An interview-based study of display space management

Technical Report GIT-GVU-03-17
May 2003

Dugald Ralph Hutchings, John Stasko
College of Computing/GVU Center
Georgia Institute of Technology
Atlanta, GA 30332 USA
{hutch, stasko}@cc.gatech.edu

ABSTRACT
There are a number of challenges for researchers in the area of window and screen space management: (1) many systems have been proposed, but little study on people's window interaction habits exists, (2) users of emerging display systems have different properties and needs than users of single-display systems, yet users might also interact with several different types of systems, and (3) evaluation is difficult since habits are unknown but more importantly there are two very different roles that managers must fulfill: allow the user to complete one task through the aid of several windows and be able to switch to or monitor a different task. To begin to answer these challenges, we present a interview-based study of window system users that investigates the way they manage screen space. Results include the characteristics common across all users as well as a classification of management styles. We also present some implications for building and evaluating window and display space management systems.

KEYWORDS
interview, window management, space management, multiple monitors, multiple desktops, user interaction

INTRODUCTION
The personal computer (PC) has undergone dramatic changes over the past 25 years. The huge gains in processor speed and physical memory and the popularity of the Internet have allowed people to use computers in an astonishing variety of ways. Throughout this period, the desktop, and its related window system, have changed very little. Many researchers, organizations, and individuals have built enhanced window systems, some of which have been evaluated on users completing very specific tasks. However, because of the existence of only a few studies of the actual practices of users of window systems (which, for PCs, is nearly *all* users), one cannot be certain whether these specific tasks are representative of the tasks that users complete through window interaction. One must also consider the age of many of these studies, which were completed in the 1980s (e.g. [Bly] and [Gaylin]).

Even supposing the existence of studies on actual window management practices, it is difficult to determine what aspect of a window system to evaluate because defining the "task" of window management is very difficult. Window managers need to serve two roles: allow the user to (1) complete one task through the aid of several windows and (2) be able to switch to or monitor a different task or set of tasks. For example, the writing of this paper was accomplished through a window with the current draft, a window with the previous draft, a window with the intended outline for the paper, several windows with references and resources, and a window displaying a list of images needed for the paper, all of which may or may not have been open simultaneously. Additionally, at several points during the writing of this paper, one author also had to prepare to run a meeting, which required a windows for editing the agenda, viewing a personal calendar, editing web pages, viewing web pages, and checking email. The same author also displayed a portion of his email nearly all of the time, in order to monitor incoming communications, whether they related to the paper, the meeting, or something else entirely. Gauging the effectiveness of the window manager requires measuring *how well* the user could both switch among different tasks and *how well* the user could complete one task; suitably defining *how well* is a major challenge unto itself.

Not only was writing this paper accomplished alongside other tasks, but it was accomplished on more than a single system: a laptop and a multiple-monitor desktop. Stable display systems are becoming more complex and more commonplace (including multiple-monitors, large displays, and multi-device systems such as handheld PDAs and PCs used in tandem). Powerful systems are increasingly able to fit in portable devices, such as laptops. Moreover, the ability to access remote systems graphically, whether through networking [VNC] or emulators [VirtualPC], is also becoming robust and oft-used. This creates a unique challenge for emerging windowing systems: the ability to effectively manage a large display space while still being relevant to the small display areas that have dominated

interaction for the past 25 years.

We have briefly discussed three challenges for window management systems: (1) many systems have been proposed, but little study on people's window interaction habits exists, (2) evaluation is difficult since habits are unknown but more importantly there are two very different roles that managers must fulfill, and (3) users of emerging display systems have different properties and needs than users of single-display systems, yet users might also interact with several different types of systems. Hence in this paper we present results from a study of 17 users in order to fill the research gap left by (1), make some headway in better understanding (2), and determine the elements common to any display system discussed in (3).

RELATED WORK

Work in the area of window management falls into two rough categories: work such as [Henderson], [Badros], and [Beaudouin-Lafon] claim to be beneficial to users because the systems attempt to imitate the way that people use space in the real world; and work such as [Kandogan], [Robertson], [Bell], [Funke], [Stille], and [Miah] claim to be beneficial to users because of the gains seen in task completion time as compared to a standard windowing system. There are potential issues with evaluation in each case. In the former case, note that affordances of computer screens are not necessarily equivalent to affordances of physical objects. Moreover, the ubiquity of personal computers and window systems implies that drawing parallels to the management of space in the "real world" may be unnecessary or even inefficient, as interacting with windows is now an event unto itself. In the latter case, evaluation may be unreliable if the selected tasks are not indicative of actual use of screen space, or if the interface is inefficient for the host of other tasks that the window systems help to support. A few systems give interaction techniques to use space to more easily manage tasks (such as [Robertson]), but do not show a great benefit in actually completing those tasks. This analysis of previous work is certainly not to detract from its contributions. Rather, it is to show the aforementioned inherent difficulty in evaluating window systems and window interaction techniques, as well as to illuminate the surprising absence of work that investigates the actual space management practices of window users (since so few can be cited, with [Bly] and [Gaylin] being notable, but old, exceptions).

