Exploring and Evaluating Task Sequences for System Control Interfaces in Immersive Virtual Environments

Ryan P. McMahan

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
In
Computer Science

Dr. Doug A. Bowman, Chair
Dr. Chris North, Committee Member
Dr. Manuel A. Perez-Quinones, Committee Member

June 4, 2007
Blacksburg, VA

Keywords: Immersive virtual environments, 3D user interfaces, system control, system control interfaces, task sequences

Exploring and Evaluating Task Sequences for System Control Interfaces in Immersive Virtual Environments

Ryan P. McMahan

Abstract

System control – the issuing of commands – is a critical, but largely unexplored task in 3D user interfaces (3DUIs) for immersive virtual environments (IVEs). System control techniques are normally encompassed by complex interfaces that define how these interaction techniques fit together, which we call system control interfaces (SCIs). Creating a testbed to evaluate these SCIs would be beneficial to researchers and would lead to guidelines for choosing a SCI for particular application scenarios. Unfortunately, a major problem in creating such a testbed is the lack of a standard task sequence – the order of operations in a system control task.

In this research, we identify various task sequences, such as the Action-Object and Object-Action task sequences, and evaluate the effects that these sequences have on usability, in hopes of establishing a standard task sequence. Two studies were used to estimate the cognitive effort induced by task sequences and, hence, the effects that these sequences have on user performance. We found that sequences similar to the Object-Action task sequence induce less cognitive time than sequences similar to the Action-Object task sequence. A longitudinal study was then used to analyze user preferences for task sequences as novices became experienced users with using an interior design application. We found that novices and experienced users alike prefer sequences like the Object-Action over sequences like the Action-Object task sequence.

Acknowledgements

Thanks to my family for always supporting my endeavors and decisions.

To Dr. Bowman for his guidance, inspiration, support, and, most importantly, his friendship.

To Dr. North and Dr. Perez for their ideas and influences on my research.

To the Virginia Tech 3DI group for providing their support and ideas. They are like a family.

To Ron Kriz and Patrick Shinpaugh for all their help with the VT CAVE.

To all the participants. This research would not have been successful without them.

To all my friends for their support … and distractions.

To Christine for being there at the end and making sure I finished.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Introduction to System Control

In the past few years, there has been much research on the topic of 3D user interfaces (3DUIs) for immersive virtual environments (IVEs). Most of this research has pertained to fundamental user interaction tasks such as selection, manipulation, travel, and wayfinding, but one task that has not been as heavily researched is *system control* [Bowman *et al.* 2005]. A system control task is a user task in which a command is issued to perform a particular function, change the mode of interaction, or change the system state, often through commands or menus [Bowman *et al.* 2005].

While working on a document, a user may want to save the current version before continuing with editing. In a traditional desktop graphical user interface (GUI), this system control task – issuing a command to perform a save function – is usually accomplished by accessing the "Save" command from the "File" menu, clicking on a button resembling a floppy disk, or using a combination of the "Ctrl" and "S" keys. Working with a painting application, another user may want to change from a brush tool to an eraser tool. In this case, the system control task is to change the mode of interaction, which is typically achieved through the use of a tool palette and by clicking on a button resembling an eraser. The third case of system control tasks – changing the system state – can be seen in the example of a standard web browser and the user adjusting the size of the text. This task is normally accomplished by accessing the "Text Size" submenu from the "View" menu or by using a combination of the "Ctrl" key and the "+" or "-" keys.

Although VE users perform most of their work through interaction tasks like selection, manipulation, and travel, system control is critical because it is the "glue" that allows the user to control the interaction flow between the other key tasks in an application [Bowman *et al.* 2005]. Even in IVEs, system control tasks are used very often. Consider an architect using a design and analysis application similar to Virtual-SAP [Bowman *et al.* 2003] to expand his latest design with a new wing and analyze the weaknesses of the new structure.

The architect begins by loading in his latest design using a file selection dropdown menu and clicking a "Load" button on a 3D window. The architect uses ray-casting to select the portion of the design to expand and then changes the mode of interaction from selection to expansion by clicking an "Expand" button on the 3D window. In expansion mode, the architect indicates the direction and extent for expanding the new wing. Once he finishes the expansion, the architect wants to analyze the structure for any weaknesses. He focuses back on the 3D window and clicks on a "Simulation" button to have the system simulate an earthquake and show what would happen to the new structure. The new design survives the simulation, and the architect clicks a "Save" button to save the changes to the structure.

In this example of the architect using a design and analysis application, there are several instances of system control tasks. The architect used system control to change the state of which file to load, to perform the load function, to change the mode of interaction to expansion, to perform an earthquake simulation, and to perform a save function. Many of these types of small interactions can be found in IVE applications, and the abundant necessity for system control has led to the development of many system control techniques.

A *system control technique* is a specific method for executing a system control task. Most of these techniques draw upon a small number of basic metaphors or their combination, such as adapted 2D menus, 3D widgets, speech recognition, gestures, and virtual tools [Bowman *et al.* 2005]. An example system control technique is using a stylus to interact with a dropdown menu on a tablet. The pen-and-tablet user interface of the original Virtual-SAP application allowed users to interact with traditional 2D menus and widgets, such as dropdown menus, within a 3D environment [Bowman *et al.* 2003]. Another system control technique is the use of collocated 3D widgets, in which the functionality of a menu is moved onto an object in the 3D environment, and geometry and functionality are strongly coupled [Bowman *et al.* 2005]. For instance, in the Home Design application, a user can select and change the position of a handle on a door via ray-casting in order to resize the dimensions of that door [Mackie *et al.* 2004], thus combining mode selection and manipulation into a single step. Sometimes system control techniques are voice commands, such as in [Zhao and Madhavan 2006]. In this system, the voice commands

"faster" and "slower" are used to respectively double or half the speed of the current interaction task, changing the state of the system.

These low level interaction techniques are normally encompassed by more complex interfaces that define how these system control techniques fit together. These encompassing, complex interfaces are known as *system control interfaces* (SCIs). The ring menu [Liang and Green 1994, Shaw and Green 1994], the TULIP menu [Bowman and Wingrave 2001], and the command and control cube (C^3) [Grosjean *et al.* 2002] are all examples of SCIs. All of these interfaces define how the user transitions between system control techniques and interaction tasks. For instance, the C^3 is a 3 x 3 x 3 cubic grid, where each of the 27 grid cubes is a menu item (Figure 1.1). For the example task of changing the mode of interaction to a pencil tool, the user activates the menu by making a pinch on a Pinch Glove™ worn on the non-dominant hand, and the C^3 appears centered at the user's hand. The user then moves her hand in the direction of the pencil menu item cube relative to the center position, and then releases the pinch to change the mode of interaction to the pencil tool.

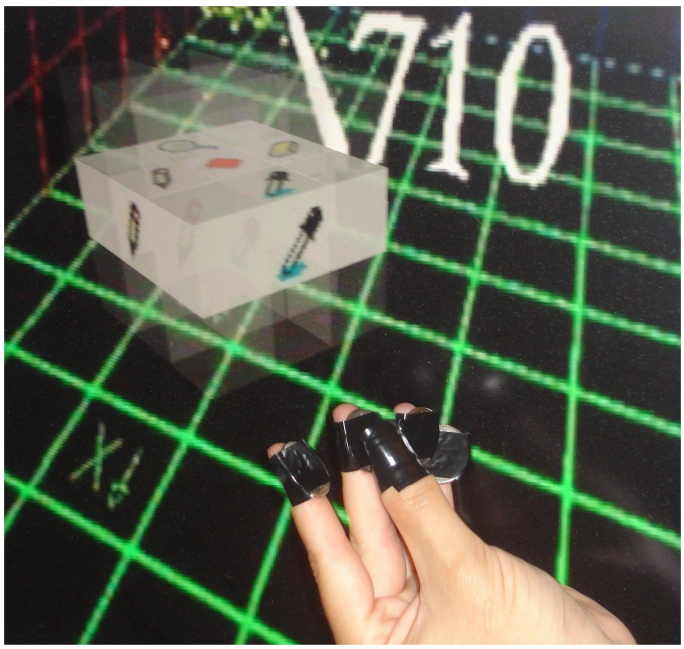


Figure 1.1 The command and control cube is an example of a system control interface. (The C^3 pictured here is not the original developed by Grosjean *et al*.)

System control interfaces are important for controlling the flow of interaction in an application and hence can make or break an application. In [Bowman *et al.* 2005], several design guidelines for 3D system control are presented:

- Avoid disturbing the flow of action of an interaction task.
- Prevent unnecessary changes of the focus of attention.
- Avoid mode errors.
- Use an appropriate spatial reference frame.
- Structure the functions in an application.
- Consider using multimodal input.

These guidelines suggest that SCIs for IVEs should be *transparent* – these interfaces should induce minimal cognitive effort so that users can focus on *what* to do instead of *how* to do it. A transparent SCI would maximize user performance and productivity by minimizing the cognitive effort required to interact with the interface itself and keeping the user focused on his current task. Despite how important these interfaces are, there has unfortunately been a relative lack of empirical evaluations of SCIs [Bowman *et al.* 2005].

1.2 Creating a Testbed for System Control Interfaces

Techniques for other 3D interaction tasks, such as selection, manipulation, and travel, have been empirically evaluated through testbed evaluations. Testbeds are environments and tasks that involve all of the important aspects of a task, test each component of a technique, consider outside influences on performance, and have multiple performance measures [Bowman *et al.* 2001]. Testbed evaluations allow multiple interaction techniques for the same type of task to be evaluated equally under several sets of circumstances.

There are several benefits of testbed evaluations. One benefit is reusability [Bowman *et al.* 2001]. If a new technique for a given interaction task is developed, that technique may be run through the testbed for that task and compared against previously tested techniques. This saves time later in comparing new interaction techniques to previously established techniques. Another benefit of testbed evaluations is the complex performance data generated from multiple

independent and dependent variables [Bowman *et al.* 2001]. This data allows discoveries to be made of interesting interactions between variables that would not have emerged from standard usability evaluations.

Because the benefits of testbed evaluation are so great and there has been a relative lack of empirical evaluations of system control, it would be reasonable and useful to build a testbed for SCIs. Such a testbed would provide an empirical method for evaluating and comparing current SCIs in detail. This would lead to guidelines for choosing a SCI based on a particular application scenario. Additionally, as future SCIs are developed, reusability would allow a new SCI to be compared to the previously established SCIs by running formal experiments solely for the new SCI. Unfortunately there are currently major problems in creating a testbed for SCIs.

One problem is that comparing two SCIs is like comparing apples and oranges. SCIs have different styles of interaction which makes their comparison difficult. For instance, some interfaces are based on graphical menus, such as the Virtual-SAP application [Bowman *et al.* 2003], while others can be based on gestural input for issuing commands, such as Polyshop [Mapes and Moshell 1995]. Comparing visual interfaces, such as menus, to non-visual interfaces, such as gestures, would be extremely complex, and it would be difficult to eliminate confounds for testbed evaluation. For example, the system control task of confirming an action could be accomplished by selecting an "OK" button on an adapted 2D menu or by using a "thumbs-up" posture with the hand. The performance times required to complete these two different confirmations can obviously be compared, but there are hidden confounds in using these two different techniques in a SCI. One obvious confound is feedback. In the example of the "OK" button, the system is obviously awaiting confirmation for an action, but in the example of the "thumbs-up" posture, unless other feedback is provided, the user may be unaware that confirmation is required.

Another major problem with creating a SCI testbed is that usually a SCI does not dictate how it is should be used. For example, an adapted 2D menu interface could be used to issue commands in at least two ways. Assume the user wants to remove an object from the environment. The user could select a "Remove" button from the menu and then select the object he wants to remove.

Alternatively, the user could select the object to remove first and then select the "Remove" button from the menu. Because a SCI doesn't dictate how it should be used, two developers may create two different flows of interaction for an application despite using the same interface. This causes problems for creating a testbed evaluation for SCIs, as the same SCI could be evaluated in more than one way.

Since creating a testbed evaluation for SCIs would be extremely beneficial to 3DUI designers, these major problems need to be addressed. For this research, we address the second major problem mentioned here, which is directly related to *task sequences*, in hopes of advancing towards a testbed for SCIs. By determining how SCIs should dictate their usage, we can create guidelines for using SCIs and provide standards for a testbed evaluation.

1.3 Task Sequences

A *task sequence* is the order of operations in a system control task [McMahan and Bowman 2007]. For example, if performing a particular function requires the indication of an object within the IVE and the indication of the function itself, then there are two obvious task sequences for performing the function. One task sequence would be the indication of the function followed by the indication of the object. An example of this would be a user selecting a "Remove" button from an adapted 2D menu and then selecting a virtual tree to remove from the environment (Figure 1.2A). We refer to this task sequence as an Action-Object task sequence because the user indicates the action before the object. The second obvious task sequence would be the indication of the object followed by the indication of the function. In the example, the user selects the virtual tree and then selects the "Remove" button from the menu (Figure 1.2B). This is referred to as an Object-Action task sequence because the user indicates the object before the action.

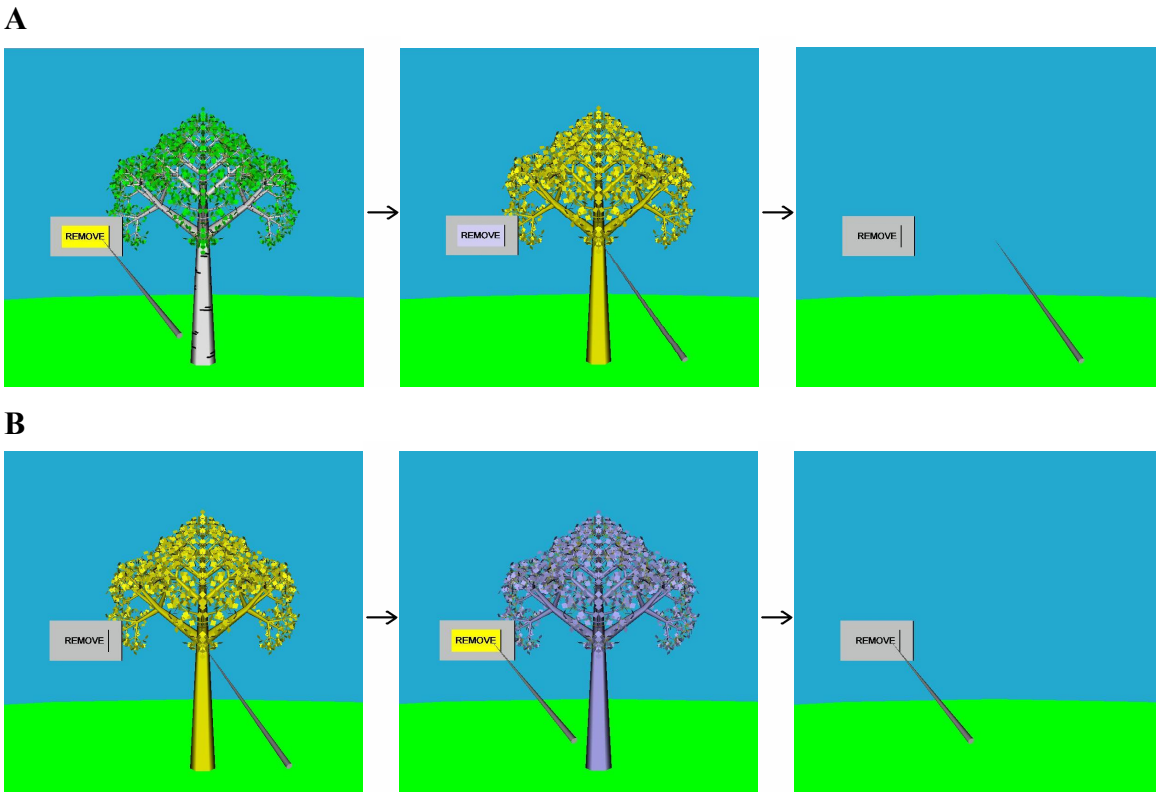


Figure 1.2 Examples of **A**) the Action-Object task sequence and **B**) the Object-Action task sequence.

As mentioned in Section 1.2, a SCI doesn't dictate how it should be used. Specifically, a SCI can be used in as many or more ways to complete a task as there are task sequences for that task. If task sequences for 3D tasks were standardized, or if their effects were understood, one confound for comparing SCIs could be eliminated, which would bring us one step closer to creating a testbed for such interfaces. In conventional environments, such task sequence standards have already been adopted.

In most command line interfaces (CLIs), a user types a command first followed by a list of parameters. We correlate this standard for CLIs with the Action-Object task sequence since the user first indicates the action (the command) and then objects (the parameters). In most GUIs, a user clicks on a target icon first and then accesses menus to execute commands on the target. We correlate this standard for GUIs with the Object-Action task sequence since the user first indicates the object (the icon) and then the action (the menu command). Both of these

conventional environments use different standard task sequences, which leads us to question if and which task sequences should be standardized for IVEs. If a single task sequence is optimal for usability, then any SCI would benefit from using that sequence and establishing a standard task sequence would be quite obvious. On the other hand, if different task sequences provide high levels of usability in different situations, understanding the combined effects of task sequence and task would still be beneficial for IVE designers.

This research focuses on evaluating task sequences in regard to usability, with the goal of potentially standardizing task sequences for SCIs. By establishing a standard for task sequences, we could eliminate one confound for comparing these interfaces and would be closer to establishing a testbed for SCIs.

1.4 Research Questions

In the course of our research we have developed several research questions regarding the design of system control interfaces and the use of task sequences. We directed our research to answer the following questions.

1. How do task sequences affect user performance in immersive virtual environments?

One part of determining the usability of an interface is evaluating the effect that that interface has on the user's performance. Since underlying task sequences can affect the interaction flow of an interface, do different task sequences change the effect that an interface has on user performance? Do different sequences of indications affect how the user thinks about the task and in turn affect the performance of the user? Do some task sequences induce less cognitive effort than others?

2. Which task sequences do users prefer in immersive virtual environments?

Another consideration in determining the usability of an interface is assessing user preference. Do most users prefer one task sequence over others and why? Are some task sequences easier to

understand for users than others? Do novices prefer the same task sequences as experienced users or do task sequence preferences evolve with expertise?

3. Is there a single task sequence that should be utilized by system control interfaces for immersive virtual environments and be accepted as the standard task sequence?

As mentioned in Section 1.3, CLIs typically utilize the Action-Object task sequence, and GUIs typically utilize the Object-Action task sequence. As of now, there is not a standard task sequence utilized by SCIs for IVEs. Should SCIs for IVEs utilize the Action-Object task sequence or the Object-Action task sequence? What should determine which task sequence to accept as the standard for SCIs?

1.5 Hypotheses

We formed the following hypotheses about system control interfaces and task sequences based on the similarities of immersive virtual environments to graphical user interfaces and our experiences with virtual environment applications.

1. The Object-Action task sequence will induce less cognitive effort than the Action-Object task sequence and will improve user performance.

With the Object-Action task sequence, users begin tasks by indicating an object, a concrete representation. With the Action-Object task sequence, users begin by indicating an action, an abstract concept. We hypothesize that task sequences beginning with the indication of a concrete representation will induce less cognitive effort for users than those beginning with the indication of an abstract concept, hence the Object-Action task sequence will induce less cognitive effort than the Action-Object task sequence. Following from this, we also hypothesize that the Object-Action task sequence will result in higher levels of user performance.

2. Novices and experienced users will both prefer the Object-Action task sequence for working in an immersive virtual environment.

In comparison to CLIs and GUIs, IVEs are more similar to GUIs. In CLIs, strings are used to represent everything, including commands, objects, and parameters. In GUIs, objects are visually represented by icons and commands are contained in graphical menus. In IVEs, objects have their own visual representations, which make IVEs more similar to GUIs than CLIs. Additionally, most people commonly use GUIs over CLIs.

Due to the similarities and familiarities that GUIs and IVEs share, we hypothesize that novices will prefer the Object-Action task sequence for working in an IVE. Based on our own experiences with IVEs, we hypothesize that experienced users will also prefer the Object-Action task sequence for working in an IVE.

3. The Object-Action task sequence should be the standard task sequence utilized by system control interfaces for immersive virtual environments.

Because both CLIs and GUIs have standard task sequences, we hypothesize that there is a single task sequence that should be the standard for SCIs. Since we hypothesized that the Object-Action task sequence will induce less cognitive effort than the Action-Object task sequence and that the Object-Action task sequence will be preferred by novices and experienced users over the Action-Object task sequence, we hypothesize that the standard task sequence for SCIs should be the Object-Action task sequence.

1.6 Approach

We used the following approach to answer our research questions and test our hypotheses.

1. Develop a method to evaluate the effects of task sequence on user performance.

One problem in evaluating the effects of task sequence on user performance is that absolute task performance is dependent on the SCI used to evaluate the task sequence. It is possible that a SCI could bias the absolute task performance of one task sequence over another due to the design and implementation of the SCI.

Consider two different 2D adapted menu SCIs. The first SCI uses a window statically located in front of the user at all times. The second SCI uses a window that dynamically moves to the last location the user pointed to within the IVE. The first SCI is biased for the Action-Object task sequence as the user will always start by focusing ahead. Additionally, for the Object-Action task sequence, this first SCI forces the user to return focus to the same location, no matter where the object is. The second SCI is biased for the Object-Action task sequence as the user is able to quickly access the window near the object. For the Action-Object task sequence, this second SCI forces the user to search in the location of the last object before beginning a new task.

Since the SCI used to evaluate task sequences could be biased, an alternative method to absolute task performance comparison for evaluating the effects of task sequence on user performance is necessary. One alternative is to compare the cognitive effort induced by each task sequence. This eliminates any effects that a specific implementation of a SCI may have on the evaluation of a task sequence.

Unfortunately it is difficult to measure cognitive effort. Because we don't know exactly what the user is thinking, it is hard to determine how much of the user's cognitive effort is used to think about the task sequence and how much is used to think about interacting with the SCI. To alleviate this problem, we will develop a model similar to the Keystroke-Level Model (KLM) [Card *et al.* 1980] and use a cognitive effort operator similar to the mental operator. We can estimate the cognitive effort operator by subtracting all other operators from the total time to complete a task. The other operators include the physical (perceptual and motor) time required for the user to make all of the indications required by the task and the cognitive time required for the user to interact with the SCI.

Since we need to know the physical and cognitive time required to interact with the SCI for a task, we need a single system control interface for evaluating the effects of task sequence on user performance. We chose to use the Virtual Environment Windowing Library (VEWL) [Larimer and Bowman 2003]. VEWL is an adapted 2D menu SCI (Figure 1.3) developed on top of DIVERSE [Kelso *et al.* 2002]. Different from most adapted 2D menu interfaces, VEWL utilizes a virtual sphere and maintains all windows tangent to this sphere. This ensures that all windows are equal distance to the user and provides a virtual surface for the users to move the windows along.

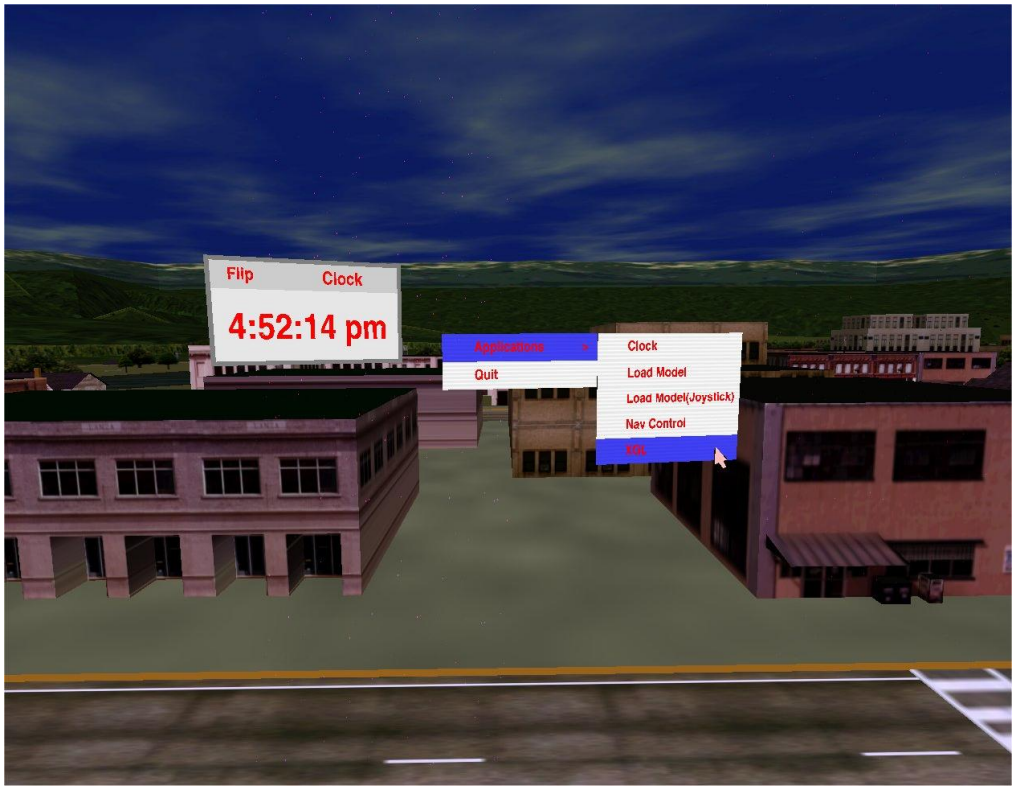


Figure 1.3 The Virtual Environment Windowing Library (VEWL).

We ran two experiments to establish the data necessary to complete our new model and evaluate the effects of task sequence on user performance. The first experiment was used to estimate the time required to complete certain system control tasks with the chosen VEWL SCI based solely on atomic-level interactions such as clicking a command button and invoking a popup menu. This helped establish the expected physical and cognitive times required to interact with the SCI for a task.

The second experiment was used to estimate the total time required to complete a task in a chosen IVE and all other operators, such as selecting an object within the virtual environment. This information was then used with the data established in the first experiment to estimate the cognitive effort operator by subtracting all other operators from the total time required to complete a task. Hence, we were able to evaluate the effects of task sequence on user performance by comparing cognitive effort operators.

2. Develop an approach to capture user preferences for task sequences as their expertise increases.

Since we are concerned with the user preferences of novices and experienced users for task sequences, we ran a longitudinal study to capture the preferences of users as their expertise with an IVE application increased. We designed an application with a single SCI that supports multiple task sequences so that users would have open preferences and would not be influenced on which task sequence to use.

3. Create guidelines for using task sequences with system control interfaces for immersive virtual environments and establish a standard task sequence.

Using the information gathered in approaches #1 and #2, we formed guidelines on which task sequences should be used for designing, evaluating, and comparing SCIs. These guidelines will contribute to the development of a testbed for SCIs.

1.7 Overview

In this thesis, we examine questions surrounding task sequences and SCIs, focusing on user performance and preference. In Chapter 2, we cover previous research on 3DUIs and work related to task sequences. In Chapter 3, we identify three different levels of task sequences and two different foci for task sequences. In Chapter 4, we explain different approaches to evaluating task sequences in full detail. In Chapter 5, we describe the first experiment, which was used to gather atomic-level performance data on using VEWL. In Chapter 6, we describe the second

experiment, which was used to gather the remaining performance data necessary to estimate the cognitive effort operator of various task sequences. In Chapter 7, we describe the final experiment, which was a longitudinal study used to gather user preference data as users progressed from novices to experienced users for using an IVE application. Finally, in Chapter 8, we discuss the implications of these studies and present some guidelines for using task sequences with SCIs for IVEs.

Chapter 2 Related Work

2.1 System Control and 3D User Interfaces

In 2D user interfaces, system control tasks are supported by the use of specific interaction styles, such as the WIMP (Windows, Icons, Menus, and Pointers) metaphor [Preece *et al.* 2002]. Many of these same interaction styles have also been adapted to 3DUIs, as seen with adapted 2D menus, but because these interaction styles may not be effective in all situations, we have already seen the development of non-conventional system control techniques [Bowman *et al.* 2005]. In this section, we describe both conventional and non-conventional system control techniques, discuss some possible classifications for these techniques, and provide some basic design guidelines for SCIs.

2.1.1 Conventional Techniques

In 2D user interfaces, interaction elements such as dropdown menus, pop-up menus, tool palettes, radio buttons, and checkboxes are everywhere and most users are very familiar with these interaction techniques. These same conventional techniques have also been adapted to 3DUIs in the form of adapted 2D menus and 3D widgets.

Adapted 2D menus are simple 3D adaptations of 2D menus and basically function in the same way as they do in desktop GUIs [Bowman *et al.* 2005]. Dropdown menus, popup menus, floating menus, and toolbars are some of the types of adapted 2D menus that have been developed. Most of these types of adapted 2D menus, including popup menus (Figure 2.1), are encompassed in VEWL, the SCI described for our approach in Section 1.6.

Adapted 2D menus can be placed in virtual environments by several different means including *world-fixed*, *view-fixed*, *object-fixed*, and *hand-held* [Lindeman *et al.* 1999]. A world-fixed menu has an absolute, fixed position in the VE and will move in and out of the user's view as the user approaches and leaves that position, respectively. A view-fixed menu has a fixed position relative to the user's viewpoint and moves along within the VE as the user does. An object-fixed

menu has a fixed position relative to an object within the VE and moves as the object moves. A hand-held menu is a special type of object-fixed menu that is fixed relative to an object held in the user's non-dominant hand for bimanual interaction techniques.

Figure 2.1 The VEWL popup menu is an example of a view-fixed, adapted 2D menu.

The 3D widget is another adaptation of conventional system control techniques. Many 2D user interfaces utilize widgets for manipulations such as resizing windows, rotating selections, and adjusting parameters. The 3D widget takes advantage of the extra degree-of-freedom (DOF) available in a 3D environment to enable more complex interactions and better visual affordances. There are two kinds of 3D widgets: context-sensitive (collocated) and non-context-sensitive [Bowman *et al.* 2005]. Context-sensitive widgets are typically used for changing geometric parameters because the functionality of the widget is moved onto an object in the 3D environment, strongly coupling functionality and geometry. Figure 2.2 shows an example of a context-sensitive widget used in the Home Design application [Mackie *et al.* 2004] for resizing a door. Non-context-sensitive widgets are normally used for standard menu item selections, such as the command and control cube (C^3) [Grosjean *et al.* 2002], mentioned in Section 1.1.

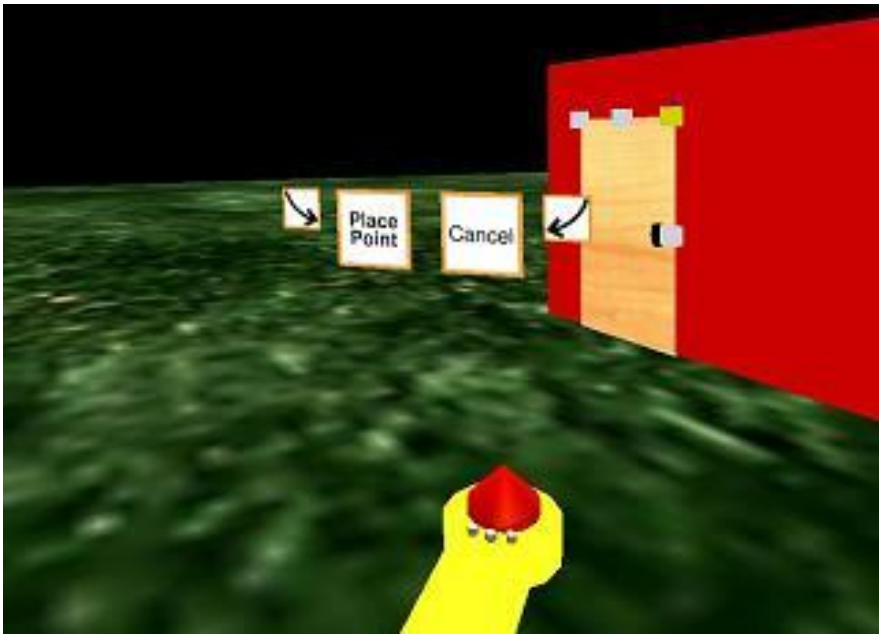


Figure 2.2 A context-sensitive, 3D widget is used for resizing a door in the Home Design application.

2.1.2 Non-conventional Techniques

In IVEs, users have to deal with input and output devices that are considerably different from the desktop, such as 6-DOF input devices instead of the standard keyboard and mouse. These differences create both new problems and new possibilities for the development of system control techniques for 3DUIs. Hence, non-conventional techniques have been discussed as appropriate replacements for system control [Bullinger *et al.* 1997].

The development of a non-conventional system control technique is normally linked to human factors, the input devices used, and the system- and application-level factors involved [Bowman *et al.* 2005]. Because of these factors, we have seen the development of several different types of non-conventional techniques, including ring menus, TULIP menus, gestural commands, and tools.

