M5: Making Multiple Monitors More Manageable

Dugald Ralph Hutchings
GVU Center & College of Computing
Georgia Institute of Technology
Atlanta, GA 30332 USA
hutch@cc.gatech.edu

ABSTRACT

The goal of M5 is to understand how window management techniques might be shaped to offer a more efficient and enjoyable multiple-monitor experience. Two observations constitute the foundation of the M5 project: (1) despite many years of system-building work in the area of window management, there is surprisingly little evaluation work in the area, and (2) the increasing ubiquity of single-user, multiple-monitor systems and the initial findings suggesting the many different ways that people use them opens an intriguing avenue of research in the area. To accomplish our goal, the research will progress along the following path: assess general window management practices (irrespective of number of monitors), compare and contrast current practices between single-monitor and multiple-monitor users, build window management techniques to address problems and opportunity areas (for multiple-monitor users) discovered in the studies, and subsequently evaluate the techniques both in the field and in the lab. More broadly, the M5 project seeks to broaden the research community's understanding of multiple-monitors and the interfaces used to interact with them.

ACM Classification: H5.2 [Information interfaces and presentation]: User interfaces – *GUI, Windowing systems*

General Terms: Design, Human Factors

Keywords: multiple monitors, window management, snip, snap, mudibo, TaskZones

INTRODUCTION

As early as 1993 the HCI community has been exposed to the notion that a multiple-monitor display system has properties that diminish the effectiveness of traditional interfaces and invalidate assumptions about the ways that users manage the display space [5]. A multiple-monitor system is one that is physically separated but in some way virtually connected if not virtually contiguous. In 2001 Grudin presented some of the very first work describing how *everyday* multiple-monitor users can vary drastically from both typical single-monitor users and from each other, documenting three broad classes of use that he found during his field studies. Grudin further indicated that most users feel that they can complete tasks much more efficiently with more monitors, despite the existence of several usability issues that exist with the many interfaces built with a single-monitor user in mind [7]. Formal and informal studies following Grudin's work have verified that users indeed can experience great gains in productivity but can also suffer from usability problems [4, 17].♣, ♠ Many of the documented usability issues relate to the behavior of the window manager, an interface that pervades the entire display space and thus is perhaps most prone to usability issues.

There is a considerable amount of research in the area of window management, which is not surprising considering that it is an interface that nearly every user encounters every day. What is surprising is how little window management research involves either studies of users' general practices or some type of evaluation of a system or technique. Card *et al*. [2] and Gaylin [6] each provide studies of users' actual window management practices, but their findings may not be as applicable today as they were in 1984 and 1986 respectively since the number and variety of applications for everyday computer use have changed quite a bit. Furthermore, since all of the participants in their studies were most likely single-monitor users (an assumption that goes unstated in both papers), the work may not offer a true view of the needs and practices of multiple monitor users.

These two situations, *i.e*. the apparent usability issues of window management techniques on multiple-monitor systems and the lack of understanding of current-day user needs and practices, provide a natural path for my doctoral research. The first part consists of two user studies that address this lack of understanding and, along with others' recent multiple-monitor research, provide a foundation for building different interfaces that should address user needs and desires. The second part consists of actually building those interfaces and then evaluating whether they meet the needs and desires. As noted in the previous paragraph, it is rare that a proposed window management technique is seriously evaluated. In the cases where evaluation takes place, it is most often in laboratory settings [1, 15]. Thus a key contribution of our work will be *field* evaluations of the window management techniques that have been built. In the following sections each part of the doctoral research is discussed in more detail.

UIST'05, October 23–27, 2005, Seattle, Washington, USA.
Copyright ACM 1-59593-023-X/05/0010...$5.00.