In addition to analyzing proposed systems or techniques in the face of multiple-window coordination, one must also consider whether the proposal is effective for managing space and tasks on emerging display systems. Recent work has analyzed the emergence of multiple-monitor systems (multimon) by investigating the differences between multimon and a single monitor [Grudin] and differences between multimon and virtual desktops [Ringel]. One might also consider how the older space management proposals scale to multimon or to very large single displays such as the Perspective Wall [Guimbretière]. Note that we specifically avoid collaborative use of multiple monitors ([Nelson] and [Rodden]), but do leave the matter open for future consideration.

STUDY SETUP

We interviewed 17 people (10 female, 7 male) in their natural workspaces to ascertain how they manage their screen space. Each interview ranged between 30 and 60 minutes and was audio-recorded. The interviewer captured screen contents at least once for each participant using built-in window system or operating system operations and snapped photographs of the working space of each participant. Each participant had the opportunity to prevent the interviewer from capturing either audio or video.

15 users had desktop PCs, 11 of which were single-monitor, 3 of which were dual-monitor, and one user had two independent single-monitor systems on the same desk; 2 users used a laptop exclusively (at least 3 others has laptops on their desks which they either used at home or infrequently at work), one of whom connected the laptop to another monitor when he used the laptop on his desk. Window systems included Windows 2000 (7), Windows XP (2), Mac OS 9 (2), CDE on SunOS (2), Enlightenment on Linux (2), KDE on Linux (1), and the participant with two independent systems ran SunOS's CDE and Windows 2000 (so 6 users used multiple-desktop systems). Occupations included computer science students (5), chemistry students (2), immunology students (2), computer science professors (2), administrative assistants (2), technology support staff members (2), a mathematics student, and a virology researcher. 11 users identified themselves as *constant* users of their systems, 2 as *occasional* users of their systems (meaning that they used their system for a one or two hours per day), and 4 as *fluctuating* users (some days they are constant users and other days they are occasional users or do not use their system whatsoever). Screen resolutions ranged wildly, with the endpoints of one 800×600 pixel display and two 1600×1200 pixel displays.

As the reader can plainly see, the group of users was rather heterogeneous with respect to system characteristics, although homogenous with respect to occupation type. We specifically targeted this group of workers because they were representative of information workers in any company or group and because we predict that they will likely be early adopters of emerging display technologies, if they had not already done so. By choosing users of such a wide variety of systems, we aimed to eliminate any effect that specific systems had on users, as well as determine if specific systems were correlated to specific display management behaviors. We first window management styles that emerged from our study and then present results that cross system lines entirely. First though, we define a few terms to allay any confusion regarding their use.

DEFINITIONS

As the reader will soon discover, it is important to carefully define the idea of *focus*. We use *input focus* to refer to the

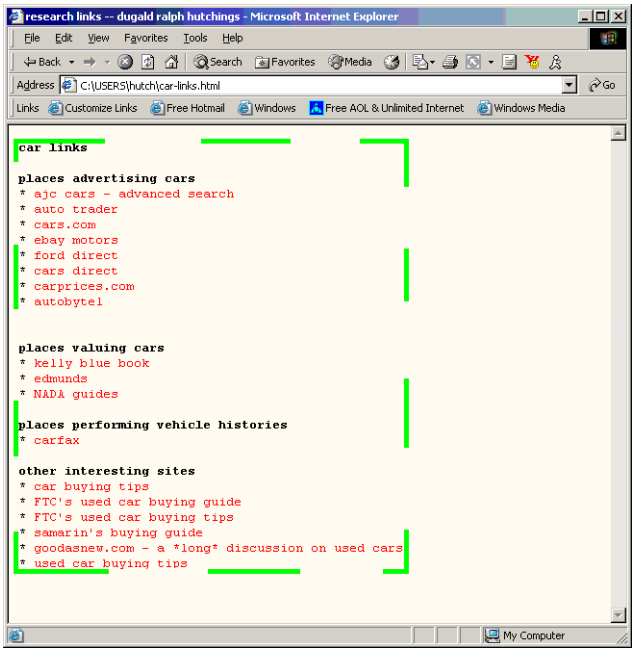

Figure 1: A web browser window, with the information region coarsely marked by the dashed rectangle.

window that has system focus, i.e., the window that exclusively receives input from the user. We use *user focus* to refer to the window that the user is actively viewing, which may or may not have input focus. This is a crucial separation since, for example, the coordination of multiple windows may involve many changes in user focus, but few changes in input focus. In particular, we use *glancing* to refer to the action of shifting user focus without shifting input focus. When we use *focus* without qualification, we mean both input focus and user focus.

We also define the notion of an *information region* of a window. This is any UI component that displays information useful to the user while the window does not have input focus. For example, a classic web browser is depicted in Figure 1, with the information region coarsely encapsulated in a green rectangle. Note how the information region is much smaller than the window, as other parts of the window are useless if the window does not have input focus.

Finally, note that a window can be repositioned in many ways. A window has a width (x-dimension), height (y-dimension), and a depth (z-dimension), all of which can be manipulated directly by the user. Furthermore, resizing a window changes its height or width (and may require a change in depth as well), so resizing may also be thought of as repositioning.

MANAGEMENT STYLES

In this section, we discuss the general ways that people interact with window systems and organize space. First though, let us define a few terms. *Resize* means to change the height or width of a window, *move* means to change the left-to-right or up-to-down position of a window, and *stack* means to change the top-to-bottom position of a window.

