Display Space Usage and Window Management Operation Comparisons between Single Monitor and Multiple Monitor Users

Dugald Ralph Hutchings

GVU Center/College of Computing
Georgia Institute of Technology
Atlanta, GA 30332 USA
hutch@cc.gatech.edu

Greg Smith, Brian Meyers
Mary Czerwinski, George Robertson
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
{gregsmi, brianme, marycz, ggr}@microsoft.com

ABSTRACT

The continuing trend toward greater processing power, larger storage, and in particular increased display surface by using multiple monitor supports increased multi-tasking by the computer user. The concomitant increase in desktop complexity has the potential to push the overhead of window management to frustrating and counterproductive new levels. It is difficult to adequately design for multiple monitor systems without understanding how multiple monitor users differ from, or are similar to, single monitor users. Therefore, we deployed a tool to a group of single monitor and multiple monitor users to log window management activity. Analysis of the data collected from this tool revealed that usage of interaction components may change with an increase in number of monitors, and *window visibility* can be a useful measure of user display space management activity, especially for multiple monitor users. The results from this analysis begin to fill a gap in research about real-world window management practices.

Categories and Subject Descriptors

D.4.9 [**Operating Systems**]: Systems Programs and Utilities – *window managers*.

H.5.2 [**Information Interfaces and Presentation (e.g., HCI)**]: User Interfaces – *evaluation/methodology*, *graphical user interfaces* (*GUI*), *windowing systems*.

General Terms

Management, Measurement, Experimentation, Human Factors

Keywords

UI logs, window management, space management, multiple monitors, user interaction, automation

1. INTRODUCTION

The variety of ways that people can and do commonly use personal computers (PCs) has broadened tremendously throughout the history of the machine. People schedule their entire day using calendar software, they conduct research or follow current events through multi-media web browsing, they store and access entire music collections, they communicate with one another through email and instant messages... the list itself could fill the pages of this document. However, the display through which people conduct these activities has changed little. Most still use one rather small monitor to conduct all of these activities. With so many activities to accomplish, users can have significant trouble effectively managing a small amount of screen real estate.

More recently, advancing technology is allowing users to easily and cheaply use additional monitors. Newer video cards support two, three, or even four independent monitors, and most PCs allow people to use many video cards simultaneously. The small form factor of newer monitors such as flat-panel displays allows an already crowded physical desk space to accommodate more monitors on a single PC. The display itself is also improving, with the ability to display more pixels in a limited space than ever before. Multiple monitor systems can help to assuage the problems with managing a small amount of space, but could simultaneously introduce new problems in managing a large amount of space. The growing popularity of multiple monitor systems has prompted us to investigate the differences user exhibit in managing systems of varying numbers of monitors. Understanding the differences can help in the design and evaluation of space management interfaces.

We have conducted a study of the window management practices of both single monitor and multiple monitor users to begin to understand the differences and similarities between the two groups. After discussing related work and some definitions, we present the tool we developed to track window management events and periodically record window configurations of our participants. We then show the characteristics we could derive from the data generated by the tool, showing how some aspects (such as how windows are accessed) are different while other aspects (such as the amount of time windows remain accessed) are similar. Finally, we discuss how these findings relate to future work and conclude that there is much work remaining for designing and evaluating multiple monitor systems.

2. RELATED WORK

More users are opting for multiple monitor systems, and initial lab research indicates that multiple monitor systems can help users be more productive [4] but that multiple monitor systems could stand to gain from advances in hardware and software design [17]. These important findings motivate field work such as ours in order to understand actual management practices that people employ. Other field research has shown the different emerging broad strategies people have developed for using multiple monitor systems. One particularly interesting finding is that people do not treat additional monitors as "additional space" but rather tend to manage windows within monitors and rarely place windows across physical monitor boundaries 0[7]. Ringel interviewed virtual desktop users to understand the difference between them and multiple monitor users [15], finding that virtual desktop users prefer to have peripheral windows (such as email) on separate desktops so that they do not distract them from their primary tasks. Hutchings and Stasko [10] interviewed office workers of a variety of window managers and display configurations to understand high-level space management practices. One interesting finding in that study was that users would activate a window not to display information in it but to hide information in non-active windows. A topic that these pieces of work do not cover is the *mechanics* of space management on multiple monitor systems, *i.e.*, how users arrange windows to produce the desired display effects. Little other work tackles this issue for any display configuration, although Gaylin [6] provides a notable exception for single monitor users. His study is somewhat limited though, as it consists of a small user population (8 programmers and 1 other) over a short period of time (22 minutes per participant) and observation rather than the capture of raw window system events, as we use in our study. We also employ a broader user population and a longer period of time in which users work with windows.

