CS-TR-3784 January 1997
ISR-TR-97-55

User Interfaces for a Complex Robotic Task: A Comparison of Tiled vs. Overlapped Windows

J. Corde Lane[1], Steven P. Kuester[1], and Ben Shneiderman[2]

Space Systems Laboratory[1]
Human-Cumputer Interaction Laboratory[2]
Department of Computer Science, Institute for Systems Research[2]
University of Maryland, College Park, MD 20742-3255
email: corde@ssl.umd.edu, ben@cs.umd.edu

Abstract
High complexity tasks, such as remote teleoperation of robotic vehicles, often require multiple windows. For these complex tasks, the windows necessary for task completion, may occupy more area than available on a single visual display unit (VDU). Since the focus of the robotic task constantly changes, modular control panels that can be opened, closed, and moved on the screen are invaluable to the operator. This study describes a specific robotic task and the need for a multi-window interface that can be easily manipulated. This paper examines two multi-window management strategies: tiled (fixed size) and arbitrary overlap. Multi-window searches were performed using the two management styles and they were compared on the basis of search completion time and error rates. Results with 35 novice users showed faster completion times for the tiled management strategy than for the arbitrary overlap strategy. Other factors such as the number of windows available, the number of displayed windows, workload of opening or closing windows, and effect of learning are discussed.

Keywords: tiled, overlapped, user interface design, window management

Date: January 1997

User Interfaces for a Complex Robotic Task: A Comparison of Tiled vs. Overlapped Windows

J. Corde Lane
corde@ssl.umd.edu
Space Systems Laboratory, University of Maryland, College Park, MD 20742

Steven P. Kuester
Space Systems Laboratory, University of Maryland, College Park, MD 20742

Ben Shneiderman
ben@cs.umd.edu
Human-Computer Interaction Laboratory, Dept. of Computer Science, University of Maryland, College Park, MD 20742

1.0 Introduction

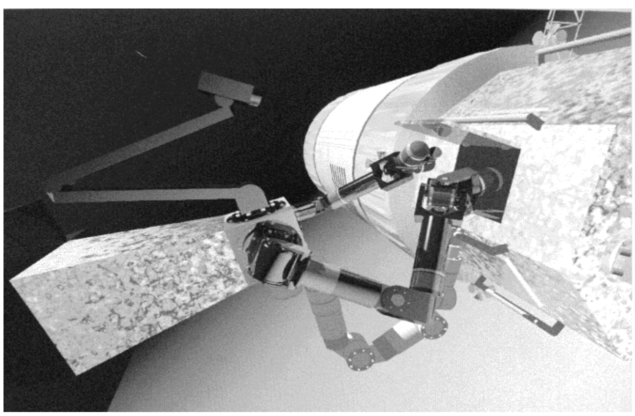


Figure 1: Simulation of the Ranger vehicle repairing a satellite

The Space Systems Laboratory conducts research in areas of human factors and space robotics. A current project is Ranger; a four-armed, free-flying robot developed to perform maintenance tasks on satellites. Creating an effective human-computer interface will require the capability for an operator to quickly reorganize the numerous control panels as the focus of the task changes.

1.1 RANGER PROJECT BACKGROUND

The Ranger vehicle has a propulsion module which allows it move freely and to rendezvous with a target satellite. This propulsion module houses all spacecraft bus subsystems (power, guidance navigation and control, communications, thermal control). The forward section holds the computer and electronics used to control the four robotic arms. Ranger has one grapple arm which attaches to the satellite and moves Ranger into position. Two dexterous arms perform the maintenance tasks. The

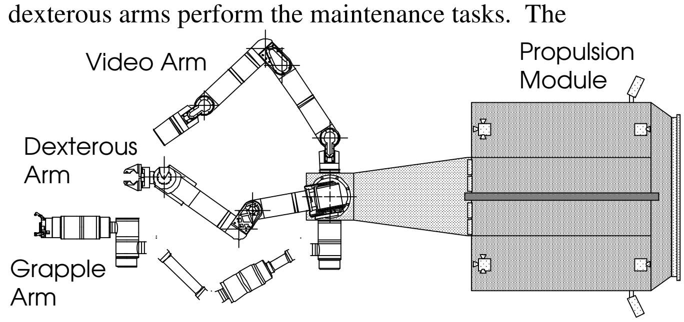


Figure 2: Ranger schematic illustrating the vehicle arms and propulsion module

video arm is controlled to provide an optimum view to the operator.

1.2 DESIGNING THE HUMAN-COMPUTER INTERFACE

The human-computer interface to control this vehicle requires multiple control panels that will need to be changed as the focus of the task changes. Controlling this robotic vehicle not only requires a computer interface to monitor changes in the propulsion module, similar to a traditional satellite, interactive and highly informative control panels are needed to be able to control each of the four arms. This vehicle will be operated remotely from a ground station at University of Maryland.

A popular paradigm for satellite control stations is to generate full size control panels representing key systems of the satellite. The operator can flip though the many screens quickly, and view the data telemetry from the satellite. This full screen control panels work well when the tasks for the operator are well defined. Viewing battery power levels, adjusting the

Figure 3: Robotic Control Station Interface in overlap and tiled mode

satellite orientation, and monitoring temperature gradients can be performed effectively with the appropriate control panel. Control of the systems of the propulsion module might be able to follow this paradigm, as only infrequent commands to the vehicle and constant monitoring is necessary. However, control of any robotic arm, without using preplaned trajectories, is a highly interactive relationship between the operator and computer interface. Many control panels are needed to control and monitor only one arm. Graphical displays of the arm's orientation, low level diagnostics for determine errors in control software, rudimentary control for testing individual parts within the arm, graphing data showing the history of power usage, and many more panels are opened to assist the operator in performing a robotic task.

