Nuance-Oriented Interfaces in Virtual Environments

Chadwick A. Wingrave

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

Master of Science

In

Computer Science and Applications

Dr. Doug Bowman, Chair
Dr. Naren Ramakrishnan
Dr. Ronald D. Kriz

July 30, 2001
Blacksburg, Virginia

Keywords: Nuance-Oriented, Nuance, Virtual Environments, Machine Learning, Virtual Reality, interface design, Human Factors

Nuance Oriented Interfaces in Virtual Environments

Chadwick A. Wingrave

Abstract

Virtual Environment (VE) user interfaces do not generally support personalized interaction or the ability to adapt to the user. Our research is developing nuance-oriented interfaces – interfaces that take advantage of subtle clues that users give in their actions. To that end, we have performed five experiments with the goal of recognizing and applying nuances for the task of selection. We found evidence that users adapt their behavior to the feedback given by the system, rather than using a preconceived mental model of the environment. In addition, users' behavior can be modeled by a reward function. Lastly, users interact by trading off exploration with exploitation. We propose a method of modeling this behavior using machine learning as future work.

Acknowledgement

I am what I am. I am a combination of my inborn self, my upbringing and the company I keep. To those that have helped me, in learning, work or play, thank you.

I wish to thank Reel Big Fish, the band whose song *Nothin'* inspired me, gave me council and a proper state of mind during my last days of writing.

I wish to thank those programmers that made the hardest part of my thesis understanding page numbering, printing, captions and text boxes in Microsoft Word. Don't let me find you in a dark alley… grrr.

To the professors that have helped me at my time here at Virginia Tech, specifically my committee of Dr. Bowman, Dr. Ramakrishnan and Dr. Kriz, and Dr. Henry. Dr. Bowman, I appreciate the advice, guidance and willingness to put up with me. I can not imagine what I must have put you through ☺. Dr. Ramakrishnan, I appreciate the coffee talks and answers to my questions; most of which started off with, "I don't understand what you are talking about Chad? Can you rephrase that?" Dr. Kriz, I appreciate you not giving up on me. I hope I can repay you for your support in some way in the future. Dr. Henry, I appreciate the talks, advice and opportunities.

To God, whom the more I learn, the more I appreciate. I will spend my life understanding your works here and never scratch the surface. When I am feeling lost, I simply look at your creations of animals, insects, plants, and yes, humans and am given comfort in knowing that there is a grand design that does not depend on my understanding of science.

To my friends, I always remember and think about you. Forgive me if I don't keep in touch as well as I should.

To my family, which seems to be growing these days, thank you for the support and patients now and in the future.

To my Mom; I still remember the concerned look on your face and weak scolding you gave me to stay in school when I was out of it. I am sure you would be proud to know I made it this far and continue on for a PhD. I miss you and dedicate this work to you…

Table of Contents

Table of Figures

Table of Tables

Preface

Benjamin Franklin used to look upon the sun chair in the Philadelphia legislative chamber and wonder… is the sun, carved into the back of the chair, rising or setting? He pondered the nation of the United States of America. I ponder computer assisted interface design. In any event, Benjamin, along with the other founding fathers, became very, very drunk after the completion of the first Continental Congress. I plan to do the same.

Chapter 1 Introduction

When I first started working on designing virtual environment (VE) interfaces, I noticed the difficulty in designing and implementing them. Due to this difficulty, the usefulness of such environments suffered because you eventually settled for something less usable than you originally set out to construct. This, in turn, placed a limit on what you could do in these environments. Also, currently the best method of evaluating the environment is through user testing and a single user test can show the flaws of the interface, ruining days of implementation. This was mixed in my head with the results of my first attempt to create a novel interface through line detection. I started to notice that user's actions were not only difficult to interpret but users themselves were not even performing the appropriate actions to cause an event to occur. So how do we create an interface to deal with the complexities of the user?

In this thesis, we introduce the concept of nuance-oriented interfaces for virtual environments. The hypothesis is that the user has an innate model of interaction that will be their first intuition for performing any action. This will also dictate the user's methods of increasing performance of an interaction technique by performing *nuances*, which can be found in input device data. A nuance, according to Webster, has three definitions; 1. a subtle distinction or variation 2. a subtle quality 3. sensibility to, awareness of, or ability to express delicate shadings. A *nuance* will be defined for our purposes as a repeatable action users make in their interactions in an environment, intentional or not, that are highly correlated with an intended action but not implicit in the interaction metaphor. If these nuances could be identified and managed as a part of the interaction technique, users would have a more responsive environment to their actions. This could lead to improved efficiency and presence; possibly even new uses for VE technology.

Consider the following scenario showing the power of nuance-oriented interaction:

> Brad, the VE user, is wandering around a model of a building being planned. He raises his hand towards the door exiting to the hallway and gives a slight rotation of his wrist and he starts to accelerate towards the door. On his way, Brad hears a comment from the building architect, Dave, who is behind him. Not hearing what is said, he raises his head with a slight tilt causing the acceleration towards the door to pause and Dave's remark to echo in his ears. "Does that electrical conduit run through here?" says Dave's automatically repeated response. Brad holds his hands in front of his face causing a finger-labeled menuing system to appear [Bowman01] and he turns on the display of the electrical system. "Nope", Brad replies as he resumes his movement towards the door. Once in the hallway, Brad points down the hall and rolls his hand forward, which causes him to zoom down the hall. Before he turns at

the end of the hall, he notices two electrical conduits intersecting when they should not. Brad holds his hand to his ear and says, “Dave, I’m in the hallway here and two conduits are overlapping, should I fix it?” “Yes, move the north- south conduit up” comes a voice from nowhere. Brad reaches midway towards the pipes with his hand aligned with the north-south conduit, pinches his fingers together, and his hand shoots-out, grabbing the conduit. Moving his hand up a little and releasing his pinch, the pipe moves up and above the other conduit.

This scenario shows multiple nuances made by a user, which the VE interprets and acts upon, leading to a seamless interface. The first nuance occurs when Brad reaches towards the door and turns his wrist. This is a natural action as a doorknob affords rotation and entry into the new environment, the hallway. The second nuance is when Brad raises his ear after Dave speaks. This again is a natural action that causes the environment to repeat, just like a human in a conversation, the last communication made. Once Brad understands Dave’s question, he moves his hands in front of his face causing a hidden menu to appear. Since the menu is hidden, it does not occlude the environment, and the intuitive method of looking at the hands where the menu is located requires little cognitive effort on the part of Brad. A few moments later, Brad, while traveling down the hallway in the direction of his pointing, rolls his hand forward causing him to zoom to the end. This is a natural motion since disapproval of the speed can be signaled by a specific change of the hand, and since the hallway affords movement in only two directions and Brad is facing one of them, the zooming action is assumed. Finally, when Brad moves the conduits, the interface assumes which conduit he intends to select based upon the affordance of the cylindrical shape. All of these nuances are assumptions made by the interface based upon the user’s input data but few interface designers would want to implement them because of the difficulty involved in recognizing the nuances and in distinguishing the situations from other user movements.

1.1 Motivation For Better Interaction

Some of the driving forces for creating virtual environment applications are visualization of multidimensional or spatial data, training in expensive or hazardous environments, and entertainment. In most cases, the VE is trying to invoke a feeling of presence [Sheridan94] in the user to make the data more understandable or to make the training more realistic. This presence can easily be interrupted by an interface that does not map user actions to expected results or worse, incorrectly maps a user action to an unexpected result. The outcome is a user that is all too aware that they are in a VE and can not interact with the environment. Beyond just the utility, the ability to have accurate, robust interfaces developed quickly can reduce VE costs. This could open more domains to benefit from the advantages from VEs.

1.2 Iterative Design for Hypothesis Testing

VE interfaces are more complex than other interfaces yet the design process remains much the same. Basically, designers create a metaphor, design the layout and feedback of the device, test it on users, make changes and iterate [Hix99]. This works fine on simple interaction but as the complexity of the interface increases, this iterative design process becomes too slow to cover the interaction possibilities (see section 2.3). Another problem is that even if we had the time to iterate over the design space, we would have difficulties dealing with the user's mental process. In traditional interfaces, users have to frame their interaction through a pointing device, such as a mouse, or discrete events, such as buttons; both leave little room for ambiguity. VEs conversely have just a button or two forcing much of the interface to be built off of a 6DOF position tracker. Additionally, users in VEs bring in their intuition in working with real-world objects and it is hard for VE designers to mimic the real world. For that matter, it is not clear that mimicking the real world is optimal.

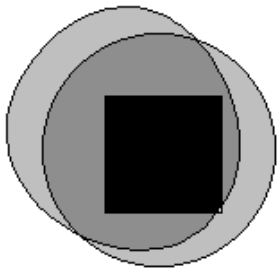

Figure 1 The interface designer would expect users to select the square in the space as shown by the perfect circle. The user might behave more like the oblong circle however.

As an example of this, consider the user pointing at the black square in Figure 1. The interface designer would typically consider an angle of error around the square as acceptable for selecting it, shown as the perfect circle around the square. The user has their own space of selection shown by the oblong circle. This shape could be explained by the hypothesis that the user points with their right hand and this makes them err to the upper left, but the end result is that the user and designer disagree. This disagreement causes the user to reshape their model of that selection space to match the designer's, requiring user experience and causing frustration. This example is just a small example of the problem we wish to address. Before we can start building interfaces that match the user's innate model, we need to discover what users consider factors to their innate models. In this case, we hypothesize it is pointing with the right hand but we are not sure.

Some objects might have affordances that cause them to be interacted with in certain ways, such as the cylindrical conduits in the scenario in the Introduction. Affordances have been shown to be a significant factor in creating usable interfaces in desktop 2D applications and the real world [Norman90]. Since most of the intuition users apply to 2D comes from their interaction in the real world and since VEs more closely align with the real world, users should more readily apply their intuition in VEs.

Another factor to consider is whether all users are the same. In the example above, the user is right handed. What happens when a left-handed user enters the environment? Chances are that they would not point with their right hand and the interface needs to adjust to the user and not vice versa. This is just a minor albeit obvious property of the user. It might be that users tend to look at objects differently or tend to like different manners of interaction. If this is the case, we should attempt to identify parameters that can explain the manner in which users interact.

Once we start to discover information about users and their interaction, we need some method of framing that data so that it can be transferred to other VEs. Currently, designers might learn that users tend to act a certain way due to some affordance of an object but it is not common practice to record that affordance in a manner where the next interface can make use of it.

As we increase the number of hypotheses through properties of objects and users, we need some new methods of hypothesis testing. This leads to the requirement of some form of complexity handling, such as machine learning (ML); before that, we need to determine if modeling the interface as a complex situation can indeed solve our problem.

1.3 Statement of the Problem

One of the purposes of an interface is to match user actions to actions in the interaction space. Despite what you might think, creating an interface that is 100% accurate at identifying what the user is trying to do is quite easy. All the designer needs do is create a one-to-one mapping of user actions, such as a button press, to all actions in the interaction space. This is obviously not feasible as the number of discrete events in a VE is limited and the possible actions are quite large. Also, the memorization required on the part of the user would be huge. An alternative is to have hierarchies of discrete events such that each event leads to another logical sub-hierarchy of events but this requires many user actions to signal just one environment action. The third alternative is to have some non-discrete interaction that can specify some part of the mapping to an action. Unfortunately, the non-discrete nature of the interaction introduces ambiguity. For example, pointing at an object is one way of selecting it but ambiguities form when multiple objects lie along the pointing direction or the user is not an accurate pointer.

Interfaces that allow ambiguity make up almost the entirety of interface design in VEs because of the speed and ease in which the user can specify ambiguous actions. The problem is that each new interaction increases the amount of ambiguity in the system and if the amount of interaction is high, as is required for newer, more ambitious VEs, then the system becomes either unusable or at best, frustrating.

Our thesis is that nuances can frame user actions and through the use of learning techniques, be optimized. We can then produce robust interfaces designed to make use of user's innate model. Also, the learning techniques can be used to deal with the errors and gaps in input data. They can also reduce the rigidity of the interface and make the interaction feel more natural and fluid.

1.4 Related Work

Most of the research in this thesis is on the VE task of selection. Therefore, we reviewed related work in human-designed selection techniques and interaction techniques that have been designed with ML techniques. There is very little work done using ML to design full interfaces and not just standalone techniques. None of this work has been specific to VEs as far as the author is aware.

1.4.1 Current VE Selection Techniques

There has been much work done in creating intuitive, less fatiguing, unambiguous selection techniques for VEs. A selection technique is, "… the specification of one or more virtual objects by the user for some purpose." [BowmanDis] The techniques can be roughly categorized into ray-based and reaching techniques with a third category being multi-modal.

1.4.1.1 Ray-Based Selection Techniques

Ray-based techniques require the user to specify a ray to intersect an object. This is generally done by extending a ray from the user's finger, called ray-casting [Mine95], but can also extend from other positions such as the eyes, referred to as gaze-based selection. These two techniques have properties that make them less fatiguing. Ray-casting allows the user to "shoot-from-the-hip" so they do not need to raise their hands high and in gaze-based selection, the movement of the user's head is not fatiguing. Problems such as occlusion, the covering of distant objects by a closer object, affect all ray-based techniques because of the difficulty in distinguishing between the intended object and the occluding objects. One way around this is to specify a cone instead of a ray, aperture-based selection, with the aperture of the cone encircling and distinguishing between the occluding and intended objects [Forsberg96]. Occlusion too can be used to specify rays through the techniques of the Sticky Finger (specify a ray from the eye through a fingertip), the Head-Crusher (specify a cone from the eye through two fingers) and Lifting Palm (specify a ray from the eye through the upturned palm) [Pierce97]. These techniques are more fatiguing because the user must raise their hand but have been shown to be more accurate in some cases.

1.4.1.2 Reaching Selection Techniques

Reaching techniques are generally more intuitive than ray-based because they are based on a human's normal cognitive process. If you want something, you reach for it. They are however very limited by the fact that arms have a limited length so most of the varieties of reaching techniques are focused on working around this limitation.

The easiest method to work around the arm length limitation is to scale the distance the virtual hand is from its natural position. So, if the furthest object you want to reach is 100 feet away and your arm is two and a half feet long, the arm movements are scaled 40 times. This is successful in some applications. Unfortunately, most applications either do not know the furthest distance of an object, making the scaling factor hard to determine, or require the user to work with objects both close and far in relation to them,

making fine-grained movements difficult to make. To deal with this, techniques were created to use a non-linear mapping function to scale the hand's position, called the Go-Go technique [Poupyrev96], such that the hand moves normally up close but extends rapidly as the user reaches into the environment. Go-Go still has difficulty selecting small objects at a distance, since the slightest movement will move the hand large distances, but is a definite improvement over linearly scaled functions. Another reaching technique is the World In Miniature (WIM) [Stoakley95]. In this reaching technique, instead of reaching into the world, the user reaches into a miniature model of the environment, usually attached to their hand. This technique makes reaching objects easy but small objects can be hard to select just like the scaled arm extensions already mentioned.

1.4.1.3 Ray-Based vs Reaching Selection Techniques

All reaching techniques suffer from having to match three dimensions of the user's hand with the three dimensions of the object to select whereas ray-based techniques only require the matching of two. One advantage of reaching over ray-based techniques is the natural shift into the manipulation task by which most selections are followed. The HOMER [Bowman97] technique was designed specifically to address this problem. It uses ray-casting to select an object and uses a linear mapping function for manipulation.

1.4.1.4 Multi-Modal and Hybrid Techniques

Multi-modal techniques combine many modes of human interaction into one interaction metaphor. This can be done through text, speech, position trackers, gestures, eye tracking, force-feedback devices, haptics, etc. These interaction metaphors tend to be more usable than if the interaction was purely through one modality alone. This has been explained in Perceptual User Interfaces as Mutual Disambiguation [Oviatt99]. Early work combining voice and gesture was "Put-That-There" [Bolt80] where the user gestured with their hand and spoke.

Hybrid techniques such as widgets and multiple tracked arms are also used. A two-handed selection technique called Framing Hands [Pierce97] can select multiple objects by specifying a selection pyramid. Menuing systems, such as drop-down and floating which were common in 2D interfaces, can select options in a list and have been implemented several ways.

Multi-modal and hybrid techniques are still a developing area both in hardware devices, software APIs and evaluations as "…multi-modal interfaces make necessary the development of software tools that satisfy new requirements. Such tools are currently few and limited in scope."[Nigay95] These techniques do provide promise because of their ability to handle complex situations and the affordances and feedback they provide.

1.4.1.5 Selection Technique Taxonomy

For a further, more in-depth selection technique taxonomy, see Figure 2. In this taxonomy, our ray-based selection is equivalent to pointing and occlusion and our reaching selection to object touching. The multi-modal and hybrid selection is equivalent to indirect selection.

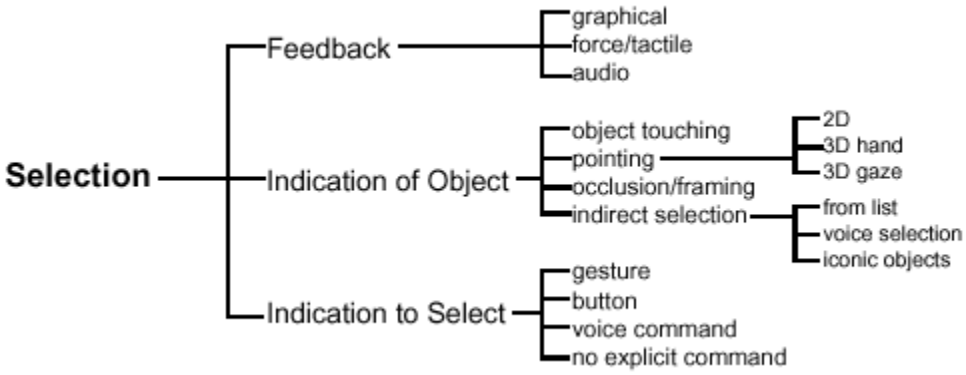

Figure 2 Taxonomy of selection techniques from [BowmanDis].

1.4.2 Related Work in ML and Interface Design

The use of ML and artificial intelligence techniques in user interface research is not new. ML has been used in handwriting recognition [Garris98], sign language recognition [Kramer89], automatically adapting interfaces to users as they work in environments [Brown90] and to support programming-by-demonstration [Cypher93].

