Supporting Window Switching with Spatially Consistent Thumbnail Zones: Design and Evaluation

Susanne Tak[1], Joey Scarr[1], Carl Gutwin[2], and Andy Cockburn[1]

[1] Computer Science and Software Engineering, University of Canterbury,
Private Bag 4800, Christchurch 8140, New Zealand
susanne.tak@pg.canterbury.ac.nz,
jls129@student.canterbury.ac.nz, andy@cosc.canterbury.ac.nz
[2] Department of Computer Science, University of Saskatchewan,
110 Science Place, Saskatoon, Saskatchewan, S7N 5C9, Canada
gutwin@cs.usask.ca

Abstract. Computer users switch between applications and windows all day, but finding the target window can be difficult, particularly when the total number of windows is high. We describe the design and evaluation of a new window switcher called SCOTZ (for Spatially Consistent Thumbnail Zones). SCOTZ is a window switching interface which shows all windows grouped by application and allocates more space to the most frequently revisited applications. The two key design principles of SCOTZ are (1) predictability of window locations, and (2) improved accessibility of recently and frequently used windows. We describe the design and features of SCOTZ, and present the findings from qualitative and empirical studies which demonstrate that SCOTZ yields performance and preference benefits over existing window switching tools.

Keywords: window switching, revisitation, spatial stability, predictability.

1 Introduction

Desktop computing involves constant window switching to navigate between various applications and documents. Previous work has found that people have more than eight windows open almost 80% of the time and that the average time between window switches is only 20.9 seconds [1].

Several window switching tools are available in most operating systems, and there are a large number of commercial and research tools that aim to enhance window switching performance. Our previous work identified two problems with current window switching interfaces [2]: first, many window switching interfaces lack spatial stability, meaning that the location of controls for acquiring particular windows can change over time, forcing users to rely on relatively slow visual search rather than rapid spatial decisions; second, most common window switching interfaces provide relatively weak support for strongly-exhibited patterns of window revisitation. We proposed spatial constancy and size morphing as design solutions to these problems, and our experiments using synthetic target acquisition tasks provided preliminary evidence of their success [2].

P. Campos et al. (Eds.): INTERACT 2011, Part I, LNCS 6946, pp. 331–347, 2011.

In this paper we extend our prior findings with the design and evaluation of a new window switcher called SCOTZ (Spatially Consistent Thumbnail Zones). SCOTZ uses application zones arranged in a treemap visualization to provide a spatially-stable and predictable window layout, and uses size morphing of the application zones to facilitate revisitation. It also provides quick access to recently used windows. SCOTZ's design is validated through qualitative and quantitative evaluations.

We make three specific contributions:

- a thorough description of our new window switcher SCOTZ and the rationale for its design choices;
- a summary of lessons learned from our qualitative study, which is useful for future iterations of SCOTZ as well being potentially informative for the design of other window switchers;
- an empirical study demonstrating the performance benefits of SCOTZ over two major commercial window switching interfaces (the Windows 7 Taskbar and Windows 7 Alt+Tab).

2 Related Work

2.1 Commercial Window Switching Interfaces

Three important commercial window switching applications are briefly reviewed below: the Microsoft Windows 7 Taskbar, Alt+Tab, and Mac OS X Exposé.

The *Microsoft Windows 7 Taskbar* is a narrow strip at the bottom of the screen which groups windows by application (see Fig. 1). Taskbar application icons can be *pinned* to the Taskbar so they are always in the same location and remain visible on the Taskbar even when there are no associated windows open. These pinned application controls remain in stable locations across sessions. However, the Taskbar provides no explicit support for switching to recent or frequently-used windows.

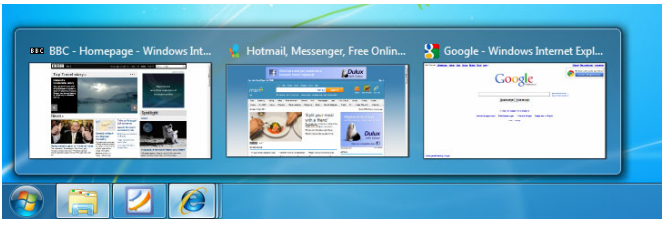

Fig. 1. The Microsoft Windows 7 Taskbar

Microsoft Windows 7 Alt+Tab (sometimes referred to as Windows *Flip*) is a modal window that is shown when the key combination Alt+Tab is pressed (see Fig. 2). The Windows 7 Alt+Tab window shows thumbnail previews of all windows: first, the top six windows in the *z*-order (see below), then a desktop preview, and then all remaining windows sorted by application name. *Z*-ordering is the depth-ordering of windows: a window that is placed in front of another window is relatively higher in the

z-ordering than the underlying windows. *Z*-ordering is similar to recency ordering, but sorting windows by *z*-order is spatially unstable: the ordering of the window representations will be different from switch to switch if the *z*-ordering of the windows changes. Also, it is unclear how well users understand and can anticipate *z*-order.

Fig. 2. Microsoft Windows 7 Alt+Tab

When activated, *Mac OS X Exposé* smoothly shrinks all windows so that they can be simultaneously viewed on the screen. The spatial location of each window in the overview is influenced by its most recent location on the screen. While this relative positioning may assist users in visually seeking windows in the overview, the locations are not spatially stable between invocations if the locations of the windows change, and consequently the locations are unpredictable.

2.2 Task-Based Window Switching Interfaces

Early studies of everyday computing [3] observed that computer users frequently switch between *tasks* such as writing a paper or programming. Many window switching interfaces have been developed to support *task management* by grouping windows by task. For example, a "writing a paper" task might contain a spreadsheet window, a statistical analysis package window, and a word-processing document into which the results are typed.

