Performance and Usability of Force Feedback and Auditory Substitutions in a Virtual Environment Manipulation Task

Gregory W. Edwards

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

Master of Science
in
Industrial and Systems Engineering

Dr. Woodrow Barfield, Co-Chair
Dr. Maury Nussbaum, Co-Chair
Dr. Brian Kleiner, Committee Member

November 3, 2000
Blacksburg, Virginia

Keywords: Virtual Environment, Haptic, Force Feedback,Sensory Substitution, Sound, Phantom

Performance and Usability of Force Feedback and Auditory Substitutions in a Virtual Environment Manipulation Task

Gregory W. Edwards

(ABSTRACT)

Recent technology developments have made possible the creation of several commercial devices and a selected number of development platforms for the inclusion of haptics (the sense of touch) in virtual environments (VE). This thesis sought to investigate and develop a better understanding of whether or not haptics or sound substitutions improved manipulation performance or usability in VE applications. Twenty-four volunteers (12 males and 12 females) participated in a 2 (haptics) x 2 (sound) x 2 (gender) mixed factorial experiment in which they completed a VE manipulation task involving the assembly and disassembly of 5 interconnecting parts. Performance for the manipulation task was measured through completion time and the number of collisions made, as well as subjective measures of usability.

Results indicated that completion times were slower and collision counts were higher for males with the addition of haptics ($p_{time} = 0.03$; $p_{collisions}<0.05$), while females exhibited a smaller increase in collision counts and no increase in completion time with the addition of haptics. Nonetheless, there were improved usability attributes when haptics were incorporated, more specifically, an increased sense of realism, perceived helpfulness and perceived utility in a design task ($p<0.05$ for all). Sound was found to be an effective substitute for haptics in most measures taken while the combination of sound and haptics versus either alone, did not demonstrate any signs of improving performance or any usability attributes. It is therefore recommended that sound substitution be used in VE manipulation tasks where the extra haptic information is desired, and minimizing completion time or collisions are the overall goal. Finally, for the utility of the feedback towards a design task, users ranked haptics as being more useful than sound, but ranked the combination of sound and haptics as being the best feedback condition ($p<0.05$). Further research is required to determine whether this belief is consistent with objective measures.

Acknowledgements

First I would like to acknowledge that the research described in this document was made possible in part by an equipment grant from ONR (NOO0149710388). This grant was made to Virginia Tech with the goal of supporting research in the areas of virtual and augmented reality. Equally important to me was a postgraduate grant that I received from the Canadian National Science and Engineering Research Council that allowed me to attend Virginia Tech in the USA. I would also like to thank Dr. Kriz and all the members of the Scientific Visualization lab (the CAVE) who supported me through a research grant while at Virginia Tech and who gave me the chance to learn and refine many of the programming skills that allowed me to complete my thesis.

A special thanks goes to the members of the Virtual Environment Lab, especially Eric Nash, who was a great source of inspiration, entertainment and ideas, and without whom I'm sure my days at Virginia Tech would have been much less enjoyable. I'd also like to thank my entire committee for their support and technical expertise. Especially, Dr. Woodrow Barfield and Dr. Maury Nussbaum, both of whom were very generous with their time and advice.

Thanks also go to my family and friends. My parents and brother for their encouragement and support during my time at Virginia Tech and my friends, Erik Olsen, Eric Nash, Jennifer Thompson, Kristen Gardberg, Jimmy Vu, Craig Hamrick, Ken Klauer and Gary Olacsi for the good times and camaraderie that made my two years at Virginia Tech genuinely enjoyable.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

As virtual environments (VE) become more popular for training, visualization and entertainment, they have begun appearing more frequently within the workplace. Useful applications such as vehicle simulation for training pilots and drivers, vehicle design-styling, architectural and landscape design, manufacturing parts design, NASA astronaut training and surgeon skill training are but a few of its current uses (Brooks, 1999).

In a typical VE application, the user will wear a head mounted display (HMD) or a pair of stereo shutter glasses that immerses them within a visual world of a given resolution and field of view (FOV). Most recent VEs have provided little haptic cues (haptics refers to the sense of touch; Fritz and Barner, 1999), and no olfactory stimulation, leaving the user to rely purely on visual and auditory cues to convey all the information (Barfield, Hendrix, Bjorneseth, Kaczmarak and Lotens, 1995). For some applications, visual and auditory cues may be sufficient, but several experiments have shown potential limitations with this approach. Examples include decreased navigational ability (Bakker, N., Werkhoven, P., Passenier, P., 1998; Gunther, 1997), difficulty in manipulating objects (Richard, Birebent, Coiffet, Burdea, Gomez, Langrana, 1996) and a decreased sense of presence (Sheridan, 1992a; Witmer and Singer, 1994). The lack of haptic stimulation has been mainly due to expensive devices, limitations associated with the technology and difficulties in integrating realistic haptic interactions with the correct physics. Yet, with the recent commercialization of new haptic technologies and software development platforms (e.g. GHOST from Sensable, LHX from Iwata, Yano and Hashimoto, 1997, G_2H by Acosta, Stephens, Temkin, Krummel, Gorman, Griswold and Deeb,1999), the use of haptics may become more widespread.

A question now faced by many VE developers is whether or not haptic technology enhances performance, making it worthwhile to incorporate despite the extra cost and development effort. In certain situations, such as training motor skills for surgeons, there is little debate over the necessity of haptic feedback. On the other hand, applications such as simulating the assembly of an industrial part may or may not benefit from haptics. Given the widespread use of auditory cues and the limitations with current haptic technologies, one alternative is the use of auditory

cues as a substitute for information typically presented through the haptic sense. A central aim of this research is to investigate whether haptic cues increase performance or usability on a VE manipulation task, and if so, whether auditory substitutions for these cues could be just as effective. Designers will be able to use the information gained form this study to create VEs that are easier to use and more productive.

This thesis has been divided into seven chapters. The first chapter (the introduction) gives an overview of the topic area and some of the motivation behind the research. The second chapter provides a brief background on the topics to be covered (including research on virtual environments, haptics and sound) and is followed by the third chapter which presents the research purpose, objectives and hypotheses. A separate description of each of the two pilot studies and the main study are then given in chapters four, five and six respectively, providing the methods, results and discussion for each. Finally, conclusions are given along with design guidelines and recommendations for future research in the seventh chapter.

Chapter 2: Literature Review

The term *virtual environment* does not yet have a consensus definition. Some believe that it is any computer generated three-dimensional graphical program that uses an HMD with head tracking (e.g. Pausch, Proffitt and Williams, 1997), while others believe that it encompasses any computer generated environment that attempts to immerse the user within it (e.g. Sheridan, 1992a). In the present work, the latter definition was used, which emphasizes the user's personal experience, as opposed to the definition referring to the exact equipment employed. Achieving a sense of immersion, or presence as it is more commonly referred to, is what distinguishes a VE from most other computer programs.

2.1 Virtual Environment History

The first attempt at a VE is often credited to Ivan Sutherland in 1968 (Harrison and Jacques, 1996). Using an HMD that employed two miniature TV screens for presenting stereo images and a sensor to track head movements, he provided the user with the illusion of residing in the same virtual space as wire-frame graphical objects. With advances in computing power, better graphics can now be processed at faster speeds for these environments. In addition, the technology used for HMDs and stereoscopic projection viewing has advanced such that the required devices are now only slightly larger than regular glasses (but still quite a bit heavier).

In terms of auditory stimuli, research and technology development began a little later, but have already reached substantial levels of fidelity both in terms of the sounds used as well as the methods of presentation (more detail in *2.4 Sound in Virtual Environments*). Most virtual environments, however, are limited to only visual and auditory stimuli, leaving out the haptic and olfactory modalities completely. One notable exception was a video game called *Sensorama,* created in the mid-1960's by Morton Heilig. Using a binocular viewing system that allowed three-dimensional visuals and a wide field of view, the game showed a stereo-film-loop of a motorcycle ride through New York City. Special aspects of this game included three-dimensional binaural sound, haptic interaction with a simulated bike seat (vibration cues) and handlebars, a fan to blow wind in the participant's face and a chemical bank to simulate smells (Fisher, 1990). There are few, if any, current simulations that come close to the amount of

sensory data and fidelity provided by this application. To give credit to our current designers however, this game used actual video footage, which is much easier to create than virtual imagery. Nonetheless, the idea of immersing all the senses was present and this is what is lacking in most current VEs. This thesis focuses on the sense of touch (haptics) in virtual environments, the next essential modality to be incorporated. The following section describes the haptic modality and then discusses relevant research that has been done in the field of haptic VEs.

2.2 Haptics

Haptics refers to the sense of touch (Fritz and Barner, 1999). Haptic information is conveyed to us through an array of different sensory organs embedded below the skin (Asamura, Tomori and Shinoda, 1998). Specialized organs sense different stimuli (such as heat, pressure, or vibration) and depending on the type, magnitude and location of the stimuli on the skin, a response is triggered in one or many of the receptors. The signal is then transmitted up the spine and into the thalamus region of the brain. From there more neurons complete the pathway of the signal ending in the somesthetic area of the cortex (Burdea, 1996).

The haptic sense is comprised of both tactile and kinesthetic perception (commonly referred to as force feedback). Tactile perception provides the feeling of texture and the frequency of vibration of surfaces. Hand receptors by themselves are connected to nearly a quarter of the total cortex surface, giving the hand a high sensitivity to external stimuli (Burdea, 1996). A human finger can sense tactile input at a bandwidth of 0-400 Hz, but can sense vibrations up to somewhere between 5,000 and 10,000 Hz during skillful manipulation tasks (Burdea, 1996).

Kinesthetic (force feedback) perception, alternatively, allows the sensing of total contact force, the compliance of objects (how hard they are) and their weight. Low-bandwidth receptors placed deeper in the body, usually on tendons (Burdea, 1996) are used for kinesthetic sensing. The maximum exertable force from a finger is approximately 50 to 100 Newtons (N) (Salisbury and Srinivasan, 1997). According to researchers at Sensable technologies however (the creators of the PHANToM™), people rarely exert more than 10 N during precise manipulation (Massie and Salisbury, 1994). Other relevant human capabilities related to the kinesthetic sense include

the frequency at which humans can sense forces, 20 to 30 Hz, and the frequency at which humans can exert forces or movements, 5 to 10 Hz (Burdea, 1996).

2.2.1 Haptics and Virtual Environments Background

Earlier work in the haptic simulation field began with Telerobotics. There was (and still is) a need to control robots to do precise manipulation tasks in areas that were either too dangerous or costly for humans to be present. Examples of such applications include handling of materials in chemically hazardous environments, movement of materials in space or on the ocean floor, the disposal and diffusing of bombs and mines, as well as the precise movements required for microsurgery or the large motions required for moving heavy materials (Ellis, Ismaeil and Lipsett, 1996). In these scenarios, the telemanipulation task involves the human receiving mainly kinesthetic force feedback from the remote robot and then responding by sending appropriate manipulation commands (Sheridan, 1992b). The device that was in contact with the human was typically called the master and the remote robot was named the slave, hence the popular master-slave terminology used in the Telerobotics community. Force feedback telerobotic work was being conducted as early as 1954 when the E-3 robot arm from Argonne National Laboratories was described in print (Goertz and Thompson, 1954). The master had six degrees of freedom and could provide feedback to the human in all six force/torque components. Although the device could provide up to 35 N of force back to the human, it suffered from high static friction and a very low update rate of about three Hz (Goertz and Thompson, 1954). Recalling that the human can sense kinesthetic feedback at a rate of 20 to 30 Hz, it is not surprising that the movements of the E-3 were perceived as being very rough and "jerky". In the 1960's, a haptic manipulator was used as a controller in a computer application for the first time with a program created by Knoll at Bell Labs (Salisbury and Srinivasam, 1997). Since then many manipulators have been created for both teleoperation and virtual environments. Nonetheless, until recently there have been few commercially available systems due to problems in providing enough force from a device that was still small, light and safe (Burdea, Richard and Coiffet, 1996). Systems that provided realistic force feedback were also heavy and bulky and would easily tire the user, in addition to being extremely complex and expensive.

2.2.2 Current Devices and Applications

Current systems have achieved a balance between less degrees of freedom and/or force feedback, but with higher update rates and smaller, lighter less complex systems. There are two general categories of current systems, tactile systems and force feedback systems. A brief overview of tactile displays will be presented first before concentrating on a description of force feedback systems (the primary technology used in this research). It should be noted, as will be discussed later in this section, that some force feedback devices also provide limited synthetic tactile sensations through the application of high bandwidth force feedback (similar to vibrotactile displays).

Tactile Feedback Interfaces: Traditional tactile display systems use pneumatic, vibrotactile or electrotactile stimulation, with functional neuromuscular stimulation (FNS) appearing recently (Burdea, 1996). Pneumatic systems employ either air jets or air pockets that inflate and push the lining of a glove against the skin. An example of such a system is the Teletact II originally made by Stone and Henequin and subsequently remodeled by Advanced Robotics Research Ltd. and Airmuscle Ltd. (Stone, 1992). The system is a glove with a series of separate air pockets along each finger and the palm of the hand. Each pocket is individually activated through a control board connected to the simulation (Stone, 1992).

Vibrotactile systems use vibrations presented to the fingertips through either voice coils, miniature loudspeakers or spatial arrays of micropins (Burdea, 1996). A system that uses voice coils at the user's fingertips is the Touch Master by Exos Inc. (Burdea et. al., 1996). These disks are voice coils that vibrate at a set frequency, but at different amplitudes to provide different sensations of touch.

Electrotactile stimulation uses electrodes placed on the skin to pass very small currents through the local tactile receptors (Burdea, 1996). This method has higher risks associated with it since it uses electrical currents applied directly to the skin. Nevertheless it is an important area of research because it is lighter, less power consuming and consists of less moving parts than the previous methods mentioned here. The Tacticon 1600 is an example of such a system that is

used to replace sounds for the deaf by substituting selected frequency bands by electrodes placed on the skin (Zhu, 1988).

FNS places electrodes in the muscles and even in the nervous system to stimulate the muscles or nervous system electrically and simulate the feeling of touch. These systems are invasive, can produce pain and are rather complex (Burdea, 1996). For this reason, the vast majority of interfaces use the non-invasive methods mentioned previously. Given the limited ability of these devices to only emulate tactile sensations, most VE developers are more interested in the force feedback devices.

Force Feedback Interfaces: Force feedback devices that are currently being marketed can be classified into two general categories: desktop and portable devices. Desktop devices generally use electrical actuators whose weight are supported by a desk or table-top. They are commonly able to produce higher resistances to motion than the portable devices, but have more limited workspaces.

These desktop devices can be further subdivided into teleoperator exoskeleton arms, more precise desktop manipulators and joysticks. Teleoperator exoskeleton arms attach to the user's full arm and were developed for controlling robots with the same geometry as the controller (Sheridan, 1992b). Generally these devices provide higher force feedback capabilities to the whole arm and wrist, but are large in size and weight and are plagued by slower update rates and larger resistances to the user's motion. Smaller resistances and much lighter configurations are available with the more precise desktop interfaces. Sensable Technologies makes such a device, called the PHANToM™, that provides feedback to one finger in three degrees of freedom (DOF) and measures position in six DOF (Salisbury and Srininivasan, 1997). The PHANToM™ provides smaller amounts of force feedback than the full arm devices, but is more precise and much less intrusive on the user's movements. This research project used a PHANTOM™ which will be described in more detail in Section *4.4 Equipment*.

A very different type of precise desktop device is the stringed force-feedback interfaces. Forces are typically provided to the hand or finger through steel cables or strings attached to

actuators on a cubic support structure. An example of such a system is the Space Interface Device for Artificial Reality (SPIDAR) II developed by Ishii and Sato (Ishii and Sato, 1994). The SPIDAR II is a cube with a string running from each of the cube's vertices to attach to the user's index and thumb (4 strings to each finger). Through actuators at the ends of the strings and rotary encoders, string length and tension can be modified and measured.

The final type of desktop force feedback equipment is the joystick, which generally provides even less feedback capabilities in only two DOF, but is much simpler in design and substantially less costly than the other two categories (Johansson and Linde, 1999). The Immersion Impulse Engine 2000 is an example of a high quality two DOF force feedback joystick with low inertia, low friction, and high bandwidth. Higher DOF joysticks also exist, such as the Haptic Master by Missho Electronics Co., which provides six DOF within a 40 cm diameter sphere. Control on the Haptic Master is provided through a small ball that is held in the air by three arms attached to a desktop platform (it has a maximum force output of 69 N).

Many of the force feedback devices just described can also provide limited tactile stimulation by providing high frequency and low amplitude force feedback. Both the PHANToM™ and Impulse Engine 2000 for example, are capable of allowing the user to feel rough or smooth surfaces through vibrations and very fine force feedback modulations transmitted through the haptic interface. High frequency control loops are used to allow this interaction.

The second category of force feedback controllers are the portable units. These mechanisms are not attached to a desktop and are able to follow the user's hand through a larger range of motions. In addition, portable units often come in the form of gloves that allow separate input/output to each finger, permitting grasping motions and sensations. These systems typically use air power or tension cables. Early air powered devices were developed for tele-robotic control of grippers in the 1960's (Springer and Gadh, 1997). Air bladders attached to a glove provide resistance to finger bending with the increase in air pressure. Unfortunately, these gloves were generally thick and provided a lot of resistance to natural hand closure (Springer and Gadh, 1997). More recent air-pressure devices include the Rutgers Master II, which utilizes air

cylinders mounted at the palm and running towards four of the fingers to provide resistance (Gomez, Burdea and Langrana, 1995). Alternatively, cable gloves have cables running along the outside surfaces of the finger and a force is simulated by the cable being pulled (tightened), providing resistance to the inward motion of the finger. These devices also have problems though, from friction of the cable and difficulties in keeping the cable in the desired position with high forces (Springer and Gadh, 1997). An example of a current cable glove is the CyberGrasp™ from Virtual Technologies Inc. (see www.virtex.com).

2.2.3 Benefits of Force Feedback

The benefits of having force feedback were first investigated with the telerobotic tasks described earlier. One such study, by Hannaford and Wood (1989), examined three different telerobotic manipulation tasks under three conditions: direct manual control, remote telerobotic control without force feedback and remote telerobotic control with force feedback. In the teleoperation conditions it was found that the addition of force feedback increased performance significantly with all measures: 30% time reduction, 60% decrease in errors, and a reduction in the force exerted by a factor of seven (Hannaford and Wood, 1989). A second experiment, by Howe and Kontarinis (1992), studied performance of a two DOF peg-insertion task with and without force feedback. Results showed that time to completion was twice as fast with force feedback, as opposed to only visual feedback. A similar study by Massimino and Sheridan (1993) showed equal performance with or without force feedback on a two DOF peg insertion task. The results may have varied due to the different types of devices used and the fidelity of the feedback (the device used by Howe and Kontarinis provided a grasping interface as well). Another study by Massimino and Sheridan (1993), using the same controller device, indicated that the addition of haptics in a three-dimensional peg insertion task actually impeded performance. These varying results may indicate that the utility of force feedback depends on the device used as well as the fidelity of the force feedback provided by the controller and the simulation.

As research with force feedback and telerobotics advanced, similar studies with virtual environments were performed. One study investigated performance of a molecular docking task with and without force feedback (the GROPE project; Brooks, Ouh-Young, Batter and

Kilpatrick, 1990). Not only was performance shown to improve by a factor of two with force feedback compared to only a visual display, but chemists were reported having a "radically improved situation awareness" of what was occurring. Another study, by Ishii and Sato (1994) varied the simulated weight of objects being manipulated in a VE. The experiment consisted of a pick-and-place task of a cube with different weights, using a desktop mounted two-handed input device (called a SPIDAR). Completion time increased with weight (going from 5.1 seconds for a 20 gram weight to 5.8 seconds for a 150 gram weight), but accuracy was best for 35 to 50 gram cubes (with little completion time degradation). Yet another pick and place experiment was conducted by Richard and Coiffet (1995) using a Rutgers Master™ interface glove. Their results showed that haptics improved time to completion (by 30%) and diminished the required grasping force (by 50%; they were instructed to grasp the ball at 10% object deformation). Improvements in completion time were hypothesized to have been caused by a decrease in mental workload. The study also examined sensory substitution and these results will be presented in Section *2.3 Sensory Substitution and Augmentation*.