Beyond studies of display space and window management, many systems have been developed to help users manage space, but all were seemingly designed with one-monitor systems in mind. We briefly describe several systems, and relate them to potential uses for multiple monitor systems. For example, one potential advantage of attaching many displays to a single computer is the ability to work with a similar amount of digital space as one might have as a workspace in the physical world. Examples of interfaces that try to emulate the real-world manipulation of the display of information include Rotating and Peeling Windows [1], which try to help users access more windows by treating the windows as physical pieces of paper, and Rooms [8], an early and comprehensive virtual desktop system. Systems that employ visualization techniques could also benefit from an increase in space. One such system is The Task Gallery [16], which was shown to help users organize tasks through the 3D metaphor of an art gallery, where each task is hung on the wall or ceiling. However, it is unclear how effective 3D environments can be on non-contiguous multiple monitor systems. The previously mentioned window managers and interaction techniques all employ overlapping as a means for managing windows, but many tiling systems have been proposed too. Bly and Rosenberg [3] and more recently Kandogan and Shneiderman [12] demonstrate situations in which advanced tiling window managers outperform overlapping ones. However, both tests were conducted on single monitor computers, so it is unclear how tiling systems would perform on multiple monitor computers (even if enhanced to reflect user practices such as avoiding placing windows across physical monitor boundaries [7]). The same adaptation issue arises for automated window operations. Non-overlapping Dragging [2] is a technique that automatically moves obscured windows to available empty screen spaces, and QuickSpace operations [11] automatically move windows so that they will not be further obscured by a growing window. Both techniques rely on the existence of empty space, which we show may not often be available even on multiple monitor systems. Related to automated operations is the notion of adaptive window management, where a common shortcoming of such systems is the hiding or moving of windows that the user would like to remain visible or anchored. CIWM [5] is a window manager that was designed specifically for a multiple monitor system, and its evaluation indicates that windows can be hidden at inopportune times. Algorithms for deciding which windows to hide generally hinge on how often users interact with the window, but in larger display systems where input generally is directed at only one monitor at a time, a better metric may include how often the user displays information windows. We give some analysis of window visibility and its meaning for multiple monitor users later in this paper. First though, we introduce some definitions of terms for window management.

3. DEFINITIONS AND BACKGROUND

For those who are unfamiliar with the common terms used for window managers, we point to Myers' excellent overview of the topic [13]. Our field study involves users of the Windows XP window manager (hereafter referred to as *Windows*) and Windows contains some features not discussed by Myers. Since a basic understanding of the user interface of Windows is essential to understanding the results of this paper, we give a brief description in the next paragraph. Readers familiar with Windows may skip to the following paragraph.

The *desktop* is the main window, *i.e.*, the one on which all other windows are displayed. Exactly one window is the *active window*, *i.e.* the only window receiving user input. For a window *w*, there are several ways to make *w* active (also called *switching* to *w*), including clicking on any part of *w*, pressing <alt>+<tab> and selecting *w* from the resulting window list, or clicking on *w*'s *taskbar* button. The *taskbar* is a special area that aids in window management by displaying a list of buttons, one for each application window. The user has the option of making the taskbar *always on top* so that other windows cannot occlude it, or *autohide*, so that the taskbar slides out of focus (and out of sight) when not being activated. Special operations for windows include *minimize* (which Myers refers to as *iconify*) which hides a window from view but maintains its taskbar button, and *maximize*, which grows a window to the size of its current monitor. A key attribute of a window is its *z*-position. Windows can overlap, so each window is assigned an integer (*z*) as its depth; the window with the lowest *z*-position is the window at the top of the desktop.

An open window can be *invisible* to the user for two main reasons: (1) the user has *hidden* the window, for example by minimizing it, and (2) the window is *occluded* because another window or set of windows lower in the *z*-order obscures it. The reader should take care to distinguish between *hidden* and *occluded*; we use the terms *not visible* and *invisible* to apply to either case.

4. INITIAL STUDY

Thirty-nine volunteers from within a research organization participated in a 3-week study of their computing event activity by using VibeLog on their work PCs. VibeLog is a window operation logging tool that we describe in detail in the next section. Occupations of our participants included Administrative Assistant (1), UI designer (1), Program Manager (3), Software Developer (9), Research Intern (8), and Researcher (17). We captured 105,402 minutes (just over 73 person-days) of participants' *active time*. A particular point in time is active if, within the previous 5 minutes, there was a mouse movement or key press. We use active time to prevent capturing data during period of inactivity, such as when the participant is eating lunch or has left work for the day. When the logging tool detects that the user is no longer active, *idle* events are inserted in the log to mark the duration of non-active time.

Throughout the study, some users changed display configurations, and across our sample we observed 29 single monitor users, 18 dual monitor users, and 2 triple monitor users. 14 multiple monitor users had less than 3 million pixels of display area (we refer to this group as *small multimon*), and 7 multiple monitor users had 3 million pixels or more of display area (we refer to this group as *large multimon*). Again, the breakdowns sum to more than 39 because some people worked with more than one configuration at different times. All participants were researchers within some sub-discipline of Computer Science.

