Using virtual menus in a virtual environment

Richard H. Jacoby

Sterling Software
NASA Ames Research Center, m/s 262-6
Moffett Field, CA. 94035
rick@eos.arc.nasa.gov

Stephen R. Ellis

NASA Ames Research Center, m/s 262-2
Moffett Field, CA. 94035
(415) 604-6147
silly@eos.arc.nasa.gov

ABSTRACT

Virtual environment interfaces to computer programs in several diverse application areas are currently being developed. The users of virtual environments will require many different methods to interact with the environments and the objects in them. This paper reports on our use of virtual menus as a method of interacting with virtual environments. Several aspects of virtual environments make menu interactions different from interactions with conventional menus. We review the relevant aspects of conventional menus and virtual environments, in order to provide a frame of reference for the design of virtual menus. We discuss the features and interaction methodologies of two different versions of virtual menus which have been developed and used in our lab. We also examine the problems associated with our original version, and the enhancements incorporated into our current version.

1. INTRODUCTION

Virtual environments (VE), or virtual reality as it is often called, have been the subject of much research and development over the past five years. Currently VE interfaces to several diverse application areas are being developed. These application areas include scientific and medical visualization, teleoperation, telerobotics, and vehicle simulation [11]. Other areas are CAD [9], architecture [4], music [25], and software design [15].

Users of an application program with a VE interface will need controls through which to interact with the environment and the application program [5]. Visualization program users may want to select varying levels of detail; robotic teleoperators may want to turn force and torque displays on or off; and CAD designers may want to change modelling tools or switch rendering techniques. Many different kinds of interactive controls will undoubtedly be desired and developed. The menu is one control that has proven immensely useful in personal computers and workstations. Its 3-dimensional analogy, a virtual menu, has been implemented in the Virtual Interactive Environment Workstation (VIEW) laboratory at NASA Ames Research Center (ARC).

The VIEW lab has created several computer programs for the purpose of demonstrating the utility of VEs. Three of these programs use virtual menus for some aspects of control. To our knowledge, no other lab has used a virtual menu for any type of control. Recently we created a telerobotic application, utilizing a VE interface, as a platform for testing force and torque displays. This application has also become a testbed for improving our virtual menus. A second version of virtual menus that addresses some of the short-comings of the initial version is now in use in the lab. A third version of the menus, with several enhancements over the second version, has been designed for use in the virtual windtunnel project at ARC [6].

There has been much study of conventional workstation menus, and their use is fairly well understood [1, 2, 8, 14, 16, 18, 22, 23, 24, 26]. While virtual menus have a lot in common with conventional menus, they also have several unique aspects that present new challenges in their design and use. Menu usage in a VE will be affected by both the user's position in the environment and the menu's placement, as well as by the user/menu interaction methods. The limitations of the interface technology, such as tracker noise, tracker response time, and viewer resolution, lead to additional considerations in the design of virtual menus.

2. BACKGROUND

This section discusses several aspects of both menus and VEs in order to provide a frame of reference for the design of virtual menus, and for the description of the particular design choices that have been made in the VIEW lab.

2.1. Menus

A computer menu is a collection of items, usually options and commands, that is displayed to a user and from which the user can interactively select an item [13].

Until the late 1960s, interactive computer sessions were typically carried out at a Teletype. The menu was printed at the user's Teletype printer, and the user made a selection by typing a character or keyword. Today most interactive computer sessions are held at workstations or personal computers where the user employs a mouse to invoke, and interact with, a menu. Although the mouse is the most common interaction device, other devices are sometimes used. These devices include keyboards, light pens, touch sensitive screens, speech recognizers, joysticks with buttons, six degree-of-freedom pointing devices with buttons, and electronic gloves with hand trackers. Each device offers specific advantages in particular applications and hardware/software configurations. The choice of a glove and tracker for menu interaction in the VIEW lab will be discussed below.

Typically, a menu is a sequential list of text items arranged in a rectangular format. In addition to, or instead of text, icons or other graphics may be used [2]. Instead of a rectangular format, other formats, such as semi-circular or circular, have been tried [7].