2.2.4 Force Feedback Applications

Applications using haptic feedback in virtual environments are now numerous and include the following listing:

- Three-dimensional medical imaging and training simulations (Zajtchuk and Satava, 1997; Poston and Serra, 1996).
- Protein molecule docking simulations (Brooks, Ouh-Young, Blatter and Kilpatrick, 1990).
- Three-dimensional data haptization (Durbeck, Macias, Weinstein Johnson and Hollerbach, 1998).
- The recent use for industrial computer automated part design and modeling (Springer and Gadh, 1997; Lin, Gregory, Ehmann, Gottschalk and Taylor, 1999).
- The simulation of industrial manipulation tasks (assembly, maintenance, training, etc.; Luecke and Zafer, 1998; Gutierrez, T. Barbero, J.I., Aizpitarte, M., Carrillo, A.R. and A. Eguidazu, 1998; Hollerbach, Cohen, Thompson, Freier, Johnson, Nahvi, Nelson and Thompson II, 1997).
- Presenting information for the visually impaired (Fritz and Barner, 1999; Grabowski and Barner, 1998; Jansson ,1999),
- Telerobotic control (via a VE interface; Baier, Buss, Freyberger, Hoogen, Kammermeier and Schmidt, 1999; Buttolo, Kung, and Hannaford, 1995), and many other applications not listed here.

2.2.5 Limitations of Haptic Technology

There are three main factors that make incorporating haptics into virtual environments difficult: 1) Input device realism and cost; 2) complexity of the VE and update rate; and the 3) type of force feedback required.

1. Input Device Fidelity and Cost: A 2 degree of freedom (DOF) high quality joystick can cost as little as $4,000 (the Impulse Engine 2000™), but prices for higher DOFs and more precision can cost up to $100,000 for a full 6 DOF robotic arm interface (Springer and Gadh, 1997). The cost of the device (a PHANToM 1.5A) used for this study was approximately $27,500 while this thesis was being written.

2. Complexity of the VE and Update Rate: Complexity in this case refers to the number and size of the objects to be simulated and the types of haptic interactions required. The number and size of the objects influences the speed of the simulation because for every graphical object that requires haptic feedback a separate rendering of the object must be done with the specific haptic properties for that object. In addition, if the objects are allowed to move or be deformed then the two renderings (graphical and haptic) must be kept synchronized such that a haptic deformation also produces a graphical deformation and possibly an auditory effect as well. Yet another factor to take into account is the realism of the interactions between objects. If the deformations and reactions are required to be physically accurate then more mathematical functions and physics equations must be incorporated (people are still working on efficient algorithms for these). Thus, more complex VEs require extra haptic programming and faster computers.

3. Type of Force Feedback Required: The third factor is whether tactile or only force feedback is required. Tactile feedback requires higher update rates to the haptic interface since the required bandwidth for sensing is much higher (tactile sensors can sense at a rate of 400 Hz, but up to 10 kHz for vibrations; Burdea, 1996). Force feedback by itself requires much less in terms of the update rate since the kinesthetic receptors only sense movements or forces between 20 to 30 Hz (Burdea, 1996).

In summary, the applications requiring the most work and resources for the integration of haptics are those that require a large number of degrees of freedom and realism in the interaction device, whose VE is large and complex in terms of the number of objects and their complexity, that requires high realism in the simulated interactions, and that requires tactile sensations. For these reasons and because of current limitations of the haptic devices, the use of sensory substitution and or augmentation of the haptic feedback may currently be a viable and required solution for some applications and developers.

2.3 Sensory Substitution and Augmentation

Sensory substitution can be defined as "the provision to the brain of information that is usually in one sensory domain by means of the receptor, pathways and brain projection, integrative and interpretative areas of another sensory system." (Bach-y-Rita, Ebster, Tompkins and Crabb, 1987) Sensory substitution is most commonly used for blind people by replacing written text with braille (the visual modality replaced by tactile) and for deaf people by replacing speech by sign language (the auditory modality replaced by vision). Another area of sensory substitution is force feedback substitution, which involves using the visual, auditory or tactile domain to replace information conveyed by forces. Force feedback substitution has been done in the past for teleoperation tasks where it is useful for the operator to know something about the grasping forces and restraints present at the remote site. One of the reasons for such substitution, in addition to the limitations of force feedback presented above, is that these applications commonly have significant transmission (and other) delays, making force feedback control extremely difficult for the operator (Massimino and Sheridan , 1993). Some of the research on force substitution will now be discussed.

A study by Massimino and Sheridan (1993) had participants complete two teleoperation tasks with two identical 6 DOF robot arms. The first task was to tap the tip of the slave arm back and forth between two objects as fast as possible. Feedback presented during the task took four different forms (all with visual feedback): no force feedback, vibrotactile feedback presented to the thumb and index finger with the vibration amplitude proportional to the contact force, force feedback presented through the master arm, and auditory feedback with the loudness proportional to the contact force magnitude. When the objective was only to tap as fast as

possible it was found that auditory feedback was the best condition followed by vibrotactile and then force feedback. When the task also involved the objects being able to move when tapped at a force of 2 lbs. and that the amount of movement involved a penalty, the results were very similar. The results from these two tasks indicate that the force required for movement of the blocks was so great that it did not impede their speed of tapping back and forth. The task therefore became one of reaction speed and the auditory and tactile feedback conditions allowed smaller forces to be detected faster than the force feedback condition. The slower detection times with force feedback were due to the high weight and design of the telerobotic arms used, which was confirmed by the authors through psychophysical testing (Massimino and Sheridan, 1993). When the force to move the objects was lowered to 0.5 lbs. the subject had to pay a lot more attention to the force and speed that they hit the objects with. In this condition it was found that the vibrotactile, auditory and force feedback conditions were approximately equal in performance. Therefore, when perceiving light forces is required, the substitution of force feedback with vibrotactile and auditory cues exhibited equal efficiency.

The second task in this same study by Massimino and Sheridan (1993) was slightly more complex, and involved inserting a peg into a hole with and without an obstructed view AND with and without a 3 second time delay. For the visual condition with no delays they found that force feedback and auditory displays actually reduced performance. The force feedback impeded movements and the auditory cues required for representation of three-dimensional movement were so complex that they were hypothesized to have overloaded the operator. When time delays were present, force feedback created instabilities for the operator (impedes movements when not expecting it, etc.), and auditory and vibrotactile feedback both outperformed the visual only condition by approximately 30% improvements in completion time. When the view was obstructed, force feedback outperformed both other conditions significantly (less than half the completion time of the other conditions), but the other conditions still allowed completion of the task.

Auditory substitution was also considered by Richard and Coiffet (1995) in the efficiency of moving a virtual ball from one location to another using a force feedback glove. Dependent measures were the amount of object deformation (an error was recorded if the participants

deformed the ball by more than 10%) and the time to complete the task. Their results showed that the task was completed faster and with more constant and low object deformation when both auditory and haptic feedback were provided. Moreover, this study looked at adding additional visual cues of force feedback, but implemented them on a separate display from the VE display, which required the user to timeshare between the two and which ultimately led to reduced performance.

A slightly more complex manipulation task (a peg-in-a-hole type insertion task) was examined by Gupta, Whitney and Zeltzer (1997), with and without force feedback and with and without auditory augmentation. They found that force feedback (provided through 2 PHANToMs) improved performance by a factor of 1.3, while the addition of auditory cues did not change performance. The authors noted, though, that the task only presented auditory information when the part was being inserted so that it was not a very auditory intensive task.

The use of the auditory modality over tactile (vibratory or electrical) and visual cues for substituting force feedback is of interest for several reasons. First, sending more information to the visual modality would most likely not show performance improvements since it is one of the most heavily loaded modalities in a typical virtual environment. Second, the auditory modality has a large area in the sensory cortex allocated to it for information processing (Richard and Coiffet, 1995). Much of this processing potential is not used in current VE simulations. Third, the auditory modality has a much wider bandwidth versus tactile senses for processing information, with 10^4 bits/sec for auditory cues as compared to 10^1-10^2 bits/sec. for tactile cues (Fritz, 1999). Fourth, the auditory modality can augment haptic interactions by providing secondary information such as the composition of the material through tapping (Ruspini and Khatib, 1998) or increasing the apparent stiffness of an object (DiFranco, Beauregard and Srinivasan, 1997). Augmenting the sense of solidity of an object can also reduce the required mechanical actuator force of afeedback device. The next section briefly describes the advances in technology regarding sound and its use in VEs.

Although it is not the focus of this thesis, research has also been done with force feedback and visual substitution for the blind. The reader is referred to Grabowski and Barner, 1998; Jansson, 1999; and Fritz, 1999 for more information on these studies.

2.4 Sound in Virtual Environments

Sound cues can produce an alerting or orienting response, can be detected more quickly than visual signals and can be heard simultaneously in three dimensions (Wenzel, 1992). Sound is also especially useful for conveying state changes over time, since the human is extremely sensitive to changes in an acoustic signal over time (Wenzel, 1992).

The technology used to emulate sounds is in many cases superior to our capabilities of detection, both in terms of threshold emulation as well as acuity and resolution performance (Barfield et. al., 1995). Nonetheless, work remains to be done with spatialized sound, model-based sound and virtual acoustics (Anderson and Casey, 1997). Spatialized sound, as the name implies, allows the user to sense sounds in all three dimensions as if the objects really were in front, behind or above the user for example. Model-based sounds incorporate the structure of objects in a physical model and will create sounds after having solved the wave equation of the appropriate model. This allows different sounds to be produced depending on where and when an object is tapped for example, creating much more realism than the popular sampling method which typically involves one sound per object. Virtual acoustics refers to the possibility of hearing sounds appropriate to the environment that is simulated (e.g. the echoes in a concert hall versus a broom closet). Research on sound in virtual environments is focusing on these issues and the associated technology (Begault, 1994; Anderson and Casey, 1997). The research described in this thesis uses simple non-three dimensional sounds to represent collisions and contacts between objects all within the forward field of vision (reducing the need for spatialized sound). The specific auditory cues and hardware to be used are described in sections *4.5 The Task* and *4.4 Equipment,* respectively.

2.5 Gender, Spatial Ability and Fine Motor Skills

2.5.1 Spatial Ability Differences by Gender

Spatial ability refers to a person's "skill in representing, transforming, generating, and recalling symbolic, nonlinguistic information" (Linn and Peterson, 1985). Spatial ability can be subdivided into three categories: spatial perception, mental rotation and spatial visualization. Spatial perception relates to a person's ability to determine spatial relationships with respect to the orientation of their own body. Mental rotation ability refers to how rapidly and accurately one can rotate a two or three-dimensional figure. Spatial visualization encompasses those spatial ability tasks that involve complicated, multistep manipulations of spatially presented information. Investigation into the effects of spatial ability, related specifically to virtual environment performance, has been limited mainly to studies investigating navigational performance in large virtual environments. These studies have found mixed results with spatial ability tests sometimes showing correlations to performance and sometimes not (Satalich, 1995; Edwards, Thompson and MacGregor, 1998).

Gender related differences in spatial ability have been widely cited in the literature with significant and consistent results showing males to perform better on spatial ability tests (Crawford and Christensen, 1995; Linn and Petersen, 1985; Vandenburg and Kuse, 1979). However, few, if any VE studies have considered gender differences. A study by Edwards, Thompson and MacGregor (1998) is one of the few studies that has taken gender into consideration. The task was a navigation task and results indicated significant gender differences in the ability of participants to gain navigational knowledge from the virtual environment. Although similar gender consideration has not been reported in the research on **VE manipulation tasks**, to the knowledge of the author, there **has** been research on gender differences in fine motor skills.

2.5.2 Fine Motor Skill Differences by Gender

Many studies have examined gender differences on fine motor skill ability and concluded that females, in general, perform better (e.g. Philips, Paterson and Pettijohn, 1985, Peters, Servos and Day, 1998). The most notable differences have been observed during a peg-in-hole insertion

task in which the participants are instructed to insert small pegs (about 3mm in diameter) into holes as fast as they can. Recently however, an alternate explanation for the differences was proposed by Peters, Servos and Day (1998). In a similar insertion task, finger and thumb size were measured for all participants and correlated to the results. The study found that using finger size as a covariate eliminated any sex differences, thus concluding that the faster performance was in fact due to finger size and not gender differences. Smaller fingers allowed the females to pick up the small pegs and manipulate them faster. A study by Kilshaw and Annett (1983) confirmed this theory with a similar peg-in-hole insertion task that used larger pegs with 9.5 mm diameters and found that males were faster than females at this task. Many theorize that the general difference in motor skill ability between genders is hormonally based (Hampson and Kimura, 1988), but a discussion of these issues is beyond the scope of this research.

2.6 Focus on Industrial Applications

Industrial design is one new area of VE simulation where it is uncertain if haptic cues will increase performance. Industrial designers often create designs for parts in computer aided design (CAD) packages, allowing them to specify specific dimensions in a very structured and exact way. Conversely, these programs are "tedious, detail oriented, and time consuming, and because of this CAD cannot be utilized in the early stages of a design project" (Springer and Gadh, 1997). Additionally, it is very difficult to visualize an assembly or disassembly procedure with multiple parts together with their interactions (Gutierrez et. al., 1998). As a result, many of the initial design decisions are done before the CAD model is created and abstract assembly and disassembly planning algorithms are used to help conceptualize the parts (Jayaram, Wang, Jayaram, Lyons and Hart, 1999; Gupta, Whitney and Zeltzer, 1997). Often mock-ups are created later in the design stage to verify the fit of the parts as well as the efficient accomplishment of assembly and disassembly procedures. However, because these mock-ups are done late in the design stage, any problems that are found can lead to considerable lost time and backtracking. To avoid the shortcomings and problems caused by the use of tedious CAD packages, many designers are now looking towards virtual environment applications to provide a more intuitive design tool. VEs can be used earlier in the design process, can allow visualization of the parts and their assembly as well as their predisposition towards ease of maintenance procedures, and

can even be used to illustrate these procedures to the mechanics for use with the final products. The addition of haptics to this process potentially allows the designers to 'feel' the parts as they are manipulated or making contact with other parts, giving them a better awareness of the interactions and how well the parts fit together. Many researchers are currently studying the effects of haptics in industrial applications, how to incorporate them effectively and how to create virtual environment extensions to existing CAD programs (e.g. Jayaram et. al., 1999; Gutierrez et. al., 1998, SensAble, 2000, Luecke and Zafer, 1998, Springer and Gadh, 1997, Gupta, Whitney and Zeltzer, 1997).

Chapter 3: Research Objectives and Hypotheses

3.1 Overall Research Purpose

There are currently many difficulties with haptic interaction in VEs, including limitations of the technology, high costs, difficult programming and possibly large processing requirements (depending on the complexity of the required VE). Given these problems, many designers of VEs are uncertain of whether or not to incorporate haptics into their applications. The research undertaken by this thesis will aid designers in this decision with both objective performance data and subjective opinions on the use of haptics and auditory substitutions in a complex VE manipulation task. Based on the results, it is also hoped that this study will encourage other research efforts to examine different tasks and methods of incorporating the two modalities (haptic and auditory) effectively.

3.2 Research Objectives

There are several specific objectives associated with this research. The first is to investigate whether haptics increases performance and ease of interaction in a complex VE manipulation task. Previous studies have looked at more elementary pick and place tasks and found that the addition of haptics was effective in decreasing completion time and increasing accuracy, but few studies have looked at more complex tasks.

The second objective is related to the realism and the utility of incorporating haptics into a visual simulation. Haptic cues are hypothesized to create more realistic simulation interactions, thus creating a more intuitive interaction medium and allowing the simulation to be more helpful and productive. Hence, the second objective is to examine whether haptic cues make the interactions more realistic for the user and potentially allows them to do their job more proficiently. If this is not the case, then designers of VE systems where haptics are not an essential ingredient (i.e. NOT surgical simulators or any other simulation meant to train motor skills) will not have to incorporate haptics or invest the extra associated cost and development time.

The third objective is to investigate whether auditory cues provide an effective substitute for haptics. Using auditory substitutes will avoid the difficulties associated with the implementation of haptics in some VE applications (as detailed in Section *2.2.4 Limitations of the Technology*). The objective is to lessen the burden on the developer and to diminish system requirements.

The fourth objective is to explore whether having both auditory and haptic cues increases performance and subjective satisfaction. Many authors have stated that virtual environments will provide better immersion for the user if more sensory cues are provided. Nevertheless, it is not always the case that this will increase performance.

3.3 Research Hypotheses

There were several hypotheses related to the above mentioned objectives:

Objective #1: Effects of Haptic Cues on Performance and Ease of Interaction within VEs

1. The use of haptic cues will increase performance (reduce times for completion) by providing essential collision and interaction cues for the task. Task Performance will be measured by completion time and number of collisions with the surrounding parts and structure. The number of collisions is expected to increase due to the addition of gravity and reaction forces to all parts, however the completion time is expected to show decreases (faster times) based on earlier work by Richard et. al., 1996; Brooks, Ouh-Young, Batter and Kilpatrick, 1990; and Howe and Kontarinis, 1992. This combined effect is in opposition to results from Ishii and Sato (1994) who showed that accuracy increases but that speed decreased in a pick and place task with the addition of force feedback. The reason for the proposed disparity is that the task used for this study requires accuracy for faster completion (given by the haptics despite increased collisions due to reaction forces and gravity), whereas the task by Ishii and Sato (1994) did not require accuracy for faster completion.
2. Ease of interaction is expected to be greatest with haptic cues. This would be shown by higher Likert-type scale ratings for the haptic conditions on the questions related to ease of use.

Objective #2: Effects of Haptic Cues on the Realism of the VE Manipulation

3. The users will rate the haptic cues as giving higher fidelity and more information to be able to do an industrial design task (what this task resembles the most). This result is expected since haptics present more realistic information (especially with the addition of gravity and the weight of parts) and the user can immerse themselves better with this information.

Objective #3: Effectiveness of Sound Substitution for Satisfaction and Performance

4. The use of haptic feedback will lead to better performance than auditory substitution. Better performance would be shown by faster completion times for the haptics conditions than the auditory conditions (based on previous findings by Richard and

Coiffet, 1995). These anticipated findings are contrary to the peg-in-hole findings by Massimino and Sheridan (1993) that found that force feedback and auditory feedback impeded the task. The task in this study differs from theirs in that the auditory information is not as complex and overwhelming and the possibility of force feedback slowing them down should be compensated for over the larger number of interactions and the intuitiveness of force feedback for these interactions.

5. Auditory cues will not provide as much information as haptic cues for producing a realistic and productive application, but will provide more than no additional sensory cues. This hypothesis is expected to be shown through participants rating the haptics cues highest on the rating scales related to part design, followed by the auditory cues and then the control condition.
6. Subjective user satisfaction with the interface will be lower with auditory substitution (than with haptic cues), but higher than with no additional feedback. This is expected to be shown by the auditory condition having higher scores on the rating scales related to ease of use (as compared to the control condition), but lower scores than the haptic condition.

Objective #4: Effects of the Combination of Haptic and Auditory Cues

7. The use of both haptic and auditory feedback will lead to the best performance, shown by the lowest completion times. These expected results are based on previous findings by Richard and Coiffet (1995) who used a task similar in auditory intensity/complexity as opposed to the study by Gupta et. al. (1997) that used a dissimilar task and found that auditory augmentation did not help.
8. The ability to use the simulation for its intended design purposes will not be improved by the combination of both sensory cues. This is anticipated to be shown by non-significant differences between the related rating scale results for the combined condition as compared to the condition with just haptic cues.
9. Subjective satisfaction with the interface will be highest with both sensory cues. Rating scale results, related to ease of use and user satisfaction, are expected to be highest for the condition with haptics and auditory cues.