5. VIBELOG

The VibeLog application is built in Visual C++ and runs on any current Windows platform. When started, the tool runs continuously and is reset only upon system shut down, display configuration change, or the passing of 24 hours since the previous reset. Static configuration information is collected at startup, including the ID and coordinates of each monitor registered with the operating system. The main feature of VibeLog is the maintenance of two logs of window system information: *events* and *windows*. The log of events contains an entry for every window management activity and the log of windows contains a series of entries enumerating the on-screen windows each minute that a user is active.

5.1 Log of Events

The event log has an entry for every window management activity that occurs. These activities include opening and closing a window, showing and hiding a window, activating a window, *and* moving, sizing, minimizing, maximizing, and restoring a window. There is also an entry when users press <alt>+<tab> to switch to a different window. VibeLog is able to maintain this log by programmatically hooking the public window system events made available by Windows; no modification or private instrumentation of the operating system is required. Each log entry has a timestamp and contains *window* and *input* information.

5.1.1 Window Information

Window information includes the window's ID, title, host application, coordinates, *size state*, *style*, and *monitor information*. The size state is one of *maximized*, *minimized*, or *normal*. The window style defines a number of attributes of the window, including whether the window is (1) a *popup* window (typically used for dialog boxes), (2) a toolbar (see Figure 1), (3) invisible, (4) transparent, and (5) always on top. There are two pieces of monitor information. The standard API call for a window's monitor returns the *main monitor*. For a single monitor system, this is the one and only monitor. For a multiple monitor system, however, this is the monitor that contains the majority of the window's area. Thus the second field is the number of monitors on which the window actually resides, as determined by rectangle intersections between coordinates of the window and coordinates each monitor.

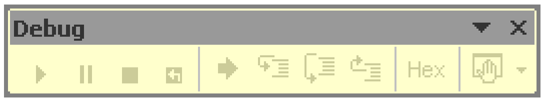


Figure 1. *Debug* is a toolbar window from a development environment – an example of a floating palette.

5.1.2 Input Information

Input information includes the input *type* and *location*. The input type is one of *keyboard* or *mouse*. Alternative input devices are abstracted by the window system, so they too appear as one of the two types. The input location is one of *window*, *taskbar*, *desktop*, or *alt+tab*. If a user clicks on a window to activate it, the location is *window*, whereas if the user activates the same window by clicking on its taskbar button, the location is *taskbar*. If a user opens a new window from a desktop icon, the location is *desktop*. If a user activates a window by using <alt>+<tab>, the location is *alt+tab*.

The reader should note that this information is not included in the event generated by the window system. To collect this information, we track the input of the user separately, and attribute the most recently generated input event to the window event, taking care to clear the most recent input event as appropriate. We videotaped ourselves generating every combination of input that we could list, and then checked the resulting logs against the tape. Very infrequently there were errors in the input information, but unfortunately we cannot provide a specific margin of error for these calculations.

5.1.3 Note on Incompleteness

For technical reasons such as OS optimization of event generation and dispatch, it is impossible to guarantee that every event generated by the user will appear in the event log. Thus each log may contain omissions of some events, although evidence suggests that such omissions will be extremely infrequent. We used the videotape method described above to also check for omissions. Only one application (a web browser) and one window management operation (activation) ever failed to generate events, and these gaps were infrequent. One omission in a 20 minute intensive-use event log was common, if present.

5.2 Log of Windows

The window log writes a series of entries, one per each open window, every minute. This log was originally designed as a checkpoint for the event log, serving as a redundant source of periodic desktop content information. However, the window log also allowed us to easily make coarse-grained calculations of *window visibility*, *i.e.*, what windows were visible to the user at a point in time, without the aid of the full event stream in the event log. Window *visibility* needs to be distinguished from window *active state*, as only one window can be active at a time but many windows can be visible at one time.

Much of the same window information in event log entries also appears in window log entries: a timestamp, handle, title, application name, coordinates, size state, style, and monitor

Table 1. MM participants used the taskbar less and window interactions more to switch among windows.

Display	Total Switches	Window Switches	Taskbar Switches
single monitor	186,708	64.7 %	26.3 %
small multimon	63,083	78.9 %	13.3 %
large multimon	90,284	87.4 %	5.2 %

Table 2. Email is (1) invisible less often, (2) visible more often, and (3) less often active when fully visible for MM participants.

Display	Email Invisible	Email Fully Visible	Active when Fully Visible
single monitor	67.1 %	6.7 %	90.0 %
small multimon	51.0 %	27.1 %	44.6 %
large multimon	26.6 %	29.5 %	13.9 %

information. Other information includes the *z*-position and active state (a binary value to indicate if the window was active).

