WindowScape: A Task Oriented Window Manager

Craig Tashman
Georgia Institute of Technology
GVU Center
Atlanta, GA 30332,USA
craig@cc.gatech.edu

ABSTRACT
We propose WindowScape, a window manager that uses a photograph metaphor for lightweight, *post hoc* task management. This is the first task management windowing model to provide intuitive accessibility while allowing windows to exist simultaneously in multiple tasks. WindowScape exploits users' spatial and visual memories by providing a stable thumbnail layout in which to search for windows. A function is provided to let users search the window space while maintaining a largely consistent screen image to minimize distractions. A novel keyboard interaction technique is also presented.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: Scaling, window management, task management, visual search, spatial memory.

INTRODUCTION
Throughout the history of personal computing, visual workspace management has been a problem. Difficulties in managing screen real estate on personal computers were recognized as early as 1983 [1]. The dearth of space provided by these small screens was one of the motivating factors for Alan Kay's initial use of overlapping windows [3]. Today we see virtual desktop managers included as standard features of many Linux and Unix GUI's, and numerous such utilities are readily available for Windows and Macintosh.

Although the increasing economic viability of multi-monitor workstations alleviates some of the workspace management problems caused by small displays, they only address part of the problem. Mobile knowledge workers depend on devices like laptops and PDA's, where screen size is limited by the need to minimize device footprint. And surprisingly, even those users for whom multiple monitors are feasible do not necessarily prefer them, sometimes choosing instead to use virtual desktop managers [6]. But even if all users possessed physically large display surfaces, larger displays may prompt users simply to keep more documents open [7]. Using modern overlapping window management systems, very large displays may face the same organization problems that occur on physical desks when they are covered with many documents.

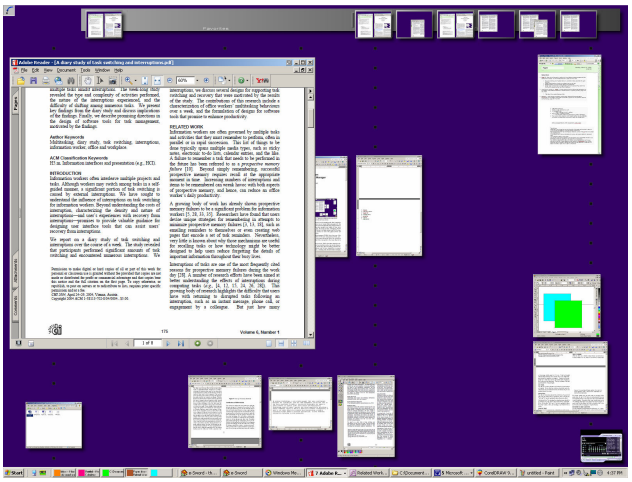

Figure 1: Several miniaturized, thumbnail windows and one full size window.

Virtual desktop managers (VDMs) have been one of the popular solutions to the space management problem, operating on the observation that people tend to use windows in groups. [1] VDMs allow users to create and switch among groups of windows explicitly. But in spite of this success, VDMs and other alternative window managers have limitations in the flexibility of the grouping mechanisms they provide, and the means offered for finding particular windows. To address these limitations, we have developed WindowScape, a zooming, task-oriented window manager. WindowScape uses photograph and history metaphors for its window grouping system, providing a lightweight mechanism for grouping windows that are often used together, and for allowing windows to reside in multiple groups simultaneously. Although the term *task* can mean various things, here we use the term to refer to these window groups. We do so since end users appear to accept the idea of equating groups of windows that are used together

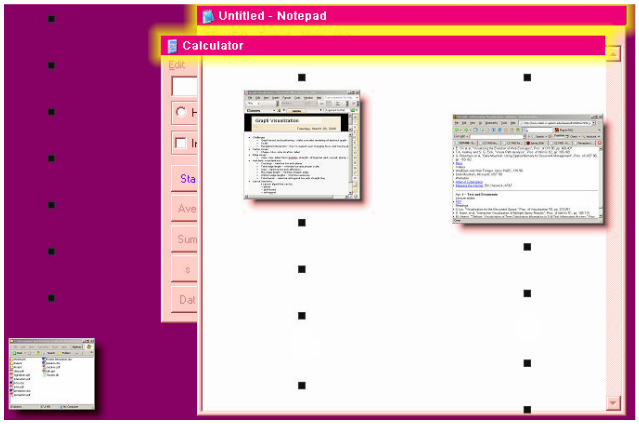

Figure 3: Two overlapping windows with their title bars and all miniaturized windows brought to the top of the z-order.

with tasks [7], and since studies of VDM usage show window groups often corresponding to tasks and subtasks [6]. In addition to task management, WindowScape provides a spatially stable, readily accessible thumbnail layout to allow users to search for windows by their appearance and leverage spatial memory in recalling locations (Figure 1).