♣ www.eetimes.com/showPressRelease.jhtml?articleID=103399

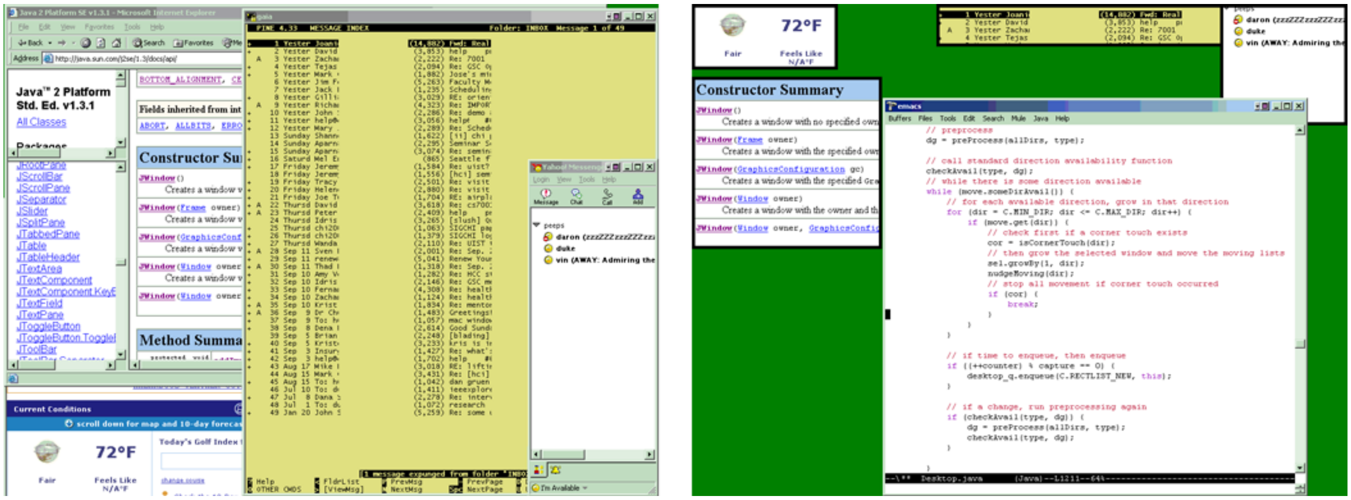
♠ codebetter.com/blogs/darrell.norton/archive/2003/11/11/3432.aspx

BASE RESEARCH – FIELD STUDIES

Two field studies constitute the base of the thesis research: (1) a series of interviews with 20 participants of a variety of occupations, window managers, and display systems [11] and (2) a collection of window management activity logs from 37 participants of similar occupation and window manager but that differ in the number of monitors that they use [13]. The big advantage of using interviews was that we were able to pose questions such as "*Why* are your windows arranged in a particular way" and "What aspects of the window interface trouble you" which to our knowledge had never been addressed by any window management study. More generally, we were able to understand underlying motivations for particular behaviors, especially those that might be "workarounds" to attaining a space management goal that is not directly supported. We found in particular that nearly all participants indicated that they would arrange windows to *hide specific content* rather than to show it. Although this study was published in 2004, the work was completed in early 2003 and closely relates to a result from Ringel's study that was being conducted at about the same time comparing single-monitor, virtual desktop users to multiple-monitor users. She indicated that many participants *avoided* multiple monitors because the additional monitors encouraged them to be distracted by information peripheral to a primary task [16]. This key finding has played a major role in our construction of interfaces for multiple monitor users, where there is more space, thus more chance that content can be distracting, thus more opportunity and need for window management operations that more tightly control the content that is displayed. More detailed information about this study can be found in our 2004 GI paper [11].

While interviews have inherent advantages, they also have the disadvantage that participants may not be able to fully explain why they perform in a particular way and cannot completely describe how they generally behave. Thus to supplement our own study as well as the findings reported by Grudin and his interview-based multiple-monitor study, we designed a tool that could log a person's window management activity to compare single-monitor and multiple-monitor users. The big advantage of the study was the large amount of data that we were able to collect: in 3 weeks, 37 users provided over 100,000 minutes of window activity (about 500 times more log data than Gaylin collected, although he also shot video [6]). We were able to provide additional evidence for claims that had already been made by others' studies, such as that the TaskBar was less often used as the number of monitors increased and that people tend to use a second or third monitor to display information peripheral to a primary task or for communication interfaces. We also found surprising results, including that there was no significant difference in the amount of time in between window focus switches between single-monitor and multiple-monitor users, and that single-monitor and dual-monitor users did not significantly differ in the number of simultaneously visible windows but there was a significant difference between dual-monitor and triple-monitor users. More detailed information about this study can be found in our 2004 AVI paper [13].