A ring menu [Liang and Green 1994, Shaw and Green 1994] is normally attached to the user's hand with menu items arranged in a circular pattern around it. With this non-conventional

technique, the user rotates his hand until the desired item falls within a "selection basket" and then the user makes the selection [Bowman *et al.* 2005]. The performance of this type of ring menu depends on the physical movement of the hand and wrist. Hence, system control designers have sought alternatives for rotating such as using a button on an input device to rotate the desired item into position. Figure 2.3 shows an example of a double-basket ring menu with two green selection baskets.

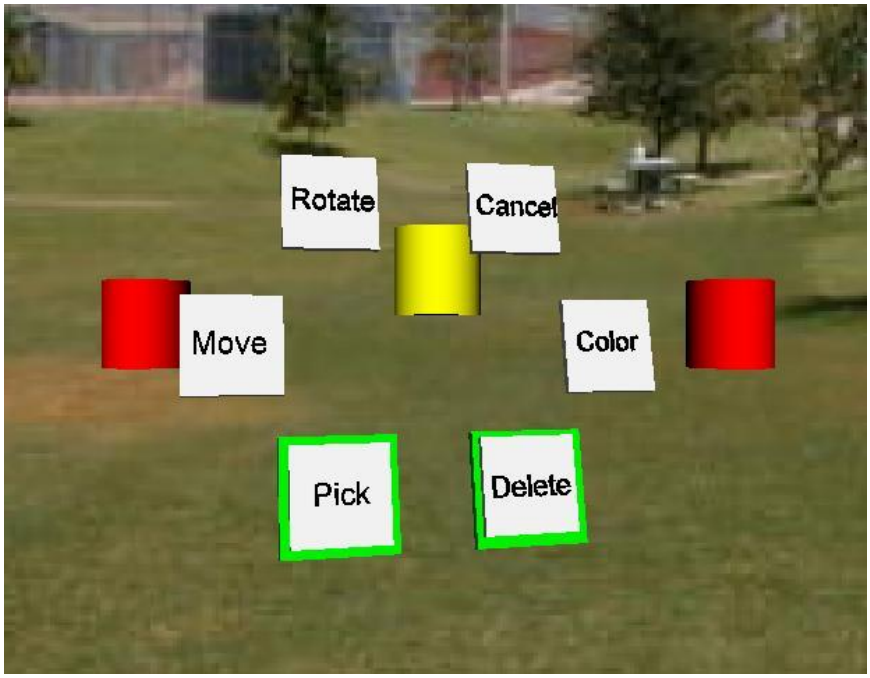


Figure 2.3 A double-basket ring menu is used as a non-conventional technique.

TULIP (Three-Up, Labels In Palm) menus [Bowman and Wingrave 2001] are menus attached to a user's hand by assigning menu items to different fingers using Pinch Gloves^{TM} (Figure 2.4). With this non-conventional technique, the user creates a pinch between a finger and the thumb of the same hand to select the corresponding menu item. When more than four items are in a menu, the first three items are displayed, attached to the user's index, middle, and ring fingers. The pinky finger is labeled "More," and selecting this menu item moves the next three items in the menu onto the user's fingers and the inactive items are displayed, in groups of three, on the palm of the virtual hand.

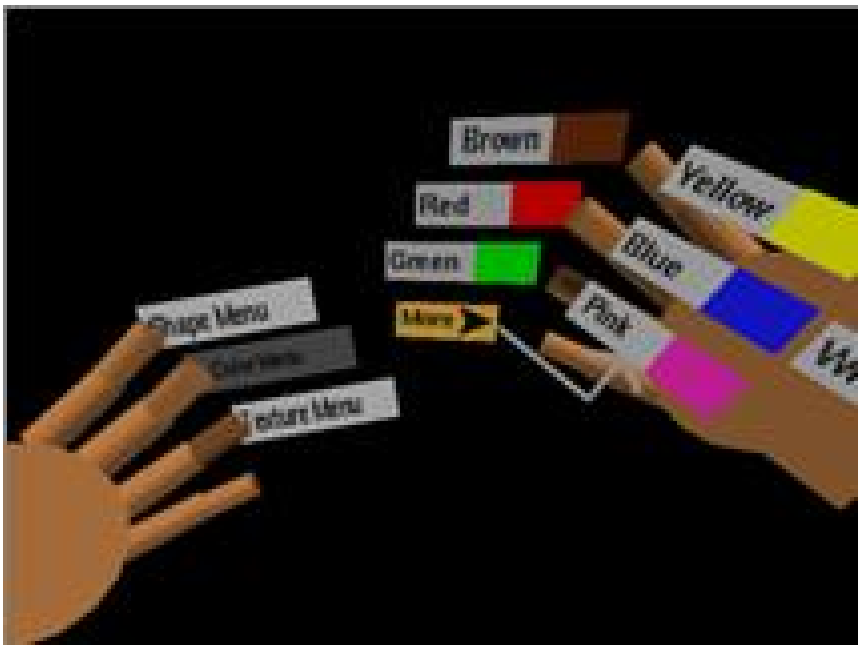


Figure 2.4 TULIP menus use finger pinches to select menu items.

Gestural commands are another type of non-conventional system control techniques and were one of the first system control techniques used for IVEs. Gestural commands can be classified as either postures or gestures [Bowman *et al.* 2005]. A posture is a static configuration of the hand, whereas a gesture is a dynamic movement. The usability of gestural commands for system control depends on the number and complexity of the gestural commands as more gestures imply more learning for the user [Bowman *et al.* 2005].

Another type of non-conventional system control techniques are tools, both physical and virtual. The use of familiar (real-world) devices for 3D interaction provides directness of interaction because of the real-world correspondence and familiarity provided by the tool itself [Bowman *et al.* 2005]. Physical tools are a collection of real physical objects with corresponding virtual representations. A user can access a physical tool by simply picking it up and using it. Virtual tools have no physical instantiations but behave just as physical tools do.

2.1.3 Classifying Techniques

With such a spectrum of conventional and non-conventional system control techniques being developed, 3DUI researchers have attempted to classify system control techniques [Bowman *et al.* 2005, Lindeman *et al.* 1999]. We find the classification presented in [Bowman *et al.* 2005], see Figure 2.5, to be the most useful for understanding the similarities of 3D system control techniques.

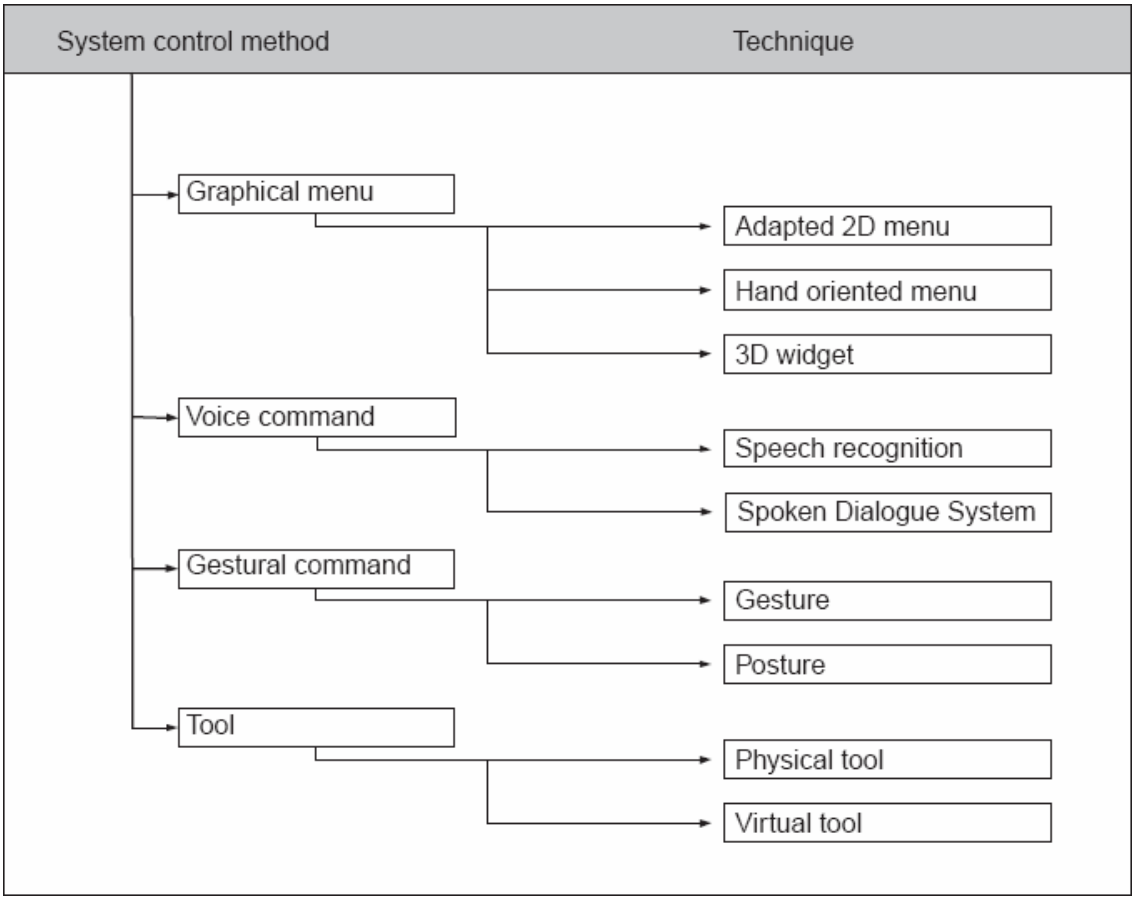


Figure 2.5 One classification of system control techniques based on metaphors.

This classification is organized around four main metaphors – graphical menus, voice commands, gestural commands, and tools – influenced by the description of non-conventional control techniques in [McMillan *et al.* 1997]. For graphical menus, visual feedback is the key connection for system control techniques as users can see what choices there are to choose from. For voice and gestural commands, direct input and memory load are key aspects. Users are

normally required to remember all of the commands and are not presented with choices although access to a command is more direct than that of a hierarchical graphical menu. Tools are an interesting combination of the other metaphors. Tools provide visual feedback in their representations, act as direct input as they are used on other objects, and require users to remember how each tool functions.

2.1.4 Design Guidelines

As mentioned in Section 1.1, several design guidelines for 3D system control techniques have been presented in [Bowman *et al.* 2005]. These guidelines are overall guidelines and in no way should these be considered the only guidelines for designing 3D system control techniques.

Avoid disturbing the flow of action of an interaction task. System control is often integrated with another 3D interaction task. Because of this integration, system control techniques should be designed to avoiding disturbing the flow of action of an interaction task.

Prevent unnecessary changes of the focus of attention. One of the major interruptions to a flow of action is a change of the focus of attention. This may occur when users have to cognitively or physically switch between the actual working area and a system control technique, or even when they must look away to switch devices.

Avoid mode errors. It is important that the user knows which interaction mode is currently active. Providing clear feedback remedies these types of errors.

Use an appropriate spatial reference frame. Placing a system control technique in an optimal position can make a big difference in its usability. If a technique is not visible at all, placed far away from the actual work area, or not oriented toward the user, user performance will be negatively affected. Also, system control techniques shouldn't occlude the user's view of the actual work area.

Structure the functions in an application. Using hierarchical menu structures and context-sensitive system control are good techniques for structuring the functionality of an application. In some cases, it may make sense to place some of the system control functionality on another device, such as a PDA.

Consider using multimodal input. System control techniques that combine multiple input streams can provide more fluid and efficient system control.

Unfortunately, these guidelines do not directly address the issue of task sequence – the order of operations in a system control task. Additionally, there has been no evidence of empirical evaluations of task sequences in SCIs for IVEs.

For this research, we want to establish guidelines involving task sequences for SCIs, in an attempt to standardize task sequences for a SCI testbed. In order to better understand task sequences for SCIs, it is instructive to consider how task sequences relate to other human-computer interfaces.

2.2 Task Sequences in Other Human-Computer Interfaces

When computers first emerged around the time of World War II, humans first started interacting with them using punch card machines and teletypes [Stephenson 1999]. For decades these already-existing technologies had been used for translating letters into bits and vice versa so it was quite natural for these devices to be grafted to computers as the first human-computer interfaces. These interfaces were initially used to batch process series of commands for programming purposes, but command languages and command line interfaces (CLIs) soon emerged.

2.2.1 Command Line Interfaces (CLIs)

When terminals and CLIs (Figure 2.6) emerged, the use of command languages allowed for interactive computing, leaving the batch processing era behind. A command language is defined

by the range of commands provided and their syntax [Newman and Sproull 1973]. The term – *command language* – infers that the underlying interface style is focused on commands or actions. For instance, using UNIX, a user would type in the command prompt "rm foo.txt" to remove a file named "foo.txt" from the current directory. This syntax shows that most CLIs force users to specify their actions first, followed by any parameters necessary to complete the actions. Since commands (or actions) are specified first and then parameters (or objects) in order to complete a task, we classify the task sequences underlying CLIs as Action-Object task sequences.

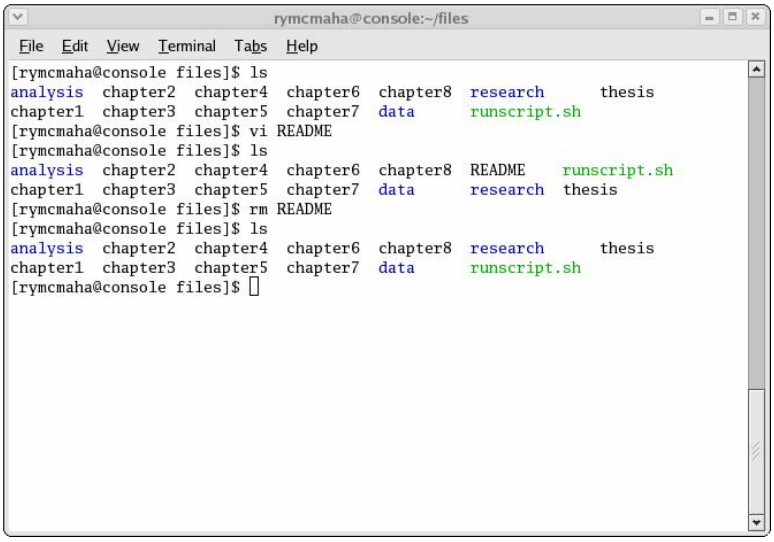


Figure 2.6 An example of a terminal for a CLI.

2.2.2 Graphical User Interfaces (GUIs)

The first major advance toward GUIs was Sutherland's *Sketchpad*, a graphical design program [Sutherland 1963]. Sutherland's goal was to create a program that would make it possible for a person and a computer to "converse rapidly through the medium of line drawings." Sutherland was one of the first to discuss the power of GUIs, the conception of a display as "sheets of paper", the use of pointing devices, the virtues of constraint representations, and the importance of depicting abstractions graphically [Hutchins *et al.* 1985].

As GUIs emerged, the idea of direct manipulation also emerged. The term *direct manipulation* was originally coined by Shneiderman and refers to systems having the following properties [Shneiderman 1998]:

1. *Continuous representation of the objects and actions of interest with meaningful visual metaphors.*
2. *Physical actions or presses of labeled buttons, instead of complex syntax.*
3. *Rapid incremental reversible operations whose effect on the object of interest is visible immediately.*

The influence of these concepts and the idea of direct manipulation can be seen in the WIMP (Windows, Icons, Menus, and Pointers) metaphor [Preece *et al.* 2002], which was codified by the Xerox PARC team in the 1970s. The WIMP metaphor first appeared commercially in the Xerox 8010 Star system in 1981 and is used in most of the current desktop interfaces of today. Essentially, the WIMP metaphor defines a GUI for a 2D desktop environment (Figure 2.7).

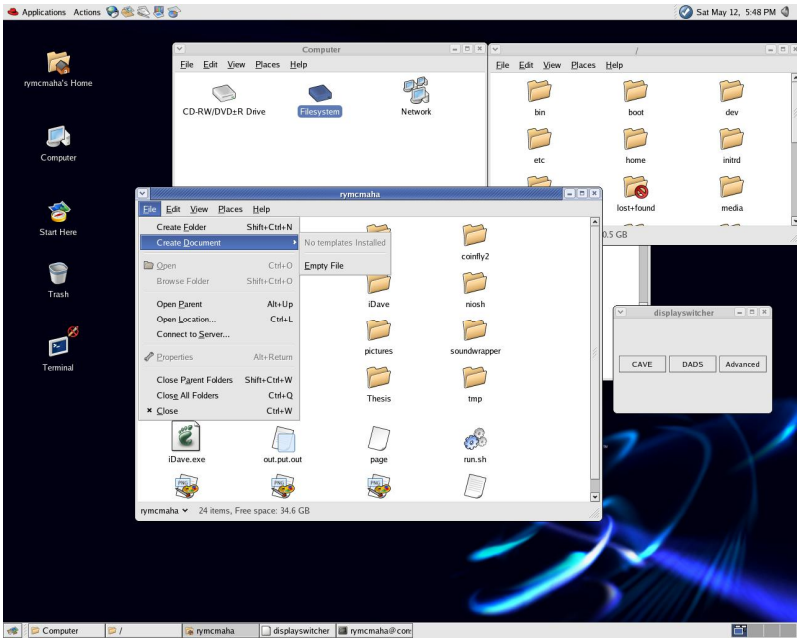


Figure 2.7 An example of a desktop GUI using the WIMP metaphor.

In the WIMP metaphor, most objects have visual representations (icons) and users complete commands by acting upon those representations with pointers. Returning to the previous

example, to remove a file named “foo.txt” from the current directory, a user would right click with a mouse device on the icon labeled “foo.txt” and then left click on a “Delete” command in a pop-up menu displayed near the icon of interest. Another common alternative for this example is for the user to click and drag the icon labeled “foo.txt” from the current folder (or directory) to the trashcan or recycling bin icon on the desktop. In this alternative example, the trashcan or recycling bin represents a command or action. This interface style of direct manipulation forces users to think about their objects of interest first and then the actions to carry out upon those objects. Since objects are specified first, followed by actions, in order to complete a task, we classify the task sequences underlying the WIMP metaphor as Object-Action task sequences.

2.2.3 Explaining the Evolution of Interfaces

To help explain the evolution of GUIs from CLIs, Shneiderman [Shneiderman 1998] presented some of the beneficial points of GUIs:

- Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user.
- Experts can work rapidly to carry out a wide range of tasks, even defining new functions and features.
- Knowledgeable intermittent users can retain operational concepts.
- Error messages are rarely needed.
- Users can immediately see whether their actions are furthering their goals, and, if the actions are counterproductive, they can simply change the direction of their activity.
- Users experience less anxiety because the system is comprehensible and because actions can be reversed easily.
- Users gain confidence and mastery because they are the initiations of action, they feel in control, and they can predict the system responses.

These benefits explain some of the reasons why interfaces have evolved from CLIs to GUIs, but these reasons are closely linked to direct manipulation and not the underlying task sequences of these two different interface styles.

2.3 Motivation to Examine Task Sequences

The evolution of GUIs from CLIs has been examined but the evolution of task sequences and the replacement of the Action-Object task sequence by the Object-Action task sequence have not been examined in the literature.

Shneiderman [Shneiderman 1998] presents the example of knowing whether to drag a file to the trashcan or to drag the trashcan to the folder as a syntactic aspect of direct manipulation. Unfortunately, there has been little or no research on these syntactic aspects of direct manipulation and thanks to the WIMP metaphor, the Object-Action task sequence has emerged as the standard for 2D desktop environments, whether this task sequence is superior to the Action-Object task sequence or not.

GUIs and 2D desktop environments have already been standardized with the Object-Action task sequence. Even if we could prove that the Action-Object task sequence would be a better syntactic choice for direct manipulation in these 2D desktop environments, it would be impossible to convince millions of users to change their standard conceptions of 2D desktop environments.

However, immersive virtual environments have not been standardized with a task sequence. Because virtual reality is still an emerging technology with few everyday users, 3DUIs for IVEs have not been standardized yet. It is still possible to evaluate task sequences to determine if one syntactic approach is better than the other and to apply that evaluation towards setting standards for IVEs. This opportunity may allow us to set proper standards for the underlying task structures involved with interacting with an IVE application. Most importantly, establishing standards for task sequences for SCIs will bring us one step closer to creating a testbed for these interfaces.

Chapter 3 Identifying Task Sequences

3.1 Identifying Tasks

A task sequence is the order of operations in a system control task [McMahan and Bowman 2007]. We have already mentioned two types of task sequences in the previous chapters: the Action-Object task sequence and the Object-Action task sequence. With the Action-Object task sequence, the user identifies the action before identifying the object, and with the Object-Action task sequence, the user identifies the object before identifying the action.

For this research, we wanted to identify all of the possible task sequences that may be used in a SCI for an IVE, before attempting to evaluate such sequences. To help identify possible task sequences, we first needed to consider as many different IVE tasks as possible. We surveyed the 3DUI mailing list (http://www.3dui.org), a discussion list with participation by researchers from around the globe, asking for basic user tasks and complex user goals that have been encountered in IVE applications. We specifically asked for lists of basic user tasks and how these encountered tasks were completed in regard to system control. We also asked for lists of complex user goals that require the combination of several basic user tasks and how these goals could be completed in regard to system control. From this survey (see Appendix A), we were able to compile the following list of basic user tasks:

- Place a predefined object within the environment
- Dynamically create an object within the environment
- Remove an object from the environment
- Translate an object
- Rotate an object
- Scale an object
- Change the visual properties of an object
- Change the abstract properties of an object
- View the abstract properties of an object
- Copy an object

We also gathered much more information on how these basic user tasks could be completed and on complex user goals comprised of these basic components, but we found the list of basic user tasks to be most beneficial to identifying task sequences. Using the list, we identified three levels of complexity for task sequences: simple, complex, and combination. Each level has multiple possible task sequences. Note that the task sequence does not specify how the SCI is designed, but only the sequence of selections the user must perform. We also classified task sequences by two different foci: object-oriented and location-oriented.

3.2 Simple Task Sequences

The first level of complexity that we identified was simple task sequences. The majority of the basic user tasks compiled from the survey required the indication of an action and an object. Based on this fact, we determined that a simple task sequence would include only the indication of one action and one object. We restricted simple task sequences to the indication of *one* action and *one* object to avoid questions regarding the use of modes and multiple-object selections. We also considered the combination of the indication of the action with the indication of the object, but classified this task sequence in our third level of complexity – combination task sequences (Section 3.4).

Therefore, we identified only two simple task sequences – the Action-Object task sequence and the Object-Action task sequence. To explain these task sequences again, consider the example of a user wanting to remove a bookshelf from an IVE. In the Action-Object task sequence, the user would indicate the "Remove" command, using the SCI, and then select the bookshelf to be removed. In the Object-Action task sequence, the user would select the bookshelf and then, using the SCI, indicate the "Remove" command. See Figure 3.1 for examples of SCIs that implement these simple task sequences (NOTE: These are not the only ways to implement SCIs for simple task sequences).

A

B

Figure 3.1 Implementations of a remove command using **A**) the Action-Object task sequence and **B**) the Object-Action task sequence.

3.3 Complex Task Sequences

The second level of complexity that we identified was complex task sequences. Some of the basic user tasks compiled from the survey required the indication of an action, an object, and one or more parameters. To simplify the process of identifying task sequences, we decided to assume that if a task required the indication of any number of parameters that all of those indications would be made simultaneously, and no task would require more than one indication for all of the parameters necessary to complete the task. Therefore, we determined that a complex task sequence would include the indication of *one* action, *one* object, and *one* parameter.

Using the definition of a complex task sequence, six complex task sequences can be identified:

- Action-Object-Parameter
- Action-Parameter-Object
- Object-Action-Parameter
- Object-Parameter-Action
- Parameter-Action-Object
- Parameter-Object-Action

We decided to not evaluate three of these complex task sequences for our research: Object-Parameter-Action, Parameter-Action-Object, and Parameter-Object-Action. All three of these complex task sequences involve the indication of a parameter before the action that will utilize that parameter. Some applications may experiment with this feature, but for most IVE applications it does not make sense to have the user indicate a parameter before knowing which command the parameter will be used for. This is why we decided to consider only the Action-Object-Parameter, Action-Parameter-Object, and Object-Action-Parameter task sequences.

Consider the task of rotating a piece of furniture within the IVE by 45^{o}. In the Action-Object-Parameter task sequence, the user uses the SCI to indicate the "Rotate" command, selects the piece of furniture to rotate, and the uses the SCI again to indicate the "45^{o}" parameter. In the Action-Parameter-Object task sequence, the user uses the SCI to indicate the "Rotate" command, followed by the "45^{o}" parameter, and then selects the piece of furniture to rotate. In the Object-Action-Parameter task sequence, the user selects the piece of furniture to rotate and then uses the SCI to indicate the "Rotate" command followed by the "45^{o}" parameter. See Figure 3.2 for examples of SCIs that implement complex task sequences (NOTE: These are not the only ways to implement SCIs for complex task sequences).

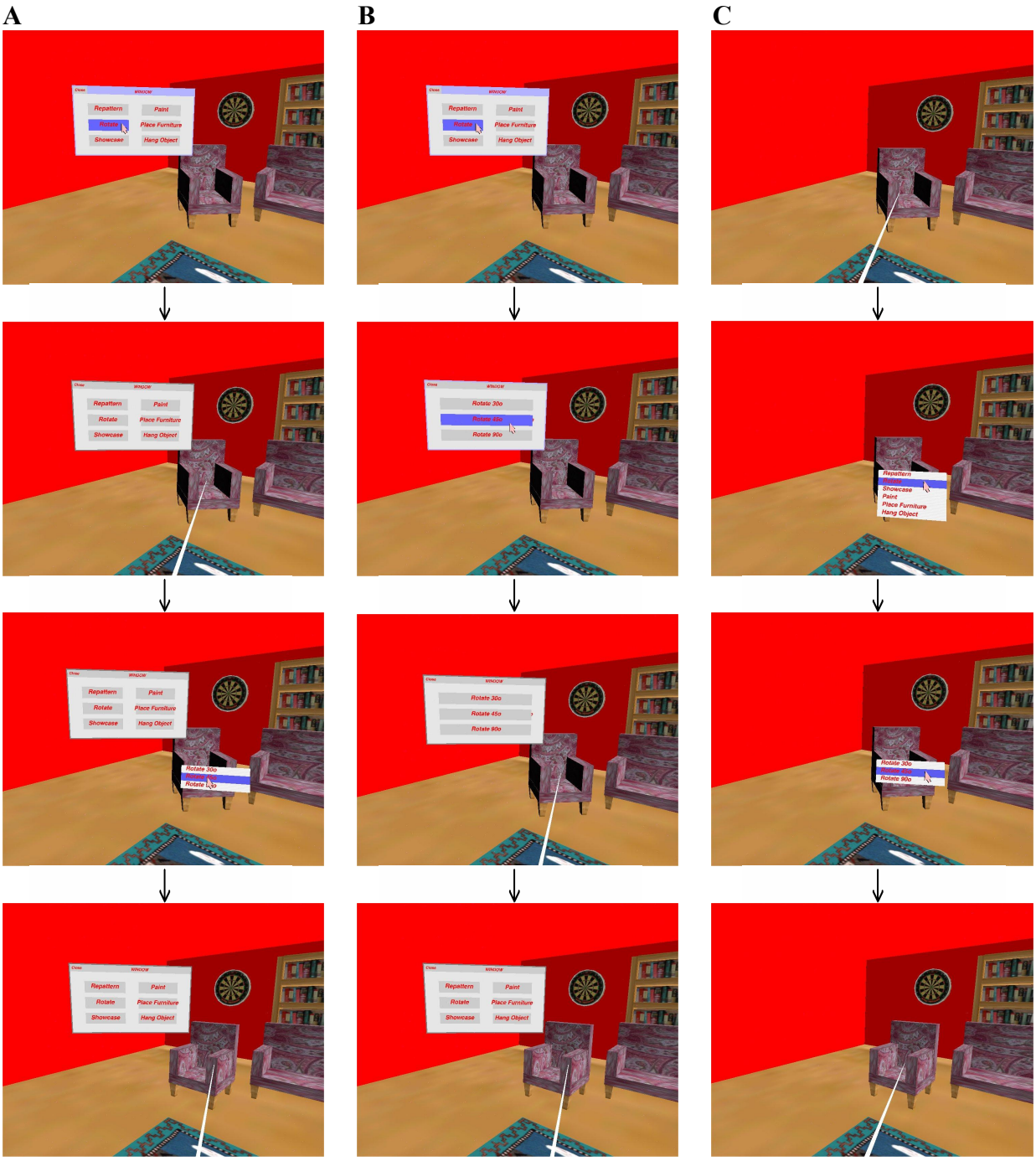


Figure 3.2 Implementations of a rotate command with a 45° parameter using **A**) the Action-Object-Parameter task sequence, **B**) the Action-Parameter-Object task sequence, and **C**) the Object-Action-Parameter task sequence.

3.4 Combination Task Sequences

The third level of complexity that we identified was combination task sequences. Looking back to the tasks compiled from the survey, we decided that some of the indications required could be combined into single indications – creating combination task sequences. Using the same assumptions made with simple task sequences and complex task sequences about using only *one* of each type of indication, we identified the following combination task sequences (NOTE: "+" indicates a combination of the two adjacent indications into a single required indication):

- Action+Object
- Action+Object-Parameter
- Action-Object+Parameter
- Action+Object+Parameter
- Action+Parameter-Object
- Action-Parameter+Object
- Action+Parameter+Object
- Object+Action
- Object+Action-Parameter
- Object-Action+Parameter
- Object+Action+Parameter
- Object+Parameter-Action
- Object-Parameter+Action
- Object+Parameter+Action
- Parameter+Action-Object
- Parameter-Action+Object
- Parameter+Action+Object
- Parameter+Object-Action
- Parameter-Object+Action
- Parameter+Object+Action

By eliminating equivalent task sequences, such as Action+Object-Parameter and Object+Action-Parameter, we can reduce the list of identified combination task sequences to the following:

- Action+Object
- Action+Object-Parameter
- Action-Object+Parameter
- Action+Object+Parameter
- Action+Parameter-Object
- Object-Action+Parameter
- Object+Parameter-Action
- Parameter-Action+Object

We decided to not evaluate four of these complex task sequences for our research: Action+Object, Action+Object+Parameter, Object+Parameter-Action, and Parameter-Action+Object. We decided not to evaluate the Action+Object and Action+Object+Parameter task sequences because we felt that these single indication task sequences would be difficult to implement in most SCIs. One possible implementation would be widgets attached to each object representing every possible combination of actions and parameters, but studying single selections is not very interesting research. We decided not to evaluate the Object+Parameter-Action and Parameter-Action+Object task sequences because the parameter is indicated before the action (see Section 3.3). Therefore, for combination task sequences, we only considered the Action+Object-Parameter, Action-Object+Parameter, Action+Parameter-Object, and Object-Action+Parameter task sequences.

To look at these combination task sequences closer, let us revisit the task of rotating a piece of furniture within the IVE by 45°. In the Action+Object-Parameter task sequence, the user would use the SCI to indicate the "Rotate" command and the piece of furniture to rotate at the same time and then use the SCI to indicate the "45°" parameter. For the Action-Object+Parameter task sequence, the user would use the SCI to indicate the "Rotate" command and then would use the SCI to indicate the piece of furniture to rotate and the "45°" parameter at the same time. For the Action+Parameter-Object task sequence, the user would use the SCI to indicate the "Rotate" command and the "45°" parameter at the same time and then would select the piece of furniture to rotate. In the Object-Action+Parameter task sequence, the user selects the piece of furniture to rotate and then uses the SCI to indicate the "Rotate" command and the "45°" parameter at the same time.

Figure 3.3 shows examples of SCIs that implement combination task sequences. Note that these examples are not the only ways to implement SCIs for combination task sequences. Additionally, these examples may not be optimal for scalability. For instance, the example SCI for the Object-Action+Parameter task sequence would be very unusable if an object had several functions and parameters associated with it. In this instance, the many 3D widgets would occlude the environment and each other.

