NPSNET: Four User Interface Paradigms for Entity Control in a Virtual World

David R. Pratt, John Locke, Paul Barham and John Falby

*Department of Computer Science
Naval Postgraduate School
Monterey, California, USA 93943
Telephone 408 656 2865
Email pratt@cs.nps.navy.mil*

Synopsis

The Department of Computer Science, Naval Postgraduate School, has been researching networked shared environments for the last four years. The NPSNET series, our current family of virtual worlds, emphasizes ground-based and low-flying networked entity simulation. We have developed four distinct user interface paradigms for NPSNET (keyboard with mouse, dial/button box, SpaceBall, stick and throttle), the choices governed by available equipment, the entity to be controlled, the purpose of the virtual world, sponsor request, and previous experience. Each paradigm has strengths and weaknesses for different purposes and entities. A wide spectrum of users has provided feedback on the various paradigms, under learning conditions ranging from training in our laboratory, minimal instruction at demonstrations like SIGGRAPH '91 and '93, and a user's guide provided with the software distribution. NPSNET has served as a networked game, surrogate crew station, analytical model display, out-the-window and sensor display, and command and control tool. Different users and operational platforms demand different interfaces. From these varied user experiences, we have developed anecdotal guidelines for developing virtual world interfaces. This paper examines the

fundamentals of entity control, the different interface paradigms and their impact on the utility of the simulation for different audiences and purposes, and our guidelines.

Keywords:

entity control, networked virtual environments, user interface, virtual environments

Introduction

We present our experience implementing four user interface paradigms in the various versions of NPSNET, a networked, shared virtual environment (Zyda *et al.*, 1992). We begin with an overview of NPSNET, followed by a description of the major user tasks it implements. Next we examine each of the four user interface paradigms employed: keyboard with mouse, dial/button box, SpaceBall, and flight stick and throttle. We then compare the different paradigms as implemented in NPSNET.

Overview of NPSNET

NPSNET was designed and built by the Department of Computer Science, Naval Postgraduate School, as a test-bed for the exploration of virtual worlds (VW) technology. Work on the system began in February 1990 as a means of exploring a proposed terrain database format. Since then, over ten postgraduate students have added to its development. What started as a simple system with a lone vehicle in a one kilometer square world, has now expanded to a fully networked system capable of supporting 500 vehicles of any of 2000 types in a world exceeding 100 kilometers on a side. As NPSNET has grown, it has been favorably compared to the U.S. Army's SIMNET system.

(Pope and Schaffer, 1991). While the systems are similar in many respects, there are fundamental differences. Foremost are the respective hardware platforms. NPSNET can run on an SGI Indigo Elan, while SIMNET requires a vehicle mock-up, computer image generator, host computer, and sound generator. SIMNET is also designed for specific weapon systems. A node configured to emulate an M1 tank cannot be reconfigured to an M2, the inflexibility owing to the use of the mock-up and the complexity of the simulator code. However, the mock-up adds a realism that NPSNET, a workstation-based system, cannot. NPSNET was designed to be as flexible and extensible as possible. To a large extent, this prevented the inclusion of more specific weapons systems features. However, since the system is extensible, we have incorporated some of these features into versions of NPSNET.

Rather than using an accurate vehicle mock-up, NPSNET users interact with the system by means of a SpaceBall, keyboard with mouse, dial/button box, and throttle and stick. A variety of controls lets the user select the most comfortable interface for the task at hand.

NPSNET was designed to run on a commercial, off-the-shelf workstation. As such, the available visual display was initially limited to a single display console. The current display is a combination of a 3-D Out-The-Window (OTW) view, a 2-D elevation shaded map, and an information panel. The size and location of the 3-D OTW view was determined by default, that is, the screen area available on the graphics display. Across the top of the screen is the information panel. This window contains textual information such as location, speed, orientation, frame rate, and weapon status. The third window is the optional 2-D elevation shaded display. It shows the vehicle's location in the world in relation to the other vehicles and the database, but can be hidden to expand the OTW view.

