% of windows for the different participants (see Table 1), with a mean of 35.1%.

Table 1. Application and window switching data for the nine participants

P.	single/ dual	# days	# switch	# apps	Pareto. 80% of switches by what %		Interface %		
					Windows	*Apps*	*Click*	*Task-bar*	*Alt+ Tab*
1	single	10	1396	40	40	15	23.4	76.7	0.0
2	dual	117	17088	45	34	7	78.4	21.4	0.2
3	single	23	2588	23	37	17	37.0	47.7	15.2
4	single	16	3019	49	37	16	35.3	61.3	3.4
5	dual	9	2454	37	40	22	64.5	35.5	0.0
6	dual	30	10373	54	31	13	64.1	35.5	0.4
7	dual	8	2922	34	35	18	82.7	14.5	2.8
8	single	13	2989	39	24	18	22.7	75.3	1.9
9	single	15	2548	23	37	17	7.6	7.7	84.7

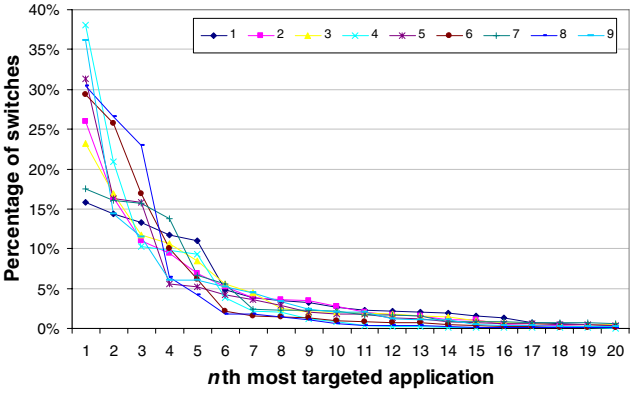


Fig. 1. Percentage of switches to the 20 most frequent applications for each participant

Revisitation to Applications across Sessions. As mentioned above, many user tasks involve targeting an application rather than a specific window. For example, a user may have several Web browser windows concurrently open, any of which can be used for a pressing search task; or the user may need to target a calculator, file explorer, etc. In each case the application is the target, not a specific window. We therefore analysed each participant's revisitation to applications for the duration of the study.

The number of distinct applications used by the participants ranged from 23 to 54, with a mean of 38 (*s.d.* 10.6). For each participant the total number of switches to each application was calculated, ranked by frequency, and converted to a percentage of their total application switches. Fig. 1 shows this data for each participant. It is clear that a few applications are used a lot, and a lot are used relatively little.

Table 1 shows Pareto principle data, revealing that most users' application revisitation roughly adheres to the 80-20 rule. For example, 22% of participant five's applications accounted for 80% of switches. All other participants' application revisitation was *more* pronounced than the Pareto principle predicts: e.g., only 7% of participant two's application switches accounted for 80% of switches.

3.2 Interface Mechanisms Used to Switch Windows

We analysed the main three interface mechanisms currently used to visit and revisit windows: direct window clicks, selection from the Taskbar, and Alt+Tab. This analysis was conducted to determine whether users made similar or divergent use of the tools available. Table 1 shows each participant's use of these tools. Two important observations from this data are as follows. First, there are substantial differences between participants in their use of Alt+Tab. Seven of the participants used it very lightly (less than 3.5% of window activations) or not at all; one used it fairly often (15.2%); and one used it almost exclusively (84.7%). The two participants who used Alt+Tab heavily had a single monitor, suggesting that it might be most valuable for users with constrained screen real-estate (see Fig. 2). Second, direct clicks and Taskbar use were also influenced by screen real-estate, with dual monitor participants using direct clicks more and the Taskbar less than single monitor participants.

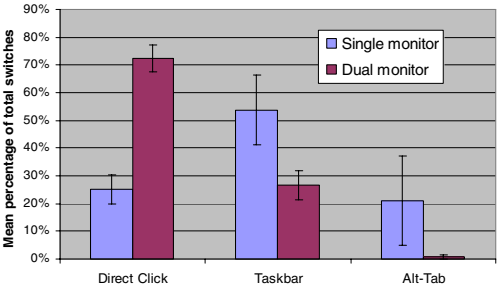


Fig. 2. Cross participant means of the percentage of window switches activated by clicking on the window, Taskbar selections, and Alt+Tab. Error bars show ±1 standard error.