A single operator has the capability of not only controlling and monitoring all spacecraft bus subsystems, but also controlling one to four of the robotic arms. Since the Ranger vehicle is designed to perform multiple generic maintenance tasks, a well developed procedure can not be developed. The vehicle and operator must constantly be able to adjust to the changing conditions while performing the task. This capability requires the computer human interface to change as the focus of the operator shifts. Instead of using full screen control panels, the Ranger control station will use small modular panels, shown in figure 3. Analogous to airplane cockpit gauges and sensor packages, these control panels are specialized and used together to build a customized cockpit for the operator. The ability to add, delete, and move around the appropriate panels is essential for successful completion of the robotic task. This study examines methods of window management which will make the operator effective in manipulating the many control panels required to control a complex robotic vehicle.

1.3 PREVIOUS RESEARCH

Several window management strategies exist that can be used for complex system interfaces. In the early eighties, Xerox developed one of the first tiled window management strategies for its STAR system. Shortly thereafter, Apple developed a much more flexible arbitrary overlap window management strategy in its Finder 1.0 using guidlines still followed today (Apple Computer, 1992). Since then, both strategies have continued to evolve. Since their conception, researchers have tried to determine which task domain is best suited for each of these two window management strategies. For example, Lifshitz and Shneiderman (Lifshitz and Shneiderman, 1991) found that for a search and retrieval task, subjects preferred a tiled interface that gave them control of tile placement as opposed to computer determined placement.

Some of the task domain that has been evaluated has included complex tasks (multiple windows and subtasks). The ROOMS system allowed users to organize many virtual workspaces (Henderson et al., 1993). Each workspace was a room where many windows were dedicated to completing a specific task. The user could quickly switch rooms to work on another task. The CUBRICON Intelligent Window Manager used a computer algorithm weighing factors to determine a window's importance and automatically change the screen layout (Funke et al., 1993). Kandogan and Shneiderman (1997) developed a tiled strategy which allowed the user to directly and incrementally reshape a target window while the computer would compensate resizing the remaining windows For complex tasks such as controlling a telerobot or nuclear reactor, it is important to determine which window management strategy is appropriate.

1.4 PROBLEM STATEMENT
This study examines two multi-window management strategies for complex systems: tiled and arbitrary overlap. The two management styles will be compared using criteria of faster performance and fewer errors.

2.0 Theory

This section describes the theory behind both the tiled and arbitrary overlap interfaces. We begin with a description of a user's cognitive model and how it relates to complex tasks, then present the particulars of each interface.

2.1 MEMORY MODELS
Although many memory models have been proposed (Sanders and McCormick, 1993), the concept of memory being separated into sensory storage, short term memory, and long term memory is one of the most common. In this model, the human perceives sensory input and retains it briefly in sensory storage. The human then determines whether the information is useful by means of the short term memory; also referred to as working memory. If the information is useful or encountered repetitively, it is stored in long term memory for later recall. In this model, it is the working memory that performs all the cognitive tasks that the human requires. Working memory is where humans formulate models of their environment based on sensory perception from sensory storage and past experiences from long term memory.

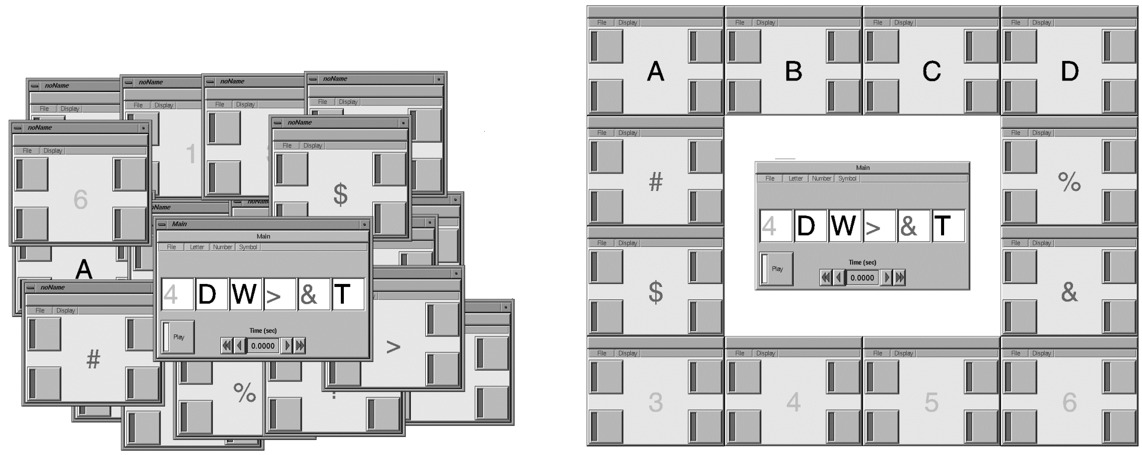
2.2 MEMORY MODELS AS APPLIED TO COMPLEX TASKS
How the human organizes their environment in working memory is critical to task performance. The Syntactic-Semantic model (Shneiderman, 1992) assumes users form a semantic model of the based on their perception of the elements presented on the display. The users' syntactic knowledge is used to interpret the set of symbols that make up this text into meaningful thoughts and ideas, while the semantic model is composed of the ideas that the reader generates. Different window management strategies, such as tiled or arbitrary overlap, can be considered different syntax for the user to apply their semantic model. In human factors, much effort is put into determining appropriate syntax for interfaces based on user's existing memory models. Unfortunately, the systems are often highly simplified and only require a single model of the system to be retained in working memory. More complex systems may require semantic models that are too large to fit in working memory. In this case, the user must continuously swap smaller subsets of a semantic model from long term memory to short term memory as the task set changes. The concept of a complex task can be defined to be a task set that maximizes a human's cognitive workload due to frequent change of semantic memory models in working memory. By choosing an appropriate window management strategy (syntax), user's already overloaded cognitive workload can be reduced.