The user sits at the workstation console and perceives the NPSNET virtual world from the point of view of a vehicle entity, usually an airplane or tank. (A "stealth" mode allowing free-form

movement through the world is also available.) The user navigates the vehicle through the virtual world, either to explore the natural terrain and man-made structures like buildings and bridges, or to seek other entities that are moving around in the world.

Since the system is distributed, users at separate workstations can view the same virtual world. Network protocols ensure that consistency between separate views is maintained efficiently. Other entities that the user sees may be computer-generated on the same or another workstation, or navigated by human users at other workstations. Ideally, the computer-generated entities should be indistinguishable from human-controlled entities. Although the technology is inherently multi-purpose, our virtual world implementation is oriented toward military applications, thus the military entities. Accordingly, a primary user activity is to seek other entities and shoot at them, or conversely to evade enemy fire. NPSNET is also applicable in a broader military context, for instance, it addresses the need for command-level officers to rehearse their roles in a complex command-and-control infrastructure, and to do so in a model of a real environment where actual exercises or operations will take place.

Since the initial release of NPSNET in Fall 1991, over seventy schools, government organizations, defense contractors, and commercial firms have received at least one version of NPSNET or some of the support code. The uses for this code include autonomous agent studies, constructive combat model visualization, terrain visualization, weapons evaluation, and low-cost stealth capability. The flexibility and modularity of the code and the database allows a single system to be readily adapted for these manifold uses.

Tasks

A crucial consideration in any user interface is the purpose for which the system was designed. While it sounds trite, this fundamental principle is frequently shortchanged in VW systems in favor of

exhibiting novel aspects of the technology. To justify our user interface paradigms, we first discuss the major tasks performed within NPSNET: free flight, tether, and inspection.

Free Flight

Free flight is the most common task and thus a major consideration in interface design. It can be separated into two sub-tasks with a common basis, flying and driving. In both, the user controls his motion through virtual space and the monitor displays an OTW view. Six degrees of freedom (DOF), three for position and three for angular placement, are available in a single scale simulation. (Ware and Osborne, 1990). The difference between the sub-tasks lies in the constraining forces. While driving, the user is constrained to a fixed distance above the ground. The underlying terrain dictates entity pitch and roll, limiting user control of the eye point motion to three DOF, X, Y, heading. The other three – Z, pitch, roll – are computed from the underlying terrain. When flying, the user controls all six DOF. (For real world consistency, the user's Z value is bounded by the underlying terrain elevation.)

Free flight is most commonly used to simulate the motion of an entity through the world. The primary user goal is to maneuver the eye point to some location and avoid other objects in the VW. A secondary goal is the interaction with another entity by means of the entity's weapon systems and attendant projectiles. This goal involves positioning and orienting the entity to line up the other entity in the sights.

Tether

Tethering is the attachment of the eye point to an entity outside user control. This is commonly called the "wing man's view" after the flying paradigm, and the tethered-to entity can be referred to as the "lead entity," or simply, the "lead." The eye point is normally positioned to

view the lead and is fixed in a relative location and orientation. While tethered, the most common sub-task is to observe the interaction of the lead with the virtual environment. The user of the tethered-to entity controls five DOF (roll excluded) for the eye point. In free flight, an entity's DOF are referenced to the world coordinate frame, while tethered they are offsets to the lead's orientation. The offsets can be relative to the lead entity coordinate frame, or absolute, based in a world-oriented frame with an origin at the lead entity's location. Experience has shown, when the eye point is tethered to the lead, it is controlled best in cylindrical coordinates. This works quite well when the look-at point is fixed on the object. The DOF the user has to control are thereby minimized to three (rotation, radius and elevation).

Tethering is done for the user to observe the motion or actions of the lead entity. The most common user task is training. The novice tethers to the expert to learn what to expect in the simulated environment.