RELATED WORK

Modern Virtual desktop managers can be traced back to Henderson and Card's *Rooms* [2]. VDMs vary considerably, but most require windows to be located in exactly one desktop. Other work such as *Scalable Fabric* [7], *Kimura* [4], and *GroupBar* [8] offer users very different systems with which to group their windows. But even these systems, which do not require windows to be in any group at all, still do not allow for windows to be in multiple groups at the same time. A disadvantage faced by these VDMs is that requiring windows to be in a single group forces users to decide ahead of time where a new window belongs. Likewise, such a single-group approach is inflexible, in that it neglects the possibility that some windows might naturally be useful in several different groups.

A partial exception to this limitation was provided by the original Rooms system. Although Rooms did not allow windows to exist simultaneously in multiple groups, it did provide an abstraction known as a *window placement* to be in multiple groups simultaneously. A window placement represented a particular window at a particular position; Rooms allowed users to copy window placements from one group (or Room) to another. The disadvantage to this mechanism is that there is no clear analog to such an abstraction in the real world. Even in other computer environments, copying something generally produces a completely independent duplicate, whereas copied placements all refer to the same underlying window. Thus, we sought to provide a more intuitive means of allowing windows to occupy multiple groups simultaneously.

Another limitation of VDMs is the strict separation they impose between window groups [7]. For example, VDMs make it difficult to interact with windows from multiple groups at the same time. One can move all of the desired

Figure 2: The timeline (right) of desktop states shown as a series of photograph-like snapshots. The left is the list of favorite snapshots, the favorites bar.

windows to the same group, but doing so alters the grouping structure; it also requires the user to remember to return the windows to their proper groups afterward. Like Scalable Fabric and GroupBar, WindowScape allows one to interact with windows from multiple groups at once, and without affecting the grouping structure [7, 8].

Tools such as Scalable Fabric, GroupBar and Kimura present the user a layout of window representations [7, 8, 4]. Scalable Fabric and Kimura lay out thumbnail views of the user's windows on the desktop; users can expand or otherwise change those thumbnails into the actual windows they represent. GroupBar lays out representations as well, but it represents windows with buttons, which can only be positioned in rows or columns along the edges of the screen. However, all of these systems relate the position of a representation to the group in which it resides. This relationship precludes the user from spatially organizing the windows in other ways, such as with respect to time, or so as to facilitate keyboard navigation among the representations.

Another property of VDMs, Scalable Fabric, GroupBar and similar systems is that the user must explicitly choose the grouping structure. Some systems, such as GroupBar and Scalable Fabric, allow the user to create groups, destroy groups, and specify what a group contains far more easily than with a VDM. Still, the user is forced to choose exactly where a window belongs, how many groups the windows should be divided into, etc. in order to leverage the benefit of window grouping.

A differentiating feature of WindowScape is its use of a timeline as part of its window group management system. WindowScape is not the first system that allows users to manage their activities based on their histories; prior work includes Kimura and Rekimoto's *Time-machine Computing* [4, 5]. However, these prior systems have generally focused on solving a different problem than WindowScape. Kimura, for example, focuses on workspace management with special, large, focus-plus-context display surfaces [4]. Time-machine Computing is more a file manager than a window manager.

WINDOWSCAPE OPERATION

In order to consider how WindowScape addresses the above limitations with prior work, we will first explain WindowScape's general operation, and draw some additional distinctions with other approaches. In the Task Management subsection we will revisit the above limitations and suggest how WindowScape addresses them.

Window Management

WindowScape represents windows as small thumbnails that we refer to as *miniatures*. These miniatures are positioned on the desktop by the user; to ensure spatial stability, they are never moved automatically. The miniatures can be in-

dependently expanded and re-miniaturized, allowing the user to display just the windows needed at that moment (Figure 1). When expanded, windows can be moved about without affecting where they will go the next time they are miniaturized. In order to provide users with maximal context, the location to which windows expand depends on which other windows are expanded as well. Specifically, and in contrast to Scalable Fabric, when a window is expanded WindowScape puts it in the same location as the last time it was expanded with the same group of other windows, whether that group is explicitly recognized as a task by the user or not. Once expanded, windows can be re-miniaturized individually or in parallel; WindowScape also provides a function for miniaturizing, and positioning the resulting miniature, in one mouse stroke.