Relationship of window size to interaction techniques

People have a variety of ways in which they organize screen space, and few, if any, people organize their windows in exactly the same way. In fact the idiosyncrasies that people have are quite amazing and could make a paper unto itself. Nevertheless, people fall into three rather broad categories: maximizers, effective maximizers, and careful coordinators. *Maximizers* simply maximize every window and use stacking to switch among windows. *Effective maximizers* are a little different. These users have one or more smaller windows with which they frequently interact or glance (such as IM clients or music players), or they have a bank of desktop icons uncovered, as described previously. Users will resize all or most other windows to occupy the remaining portion of the screen and use stacking to switch among these effectively maximized windows. Interestingly, no effective maximizers use "always on top" window features, if such features were available in the window system. When asked about this, each replied in the following vein: "Every once in awhile, I have to fully maximize a window, which means having an always on top window is annoying. It's easier for me to just manually resize the other windows." *Careful coordinators* are those who tend to have many windows visible simultaneously (which means none of them are maximized) or, when they have a maximized window, are working in an application that itself has many windows. Careful coordinators also seemed to have similar *widths* for similar applications. In each case, web browsers, terminal windows, email windows, IM windows, and text editing windows all had the same width, although the length of each window might vary. All but one of the multiple desktop users that we interviewed were careful coordinators (with the other being an effective maximizer).

Participants' techniques for interacting with windows were much more broad. We found five different ways people switched among windows: (1) moving the mouse directly to the window (sometimes requiring a click as well), (2) using the keyboard (usually "alt-tab"), (3) using a window-system supplied interaction area, such as a taskbar, (4) minimizing the current window and stacking the desired window, and (5) moving windows at the top of the stack until the desired window was found (although this technique was rarely used). Most participants indicated that they exclusively use one technique, although a few mixed techniques (mostly mixing (1) with (3)). Among multiple desktop users, methods (2) and (3) were used to switch to entirely different desktops.

The relationship between placement and interaction is very interesting. All maximizers used a taskbar to switch among windows. This makes a lot of sense, as the taskbar is easily accessed regardless of window size, making window switching very easy for this type of user. These users also had similar screen setups and resolutions (single-monitor

and 1024 × 768 or smaller) and all used Microsoft Windows. Although one might expect a similar statement about effective maximizers, this is not the case. Every effective maximizer used direct switching to move between an effectively maximized window and a window or icon elsewhere in the screen. But effective maximizers composed a variety of window systems and interaction techniques (for switching among effectively maximized windows). Windows users used the taskbar, Mac9 users used minimize/restore cycles, and multiple-desktop-system users tended to keep only one effectively maximized window per desktop, so switching windows equated to switching desktops. Careful coordinators either use direct switching exclusively or in concert with a taskbar, although the taskbar seems to be used only when the window to be switched to is nowhere on screen. All careful coordinators except one had screen resolutions greater than 1024 × 768 pixels, and many used multiple desktops.

CROSS-CUTTING RESULTS

It is certainly not surprising that all of the participants, whether occasional users or system administrators, indicated that everyday interaction involves coordination of multiple windows, whether completing one task with multiple windows or managing several tasks, each with one or more windows. However, several factors contribute to the way that people manage many windows occupying the screen space of the participants. We present those factors in the following subsections, and conclude each subsection with an implication for design and evaluation.

Invisibility is as important as visibility

Using information from one window to interact with another window is quite common, whether it be consulting an outline in order to write a paper, compiling email message comments into a coherent digest, grabbing images from a web page, or using documentation to write a piece of computer code. Users employ moving, resizing, and re-layering of windows in various ways to accommodate the use of information. But participants also use these techniques to purposefully hide information as well! We discuss three different ways people hide windows.

One instance of hiding is when focus is directed at several windows in short frequencies or when focus is dominated by only a few windows. Being able to glance at the different windows is unimportant, but being able to quickly access a window is. Since users find it much quicker to change the depth of a window than to move and especially resize windows, this is much preferred over a tiling window manager. Many examples are evident in participant data; a very common example is leaving some portion of an email window visible while completing another task.

Another instance of purposeful hiding is when the interaction in one window (the *main* window) can be aided by the information area of one or more windows (the *secondary* window(s)). Usually, user focus will shift among the entire set of windows, while input focus will mostly remain in the main window. Secondary windows can be distracting for many reasons. One factor is the presence of non-change-blind animations in the non-information areas. However, a portion of the information area can be important for users to view, making minimization of the window impossible. Thus, users will attempt to hide the distracting areas by moving those areas off-screen or by allowing the main window or a secondary window cover the area (a common case among participants for this type of hiding is for advertisements in web pages). Another distracting factor could simply be the sheer amount of information contained in a secondary window, relative to the information that is relevant to the main window. Hiding a large portion of the secondary window(s) allows users to more quickly locate the relevant information and focus on the task at hand. Additionally, resizing the secondary window is often undesirable as the layout of the information is then subject to change, causing disorientation and unnecessary interaction. For example, participant 07 displayed a portion of the outline of a very large document in a secondary window.