The results of the two studies open a number of potential research avenues. We have chosen to focus in depth on the combination of two specific findings: (1) there is often a desire to either hide or show specific window content (2) users often employ secondary and tertiary monitors so show information in support of or peripheral to the main window of a task but do not necessarily desire to interact with this information. As a result, we have built two interfaces that allow users to show critical information in a smaller space or hide distracting information, leaving more space for additional helpful information or the opportunity to simply leave it blank. We have also built a window grouping interface called TaskZones that is intended to replace



```
        // preprocess
        dg = preProcess(allDirs, type);

        // call standard direction availability function
        checkAvail(type, dg);
        // while there is some direction available
        while (move.someDirAvail()) {
            // for each available direction, grow in that direction
            for (dir = C.MIN_DIR; dir <= C.MAX_DIR; dir++) {
                if (move.get(dir)) {
                    // check first if a corner touch exists
                    cor = isCornerTouch(dir);
                    // then grow the selected window and move the moving lists
                    sel.growBy(1, dir);
                    nudgeMoving(dir);
                    // stop all movement if corner touch occurred
                    if (cor) {
                        break;
                    }
                }
            }

            // if time to enqueue, then enqueue
            if ((++counter) % capture == 0) {
                desktop_q.enqueue(C.RECTLIST_NEW, this);
            }

            // if a change, run preprocessing again
            if (checkAvail(type, dg)) {
                dg = preProcess(allDirs, type);
                checkAvail(type, dg);
            }

        }
--\**  Desktop.java       (Java)--L1211--64%----------------------------------
```

Figure 1. On the left, the user has some code documentation, an email client, an instant messaging client, and the weather in secondary windows, but has no room to place the primary code editing window. On the right is the result of the user snipping the secondary windows, allowing her to monitor new email, see who is online, use the relevant code documentation, and track the weather while having a large amount of space in which to edit code. The result is an information-dense, visually sparse display.

virtual desktops in order to address concerns of distraction and ease-of-use of such interfaces on multiple-monitor systems. In this paper we will avoid further discussion of TaskZones since we do not plan any further evaluation of it, but it has been submitted as a poster to UIST and will hopefully appear. So we now move onto the snip and snap interfaces for showing and hiding content in a window.

SNIP AND SNAP – MANIPULATION INTERFACES

Overview: Snip and snap are simple-to-understand, easily described interfaces. They do not involve advanced animation or employ intricate adaptive techniques, as some recent window management techniques and systems have. But they *directly address* the problems and issues that we uncovered with our base studies, which is why we think that they will be both simple and powerful. In the next section we discuss how we plan to determine whether the interfaces truly are powerful.

Snip

The snip interface was initially presented as a prototype at AVI 2004 [12], but since then we have developed it as a fully-functional interface for Windows XP. Snip allows a user to specify a rectangular region of a window and keep only that region visible on the screen. Figure 1 illustrates a scenario where snip might be useful (previous page, copied directly from the AVI paper [12]). Note that the window is still "live," which allows the user to continue to interact with it even in snipped form. For example, in the instant message window in Figure 1, the user can still right click on a name to send an instant message. The reason that something like snip is more useful than the standard resize operation is that resizing a window typically alters the size of the content area, keeping all interaction components like buttons and menus visible. However, when used in support of some other window or monitored peripherally, these interface components are often unnecessary. Furthermore resizing a window may cause unusual, unpredictable, or undesirable alterations to the presentation of content. Snip allows a user to select a current view and show only the relevant information in the view. Note that snip is similar WinCuts, which is an interface that we proposed in our early work [10] but was ultimately implemented by Tan *et al.* [18].