One of the primary advantages of using these techniques is their ability to generalize to situations not encountered before. This generalization ability is aided by model-based techniques such as neural networks, decision trees, production systems, rules, and navigation maps. Such techniques require a reasonable amount of both "training data" and "training time" in order to construct a model. Evaluation of such techniques thus involves a distinct training phase followed by a test phase to validate the models. The techniques differ in their complexity of learning the representations (models), amount of training data required, the nature of their induced representations, and their ability (or lack thereof) to incorporate new data on a continual basis.

While ML techniques are prevalent in many desktop user interfaces, VE interfaces constitute a relatively nascent field of application. Slater et al. describe the use of neural networks to learn when users are walking in place to create a VE travel technique [Slater95]. This was only the recognition of a user action and not the creation of an entire interface, however. Neural networks (NNs) can approximate any function to any required level of accuracy (perhaps with exponential increase in complexity). They use one or more layers of intermediate functional elements to model the dependence of output signal(s) on given input parameters. The general problem of learning NNs is NP-complete [Blum88], but that has not dissuaded engineers and scientists from employing them as a tool to solve functional modeling problems, particularly noisy ones. Similarly, models such as decision trees [Ruvini00] and version spaces [Eisenstein00] have been employed in interface research.

1.4.3 Why Nuances Differ from other VE Interface Approaches

We are trying to look at a larger problem in VE interaction. Research has up to this time been spent studying individual types of interaction techniques and trying to compare and contrast them by observing the user's responses. This includes even those applications

that have applied ML to optimize a technique. Our approach is to discover how the user wants to work in a system as a whole and what their innate models are of the interaction before them, not of a single particular interaction studied by itself. With this knowledge, we hope to build nuance-oriented VE interfaces that are tuned by the user and not imposed on them.

1.5 Goals

The goals we will achieve are a proof that users do in fact interact using nuances and that they can be modeled by a learning system. We also need to gain a better understanding of how users work inside a VE system. Since there is no work in the area of nuance-oriented interfaces, we also need to create a method of classifying nuances as we learn them. Finally, we will investigate what is required of a complete nuance-oriented system so that we can later build one.

1.6 Overview

This thesis is organized into six chapters. Chapter 1 of this thesis gave an introduction to the problem and covered related work. Chapter 2 gives a complexity comparison of VEs and typical desktop interfaces to show why they are different and what to do about the differences. Chapter 3 introduces the concept of nuance-oriented interfaces and how they can be used to frame and design interfaces in VEs. Chapter 4 reviews typical ML methods and provides an existence proof that ML can be used to capture nuances. Chapter 5 is an experimental investigation of the user through multiple VE user tests. This is followed by conclusions and future work. Chapter 6 discusses the implications of the results of these experiments. The thesis concludes with a summary and list of future work.

Chapter 2 Interface Complexity in Virtual Environments

This chapter gives an argument for why VE interfaces are complex and therefore should be investigated with complexity handling techniques. The argument is based upon the comparison of input devices, their data and the metaphors used in command-line, 2D and VE interfaces.

We classify input data into two distinct types, discrete and continuous. Discrete data has a very distinct start and stop, such as a button press or finger pinch. This makes it fit easily into the event-based architectures of 2D applications with the user, not the interface's recognition function, creating the errors. Continuous data is a running stream of data generated through such devices as the mouse and position trackers. It is not as easy as discrete data to work with, as placing it into an event-based interface requires a recognizer function, created by the interface designer, to identify when an event has occurred. This recognizing function is part of the interaction technique which maps input data to actions in the application. This function must also deal with errors in input data as most continuous devices are prone to precision problems. Both types of data can be expressed in Degrees of Freedom (DOF) in that each DOF is a range in which the device can report with a button having 1 DOF, a mouse having 2 DOF and a position tracker 6 DOF (X, Y, Z, Yaw, Pitch, Roll).

Building a new type of interface to handle complexity is a long and difficult task and we have to ask ourselves why this is necessary. Jacob gives the major reasons why VE interfaces differ from traditional WIMP interfaces [Jacob99]. These are:

- single-thread input/output versus parallel, asynchronous, but interrelated dialogues
- discrete tokens versus continuous and discrete inputs and responses
- precise tokens versus probabilistic input, which may be difficult to tokenize
- sequence, not time, is meaningful versus real-time requirements, deadline-based computations
- explicit user commands versus passive ("non command-based") monitoring of the user

There are other less obvious reasons as well:

- Users have evolved to operate in 3D spaces and are quite adept at it. Their methods of interaction will be based upon their existing knowledge of the world and this only becomes more important as the levels of presence rise in VEs.

- Often, actions in a VE are not discrete so creating an undo feature is difficult. For example, users in a travel technique trying to revert to their previous position have difficulties since position is a stream of locations over time and not discrete states. Since the ability to reverse unwanted actions is one of the most important features of interface design, this could lead to severe difficulties, especially considering that VEs have difficulties recognizing user actions.
- VE interaction devices have higher DOF than 2D and also are more error prone. Recognizer functions created by the interface designer have more DOF in which to not align with the user's belief when an event occurs. This can lead to usability problems.
- Interface designers usually ignore much of the input data to simplify the recognition function. This process removes much of the possible richness of the interaction so users give more data than the interface takes advantage of.

To investigate these difficulties, let us compare input devices and the metaphors designed to handle them. Afterwards, we will give a numerical argument for why VEs are so complex.

2.1 Comparison of Input Devices

A review of the types of input devices for command-line, 2D and 3D applications leads to a very good understanding of why interaction in 3D is difficult. Command-line interfaces tend to operate on discrete data that has little ambiguity. 2D input devices tend to still be mostly discrete in nature with some continuous data and 3D input devices tend to be nearly totally continuous with the few discrete inputs that do exist having few affordances.

2.1.1 Command-Line Input Devices

Command-line input devices are completely discrete in nature where the input device is a device with many buttons called, obviously, a keyboard. This means there is little ambiguity in identifying what the user is trying to do, since they either pushed a button or not (users can mistakenly press a button but this is not an error in recognition). In good interface design, keyboard keys have an existing knowledge in the users mind because an action such as "Copy" will be mapped to the key that starts the same letter, "C", but most interaction must be memorized by the user, leading to a high cognitive load.

2.1.2 2D Desktop Input Devices

The typical 2D interaction devices are the pointing device (mouse, trackball, touchpad, etc.) and keyboard. The pointing devices for 2D interaction generally give accurate data in 2 DOF to the application. There can be errors on the part of the user, for example missing with the mouse, but it is usually hard to generate these errors except when performing detailed work. The DOF are usually very constrained which is quite acceptable since the screen is all that needs coverage. Also, since the user has no pre-existing or innate intuition about how to use the mouse for anything other than pointing (i.e. our ancestors never felt the urge to rub rocks in circles on tree bark to communicate),

we can assume that they will use the mouse as intended. Many users did make the mistake of placing the mouse on the floor and stepping on it like a foot-pedal but those were mistakes in how they perceived the device to work. Once they realized that they were supposed to move it with their hand, they had no other errors in their intuition.

There are a few desktop 6 DOF input devices. One type is basically a mouse with a few buttons that you can hold in your hand to give 6 DOF data. Ascension, Fakespace, Logitech and Polhemus have made or currently make such a device. Alternatively, there are devices much like a trackball except they are 6 DOF devices. One is made by Labtec called a Spaceball and the other by LogiCad, called a Cyberpuck. These devices allow people to work at the desktop and still work with higher dimensional data.

2.1.3 VE Input Devices

There are many more input devices in VEs than in 2D desktop applications. There are position trackers for points in space, gloves that offer various amounts of detail of the hand, discrete inputs of various types and even newer devices to incorporate eye and voice data (which will not be covered here). Unfortunately, the successful input devices of the 2D world do not enter into this environment. The keyboard can not be brought in because displays like head mounted displays would block the view of the keyboard and in rear projected displays, the keyboard would block the view of the user. There has been some work to develop keyboards in VEs [BowmanHCI01] but it is not widespread. Also, since the mouse only gives 2DOF, its usefulness is limited in VEs.

2.1.3.1 Position Trackers

The most important and common input device to a VE is the position tracker. The existing tracker devices have various ranges in which they operate but most are able to sufficiently cover an area for what researchers require. Though a small area, this is easily larger than the coverage of a 2-D pointing device. Additionally, VE trackers give data for multiple points as opposed to one and in four extra DOF than a 2D pointing device. Additionally, VE trackers have more errors in accuracy, called drift, and precision, called jitter, than the 2-D pointing devices.

2.1.3.2 Gloves and User Hand Data

Many systems have attempted to harness the data produced by the hand at various levels of fidelity and with various levels of success. The Fakespace PinchGlove device gives discrete, non-error prone data to an application by recording pinches between fingers using conductive cloth sewn into the fingertips. The total number of combinations is very large[1] but only a few of the combinations make sense and are physically possible to make. Additionally, unlike the discrete input of a keyboard, there are no affordances offered by the pinches. For example, the interface that requires the user to touch their thumb, pinky, middle and index fingers together to select the 11th item in a list will end up with many annoyed users with hand cramps desperately trying to remember which fingers do what. A step up in data fidelity are the DataGloves which measure the degree of curvature of the user's fingers, making them a continuous device and adding 5DOF to each hand, one per finger, with unfortunately high error rates [Kessler95]. The last step

[1] There are 115,974 distinct combinations of pinches.

in fidelity are mechanical gloves that give full information about the hand to the application. They provide 20 DOF per hand with little error but require precise fits and much time to put on.

2.1.3.3 Discrete Devices

Many tracking devices augment their input data with discrete devices. Devices such as the Polhemus Fastrak and Intersense have styluses with one or two buttons. Also, buttons can be found on devices such as CAVE wands, 3D mice and the Polhemus 3Ball.

2.2 Comparing Interface Metaphors

Interaction metaphors are ways to think about an interaction with a system and several have been successfully developed to deal with different types of interfaces.

2.2.1 Command Line Interfaces

Command-line metaphors really do not exist. Commands are given explicitly and precisely by the user. The rest is best stated by a quote from an article by Neal Stephenson:

> Why are we rejecting explicit word-based interfaces, and embracing graphical or sensorial ones--a trend that accounts for the success of both Microsoft and Disney?
> Part of it is simply that the world is very complicated now--much more complicated than the hunter-gatherer world that our brains evolved to cope with--and we simply can't handle all of the details. [Stephenson01]

2.2.2 2D WIMP Metaphor

The most common interface metaphor for 2D use is the Window, Icon, Menu and Pointing device (WIMP) metaphor. It increases the knowledge in the world for the user, making interaction easier, and gives them a way to think about the interaction as if they were on a desk. The continuous data of the 2D pointing device (a mouse usually) is managed with an event-based paradigm which responds with actions when the mouse buttons are pressed. At first, discrete buttons on the pointing device offered the only means of triggering an event but later events, such as passing over an icon or leaving a window, increased the usability of applications. These later events were methods of handling more complexity in the environment by recognizing useful points of interest.

2.2.3 VE Metaphors

VE metaphors are still very much in the research domain. The first step in creating an understanding of VE interaction should be the understanding of tasks that need to be accomplished in VEs. From there, techniques need to be applied to those tasks, evaluated and merged together to form a metaphor. There may not be a standout metaphor for VEs such as WIMP for 2D interaction but current implementations tend, for good or bad, to use a real-world or natural metaphor for interaction.

2.2.3.1 Categorizing Tasks of a VE

"Applications of VE are currently in the realm of vehicle simulation, entertainment, vehicle design, architectural design and arrangement, training, medicine and probe microscopy" [Brooks99]. Each of these application fields has similar tasks which have been categorized into travel, selection, manipulation and system control [BowmanDis]. Any metaphor that would be developed would have to cover some part of these tasks. Also, some metaphors are not appropriate for some applications. For example, in a car simulation, one would need to develop a travel metaphor that operates like the car but for architectural walk-throughs, traveling where you point may be more appropriate.

2.2.3.2 2D for 3D

Users are familiar with 2D WIMP metaphors so it makes sense to use that existing knowledge to design interaction for 3D applications. Menus and icons are easy for users to understand and select, using pointing or occluding hand techniques, but extended use is not possible due to arm strain. Additionally, occlusion is a problem for these techniques because the menus and icons either occlude objects in the environment or are themselves occluded, something that was not as much of a problem in 2D displays.

2.2.3.3 Natural

Natural metaphors for interaction are those that operate the same in the VE as in the real world. They are the easiest for users to understand due to prior knowledge but are severely limited by two factors: reality and complexity. The most natural notion for selecting an object may be walking up to it and grabbing it with a hand but this requires the user to exert themselves in walking and raising their arm. If the application space is large, this natural technique will not be usable. The second limiting factor occurs when a user is immersed within a VE and starts to interact naturally. A simple task such as removing the lid from jar is complicated for interface designers to recognize and so the effort on the part of the user to twist their wrist to open the jar is useless. This leads to a mismatch in what the user expects and what occurs.

2.2.3.4 Magic

Magic techniques for interaction are those that are not natural in the real world but possible in VEs. Examples of magic travel metaphors are flying and teleportation. They have the possibility of not being the most intuitive to the user at first but in some cases are better in terms of speed and user comfort. For example, to get to the top of a building it is easier to fly or teleport than to walk to an elevator. Also, instead of twisting the top off a jar, it might be easier to pinch your fingers or push a button when near the jar lid. Unfortunately, in some cases, magic techniques cannot be implemented because the VE itself requires natural interfaces as in the case of training applications. Many magic techniques work because they reduce the amount of data to be processed by the application to a minimum. In the case of the previously discussed selection by ray casting or occlusion, the technique reduces the number of dimensions that the user must align making it easier to specify.

So, what makes VEs different? In going from the command-line to VEs, the amount of ambiguity in the input has data increased. Also, the amount of data has increased. The

metaphors to handle the data also become more realistic. A numerical explanation will shed more light on this issue.

2.3 An Argument for VEs as Complex Systems

Input data can be thought of as an interaction space of whatever dimensionality it possesses. For example, a mouse gives an X and Y position on the screen; so two dimensions of data; so it has a dimensionality of two. In order to recognize user events in an interaction space, the event must be recognized by some bounding function on the dimensions of the data. To discretize the space for a two-dimensional device, like a mouse, we must divide the space of interaction through such boundings as planar divisions, circular or pologonal envelopments, or a combination of both. An example of divided space is an icon in that the recognizer function must identify when the mouse enters the space in which the icon exists. This requires four points to be specified; one dividing space for an icon in three dimensions requires eight points. The simplest bounding function uses two hyper-planes to bound each dimension at an upper and lower value. So, specifying an icon in x dimensions requires at least 2^x points because every dimension must be bounded by at least two values; a lower and upper. Does this mean that the recognizer function for the 20 DOF glove we would require 2^{20} (1,048,576) points? No, it requires 2^{26} (67,108,864) points because the hand adds another 6 DOF. In this situation, the interface designer ignores many of the DOF which unfortunately removes many of the clues, or nuances, in the data that users consciously or unconsciously give a system. So, to make greater use of input data, we need methods to incorporate all the details in the input data.

The first step in dealing with complexity is identifying a model that can represent the data. Once this model is identified, then methods of dealing with complexity can be applied and simplicity can be weaned from the morass of input data. These topics are covered in the chapters to follow but first, let us get an idea of what nuances are.

2.4 Summary

In this chapter, we have shown that interfaces for VEs exhibit a much greater complexity than more traditional types of interfaces. Both the input devices and metaphors contribute to this complexity. For our work, this complexity is both encouraging and challenging. The encouragement is that it supports our notion of why we need some new methods to deal with VE interfaces. The challenge is to now investigate how to create that system for VEs which will have to deal consistently and accurately with very complex interaction data. The next two chapters will detail our vision of nuance-oriented interfaces and the use of ML algorithms on them.

Chapter 3 Nuance-Oriented Interfaces

Nuances can frame user intentions by recognizing patterns in user actions and mapping them to interface actions and identifying user goals and subgoals to assist the user. The trick is that in many situations, user intentions depend not only on the properties of a situation itself but also on the gestalt of several sub-situations, maybe even those situations over time. A nuance representation allows interfaces to process parallel input data, form their own bounds for continuous data, and be probabilistic and based upon time. A system like this has not fully been attempted in the past because of the work required in recognizing all the parameters to the interface. An interface VE that framed users' nuances and intelligently acted on them would not require the interface designer to work out all the details of a robust interface but would let the user flesh out the environment with their innate model. Since user's brains can process information from various sources and work on multiple tasks in the real world, we would expect that the created nuances would handle the difficulties of VE interfaces, roughly based on reality, quite well.

Nuances operate on nuance **properties**, nuance **concepts** and other nuances which we shall refer to collectively as **atomic nuances** or even **atoms**. These atomic nuances are what the nuance-oriented interface is built upon and incorporate into their recognition function. For ray-based selection tasks, some of the properties would be the size, shape and location of the objects. A concept could be the angle of error an object lies off of the specified ray. The recognizer function of the interface is created off of these atoms and for ray casting, the function would be to select the object that has the smallest angle of error from the specified ray when the user pushes a button. Other nuances, as they become recognized and incorporated to the nuance-oriented interface, will modify the behavior of the interface, thereby altering and optimizing the interaction.

There are two parts to interface design with which nuances can help. The first part is the recognition of when the user wants to perform an action. This, in the past, has been a trivial task since there is little ambiguity in 2D and command-line interfaces. For VEs, as was covered in the last chapter, this is not so trivial. The second part is the optimization of how the user performs tasks. For instance, when a user wants to perform an action, there are several steps they must take to complete their goal while each of these steps identified by the interface. Nuances help because they can help in the recognition of the actions a user takes, or the induction based on input data of how users perform an action, and the optimization of the paths to complete a task, or the deduction of how to perform their actions faster.

In this chapter, we will explain a nuance-oriented interface, with examples applying to VEs. We will look at the categories of nuances and guidelines for them to follow. Also, we will review a methodology for building interfaces with them called the Formative Nuance-Oriented User-Centered Evaluation Cycle.

3.1 Scenarios of Nuance Usage

Here are a few scenarios of how nuances can improve the interaction of a VE.

Example 1.

> *Dave is back and he is trying to place walls in a model of a building he is working on. The interface uses a non-linear arm mapping function to move his hand when he manipulates objects in the environment. This allows him to move objects precisely up close but still reach way off in the distance. The problem he is having now is trying to place a wall on the other side of the building. He tries to put it into place but even the jitter of the tracking system is enough to make his wall bounce all around the environment.*

The non-linear mapping is helping him reach the other side but it is not allowing him the ability to work with enough precision once he is there. The system should build a nuance that recognizes when the user is getting frustrated and should change the mapping function to something more appropriate. For instance, as he keeps his hand at a distance longer, the mapping function changes to allow more detailed movements at that distance. When Dave brings his hand back to his body, it reverts to the old mapping function.