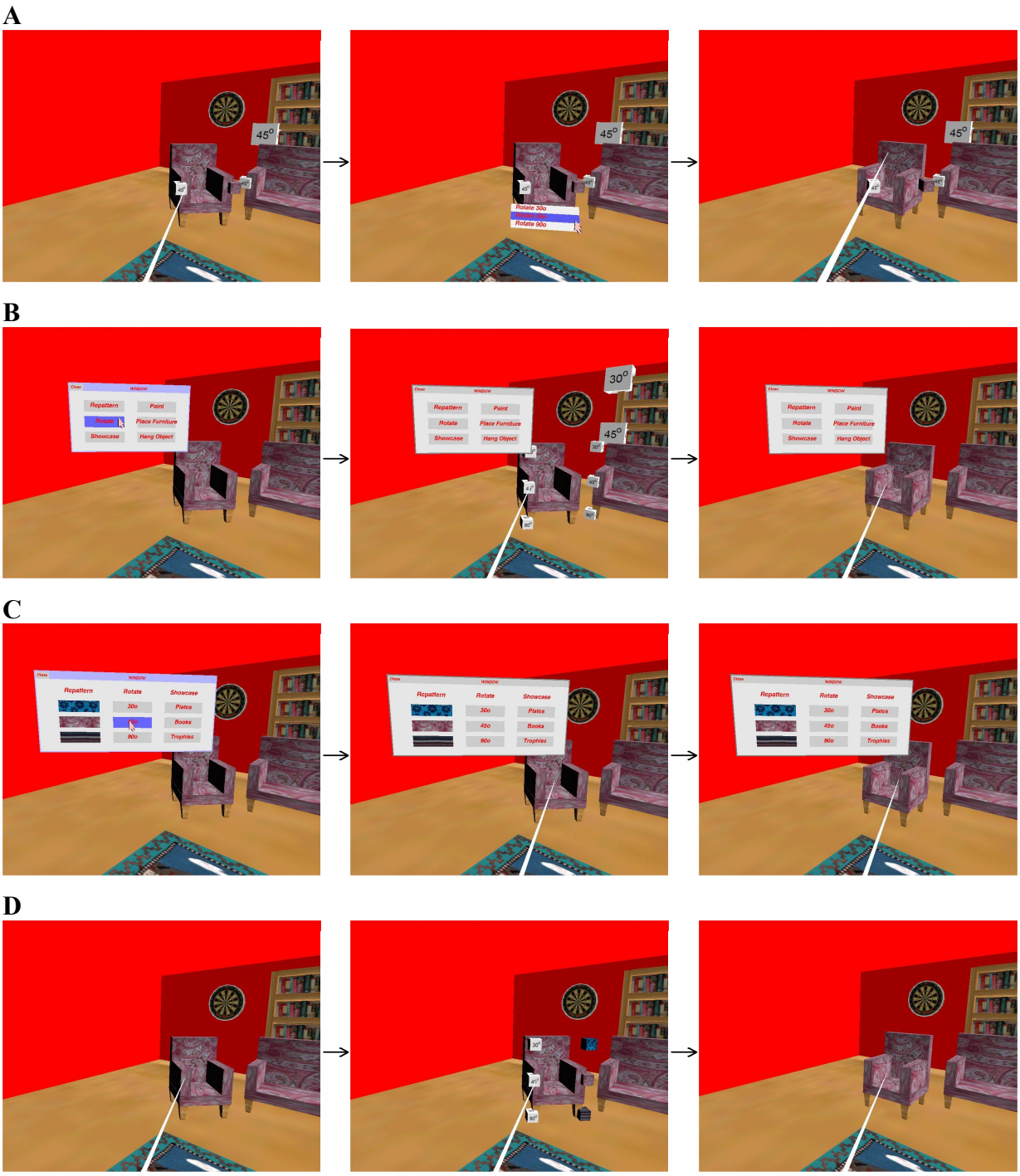


Figure 3.3 Implementations of a rotate command with a 45° parameter using A) the Action+Object-Parameter task sequence, B) the Action-Object+Parameter task sequence, C) the Action+Parameter-Object task sequence, and D) the Object-Action+Parameter task sequence.

3.5 Foci of Task Sequences

For the previous sections on simple, complex, and combination task sequences, the task sequences discussed address object-oriented tasks. In IVEs, some tasks may not involve an object but instead a location within the environment. This fact caused us to classify task sequences by two different foci: object-oriented and location-oriented. Location-oriented task sequences are the same as their object-oriented counterparts except that the user selects a location within the IVE instead of selecting an object. For example, consider the task of placing a cube within the VE. The location-oriented version of the Action-Parameter-Object task sequence for this task would have the user use the SCI to indicate a place command followed by a cube parameter, and then the user would select the desired location of the cube in the environment.

By classifying task sequences as object-oriented or location-oriented, we essentially double the number of task sequences identified. We could re-list all of the task sequences, replacing the term "Object" with "Location". For example, we could talk about the Action-Location and Location-Action task sequences. To avoid doubling our lists, we simply refer to location-oriented task sequences by their object-oriented counterparts and note the foci, location-oriented. Therefore, a task sequence that involves the indication of an action and then a location would be referred to as a location-oriented Action-Object task sequence. The decision to choose "Object" over "Location" was based on the greater number of object-oriented basic user tasks than location-oriented basic user tasks.

Another consideration about the foci of a task sequence is that location-oriented combination task sequences are extremely hard to implement with a SCI. Consider the Action+Object-Parameter combination task sequence. For an object-oriented version of this task sequence, simple 3D widgets solve the design problem of combining the indication of the action and the object into a single indication (see Figure 3.3A). For a location-oriented version of this task sequence, if 3D widgets were used, the entire virtual environment would be cluttered with widgets that would occlude the environment itself and each other. For this reason, we decided to not consider location-oriented combination task sequences for our research, since this is impractical for SCIs to implement.

3.6 Summary

In this chapter, we have identified the following possible task sequences for SCIs:

Simple Task Sequences

- Action-Object
- Object-Action

Complex Task Sequences

- Action-Object-Parameter
- Action-Parameter-Object
- Object-Action-Parameter
- Object-Parameter-Action
- Parameter-Action-Object
- Parameter-Object-Action

Combination Task Sequences

- Action+Object
- Action+Object-Parameter
- Action-Object+Parameter
- Action+Object+Parameter
- Action+Parameter-Object
- Object-Action+Parameter
- Object+Parameter-Action
- Parameter-Action+Object

Additionally, we identified two classifications for task sequences: object-oriented and location-oriented.

We have also explained our rationale for considering only the following task sequences for our research:

Simple Task Sequences

- Action-Object (object- and location-oriented)
- Object-Action (object- and location-oriented)

Complex Task Sequences

- Action-Object-Parameter (object- and location-oriented)
- Action-Parameter-Object (object- and location-oriented)
- Object-Action-Parameter (object- and location-oriented)

Combination Task Sequences

- Action+Object-Parameter (object-oriented only)
- Action-Object+Parameter (object-oriented only)
- Action+Parameter-Object (object-oriented only)
- Object-Action+Parameter (object-oriented only)

Chapter 4 Evaluating Task Sequences

4.1 Motivation

As mentioned in Section 1.3, this research focuses on evaluating task sequences in regard to usability, with the goal of potentially standardizing task sequences for SCIs. By establishing a standard for task sequences, we could eliminate one confound for comparing these interfaces and would be closer to establishing a testbed for SCIs.

The first step in evaluating task sequences is to decide *what* to evaluate and *how* to accomplish that evaluation. In the course of our research we attempted to evaluate task sequences using three different methods. These three methods focused on user instincts (Section 4.2), user performance (Section 4.3), and user preferences (Section 4.4). We decided to focus upon these user aspects because they all affect and contribute to usability.

4.2 User Instincts for Task Sequences

The first method of evaluating task sequences that we considered was focused on user instincts for task sequences. We believe that for certain tasks a user will have inherent ideas of how to achieve these tasks. For instance, for the task of moving a chair across a room, the user might break down the task into the subtasks of picking up the chair, carrying the chair to the desired location, and then setting the chair back down. If this breakdown was an inherent idea for everyone, it could be considered an instinct, specifically a *task sequence instinct*.

The concept of task sequence instincts also contributes to the concept of making 3DUIs more natural, which researchers have been striving for in the past few years. If task sequence instincts exist, then 3DUIs that conform to these inherent ideas of how to complete certain tasks would be more natural. By making more natural 3DUIs, user performance and usability improve as users know what to expect of and how to interact with these interfaces.

With the possible benefits of determining task sequence instincts in mind, we set out to design an experiment to evaluate user instincts for task sequences. Specifically, we wanted to determine if users have instincts about completing certain types of tasks. We decided to reuse the list of basic user tasks from Section 3.1 as a list of tasks to evaluate user instincts for, but we still did not know *how* to evaluate task sequence instincts.

Our first experimental design was to present a simple SCI that users would be free to decide how to use for completing a task. Consider the task of removing an object from the environment. We could present the user with an adapted 2D menu with a single button labeled "Remove", a single object within the environment to be removed, and a verbal directive of the task. If the user had an Action-Object task sequence instinct for this remove task, the user would presumably select the "Remove" button first and then select the object. If the user had an Object-Action task sequence instinct, the user would presumably select the object first and then the "Remove" button.

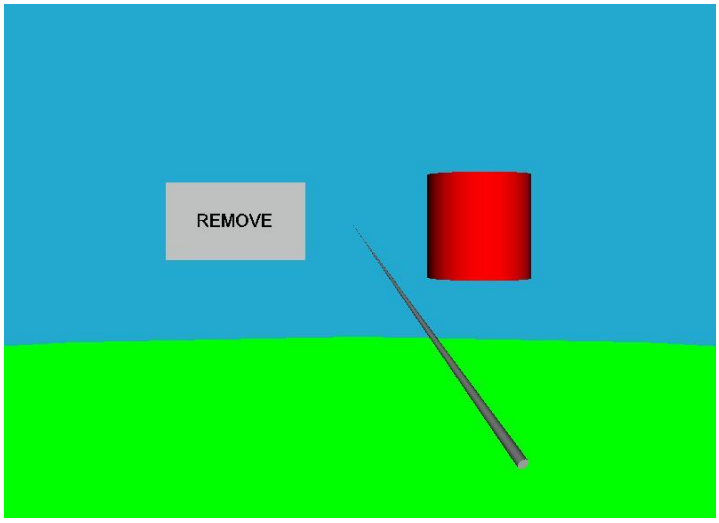


Figure 4.1 First experimental design for evaluating task sequence instincts.

This initial experimental design for evaluating user instincts for task sequences seemed promising until we began developing the details of the experiment. The major problem we encountered was how to eliminate interface affordances. For instance, people tend to notice large objects before smaller objects. Therefore in the aforementioned interface, if the "Remove" button was larger than the object to be removed, the interface affords an Action-Object task sequence as the user will probably notice the action before the object. On the other hand, if the object to be

removed was larger than the "Remove" button, then the interface affords an Object-Action task sequence. There are several other types of affordances to consider: people tend to read left to right, people tend to read top to bottom, people notice closer objects before objects farther away, and people notice highly contrasted objects before objects that have little contrast with the background.

We attempted to address the major problem of interface affordances but found that stripping away these affordances is a daunting task. Deciding where to place a menu in relation to the object of focus is the most daunting part of this task. If the menu is placed to the left of the object or above the object, the interface affords an Action-Object task sequence. If the menu is placed to the right of the object or below the object, the interface affords an Object-Action task sequence. Regardless of the position of the menu, the interface will afford one task sequence over another. One idea would be to balance other affordances, such as making objects larger or using higher contrasts, with menu position. Unfortunately, it is impossible to determine that balance as some users may be more affected by menu position while others may be more affected by larger objects.

Due to the affordances that interfaces provide and our inability to strip away these affordances, we decided to abandon our first experimental design. Instead, we began to pursue an alternative experimental design using "think-aloud" exercises and no SCI at all. The concept behind our second experimental design was to present the user with a verbal description of a situation and a task to complete in that situation, and then have the user explain how they would complete the task. An example of such a think-aloud exercise would be "There is a cube in front of you that you want to paint red. How would you complete this task?" Some possible responses may be "Use a paintbrush and paint it red" or "Pick up the cube and dip it in red paint."

Unfortunately there were problems with this experimental design as well. The most important problem discovered was that how the task is verbally communicated to the user can affect how the user thinks about the task. Telling the user "there is a cube in front of you that you want to paint red" suggests an Object-Action task sequence to the user since the object is mentioned to the user before the action. Alternatively, telling the user "you want to paint a cube in front of you

red" suggests an Action-Object task sequence since the action is mentioned to the user before the object. Another problem with this experimental design would have been correlating user responses to task sequences. For example, what task sequence would "use a paintbrush, pick up the cube, and paint it" correlate to?

Since our first experimental design failed to strip away interface affordances and our second experimental design failed to provide a non-biased form for presenting tasks, we decided to abandon the entire evaluation of user instincts for task sequences. We still believe that task sequence instincts may exist and would give great insight for designing 3DUIs, but we were unable to successfully evaluate such instincts.

4.3 Effects of Task Sequences on User Performance

The second method of evaluating task sequences that we considered was determining the effects of task sequences on user performance. User performance is closely tied to usability and is often considered the most important factor of usability. Often absolute user performance – how long it takes a user to complete a task – is used as a metric for user performance evaluations. Unfortunately, because absolute task performance is dependent on the SCI used to evaluate the task sequence, there is a problem in evaluating the effects of task sequence on absolute task performance.

A SCI can bias the absolute task performance of one task sequence over another due to the design and implementation of the SCI as seen in the example from Section 1.6 of the two different 2D adapted menu SCIs. The first SCI – a statically located window – is biased for the Action-Object task sequence since the user will always start by focusing ahead and, for the Object-Action task sequence, the SCI forces the user to return focus to the same location, no matter where the object is. The second SCI – a dynamically located window – is biased for the Object-Action task sequence since the user is able to quickly access the window near the object and, for the Action-Object task sequence, the SCI requires the user to search in the location of the last object before beginning a new task.

Since the SCI used to evaluate task sequences could be biased, an alternative method to absolute task performance comparison for evaluating the effects of task sequence on user performance is necessary. An alternative is to compare the cognitive effort induced by each task sequence. This eliminates any effects that a specific implementation of a SCI may have on the evaluation of a task sequence and provides generalized results. Unfortunately it is difficult to measure cognitive effort because we do not know exactly what the user is thinking. It is hard to determine how much of the user's cognitive effort is used to think about the task sequence and how much is used to think about interacting with the SCI. To alleviate this problem, we developed a model similar to the Keystroke-Level Model (KLM) [Card *et al.* 1980].

The KLM was proposed as a model for predicting the time it takes an expert user to perform a given task on a given computer system. The model is based on counting keystrokes and other low-level operations, including the system's responses and the user's mental preparations. The KLM asserts that executing a task can be described in terms of four different physical-motor operators (keystroking, pointing, homing, and drawing), one mental operator (by the user), and one response operator (by the system). Execution time is simply the sum of the time for each of these operators, as seen in Equation 4.1.

$$T_{Execute} = T_K + T_P + T_H + T_P + T_M + T_R \qquad (4.1)$$

Using a similar concept, we created a model which defines the total time to complete a system control task as the sum of the time for the perceptual and motor processes needed to select the necessary components (action, object, and parameter) and the cognitive time induced by the underlying task sequence (Equation 4.2). These times are not necessarily non-overlapping as humans are able to perform actions while thinking about other tasks. For example, a user could think of the next indication to perform in the middle of moving his arm to perform the current indication. Regardless, we are still accounting for the additional time required when the user is forced to think about the current task sequence.

$$T_{Total} = T_{Perceptual\text{-}Motor} + T_{Cognitive} \qquad (4.2)$$

Using our model for user performance, we are able to estimate the cognitive time induced by a task sequence by subtracting the estimated time to make all necessary selections from the total time the user required to complete the task. For example, consider the task of removing a cube from the IVE. If the user takes 1.0 second to perceptually recognize and physically select the SCI component for the remove command and takes 1.0 second to perceptually recognize and physically select the cube, then $\mathbf{T_{Perceptual\text{-}Motor}}$ is 2.0 seconds. If the user takes a total time, $\mathbf{T_{Total}}$, of 2.5 seconds to remove the cube from the IVE, then the cognitive time induced by the task sequence, $\mathbf{T_{Cognitive}}$, is 0.5 seconds.

An important factor in this calculation is the accuracy of estimating $\mathbf{T_{Perceptual\text{-}Motor}}$. Specifically we need to know the cognitive and physical time required to interact with the SCI for a task. Therefore a single SCI for evaluating the effects of task sequence on user performance was necessary for our research. As mentioned in Section 1.6, we chose to use the Virtual Environment Windowing Library (VEWL) [Larimer and Bowman 2003]. VEWL is a view-fixed, adapted 2D menu SCI developed on top of DIVERSE [Kelso *et al.* 2002] that utilizes a virtual sphere to provide a surface for moving windows around.

We designed our first experiment (see Chapter 5) to establish estimates for the time required to complete certain system control tasks with the VEWL SCI based solely on atomic-level interactions, such as clicking a command button and moving a window. This experiment established the expected cognitive and physical times required to interact with VEWL for various system control tasks. We then designed a second experiment (see Chapter 6) to obtain an average for the total time required to complete various tasks in an IVE application.

By subtracting $\mathbf{T_{Perceptual\text{-}Motor}}$, established in the first experiment, from $\mathbf{T_{Total}}$, established in the second experiment, we estimated $\mathbf{T_{Cognitive}}$ for different task sequences. By estimating $\mathbf{T_{Cognitive}}$ of the task sequences we identified in Chapter 3, we were able to compare these values to determine which task sequences should be used for certain tasks. More importantly, we were able to evaluate the effects of task sequences on user performance. Chapter 6 covers these results in detail.

4.4 User Preferences for Task Sequences

The third method of evaluating task sequences that we considered was determining which task sequences users prefer. User preference is a highly regarded aspect of usability within the human-computer interaction community. Sometimes developers will even design systems that have multiple methods of completing the same task as to provide users the ability to decide which method to use. For example, in most applications there are three standard methods to save a current document – accessing the "Save" command from the "File" menu, clicking on a save icon, or using the "Ctrl + S" keyboard shortcut. Providing different methods to complete the same task increases usability as users chose methods they prefer and understand best.

For this research, we were concerned with two aspects of user preferences – which task sequences novices prefer and how those preferences evolve as novices become experienced users. If the majority of novices prefer the same task sequences, then SCIs that utilize those task sequences should be easier for novices to learn than SCIs that utilize other task sequences. Hence, these SCIs would be more usable and more natural for novices. Additionally, if we track the evolution of task sequence preferences as novices become experienced users, we can determine the range of task sequences that SCIs should provide. For instance, if novices prefer the Action-Object task sequence and this preference evolves to the Object-Action task sequence, then SCIs should provide methods for using either task sequence, to accommodate novices and experienced users alike. On the other hand, if novices and experienced users both prefer the Action-Object task sequence then SCIs would be able to utilize a single task sequence without causing much detriment.

In order to evaluate which task sequences novices prefer and how those preferences evolve, we designed a longitudinal study of five sessions in which users would have choices of which task sequences to use for completing tasks (see Chapter 7). The SCI for this experiment was designed to support multiple task sequences so that users would have open preferences and would not be influenced on which task sequence to use. Throughout the sessions, we were able to record the preferences of the participants, capturing data on novices and the evolution of their preferences as they became experienced users. Chapter 7 covers the results of this study in detail.

4.5 Summary

In this chapter, we presented three different methods for evaluating task sequences. Our first method was concerned with task sequence instincts, inherent ideas shared by users about which task sequences should be used for certain tasks. Unfortunately, we were unable to design an experiment to evaluate task sequence instincts due to inabilities to strip away interface affordances and avoid biased directions for think-aloud exercises. Our second method for evaluating task sequences was concerned with user performance. Specifically, to evaluate the effects of task sequence on user performance, we examined the cognitive effort induced by task sequences. We were able to estimate these cognitive effort by building a task model (see Equation 4.2) very similar to the KLM. Our third method for evaluating task sequences was concerned with user preferences. By providing choices for which task sequences to use and capturing data from several sessions of a longitudinal study, we were able to evaluate user preferences for task sequences.

Chapter 5 VEWL Performance Study

5.1 Virtual Environment Windowing Library

As mentioned in Section 4.3, for evaluating the effects of task sequences on user performance, we decided to use the Virtual Environment Windowing Library (VEWL) [Larimer and Bowman 2003] for the SCI. Different from most adapted 2D menu interfaces, VEWL utilizes a virtual sphere as a surface for content to be maintained tangent to and to be moved along. This ensures that all windows and widgets are equal distance to the user and allows for placement of content anywhere around the user. Users can "click on" these windows and widgets using ray-casting and the VEWL mouse, an adapted 2D mouse that moves along the inside of the virtual sphere.

Windows are perhaps the most important widget in VEWL. These widgets are view-fixed, adapted 2D windows that enable all of the other VEWL widgets aside from popup menus. VEWL windows remain tangent to the VEWL virtual sphere at all times, but can be moved along the surface of the virtual sphere by the user. A VEWL window also has a single button in the upper left corner for closing or removing the window. A VEWL window can be seen in Figure 5.1.

Figure 5.1 A VEWL window enabling no other widgets.

Aside from windows, VEWL supports several widgets including checkboxes, command buttons, dropdown menus, and scrolling lists. The VEWL checkbox widget is nearly identical to the checkbox widgets found in most desktop GUIs. The checkbox has two main parts – a box which can appear checked or unchecked and a label which may contain a string to identify the widget (see Figure 5.2). Both the box and the label can be clicked once to change the state of the checkbox from unchecked to checked or vice versa.

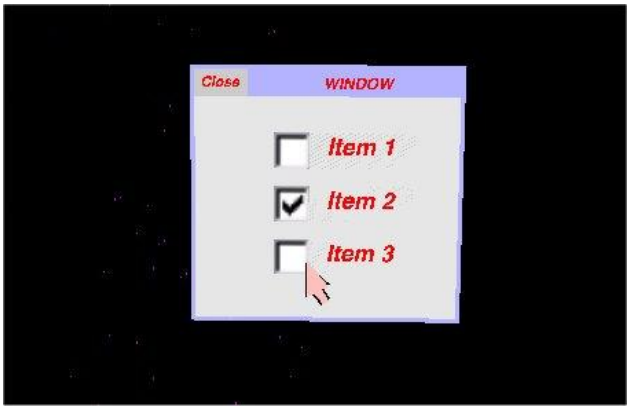


Figure 5.2 VEWL checkboxes can be unchecked or checked.

The VEWL command button widget is equivalent to the standard buttons found in most desktop applications. The command button is presented as a raised rectangular box on a VEWL window and may either be textured with a pattern or contain a label to help identify the widget (see Figure 5.3). Command buttons are normally clicked once to execute or indicate a command or system control task.

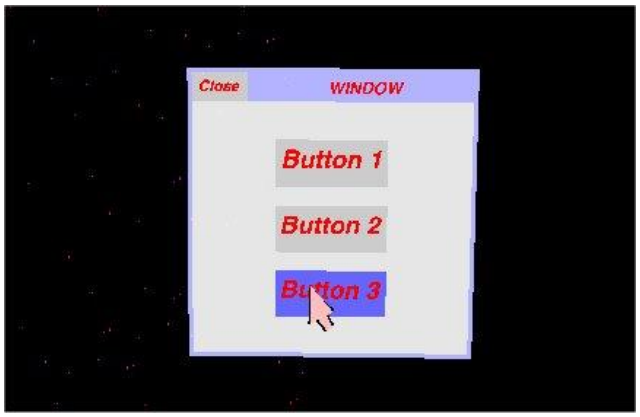


Figure 5.3 VEWL command buttons are used to execute or indicate commands.

The VEWL dropdown menu widget is similar to the dropdown menus found in most desktop applications. The widget has two states and consists of several parts. The first dropdown menu state is non-activated. During this state, the dropdown menu appears as a single widget with a label representing the item selected from the menu and a down arrow indicating that the widget can be activated (see Figure 5.4A). The second dropdown menu state is activated. During this state, the dropdown menu appears as a list of items with several mini-widgets, similar to a scrollbar, located on the right side of the list. There are a total of five mini-widgets that are used to scroll the list (see Figure 5.4B). The first mini-widget is the *UpArrow* and moves the list up by one single item when clicked. Below the *UpArrow* is the *PageUp*, which moves the list up by an entire page when clicked. The middle mini-widget is the *CurrentPage*, which only indicates the location of the current page within the entire list. The *PageDown* mini-widget is below the *CurrentPage* and moves the list down by an entire page when clicked. The last mini-widget is the *DownArrow*, which moves the list down by a single item when clicked. Once an item in the list has been clicked, the dropdown menu changes state from activated back to non-activated.

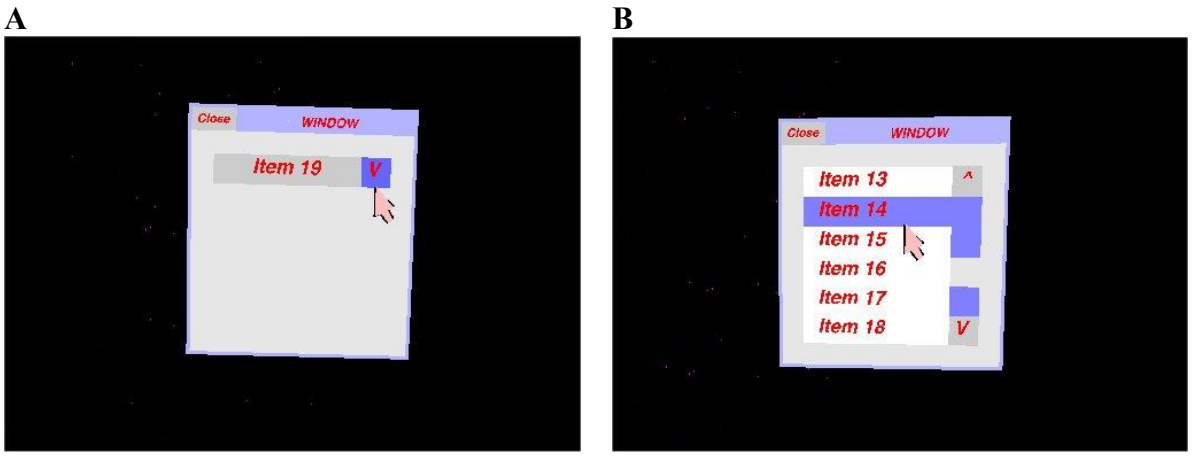


Figure 5.4 VEWL dropdown menus have **A)** non-activated states and **B)** activated states.

The VEWL scrolling list widget is essentially equivalent to a VEWL dropdown menu in activated state. Like VEWL dropdown menus, scrolling lists have a list of items with the same five mini-widgets – *UpArrow*, *PageUp*, *CurrentPage*, *PageDown*, and *DownArrow* (see Figure 5.4B). Once an item in the list has been clicked, the scrolling list keeps that item highlighted until another item is clicked.

The VEWL popup menu widget is the only widget that does not require a VEWL window. VEWL popup menus are similar to the context menus found in most desktop GUIs. The popup menu is presented as a column of buttons that are considered menu items. Some menu items can be secondary popup menus and cause adjacent columns of buttons to appear once clicked (see Figure 5.5). The menu items in popup menus are normally clicked once to execute or indicate a command or system control task and then the popup menu disappears, as a context menu would.

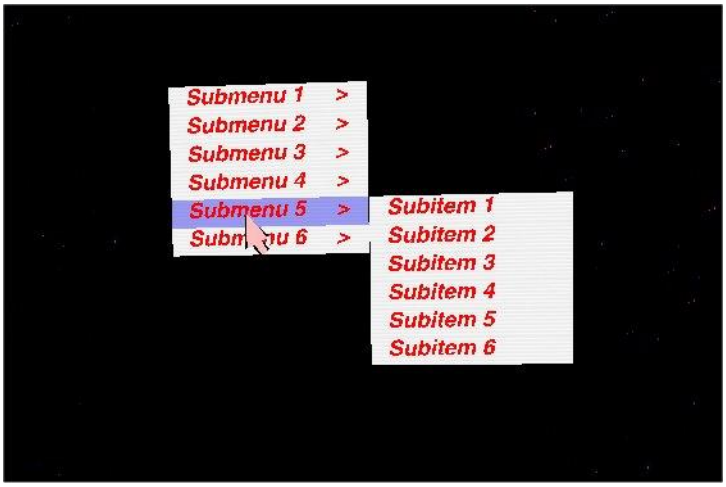


Figure 5.5 VEWL popup menus are the only widgets that don't require a VEWL window.

5.2 Goals

Since we wanted to use the task sequence performance model described by Equation 4.2 for estimating the cognitive effort of different task sequences, the primary goal of the VEWL performance study was to establish expected estimates for $\mathbf{T_{Perceptual\text{-}Motor}}$ by evaluating the time required to complete certain system control tasks with VEWL. Since we had not designed the second study for evaluating the effects of task sequences on user performance (see Chapter 6), we did not know which atomic-level VEWL interactions would be used in the SCI for the second study. Hence, we decided to establish estimate performance data for all of the possible atomic actions found in VEWL.

To accomplish the goal of establishing estimate performance data on all of the atomic actions in VEWL, we first listed each widget and its corresponding atomic actions:

Window

- Focusing on
- Moving
- Closing

Checkbox

- Changing the state

Command Button

- Clicking

Dropdown Menu

- Activating
- Scrolling the list
- Selecting an item from the list

Scrolling List

- Scrolling the list
- Selecting an item from the list

Popup Menu

- Selecting an item from the root level
- Selecting an item from a secondary level

For the VEWL performance study, our goals were to establish expected estimates for completing the atomic actions listed above under different circumstances.

5.3 Implementation

For the VEWL performance study, we used a four-screen CAVE™ [Cruz-Neira *et al.* 1993] in our experiments. The CAVE has three sides (10' x 9') and a floor (10' x 10') and uses projection technology, stereoscopic displays, and head tracking to create an IVE. The CAVE provides a 270-degree horizontal field of regard (FOR) and a 100-degree horizontal field of view (FOV) when viewed through active stereo glasses. Unfortunately, for our study, the floor projector of the CAVE was unavailable and we were forced to avoid using this area of the IVE for our study.

In addition to the CAVE, we used an Intersense IS-900 tracking system with a 6-DOF head tracker and a 6-DOF wand device with four buttons and a 2-DOF joystick. We used DIVERSE [Kelso *et al.* 2002] and VEWL [Larimer and Bowman 2003] to create the software for the study.

5.4 Experimental Design and Procedure

Since our goals for the VEWL performance study were to establish expected estimates for completing the atomic actions listed in Section 5.2 under different circumstances, we designed six different experiments for each of the six VEWL widgets – windows (Section 5.6), checkboxes (Section 5.7), command buttons (Section 5.8), dropdown menus (Section 5.9), scrolling lists (Section 5.10), and popup menus (Section 5.11). Each experiment consisted of several tasks to be completed for collecting data on user performance and would take approximately thirty minutes to complete.

For each task, participants were asked to stand in the center of the CAVE floor with their hands down by their sides and looking at the center of the front CAVE wall before starting. This was to establish a standard starting body position and eliminate variances in body positions between participants. The participants were also asked to complete each task as effectively and quickly as possible. The software used for each experiment was used to record the total time for each task, indicated by the experimenter entering an identifying number of the task and the user completing the identified task.

Participant sessions began with the administration of a background survey (Appendix B) to collect information about the participants. After completing the background survey, each participant was randomly assigned to attempt two of the six experiments, which would make each session approximately one hour long in time. After completing the two experiments, participants were thanked for their time and were finished with their sessions.

5.5 Participants

For the entire VEWL performance study, we recruited 30 unpaid participants (21 male, 9 female) through various Virginia Tech listservs and classes. The age range of the participants was 18 to 31 years old with 22 being the average age. All of the participants used computers daily and all but one used Windows as their operating system of choice. Of the 30 participants, 14 had corrected vision (glasses or contacts). Six of the participants used their left hands for the experiments while the other 24 used their right hands. Twelve of the participants had previous virtual reality experiences with nine having experiences in the CAVE.

Unfortunately, we did not keep records of which participants completed which experiments so we were unable to analyze the statistics of participants per experiment.

5.6 Window Experiment

For this experiment, we were concerned with the three atomic actions of focusing a window, moving a window, and closing a window. For the atomic action of focusing a window, we evaluated two independent variables – *CAVE location* and *window size* – for significant effects on user performance.

CAVE location refers to what location within the IVE the widget is located, relative to the CAVE itself. For example, a front wall CAVE location indicates that the widget was located in the IVE such that the widget appears at the center of the front wall of the CAVE. A left seam CAVE location indicates that the widget appears across the seam between the left and front CAVE walls.

We evaluated CAVE location for windows at five levels – *left wall*, *left seam*, *front wall*, *right seam*, and *right wall*. We could have evaluated more levels if the floor projector of the CAVE had been available at the time of the VEWL performance study. For window size, we evaluated the three levels of *small* (250 pixels x 250 pixels), *medium* (500 pixels x 500 pixels), and *large* (750 pixels x 750 pixels). Table 5.1 details the evaluation of focusing a window.

Table 5.1 Experimental Design for Focusing a Window

Task ID	CAVE Location	Window Size
1	Left	Small
2	Left Seam	Small
3	Front	Small
4	Right Seam	Small
5	Right	Small
6	Left	Medium
7	Left Seam	Medium
8	Front	Medium
9	Right Seam	Medium
10	Right	Medium
11	Left	Large
12	Left Seam	Large
13	Front	Large
14	Right Seam	Large
15	Right	Large

For the atomic action of moving a window, we evaluated the independent variable *walls spanned* for significant effects on user performance. The walls spanned equate the number of CAVE walls that the movement of the window spans. For example, 1 walls spanned is equivalent to moving a window from the front wall to the right wall while 1.5 walls spanned is equivalent to moving a window from the right seam to the left wall. We evaluated walls spanned at the four levels of *0.5*, *1.0*, *1.5*, and *2.0* walls. Table 5.2 details the evaluation of moving a window.