Finding Obscured Windows

By default, miniatures remain below expanded windows in the z-order. If the user wants to view the miniatures being obscured by an expanded window, she must either drag the occluding window aside or miniaturize it in order to see the miniatures beneath it. While this is a problem for expanded windows which routinely cover one another, the problem is especially acute for miniatures, which are always last on the Windows z-order. To mitigate this problem without the distraction of dramatically changing the overall screen image, we added a feature that allows the user to bring all miniatures to the top of the z-order, as well as the title bars of all expanded windows. The user simply drags the cursor over the desktop background and all miniatures and title bars appear, while everything else tints red to make the miniatures visually stand out (Figure 3). When the mouse is released, the display returns to normal, and if the mouse was over a miniature or a title bar, it is expanded or its window is brought to the front respectively.

Like *WindowScape*, *Scalable Fabric* allows users to bring all window thumbnails to the front [7]. However, *Scalable Fabric* does not provide comparable mechanisms for helping users search through their already-expanded windows. Besides *Scalable Fabric*, we are not aware of any other systems that let users visually search among their windows while maintaining the user-defined spatial relationships during the search.

Keyboard Navigation

When no windows are expanded, users can navigate among the miniatures by keyboard. Mapping the four directional keys on standard keyboards to transitions among arbitrarily located items (such as the miniatures) involves a tradeoff. If we use an algorithm that devises intuitive mappings, then some miniatures may be unreachable (Figure 4a, where, starting from item A, intuitive transitions would leave item E unreachable). Alternately, we could guarantee reachability, but leave some transitions counter-intuitive. This type of algorithm is used in navigating among icons on the Windows desktop, and we experimentally found to be the case in *Scalable Fabric* (Figure 4b, showing the directional-key transitions among five thumbnails). Note how in figure 4b,

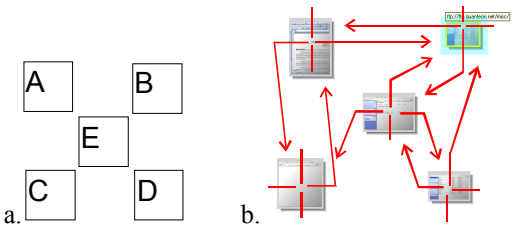

Figure 4: (a) A representation of 5 miniatures. (b) Five thumbnails in Scalable Fabric with arrows showing the keyboard transitions between them.

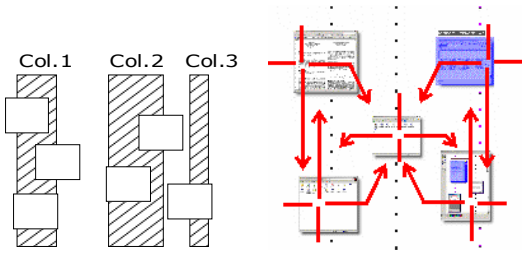

Figure 5: (a) How several miniatures would be grouped into columns. (b) Five miniatures in WindowScape with arrows showing the keyboard transitions between them.

pressing the 'right' directional key when the bottom left thumbnail is selected will select the top left thumbnail. Our solution to this problem was to use a simple mapping algorithm where we could visually represent what the transitions would be among the miniatures.

WindowScape's keyboard navigation algorithm groups all miniatures into columns. It begins with the leftmost miniature M_1, and proceeds right, grouping any miniatures that overlap with M_1 into the same column. This process is repeated for all remaining miniatures, resulting in a columnar grouping (Figure 5a). To navigate among the miniatures, the left/right arrow keys change the active column, and the up/down arrow keys change the selected miniature within that column (Figure 5b, showing directional key transitions among 5 miniatures). When moving from column C_1 to C_2, the miniature that will be activated in C_2 is that with the closest height to the last miniature active in C_1. We depict the columns for the user by a drawing a vertical, dotted line through each columnar group. When a column is active, the dots (which are small squares) are red, otherwise they are black. We recognize that not all the transitions our algorithm generates are intuitive, but we hope that representing the columnar groupings by which the algorithm generates transitions will let users more easily predict the effects of their actions, and figure out how to get from one miniature to another.

WindowScape Task Managament

Unlike earlier window group management systems that require explicit user creation of groups, and explicit placement of windows in groups, WindowScape provides task management implicitly, through a history metaphor. Each time the user expands one or more miniatures, or miniaturizes one or more windows, a small photograph-like snapshot is added to the timeline pane, located by default on the

right half of a panel at the top of the display (Figure 2). If the panel is full, the leftmost snapshot is discarded. Each snapshot depicts the appearance and organization of the expanded windows at the time the snapshot was taken. In order to return the windows to earlier expansion (i.e., miniaturized or expanded) and position states, the user clicks on the appropriate snapshot. To give users a better sense of how the screen will appear if a snapshot is selected, the contents of the windows depicted in the snapshots are occasionally updated to reflect the current content of the actual windows.