Example 2.

> *Our friend Dave has built his building and now wants to see what type of lighting looks best. As he is standing under the hanging lamp in the center of the room, he realizes that there is not enough light. To remedy the situation, he points at the light and a ray extends from his finger. As he points, the ray keeps flipping back and forth between the light bulb and the lamp since the bulb is so small and hard to point at. Dave is frustrated.*

In this situation, it is probable that Dave is quite accurate in pointing at the bulb but his precision is varying and overlapping onto the light fixture. The nuance could be that the object to be selected has some dependency upon where the ray has been pointing over time.

Example 3.

> *Now that Dave can see the room better, he realizes that the light fixtures are a bit much and wants to downplay their importance in the room. To do so, he needs to select them to change the styles. When he tries to do so,*

the environment keeps selecting the light bulb in the center of the light fixture, even when he veers around the light bulb.

Example 2's nuance is probably causing this difficulty. In this case, the precision errors Dave is now exhibiting are far outside the light bulb but since the interface is now paying attention to the accuracy of his pointing, things are going sour. To remedy the situation, the last nuance could probably have some concept added to it to deal with the amount of precision and use it to outline the objects that are trying to be selected.

NOTE: The author realizes that the best interface for changing room lighting is probably not pointing at a light bulb but it serves only as an example. Scenario-based design with user testing should recognize that the task could better be handled by some other means.

3.2 Categories of Nuances

We have identified four categories of nuances. These include *object nuances*, nuances that arise from some affordance of the object, and *environmental nuances*, nuances that arise from an object existing in relation to the environment. *Refinable nuances* adjust boundaries for existing techniques such as correcting for constant errors in interaction. There are also *supplementary nuances* that are not intuitive but exist and can be mined from interaction data. These will take time and research to discover.

Figure 3 Object nuances of similar shaped objects such as a cup (left) and a column (right) afford the same selection principle.

3.2.1 Object Nuances

Object nuances are better described as affordances that the object possesses. An affordance is "the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used" [Norman90]. These object nuances will change the manner in which the user chooses to interact with the object.

For example, we might intuitively want to grab a stone column and a glass of water in the same manner because of their similar cylindrical shape (see Figure 3). Objects of this

shape may make the user feel the need to select it with the orientation of the hand matching the orientation of the object. So, a VE that had the user select between various pots on a stove with handles at various angles, could use the orientation of the hand as a nuance in determining the correct pot. If the user were trying to select a glass on the stove, the orientation of the hand would be key to differentiating between the pots and the glass because the hand would be at a 90-degree rotational difference to the panhandles. These both work off of the same nuance, "cylinders tend to be selected in the same orientation of the hand."

If we were to select a flower and a golf tee however, we would grab each in a different manner even though they are both cylindrical. In the case of the flower, we would grab along the stem, away from the leaves, and in the case of the tee, at the tee's head. This is because the flower offers affordances that say the petals are fragile so grab at the base while the tee's affordances state that the head would offer the best grip. These two nuances are different in that they are based upon the properties of the object and the nuance of the object's shape.

3.2.2 Environmental Nuances

Environmental nuances are nuances representing changes rooted not in an object but intrinsic in either the user's perceptual belief or the environment's design. In other words, there is a property the user is working with and reacting to that causes them to work this way. In the real world, we all walk on the ground because of gravity, not because the ground has the object nuance of being "walked on".

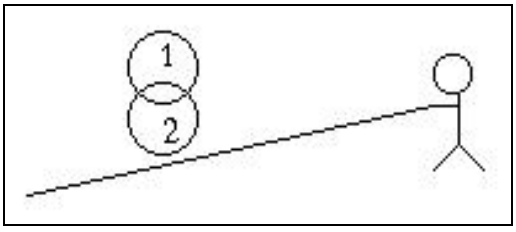

Figure 4 Using Ray Casting to select between objects 1 and 2, the user errs in the direction away from the object they do not want to select.

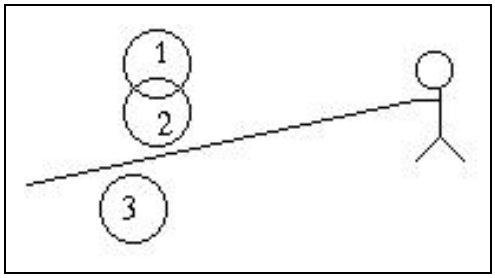

Figure 5 Because the user errs away from objects they don't want to select, we make the assumption that object 2 is the desired object even though objects 2 and 3 appear similar distances away from the user's Ray Casting selection.

An example of an environmental nuance can be seen in the ray-casting selection technique [Mine95]. In the selection of objects that are close together, users should be able to produce small nuances and the VE should interpret the extra data. For example (Figure 4), the user might try to err in the direction away from object 1 when they are trying to select object 2. This could help differentiate between objects 1 and 2. In another case (Figure 5), there is an object 3 that appears to be fairly similar in distance from the ray as object 2. Since we know that the user consistently errs to distinguish between two close objects, we would assume that the intended object is object 2 and not 3. In this

case, it is not the property of a single object that cause the user to change behavior but multiple objects acting together.

Another example would be grabbing the same stone column as mentioned before but at different distances; try this yourself. Imagine the stone column a few inches in front of your nose and try to grab it. Then, imagine it further out and grab it. If the object appeared to be in front of your nose, you tend to approach it from the side. If you imagine grabbing it at a distance, you tend to approach it with your wrist pointing away from you. This is a simple nuance but helpful if the user were standing on the steps of the Athenian Acropolis and the interface had to decide if the user was attempting to grab the inner or outer columns. Such an interface created by a designer would be dauntingly difficult and costly, if possible at all, to create. Additionally, it would most likely be limited in scope to selecting columns of Greek temples.

3.2.3 Refinable Nuances

Refinable nuances can be used to alter the existing behavior of an interface to make it more usable. This can be in order to correct for errors in input data or the user. It can also be used to increase the precision or increase the usability of boundary conditions for environments that require it.

It has been shown that users err more in depth than in the horizontal and vertical dimensions [Werkhoven98]. A refinable nuance would reduce the emphasis placed on accuracy in the depth dimension or maybe even model how the user errs in that dimension. In this way, if the user is trying to select an object using arm extension, the refinable nuance will widen the acceptable depth error.

Also, ray casting implementations use a ray extending from the tip of the user's finger and have the user align that ray with the object they wish to select. A refinable nuance would discover that a user prefers the ray to extend at a slightly adjusted angle to reduce their stress and increase accuracy.

3.2.4 Supplemental Nuances

Not based upon object affordances or environmental effects, these supplemental nuances can better be grouped into the two subclasses of strategies and discoverables. These nuances take advantage of user innate models. Also, these could be crafted from non-intuitive means to create expert interfaces.

3.2.4.1 Supplemental Nuance Strategies

Strategies are means in which users tend to work within an interaction framework to increase the usability of it. Strategies can be small improvements such as stiffening their wrists to improve pointing accuracy, up to large improvements such as rotating their head to move occluding objects out of their way. When users have the option of signaling a selection by pinching during ray casting selection, they eventually start to use their middle and thumb as opposed to their index and thumb. This allows the index finger to stay pointing and the middle finger and thumb to move closer together so when a pinch occurs, moving the finger a smaller distance upsets the tracker less. Larger strategies are learned too. For example, many users use body-centered references such as pointing to

indicate objects or locations they want to remember later [Bowman99]. Then, as they move through the environment, they update the body-centered reference to serve as a physical mnemonic.

3.2.4.2 Supplemental Nuance Discoverables

Discoverables are methods of interaction that users develop on their own inside an interaction framework. Users tend to develop methods dealing with difficult to handle situations within the existing interface in ways that were not originally intended. In one experiment in section 5.1, the users were to select an object through occlusion selection of an object with the ray extending from the user's eye through their palm. Since the model of the palm of the hand was circular and the object they were selecting was a sphere, many users matched the perimeter of the palm to the sphere. In our implementation at the time, this extra information the user provided was ignored but could have been taken advantage of through aperture based selection [Fosberg96]. Nuances should recognize and formalize such discoverables, forming new nuances that will enhance the features of this discovered nuance. Recognizing discoverables quickly is critical since it is only a matter of time until users recognize that their extra effort is not being recognized and they stop putting in the effort to create the nuance.

3.2.5 A Note on Nuance Applicability

Not all nuances are applicable to all users. Some nuances can be used to personalize an interface to a specific user or to a category of user. For example, many websites allow users to build a personalized site that only holds information the particular user is interested in [Manber00]. Other websites recommend options to users based upon the likes and dislikes of users similar to them [Resnick97]. It should therefore be noted that not all nuances are applicable to all users as some users are left-handed or have different length arms. There might also be properties of users that can be recognized so as to compare users and build categories to suggest how they might prefer to interact. Using this, a user entering a VE for the first time can have an interface personalized for them because of the nuances of users like them. Much work can be pulled from the user modeling community.

3.3 Guidelines for Nuances

As can be seen from the work that nuance-oriented VEs must do, there are certain requirements that are placed on such a system. These guidelines can be adapted from guidelines set forth in similar fields such as gesture recognition [Rubine92] and automated user interface design [Eisenstein00][Ruvini00]. They are easily remembered by the cumbersome acronym A FUR STAGE: accurate, fast, understandable, refining, sensitive, trainable, attribute-based, general, extensible.

3.3.1 Accurate

The VE needs to be accurate in its ability to recognize when a nuance has occurred. Incorrectly identifying a nuance may make an incorrect action occur for the user. This could increase frustration as well as produce incorrect results. As stated earlier, it can be

difficult to implement an undo feature in VEs, so incorrect results can be very painful to correct.

3.3.2 Fast

Response time greatly affects user satisfaction with an interface [Baecker87] and in VEs, most systems are already heavily taxed so any extra processing greatly affects performance. Many ML algorithms are not created to be run in real-time so the algorithms need to be specially handled or selected. Another solution might be distributing the computation but that is left up to the implementer.

3.3.3 Understandable

Users need to be able to perceive that a VE actually used their nuance in the interaction. If users feel that their nuance was ignored, they will stop making such actions and the nuance will go unused. This is not necessarily a conscious action on the part of the user as some refinement nuances simply expand on the boundary conditions of interactions.

3.3.4 Refining

Most interactions already have known affordances and users have existing knowledge. This should be supported by nuances and not rebuilt by them. The nuance interfaces should only increase the accuracy and robustness of the interface in addition to handling special cases.

3.3.5 Sensitive

Most nuances will be slight modifications to an existing interaction that a user makes, so VEs will have to be very sensitive to recognize when a nuance has occurred. This is made difficult by the guidelines of being accurate and understandable because we must notice every detail and avoid mistakes while accounting for each nuance whenever it first occurs. When performed properly, in some cases the user may not even notice a change in the interface from a new nuance; they will only notice that they are struggling less with the interface.

3.3.6 Trainable

VEs need to recognize quickly that a nuance has occurred from the input data. If not, then users will stop making such actions and the possibility of supporting users with the nuance will be lost. Additionally, some VEs might want to customize themselves to individual users or groups of users over time. Since nuances will hopefully be carried over from one environment to another, it will be important to have the transplanted nuance train for a specific environment's details in addition to just the general case.

3.3.7 Attribute-Based

The nuance itself should convey information from its size, duration, location, orientation, etc. Some nuances might be purely boolean but we should assume that most would apply some amount of numerical modification to an existing interaction and should be a function of user action. So, if the user does a nuance movement in a large sweeping motion, then we should increase the amount of change the nuance coveys. This is part of the functional representation of nuances.

3.3.8 General

Nuances should be identifiable in various sizes and orientations because users are of various sizes and orientations. A nuance that only works when the user is performing it along a certain axis is not general, as the chance that the user will be facing along that axis every time is not high (unless that is a specific characteristic of the environment). This may convey some attribute but it will also be due to the fact that users are not consistent and are error prone. So, if the nuance is to be useful, it needs to have a form general enough to be easily performed and structured enough to not be accidentally triggered.

3.3.9 Extensible

The nuance should be extensible to new interfaces both in the same and other VEs, and could be device independent. There might be a need to recognize a VE-specific nuance, but most likely an action in one VE will be performed by the user in another, especially if the user becomes accustomed to that nuance.

3.4 The Formative Nuance-Oriented User-Centered Evaluation Cycle

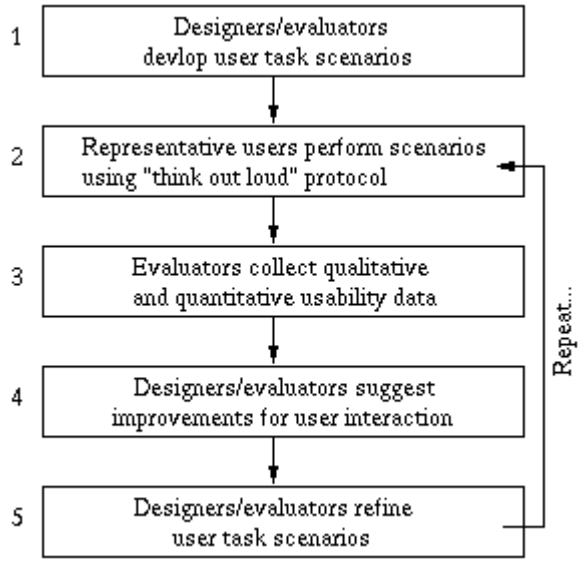

Figure 6 The Formative User Centered Evaluation Cycle. [Gabbard99]

It is my hope that some day, the creation of a VE will entail a bit of hand-waving and "poof", a VE will appear, built from expert guidelines and user models, extracted from databases of user nuances, and updated on-the-fly as users gain experience in the new VE. This type of a scenario requires near AI levels of intelligence on the part of the computer and thus we should develop a process to work with nuances in a more computationally tractable manner. The most promising approach would be to incorporate nuances into a user-centered design and evaluation cycle such as the formative user-centered evaluation cycle[Hix99][Gabbard99]. See section 6.3 for a discussion of on-line vs off-line learning.

3.4.1 The Formative User Centered Evaluation Cycle

The formative user centered evaluation cycle (FUCEC) is composed of five steps (Figure 6). The first step requires the designers and evaluators to create user task scenarios. This is followed by the second step where users perform the scenarios with the "think out loud" protocol. In the third step, evaluators collect usability data and then in the fourth step, the evaluators and designers suggest improvements to the interface. In the last step,

the user task scenarios are refined and the process returns to the second step. This cycle is repeated as necessary. The benefits are an interface designed to work with the user and not imposed upon them, a major impetus for nuance-oriented interfaces too.

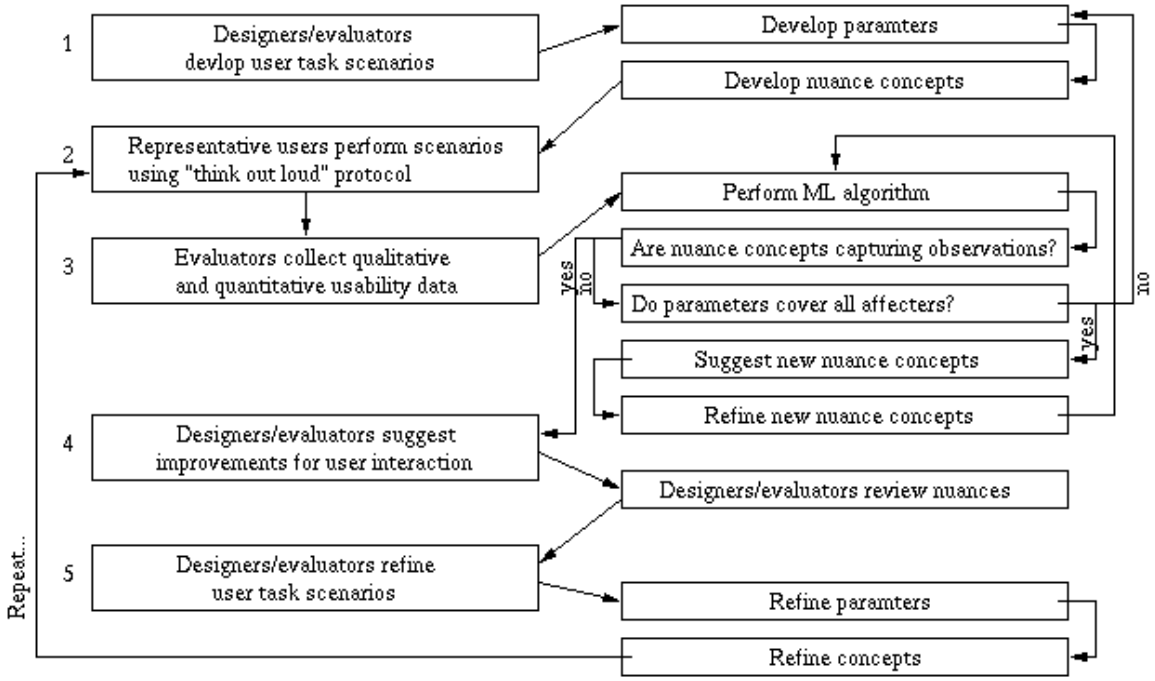

Figure 7 The Formative Nuance-Oriented User Centered Evaluation Cycle.

3.4.2 Incorporating Nuances Into the Cycle

Because the user is the important factor in the FUCEC and it deals with iterating the interface, it is a natural fit for incorporating the nuance-oriented framework. In this sense, the nuances can be seen as another expert, generating suggestions and refining the interface, but it does raise difficulties. For instance, nuances require ML algorithms which are difficult to work with and require the designers and evaluators to help mold and shape the learning process. Therefore, the major change to the cycle is to have the designers and evaluators focus on the nuances as well as the interface. There are six major changes to the FUCEC as shown in Figure 7.

3.4.2.1 Change 1: Identify Initial Atomic Nuances

The first change is after the first step in FUCEC. After task scenarios have been created, the hypothesized parameters and concepts incorporating those parameters need to be identified. This allows the parameters and concepts to be built upon user scenarios. After this, the cycle goes into step 2 as normal.

3.4.2.2 Change 2: Enrich the Nuances

Chapter 4 The second change occurs after step three in FUCEC. At this point, the designers and evaluators have collected qualitative and quantitative data.