Menus are useful in a variety of interactive situations. Since menus display available options simultaneously, they reduce the number of options a user needs to remember. By limiting the options to a well-defined set, they help the user make an appropriate selection [21]. By visually presenting the options in a consistent manner, they facilitate learning the interaction [10].

Menus inform the user, in addition to accepting the user's input. Displaying the available options tells the user something about the application. A context sensitive menu is very informative, since its text can change over time as the system state changes. The new information provides the user with feedback about previous operations [19].

One or two character keystroke shortcuts are often provided to accomplish the same actions as some of the menu items, so that a menu and its shortcuts provide redundant methods of interaction [20]. Menus are useful, of course, when there are not enough shortcuts available to cover all the commands.

Menus can be structured in several ways. For a simple application, one menu with all the available options may suffice. However, it may be easier to use a menu bar with several menus, where each menu has a group of commands associated with a particular aspect of the application. Since each menu deals with a particular aspect of the application, the menu structure helps the user to learn the application more quickly, and helps organize her interactions with it [23]. Hierarchical menus are useful when there are more related options than will fit on a single menu. Implementing hierarchical menus should be done with caution however, since poorly designed hierarchical menus may confuse the user [18]. It will also take longer for a user to access an item from deep within a hierarchy than it will from a single menu [2, 18].

Two popular styles of menus are pop-up menus and pull-down menus. Pull-down menus are "pulled-down" by the user from a menu bar. The menu bar, which contains the menu titles, serves as a visual reminder of the types of operations available to the user [16], and as an element of familiarity in the environment [2]. The menus appear at standard locations (under the menu title in the menu bar) and interactions can be accomplished with a pointing device and button. After the interaction, the menu usually disappears, although some applications allow a pull-down menu to be "torn-off" and remain visible for easier future use.

Pop-up menus are "popped-up" by the user at the current cursor location. With pop-up menus, there are no menu bars containing individual menus titles, so if there is more than one menu available, additional input is needed to determine which menu to activate. One determination scheme is to use a main menu with hierarchical branching to secondary menus. Other schemes use the cursor position, a second mouse button, or a keyboard character to determine which menu to activate. Interactions with most pop-up menu implementations use a pointer and button, with additional discriminating information provided by a keystroke, situation context, or current cursor position. Like pull-down menus, pop-up menus can disappear after an interaction, or be made to remain for future interactions.

The fact that menus can disappear makes them useful in reducing workspace clutter. Some menu implementations allow menus to be 'torn-off" (i.e.: moved to another location and made to remain visible). The aspect of remaining visible makes the menus quicker to use.

2.2. Virtual environments

The Virtual Interactive Environment Workstation (VIEW) lab at NASA Ames Research Center has been in existence since 1984. The initial work centered around developing a general purpose VE facility with many capabilities [12]. In late 1989 the initial operating configuration (IOC) was completed. After IOC, work shifted toward using the facility to run experiments and examine engineering issues related to improving specific aspects of VE [3, 17].

One of our areas of concentration has been the improvement of robotic teleoperation through the use of a VE. To this end, a virtual environment telerobotic application, *Teleop*, was created. The application runs alone or connected to a real PUMA robot arm. There are several aspects of the program that the operator needs to control including engaging and disengaging control of the virtual and real robots, displaying or hiding a force and torque display, executing robot macro instructions, changing levels of information sent from the robot, and enabling or disabling “flying” of the user's viewpoint. These and other options are controlled by the user via menus from within the VE. Virtual menus have also been used to control options in other smaller demonstration programs in the lab.

The current configuration of the VIEW hardware uses an HP 9000/835 as a host and an ISG Technologies computer as the graphics processor. Peripherals attached to the host include a VPL DataGlove, a Polhemus six degree-of-freedom tracking source and two sensors, a Convolvotron sound localizer, and a VocaLink voice recognizer. Output to the user is provided through a pair of stereo head-mounted, monochrome liquid crystal displays, Dectalk voice synthesizer, audio headphones and room speakers.

