Supporting Lightweight Customization for Meeting Environments

Edward Tse and Saul Greenberg
University of Calgary
2500 University Dr. N.W.
Calgary, Alberta
T2H 1N4, Canada
Tel: 1-403-210-9501
[tsee, saul]@ cpsc.ucalgary.ca

ABSTRACT
Digital wall-sized displays commonly support authoring and presentation in face to face meetings. Yet most meeting applications show not only meeting content (i.e., the material being developed) but authoring tools as well – the usual controls, palettes, and menus. Attendees are distracted when the author navigates the (usually complex) interface as part of the authoring process the tools themselves unnecessarily clutter the display. The problem is that current customization techniques are not suited for meeting environments as complex customization interfaces take attention away from the meeting agenda thus making customization a socially unacceptable practice.

In this paper, we present the solution of *lightweight* customization, a customization technique designed to minimize time and cognitive effort. This paper illustrates lightweight customization through two implementations: First, *customized views* provide a scribe with full application functionality while presenting the important presentation content to the other meeting collaborators on a secondary projected display. Second, *customized interfaces* allow meeting collaborators to rapidly recall previous functionality and build customized interfaces through a history of previous actions.

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

General terms: Customization, History, Views, Interfaces, Meetings

Keywords: Lightweight History Customization, Customized Views, Microsoft Office

INTRODUCTION
This research began with a preliminary pilot study performed at Smart Technologies. This study explored the use of conventional desktop applications on a large interactive display (i.e., a Smart Board). The purpose of this task was to understand the nuances of working on a large wall sized display.

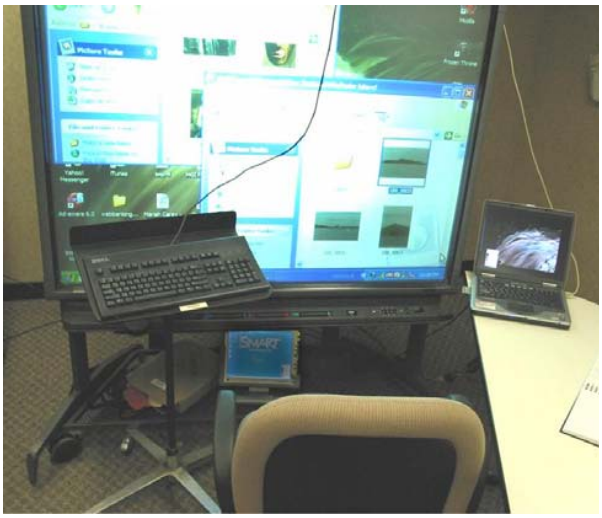

Figure 1. The pilot study setup

The task was to organize and edit a collection of photos using Adobe Photoshop and the Microsoft Windows XP Explorer shown on a large Smart Technologies DViT Smart Board (see Figure 1). The Smart board could be lowered so that all points could be reached while seated. Video recordings were taken and analyzed for one pilot study.

The pilot study revealed two significant problems that motivated my exploration of customization: Screen clutter and keyboard shortcuts.

Screen Clutter
Screen Real Estate was a serious problem for touch interactions over a large display. It was not uncommon to have 30-50% of the screen real estate devoted to graphical user interface elements. For example, windows task bars on the side of each explorer window used up the majority of the screen space in the photo organizing task. While the task bars could be disabled by adjusting the folder settings, this option was never used in the pilot study because the participant did not want to lose focus on the task at hand.

Cite as:
Tse, Edward and Greenberg, Saul (2005) Supporting Lightweight Customization for Meeting Environments. Report 2005-784-15, Department of Computer Science, University of Calgary, Calgary, Alberta CANADA T2N 1N4. April..

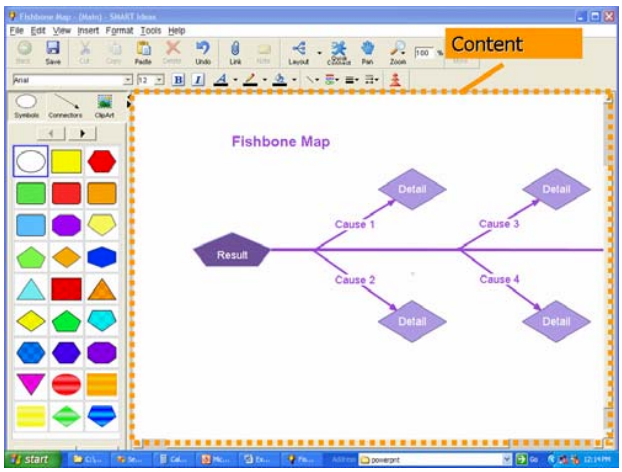

Figure 2. The Smart Ideas Meeting Software. The meeting content is highlighted.

This same problem occurred for Adobe Photoshop where the floating tool palettes made it difficult to interact with the photo canvas. These palettes were eventually moved off of the main display and instead, keyboard shortcuts were used. The repositionable keyboard was moved to the left side of the screen so that it would not occlude any part of the application.

These problems are also reflected in current meeting software. For example, the user interface in an early version of the Smart Ideas Concept Mapping Software (Figure 2) consumes 42% of the total screen real estate. By displaying meeting notes on a large public display, participants can suggest corrections to the minutes and quickly recall previously discussed ideas. However, the public display distracts from the meeting agenda when one explores elements within the user interface.

Keyboard Shortcuts

While keyboard shortcuts were a useful replacement for tool palettes and menus, problems arose when the participant forgot the appropriate keyboard shortcut and made a wrong guess. This would lead to an unexpected action or an unwanted dialogue opening up. The participant would then have to undo the previous action or close the dialog box and then search through the menus to read what the keyboard shortcut was. The participant did not write down the keyboard shortcuts because the participant did not expect to forget the keyboard shortcut so often. This indicates that keyboard shortcuts should be simple and that visual references should be provided so that keyboard shortcuts do not need to be memorized.