This data should be recognized in the ML algorithm and generate many of the same observations as the experts in the form of nuances. If not, the parameters and concepts should be massaged until they do. The method is to see if the concepts are at fault in missing the observations. If this is the case, we must suggest new concepts to replace or add to the system, then reapply the ML algorithm. This may require adding new parameters to the data that we are collecting, in which case we need to revert back to the beginning with developing parameters and concepts. We repeat this process until we are satisfied that we are getting useful results.

3.4.2.3 Change 3: Expert Review of Nuances

The third change is after step four in FUCEC. At this point, the designers and evaluators have suggested improvements for the interface and now can review the nuances that the ML algorithm has generated. This is where useful nuances can be added to the interface or modified to match what the designers and evaluators consider useful. The designers and evaluators might have recognized that a concept exists but not the specifics of the relationship. For example, they might have recognized that the user likes to err in a certain direction when they point but not the specific amounts of error. Also, this step will remove obviously incorrect nuances such as those that are overly specific, overly general or flat-out wrong. For example, designers and evaluators might have noticed that users err but not that they err differently depending upon the shape, size or distance of the object. They may have learned that every selection was made with the right hand which should be generalized to work depending upon which handedness the user is. There may also be useless nuances learned such as every selection was made with one hand pointing down. An example of a completely incorrect nuance is one that selects an object if it is blue. This nuance could be supported by the example data but is obviously wrong (unless this is something specific to that VE).

3.4.2.4 Change 4: Refine Parameters and Concepts

The fourth change is after step five in FUCEC where the designers and evaluators refine scenarios to incorporate all the changes. Since the scenarios might have changed, we need to make sure we make any necessary changes to parameters or concepts. With new scenarios, the nuances that were previously useful might no longer be, so the usefulness of old nuances and data is unknown but the iterative process will make sure that only the best are incorporated into the final interface.

3.4.3.5 Changes 5 and 6: Changes to Data Collection

There are two changes that need to be made to step 3. The first is as interfaces become better through iteration, it may become hard to tell if an improvement has taken place. For instance, many users may not even notice a change in the interface but will take advantage of it unknowingly. This problem was identified in a dialog system: “Thus the differences … probably fell below the level of conscious awareness for most subjects. We believe this is a general problem for systems … without gross infelicities.” [Tsukahara01]. They solved the problem by having the user listen to a recording of the system which they called ‘evaluation after relistening’. They later state this “… is an effective way to amplify weakly-detected user preferences.” Using this type of relistening, users may be able to judge a new interface after the experts have redesigned

it. The second change is to log quantitative data from the environment in step 3. This is not normally done because quantitative data is usually gathered by experts counting certain events. The machine learning requires the input data of the user and the location of objects in the environment, event over time.

3.5 Summary

This chapter has explained nuances but carefully ignored the actual methods of using ML. This will be covered in the following chapter where the subject of ML can be introduced and its applications to nuances explained in more depth.

Chapter 4 The Usage of Machine Learning

Machine learning is defined by Tom Mitchell as, "*…the study of computer algorithms that improve automatically through experience*." [Mitchell97] It is a sub-field of artificial intelligence comprising several of its own such as Bayesian networks, genetic algorithms, inductive logic programming, inverse reinforcement learning, neural networks and decision trees among others. Each of these ML has advantages and disadvantages in what it can learn, termed the induction bias models [Mitchell80], and how they go about doing it.

The first part of the chapter will focus on the types of machine learning. Also, up to this point, we are operating on the belief that ML can capture the nuances so we will discuss our existence proof performed using a neural network (NN).

4.1 Models of Machine Learning

Many types of machine learning exist, each with advantages and disadvantages as stated earlier. There are two important concepts to note in ML. The first is that a generated solution in ML does not prove a situation. It merely presents a solution given the representation and example cases in the hope that past examples are an indicator of future actions. This is important, as we will require experts to review any induction through a ML algorithm. This leads to the second concept which is ML is not a magical solution where data is put in and answers are forthcoming. It requires time and hopefully the results through ML are faster achieved than through human review. It will require time to build systems capable of what we want but since it has been successfully employed in several problematic domains, ML should work for nuances too.

4.1.1 Neural Networks

Studying the brain's structure first inspired neural networks [Bishop95]. The construction was of input nodes connected by paths to output nodes with the possibility of layers between. The data is placed in the input nodes and operated on through the weighted connections to have an effect on the nodes of the next layer, eventually ending in the output nodes. Creating a representation can be difficult and so too the deciding on the number of nodes in each layer. Training a neural network can be done through many methods such as back-propagation or genetic algorithms but these methods are all just ways to change the weights of the links between the nodes. At this point, training examples are given to the learning algorithm and the weights of the nodes are set to approximately fit the space being represented with the network. The problem is that the

final functioning of the NN is very hard to decipher, which is a problem for applications that require readability. Since NNs require very little understanding of the problem domain, they are heavily used in situations where there is no other option. The NN also has the advantage of dealing with noisy data well.

4.1.2 Bayesian Networks

Bayesian statistics [Heckerman96] can be used to generate an understanding of a problem domain. The statistics give a predictive probability of a new state given the existing state through a network of causal relationships. The problem with such a system is generating the initial structure of the network when no data is known. Such a system will require lots of training data to become sound but is much more readable than a neural network because it has a formal mathematical basis. The representation of graphical connections is in the same but the learning algorithms and interpretations are different from that of a neural network.

4.1.3 Inductive Logic Programming

Inductive Logic Programming (ILP) [Muggleton92] is a rule-based inductive process using a first order logic representation. It is able to represent a large number of possibilities and generalizes to new situations well. The learning also could potentially require few test cases. The problem is the difficulty in trying to induce the rules. Also, dealing with numerical data is not handled well and requires other areas of learning such as Constraints-Based Reasoning. The readability of the resulting rules from ILP is very valuable.

4.1.4 Decision Trees

Decision trees [Russell94] model data by classifying data into nodes of a tree where a criterion splits the data. Decision trees are easily read and understood by humans but are unfortunately limited in what they can represent; any decision tree can be represented as a rule-based system but the reverse is not true. Algorithms for decision trees are very dependent upon the initial criteria chosen and also have trouble with large data sets. This can be remedied to some extent using decision trees that allow for errors through post pruning or other methods.

4.1.5 Inverse Reinforcement Learning

Inverse reinforcement learning [Ng00] deduces a reward function from observed behavior. It is not a type of machine learning but a method that creates a reward function in a Markov Decision process so it can be used around the afore mentioned learning methods. A Decision process is where there exists a set of states S, a set of actions A and an agent, in this case the user. At each step, the agent knows its current state and chooses an action, receiving a reward which is a function of the state and action chosen ($r = (s,a)$ where $s \in S$, $a \in A$). A Markov process is one where the transition probabilities are stationary.

4.2 Neural Network Existence Proof Experiment

We wanted to create an example system to test our theory that users employ nuances and that these nuances can be recognized and used to enhance interaction [Wingrave01]. We chose to focus on the simple task of arm-extension selection because it is understandable by the user and users have existing knowledge and mental models of this task thus making the task interesting for study. As a preliminary exploration of the modeling choices in nuance research, we employed a NN and the JIVE Toolkit [WingraveJive] built on top of DIVERSE [Arsenault01].

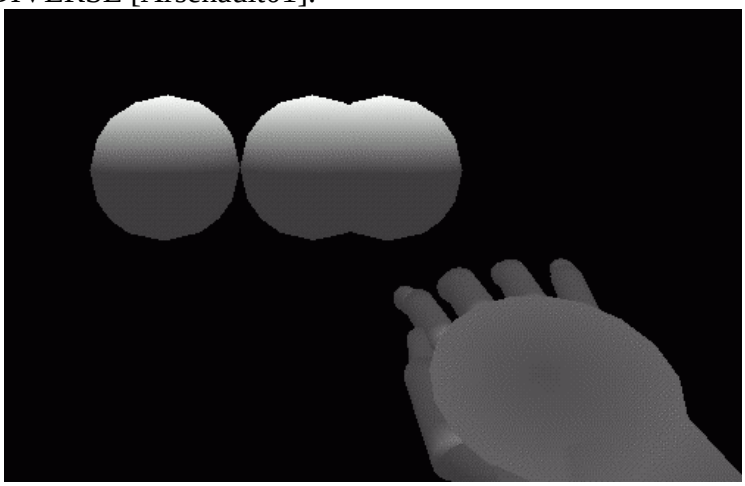

Figure 8 The proof of concept VE with three balls, two of which are overlapping, and the user's hand. When the user pinches his fingers, the VE will tell them which ball they were trying to select based upon the trained NN.

The environment consisted of three balls placed slightly out of reach of the user in a horizontal line in front of the user. There were two blue balls and one red ball and the user was to select the red ball using whatever they felt would distinguish that ball from the other two using their tracked hand (Figure 8). Fakespace Pinch Gloves were used to indicate selection at the hand's current location. There were seven (7) possible positions for the balls and neighboring positions had the balls partially overlapping to increase pressure on the user to select precisely. The data collected was the position of the user's head and hand, as well as the position of the balls and which was red, at the time of the pinch.

A NN was used and chosen because NNs are easy to train and are fairly fast to run when plugged into the environment after being trained. They do not fully match our guidelines for a nuance system, so they are not suggested for full-scale nuance-oriented systems but their ability to handle a wide variety of situations in a proof-of-concept system was considered more important than their shortcomings. The collected user data was normalized and split into a training and a test group. The training data for the groups was fed into a feed-forward NN with one hidden layer and three outputs. Each output gave the level of selection of the corresponding ball. Training was performed with back propagation; then the trained network was loaded into the same environment with three balls. After each pinch, in the test environment, the NN printed which ball had been selected.

The results were both encouraging and frustrating. The NN worked well when used in exactly the same manner in which the testing data was taken. It very easily distinguished the object the user was trying to select, even when all three of the objects were overlapping and the center object was to be selected. While trying to understand the NN, it appeared to take advantage of how the user's hand was oriented while making selections, a fairly subtle but effective technique. However, a small change, such as the user standing on a stool or kneeling for example, changes the arm position and the manner of selection so the NN very quickly leaves its trained area. While some of these aspects could be addressed by using translation invariant testing, NNs have other disadvantages including (i) excessive dependence on the original network topology (and hence, experimental setup), (ii) the black-box nature of their function (rendering their results inscrutable), and (iii) their capacity to incorporate knowledge in only a limited form. Additionally, the training of the NN takes quite a bit of computation and requires much training data. Because of this, full user trials were not considered practical and no statistical results are reported here.

The conclusion was that NNs were not feasible for large-scale nuance-oriented interfaces in the representation we used as we had predicted, as simple changes to the VE would require a completely new dataset and more computation. Also, collecting data for all the probable situations is not practical as we shall show. The concept of nuance interaction has successfully been shown, however. Users do operate with nuances and an interface can make use of this to increase precision.

So why do we say that our representation is not practical? In our representations, the time of training and the amount of data required grows as a function of the size of the inputs, so as we raise the maximum number of objects or increase the space in a VE, we are increasing the amount of work required. If we fix the number of objects in an environment, we must train our model for all the environments that are smaller than it. The reason is that object-to-object interactions occur in VEs. For example, selecting a single object from among a group of objects is not done without regard for the other objects. Users will put nuance information into their hand movements to select the object they desire. Whether that is angular error of a ray cast from the user's finger in a direction that contains no other object or the motion of reaching around occluding objects, that nuance information is there but ignored. This means that for n objects, we must train n environments. Another problem is that users will treat objects differently because of the object's affordances. For example, users selecting a cylindrical object will select differently than if the object were a ball or shaped like a pitcher. To generalize, for r different types of objects in an environment with n objects we get n^r cases to train for and since we must do this for every environment smaller than the one we are in, we get $1^r + 2^r + ... + n^r$ cases. Using the binomial series we can rewrite this as $\left(\frac{n(n+1)}{2}\right)^r$ cases. For an environment of up to five (5) objects and of three (3) types, we get an unwieldy 3,375 cases to train.

4.3 Summary

There are many problems to be looked at for a nuance system. Much of this work so far has been a thinking exercise of imagining future problems and planning for them with chapter 3 considering the VE interface aspects and this chapter the ML aspects. The following chapter is the experimental chapter where we work with users to investigate the nuance framework.

Chapter 5 Chronological Plan and Results

To try and prove some of our ideas about Nuance-Oriented interfaces in VEs, we created two plans to investigate the interaction. The first plan consisted of four phases with each phase exploring each type of nuance. This plan was replaced with a three phase plan for reasons to be explained. The results of these phases were counter intuitive but very illuminating nonetheless.

In all the phases, we had the same group of eight users, all taken from a graduate level course on virtual environments, though not all were computer science students. All had some familiarity and interest in the field of virtual environments but not necessarily experience. There were five males and three females between the ages of 24 and 54, with most towards the lower end of the age-group. Their compensation was receiving extra credit in the graduate level course.

We start this chapter explaining the software and hardware we used and then go into the experimental phases. Each phase starts with an explanation of the purpose followed by the description of the environment. Sections appropriate to each phase follow this and then we conclude with sections explaining the data collected, results and what was learned.

5.1 Software and Hardware

5.1.1 DIVERSE

DIVERSE (Device Independent Virtual Environments Reconfigurable Scalable Extensible) was developed at Virginia Tech as an API for Virtual Environment development and research. It is released under the GPL and runs on IRIX and Linux systems.

DIVERSE is built in two parts, dtk and dgiPF. The dtk stands for DIVERSE ToolKit and manages typical hardware devices through networked shared memory. This allows for easy debugging by manipulating the shared memory through a graphic interface of sliders and buttons during testing. dgiPF stands for DIVERSE Graphics Interface Performer. It focuses on managing the different display types from desktops to multi-walled CAVEs. DIVERSE also allows for interaction in the environment to occur through dynamically shared objects (DSOs), so loading new interfaces, hardware, objects, etc. can be performed without recompiling. For my experiments, this was not necessary.

5.1.2 JIVE

JIVE stands for Just In a Virtual Environment and was developed at Virginia Tech by the author. It was originally built on top of Limbo [Leigh01] and the CAVE Libs [Cruz-Neira93] but when DIVERSE became available, it was built on it and Performer. The purpose was to create a simple API for building VEs. JIVE implements interaction by events cascading through the system of dependencies. It contains a library of interaction techniques and devices, and provides a simple interface for dealing with objects and implementing new interaction. It is highly object-oriented and still in development.

5.1.3 Virtual Research's V8 Head Mounted Display

Virtual Research's V8 Head Mounted Display is worn by the user about the face and adjusted by three dials for comfort and clear vision. The HMD has a 640x480 resolution with a 60° field of view. It also has the ability, which we did not take advantage of, to operate in stereo.

5.1.4 Polhemus's 3Space Fastrak

Polhemus's 3Space Fastrak is a magnetic six-dimensional tracking system. In these experiments, it was used to track both of the user's hands and their head. The user was told to stand close to the transmitter, which was on a platform, and the area in front of that transmitter was tracked.

5.1.5 Fakespace's Pinch Gloves

Fakespace's Pinch Gloves are a pair of gloves with magnetic material on the tips and back of the fingers and thumb. This conductive material, when placed in contact with another piece of material on the same glove or the paired glove, completes a circuit and tells the computer that; 1) the user is pinching and 2) which fingers are being pinched. In these experiments, the pinch gloves were also tracked with a Polhemus tracker so that the user's hand location was known by the system.

5.2 Original Plan: 4 Phases

Our original plan was of four phases to generate an optimal selection technique based upon the concepts of a nuance-oriented interface. The goal of this process was to create the rules and adjustments on selection techniques based upon regions of space, attributes of the object being selected and properties of the environment. In each phase, using rules and findings from the previous phase, selection trials were to be performed to collect data from the users. The reasoning for breaking the experiment into these steps is that performing all the experiments simultaneously encounters far too many variables to be isolated at once.

In the first phase, the user was told which technique to use for each trial. This was to allow the tuning of a set of interaction techniques for use in the later phases. The second phase tunes selection based upon regions of space. It was the hope that the optimized selection techniques from phase one could be used to isolate how people liked to perform selection in space. These preferences would then be turned into maps showing which

metaphors the users tended to use for selection in each general area. Phase three was to account for attributes of each object with the selection maps of phase two and tuned selection techniques of phase one. Phase four was to enter environmental factors into the rules as to how selection changes due to interaction with other objects. Only phase one was performed.

5.2.1 Original Phase 1: Tuning Selection Metaphors

Given a base set of selection metaphors, we can tune them to fit the user thus making a more usable system. The techniques chosen were ray casting, occlusion selection and arm extension. The arm extension technique was originally going to use three different functions for scaling the location of the hand: one-to-one linear, scaled linear, and Go-Go. This was changed when we decided to remove the hand from the scene since the hand provides feedback, which affected how the user interacted.

Each technique has tunable values for its selection and the purpose of this phase was to discover what values the users preferred. For ray casting and occlusion selection, the tunable values we used were the offsets of the rays extending from the hand or eye in the yaw and pitch dimension. For arm extension, tunable values were the x, y and z positions of the hand. Simple statistics were to be used to discover the optimal values from the collected data. This tuning also was designed to take into account some of the consistent errors that user make, such as the difficulty with depth as stated in section 3.2.3 under refinable nuances.

5.2.1.1 Environment

The environment showed the ground and the shadow of the ball on the ground for depth cues. The hands were shown and the ball was positioned inside one of the twenty-seven general locations, in a 3x3x3 grid, in front of the user and moved to each location randomly after each selection on the user. Before each technique was tested, the user was told about the technique and how it worked. They then performed the selection trials.

5.2.1.2 Hypothesis

There should be well-defined error values in the selection techniques for which we can tune the interfaces for the further phases.

5.2.2 Original Phase 2: Tuning for Regions of Space

Once we had tuned the techniques, we planned to allow the user to choose a technique for the selection of an object. The object to select in each trial was to be a sphere placed in twenty-seven general regions as before. The hope was that users tend to use certain metaphors to select in each region and that we can mine the data to produce 3D selection maps of the preference of users.

5.2.2.1 Experiment

The users were to be told they are entering an environment where a ball will be shown. They were to use whatever selection technique they prefer to select the object. They were not to be allowed to walk towards the object.

5.2.2.2 Hypothesis

Novice users were expected to use the same interaction technique through this phase. They might, towards the end, become more natural and start mixing the techniques. On the other end, the more experienced users were expected to like a certain type of selection technique and therefore not mix them. It was the users that had a bit of experience that were expected to mix techniques the most. Those users that do mix the interaction were expected to most likely use arm extension and occlusion selection for middle distance objects, arm extension for close objects and ray casting and occlusion selection for distant objects.