The Rooms system [4] is an early example of a window desktop management system that allows users to manually partition space for different tasks. Examples of task-based window managers that require *manual* grouping of windows into tasks are GroupBar [5], Activity-Based Computing [6], Scalable Fabric [7] and Task Gallery [8]. GroupBar and Activity-Based Computing use interfaces similar to the Windows Taskbar. With Scalable Fabric users can place groups of related windows in the periphery of the screen and switch to these groups of windows at once. Task Gallery uses a 3D visualization of task groups. The primary limitation of manual creation of task groups is that users must carry out explicit actions to gain potential benefits and it creates additional cognitive overhead for the user [9, 10].

Some window switching interfaces automatically identify tasks. SWISH [11] is an algorithm to automatically group windows into tasks based on their temporal and semantic relatedness. RelAltTab [12] modifies the Alt-Tab list to incorporate a system-generated list of related windows, similar to SWISH. Push-and-pull switching [13] automatically identifies window groups based on window overlap. 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 [14]. When this happens users must resort to time-consuming visual search of candidate targets.

WindowScape [15] uses a combination of automatic grouping by taking snapshots of the desktop (which can then later be revisited) as well as manual configuration of window miniatures on the desktop.

Another issue related to task-based grouping of windows is whether or not windows can be associated with multiple tasks at the same time. Some windows may not be 'task-specific' at all. For example, generic applications, such as a web browser or an e-mail client are likely to be used across tasks, rather than in one specific task. If a task-based window switcher only allows for exclusive grouping (i.e., a window can only be associated with one task, e.g. [5, 6, 8]) the user is forced to make 'impossible' choices about where a window should belong, or open multiple windows for these applications, which some users find unnatural or difficult (as one user stated, "[I'm] s*till trying to get used to having multiple internet windows open*" [5], p. 40). Using Activity-Based Computing [6], windows can be classified under more than one task, although the authors report on problems with achieving this in "a simple manner."

Last, another problem associated with task-based grouping of windows is that some windows might not be associated with any task at all. This is reported in [6]: "*The worst thing? Well* [...] *if you have to put everything into activities, then you need to constantly consider 'where does this one belong'. In many situations something just appears quickly and then you start up some application and do some things in it.* [...]" (p. 219).

Taskposé [16] uses a fuzzy approach to defining tasks, using spatial proximity to illuminate task-based window relationships – windows that are often temporally adjacent drift closer to one another and those that are temporally distant drift apart.

2.3 Studies of Window Switching

In this section, we provide an overview of studies related to window switching behaviour and window switching tools.

Window Switching Behaviour. Very early studies [17] of window switching showed that window switching is much more common that window creation, deletion and geometry management such as moving and resizing. Recent studies have showed the same: Hutchings et al. [18] found that the average time a window is active is only 20.9 seconds. In terms of *how* people switch between windows, previous work has found a relationship between monitor setup and window switching methods, with dual monitor users more likely to use a direct click on a window and less likely to use the Windows Taskbar, than single monitor users [2, 18]. In terms of *which* windows people switch to, previous work [2] has found that window switching follows an inverse exponential distribution, with 80% of window switches involving only 35% of windows.

Window Switching Interfaces. Analysis of the efficiency and effectiveness of current commercial methods for switching between windows is often anecdotal and sometimes conflicting. For example, some previous work labels the ordering of windows in Alt+Tab as "very effective" [19], but Alt+Tabbing is also labelled as "tedious" [20]. There are very few studies that evaluate the use, efficiency and/or effectiveness of the common commercially available window switching interfaces in

a formal manner. Alt+Tab is found to be relatively fast when the number of windows is small [21], but performance decreases as the number of windows increases. Users make more errors when using the Windows Taskbar than when using Mac's Exposé, which may be due to the smaller target sizes on the Windows Taskbar [21].

3 SCOTZ

In this section, we introduce our new window switcher SCOTZ (Spatially Consistent Thumbnail Zones)[1]. SCOTZ is designed to satisfy two main goals:

1. **Predictability of window locations.** Comprehensibility and predictability are important attributes of a user interface [14]. If a window switcher places window representations in predictable locations the user will able to correctly anticipate where a window representation is (or will be) placed, reducing the need for a costly visual search.
2. **Improved accessibility of frequently/recently used applications and windows.** Previous work [2] has identified strong revisitation patterns to applications and windows, so window switching interfaces should support rapid acquisition of recent or frequent targets.

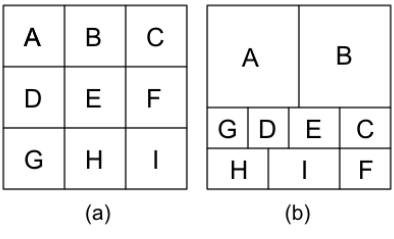

Fig. 3. Abstraction of the design of SCOTZ with nine application zones, labelled A to I: (a) before size morphing, (b) after several iterations of size morphing; applications A and B have been switched to frequently, and are therefore allocated more space, while keeping all application zones in relatively stable positions.

SCOTZ is a modal interface which groups windows by application in so-called *application zones*. Size morphing is applied to allocate more space to the most frequently switched-to applications. The basic concept behind SCOTZ is shown in Fig. 3, and Fig. 4 shows an actual SCOTZ window in full-screen mode with eight application zones and six windows. SCOTZ retains application zones' size and position even when the computer has been restarted.