Inspection

Inspection – examining a point on the database – combines free flight and tethering. A typical inspection might be to evaluate how a house icon fits the terrain. The user must position the eye point with great accuracy, moving with all six DOF. Once the object/point to be inspected is centered on the screen, it is then fixed as the look-at point. The user may move the eye point through all DOF to study the object from any angle, unburdened with maintaining the desired view direction.

Inspection is usually performed by database developers. In addition, there are user systems, such as Unmanned Air Vehicle (UAV) reconnaissance, best modeled as inspection tasks.

Keyboard With Attached Mouse

The keyboard with attached mouse is by far the most common computer input device, thus the lowest common denominator. With approximately one hundred keys, including the mouse buttons (hereafter grouped with the keyboard keys), it is also the most flexible. While keys have different labels, they are fundamentally alike. As buttons, they input discrete value changes.

Applications

Since keyboard keys input discrete changes, they are used to control values that can be modeled in a step-wise manner. The most common use is as a toggle button, a device that has a discrete on/off state. Since the keyboard key springs up after pressure is removed, the transition must be monitored. When the key is pressed, the state is changed; another press restores the original state. An extension is to use a single key to cycle through a small set of discrete values. Each key press increments a counter. When the counter reaches its minimum (or maximum) value, it resets to the maximum (or minimum), creating a loop. This works well when there are five or less transitions. More transitions cause users to become impatient and repeatedly press the key/button, tending to overshoot the target value.

The mapping of key to action is also important. Ideally the key's label should be a mnemonic for the desired action. For example, the "T" key is a better choice to control time of day than, say, "N."

When there are many states across a wide continuous spectrum, as in vehicle speed, two keys are used, one each to increment the positive and negative directions. In addition to key labeling, key location is important. Both keys should be accessible to the hand without repositioning, ideally side by side. The user can manipulate states without looking at the keyboard, avoiding interruption to their interaction with the VW. Likewise, common secondary tasks, such as firing a weapon, should be mapped to the hand not controlling the

primary action, or the thumb. Mapping to the other hand is often preferable as the use of both hands further immerses the user in the VW. The sense of immersion can be decreased, or even lost, if the user has to look at the keyboard for a key.

Implementation

Figure 1 shows the key mapping of NPSNET-Stealth, the third version of NPSNET. The system allows passive observation of the networked VW. The user can monitor action in the VW by either free flight or by tethering to existing entities. The observer is invisible and does not impact other entities.

Silicon Graphics Inc. (SGI) keyboards have both arrow keys and a numeric keypad. The arrow keys work well for entity motion control in a 2-D plane or over the terrain. The left and right keys are used for turns while the up and down keys control acceleration and deceleration. The view direction is tied to the motion vector. However, as in an automobile, the desired view direction is not necessarily aligned with the direction of motion. The ability to change the viewing direction is provided by the numeric keypad. Note that the middle key resets the viewing direction, an important feature since users tend to become disoriented when not looking in the direction of travel for extended periods. Resetting the view direction to the motion vector provides immediate reorientation.

Implementing the number keys to control field of view and the arrow keys to control direction of motion, are two examples of "radio buttons." The keys are linked so that only one can be selected at a time, much like selecting a preset station on a car radio. Unlike toggle buttons, radio buttons give direct access to values. Each key controls a distinct value within the radio button group. The number of available buttons, their positioning, and labeling make the number keys an ideal choice for controlling field of view.

Tethering to an entity is easily and intuitively accomplished by selecting an icon representing the entity with the left mouse button. Once tethered, the user's speed vector is set by the lead entity. The viewpoint is now controlled by the six keys above the arrow keys. This arrangement lends itself to the layout in Figure 1. Using keys to control the viewpoint during tethering which are different from those used to control the viewpoint during free flight avoids overloading of keys and user paradigm shifts.