5.2.3 Original Phase 3: Tuning for Attributes of the Object

I believe that certain objects afford techniques for selecting them. If that is true, then we should be able to develop rules based upon the technique and affordances of the object. Basically, these rules will increase or decrease the weighting of certain objects to be selected. The difficulty will be in identifying the affordances of the objects. The best we can do is test a few affordances and see if they have merit since the identification of all is beyond the scope of this phase.

The affordances of objects that we considered testing were the orientation of different shapes (beams, cylinders), the size of the shapes (large/small), dimensional differences (narrowness, circular, donut), color, curvature of the surfaces, sharp looking surfaces, textures, hot/cold looking, handle-like, spiked, similarities to existing objects (umbrella/pots and pans), etc.

5.2.3.1 Experiment

The users were to be informed of the same information as before regarding the selection techniques. They were to then have some number of trials with objects with certain affordances. Trials of similar affordances were to be spaced so the user does not get "bored" with selecting each item and start selecting without paying attention to the affordance. Some of the scenarios might be:

- orientation of an object compared to hand (narrow pencil-like objects)
- curved surfaces (cylinders), hemispheres and cones
- handles (lollipop, protrusion, barbell)
- painful (something that looks hot(flames)/cold(dry-ice effect)/sharp(knife blade on one side)/spiked (porcupine))
- real-world techniques for shape (Christmas tree, bowling ball, trashcan)

Pilot tests were to be performed to decide which affordances were the best to study.

5.2.3.2 Hypothesis

Certain properties such as "handle-like", or "uncomfortable to touch" or "real-world technique" were expected to arise. The main purpose of this phase was to show that these affordances exist and let future research quantify the results.

5.2.4 Original Phase 4: Tuning for Environmental Nuances

Certain properties of the environment will impact how the user selects with the techniques. For instance, if the user is selecting an object that is close to another, they

might purposefully err away from the object they do not want to select. The effect would increase the error on the object they wish to select but the hope is that the other object's error would increase more. Other environmental nuances might exist in observing occluding objects. This phase was to require machine learning to help recognize the rules.

5.2.4.1 Environment

There were to be several objects in the environment of various shapes, sizes and affordances with the user, a change from the previous phases. One object was to be shown alone in the environment to the user for a brief period of time and then the rest of the objects were to be entered. The user's task was to select the object that was shown first. This was to keep users from entering into a video-game state of mind where their actions become reflex-based and not cognitive (The belief being that this is closer to a normal environment's interaction). The existing selection maps and tuned techniques from the previous phases were to be used.

5.2.4.1.1 Experiment I

The users were to go through a predefined number of trials of randomly generated scenes of objects. The data was to be recorded and mined.

5.2.4.1.2 Experiment II

The users were to be placed back into the same environment two more times after the data had been mined from experiment I. One time was to be without the rules generated and the other was to be with the generated rules (order permuted). User actions were again to be recorded but this time the trial would not end until they successfully select the correct object using a tuned selection technique. They were to be asked afterwards if they liked the rule-based environment more than the non-rule-based.

5.2.4.2 Hypothesis

Users were expected to develop strategies to adapt to occlusion and closeness such as going around an object to point. It was expected that not all users will use strategies but some will and after the environment realizes those strategies, users will exploit the rules through the rest of the phase. It was hoped that the users would like the rule-based environments better.

5.3 Phase 1: Optimizing Selection Techniques

Phase 1, as we have already stated, was to discover refinable nuances for three VE selection techniques: arm extension, ray casting and occlusion. Most interface designers stop at these high-level techniques and do not try to tune them for the user beyond the obvious. For example, ray casting is almost always implemented with a ray extending directly from the hand without every trying to discover if this is the optimal configuration. For each technique, users were told how each technique worked in wording vague enough to not guide their actions but informative enough to let them know how it works and is implemented (Table 1). The concept was that users have an existing model of how they wish to interact with the environment and if we isolate that underlying model, then we can use that knowledge to recognize their actions, reducing the Gulf of

Execution [Norman90]. This information can then be used in further phases where we isolate other factors of the user.

Ray Casting The Ray Casting technique involves pointing a hand at an object and pinching when you believe that a line extends from your hand to the object. **Arm Extension** The Arm Extension technique involves reaching your hand out to the object to be selected. When you feel that your extension has indicated the object that you wish to select, you will pinch your finger and thumb. This is rather vague so discuss this technique with the researcher to make sure that you have a good understanding of it. **Occlusion Selection** The Occlusion Selection technique involves placing the tip of your finger between your eye and the object so that the fingertip occludes or covers the object you wish to select.

Table 1 The wording of the selection techniques to the user. The descriptions were meant to be informative but not enough to actually guide their usage of a technique.

The difficulty we encountered was to make an interface where the user would act naturally and not adapt. To this end, we attempted to remove all forms of feedback from the environment relevant to each selection technique. For this reason we did not implement ray casting with a ray extending from the user's finger or even a hand for arm extension. We then assumed that since the user was operating according to their own definition of optimality, and we knew their goal to be the selection of the orange sphere, then each time they conclude a selection with a pinch, they were correct in their selection.

We had the opportunity to use real world objects, like chairs, lamps, cups and such. This would have helped give the environment a more realistic appearance and helped the users become oriented in the space around them but the affordances of the objects we felt would alter the ways in which the users interacted. Since the goal was to understand how the users worked without any external influences, we used orange spheres.

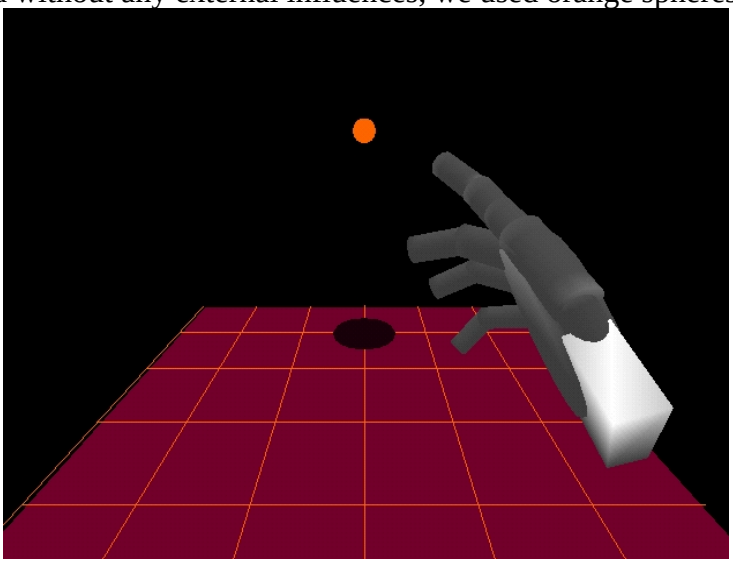

Figure 9 Phase 1 with the ball at a distant position. Notice the shadow of the sphere and the gridded floor.

5.3.1 Environment

The environment (Figure 9) in all three sets of trials, one for each selection technique, had the user standing on a platform overlooking a floor with one orange sphere that they were told to select using the selection technique that was currently being tested. Their head was tracked and a virtual hand, at the same position and orientation as the user's physical hand, was shown except in the arm extension technique. It was felt that the hand being shown would alter the way in which the user acted since the hand is the major form of feedback in arm extension (see 5.2.1). To account for a lack of depth cues, the users were told the sphere was the same size throughout the experiment and that the floor was a grid of one-meter squares. There was also a shadow, properly scaled for depth and approximately scaled for height, placed below the sphere on the ground. Each set had 30 trials where the sphere was moved through different locations with the first three trials being the sphere at its furthest distance, middle distance and closest distance to the user to help them get an idea of the environment's depth. Twenty-seven trials had the sphere randomly located at a position of near, mid and far; low, level and high; left, center and right. The final eight had the sphere positioned very near the user at a position of close left, center and close right; close low, level, and close right. The center, near, level was only performed once and the positions can be seen in Appendix B. One side effect noticed in the pilot study was that users were able to cycle quickly through selections because of our assumption that each episode was correct. To counter, we added a three second pause between each selection episode and added an audible sound when the orange sphere reappeared.

5.3.2 Data Collected

The data from this phase was the position of the user's head and hands as well as the position of the sphere at the time of pinch. This data was then used to discover if users made consistent errors based upon the location of the intended object of selection and the selection technique. Clustering techniques such as k-means were then to be used on the data to group user actions.

5.3.3 Results

With users free from the feedback of the environment, we expected them to revert to their most natural model of interaction built off of innate and proprioceptive intuition. What occurred was an amazing display of adaptation on the part of the user, completely unnatural and inefficient but incredibly adept at making use of the scarce feedback that was left in the system. As an example, one subject spent the entire occlusion selection trial making selections with their palm facing out. This is a very uncomfortable position, even for short periods of time, and especially for objects elevated in the environment. Most importantly, the palm being turned out *completely* occludes the environment thus reducing accuracy.

There were some supplemental nuance strategies observed. One was that users tended to roll their wrists clock-wise when selecting objects that were high or close to them. This was predicted in the column selection example in 3.2.1 and was nice to see in the data. Another was that since the objects being selected were spherical and since the palm of the hand was circular, some users attempted to position their hand such that the sphere was perfectly occluded by the palm of the hand achieving an eclipse, much like aperture based selection. Another strategy was to switch to pinching with the middle finger and thumb instead of the index and thumb to make a selection, thus avoiding the "Heisenberg effect" (Figure 10) of selection [BowmanHCI01]. This is when performing the action to signal an event induces errors.

Figure 10 The "Heisenberg" effect of making a selection induces errors in the orientation of the tracker device.

Users of arm extension were found to not have a concept of depth. We expected users to scale the extension of their arm to the objects being selected but found that users only divided space into "far" and "near" with far being a fully extended arm and near being a half-way extension. We were hoping that since objects appeared at different depths over time, the user would learn this and scale accordingly. The lack of other objects in the scene at the time of pinch could have made indicating depth unnecessary so the result may be inconclusive. Users found it very unsatisfactory not being able to see their hand as could be expected. Also, users were very slow and hesitant compared to the other two selection techniques. Because of this, the usefulness of arm extension was questioned since it requires specifying depth and users do not seem to have a good grasp of it except through proprioception [Mine97].

Occlusion selection contained the most interesting results. Since we did not remove the hand from the scene, users had almost all the feedback of the full implementation. Because of this, we expected nearly optimal usage. The users however choose unusual points on the hand as the occluding points. The two most common were actually the palm of the hand and the knuckle where the thumb meets the hand (Figure 11). The palm of the hand occlusion technique occluded most of the scene making the accuracy very low. The thumb knuckle technique is inaccurate and again occluding. It does however leave the hand in a natural and thus

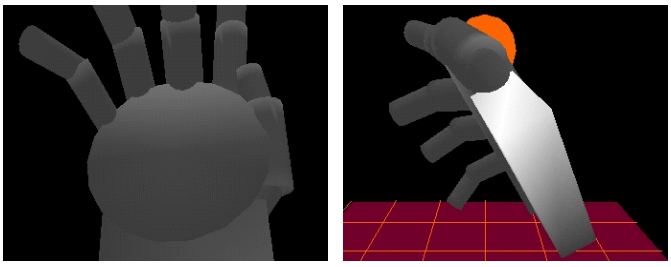

Figure 11 Two occlusion selections used most commonly in phase 1. Left is the palm occlude (with the sphere behind the palm) and right is the thumb knuckle occlude. Both are inaccurate and highly occlude the scene but for some reason users converged to them.

non-fatiguing state. A few users did choose to use the more accurate and less occluding fingertips.

For ray casting selection, only one user did true ray casting. All the other users occluded the object with the tip of their finger and considered that pointing at the object (Figure 12). This completely voided the concept of "shooting-from-the-hip" to reduce fatigue but with the lack of a ray extending from the fingertip, this provided the most feedback to the user.

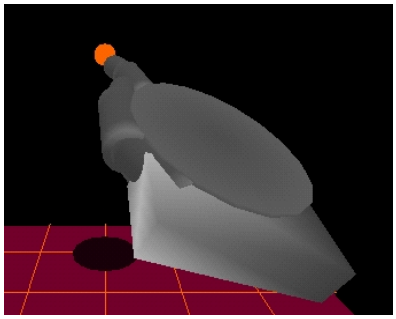

Figure 12 All but one user considered ray casting to be like a fingertip occlusion technique.

5.3.4 What We Learned

The results lead to the following conclusion:

> *Users largely do not have a model of interaction with the environment but a model of how to respond to feedback the environment provides.*

Stated another way, users attempt to align their actions with feedback and not their senses. The effect of user experience with VEs may play an important role in this conclusion.

Also, it was interesting to see that users started to hold their fingers in the same position of the model hand. This might be a method of guiding users because if they feel the need to imitate, we could guide them using this.

Our original intent was to build personalized selection techniques for the users. After reviewing the results, it was not considered possible to use the data since the users were so inefficient with their interaction in virtual environments without feedback to guide them. Clustering was performed to see if trends existed in user data but the trends just mimicked the observations. Because of the errors in our assumptions, we replaced the four phase plan with a new three phase plan with the 1st phases the same.

5.4 Phase 2: Optimizing Selection Techniques II

If users do not have a model of an environment but of how to respond to the feedback it produces, then the factors we should be isolating are those of the feedback. So, how do we go about dealing with the problem of the adaptability of the user, since they will adapt to even a poorly tuned interface? To do this, we need some sort of pressure to improve as simply asking the users to do so will create slight tuning and much adaptation. We also require some notion of getting users to search for alternate possibilities of selection within a technique. Looked at from a ML perspective, there are several local maxima of the tuned techniques the user can create and we need to make sure they search through many of them to find their global maxima. Another problem is making the system friendly and usable so that users will not mind staying in the environment long enough to tune the system to their needs. Lastly, what types of feedback should the selection techniques have and what parameters of the interaction techniques should be tunable?

To provide pressure to improve, we framed the selections into sets of selection trials where the user was told to select as quickly and accurately as possible. At the end of each set, a qualitative ranking of their performance was returned to the user in the hope that the competitiveness of the user would make them want to achieve better and better rankings. To provide pressure to search, we asked the users to try several predefined configurations of the selection technique. Each configuration was either created by an expert or was a typical method of selecting from Phase 1. This was done to see how phase 1's results compared against the expert configurations now that feedback was enabled.

To make the environment user-friendly and non-threatening, we did several things. The first was to have the researcher direct the user through the experiment, answering questions along the way. We also allowed the user to work at their own pace and change the predefined paths of the experiment by being able to redo a trial with the current configuration or go back to a previously tried configuration. Because of the need to have the user control so much of the experiment, we implemented the TULIP menuing system [Bowman01] to handle the experiment control. It was chosen because it has low fatigue, allows the occluding hand to be removed from the scene easily and is fairly fast. These properties of TULIP outweigh its steeper learning curve, and with the researcher able to answer questions, users quickly understood TULIP's use. Lastly, personalization of a selection technique was done by selecting a parameter of that technique to tune using TULIP and then rotating the left hand to change the value.

5.4.1 Environment

The environment was a 3x3x3 array of 27 light blue cubes placed in front of the user. The head and hands were tracked and the user was wearing PinchGloves™. In the occlusion selection environment, there was a bull's-eye on the hand and in the ray casting environment there was a ray extending from the fingertip. The left hand was labeled with the TULIP menuing system at all times and its submenus were displayed on the right hand when the choices need to be displayed. The three menu types were: "Configure", "Trials" and "Personalize". The "Configure" menu had seven configurations for occlusion selection and six for ray casting. The "Trials" menu allowed the user to select a set of 2, 4, 10 and 27 selection trials as well as to stop the current set. The options of the "Personalize" menu will be explained below. The environment displayed text to the user such as the ranking they receive for each trial, if a selection was correct or not and when each trial started.

There was one environment for each selection technique. In each environment, the user was asked to do at least one set of 10 selections for each predefined configuration. They were then asked to rate that configuration on a scale of 1 to 5. After going through all the configurations, they were introduced to the methods of personalizing the interface. The researcher encouraged them to do several small sets of trials with each personalized technique and then at least two full sets (27 selections) with their final settings.

5.4.2 Selection Techniques: Their Feedback and Tunable Attributes

There were two selection techniques in Phase 2; ray casting and occlusion selection. Arm extension was dropped from further phases because it is not well suited for selecting at a distance. This was due to the need to specify a parameter specifying the furthest distance an object can be from a user and also because users in the first study only had the concept of selecting "near" and "far", as explained in 5.2.6. Again, all selection techniques were triggered by either a middle and thumb or index and thumb pinch.

The passive feedback for ray casting was the ray extending from the fingertip. This guided users in their selection, such as when the ray passed over objects, so it was easy to see where the ray was in relation to the object. Active feedback was also added such that when the user's ray gets close to an object, the ray snaps to that object, changing the color of the ray and object. The properties that were tunable with this implementation were the yaw and pitch values of how the ray extends off of the fingertip with yaw being the angle across the fingers and pitch being the direction the fingers move when the hand closes. When the user tuned the values by rotating their left hand, the angles change immediately so they had an immediate feedback as to what the newly tuned position of the ray was. The user also had control over the snap-to angle and when they were tuning it, a cone representing the snap-to angle was drawn.

In occlusion selection, a user-facing, passive feedback, bull's-eye was attached to the hand representing the point where the ray from the eye passes through and extends into the environment. Additionally, there was an active feedback snap-to angle but instead of a ray snapping-to, the bull's-eye snapped-to and changed color. The tunable properties were the x (across fingertips), y (along fingers) and z (out from palm) positions of the bull's-eye in relation to the hand as well as the snap-to angle.

The predefined configurations for each selection technique are as follows:

Ray Casting Selection

Config 1: The ray extends straight from the fingertip with a 10-degree snap-to angle.
Config 2: The ray has negative pitch with a 10-degree snap-to angle.
Config 3: The ray has positive pitch with a 10-degree snap-to angle.
Config 4: The ray is straight but with a 40-degree snap-to angle.
Config 5: The ray is straight but with a 3-degree snap-to angle.
Config 6: The ray has positive pitch and heading with a 10-degree snap-to angle.