We have defined eleven standard hand gestures in our system for use with the VPL DataGlove. All make use of four fingers and do not consider thumb position. Five of these gestures are used with menus. These five are: NULLGESTURE, FIST, SINGLEPOINT, DOUBLEPOINT, and TRIPLEPOINT. The SINGLEPOINT is a fist with the index finger extended (as if pointing with it). DOUBLEPOINT and TRIPLEPOINT positions are like a SINGLEPOINT with the additional extension of the middle finger, and middle and ring fingers respectively. The NULLGESTURE position has the fingers slightly curled (i.e., relaxed) so as not to make any of the other gestures.

Three things need to be considered when using gestures to interact with a computer. First, can the user easily make the required gesture? Second, how well can the computer distinguish between the different gestures, and last, is the required gesture appropriate for the action it is invoking? The five gestures used for menu interactions have been observed to be relatively easy for a user to make. Further work is necessary to ascertain how well the computer can distinguish between them, and some planning for this evaluation has begun. Two of the gestures seem very appropriate for their actions, i.e.: FIST for grabbing the menu, and SINGLEPOINT for selecting an item. The appropriateness of the remaining gestures is difficult to judge.

3. DISCUSSION

Two factors must be considered when designing a menu system: readability of menus, and ease of use. These factors take on several aspects in a VE, which we discuss below.

3.1. Menus within a virtual environment

Menus in a VE differ from conventional workstation menus in several ways. Perhaps the most significant difference is that conventional menus appear on the computer screen facing the user from about two feet away, while VE menus exist in 3-dimensional space and may be at any position and orientation. Also, the user is immersed in the same 3-dimensional space and can have an arbitrary position and orientation. This implies that in a VE the user may have trouble seeing or reading the menu—objects in the VE may appear between the user and menu, or the user may be at a bad angle, too far, or to near to read the menu easily. If the menu is too near, the user will need to turn her head in order to see all the items.

Menus in a VE are not restricted to being flat 2-dimensional objects; they may also be 3-dimensional objects with depth. They can be thin, like a menu printed on cardboard, or any arbitrary shape, such as a cube with menus on all sides.

The mode of interaction is another difference between virtual and conventional menus. While interaction with a conventional menu is done with a mouse or keyboard, these devices may be unavailable or awkward for the VE user, or there may be a more

desirable method of interaction. Menu interactions in a VE may be more effective with an electronic glove and hand tracker, or with a different pointing device and buttons.

3.2. Limitation of the technology

Limitations of the current VE technology contribute additional constraints to menu design. These limitations are viewer resolution, viewer field-of-view (FOV), electromagnetic tracker noise, tracker lag, display update rate, and glove calibration accuracy. The current head-mounted LCD displays have less than normal FOV and low resolution. The VIEW lab viewer has a FOV of 120 degrees (compared to 180 for average human vision) and a resolution of 320 x 220 in monochrome. While viewers are improving with technology, these performance specifications are fairly typical for current low-cost state-of-the-practice equipment. An implication of low resolution is that text must be fairly large to be readable. An implication of the FOV is that a menu may take up a large portion of the user's view.

Tracker noise increases as the distance between the source and sensor increase. The signal from the Polhemus tracker we currently use becomes noisy at approximately 1.5 meters from the source. This makes pointing at menu items difficult. Viewer and CRT electronics, when close to the tracker's sensor, also add appreciable signal noise.

Head and hand tracker lag time and slow graphic update rates of the displays tend to make menu interactions sluggish. A typical update rate of many VE systems is about 10 Hz with tracker lag of a quarter of a second or more. Sluggish update rates make interacting with the menus difficult.

The DataGlove can be calibrated for each user's hand in less than a minute. A problem arises, however, as the fibers and sensors of a glove finger age and lose their dynamic range. This makes it difficult for the computer program to recognize how much a finger is bent, and therefore difficult to distinguish between gestures.

Another consideration of menu design is related to fatigue: will repeated use of the menus cause unnecessary eyestrain and physical tiring of the hands, arms or head?