For the atomic action of closing a window, we evaluated the independent variable *CAVE location*. We evaluated CAVE location at five levels – *left wall*, *left seam*, *front wall*, *right seam*, and *right wall*. Table 5.3 details the evaluation of closing a window.

Table 5.2 Experimental Design for Moving a Window

Task ID	Walls Spanned
1	1.0
2	1.0
3	2.0
4	0.5
5	1.5
6	1.0
7	1.0
8	2.0
9	0.5
10	1.5
11	1.0
12	1.0
13	2.0
14	0.5
15	1.5

.

Table 5.3 Experimental Design for Closing a Window

Task ID	CAVE Location
1	Left
2	Left Seam
3	Front
4	Right Seam
5	Right

5.6.1 Procedure

The procedure for the window experiment consisted of three parts: focusing a window, moving a window, and closing a window. For focusing a window, participants were told a CAVE location. After being told to start, an unfocused VEWL window would appear at the CAVE location. The participant was then expected to click anywhere on the VEWL window to bring the window to a focused state. Tasks in Table 5.1 were randomized for each participant to avoid the effects of ordering.

For moving a window, participants were told two CAVE locations. After being told to start, a VEWL window would appear in the first CAVE location and a red cube would appear in the second CAVE location. The participant was expected to then click and drag on the window to

move it from its starting location to the location of the red cube. Tasks in Table 5.2 were randomized for each participant to avoid the effects of ordering.

For closing a window, participants were told a CAVE location. After being told to begin, a VEWL window would appear at the CAVE location. The participant was expected to then click the close button on the window to close it. Tasks in Table 5.3 were randomized for each participant to avoid the effects of ordering.

5.6.2 Results

For the atomic action of focusing a VEWL window, we performed a two-factor ANOVA (CAVE location and window size). For this action, neither independent variable had a significant effect on user performance for focusing a VEWL window. There were also no significant interactions between these two variables. Therefore, regardless of these independent variables, the expected time to focus a VEWL window is 0.440 seconds (standard error of 0.016 seconds).

Table 5.4 Two-Factor ANOVA of Focusing a Window

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	.359(a)	14	.026	.659	.811
Intercept	28.552	1	28.552	733.240	.000
CAVE Location	.028	4	.007	.178	.949
Window Size	.173	2	.087	2.227	.112
CAVE Location * Window Size	.157	8	.020	.505	.851
Error	5.179	133	.039		
Total	34.059	148			
Corrected Total	5.538	147			

(a) R Squared = .065 (Adjusted R Squared = -.034)

Table 5.5 Expected Time for Focusing a Window

Mean	Std. Error	95% Confidence Interval	
		Lower Bound	Upper Bound
.440	.016	.407	.472

For the atomic action of moving a VEWL window from one CAVE location to another, we performed a one-factor ANOVA (walls spanned). For this action, the walls spanned had a significant effect ($F=11.420$, $p<0.001$) on user performance. Considering this significant effect, we can estimate the time required to move a VEWL window based on the walls spanned during movement. Table 5.7 outlines the expected times for moving a VEWL window based on the walls spanned.

Table 5.6 One-Factor ANOVA of Moving a Window

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	1.759(a)	3	.586	11.420	.000
Intercept	84.529	1	84.529	1646.346	.000
Walls Spanned	1.759	3	.586	11.420	**.000**
Error	7.496	146	.051		
Total	99.721	150			
Corrected Total	9.255	149			

(a) R Squared = .190 (Adjusted R Squared = .173)

Table 5.7 Expected Times for Moving a Window

Walls Spanned	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
0.5	.608	.041	.527	.690
1.0	.743	.029	.685	.800
1.5	.886	.041	.805	.968
2.0	.903	.041	.821	.985

For the atomic action of closing a VEWL window, we performed another one-factor ANOVA (CAVE location). For this action, the CAVE location did not have a significant effect on user performance. Therefore, regardless of CAVE location, the expected time for closing a VEWL window is 0.591 seconds (standard error of 0.032 seconds).

Table 5.8 One-Factor ANOVA of Closing a Window

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	.299(a)	4	.075	2.475	.069
Intercept	10.343	1	10.343	342.051	.000
CAVE Location	.299	4	.075	2.475	.069
Error	.786	26	.030		
Total	11.595	31			
Corrected Total	1.086	30			

(a) R Squared = .276 (Adjusted R Squared = .164)

Table 5.9 Expected Time for Closing a Window

Mean	Std. Error	95% Confidence Interval	
		Lower Bound	Upper Bound
.591	.032	.526	.657

5.7 Checkbox Experiment

For the checkbox experiment, we were concerned with the single atomic action of changing the state of the checkbox widget. We evaluated three independent variables – *CAVE location*, *part clicked*, and *checkbox quantity* – for significant effects on user performance.

We evaluated CAVE location for checkboxes at five levels – *left wall*, *left seam*, *front wall*, *right seam*, and *right wall*. For part clicked, we evaluated the two levels of *box* and *label*, because these are the two parts that can be clicked to change the state of a checkbox. We evaluated the checkbox quantity variable, the total number of checkboxes presented to the participant, at the three levels of *1*, *3*, and *6* checkboxes. Table 5.10 details the evaluation of changing the state of a checkbox.

Table 5.10 Experimental Design for Changing the State of a Checkbox

Task ID	CAVE Location	Part Clicked	Checkbox Quantity
1	Left	Box	1
2	Left Seam	Box	1
3	Front	Box	1
4	Right Seam	Box	1
5	Right	Box	1
6	Left	Box	3
7	Left Seam	Box	3
8	Front	Box	3
9	Right Seam	Box	3
10	Right	Box	3
11	Left	Box	6
12	Left Seam	Box	6
13	Front	Box	6
14	Right Seam	Box	6
15	Right	Box	6
16	Left	Label	1
17	Left Seam	Label	1
18	Front	Label	1
19	Right Seam	Label	1
20	Right	Label	1
21	Left	Label	3
22	Left Seam	Label	3
23	Front	Label	3
24	Right Seam	Label	3
25	Right	Label	3
26	Left	Label	6
27	Left Seam	Label	6
28	Front	Label	6
29	Right Seam	Label	6
30	Right	Label	6

5.7.1 Procedure

The procedure for the checkbox experiment was divided into two parts: changing the state of the checkbox by clicking the box and changing the state of the checkbox by clicking the label. These were counterbalanced between participants to avoid effects of ordering.

For clicking the box, participants were told a CAVE location and a label number. After being given a verbal indication to start, a VEWL window appeared within the IVE at the CAVE

location with a quantity of checkboxes corresponding to the checkbox quantity variable. The participant was expected to click the box corresponding to the checkbox with the label number. Tasks 1 through 15 (Table 5.10) were randomized for each participant to avoid the effects of ordering.

For clicking the label, participants were told a CAVE location and a label number. After being given a verbal indication to start, a VEWL window appeared within the IVE at the CAVE location with a quantity of checkboxes corresponding to the checkbox quantity variable. The participant was expected to click the label corresponding to the checkbox with the label number. Tasks 16 through 30 (Table 5.10) were randomized for each participant to avoid the effects of ordering.

5.7.2 Results

We performed a three-factor ANOVA (CAVE location, part clicked, and checkbox quantity) for the atomic action of changing the state of a checkbox widget. For this action, the part clicked (box or label) had a significant effect ($F=17.511$, $p<0.001$) on user performance while CAVE location and checkbox quantity did not. There were no significant interactions between any of these three variables.

Considering that the only significant effect was caused by the part clicked, we can estimate the expected time for changing the state of a checkbox by determining what part of the widget the user will click. The expected time for clicking the box is 0.866 seconds (standard error of 0.023 seconds) while the expected time for clicking the label is 0.732 seconds (standard error of 0.023 seconds). This result is due to the fact that checkbox labels are larger in area than checkbox boxes and therefore the user can be less precise but faster.

Table 5.11 Three-factor ANOVA of Changing the State of a Checkbox

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	3.568(a)	29	.123	1.617	.027
Intercept	189.196	1	189.196	2487.123	.000
CAVE Location	.601	4	.150	1.976	.099
Part Clicked	1.332	1	1.332	17.511	**.000**
Checkbox Quantity	.383	2	.192	2.519	.082
CAVE Location * Part Clicked	.361	4	.090	1.186	.317
CAVE Location * Checkbox Quantity	.491	8	.061	.806	.598
Part Clicked * Checkbox Quantity	.130	2	.065	.857	.426
CAVE Location * Part Clicked * Checkbox Quantity	.247	8	.031	.406	.917
Error	20.311	267	.076		
Total	213.128	297			
Corrected Total	23.879	296			

(a) R Squared = .149 (Adjusted R Squared = .057)

Table 5.12 Expected Times for Changing the State of a Checkbox

Part Clicked	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
Box	.866	.023	.821	.910
Label	.732	.023	.687	.776

5.8 Command Button Experiment

For the command button experiment, we were concerned with the atomic action of clicking a command button. We evaluated three independent variables – *CAVE location*, *button size*, and *button quantity* – for significant effects on user performance.

We evaluated CAVE location for command buttons at five levels – *left wall*, *left seam*, *front wall*, *right seam*, and *right wall*. For button size, we evaluated the two levels of *small* (120 pixels x 50 pixels) and *large* (240 pixels x 100 pixels), because these should demonstrate if button size has a significant effect. We evaluated the button quantity variable, the total number of command

buttons presented to the participant, at the three levels of *1*, *3*, and *9* command buttons. Table 5.13 details the evaluation of clicking a command button.

Table 5.13 Experimental Design for Clicking a Command Button

Task ID	CAVE Location	Button Size	Button Quantity
1	Left	Small	1
2	Left Seam	Small	1
3	Front	Small	1
4	Right Seam	Small	1
5	Right	Small	1
6	Left	Small	3
7	Left Seam	Small	3
8	Front	Small	3
9	Right Seam	Small	3
10	Right	Small	3
11	Left	Small	9
12	Left Seam	Small	9
13	Front	Small	9
14	Right Seam	Small	9
15	Right	Small	9
16	Left	Large	1
17	Left Seam	Large	1
18	Front	Large	1
19	Right Seam	Large	1
20	Right	Large	1
21	Left	Large	3
22	Left Seam	Large	3
23	Front	Large	3
24	Right Seam	Large	3
25	Right	Large	3
26	Left	Large	9
27	Left Seam	Large	9
28	Front	Large	9
29	Right Seam	Large	9
30	Right	Large	9

5.8.1 Procedure

The procedure for the command button experiment consisted of one part: clicking the correct command button. Participants were told a CAVE location and a button number. After being told to start, a VEWL window appeared within the IVE at the CAVE location with a quantity of command buttons corresponding to the button quantity variable. The participant was expected to

click the command button with the corresponding button number. Tasks from Table 5.13 were randomized for each participant to avoid the effects of ordering.

5.8.2 Results

We performed a three-factor ANOVA (CAVE location, button size, and button quantity) for the atomic action of clicking a command button. For this action, the button size (standard or large) had a significant effect ($F=23.239$, $p<0.001$) on user performance while CAVE location and button quantity did not. There were no significant interactions between any of these three variables.

Table 5.14 Three-factor ANOVA of Clicking a Command Button

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	4.055(a)	29	.140	1.917	.004
Intercept	138.897	1	138.897	1904.342	.000
CAVE Location	.501	4	.125	1.719	.146
Button Size	1.695	1	1.695	23.239	**.000**
Button Quantity	.438	2	.219	3.000	.051
CAVE Location * Button Size	.241	4	.060	.825	.510
CAVE Location * Button Quantity	.409	8	.051	.700	.691
Button Size * Button Quantity	.053	2	.026	.362	.696
CAVE Location * Button Size * Button Quantity	.719	8	.090	1.232	.280
Error	19.693	270	.073		
Total	162.645	300			
Corrected Total	23.748	299			

(a) R Squared = .171 (Adjusted R Squared = .082)

Considering that the only significant effect was caused by the button size, we can estimate the expected time for clicking a command button by determining the size of the button. The expected time for clicking a small command button is 0.756 seconds (standard error of 0.022 seconds) while the expected time for clicking a large command button is 0.605 seconds (standard error of

0.022 seconds). This result is due to the fact that users can be less precise and faster with large command buttons than with small command buttons.

Table 5.15 Expected Times for Clicking a Command Button

Button Size	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
Large	.605	.022	.562	.649
Standard	.756	.022	.712	.799

5.9 Dropdown Menu Experiment

For the dropdown menu experiment, we were concerned with the three atomic actions of activating a dropdown menu, scrolling the list of a dropdown menu, and selecting an item from the list of a dropdown menu. Since scrolling the list of items and selecting an item from the list both require the dropdown menu widget to be activated, we chose to couple the evaluation of activating dropdown menus with these other two atomic actions. Therefore, we asked users to activate the menus and then either scroll the list or select a visible item from the list.

For the atomic action of scrolling a menu list, we evaluated three independent variables – *CAVE location*, *mini-widget used*, and *scroll units* for significant effects on user performance. We evaluated CAVE location for dropdown menus at five levels – *left wall*, *left seam*, *front wall*, *right seam*, and *right wall*, but due to the limited number of participants and time, we did not fully balance these levels in the experimental design. We evaluated the mini-widget used at four levels – *UpArrow*, *PageUp*, *PageDown*, and *DownArrow* – since the *CurrentPage* mini-widget does not actually scroll the dropdown menu list. We evaluated scroll units, the number of times the mini-widget needed to be used to achieve the desired position, at the three levels of *1*, *2*, and *3* units (or clicks). Table 5.16 details the evaluation of scrolling a dropdown menu.

Table 5.16 Experimental Design for Scrolling a Dropdown Menu

Task ID	CAVE Location	Mini-Widget Used	Scroll Units
1	Left	UpArrow	1
2	Front	PageUp	1
3	Right Seam	DownArrow	1
4	Left Seam	PageUp	1
5	Front	UpArrow	2
6	Right	PageUp	2
7	Left	DownArrow	2
8	Front	PageUp	2
9	Right Seam	UpArrow	3
10	Left Seam	PageUp	3
11	Front	DownArrow	3
12	Right	PageUp	3

For the atomic action of selecting an item from a menu list, we evaluated three independent variables – *CAVE location*, *list quantity*, and *item position* for significant effects on user performance. We evaluated CAVE location for dropdown menus at three levels – *left seam*, *front wall*, and *right seam* due to the limited number of participants and time. We evaluated the list quantity, the number of items shown in the list at any one time, at the three levels of *3*, *6*, and *9* items. We evaluated the item position, the relative position of the item in the shown list, at three levels – *top*, *middle*, and *bottom*. Table 5.17 details the evaluation of selecting an item from a dropdown menu.

Table 5.17 Experimental Design for Selecting an Item from a Dropdown Menu

Task ID	CAVE Location	List Quantity	Item Position
1	Front	3	Top
2	Front	3	Middle
3	Front	3	Bottom
4	Front	6	Top
5	Front	6	Middle
6	Front	6	Bottom
7	Front	9	Top
8	Front	9	Middle
9	Front	9	Bottom
10	Right Seam	3	Top
11	Left Seam	3	Middle
12	Right Seam	3	Bottom
13	Left Seam	6	Top
14	Right Seam	6	Middle
15	Left Seam	6	Bottom
16	Right Seam	9	Top
17	Left Seam	9	Middle
18	Right Seam	9	Bottom

5.9.1 Procedure

The procedure for the dropdown menu widget consisted of two parts: scrolling the menu and selecting an item from the menu. For scrolling the menu, participants were told a CAVE location, a mini-widget, and a list item number. After a verbal directive to start, a VEWL window appeared within the IVE at the CAVE location and contained a single dropdown menu. The participant was expected to activate the dropdown menu, and then click the appropriate mini-widget until the list item number was positioned at the top of the list, which would correspond to the scroll unit variable. Tasks from Table 5.16 were randomized for each participant to avoid the effects of ordering.

For selecting an item from the menu, users were told a CAVE location and a list item number. After a verbal directive to start, a VEWL window appeared within the IVE at the CAVE location and contained a single dropdown menu. The participant was expected to activate the dropdown menu, and then select the appropriate list item corresponding to the list item number. Tasks from Table 5.17 were randomized for each participant to avoid the effects of ordering.

5.9.2 Results

We performed a single-factor ANOVA (CAVE location) for the atomic action of activating a dropdown menu since scrolling the menu or selecting an item from it should not have any effect upon the activation of the menu itself. We determined that CAVE location does not have a significant effect on user performance for activating a dropdown menu. Therefore, regardless of CAVE location, the mean time for activating a dropdown menu is 0.635 seconds (standard error of 0.017 seconds).

Table 5.18 One-factor ANOVA of Activating a Dropdown Menu

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	.388(a)	4	.097	1.865	.117
Intercept	71.316	1	71.316	1372.192	.000
CAVE Location	.388	4	.097	1.865	.117
Error	15.228	293	.052		
Total	127.823	298			
Corrected Total	15.616	297			

(a) R Squared = .025 (Adjusted R Squared = .012)

Table 5.19 Expected Times for Activating a Dropdown Menu

Mean	Std. Error	95% Confidence Interval	
		Lower Bound	Upper Bound
.635	.017	.602	.669

For the atomic action of scrolling a dropdown menu, we performed a three-factor ANOVA (CAVE location, mini-widget used, and scroll units). For this action, the mini-widget used had a significant effect ($F=3.533$, $p=0.018$) on user performance as well as the scroll units required ($F=17.702$, $p<0.001$). CAVE location did not have a significant effect on user performance. There were no significant interactions between any of these three variables.

Considering that both the mini-widget used and the scroll units required have significant effects on user performance for scrolling a dropdown menu, we can estimate how long it will take a user to scroll a dropdown menu based on the mini-widget chosen and the scroll units required to reach

the desired position with that widget. Table 5.21 provides the expected times for using the mini-widgets based on scroll units of 1, 2, and 3.

Table 5.20 Three-factor ANOVA of Scrolling a Dropdown Menu

Source	Type I Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	10.117(a)	11	.920	4.651	.000
Intercept	123.911	1	123.911	626.649	.000
CAVE Location	.285	4	.071	.361	.836
Mini-Widget Used	2.096	3	.699	3.533	**.018**
Scroll Unit	7.001	2	3.500	17.702	**.000**
CAVE Location * Mini-Widget Used	.735	2	.368	1.860	.161
CAVE Location * Scroll Unit	.000	0	.	.	.
Mini-Widget Used * Scroll Unit	.000	0	.	.	.
CAVE Location * Mini-Widget Used * Scroll Unit	.000	0	.	.	.
Error	19.971	101	.198		
Total	154.000	113			
Corrected Total	30.088	112			

(a) R Squared = .336 (Adjusted R Squared = .264)

Table 5.21 Expected Times for Scrolling a Dropdown Menu

Mini-Widget Used	Scroll Unit	Mean	Std. Error	95% Confidence Interval	
				Lower Bound	Upper Bound
DownArrow	1	.727(a)	.141	.448	1.006
	2	1.423(a)	.148	1.129	1.717
	3	1.393(a)	.141	1.114	1.672
PageDown	1	1.002(a)	.141	.723	1.281
	2	1.083(a)	.141	.804	1.362
	3	1.459(a)	.157	1.147	1.771
UpArrow	1	.737(a)	.148	.443	1.031
	2	.768(a)	.141	.489	1.047
	3	1.432(a)	.148	1.138	1.726
PageUp	1	.681(a)	.141	.402	.960
	2	.771(a)	.148	.477	1.065
	3	1.206(a)	.148	.912	1.500

(a) Based on modified population marginal mean.

For the atomic action of selecting an item from a dropdown menu, we also performed a three-factor ANOVA (CAVE location, list quantity, and item position). For this action, the list quantity had a significant effect ($F=6.225$, $p=0.003$) on user performance as well as item position did ($F=11.719$, $p<0.001$). CAVE location did not have a significant effect on user performance. There were no significant interactions between any of these three variables.

Table 5.22 Three-factor ANOVA of Selecting an Item from a Dropdown Menu

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	3.309(a)	17	.195	3.155	.000
Intercept	43.331	1	43.331	702.477	.000
CAVE Location	.095	2	.048	.774	.463
List Quantity	.768	2	.384	6.225	**.003**
Item Position	1.446	2	.723	11.719	**.000**
CAVE Location * List Quantity	.052	2	.026	.424	.655
CAVE Location * Item Position	.170	2	.085	1.381	.254
List Quantity * Item Position	.401	4	.100	1.624	.171
CAVE Location * List Quantity * Item Position	.001	1	.001	.021	.886
Error	9.623	156	.062		
Total	59.919	174			
Corrected Total	12.931	173			

(a) R Squared = .256 (Adjusted R Squared = .175)

Table 5.23 Expected Times for Selecting an Item from a Dropdown Menu

List Quantity	Item Position	Mean	Std. Error	95% Confidence Interval	
				Lower Bound	Upper Bound
3	Bottom	.533(a)	.056	.423	.643
	Middle	.372(a)	.059	.256	.488
	Top	.335(a)	.056	.225	.445
6	Bottom	.591(a)	.057	.478	.704
	Middle	.570(a)	.057	.457	.683
	Top	.512(a)	.056	.402	.621
9	Bottom	.762(a)	.056	.652	.871
	Middle	.600(a)	.059	.484	.716
	Top	.394(a)	.056	.284	.503

(a) Based on modified population marginal mean.

Considering that both the list quantity and the item position have significant effects on user performance for selecting an item from a dropdown menu, we can estimate how long it will take a user to select an item from a dropdown menu based on the size of the menu and the position of the item. Table 5.23 provides the expected times for selecting an item based on the list quantity (3, 6, or 9) and the position of the item (top, middle, or bottom).

5.10 Scrolling List Experiment

For the scrolling list experiment, we were concerned with the two atomic actions of scrolling the list and selecting an item from the list. For scrolling the list, we evaluated three independent variables – *CAVE location*, *mini-widget used*, and *scroll units* for significant effects on user performance. We evaluated CAVE location for scrolling lists at five levels – *left wall*, *left seam*, *front wall*, *right seam*, and *right wall*, but due to the limited number of participants and time, we did not fully balance these levels in the experimental design. We evaluated the mini-widget used at four levels – *UpArrow*, *PageUp*, *PageDown*, and *DownArrow* – since the *CurrentPage* mini-widget does not actually scroll the scrolling list. We evaluated scroll units, the number of times the mini-widget needed to be used to achieve the desired position, at the three levels of *1*, *2*, and *3* units (or clicks). Table 5.24 details the evaluation of scrolling a scrolling list.

Table 5.24 Experimental Design for Scrolling a Scrolling List

Task ID	CAVE Location	Mini-Widget Used	Scroll Units
1	Left	UpArrow	1
2	Front	PageUp	1
3	Right Seam	DownArrow	1
4	Left Seam	PageUp	1
5	Front	UpArrow	2
6	Right	PageUp	2
7	Left	DownArrow	2
8	Front	PageUp	2
9	Right Seam	UpArrow	3
10	Left Seam	PageUp	3
11	Front	DownArrow	3
12	Right	PageUp	3

For the atomic action of selecting an item from a scrolling list, we evaluated three independent variables – *CAVE location*, *list quantity*, and *item position* for significant effects on user

performance. We evaluated CAVE location for scrolling lists at three levels – *left seam*, *front wall*, and *right seam* due to the limited number of participants and time. We evaluated the list quantity, the number of items shown in the list at any one time, at the three levels of *3*, *6*, and *9* items. We evaluated the item position, the relative position of the item in the shown list, at three levels – *top*, *middle*, and *bottom*. Table 5.25 details the evaluation of selecting an item from a scrolling list.

Table 5.25 Experimental Design for Selecting an Item from a Scrolling List

Task ID	CAVE Location	List Quantity	Item Position
1	Front	3	Top
2	Front	3	Middle
3	Front	3	Bottom
4	Front	6	Top
5	Front	6	Middle
6	Front	6	Bottom
7	Front	9	Top
8	Front	9	Middle
9	Front	9	Bottom
10	Right Seam	3	Top
11	Left Seam	3	Middle
12	Right Seam	3	Bottom
13	Left Seam	6	Top
14	Right Seam	6	Middle
15	Left Seam	6	Bottom
16	Right Seam	9	Top
17	Left Seam	9	Middle
18	Right Seam	9	Bottom

5.10.1 Procedure

The procedure for the scrolling list experiment consisted of two parts: scrolling the list and selecting an item from the list. For scrolling the list, participants were told a CAVE location, a mini-widget, and a list item number. After a verbal directive to start, a VEWL window appeared within the IVE at the CAVE location and contained a single scrolling list. The participant was expected to click the appropriate mini-widget until the list item number was positioned at the top of the list, which would correspond to the scroll unit variable. Tasks from Table 5.24 were randomized for each participant to avoid the effects of ordering.

For selecting an item from the scrolling list, participants were told a CAVE location with a list item number. After a verbal directive to start, a VEWL window appeared within the IVE at the CAVE location and contained a single scrolling list. The participant was expected to select the appropriate list item corresponding to the list item number. Tasks from Table 5.25 were randomized for each participant to avoid the effects of ordering.

5.10.2 Results

For the atomic action of scrolling a scrolling list, we performed a three-factor ANOVA (CAVE location, mini-widget used, and scroll units). For this action, the scroll units had a significant effect on user performance ($F=12.540$, $p<0.001$) while CAVE location and the mini-widget used did not. There were no significant interactions between any of these three variables.

Table 5.26 Three-factor ANOVA of Scrolling a Scrolling List

Source	Type I Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	8.805(a)	11	.800	2.733	.004
Intercept	229.600	1	229.600	783.913	.000
CAVE Location	.504	4	.126	.431	.786
Mini-Widget Used	.578	3	.193	.658	.580
Scroll Unit	7.345	2	3.673	12.540	**.000**
CAVE Location * Mini-Widget Used	.000	0	.	.	.
CAVE Location * Scroll Unit	.000	0	.	.	.
Mini-Widget Used * Scroll Unit	.376	2	.188	.643	.528
CAVE Location * Mini-Widget Used * Scroll Unit	.000	0	.	.	.
Error	30.753	105	.293		
Total	269.158	117			
Corrected Total	39.558	116			

(a) R Squared = .223 (Adjusted R Squared = .141)

Considering that the only significant effect was caused by scroll units, we can estimate how long it will take a user to scroll a scrolling list based on the number of scroll units required to reach the desired list position. The expected time for one scroll unit is 1.092 seconds (standard error of

0.086 seconds), 1.432 seconds (standard error of 0.087 seconds) for two scroll units, and 1.698 seconds (standard error of 0.088 seconds) for three scroll units.

Table 5.27 Expected Times for Scrolling a Scrolling List

Scroll Unit	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
1	1.092(a)	.086	.922	1.261
2	1.432(a)	.087	1.260	1.604
3	1.698(a)	.088	1.523	1.873

(a) Based on modified population marginal mean.

For the atomic action of selecting an item from a scrolling list, we performed a three-factor ANOVA (CAVE location, list quantity, and item position). For this action, the CAVE location, list quantity, and item position had no significant effects on user performance. In addition, there were no significant interactions between any of these three variables. Therefore, regardless of these independent variables, the expected time required to select an item from a scrolling list is 0.848 seconds (standard error of 0.024 seconds).

Table 5.28 Three-factor ANOVA of Selecting an Item from a Scrolling List

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	1.426(a)	17	.084	.825	.662
Intercept	115.118	1	115.118	1132.394	.000
CAVE Location	.029	2	.014	.141	.868
List Quantity	.337	2	.168	1.656	.194
Item Position	.541	2	.270	2.659	.073
CAVE Location * List Quantity	.101	2	.051	.498	.609
CAVE Location * Item Position	.026	2	.013	.130	.878
List Quantity * Item Position	.329	4	.082	.810	.521
CAVE Location * List Quantity * Item Position	.016	1	.016	.159	.690
Error	15.757	155	.102		
Total	142.618	173			
Corrected Total	17.183	172			

(a) R Squared = .083 (Adjusted R Squared = -.018)

Table 5.29 Expected Time for Selecting an Item from a Scrolling List

Mean	Std. Error	95% Confidence Interval	
		Lower Bound	Upper Bound
.848(a)	.024	.800	.896

(a) Based on modified population marginal mean.

5.11 Popup Menu Experiment

For this experiment, we were concerned with the two atomic actions of selecting menu items from the root level of a popup menu and from a secondary level of a popup menu. For selecting root-level popup menu items, we evaluated three independent variables – *CAVE location*, *menu quantity*, and *item position* – for significant effects on user performance. We evaluated CAVE location at three levels – *left seam*, *front wall*, and *right seam* – due to a limited number of participants and time. For the menu quantity variable, the number of menu items presented in the popup menu, we evaluated the three levels of *3*, *6*, and *9* menu items. We evaluated the item position, the relative position of the menu item in the popup menu, at three levels – *top*, *middle*, and *bottom*. Table 5.30 details the evaluation of selecting a root-level popup menu item.

Table 5.30 Experimental Design for Selecting a Root-Level Popup Menu Item

Task ID	CAVE Location	Menu Quantity	Item Position
1	Front	3	Top
2	Front	3	Middle
3	Front	3	Bottom
4	Front	6	Top
5	Front	6	Middle
6	Front	6	Bottom
7	Front	9	Top
8	Front	9	Middle
9	Front	9	Bottom
10	Right Seam	3	Top
11	Left Seam	3	Middle
12	Right Seam	3	Bottom
13	Left Seam	6	Top
14	Right Seam	6	Middle
15	Left Seam	6	Bottom
16	Right Seam	9	Top
17	Left Seam	9	Middle
18	Right Seam	9	Bottom

For the atomic action of selecting a secondary-level popup menu item, we also used three independent variables – *CAVE location*, *menu position*, and *item position* – for significant effects on user performance. For evaluating CAVE location, we used the three levels of *left seam*, *front wall*, and *right seam* due to a limited number of participants and time. For menu position, the position of the secondary popup menu within the root popup menu, we evaluated three levels – *top*, *middle*, and *bottom*. For item position, the position of the menu item within the secondary popup menu, we also evaluated the levels of *top*, *middle*, and *bottom*. Table 5.31 details the evaluation of selecting a secondary-level popup menu item.

Table 5.31 Experimental Design for Selecting a Secondary-Level Popup Menu Item

Task ID	CAVE Location	Menu Position	Item Position
1	Front	Top	Top
2	Front	Top	Middle
3	Front	Top	Bottom
4	Front	Middle	Top
5	Front	Middle	Middle
6	Front	Middle	Bottom
7	Front	Bottom	Top
8	Front	Bottom	Middle
9	Front	Bottom	Bottom
10	Right Seam	Top	Top
11	Left Seam	Top	Middle
12	Right Seam	Top	Bottom
13	Left Seam	Middle	Top
14	Right Seam	Middle	Middle
15	Left Seam	Middle	Bottom
16	Right Seam	Bottom	Top
17	Left Seam	Bottom	Middle
18	Right Seam	Bottom	Bottom

5.11.1 Procedure

The procedure for the popup menu experiment consisted of two parts: selecting a root-level popup menu item and selecting a secondary-level popup menu item. For selecting a root-level popup menu item, participants were told a CAVE location and a menu item number. After being told to begin, a red cube would appear within the IVE at the CAVE location. The participant was expected to click on this red cube, thereby invoking the popup menu at the same location. The

participant was then expected to select the menu item with the corresponding number. Tasks from Table 5.30 were randomized for each participant to avoid the effects of ordering.

For selecting a secondary-level popup menu item, participants were told a CAVE location, a root-level menu item number, and a secondary-level menu item number. After being told to start, a red cube would appear within the IVE at the CAVE location. The participant was expected to click on this red cube, thereby invoking a popup menu at the same location. The participant was then expected to click the root-level menu item corresponding to the root-level menu item number, which activated a secondary-level popup menu. The participant then was expected to select the appropriate secondary-level menu item from the new popup menu. Task from Table 5.31 were randomized for each participant to avoid the effects of ordering.

5.11.2 Results

For the atomic action of selecting a root-level popup menu item, we performed a three-factor ANOVA (CAVE location, menu quantity, and item position). For this action, CAVE location, menu quantity, and item position had no significant effects on user performance. There were no significant interactions between any of these variables as well. Therefore, regardless of these independent variables, the expected time for selecting a root-level popup menu item is 0.725 seconds (standard error of 0.029 seconds).

For the atomic action of selecting a secondary-level popup menu item, we also performed a three-factor ANOVA (CAVE location, menu position, and item position). For this action, the item position had a significant effect ($F=4.640$, $p=0.011$) on user performance while CAVE location and menu position did not have significant effects. There were no significant interactions between any of these three variables.

Table 5.32 Three-factor ANOVA of Selecting a Root-Level Popup Menu Item

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	1.591(a)	17	.094	.640	.855
Intercept	83.185	1	83.185	569.177	.000
CAVE Location	.005	2	.003	.018	.982
Menu Quantity	.245	2	.122	.838	.435
Item Position	.025	2	.013	.086	.918
CAVE Location * Menu Quantity	.027	2	.014	.093	.911
CAVE Location * Item Position	.055	2	.028	.189	.828
Menu Quantity * Item Position	.820	4	.205	1.402	.236
CAVE Location * Menu Quantity * Item Position	.095	1	.095	.651	.421
Error	22.945	157	.146		
Total	116.586	175			
Corrected Total	24.537	174			

(a) R Squared = .065 (Adjusted R Squared = -.036)

Table 5.33 Expected Times for Selecting a Root-Level Popup Menu Item

Mean	Std. Error	95% Confidence Interval	
		Lower Bound	Upper Bound
.725(a)	.029	.668	.782

(a) Based on modified population marginal mean.