These timeline snapshots constitute short-lived window groupings since clicking on different snapshots can expand different groups of windows from their miniature form to their full, interactive form or vice versa. In contrast to other window group management systems, WindowScape represents window groups in terms of abstractions that are independent of the window representations (i.e., the miniatures) themselves. But, since these abstractions are metaphoric photographs, they retain intuitive accessibility. This method of group representation has various benefits, including allowing the miniatures to be positioned independently of their group membership.

Despite its benefits, an implicit timeline-based model does not have good stability. If a user desires to return to a state from a dozen snapshots ago, it is likely that snapshot is no longer visible on the timeline. But even if we made the timeline larger or the snapshots smaller, the timeline's spatial instability may make it hard for users to leverage spatial memory in finding the desired snapshot. Therefore, we use the left half of the top panel seen in Figure 2 as a favorites bar. If a user thinks a snapshot may be important, or finds herself returning to it often, she can simply copy it to the favorites bar. Also, the user can take a snapshot of the current set of expanded windows and put that directly in the favorites bar with a key combination.

In this model, unlike conventional window grouping systems, the user never makes an explicit choice about where a window belongs. Rather, the user just expands windows and returns to prior states (via snapshots) as needed. The choice is made implicitly, after the fact, by the snapshots that the user selects frequently, or copies to the favorites bar. This approach also solves the problem of allowing windows to occupy multiple groups simultaneously. The use of the photograph metaphor provides an intuitive way for a single window to be represented in several groups, while making it clear that there really is only one underlying window. People understand that there can be several photos of an object with there being only one underlying object.

FUTURE WORK

In the near future, we plan to conduct a long-term deployment of WindowScape with about twenty participants and capture comprehensive statistics on how it is used, as well as user opinions. This study will give us insight into whether users find WindowScape useful, where they have difficulties, and how we could improve it.

In the longer term, we plan to study the integration of WindowScape with a traditional virtual desktop manager. We suspect that a tool like WindowScape would be better for managing sub-tasks where a stronger sense of context may be desirable, and a VDM would be better for dividing whole tasks, where pushing unrelated material out of sight and out of mind may be more desirable [6].

CONCLUSIONS

WindowScape is a zooming window manager that uses photograph and history metaphors to provide lightweight task management. Windows are represented as small miniatures which can be positioned by the user and expanded or miniaturized individually or in groups. Users can bring the miniatures and the title bars of expanded windows to the top of the z-order to search through them while keeping their currently expanded windows visible in the background. To facilitate keyboard navigation, the miniatures are grouped into columns. WindowScape allows tasks to be defined implicitly, by taking regular photograph-like snapshots of expanded window positions, to which users can return. This allows users to defer consideration of tasks until they need to return to them. Snapshots that are used often can be copied from the default timeline area to the favorites bar to avoid the lack of persistence of the timeline. In the near future, we plan to conduct user studies to investigate the real world usability of WindowScape.

REFERENCES

1. Bannon, L., Cypher, A., Greenspan, S., and Monty, M. (1983). Evaluation and analysis of user's activity organization". In Proc. CHI'83 (pp. 54-57). NY: ACM.
2. Henderson, D. A., Jr., Card, S. K. Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. In *ACM Transactions on Graphics 5* (3, July 1986), 211-243.
3. Kay, A. "The Reactive Engine." Doctoral dissertation, Electrical Engineering and Computer Science, University of Utah, 1969.
4. MacIntyre, B., Mynatt, E. D., Voida, S., Hansen, K. M., Tullio, J., Corso, G. M. Support for multitasking and background awareness using interactive peripheral displays. *CHI Letters*, 3 (2), 2001.
5. Rekimoto, J. Time-machine computing: a time-centric approach for the information environment. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 45 - 54, 1999.
6. Ringel, M. When one isn't enough: an analysis of virtual desktop usage strategies and their implications for design. *CHI Extended Abstracts* 2003, ACM Press, 762-763.
7. Robertson, G., Horvitz, E., Czerwinski, M., Baudisch, P., Hutchings, D., Meyers, B., Robbins, D., and Smith, G. Scalable Fabric: Flexible Task Management. In *Proceedings of AVI'04*, pages 85–89, May 2004.
8. Smith, G., Baudisch, P., Robertson, G., Czerwinski, M., Meyers, B., Robbins, D., and Andrews, D. (2003). GroupBar: The TaskBar Evolved. In Proc. OZCHI'03.