3.3. Menu paradigms

Menu behavior can be designed around a variety of different paradigms. The selection of a particular paradigm will, of course, affect the way a user interacts with the menu. To help examine the interaction paradigms that follow, several aspects of menu behavior are listed here.

Invocation:	What actions are necessary to make a menu appear and be ready for use?
Location:	Where does a menu appear?
Reference Frame:	When a menu appears, is it fixed in space or in field-of-view (or neither), and how is it moved?
Cursor:	Is there a cursor or other aid to help the user with pointing?
Highlighting:	How are the different items highlighted?
Selection:	Once highlighted, how is an item chosen?
Removal:	Does the menu go away after selection or can it be made to remain visible?

4. VIEW MENUS

Teleop, the robot teleoperation application, has been the VIEW lab's primary testbed for different menu implementations. This application has been used by the first author hundreds of times and by others several dozen times over the last year and a half. Virtual menus have also been part of demonstration applications in the VIEW lab for about three years. During that time, well over two hundred people have tried these programs and experienced VE. During the development, demonstration and use of these applications, we have had the opportunity to observe the interactions between users and our menus. The results of informal observation and discussion have lead us to identify weaknesses in the initial version of the menus, and to develop a newer version that addresses those weaknesses.

Until this fall, the interactions with the VIEW menus have been performed entirely through hand gestures and pointing. This has been due, in part, to our hardware configuration: an electronic glove and tracker that the user already wears to interact with the environment in other ways. In addition, we wanted to evaluate the premise that gesturing is a natural and efficient method of interacting with the computer.

The VIEW menus are text based. They are very thin rectangular 3-dimensional objects that can be displayed at any position and orientation in the environment. Highlighting is accomplished by placing a dimly lit rectangle between the menu and the highlighted text. Menu interactions are initiated by the user making a specific hand gesture. Before the menu appears, the user sees a model of a hand and fingers at the position and orientation of his real hand and fingers. In the newer version of the menus, a pointer and cursor also become visible.

4.1. Initial menu system

The initial version of the menus used four different hand gestures: TRIPLEPOINT to invoke a menu, NULLGESTURE to highlight an item, SINGLEPOINT to select an item, and FIST to move the menu. When the user wanted to interact with the menu, she would make a TRIPLEPOINT. The menu would then appear facing the user's head, centered around the user's fingers, but a few inches further away. Once the menu appeared, the user would relax her hand into the NULLGESTURE position. To highlight an item, the user would move her hand up or down to be in proximity of the desired item. The computer would calculate a small box around the user's hand, and if that box intersected a menu item, the item would be highlighted. To select an item, the user would make a SINGLEPOINT while the item was highlighted. Depending on which option the programmer had implemented for that item, the menu would stay visible or disappear after the selection was made. To move a menu, the user would highlight any item, make a FIST, move the fist (and the menu) to a new position and orientation, and then relax the hand into the NULLGESTURE position. The menu would stay in the new position and orientation. This scheme is similar in style to that of a pop-up menu in that the menu appears at the user's hand location, analogous to a pop-up menu appearing at the cursor location. This scheme appears to be an easy to learn and natural method for interaction with menus. In practice, we found that although the scheme was a useable system, the novice user had a significant amount of difficulty with it. The experienced user, while having less difficulty than the novice, also had some problems with it.

One of the biggest problems with this implementation was insufficient feedback to the user. Because the user could not the see the computed box around the hand she could not tell when an item was going to be highlighted. Also, when looking at the menu from the edge and in low resolution stereo, it was difficult to judge the location of the hand relative to the menu. This resulted in the user perceiving the hand as being in the correct place to highlight a particular item, however the wrong item, or no item at all, would be highlighted. Thus the user had little feedback as to how to move the hand to correct this situation.

Another feedback related problem would occur when the user thought that she had correctly highlighted an item and made a SINGLEPOINT but the selection did not occur. The user had no information as to whether the difficulty was due to the computer misrecognizing the selection gesture, or a different problem. Another consequence of insufficient feedback occurred when the user would have the correct item highlighted, but the hand position was very close to highlighting another item. The user would make a SINGLEPOINT, but as the gesture was being made, either a slight hand movement or tracker noise would cause the highlight to move and the wrong item to be selected.