2.3 OVERVIEW OF THE INTERFACES
Two multiple window management strategies were considered for this study: tiled and arbitrary overlap. The tiled interface includes a group of non-resizable windows that can occupy twelve discrete locations on the perimeter of the display. The arbitrary overlap interface allows users to organize windows in any location on the screen. Figure 4 shows a graphical representation of each interface.

The two management styles diverge in how windows are organized. The arbitrary overlap window strategy allows the user to open, close, and move windows throughout the entire screen (Shneiderman, 1992). Also, windows are free to overlap each other. This style is similar to the Macintosh Finder and Microsoft Windows 95. This gives the effect of "two-and-a-half dimensions" since windows can appear behind or in front of other windows. This management style is popular because it allows direct manipulation which gives users greater control over the system. However, many users fall into the cluttered desktop syndrome: many windows on the screen overlap each other, making it difficult to locate the appropriate window. Also, some windows are buried deep beneath others and cannot be found until certain windows are moved or closed. Unlike the arbitrary overlap interface, the tiled interface allows opening, closing, and movement of windows without overlap. If windows are hidden completely by other windows, they are closed to prevent "piles-of-tiles". This prevents the necessity for user's to remember where hidden tiled windows might be located.

3.0 Implementing the Interfaces

This study examined the performance speed and errors of 35 subjects using one of the two management styles. The test was run using Silicon Graphics Indigo^2 (SGI) workstations. The two interfaces were created using the SGI's graphics library, *Forms*, developed by Mark Overmars. Syntax for manipulating the main window and both tiled and arbitrary overlap interfaces are described below.



(a) **(b)**

Figure 4: Two multiple window management strategies: (a) arbitrary overlap and (b) tiled

3.1 MAIN AND SUBTASK WINDOWS

The interface consisted of one main window and many subtask windows. The main window was intended to have the primary focus of the subject. It dictated to the subjects what tasks were needed to be performed, showed current progress, scored users past performance, and was used in the creation of subtask windows. Subtask windows were smaller and contained four buttons in each of the four corners. An alphanumeric symbol in the center identified the window. By clicking on the four buttons of the appropriate subtask window, as indicated by the main window, subjects completed a component of the task set called a trial.

3.2 THE ARBITRARY OVERLAP INTERFACE

In the arbitrary overlap interface, the SGI's IRIX window management environment was used to manipulate the subtask windows. By clicking the right mouse button on the title bar of a window, a pop-down menu would show window management options. The functions included: moving the window, raising the window above all other overlapping windows, and lowering the window below all overlapped windows. There were no options to allow window resizing. These window management functions could also be performed in other ways. A window could be raised by clicking the left mouse button on the title bar or window border. If the mouse button is held, the window could be dragged to any location on the screen. Finally, both interfaces could close windows by choosing 'Close' from the pull down menu.

3.3 THE TILED INTERFACE

The tiled interface did not take advantage of the IRIX window management environment. A different title bar was used and it did not retain the window management border. To move a subtask window, the user would instead use the mouse to activate a pull down menu from the title bar and choose the 'Arrange..' option. This would open the 'Arrange' dialog box with thumbnail representations showing the locations that subtask windows could be moved to. The user would choose the button corresponding to the desired location, then hit the 'OK' button. The subtask window would then be moved to that location. If another subtask window already existed there, the two subtask windows would swap locations. A 'Cancel' button was also provided on the 'Arrange' dialog box to abort the move. Since no overlapping was allowed, raise or lower features were not needed.

3.4 CONTROL FOR BIAS

Control for biases were made in the placement of subtask windows. Because the tiled interface required the 'Arrange' dialog box, the same dialog box, without its functionality, was included in the arbitrary overlap interface. This gave equal number of mouse clicks in the placement of subtask windows for both window management strategies.

4.0 The Hypothesis

Our hypothesis predicts that, after learning, performance times will be faster and error rates will be lower for the tiled interface. The tiled interface simplifies users cognitive workload by relieving them of searching difficulties that may be associated with overlapping windows. We expect expert users to show faster completion times with lower error rates on the tiled interface for reasons similar to those mentioned above.

5.0 The Test

A test was developed which required subjects to perform several searches on a group of items that either were not on the screen (tiled) or were hidden by other overlapping windows (arbitrary overlap). This test was selected to determine how the two interface types affected memory retention capabilities of the subjects. The test was the same for all subjects, independent of the interface being manipulated. The only difference was the syntax required to complete the tasks on the two interfaces. The subjects were instructed to perform a series of subtasks as quickly, and with as few errors, as possible. Each subtask consisted of eliminating an alphanumeric from the main window by clicking all four buttons in the corresponding subtask window.