The following sections describe the design of SCOTZ and explain how its features fulfil the two main goals above. Also, we describe four additional properties of the system: support for different display sizes, support for keyboard and mouse input, support for application launching, and options for personalization.

[1] A fully functional version of SCOTZ is available for download at http://www.cosc.canterbury.ac.nz/scotz

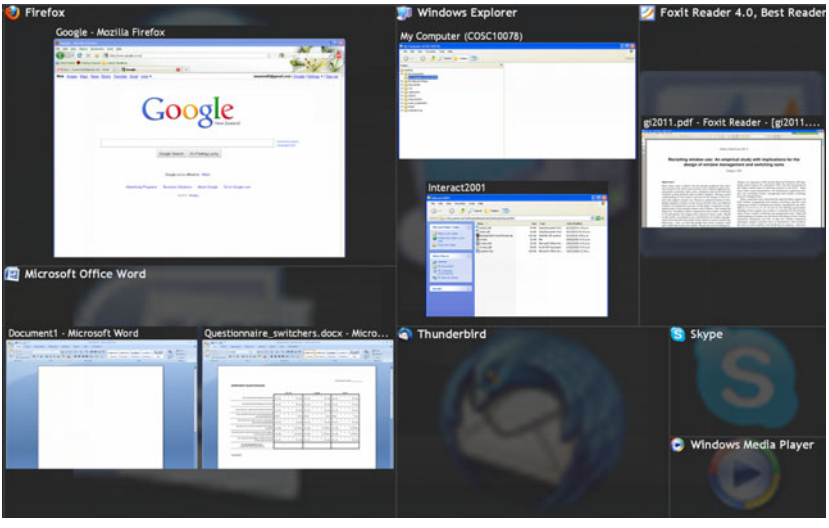

Fig. 4. SCOTZ in full-screen mode showing eight application zones and six windows

3.1 Predictable Window Locations

SCOTZ groups windows by application in *application zones*, which remain in stable locations. This results in comprehensible and predictable window locations. SCOTZ offers three different layouts for the application zones: two treemap layouts, where size morphing is applied, and a grid layout where no size morphing is allowed (see Fig. 5). A treemap is a space-filling layout that recursively divides the screen in rectangles, sized relative to some underlying data attribute [22]. In SCOTZ, the size of the application zone reflects how often an application has been switched to.

Various algorithms for generating treemaps exist, each offering advantages for specific contexts. For example, some treemaps are designed to keep items in relatively stable positions as the underlying data changes. An example of such a treemap is the spiral treemap [23] (see Fig. 5a), which preserves the ordering of the items and results in treemaps with relatively high stability – that is, item locations do not change a great deal when the underlying data changes. A squarified treemap [24] (see Fig. 5b) arranges items from top-left to bottom-right sorted by value, and creates a layout with items with very low aspect ratios. Low aspect ratios are not only attractive from an aesthetic point of view, but treemaps where the items have low aspect ratios can also be expected to lead to lower Fitts' Law targeting times than treemaps that have items that are long and narrow. However, the squarified treemap can be unstable in very early stages of use or when application revisitation patterns change a lot over time. Because of their favourable performance in terms of stability and aspect ratio, the spiral and squarified treemaps are good candidates to be used in SCOTZ.

Even though previous work has demonstrated that the slight instability of the layout caused by size morphing (i.e., items unavoidably move as they grow/shrink) does not harm user performance [2], SCOTZ also offers a grid layout (see Fig. 5c) where no size morphing is applied, and is therefore very stable.

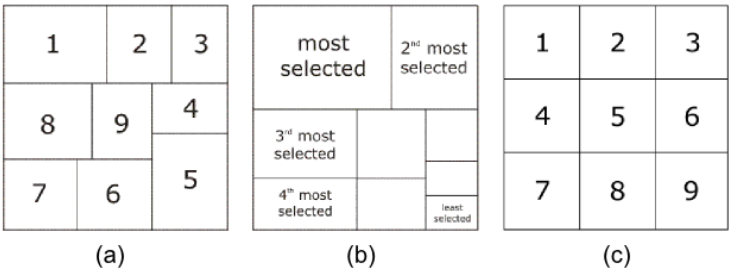

Fig. 5. Abstraction of the three layout options for application zones in SCOTZ: (a) spiral treemap, where applications are (arbitrarily) sorted once and this ordering is retained forever, and where zones are laid out in order in a clockwise spiral from top left; (b) squarified treemap, where application zones are sorted by size and laid out from top left to bottom right; (c) grid, where all application zones are of equal size and are laid out in a grid pattern.

The ordering of the window representations is either alphabetic by window title, by frecency (a combination of frequency and recency), or by the order in which the respective windows were opened, using a row-major order. Alphabetic ordering is very predictable and understandable; sorting by frecency supports window revisitation; and sorting the representations by the order in which the respective windows were opened is very stable when additional application windows are opened.

3.2 Improved Accessibility of Frequently Used Windows

SCOTZ allocates more space to the most frequently switched-to applications, which reduces the Fitts's Law [25] targeting time of these application zones, as well as making them easier to find [26].

3.3 Improved Accessibility of Recently Used Windows

When SCOTZ is bound to Alt+Tab, pressing Alt+Tab will bring up the SCOTZ interface as normal, but repeated presses of Tab will cycle through the windows according to *z*-order, similar to the implementation of Microsoft Windows Alt+Tab (see Fig. 6). This provides quick and easy access to recently used windows.