Occlusion Selection

Config 1: The bull's-eye is on the index finger and has a 10-degree snap-to angle.
Config 2: The bull's-eye is on the middle finger and has a 10-degree snap-to angle.
Config 3: The bull's-eye is on the thumb's knuckle with a 10-degree snap-to angle. This was a configuration that was used by almost every user in the first implementation.
Config 4: The bull's-eye is on the palm of the hand with a 10-degree snap-to angle. This was a configuration that was used by almost every user in the first implementation.
Config 5: The bull's-eye is placed a few centimeters off of the palm with a 10-degree snap-to angle.
Config 6: The bull's-eye is placed on the index finger and has a 45-degree snap-to angle.
Config 7: The bull's-eye is placed on the index finger and has a 3-degree snap-to angle.

Table 2 The predefined selection techniques for phase 2

5.4.3 Data Collected

Data was collected from several sources in this experiment. The system logged data on user trials, accuracy and preference. The user used the speak-aloud method for qualitative data and gave their rating of each configuration, which was recorded by the researcher along with the researcher's own observations. There was also a comfort ratings form and a post experiment questionnaire.

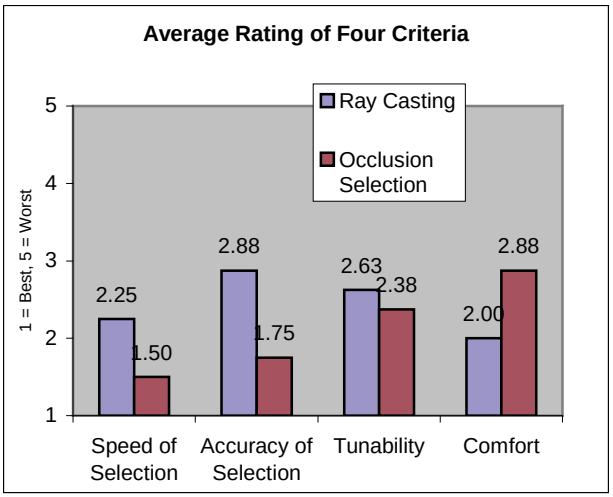

Figure 13 Users rated occlusion superior to ray casting in all criteria except comfort

5.4.4 Results

After this evaluation, we obtained acceptable results for tunable properties of the selection techniques. Users were able to modify the selection techniques easily with an average rating of about two out of five (one being the best) for the usability of the tuning task. They also seemed eager enough to try extra trials with the average number of trials being 15 when the required amount was 8 for occlusion selection and 7 for ray casting. This was encouraging and made us believe that they did search the interaction space well.

Quantitative ratings were taken from user surveys and log files of the user experiments. Figure 13 gives the user's ranking of four criteria for each selection technique. Occlusion selection was preferred to ray casting in all but the comfort category. The configurations for occlusion selection that were ranked highest were generally those where the bull's-eye was on one of the fingertips, as was expected. Configurations 3 and 4 of occlusion selection, which were preferred in Phase 1, were ranked very low as compared to the other configurations, as was expected. This adds support to the notion that users need feedback to determine the value of a configuration. In the ray casting configurations, there was little variance among the different configurations. Also, various snap-to angles made little difference. In general, snap-to angles were set low in occlusion configurations, possibly because there was no need for feedback to try and refine a selection since the proprioceptive sense helps guide the user. In ray casting, snap-to angles were larger though users complained about the ray flickering to other objects. This snap-to could be more useful in sparser environments where the flickering would not be such a problem.

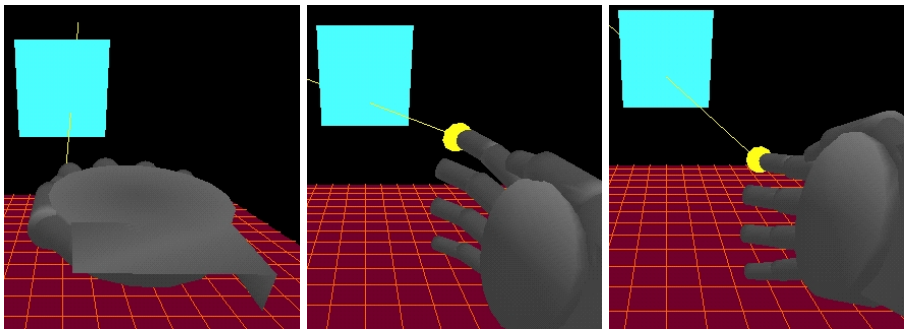

Figure 14 Unusual ray casting configurations created by the users: (left) the ray extends up and to the right (center) the ray extends down and to the left (right) the ray extends up

There were a few different configurations that users preferred (Figure 14). For the most part, there was very little tinkering with factors other than the snap-to angles as most users knew what they wanted in their configuration from the start. Users of occlusion selection had no preference between either their index or middle finger for selection. The major difference between the occlusion techniques was the snap-to angle which was set to (in degrees) 20.60, 16.02, 0.0, 10.0, 4.47, 1.83 and 3.0 twice. Ray-casting had four general configurations with the first being the ray extending straight from the hand with normal to larger snap-to angles. A second personalized ray-casting configuration has the ray extending up and to the right when the hand is parallel with the ground. This lets the user hold the hand in front of themselves in a natural 45-degree angle and make selections with a relatively small snap-to angle of 4.54 degrees. This configuration was not anticipated. Another configuration not anticipated had the ray extending down and to the left when the hand was placed flat. This allowed them to keep the hand in a natural position of close to a 45-degree angle and low, which reduced stress. Users of these two unanticipated tunings also liked large snap-to angles which might be explained by a difficulty to aim the ray when the hand is so far from the head. In the final unanticipated configuration, the hand was held pitched up and rotated outward. This placed the hand close to the face and yet not held too far in front of the user so as to be accurate (close to the eyes) and less fatiguing (not extended). There was a relatively small snap-to angle which makes sense because with the hand close to the eyes, it was already very easy to see where the ray was pointing.

5.4.5 What We Learned

Many users tended to select in the environment rather quickly until they encountered a selection that was difficult to make, due to the object being at a distance, near other objects or the user being sloppy. This difficult selection caused them to slow down and select with more accuracy. Other times this happened were when the feedback was not matching what they were expecting, such as when the bull's-eye was placed off-the palm or when the snap-to angle flickered between objects. This "bad vibes" behavior tended to last for a few selections until the user ramped back up to speed or discovered the source of their confusion.

There were six general parameters that users demonstrated that could be used to explain their actions. The first was *spatial awareness*. Some users had the ability to understand

where they existed in the space and this greatly improved their ability to use ray casting and most likely extends to other proprioceptive tasks. The next parameter was *feedback alignment*. Users responded to feedback with varying degrees of acceptance. Those with high feedback alignment considered the feedback to be infallible such that any indication of correctness, such as a snap-to event occurring, immediately caused them to pinch their fingers. Users with these tendencies preferred smaller snap to angles so they could wave their hand at the object and rely on their reflexes to pinch when they were pointing at the object. A third parameter was *exploration* or their ability to turn lemons into lemonade. Those with high exploration were able to create strategies to adapt to bad configurations quickly. They would inspect the current interface and make changes to how they operate such as orienting the hand differently to reduce occlusion or fatigue. Those with little or no exploration basically never changed the way they selected throughout a configuration no matter the fatigue or difficulty. The forth parameter of a user was *resilience*. A resilient user would not mind fatiguing selection techniques. This can be an advantage because they can use techniques that are accurate or fast but also highly fatiguing. This could be a problem for users if they spend a long time in VEs. The fifth parameter was user *precision*. This affected how much users test the boundary conditions of a selection technique. So, if a technique accepts up to fifteen degrees of error, a non-precise user will expand their sloppiness to fit the fifteen-degree boundary condition. Those with a high precision rating will always provide a certain level of precision even though the environment may require less. This extra effort to give that precision can cost time and fatigue and those with low precision will test the boundary conditions more and keep their precision errors within those bounds. The last parameter and near antithesis of precision is user *speed*. This affects how fast the user moves and how long they must receive positive feedback before they go forward with an action. For example, some users in ray-casting would move their hands quickly and pinch quickly where as others would move slowly, letting the ray dwell on an object for a moment or two, and then pinch. All of these parameters, taken merely from observation, are most likely part of a set of parameters that can be used to explain the user's model of how to perform a selection technique.

The implications are this:

> *There exists an innate set of parameters of the user and the interface that can be used to explain the actions of users.*

In machine learning, a reward function is a reward given based upon the state of the system and the action of the agent. So, a reward function can be created where the state is based on the environment and these parameters and the actions are those the environment allows. Using Inverse Reinforcement Learning from section 4.1.5, we can discover the user reward function so that we can recognize and predict user actions.

5.5 Phase 3: Selection Maps

Under the assumption that we have tuned the selection techniques appropriately, then the next understanding we need is which types of selection techniques are preferred for various locations around the user. In effect, this phase creates a 3D map of selection

preferences between selection techniques just like the old phase 2 was supposed to learn. This selection map incorporates user's tuned settings of the selection techniques from the previous phase implicitly.

Before we consider a map of selection space, one trouble we must overcome is handling the situation of two concurrently running selection techniques selecting two different objects at the same time. In previous environments, there was one technique we were working with at a time but now we want the user to be able to work with multiple techniques. A nuance-based system would be able to handle such a situation because it would have a normalized error value between possible techniques, with the selection technique with the lowest error being the technique to use. In our experiment, ray casting and occlusion selection both have error values in angular units. The measure of how comparable they are will be a topic of future research but for now they seem similar enough to consider each angular unit of error equivalent. Once a decent selection map is created, it could be used to adjust the error values for the techniques. For now, any mistake between which technique the user was using will be noted and corrected by the researcher.

5.5.1.1 Isolating Factors Affecting Selection Technique Preference

To correctly isolate a map of selection, we must identify the nuances that affect the user and minimize those effects. Those identified nuances are the hand's previous location, the selection technique previously used and other minor factors.

The previous location of the hand will affect which technique users prefer. As an example, if a user is pointing at an object across their view and the next object that appears is directly behind their hand, occlusion selection will be highly emphasized. To handle this, we added a delay of one second to the environment. This was enough time for the user to instinctively bring their hand to a more comfortable state and in many cases, to remove it as an occluding object from the scene.

The selection technique previously used will be an indicator of what they might use in the next step. This will be due to the hand already being in a position that is used for a selection technique such as pointing forward for ray casting. Also, the user will be in the mindset of that technique and switching mindsets will be a factor that needs to be considered.

Some of the other factors that affect the choice of selection techniques are object nuances, the existence of other objects in the scene, fatigue and "bad vibes". Object nuances can tell the user how to select the object or at least how to act around it. We used grey cubes so this should not be a factor. Other objects in the scene can interfere with a selection technique by occluding an object, affecting the active feedback, etc. We removed all objects in the scene except for the hands, the cube and the floor with only the cube being selectable so again, there should not be a problem. Fatigue will affect how users work in an environment and only by users conserving energy by using less fatiguing selection techniques will they be able to avoid this. Since the selection of elevated objects is more fatiguing for occlusion selection than ray casting, the map should

reflect this. "Bad vibes" also affects what people do. This is where a user who uses a selection technique incorrectly a few times will shy away from it in future trials.

5.5.1.2 Selection Maps as Predictors

We created selection maps using examples of previous selection under the assumption that past experience will be a good indicator of future actions. To that end, data for the map is a Cartesian point and the type of selection technique used at that point. Our definition of a map will be the space in front of the user, 120 degrees in the yaw and 80 degrees in the pitch. The need to be efficient and locate objects quickly should keep users oriented with the world. Also, users do not interact out of their vision without at least some sort of proprioceptive sense so again, they will want to keep in line with the world.

Once data is collected, it can be used to predict by two general methods; model-based and model free (memory-based or lazy). Model-based representations take data and generalize it into a predictive model of the data. Model-free keeps the data around and use the data itself to predict. In our experiment, we used two model-free predictors and anticipated using one model-based but declined for reasons to be mentioned. The benefits of model-based predictions are a reduced amount of computation since the model is a generalization of the data it represents. The downside is that with any generalization, data is lost. Also, the ability to incorporate additional data varies among models.

The two model-free predictors used were sphere and cone-based. The sphere-based predictor takes the point of the current object, input point, to be selected and draws a sphere of a given radius around the point identifying all the previously sampled points contained. The cone-based predictor draws a ray from the user's eye through the input point and creates a cone around that ray using a certain angle again identifying all the sampled points contained. These identified sample points are then split among which selection technique was used when they were created and the selection technique with the most data points is the predicted technique of a certain probability (Table 3). The radius and angle used is dependent upon the number of points you desire to be used and the density of your sampled points that you have collected. In our experiment, we used a radius of four meters and an angular error of twenty degrees which we determined by trial and error.

For each $t \in$ T, where T is the set of all selection techniques used, the probability of t being predicted is $\frac{SP_t}{SP_{total}}$, where SP is the set of all sampled points for the given model.

Table 3 Determining the probability of a selection technique being performed based upon the sampled points

Alternatively, model-based techniques induce a high-level representation that can be used as predictors for future interaction scenarios. A promising class of such techniques involves finding optimal and "admissible" regions [Fukuda96]. These algorithms exploit spatial continuity to postulate areas and regions of the 3D space that have high confidences for one selection technique to be preferred over all others. A union of such areas for all techniques constitutes what we can call a "selection map."

5.5.1.3 Environments Used

There were two environments used for this phase (Figure 15). The first environment is similar to Phase 1's environment. The user stood above a gridded floor and saw one cube floating in space before them. The head and gloves were both tracked with models of hands attached to the gloves. TULIP was used to let the user choose when to start a set of trials, in this case a demo set trial of four selections or a full set of 100 selections. The second environment is much like Phase 2's environment. There was an array of 27 objects laid-out 3x3x3 in front of the user. Again, the head and gloves were tracked and again models of hands were attached to the gloves. In both environments, both selection techniques could have been used at any time with active and passive feedback operating. In order to give the users a break and keep the pressure to perform up, there was a pause every 10 selections where the environment told the user how many selections are left in the current trial and their qualitative ranking so far.

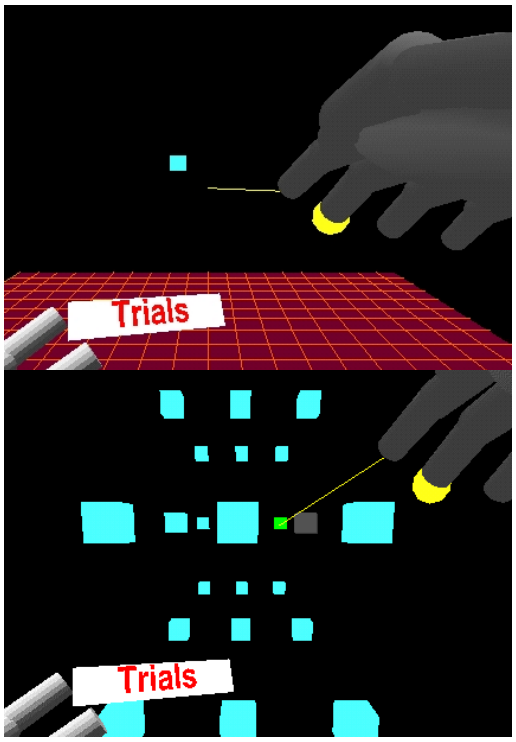

Figure 15 Phase 3 environments 1 (top) and 2 (bottom)

5.5.1.4 Experimental Design

There were three experiments in this phase. The first two experiments used environment 1 and the last used environment 2. Experiment 1 was designed to collect data points and experiment 2 was designed to test how predictive the environment could be of the users actions in the same environment using the sampled data. Experiment 3 used the sampled data and predicted user action inside environment 2 to test how applicable the map was to an alternate environment. To collect the points in environment 1 and 2, the grey cube would disappear after a selection and reappear somewhere else in the map region. In experiment 3, one object was colored dark blue and that coloration would move randomly from object to object after each selection. The selection techniques were tuned to the same settings as the user preferred in Phase 2. Additionally, users were asked to perform a few selections of both techniques to help them regain familiarity before the experiment started and they could use either technique for any selection.

5.5.1.5 Data Collected

Experiment 1 sampled 100 Cartesian coordinates of the cube when selected and also the selection technique used at that point. Additionally, the researcher recorded qualitative information and any comments the users made during the experiment. Since the users were trying to behave optimally, they were not explicitly told to think-aloud. Experiment 2 sampled 50 Cartesian coordinates of the cube when selected and additionally the results

of the two prediction techniques with their corresponding supporting data. Experiment 3 recorded the same data as experiment 2.

5.5.1.6 Results

Predicting which technique users would use, using data in the same environment it was generated in, was nearly always correct. The reason for this accuracy was that users do not like to change between selection techniques. Users tended to pick their favorite selection technique and use it throughout the trial. When the environment switched to having multiple objects (experiment 3), the users still tended to use their favorite selection technique but when selections became difficult, they had a tendency to switch. Usually, they would switch back to their favorite selection technique immediately but some users would continue for a trial or two, then switch back to their favorite selection technique. This seemed to suggest a high, user dependent, cognitive cost of switching techniques.

The questionnaire returned a few interesting results, the first being that in the first two experiments, the selection map was not a major factor in the prediction of the selection technique. There was a difference in the results but it was so small as not to be significant. In experiment 3 however, a selection map based upon user preference was considered by the users to be slightly a factor where ray casting was liked at a distance and to the left and occlusion selection in the other locations. Other results showed that users felt switching between selection techniques in the second environment was less difficult than in the first even though they claimed the utility to be about the same. In all environments, the users liked their configuration for the selection techniques, so the results of Phase 2 transferred.

5.5.1.7 What We Have Learned

When comparing selection techniques, there needs to be some sort of distracter task to remove the users from their current mindset. Without this, selection maps are overshadowed by the cognitive costs of switching so the extent to which selection maps are useful is still a topic of research though it appears to have impact.

Our experience can be summarized in this way:

> *When given a choice between interaction techniques, users have a perceived utility of each and an associated penalty for switching from their current technique to another; in a sense exploration versus exploitation.*

Additionally, incorrect selections, frustration or the “bad vibes” concept reduces the perceived utility of a selection technique.

5.6 Summary

Though not proving what we originally intended, our results are quite encouraging. These experiments have helped outline how users work in environments and how we should build a nuance-oriented system for them. Phase 1 showed that users have a model of feedback and not of the environment, the implication being that a nuance-oriented

framework should be based upon feedback and not the model of the user. Phase 2 confirmed this and led us to believe that users have a reward function based upon themselves and the VE. We also hypothesized about some of those parameters which fits nicely into the nuance parameter framework. Phase 3 showed that users do not like to switch between techniques unless they sense they are failing, which can be modeled with environmental nuances. This reluctance to switching again supports the fact that users are machine learners themselves. All of these results can be modeled by the system.