4 Design Principles for Window Switchers

The patterns observed in the log study strongly suggest that supporting revisitation should be a main design goal in window switching tools. More specifically, this support should allow users to quickly find and distinguish between previously-visited windows in a switcher's display. There are several possible ways to provide this support (for example, previously-visited windows could be highlighted in the display, or the most frequently-used windows could be shown first), but previous research in psychology argues for an approach that makes use of *spatial constancy*, discussed in Section 4.1. However, user's patterns of behaviour change over time: for instance, a user may replace an application with a different vendor's system. Complete spatial constancy does not allow for the addition/removal of items. We address this problem with the use of *size morphing* (Section 4.2). Morphing target sizes allow for the addition of new items while maintaining as much spatial stability as possible. Also, allocating more space to frequently-used applications and windows will reduce their Fitts' Law targeting times. Given that some applications and windows are used much more frequently than others this might increase overall performance.

4.1 Supporting Window and Application Revisitation with Spatial Constancy

Spatial location memory is a person's memory of where objects are in space. It is well developed and can be extremely fast: studies have shown that items can be found in time proportional to the logarithm of the size of the set, which can be much faster than the linear search time needed for unorganised sets [19]. This fast performance is enabled by spatial constancy – that is, items remaining in the same location over time. This idea has been known since the first interface design guidelines [21], and many studies have demonstrated its effectiveness.

There is also evidence that spatial location memory in user interfaces is surprisingly robust. In studies of the Data Mountain's spatial layout of thumbnail images [22] users were able to quickly and accurately recall the location of specific targets four months after originally creating a layout of 100 images. Furthermore, their retrieval performance was not significantly harmed when the images were replaced with blank outlines. Spatial constancy, however, has received little attention in research on task- and window switching tools (with a few exceptions [13, 14]).

Applying the idea of spatial constancy in a window switcher implies that windows and applications should not move in the switcher's display. Within a work session, this design approach has clear advantages: since revisitation is strong, people will quickly learn the locations of the most frequently-used windows. Spatial stability offers similar advantages across work sessions. Since people regularly return to a relatively small number of applications, the problem of where to place individual windows in the switcher display can be solved by creating 'application zones' that are themselves spatially constant, and that reflect people's longer-term repeating work patterns. Spatial constancy can thus be applied in a hierarchical fashion: applications are given stable zones in the switcher display, since these change slowly over the long term; and within each zone, the application windows used in the current work session are placed in stable spatial locations. The application-based organisation provides an initial guide to the location of a new window, but as the window is used more and more frequently, users will start to remember its location as a separate entity.

4.2 Size Morphing to Accommodate Change and Optimise Performance

The design principle of spatial constancy potentially conflicts with changes in patterns of behaviour. For example, when users replace one application with another how can spatial stability be maintained? Also, given that some applications and windows are used much more frequently than others, how can the switching interface depict the relative importance of applications and windows, and optimise the acquisition of frequent targets?

Our proposed solution is to use gradual size 'morphing' to adjust the sizes of application zones and window thumbnails. Morphing avoids abrupt changes in layout that would damage spatial memory, while allowing the introduction of new items and enabling frequent targets to be enlarged to enhance their visibility and to reduce pointing time.

Fig. 3 shows a mock-up of our design, which uses the entire screen when activated. Clicking on a zone or thumbnail immediately switches to the associated application or window. Each zone contains thumbnails representing all windows associated with the

(a) Initial view. (b) After morphing.

Fig. 3. Prototype application zones, before and after size morphing. The application and window in the top left have been used most frequently.

application, scaled and tiled to fit. Initially all application zones are of equal size (see Fig. 3a), but they gradually morph in size to reflect frequency of use (see Fig. 3b).

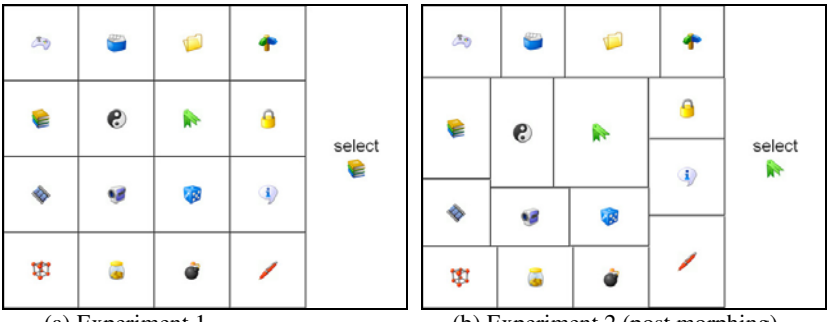
5 Evaluating Spatial Constancy and Morphing

Our design principles assume that users will perform better at switching to windows when thumbnails remain in spatially stable locations than when they are rearranged according to recency or other properties. It also assumes that size morphing will allow users to maintain their spatial memory while the interface adapts. We conducted two experiments to separately investigate these issues.