Considering that the only significant effect was caused by the position of the menu item within the secondary-level popup menu, we can estimate how long it will take a user to select a secondary-level popup menu item based on the position of that item within the secondary-level popup menu (top, middle, or bottom). Table 5.35 shows the expected times based on these positions.

Table 5.34 Three-factor ANOVA of Selecting a Secondary-Level Popup Menu Item

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	4.275(a)	17	.251	1.250	.234
Intercept	232.638	1	232.638	1156.053	.000
CAVE Location	.828	2	.414	2.058	.131
Menu Position	.453	2	.226	1.125	.327
Item Position	1.867	2	.934	4.640	**.011**
CAVE Location * Menu Position	.036	2	.018	.090	.914
CAVE Location * Item Position	.047	2	.023	.116	.890
Menu Position * Item Position	.101	4	.025	.125	.973
CAVE Location * Menu Position * Item Position	.129	1	.129	.639	.425
Error	29.179	145	.201		
Total	285.200	163			
Corrected Total	33.454	162			

(a) R Squared = .128 (Adjusted R Squared = .026)

Table 5.35 Expected Times for Selecting a Secondary-Level Popup Menu Item

Item Position	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
Bottom	1.393(a)	.063	1.268	1.518
Middle	1.223(a)	.059	1.105	1.340
Top	1.127(a)	.061	1.007	1.247

(a) Based on modified population marginal mean.

5.12 Summary

Reviewing the results of each of the six experiments, we have determined the following facts regarding completing atomic actions in the VEWL SCI.

- The expected time for focusing a window is 0.440 seconds.
- The expected time for moving a window is significantly affected by the number of walls spanned (see Table 5.7).
- The expected time for closing a window is 0.591 seconds.

- The expected time for changing the state of a checkbox is significantly affected by the part clicked (box or label) to change the state (see Table 5.12).
- The expected time for clicking a command button is significantly affected by the size of the button (see Table 5.15).
- The expected time for activating a dropdown menu is 0.635 seconds.
- The expected time for scrolling a dropdown menu is significantly affected by both the mini-widget used to scroll the menu and the scroll units required to reach the desired position within the menu (see Table 5.21).
- The expected time for selecting an item from a dropdown menu is significantly affected by both the number of items shown in the menu and the position of the item within the menu (see Table 5.23).
- The expected time for scrolling a scrolling list is significantly affected by only the scroll units required to reach the desired position within the menu (see Table 5.27). The reason the mini-widget used significantly affects dropdown menus and not scrolling lists is because a user must identify the mini-widget after activating the dropdown menu but can identify the mini-widget while moving the virtual mouse toward the scrolling list, hence saving time.
- The expected time for selecting an item from a scrolling list is 0.848 seconds. The reason the number of items shown and the position of the item significantly affect dropdown menus and not scrolling lists is because users must identify everything after activating the dropdown menu but can identify things while moving the virtual mouse toward the scrolling list, hence saving time.
- The expected time for selecting a root-level popup menu item is 0.725 seconds.
- The expected time for selecting a secondary-level popup menu item is significantly affected by the position of the menu item within the secondary-level popup menu (see Table 5.35).

With user performance data collected on completing atomic actions in the VEWL SCI, we were ready to create our second study for collecting data on completing entire tasks in an IVE application, in order to be able to estimate $\mathbf{T_{Cognitive}}$ and evaluate the effects of task sequences on user performances.

Chapter 6 Task Sequence Performance Study

6.1 Goals

With the establishment of the estimates of $\mathbf{T_{Perceptual\text{-}Motor}}$ for the VEWL SCI (see Chapter 5), we needed to establish $\mathbf{T_{Total}}$ and $\mathbf{T_{Perceptual\text{-}Motor}}$ for a specific testbed of tasks, in order to estimate $\mathbf{T_{Cognitive}}$. We were interested if task sequences have a significant effect on the cognitive effort induced when completing tasks in IVEs and if the foci of task sequences (object- or location-oriented) have a significant effect on the cognitive effort of the user as well. We hypothesized that task sequences beginning with the indication of objects, such as the Object-Action task sequence, would induce less cognitive effort than task sequences beginning with the indication of actions, such as the Action-Object task sequence. We also predicted that the foci of task sequences would have a significant effect on user performance and that object-oriented task sequences would be less cognitively demanding than location-oriented tasks.

Our first goal for the task sequence performance study was to establish a testbed of tasks to evaluate the various task sequences and foci. We chose an interior design scenario for this testbed of tasks. This scenario provided enough rich tasks to support the various task sequences that we had defined in Chapter 3 (see Table 6.1).

Table 6.1 Interior Design Scenario – Support of Task Sequences

Task	Level of Task Sequence	Focus of Task Sequence
Remove	Simple	Object-oriented
Scale Down	Simple	Object-oriented
Copy	Simple	Object-oriented
Paste	Simple	Location-oriented
Place a Painting	Simple	Location-oriented
Place a Rug	Simple	Location-oriented
Repattern	Complex and Combination	Object-oriented
Rotate	Complex and Combination	Object-oriented
Showcase	Complex and Combination	Object-oriented
Paint	Complex	Location-oriented
Place Furniture	Complex	Location-oriented
Hang an Object	Complex	Location-oriented

For simple task sequences, the interior design scenario included three object-oriented tasks (remove, scale down, and copy) and three location-oriented tasks (paste, place a painting, and place a rug). The remove task involved removing a selected object from the IVE. For the scale down task, a selected object was scaled to fifty percent of its current size. The copy task involved storing a copy of a selected object in memory. For the paste task, a copy of an object in memory was placed at a selected location. The place a painting task involved placing a generic painting object at a selected location on the floor. Similarly, the place a rug task involved placing a generic rug object at a selected location on a wall.

For complex and combination task sequences (those involving parameters), the interior design scenario included three object-oriented tasks (repattern, rotate, and showcase) and three location-oriented tasks (paint, place furniture, and hang an object). The repattern task involved changing the pattern of a selected chair or sofa to a blue, pink, or striped material. For the rotate task, a selected piece of furniture was rotated to the left by 30^{o}, 45^{o}, or 90^{o}. The showcase task involved changing the contents of a selected bookcase to books, plates, or trophies. For the paint task, a selected location would be painted red, green, or blue. The place furniture task involved placing a chair, sofa, or bookcase at a selected location on the floor. The hang an object task involved hanging a dartboard, painting, or poster at a selected location on a wall.

With a testbed of tasks establish with the interior design scenario, our remaining goals were to establish estimates of $\mathbf{T_{Percpetual\text{-}Motor}}$ for selecting the various objects in the interior design scenario and to establish $\mathbf{T_{Total}}$ for the tasks provided by the testbed, in order to estimate $\mathbf{T_{Cognitive}}$ for each task.

6.2 Implementation

For the tasks sequence performance study, we used a four-screen CAVE™ [Cruz-Neira *et al.* 1993] in our experiments. The CAVE has three sides (10’ x 9’) and a floor (10’ x 10’) and uses projection technology, stereoscopic displays, and head tracking to create an IVE. The CAVE provides a 270-degree horizontal field of regard (FOR) and a 100-degree horizontal field of view (FOV) when viewed through active stereo glasses. Since the floor project of the CAVE was

unavailable at the time of our the VEWL performance study, we decided to not use the floor of the CAVE to maintain a constant setup and reduce confounds in our performance studies.

In addition to the CAVE, we used an Intersense IS-900 tracking system with a 6-DOF head tracker and a 6-DOF wand device with four buttons and a 2-DOF joystick. We used DIVERSE [Kelso *et al.* 2002] and VEWL [Larimer and Bowman 2003] to create the software for the study.

6.3 Experimental Design and Procedure

In order to establish the estimates of $\mathbf{T_{Percpetual\text{-}Motor}}$ for selecting the various objects in the interior design scenario and to establish $\mathbf{T_{Total}}$ for the tasks provided by the testbed, we had participants to complete two separate experiments – a series of selection trials and a series of interior design tasks.

For the series of selection trials, we created an empty IVE that objects from the interior design scenario would appear in random CAVE locations (left wall, left seam, front wall, right seam, and right wall). Only one object would appear at a time and once that object was selected another random object would appear. Using ray-casting, participants were expected to select objects as quickly and efficiently as possible. The software for the IVE would record the amount of time between selections and the type of object selected. Once 100 selections were made, the series of selection trials would be finished.

For the series of interior design tasks, we created nine separate IVEs for the nine different task sequences that we wanted to evaluate. Table 6.2 details the design of the SCI for each of these task sequence IVEs. Based on these designs, the only data from the VEWL performance study necessary is the expected times for clicking a command button from a starting position, clicking a command button on another VEWL window, selecting a root-level popup menu item from a starting position, and selecting a root-level popup menu item from another VEWL popup menu.

Table 6.2 Design of SCI for Each Task Sequence IVE

Task Sequence IVE	Design of SCI
Action-Object	Click command button on window (Action), select object using ray-casting (Object)
Object-Action	Select object using ray-casting (Object), select root-level popup menu item (Action)
Action-Object-Parameter	Click command button on window (Action), select object using ray-casting (Object), select root-level popup menu item (Parameter)
Action-Parameter-Object	Click command button on window (Action), click command button on another window (Parameter), select object using ray-casting (Object)
Object-Action-Parameter	Select object using ray-casting (Object), select root-level popup menu item (Action), select root-level popup menu item from another popup menu (Parameter)
Action+Object-Parameter	Select 3D widget representing object and action using ray-casting (Action+Object), select root-level popup menu item (Parameter)
Action-Object+Parameter	Click command button on window (Action), select 3D widget representing object and parameter using ray-casting (Object+Parameter)
Action+Parameter-Object	Click command button on window (Action+Parameter), select object using ray-casting (Object)
Object-Action+Parameter	Select object using ray-casting (Object), select 3D widget representing action and parameter using ray-casting (Action+Parameter)

Since button size was the only independent variable to have a significant effect on clicking command buttons and there were no independent variables that had a significant effect on selecting a root-level popup menu item, we already had the expected times for clicking a command button and selecting a root-level popup menu from a starting position. Unfortunately, for the VEWL performance study, we had not accounted for atomic actions not from a starting position and, therefore, did not have expected times for clicking a command button and selecting a root-level popup menu item from another window or popup menu, respectively. To account for these, we looked at the dropdown menu experiment and used the expected time required to select an item, after activation, from a dropdown menu with a list quantity of 3 items (since all of our interior design tasks provide for 3 parameters and list quantity had a significant effect on user performance). Table 6.3 details the VEWL performance results necessary for calculating $\mathbf{T_{Percerptual\text{-}Motor}}$ for all nine task sequence IVEs.

Table 6.3 Necessary VEWL Performance Results

Atomic VEWL Task	Mean	Std Dev.
Clicking a command button (from starting position)	0.7556	0.32286
Clicking a command button (from another window)	0.4133	0.33104
Selecting a root-level popup menu item (from starting position)	0.7253	0.37552
Selecting a root-level popup menu item (from another popup menu)	0.4133	0.33104

For the IVEs corresponding to simple and complex task sequences, there were twelve total interior design tasks (six object-oriented and six location-oriented) for participants to complete. For the combination task sequences, there were six object-oriented, interior design tasks for participants to complete. Before each task, the participants were asked to stand in the center of the CAVE floor with their arms down by their sides, looking forward at the front CAVE screen as the participants did for the VEWL performance study. The interior design tasks were read aloud to the participants and a verbal "GO!" signaled participants to begin each task. The time from the "GO!" signal to the completion of the task was measured and recorded for each task by the software for the IVEs.

Participant sessions began with the administration of a background survey (Appendix C) to collect information about the participants. After completing the background survey, participants were trained on selecting objects using ray-casting and then completed the series of selection trials. After the series of selection trials, participants were trained on using the task sequence IVEs and completed each set of interior design tasks per IVE. Both independent variables (task sequence and task focus) were varied within-subjects. Additionally, the ordering of the task sequence IVEs was counterbalanced between-subjects to avoid any learning effects. After the series of interior design tasks, an exit survey (Appendix C) was administered to collect participants' thoughts about the most intuitive task sequences and any task sequences that were difficult to understand or execute.

6.4 Participants

For the task sequence performance study, we recruited 24 unpaid participants (17 male, 7 female) through various Virginia Tech listservs and classes. The age range of the participants was 19 to 29 years old with 22 being the average age. All of the participants used computers daily. Of the 24 participants, 10 had corrected vision (glasses or contacts). All of the participants used their right hands for the experiments. Eleven of the 24 participants had had some sort of previous virtual reality experience, ranging from a limited HMD demo to other studies in the CAVE.

6.5 Results

From the series of selection trials, we averaged the data collected from all participants on the time required to make selections based on the object being selected. Table 6.4 details these object selection results.

Table 6.4 Results from Series of Selection Trials

Object Selected	Mean	Std Dev.
3D Widgets (Large)	1.5011	0.18246
3D Widgets (Small)	2.3367	0.34726
Bookcase	1.2697	0.20435
Chair	1.4784	0.19324
Dartboard	1.7623	0.24836
Floor	1.2614	0.36039
Painting	1.5315	0.19455
Poster	1.5153	0.19612
Rug	1.5607	0.19062
Sofa	1.3337	0.1932
Wall	1.4957	0.57139

Using our task sequence performance model (Equation 4.2), the necessary VEWL performance results (Table 6.3), and the results from the series of selection trials (Table 6.4), we were able to estimate the cognitive time associated with each of the task sequences that we had identified. For example, consider the Action-Parameter-Object task sequence for repatterning a chair with blue material. According to Table 6.2, the actions for this task require clicking a command button (0.7556 seconds), clicking a command button on another window (0.4133 seconds), and selecting the chair (1.478 seconds). We would therefore calculate $\mathbf{T_{Perceptual\text{-}Motor}}$ to be 2.647 seconds. If the repattern task took the user 6.824 seconds to complete ($\mathbf{T_{Total}}$), then we estimate the cognitive effort ($\mathbf{T_{Cognitive}}$) to be 4.177 seconds. (NOTE: Due to participants being informed where to expect widgets to appear during the VEWL performance study, the estimates for activating VEWL widgets are smaller than the estimates for selecting objects. This accounts for users knowing where to expect the SCI and for users not knowing which objects the task will involve.)

6.5.1 Objective Results

We performed two-factor ANOVAs (task sequence and focus) for the simple task sequences and complex task sequences. We performed single-factor ANOVAs (task sequence) for combination task sequences (since combination task sequences did not support location-oriented tasks).

For simple task sequences, the task sequence had a significant effect ($F=68.957$, $p<0.0001$) and the focus also had a significant effect ($F=16.325$, $p<0.0001$). There was not a significant interaction between these two variables. The mean cognitive time was 1.153 seconds for the Object-Action task sequence and 2.909 seconds for the Action-Object task sequence. The mean cognitive time was 1.604 seconds for an object-oriented task sequence and 2.458 seconds for a location-oriented task sequence. Figure 6.1 shows the mean times for each condition.

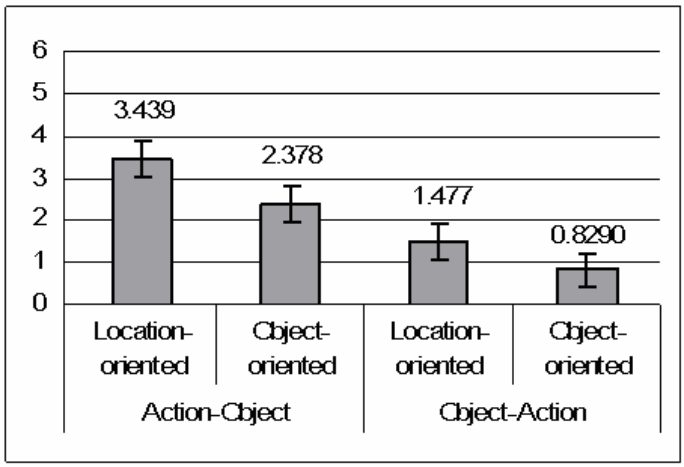


Figure 6.1 Estimated Means of Cognitive Time (in seconds) for Simple Task Sequences with Standard Error Bars

For complex task sequences (Figure 6.2), the task sequence had a significant effect ($F=4.536$, $p=0.011$) and the focus also had a significant effect ($F=32.767$, $p<0.0001$). Again, there was not a significant interaction between these two variables. The mean cognitive time was 3.996 seconds for the Object-Action-Parameter task sequence, 4.177 seconds for the Action-Parameter-Object task sequence, and 4.640 seconds for the Action-Object-Parameter task sequence. Based on a post-hoc test, the only pair not significantly different was Object-Action-Parameter and

Action-Parameter-Object. The mean cognitive time was 3.755 seconds for an object-oriented task sequence and 4.786 seconds for a location-oriented task sequence.

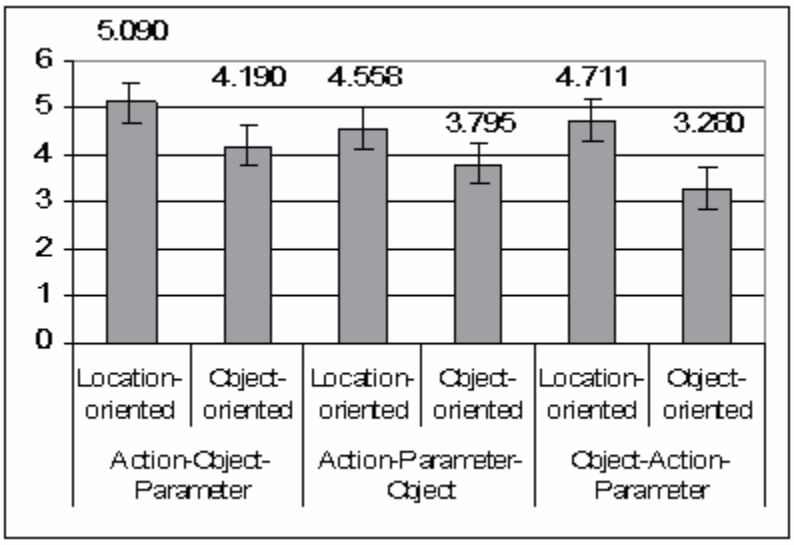


Figure 6.2 Estimated Means of Cognitive Time (in seconds) for Complex Task Sequences with Standard Error Bars

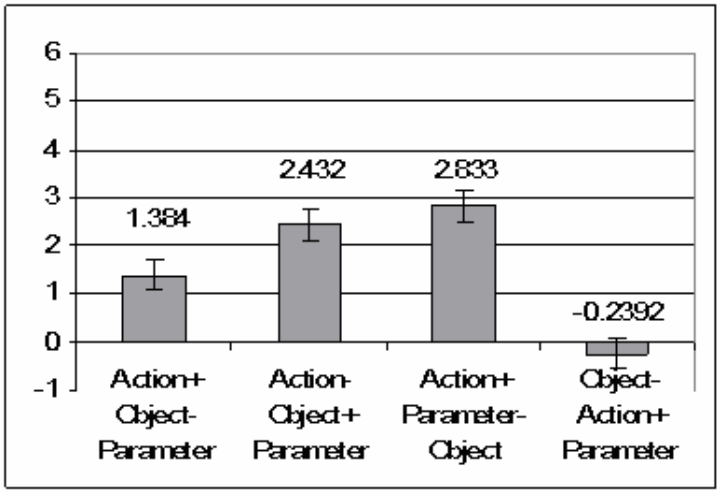


Figure 6.3 Estimated Means of Cognitive Time (in seconds) for Combination Task Sequences with Standard Error Bars

For the combination task sequences, the task sequence had a significant effect ($F=72.65$, $p<0.0001$). As shown in Figure 6.3, the mean cognitive time was -0.239 seconds for the Object-Action+Parameter task sequence (we discuss possible interpretations of this negative cognitive

time in Section 6.6), 1.384 seconds for the Action+Object-Parameter task sequence, 2.432 seconds for the Action-Object+Parameter task sequence, and 2.833 seconds for the Action+Parameter-Object task sequence. Based on a post-hoc test, the only pair not significantly different was Action-Object+Parameter and Action+Parameter-Object.

6.5.2 Subjective Results

From the exit surveys administered, we discovered that for simple tasks, 19 participants (79.2 percent) found the Object-Action task sequence more intuitive than the Action-Object task sequence. Similarly, for complex tasks, 15 participants (62.5 percent) found the Object-Action-Parameter task sequence more intuitive than the Action-Object-Parameter and Action-Parameter-Object task sequences. For combination tasks, 16 participants (66.7 percent) found the Object-Action+Parameter task sequence more intuitive than the other three combination task sequences. Figure 6.4 shows the number of participants (24 total) reporting that a particular task sequence was difficult to understand or execute.

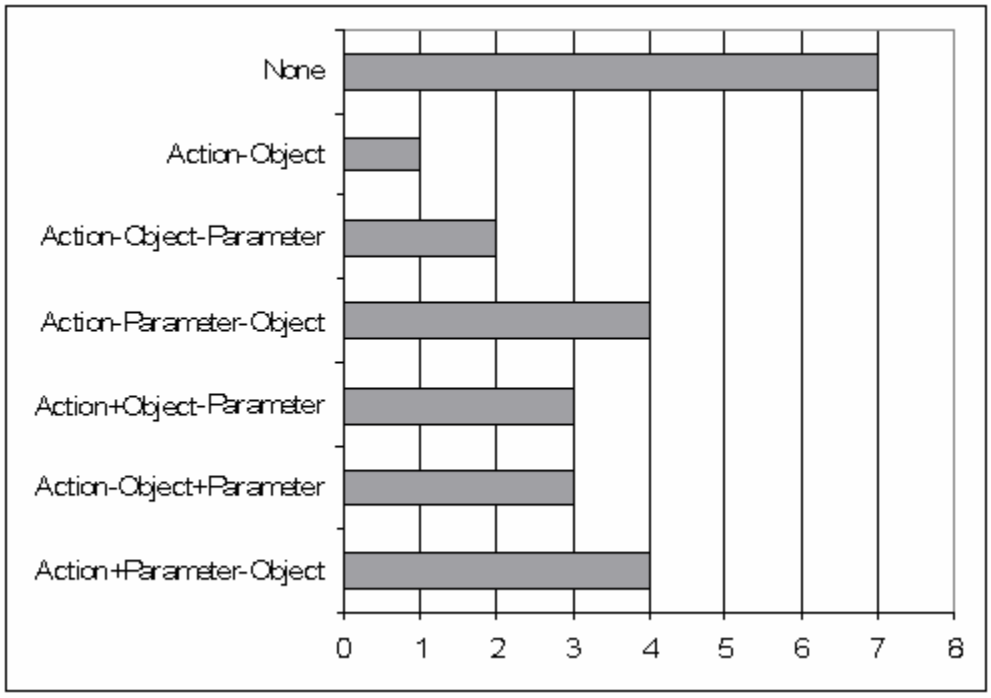


Figure 6.4 Number of participants reporting that a particular task sequence was difficult to understand or execute.

6.6 Discussion

Overall, we found that task sequence has a significant effect upon the cognitive effort experienced by users when completing a task within an IVE. Though the task sequence had a significant effect on cognitive effort, the absolute differences in time are small and the importance of the significant effect is questionable. We believe there are three considerations that make the significant effect important.

First, if an application is used for long periods of time, even saving fractions of a second per system control task can add up to valuable time saved overall [Gray *et al.* 1990]. By choosing the correct task sequence, for instance the Object-Action task sequence, when designing an IVE application, the designer can create an application that will ultimately save the user valuable time in operation and use.

Second, system control interfaces, which do not directly support the user's goal, should not induce more cognitive effort than what the user task already induces. System control interfaces should be transparent and induce minimal cognitive effort. By using the appropriate task sequences when designing 3DUIs and SCIs, we can ensure that these interfaces are not distracting users and inducing unnecessary cognitive effort.

Third, the tasks we evaluated were not difficult (even the complex and combination tasks). The differences in time might grow larger with tasks that are more cognitively-demanding and involve user strategies. Most tasks are generated by the user and not explicitly given to them. With larger acquisition times for these tasks, the execution times will probably also increase, leading to an increased difference in the cognitive time induced by different task sequences.

Our experiment does not tell us *why* Object-first task sequences lead to better performance, but there are several possible causes. Humans live in a physical world that consists of many objects. This fact may enable users to complete tasks easier when the focus (or start of the task) is an object instead of an abstract action. Another reason for the success of these task sequences could be due to the subjects' prior experiences and familiarity with GUIs. Since GUIs are object-

oriented, experience with these interfaces may afford better performance in IVEs that use Object-first task sequences.

In addition to finding that task sequence has a significant effect on user performance, we found that the focus of the task sequence (object- and location-oriented) also has a significant effect upon the cognitive effort. Location-oriented tasks induced larger cognitive effort than object-oriented tasks. This could be because objects are visible and distinct, while locations are invisible and ambiguous. Users must visualize the results of their actions (e.g., what the sofa will look like against that wall) before selecting a location.

Another interesting discovery was that the mean cognitive times for complex task sequences were higher than those for the simple task sequences, but the mean cognitive times for combination task sequences were not any higher than those for the simple task sequences. Clearly, complex task sequences induce more cognitive effort than simple task sequences due to the addition of the parameter. For combination task sequences, we believe the combination of two of the three induced selections decreases the cognitive effort. In fact, the estimate for the Object-Action+Parameter sequence is negative! Of course, this cannot be literally true and is the result of users performing the perceptual-motor tasks faster than the mean values used to calculate the cognitive times. Even with perceptual-motor performance faster than the expected mean values, the actual cognitive times for this task sequence must have been close to zero for the estimated cognitive times to be negative. This indicates the power of this combination task sequence.

A concern with our task sequence performance model (Equation 4.2) and our task sequence performance study is the accuracy of our estimates for the cognitive effort induced by the various task sequences. There are several possible errors in our estimating cognitive effort. For instance, a participant could perform the perceptual-motor actions faster than the means that we use for these actions, but the participant is most likely performing these actions faster for all of the task sequences. Another possible error is in the participant's understanding of the verbal directives given during the experiments as participants may mishear or misunderstand the tasks given to them. We tried to eliminate this type of error by ignoring trials in which the participant

questioned or restated what the given task was. We also ignored trials in which the participant did not complete the given task. Nevertheless, we believe that the times for these possible errors are nearly equivalent for all task sequences and do not reduce the value of our results

Chapter 7 Task Sequence Preference Study

7.1 Goals

As mentioned in Section 4.4, another method for evaluating task sequences that we considered was determining which task sequences users prefer. For the task sequence preference study, we had two main goals – determining which task sequences novices prefer and how those preferences evolve as novices become experienced users. We wanted to determine which task sequences novices prefer because SCIs that utilize those task sequences should be easier for novices to learn. Additionally, if we track the evolution of task sequence preferences as novices become experienced users, we can determine the range of task sequences that SCIs should provide.

In order to achieve these goals, we designed a "longitudinal" study of five sessions in which users would have choices of which task sequences to use for completing tasks. Even though there were only five sessions, we call this a longitudinal study because most studies in virtual reality and 3D interaction are single sessions. The SCI for this study was designed to support multiple task sequences so that users would have open preferences and would not be influenced on which task sequence to use. Throughout the sessions, we were able to record the preferences of the participants, capturing data on novices and the evolution of their preferences as they became experienced users.

7.2 Implementation

For the tasks sequence preference study, we used a four-screen CAVE™ [Cruz-Neira *et al.* 1993] in our experiments. The CAVE has three sides (10' x 9') and a floor (10' x 10') and uses projection technology, stereoscopic displays, and head tracking to create an IVE. The CAVE provides a 270-degree horizontal field of regard (FOR) and a 100-degree horizontal field of view (FOV) when viewed through active stereo glasses. For this experiment, we were able to use the floor projector of the CAVE.

In addition to the CAVE, we used an Intersense IS-900 tracking system with a 6-DOF head tracker and a 6-DOF wand device with four buttons and a 2-DOF joystick. We used DIVERSE [Kelso *et al.* 2002] and VEWL [Larimer and Bowman 2003] to create the software for the study.

7.3 System Control Interface

Since we wanted to run a longitudinal study in which users would have choices of which task sequences to use for completing tasks, we had to design a SCI that would support multiple task sequences so that user would have open preferences and would not be influenced on which task sequence to use. Also, we needed to choose another testbed of tasks. For sake of consistency and ease, we decided to use VEWL to create the SCI and we decided upon another interior design scenario, similar to the one used in the task sequence performance study, for a testbed of tasks.

The interior design scenario used for the task sequence preference study consisted of nine different tasks – place, edit, move, rotate, cut, copy, paste, save, and load. Table 7.1 details how these tasks support the various task sequences and foci.

Table 7.1 Interior Design Scenario – Support of Task Sequences

Task	Level of Task Sequence	Focus of Task Sequence
Place	Complex	Location-oriented
Edit	Complex	Object-oriented
Move	Simple	Object-oriented
Rotate	Simple	Object-oriented
Cut	Simple	Object-oriented
Copy	Simple	Object-oriented
Paste	Simple	Location-oriented
Save	None	None
Load	None	None

For the place task, users could place beds, chairs, desks, paintings, posters, shelves, sofas, and stands with choices of attributes such as the pattern of the bed comforter and the contents of the shelves. For the edit task, users could edit similar attributes of any object placed in the environment, including the doors, floor, and walls of the room being decorated. The place and edit tasks supported complex and combination task sequences as both of these tasks required the choice of several parameters, such as the pattern of the bed comforter.

For the move task, users could move an object to any location and then specify to stop moving that object once a desired location was reached. Similarly, for the rotate task, users could rotate an object to any degree of heading (for floor objects) or roll (for wall objects) until the desired rotation was reached. The cut task enabled users to remove an object from the IVE and store it in memory. The copy task enabled users to copy an object to memory. For the paste task, users were able to place an object in memory at any specified location. The move, rotate, cut, copy, and paste tasks all supported simple task sequences as these tasks required only the indication of a single object or location.

For the save task, users could choose from four abstract "slots" to save the current room design to. Similarly, for the load task, users could choose from the same four abstract slots to load a room design from and replace the current room design. The save and load tasks did not support any of the task sequences that we had defined in Chapter 3, since no objects or locations were required to complete these tasks, but we still provided these tasks to create a lush interior design scenario that users would want to use.

With our interior design scenario created, we needed to create a SCI that would support all of the tasks with multiple task sequences. Due to the difficultly of implementing SCIs for combination task sequences, we decided to focus on only simple and complex task sequences. Since simple task sequences are complex task sequences without the indication of parameters, we decided to design an SCI that would support the complex task sequences we evaluated before – the Action-Object-Parameter, Action-Parameter-Object, and Object-Action-Parameter task sequences.

For our design of the SCI, we decided to utilize two VEWL windows and several popup menus (Figure 7.1). The first VEWL window was designed for choosing one of the nine tasks or actions. We refer to this as the action window. The second VEWL window was designed for choosing parameters for the place and edit tasks. We refer to this as the parameter window. We also designed similar popup menus that supported choosing actions and parameters. Now consider the example location-oriented task of a user placing a bed within the IVE.

If the user wants to use the Action-Object-Parameter task sequence for placing a bed, the user would first click on a command button labeled “Place” on the action window, invoking command buttons regarding types of objects to appear on the parameter window. Disregarding the parameter window, the user uses ray-casting to select the location to place the bed. A popup menu appears in the specified location with types of objects to place. The user selects the “Bed” menu item and continues through a series of popup menus for choosing the bed’s size, style, wood color, and comforter pattern. After choosing the final parameter, the comforter pattern, the desired bed appears in the specified location, the popup menus disappear, and the parameter window returns to normal.

Figure 7.1 The SCI included the action and parameter windows and several popup menus.

If the user wants to use the Action-Parameter-Object task sequence for placing a bed, the user clicks on the “Place” command button on the action window. This time, the user clicks the “Bed” command button that appeared on the parameter window and proceeds through a series of different command buttons on the parameter window for choosing the parameters of the bed. After clicking on the final command button for the comforter pattern, the parameter window alerts the user to select a location. Using ray-casting, the user selects the desired location for the bed and the desired bed appears and the parameter window returns to normal.

If the user wants to use the Object-Action-Parameter task sequence for placing a bed, the user begins by disregarding the action window and selecting the desired location using ray-casting. A popup menu appears in the specified location with the actions to choose from. The user selects the "Place" menu item and another popup menu appears with the types of objects to place. The user selects the "Bed" menu item and continues through a series of popup menus for choosing the parameters of the bed. After choosing the final parameter, the comforter pattern, the desired bed appears in the specified location and the popup menus disappear.

With this design, the SCI for the task sequence preference study provides users with the opportunity to use any of the simple and complex task sequences.