5.1 MAIN WINDOW
At the beginning of the test, the main window would appear on a blank screen (see figure 5).

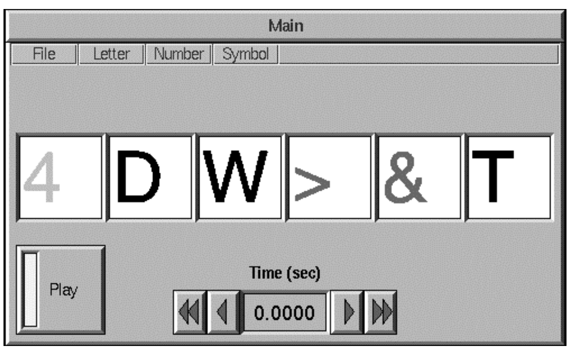


Figure 5: The Main Window

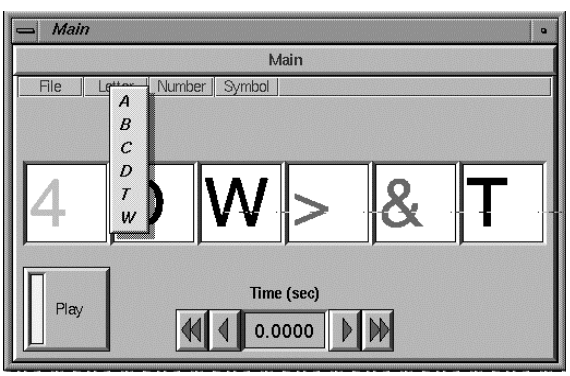


Figure 6: Menu Activation

The main window was fixed in the center of the screen when using the tiled interface. The subjects using the arbitrary overlap interface could place the main window wherever they desired. After the 'Play' button was pressed, the test would begin. Six alphanumerics would fill the white boxes in the main window. These alphanumerics represented which subtask windows had to be manipulated. Each subtask window corresponded to a single alphanumeric character. The windows were divided into three categories: letters (A,B,C,D,T,W), numbers (1,2,3,4,5,6), and symbols ($, #, %,&, ?, >). This gave a total of 18 subtask windows which could be manipulated. The tiled interface could only display twelve subtask windows at a time while the arbitrary overlap could only display about nine without overlap. The alphanumerics were chosen to prevent confusion. For example, the letter 'O' and the number '0' were not used to avoid conflict.

5.2 CREATING SUBTASK WINDOWS
At the beginning of the test, no subtask windows were open. Therefore, subjects had to open the specified subtask windows to be manipulated. To open a subtask window, subjects would move the mouse to the appropriate pull-down-menu (letters, numbers, or symbols) and choose the appropriate alphanumeric item. The 'Arrange' dialog box would pop up and could be placed anywhere on the screen (see figure 7).

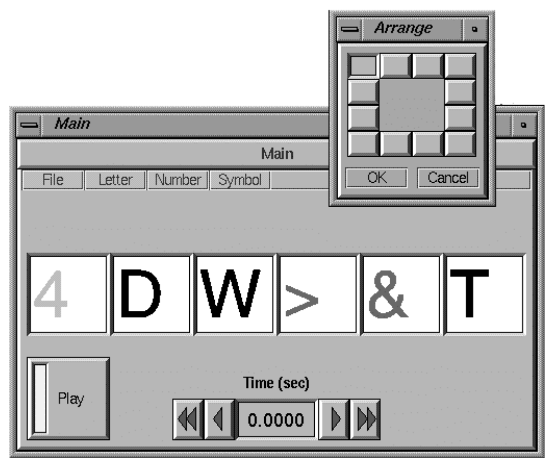


Figure 7: The 'Arrange' Dialog Box

For the arbitrary overlap interface, subjects pressed the 'OK' button in the 'Arrange' dialog box. The 'Arrange' dialog box disappeared and the chosen subtask window could be placed anywhere on the screen (see figure 8).

For the tiled interface, the 'Arrange' dialog box's thumbnail representations allowed subjects to place the subtask window at a discrete location on the screen. Then, after pressing 'OK' button, the 'Arrange' dialog box disappeared and the subtask window was opened in the specified location. In figure 8, the subtask window would appear in the upper left corner of the screen. If a subtask window already existed in that location, it would have been destroyed. No piles-of-tiles were allowed. In both interfaces, a subtask window that already existed on the screen could not be reopened.