Chapter 4: Pilot Study 1

There were two pilot studies that were conducted to determine parameters for the main study. This chapter describes the first of these studies, which had as its main objectives:

1. To determine how many trials were required at each condition to obtain a steady level of performance.
2. To determine if there was a significant learning effect over the duration of the study.
3. To determine a suitable sample population for the main study (do a power calculation).
4. To determine if the task was of a suitable difficulty level.

4.1 Participants

Two males and two females between the ages of 20 and 30, with normal or corrected to normal vision were recruited from the Virginia Tech university student population. Participants were screened out of the experiment if they had extensive previous VE experience or any severe wrist problems. Screening based on VE experience was implemented during recruiting and was based on findings by Gunther (1997) that showed experienced participants were more adept at orienting themselves in VEs than the general population. Implementation of the screening on wrist problems was in response to the first participant in the study mentioning that the device aggravated a wrist problem that they had. Participants were also tested on their spatial ability through a Mental Rotations Test (Crawford and Christensen, 1995), to be monitored as a possible cofactor during the analysis (described in Section *3.4 Procedures)*. Compensation of $15 was given for participation in the study and no symptoms of simulator sickness were experienced.

4.2 Experimental Design

The design was a 2 x 2 x 4 within study with the first factor being haptic feedback (with or without), the second factor being sound (with or without) and the third factor being the number of trials (four trials). Order of presentation was determined by a balanced latin square with each subject seeing all conditions (visual only, visual with sound, visual with haptics, visual with sound and haptics) for a total of 4 times each (4 trials). The order of presentation is shown in Table 1, with the first participant (S_1) seeing condition A, then B, D and finally C. Gender effects were not analyzed in the study.

Table 1: Pilot study 1 experimental design

		Participants A = Control; B = Sound only; C = Haptics only; D = Sounds and Haptics															
		S_1				S_2				S_3				S_4			
	Trials	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4
Order	O_1	A_{11}	A_{12}	A_{13}	A_{14}	B_{11}	B_{12}	B_{13}	B_{14}	C_{11}	C_{12}	C_{13}	C_{14}	D_{11}	D_{12}	D_{13}	D_{14}
	O_2	B_{21}	B_{22}	B_{23}	B_{24}	C_{21}	C_{22}	C_{23}	C_{24}	D_{21}	D_{22}	D_{23}	D_{24}	A_{21}	A_{22}	A_{23}	A_{24}
	O_3	D_{31}	D_{32}	D_{33}	D_{34}	A_{31}	A_{32}	A_{33}	A_{34}	B_{31}	B_{32}	B_{33}	B_{34}	C_{31}	C_{32}	C_{33}	C_{34}
	O_4	C_{41}	C_{42}	C_{43}	C_{44}	D_{41}	D_{42}	D_{43}	D_{44}	A_{41}	A_{42}	A_{43}	A_{44}	B_{41}	B_{42}	B_{43}	B_{44}

4.3 Procedures

Participants were randomly assigned to one of the four orders of presentation prior to their arrival. Upon arrival, the participants were asked to read and sign the informed consent form (see *Appendix A: IRB Protocol and Informed Consent Form*). A pre-questionnaire was then administered to assess standard demographic information, including their previous computer and virtual environment experience (see *Appendix B: Questionnaires*). The next step was to have the participants fill out the Mental Rotations Test (MRT; Crawford and Christensen, 1995; explained more in section *4.5.1 Measures*). Following the MRT, participants were asked to interact with a demo program using the PHANToM™ for 2 minutes. The demo used the same pick and place interaction techniques as the VE. A briefing on the specifics of the task was then given (discussed in section *4.5 The Task*) and participants were brought into the virtual environment to begin. There were four conditions, one for each of the four different feedback combinations (no feedback, sound only, haptics only, sound and haptics). For each condition the participant would complete the task 4 times and at the end of the fourth trial they were asked to take off the HMD and answer rating scales on the condition they had just experienced (see section *4.5.1 Measures* for more details on this measure and *Appendix B: Questionnaires* for the actual questionnaire).

At the end of all four conditions the participants were given a subjective questionnaire including rating scales and open-ended questions comparing the conditions. The VE was designed to minimize the possibility of simulator sickness by making the participant's exposure relatively short, keeping motion to a minimum (the person's view never changed) and providing

rests after each set of four trials (the questionnaires). The time sequence of these procedures is shown below in Table 2.

Table 2: Breakdown of Experiment 1 Timing

Event	Approximate Time (minutes)
Explanations and Informed Consent	5
Pre-Questionnaire	5
Mental Rotations Test	10
Practice and Familiarization	2
Briefing	2
4 Task Iterations, 4 conditions + questionnaires	16 X 5 + 4 X 1.5 = 86
Post-Questionnaire	5
Total Participant Time	115 = 1 hour and 55 minutes

4.4 Equipment

All studies were carried out in the Virtual Environment Lab within the Human Factors Engineering and Ergonomics Center (HFEEC) of the Department of Industrial and Systems Engineering at Virginia Polytechnical Institute and State University. The virtual environment itself was run on an Intergraph Pentium II 300 MhZ with 128 Mb of RAM, a MultiRealiZm Z13-GT graphics card with 16 Mb of frame buffer memory and 32 Mb of texture memory and an on-board Crystal CS4236B 16-Bit sound controller.

The software used to develop the virtual environment was WorldToolKit release 8 (patch 5) from Sense8™, Microsoft™ Visual C++ 6.0 and Ghost SDK™ v.3.0 for the haptics. Graphics were viewed using a V8 head mounted display from Virtual Research Systems with a resolution of 640 X 480 and a 60° diagonal field of view (FOV). Sound was presented using the Sennheiser HD25 headphones on the V8. Lastly, haptic interaction was provided through the Sensable technologies PHANToM™ 1.5A Haptic Interface using the stylus and extra Gimbal attachment (so as to use the additional button provided on the stylus). The PHANToM™ (Personal Haptic Interface Mechanism) is a force-feedback device that measures the motion of the user's finger or an attached stylus in 6 degrees of freedom and applies an appropriate force in return in three degrees of freedom (no rotation). Position is accurate to 0.02 mm with the device and its low friction and inertia make it such that the motion feels realistic and unconstrained (Salisbury and Srinivasan, 1997). A peak force of 10 N can be returned by the PHANToM™,

with a maximum continuous output of 1.5 N (Massie and Salisbury, 1994). This amount of force realisitically simulates what people sense in typical manipulation tasks (Massie and Salisbury, 1994). In order to maintain smooth motion and distinct tactile sensations it is necessary to update the PHANToM™ control loop at a minimum rate of 500 Hz and it is recommended to maintain the update rate at 1 kHz (Salisbury and Srinivasan, 1997). The required update rate highly influenced the development of the virtual environment control loop, associated algorithms and models (typical visual simulations only require a 15 to 30 Hz update rates). The PHANToM, virtual environment and HMD are shown in Figure 1.

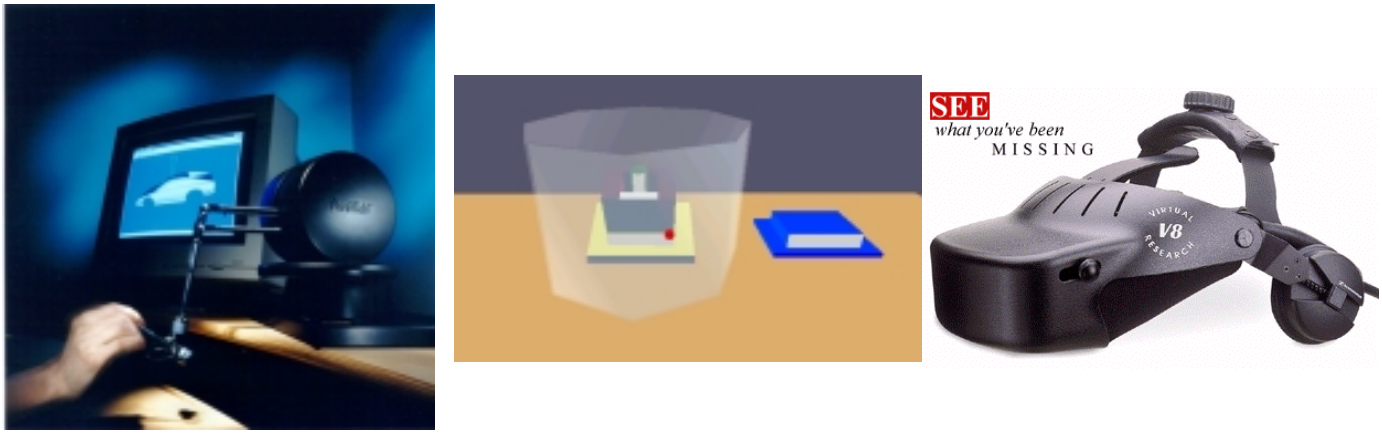


Figure 1: From left to right: PHANToM™ haptic interaction device (http://www.sensable.com/products/premium.htm), an image of the virtual environment and the head mounted display (http://www.virtualresearch.com).

4.5 The Task and Virtual Environment

The experiment task consisted of disassembling a system of parts, replacing one part and then reconstructing the system. The parts numbered seven, as seen in Figure 2, and were all of different colors. The system represented a simplified mechanical assembly similar to a real assembly illustrated in Boothroyd (1992; the purpose of the assembly shown was unknown). Since participants are manipulating three-dimensional objects rapidly and with accuracy, the task not only simulated a maintenance task but also a general VE manipulation task. Each participant was expected to complete the task with all four conditions of feedback (no extra feedback; sound; haptics; haptics and sound), four trials at each. Before beginning the task, participants were given instructions and shown a diagram of the parts (see Figure 2). They were informed that the objective of the task was to try to complete the disassembly, replacement and reassembly as quickly as possible while minimizing collisions with other parts (both were equally

important). Upon donning the HMD the user saw a workbench, the part system, the additional part used for replacement, a barrier around the part system, and a cursor used for manipulation

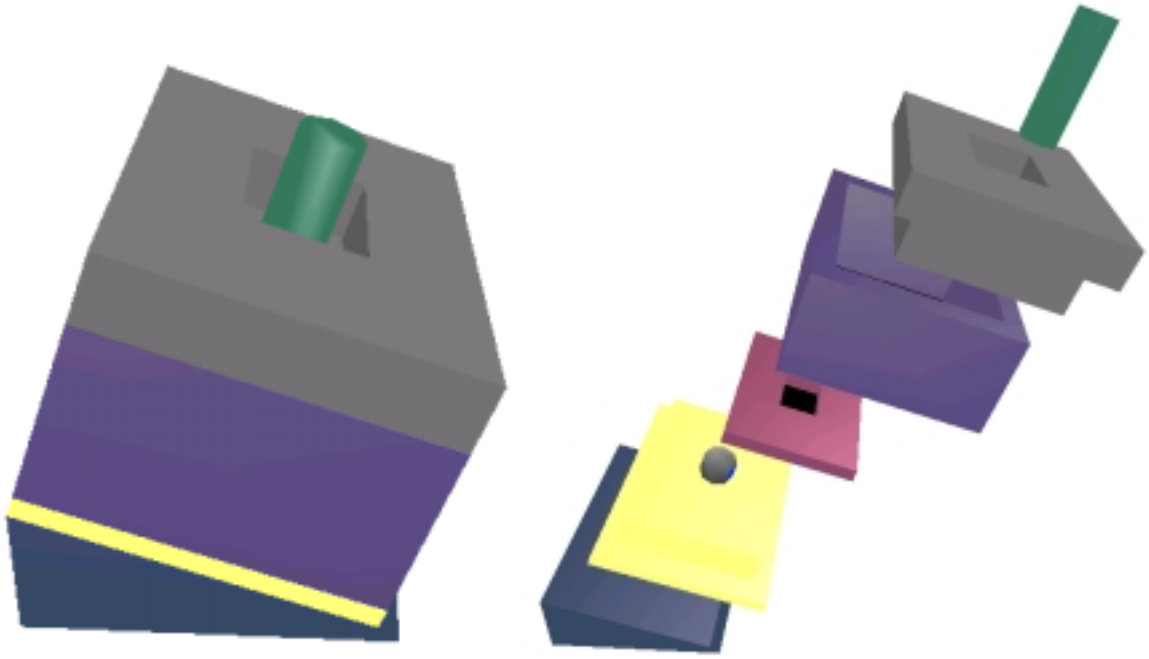


Figure 2: Part system for first pilot experiment, assembled (left) and disassembled (right).

The parts began assembled (Figure 2), resting on a workbench and surrounded by a see-through barrier to simulate that the system was part of a larger mechanical assembly. Manipulation of the components was done through the stylus attachment of the PHANToM™ device, with the movements of the stylus corresponding to the position of a virtual cursor in the VE. When the cursor touched a component, the user was able to grasp the object by pressing the button on the stylus. Keeping the button depressed while moving the cursor had the effect of dragging the part with the cursor. By letting go of the button, the part could then be dropped.

Assembly and disassembly procedures were greatly simplified for this application, as compared to real life assembly procedures. To assemble two parts together the user had only to push them together in the correct orientation and they would be connected (shown through flashing and a clicking sound if sounds were enabled, more details are given in Table 3). The same held true for disconnecting parts.

Haptic feedback in this experiment refers mainly to force feedback sensed through the end of a stylus pointing device, but with the added ability to create simulated surface friction. The auditory feedback was presented to both ears (but not localized or head tracked) through headphones and produced by prerecorded samples that were replayed through a standard 16-bit sound card. The auditory cues were used to convey similar information to the force feedback (see Table 3), but within the auditory modality. For example, picking up a part was felt by the weight added to the stylus interactions whereas auditorily the selection of an object made a clicking sound to indicate that the part was being held. When a part was dropped, the haptic modality allowed the user to feel the release of the weight and a tone was heard. Similarly, collisions between objects were felt by force feedback through the stylus and a collision sound was created (see Table 3).

Table 3: Sensory cues used for industrial part manipulation task.

Effect	Auditory Cues	Haptic Cues	Visual Cues (always)
Collision	Realistic metallic sound for each collision made.	Feel of collision in relation to the shapes of the parts (i.e. direction of force based on polygon normals).	Part visually stops upon hitting something.
Grasping	A reload sound is heard when an object is selected and let go.	The part's mass is felt through the stylus. Each part had a different simulated mass (from 200 to 700 grams).	The part flashes with a different color when selected.
Assembly or disassembly	The component makes a popping noise when connected or disconnected correctly.	None	Visually the two components blink momentarily when assembled, but not when pulled apart.
Gravity	None	A small sense of the weight of the parts was added for realism (but not full weight since the user was holding a stylus and did not have a firm grasp of the object). Simulated weights ranged from 200 to 700 grams per part (lighter than 200 grams was hard to feel at all).	See the part fall if it is released in the air.
Depth	Feedback in cases of collision.	Feedback in cases of collision.	Stereoscopy from HMD, lighting.

Because of the high computational demands of the haptic and graphical rendering, the sounds were kept non-computationally demanding (not three-dimensional or with any virtual acoustics or modeling). This was consistent with the goals of this study, to observe whether sounds that are simple and non-processor intensive can be substituted for more processor intensive haptic feedback. Thus, developers with limited computational resources may have an alternate recourse to haptics with the use of non-computationally demanding sounds. More complex and realistic sounds were beyond the scope and focus of this study. Every condition used the same stylus pointing device (PHANToM™). Using the same device for all conditions was possible since various options were created in the VE program that allowed the haptics to be turned on or off.

4.5.1 Limitations of the Virtual Environment

Although the intent was to make the virtual environment as realistic as possible, there were some limitations that could not be overcome in the required timeframe. The first was that objects (both graphically and haptically) did not slide against one another. When an object hit another object the participant would pull the first object away from the collision before continuing. A second limitation was a lack of haptic stability in some cases, exhibited as small temporary oscillations of the part as if it was on a spring (which in fact it was). Efforts were made to reduce the oscillations (and attain critical damping) as much as possible, but they still occurred with collisions at higher velocities or when a part was initially picked up. The third limitation was discovered halfway through the third experiment when a user complained of having difficulty grabbing objects in the non-haptics conditions. During these conditions the user was unable to determine when the cursor hit things since there was no sound that had been substituted for cursor collisions. Sounds should have been incorporated for this particular action to have a true sensory substitution for haptics since there was haptic feedback when the cursor hit the parts. Other than this feature all other haptic cues had related sound substitutions.

4.6 Measures

Dependent measures were those that indicated the user's ability to manipulate objects in the virtual world, the ease with which they accomplished this task and the amount of information that they felt they had gained (i.e. how useful the feedback would be in a real task). These

measures were gathered through both objective and subjective means. Objective measures included the time it took to complete each task and the number of collisions with the surrounding structure and parts. Task time was measured from the time the person grabbed the first part to the time that they reconnected the last part and was kept track of by the computer. Collisions were also tallied by the computer and were incremented each time the part being held hit another part or the surrounding barrier. This was a measure of accuracy since precision was required to avoid hitting the barrier and any other parts as well. The more times they hit parts and the barrier the less relatively accurate they were (relatively since the person must make a minimum of 5 collisions to connect the parts). These objectives measures were important as they relate to any VE manipulation task where precision and or speed are of importance. Subjective measures included 10-point rating scales to evaluate the realism, perceived task difficulty and performance, ease of learning, perceived system speed and overall reaction to the feedback received. The 10-point rating scales were based on the QUIS questionnaire (version 5.0, Chin, Diehl, and Norman, 1988). Additionally, after the user had finished all conditions they were asked to complete a set of 7-point rating scales and open-ended questions comparing the feedback cues (questionnaires shown in *Appendix B: Questionnaires*). The 7-point rating scales asked the user to compare how well the different feedback cues helped them to do the task (speed and accuracy) as well as how they foresaw these cues helping in a real design application.

A measure of spatial ability was also taken at the beginning of the experiment through a mental rotations test (MRT). Originally developed by Shepard and Metzler (1971), the MRT mainly measures mental rotation ability, but it is also hypothesized to have some relation to spatial visualization (Linn, and Peterson, 1985). Since the task being conducted used simple manipulations of three-dimensional objects, a cross between mental rotation and spatial visualization abilities, the MRT was deemed one of the more relevant tests that could be used for this study. The MRT was modified by Vandenburg and Kuse to computer format (1978) and finally modified to a paper format by Crawford and Christensen (1995). Crawford and Christensen's (1995) version was used for this experiment. Their version has a picture of a three-dimensional object on the left and four similar drawings on the right. Two of the four drawings on the right represent the same object, but rotated and the user must identify the two

correct objects for each of 20 such iterations in a limited time frame (6 minutes). An example of the figures used in the test are shown in Figure 3.

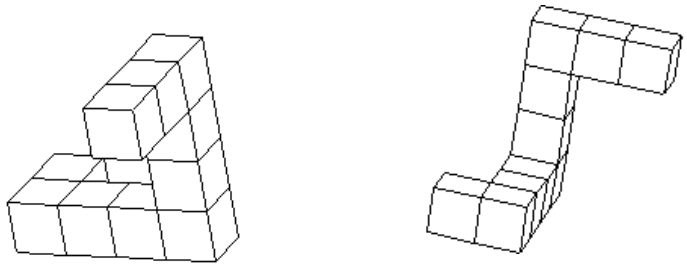


Figure 3: Example of figures in mental rotation test.
http://www.personal.psu.edu/faculty/i/b/ib4/html/mrt-ex1-p1.html

4.7 Statistical Analysis

Results from trial performances were analyzed using paired t-test comparisons. For each order of presentation, planned paired one-tailed (in the direction of the difference) t-test comparisons were made between trials 1 and 2, 2 and 3, and 3 and 4. Paired comparisons were used since specific differences between the trials were required and the statistical power of planned comparisons are greater than a one-way ANOVA followed by post-hoc tests (Huck, 2000, p.355). However, no correction for inflated type 1 errors was done since it was more important to reduce type II errors and detect differences that may exist between trial performances. The downfall of detecting a difference that did not exist (type I error) was minimal (an extra trial would get added into the design of the third study). In addition to paired comparisons, percent differences between trial totals were also calculated.