7.4 Experimental Design and Procedure

As mentioned in section 4.4, we designed a longitudinal study of five sessions in which users would have choices of which task sequences to use for completing tasks. We chose to use five one-hour sessions per participant to allow enough time for each participant to progress from novice to experienced user with our interior design scenario. Before the first session, we administered a background survey (Appendix D) to each participant to collect information on the participants.

For the first session, we wanted to introduce the participant to the interior design scenario and the SCI through training. For each of the nine tasks, we stepped through the possible task sequences for completing the task with the participant. Then after showing the participant how to complete all the tasks using all of the various task sequences, we asked the participant to show us how to complete all of the tasks. Once the participant successfully showed us how to complete all the tasks using all of the various task sequences, the participant was allowed to "play" with the interior design scenario for the remainder of the session. During this play time, the software used for the IVE recorded the different tasks the participant chose to complete and *which* task sequences were used.

For the second session, we wanted the participant to review the different tasks and task sequences through the process of replicating two room designs. The software for the IVE was designed to allow a "goal" room design to be loaded when the software is run. This goal room design is not normally visible, but users are able to hide the current room design and view the goal room design by holding down the upper right button on the wand device. When users release this button, the current room design returns and the goal room design disappears. This software allowed us to provide tasks for the users to complete without having the problem of biased commands that may influence which task sequences are used. The goal room designs were also designed to ensure that participants had to use certain tasks such as place, edit, move, rotate, and cut.

So for the second session, we first provided participants with an empty room and a goal room design to be replicated using the task sequences beginning with the indication of an action – Action-Object, Action-Object-Parameter, and Action-Parameter-Object. After the participant replicated the goal room design to the best of her ability, we conducted a "step" interview (Appendix D), in which we asked how the participant like using the task sequences for the previous step and why. After this step interview, we provided her with another empty room and another goal room design to be replicated using the task sequences beginning with the indication of an object or location – Object-Action and Object-Action-Parameter. Once the participant replicated this second goal room design, another step interview was conducted. Finally, she was allowed to have play time for the remainder of the session while the software recorded which tasks and task sequences she used.

For the third session, we wanted the participant to review the different tasks and task sequences one more time. We used the process of room replication again, with the participant using Object-first task sequences for the first goal room design and Action-first task sequences for the second goal room design. This was to counterbalance any effects of ordering for sessions 2 and 3. Again, once the participant replicated each goal room design, a step interview was conducted. Finally, he was allowed to have play time for the remainder of the session while the software recorded which tasks and task sequences he used.

For the fourth session, we wanted to provide another structured set of tasks for the participants to complete. We used the process of room replication one more time, but did not specify which task sequences the participants were to use. Each participant was allowed to replicate the goal room design using any task sequences he desired and the software recorded which tasks and task sequences he used. After completing this replication, the participant was given the remainder of the session as play time while the software recorded the task sequences he used.

For the fifth session, we wanted the participants to act as experienced users and create their own room designs using whichever task sequences they desired. To inspire such actions, we created a contest based on the designs derived during this fifth session for the participants. The participant that created the best room design during the fifth session would win the contest. So participants were given the entire session to create their best room design, starting from an empty room. The participants were encouraged to use whichever tasks and task sequences they desired. The software recorded these tasks and which task sequences were used. At the end of the session, we administered a final interview (Appendix D) concerning which task sequences the participant preferred and an exit survey (Appendix D) concerning the participant's self-assessment of expertise after each session.

7.5 Participants

For the task sequence preference study, we recruited 8 paid participants (4 male, 4 female) through various Virginia Tech listservs and classes. The participants were compensated $5 for every session that they began, a total of $25 for the entire study. The age range of the participants was 20 to 23 years old with 22 being the average age. All of the participants used computers daily. Of the 8 participants, 3 had corrected vision (glasses or contacts). All of the participants used their right hands for the experiments. All of the participants had had some sort of previous virtual reality experience, but no experiences with IVEs similar to the interior design scenario.

7.6 Results

With the data collected from five sessions and six recordings (remember the fourth session had two separate recordings), we began looking at task sequence preferences by participant. We examined the results by participant to observe the evolution of preferences as novices became experienced users, which could not have been done with aggregate results.

7.6.1 Participant One

By first looking at the number of times Participant One (P1) used the various task sequences we get an idea of which task sequences the participant prefers. To simplify analysis, we considered Object-Action task sequences as Object-Action-Parameter task sequences and Action-Object task sequences as Action-Parameter-Object task sequences (NOTE: We chose to count Action-Object task sequences as Action-Parameter-Object task sequences instead of Action-Object-Parameter task sequences because we found in the task sequence performance study that the separation of the action and parameter causes decreased user performance). Figure 7.2 shows the percentages that P1 used the task sequences by each session.

As seen in Figure 7.2, P1 had a very clear task sequence preference for the Object-Action-Parameter task sequence, indicating that the participant preferred starting tasks by indication the object or location first. The only task P1 completed not using the Object-Action-Parameter task sequence was a place task, during the first session, using the Action-Object-Parameter task sequence.

As a novice and an experienced user, P1 preferred the Object-Action-Parameter task sequence. The subjective responses of P1 during the four step interviews and the final interview confirm this.

Additionally, according to the exit survey, P1 felt that he/she had progressed from a moderate user in the first session to an experienced user in the final session.

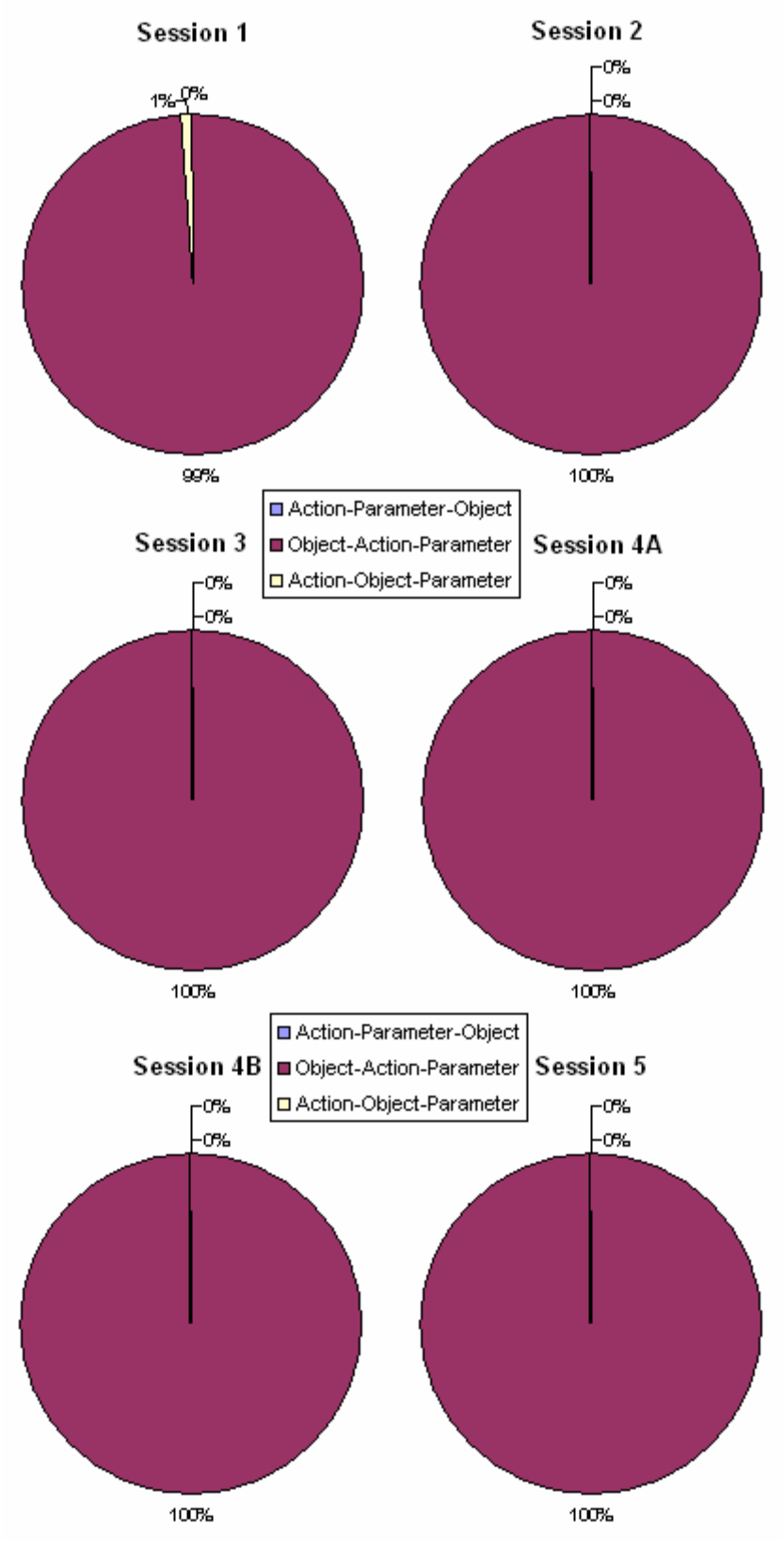


Figure 7.2 Percentages P1 used task sequences by session.

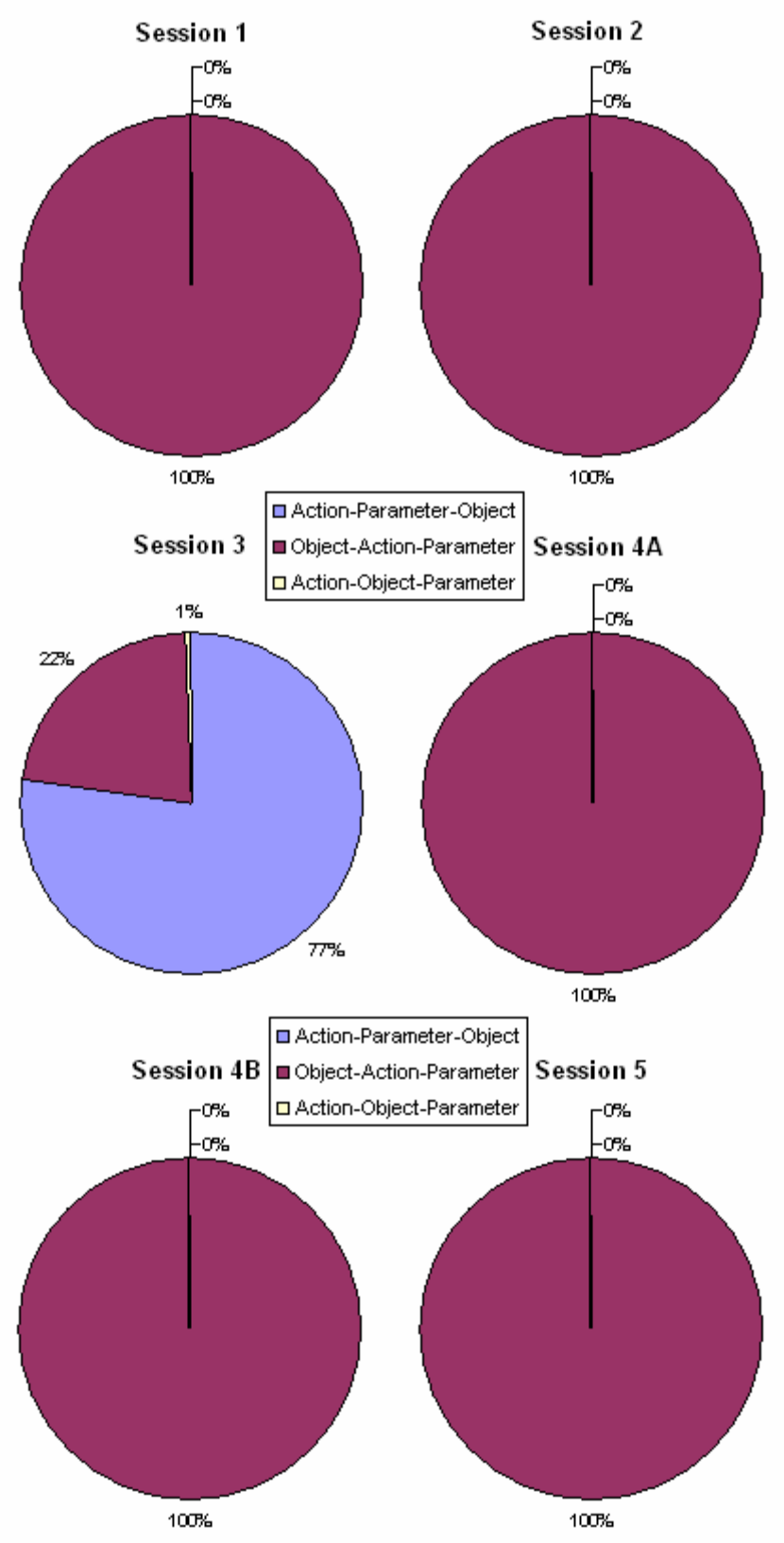


Figure 7.3 Percentages P2 used task sequences by session.

7.6.2 Participant Two

As seen in Figure 7.3, aside from the third session, Participant Two (P2) had a very clear task sequence preference for the Object-Action-Parameter task sequence, indicating that the participant preferred starting tasks by indication the object or location first.

As for Session 3, we believe that the change in P2's task sequence preferences was induced by the ordering of the task sequences used for the replication portions of the third session. As mentioned in Section 7.4, the third session consisted of replicating a goal room design using Object-first task sequences, *then* replicating a goal room design using *Action*-first task sequences, and then the play time seen in the percentages of Session 3. We believe P2 used the Action-Parameter-Object task sequences because the participant had just finished using those task sequences for the replication of the last goal room design. By looking at P2's usage of the Object-Action-Parameter task sequence over the time period of Session 3 (Figure 7.4), this idea seems to hold true since the usage increases towards the end of the session, hence, the usage of the Action-Object-Parameter task sequence reduced.

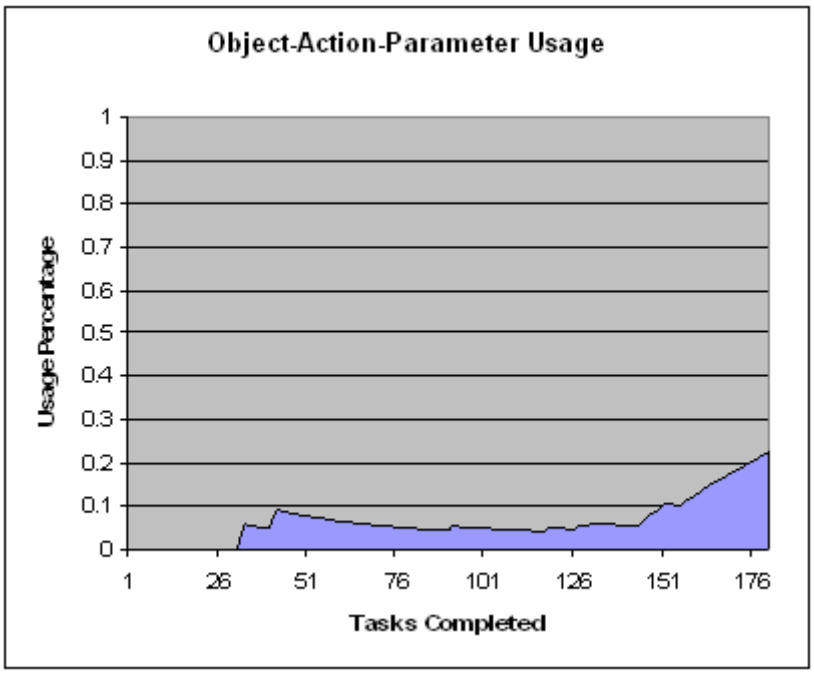


Figure 7.4 Percentage P2 used the Object-Action-Parameter task sequence during Session 3.

As a novice and an experienced user, P2 preferred the Object-Action-Parameter task sequence. The subjective responses of P2 during the first two step interviews and the final interview confirm this.

Additionally, according to the exit survey, P2 felt that he/she had progressed from a moderate user in the first session to an experienced user in the final session.

7.6.3 Participant Three

As seen in Figure 7.5, Participant Three (P3) started with a task sequence preference leaning toward the Action-Parameter-Object task sequence and quickly adopted a preference for the Object-Action-Parameter task sequence throughout the sessions.

During the first session, P3 used the Action-Parameter-Object task sequence the majority of the time even after completing the first ten tasks using the Object-Action-Parameter task sequence, but P3's usage of the Action-Parameter-Object task sequence was only marginally more than the participant's usage of the Object-Action-Parameter task sequence. We looked at the breakdown of tasks for P3 in Session 1 (Table 7.2) and determined that the participant did not have preferences based on the type of task. Therefore, we believe that P3 as a novice did not have a task sequence preference and used whatever task sequence the participant believed was easiest at the time of each task.

Table 7.2 Breakdown of Tasks for P3 during Session 1

	Action-Parameter-Object (Action-Object)	Object-Action-Parameter (Object-Action)	Action-Object-Parameter
Place	29	5	1
Edit	31	27	0
Move	2	18	
Rotate	5	7	
Cut	0	5	
Copy	0	1	
Paste	0	0	

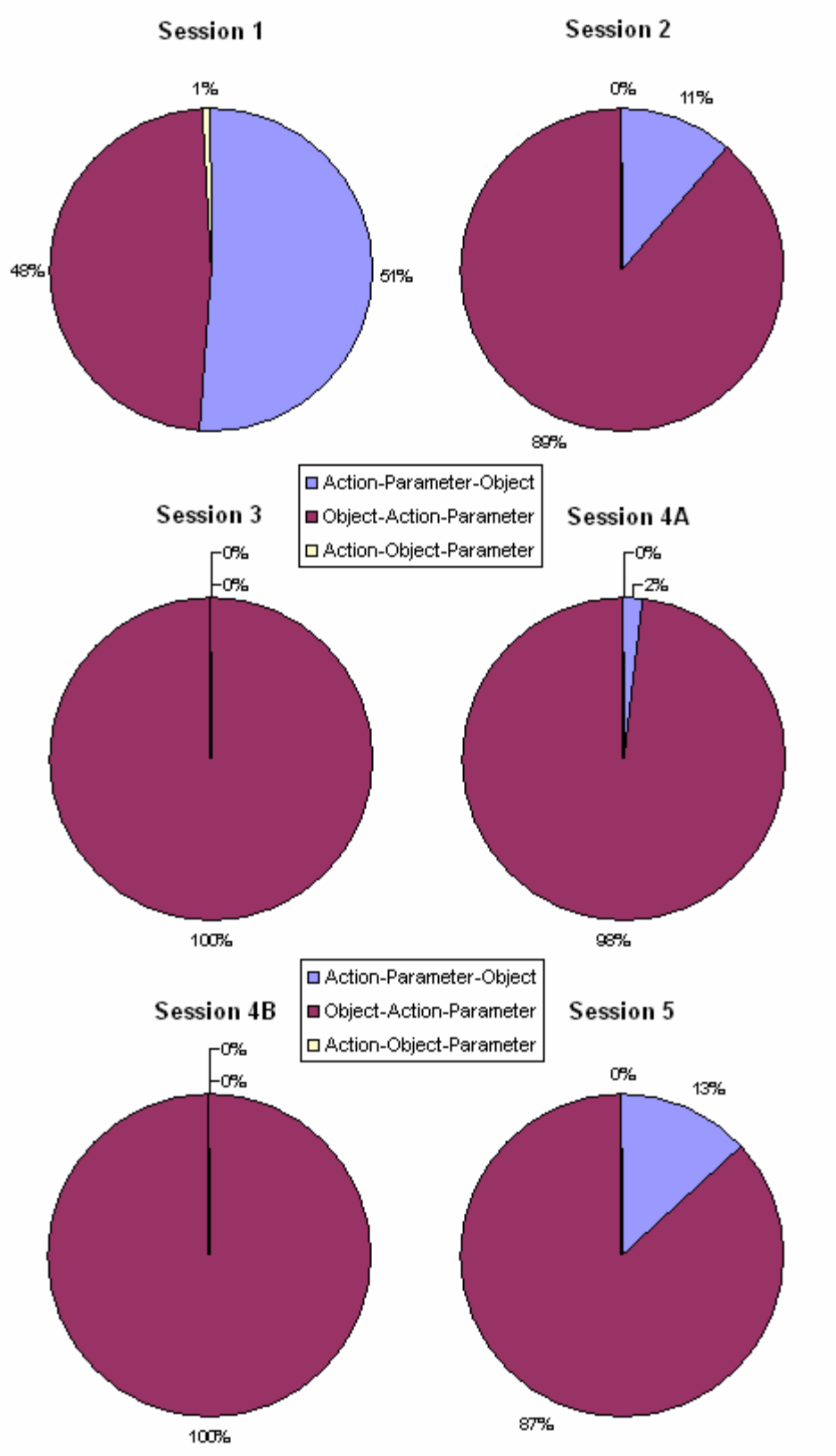


Figure 7.5 Percentages P3 used task sequences by session.

After the first session, P3 seems to have clearly adopted the Object-Action-Parameter task sequence as the task sequence of choice except for a small usage of the Action-Parameter-Object task sequence during Session 5. This small usage consisted mainly of edit tasks and we believe P3 was making adjustments for not being able to see parameters clearly on some of the popup menus in certain situations.

As a novice, P3 showed no clear task sequence preferences and, as an experienced user, P3 seemed to have preferred the Object-Action-Parameter task sequence. The subjective responses of P3 during the four step interviews and the final interview confirm this.

Additionally, according to the exit survey, P3 felt that he/she had progressed from a novice user in the first session to a near-expert user in the final session.

7.6.4 Participant Four

As seen in Figure 7.6, Participant Four (P4) began with a moderate preference for the Object-Action-Parameter task sequence and ended with a clear preference for the task sequence.

For Session 1, P4 used the Object-Action-Parameter task sequence predominantly except for place tasks. For this location-oriented task, the participant mainly used the Action-Parameter-Object task sequence as a user would use a tool palette to place shapes in a 3D modeling application. Session 2 was similar with the participant mainly using the Action-Parameter-Object task sequence except for place tasks and a few edit tasks.

As a novice and an experienced user, P4 preferred the Object-Action-Parameter task sequence. The subjective responses of P4 during the four step interviews and the final interview confirm this.

Additionally, according to the exit survey, P4 felt that he/she had progressed from a moderate user in the first session to an expert user in the final session.

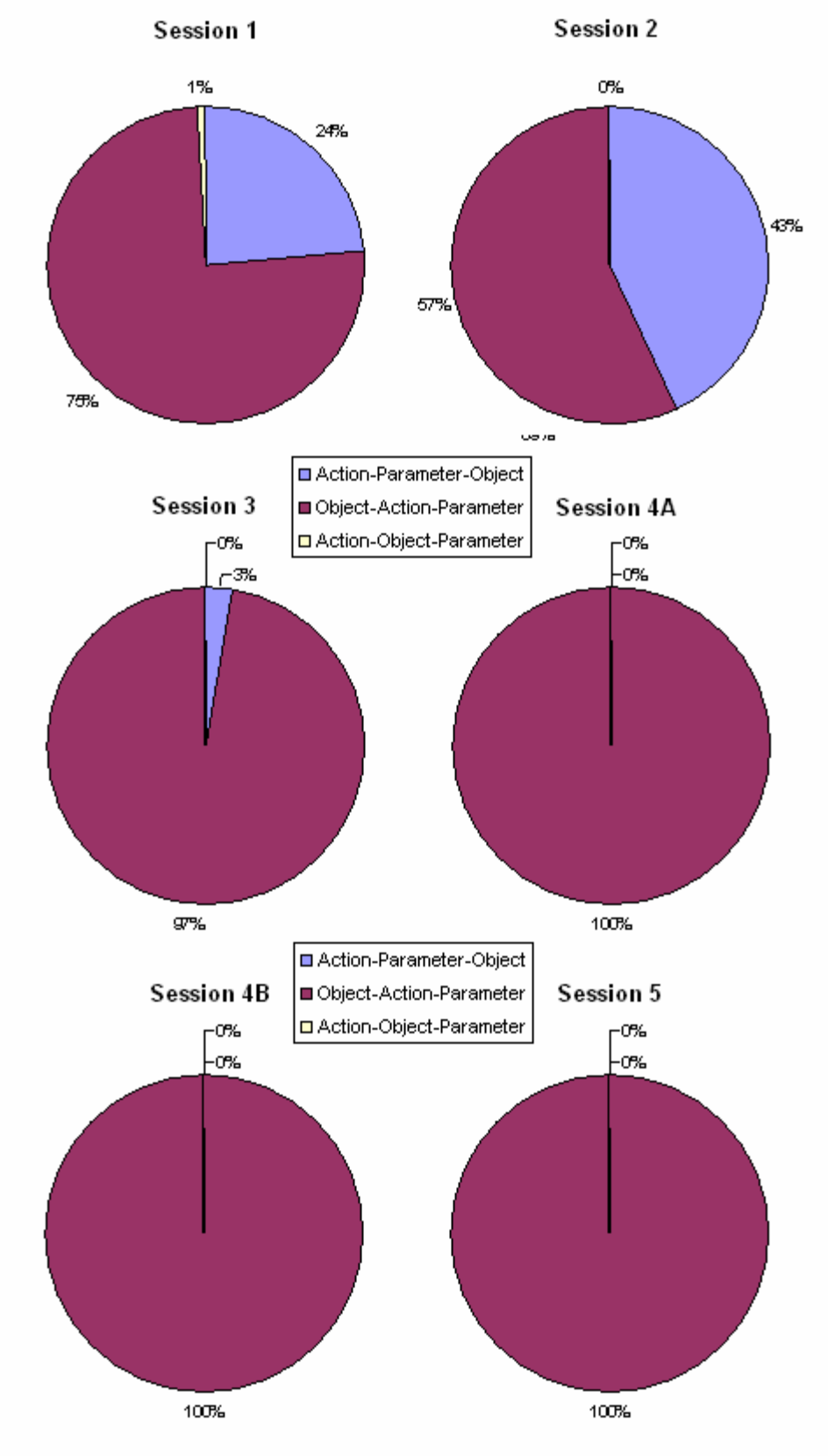


Figure 7.6 Percentages P4 used task sequences by session.

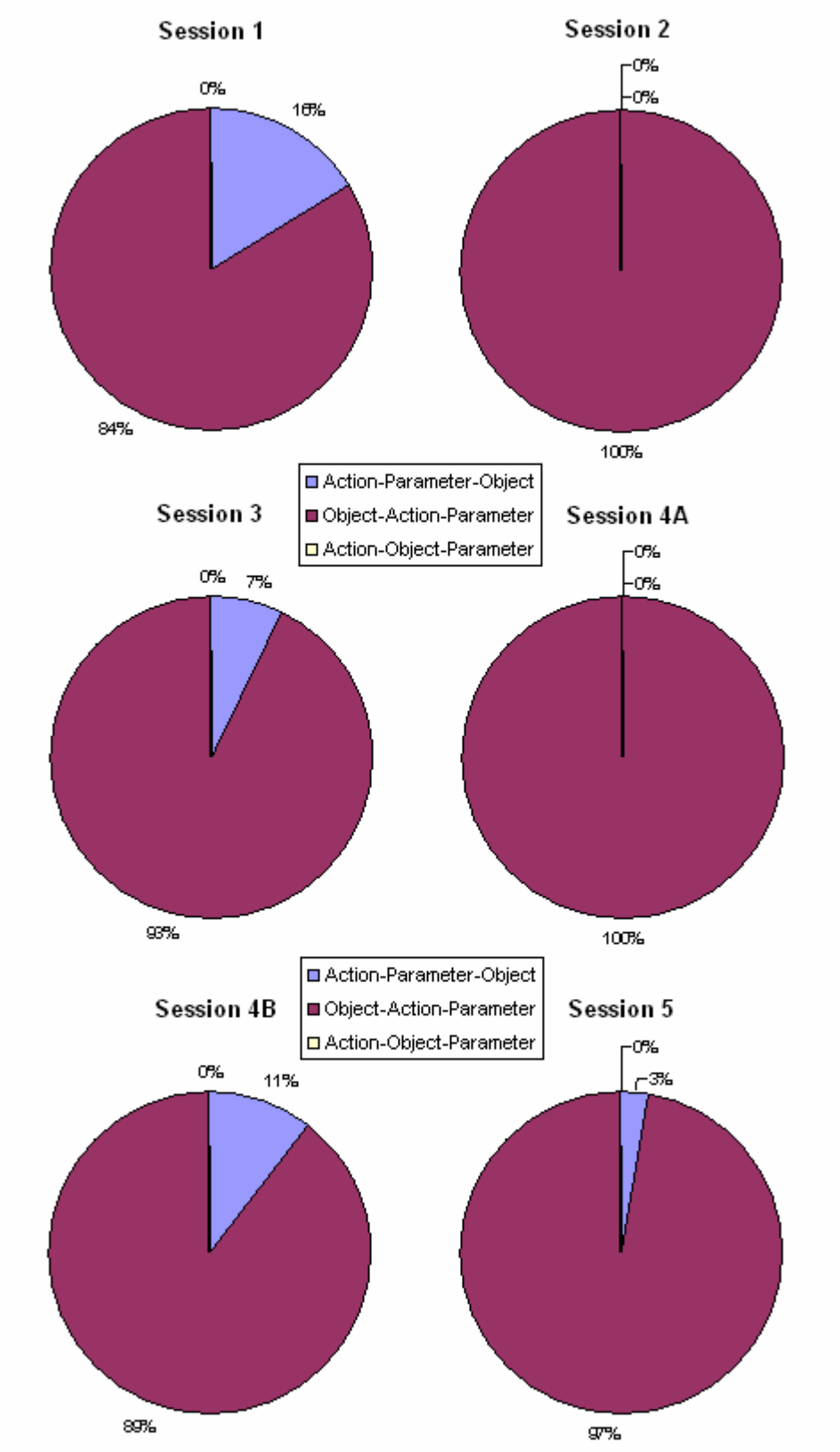


Figure 7.7 Percentages P5 used task sequences by session.

7.6.5 Participant Five

As seen in Figure 7.7, Participant Five (P5) had a consistent preference for the Object-Action-Parameter task sequence throughout the five sessions. Except for most of the place tasks in Session 1 and some of the edit tasks in Sessions 3 and 4B, P5 predominantly used the Object-Action-Parameter task sequence.

As a novice and an experienced user, P5 preferred the Object-Action-Parameter task sequence. The subjective responses of P5 during the four step interviews and the final interview confirm this.

Additionally, according to the exit survey, P5 felt that he/she had progressed from a moderate user in the first session to an expert user in the final session.

7.6.6 Participant Six

As seen in Figure 7.8, Participant Six (P6) had a consistent preference for the Object-Action-Parameter task sequence throughout the five sessions. Except for a minor number of place tasks in Session 2 and a minor number of place and edit tasks in Session 4A, P6 predominantly used the Object-Action-Parameter task sequence.

As a novice and an experienced user, P6 seemed to prefer the Object-Action-Parameter task sequence although the subjective responses of P6 during the four step interviews and the final interview show no preference for any task sequences.

Additionally, according to the exit survey, P6 felt that he/she had progressed from a moderate user in the first session to an expert user in the final session.