Alternatively, customized interface could have been built but this is not practical as customization is a complex and highly cognitive task. Customization interfaces such those in Microsoft Word require one to stop what they are doing to enter a customization mode. Then, they must remember the functionality that is to be customized (e.g., Searching through categories of functionality in the Office customize dialog, Figure 3). This is not suitable for meeting environments as this takes time and focus away from the goals of the meeting.

These two findings formed the basis of my motivation for exploring lightweight customization in meeting environments. Small usability issues such as having to enter a menu to reduce screen clutter or having to write down keyboard shortcuts dramatically affected their use in this time sensitive environment.

The fundamental problem is that customization is a feature designed exclusively for experts. The assumption is that experts will be able to easily navigate through complex menus, use special customization modes, and memorize a large number of keyboard shortcuts. This introduces hurdles to customization that effectively prohibit its use in meeting environments.

To resolve this problem this paper introduces the concept of lightweight customization: A technique for minimizing the cost of customization so that it is suitable for a meeting environment.

TERMINOLOGY

Customization

This paper we define customization as any variation of display or use from the standard display and settings provided in the out of the box application. Customization does not necessarily require any explicit user effort (e.g., entering a customization mode), thus moving a tool palette in Adobe Photoshop to be considered an instance of customization.

Meetings

This paper argues that existing display and interface customizations are inadequate for meeting environments. In this paper we focus on two types of meetings environments.

1. ***Brainstorming meetings*** where one person takes on the role of a note taker or "scribe" and every member is an

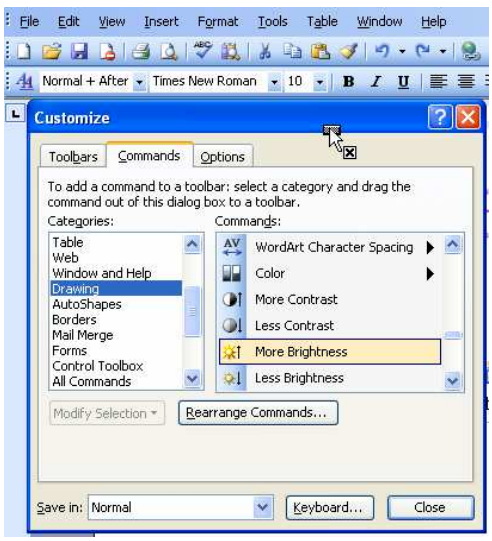

Figure 3. The customization interface in Microsoft Office

intellectual contributor. E.g., A product features meeting where all the notes are being written on a projected screen.

2. ***Planning meetings*** that involve layout design tasks. For example, laying out a weekly newspaper where contributions have different roles (e.g., photo editor, layout editor) and dynamically changing tasks (e.g., photo editing, caption editing).

Our description of brainstorming meetings is similar to Mantei's (1988) description of the designated scribe. That is, one person (e.g., a secretary) would remove themselves from the conversation and focus solely on the task of note taking. The primary difference in brainstorming meetings is that the scribe is not a secretary but rather a meeting contributor who is knowledgeable about the discussion domain. The meeting scribe thus becomes a specialist who can effectively capture meeting notes while also intellectually contributing to the meeting.

Planning meetings are similar to Mantei's (1988) rotating scribe protocol. Each person in the meeting has their own networked personal computer while meeting information is displayed on a large public display. Mantei describes how people skilled in using personal workstations were more likely to take on the role of scribes. In this paper we take the role of the rotating scribe a step further by focusing on meetings of specialists with clearly defined roles in the meeting environment. For example, in a newsroom meeting there might be a photo editor, a news editor, an entertainment editor and a chief editor, each with very different roles and tasks in the meeting.

LIGHTWEIGHT CUSTOMIZATION

While most previous work focuses on making customization possible [e.g., 1, 2, 6, 7], this paper focuses on making customization lightweight. We believe that customization fails in the meeting environment for two reasons:

1. ***Time Cost***: customization requires a member to take extra time (sometimes between 5 to 10 minutes) to learn and use a customization interface. Since many meetings last less than an hour, this makes customization an ineffective use of meeting time.
2. ***Attention Loss***: the attention of a meeting member has to be diverted away from the meeting agenda in order to perform the customization task. This makes customization a socially unacceptable behaviour in meeting environments.

Lightweight customization is a technique designed to mitigate the aforementioned problems. Its core principles are:

1. ***Require Minimal / No Time Effort***: Customization should be provided automatically and should not use customization modes. Any user customization effort must be justified with a proportionate gain in meeting productivity. The time spent customizing should at least match the time saved in improved meeting productivity.
2. ***Minimize Cognitive Effort***: Customization must not be a cognitively demanding task so that it does not divert the attention of the meeting collaborators away from the meeting agenda.

Following the description of related work, this paper describes the application of the principles of lightweight customization to views and interfaces.

RELATED WORK

Customized Views

Presentation programs such as the Microsoft PowerPoint Slideshow allow meeting content to be exclusively displayed on a large projected display. Since these programs are designed as independent applications there is no direct correspondence between the content on the projected display and the scribe's view. For example, while textual changes in the slide editor get reflected in the slideshow slide navigations are not replicated. Also, Microsoft PowerPoint provides a "Presenter View" application with a direct correspondence to the scribe view but it has no editing capabilities and thus would not be suited for a brainstorming meeting environment.

Recent research has explored real time user specified screen captures. Tan et al. (2004) presented WinCuts: an interaction technique for manually selecting regions of a screen for personal use or presentation on a large display. For example, an end user could select regions of an email client to create a simplified email client with significantly less screen real estate. Fujima et al. presented a similar technique that allowed regions of a web page to be displayed in an uncluttered form. Both of these systems provided a direct connection between the customized view and the original application that was screen captured. This means that clicking on a button in the customized view is equivalent to clicking on a button in the scribe view.