Dial/Button Box

The dial and button box is actually made up of two separate boxes, a dial box and a button box. The button box contains thirty-two unlabeled keys, each associated with an LED. The keys are arranged in six rows, in the pattern four-six-six-six-six-four, and operate like keyboard keys. The LEDs can be programmed through use of a thirty-two-bit integer to provide a status indicator of the key values. The dial box contains a set of eight dials arranged in four rows of two. The dials are free running (no mechanical stops) with the dial value represented as an integer. However, the output is bounded by user-definable minimum and maximum values. By sending commands to the box, it is possible to rezero the dial, allowing the dials to be used in both absolute and relative modes. In the absolute mode, the value is read directly and applied to the controlled parameter. In the relative mode, the change from the previous value is measured and then applied to the controlled parameter.

Applications

The button and dial box is a very flexible interface device. By combining both discrete and continuous input devices, most needs can be covered. However, two continuous devices cannot be made to act

well in concert as, analogously, a straight diagonal line cannot easily be drawn on an Etch-a-Sketch.

Implementation

Figure 2 illustrates the layout of the dial box used for NPSNET-1, the first version of NPSNET. As shown, only six dials were employed. Dial 0 controlled the direction of movement, with clockwise rotation causing a right turn, counterclockwise a left. After each input the dial reset to zero. Thus each input changed the heading incrementally and turns did not have to be unwound. Dial 7, the look angle, worked the same. The speed dial, Dial 2, controlled the vehicle absolute speed. In this case, the dial value was applied directly to the entity under control. As a result, the stops imposed by the user on the dial act as minimum and maximum velocity limits. The same approach was taken for the tether controls. However, rather than using polar coordinates centered on the lead entity, we treated each axis independently in world coordinates. Movement of the eye point along the X, Y, and Z axes were accomplished by Dials 5, 3, and 1, respectively. But this was confusing and difficult to operate. When tethered to an entity, the user tends to think in the coordinate frame of the entity rather than of the world.

SpaceBall

The SpaceBall is a six-DOF isometric joystick-type device which is particularly applicable to maneuvering objects in 3-D space (Chen *et al.*, 1988). The user pushes, pulls, or rotates the approximately four-inch diameter ball mounted in a fixed base. The force exerted on the ball is measured by a series of strain gauges which transmit encoded values to the application. The user can select any or all of the DOF values for consideration. In addition to the ball, there are nine buttons on the SpaceBall, eight on the base and one on the ball itself. The

Figure 2

button on the ball, the pick button, is under the ball's rubber in a small indentation on the front of the ball.

Applications

The SpaceBall is well-suited to be an integrated throttle and direction input device. Twisting the ball around the vertical axis is an intuitive way of inputting steering commands for an entity. At the same time, the fore/aft translation models the action of a throttle quite well. The SpaceBall can also model the movement of an entity along the three axes by fore/aft ball translation corresponding to front/back motion, side-to-side translation corresponding to horizontal motion, and lifting or pushing down on the ball corresponding to vertical motion. Likewise, the three DOF associated with orientation (heading, pitch, and roll) can also be modeled. However, when rotations and translations are combined, most novices and some experts have trouble de-coupling the DOF to achieve the desired results, leading to dissatisfaction with the device. If the user has small hands, the size of the ball inhibits adequate vehicle controlling.

Implementation

Figure 3 shows the SpaceBall control inputs and the button mappings for NPSNET-2. As above, most novices had considerable difficulty uncoupling the axis. The dominant coupling was the increase and decrease in speed (forward and rearward translation) with the elevation of the gun/view. Eventually most users could de-couple the two actions. However, many were unable to raise the gun without going backwards. To eliminate the problem, we simply defined a toggle keyboard key that prevented a negative entity speed.

An important observation was that novices tend to oversteer. To avoid this, we tried making the controls sluggish. But this frustrated experienced users. To compromise, we defined two buttons on the

Figure 3

base which allowed the user to adjust the sensitivity by means of a scaling factor.

When the SpaceBall has been used for extended periods, it tends to exhibit drift and confuse the user. To compensate, one of the buttons was defined as the rezero key. It sets all outputs to zero.