6. ACROSS-USER ANALYSIS

We were interested in understanding how people used windows differently (or if they used windows similarly) with respect to monitor configuration. In this section, we present information on how people switched among windows, how long windows remained active once switched to, and how people kept both active and inactive windows visible.

6.1 Switching Windows and Taskbar Usage

As previously noted, one of the main methods for switching to a different window is to click on the window's taskbar button. Recent research has hinted that the taskbar has potential usability problems. One issue raised by some participants in a qualitative study is that when eight or more windows are open, and the taskbar is in its default position at the bottom of the primary monitor, only a few (or none!) of the letters in the windows' titles are visible [10]. Analysis of our participants' data revealed that 78.1% of the time people had eight or more windows open, so users may often experience problems with using the taskbar. Others have suggested taskbar issues specific to multiple monitor users. Observations and follow-up interviews of participants in a controlled multiple monitor study indicated that the participants had trouble using the taskbar because of the amount of screen real estate they had to traverse to interact with it [4]. This issue led us to hypothesize that multiple monitor users will tend to access windows directly more and use the taskbar less than single monitor users.

The study data in fact supports the hypothesis. Table 1 shows the higher percentage of times that single monitor participants switched windows using taskbar as compared to the two multiple monitor groups (MM participants). Window switches include clicking on a window and minimizing another window but do not include *alt+tab*, *i.e*., using a keyboard shortcut.

6.2 Amount of Time that Windows Are Active

Participants produced 360,084 activate events, accounting for both the opening of new windows and switching to already opened windows. The average amount of time that any window was active was 20.9 *seconds*. (This average excludes activation durations of less than 150 milliseconds, which are likely to be artifacts of multiple windows and sub-windows popping up near-simultaneously in response to a single user action, rather than representing multiple user actions). Perhaps more revealing, however, is that the median amount of activation time is 3.77 seconds, *i.e*., half of all window activation lengths are quite short. One major implication of this finding is simply that users frequently shift their attention among several windows – this can be due either to user action in proactively switching tasks, or standard application tendencies to pop up many short-lived sub-windows and dialogs.

An interesting aspect to the window switching statistics is that they hold within each of the single monitor, small multimon, and large multimon user groups, but are likely to affect each group differently. For single monitor users, the visible regions of on-screen windows are likely to frequently change, because activating a window causes the depths of other windows to change. This may not hold for multiple monitor users since, for example, a user could be switching back and forth between two windows that are each completely visible on separate monitors. But if a person uses a second or third monitor as a mainly peripheral display (where windows are shown but seldom become active), then the same "changing window depth" issue arises for the monitors in active use. Each type of use calls for different design considerations: if depth frequently changes, designers may try to develop techniques to create stable display of information, whereas if user focus frequently changes among the monitors, designers may develop better navigation techniques to more easily switch among windows (as pointed to in the previous subsection).

6.3 Window visibility

The length that windows are active is only one measure of the use of screen space. One of the major advantages of a multitasking window system is the ability to both run and display many applications simultaneously. Since screen space is a limited resource, it seems likely that any open window with some part visible at a particular point in time is of some importance to the user. We thus developed a line of analysis that focuses on this measure of importance by calculating the percentage of visible area of a window for each entry in the window logs.

6.3.1 Number of visible windows

It might be expected that multiple monitor users would have more windows visible than single monitor users, and our results confirm that expectation. However, the gap between single monitor users, who averaged 3.5 visible windows, and small multimon users, who averaged 4.1 visible windows, is surprisingly small. On the other hand, large multimon users averaged 6.8 visible windows. The median for each group was 3, 4, and 6 visible windows, respectively. One possibility for the small gap is that the small multimon users favored the display of larger windows over the display or more windows. There were significant negative correlations between the number of windows visible and the number of monitors a user has (r = - 0.85 for both single monitor users and dual monitor users). In other words, both single monitor users and small multimon users have a significantly lower

Table 3. With more screen space, users tend to have some part of the screen empty. However, no clear pattern emerges for the amount of space that will be empty. In general, all participants frequently had very little empty screen space.

Display	Time with no empty space	Time with less than 20% empty
single monitor	48.0 %	89.9 %
small multimon	33.5 %	71.0 %
large multimon	14.3 %	80.8 %

likelihood of having a large number of windows visible. There was no correlation between large multimon users and the number of windows visible.

6.3.2 Visibility and Activity of the Email Inbox

One particular application window that demonstrates what visibility can indicate about screen space usage is the email window. Since each person in the study used the same email client to interact with email, we were able to easily gather statistics about the use of the email inbox. Table 2 outlines the data that we presently discuss.