Using the fourth trial from each condition, a three-factor (sound, haptics and order) analysis of variance (ANOVA) was conducted on trial times to determine if there was an order effect or any indications of effects from sound and haptics. Results achieving a probability level (p-value) of smaller than 10% were considered significant. An alpha of 0.1 was used since this was a pilot study with few participants (an increased alpha increases power) and also because it was more important to detect any possible effects and avoid type II errors than it was to avoid type I errors. Finally, a power analysis was conducted to determine an appropriate number of subjects for the main study (Keppel, 1991, p.228). No parametric analysis for gender effects, the number of collisions or the questionnaire data was done (since the study included a sample size insignificant for such analysis).

4.8 Results

4.8.1 Number of Trials

For the first order there was a significant difference between trials 1 and 2 (μ_1 = 607.8s, μ_2 = 324.3s; $t_{(3)}$ = -2.58, p = 0.04) and trials 2 and 3 (μ_2 = 324.3s, μ_3 = 252s; $t_{(3)}$ = -2.32, p = 0.05) (Table 4 and Figure 4). Percent differences between average trial performances were also calculated and differences greater than 10% were found between trials 1 and 2 for orders 1,2, and 4; as well as between trials 2 and 3 for orders 1,3 and 4 (Table 5).

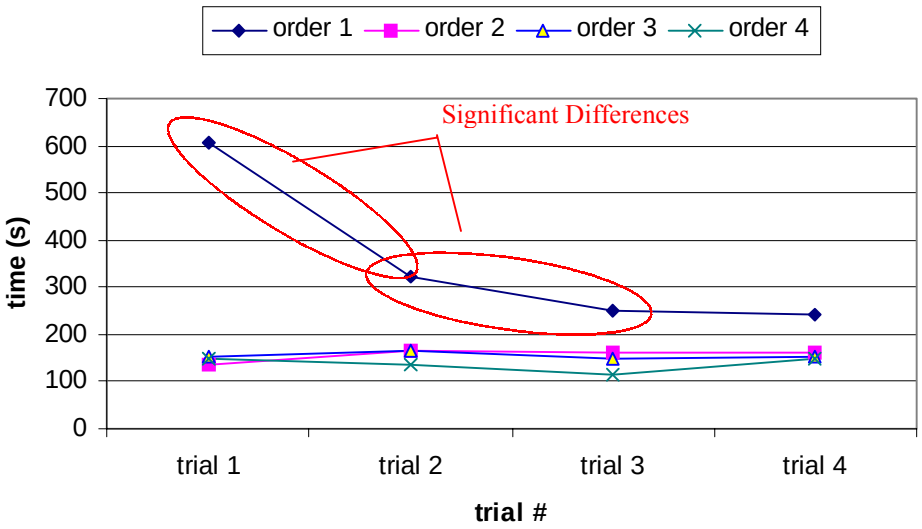


Figure 4: Time vs. trial # vs. order of presentation for first pilot study.

Table 4: Probability levels of paired t-test comparisons for first pilot study.

	Probability level for trial differences (negative indicates second trial time larger than first in the pair)		
Order	**$trial_2 - trial_1$**	**$trial_3 - t_2$**	**$trial_4 - trial_3$**
1	**0.04**	**0.05**	0.33
2	-0.12	0.39	0.49
3	-0.39	0.35	-0.38
4	0.29	0.28	-0.16

Table 5: Percent differences between trials for each order of presentation in first pilot study.

	Trial 1 / Trial 2 (%)	**Trial 2 / Trial 3 (%)**	**Trial 3 / Trial 4 (%)**
Order 1	**46**	**22**	5
Order 2	**22**	2	0
Order 3	-8	**12**	3
Order 4	**10**	**14**	-25

4.8.2 Order, Haptics and Sound

Significant effects for the analysis on time to completion were found for sound ($F_{(1,6)} = 5.87$, $p = 0.05$) and order ($F_{(3,6)} = 4.59$, $p = 0.05$) (Table 6). The addition of sound reduced performance ($\mu_s = 198.4$s, $\mu_{ns} = 145.8$s) and later orders of presentation showed increased performance (decreasing completion times) over earlier presentation orders ($\mu_1 = 240.5$s, $\mu_2 = 161.8$s, $\mu_3 = 141.3$s, $\mu_4 = 144.8$s). A Bonferonni t-Test (Dunn test) post hoc analysis on orders 1 vs. 2 and orders 1 vs. 3 showed the first and third ordered tasks to be significantly different (Table 7). Although not statistically significant, the haptics factor approached significance ($F_{(2,6)} = 3.76$, $p = 0.10$) and showed what could be a decrease in performance with the presence of haptics.

Table 6: Effect tests on completion time in study 1.

Source	DF	Sum of Squares	F Ratio	Prob > F
Sound (y/n)	1	11077.563	5.87	**0.05**
Haptics (y/n)	1	7098.063	3.76	0.10
Order	3	25941.688	4.59	**0.05**
Subject	3	59615.188	10.54	**0.01**
Sound (y/n)*Haptics (y/n)	1	8883.063	4.71	**0.07**

Table 7: Post-hoc order analysis, study 1.

	Order 1	Order 2	Order 3	Order 4
Totals (S_1 -> S_4)	962	647	565	579
Difference O_1 and O_2		315		
Difference O_1 and O_3		**398**		
Critical Bonferroni Difference		365		

4.8.3 Power Analysis

Using the variances found in study 1, a power analysis was conducted to determine an appropriate number of subjects. The method used required an F-value from the study. The F-value for the completion time interaction (sound and haptics) was chosen as interactions generally produce the largest estimate of sample size (Keppel, 1991, p.226). An alpha level of 0.05 was used for the power calculation since this is the alpha level that will be used in the third study (the study for which the calculations are being made). The results of the power analysis using the equation given by Keppel (1991, p.228) are shown in Figure 5.

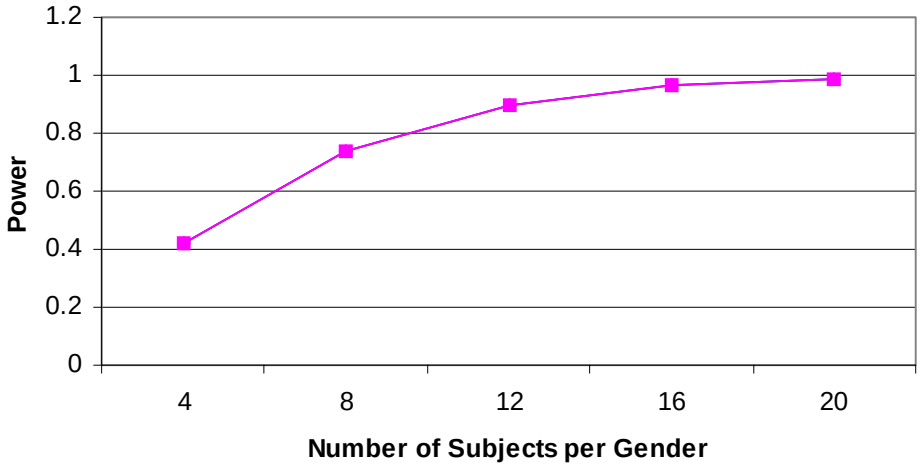


Figure 5: Power vs. number of subjects based on pilot study.

Results are shown for multiples of 4 participants since 4 x 4 balanced latin squares were required to balance order for the 4 possible feedback conditions (Sound × Haptics).

4.8.4 Observations on Task Difficulty

The computer crashed once due to the update being too slow. Moreover, multiple observations of lag in the graphical update rate were found when the participants manipulated the large gray part and the large encompassing purple part (please refer to Figure 2). Participants also encountered considerable difficulty in assembling these two parts due to the slow computer system. Other problems worthy of mention include the parts being too heavy (the weight parameter set in the software, actual weights not measured), the sounds lasting too long and being too high-pitched, and the discovery of multiple small bugs in the program (that were subsequently fixed).

4.9 Discussion

Results of the pairwise t-test comparisons on trial completion times indicated that at least 3 trials were required for the first presented feedback condition to reach a steady level of performance. For those feedback conditions received after the first, no significant improvement in completion times were observed over the trials (see Table 4 and Figure 4). However, the percent differences indicated some noticeable differences in performance between the second and third trials (Table 5). Most noticeably, the 12% and 14% differences between trials 2 and 3 on the third and fourth presented feedback conditions respectively. This data indicated that 4 trials at each feedback condition would be a safe number of trials to be certain that a steady level of performance had been attained.

The analysis for sound, haptics and order returned significant effects for sound and order (Table 6). Although the sound effect was statistically significant, the small magnitude of the difference and large standard deviations indicate that this result may not be practically significant (too few participants to be certain at this stage). In contrast, the significant order effect and subsequent post-hoc analysis (Table 7) indicate that a learning effect was occurring over the first three conditions. To reduce this effect a set of 4 practice trials were added for subsequent studies and the task was made easier (discussed more below). The practice session consisted of one trial

at each condition, presented in the same balanced order that was determined for that participant's real trials.

According to many methodologists, a reasonable and desirable level of power for studies conducted in the behavioural sciences is 0.80 (Keppel, 1991, p.75). This thesis has adopted this convention and found that a level of at least 0.8 was achieved with 12 participants for each gender (see Figure 5). Increasing the number of subjects above this level shows little predicted benefit for the extra work required.

Based on the difficulties encountered with the speed of the computer and the assembly of the purple and gray part, these two parts were removed. This reduction by itself would make the task too easy though. Therefore, one of the parts was modified to make the task more challenging. A second pilot study of three participants was conducted to verify that the level of difficulty was appropriate.

Chapter 5: Pilot Study 2

The second pilot study was run to confirm if the modified task difficulty was appropriate, whether 3 trials at each condition were now sufficient and whether the newly implemented practice session eliminated the learning effect across conditions.

5.1 Participants

Two males and one female between the ages of 20 and 30, with normal or corrected to normal vision were recruited from the Virginia Tech university student population. Participants were pre-screened just as in the first pilot study for VE experience and wrist problems. Participants were remunerated $10 for the study and nobody experienced any symptoms of simulator sickness.

5.2 Experimental Design

The experimental design was identical to the first pilot study with the exception of using 3 trials at each condition and not fully balancing presentation order (only 3 participants).

5.3 Procedures

For the second pilot study, participants were asked to complete a practice session consisting of one trial at each feedback condition, in the same order as the normal trials. This practice session was completed immediately before beginning the normal trials. Otherwise, procedures were identical to the first pilot study with the time sequence shown in Table 8.

Table 8: Breakdown of Experiment 2 Timing.

Event	Approximate Time (minutes)
Explanations and Informed Consent	5
Pre-Questionnaire	5
Mental Rotations Test	10
Practice and Familiarization	2
Briefing	2
Practice Session (4 trials)	4 x5 = 20
3 Task Iterations by 4 conditions + questionnaires	12 trials X 3 mins. + 4 questionnaires X 1.5 mins = 40
Post-Questionnaire	5
Total Participant Time	89 = 1 hour and 29 minutes

5.4 Equipment

The same equipment as the first pilot study was used.

5.5 The Task and Virtual Environment

The second experiment used a modified version of the part system with only 5 parts and is shown in Figure 6. All other features of the task and virtual environment were identical to the first pilot study except that the weights of the parts were lowered to a maximum of 400 grams.

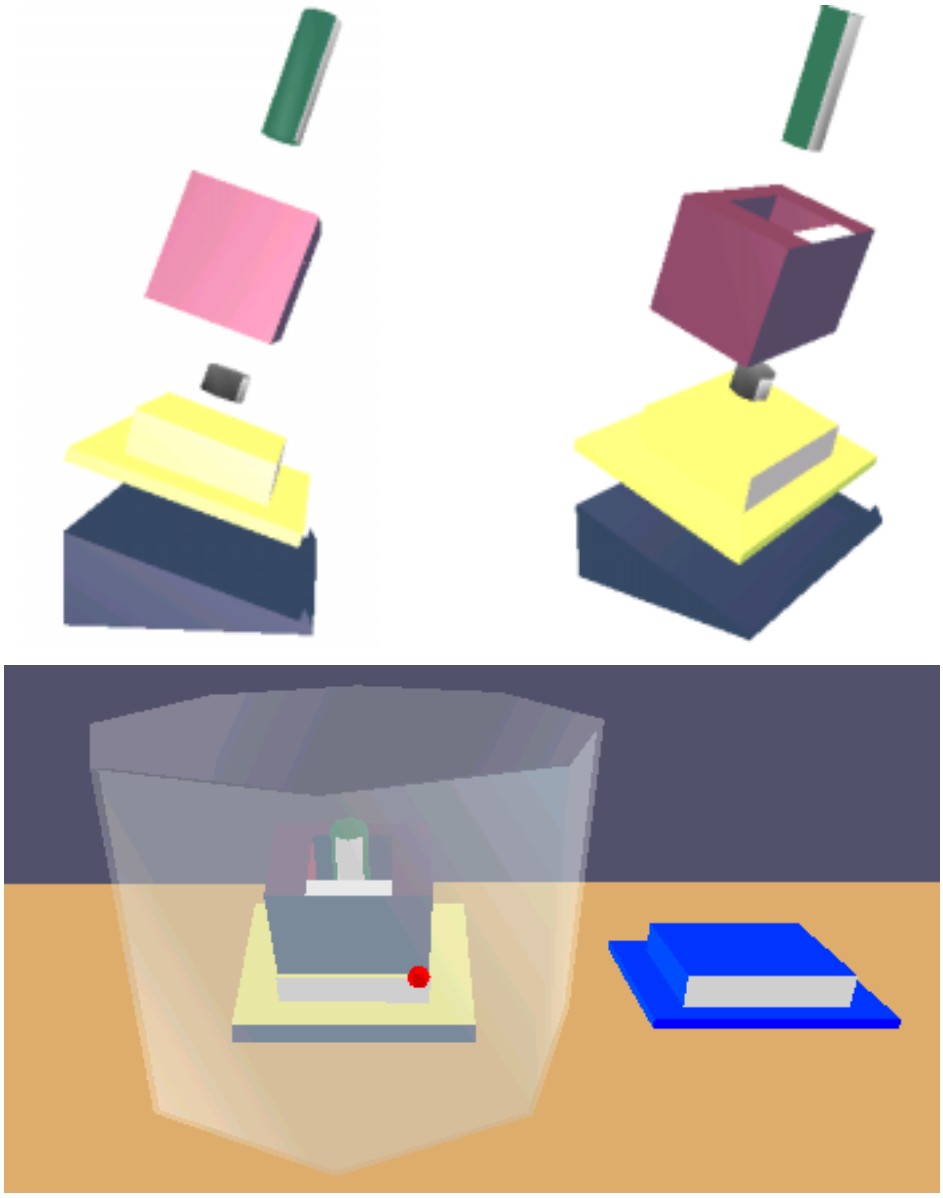


Figure 6: Part system for experiment 2 and 3, disassembled (top) and assembled in the VE (bottom).

5.6 Measures

The same dependent variables as the first pilot study were used.

5.7 Statistical Analysis

Results were analyzed using the same procedures as in the first pilot study, with the exception of the power calculation (already calculated from first pilot study).

5.8 Results

5.8.1 Number of Trials

There was a significant difference between trials 1 and 2 for order 1 ($\mu_1 = 118s$, $\mu_2 = 79s$; $t_{(2)} = -5.57$, $p = 0.02$) and for order 3 ($\mu_1 = 114.5s$, $\mu_2 = 78s$; $t_{(2)} = -4.29$, $p = 0.07$) (Figure 7, Table 9). Percent differences between average trial performances were also calculated and differences greater than 10% were found between trials 1 and 2 for the first 3 orders and between trials 2 and 3 for the second order (Table 10).

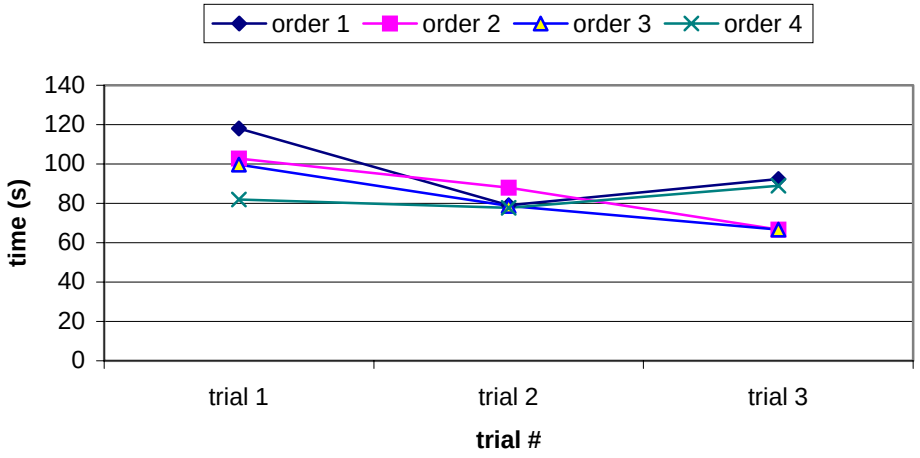


Figure 7: Time vs. trial # for each order of presentation in the second pilot study.

Table 9: Probability levels of paired t-test comparisons for second pilot study.

	Probability level for trial differences (negative indicates second trial time larger than first in the pair)	
Order	**$Trial_2 - trial_1$**	**$trial_3 - trial_2$**
1	**0.02**	-0.19
2	0.25	0.15
3	**0.07**	0.35
4	0.38	-0.15

Table 10: Percent differences between trials for each order of presentation in second pilot study.

	Trial 1 / Trial 2 (%)	**Trial 2 / Trial 3 (%)**
Order 1	**33**	-14
Order 2	**10**	**20**
Order 3	**32**	8
Order 4	5	-15

5.8.2 Order Analysis

There were no significant effects for the order analysis (Table 11).

Table 11: Order analysis on completion time for second study.

Source	**DF**	**Sum of Squares**	**F Ratio**	**Prob > F**
Sound (y/n)	1	6.0000	0.02	0.89
Haptics (y/n)	1	108.0000	0.37	0.59
Order	3	1661.0000	1.90	0.31
Subject	2	132.6667	0.23	0.81
Sound (y/n)*Haptics (y/n)	1	120.3333	0.41	0.57

5.8.3 Task Difficulty

In terms of the difficulty of the new task, it was observed that many less critical incidents occurred with the new part system, yet some manipulation ability was still required. With the new reduced part system the simulation updated faster, the computer was less prone to slowing down and no crashes were observed (neither were there any crashes for the remainder of the study). Graphical update rates between 12 to 20 frames per second were observed with the new part system whereas update rates of 6 to 15 frames per second were observed for the older part system (much slower). A study by Barfield, Baird and Bjorneseth (1998) showed that an update rate between 15 to 20 fps. (consistent with the current VE and modified part system) is sufficient to maintain performance and the feeling presence in a VE.

5.9 Discussion

The modified part system and implementation of a practice session were successful in eliminating the learning effect (see Table 11). Nevertheless, there was still an observed need to have four trials for each feedback condition as was shown from the decreasing completion times. Although these decreasing times had mostly insignificant differences with pairwise t-test comparisons (except between trials 1 and 2 on orders 1 and 3) it was still deemed a necessary precaution to add a 4^{th} trial based on the percent differences and the shape of the graph (Figure 7, Table 9 and Table 10).

It was also noticed during the second pilot study that the task itself produced irregular performance data at times. This was due to the participant experiencing difficulties with a part that they were usually able to avoid, or the opposite, they would assemble a part much quicker than they usually could. To alleviate some of this randomness, the third and fourth trials of each feedback condition were averaged together to obtain the measure of performance in the third and final study (both for time and the number of collisions).

Chapter 6: Main Study

The main study used the results from the two pilot studies to investigate the four main objectives described in Chapter 3.