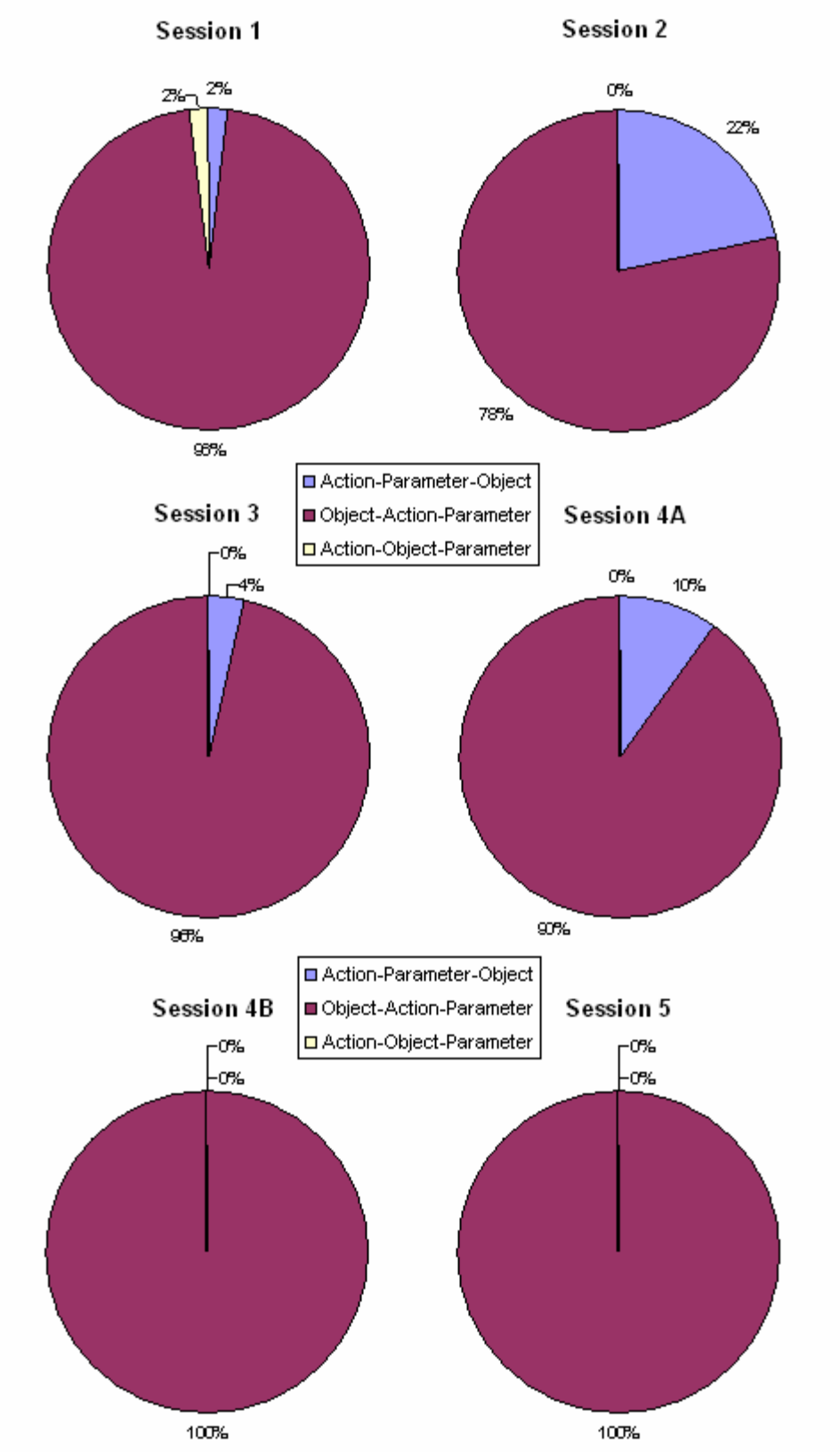


Figure 7.8 Percentages P6 used task sequences by session.

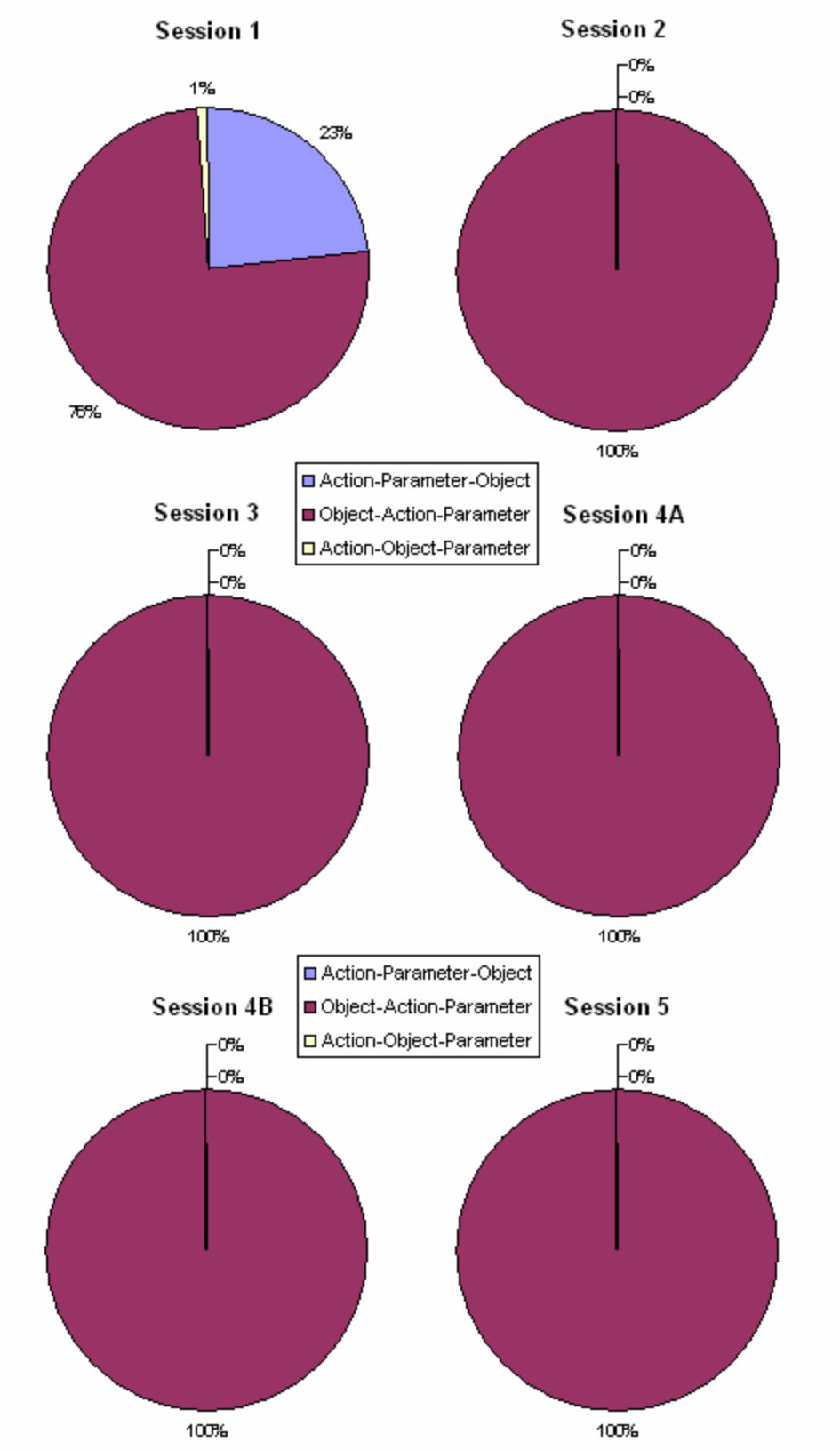


Figure 7.9 Percentages P7 used task sequences by session.

7.6.7 Participant Seven

As seen in Figure 7.9, Participant Seven (P7) began with a moderate preference for the Object-Action-Parameter task sequence and quickly adopted a clear preference for the task sequence.

For Session 1, P7 used the Object-Action-Parameter task sequence predominantly except for place tasks. For this location-oriented task, the participant mainly used the Action-Parameter-Object task.

As a novice and an experienced user, P7 preferred the Object-Action-Parameter task sequence. The subjective responses of P7 during the four step interviews and the final interview confirm this.

Additionally, according to the exit survey, P7 felt that he/she had progressed from a moderate user in the first session to an expert user in the final session.

7.6.8 Participant Eight

As seen in Figure 7.10, Participant Eight (P8) had a consistent preference for the Object-Action-Parameter task sequence throughout the five sessions. Except for some place tasks in Session 1, some random tasks in Session 2, most of the place tasks in Session 3, and some place tasks in Sessions 4B and 5, P8 predominantly used the Object-Action-Parameter task sequence.

As a novice and an experienced user, P8 preferred the Object-Action-Parameter task sequence. The subjective responses of P8 during the four step interviews and the final interview confirm this.

Additionally, according to the exit survey, P8 felt that he/she had progressed from a near-expert user in the first session to an expert user in the final session.

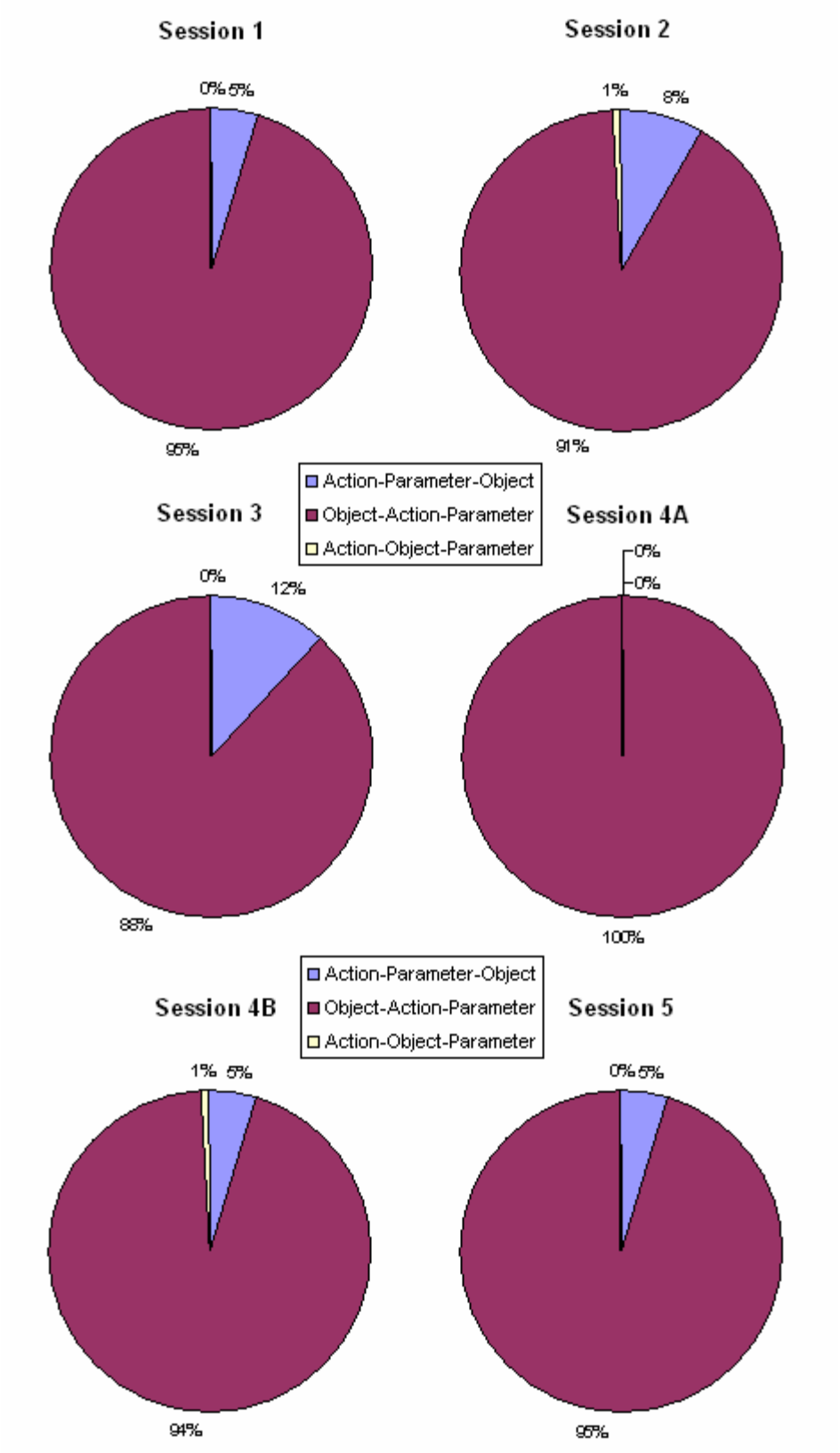


Figure 7.10 Percentages P8 used task sequences by session.

7.7 Discussion

From the task sequence preference study, we have clearly established the preferences of most novices and experienced users.

As novices, all of the participants except for P3 used the Object-Action-Parameter (or Object-Action) task sequence for most of the tasks. In fact, half of the participants (P1, P2, P6, and P8) used the Object-Action-Parameter task sequence for 95% or more of the tasks during Session 1.

As they became experienced users, all of the participants adopted using the Object-Action-Parameter task sequence for most tasks. In fact, P2 was the only participant to use another task sequence for the majority of tasks during a session (we discussed this phenomenon in Section 7.6.2). For the final session, all of the participants except P3 used the Object-Action-Parameter task sequence for 95% or more of the tasks with P1, P2, P4, P6, and P7 using solely the Object-Action-Parameter task sequence.

Clearly from the results of this study, it appears that novices and experienced users alike prefer the Object-Action-Parameter task sequence for completing tasks in an IVE. One implication of this is that both novices and experienced users are likely to prefer SCIs utilizing task sequences that begin with the indication of objects and locations over SCIs utilizing other task sequences. Another implication is novices and experienced users must prefer these Object-Action and Object-Action-Parameter task sequences because these task sequences are easier to understand and execute than other task sequences. This also implies that these task sequences must increase usability of SCIs that utilize them.

Another interesting result of this study was that the Action-Object-Parameter task sequence was actually used and preferred the least by the participants. We believe this was caused by the separation of the indications of the action and parameter, which we think most users associate together. This would explain why the Action-Parameter-Object task sequence was preferred more than the Action-Object-Parameter task sequence.

Chapter 8 Conclusions and Future Work

8.1 Summary

This research has focused on evaluating task sequences in regard to usability, with the goal of potentially standardizing task sequences for SCIs. By establishing a standard for task sequences, we hope to eliminate one confound for comparing these interfaces and would be closer to establishing a testbed for SCIs.

In the course of our research, we developed several research questions regarding the design of system control interfaces and the user of task sequences. We now attempt to answer those questions.

1. How do task sequences affect user performance in immersive virtual environments?

One part of determining the usability of an interface is evaluating the effect that that interface has on the user's performance. Since underlying task sequences can affect the interaction flow of an interface, we wanted to know if different task sequences change the effect that an interface has on user performance.

In Section 4.3, we discussed determining the effects of task sequences on user performance. We determined that it was insufficient to evaluate absolute task performance in regard to the effects of task sequences because a SCI can bias the absolute task performance of one task sequence over another due to the design and implementation of the SCI. Because the SCI used to evaluate task sequences could be biased, we developed an alternative method, to compare the cognitive effort induced by each task sequence.

In order to determine the cognitive effort induced by each task sequence, we developed a model which defines the total time to complete a task as the sum of the time for the perceptual and motor processes needed to select the necessary components and the cognitive time induced by the underlying task sequence:

$$\mathbf{T_{Total} = T_{Perceptual\text{-}Motor} + T_{Cognitive}}$$

Using this model for user performance, we designed two studies to collect the data necessary to estimate the cognitive effort induced by task sequences. In the first study (see Chapter 5), we established estimates for the time required to complete certain system control tasks with the VEWL SCI based on atomic actions such as clicking a command button and selecting a popup menu item. In the second study (see Chapter 6), we established the times required to complete certain interior design tasks. By subtracting $\mathbf{T_{Perceptual\text{-}Motor}}$, established in the first study, from $\mathbf{T_{Total}}$, established in the second study, we estimated $\mathbf{T_{Cognitive}}$ for different task sequences.

Comparing $\mathbf{T_{Cognitive}}$ for the various task sequences we identified in Chapter 3, we found that task sequence does have a significant effect on the cognitive effort induced and that Object-first task sequences, such as the Object-Action task sequence, induce smaller cognitive effort than Action-first task sequences, such as the Action-Object task sequence. We also found that the focus of a task sequence (object- or location-oriented) also has a significant effect on the cognitive effort induced and object-oriented task sequences induce smaller cognitive effort than location-oriented task sequences.

2. Which task sequences do users prefer in immersive virtual environments?

Another consideration in determining the usability of an interface is assessing user preference. For this research, we wanted to know if most users prefer one task sequence over others and why. We also wanted to know if novices prefer the same task sequences as experienced users.

In Section 4.4, we discussed determining which task sequences users prefer. For this research, we were concerned with two aspects of user preferences – which task sequences novices prefer and how those preferences evolve as novices become experienced users. In order to evaluate these two aspects, we ran a longitudinal study in which users would have choices of which task sequences to use for completing interior design tasks (see Chapter 7). The SCI for this study was designed to support multiple task sequences so that users would have open preferences and would not be influenced on which task sequence to use. By recording data on which task

sequences participants used throughout the longitudinal study, we were able to analyze the preferences of novices and how those preferences evolved as the novices became experienced users.

From our longitudinal study, we found that both novices and experienced users alike prefer Object-first task sequences, such as the Object-Action task sequence, over Action-first task sequences, such as the Action-Object task sequence. These results indicate that most users would probably prefer the Object-Action task sequence and similar task sequences for completing tasks in IVEs.

3. Is there a single task sequence that should be utilized by system control interfaces for immersive virtual environments and be accepted as the standard task sequence?

As of now, there is not a standard task sequence utilized by SCIs for IVEs. For this research, we wanted to know if SCIs for IVEs should utilize a single task sequence and if there should be a standard task sequence, which could be used in the development of a SCI testbed.

With the results of the task sequence performance study and the task sequence preference study indicating that Object-first task sequences, such as the Object-Action task sequence, induce smaller cognitive effort and are preferred by novices and experienced users alike, we believe that we have found a single category of task sequences that should be utilized by SCIs for IVEs and be accepted as standards. We discuss this topic in more detail in the next section.

8.2 Contributions

Given the results of our studies, we feel that this research makes several contributions including setting standard task sequences for a SCI testbed and making recommendations for SCI developers on task sequences. Despite these contributions, it should be noted that we only studied task sequences in a single specific task context (interior design scenarios) and with only one specific SCI (VEWL), so the results of our research should not be over-generalized. Further research is needed to validate these results in other task contexts and with other SCIs.

8.2.1 Standard Task Sequences for a SCI Testbed

As mentioned in Chapter 1, a major problem with creating a SCI testbed is that usually a SCI does not dictate how it should be used in regard to task sequences, which means that the same SCI could be evaluated in more than one way. If standards for task sequences were established for evaluating SCIs, we would eliminate this problem and would be one step closer to establishing a SCI testbed.

Based on the results of our task sequence performance study and our task sequence preference study, which indicate that Object-first task sequences induce smaller cognitive effort and are preferred by novices and experienced users, we believe that we have found a single category of task sequences that should be used as the standard task sequences for a SCI testbed. For simple tasks, those involving only an action and an object or location, we believe that the Object-Action task sequence should be the standard task sequence. For complex tasks, those involving an action, an object or location, and parameters, we believe that the Object-Action-Parameter task sequence should be the standard task sequence.

For example, assume that we have designed a SCI testbed and want to evaluate the VEWL SCI and a ring menu SCI. Since both of these SCIs do not dictate how they should be used in regard to task sequences, we can design each SCI to use the Object-Action task sequence for simple tasks and the Object-Action-Parameter task sequence for complex tasks for evaluation. This should improve the usability of each interface and most importantly eliminates task sequence as a possible confound for comparing the evaluations of these two SCIs.

Establishing the Object-Action and Object-Action-Parameter task sequences as the standard task sequences for a SCI testbed has the obvious benefit of eliminating task sequence as a possible confound, if the SCIs evaluated do not dictate how they should be used in regard to task sequences. If a SCI does dictate how it should be used in regard to task sequences and it does not utilize the Object-Action and Object-Action-Parameter task sequences, these established standards are of no use. On the other hand, from this research, we do know how task sequences affect user performance and usability and should be able to account for these effects.

For example, consider a toolbelt SCI that uses multiple physical tools to accomplish various tasks. This SCI forces users to use the Action-Object task sequence since for each task the user must select the proper tool from the toolbelt and then use that tool on the desired object or at the desired location. Since we cannot evaluate this toolbelt SCI with the Object-Action and Object-Action-Parameter task sequences, we must instead account for the SCI using different task sequences.

Assume that we want to compare the toolbelt SCI to a SCI previously evaluated using the Object-Action task sequence. Now consider the task of pasting an object at a specified location in an IVE. Without accounting for task sequences, if the toolbelt SCI required 1.0 seconds longer to complete the task than the previously evaluated SCI and this difference in task completion time was significant, we would have to say that the previously evaluated SCI was more usable for pasting tasks than the toolbelt SCI. Now by accounting for the toolbelt SCI using a location-oriented Action-Object task sequence instead of a location-oriented Object-Action task sequence (a cognitive effort difference of 1.962 seconds), the toolbelt SCI actually required 0.962 seconds less to complete the task than the previously evaluated SCI. If this difference was significant, we would instead say that the toolbelt SCI was more usable for pasting tasks than the previously evaluated SCI.

As seen in this discussion, by establishing the Object-Action and Object-Action-Parameter task sequences as standards for a SCI testbed and knowing how task sequences affect user performance and usability, we have eliminated task sequence as a possible confound for a SCI testbed and provided information that improves the evaluating of such interfaces.

8.2.2 Considerations for Developing SCIs

Another contribution that this research makes is several considerations for the process of developing SCIs. One obvious consideration is to try to develop SCIs that can use the Object-Action and Object-Action-Parameter task sequences. Because these task sequences induce smaller cognitive effort and are preferred by novices and experienced users, SCIs that can use these task sequences will have increased user performance and usability. And since developers

are always trying to develop better user interfaces, it only makes sense to provide for these increases in usability.

A second consideration in developing SCIs is to look for alternative task sequences to use. For instance, assume we are developing a toolbelt SCI, which requires task sequences to begin with the indication of an action (or physical tool in this case). We are obviously unable to design the toolbelt SCI to utilize the Object-Action-Parameter task sequence for complex tasks, but we do have a choice between the Action-Object-Parameter and Action-Parameter-Object task sequences. Though the difference in cognitive effort induced by these two task sequences is not a significant difference, it might be better to use the Action-Parameter-Object task sequence judging by this difference.

Another consideration in developing SCIs is to attempt to design SCIs that utilize combination task sequences. An interesting discovery made in the task sequence performance study was that the mean cognitive times for complex task sequences were higher than those for the simple task sequences, but the mean cognitive times for combination task sequences were not any higher than those for the simple task sequences. For combination task sequences, we believe the combination of two of the three induced selections decreases the cognitive effort. Hence, by developing a SCI that utilized these combination task sequences, we could harness the power of these sequences to create a more usable interface. For example a SCI designed for the Object-Action+Parameter task sequence would probably induce about 3 seconds less cognitive time per complex task than a SCI designed for the Object-Action-Parameter task sequence.

8.3 Future Work

For future work, there are several different research directions that could be pursued based on the research presented in this thesis. The first research direction for consideration, and what we consider the most important, is the design and development of a testbed evaluation for SCIs. The benefits of such a testbed would be great as it would provide an empirical method for evaluating and comparing current SCIs in detail. It would also lead to guidelines for choosing a SCI based on a particular application scenario. Additionally, as future SCIs are developed, reusability

would allow a new SCI to be compared to previously evaluated SCIs by running formal experiments solely for the new SCI.

The second research direction for consideration of future work would be running experiments similar to the ones presented here with other task contexts and other SCIs. In our research, we only studied task sequences in a single task context (interior design scenarios) and with only one specific SCI (VEWL), so the results of our research should not be over-generalized. By running similar experiments in other task contexts and with other SCIs, we can hopefully validate the results presented here.

A third research direction to consider would be attempting the task sequence instinct study again. Our second experimental design failed due to biased instructions. At the time, we did not know how to overcome this fault, but after creating the software for the task sequence preference study, with the feature of goal room designs for providing tasks to the participants, we believe that a think-aloud experiment could be conducted with the use of physical models to represent current situations and goal situations. Consideration of how to determine which task sequence corresponds to a participant's thoughts would still have to be taken.

Another research direction for future work would be to evaluate the effects of natural language on the studies presented in this research. All of the experiments for this research were conducted in the English language. All of the verbal directives were given in English. Would the results have differed if the research was conducted in another language? Could the results have differed if the participants were English Second Language (ESL) users? These are questions that could provide interesting insight into natural languages and cultures for human-computer interaction.

A final research direction for consideration is further developing our task sequence performance model to produce more accurate estimates of cognitive effort and to better understand how users think about and perform different task sequences. Instead of defining the total time to complete a task as the sum of two operators, research could lead to the development of a more complex task sequence performance model in which perceptual and motor actions are not combined and the fact that these actions and cognitive effort may overlap is accounted for.

References

Bowman, D., Johnson, D. & Hodges, L. (2001). Testbed evaluation of virtual environment interaction techniques. *Presence: Teleoperators and Virtual Environments* 10: 75-95.

Bowman, D. A., Kruijff, E., Joseph J. LaViola, J. & Poupyrev, I. (2005). *3D User Interfaces: Theory and Practice* (p. 478). Pearson Education, Inc.

Bowman, D. A., Setareh, M., Pinho, M. S., Ali, N., Kalita, A., Yunha, L., Lucas, J., Gracey, M., Kothapalli, M., Qinwei, Z., Datey, A. & Tumati, P. (2003) Virtual-SAP: an immersive tool for visualizing the response of building structures to environmental conditions. pp. 243-250).

Bowman, D. A. & Wingrave, C. (2001) Design and Evaluation of Menu Systems for Immersive Virtual Environments. *IEEE Virtual Reality 2001* pp. 149-156).

Bullinger, H., Kern, P. & Braun, M. (1997). Controls. In: Salvendy, G., (Ed) *Handbook of Human Factors and Ergonomics*. John Wiley and Sons.

Card, S. K., Moran, T. P. & Newell, A. (1980). The Keystroke-Level Model for User Performance Time with Interactive Systems. *Communications of the ACM* 23: 396-410.

Cruz-Neira, C., Sandin, D. J. & DeFanti, T. A. (1993) Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In: Proceedings of the 20th annual conference on Computer graphics and interactive techniques. ACM Press.

Gray, W. D., John, B. E., Stuart, R., Lawrence, D. & Atwood, M. E. (1990) GOMS meets the phone company: Analytic modeling applied to real-world problems. *INTERACT '90: Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction* pp. 29-34). Amsterdam, The Netherlands, The Netherlands.

Grosjean, J., Burkhardt, J. M., Coquillart, S. & Richard, P. (2002) Evaluation of the Command and Control Cube. pp. 473-478).

Hutchins, E. L., Hollan, J. D. & Norman, D. A. (1985). Direct Manipulation Interfaces. In: Norman, D. A. & Draper, S. W., (Eds), *User Centered System Design: New Perspectives on Human-Computer Interaction*.

Kelso, J., Arsenault, L., Satterfield, S. & Kriz, R. (2002) DIVERSE: A Framework for Building Extensible and Reconfigurable Device Independent Virtual Environments. *IEEE Virtual Reality 2002*.

Larimer, D. & Bowman, D. A. (2003) VEWL: A Framework for Building a Windowing Interface in a Virtual Environment. *INTERACT: IFP TC13 International Conference on Human-Computer Interaction* pp. 1-7).

Liang, J. & Green, M. (1994). JDCAD: A Highly Interactive 3D Modeling System. *Computers and Graphics* 18: 499-506.

Lindeman, R. W., Sibert, J. L. & Hahn, J. K. (1999) Hand-held windows: towards effective 2D interaction in immersive virtual environments. pp. 205-212).

Mackie, C., Cowden, J., Bowman, D. A. & Thabet, W. Y. (2004) Desktop and Immersive Tools for Residential Home Design. *CONVR Conference on Construction Applications of Virtual Reality*.

Mapes, D. P. & Moshell, J. M. (1995). A two-handed interface for object manipulation in virtual environments. *Presence: Teleoperators and Virtual Environments* 4: 403-416.

McMahan, R. P. & Bowman, D. A. (2007) An Empirical Comparison of Task Sequences for Immersive Virtual Environments. *IEEE Symposium on 3D User Interfaces (3DUI '07)*.

McMillan, G., Eggelston, R. & Anderson, T. (1997). Nonconventional Controls. In: Salvendy, G., (Ed) *Handbook of Human Factors and Ergonomics*. John Wiley and Sons.

Newman, W. M. & Sproull, R. F. (1973). *Principles of Interactive Computer Graphics* New York: McGraw-Hill.

Preece, J., Rogers, Y. & Sharp, H. (2002). *Interaction Design: Beyond Human-Computer Interaction*: John Wiley and Sons.

Shaw, C. & Green, M. (1994) Two-Handed Polygonal Surface Design. *1994 ACM Symposium on User Interface Software and Technology (UIST '94)* pp. 205-212).

Shneiderman, B. (1998). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*: Addison Wesley Longman, Inc.

Stephenson, N. (1999). *In the Beginning...was the Command Line* (p. 160). Harper Perennial.

Sutherland, I. E. (1963) Sketchpad: A man-machine graphical communication system. *SpringJoint Computer Conference* pp. 329-346). Baltimore, MD.

Zhao, W. & Madhavan, V. (2006) Virtual assembly operations with grasp and verbal interaction. *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications* pp. 245-254). Hong Kong, China.

Appendix A Task Survey

A.1 Task Survey of 3DUI Listserv

Introduction

We are collecting information about tasks and goals involving system control (menus and other forms of commands) in VE applications. We are interested in both basic user tasks and complex user tasks and goals. The set of basic user tasks will be the focus of a verb-object versus object-verb experiment. The set of complex user tasks and goals will be used to run a longitudinal experiment that examines various possible task structures used in VEs. This experiment will help determine which task structures are best for novice users and which are best for experienced users.

A system control task is a user task in which a command is issued to request the system to perform a particular function, change the mode of interaction, or change the system state. A system control technique is the method by which system control tasks are accomplished. In 2D interfaces, pull-down menus, pop-up menus, toolboxes, palettes, toggles, radio buttons, and checkboxes are all examples of system control techniques. A system control interface (SCI) is composed of system control techniques to form a flow of system control tasks for the user. Examples of VE SCIs are VEWL (Virtual Environment Windowing Library), ring menus, TULIP (Three-Up, Labels In Palms) menus, and voice command interfaces.

Instructions – Part 1

For the first part of this survey, we ask you to list the basic user tasks that you have encountered in VE applications. We are also interested in how these basic user tasks were performed in regards to the application's system control interface and other interaction techniques. We also request that similar tasks are defined as a single universal task.

Example – Part 1

In an architecture and construction application known as Home Design, two of the basic user tasks are placing a window in a wall and placing a door in wall. For both of these tasks, the window and door are prefabricated objects. So these two tasks can be combined into the following:

Application: Home Design – an architecture and construction application concerned with residential homes.
Task: Place a prefabricated object in a wall.
Details: A user interacts with the SCI (a floating window) to select which prefabricated object is desired. This changes the mode of interaction to placing an object. Then using ray casting, the user places the object in a wall at the desired location. After placement, the mode of interaction returns to selection and manipulation which is the application's normal mode of interaction. Prefabricated objects include windows and doors.

Feel free to list as many basic user tasks as you can.

Application:
Task:
Details:

Instructions – Part 2

For the second part of this survey, we ask you to list possible complex user tasks or goals for the applications you listed basic tasks for. Complex user tasks or goals should consist of many steps or basic tasks. They need not be tied to specific features of the application but should be realistic complex user tasks or goals that you have encountered. We also ask for possible task structures, that is, different sets of actions that can be used to accomplish the task or goal.

Example – Part 2

For example, in Home Design, a user may wish to move all the windows from the north side of a house to the south side. Assuming that deleting an object from a wall and moving an object from one wall to another were defined in the basic user tasks, the goal would be listed as the following:

Application: Home Design – an architecture and construction application concerned with residential homes.
Complex Task/Goal: Move all the objects of the same type from one wall to another wall.
Subtasks: Delete an object from a wall.
Place an object in a wall.
Move an object from one wall to another.
Possible task structures:

1. User can delete all the objects of the same type from the north wall. The user will then place the same number of objects in the south wall.
2. User can place the same number of objects in the south wall as the north wall. The user then deletes all the objects of the specified type from the north wall.
3. User can move each object of the same type from the north wall to the south wall, one at a time.

Feel free to list as many complex user tasks or goals as you can.

Application:
Complex Task/Goal:
Subtasks:
Possible task structures:

Thank you for your time and information.

Appendix B VEWL Performance Study Forms

B.1 Background Survey

Please help us to categorize our user population by completing the following items.

Gender (circle one): Male Female

Age: _____

Do you wear glasses or contact lenses? (circle one)
No Glasses Contact Lenses

Are you left or right handed? _____

Occupation (if student, indicate graduate or undergraduate):
_____

Major / Area of specialization (if student): _____

Rate your familiarity with computers: (circle one)

•----------•----------•----------•----------•----------•
not at all familiar not very familiar somewhat familiar fairly familiar very familiar

How often do you use computers? (circle the best answer)

a. not at all
b. once a month
c. once a week
d. several times a week
e. daily

Which operating system do you use the most? (for example: Windows XP, Mac OS X)

_____

Have you ever used a virtual reality (VR) system? If so, please describe it (what type of display was used, what kind of application was it, how you interacted with the system.).

_____

_____

_____

B.2 Instructions

B.2.1 Instructions for Windows Experiment

For this study, you will be asked to perform a set of tasks using VEWL (Virtual Environment Windowing Library) within the CAVE™. Specifically, your role in this experiment is that of evaluator of VEWL's windows. We are not evaluating you. All information that you help us attain will remain anonymous. The time you take to do each task and other aspects of your interaction with the system will be measured.

The session is divided into three parts: focusing windows, moving windows, closing windows. You will be asked to stand in a designated spot within the CAVE with your hands down by your side before starting each task. You will also be asked to start each task looking at a designated spot within the CAVE. You will be asked to complete each task as effectively and quickly as you can.

For the focusing of windows, you will be told a position within the CAVE. A VEWL window will then appear within the virtual environment at the position told. You will point to this window and focus it by pressing either of the left wand buttons. This process will be repeated 15 times for different positions within the CAVE and size of windows that appear.