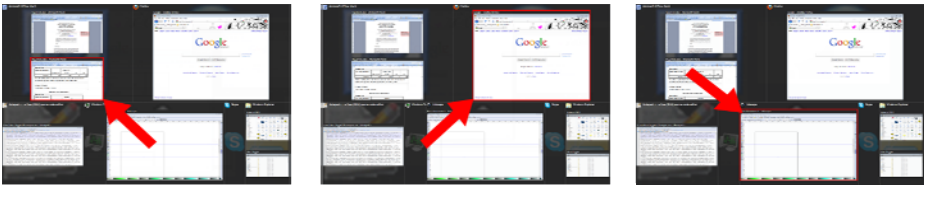

Fig. 6. SCOTZ Using Alt+Tab to cycle through the windows in SCOTZ by z-order. A pulsating red border indicates the currently selected window (indicated by a red arrow in this figure).

3.4 Support for Various Display Sizes

Modern display sizes range from very small (netbooks) to very large or multi-monitor setups. To accommodate this wide range of display sizes, SCOTZ can either be a full screen window (see Fig. 4), a smaller window in the centre of the screen, or a smaller window positioned under the mouse cursor (see Fig. 7). As the design of SCOTZ is aimed at keeping the application zones in spatially stable locations, positioning the SCOTZ interface relative to the mouse cursor means that a target can always be acquired with the same mouse gesture. Previous work has shown that despite some difficulties in the learning/training phase, mouse gestures can be very efficient and accurate [27].

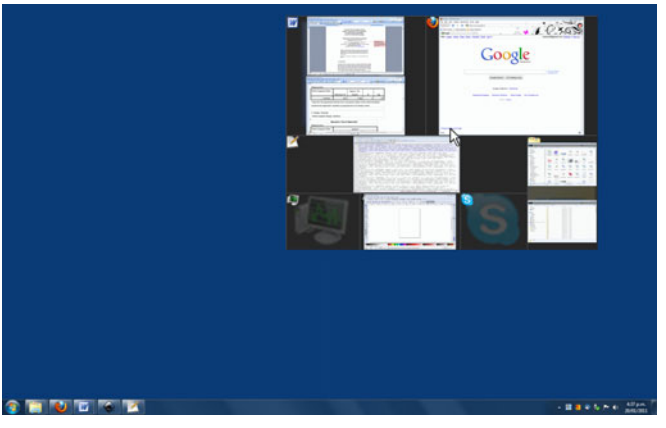

Fig. 7. SCOTZ in a smaller display mode, positioned under the mouse cursor

3.5 Support for Keyboard and Mouse Input

Previous work has shown that while most users prefer mouse-based methods for window switching, such as the Windows Taskbar, there is a small but significant subset of users who prefer keyboard-based methods, such as Alt+Tab [2]. Therefore, SCOTZ has rich options for both keyboard and mouse input.

3.6 Support for Application Launching

SCOTZ provides a single interface mechanism for both window switching and application launching. After having used SCOTZ for a long period of time, we expect users to be familiar with the locations of application zones, and if an application has no open windows associated with it, the application zone is still visible in SCOTZ (for example, see the Skype, Thunderbird and Window Media Player zones in Fig. 4). Therefore, it makes sense for these zones to double as efficient application launchers. Clicking on an empty application zone launches the application.

3.7 Options for Personalization

Personalization of an interface is driven by a variety of motivations [28], such as the personal goals of the user, accommodation for individual differences, or personal preference. Users can make several functional personalizations in SCOTZ to accommodate their personal goals (in addition to choices about layout and input methods, as presented in previous sections): for example, users can include or exclude certain applications in SCOTZ, and can customize the rate of growth for application zones. The appearance of SCOTZ is also customizable: users can change the font size and type to accommodate various levels of visual acuity, and the colour scheme to accommodate for various types of colour blindness as well as personal taste and preference. Also, users can adjust the opacity of SCOTZ (see Fig. 8).

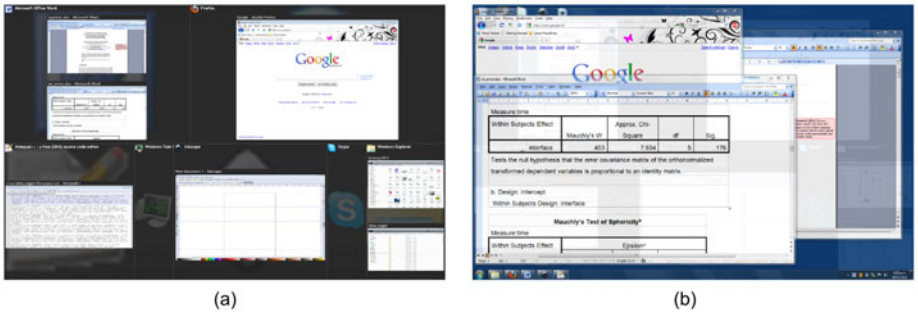

Fig. 8. Different opacity levels of the SCOTZ window, from (a) opaque to (b) almost transparent

4 Qualitative Study

A beta version of SCOTZ was given to five volunteers, who were asked to use it for at least several days. Next, they were given a questionnaire and were interviewed. Based on the results, we identified five main observations:

1. **People did not notice the slight location changes of application zones when using the spiral treemap layout.** Participants commented on never noticing *any* location changes of the application zones when the spiral treemap layout was used, even though gradual changes will have occurred as zones grew and shrunk.
2. **No clear preference for either the spiral or the squarified treemap layout.** The spiral layout was the default layout for SCOTZ, but some participants did (temporarily) switch to the squarified layout. However, there is no consensus on which layout is preferred. Some participants regarded the squarified layout as being too unstable (i.e., the application zones move too much), while others really liked the squarified layout and clearly preferred it over the spiral layout.
3. **Even users that mainly used Alt+Tab appreciated the size morphing of the application zones.** Though the size morphing of the application zones does not seem to have direct benefits for people that mainly use Alt+Tab for switching

between windows, participants commented that it helped them to guide their attention towards the application they were aiming for.