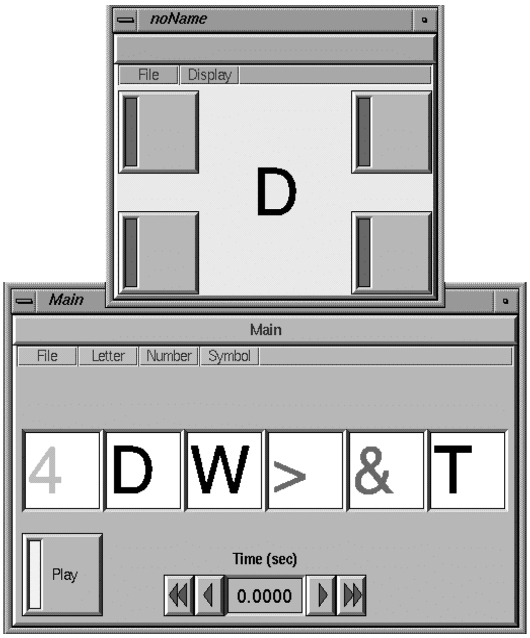


Figure 8: The 'Subtask Window'

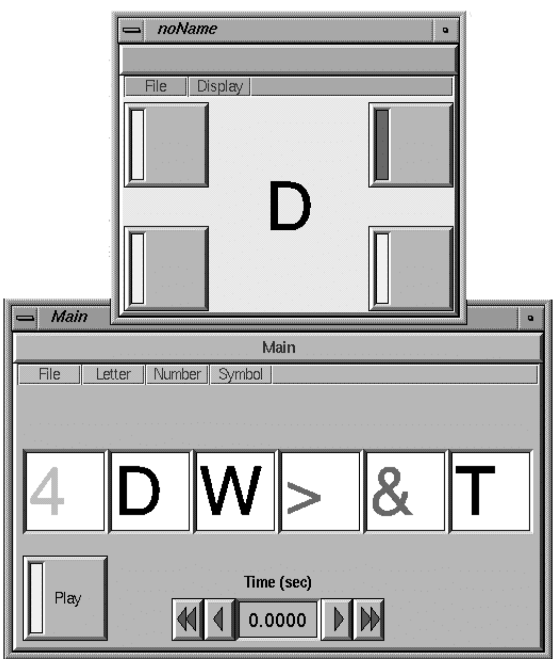


Figure 9: The 'D' Subtask Window Being Manipulated

5.3 MANIPULATING SUBTASK WINDOWS

The subtask window could then be manipulated to complete one of the six subtasks. To successfully manipulate a subtask window, each of the four buttons in the corners of the window were pressed. After each button was pressed, it would light up. Figure 9 shows the 'D' subtask window with three of the four buttons successfully manipulated. When all four buttons were pressed, in any order, the lights in the four buttons would turn off and the corresponding alphanumeric would disappear in the main window. After the last button is pressed in the 'D' subtask window the screen would appear as in Figure 10.

Subjects then continued by creating another appropriate subtask window and manipulating it. The six subtasks could be completed in any order. As each alphanumeric subtask window was correctly manipulated another field in the main window would disappear.

5.4 COMPLETING TRIALS AND TEST

When the sixth subtask was performed, that trial was completed. The next trial would begin and another

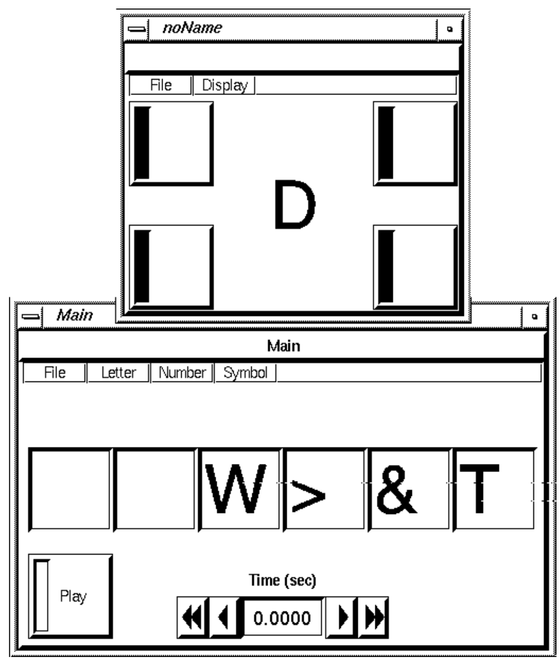


Figure 10: The 'D' Subtask Completed

six alphanumerics were displayed in the main window. The time indicator would show the amount of time, in seconds, it took for the last trial to be completed. Figure 11 shows the first trial near completion, only one more button in the '&' subtask window needs to be manipulated. Once that button was pressed, the display would look like figure 12. Six new alphanumerics were displayed as the next trial began. After the completion of a trial, any subtask windows currently on the screen became immediately available for manipulation. It was not necessary for them to be closed and reopened for use on subsequent trials. Other subtasks may have required the creation of new subtask windows. As the trials continued, the subject had to work within the specific window manager. Many activities the subjects performed included: creating the necessary subtask windows, possibly eliminating subtask windows not needed, moving subtask windows to related groups, and scanning for the correct subtask window. After ten trials were completed, the system indicated the termination of the test. Many variables calculated by the system, during the test, were then sent to a data file. These variables included trial completion times, error rates, number of subtask windows opened during each trial, and the number of subtask windows displayed at the beginning of each trial.