6.1 Participants

The main study recruited 24 volunteers (12 males and 12 females) between the ages of 18 and 45, with normal or corrected to normal vision from Virginia Tech. Older participants were not solicited because of findings from Youngs (1999) indicating performance differences for participants over 55 using new technology. Participants were pre-screened just as in the first pilot study for VE experience and wrist problems. A remuneration of $10 was given to the participants of the study and none experienced any symptoms of simulator sickness. Unfortunately, one participant dropped out of the study part-way through due to a commitment they had forgotten about. This participant's data was not analyzed and another volunteer was recruited to maintain the required number of participants.

6.2 Experimental Design

The main experiment was a 2 x 2 x 2 mixed factorial design with gender as the between factor and haptic feedback (with or without) along with audio feedback (with or without) being the within factors. The structural model for this design is shown in the following statistical linear model:

$$Y_{ijklm} = \mu + \alpha_i + \beta_j + \delta_k + \gamma_{l(k)} + \alpha\beta_{ij} + \alpha\delta_{ik} + \beta\delta_{jk} + \alpha\gamma_{il(k)} + \beta\gamma_{jl(k)} + \alpha\beta\delta_{ijk} + \alpha\beta\gamma_{ijl(k)} + \varepsilon_{m(ijkl)}$$

Where:

Y_{ijklm} = The dependent measure of time or number of collisions for the i^{th} sound level, j^{th} haptics level, k^{th} gender, l^{th} subject and m^{th} error term.

μ = The grand mean of the treatment population.

α_I = The treatment effect for the i^{th} level of haptic feedback.

β_j = The j^{th} level of sound feedback.

δ_k = The k^{th} gender.

$\gamma_{l(k)}$ = The l^{th} subject within gender k.

$\varepsilon_{m(ijkl)}$ = The m^{th} experimental error for the i^{th} sound level, j^{th} haptics level, k^{th} gender and l^{th} subject.

For the four within feedback conditions (Sound × Haptics), the order was determined by 3 balanced latin squares, providing 12 different orders of presentation for each gender. Each participant was randomly assigned to one of the orders as shown in Table 12. Although each participant went through 4 trials at each feedback condition, only data from the third and fourth trials were used in a combined average score (based on learning curve results and data variability found in the first two studies; see sections *4.9 Discussion* and *5.9 Discussion*).

Table 12: Presentation Order and Balanced Latin Square Experimental Design.

Subjects

M = Male; F = Female; A = Control; B = Sound only; C = Haptics only; D = Sounds and Haptics

		M_1/ F_1	M_2/ F_2	M_3/ F_3	M_4/ F_4	M_5/ F_5	M_6/ F_6	M_7/ F_7	M_8/ F_8	M_9/ F_9	M_{10}/ F_{10}	M_{11}/ F_{11}	M_{12}/ F_{12}
Order	O_1	A	B	C	D	D	C	B	A	A	B	C	D
	O_2	B	C	D	A	C	B	A	D	C	D	B	A
	O_3	D	A	B	C	A	D	C	B	D	C	A	B
	O_4	C	D	A	B	B	A	D	C	B	A	D	C

Balanced Latin Squares

The haptic feedback and auditory cues were the same as the first and second pilot studies. The only exception was the hit/collision sound, which was replaced with a lower pitched sound. Gender was included as a blocking variable in this study because of previous studies showing that males have a substantial and consistent advantage in tasks requiring the manipulation of three-dimensional objects (Linn and Peterson, 1985; Vandenburg and Kuse, 1979). As such, it was theorized that there may also be gender-related differences in how subjects react (positively or negatively) to the sound and haptic cues given during the manipulation task.

6.3 Procedures

The main study was identical in procedures to the second pilot study. Please see *Appendix C: Participant Instructions and Procedures* for the exact instructions that were used. The time sequence of these procedures is shown in Table 13.

Table 13: Breakdown of Main Experiment Timing.

Event	Approximate Time (minutes)
Pre-Questionnaire	5
Explanations and Informed Consent	5
Mental Rotations Test	10
Demo	2
Briefing	2
4 Practice Trials	4 X 5 mins = 20
Task Iterations + questionnaires	16 X 3 mins + 4 X 1.5 mins = 54
Post-Questionnaire	5
Total Participant Time	103 = 1 hour and 43 minutes

6.4 Equipment

The same equipment that was used in the first and second pilot studies was used for the main study.

6.5 The Task and Virtual Environment

The part manipulation task was the same as the task used in the second pilot study.

6.6 Statistical Analysis

The results were calculated from the average of the third and fourth trials of each task (based on findings from the second pilot study, 5.9 Discussion) with average completion times analyzed using a four-factor (gender, sound, haptics, order) analysis of variance (ANOVA) and partial omega squares ($\hat{\omega}^2_{effect}$) calculated for treatment magnitude estimates (Keppel, 1991, p.432). According to Cohen (1977, p.284-288) the sizes of the $\hat{\omega}^2_{effect}$ can be interpreted as indicating small to large treatment effects (Table 14).

Table 14: Meaning of omega squared values based on Cohen, 1977.

Effect Size	Omega Squared Value ($\hat{\omega}^2_{effect}$)
Small	0.01
Medium	0.06
Large	0.15

Average collision counts were analyzed using interval nonparametric techniques (Quade test, Mann-Whitney test, Wilcoxon Signed Ranks test). Nonparametric techniques were used since the number of collisions was non-continuous and the data was also non-normally distributed. Questionnaire data was analyzed using ordinal and nominal nonparametric procedures (Wilcoxon Signed Ranks Test, Friedman two-way ANOVAs and Chi-square procedures). The mental rotations test (MRT) score was analyzed for correlation with the dependent measures. Any results achieving a probability level (p-value) smaller than 5% were reported as statistically significant.

6.7 Results

6.7.1 Completion Times

A four-way (gender, sound, haptics, order) ANOVA on average completion times (between trials 3 and 4) yielded significant results for gender ($F_{(1,63)} = 7.71$, $p = 0.01$) and an interaction between gender and haptics ($F_{(1,63)} = 4.91$, $p = 0.03$). The gender by haptic interaction was characterized by males performing worse with haptics, while females showed no significant change (Table 16, Figure 8; $\mu_{mno_haptics} = 90.4$s, $\mu_{mhaptics} = 121.4$s, $\mu_{fno_haptics} = 85.9$s, $\mu_{fhaptics} = 81.9$s, $\widehat{\omega}^2_{effect} = 0.039$). This interaction was the main cause behind the gender effect ($\mu_f = 83.9$s, $\mu_m = 105.\ 9$s, $\widehat{\omega}^2_{effect} = 0.065$) since there was no noticeable difference between genders in the non-haptics conditions ($\mu_{mno_haptics} = 90.4$s, $\mu_{fno_haptics} = 85.9$s). There was no significant effect for order ($F = 0.94$).

Table 15: Completion time and number of collision means.

Gender	Feedback Condition	Mean Completion Time (s)	Mean # of Collisions
Both	None	85.8	29
	Sound	90.5	24.1
	Haptics	94.3	50.5
	Sound and Haptics	109.0	57.2
Males	None	90.0	28.5
	Sound	90.8	25.8
	Haptics	112.5	63.1
	Sound and Haptics	130.3	70.3
Females	None	81.7	29.5
	Sound	90.2	22.4
	Haptics	76.0	38
	Sound and Haptics	87.8	44.1

Table 16: 4-way ANOVA results for completion time.

Source	DF	Sum of Squares	F Ratio	Prob > F	$\hat{\omega}^2_{effect}$
Sound (y/n)	1	2262.042	1.50	0.22	0.00
Haptics (y/n)	1	4387.510	2.92	0.09	**0.01***
Gender	1	11594.010	7.71	**0.01***	**0.03***
Order	3	4244.479	0.94	0.43	-
Subject[Gender]	22	76621.573	2.32	0.01	-
Sound (y/n)*Haptics (y/n)	1	615.094	0.41	0.52	-
Sound (y/n)*Gender	1	3.760	0.00	0.96	-
Haptics (y/n)*Gender	1	7385.042	4.91	**0.03***	**0.02***
Sound (y/n)*Haptics (y/n)*Gender	1	287.042	0.19	0.66	-

*Indicates statistically significant or at least a small magnitude effect; a dash indicates that the $\hat{\omega}^2_{effect}$ could not be calculated (F-value too small) or did not make sense to be calculated (subjects and order))

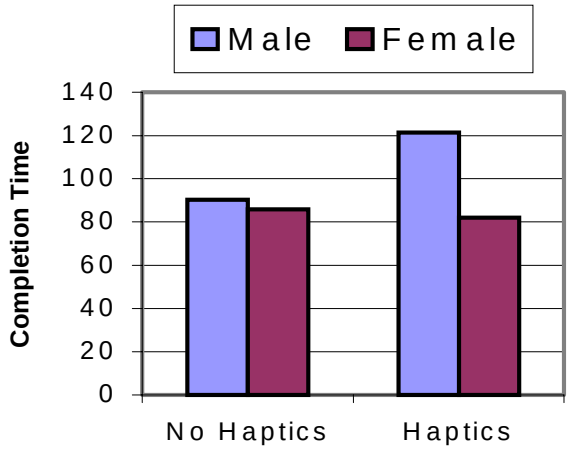


Figure 8: Gender x haptics interaction for completion time.

Due to the gender interaction, the data was also analyzed separately for each gender. As the interaction predicted, males showed a significant decrease in performance with the addition of haptics ($\mu_{mno\_haptics}$ = 90.4s, $\mu_{mhaptics}$ = 121.4s, $\hat{\omega}^2_{effect}$ = 0.08; $F_{(1,30)}$ = 5.31, p = 0.03, Table 17). Females, on the other hand, showed a significant order effect ($F_{(3,30)}$ = 3.18, p = 0.04; Table 18). A post-hoc analysis (Bonferonni t-test between orders 1&2 and orders 1&3) revealed a critical difference between orders 1 and 2 (critical difference = 24.6, actual difference = 27.4; Figure 9).

Table 17: Male 3-way ANOVA for completion time.

Source	DF	Sum of Squares	F Ratio	Prob > F	$\widehat{\omega}^2_{effect}$
Sound (y/n)	1	1040.672	0.48	0.50	-
Haptics (y/n)	1	11578.547	5.31	**0.03***	**0.08***
Order	3	7886.349	1.21	0.32	-
Subject	11	57638.182	2.40	0.03	-
Sound (y/n)*Haptics (y/n)	1	871.255	0.40	0.53	-

*Indicates statistically significant or at least a small magnitude effect; a dash indicates that the $\widehat{\omega}^2_{effect}$ could not be calculated (F-value too small) or did not make sense to be calculated (subjects and order)).

Table 18: Female 3-way ANOVA for completion time.

Source	DF	Sum of Squares	F Ratio	Prob > F	$\widehat{\omega}^2_{effect}$
Sound (y/n)	1	1225.130	1.88	0.18	**0.02***
Haptics (y/n)	1	194.005	0.30	0.59	-
Order	3	6198.891	3.18	**0.04***	-
Subject	11	18983.391	2.65	0.02	-
Sound (y/n)*Haptics (y/n)	1	30.880	0.05	0.83	-

*Indicates statistically significant or at least a small magnitude effect; a dash indicates that the $\widehat{\omega}^2_{effect}$ could not be calculated (F-value too small) or did not make sense to be calculated (subjects and order)).

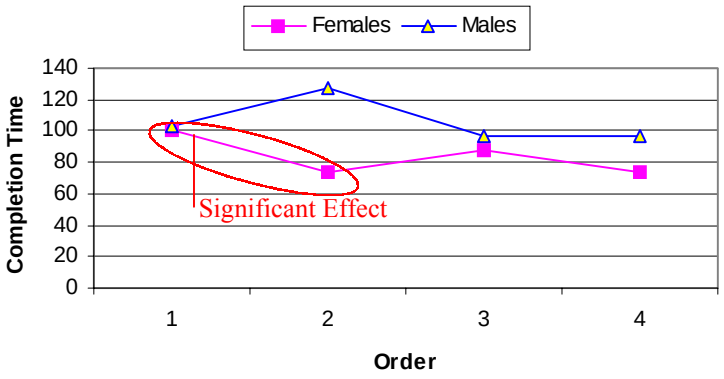


Figure 9: Order effect for females compared to none for males.

6.7.2 Number of Collisions

A Mann-Whitney test for gender, using the average of the four collision scores for each participant (there's one average score for each condition, and 4 conditions for each participant), indicated a significant effect ($T = 115.5 < w_{p(\alpha/2=0.025)} = 116$) with females experiencing less collisions on average than males ($\mu_f = 33.5$, $\mu_m = 46.9$; Table 19). Multiple Wilcoxon Signed Ranks tests showed haptics to be a significant effect for both genders ($\mu_{male\_haptics} = 66.7$,

$\mu_{male\_nohaptics} = 27.1$, $T^{+} = 78 > w_{(1-\alpha/2=0.975)} = 64$; $\mu_{female\_haptics} = 41.0$, $\mu_{female\_nohaptics} = 26.0$, $T^{+} = 72.5 > w_{(1-\alpha/2=0.975)} = 64$; Table 19, Figure 10). Interactions were investigated using a Quade test for each gender on all 4 conditions (Table 21). Significant differences were found, but post-hoc multiple comparisons revealed that the differences amounted to nothing more than the haptics effect (no interactions, Table 22, Table 23). A Quade test for order was done for both females and males and found no significant results ($T_{3\_males} = 0.16$ and $T_{3\_females} = 1.59$, both $< F_{(3,33)} = 2.9$; Table 24).

Table 19: Mann-Whitney test on gender, # of collisions.

Gender	N	Mean	T
Males	12	46.9	184.5
Females	12	33.5	116
$W_{p(\alpha=0.05)} = 116$, **p<0.05***			

Table 20: Wilcoxon Signed Ranks test for sound and haptics, # of collisions.

Feedback/Gender	Mean with Feedback	Mean without Feedback	p-value
Sound / Males	48.0	45.8	$p > 0.05$
Haptics / Males	66.7	27.1	**$p < 0.05$***
Sound / Females	33.3	33.7	$p > 0.05$
Haptics / Females	41.0	26.0	**$p < 0.05$***

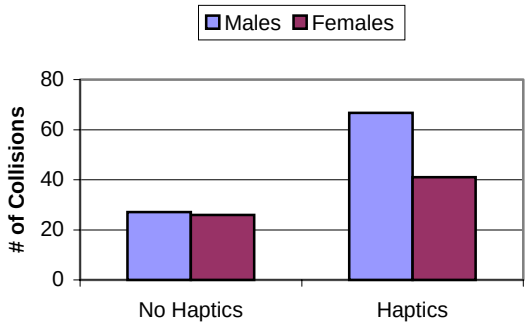


Figure 10: Haptic effect for collisions on each gender.

Table 21: Quade test for interactions on # collisions.

	Means for each Feedback condition				
Gender	**None**	**Sound**	**Haptics**	**Sound and Haptics**	**p-value**
Males	28.5	25.8	63.1	70.3	$p < 0.05$
Females	29.5	22.4	38.0	44.0	$p < 0.05$

Table 22: Post-hoc multiple comparison for males, # of collisions.

Condition	S_i	None (-64.5)	Sound (-77.5)	Haptics (77)	H. & S. (65)
None	-64.5	0	-13	**141.5***	**129.5***
Sound	-77.5	-	0	**154.5***	**142.5***
Haptics	77	-	-	0	-12
Sound and Haptics	65	-	-	-	0
Test Statistic (for 1-α/2 = 0.975) = 67.8; **bold*** indicates significant rating difference					

Table 23: Post-hoc multiple comparison for females, # of collisions.

Condition	Sum of Ranks	None (-52.75)	Sound (-65.25)	Haptics (53.5)	H. & S. (64.5)
None	-52.75	0	12.5	**106.25***	**117.25***
Sound	-65.25	-	0	**118.75***	**129.75***
Haptics	53.5	-	-	0	11
Sound and Haptics	64.5	-	-	-	0
Test Statistic (for 1-α/2 = 0.975) = 79.2; **bold*** indicates significant rating difference					

Table 24: Quade test for order on # collisions.

Factor / Gender	Mean Collisions for Level 1 / Level 2 / Level 3 / Level 4	p-value
Order / Males	42.1 / 58.3 / 39.8 / 47.4	$p > 0.05$
Order / Females	45.4 / 26.2 / 34.5 / 27.9	$p > 0.05$

6.7.3 Time and Collisions Relation

Time and collisions were related to one another as shown in the scatter plot in Figure 11.

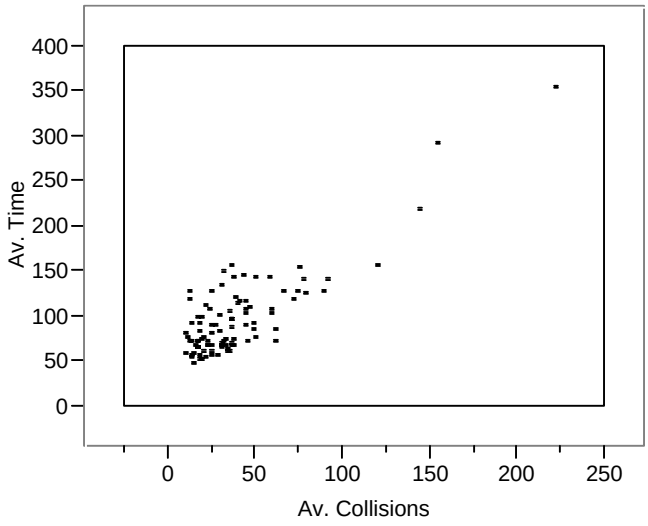


Figure 11: Scatter plot for average time versus average number of collisions.

6.7.4 Mental Rotations Test

MRT scores were affected by gender ($\chi^2 = 4.45$, $p = 0.04$) with males scoring higher than females ($\mu_f = 18.9$, $\mu_m = 22.2$, scores out of 40). No correlation was found between the MRT score and times to completion or the number of collisions (Table 25).

Table 25: Pearson Correlation of MRT with Objective Measures.

		Pearson	Probability Level
MRT	**Average Time**	0.05	0.63
	Average # of Collisions	-0.03	0.78

6.7.5 Between Condition Questionnaire Data

Data obtained for the questionnaire presented at the end of each feedback condition was analyzed for order (using Friedman two-way ANOVAs), sound and haptics factors (using Wilcoxon Signed Ranks tests) and for interactions (using Friedman two-way ANOVAs); all using an alpha level of 0.05.

The order analysis (Table 26) was conducted for both genders combined as well as each separately. One significant result was obtained for males on question 4 (learning the task), indicating that on average, the second condition presented (regardless of feedback type) was judged harder to learn ($S = 11.86$, $p = 0.01$, Table 26).

Table 26: Friedman two-way ANOVA for order on questionnaire data.

Question	Gender	p-value (Adjusted for ties)
Q1 (Overall Reaction)	Both	0.48
Q2 (Realism)	Both	0.92
Q3 (Task Difficulty)	Both	0.47
Q4 (Ease of Learning)	Both	0.23
Q5 (Update Rate)	Both	0.97
Q6 (Perceived Proficiency)	Both	0.12
Q1 (Overall Reaction)	Males	0.41
Q2 (Realism)	Males	0.43
Q3 (Task Difficulty)	Males	0.06
Q4 (Ease of Learning)	Males	**0.01***
Q5 (Update Rate)	Males	0.61
Q6 (Perceived Proficiency)	Males	0.18
Q1 (Overall Reaction)	Females	0.52
Q2 (Realism)	Females	0.41
Q3 (Task Difficulty)	Females	0.83
Q4 (Ease of Learning)	Females	0.87
Q5 (Update Rate)	Females	0.34
Q6 (Perceived Proficiency)	Females	0.21

The sound and haptics factor analysis (Table 27) was calculated for both genders combined as well as for each separately. For genders combined, significant results were found for question 1 (overall satisfaction), question 2 (realism) and question 4 (ease of learning). Question 1 showed significantly higher satisfaction ratings with haptics versus no haptics ($T^{+} = 134.5$, n=18, $w_{(1-\alpha/2=0.975)} = 130$). Question 2 showed significantly higher realism ratings for sound versus no sound AND for haptics versus no haptics ($T^{+}_{sound} = 164.5$, n=20, $w_{(1-\alpha/2=0.975)} = 157$; $T^{+}_{haptics} = 178.5$, n=20, $w_{(1-\alpha/2=0.975)} = 157$). Question 4 showed significantly higher ease of learning ratings for haptics versus no haptics ($T^{+} = 148$, n=19, $w_{(1-\alpha/2=0.975)} = 143$). There were also two results that were not statistically significant, but that are worth mentioning here ($p = 0.05$ as opposed to $p<0.05$). These were question 1 with sound showing potentially higher ratings than no sound for overall satisfaction, and question 5 with sound showing potentially higher ratings than no sound for perceived simulation update rate.