For the moving of windows, you will be told a start position and an end position. A VEWL window will then appear within the virtual environment at the start position told and a red dot will appear at the end position told. You will point to this window and hold either of the right wand buttons down to begin moving the window. You will then point to the red dot and

release the button held down to finish moving the window. This process will be repeated 15 times for different start positions, end positions, and accuracy of placement desired.

For the closing of windows, you will be told a position within the CAVE. A VEWL window will then appear within the virtual environment at the position told. You will point to the "Close" button at the top of this window and click it by pressing either of the left wand buttons. This process will be repeated 5 times for different positions within the CAVE.

Remember, you are free to withdraw from this study at any time for any reason.

B.2.2 Instructions for Checkbox Experiment

For this study, you will be asked to perform a set of tasks using VEWL (Virtual Environment Windowing Library) within the CAVE™. Specifically, your role in this experiment is that of evaluator of VEWL's checkboxes. We are not evaluating you. All information that you help us attain will remain anonymous. The time you take to do each task and other aspects of your interaction with the system will be measured.

The session has two parts: clicking a checkbox and clicking a checkbox label. You will be asked to stand in a designated spot within the CAVE with your hands down by your side before starting each task. You will also be asked to start each task looking at a designated spot within the CAVE. You will be asked to complete each task as effectively and quickly as you can.

For clicking a checkbox, you will be told a position within the CAVE and a label number. A VEWL window will then appear within the virtual environment at the position told. A number of checkboxes will be on this window. You will point to the checkbox with a label corresponding to the label number told and click it by pressing either of the left wand buttons. This process will be repeated 15 times for different positions within the CAVE, label numbers, and number of checkboxes on the window.

For clicking a checkbox label, you will be told a position within the CAVE and a label number. A VEWL window will appear within the virtual environment at the position told. A number of checkboxes will be on this window. You will point to the checkbox label corresponding to the label number told and click it by pressing either of the left wand buttons. This process will be repeated 15 times for different positions within the CAVE, label numbers, and number of checkboxes on the window.

Remember, you are free to withdraw from this study at any time for any reason.

B.2.3 Instructions for Command Button Experiment

For this study, you will be asked to perform a set of tasks using VEWL (Virtual Environment Windowing Library) within the CAVE™. Specifically, your role in this experiment is that of evaluator of VEWL's command buttons. We are not evaluating you. All information that you help us attain will remain anonymous. The time you take to do each task and other aspects of your interaction with the system will be measured.

The session has one part: clicking a command button. You will be asked to stand in a designated spot within the CAVE with your hands down by your side before starting each task. You will also be asked to start each task looking at a designated spot within the CAVE. You will be asked to complete each task as effectively and quickly as you can.

For clicking a command button, you will be told a position within the CAVE and a button number. A VEWL window will then appear within the virtual environment at the position told. A number of command buttons will be on this window. You will point to the button corresponding to the button number told and click it by pressing either of the left wand buttons. This process will be repeated 30 times for different positions within the CAVE, button numbers, and number of command buttons on the window.

Remember, you are free to withdraw from this study at any time for any reason.

B.2.4 Instructions for Dropdown Menu Experiment

For this study, you will be asked to perform a set of tasks using VEWL (Virtual Environment Windowing Library) within the CAVE™. Specifically, your role in this experiment is that of evaluator of VEWL's dropdown menus. We are not evaluating you. All information that you help us attain will remain anonymous. The time you take to do each task and other aspects of your interaction with the system will be measured.

The session has two parts: scrolling a dropdown menu and choosing an item from a dropdown menu. You will be asked to stand in a designated spot within the CAVE with your hands down by your side before starting each task. You will also be asked to start each task

looking at a designated spot within the CAVE. You will be asked to complete each task as effectively and quickly as you can.

For scrolling a dropdown menu, you will be told a position within the CAVE, a scroll bar element, and a list item number. A VEWL window will then appear within the virtual environment at the position told. A dropdown menu will be on this window. You will point to the dropdown menu and activate it by pressing either of the left wand buttons. A scrolling list will then appear. You will point to the scroll bar element told and click it by pressing either of the left wand buttons. You will continue to click the element until the first item in the list corresponds to the list item number told. This process will be repeated 12 times for different positions within the CAVE, scroll bar elements, and list items.

For choosing an item from a dropdown menu, you will be told a position within the CAVE and a list item number. A VEWL window will then appear within the virtual environment at the position told. A dropdown menu will be on this window. You will point to the dropdown menu and activate it by pressing either of the left wand buttons. A scrolling list will then appear. You will point to the list item corresponding to the list item number told and click it by pressing either of the left wand buttons. This process will be repeated 18 times for different positions within the CAVE, list item numbers, and number items in the list visible.

Remember, you are free to withdraw from this study at any time for any reason.

B.2.5 Instructions for Scrolling List Experiment

For this study, you will be asked to perform a set of tasks using VEWL (Virtual Environment Windowing Library) within the CAVE™. Specifically, your role in this experiment is that of evaluator of VEWL's scrolling list. We are not evaluating you. All information that you help us attain will remain anonymous. The time you take to do each task and other aspects of your interaction with the system will be measured.

The session has two parts: scrolling a list and choosing an item from a list. You will be asked to stand in a designated spot within the CAVE with your hands down by your side before starting each task. You will also be asked to start each task looking at a designated spot within the CAVE. You will be asked to complete each task as effectively and quickly as you can.

For scrolling a list, you will be told a position within the CAVE, a scroll bar element, and a list item number. A VEWL window will then appear within the virtual environment at the position told. A scrolling list will be on this window. You will point to the scroll bar element told and click it by pressing either of the left wand buttons. You will continue to click the element until the first item in the list corresponds to the list item number told. This process will be repeated 12 times for different positions within the CAVE, scroll bar elements, and list items.

For choosing an item from a list, you will be told a position within the CAVE and a list item number. A VEWL window will then appear within the virtual environment at the position told. A scrolling list will be on this window. You will point to the list item corresponding to the list item number told and click it by pressing either of the left wand buttons. This process will be repeated 18 times for different positions within the CAVE, list item numbers, and number items in the list visible.

Remember, you are free to withdraw from this study at any time for any reason.

B.2.6 Instructions for Popup Menu Experiment

For this study, you will be asked to perform a set of tasks using VEWL (Virtual Environment Windowing Library) within the CAVE™. Specifically, your role in this experiment is that of evaluator of VEWL's popup menus. We are not evaluating you. All information that you help us attain will remain anonymous. The time you take to do each task and other aspects of your interaction with the system will be measured.

The session is divided into two parts: selection of top level popup menu items and selection of second level popup menu items. You will be asked to stand in a designated spot within the CAVE with your hands down by your side before starting each task. You will also be asked to start each task looking at a designated spot within the CAVE. You will be asked to complete each task as effectively and quickly as you can.

For the selection of top level popup menu items, you will be told a position within the CAVE and an item number. A red dot will then appear within the virtual environment at the position told. You will point to this red dot and press either of the right wand buttons to display a VEWL popup menu. The popup menu will appear at the location of the red dot. You will then choose the item from the menu corresponding to the item number previously told to you by

pointing to the item and pressing either of the left wand buttons. This process will be repeated 18 times for different positions within the CAVE, item numbers, and number of items within the popup menu.

For the selection of second level popup menu items, you will be told a position within the CAVE, a submenu number, and an item number. A red dot will then appear within the virtual environment at the position told. You will point to this red dot and press either of the right wand buttons to display a VEWL popup menu. The popup menu will appear at the location of the red dot. You will then choose the submenu corresponding to the submenu number previously told to you by pointing to the submenu. A second popup menu will then appear next to the submenu item pointed to. You will then choose the item from the second popup menu corresponding to the item number previously told to you by pointing to the item and pressing either of the left wand buttons. This process will be repeated 18 times for different positions within the CAVE, submenu numbers, and item numbers.

Remember, you are free to withdraw from this study at any time for any reason.

Appendix C Task Sequence Performance Study Forms

C.1 Plan of Procedure

- Informed Consent Form
- Background Survey
- Written Instructions
- Object Selection Portion (EV0)
    - Equip
    - Show designated spot
    - Start EV0
    - "**Notice the white ray coming out of the wand device you are holding. This is the wandray. The wandray is directly controlled by the wand device. You can use this wandray to point at and select objects by clicking the lower left button on the wand device.**

      Point to the pink chair on your left with the wandray and click the lower left button on the wand device. You just selected this object. For this portion of the study, when you select an object, it disappears and a new object appears within the environment. Notice the new pink chair to your diagonal left. You will do this for several various objects in several different locations. When no more objects appear, you have finished this portion. Please complete this as quickly and efficiently as possible. You may begin."
    - Check and save log file.

- Simple Task Structures (EVA & EVB)
    - Reshow designated spot
    - "For the next two portions of the study, please stand in this spot with your hands down by your side before starting each task. You are asked to start each task looking directly ahead at the front CAVE wall."
    - Start EV(?)

- "**Notice the window to your diagonal left. This is a VEWL window. The six boxes on it are buttons. Notice when you point the wandray at the VEWL window, it changes to a mouse pointer. You can use this pointer to click on the buttons by clicking the lower left button on the wand device. I am now going to show you how to remove, size down, copy, hang painting, place rug, and paste.**"
- Show how to remove, size down, copy, hang painting, place rug, and paste.
- "**I will now restart the program and I will give you tasks to complete. Please stand in the designated spot with your hands down by your side and looking ahead until I have finished giving you the task and I have said "GO!" Remember to complete these tasks as quickly and efficiently as possible.**"
- Run EV(?)
- Check and save log file.
- Repeat last 6 steps for other evaluation.

- Break (5 minutes)
  - Unequip
  - Equip
  - Reshow designated spot

- Complex Task Structures (EVC, EVD, EVE, EVF, EVG, EVH, & EVI)
  - Start EV(?)
  - "**Notice the window to your diagonal left. This is a VEWL window. The six boxes on it are buttons. Notice when you point the wandray at the VEWL window, it changes to a mouse pointer. You can use this pointer to click on the buttons by clicking the lower left button on the wand device. I am now going to show you how to repattern, rotate, showcase, paint, place furniture, and hang object.**"
  - Show how to repattern, rotate, showcase, paint, place furniture, and hang object.
  - "**I will now restart the program and I will give you tasks to complete. Please stand in the designated spot with your hands down by your side and looking ahead until I have finished giving you the task and I have said "GO!" Remember to complete these tasks as quickly and efficiently as possible.**"

- Run EV(?)
- Check and save log file.
- Repeat last 6 steps for other evaluations.

- Exit Survey

C.2 Background Survey

Please help us to categorize our user population by answering the following questions.

1. What is your gender?

() Male () Female

2. How old are you?

3. Are you wearing glasses or contact lenses during the experiment?

() Glasses () Contact Lenses () No

4. Which hand will you use as your dominant hand during the experiment?

() Left () Right

5. What is your occupation? (if you're a student, indicate graduate or undergraduate)

6. If you indicated you're a student in question 5, what is your major?

7. You are familiar with computers.

() Strongly Disagree () Disagree () Neutral () Agree () Strongly Agree

8. How often do you use computers?

9. Have you ever used a virtual reality (VR) system? If so, please describe it.

C.3 Instructions

For this study, you will be asked to perform a set of tasks using VEWL (Virtual Environment Windowing Library) within the CAVE™. Specifically, your role in this experiment is that of evaluator of various task structures. We are not evaluating you. All information that you help us attain will remain anonymous. The time you take to do each task and other aspects of your interaction with the system will be measured.

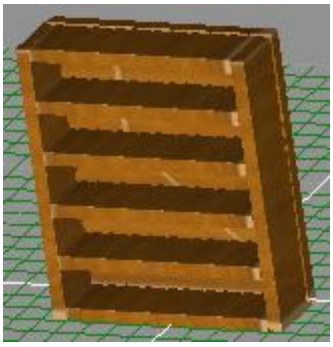
The session has three parts: gathering baseline performance data for selecting various objects, gathering performance data for completing simple tasks, and gathering performance data for completing complex tasks. For all three parts, you will be asked to stand in a designated spot within the CAVE with your hands down by your side before starting each task. You will also be asked to start each task looking directly ahead at the front CAVE wall. You will be asked to complete each task as effectively and quickly as you can.

The first part of the session, gathering baseline performance data for selecting various objects, will involve the usage of a "wandray". This wandray is directly controlled by the wand device you will be given. You can use this wandray to point at and select objects by clicking a wand button. Various objects will appear one at a time in an empty environment. After you have selected an object, that object will disappear and a new one will appear at a different location within the environment. You will do this for several different objects in several different locations.

The second part of the session, gathering performance data for completing simple tasks, will involve the usage of the wandray and VEWL. Using a mouse pointer, you will interact with VEWL. This mouse pointer, like the wandray, is directly controlled by the wand device. This session part will involve you completing six different simple tasks as they are verbally given to

you by the experimenter. The six different simple tasks for this part include: remove, size down, copy, hang painting, place rug, and paste. Remove will cause a selected object to be removed from the environment. Size down will cause a selected object to be scaled down to half its original size. Copy will copy a selected object for later pasting. Hang painting will cause a painting to appear at the selected location on a wall. Place rug will cause a rug to appear at the selected location on the floor. Paste will cause a previous copy to be placed at the selected location if valid.

The third part of the session, gathering performance data for completing complex tasks, will involve the usage of the wandray and VEWL also. This part will involve you completing six different complex tasks as they are verbally given to you by the experimenter. The six complex tasks include: repattern, rotate, showcase, paint, place furniture, and hang object. Repattern will cause a selected chair or sofa to be changed to a new pattern. Possible patterns include blue, pink, or stripe. Rotate will cause a selected object to be rotated to the left by a certain number of degrees. Possible degrees include 30, 45, and 90. Showcase will cause the items of a selected bookcase to change. Possible items include plates, books, and trophies. Paint will cause a selected wall's color to be changed. Colors for painting include red, blue, and green. Place furniture will cause a piece of furniture to be placed at a selected location on the floor. Possible furniture includes a chair, sofa, or bookcase. Hang object will cause an object to be placed at a selected location on a wall. Possible objects to hang include a painting, dartboard, or poster.
Following are pictures of items you will see during the session:



BOOKCASE

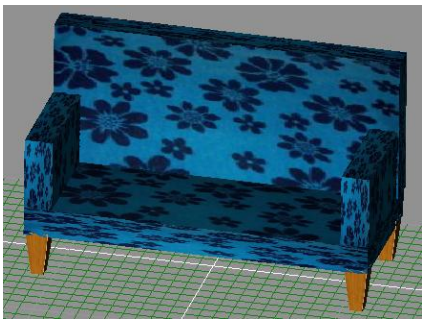
PAINTING

DARTBOARD

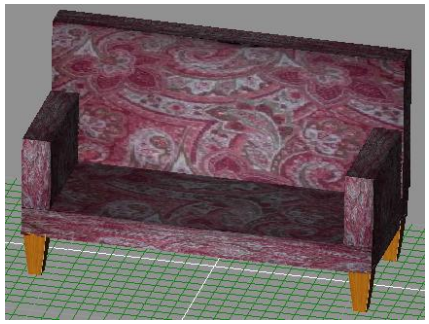
BLUE CHAIR

PINK CHAIR

STRIPED CHAIR

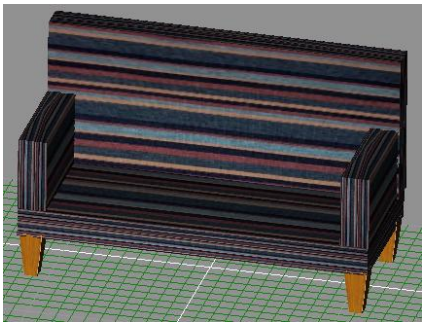


BLUE SOFA

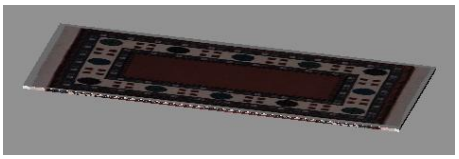


PINK SOFA

POSTER



STRIPED SOFA



RUG

Remember, you are free to withdraw from this study at any time for any reason.

C.3 Evaluation Tasks

EVA

1. Remove the blue sofa. **GO!**
2. Size down the striped chair. **GO!**
3. Copy the blue chair on the right. **GO!**
4. Paste the copy between the other two blue chairs. **GO!**
5. Place a rug in front of the blue chairs. **GO!**
6. Place a painting on the wall behind the blue chairs. **GO!**
7. Size down the bookcase. **GO!**
8. Copy the blue chair in the middle. **GO!**
9. Paste the copy in the right corner of the room. **GO!**
10. Remove the rug. **GO!**
11. Place a painting above the striped chair. **GO!**
12. Place a rug in front of the blue chairs. **GO!**

EVB

1. The blue sofa, copy it. **GO!**
2. The blue sofa, size it down. **GO!**
3. The blue sofa, remove it. **GO!**
4. Between the blue chairs, paste the copy. **GO!**
5. In front of the blue sofa, place a rug. **GO!**
6. On the wall behind the blue sofa, place a painting. **GO!**
7. The rug, remove it. **GO!**
8. In front of the blue sofa, place a rug. **GO!**
9. The blue chair on the left, copy it. **GO!**
10. The blue chair on the left, size it down. **GO!**
11. In the right corner of the room, paste the copy. **GO!**
12. To the left of the painting, place another painting. **GO!**

EVC

1. Repattern with blue material the pink sofa. **GO!**
2. Rotate 45o the pink chair. **GO!**
3. Showcase trophies in the bookcase. **GO!**
4. Paint with blue the wall in front of you. **GO!**
5. Place a chair to the right of the blue sofa. **GO!**
6. Hang a poster to the right of the bookcase. **GO!**
7. Repattern with blue material the pink chair. **GO!**
8. Showcase plates in the bookcase. **GO!**
9. Paint with red the blue wall. **GO!**
10. Place a sofa under the painting. **GO!**
11. Rotate 90o the blue sofa under the painting. **GO!**
12. Hang a painting to the left of the painting already in the room. **GO!**

EVD

1. Place under the painting a sofa. **GO!**
2. Rotate the blue sofa 90o. **GO!**
3. Repattern the blue sofa with pink material. **GO!**
4. Showcase in the bookcase plates. **GO!**
5. Paint the wall in front of you with green. **GO!**
6. Paint the green wall with red. **GO!**
7. Hang to the right of the bookcase a dartboard. **GO!**
8. Hang on the left wall a painting. **GO!**
9. Place to the right of the pink sofa in front of you a chair. **GO!**
10. Repattern the blue chair with pink material. **GO!**
11. Rotate the pink chair on the right 45o. **GO!**
12. Showcase in the bookcase books. **GO!**

EVE

1. To the right of the bookcase, place another bookcase. **GO!**
2. Under the painting, place another bookcase. **GO!**
3. The pink sofa, repattern with striped material. **GO!**
4. The bookcase under the painting, rotate 90o. **GO!**
5. The pink chair, repattern with striped material. **GO!**
6. The striped sofa, rotate 30o. **GO!**
7. The wall to your left, paint blue. **GO!**
8. The wall to your right, paint green. **GO!**
9. In the bookcase, next to the right wall, showcase trophies. **GO!**
10. On the wall to your left, hang a poster. **GO!**
11. In the empty bookcase, showcase books. **GO!**
12. On the wall to your right, hang a poster. **GO!**

EVF

1. Repattern with blue material the pink chair on the right. **GO!**
2. Rotate 45o the blue chair. **GO!**
3. Showcase trophies in the bookcase. **GO!**
4. Showcase books in the bookcase. **GO!**
5. Rotate 45o the blue chair. **GO!**
6. Repattern with blue material the pink chair. **GO!**

EVG

1. Repattern the pink sofa with striped material. **GO!**
2. Repattern the pink chair on the right with striped material. **GO!**
3. Showcase in the bookcase plates. **GO!**
4. Rotate the pink chair 45o. **GO!**
5. Rotate the striped sofa 90o. **GO!**
6. Showcase in the bookcase trophies. **GO!**

EVH

1. In the bookcase, showcase plates. **GO!**
2. The pink sofa, rotate 90o. **GO!**
3. The pink sofa, repattern with blue material. **GO!**
4. In the bookcase, showcase books. **GO!**
5. The blue sofa, rotate 90o. **GO!**
6. The blue sofa, repattern with pink material. **GO!**

EVI

1. The pink sofa, rotate 30o. **GO!**
2. In the bookcase, showcase trophies. **GO!**
3. The pink sofa, repattern with striped material. **GO!**
4. The pink chair on the left, repattern with striped material. **GO!**
5. The striped chair, rotate 45o. **GO!**
6. In the bookcase, showcase plates. **GO!**

C.4 Exit Survey

Please answer the following questions to help us understand your evaluation of the various task structures you used.

1. For the non-parametric task structures, which task structure was more intuitive for you?

   () Object-Verb () Verb-Object () Neither () Both

2. For the parametric task structures, which task structure was more intuitive for you?

   () Object-Verb-Parameter () Verb-Parameter-Object () Verb-Object-Parameter

   () None of these () All of these

3. For the parametric combination task structures, which task structure was more intuitive for you?

() Object + Verb – Parameter () Object – Verb + Parameter

() Verb + Parameter – Object () Verb – Object + Parameter

() None of these () All of these

4. Were any of the following difficult to understand or execute?

() Object-Verb () Verb-Object

() Object-Verb-Parameter () Verb-Parameter-Object () Verb-Object-Parameter

() Object + Verb – Parameter () Object – Verb + Parameter

() Verb + Parameter – Object () Verb – Object + Parameter

5. Please list any comments you have about the tasks or the session here:

Appendix D Task Sequence Preference Study Forms

D.1 Plan of Procedure

Session 1

Have user sign receipt for compensation.

1. Show the user how to navigate around an empty room.
2. Show the user how to use the wandray and VEWL.
3. Show the user how to place objects: (alternate orderings)
    a. Verb-Parameter-Object: Bed (Queen / Style 4 / Black / any)
    b. Verb-Object-Parameter: Painting (Large / Brown / Dali)
    c. Object-Verb-Parameter: Desk (Style 2 / Black / Desktop)
4. Show the user how to move objects: (alternate orderings)
    a. Verb-Object: Bed
    b. Object-Verb: Painting
5. Show the user how to rotate objects: (alternate orderings)
    a. Verb-Object: Desk
    b. Object-Verb: Bed
6. Show the user how to copy and paste: (alternate orderings)
    a. Verb-Object: Painting
    b. Object-Verb: Bed
7. Show the user how to cut and paste: (alternate orderings)
    a. Verb-Object: Desk
    b. Object-Verb: Bed
8. Show the user how to edit objects: (alternate orderings)
    a. Verb-Parameter-Object: Bed Comforter (any)
    b. Verb-Object-Parameter: Painting Artwork (Monet)
    c. Object-Verb-Parameter: Desk Content (Laptop)
9. Show the user how to save: (alternate orderings)
    a. Verb-Object: Slot 1
    b. Object-Verb: Slot 2

10. Show the user how to load: (alternate orderings)
    a. Verb-Object: Slot 3
    b. Object-Verb: Slot 4
11. Have the user show 3 ways to place a chair.
12. Have the user show 2 ways to move a chair.
13. Have the user show 2 ways to rotate a chair.
14. Have the user show 2 ways to copy a chair.
15. Have the user show 2 ways to cut a chair.
16. Have the user show 2 ways to paste a chair.
17. Have the user show 3 ways to edit a chair's pattern.
18. Have the user show 2 ways to save.
19. Have the user show 2 ways to load.
20. ***Have the user explore iDave and create a room design with the remaining time. (Make observations)

*** - Portions of study that user preferences will be recorded.

Session 2 (alternate step 1 and 3 for counterbalance)

Have user sign receipt for compensation.

1. Give the user a sample room design to replicate using a preloaded environment and only Verb-Object task structures. Replication suggests using each command twice as follows:
    a. Place – object not already in room (V-P-O)
    b. Place – object not already in room (V-O-P)
    c. Move – object already in room but in wrong place
    d. Rotate – object already in room but in wrong orientation
    e. Copy & Paste – duplicate object already in room
    f. Cut – unneeded object already in room
    g. Edit – object already in room but needs one attribute changed (V-P-O)
    h. Edit – object already in room but needs one attribute changed (V-O-P)
    i. Save – explicit command after room has been replicated
    j. Load – explicit command after save has been performed

2. Interview the user about step 1.
3. Give the user a sample room design to replicate using a preloaded environment and only Object-Verb task structures. Replication suggests using each command twice as above.
4. Interview the user about step 3.
5. ***Have the user create a new room design with the remaining time.

Session 3 (alternate step 1 and 3 for counterbalance)

Have user sign receipt for compensation.

1. Give the user a sample room design to replicate using a preloaded environment and only Object-Verb task structures. Replication suggests using each command twice as above.
2. Interview the user about step 1.
3. Give the user a sample room design to replicate using a preloaded environment and only Verb-Object task structures. Replication suggests using each command twice as above.
4. Interview the user about step 2.
5. ***Have the user create a new room design with the remaining time.

Session 4

Have user sign receipt for compensation.

1. ***Give the user a sample room design to replicate using a preloaded environment and any task structures. Replication suggests using each command six times as above.
2. ***Have the user create a new room design with the remaining time.

Session 5

Have user sign receipt for compensation.

1. ***Have the user create a new room design with the time. Inform that the design will be entered into a contest, and the contest winner will receive $10.
2. Interview the user about satisfaction, choices, etc.

*** - Portions of study that user preferences will be recorded.

D.2 Background Survey

Please help us to categorize our user population by answering the following questions.

1. What is your gender?

() Male () Female

2. How old are you?

3. Are you wearing glasses or contact lenses during the experiment?

() Glasses () Contact Lenses () No

4. Which hand will you use as your dominant hand during the experiment?

() Left () Right

5. What is your occupation? (if you're a student, indicate graduate or undergraduate)

6. If you indicated you're a student in question 5, what is your major?

7. You are familiar with computers.

() Strongly Disagree () Disagree () Neutral () Agree () Strongly Agree

8. How often do you use computers?

9. Have you ever used a virtual reality (VR) system? If so, please describe it.

D.3 Step Interview

1. On a scale of 1-7 (1 being terrible and 7 being excellent), please rate your satisfaction of using the application and given task sequence for completing the task of replicating the given room.

2. Please explain your reasons for giving this rating.

3. Did you find any particular parts of the task difficult and why?

D.4 Final Interview

1. On a scale of 1-7 (1 being terrible and 7 being excellent), please rate your satisfaction of using the application for completing the task of creating your own room design.

2. Please explain your reasons for giving this rating.

3. In what situations do you prefer to use the Action-Object task sequence? Why?

4. In what situations do you prefer to use the Object-Action task sequence? Why?

5. List of questions from study.

D.5 Exit Survey

Please answer the following questions to help us understand your evaluation of the various task structures you used during the entire experiment.

1. Please rate your own usage of the interior design application after session 1.

() Very Poor () Poor () Moderate () Good () Very Good

2. Please rate your own usage of the interior design application after session 2.

() Very Poor () Poor () Moderate () Good () Very Good

3. Please rate your own usage of the interior design application after session 3.

() Very Poor () Poor () Moderate () Good () Very Good

4. Please rate your own usage of the interior design application after session 4.

() Very Poor () Poor () Moderate () Good () Very Good

5. Please rate your own usage of the interior design application now.

() Very Poor () Poor () Moderate () Good () Very Good

6. Please list any comments you have about the experiment or any particular session here:

Appendix E IRB Approval Forms

E.1 IRB Approval of VEWL Performance Study

Virginia Tech
VIRGINIA POLYTECHNIC INSTITUTE
AND STATE UNIVERSITY

Institutional Review Board

Dr. David M. Moore
IRB (Human Subjects) Chair
Assistant Vice President for Research Compliance
1880 Pratt Drive, Suite 2006(0497), Blacksburg, VA 24061
Office: 540/231-4991; FAX: 540/231-0959
email: moored@vt.edu

DATE: September 27, 2005

MEMORANDUM

TO: Doug A. Bowman Computer Science 0106
Ryan McMahan

FROM: David Moore

SUBJECT: **IRB Expedited Approval:** "Baseline Performance of Users Using VEWL"
IRB # 05-580

This memo is regarding the above-mentioned protocol. The proposed research is eligible for expedited review according to the specifications authorized by 45 CFR 46.110 and 21 CFR 56.110. As Chair of the Virginia Tech Institutional Review Board, I have granted approval to the study for a period of 12 months, effective September 27, 2005.

Virginia Tech has an approved Federal Wide Assurance (FWA00000572, exp. 7/20/07) on file with OHRP, and its IRB Registration Number is IRB00000667.

E.2 IRB Approval of Task Sequence Performance Study

VirginiaTech

Office of Research Compliance
Institutional Review Board
1880 Pratt Drive (0497)
Blacksburg, Virginia 24061
540/231-4991 Fax: 540/231-0959
E-mail: moored@vt.edu
www.irb.vt.edu
FWA00000572(expires 7/20/07)
IRB # is IRB00000667.

DATE: April 11, 2006

MEMORANDUM

TO: Doug A. Bowman
Ryan McMahan

Approval date: 4/10/2006
Continuing Review Due Date:3/26/2007
Expiration Date: 4/9/2007

FROM: David M. Moore

SUBJECT: **IRB Expedited Approval:** "Exploring Object-Verb Vs. Verb-Object Task Performances" , IRB # 06-247

This memo is regarding the above-mentioned protocol. The proposed research is eligible for expedited review according to the specifications authorized by 45 CFR 46.110 and 21 CFR 56.110. As Chair of the Virginia Tech Institutional Review Board, I have granted approval to the study for a period of 12 months, effective April 10, 2006.

As an investigator of human subjects, your responsibilities include the following:

1. Report promptly proposed changes in previously approved human subject research activities to the IRB, including changes to your study forms, procedures and investigators, regardless of how minor. The proposed changes must not be initiated without IRB review and approval, except where necessary to eliminate apparent immediate hazards to the subjects.
2. Report promptly to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.
3. Report promptly to the IRB of the study's closing (i.e., data collecting and data analysis complete at Virginia Tech). If the study is to continue past the expiration date (listed above), investigators must submit a request for continuing review prior to the continuing review due date (listed above). It is the researcher's responsibility to obtained re-approval from the IRB before the study's expiration date.
4. If re-approval is not obtained (unless the study has been reported to the IRB as closed) prior to the expiration date, all activities involving human subjects and data analysis must cease immediately, except where necessary to eliminate apparent immediate hazards to the subjects.

Important:
If you are conducting **federally funded non-exempt research**, this approval letter must state that the IRB has compared the OSP grant application and IRB application and found the documents to be consistent. Otherwise, this approval letter is invalid for OSP to release funds. Visit our website at http://www.irb.vt.edu/pages/newstudy.htm#OSP for further information.

cc: File
Department Reviewer:Doug A. Bowman

Invent the Future

E.3 IRB Approval of Task Sequence Preference Study

VirginiaTech

Office of Research Compliance
Institutional Review Board
1880 Pratt Drive (0497)
Blacksburg, Virginia 24061
540/231-4991 Fax: 540/231-0959
E-mail: moored@vt.edu
www.irb.vt.edu

FWA00000572(expires 7/20/07)
IRB # is IRB00000667.

DATE: July 7, 2006

MEMORANDUM

TO: Doug A. Bowman
Ryan McMahan
Farid Sultani

Approval date: 7/6/2006
Continuing Review Due Date:6/21/2007
Expiration Date: 7/5/2007

FROM: David M. Moore

SUBJECT: **IRB Expedited Approval:** "Exploring Task Structure Preferences for 3D Virtual Environments" , IRB # 06-383

This memo is regarding the above-mentioned protocol. The proposed research is eligible for expedited review according to the specifications authorized by 45 CFR 46.110 and 21 CFR 56.110. As Chair of the Virginia Tech Institutional Review Board, I have granted approval to the study for a period of 12 months, effective July 6, 2006.

As an investigator of human subjects, your responsibilities include the following:

1. Report promptly proposed changes in previously approved human subject research activities to the IRB, including changes to your study forms, procedures and investigators, regardless of how minor. The proposed changes must not be initiated without IRB review and approval, except where necessary to eliminate apparent immediate hazards to the subjects.
2. Report promptly to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.
3. Report promptly to the IRB of the study's closing (i.e., data collecting and data analysis complete at Virginia Tech). If the study is to continue past the expiration date (listed above), investigators must submit a request for continuing review prior to the continuing review due date (listed above). It is the researcher's responsibility to obtained re-approval from the IRB before the study's expiration date.
4. If re-approval is not obtained (unless the study has been reported to the IRB as closed) prior to the expiration date, all activities involving human subjects and data analysis must cease immediately, except where necessary to eliminate apparent immediate hazards to the subjects.

Important:
If you are conducting **federally funded non-exempt research**, this approval letter must state that the IRB has compared the OSP grant application and IRB application and found the documents to be consistent. Otherwise, this approval letter is invalid for OSP to release funds. Visit our website at http://www.irb.vt.edu/pages/newstudy.htm#OSP for further information.

cc: File
Department Reviewer:Doug A. Bowman

Invent the Future