As a vehicle control mechanism, the SpaceBall is much more intuitive than the keyboard or button/dial box for most users. Still, there is a significant number who cannot seem to grasp the metaphor and feel more comfortable with the keyboard.

Stick and Throttle

The Thrustmaster stick and throttle set, Figure 4, replicates the appearance and functionality of an F-4 Phantom jet's controls. For simplicity, we use a small subset of the available switches and knobs on the devices.

The throttle is a single DOF device, rotation around a horizontal axis. Minimum is full back, maximum full forward. The stick is a two DOF device, rotations around horizontal and fore/aft axes. In many ways it is like the common continuous-range, self-centering joystick. The neutral position is vertical. We use three of the buttons on the stick for toggle inputs, missile firing, bomb release, and gun firing (trigger button).

Applications

This is the most specialized of the interface devices we have used and has a narrow range of applications. The throttle, being a single DOF device with a minimum and maximum, is restricted to single variable control, speed. The stick, with two DOF, can be mapped to pitch and roll. We have used the stick to steer a ground vehicle as well as control flying vehicles. When used to steer ground vehicles, the stick pitch

Figure 4

input is disabled since ground vehicle pitch depends on the slope of the underlying terrain.

We have observed that many users have difficulty using the stick for anything other than flying an airplane. They feel uncomfortable using it to steer ground vehicles or “taxi” air vehicles on the ground. From this we have learned that perception of a device is nearly as important as the device characteristics in determining suitability for an application.

Implementation

The implementation of the throttle and stick was relatively straightforward, owing to the strong cognitive model that users have about the devices. As shown in Figure 4, the throttle controlled aircraft velocity, while the stick controlled aircraft orientation, hence the direction of travel. It is interesting to note that we have implemented two aerodynamics models within NPSNET-IV. One is so realistic that only real pilots can successfully fly it. The second model, the default, is much simplified, in deference to everyone else.

Comparisons

As shown in Table 1, many controls in NPSNET-IV are mapped to more than one interface paradigm, thus ensuring that the lowest common denominator, the keyboard, was always available to interact with the VW. Any blank spaces under the “SpaceBall” and “Stick and Throttle” columns indicate the availability of the keyboard to perform that function.

Conclusions

Since NPSNET research currently covers a multitude of disciplines – distributed VW consistency, entity representation, network throughput,

Table 1

Action	Keyboard	SpaceBall	Slick and Throttle
Steer/bank left	Left Arrow	Rotate ball left	Move stick left of center
Steer/bank right	Right Arrow	Rotate ball right	Move stick right of center
Air vehicle nose up	Up Arrow	Rotate ball back	Move stick backward of center
Air vehicle nose down	Down arrow	Rotate ball forward	Move stick forward of center
Rezero roll (wings level)		Button 3	
Rezero pitch (nose level)		Button 4	
Transport to defined location	F9, F10, F11, F12		
Transport to HUD location	Middle mouse button at HUD location		
Accelerate	A		Move throttle forward
Decelerate	D		Move throttle backward
Stop	S	Button 1	Move throttle all backward
Change view to look left	Pad4		
Change view to look right	Pad6		
Change view to look up	Pad8		
Change view to look down	Pad2		
Reset view	Pad5	Button 2	
Insert view as direction	Insert		
Fire Missile	End	Pick button	Top red button
Fire Guns	Home	Button 7	Trigger
Drop Bomb	Space Bar		Middle red button
Targeting Toggle	4		
Engage Video Missile	V		

etc. – user interface design competes for attention as one among many issues. Therefore, we have employed usability tests to assess the efficacy of the software while the system remains under development. Usability testing tends to uncover most serious problems (Jeffries and Desurvire, 1992).

We found that the user's perception of an interface device made it more intuitive for some tasks than others. For example, the flight stick and throttle was the most intuitive for the free flight of an aircraft, but worked poorly for the inspection task. In addition, within free flight, the type of vehicle being controlled also impacted how intuitive an interface was. The flight stick and throttle was not as intuitive as the SpaceBall for controlling a ground vehicle.