Table 27: Wilcoxon Signed Ranks test for sound and haptics, questionnaires, both genders.

Question / Factor	Mean with Feedback	Mean without Feedback	Statistics
Q1 / Sound	6.562	5.917	T+ = 156.5 (for sound), n = 20, **p = 0.05**
Q1 / Haptics	6.583	5.896	T+ = 134.5 (for haptics), n = 18, **p < 0.05***
Q2 / Sound	6.438	5.917	T+ = 164.5 (for sound), n = 20, **p < 0.05***
Q2 / Haptics	6.729	5.625	T+ = 178.5 (for haptics), n = 20, **p < 0.05***
Q3 / Sound	5.354	5.458	T+ = 131.5 (for sound), n = 21, p > 0.05
Q3 / Haptics	5.521	5.292	T+ = 137 (for haptics), n = 22, p > 0.05
Q4 / Sound	6.483	6.083	T+ = 118.5 (for sound), n = 19, p > 0.05
Q4 / Haptics	6.5	6.021	T+ = 148 (for haptics), n = 19, **p < 0.05***
Q5 / Sound	6.917	6.792	T+ = 106 (for sound), n = 16, **p = 0.05**
Q5 / Haptics	6.854	6.854	T+ = 62.5 (for haptics), n = 15, p > 0.05
Q6 / Sound	5.792	5.979	T+ = 92.5 (for sound), n = 21, p > 0.05
Q6 / Haptics	5.792	5.979	T+ = 118.5 (for haptics), n = 23, p > 0.05

For each gender separately, males showed significant results on questions 1 and 2 (Table 28), but females showed no significant results (Table 29). For question 1, male participants ranked sound significantly higher than no sound for overall satisfaction (T^{+} = 59.5, n=11, $w_{(1-\alpha/2=0.975)}$ = 55), and for question 2, males ranked sound and haptics significantly more realistic than no sound and no haptics respectively (T^{+}_{sound} = 56.5, n=11, $w_{(1-\alpha/2=0.975)}$ = 55; $T^{+}_{haptics}$ = 47.5, n=10, $w_{(1-\alpha/2=0.975)}$ = 46).

Table 28: Wilcoxon Signed Ranks test for sound and haptics, questionnaires, males.

Question / Factor	Mean with Feedback	Mean without Feedback	Statistics
Q1 / Sound	6.3	5.9	T+ = 59.5 (for sound), n = 11, **p < 0.05**
Q1 / Haptics	6.5	5.8	T+ = 44 (for haptics), n = 10, p > 0.05
Q2 / Sound	6.5	5.7	T+ = 56.5 (for sound), n = 11, **p < 0.05**
Q2 / Haptics	6.8	5.4	T+ = 47.5 (for haptics), n = 10, **p < 0.05**
Q3 / Sound	5.6	5.5	T+ = 26 (for sound), n = 9, p > 0.05
Q3 / Haptics	5.5	5.6	T+ = 23.5 (for haptics), n = 10, p > 0.05
Q4 / Sound	6.3	5.8	T+ = 33.5 (for sound), n = 10, p > 0.05
Q4 / Haptics	6.3	5.8	T+ = 44 (for haptics), n = 11, p > 0.05
Q5 / Sound	6.6	6.3	T+ = 27 (for sound), n = 8, p > 0.05
Q5 / Haptics	6.3	6.5	T+ = 13 (for haptics), n = 7, p > 0.05
Q6 / Sound	6.0	6.2	T+ = 24 (for sound), n = 10, p > 0.05
Q6 / Haptics	5.8	6.4	T+ = 20 (for haptics), n = 11, p > 0.05

Table 29: Wilcoxon Signed Ranks test for sound and haptics, questionnaires, females.

Question / Factor	Mean with Feedback	Mean without Feedback	Statistics
Q1 / Sound	6.8	6.0	T+ = 36.5 (for sound), n = 9, p > 0.05
Q1 / Haptics	6.7	6.0	T+ = 25.5 (for haptics), n = 8, p > 0.05
Q2 / Sound	6.4	6.2	T+ = 29.5 (for sound), n = 9, p > 0.05
Q2 / Haptics	6.7	5.9	T+ = 45 (for haptics), n = 10, p > 0.05
Q3 / Sound	5.1	5.5	T+ = 31 (for sound), n = 12, p > 0.05
Q3 / Haptics	5.6	5.0	T+ = 53 (for haptics), n = 12, p > 0.05
Q4 / Sound	6.6	6.3	T+ = 30 (for sound), n = 9, p > 0.05
Q4 / Haptics	6.7	6.2	T+ = 24 (for haptics), n = 8, p > 0.05
Q5 / Sound	7.3	7.3	T+ = 17 (for sound), n = 8, p > 0.05
Q5 / Haptics	7.4	7.2	T+ = 25.5 (for haptics), n = 8, p > 0.05
Q6 / Sound	5.5	5.8	T+ = 24.5 (for sound), n = 11, p > 0.05
Q6 / Haptics	5.8	5.6	T+ = 43 (for haptics), n = 12, p > 0.05

Nonparametric interactions were analyzed by looking for differences between all four conditions (using Friedman two-way ANOVAs) and then, in the presence of significant differences, analysis of multiple comparisons were completed to identify any interactions. These analyses were conducted for both genders combined as well as separate (Table 30). Significant results were found for question 2 (related to realism) for both genders combined ($S = 12.9$, $p = 0.01$; Table 30) and for just males ($S = 10.2$, $p = 0.02$; Table 30). Post-hoc multiple comparisons for genders combined and males only, indicate that both the haptics only condition and the joint haptics and sound condition were rated significantly higher for realism than the no feedback condition (Table 31, Table 32). The corresponding interaction is that the addition of haptics when sound is not present increases realism, whereas if sound is already present, the addition of haptics does not significantly increase realism (Figure 12, Figure 13). However, as can be seen by Figure 12 and Figure 13, this is not a practically significant interaction since when sound is present the addition of haptics was close ($p<0.1$) to being significantly more realistic.

Table 30: Friedman two-way ANOVA for interactions on questionnaire data.

Question	Gender	p-value (Adjusted for ties)
Q1 (Overall Reaction)	Both	0.13
Q2 (Realism)	Both	0.01
Q3 (Task Difficulty)	Both	0.71
Q4 (Ease of Learning)	Both	0.49
Q5 (Update Rate)	Both	0.33
Q6 (Perceived Proficiency)	Both	0.66
Q1 (Overall Reaction)	Males	0.38
Q2 (Realism)	Males	**0.02***
Q3 (Task Difficulty)	Males	0.98
Q4 (Ease of Learning)	Males	0.60
Q5 (Update Rate)	Males	0.30
Q6 (Perceived Proficiency)	Males	0.46
Q1 (Overall Reaction)	Females	0.37
Q2 (Realism)	Females	0.30
Q3 (Task Difficulty)	Females	0.27
Q4 (Ease of Learning)	Females	0.76
Q5 (Update Rate)	Females	0.64
Q6 (Perceived Proficiency)	Females	0.46

Table 31: Post-hoc multiple comparison for both genders, question 2.

Condition	Sum of Ranks	None (44)	Sound (58)	Haptics (67)	H. & S. (71)
None	44	0	14	**23***	**27***
Sound	58	-	0	9	13
Haptics	67	-	-	0	4
Sound and Haptics	71	-	-	-	0
Test Statistic (for 1-α/2 = 0.975) = 15.1; **bold*** indicates significant rating difference					

Table 32: Post-hoc multiple comparison for males, question 2.

Condition	Sum of Ranks	None (20)	Sound (28.5)	Haptics (33.5)	H. & S. (38)
None	20	0	8.5	**13.5***	**18***
Sound	28.5	-	0	5	9.5
Haptics	33.5	-	-	0	4.5
Sound and Haptics	38	-	-	-	0
Test Statistic (for 1-α/2 = 0.975) = 10.68; **bold*** indicates significant rating difference					

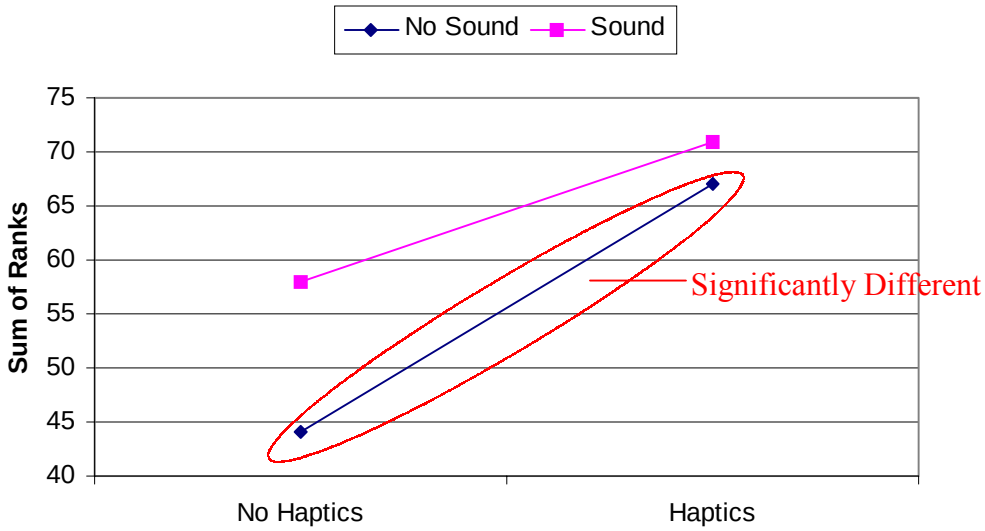


Figure 12: Question 2 (Realism), haptics x sound interaction for genders combined.

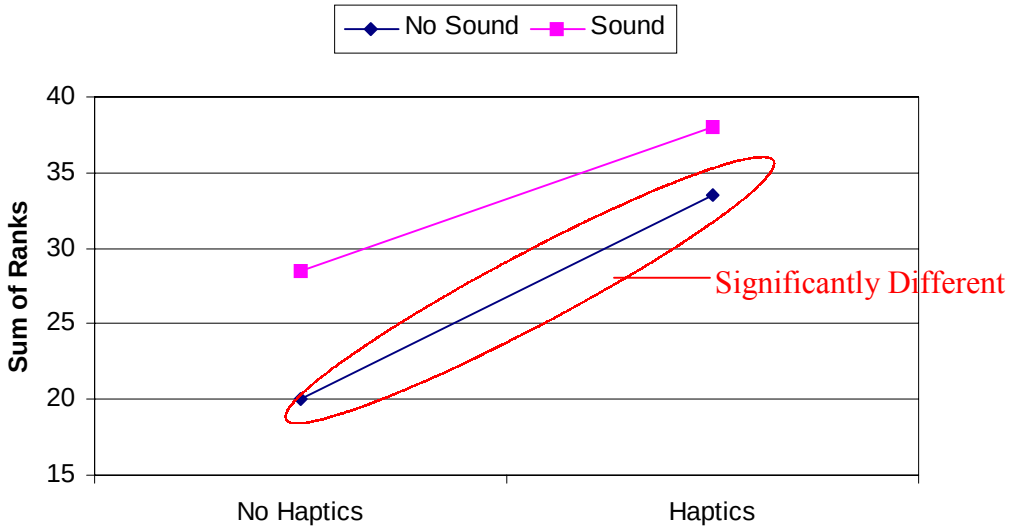


Figure 13: Question 2 (Realism), haptics x sound interaction for males.

A correlation analysis was done between question 6 (perceived performance) and the average time as well as the average number of collisions. A strong correlation was found for both measures indicating that participants were accurate in reporting how well they performed (Table 33).

Table 33: Pearson Correlation of Q6 with Objective Measures.

		Pearson	Probability Level
Q6	**Average Time**	-0.43	0.00
	Average # of Collisions	0.26	0.01

6.7.6 Post-hoc Questionnaire Data

The question asking the participants for their favorite and worst conditions was analyzed using a Chi-square test for two independent samples to compare between the genders and then a Chi-square Goodness of fit test was used to evaluate differences in frequency counts between each feedback condition. Results from the seven-point rating scales were analyzed using Friedman two-way ANOVAs for both genders combined and for each separately.

Results from the statistical analyses indicated that there was no significant difference between genders on preferred feedback condition (Table 34). Data from each gender was therefore combined for the analysis on frequency counts, but no condition was ranked significantly higher than any other as being the user's favorite or worst condition (Table 35). Nonetheless, the no feedback condition did have 11 participants rank it as the worst condition (not statistically significant at alpha of 0.05, but $p < 0.1$).

Table 34: Chi-Square test for two independent samples on gender for best and worst condition.

	Frequency Count for Best Condition		**Frequency Count for Worst Condition**	
Feedback Condition	**Females**	**Males**	**Females**	**Males**
None	3	3	6	5
Sound	3	2	4	2
Haptics	2	2	0	4
Sound and Haptics	4	5	2	1
Statistical Results	$\chi^2_{observed} = 0.31 < \chi^2_{tabled} = 7.82$		$\chi^2_{observed} = 5.09 < \chi^2_{tabled} = 7.82$	

Table 35: Chi-Square Goodness of Fit test on best and worst condition.

	Frequency Counts	
Feedback Condition	**Best**	**Worst**
None	6	11
Sound	5	6
Haptics	4	4
Sound and Haptics	9	3
Statistical Results	$\chi^2_{observed} = 2.33 < \chi^2_{tabled} = 7.82$	$\chi^2_{observed} = 6.33 < \chi^2_{tabled} = 7.82$

For the seven-point rating scales, question 1 asked the participants to rate how helpful each feedback condition was in terms of helping them to do the task quickly and accurately, while question 2 asked them how useful each feedback condition would be if the system was used to evaluate industrial designs of parts (*Appendix B: Questionnaires*). The data for both genders combined showed there to be a significant effect for both seven point rating scale questions ($S_{question\_1} = 14.55$, $p = 0.00$; $S_{question\_2} = 37.12$, $p = 0.00$, Table 36). A post-hoc multiple comparison test indicated that for question 1 there were significant differences between no feedback versus haptics, no feedback versus sound & haptics, and sound versus sound & haptics (Table 37, Figure 14). A similar test for question 2 indicated pairwise differences between all combinations except the no feedback versus sound comparison (Table 38).

For males both questions showed significant differences between their ratings ($S_{question\_1} = 9.62$, $p = 0.02$; $S_{question\_2} = 37.12$, $p = 0.00$, Table 36). Post-hoc multiple comparisons for question 1 revealed significant differences between no feedback versus sound & haptics and between sound versus sound & haptics (Table 39, Figure 15). Post-hoc multiple comparisons for question 2 revealed significant differences between all possible pairwise combinations except between no feedback and sound (Table 40).

For females, only question 2 exhibited significant results ($S_{question\_2} = 15.66$, $p = 0.00$, Table 36). A post-hoc multiple pairwise comparison revealed significant differences for no feedback versus haptics, no feedback versus sound & haptics and for sound versus sound & haptics (Table 41, Figure 16).

Table 36: Friedman two-way ANOVA for post-questionnaire data.

Question	Gender	p-value (Adjusted for ties)
Q1 (Helpfulness)	Both	**0.00***
Q2 (Hypothesized Design Utility))	Both	**0.00***
Q1 (Helpfulness)	Males	**0.02***
Q2 (Hypothesized Design Utility))	Males	**0.00***
Q1 (Helpfulness)	Females	0.11
Q2 (Hypothesized Design Utility))	Females	**0.00***

Table 37: Post-hoc multiple comparison for both genders, post question 1.

Condition	Sum of Ranks	None (43)	Sound (57)	Haptics (65)	H. & S. (75)
None	43	0	14	**22***	**32***
Sound	57	-	0	8	**18***
Haptics	65	-	-	0	10
Sound and Haptics	75	-	-	-	0
Test Statistic (for 1-α/2 = 0.975) = 15.84; **bold*** indicates significant rating difference					

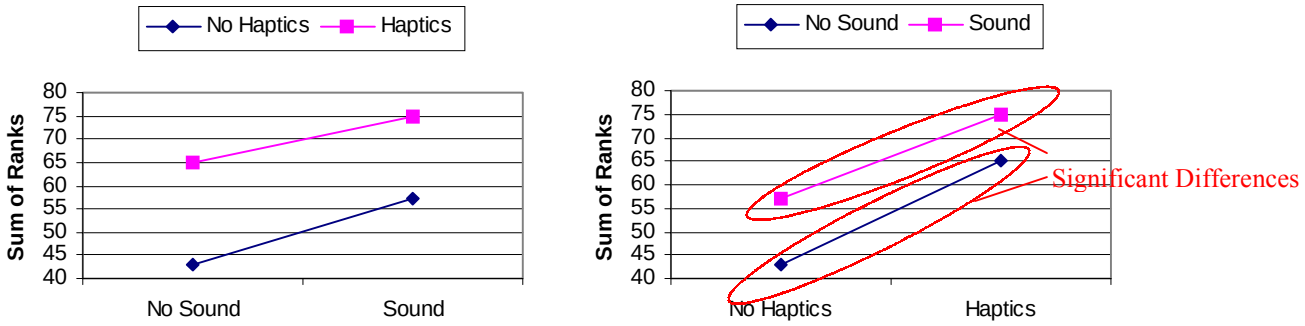


Figure 14: Interaction for post-question 1, both genders.

Table 38: Post-hoc multiple comparison for both genders, post question 2.

Condition	Sum of Ranks	None (38)	Sound (49)	Haptics (67.5)	H. & S. (85.5)
None	38	0	11	**19.5***	**47.5***
Sound	49	-	0	**18.5***	**36.5***
Haptics	67.5	-	-	0	**18***
Sound and Haptics	85.5	-	-	-	0
Test Statistic (for 1-α/2 = 0.975) = 11.95; **bold*** indicates significant rating difference					

Table 39: Post-hoc multiple comparison for males, post question 1.

Condition	Sum of Ranks	None (21.5)	Sound (27.5)	Haptics (31)	H. & S. (40)
None	21.5	0	6	9.5	**18.5***
Sound	27.5	-	0	3.5	**12.5***
Haptics	31	-	-	0	9
Sound and Haptics	40	-	-	-	0
Test Statistic (for 1-α/2 = 0.975) = 11.16; **bold*** indicates significant rating difference					

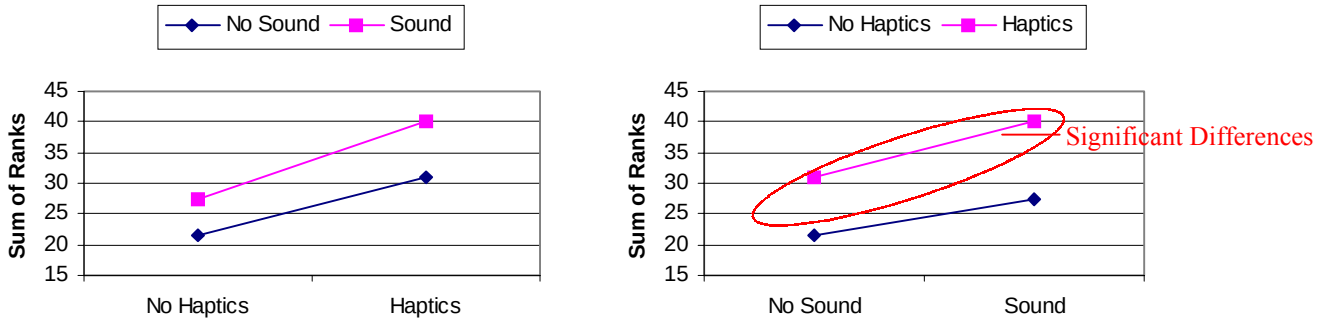


Figure 15: Interaction for post question 1, males.