For single monitor users, the inbox was invisible 67.1% of the time and completely visible 6.7% of the time. When completely visible, the inbox was active 90.0% of the time, meaning that users rarely displayed the entire email window while interacting with another window or application. However, for small multimon users, the inbox was invisible only 51.0% the time and was fully visible 27.1% of the time. Furthermore, when fully visible, the inbox was active only 44.6% of the time. Large multimon participants exhibited even more dramatic differences. Inboxes for large multimon participants were invisible 26.6% of the time and were fully visible 29.5% of the time. For only 13.9% of the fully visible time, the inbox window was active. This seems to indicate that users with more space use the inbox as a *glancing window*, watching for incoming email but not necessarily interacting with it, and making it very easy to access email when new messages arrive. While we cannot make a strong claim about the prominence of email in the presence of multiple monitors, there appears to be a pattern emerging that bears further exploration.

6.3.3 Empty Space

As outlined earlier, recent work in space management has focused on using empty space and dynamic window movements to help keep more windows visible simultaneously [[2], [11]]. The data we collected yields the opportunity to understand how much empty space tends to be available on users machines. In Windows, the bottommost window is the desktop window, and measuring its visibility indicates the amount of screen space unoccupied by any window. Table 3 shows empty space information.

Among single monitor users, there was no empty space for almost half of the time logged (48.0%), and for 89.9% of the time logged, less than one-fifth of the desktop was visible. Small multimon users tended to have screen space open more frequently though, having no space only 33.5% of the time and less than one-fifth of the screen 71.0% of the time. Large multimon had no empty space only 14.3% of the time, yet surprisingly 80.8% of the time less than one-fifth of screen space was empty. Therefore, window management techniques have an increased chance to exploit empty space on multiple monitor systems, but since users rarely arrange windows across monitors [7], exploiting this space requires careful consideration. We conclude that empty space based management ideas show some promise when augmented by an understanding of multiple monitor users' practices and where the monitor bezels are configured.

7. PER USER ANALYSIS -VISUALIZATION OF WINDOW VISIBILITY

Besides gathering data across a group of users, our tool allows us to inspect space management behaviors of individual users. We have developed a visualization to inspect broader patterns of visibility for individuals, and our data suggests that visibility can indicate quite a number of characteristics of individuals.

7.1 Visibility with Monitor Information

Color Figure 1 is a visualization of a particular participant's window visibilities over a period of 22 minutes of active time.[†] The x-axis is labeled with time, and each tick is one minute. The y-axis has an entry for each window that was visible, and is labeled with the host application name. The color of any block (x, y) indicates the monitor on which the window resided. Red, green, and blue in Color Figure 1 each represent one of the three monitors of the user. Any window that is situated across more than one monitor is represented in grayscale (note how the desktop is gray). The amount of shading of (x, y) is the amount of y's window area that was visible at time x. Pure red, green, blue, or black indicates that a window was fully visible. Lighter shades indicate lesser visibility, and pure white indicates that the window was not visible. If (x, y) contains a white dot, then y was the active window at time x. There is at most one white dot in each column.

7.1.1 Stability of Window-Monitor Placement

In Color Figure 1, the absence of gray from any window (other than the desktop) indicates that the user places windows completely on one monitor. But the visualization actually shows something stronger. Looking from left to right at each window, we see that almost all of them only have one color, *i.e.*, that they each stay on one *specific* monitor. Only the inbox window (second from the bottom of the figure) ever appears on more than one monitor, and even in this case spends all of the time other than one minute on the green monitor.

While most visualizations were similar to Color Figure 1 with respect to window monitor stability, Color Figure 2[†] illustrates a dual monitor user who straddled an active window across monitors, again showing particular windows behaving in different ways. An interesting observation about this case is that the result of positioning the window across monitors was an increase in the amount of visibility of the email inbox.

7.1.2 Monitor Usage

Another pattern involving the green monitor also appears in Color Figure 1. While the active window appears 14 times on the red monitor and 7 times on the blue monitor, the green monitor has the active window only one time. It appears that this user prefers

[†] Grayscale versions of Color Figure 1 and Color Figure 2 are located on the last page of this paper. The actual color versions appear in the color plates section of the proceedings. The electronic version of this paper uses color images on the last page.