Manually customizing views takes time and focus away from tasks in a meeting. Most applications have very obvious content windows (e.g., the Mind Map in Figure 1) that would be tedious and prone to error if required to specify the correct region each time the presentation application was run. A customized view should be able to automatically project the region of interest on the large projected display without end user intervention.

Customized Interfaces

There is a large body of early work in the area of programming by example and customization, notably Cypher (1993) and Greenberg (1993) describe early programming by example and customization systems using graphical user interfaces. While Cypher surveyed existing programming by example systems, Greenberg created a graphical user interface customization that allowed end users to rapidly recall previous command line functions through a history list of recent commands. These systems were designed as general customization techniques for desktop computers rather than for groups of individuals in a meeting environment.

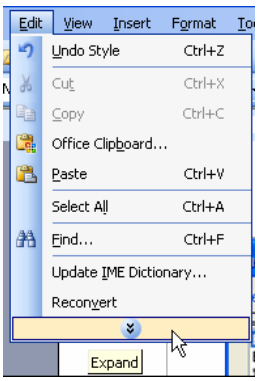

Figure 4. Adaptive Menus in Microsoft Word

More recently, researchers have explored the application of customization and history systems in applications such as Microsoft Word and Internet Explorer.

McGrenere (2002) explored the provision of multiple customized interfaces in Microsoft Word to simplify interfaces based on their frequency of use. Thus, commonly used menus and toolbar items would be available in the simplistic interface while infrequently used items would be accessibly in an intermediate interface and full application functionality would be available through an advanced interface. End users could switch between interfaces by selecting an item in a drop down list box. Simplified interfaces such as the adaptive menus in Microsoft Word require end user effort to access infrequently used functionality. For example, Figure 3 shows an adaptive menu in Microsoft Word that requires an end user to click on an expand button or hover over a menu to access infrequently used functions such as "Replace".

People constantly revisit web pages. Studies have shown that 81% of all web navigations are page revisits [McKenzie, et al., 2001]. Kaasten (2001) exploited this fact and created a history system for Internet Explorer that integrated Back, History and Bookmarks in a web browser. The history system provided web page thumbnails and titles in a recency ordered list that moved duplicates to the most recent position. Customized interfaces use a similar history mechanism for repeating commonly used functionality in meeting software.

Commercial products such as Microsoft Office and Aliaswavefront Maya offer interface customization but require the user to enter a customization mode. I emphasize that secondary customization tasks prohibit customization in meeting environments because they require meeting contributors to shift their focus to the customization mechanism rather than the meeting agenda.

LIGHTWEIGHT CUSTOMIZED VIEWS

In a brainstorming meeting, scribes need to print meeting notes, reference previous week notes and link attachments. Essentially they need access to most of the meeting software functions. While existing meeting software (e.g., Smart Ideas, Figure 2) provide a full screen mode that increased the size of the content area it has limited capabilities. The full screen mode is not normally used in brainstorming meetings as the scribe has to continually switch between the full screen mode and the standard user interface.

Consequently, I developed a technique to automatically capture the meeting content and resize it to completely fill

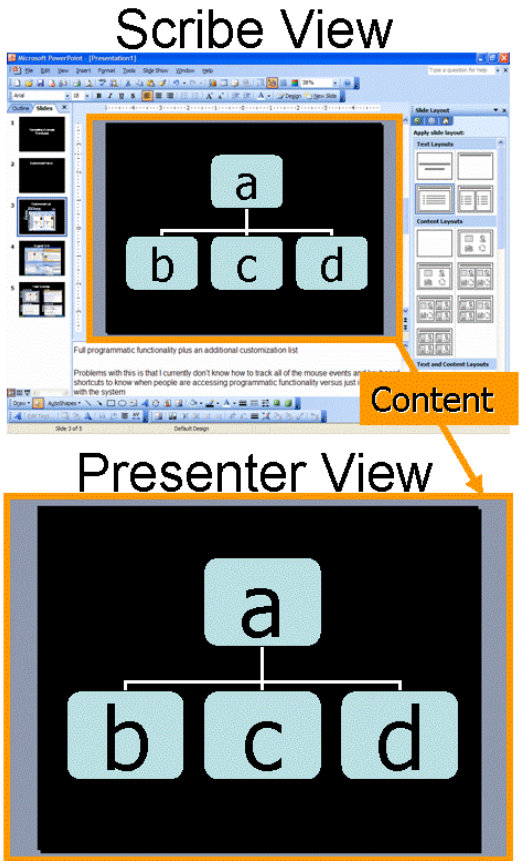

Figure 6. The Scribe and Presenter Customized Views

a large projected display without the use of a specialized view mode. This technique was applied to Microsoft Office applications such as PowerPoint, Word and Excel. As seen in Figure 6, the scribe has complete access to all of Micro

soft PowerPoint's functionality. They could easily open a print dialog, insert an attachment or email the slides to the meeting collaborators. The scribe does not need to modify their interactions with Microsoft PowerPoint in any way, PowerPoint remains fully functional.

The major difference between Customized View and existing manual selection techniques [e.g., 6, 7] is that the content windows are selected automatically. This follows the first lightweight customization principle: require minimal / no time effort. The content window of PowerPoint in Figure 6 is automatically grabbed and expanded to fit the full size of the secondary display. Currently, the only user effort required to use Customized Views is that the end user must start the Customized View application. Alternatively, one could begin by running the customized view application, it would automatically detect that PowerPoint was not running and would automatically start an instance of PowerPoint. I argue that this effort is justified because the cost of starting the application is small compared to the benefits.

In addition, the Customized View allows sequential interaction on the scribe view and the presenter view. That is,

only one person can control the cursor at any given time. It allows someone using an interactive Smart Board to manipulate objects and enter text.