5.1 Experiment 1: Spatial, Recency, or Frequency Ordering?

A window switching interface can order items in a variety of ways, including spatially or by recency (like Alt+Tab). Other orders are also possible: frequency reordering might work well for Zipf-like distributions; random orders may perform well if visual popout effects are strong. This experiment, therefore, investigates the performance impact of four different orderings (spatially stable, recency order, frequency order, and random order) for tasks involving acquisition of targets in a Zipf-like distribution.

The experimental interface consisted of a grid of distinct icons (Fig. 4a) with a cued target on the right. Participants were instructed to click on the target icon region as quickly and accurately as possible, with each successful acquisition immediately cueing the next. The *spatially stable* layout used an arbitrarily but stable order (i.e., icons never moved). In the *random layout* all items were randomly repositioned after each selection. The *recency reordering* condition moved the most recently selected item to the top left of the grid, pushing earlier items along in row-major order (similar to Alt+Tab). Finally, *frequency reordering* repositioned items according to their cumulative selection counts (most frequent at top left, in row-major order). Note that frequency reordering rapidly stabilizes with a Zipf-like distribution of targets, while recency ordering is less stable.



(a) Experiment 1. (b) Experiment 2 (post morphing).

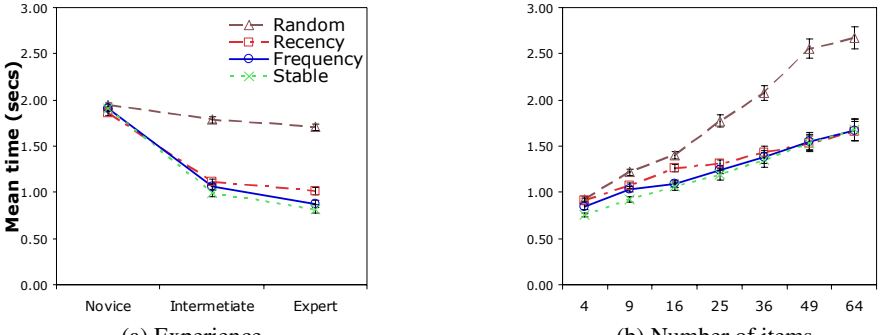
Fig. 4. Experimental interfaces showing 16 icon regions and the target cue on the right

To understand how performance with these interfaces is influenced by number of targets, participants completed trials with 4, 9, 16, 25, 36, 49 and 64-item grids. The Zipfian distribution of targets was generated by randomly selecting eight targets from among the candidates (all four for the 4 item grid): one was cued 10 times, one 5 times, then 3, 2, 2, 1, 1, and 1 for the others.

Twenty-six students volunteered for the experiment (16 male, 10 female, 16-44 years old). Their participation lasted approximately 45 minutes.

Results and Discussion. The selection time dependent measure was analysed using a 4×3×7 RM-ANOVA for factors *layout* (stable, frequency, recency, random), *experience* (novice, intermediate or expert) and *items* (4-64). Experience was determined by assigning first-time icon selections as novice, 2nd-7th selections as intermediate, and 8th-10th as expert.

All factors showed significant main effects: *layout* ($F_{3,60}$=72, p<.001), *experience* ($F_{2,40}$=409, p<.001) and *items* ($F_{6,120}$=135, p<.001) (see Fig. 5). Stable layouts were the fastest (mean 1.2s) followed by frequency reordering (1.3s), recency reordering (1.3s), and random (1.8s). Post hoc comparisons show pairwise differences between all layouts except frequency and recency reordering, and frequency reordering and



(a) Experience. (b) Number of items.

Fig. 5. Experiment 1 mean selection times (±1 standard error) with the four layouts

stable (Bonferroni, $p<.05$). Fig. 5a shows a significant *layout×experience* interaction ($F_{6,120}=47$, $p<.001$) caused by relatively constant performance across *experience* with random layouts in contrast to marked improvement with other layouts. Fig. 5b shows a significant *layout×items* interaction ($F_{18,360}=14$, $p<.001$), caused by the random layout worsening much more rapidly across increased number of items than the other three layouts. The results show that spatial constancy is beneficial. The stable, frequency, and recency layouts all supported expertise development; random did not. The recency layout (similar to Alt+Tab order) was outperformed by the stable layout.