4. **The application launching functionality was only used by some participants, but it did not bother those who did not use it.** The option to use SCOTZ as an application launcher was only used by some users, yet this functionality did not bother non-users. If anything, retaining the application zones whilst no windows are open enhances the spatial stability of SCOTZ's layout.
5. **Overriding existing mappings such as Alt+Tab is useful, but risky.** Because SCOTZ was bound to Alt+Tab by default and SCOTZ retained Alt+Tab's (*z*-order) functionality, SCOTZ could be used without any additional learning. However, cycling to the correct application using SCOTZ (instead of clicking on the zones/thumbnails with the mouse) can be confusing, because it is harder to keep track of the selected item (also see Fig. 6). Possible solutions are (1) mapping SCOTZ under another key combination, (2) not retaining the *z*-ordering, but picking an ordering that matches the layout of SCOTZ better, or (3) providing better feedback on the *z*-ordering (e.g., with a small strip at the bottom of the screen showing the full order).

5 Lab Study

We performed a lab study to empirically compare the performance of SCOTZ, the Microsoft Windows 7 Taskbar, and Alt+Tab in a controlled environment. We chose the Taskbar and Alt+Tab for comparison because (1) Microsoft Windows is the most commonly used operating system, and therefore a comparison with the tools available in Windows 7 is relevant, and (2) these two tools present a *challenging* condition to compare SCOTZ against in terms of the key design principles of SCOTZ. The Windows 7 Taskbar places application icons in stable locations (unless a program is closed and re-opened), and Windows Alt+Tab provides explicit support for switching back to recently used windows because it places (the first six) window representations by their place in the z-ordering of windows, which is very similar to a recency ordering. We considered including Mac Os X Exposé as another comparison point, but excluded it because (although beautiful) its target acquisition time necessarily includes visual search time. In Exposé items do not appear in spatially stable locations; Exposé's representation of available windows is altered whenever windows are moved, opened, or closed, so users can never be certain where their target windows will appear. Several previous studies (including [2]) demonstrate that this type of spatial instability reduces target acquisition performance.

5.1 Method

In the experiment, participants were presented with a set of windows in Microsoft Windows 7, such as a word processor with several documents open, an e-mail application, a game, a video player, etc. Participants completed a series of tasks in which they had to switch to a particular window using the Windows 7 Taskbar, Windows 7 Alt+Tab, or SCOTZ in a successive series of tasks (see Fig. 9). In each condition, participants were instructed to use one particular switching tool *exclusively*.

Some windows were prompted often, while others were hardly ever prompted, following the findings reported in [2]: 80% of switches were to 35% of windows.

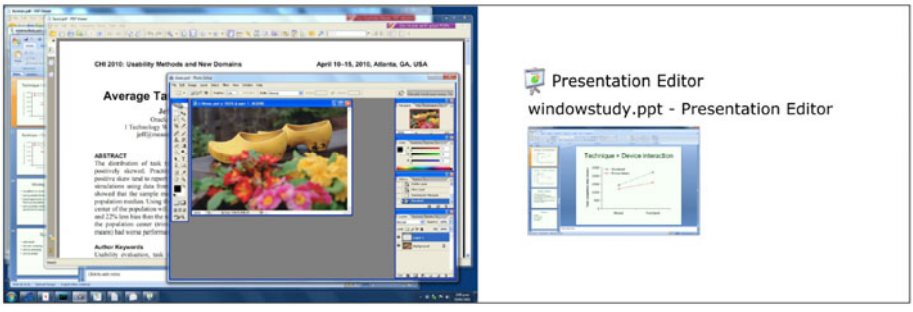

Fig. 9. The experimental interface: all windows are on the primary screen on the left, and the current task is on the secondary screen on the right

5.2 Design

Switching time and *errors* (switching to a non-target window) were measured across three levels of the independent variable *interface* (Taskbar, Alt+Tab and SCOTZ), and analysed using a one-way repeated-measures ANOVA. The experiment used a within-subject design and the order in which the conditions were presented to the participants was counterbalanced.

5.3 Procedure

At the start of each condition participants were given a verbal explanation and a demo of the window switching tool used in that particular condition. Participants then performed a series of 20 practice tasks with the window switcher before starting the experimental tasks.

At the start of each task, all windows were temporarily hidden from view and the user was prompted to press a 'Next' button at the centre of the secondary screen. Pressing this button revealed the next target window on the secondary screen (by showing the application icon, the window title and a window preview thumbnail of the target). Next, participants were prompted to click a 'Start' button in the centre of the primary screen, after which all the windows were unhidden, and participants then had to switch to the target window. If the participant switched to the incorrect window nothing happened. In total, participants performed 80 tasks in each condition (excluding the practice tasks).

After each condition, the participant filled out a short questionnaire regarding the window switching tool that had just been used.