Additionally, working with one's arm fully extended is tiring and uncomfortable. After a short time a typical user tends to drop his arm and bring it in closer to his body. Insufficient arm extension seemed to be the root of additional problems with the original menu implementation. If a user's arm was not extended far enough, the menu would appear too close for comfortable reading, and also require the user to use head movements to look up and down in order to see the entire menu. Having to look up or down meant most items were viewed from their edge—contributing to the difficulty in reading. Interaction with the menu was made more difficult because the glove was either close to the face where the viewer electronics made the sensor output noisy, or because the user had to reach way up or down to where it was hard to see the highlighting and hard to judge the hand position.

Another problem caused by insufficient arm extension was that sometimes a user would extend the arm to invoke the menu and then retract the arm so that the hand was no longer close enough to the menu to make a selection. If the user then moved forward and re-extended the arm, his hand would sometimes pass through the menu so that he could not see his hand and consequently could not tell how to move it to highlight an item. Not being able to see the hand also led to confusion as to whether there was a problem with the hand tracker or with the user not positioning the hand properly.

Moving a menu posed an additional problem. In order to move the menu, one had to first highlight an item, which was sometimes difficult to do. If the user accidentally brought the menu up in a place where it was difficult to see, such as behind another object, it was then hard to move.

Finally, there was no way to remove a menu without selecting an item from it. Therefore, if the user accidently brought a menu up he had to select an innocuous item that would remove the menu when selected, or work with the menu remaining up.

4.2. Current Menu System

Several changes to the menu software were made to address the above shortcomings. The current menu system uses a total of three hand gestures. Interaction with a menu that is not visible requires two gestures: the DOUBLEPOINT to invoke the menu and highlight items, and the NULLGESTURE for selection. Menus that are already visible also require two gestures: the TRIPLEPOINT to highlight items, and the NULLGESTURE for selection. When a user wants to interact with a menu that is not visible, she invokes it with a DOUBLEPOINT. The menu appears as if it were one meter from, and facing, the user's head. The menu's center is positioned slightly to the right side of the FOV, with its top border slightly above the center of the FOV. While the user maintains the DOUBLEPOINT, a ray is projected from the hand position to the plane of the menu and a small clover-leaf cursor is displayed at the end of the ray on the plane of the menu (Figure 1). The cursor and ray move as the user moves her hand. The direction of the ray is determined by the direction in which the hand is pointing. If the cursor is moved off the menu and the user makes a NULLGESTURE (or any gesture other than a DOUBLEPOINT), the menu disappears. As the cursor is moved from item to item on the menu, those items are highlighted in turn. If the user makes a NULLGESTURE when an item is highlighted, that item is selected and the menu goes away. Because the menu appears at a fixed point in the user's FOV, this scheme resembles a pull-down menu.