Snap

The snap interface is based on work from Guimbretière *et al.* about interaction with wall-sized high-resolution displays [8]. He allowed users to take images from web pages out of the browser and into separate windows to help facilitate brainstorming. Our version of the snap interface allows a user to specify any region of a window and produce that region as an image in its own window. As opposed to snip, the resulting image from a snap operation is not "live" but is truly a static image. Snap can be useful in situations where further interaction with the information is unneeded or would be too complex. A common example is locating an address with one search and then typing that address into another search to get a map of the location. Since addresses do not tend to be easily copied and pasted from one webpage to another due to line breaks, it might be better to take a quick "snapshot" of the address, load the map page and just type the address in. Other possible uses of snapshots are as visual reminders to do something (allowing the window to be used for other tasks in the meantime) and as a possible technique to avoid navigating to different parts of a document for reference. In this latter example, imagine taking a snapshot of an outline in the beginning of a paper and using it to help remember what to write next while writing the rest of the paper. Rather than scrolling back and forth or setting a bookmark, a snapshot can allow a person to continue to write without having to constantly switch views, which exploits the extra space provided by multiple monitors.

EVALUATING SNIP AND SNAP

We have planned two phases of evaluation of these window management operations. Phase I involves a deployment of the interfaces to actual users in their work environments, which is a rarity in the window management area (the formative evaluation of Rooms is a notable exception [9]). We endeavor to understand how the availability of each operation modifies the general space management practices of users: will they tend to have more windows visible at one time as a result? Do the smaller windows tend to appear on only specific monitors or do they appear all over the display space? We also plan to assess properties of each operation individually: will people use snip on windows that they are actively working on or only on windows that help support interaction in another window? Will snipped windows remain snipped for short periods of time or long periods of time? Will snapshots be used more actively (such as helping in a focus task) or more passively (such as just a visual, in-context reminder to do something)? Two primary data collection methods should help us to provide these answers: logging and interviews. We will log each time a participant uses an operation, as well as periodically list all of the open, visible windows, in order to reconstruct a general picture of the arrangement of the display space over time. Interviews should allow us to better understand the intention of users behind their logged behaviors as well as allow the users to describe how they found the operations useful, how they could be improved, and the situations in which they most frequently used the operations.

Phase II of the evaluation consists of controlled laboratory studies of the implemented window operations, which as mentioned are relative rarities in this research area. We hypothesize that certain types of tasks can be completed more quickly or efficiently through use of the snip or snap operations. Snip should aid in tasks where several reference windows are needed or where several information sources need to be frequently monitored and snap should aid in tasks where view changes are frequent (such as scrolling in a long document or navigating in a map). However, these studies occur in the second phase because information from the first phase may reveal or help construct more representative tasks. There may be task types that we have not yet considered that appear to benefit from one of the window operations. We would like to test these types of tasks to both (1) validate the opinions of our participants gathered from Phase I and (2) increase the external validity of the Phase II studies.

DISCUSSION AND OTHER WORK

Doctoral research necessarily involves deeper exploration of a narrow range of topics as opposed to shallower exploration of a wide range of topics. Snip and snap allow exploration of the use of manipulation techniques that can be applied to essentially any software application on personal, multiple-monitor display systems as opportunities to show additional information that can help support a primary task. The broader context of the work is the general exploration of how to exploit the space afforded by any multiple-display system, which is a topic of great current interest to the UIST community and should allow for an engaging discussion at the Doctoral Symposium. In addition to these interfaces, I have also been examining other aspects of multiple-monitor systems. The *mudibo* interface includes the notion of replication of information across monitors to aid in the use of dialog boxes: rather than forcing the window manager to decide "*the* correct location" of a dialog box, mudibo replicates the dialog box across all of the monitors until the user starts to interact with one of the copies, then hides the copies with which the user did not interact. We expand at length on the potential power of replication coupled with automation in our CHI Technote [14]. As mentioned we have also developed a window grouping interface that re-examines the usability and utility of virtual desktops on multiple-monitor systems. I would be happy to discuss these other minor aspects of the research at the Symposium.