5.2 Experiment 2: The Effect of Size Morphing

The second experiment examined the performance impact of the morphing behaviour used for two purposes: to allow new windows/applications to be introduced, and to change target sizes in response to the Zipf-like frequency distribution. Experiment 1 suggests that total stability is the 'gold standard', but total stability would prohibit the addition of zones for new windows/applications, as well as prohibiting morphing size adaptation to reduce the Fitts' Law targeting time.

We therefore experimentally compared target acquisition performance using totally stable placement (the gold standard control condition) as well as morphing behaviour implemented with *squarified* [23] and *spiral* [24] treemaps. Treemaps recursively divide 2D spaces into rectangles of various sizes, with size representing an underlying quantitative data attribute. Various algorithms for creating treemaps exist, and two main properties are the average aspect ratio and spatial stability (see [24] for a recent review). *Squarified* and *spiral* treemaps were used in this experiment because they respectively support low and high spatial stability. Fig. 4b shows the spiral experimental condition after morphing. The experimental method, procedure and design were similar to Experiment 1. There were seventeen participants (fifteen male, two female, 21-35 years old).

Results and Discussion. Task time was analysed using a 3×3×7 RM-ANOVA for factors *layout*, *experience*, and *items*. There were significant main effects for *items* ($F_{6,96}=204$, $p<.001$) and *experience* ($F_{2,32}=444$, $p<.001$), but not for *layout* ($F_{2,32}=1$, $p=.3$). Mean times for the three layouts were similar at 1.3s, 1.4s and 1.4s with stable, squarified and spiral respectively.

The absence of a significant difference between layouts is interesting and warrants further analysis. The stable interface is not a practical solution to window switching because it prohibits the addition of new applications and windows. However, total stability best supports users in exploiting their spatial memory, so it is interesting that it did *not* significantly outperform either of the two treemap layouts. It is reasonable to suspect that the treemap interfaces allowed users to benefit from reduced Fitts' Law targeting performance with frequently selected items (which gradually morphed to larger sizes), and that this counteracted the penalties associated with reduced absolute stability, but this explanation is not supported by a significant *layout×experience* interaction ($F_{4,64}=2$, $p=.1$). More powerful experimental analysis is required.

6 Discussion

The studies provide important findings for the design of window switching tools:

- Within a work session, people frequently revisit previously-used windows;
- Revisitation to a small set of applications is also strong across sessions;
- Most users rely heavily on mouse-based window switching methods, but some use keyboard methods almost exclusively;
- Spatially-constant item layouts are significantly faster than recency-based layouts for a Zipfian distribution of target stimuli;
- Gradual size 'morphing' of item areas in the display did not affect performance – importantly, morphing did not substantially harm the participant's spatial memory for target locations.

In the following paragraphs, we consider reasons why these design principles succeeded and discuss design issues for real-world window switching tools.

6.1 Why the Principles Succeeded

The results of Experiment 1 add to previous findings showing performance advantages for spatial constancy in comparison to alternatives. Furthermore, these are the first results (that we know of) comparing spatially stable 2D layouts with frequency and recency based layouts.

The stable layout was significantly faster than the recency layout, with performance benefits increasing with expertise. This is easily explained – after each selection in the recency layout users must either calculate the new position of their targets or visually search for them, both of which demand time.

Performance with stable and frequency layouts was similar to each other, raising the question of whether window switching interfaces should use frequency ordering. However, the experiment used a Zipfian distribution of stimuli, causing the frequency layout to quickly settle to spatial stability, so we believe that the frequency layout's comparative success is also explained by its spatial stability (after a short period of reorganisation). In practical use, a frequency layout for windows is unlikely to succeed due to the transient nature of most windows. Consequently, some other basis for organisation is required, such as application zones. Although application zones could be placed in a frequency layout, doing so would create placement instability during early use, which might cause users to discard the system due to its unpredictable behaviour. We therefore believe that using spatial stability as the main layout principle is a preferable solution. Another reason for believing application zones to be a useful placement strategy is that it quickly narrows the user's search space when there are several candidate windows from the same application – rather than having to search the whole display, users can quickly dismiss the majority of candidates by homing in on only those in the appropriate application zone.