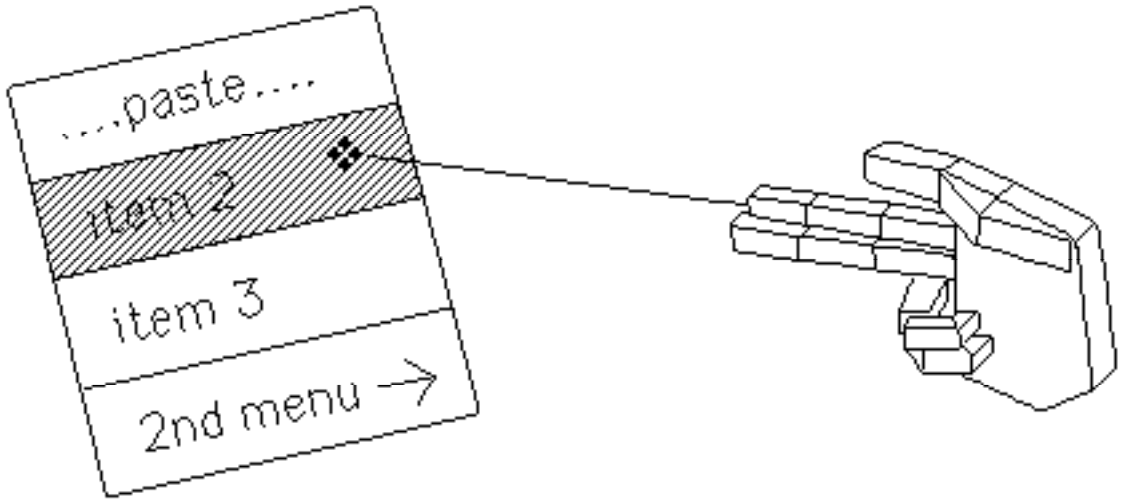

Figure 1. Hand and menu in virtual environment showing ray and cursor, menu items, hot area, and highlighting.

This current version also gives the programmer the option of making one of the menu items a paste item. (We use the top item and its text is "....Paste....".) When the paste item is selected, the menu stays pasted up in space for repeated use. To interact with a pasted-up menu, the user employs the TRIPLEPOINT to highlight an item, and the NULLGESTURE to select it. A ray and cursor are also present for the TRIPLEPOINT. To remove a pasted-up menu, the user selects the paste item again causing the menu to disappear.

The current version corrects many of the difficulties with the initial version. If the menu is invoked accidentally, the user points off the menu and releases the gesture. If the menu is invoked in the wrong place, the user points off the menu and releases the gesture, then re-invokes it at a better location. By being presented with the ray and cursor, the user receives visual feedback that the computer has recognized the gesture (DOUBLEPOINT or TRIPLEPOINT). The ray and cursor also give the user feedback as to which way to move the hand to highlight an item, and how close he is to highlighting a different item. The menu always appears at a comfortable reading distance where the user can see the entire menu without turning his head. The menu can be invoked with the arm and hand in any comfortable position. Since the cursor is positioned with a combination of pointing and moving of the hand, long reaches up or down are not needed, and movement of the hand into areas of high tracker noise can be minimized.

Hierarchical menus are possible with either the initial or current version of menus. In the initial version, the user had to select an item that would bring up a sub-menu, and then select an item from the sub-menu. A NULLGESTURE and SINGLEPOINT combination was used for both selections. The current version makes the use of hierarchical menus easier by implementing "hot areas". The programmer has the option of making either end of an item a hot area. We place an arrow ("-->" or "<--") in an item's text to indicate the hot area (Figure 2).. When the user moves the cursor into a hot area, the item is automatically selected. So far, the use of hot areas has been restricted to bringing up sub-menus. Selection from the sub-menu is then performed in the usual way, that is by changing from a DOUBLEPOINT or TRIPLEPOINT to NULLGESTURE.

By making a main menu that consists of menu titles and hot areas, we can simulate a pull-down menu bar (Figure 2). This allows the user to access any of several menus and interact with them by using only two gestures–the DOUBLEPOINT and NULLGESTURE.

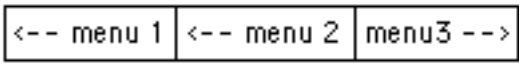

Figure 2. Simulated pull-down menu bar showing main menu with three sub-menu titles and hot areas.

5. FUTURE WORK

Although the current menu version is substantially easier to operate than its predecessor, many improvements to menu interactions can still be made.

One possible improvement to the current interaction method would be to make the cursor "sticky". Such a cursor would be free to move as it currently does, but an item would stay highlighted until the cursor is near the center of the next item. Only then would the next item be highlighted. This change could help prevent tracker noise and slight hand movements from highlighting the wrong item just before selection.

Another variation on the current paradigm is to not render the graphics model of the hand during menu interaction. Once the user makes the DOUBLEPOINT or TRIPLEPOINT gesture to interact with a menu, the hand graphic hand is not needed. Feedback of the gesture recognition would be provided by the presence of the menu, ray and cursor and by the absence of the hand model. Feedback for pointing would still be provided by the ray and cursor. An advantage may be a faster running simulation which would make pointing easier because of reduced transport delay.