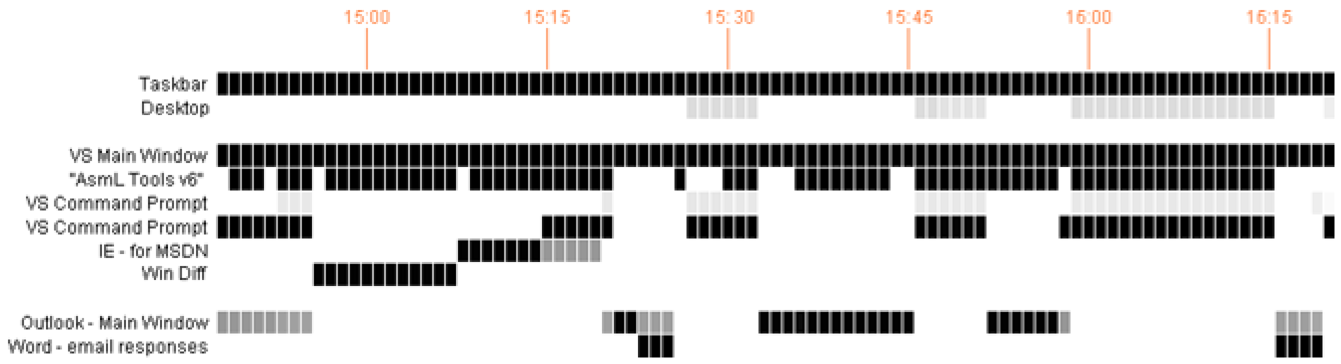


Figure 2. A visualization of window visibility from a window log file without monitor information.

to place windows that provide information without needing user input on the green monitor. For example, media player can play music or show videos for hours after clicking a "play" button once, and the inbox automatically shows new messages as they arrive without any user input. So by arranging the visualization to group windows by monitor, we can easily see if each monitor serves a particular role for the user. Color Figure 2 also demonstrates how no window fully on the green monitor received input, indicating that this participant used the green monitor more to display information than to interact with windows. However, the user gave much of the active time to a window across both monitors.

7.2 Visibility without Monitor Information

Figure 2 is a visualization similar in layout and structure to that of the color figures, but does not include the active window dot or monitor information. Instead, each block is shaded in grayscale. Figure 2 also looks at a much longer period of active time of 92 minutes. By eliminating monitor information, one is able to more easily focus on more general patterns of visibility, as we outline in the following two subsections.

7.2.1 Task Switches

Figure 3 is a cut-away of Figure 2. Analyzing the image from left to right, we first see that the command prompt window (top) becomes invisible at the same time that the text file comparison window (bottom) becomes visible. A similar situation then occurs as the help/documentation window becomes visible. Finally, the help window becomes partially visible as the command prompt window once again becomes fully visible. Note that this last switch could indicate that the help window is being used to aid in interaction in the command prompt window; others have shown that many people often show just a small portion of a window to use its information [10].

Whereas Figure 3 demonstrates window switching within one larger task (in that example, writing a piece of computer code), we can see complete primary task switches in Figure 2. Notice how the Command Prompt, IE, and WinDiff windows all become invisible whenever the Outlook or Word email responses become fully visible. A slight difference may be seen in the recurrence of this pattern, however. Although subtask window visibilities occasionally overlapped or are not as precisely sharp as in Figure 3, separate task visibilities maintain their sharpness throughout Figure 2. Future work will include a closer examination of the phenomenon, with the possibility of automatically detecting the difference between subtask switches and primary task switches.

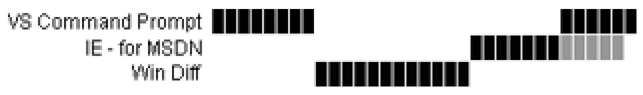


Figure 3. A cut of Figure 2. Notice the sharp change in visibility of different windows all related to a primary task.

7.2.2 Window Types

Another aspect of the visualization is the ability to compare the visibility behaviors of different windows, thus allowing the classification of windows. For example, in Figure 2 the taskbar is fully visible throughout the entire time, meaning that the user has probably set the taskbar to be always on top. The desktop is only visible when the email window is not visible, leading one to conclude that the email window uses all of the remaining desktop space. The email window itself is also interesting, as it appears in short bursts and then disappears. This indicates that the user does not monitor the email window while working on other tasks. Figure 4 compares a cut of the email window from Figure 2 with a cut of a visualization of an email window for a different participant. For the top participant, email is continuously visible with a relatively high percentage of the window showing, indicating that email may be referenced or monitored during the completion of other tasks.

8. FUTURE WORK

The work presented here demonstrates that the analysis of computer usage visualizations reveals that visibility can be a powerful tool in measuring the importance of a window; simply because a window has ceased being the active window does not imply that it is unimportant. Since visibility could be more useful than measuring activation for many situations, a question for future exploration is what other advantages visibility measures have, as well as developing more alternatives for measuring importance and use. The visualizations themselves could be altered to provide different information. For example, a block at

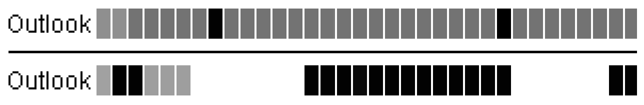


Figure 4. Visibilities of email windows for two different participants. The top image shows that the email window is constantly visible, whereas the bottom image shows how the email window often becomes invisible. Such patterns can aid in classifying the ways that people use windows.