Another type of information hiding relates to privacy. All participants use email and many had email programs running constantly. A number of participants also used instant messaging to communicate. Others ran programs that contained secure or proprietary information. This is a difficult situation, since the information is frequently accessed or consulted, yet should remain invisible when not in use. As participant 02, who works in a laboratory, said, "The desktop is not as personal as just one person's vision," or, as participant 14, who has a private office, stated, "I don't want to have [my email] visible on the screen when people walk in. I'm pretty private about it ... hiding things is good." Sometimes users minimize communications client windows or place them on dedicated a desktop to easily and completely hide them. However, because these clients are frequently used, they are often partially hidden behind other windows. In particular, instant messaging requires the focus to switch over short intervals, making minimization or placement to exclusive desktops inefficient due to interaction overhead (although one user of instant messaging placed all communication windows on one desktop).

Implication for design: When developing new systems, worthwhile operations for interaction could include the ability to automatically arrange a selected set of windows so that each one has some portion visible regardless of its depth (for quick access), the ability to hide specific portions of a window without changing the layout of information contained in that window (for both coordination and privacy maintenance). In fact, we discuss the latter later in this paper.

Implication for evaluation: Evaluators of current and new systems could test how easily a group of windows can be displayed so that each has a visible part, and how easily

portions of a window can be hidden by the user, testing separately window coordination and privacy maintenance.

Strict tiling is never employed

An observation related to intentional hiding of windows is that users *never* strictly tile all windows. This is probably not too surprising for users of one-screen, one desktop systems, especially with the importance of window content hiding explained in the previous subsection and the very constrained resource of screen space. However, the same observation holds for users for virtual desktop systems, multiple monitor systems, and even the one participant who employed both.

Strictly speaking, virtual desktop systems cannot allow for tiled windows because by the definition of tiling, all visible windows are displayed on the screen simultaneously. However, even relaxing the definition of tiling to the current screen contents, we found that users did not tile windows. Excluding the reasons described above, we found two additional reasons for hiding contents in other windows. For windows that do not have input focus, it is unnecessary to see the window components because they do not display any information. Some users would cover the components to yield more space to the window of input focus while keeping the window information visible. Other users wanted to temporarily allocate a large amount of space to one window without disturbing the locations of other windows on the screen, since the other windows had been carefully placed. Enforcing a tiling system could make these users much more inefficient. Participant 10 perhaps phrased it best: "[There are] times when I want to look at one window, edit another window, [and] I don't really need to see [all of] what I am editing, so I have them overlap so I can look at the one and focus on the other."

Implication for design: One of the key disadvantages of a tiling window manager is the disturbance caused to carefully positioned windows. Thus tiling systems must allow users the ability to quickly recover to a previous window configuration. Moreover, when systems have more than one display (possibly separated by some amount of physical space), tiling mechanisms may fail to work. Possibly exploring alternate ways of tiling (for example where some parts of a window can be at different depths than other windows) may also be in order.

Implication for evaluation: Since overlapping windows is such a dominant paradigm, comparing any new tiling systems to overlapping counterparts seems to be in order. Furthermore, the two dominant roles (completing one task with several windows and the ability to switch among different tasks rapidly) could be measured.

Empty space is not really empty

When we first started the interview with each participant, we asked the person to explain the layout of the display (which we requested that the participant leave "as is" before we came to interview), and then asked to show a typical layout of the screen. Some participants completely filled the screen(s) with windows, leaving no pixel of screen space without a window. Others left some screen space vacant, but of those participants, all using exactly one desktop left a bank of desktop icons uncovered. See Figure 2 for an example of this phenomenon.

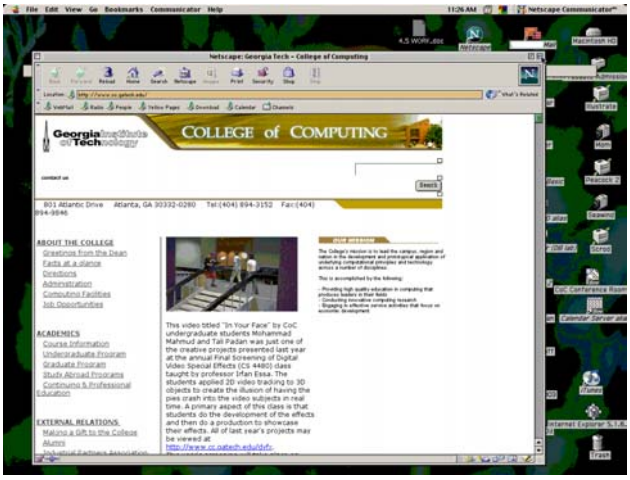

Figure 2: Icons left uncovered on Participant 04's desktop.

Part of the management activities of these people included ensuring that important icons always remained uncovered. Icons serve many functions (sometimes more than one simultaneously). Desktop icons can act as "quick launches" for commonly used applications or file system locations, "status monitors" for events such as print jobs, important interactive components (such as the "Trash" or "Recycle Bin" icon, which chiefly allows people to delete files which will later be recoverable), easily accessed temporary files, or as reminders for the user to do something. The difference between reminders and "quick launches" or "temporaries" is that reminder files need urgent user attention, whereas the other two do not. So while covering the former (and also the interactive icons) might be an annoyance, covering the latter might be detrimental to the work of the user. Users in this category avoid the "maximize window" function specifically to keep important icons uncovered.

Implication for design: Future systems might explore how to designate a group of icons as "non-empty space," where, for example, a "maximize window" function does not cover the icons, but manual resizing of windows will allow the icons to be covered. One might also allow arbitrary regions to remain visible under "automatic" operations (such as *maximize*) while able to be occluded under "manual" operations (such as a resizing of a window).