The finding from the second experiment that morphing was no worse than absolute spatial stability is important because it suggests that gradual layout changes allow users to successfully use spatial memory while the display adapts to changes in their behaviour. Although we have not yet tested item addition and deletion, we believe that the spiral treemap algorithm used in Experiment 2 will allow users to maintain

their spatial understanding of the layout. We will test this in future work. We will also model and test the Fitts' Law performance advantages gained from morphing.

6.2 Design Considerations for Real-World Switching Tools

The log study and experiments suggest that our design principles can be successful, but several design issues arise in translating these findings into a real-world switching tool, as discussed below.

Number of windows and applications. Fig. 3 portrays our design intention, showing four application zones and seven windows. However, the log study revealed that users actually work with dozens of applications and windows. Can our design scale? We are confident it can for several reasons. First, prior work (e.g., [22]) shows that users can successfully learn and remember the locations of hundreds of targets. Second, our experiments included up to 64 targets, and the benefits of spatial stability increased with the number of targets. Third, our design uses the full display space to show and access all windows and applications at once, allowing much larger targets than is possible with visually compact tools such as the Windows Taskbar.

Flexibility in input device and interface behaviour. The log study showed that most users relied on mouse input for window switching, but that one user used the Alt+Tab key combination almost exclusively. Convenience will clearly influence the choice of input device (e.g., using the mouse when a coffee cup is held in the other hand, or using Alt+Tab when typing to avoid repositioning the hands). However, Alt+Tab is also powerful when the user needs to 'flip' between a few recent windows. It is therefore desirable that next generation window switching tools continue to support flexibility in input devices and recency-based traversal. Our design can easily accommodate both. For example, the tool could be activated by a key combination (such as Alt+Tab) or by a dedicated mouse button or wheel; and the traditional Alt+Tab recency list can be traversed by highlighting items on subsequent key combination or button/wheel press.

Integrated support for application launch and window switching. Another potential advantage of our design (yet to be evaluated) is that it integrates support for different types of window switching tasks. In current interfaces, application launch facilities are largely partitioned from window switching tools, yet the user activities are closely related. For example, to "search on the Web for topic X" the user needs to acquire a browser window. If the user starts by searching the Taskbar they will be unsuccessful if no browser windows are active, necessitating a second action such as navigating through the Start menu hierarchy to launch the browser. Alternatively, if the user begins by launching the application, they will often gain a superfluous window (when others were already available) adding to their window management load. Our design, in contrast, provides a single interface mechanism for all window and application activities: if the application zone is empty, the user clicks to launch; but if windows are already available, any candidate can be immediately brought into focus.

Tailored layouts. Finally, the design could be adapted to allow users various manual controls, such as ensuring that certain application zones are always displayed in specific locations (perhaps to ensure consistency between a desktop and laptop display), or excluding particular applications from appearing in the layout.

7 Conclusions and Future Work

Log analysis of real window and application switching showed that revisitation (returning to previously used windows and applications) is an extremely frequent activity in computer use. It also showed that most users activate their window and application switches using the mouse as the input device, but that some users rely on keyboard methods almost exclusively. These empirical characterisations of window and application switching are new, and although they may confirm (or refute) designers' intuitions, empirical confirmation is necessary for informed design. The main design message of the log analysis is that window and application switching interfaces should improve efficiency by explicitly supporting revisitation.

We proposed two design principles to better support revisitation: spatial constancy of the interface controls to activate applications and the windows belonging to them; and size morphing to allow the spatial display to adapt to changes of behaviour and to optimise selection of frequent targets. Two empirical studies showed, first, that spatially stable layouts allow faster acquisition than recency and random layouts for skewed distributions such as those occurring in window switching tasks, and second, that size morphing is not significantly slower than the (idealistic but impractical) gold standard of absolute spatial stability.

In future work we will conduct a longitudinal study of user performance with a complete tool implementing the principles derived and tested in this paper.