5.4 Software and Hardware

All the content of the windows used in the experiment was non-modifiable to minimize distraction, to prevent accidental interaction with the windows, and to allow for

consistent window previews in the window switching tools throughout the tasks. To prevent learning effects across conditions, three different window sets were generated, all with unique applications and windows. The window sets were counterbalanced across the three conditions. Each window set contained 8 applications and 15 windows. For example, one of the window sets contained a PDF reader (with 4 windows open), a photo editor (3 windows), a presentation editor (2 windows), an HTML editor (2 windows), a music player, an email application, a command prompt, and a card game.

No window icons that are already in use by well-known applications were used, to ensure that all participants started off with equal knowledge about the application icons. This is particularly important for the evaluation of the Windows 7 Taskbar, which shows only application icons.

The full-screen version of SCOTZ using the squarified treemap algorithm was used. All application zones in SCOTZ were fixed (i.e., did not change during the experiment) and were pre-set to reflect the various frequencies with which applications/windows were switched to during the experiment.

A mouse with an extra side button was used in the experiment, and this side button was used to invoke SCOTZ.

5.5 Questionnaire

We used the NASA Task Load Index (NASA-TLX)[2] to assess perceived workload in each of the conditions. Two more questions were added to assess the perceived ease of learning to operate the window switcher (*operation*) and the perceived ease of learning window locations (*location learning*) in the window switcher. Also, participants were asked to rank the three window switching interfaces from most to least preferred.

5.6 Participants

Twelve people, all university students, participated in the experiment. Age ranged from 20 to 35 years old (mean 27); two participants were female. All participants were experienced computer users: computer use was at least 30 hours per week for each of the participants. Participants were reimbursed with a shopping voucher. The experiment took approximately 40 minutes to complete.

5.7 Results

Selection Times. We analysed the mean time to switch to a window for each of the methods (see Fig. 10). The results for the Windows 7 Taskbar are split by Taskbar button (for applications with only one associated window) and Taskbar thumbnail (for applications with more than one associated window, where the user first has to select the application icon and *then* the window in the fanned out sub-menu, see Fig. 1).

[2] http://humansystems.arc.nasa.gov/groups/TLX/downloads/TLX.pdf

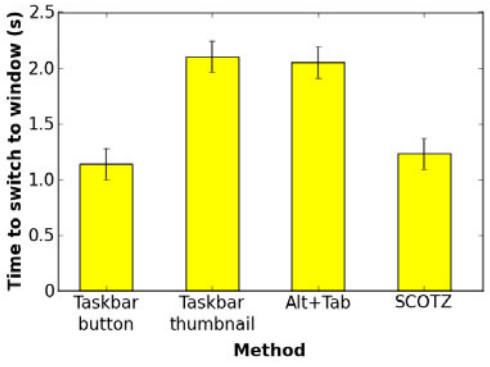

Fig. 10. Window selection times for the various methods. Error bars denote the 95% within-subject *CI* [29].

Mean window switching times when using a Taskbar button, a Taskbar thumbnail, Alt+Tab and SCOTZ are 1.1s, 2.1s, 2.1s and 1.2s, respectively, giving a significant effect of *interface*: ($F_{3,33}$=53.3, p<.001). Post hoc analysis (Bonferroni correction, α=.05) reveals pairwise differences between all tools (all p <.001) except the Taskbar button and SCOTZ, and the Taskbar thumbnail and Alt+Tab.

By design, some of the target windows in the experiment were high up in the Alt+Tab ordering, and others further down, following a nearly uniform distribution across all possible positions. A detailed analysis of window switching times when Alt+Tab is used is shown in Fig. 11, which shows the selection times for Alt+Tab ranked by position of the target window in the Alt+Tab ordering, and split by input method (using the keyboard to sequentially step through the list of thumbnails, or using the mouse to click on the target thumbnail). Three observations are apparent from Fig. 11: (1) window selection time when using Alt+Tab with mouse input is relatively constant across positions of the target thumbnail, (2) window selection time when using Alt+Tab with keyboard input increases linearly as the position of the target thumbnail in the list of windows becomes higher (r=.963, p<.01), and (3) Alt+Tab with keyboard input is very efficient for switching back to the previously used window (position 1 in the ordering); the mean window switching time is 0.9 seconds for this particular type of window switch, which is shorter than the mean switching times for both the Taskbar and SCOTZ. However, this performance benefit quickly disappears when the target window is further down the list of windows.

Errors. Mean error rates for Taskbar button, Taskbar thumbnail, Alt+Tab and SCOTZ are 0.8%, 5.8%, 2.8% and 2.7%, respectively. The difference between these switching times is significant ($F_{3,33}$=5.2, p<.01). Post hoc analysis (Bonferroni correction, α=.05) reveals a pairwise difference between the Taskbar button and Taskbar thumbnail (p< .05).

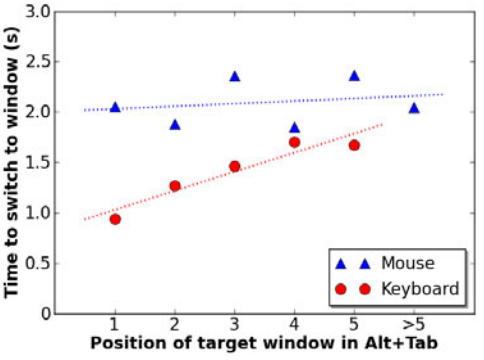

Fig. 11. Window selection times for Alt+Tab sorted by position of the target window in Alt+Tab and split by mouse and keyboard input. Participants almost never used the keyboard to select a window further than position 5 in the Alt+Tab ordering, hence there is no (reliable) data for this value.