A different interaction paradigm that has already been partially implemented entails the use of a two-button hand-grip that is held in the user's left (ungloved) hand. The buttons are used to invoke menus and cycle through menu items. It is hypothesized that this type of interaction may be quicker than pointing to a menu item. Another advantage of this paradigm is that the user's right (gloved) hand will be free for other uses. An experiment is being be designed to test the hypothesis.

Another interesting paradigm for menu interaction is the two-handed menu where the menu would appear and stay in one hand while interaction is carried out with the other, similar to the use of a hand-held calculator. This may offer some advantages in the positioning of a menu.

Further investigation is necessary to evaluate how the menu's frame of reference should be related to the user's frame of reference. Presently, menus do not move once they are invoked. This has an advantage in that the user can look around at the rest of the scene while the menu is up. It may be desirable, however, to keep the menu fixed in the user's FOV until a selection has been made or until she lets the menu go away. A mixed mode, where only orientation or position changes with the user's frame, may actually be desirable.

Work is also needed in the area of gesture recognition. A study will be performed to determine how well the computer recognizes gestures and how well it can distinguish between the user's gestures.

6. SUMMARY

We have found the use of virtual menus to be an effective method for interacting with a virtual environment. The initial implementation of a menu system allowed the user to control many aspects of the VE. It also provided us with the opportunity to observe user interactions with menus and identify problem areas. Our current implementation addresses many of those problems and appears to be much easier to use.

Feedback, showing the user the location of pointing and recognition of gestures, was implemented as a ray and cursor. This feedback significantly improved the use of virtual menus.

We were able to facilitate the use of the menus by incorporating two standard features of pull-down menus: the position of menus at a fixed location, and simulated menu bars. We have also placed menus at a comfortable reading and interaction distance. Additionally, we incorporated hot areas that allow quicker and easier interactions with hierarchical menus.

We have reduced the number of gestures needed to interact with a single menu from four to two. This simplified user interaction, and allowed the system to distinguish between gestures more easily.

We have incorporated pointing, rather than simple hand positioning, to move the cursor. This gives the user more flexibility in positioning the arm and hand resulting in less fatigue when interacting with the menus, and greater accuracy in positioning the cursor.

We are continuing to work on improving menu interactions, exploring new interaction methods, and testing the effectiveness of the implemented paradigms.

7. ACKNOWLEDGEMENTS

This research has been jointly funded by the Virtual Windtunnel project, branch RNR, and the Advanced Displays and Perception laboratory, branch FLM, both of NASA Ames Research Center.

The first author would like to thank Steve Bryson for his implementation of the initial menu system, Deborah Silver, and Dov Adelstein, for their encouragement and discussions about menus, and Robert F. Esswein for his discussions of the menu mathematics.

8. REFERENCES

1. Allen, R., Bailey, R., McIntyre, G., and Bozza, M. Placement of Menu Choices. In *Proceeding of the Human Factors Society 35th Annual Meeting*, (San Francisco, Ca., Sep. 1991). Human Factors Society, 1991, pp. 379-382.

2. Apple Human Interface Guidelines: The Apple Desktop Interface. Addison-Wesley, Reading, Ma, 1987.

3. Brody, A., Jacoby, R., and Ellis, S. Simulation of Extra-Vehicular (EVA) Self-Rescue. In Society of Automotive Engineers Report #911574, S. A. E., Warrendale, Pa., 1991.

4. Brooks, Jr., F. Walkthrough-a dynamic graphics system for simulating virtual buildings. In *ACM 1986 Workshop on Interactive Graphics*, (Chapel Hill, N. C., Oct. 1986).

5. Brooks, Jr., F. Grasping Reality Through Illusion-Interactive Graphics Serving Science. In *CHI '88 Conference Proceedings - Human Factors in Computing Systems*, (Washington, D.C., May 15-19, 1988). Addison-Wesley, 1988, pp. 1-11.