Implementation

The automatic screen capture is completed using a screen capture utility called Grouplab.WidgetTap [8]. This application uses a Microsoft Spy like utility to perform text queries on the names of all windows. The Customized View application searches for a predefined content window within a particular named process (e.g., powerpnt). The first window that matches the name query is returned to the Customized View application. If no window is found an instance of the appropriate application can be started automatically. The handle of the window can be used to obtain the size and location of the content. The Grouplab Collabrary [9] is used to capture the content on a timer interval and stretch the image to fit on a window placed on the secondary monitor.

For interaction on the secondary display, customized views use a technique similar to Tan's Wincuts [6]. When the cursor is moved to the presenter view, an artificial cursor is drawn on the presenter view while the real cursor is moved to the equivalent position in the scribe view. The input from the presenter view is scaled so that it accurately corresponds to a position in the scribe view. If both the scribe and the presenter try to interact simultaneously the mouse cursor will move around in strange ways. This is a limitation of the one user per computer assumption of current operating systems.

Discussion

The concept behind customized views is to automatically display the most pertinent screen information (i.e. the content) to the collaborators of a meeting. By reducing the user interface clutter meeting members could focus on the brainstorming activity and/or meeting agenda without the risk of losing focus when some user interface task was performed by the scribe.

While this technique does not allow live videos to be displayed, I emphasize that it was designed to display meeting notes where the important window content usually comprises of text and (still) images or drawings.

Lightweight customized views are not suited for all presentation meeting environments. If someone was giving a tutorial on using Microsoft Word, they would want to show the entire user interface. This approach is targeted at brainstorming meetings where the focus should be on the content rather than the user interface.

A similar approach could be used for school teachers as they would have full access to the user interface of the application and only make the important content visible to their students. Any drawings or annotations made on the scribe view would be immediately reflected on the project display. For example, a school teacher could bring her wireless tablet to ask a student to write down the solution of 3 * 8 = ?, the writings of the student would be immediately shown on the large display so that their thoughts could be shown to the rest of the class.

While this early implementation has focused on the provision of customized views where the content window is fairly obvious (e.g., Powerpoint, Smart Ideas), there are cases where the content window is not immediately apparent (e.g., a perspective view in a 3D scene). In such systems, a customized view could provide the scribe with several options of view for the secondary projected display. For example, a 3D animation system could specify which camera they would like projected on the secondary display.

LIGHTWEIGHT CUSTOMIZED INTERFACES

In a planning meeting each member is equipped with a networked personal computer and has a specific role (with varying tasks). For example, while a photo editor could memorize the functionality needed to perform a particular computer task there is the risk that the shortcuts may be forgotten. Our goal was to exploit the benefits of customization in meeting software while at the same time minimizing time and cognition effort.

There are two components to our lightweight customization interfaces: a history list that reduces the amount of time required to access commonly used functionality and a customization window that minimizes cognitive effort.

The subsequent section describes the combination of both lightweight customized views and interfaces to support roles in a meeting environment.

History List

The purpose of the history list is to provide the meeting collaborator a simple and easy means of quickly accessing commonly used functionality. By detecting all toolbar selections, the History list displays previously used functions based on recency. For example, the oval function in Figure 7 appears on the top of the list because it was most recently selected. Duplicate entries are automatically moved to the top of the list (e.g., clicking on "More Contrast" in Figure 7 would move More Contrast to the top of the list.

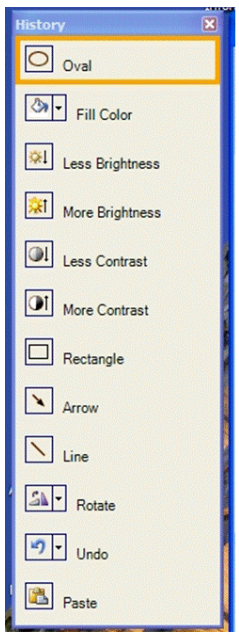

Figure 7. The History Window

The list consists of sets of icons and text descriptions. If a history list item is hovered over in the Microsoft Office application an orange outline appears around that item. For example, in Figure 7, the Oval item is highlighted because the mouse cursor is currently hovering over the oval button in the Microsoft Office Application (see Figure 9. The outline is designed to help the end user to understand the direct correlation between selections made in the

meeting application and their appearance in the history list.

Clicking on an item in the history list performs the operation on the meeting application. Clicking on an item in the history list also brings it to the top of the list since that becomes the most recently used function. For example, clicking on the Line button in the history list would move it to the top of the list.

End users can exploit the fact that the most recently used item in the history list is moved to the top. For example, in Figure 8, if I clicked on copy, paste and rotate 90 degrees in Microsoft Office, the top item in the history list would be rotate 90 degrees followed by paste and finally copy. By clicking on the copy item in the History list its position would move to the top of the history list followed by rotate 90 degrees and paste. Thus, if I did not move my cursor it would be hovering over the paste item in the history list. By clicking in the same location I could rapidly call paste and rotate. This methodology allows one to build temporary macros through the history list without having to go into a special macro mode. This means that an end user does not need to plan ahead to record a macro to be able to exploit macro functionality.

While the macro capability may seem incidental and clumsy since an end user must repeatedly click in the same location, its design is rooted in the principles of lightweight customization. Having to enter a specialized macro mode (e.g., those in the programming by example community) is enough to prohibit macro usage in a meeting setting. By automatically displaying a history list of previously used functions users can exploit macro capabilities immediately after any sequence of actions are performed. There is no need to remember what functions need to be recorded in a macro since they are automatically saved in the history list. The user need only locate the first action in the history list and repeatedly click until the total number of steps has been achieved. Thus the effort and cognitive load is minimized. We argue that the benefits of having an always on macro capability that minimizes cognitive overhead outweighs the cost of having to click on a history item multiple times.