References

1. Hutchings, D., Smith, G., Meyers, B., Czerwinski, M., Robertson, G.: Display Space Usage and Window Management Operation Comparisons between Single Monitor and Multiple Monitor Users. In: Proc. AVI 2004, pp. 32–39. ACM Press, New York (2004)
2. Smith, G., Baudisch, P., Robertson, G., Czerwinski, M., Meyers, B., Robbins, D., Horvitz, E., Andrews, D.: GroupBar: The TaskBar Evolved. In: Proc. OzCHI 2003, pp. 34–43 (2003)
3. Henderson, A., Card, S.: Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. ACM Transactions on Graphics 5(3), 211–243 (1986)
4. Robertson, G., Horvitz, E., Czerwinski, M., Baudisch, P., Hutchings, D., Meyers, B., Robbins, D., Smith, G.: Scalable Fabric: Flexible Task Management. In: Proc. AVI 2004, pp. 85–89. ACM Press, New York (2004)
5. Kumar, M., Paepcke, A., Winograd, T.: EyeExposé: Switching Applications with Your Eyes, Stanford University (2007), `http://hci.stanford.edu/cstr/reports/2007-02.pdf`
6. Grudin, J.: Partitioning Digital Worlds: Focal and Peripheral Awareness in Multiple Monitor Use. In: Proc. CHI 2001, pp. 458–465. ACM Press, New York (2001)
7. Oliver, N., Smith, G., Thakkar, C., Surendran, A.: SWISH: Semantic Analysis of Window Titles and Switching History. In: Proc. IUI 2006, pp. 194–201. ACM Press, New York (2006)
8. Kaptelinin, V.: UMEA: Translating Interaction Histories into Project Contexts. In: Proc. CHI 2003, pp. 353–360. ACM Press, New York (2003)

9. Dragunov, A., Dietterich, T., Johnsrude, K., McLaughlin, M., Li, L., Herlocker, J.: TaskTracer: A Desktop Environment to Support Multi-tasking Knowledge Workers. In: Proc. IUI 2005, pp. 75–82. ACM Press, New York (2005)
10. Oliver, N., Czerwinski, M., Smith, G., Roomp, K.: RelAltTab: Assisting users in switching windows. In: Proc. IUI 2006, pp. 385–388. ACM Press, New York (2006)
11. Bernstein, M., Shrager, J., Winograd, T.: Taskposé: Exploring Fluid Boundaries in an Associative Window Visualization. In: Proc. UIST 2008, pp. 231–234. ACM, New York (2008)
12. Shneiderman, B.: Direct Manipulation for Comprehensible, Predictable, and Controllable User Interfaces. In: IUI 1997, pp. 33–39. ACM Press, New York (1997)
13. Tashman, C.: WindowScape: A Task Oriented Window Manager. In: Proc. UIST 2006, pp. 77–80. ACM Press, New York (2006)
14. Kandogan, E., Shneiderman, B.: Elastic Windows: Improved Spatial Layout and Rapid Multiple Window Operations. In: Proc. AVI 1996, pp. 29–38. ACM Press, New York (1996)
15. Gaylin, K.: How are Windows Used? Some Notes on Creating an Empirically-Based Windowing Benchmark Task. In: Proc. CHI 1986, pp. 96–100. ACM Press, New York (1986)
16. Mackinlay, J., Royer, C.: Log-based Longitudinal Study Finds Window Thrashing, Palo Alto Research Center, UIS-2004-06 (2004)
17. Zipf, G.: Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology. Addison-Wesley, Reading (1949)
18. Juran, J.: Quality Control Handbook. McGraw-Hill, New York (1951)
19. Cockburn, A., Gutwin, C., Greenberg, S.: A Predictive Model of Menu Performance. In: Proc. CHI 2007, pp. 627–636. ACM Press, New York (2007)
20. Tauscher, L., Greenberg, S.: How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. IJ. Hum. Comp. Stud., 47(1), 97–138 (1997)
21. Hansen, W.: User Engineering Principles for Interactive Systems. In: Barstow, D., Shrobe, H., Sandewall, E. (eds.) Interactive Programming Environments, pp. 288–299. McGraw-Hill, New York (1984)
22. Czerwinski, M., van Dantzich, M., Robertson, G., Hoffman, H.: The Contribution of Thumbnail Image, Mouse-Over Text and Spatial Location Memory to Web Page Retrieval in 3D. In: Proc. INTERACT 1999, pp. 163–170. Springer, Heidelberg (1999)
23. Bruls, M., Huizing, K., van Wijk, J.: Squarified Treemaps. In: Proc. Eurographics and IEEE TCVG Symposium on Visualization, pp. 33–42. IEEE Press, Los Alamitos (2000)
24. Tu, Y., Shen, H.: Visualizing Changes of Hierarchical Data Using Treemaps. IEEE Trans. on Visualization and Computer Graphics 13(6), 1286–1293 (2007)