Implication for evaluation: Evaluators might measure how easy it is for users to keep a certain section of empty screen space visible. For systems that do not explicitly allow such functionality, evaluators might measure how easily users can switch to the desktop then revert to an original window configuration (as well as how obvious it is

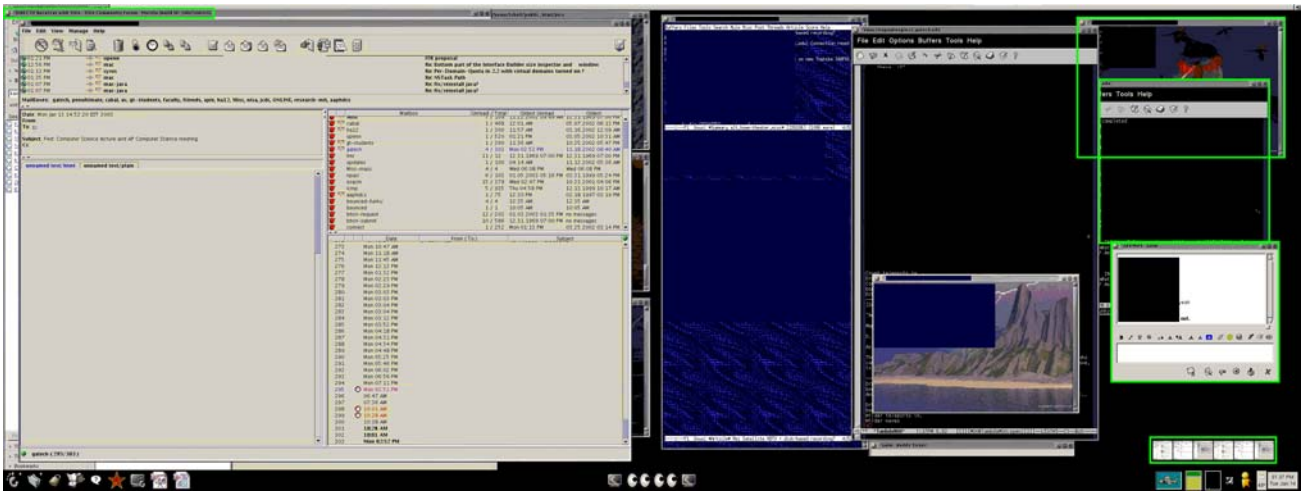

Figure 3: Participant 05's desktop. Windows that act as reminders to return to a task as encapsulates in green rectangles.

to do so).

Windows can act as reminders

Icons are not the only elements of the display system that act as reminders. Often, people are working on one window or one task and must switch to another window or task. Other times, windows automatically grab the input focus of the user, causing the user to abruptly switch user focus to the new window. Whereas in the past, with low power or low memory machine, this situation might call for users to close windows and force themselves to remember to later return to the application contained in the window, users today can simply move one more or windows to the side, to another desktop, or minimized to a smaller screen location and return to it later. These *reminding* windows were especially prevalent among participants who were commonly interrupted (such as the administrative assistants, who both had several windows opened to tasks that had to be abandoned to attend to an urgent task) or participants who constantly monitored and interacted with communication windows (such as participant 05, who said "[There are usually] at least 6 things [in the dock] as reminders to come back to a task [that I have not yet finished]... email is the center of my universe [and] dominates everything I do." See Figure 3 for a view of one of participant 05's desktops). The interesting point to note here is that although information in windows occupying the screen is not used for the task at hand, the display of such information is very important to the user and can aid in switching tasks. Users frequently interrupted all mentioned a desire to have an area on the screen to drop windows that should be returned to later (similar in nature to the use of the Trash/Recycle Bin icon).

Implication for design: Designers of adaptive window managers could heed that "inactive" or "ignored" windows may be very important to the user, and should not be removed from sight under any circumstances. New systems may consider how to place windows that need later attention, perhaps by dedicating a special area for users to drop such windows, or a "super window" that contains all of the reminder windows and uses change-blind animations to cycle through windows that need later attention.

Implication for evaluation: Evaluators should simply note that windows appearing on the screen or minimized away to the taskbar simply represent tasks that need later completion. Evaluators could possibly measure whether users can separate windows needing later attention (such as tasks-in-progress) from groups of windows representing common tasks.

The effect of input devices

Throughout our interviews, we found that the type of input devices available to the participants guided the ways that they managed screen space. There are a number of specific examples, but space prohibits the publishing of all of them. Thus we present the following example.

Participant 11 uses a laptop system that sits in a docking station on his desk as shown in Figure 4. Attached to the docking station is a flat panel display, giving a multimon setup. Initially, 11 used the touchpad mouse on his laptop's keyboard. In order to move the mouse among monitors, one to three additional runs of the finger across the touch pad were needed. Because of this overhead, the second monitor often contained windows which displayed information but seldom or never received interaction (such as web browsers). Then, 11 happened to attach a standard desktop mouse to the docking station. From that point forward, 11 mixed interaction more evenly across the screens. For example, he commonly edits documents on the attached monitor while interacting with emails about the document on the other screen, or edits a web page on one screen while viewing and updating the view on the other screen.