Like any macro system, the macro operations only work on a sequence of actions that match the object selection. For example, you cannot apply bold to a photograph. Certain programming by example [1] features such as variables, loops and conditionals are not included in lightweight customized interfaces because they violate the minimizing cognitive effort principle. That is, it is difficult for a non programmer to use an interface supporting a conditional loop because properly using a conditional loop increases the cognitive load on the end user.

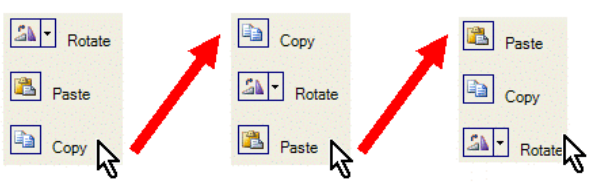

Figure 8. Using the history list to rapidly copy, paste and rotate an object.

Lightweight Interface Customizer

To mitigate the issue of memorizing keyboard shortcuts of the floating keyboard pilot study I required a customization interface that would provide simple keyboard shortcuts and a visual reference. Rather than requiring the end user to memorize complicated multi key sequences I wanted functionality to be literally available at the finger tips of the user.

For this reason, I provided a customization window with room for exactly five items. This allows one hand to be dedicated to switching between commonly used functions while the other hand could operate a mouse. This design is similar to those used in First Person Shooter (FPS) games where commonly used functionality is available at a player's fingertips (e.g., using the W, A, S and D keys to navigate in a virtual game environment). The main difference between the lightweight interface customizer and FPS games is that FPS games require a specialized mode in order to change the key bindings.

The customization window displays up to five items in a horizontal fashion with clearly labelled mapped function keys so that the user does not have to memorize any keyboard mappings. For example pressing the F8 key in Figure 9 would undo the previous action while F9, F10, F11 and F12 could be used by a graphic artist to rapidly change the drawing mode.

Similar to the workbench system for Unix commands described in [2] items can be added to the customization window by clicking on an empty slot on the customization shelf and selecting an item from the history window. This two click procedure was chosen instead of click a drag mechanism since this technique was designed for touch sensitive displays where click and drag operations were difficult to do and prone to error.

The customization window is designed to be a temporary shelf of functionality that can be used to support changing tasks and roles. The title bar of the customization window tells the end user that they can right click on an item to remove it from the customization window. This allows key bindings to be rapidly remapped to support changing tasks in a meeting environment. For example, I could right click on the items in Figure 9 and replace them with the photo control items (less brightness, more contrast, rotate) that I have begun to use and already appear in the History list.

The customization window is always available and does not require any special modes or dialog boxes to use. The history customization system can be used with any Microsoft Office application (e.g., Powerpoint, Word, Excel, Publisher). Like the History List, clicking on an item in the customization window calls corresponding function in the meeting application.

Toolbar items with single level menu selections (e.g., the rotate toolbar menu) activate the most recently used selection menu selection (e.g., Rotate 90°) when selected in the History menu. Hierarchical menu selections were not included in this version because I wanted a consistent interface (icons and textual descriptions for each item) and menu items often do not have a toolbar icon.

The customization window could be used to support the activities of a photo editor in a newsletter. For example, a photo editor might be interested in photo control functionality such as less brightness, more brightness, less contrast and more contrast. If a photo editor added these items to their customization window they would be able to rapidly recall this functionality using the customization window or using keyboard shortcuts. If the photo editor wanted to start the task of formatting picture captions he or she could right click on the customized photo control items and add formatting buttons such as bold, italics and underline. This example shows how the history customization system supports the changing tasks of a meeting collaborator.

Implementation

The first step to tracking all mouse movements, key presses and toolbar clicks is to use a Windows Hook. A Windows Hook allows us to bind to chain of events that occur during every input event so that we can track the position of the keyboard and mouse even when an application does not have focus.

Next, one needs to determine if input events occurred over tool bar item. This is done with a modified version of the Grouplab WidgetTap Library [8] that uses active accessibility (a programmer's utility for creating interfaces for people with sensory or motor impairments) obtain the location and size of each toolbar item.

Next, one needs to add items to the history list when a toolbar item is clicked. This is done by using the Grouplab Collabrary [9] to capture the toolbar icon. The gradient is removed to reduce visual clutter and a simplified icon is displayed on the history list. Toolbar icons are captured in real-time because toolbar items can be disabled. For example, capturing the icon of the bold button in Figure 9 would result in a greyed out image that would look be hard to see in the History view.

When an item is clicked on the history list the system uses the Grouplab WidgetTap Library [8] to send an event (through Active Acessibility) to the appropriate toolbar item in the meeting application.

This system implementation runs efficiently, that is, there is no noticeable difference in performance between using Microsoft Office and using the customized interface version of Office.

Discussion

The decision to sort items in the history list based on recency and moving duplicates to the top was motivated by work by Greenberg [2] who showed in studies that history interfaces that exploited recency and removed duplicates