Table 40: Post-hoc multiple comparison for males, post question 2.

Condition	Sum of Ranks	None (18.5)	Sound (22.5)	Haptics (35)	H. & S. (44)
None	18.5	0	4	**16.5***	**25.5***
Sound	22.5	-	0	**12.5***	**21.5***
Haptics	35	-	-	0	**9***
Sound and Haptics	44	-	-	-	0
Test Statistic (for 1-α/2 = 0.975) = 8.14; **bold*** indicates significant rating difference					

Table 41: Post-hoc multiple comparison for females, post question 2.

Condition	Sum of Ranks	None (19.5)	Sound (26.5)	Haptics (32.5)	H. & S. (41.5)
None	19.5	0	7	**13***	**22***
Sound	26.5	-	0	6	**15***
Haptics	32.5	-	-	0	9
Sound and Haptics	41.5	-	-	-	0
Test Statistic (for 1-α/2 = 0.975) = 9.26; **bold*** indicates significant rating difference					

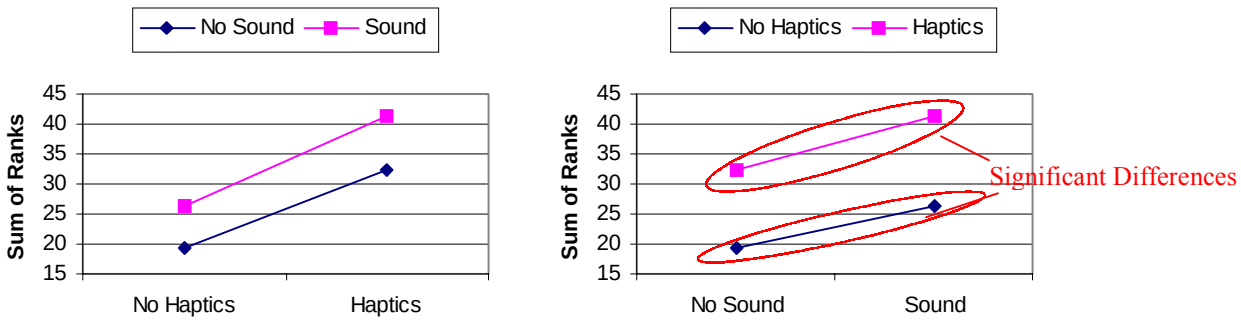


Figure 16: Interaction for post question 2, females.

User comments for each feedback condition are presented in *Appendix D: User Comments for each Feedback Condition.*

6.8 Discussion

The research described in this thesis was conducted with four objectives in mind (3.2 Research Objectives). The first was to investigate whether haptics increased performance and ease of interaction in a complex VE manipulation task. Results showed that haptics actually decreased performance and that there was no indicated change in ease of use (question 3 of rating scales). Males made more collisions with objects (an average of 2.5 times more) and took longer to complete the task (33% longer on average). Females also exhibited an increase in the number of collisions (1.5 times more on average), but their completion times showed no significant change. The increased completion time for males had a calculated $\hat{\omega}^2_{effect}$ value of 0.08, indicating a medium sized effect due to the haptics.

These results refute the hypotheses set forth earlier in the thesis predicting better ease of use and decreased completion times in the presence of haptics. These hypotheses however, were based on previous studies where it was assumed that results indicating increased accuracy from haptics on simple manipulation tasks (Richard and Coiffet, 1995; Ishi and Sato, 1994) would outweigh the decrease in performance of haptics in constrained insertion maneuvers (Massimino and Sheridan, 1993). This was clearly not the case in this experiment, where any increase in movement accuracy must have been offset by the impeded movements encountered during the

many constrained maneuvers. Observations from the experimenter and the participants clearly indicated that movements were especially impeded when inserting parts into small enclosed spaces (as predicted by Massimino and Sheridan, 1993). Although the occasional lack of stability in the haptic feedback also increased the difficulty level, it is hard to justify that this alone could have caused the observed decrease in performance (on average 31 seconds slower for males, 90s versus 121s). One possible explanation is that the simulation was not realistic enough to benefit from haptics since the VE also lacked the ability to provide sliding (for conditions with and without haptics). Getting stuck visually on something may not have been as performance diminishing as being stuck haptically on something. If this were the case, then there may be some minimal level of realism required before haptics begins to increase performance. Further research on this topic is required to determine whether this hypothesis is correct.

The second research question was to examine whether haptic cues made the interactions more realistic and could potentially allow users to gain more benefit from the simulation. The second rating scale question addressed realism and the second question in the post-questionnaire addressed the hypothesized utility of the feedback towards an industrial design task. Results indicated that males felt an increased sense of realism with the addition of haptics (1.4 rankings higher on average out of 10), while females indicated no statistically significant preference. Nonetheless, females did exhibit a tendency to rank the inclusion of haptics higher for realism (0.9 rankings higher out of 10), with a probability level between 0.05 and 0.1. As the results indicate however (0.9 to 1.4 rankings difference), neither gender ranked the haptics condition a lot higher for realism, indicating only a small preference for haptics. For hypothesized utility, participants ranked the haptic conditions higher than the non-haptic conditions (1.5 rankings higher on average out of 7), indicating that after having used the system with and without haptics almost all participants believed that haptics would be useful in a real design application. In terms of the strength of the result, a difference of 1.5 ranks out of 7 indicates more than a small preference, but not a large difference. These results agree with the hypotheses set forth.

The third question was to investigate if auditory cues provided an effective substitute for haptics. Based on objective performance measures, sound did not have the negative effects that

haptics did; sound neither increased or decreased completion time or the number of collisions. Realism, on the other hand, was improved by the addition of sound (ratings were on average 0.5 rankings higher for sound versus no sound) and found to be an equivalent substitute for haptics. For overall satisfaction, even though sound was not rated significantly higher than no sound, the p-value of the result was close to being significant ($p = 0.05$ as opposed to $p<0.05$) and will be considered here. Haptics and sound received the same average ratings for overall satisfaction as well (both approximately 0.6 rankings higher than their respective no feedback conditions), and for this reason the two will be considered equivalent in user satisfaction. However, when participants were asked to rate the conditions that helped them to do the task and those that would help in a real task, sound was ranked between haptics and no feedback, with haptics ranked higher and no feedback showing a trend of being ranked lower than sound ($0.1 > p > 0.05$).

These results confirm the hypotheses set forth earlier, which assumed that sound would provide better usability than no feedback but not as good as haptics. Nevertheless, sound may be a more effective means of conveying haptic information, since sound does not negatively affect manipulation performance while still slightly increasing user perceived realism and overall satisfaction. A downside, however, is that sound is not hypothesized to be as helpful as haptics in a design scenario. This can be explained by user comments which clearly indicate that they believed the haptics condition presented more potentially useful information than sound. Examples of such comments are given below:

- "Force told you more specifically what you were doing wrong"(than sound).
- "Sound enhanced realism, but was not very helpful."
- "Sound on the other hand was almost completely irrelevant when present in the absence of touch. It was quite distracting when I was trying to accomplish the task."
- "Sound worked well when I also had the force feedback, alone it was just annoying."
- (Sound) "Not that helpful just neat to be able to hear."

The fourth question was to explore whether the combination of sound and haptics led to increased user satisfaction or performance. Results indicated no interactions for the two factors with completion time, the number of collisions, nor any of the rating scale questions except the

question related to usefulness towards a real industrial task. For the latter question, participants rated the combination of sound and haptics higher than all other conditions (on average one ranking above haptics, which was the next highest ranked condition). Consequently, participants felt the two feedback conditions together would be the most useful condition if this system were used in a real design task. The reason that no other interactions were found was mainly due to haptics not improving performance. It should be noted as well that some participants remarked that the two feedback modalities together did augment the level of stress they felt while completing the task, while others felt the opposite. These comments indicate individual differences in user preferences for the amount of feedback that should be presented.

The generalizability of these results are limited to applications using desktop haptic manipulators and for which complex manipulations are involved (a combination of pick and place and insertion type tasks). The subjective user opinions can be applied to all these applications, however, the objective results can only be applied to a subset of these applications that depend on speed and accuracy. Applications that meet these criteria include industrial part design tasks, industrial maintenance simulations, telerobotic control through VE interfaces, three-dimensional data manipulation and visualization (where accuracy of movement may be important) and any other task involving complex manipulations. The relevance of subjective versus objective measures depends on the application and how important performance is versus user satisfaction and comfort. If the application is seldom used and time to completion is important, than the objective measures may be more applicable to that situation. On the other hand, if the simulation will be used for extended periods of time and completion time is not the most important measure of success, then user comfort and satisfaction may be more important while doing the task. Other factors concerning the limitations and generalizability of this study are addressed in section *6.2 Future Research*.

6.8.1 Gender Differences

The gender difference, indicating that females performed better with haptics than males, was unexpected. Previous studies in spatial ability have shown males to score higher in spatial ability (Crawford and Christense, 1995; Linn and Petersen, 1985; Vandenburg and Kuse, 1979). This

experiment was no exception, with males scoring significantly higher on a mental rotations test (MRT) (an average score of 22.2 for males versus 18.9 for females). However, there was no correlation between the MRT score and performance. In fact, manipulation ability appeared stronger for females than males with haptics, shown by significantly lower completion times and fewer collisions. Studies showing females to be superior in fine motor skill were supposedly due to the small size of the objects being manipulated and the smaller hand size of females according to Peters, Servos and Day (1998). However, in this experiment, the stylus interaction on the PHANToM was 9.5 mm in diameter. This diameter was supposedly big enough to reduce any hand size differences in manipulation ability (Peters, Servos and Day, 1998; Kilshaw and Annett, 1983). It may be then that there is still reason to believe that females do truly have superior fine motor skill abilities, which exhibited themselves when haptics were present in this study. It would seem that females were able to utilize the increased accuracy provided by haptics more effectively than males, allowing the former to balance out the increased number of collisions that haptics contributed to. Based purely on informal observations it did seem that females were more precise in their movements, shown by the decreased number of collisions.

In case these gender differences were actually the manifestation of individual differences not related to gender, times to completion were plotted for each subject as shown in Figure 17 (collision results were very similar). From the figure we can see that males varied more in their performance than females did, but that in general, females outperformed males. Even if the two suspect data points in the sound and collisions combination are removed (shown circled in Figure 17), the data still indicates that males performed worse than females ($F_{(1,61)} = 5.69$, $p = 0.02$).

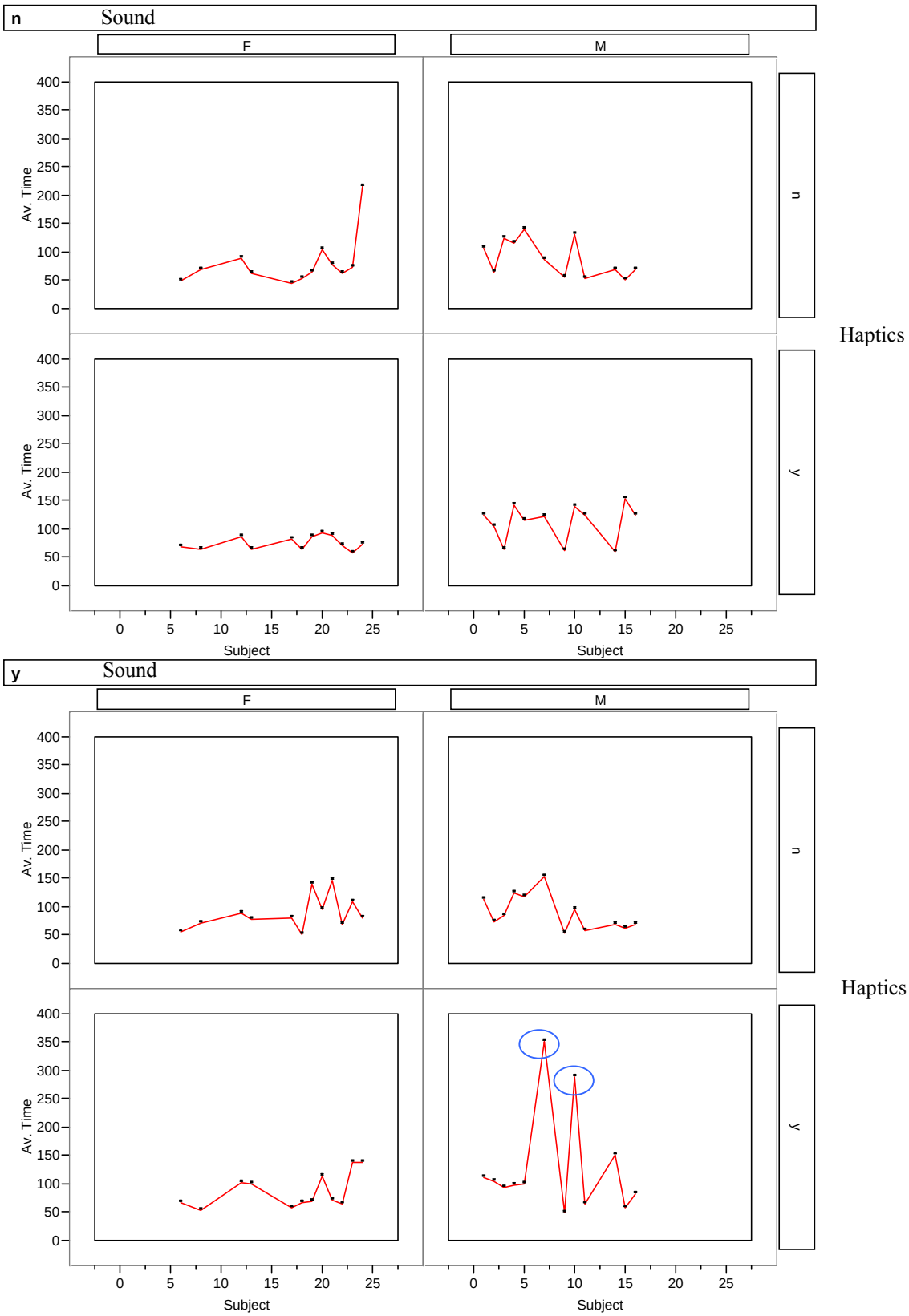


Figure 17: Individual Differences for time to completion in each feedback combination.

In addition to the gender by haptic interaction there was also a gender learning difference that should be mentioned here. Even with the practice session, a learning effect was found for females between the first and second condition (a decrease of 27% in completion time). Males actually showed performance that was quite the opposite as can be seen in Figure 9 (and discussed in the next paragraph). Since a balanced latin square was repeatedly used for each gender this averaged out any of the effects this would have had on the results for any particular feedback condition. Nevertheless, the result does suggest that females were able to learn more from the task and improve their performance more over time than males could (both genders started out approximately equivalent in performance on the first condition, Figure 9), indicating that females would have benefited from additional practice before the task began. Once again this may lend credibility to females having better fine motor skills despite the findings by Peters, Servos and Day (1998).

Male performance on the second condition exhibited the opposite reaction (an increase in completion time by 25% shown in Figure 9). Although this was not a statistically significant difference, males did consistently rate the second condition as significantly harder to learn (no matter what feedback was given, Table 26). Observations of the male participants seemed to indicate that on the second condition, many male participants would begin encountering difficulties in the angle or placement of components in the assembly and repeatedly get stuck on those difficulties. By the fourth trial at the second condition they would have solved how to get around these difficulties (e.g. they might determine the exact angle that two components connect at) and their performance would improve for the third and fourth conditions. Females did not tend to encounter this difficulty.

The exact reason behind these gender differences has long been a subject of debate. One argument is that the differences are based on hormonal levels and neurological differences in right versus left hemisphere dominance between the sexes (the nature argument). The other main argument is that each gender is subject to different environmental factors while maturing (the nurture argument). Traditionally, males have been encouraged to do things involving strength and speed while females have been encouraged to follow more artistic endeavors

involving fine motor skills. The research in this field is still far from coming to a consensus, especially with these environmental factors now beginning to shift (both male and female participation are now encouraged in both types of domains). Although this research sheds no new light on the reasons behind the dissimilarities, it does add another documented task with gender differences that may be subject to further studies to increase our understanding.

Chapter 7: Conclusions

The overall purpose of this research was to investigate whether or not haptics or sound substitutions improved performance or usability in VE applications requiring complex manipulation and where haptics are not essential (i.e. not surgical simulators were haptics are essential for training motor skills). Based on the results of this research, it is recommended that sound substitution be used for these types of VEs since it slightly improves the perceived realism of the application and overall satisfaction while not deterring from manipulation performance as haptics did in this study. Nonetheless, haptics may still be a useful and required feedback cue in a VE design or visualization application where thought provocation and understanding are the main goals (and manipulation completion time is not the main objective). Haptics were **hypothesized** by the users to be more useful than sound in these types of applications, with the combination of the two being ideal if resources permitted. Further research is required to determine whether haptics actually improves the utility of design or visualization VEs. The rest of this chapter expands on these conclusions by providing more detailed design guidelines for the VE designer and additional research questions that emerged from this thesis.

7.1 Design Guidelines

The following design guidelines have been compiled from the results of this research to aid designers in choosing whether or not to use haptics or sound substitutions in their applications (Table 42).

Table 42: Design Guidelines

Design Guidelines based on Objective measures and questionnaire rating scales
If minimizing completion time and/or the number of collisions are the only goals in a similar VE manipulation task then haptics should not be incorporated (it does not improve performance).
For small increases in realism and user satisfaction either sound or haptics could be used (both showed equivalent increases in the two measures). Sound, however, will not deter from manipulation performance if this is an objective of the VE.
To increase the user's perception of helpfulness and the utility of the feedback provided in the application, haptics should be incorporated (sound did not increase either perceived helpfulness or utility as significantly as haptics).
Using a combination of sound and haptics may improve the utility of a VE for design or visualization tasks (as predicted by participants), but the combination does not improve manipulation performance or the usability of the application.
If minimizing time to completion and the number of collisions are essential, and haptic cues are being employed, then the selection of female users would stop performance from deteriorating (results showed females performed better with haptics than males, Figure 8).
Design Guidelines based on user comments and experimenter observation
If haptic feedback is incorporated it should be very realistic, otherwise it deters from performance and from the realism of the simulation and frustrates people.
When using limited computer power, using less graphics and haptics in order to make the simulation more realistic is better than attempting to incorporate more haptics and graphics at the cost of performance (based on pilot study).
If neither sound or haptics can be incorporated, visual enhancements (e.g. flashing of parts when connected) and substitutions (visual collision detection) are appreciated by users (based on user comments in the no feedback condition).

7.2 Future Research

Previous studies have investigated the use of haptics with very simple pick and place tasks and peg-in-a-hole insertion tasks. This study examined the effects of haptics in a more complex assembly-type manipulation task and has also included measures of usability for the different feedback conditions. The research was kept general and fundamental purposefully, so as to apply to as many applications as possible. As such, it is left to the designer to determine if the recommendations brought forth from this research apply to their specific situation. The next step is to apply the findings of this research to the creation of future VE applications and to determine how haptics affects performance in specific design and visualization tasks. One such study, that has already been completed, is the GROPE project (Brooks et. al., 1990), which demonstrated that haptics increased the user's ability to dock molecules and understand the reactions occurring. It is these types of applications, relying on an increased sense of understanding (as

opposed to completion times or the number of errors), that may benefit from the increased realism or extra sensory information that haptics can provide. More research is required though to determine if other types of design and visualization applications benefit from force feedback and if so, how the feedback should be presented to maximize its utility.