The keyboard, being a toggle-type device, was a poor choice for controlling orientation since it only makes discrete changes. In addition, step size was problematic. Smaller step size resulted in smaller resolution and greater control, a user preference. However, it took longer to make large changes, which users found tedious. A larger step size made fine tuning difficult, but resulted in a shorter time to make large changes. Providing a means of altering the step size successfully alleviated these problems.

For the inspection task, the dial/button box dials were the best paradigm because of separate axis control. However, for the free flight task, they were the least intuitive due to their inability to act in concert.

The coupling of axes on the SpaceBall was difficult to master. This input device has few, if any, counterparts in most users' experience. Much training and practice was required. However, once learned, the SpaceBall was quite effective for tasks involving only translations, only rotations, or two rotations and a single translation. For other functions, the interface proved difficult, even for experts.

When keys/buttons were overloaded with multiple uses, users tended to forget which parameter was being controlled, leading to confusion and a decrease in the sense of immersion. For this reason, we avoided overloading. Many VW systems we have seen use the mouse

for free flight. We have found this unsatisfactory if we overload the mouse for the selection task of tethering.

Perhaps the most significant user interface issue was finding beta-testers outside the development team since, "years of research on individual differences in human-computer interaction have shown that user populations are not homogeneous in their ability to utilize a given interface" (Borgman, 1992). Luckily, we gave numerous demonstrations, publicly and privately, even to our own children. These users varied from six years of age to people in their seventies, with diverse backgrounds and motivations. These "novices" proved invaluable in obtaining feedback.

VW interface design is as much an intuitive art as an evolving empirically-based field. Effective design techniques exist, but are to be discovered in the nebulous realm of human perception, governed by the variability of humans and human experience. However, as with other arts, there are fundamentals of established validity. This paper represents some of the fundamentals we have come to understand in four years of developing VW systems.

References

- Borgman, C. L. (1992). Cultural Diversity in Interface Design. *SIGCHI Bulletin*, ACM Press.
- Chen, M., Mountford, S. and Sellen, A. (1988). A Study in Interactive 3-D Rotation Using 2-D Control Devices. *Computer Graphics*, ACM SIGGRAPH, August.
- Jeffries, R. and Desurvire, H. (1992). Usability Testing vs. Heuristic Evaluation: Was there a contest? *SIGCHI Bulletin*, ACM Press, October.
- Lipscomb, J. S. and Pique, M. E. (1993). Analog Input Device Physical Characteristics. *SIGCHI Bulletin*, ACM Press, July.

- Mackenzie I. S., Sellen, A. and Buxton, W. (1991). A Comparison of Input Devices in Elemental Pointing and Dragging Tasks. *CHI'91 Conference Proceedings*, ACM Press.
- Mackinley, J. D., Card, S. K. and Robertson, G. G. (1990). Rapid Controlled Movement Through a Virtual 3D Workspace. *Computer Graphics*, ACM SIGGRAPH, August.
- Nielsen, J. and Phillips, V. L. (1993). Estimating the Relative Usability of Two Interfaces: Heuristic, Formal, and Empirical Methods Compared. *INTERCHI '93 Conference Proceedings*, ACM Press.
- Pope, A. R. and Schaffer, R. L. (1991). The SIMNET Network and Protocols. *BBN Systems and Technologies Corporation: Report No. 7627*.
- Sanders, M. S. and McCormick, E. J. (1987). *Human Factors in Engineering and Design* (Sixth Edition). McGraw-Hill, 1987.
- Shneiderman, B. (1992). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (Second Edition). Addison-Wesley, 1992.
- Ware, C. and Osborne, S. (1990). Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. *Computer Graphics*, ACM SIGGRAPH, March.
- Zyda, M. J., Pratt, D. R., Monahan, J. G. and Wilson, K. P. (1992). NPSNET: Constructing a 3D Virtual World. *Computer Graphics*, ACM SIGGRAPH, March.