Implication for design: There are many possible avenues for future work. One is to simply study the management

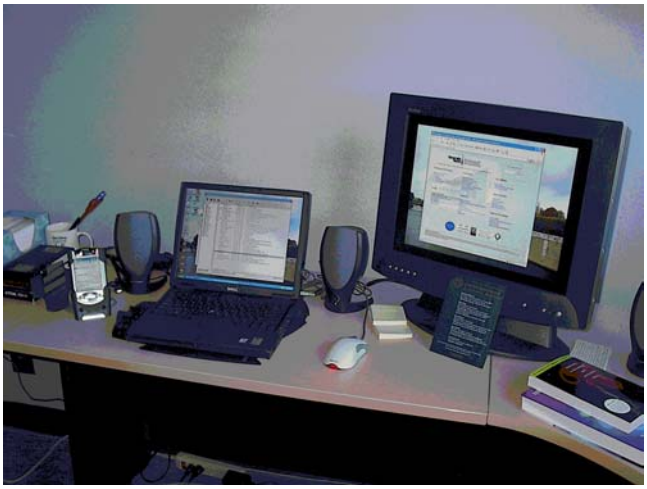

Figure 4: Participant 11's desk. A laptop is connected to a docking station to allow the participant to easily attach a standard desktop mouse (rather than use the touchpad on the keyboard) and another monitor. Furthermore, the PDA on the left is attached to the docking station and in sync with the information on the laptop computer.

affordances of different devices in order to tailor management techniques. However, if supplying this information to the windowing system (which is traditionally a layer on top of the operating system), then this observation might simply be damning. Another possibility is for the window system to track *how* users use input devices, and the dynamically provide information on how the user can more effectively use the system (or determine display techniques that aid the user's habits).

Implication for evaluation: When evaluating a new management system, researchers may take care to have participants use a variety of input techniques to interact with the system to provide additional reliability of results. This will take particular importance as advanced input techniques become more robust (such as voice commands, eye-gaze tracking, and possibly brain-based interactions).

The power of defaults

Most modern window systems offer more than just the standard window operations of add, delete, move, and resize. Microsoft Windows and KDE, for example, have an interaction area called a taskbar that chiefly contains (1) an area for users to place icons as shortcuts to applications and files, (2) an area of buttons for each window that displays a small bit of the title bar of the window and allows users to minimize, maximize, and raise the window. Many users use this area to switch to and interact with different windows and launch applications; some use it exclusively to perform these actions. As a default, the taskbar is located at the bottom of the display, orienting the buttons horizontally, but the taskbar can be moved to any edge of the screen, including the left or right, orienting the buttons vertically. In the default case, once the number of windows reaches a certain size (10 – 15 windows for a 1024 × 768 pixel resolution display), very little, if any, of the title bar can be read and only the icons remain. *n* participants we interviewed indicated that they will start closing windows on the desktop *for no other reason* than to make the taskbar readable. This problem rarely occurs when the taskbar is oriented vertically, but users do not think to move the taskbar to avoid the problem.

Another example concerns the multiple desktops on Sun's CDE. The bottom of the screen has a window with several functions, once of which is to allow users to switch among four different desktops. People we interviewed who use Suns used between two and four of the desktops, but never more, despite the fact that more desktops could be added and two indicated that they could often use more than four (one of whom was a system administrator!). A more careful design of this system window might be able to indicate how people could more easily add, delete, and otherwise manage the desktops.

Implication for design: Developers may begin to consider different default settings for different types of users, different system setups (for example by attempting to understand the number of displays). Additionally, more obvious interaction techniques (such as a button to click to add a desktop) could be useful, although the tradeoff for screen space needs careful consideration.

Implication for evaluation: It is somewhat difficult to gauge the effectiveness of system defaults when measuring users against specific tasks. Longitudinal studies (even as short as one week) would probably be more appropriate, since users will have developed a management style for the system (management styles are discussed later).

The effect of the physical environment

Perhaps the most surprising finding in this section is that the physical environment surrounding the computer system can have an effect on how people use screen space. The amount and layout of space for input devices affects how those input devices can be used, and this notion is best illustrated through examples from our interviews.

Consider the case of Participant 12 (see Figure 5). She has two independent systems on her desk, which means two monitors, two mice, and two keyboards (and occasionally also uses a laptop on the desk). The desk has a drop-down tray in which the keyboard can be placed. Because of her RSI, she uses the tray for both a keyboard and a mouse. This has two effects: (1) the mouse has very little room to move, which requires her to pick it up and drop it frequently when moving the mouse pointer, and (2) she uses the other system for absolutely nothing but email. For awhile, she ran x2vnc [Hubbe], which allows one set of input devices to control multiple systems. However, the amount of pixels that she had to traverse was too large for such a small mouse space: "I tried the one keyboard and mouse, but it didn't work because of the stupid little space for the mouse... I'm limited by the physical desk."

Figure 5: Participant 12's desk. The black keyboard corresponds to the monitor on the left because it is used for all interaction not involving email communication, and thus needs to be in the most comfortable location. The black mouse is also placed next to the keyboard, but has a very small space in which it can be moved.