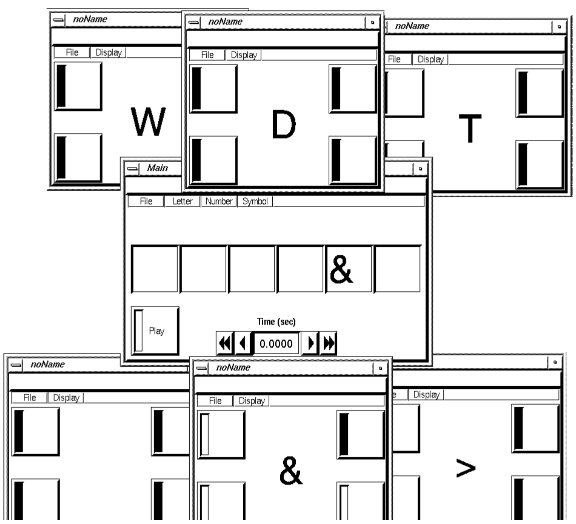


Figure 11: The Last Subtask to be Completed for Arbitrary Overlap Interface

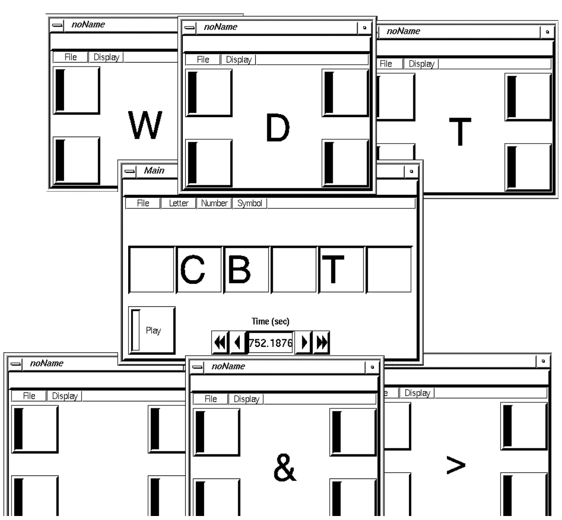


Figure 12: Trial One Complete, Trial Two Initiated for Arbitrary Overlap Interface

6.0 Subjects

The demographics survey showed that ninety-one percent of the subjects used some kind of computer windows environment often, while the other nine percent had still occasionally used a windows environment. To acquaint the subjects with the test each subject viewed a four minute video which briefly explained the experiment and the two management interfaces. Next, the subjects filled out a demographics survey. This was used to determine biases in the experiment. Subjects then went to a Silicon Graphics Indigo2 workstation and were informed which interface they would use. The test coordinator then went through a checklist of items that subjects would perform using the interface. This was a step-by-step tutorial explaining what needed to be done to perform the tasks. Throughout the tutorial, subjects were able to ask questions. After completing the orientation , subjects filled out a consent form. The test was then initiated. Upon completion, subjects filled out a questionnaire rating many of the system's attributes.

7.0 Results

Thirty-five subjects completed the experiment. Eighteen performed the test using the arbitrary overlap interface. The other seventeen used the tiled interface. In addition, two subjects were tested as experts. As the subjects went through each trial of six subtasks, several variables were monitored. At the completion of the test, a data file was created which contained the following information about each of the ten trials: completion time, number of errors, number of windows opened, and the number of window on the screen at the beginning of the trial.

7.1 COMPLETION TIMES

Subjects were instructed to move through the test as fast as possible. The test had a pseudo-random sampling of the eighteen alphanumeric subtasks. Equal amounts of letters, numbers, and symbols appeared as subtasks. However, the seed for generating the random subtasks remained the same for all subjects. Therefore, all subjects experienced the same sequence of subtasks (i.e. 4DW>&T, then 3CB2T1, then >4#6$C, etc.). This level of consistency shows in the data. Figure 13 shows the average completion times of all subjects for each trial.

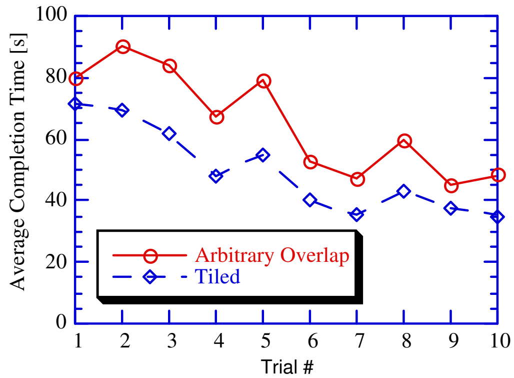


Figure 13: Average Completion Time of All 35 Subjects

Trial	Arbitrary Overlap Average	Tiled Average	Arbitrary Overlap Std. Dev.	Tiled Std. Dev.
1	80.1	71.6	15.5	16.3
2 *	89.9	69.3	21.4	13.6
3 *	84.3	61.7	23.3	23.2
4 *	67.2	48.1	27.1	13.8
5 *	79.2	54.6	28.0	12.5
6	53.1	40.2	17.1	12.4
7	47.0	35.4	17.5	11.9
8	59.5	43.2	19.6	14.9
9	45.1	37.1	17.4	9.8
10	48.6	34.5	25.6	11.0

*** statistically significant at the 0.05 level**
Table 1: Completion Time in Seconds and Standard Deviations in a Group for Each Subtask for 35 subjects

Both the arbitrary overlap and tiled interfaces follow similar trends. General learning can be seen as the completion times decrease until they stabilize during the final trials. Both interfaces spike on the fifth and eighth trials. The difficulty for those trials was higher for all subjects. Table 1 shows the averages and standard deviations for each interface. All data has units of seconds.

7.2 EXPERT STUDY

Two expert subjects were evaluated on both interfaces. These subjects had over one hour of experience on both interfaces. For each interface, the two subjects' completion times were averaged for each trial. Figure 14 shows the two averages.