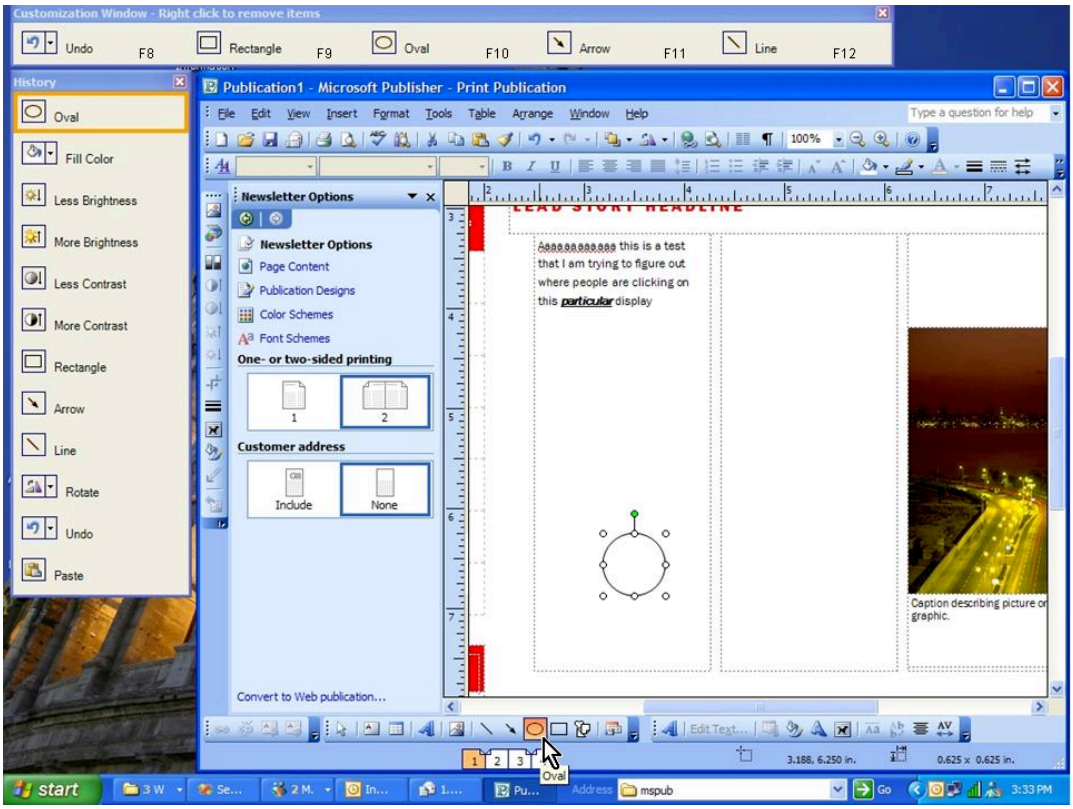

Figure 9. The customization window and history interface for Microsoft Publisher

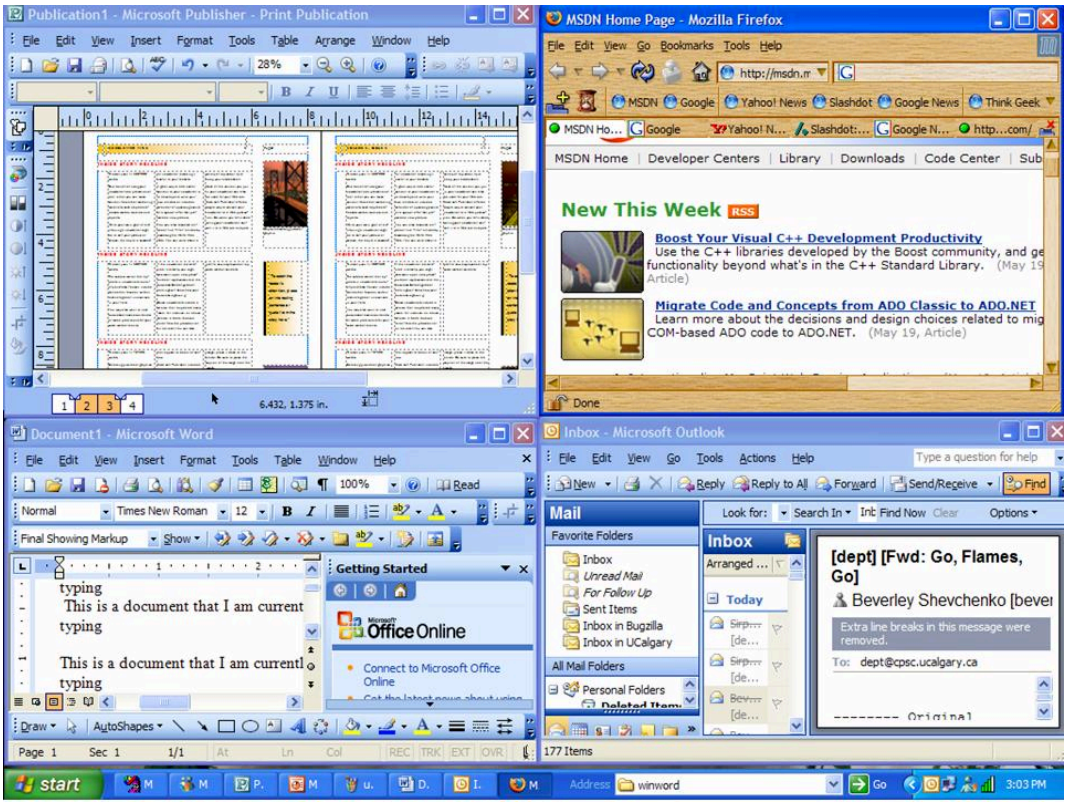

Figure 10. Multiple tiled applications used by a News Editor

were easy to understand for most users.

I chose to represent each history and customization item as a large button containing an icon and a caption since the large button made it easier to click on. The larger button provides accurate target acquisition even when moving around the screen quickly. It is important to ensure that selections can be made quickly and accurately in a meeting scenario.

Prior work on history systems suggest a frequency distribution that rapidly tails off. That is, after about 5-7 objects, the probability that the next one will be used is increasingly small. The twelve item limit in the History list was chosen from readings of previous literature and personal experience. Less than twelve items often meant that previous functions were too quickly removed from the list and more than twelve items increased the amount of searching that had to be done in order to find the appropriate item in the History List.

The purpose of the lightweight customized interface was to allow people to take advantage of the powers of customization with minimal time and cognition effort. The focus of the meeting should be on the agenda rather than the technology. The customized interfaces achieved its purpose by minimizing time effort through a history list and minimizing cognition effort by providing automated key bindings with a visual reference.