Subjective Measures. The NASA-TLX worksheet results showed significantly different ratings for *mental demand*, *effort*, *location learning*, *frustration* (Friedman test, all $p<.001$), and *operation* ($p<.01$), also see Fig. 12. Post hoc pairwise comparisons (Bonferroni correction, $\alpha=.05$) reveal significant differences between Alt+Tab and SCOTZ on all five aforementioned factors, between the Taskbar and Alt+Tab on all these factors except *frustration*, and between the Taskbar and SCOTZ on the *mental demand* and *location learning* factors. All participants preferred SCOTZ the most, and 9 out of 12 participants preferred Alt+Tab the least (i.e., 3 out of 12 participants preferred the Taskbar the least).

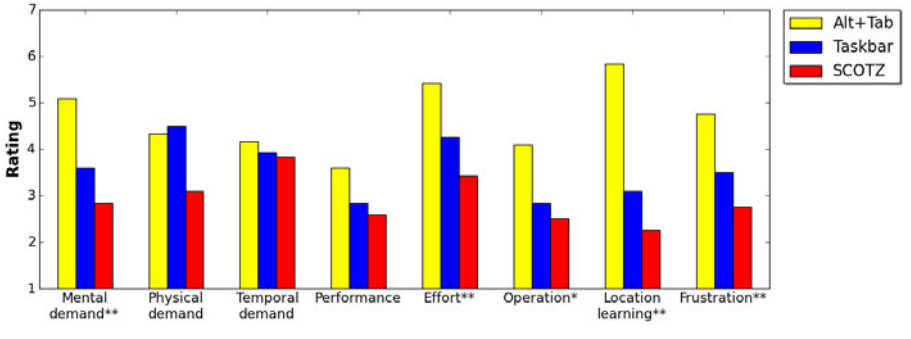

Fig. 12. Questionnaire results; lower ratings are better. * Difference is significant, $p<.01$. ** Difference is significant, $p<.001$.

5.8 Discussion

Window Switching Times. SCOTZ was faster than Taskbar thumbnails and Alt+Tab. Although there was no significant difference between SCOTZ and Taskbar buttons,

Taskbar buttons are not available when there is more than one window associated with the application (forcing users to resort to Taskbar thumbnails).

Subjective Measures. All participants ranked SCOTZ as the most preferred tool. Users perceived SCOTZ as less mentally demanding, costing less effort, and less frustrating than Alt+Tab, as well as finding it easier to learn to operate and to learn item locations in SCOTZ compared to Alt+Tab. Also, window locations in SCOTZ were perceived as easier to learn that those on Taskbar, and SCOTZ was perceived as less mentally demanding than the Taskbar. These two factors are likely related: SCOTZ is less mentally demanding because it is easier to learn locations, thereby reducing the cognitive burden for users. It is interesting that users found locations in SCOTZ easier to learn than locations of items on the Taskbar, as in both cases these were completely stable in the current study.

Alt+Tab. Overall, Alt+Tab was unpopular, with 75% of participants ranking it as least preferred. Alt+Tab was also judged to be more mentally demanding and costing more effort than the Windows Taskbar. Last, users found it harder to learn to operate and learn item locations in Alt+Tab than with the Taskbar. One participant commented that he/she "hated how Alt+Tab icons moved around". However, these results for Alt+Tab might have been negatively influenced by the fact that users had to use Alt+Tab for *all* window switches in the experiment. Our results show that Alt+Tab is very efficient for switching back to the most recently used window, with this particular type of switch outperforming both the Taskbar and SCOTZ. One participant commented "I use Alt+Tab to switch between the most recent windows, and other methods for older windows." Such a 'mixed approach', i.e. using Alt+Tab to switch back to the most recently used window, but another method for other types of window switches might lead to higher user satisfaction than the 'enforced' use of Alt+Tab for all window switches that was the case in the experiment.

Comparison to other window switching tools. Our experiment compared user performance with SCOTZ against that with the Windows 7 Taskbar and Alt+Tab. Further work is needed to compare SCOTZ performance with that of the wide range of research and commercial tools reviewed in Section 2. However, we believe SCOTZ's key design goals – supporting spatially stable means for acquiring windows and applications, and providing support for rapidly retrieving frequently and recently retrieved windows – are important for enabling high performance window acquisition. In particular, inconstant spatial locations are likely to force users to resort to time consuming visual search (to seek a target) or decision making (to calculate the effect of an algorithm, for example).

6 Conclusions and Future Work

While previous work has found that window revisitation is very common, no tools developed so far explicitly support this revisitation. We used this finding to inform the design of a new window switcher called SCOTZ, which supports window revisita-

tion by increasing the size of the most switched-to applications, and keeping them in positions that are as stable as possible.

Our lab study demonstrates the performance benefits of SCOTZ over two common window switching tools: the Microsoft Windows 7 Taskbar and Alt+Tab. This study also generated valuable insights regarding the most recent window switching tools available in Microsoft Windows 7.

More research into the suitability of SCOTZ for Alt+Tab users could shed more light on how these users can best be supported in their window switching activities. Interestingly, even Alt+Tab users reported benefits from the size morphing of the application zones in SCOTZ, but retaining the Alt+Tab order in SCOTZ did confuse these users. Ideally, SCOTZ should retain the rapid back-and-forth switching between two windows that Alt+Tab offers (see results of the lab study) while also assisting users in finding items further down the Alt+Tab ordering.

Finally, we are developing a new treemap algorithm to better suit SCOTZ than existing algorithms, in terms of enhanced spatial stability of the application zones.