Participant 09 is another interesting case. He has a heavily customized window manager and multiple desktop system on a laptop that he uses at home and at work. He uses the keyboard whenever possible to interact with the computer, but curiously has customized window and desktop manipulations to occur through mouse movements and button clicks. When asked about this, 09 responded "because the mouse buttons are 'right there.'" On the laptop, mouse buttons are not much further away from keyboard control keys than the rest of the keys themselves, which allows him to use the mouse buttons in many ways, including surrogate keyboard keys. He indicated that if he used a desktop system, he would probably switch to keyboard shortcuts in order to switch more quickly.

Implication for design and evaluation: Unfortunately, there is very little that we can do about this aspect because a system designer rarely has control over the physical environment in which the system will be placed and used. Furthermore, users at work rarely have control on the physical devices available to them for interaction (for example, participant 12 has been unable to secure funds to purchase a more appropriate mouse).

DESIGN DIRECTIONS

Technology marches onward: displays support higher resolution at lower cost, and operating systems and video cards support more than one physical display on the same computer. The analysis of our interviews indicate that the maximizers will grow extinct as the technology that used to engender such behavior also dies. The effective maximizers and careful coordinators that exist now could be aided by rethinking window interaction. We present some ideas below based on our findings.

Invisibility and Rethinking Minimize & Resize

We noted previously that many instances of window manipulation involved the purposeful hiding of window contents. Other than "minimize," window systems do not have built-in functions that help users hide window contents (although yet again the games industry has led the way for the rest of the field by including in their games "boss keys" that allow the user to place a fake picture of a spreadsheet or other work-like application on the screen at a moment's notice). Moreover, hiding window contents (and sometimes keeping other contents visible) is important to users regardless of the amount of screen space that they have available. So one possible direction is to aid users in managing space by developing operations that better decrease the amount of space occupied by a window.

Relevant Regions

One idea for diminishing the amount of space occupied by a window is to allow the user a simple way to tell the window system what parts of a window should be visible. Consider for a moment AOL's IM client window shown in Figure 6. It not only contains a list of people who are also online, but a menu, a logo, an advertisement, a label indicating who's buddy list this is, and two banks of large graphical buttons for interacting with the client, all of

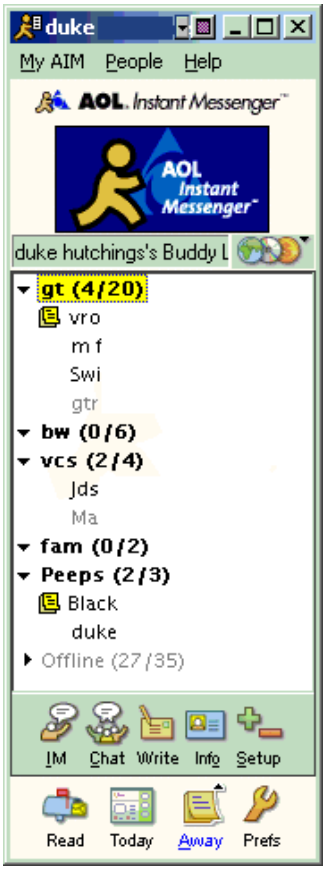

Figure 6: The AIM client window. Notice how many items in the interface are irrelevant when the window does not have input focus.

which is useless when the window has user focus but not input focus. Moreover, if a user wants to send a message to a buddy, the user need only click on the buddy's name to open a new IM window. The relevant region of the window is only the buddy list for most of the time that the window is on the screen.

So, we could diminish the amount of space occupied by the window by simply allowing the user to draw a rectangle about this relevant region, and removing the remainder of the window contents, so the user can treat the region as a window unto itself. Note how the standard version of resize would not accomplish the same thing. If we resize the window, the only part that becomes smaller is the buddy list, which is exactly the important part. Additionally, if we just scaled the window to the size of the relevant region, it is unlikely that we could read the text of the list, making glancing useless. Finally, relevant regions is an ideal candidate for the window manager because they are user-defined and may cover sections of UI components (and users could use the regions regardless of applications).

The idea of relevant regions could also be extended to other areas of space management for larger screens or multimon environments. Consider the task of editing a web page: commonly, the user has an editing window and a browser, and the user must move the mouse back and forth a considerable distance from the editing window to the browser to update the changes by clicking the refresh button. Suppose though that the user could draw a relevant region around the refresh button, and *make a copy* of the region that could be treated as its own window, even though it is just a view onto the original window. Now, the user could place the refresh button right next to the editor, and just click the button directly and view the document in the other screen. This allows the user to *remotely control* applications, especially when they related to the task at hand. Another example might be copying email UI components to all of the screens and being able to control the email display without having to constantly relocate the mouse.

Rethinking Maximize

On a small resolution, single-display machine in which no other windows need to be visible all of the time, the maximize function serves as a nice way to quickly allocate screen space to a window. However, we saw a group of users who effectively maximize. We could alter maximize slightly to be just the region of the screen for such windows, rather than making users resize windows all of the time. One example of allowing users to do this easily would be to allow for the simple drawing of a "maximize to" region on the desktop, where all maximized windows would be resized and placed (perhaps similar in nature to tabbed windows [Beaudouin-Lafon]). The advantages of such a concept is that windows do not have to be assigned "preferred sizes" and "preferred locations" and that when the entirety of the screen space is needed, users can still employ direct manipulation techniques to resize the window further. A careful design is needed though, as we noted previously that users often do not change window system settings unless it is obvious how to do so. Note that this technique might also extend to multiple-monitor systems, where each monitor has a "maximize to" region, given that the window system is aware of the number of monitors in the system.