SUPPORTING ROLES IN MEETING ENVIRONMENTS

Lightweight customized views interfaces can be combined together to rapidly develop minimalist interfaces in a meeting environment. Often the tasks of a meeting collaborator will span across multiple applications. For example a news editor for a Newspaper might want to obtain information from an email client and a web browser, write articles in a word processor and paste completed articles in a news paper layout manager.

The news editor could switch between multiple applications using the task bar but this would add unnecessary steps to the simple task of transferring information from one application to another. An alternative would be to tile all of the crucial windows such as in Figure 10 but we see that the screen is quickly consumed by user interface elements and the content is almost impossible to see.

By combining customized views and customized interfaces we can create a interface overview that is similar to Henderson and Card's (1986) notion of multiple virtual workspaces. The major difference is that the tiled displays show the important content (not just rectangles) using lightweight customized views and allow interaction through commonly used functionality through lightweight customized interfaces (Figure 11). The top bar of each window represents the currently customized interface while the left bar represents a history of previous actions. This would reduce the total time spent switching between applications and allow the news editor to focus on the task of writing an article from information found in email and on the web

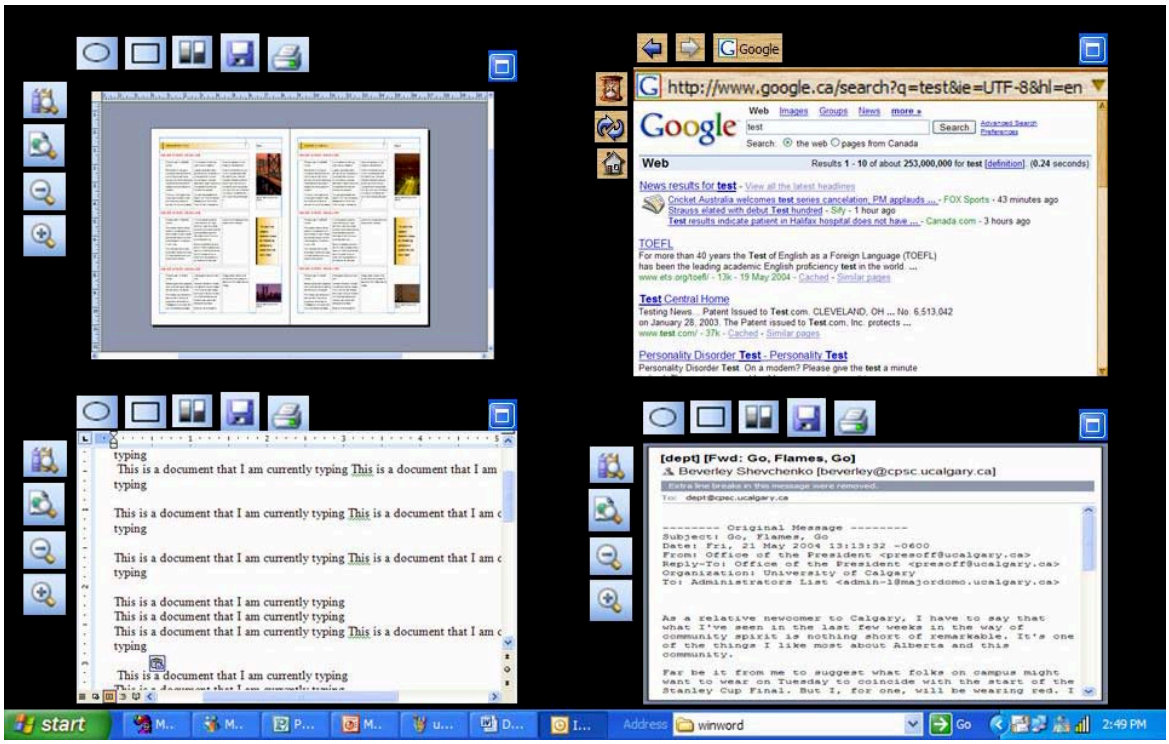

Figure 11. Combining customized views and customized interfaces for a News Editor

rather than focusing on the technology of switching between applications.

By clicking on the blue expand button one could see the appropriate application in full screen mode while minimizing the full screen application would return to the tiled customized interface and view.

By providing each meeting collaborator with customized views and customized interfaces we can support roles in customized interfaces. For example, Figure 12 shows four different views and interfaces for a simulated news room setting. A news editor (top left) might be interested in having an email client, web browser and word processor open while a photo editor (bottom left) might want a photo manipulation utility and an entertainment editor (bottom right) might want a photo browser. The current state of the newsprint could be projected onto a secondary projected display so that editor in chief (top right) could observe the current state of the newspaper and provide feedback to the editors.

A similar technique could be applied to a 3D animation studio. Specialists with lighting, character animation, modelling and special effects could collaborate together using customized interfaces and views. The final image could be projected onto a large display viewed by the director. The director could make suggestions and have the various specialists rapidly implement the suggestions so that an animation could be quickly iterated with feedback from the director.

CONCLUSION

The primary contribution of this paper is the concept of lightweight customization as a solution to using aspects of customization in meeting environments. The two core principles are minimal / no time effort and minimize cognitive effort. The principles of lightweight customization were applied to two settings:

Customized views automatically grab the content of a meeting application and project it on a secondary display. Customized views allow members of a brainstorming meeting to focus on the meeting agenda instead of the technology.

Customized interfaces provide a history of previously used functions so that commonly used sequences of functionality can be rapidly recalled. The customization window puts commonly used functionality at the fingertips of users (literally) while reducing the time and cognitive effort of customization.

Customized views and customized interfaces can be combined to create custom interfaces that support the roles and respective tasks in a meeting environment.

Customization is an extremely powerful tool that is not being used to its full potential in existing applications. This paper is an important first push towards going beyond making customization possible and trying to make it useable in time sensitive situations.

One day customization may become a powerful tool that is exploited by even novice users. The goal of this paper is to encourage the research and development community to consider lightweight customization as a technique for improving software in meeting environments.