Chapter 6 Methodological Aspects

Given the discoveries of chapter 5, we have a better understanding of the problem that we are trying to solve. The two main problems we are dealing with are the recognition of user actions, split into three smaller problems, and the deduction of user goals and sub-goals from the user actions. A section on on-line versus off-line learning is also presented in this chapter.

6.1 How a Nuance-Oriented VE Might Optimize Recognition

If all of these nuances are changing the ways in which the user will interact in an environment, then we must focus on how the interface will recognize the user actions. The first question is then, "How do we transfer the results of one environment to another?" Also, "How can we allow what we have learned in one environment to be transferred into another?" Finally, "How do we incorporate new concepts into our learning and still take advantage of the old example data?" We will refer to these questions as the Decomposition Problem, the Representation Problem and the Updating Problem.

6.1.1 The Decomposition Problem

The Decomposition problem is, "How do we get the results of one environment to transfer to another?" Stated another way, "Is the interaction space decomposable?" If so, then we can look at parts of the space and then join the learned representations into a larger solution for the entire interaction space. Our problem is most likely not decomposable because every new property, as we have stated earlier, affects every other property such that a learned representation with something new added, has to be learned again. This is because the effects of combined nuances are not necessarily a combination of the two. For example, if we are gathering data in one environment which has stone columns, carrying the cylindrical object nuance, and another environment with a pair of scissors, carrying the grab-at-top object nuance, we can learn a representation. If we are working in a third environment which has a golf tee with both the cylindrical and grab-at-top object nuance then the manner in which we grab the golf tee is not a mix. There are five possibilities; 1. the user will behave like the cylindrical nuance 2. the user will behave like the grab-at-the-top nuance 3. the user will act as if both nuances are in effect to some degree 4. some new form of grabbing will be performed by the user 5. we will have to split either or both of the nuances into smaller nuances because they are not atomic.

6.1.2 The Representation Problem

The Representation problem can be solved through a variety of representations; we present two. The first is an isometric hyper-box representation. This has the advantage of being easy to understand by human experts but rules are hard to learn and do not work well with the amount of noise we will have. A continuous function representation is easy to learn, even with noisy data, but does not produce a representation that is understandable by an expert. However, there has been some work done to extract rules with some success.

6.1.3 The Update Problem

The Update problem is, "How do we incorporate new concepts into our learning and still take advantage of the old example data?" This problem would be easily solved if we knew of a higher representation for the data. If we did, the representation could be stored and the data thrown away. This again relates to the decomposition problem and since we do not know if the problem space is decomposable, we do not know if this representation is possible (it is our intuition that it is not). The update problem is further compounded by the expense at obtaining data. If data were abundant, we would not need to worry about the update problem because as new atomic nuances were learned, we would just generate new training data and relearn the representation. One way to solve the Update problem is by storing all the supporting data from user trials and creating methods to deal with new atomic nuances as they are discovered.

6.1.3.1 What data do we record?

The simplest notion is to store data as an episode with that episode being a segment of the stream of data that was occurring in the VE. Unfortunately, the start and stop of each episode is recognized by the interface and that recognition changes so the start and stop of each episode may change. To solve this problem, we have to record all interaction data made by users. Additionally, since the interface is not perfect in its recognition, some episodes may be missed at first but recognized when the interface changes due to further data. For instance, if we are creating an interface that is triggered by the user nodding their head several times, as the recognition of the nodding nuance becomes better, it may require fewer nods to recognize. This is problematic since in the original interaction, the user, realizing the nuance is failing, diverged from their optimal path of interaction to adapt to the interface. In this case, the user performs more nods than is needed and this will cause the interface to expect the users to do more nods than required, propagating the incorrectness, not because of a lack of recognition, but because it expects more nods. Because of this, many episodes will have to be removed so as not to mislead the learning system.

6.1.3.2 Dealing with New Atomic Nuances

All nuances are created because of supporting data and that data is the result of all the atomic nuances in the VE at that time. As new environments are created, new atoms will be uncovered and older atoms, thought to be atomic, will split into multiple now thought-to-be-atomic, nuances. So, not all the atoms will come into contact with each other in every environment and all atoms must be able to change because of new atoms being discovered. Therefore, in addition to recording interaction data, the nuance-oriented

interface must store all atoms in the environment when the user data is recorded. Additionally, new atoms, as they are discovered need to be added and existing atoms replaced by multiple, more “atomic”, atoms.

6.2 How a Nuance-Oriented VE Might Optimize User Work

Most of the discussion so far has been about the recognition of interaction techniques. The other task that nuances can help perform is the optimization of how users work. So, how does one go about creating a VE system that uses nuance-oriented interaction? At first glance, the problem of designing a nuance oriented VE looks suspiciously similar to programming-by-demonstration, with just a more complicated (and richer) demonstration sequence of interactions. This model of learning typically involves recording user scenarios, replaying them, and (in a limited way) generalizing the scenarios. A nuance is more than an enumerated (or captured) list of scenarios. A nuance implies an internal model that a user brings to the interaction task and employs (in the manner of a decision procedure) actively when interacting with the VE. In other words, a nuance is best modeled as a decision procedure, itself, imitating and mimicking the user's decision procedure.

This problem is formally referred to in machine learning as “inverse reinforcement learning (IRL)” [Ng00]. The assumption in IRL is that an agent's behavior (which can be observed) is the result of a deliberative process of choosing and weighting actions. If the agent (a VE user) can be assumed to be behaving “optimally” (based on his or her own notion of what this means), then the IRL problem can be formulated as one of 1) uncovering the user's “reward function,” 2) finding a policy (a list of the nuances in the system) that works as well as the user's nuance, or 3) both. For example, perhaps a user employs a nuance to minimize hand fatigue but is otherwise unconcerned with the strain on his eye. The user's notion of optimality then would correspond to a weighted linear combination of these response variables with hand fatigue having a higher additive contribution than eyestrain. Using IRL, we can uncover this nuance and attempt to model the decision procedure that optimizes the user's reward function.

In many instances, the user might not be explicitly aware of their nuances or why/when they employ them. The observation-based approach to IRL will work even in these cases, as long as it is reasonable to assume that the user is systematic in their choices and chooses actions and interaction techniques in a consistent manner. Notice that the learned nuance may or may not be the same as the user's decision procedure but can serve as a meaningful model of the original nuance.

Once this problem is formulated, several challenges remain. The richness of the nuances employed (and the associated decision procedures) directly impact the choice of model. The model must be able to capture the full complexity of the interaction metaphor, and at the same time be computationally cheap to learn, update, and maintain. A good first step would be to qualify a design vocabulary of nuances and their various forms. A careful analysis will provide insights into model selection.

Once a preliminary representation of a nuance is available (either seeded directly, or by some basic data mining), integrating it into the control flow of a VE application is important for IRL. The system might be beset with judgment calls involving both false positives and false negatives. False positives are more serious (and irritating) than false negatives and this could be factored into the IRL training algorithm. When IRL is employed with sampled data (as it most likely will be in a VE), heuristics will become important to "shape" the nuances [Ng00][Kaelbling96], and steer them away from suboptimal solutions.

6.3 On-Line vs Off-Line Learning

Off-line training has been assumed up to this point as it allows an interface designer to manage data and validate nuances before they are thrust upon the user. This could help weed-out the misjudgments by the learning system. Unfortunately, if a user is using a system and it does not respond to their nuance, the user will likely stop exhibiting that nuance. Two possible types of systems are "train by example" systems and systems that mine user logs. The example learning systems have the problem of letting the user act naturally during the training periods such that they will repeat a nuance enough to be recognized. Mining systems can bring nuances to the attention of the UI designer and be less selective because the UI designer will ultimately decide if there is value in a nuance.

On-line systems do not have to worry about users not repeating a nuance due to lack of recognition, because those systems can recognize the nuance and immediately change the VE. Such a system may confuse a user because of incorrectly created nuances. Also, it has the major disadvantage of being computationally expensive on a system that is taxed by the VE it is already running. This could be implemented as a local or remote agent, mining user data concurrently with program execution. This type of system would be more difficult to implement than off-line learning. Also, given the task to be performed, it may require true artificial intelligence because of the requirements of the learning which is a rather large requirement. This type of a system could be seen as the Formative Nuance-Oriented User-Centered Evaluation Cycle without the experts; a step beyond even what we are considering.

6.4 Summary

The implications of dealing with these problems lead toward obvious paths of future work. The split between recognizing and deducing interfaces was not completely obvious until after these experiments in chapter 5. Also, the discussion of when learning takes place can be seen as the next step, after nuance-oriented interfaces, in interface design.

Chapter 7 The Wrap-Up

This work herein is the start of a larger research agenda to bring complexity handling strategies to bear on VE interface design. At each step, we encountered results that were not expected but furthered our knowledge of the user and how to better recognize and optimize their actions. Here we present four conclusions, five contributions and eight paths of future work towards the goal of using ML to handle Nuance-Oriented VE interface design. These contributions and the experiments that revealed them tended to create more avenues for future work than they did in answering open questions in my mind.

7.1 Conclusions

The following four conclusions have helped strengthen our belief that nuance-oriented interfaces are possible and have provided an understanding of what that system should look like:

1) Users do not have an innate model of the environment but of the feedback.
Phase 1 in section 5.3 showed that users do not have a model of the environment but of the feedback and affordances of the environment. Because of this, we have to pay close attention to what our environment is telling the user when we design interfaces. Also, the nuances are going to be based upon the feedback and affordances, and not so much an innate model of the environment.

2) Properties can be used to explain user behavior.
In section 5.4, six properties were identified that can help in explaining user behavior. Those properties were spatial awareness, feedback alignment, exploration, resilience, precision and speed. These are just the hints at the properties that will be used to frame user behavior. With these, in a nuance-oriented interface, we can assist with the recognition of their actions and the deduction of their intentions.

3) Users have an internal reward function.
This was concluded in section 5.4 when identified parameters that could explain user behavior. Since those parameters exist, then we assume users have a reward function based on those parameters which can be used to explain their actions. Given this, we can treat users as reinforcement learners and apply learning methods to deduce their goals.

4) Users have a perceived utility of approaches to an interaction task and an associated penalty for switching.
This was explained in section 5.5, after phase 3. This again lends support to users having a reward function and that one parameter to that system would be a penalty for switching techniques.

7.2 Contributions

In addition to these conclusions that we have come to, we have the following contributions:

1) An explanation as to why virtual environment interfaces are complex.
An argument was given in chapter 2 for why VE interfaces are complex, based upon a review of their input data, input devices and metaphors. Additionally, an argument was given based upon the bounding requirements of interfaces with larger DOF.

2) Machine learning can find important nuances in human behavior to exploit.
The neural network experiment in section 4.2 showed that interfaces could be made that could take advantage of user nuances by discovering both the trivial and non-trivial nuance in the selection interface. This is the basis of our current work and future work will be based upon this belief.

3) Four categories of nuances are given.
We gave four categories of nuances that can be used to classify the types of nuances that users give, consciously or subconsciously. The categories in section 3.2 are; object, environment, refinable, and supplemental. Future work will be on identifying, classifying, supporting and propagating nuances.

4) Guidelines for nuances are given.
We provided nine guidelines and an acronym to remember them by in section 3.3. The acronym is A FUR STAGE and the guidelines are Accurate, Fast, Understandable, Refining, Sensitive, Trainable, Attribute-based, General and Extensible. These will help in future work when we build interfaces and must have guidelines about the handling of each nuance.

5) A methodology of building a nuance-oriented interface is given.
The Formative Nuance-Oriented User-Centered Evaluation Cycle is explained to deal with incorporating the machine learning into a design process. This is explained in section 3.4. The additions to an existing process for interface development will hopefully make working with nuances an easier transition for interface designers.

7.3 Future Work

During the course of this research, these eight concepts were noticed to be of interest and relevant but beyond the scope of this work:

1) How to take advantage of user properties?
We showed that properties of users existed in section 5.4 but did not investigate methods to use these properties. This could lead to personality profiles or categories of users or both. This would allow similar users to share experience in differing environments. The most likely implementation would be to incorporate these parameters as nuance properties.

2) Can we create a simple nuance-oriented interface?
The nuance concepts and nuance parameters need to be identified that affect interaction tasks in an example interface. This recovers solving the representation problem and performing tests in an environment as constrained as possible. The coverage of the identified concepts and nuances will be hard to determine and will most likely require large scale user studies.

3) How to identify and prove new user properties?
We need to see how many properties it takes to describe users. This would entail studies of large numbers of users which we have not done here. Hopefully, these studies would also generate valuable example data for training the interface representations.

4) Can nuance libraries be built?
We need to formalize nuances in their representation and build a system of nuance libraries based on that formalization. This library also needs to store the supporting data for each nuance as was discussed section 6.1. This would facilitate rapid interface development if we could also strengthen our vocabulary and taxonomy.

5) Can known nuances be added automatically?
Once nuances have been identified, can we develop interfaces that take existing training data, or episodes, and learn which nuances are applicable to the existing interface? This would allow experts to verify the addition of nuances to a nuance library and interface designers to verify the addition of nuances to a particular interface from the nuance library.

6) Are our categories and guidelines complete?
The categories of nuances are hypothesized and cover, according to our intuition, the nuances that might exist. Nuances need to be identified for each category and the coverage of our categories needs to be tested. This can only happen after a system to discover nuances has been built.

7) How can we guide the user to better interaction?
It has been shown that users can be manipulated. In a pilot study, we noted that the user only learned a strategy when they were forced to do so because of the discomfort of the interface. Other users were guided as to how to hold their hand based upon the model of the hand. If these simple types of feedback can modify user behavior then there should be ways to use this to create better interfaces. Also, only the parts of the

interface that users utilize will help them work better, so if we can guide, how do we guide them to use the entire interface? This will become more important as more complex interfaces are built.

8) Can we reduce the computation required in the interface?
The nuance interfaces will require a large amount of processing to compute and methods to reduce this should be looked into. One method could reduce computation by having multiple states and each state only have a subset of the entire interface. This should be possible since many interactions flow into another and are not all active at the same time. Another would be to only process the most probable parts of an interface at any given point; the amount based upon the computational resources available.

7.4 Concluding Remarks

This thesis has laid the groundwork for basically more work. The experiments performed have helped to understand the user and shape how a nuance system should be built. The future will require experiments to start investigating the system to support the nuances and concepts mentioned in this work. Since designing interfaces is more of an art than a science, and the same can be said of the methodology of applying machine learning, I fear that creating interfaces with machine learning will be much like trying to paint the Mona Lisa on the body of Michelangelo's David and expecting both to gain praise. Future work will be the ultimate judge but given the limits of human understanding and the complexity of human expression and adaptation, I can only imagine that nuance-based systems will be the salvation of the man-machine interface.

References

[Arsenalt01] Arsenault, Lance Kelso, John Kriz, Ronald. "DIVERSE" [Online] http://www.diverse.vt.edu.

[Baecker87] Baeker, R. Buxton, W.A.S. (1987) Readings in Human-Computer Interaction – A Multidisciplinary Approach. Morgan Kaufman Readings Series. Mrogan Kaufmann, Los Altos, CA.

[Blum88] Blum, Avrim, Rivest, Ronald L. (1988) Training a 3-node neural network is NP-complete. P*roceedings of the Annual Workshop on Computational Learning Theory*, 1988.

[Bolt80] Bolt, Richard A. (1980) "Put-that-there: voice and gesture at the graphics interface". *Computer Graphics*, August, vol 14, no 3, pg 262-270.

[Bishop95] Bishop, Christopher M. (1995) Neural Networks for Pattern Recognition. Oxford Press, 1995.

[Brown90] Brown, D. Totterdell, P. Norman, M. (1990) Adaptive User Interfaces. London: Academic Press.

[Bowman97] Bowman, Doug A. Hodges, Larry (1997) "An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments" *Proceedings of the 1997 symposium on Interactive 3D graphics*, pg 35.

[Bowman99] Bowman, D., Davis, E., Badre, A., and Hodges, L. (1999) "Maintaining Spatial Orientation during Travel in an Immersive Virtual Environment." *Presence: Teleoperators and Virtual Environments*, vol. 8, no.6, pp. 618-631.

[BowmanDis] Bowman, Doug A. (1999) "Interaction Techniques for Common Tasks in Immersive Virtual Environments: Design, Evaluation, and Application" Georgia Tech Dissertation.

[Bowman01] Bowman, Doug A. Wingrave, Chadwick (2001) "Design and Evaluation of Menu Systems for Immersive Virtual Environments", *Proceedings of IEEE Virtual Reality*, pg 149-156.

[BowmanHCI01] Bowman, Doug, Wingrave, Chadwick, Campbell, Joshua, Ly, Vinh. (2001) "Using PinchGloves for both Natural and Abstract Interaction Techniques in Virtual Environments." *HCI International*.

[Brooks99] Brooks, R. (1999) "What's Real about Virtual Reality", *IEEE VR*, pg 2-3.

[Cruz-Niera93] Cruz-Neira, Carolina Sandin, Daniel DeFanti, Thomas. (1993) "Surround-screen projection-based virtual reality: the design and implementation of the CAVE." *Proceedings of the 20th annual conference on Computer graphics*, pg 135 – 142.

[Cypher93] Cypher, A. (Ed.) (1993) Watch What I Do: Programming by Demonstration. Cambridge, MA: MIT Press.

[Eisenstein00] Eisenstein, J. Puerta, A. (2000) "Adaption in Automated User Interface Design." *Intelligent User Interfaces*, pg 74-81.

[Forsberg96] Forsberg, Andrew Herndon, Kenneth Zeleznik, Robert. (1996) "Aperture Based Selection for Immersive Virtual Environments", *Proceedings of the ACM symposium on User interface software and technology*, pg 95 – 96.

[Fukuda96] Fukada, T. Morimoto, Y. Morishita, S. Tokuyama, T. (1996) "Data Mining Using Two-Dimensional Optimized Association Rules: Schema, Algorithms and Visualization", *Proceedings of the ACM SIGMOD Conference on Management of Data"*, pg 13-23.

[Garris98] Garris, M. D. (1998) "Intelligent System for Reading Handwriting on Forms", *International Conference on System Science*, vol 3, pg 233-242.

[Gabbard99] Gabbard, Joseph L. Hix, Deborah Swan II, J. Edward. (1999) "User-Centered Design and Evaluation of Virtual Environments." *IEEE Computer Graphics and Applications*, November/December, pg 51-59.