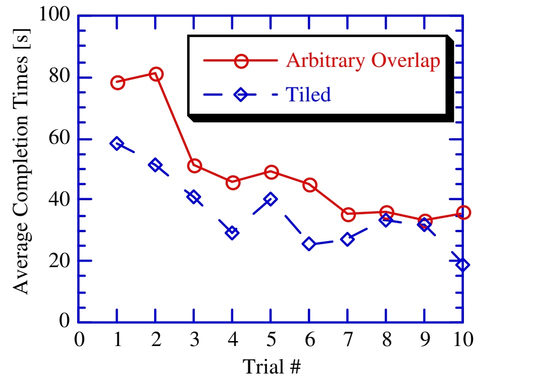


Figure 14: Average Completion Times For 2 Experts For Each Trial

7.3 NUMBER OF WINDOWS OPENED

Figure 15 shows the average number of subtask windows that were opened for each trial. The general shape of the curve is similar to the average completion time. This shows a possible relationship between the number of windows opened and the trial completion time. After the initial trials, the subjects using the tiled interface were forced to open more subtask windows than subjects using the arbitrary overlap interface.

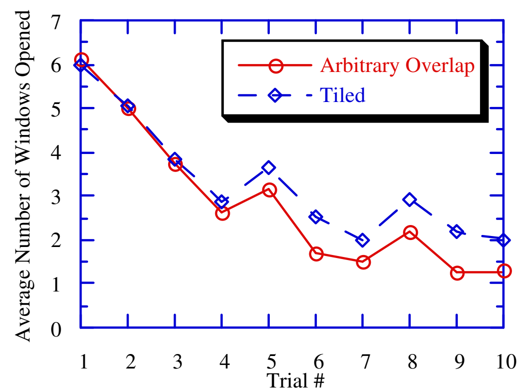


Figure 15: Average Number of Subtask Windows Opened for Each Trial for Both Interfaces

Figure 16, shows the average trial completion time as a function of the amount of windows opened for that trial. Unfortunately, the results are not statistically valid at the 0.05 level. This may be due to biases in the experiment and the observable trends merit further study.

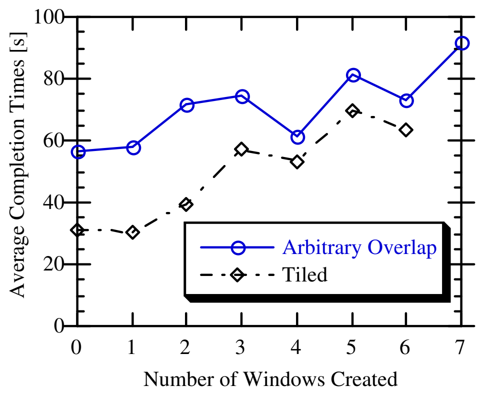


Figure 16: Average Trial Completion Time Dependent On The Amount Of Windows Opened That Trial

7.4 NUMBER OF WINDOWS INITIALLY DISPLAYED

Figure 17 shows the average number of windows displayed on the screen at the beginning of each trial. The curves for each interface appear to assymptotically approach the maximum number of diplayed windows: twelve for the tiled and eighteen for the arbitrary overlap.

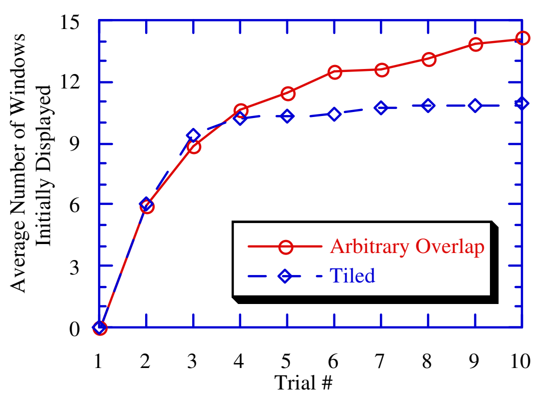


Figure 17: Average Number of Subtask Windows Initially Displayed at Each Trial for Both Interfaces

8.0 Discussion

This study presents supporting evidence that a tiled interface may provide faster navigation times than an arbitrary overlap interface for certain circumstances. A task that challenged subjects' memory retention capabilities was used. This simulated complex task was designed to be similar to the workload associated with controlling a mobile telerobotic vehicle.

8.1 COMPLETION TIME

In figure 13, for all ten trials the overall average completion time was shorter for the tiled interface than for the arbitrary overlap interface. A t-test was used to confirm statistical significance at the 0.05 level (Devore, 1991). This quicker completion time may be due to the tiled interface's lack of overlap. The subjects spent less time reorganizing the screen and were able to complete the trials faster. By not allowing to overlap, subjects could quickly scan the screen to see if a window existed. With the arbitrary overlap interface, many subjects lost a window underneath the cluttered desktop. One subject spent minutes moving and searching windows to find the last window necessary to complete a trial. The benefits of overlapping multiple windows in any subjective fashion became a trap for many subjects. With the arbitrary overlap interface it was difficult for the subject to optimize all of the screen space and to place windows in an orderly fashion. Since the tiled interface constrains the subject to place windows in discrete location, it creates an organized environment which produces faster and easier work. Also as the trials progressed for both interfaces, completion times were reduced. This suggests learning.