(*x*, *y*) might indicate the overall percentage of screen space (or monitor) that the entire window or just visible portion constitutes. Another possibility is further separation of invisible windows into hidden, occluded, and closed windows to gather a better picture of how the user is interacting with windows. Cautious analysis is in order though, as some have shown that users hide windows and window contents for a variety of reasons [10]. We could even bring the visualization "on-line" to help remind users about their activities; previous work has indicated that visualization can help users remember what they were doing [14].

Whereas work on multiple monitor usage tends to focus on the differences between single monitor and multimon users, one contribution of our work is that we have shown some similarities. For example, tasks often involve coordinating among several windows, using information from one or more windows to aid in interaction with another window. Given more display space, it seems likely that users need less window switching to get information from other windows, because those windows can be displayed without occlusion. But in fact single monitor and multiple monitor users switch windows equally often, signaling that multiple monitor users may use larger windows, or that the phenomenon is not necessarily related to display size whatsoever. However, our data has shown that the *way* that users switch windows is somewhat different in that multiple monitor users are less likely to rely on the taskbar to switch, and that information is spread across many monitors. This provides the future opportunity to explore different window switching mechanisms for these different classes of users.

An additional contribution to the field of HCI from collecting this data on how windows are used is the ability to continuously ask questions about users' window usage habits and to develop new methods for answering those questions. For example, windows are generally active for short periods of time, but looking at visualizations of window visibility, many elements of the display are relatively stable. These apparently two conflicting findings call for enhanced visualization, showing for example which windows were active over a minute's time by combining the window log and event log data. There are also a host of questions we did not present in this paper, including how users tend to position windows both absolutely and relative to other windows.

9. CONCLUSION

As display costs drop and processors and video cards continue to increase in power, the use of multiple-monitor systems or larger displays is likely to increase. It would be difficult to understand how (or, in fact, *whether*) to design for this coming change if one does not understand both how people generally interact with and manage windows, and how multiple monitor practices differ from those of the past. The overall value of the results we present in this paper is that we have starting points from which to further investigate these practices, similarities, and differences. We have also presented several ideas for how to begin novel user interface designs which leverage how users interact with windows across multiple displays.

10. REFERENCES

[1] Beaudouin-Lafon, M. Novel interaction techniques for overlapping windows. *Proc. UIST 2001*, ACM press, 153-154.

[2] Bell, B.A. and Feiner, S. Dynamic space management for user interfaces. *Proc. UIST 2000*, ACM Press, 239-248.

[3] Bly, S. A. and Rosenberg, J. K. A comparison of tiled and overlapping windows. *Proc. CHI 1986*, ACM Press, 101-106.

[4] Czerwinski, M., Smith, G., Regan, T., Meyers, B., Robertson, G. and Starkweather, G. Toward characterizing the productivity benefits of very large displays. *Proc. 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. Man-Machine Studies 38* (1993), 949-983.

[6] Gaylin, K. How are windows used? Some notes on creating empirically-based windowing benchmark task. *Proc. CHI 1986*, ACM Press, 96-100.

[7] Grudin, J. Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *Proc. CHI 2001*, ACM Press, 458-465.

[8] 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.

[9] Hilbert, D. M. and Redmiles, D. F. Extracting usability information from user interface events. *ACM Computing Surveys 32*, 4 (2000), 384-421.

[10] Hutchings, D. R. and Stasko, J. Revisiting display space management: Understanding current practice to inform next-generation design. *Proc. Graphics Interface 2004*, to appear.

[11] Hutchings, D. R., and Stasko, J. QuickSpace: New operations for the desktop metaphor. *CHI Extended Abstracts 2002*, ACM Press, 802-803.

[12] Kandogan, E. and Shneiderman, B. Elastic windows: Evaluation of multi-window operations. *Proc. CHI 1997*, ACM Press, 250-257.