The contributions of the work arise from the results of the base field studies and the evaluation of the interfaces, not necessarily from building and developing the interfaces themselves. Other members of the UIST community, traditionally centered on system-building challenges, are themselves becoming more interested in the prospect of new window management work, multiple-monitor systems, *and* evaluation. For example consider Metisse, a recent virtual window manager that is provided to allow a variety of researchers to easily build and evaluate real window manager interfaces in a variety of research contexts [3]. Our work demonstrates that Metisse is a worthwhile venture since there is a desire to conduct serious evaluation.

The M5 project incorporates more evaluation and less system-building than the usual UIST research, but expands the understanding of an emerging important topic in the UIST community and further demonstrates how evaluation can assist researchers. For this and the other reasons outlined in this paper, we hope to contribute to vigorous discussion of current research at the Doctoral Symposium.

ABOUT THE AUTHOR

Duke Hutchings is a Ph.D. candidate under the advisement of Professor John Stasko in the GVU Center of the College of Computing at the Georgia Institute of Technology. In addition to multiple-monitor window management, his interests include collaborative multi-display systems, information visualization, and peripheral awareness tools. He recently hosted a workshop on the topic of Distributed Display Environments at CHI 2005. More information is available at http://www.cc.gatech.edu/~hutch.

REFERENCES

1. Bly, S. A. and Rosenberg, J. K. A comparison of tiled and overlapping windows. *CHI 1986*, ACM Press, 101 – 106.
2. Card, S., Pavel, M., and Farrell, E. Window-based computer dialogues. *INTERACT 1984*, Elsevier Science Publishers and IFIP, 239 – 243.
3. Chapuis, O. & Roussel, N. Metisse is not a 3D desktop! TR 1407, LRI, Université Paris-Sud, 2005.
4. Czerwinski, M., Smith, G., Regan, T., Meyers, B., Robertson, G. and Starkweather, G. Toward characterizing the productivity benefits of very large displays. *INTERACT 2003*, IOS Press, 9 – 16.
5. Funke, D. J., Neal, J. G, and Paul, R. D. An approach to intelligent automated window management. *Int. J. of Man-Machine Studies 38* (1993), 949 – 983.
6. Gaylin, K. How are windows used? Some notes on creating empirically-based windowing benchmark task. *CHI 1986*, ACM Press, 96 – 100.
7. Grudin, J. Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. *CHI 2001*, ACM Press, 458 – 465.
8. Guimbretière, F., Stone, M., and Winograd, T. Fluid interaction with high-resolution wall-size displays. *UIST 2001*, ACM Press, 21 – 30.
9. Henderson, D. A. Jr. and Card, S. K. Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Trans. on Graphics 5*, 3 (1986), 211 – 243.
10. Hutchings, D. R. and Stasko, J. New operations for display space management and window management. *GVU Technical Report GIT-GVU-02-18*, August 2002.
11. Hutchings, D. R. and Stasko, J. Revisiting display space management: Understanding current practice to inform next-generation design. *Graphics Interface 2004*. Canadian Human-Computer Comm. Soc., 127 – 134.
12. Hutchings, D. R. and Stasko, J. Shrinking window operations for expanding display space. *Advanced Visual Interfaces 2004*, ACM Press, 350 – 353.
13. Hutchings, D. R., Smith, G., Meyers, B., Czerwinski, M., and Robertson, G. Display space usage and window management operation comparisons between single monitor and multiple monitor users. *Advanced Visual Interfaces 2004*, ACM Press, 32 – 39.
14. Hutchings, D. R. and Stasko, J. mudibo: Multiple Dialog Boxes for Multiple Monitors. *CHI 2005 Extended Abstracts*, ACM Press, 1471 – 1474.
15. Kandogan, E. and Shneiderman, B. Elastic windows: Evaluation of multi-window operations. *Proc. CHI 1997*, ACM Press, 250 – 257.
16. 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.
17. Tan, D.S. and Czerwinski, M. Effects of visual separation and physical continuities when distributing information across multiple displays. *Proc. INTERACT 2003*, IOS Press, 252 – 265.
18. Tan, D. S., Meyers, B., and Czerwinski, M. WinCuts: manipulating arbitrary window regions for more effective use of screen space. *CHI 2004 Extended Abstracts*, ACM Press, 1525 – 1528.