Because the completion times decreased greatly between the first and last trials, an overall average may not be the best indicator of performance. A t-test was performed for each trial revealing statistical significant differences for trials 2-5. It appears that the tiled interface is useful for novices but, as experience level increases, subjects develop schemes independent of the interface type that assists them in completing the complex tasks. This shows that, given a learning period, humans are highly adaptive to complex tasks.

8.2 EXPERIENCED USER RESULTS

With only two expert subjects tested, the average completion time for all trials was lower for the tiled interface. The creation of subtask windows seemed to be a factor in their subtask completion times. During the first two trials the expert subjects would "set up" the windows in strategic groupings. Sometimes the experts opened additional windows to complete the groups. This would then speed up the completion times for successive trials. Humans use these schemes to reduce the large cognitive workload created by the complex task. This suggests that humans will find other ways of adapting to complex tasks outside of taking advantage of an interface's syntax. This would explain the long initial completion times followed by a steep drop. The tiled interface forced organization of the windows and therefore caused substantially lower completion times for the beginning trials. The expert's experience can also be seen when comparing their results (figure 14) with the 35 novice subjects (figure 13). The expert average completion time for all trials was 18.9 seconds and 14.8 seconds shorter than novice times for the arbitrary overlap and tiled interfaces respectively. More expert subjects may substantiate these results.

8.3 DEPENDENCY OF COMPLETION TIMES ON OPENING WINDOWS

The general shape of the completion time curves (figure 13) suggest a dependency on the number of windows opened for each trial (figure 15). Further analysis of the data led to no statistical significance. In figure 16, a general increase in completion time can be noted as the number of windows opened grows. This increase is more pronounced with the tiled interface. However due to the large standard deviations, no statistical significance was found. With a greater number of subjects, a statistical relationship may be derived. As figure 15 confirms, more windows are opened by the users of the tiled interface than the arbitrary overlap. This is because the tiled interface limits the total displayed windows to twelve. However, even with the increased workload of creating additional windows, the tiled interface completion times were lower. The benefits of the organizational structure of the tiled interface outweigh its disadvantages of creating additional windows.

8.4 GUIDELINES FOR FUTURE RESEARCH

More extensive testing may find a statistical difference in completion times for expert subjects. Since only two subjects were tested, only broad based comparisons could be made. Also the effects of learning could be calculated to see how a novice develops into an expert. An investigation could define the connection between the creation of windows and completion time. By making the test more difficult, the amount of errors a subject commits on an interface would also prove useful. Further analysis could find out to what extent the "cluttered desktop" slows the completion time. In this study, the difference in error rates between the two interfaces was not statistically significant. However, the task used in this study was simple to understand and perform. Further study, looking into more difficult tasks may bring out greater difference in error rates.

9.0 Concluding Remarks

Results show that for novice users, a tiled interface produces faster completion times than an arbitrary overlap interface in performing complex multiple window management tasks. As users become more experienced, completion times are reduced while performance differences between tiled and arbitrary overlap less defined. This suggests that humans are highly adaptive and selection of appropriate window management strategies is only important for novice users. Future research should focus on further analysis of expert users. Further research also should pursue whether such independent variables as number of windows opened, percentage of windows hidden, and frequency of subtasks affect user's completion times and error rates.

10.0 References

Apple Computer, Inc. (1992). *Macintosh Human Interface Guidelines.* Addison Wesley, Reading, MA.

Bly, Sara A., and Rosenberg, Jarrett K. (1986). A Comparison of Tiled and Overlapping Windows. *Proceddings of ACM CHI'86 Conference on Human Factors in Computing Systems*, ACM, New York, 101-106.

Devore, Jay L. (1991). *Probability and Statistics for Engineerring and the Science*. Brooks/Cole Publishing Company, CA.

Funke, Douglas J., Neal, Jeannette G., and Paul, Rajendra D. (1993). An Approach to Intelligent Automated Window Management. *International Journal of Man-Machine Studies 38*, 949-983.

Henderson, D. Austin, Jr. and Card, Stuart K. (1993). ROOMS: The Ise of Muliple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface. *NASA Technical Report 93N70287*.

Kandogan, Eser and Shneiderman, Ben (1997). Elastic Windows: Evaluation of Multi-Window Operations *Proceddings of ACM CHI'97 Conference on Human Factors in Computing Systems,* ACM, New York.

Lifshitz, J. and Shneiderman, B. (1991). Multi-Window Browsing Strategies for Hypertext Traversal. *Proc. Thirtieth Annual Technical Symposium of the Washington, DC Chapter of the ACM,* 121-131.

Norman, Kent L., Weldon, Linda J., and Shneiderman, Ben (1986). Cognitive Layouts of Windows and Multiple Screens For User Interfaces. *International Journal of Man-Machine Studies 25*, 229-248.

Plaisant, Catherine, Carr, David, and Shneiderman, Ben (1995). Image browsers Taxonomy and Design Guidleines. *IEEE Software 12*, 21-32.

Sanders, Mark S. and McCormick, Ernest J. (1993). *Human Factors In Engineering And Design*. McGraw-Hill, Inc., New York .

Shneiderman, Ben (1992). *Designing the User Interface Second Edition.* Addison Wesley, Reading, MA.

Woods, David D. (1984). Visual Momentum: A Concept to Improve the Cognitive Coupling of Person and Computer. *International Journal Man-Machine Studies*, 21, 229-244.

Woods, David D. (1989). The Cognitive Engineering of Problem Representation, *Human-Computer Interaction on Complex Systems*, Academic Press: London.