Window-level Undo

A large number of applications have an "undo" function that allows a user to return to a previous state of the application, usually to correct an error made in input. Surprisingly, or review of literature in the area reveals that this idea has never been applied to window and space management. As mentioned, a class of users carefully arrange their windows, and many techniques for interacting with windows can disturb the stacking order of windows. Furthermore, people make mistakes in arranging their windows. Being able to revert to a previous state of window arrangement can be very helpful. Obviously, a corresponding "redo" function would complement a window-level undo function.

Undo is somewhat hard to implement, however. For example, how does one "undo" from creating a window? One possibility is to minimize windows created since the last window interaction. A more problematic recovery is for the closing or deleting of a window. We could introduce a "trash" or "recycle bin" functionality for the windows themselves, but sometimes not allowing users to directly cease an application could have deleterious effects (such as hogging the CPU). A full implementation would require careful thought and analysis of the situation, but it appears that undo functionality could be extremely helpful.

CONCLUSION

We have presented several challenges for building and evaluating window management systems: (1) many systems have been proposed, but little study on people's window interaction habits exists, (2) evaluation is difficult since habits are unknown but more importantly there are two very different roles that managers must fulfill, and (3) users of emerging display systems have different properties and needs than users of single-display systems, yet users might also interact with several different types of systems. The study we presented begins to meet these challenges, suggesting many ways that people can build and evaluate systems. In particular, we attempt to identify the overriding characteristics of space management, regardless of window manager and display space configuration. Furthermore, we presented some methods that attempt to satisfy key findings from our study.

Display spaces will simultaneously become more complex (by adding more monitors to single systems and increasing the resolution achieved by each monitor), yet many systems and "views onto remote systems" will remain in a small display world. Because space management sits above all of the interaction activities of users, it is a worthy endeavor to continue to analyze how space is used and what

possibilities can be developed to aid users' tasks. Moreover, as the application areas of computers continue to broaden, we should continue to study people's habits in order to effective evaluate the possibilities. The windows onto the world are unlikely to change, but current interaction should continue to be refined and explored.

REFERENCES

Badros, G.J. et. al. SCWM: An intelligent constraint-enabled window manager. In *Proc. AAAI Spring Symposium on Smart Graphics 2000* (Cambridge, MA), AAAI press, 76 – 83.

Beaudouin-Lafon, M. Novel interaction techniques for overlapping windows. In *Proc. UIST 2001* (Orlando, FL), ACM press, 153 – 154.

Bell, B.A. and Feiner, S. Dynamic space management for user interfaces. In *Proc. UIST 2000* (San Diego, CA), ACM Press, 239 – 248.

Bly, S. A. and Rosenberg, J. K. A comparison of tiled and overlapping windows. In *Proc. CHI 1986* (Boston, MA), ACM Press, 101 – 106.

Funke, D. J. et. al. An approach to intelligent automated window management. *Int. J. of Man-Machine Studies 38*, (1993), 949 – 983.

Gaylin, K. How are windows used? Some notes on creating empirically-based windowing benchmark task. In *Proc. CHI 1986* (Boston, MA), ACM Press, 96 – 100.

Grudin, J. Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *Proc. CHI 2001* (Seattle, WA) ACM Press, 458 – 465.

Guimbretière, F. et. al. Fluid interaction with high-resolution wall-size displays. In *Proc. UIST 2001* (Orlando, FL), ACM Press, 21 – 30.

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.

Hubbinette, F. x2vnc.
http://www.hubbe.net/~hubbe/x2vnc.html

Kandogan, E. and Shneiderman, B. Elastic windows: Evaluation of multi-window operations. In *Proc. CHI 1997* (Atlanta, GA), ACM Press, 250 – 257.

Miah, T. and Alty, J. L. Vanishing windows: An empirical study of Adaptive Window Management. In *Proc. Computer Aided Design of User Interfaces 1999* (Louvain-la-Nueve, Belgium), Kluwer Academic Press, 171 – 184.

Nelson, L., Denoue, L., and Churchill, E. AttrActive windows: Dynamic windows for digital bulletin boards. In *CHI Extended Abstracts 2003* (Ft. Lauderdale, FL), ACM Press, 746 – 747.

Ringel, M. When one isn't enough: an analysis of virtual desktop usage strategies and their implications for design. In *CHI Extended Abstracts 2003* (Ft. Lauderdale, FL), ACM Press, 762 – 763.

Robertson, G. et. al. The task gallery: a 3D window manager. In *Proc. CHI 2000* (The Hague, The Netherlands), ACM Press, 494 – 501.

Rodden, T. et. al. Designing novel interactional workspaces to support face to face consultations. In *Proc. CHI 2003* (Ft. Lauderdale, FL), ACM Press, 57 – 64.

Stille, S., and Ernst, R. Adaptive layout calculation to handle the visual chaos in graphical user interfaces: A retrospective on the A^2DL-Project. In *Proc. Computer Aided Design of User Interfaces 1999* (Louvain-la-Nueve, Belgium), Kluwer Academic Press.

VirtualPC. http://www.connectix.com.

VNC. http://www.uk.research.att.com/vnc.