References

1. Hutchings, D.R., Stasko, J.: Revisiting display space management: understanding current practice to inform next-generation design. In: Proc. of GI 2004, pp. 127–134. Canadian Human-Computer Communications Society (2004)
2. Tak, S., Cockburn, A., Humm, K., Ahlström, D., Gutwin, C., Scarr, J.: Improving window switching interfaces. In: Gross, T., Gulliksen, J., Kotzé, P., Oestreicher, L., Palanque, P., Prates, R.O., Winckler, M. (eds.) INTERACT 2009. LNCS, vol. 5727, pp. 187–200. Springer, Heidelberg (2009)
3. Bannon, L., Cypher, A., Greenspan, S., Monty, M.L.: Evaluation and analysis of users' activity organization. In: Proc. of CHI 1983, pp. 54–57. ACM Press, New York (1983)
4. Henderson, D.A., Card, S.: Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. ACM Trans. Graph. 5(3), 211–243 (1986)
5. Smith, G., Baudisch, P., Robertson, G., Czerwinski, M., Meyers, B., Robbins, D., Horvitz, E., Andrews, D.: Groupbar: The taskbar evolved. In: Proc. of OzCHI 2003, pp. 34–43 (2003)
6. Bardram, J., Bunde-Pedersen, J., Soegaard, M.: Support for activity-based computing in a personal computing operating system. In: Proc. of CHI 2006, pp. 211–220. ACM Press, New York (2006)
7. Robertson, G., Horvitz, E., Czerwinski, M., Baudisch, P., Hutchings, D., Meyers, B., Robbins, D., Smith, G.: Scalable fabric: Flexible task management. In: Proc. of AVI 2004, pp. 85–89. ACM Press, New York (2004)
8. Robertson, G., van Dantzich, M., Czerwinski, M., Hinckley, K., Thiel, D., Robbins, D., Risden, K., Gorokhovsky, V.: The task gallery: A 3D window manager. In: Proc. of CHI 2000, pp. 494–501 (2000)
9. Kaptelinin, V.: Umea: translating interaction histories into project contexts. In: Proc. of CHI 2003, pp. 353–360. ACM, New York (2003)
10. Dragunov, A.N., Dietterich, T.G., Johnsrude, K., McLaughlin, M., Li, L., Herlocker, J.L.: Tasktracer: a desktop environment to support multi-tasking knowledge workers. In: Proc. of IUI 2005, pp. 75–82. ACM, New York (2005)

11. Oliver, N., Smith, G., Thakkar, C., Surendran, A.: Swish: Semantic analysis of window titles and switching history. In: Proc. of IUI 2006, pp. 194–201. ACM Press, New York (2006)
12. Oliver, N., Czerwinski, M., Smith, G., Roomp, K.: Relalttab: assisting users in switching windows. In: Proc. of IUI 2008, pp. 385–388. ACM, New York (2008)
13. Xu, Q., Casiez, G.: Push-and-pull switching: window switching based on window overlapping. In: Proc. of CHI 2010, pp. 1335–1338. ACM, New York (2010)
14. Shneiderman, B.: Direct manipulation for comprehensible, predictable and controllable user interfaces. In: Proc. of IUI 1997, pp. 33–39. ACM, New York (1997)
15. Tashman, C.: Windowscape: A task oriented window manager. In: Proc. of UIST 2006, pp. 77–80. ACM Press, New York (2006)
16. Bernstein, M., Shrager, J., Winograd, T.: Taskposé: exploring fluid boundaries in an associative window visualization. In: Proc. of UIST 2008, pp. 231–234. ACM, New York (2008)
17. Gaylin, K.B.: How are windows used? Some notes on creating an empirically-based windowing benchmark task. In: Proc. of CHI 1986, pp. 96–100 (1986)
18. 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. of AVI 2004, pp. 32–39. ACM Press, New York (2004)
19. de Chiara, R., Erra, U., Scarano, V.: A visual adaptive interface to file systems. In: Proc. of AVI 2004, pp. 366–369. ACM, New York (2004)
20. Grudin, J.: Partitioning digital worlds: Focal and peripheral awareness in multiple monitor use. In: Proc. of CHI 2001, pp. 458–465 (2001)
21. Kumar, M., Paepcke, A., Winograd, T.: Eyeexpose: Switching applications with your eyes. Tech. rep. (2007)
22. Shneiderman, B.: Tree visualization with tree-maps: 2-d space-filling approach. ACM Trans. Graph. 11(1), 92–99 (1992)
23. Tu, Y., Shen, H.: Visualizing changes of hierarchical data using treemaps. IEEE Transactions on Visualization and Computer Graphics, 1286–1293 (2007)
24. Bruls, M., Huizing, K., Wijk, J.v.: Squarified treemaps. In: Proceedings of Joint Eurographics and IEEE, pp. 33–42. IEEE Press, Los Alamitos (2000)
25. Fitts, P.M.: The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology 47, 381–391 (1954)
26. Wolfe, J.M., Horowitz, T.S.: What attributes guide the deployment of visual attention and how do they do it? Nature Reviews Neuroscience 5(6), 495–501 (2004)
27. Dulberg, M.S., Amant, R., Zettlemoyer, L.: An imprecise mouse gesture for the fast activation of controls. In: Proc. of INTERACT 1999, pp. 375–382 (1999)
28. Blom, J.: Personalization: a taxonomy. In: Proc. of CHI 2000 Extended Abstracts, pp. 313–314. ACM, New York (2000)
29. Masson, M.E.J., Loftus, G.R.: Using confidence intervals for graphically based data interpretation. Canadian Journal of Experimental Psychology 57(3), 203–220 (2003)