[Heckerman96] Heckerman, David. (1996) "A Tutorial on Learning With Bayesian Networks." Technical Report MSR-TR-95-06, Microsoft Research, Redmond, Washington, 1995. Revised June 96.

[Hix99] Hix, D. Swan II, J.E. Gabbard, J.L. McGee, M Durbin, J. King, T. (1999) *"User-Centered Design and Evaluation of a Real-Time Battlefield Visualization Virtual Environment"*, IEEE VR, pg 96 –103.

[Jacob99] Jacob, R. J. K. Deligiannidis, L. Morrison, S. (1999) "A Software Model and Specification Language for Non-WIMP User Interfaces." *ACM Transactions on Computer-Human Interaction*, vol 6, no 1, pg 1-46.

[Kaelbling96] Kaelbling, L. Littman, M. Moore, A. (1996) "Reinforcement Learning: A Survey." *Journal of Artificial Intelligence Research*, vol 4, pg 237-285.

[Kessler95] Kessler, G. Drew Hodges, Larry F. Walker, Neff. (1995) "Evaluation of the CyberGlove as a whole-hand input device", *ACM Transactions on Computer-Human Interaction*, vol 2, issue 4, pg 263-283.

[Kramer89] Kramer, J. Leifer, L. (1989) "The Talking Glove: An Expressive and Receptive 'Verbal' Communication Aid for the Deaf, Deaf-Blind and Nonvocal." Tech Report, Stanford University, Department of Electrical Engineering, Stanford, CA.

[Leigh01] Leigh, Jason. (2001) [Online] http://www.evl.uic.edu/cavern/cavernsoft/limbo.html.

[Manber00] Manber, Udi Patel, Ash Robinson, John. (2000) "Experience with personalization of Yahoo!" *Communications of the ACM*, Vol 43, Issue 8, pg 35-39.

[Mine95] Mine, M. "Virtual Environment Interaction Techniques", Technical Report TR95-018: UNC Chapel Hill CS Department.

[Mine97] Mine, M. Brooks, F. Sequin, C. "Moving Objects in Space: Exploiting Proprioception in Virtual Environment Interaction", *Proceedings of SIGGRAPH*, pg 19-26.

[Mitchell80] Mitchell, T. M. (1980) "The need for biases in learning generalizations." *Technical Report CBM-TR-117*, Department of Computer Science, Rutgers University.

[Mitchell97] Mitchell, Tom (1997) Machine Learning. McGraw Hill.

[Muggleton92] S. Muggleton, ed, (1992) Inductive Logic Programming, (Academic Press, San Diego, CA, 1992).

[Ng00] Ng, A. Y. Russell, S. (2000) "Algorithms for inverse reinforcement learning." *International Conference on Machine Learning*, Stanford, CA: Morgan Kaufmann.

[Nigay95] Nigay, Laurence Coutaz, Joëlle *(1995)* "A Generic Platform for Addressing the Multimodal Challenge". *Conference proceedings on Human factors in computing systems*, pg 98-105.

[Norman90] Norman, D. The Design of Everyday Things. 1990.

[Oviatt99] Oviatt, S. Mutual Disambiguation of Recognition Errors in a Multimodal Architecture. Proceeding of the CHI 99 conference on Human factors in computing systems: the CHI is the limit, 1999, Pages 576 – 583.

[Pierce97] Pierce, Jeffrey S. Forsberg, Andrew S. Conway, Matthew J. Hong, Seung Zeleznik, Robert C. Mine, Mark R. (1997) "Image plane interaction techniques in 3D immersive environments" *Proceedings of the 1997 symposium on Interactive 3D graphics*, pg 39.

[Poupyrev96] Poupyrev, I. Billinghurst, M. Weghorst, S. Ichikawa, T. (1996) "The Go-Go Interaction Technique: Non-Linear Mapping for Fdirect Manipulation in VR", *Proceedings of the ACM Symposium on User Interface Software and Technology*, pg 78-80.

[Poupyrev97] Poupyrev, I. Weghorst, S. Billinghurst, M. Ichikawa, T. (1997) "A Framework and Testbed for Studying Manipulation Techniques for Immersive VR", *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*", pg 21-28.

[Resnick97] Resnick, Paul Varian, Hal R. (1997) "Recommender Systems", *Communications of the ACM*, Volume 40 , Issue 3, pg 56-58.

[Rubine92] Rubine, D. Beale, R. (Ed.) (1992) "Criteria for Gesture Recognition Technologies", Neural Networks and Pattern Recognition in Human-Computer Interaction, West Sussex, England: Ellis Horwood Limited.

[Russell94] Russell, Stuart, Norvig, Peter, (1994) Artificial Intelligence: A Modern Approach, Prentice Hall, 1994.

[Ruvini00] Ruvini, J. D. Dony, C. (2000) "APE: Learning User's Habits to Automate Repetitive Tasks", *Intelligent User Interfaces*, pg 229-232.

[Sheridan94] Sheridan, T. B. (1994) "Further musings on the psychology of presence", *Systems, Man and Cybernetics, Humans, Information and Technology, IEEE International Conference on*, Vol. 2, pg 1073 –1077.

[Slater95] Slater, M. Usoh, M. Steed, A. (1995) "Taking Steps: The Influence of a Walking Technique on Presence in Virtual Reality." *ACM Transactions on Computer-Human Interaction*, vol 2, no 3, pg 201-219.

[Stephenson01] Stephenson, Neal. (2001) "In the Beginning was the Command Line" [Online] http://www-classic.be.com/users/cryptonomicon/beginning_print.html.

[Stoakley95] Stoakley, Richard Conway, Matthew J. Pausch, Randy. (1995) "Virtual reality on a WIM: interactive worlds in miniature", *Conference proceedings on Human factors in computing systems*, Pages 265 – 272

[Tsukahara01] Tsukahara, Wataru Ward, Nigel (2001) “Responding to Subtle, Fleeting Changes in the User’s Internal State.” *CHI 2001*, vol 3, issue 1, pg 77-84.

[Werkhoven98] Werkhoven, P. J. Groen, J. (1998) “Manipulation performance in interactive virtual environments.” *Human Factors*, vol 40, no 3, pg 432-442.

[Wingrave01] Wingrave, Chadwick A. Bowman, Doug A. Ramakrishnan, Naren. (2001) “A First Step Towards Nuance-Oriented Interfaces for Virtual Environments.” *Proceedings of the Virtual Reality International Conference*, pp. 181-188.

[WingraveJive] Wingrave, Chadwick A. (2001) “JIVE: Just In a Virtual Environment.” [Online] http://csgrad.cs.vt.edu/~cwingrav/Jive/home.html.

Appendix A Forms

Appendix B Results

Phase 1: Comfort Ratings Results

User	1	2	3	4	5	6	7	8
Experiment Order[2]	aro	ora	roa	oar	aor	rao	aor	roa
Gender	F	M	M	M	M	F	F	M
Hand	R	R	R	R	L	R	R	R
Age	25	28	26	54	25	24	24	24
Eyes	Contacts	Glasses	Glasses	Contacts	Glasses	N/A	N/A	N/A
Major	Architecture	Biomedical Engineering	Physics	Industrial Systems Engineering	Architecture	ISE-Human Factors	Computer Science	Computer Science
Computer Familiarity[3]	4	4	3	5	5	5	5	4
Computer Usage Frequency[3]	5	5	5	5	5	5	5	5
Computer Game Frequency[3]	5	5	4	1	5	3	4	5
Comfort Ratings Form								
1st Experiment								
Arm strain[4]	1	1	1	1	1	1	1	1
Hand strain[4]	1	1	1	1	1	1	1	1
Dizziness[4]	1	1	1	1	1	1	1	1
Nausea[4]	1	1	1	1	1	5	1	1
2nd Experiment								
Arm strain[4]	1	1	1	1	3	1	1	1
Hand strain[4]	1	1	1	1	1	1	1	1
Dizziness[4]	1	1	1	1	1	1	2	1
Nausea[4]	1	1	1	1	1	7	1	1
3rd Experiment					2			
Arm strain[4]	2	1	1	1	4	1	2	2
Hand strain[4]	1	1	1	1	2	1	1	1
Dizziness[4]	1	1	1	1	1	3	3	1
Nausea[4]	1	1	1	1	1	7	1	3
4th Experiment								1
Arm strain[4]	1	1	1	1	5	1	2	2
Hand strain[4]	1	1	1	1	2	1	1	1
Dizziness[4]	1	1	2	1	1	3	3	1
Nausea[4]	1	1	1	1	1	3	1	1

[2] A is arm extension, O is Occlusion Selection, R is ray casting
[3] On a scale from 1 to 5 where 1 is low and 5 is high
[4] On a scale from 1 to 10 where 1 is low and 10 is high

Phase 1: Position of Objects

Near, Mid, Far: 1.0, 3.0 10.0
Low, Level, High: -5.0, 0.0, 5.0
Left, Center, Right: -3.0, 0.0, 3.0
Close Left, Center, Close Right: -0.5, 0.0, 0.5
Close Low, Level, Close Right: -0.5, 0.0, 0.5

Phase 2: User Results

User	1	2	3	4	5	6	7	8
First Technique	Occlusion	Ray Cast	Ray Cast	Occlusion	Ray Cast	Ray Cast	Ray Cast	Occlusion
Occlusion Configurations[1]								
1	1	3	2	2	1	2	2	2
2	1	1	2	2	1	2	1	1
3	3	5	5	5	5	5	5	5
4	4	5	5	3	3	5	5	4
5	4	3	2	1	2	3	3	4
6	2	2	2	4	2	4	3	3
7	1	3	1	2	1	2	3	2
Average:	2.285714	3.142857	2.714286	2.714286	2.142857	3.285714	3.142857	3
Ray Casting Configurations[1]								
1	1	4	4	3	2	4	3	2
2	2	2	2	3	3	2	4	1
3	3	4	1	3	1	4	2	4
4	2	1	1	5	1	3	1	3
5	1	1	2	5	3	2	3	2
6	1	3	1	4	1	2	4	5
Average:	1.666667	2.5	1.833333	3.833333	1.833333	2.833333	2.833333	2.833333

1: On a scale of 1 to 5 with 1 meaning liked and 5 meaning disliked.

Phase 2: Post Questionnaire

User Satisfaction[1]								
Ray Casting	2	3	2	3	1	2	3	2
Occlusion	2	1	2	3	2	5	2	3
TULIP	*	1	3	3	1	1	5	1
3-D Sliders	*	3	1	3	1	*	4	2
Comparison of Ray Casting and Occlusion[2]								
Close	1	4	1	2	5	2	5	2

Distant	2	3	5	4	1	1	1	2
High	1	4	3	4	1	4	1	3
Low	1	3	1	4	5	3	4	3
Score Applicability[3]								
Ray Casting	2	4	2	3	2	2	1	4
Occlusion	2	1	1	3	1	2	1	4
Determination to Score Well	1	2	2	3	1	2	2	1
Satisfaction with…[1]								
Ray Casting								
Speed of Selection	2	4	4	3	1	1	2	1
Accuracy of Selection	3	4	4	3	1	3	2	3
Tunability	2	5	2	3	1	1	3	4
Comfort	1	4	1	3	1	1	3	2
Occlusion Selection								
Speed of Selection	1	2	1	2	1	2	2	1
Accuracy of Selection	2	1	1	2	2	1	2	3
Tunability	2	1	2	4	1	4	2	3
Comfort	2	1	2	3	4	5	3	3
Best	Ray	Occlusion	Occlusion	Occlusion	Ray	Ray	Occlusion	Ray
User Comfort[4]								
Beginning								
Arm Strain	1	1	1	1	1	1	1	1
Hand Strain	1	1	1	1	1	1	1	1
Dizziness	1	1	1	1	1	1	1	1
Nausea	1	1	1	1	1	1	1	1
1st Experiment	Occlusion	Ray	Ray	Occlusion	Ray	Ray	Ray	Occlusion
Arm strain	2	1	1	1	2	1	3	4
Hand strain	1	1	1	1	2	1	2	4
Dizziness	1	1	1	1	1	9	2	3
Nausea	1	1	1	1	1	10	1	3
2nd Experiment	Ray	Occlusion	Occlusion	Ray	Occlusion	Occlusion	Occlusion	Ray
Arm strain	1	1	2	1	5	9	5	1
Hand strain	1	1	1	1	5	1	3	1
Dizziness	1	1	1	1	1	1	3	3
Nausea	1	1	1	1	1	1	*	5

1: On a scale of 1 to 5 with 1 meaning satisfied and 5 meaning unsatisfied.
2: On a scale of 1 to 5 with 1 being ray casting and 5 being occlusion selection.
3: On a scale of 1 to 5 with 1 being agree and 5 being disagree.
4: On a scale of 1 to 10 with 1 being comfortable and 10 being extreme discomfort.

Phase 2: User Performance

User	Configuration	1	2	3	4	5	6	7	8	Configuration Average

Occlusion	1	7	6	3	2	3	5	2	4	3.5
	2	2	2	2	5	7	2	1	3	2.875
	3	3	5	5	6	7	7	4	7	4.4375
	4	4	6	6	2	2	5	3	7	3.734375
	5	7	2	2	2	1	5	6	7	3.5
	6	3	3	2	3	3	7	2	7	3.34375
	7	2	4	2	2	0	3	6	2	2.640625
	User # 1	4	2	2	3	3	3	4	2	2.796875
	User # 2	2	1	3	2	2	3	2	3	2.40625
Ray Casting	1	2	6	7	2	4	7	7	5	4.125
	2	4	3	6	2	5	3	5	1	3.265625
	3	6	3	3	3	3	7	4	4	3.578125
	4	4	1	2	4	4	6	2	2	2.953125
	5	4	2	4	2	4	6	3	2	3.109375
	6	2	3	2	1	3	4	5	2	2.71875
	User # 1	3	2	5	4	6	5	2	3	3.34375
	User # 2	2	2	3	2	6	4	3	2	2.875

Textual Representations of Scores	
Master	0
Excellent	1
Good	2
Fair	3
Poor	4
Bad	5
Are you trying?	6
Eeek!	7

Phase 2: User Tuned Preferences Per Selection Technique

Selection Technique	1	2	3	4	5	6	7	8
Occlusion Selection	-6.99 25.0 0.0 20.60	-1.16 29.57 -1.64 1.83	-8.00 25.00 0.0 16.02	-6.52 32.09 3.27 4.47	-8.86 28.42 .89 0.00	-8.00 25.00 0.00 3.00	-.10 25.88 -3.87 3.00	-8.00 25.00 0.00 10.00
Ray Casting	0.00 0.00 18.63	-1.53 2.35 4.54	35.0 -10.25 25.00	-7.32 -8.17 6.91	31.52 -22.46 22.86	-5.00 0.00 10.00	22.23 5.13 10.59	-5.00 0.00 10.00

Phase 3: Questionnaire

Environments 1 and 2									
User	1	2	3	4	5	6	7	8	

Satisfaction with[1]										
	Ray Casting	2	5	2	1	5	1	1	5	2.75
	Accuracy	2	5	2	2	5	1	1	3	2.625
	Speed	3	5	1	2	5	1	1	3	2.625
	Configuration	2	5	2	1	5	1	1	*	2.428571
	Occlusion	2	1	1	5	1	3	4	1	2.25
	Accuracy	2	1	1	5	1	1	4	1	2
	Speed	2	1	1	5	1	4	3	2	2.375
	Configuration	1	1	1	5	1	2	2	2	1.875
Compare Ray and Occlusion Selection[2]										
	Close	3	5	5	1	5	4	1	4	3.5
	Distant	1	5	5	1	5	1	1	4	2.875
	High	1	5	5	1	5	1	1	4	2.875
	Low	2	5	5	1	5	1	1	4	3
	Left	2	5	4	1	5	1	1	4	2.875
	Right	2	5	5	1	5	1	1	4	3
Consistent[3]		2	1	2	2	1	1	2	1	1.5
Effort[3]		1	1	5	3	4	4	4	2	3
Usefulness of multiple techniques[3]		1	5	4	5	5	3	3	5	3.875
Environment 3										
	Ray Casting	3	5	3	1	1	2	1	5	2.625
	Accuracy	2	5	3	1	1	2	1	4	2.375
	Speed	3	5	3	1	2	1	1	4	2.5
	Configuration	2	5	2	1	1	1		3	2.142857
	Occlusion	2	1	1	5	1	3	4	1	2.25
	Accuracy	1	1	1	5	1	2	3	1	1.875
	Speed	2	1	1	5	1	1	4	2	2.125
	Configuration	1	1	1	5		1	1	2	1.714286
Compare Ray and Occlusion Selection[2]										
	Close	3	1	5	1	5	3	1	5	3
	Distant	1	5	4	1	1	1	1	5	2.375
	High	1	5	5	1	1	1	1	5	2.5
	Low	2	5	4	1	1	1	1	5	2.5
	Left	2	5	5	1	5	1	1	5	3.125
	Right	2	5	5	1	5	2	1	5	3.25
Consistent[3]		3	1	4	3	1	3	2	1	2.25
Consistent vs Experiments 1 and 2[4]		3	1	3	3	1	5	1	3	2.5
Effort[4]		2	1	5	3	5	5	4	3	3.5
Usefulness of multiple techniques[3]		1	5	4	5	1	5	4	5	3.75

1: On a scale of 1 to 5 with 1 meaning satisfied and 5 meaning unsatisfied.

2: On a scale of 1 to 5 with 1 being ray casting and 5 being occlusion selection.

3: Lower values mean more emphasis

Phase 3: Selection Preference for Environments 1 and 2

(calculated from user questionnaire)

Close	Left	Right
High	3.083333	3.125
Low	3.125	3.166667
Distant	Left	Right
High	2.875	2.916667
Low	2.916667	2.958333

Phase 3: Selection Preference for Environment 3

(calculated from user questionnaire)

Close	Left	Right
High	2.875	2.916667
Low	2.875	2.916667
Distant	Left	Right
High	2.666667	2.708333
Low	2.666667	2.708333

Vita

Chadwick A. Wingrave

The author graduated with a B.S. degree from Virginia Tech in Spring 1999. He subsequently enrolled as a Masters student and graduated in the Summer of 2001. His ambition is to address complexity issues with realistic, user-centered tools. To this end, he completed this thesis to address the complexity issues in virtual environment interface design with machine learning techniques. He has enrolled as a PhD student at Georgia Tech.

His dislike of writing useless information, as can be noted by the nuances of this work, prevents him from writing further. Find a webpage on the author, as it will undoubtedly be more up-to-date.