6. Bryson, S., and Levitt, C. The Virtual Windtunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flows. In *Proceedings of the IEEE Computer Society Visualization '91 Conference*, (San Diego, Ca., Oct. 21-25 1991).

7. Callahan, J., Hopkins, D., Weiser, M., and Schneiderman, B.. An Empirical Comparison of Pie vs. Linear Menus. In *CHI '88 Conference Proceedings - Human Factors in Computing Systems*, (Washington, D.C., May 15-19, 1988). Addison-Wesley, 1988, pp. 95-100.

8. Card, S. K., Moran, T. P., and Newell, A. The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, Hillsdale, N. J., 1983.

9. Carrabine, L. Plugging into the Computer to Sense. *Computer-Aided Engineering* (June 1990), pp. 19-26.

10. Elkerton, J., and Williges, R. C. "Dialog Design for Intelligent Interfaces" in *Intelligent Interfaces: Theory, Research an Design*, Hancock, P. A. and Chignell, M. H. (eds.), Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1989.

11. Ellis, S. Nature and Origins of Virtual Environments: A Bibliographical Essay. *Computer Systems and Engineering*, 2,4, pp. 321-347.

12. Fisher, S., McGreevy, M., Humphries, J., and Robinett, W. Virtual Environment Display System. In *Proceedings of the ACM 1986 Workshop on Interactive Graphics*, (Chapel Hill, N. C., Oct. 1986).

13. Freedman, A. The Computer Glossary : The Complete Illustrated Desk Reference. American Management Association, New York, 1991.

14. Hammond, M., and Barnard, P. "Dialogue Design: Characteristics of User Knowledge" in *Fundamentals of Human-Computer Interaction*, Monk, A. (ed.), Academic Press, Orlando, Fl, 1984.

15. Hirose, M., and Amari, H. A Study on Visualization of Control Software Design. In *MIT-JSME Workshop on Cooperative Product Development*, Nov. 1989).

16. NASA Menus Available at User Request. In Space Station Freedom Program Human-Computer Interface Guide, v. 2. 1. NASA Lyndon B. Johnson Space Center, Houston, Tex., 1988, pp. 3:20-3:23.

17. Nemire, K., Jacoby, R., and Ellis, S. A Comparison of Spatial Orientation in a Physical and a Virtual Environment (in preparation).

18. Norman, K. L. The Psychology of Menu Selection. Ablex, Norwood, N. J., 1991.

19. Perlman, G. "Making The Right Choices With menus" in *Human-Computer Interaction-interact '84*, Shackel, B. (ed.), Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1984.

20. Rawlins, M. G., and Uttley, V. "Communications and Graphics" in *Computer Graphics - Visual Technology and Art*, Kunii, T. l. (ed.), Springer-Verlag, Tokyo, Japan, 1985.

21. Reid, P. "Work Station Design" in *Fundamentals of Human-Computer Interaction*, Monk, A. (ed.), Academic Press, Orlando, Fl., 1984.

22. Schneiderman, B. Designing the User interface: strategies for effective human-computer interaction. Addison-Wesley, Reading, Ma., 1987.

23. Smith, S. L., and Mosier, J. N. Guidelines for Designing User Interface Software. In MITRE Corporation Report ESD-TR-86-278, MITRE Corporation, Bedford, Ma., 1986, pp. 231-247.

24. Somberg, B. L. A Comparison of Rule-Based and Positionally Constant Arrangements of Computer Menu Items. In *CHI+GI 1987 Conference Proceedings*, Carroll, J. M. and Tanner, P. P. (eds.), (Toronto, Canada, April, 1987). ACM, New York, N. Y., 1987, pp. 255-260.

25. Trubitt, D. R. Into New Worlds: Virtual Reality and the Electronic Musician. *Electronic Musician* (July 1990), pp. 30-40.

26. Walker, N., and Smelcer, J. A Comparison of Selection Times from Walking and Pull-Down Menus. In *CHI '90 Proceedings*, (Seattle, Wa., Apr. 1990). Addison-Wesley, Reading, Ma., 1990, pp. 221-225.