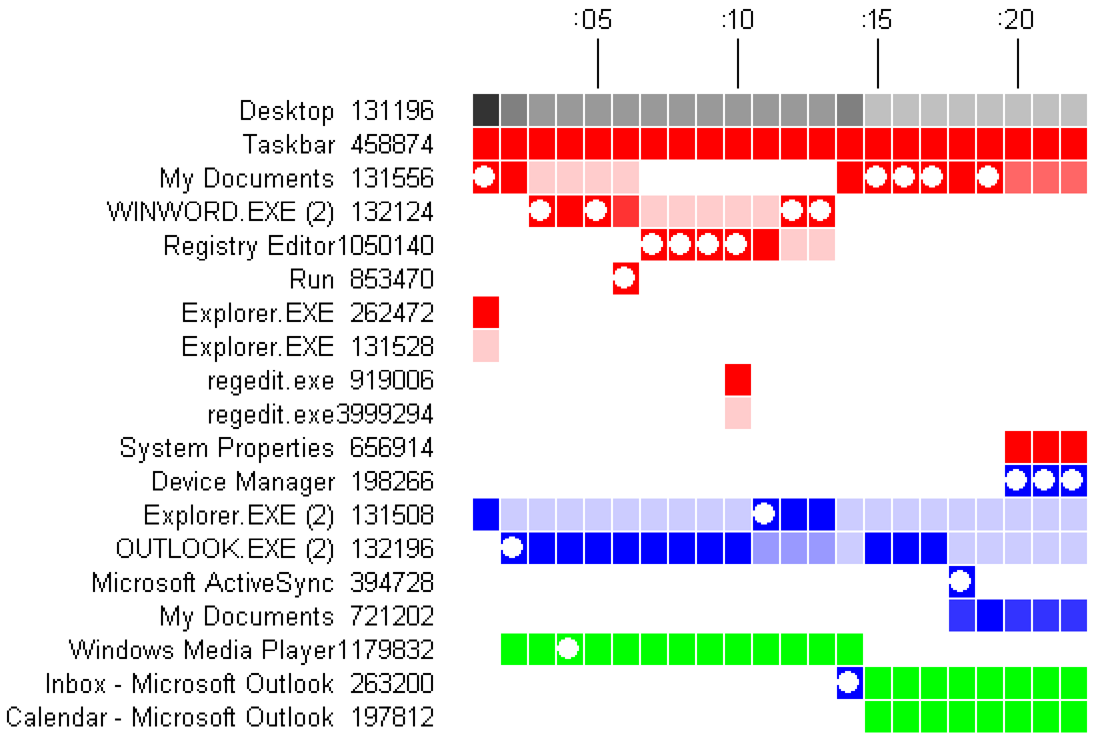
[13] Myers, B. Window interfaces: a taxonomy of window manager user interfaces. *IEEE Computer Graphics and Applications 8*, 5 (1988), 65-84.

[14] Renaud, K. Expediting rapid recovery from interruptions by providing a visualization of application activity. *Proc. OZCHI 2000*, 348-355.

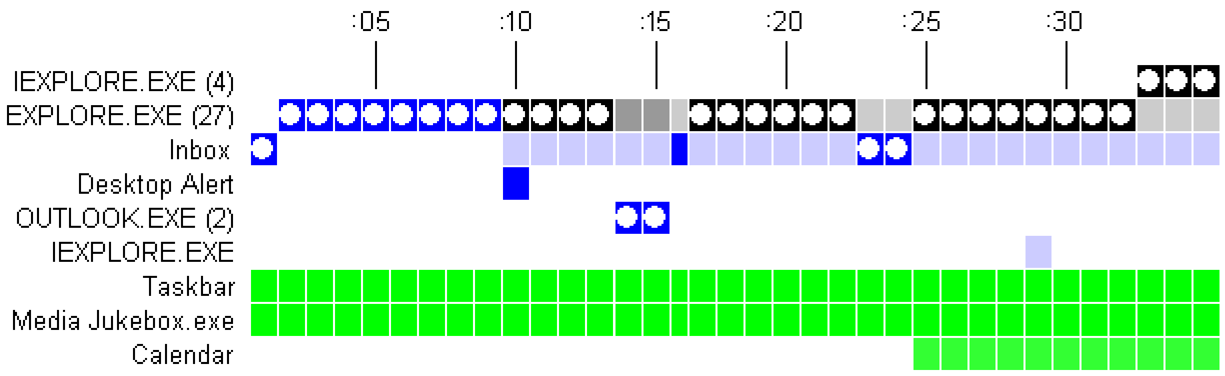
[15] Ringel, M. When one isn't enough: an analysis of virtual desktop usage strategies and their implications for design. *CHI Extended Abstracts 2003*, ACM Press, 762-763.

[16] Robertson, G. , van Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D., and Gorokhovsky, V. The task gallery: a 3D window manager. *Proc. CHI 2000*, ACM Press, 494-501.

[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.



Color Figure 1. A visualization of window visibility from a window log file. The *x*-axis is time in minutes. The *y*-axis has an entry for each individual window. The color of (*x*, *y*) corresponds to a specific monitor; this user has three monitors (red, green, and blue). The amount of shading of (*x*, *y*) indicates the amount of visible area of the window at that time: pure color means full visibility and lighter color means less visibility, with white indicating that (*x*, *y*) is invisible. If (*x*, *y*) is in grayscale, then window *y* was straddling more than one monitor at time *x*. If (*x*, *y*) has a white dot, then window *y* was active at time *x*.



Color Figure 2. A visualization similar to Color Figure 1. The user changes a web browser window so that it straddles both screens. The result is that part of the email inbox is visible throughout the remainder of the window interactions.