ACKNOWLEDGMENTS

The authors would like to acknowledge Smart Technologies for their support during this Internship. Michael Boyle provided most of the tools needed to build the customized view and customized interface prototype.

REFERENCES

1. Cypher, A. *Watch What I Do: Programming by Demonstration*. MIT Press, Cambridge, MA, 1993.
2. Greenberg, S. *The Computer User as Toolsmith: The Use, Reuse, and Organization of Computer-based Tools*. Cambridge University Press, Cambridge, MA, 1993.
3. McGrenere, J. *The design and evaluation of multiple interfaces: A solution for complex software*. PhD Dissertation. Department of Computer Science, University of Toronto, Toronto, ON, 2002.
4. Kaasten, S. *Integrating Back, History and Bookmarks in Web Browsers*. MSc Thesis, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, 2001.
5. McKenzie, B., Cockburn, A. An Empirical Analysis of Web Page Revisitation, In *Proceedings of the 34th Hawaiian International Conference on System Sciences (HICSS34)*, IEEE Computer Society Press, Maui, Hawaii, 2001.
6. Tan, D.S., Meyers, B., Czerwinski, M. WinCuts: Manipulating arbitrary window regions for more effective use of screen space. In Extended Abstracts of *Proceedings of the ACM Human Factors in Computing Systems Conference (CHI)*, p. 1525-1528, 2004.
7. Fujima, J., Lunzer, A., Hornbæk, K., Tanaka, T., Clip, Connect, Clone: Combining Application Elements to Build Custom Interfaces for Information Access. In *Proceedings of the ACM User Interface Software and Technologies Conference (UIST)*, pp. 175-184, 2004.
8. Greenberg, S. and Boyle, M. Customizable physical interfaces for interacting with conventional applications. In *Proceedings of the ACM User Interface Software and Technologies Conference (UIST)*, pp. 31-40, 2002.
9. Boyle, M., *Privacy in Video Media Spaces*, PhD Thesis, Department of Computer Science, University of Calgary, Alberta, Canada, 2005
10. Mantei, M., Capturing the Capture Concepts: A Case Study in the Design of Computer Supported Meeting Environments, In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp. 257-270, 1988.
11. Henderson, A., Card, S., Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface. In *ACM Transactions on Graphics*, Vol 5., No 3., pp. 211-243, July 1986

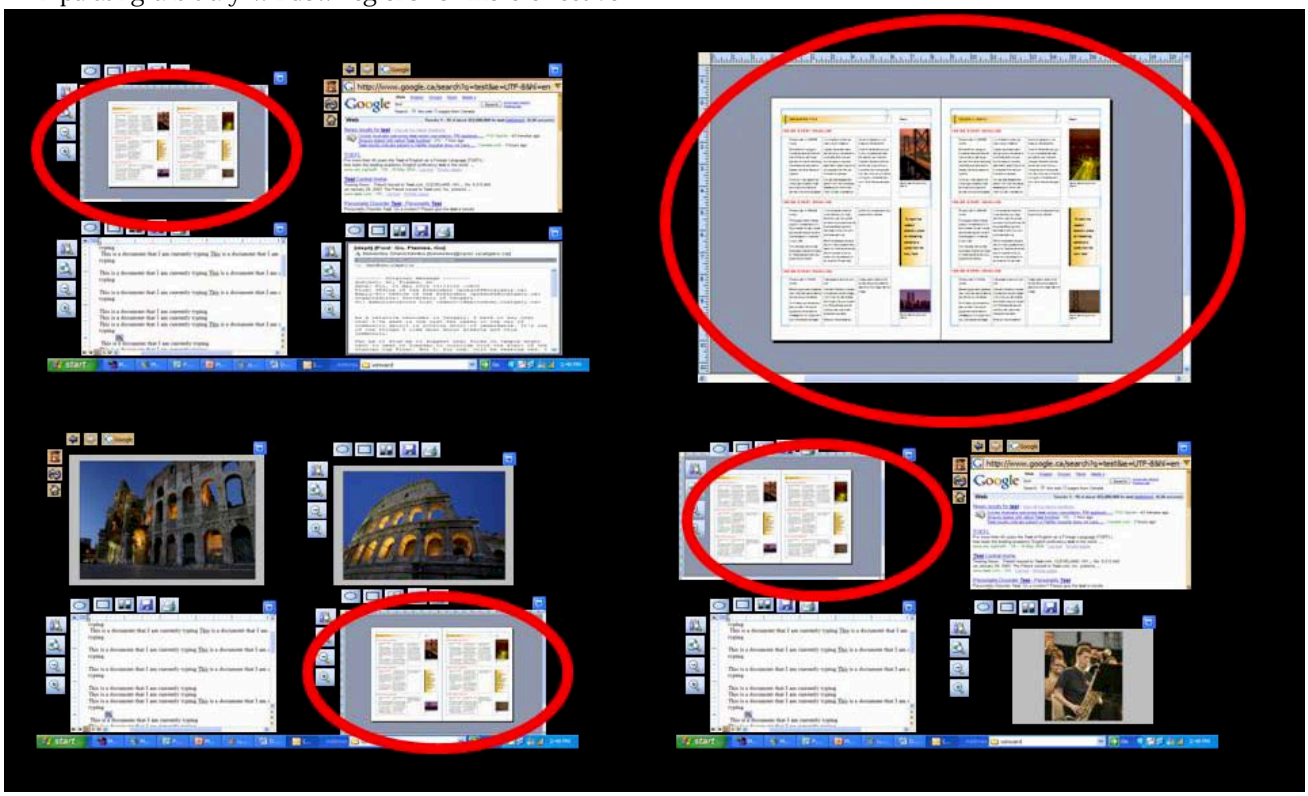

Figure 12. Customized Collaboration used in a News Room