Additional research into different devices should also be investigated. As this study used one particular device (a PHANToM™), it is difficult to generalize the results to all devices. More studies like this with other devices are required to obtain a more thorough understanding of the effects of haptics. Another limitation of this study was the reduced fidelity of the haptics due to small haptic instabilities, the inability to slide, and, at times, a reduced update rate. These problems were caused by insufficient computing power and inadequate development tools for the incorporation of haptics. In this case, because a graphical VE creation tool could not be found that would integrate with the haptics, the application had to be programmed using C++ and various libraries (WTK and GHOST). Not only was the programming involved difficult and time consuming, but there were many problems encountered in the creation of realistic 3-d haptic and graphical collision detection that had to be solved (i.e. there was a severe lack of useful functions available for accurate inter-object collision detection in GHOST and WTK). There is a great need for easy to use development tools for the creation of VEs that incorporate haptics. These applications should allow researchers that know some programming basics (i.e. they should not have to be a computer science major), to be able to create a realistic application on a desktop computer.

Another fruitful area for research, that was uncovered during this thesis, is the investigation of gender differences with haptics and VEs in general. Some surprising gender differences both in terms of performance as well as subjective opinions on the usability of haptic feedback were discovered in this thesis. Additional research into the reasons behind these gender differences should be conducted so as to determine whether different options should be incorporated into VEs based on the gender of the user. And if so, what these options should be.

Finally, although it was observed in this study that haptics and sound did not increase performance on the manipulation task, this does not mean that an alternative method of

presenting haptics and sound will not aid performance. Alternative methods could include more fidelity in the sounds without any redundancy between haptics and sound or perhaps only a subset of selected feedback cues from each modality. It is also possible that haptics did not increase performance in this study because the simulation fidelity was not high enough. Simulations with more fidelity should also be studied to determine if a certain level of fidelity in the task is required before haptics benefits performance.

References

Acosta, E., Stephens, B., Temkin, B., Krummel, T.M., Gorman, P.J., Griswold, J.A. and Deeb, S.A. (1999) Development of a haptic virtual environment. Proceedings-of-the-IEEE-Symposium-on-Computer-Based-Medical-Systems. 1999, p 35-39.

Anderson, D.B. and Casey, M.A. (1997). The sound dimension. IEEE-Spectrum. v 34 n 3 Mar 1997, p 46-50.

Asamura, N. Tomori, N. and Shinoda, H. (1998) A Tactile Display based on Selective Stimulation to Skin Receptors. In 1998 IEEE Annual Virtual Reality International Symposium, (pp.36-42). Atlanta, GA: IEEE.

Bach-y-Rita, P. Webster, J.G., Tompkins, W.J. and Crabb, T. Sensory substitution for space gloves and for space robots. Proceedings of Workshop on Space Telerobotics, Vol.2, pp.51-57, 1987. As referred to by Richard and Coiffet, 1995.

Baier, H., Buss, M., Freyberger, F., Hoogen, J., Kammermeier, P. and Schmidt, G. (1999). Distributed PC-based haptic, visual and acoustic telepresence system – Experiments in virtual and remote environments. Proceedings of the IEEE Virtual-Reality-Annual-International-Symposium, VR'99. Houston, TX, USA. 1999, p 118-125.

Bakker, N., Werkhoven, P., Passenier, P. (1998). Aiding orientation performance in virtual environments with proprioceptive feedback. Proceedings of the IEEE Virtual Reality Annual International Symposium, IEEE Computer Society, Los Alamitos, CA, USA, 28-33. As referred to by Nash, Edwards, Thompson and Barfield (2000).

Barfield, W., Hendrix, C. Bjorneseth, O., Kaczmarek, K.A. and Lotens, W. (1995). Comparison of human sensory capabilities with technical specifications of virtual environment equipment. Presence: Teleoperators and Virtual Environments. Vol. 4(4), pp.329-356.

Barfield, W., Baird, K.M. and Bjorneseth, O.J. (1998). Presence in virtual environments as a function of type of input device and display update rate. Displays. Vol. 19, pp. 91-98.

Begault, D.R. (1994). 3-D sound for virtual reality and multimedia. AP Professional: Boston.

Boothroyd, G. (1992). Assembly automation and product design. NewYork: Marcel Dekker, Inc.

Brooks, F.P. (1999). What's real about virtual reality. Proceedings of IEEE Virtual Reality. Houston, Texas, March 13-17, 1999. pp.2-3.

Brooks, F.P. Jr.; Ming-Ouh Y., Batter, JJ. and Kilpatrick, J.P. (1990). Project GROPE. Haptic displays for scientific visualization. Computer-Graphics-(ACM). v 24 n 4 Aug 1990, Publ by ACM, New York, NY, USA. pp. 177-185.

Bryson, S. (1996). Virtual Reality in Scientific Visualization. Communications of the ACM. May 1996, Vol. 39(5), pp. 62-71.

Burdea, G. (1996). Force and touch feedback for virtual reality. New York: Wiley.

Burdea, G., Richard, P. and Coiffet, P. (1996). Multimodal virtual reality: Input-output devices, system integration, and human factors. International Journal of Human-Computer Interaction. Vol. 8(1), 1996, pp. 5-24.

Buttolo, P., Kung, D. and Hannaford, B. (1995). Manipulation in real, virtual and remote environments. Proceedings of the IEEE conference on System, Man and Cybernetics. Vol. 5, October 1995, pp.4656-61. Vancouver, BC, Canada.

Chin, J.P., Diehl, V.A. and Norman, K.L. (1988). Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface. Proceedings of ACM CHI'88 Conference on Human Factors in Computing Systems, p.213-218. As referred to by Williges, 1998, 392-396.

Cohen, J. (1977). Statiscal power analysis for the behavioral sciences (rev. ed.). New York: Academic Press. As referred to by Keppel, 1991, p.66.

Crawford, H.J. and Christensen, L. (1995). Developing research skills: A laboratory manual (3rd ed.). Boston: Allyn and Bacon.

DiFranco, D.E., Beauregard, G.L. and Srinivasan, M.A. (1997). The effect of auditory cues on the haptic perception of stiffness in virtual environments. ASME Dynamic Systems and Control Division American Society of Mechanical Engineers, v 61, 1997, ASME, Fairfield, NJ, USA. p. 17-22.

Durbeck, L.J.K., Macias, N.J., Weinstein, D.M., Johnson, C.R. and Hollerbach, J.M. (1998). SCIRun Haptic Display for Scientific Visualization. Proceedings of the Third PHANTOM Users Group Workshop. AI Lab Technical Report No. 1643 and RLE Technical Report No. 624, MIT, December 1998. http://www.sensable.com/community/PUG98/papers.htm. Sensable Technologies. Viewed on January 15, 2000. Last updated 01/07/00.

Edwards, G., Thompson, J. and MacGregor, C. (1998). Investigating the role of guided tours in virtual reality environments. Proceedings of the Human Factors Association of Canada, Missausauga, Ontario, v 1, 1-4.

Ellis, R.E., Ismaeil, O.M. and Lipsett, M.G. (1996). Design and Evaluation of a High-performance Haptic Interface. Robotica. Vol. 14, no. 3, pp. 321-327.

Fisher, S. (1990). Virtual Interface Environments. In B. Laurel (Ed.), The art of human computer interface design. (pp. 423-438). Reading, MA: Addison-Wesley.

Fritz, J.P. and Barner, K.E.. (1999). Design of a haptic data visualization system for people with visual impairments. IEEE Transactions on Rehabilitation Engineering, vol.7, no.3, Sept. 1999, pp.372-84. Publisher: IEEE, USA

Goertz, R.C. and Thompson, R.C. (1954). Electronically controlled manipulator. Nucleonics, pp.46-47. As referred to by Ellis, Ismaeil and Lipsett, 1996.

Gomez, D., Burdea, G., Lagrana, N. (1995). Integration of the Rutgers Master II in a virtual reality simulation. Proceedings of Virtual Reality Annual International Symposium '95. IEEE Computer Society Press, Los Alamitos, CA, pp. 198-202.

Grabowski, N.A. and Barner, K.E. (1998). Data visualization methods for the blind using force feedback and sonification. SPIE Conference on Telemanipulator and Telepresence technologies V. Boston, Massachusetts, November 1998. SPIE vol. 3524, pp.131-139.

Grant, S.C. (1997). DCIEM Fact Sheet: Navigation in Virtual Environments. North York: DCIEM, Simulation and Training Sector.

Gupta, R., Whitney, D. and Zeltzer, D. (1997). Prototyping and design for assembly analysis using multimodal virtual environments. Computer Aided Design, Vol. 29(8), pp.585-597. August, 1997.

Gutierrez, T., Barbero, J.I., Auzpitarte, M., Carrillo A.R., and Eguidazu, A. (1998). Assembly simulation through haptic virtual prototypes. Proceedings of the Third PHANTOM Users Group Workhop, AI Lab Technical Report No. 1643 and RLE Technical Report No. 624, MIT, December 1998. http://www.sensable.com/community/PUG98/papers.htm. Sensable Technologies. Viewed on January 15, 2000. Last updated 01/07/00.

Hampson, E. and Kimura, D. (1988). Reciprocal effects of hormonal fluctuations on human motor and perceptual skills. Behavioural Neuroscience. Vol.102, pp.456-459. As referred to by Peters, et. al., 1998.

Hannaford, B. and Wood, L. (1989). Performance evaluation of a 6 axis high fidelity generalized force reflecting teleoperator. Proceedings of NASA Conference on Space Telerobotics. Vol. II, NASA, Greenbelt, MD, pp.87-96, January.

Harrison, D. and Jacques, M. (1996). Experiments in virtual reality. Oxford: Butterworth-Heinemann, 117 p.

Hendrix, C., and Barfield, W. (1996). The sense of presence within auditory virtual environments. Presence: Teleoperators and Virtual Environments, v 5, n 3, 290-301

Hollerbach, J.M., Cohne, E., Thompson, W., Freier, R., Johnson, D., Nahvi, A., Nelson, D. and Thompson, T.V. (1997). Haptic interfacing for virtual prototyping of mechanical CAD designs. Proceedings of the ASME Design Engineering Technical Conferences (DETC). September 14-17, 1997, Sacramento, California.

Howe, R. and Kontarinis, D. (1992). Task performance with a dextrous teleoperated hand system. Proceedings of SPIE. Vol. 1833, Boston, MA, pp. 199-207, November.

Huck, S.W. (2000). Reading statistics and research. 3rd Ed. New York: Addison Wesley Longman. 700 p.

Infield, S.D., (1991). An Investigation into the Relationship Between Navigation Skill and Spatial Abilities. *Doctoral Dissertation.* University of Washington. As cited by Nash et.al. (In Press).

Ishii, M and Sato, M. (1994). Force sensations in pick-and-place tasks. Proceedings of ASME WAM. DSC, Vol. 55(1), ASME, New York, pp.339-344.

Iwata, H., Yano, H. and Hashimoto, W. (1997). LHX: An integrated software tool for haptic interface. Computers & Graphics (Pergamon), vol. 21, no. 4, pp. 413-420, Aug 1997.

Jansson, G. (1999). Can a haptic display rendering of virtual three-dimensional objects be useful for people with visual impairments. Journal-of-Visual-Impairment-and-Blindness. 1999 Jul; Vol 93(7), pp. 426-429.

Jayaram, S., Wang, Y. Jayaram, U, Lyons, K. and Hart, P. (1999). A virtual assembly design environment. Proceedings of the IEEE Virtual-Reality-Annual-International-Symposium, VR'99. Houston, TX, USA. 1999, p 172-179.

Johansson, A.J. and Linde, J. (1999). Using simple force feedback mechanisms as haptic visualization tools. Proceedings of the IEEE Instrument Measurement and Technology Conference, vol. 2, pp. 820-824, 1999.

Kaczmarek, K.A. and Bach-Y-Rita, P. (1995). Tactile Displays. Chapter 9 of Virtual Environments and Advanced Interface Design. Edited by Barfield, W. and T.A. Furness III. New York: Oxford University Press. Pp.349-414.

Keppel, G. (1991). Design and analysis: A researcher's handbook. 3rd Ed. New Jersey: Prentice Hall, 594 p.

Kilshaw, D. and Annett, M. (1983). Right- and left-hand skill: I. Effects of age, sex, and hand preference showing superior skill in left-handers. British Journal of Psychology. Vol. 74, pp.253-268. As referred to by Peters, et. al., 1998.

Lin, M.C., Gregory, A., Ehmann, S., Gottschalk, S., Taylor, R. (1999). Contact determination for real-time haptic interaction in 3D modeling, editing and painting. Proceedings

of the Fourth PHANTOM Users Group Workshop. AI Lab Technical Report No. 1675 and RLE Technical Report No. 633, MIT, November 1999. http://www.sensable.com/community/PUG99/papers.htm. Sensable Technologies. Viewed on January 15, 2000. Last updated 01/07/00

Linn, M.A., and Peterson, A.C. (1985). Emergence and characterization of sex differences in spatial ability: A meta-analysis. Child Development, Vol. 56., p.1479-1498.

Luecke, G.R. and Zafer, N. (1998). Haptic feedback for virtual assembly. SPIE Conference on Telemanipulator and Telepresence technologies V, Boston, Massachusetts, November 1998. SPIE vol. 3524, pp. 115-122.

Maccoby, E.M. and Jacklin, C.N. (1974). The psychology of sex differences. Stanford, CA: Stanford University Press. As referred to by Crawford and Christensen (1995).

Massie, T.H. and Salisbury, J.K. (1994). The PHANToM™ Haptic Interface: A Device for probing virtual objects. Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Chicago, IL, Nov. 1994.

Massimino, M. and Sheridan, T. (1993). Sensory substitution for force feedback in teleoperation. Presence: Teleoperators and Virtual Environments, Vol. 2(4), pp.344-352.

McGee, M.K. (1999). Integral Perception in Augmented Reality. Published Dissertation. Virginia Polytechnical Institute and State University. 276 pp.

Nash, E.B, Edwards, G.W., Thompson, J.A. and Barfield, W. (2000). A review of presence and performance in virtual environments. International Journal of Human Computer Interaction, Vol.12(1), pp.1-36.

Pausch, R., Proffitt, D. and Williams, G. (1997) Quantifying immersion in virtual reality. Computer Graphics Proceedings. Annual Conference Series. Los Angeles, p.13-18.

Peters, M., Servos, P. and Day, R. (1998). Marked sex differences on a fine motor skill task disappear when finger size is used as covariate. Journal of Applied Psychology. Vol. 75(1), pp. 87-90.

Philips, C.W., Paterson, C.E. and Pettijohn, T.F. (1985). Feedback conditions and motor skills performance. The Journal of General Psychology. Vol.112(1), pp.53-57.

Poston ,T. and Serra, L. (1996). Dextrous Virtual work. Communications of the ACM. May 1996, Vol. 39(5), pp. 37-45.

Richard, P., Birebent, G., Coiffet, P., Burdea, G., Gomez, D., Langrana, N. (1996). Effect of frame rate and force feedback on virtual object manipulation. Presence: Teleoperators and Virtual Environments, v 5, n 1, 95-108.

Richard, P. and Coiffet, P. (1995). Human perceptual issues in virtual environments: sensory substitution and information redundancy. IEEE Workshop on Robot and Human Communication, 1995, pp. 301-306.

Ruspini, D. and Khatib, O. (1998). Acoustic cues for haptic rendering systems. Proceedings of the Third PHANTOM™ Users Group Workhop, AI Lab Technical Report No. 1643 and RLE Technical Report No. 624, MIT, December 1998. http://www.sensable.com/community/PUG98/papers.htm. Sensable Technologies. Viewed on January 15, 2000. Last updated 01/07/00.

Salisbury, J.K. and Srinivasan, M.A. (1997) Projects in VR: Phantom-Based Haptic Interaction with Virtual Objects. IEEE Computer Graphics and Applications, Sept. /Oct. 1997.

Satalich, G.A. (1995) Navigation & wayfinding in virtual reality: Finding proper tools and cues to enhance navigation awareness. Published thesis dissertation, University of Washington. As referred to in Nash et. al. (2000).

SensAble Technologies. (2000). 3-D Touch Applications. http://www.sensable.com/apps.htm. Viewed on Jan. 15, 2000. Last modified Jan 7, 2000.

Shepard, R.N. and Metzler, J. (1971). Mental Rotation of three-dimensional objects. Science, Vol. 171, p.701-703. As referred to McGee, 1999.

Sheridan, T.B. (1992a). Musings on telepresence and virtual presence. Presence: Teleoperators and Virtual Environments. Vol. 1(1), p. 120-125.

Sheridan, T.B. (1992b). Telerobotics, Automation, and Human Supervisory Control. MIT Press, Cambridge MA.

Sherman, J.A. (1978). Sex-related cognitive differences: an essay on theory and evidence. Springfield, Ill.: Thomas, p.44-54.

Siegel, S. and Castellan, N.J., Jr. (1988). Nonparametric statistics for the behavioral sciences. 2nd Ed. Boston: McGraw-Hill. 399p.

Snow, M.P. (1996). Charting Presence in Virtual Environments and its Effects on Performance. Unpublished doctoral dissertation, Virginia Polytechnic Institute and State University.

Springer S.L. and Gadh, R. (1997). Haptic feedback for virtual reality computer aided design. Concurrent-Product-Design-and-Environmentally-Conscious-Manufacturing-American-Society-of-Mechanical-Engineers,-Design- v 94 1997, ASME, Fairfield, NJ, USA. p 1-8.

Stone, R. (1992). Virtual reality tutorial. MICAD Conference, Micado, Paris, France. As referred to by Burdea, 1996.

Tate, D.L., Sibert L., and King, T. (1997). Virtual environments for shipboard firefighting training. Proceedings of the IEEE Virtual Reality Annual International Symposium, IEEE Computer Society, Albuquerque, New Mexico, USA, 61-68.

Thorndyke, P.W. (1980). Performance models of spatial and locational cognition. Technical Report R-2676-ONR. Washington, DC: Rand.

Vandenburg, S. and Kuse, A.R. (1978). Mental rotations: A group test of three-dimensional spatial visualization. Perceptual and Motor Skills, Vol. 47, p. 599-604. As referred to by Crawford and Christensen, 1995.

Vandenburg, S. and Kuse, A.R. (1979). Spatial Ability: A critical review of the sex-linked major gene hypothesis. In M. Wittig & A.C. Peterson (Eds.), Sex-related differences in cognitive functioning. New York: Academic Press, p.67-95.

Wenzel, M.E. (1992). Localization in virtual acoustic displays. Presence: Teleoperators and Virtual Environments. Vol. 1(1), pp.80-107.

Williges, R.C. (1998). ISE 5615: Human Factors Research Design Class Notes. Virginia Polytechnical Institute and State University, Fall 1998.

Winer, B.J., Brown, D.R. and Michels, K.M. (1991). Statistical principles in experimental design. 3rd Ed. New York: McGraw-Hill. 1057 p.

Witmer, B.G., Bailey, J.H., Knerr B.W., and Parsons, K.C. (1996). Virtual spaces and real world places: transfer of route knowledge. International Journal of Human-Computer Studies, v 45, 413-428.

Youngs, E. (1999). Old and young user interfaces: is there a "Nintendo Effect"? CTG Newsletter, Spring 1999. In CTG Digest, Vol.2(1), January 2000.

Zajtchuk, R. and Satava, R.M. (1997). Medical applications of virtual reality. Communications of the ACM. September 1997, Vol. 40(9), pp. 63-64.

Zhu, H. (1988). Electrotactile Stimulation. In J. Webster Ed., Tactile Sensors for Robotics and Medicine, John Wiley & Sons, New York, pp. 341-353. Referred to by Burdea, 1996.

Appendix A: IRB Protocol and Informed Consent Form

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY

IRB Protocol for:

"Investigating the Use of Force Feedback and Auditory Substitutions in a Virtual Environment Task". Towards the M.S. Requirement in Human Factors Engineering

Investigators: Gregory W. Edwards and Dr. Woodrow Barfield

1. Justification

There are currently many limitations with haptic interaction (the sense of touch) in virtual environments (VE), including limitations of the technology, high costs, difficult programming and possibly large processing requirements. Given these problems, many designers of VEs are uncertain of whether or not to incorporate haptics into their applications. The research proposed by this document will attempt to aid designers in this decision by providing the results of an experiment that has users try a general VE task with and without haptic feedback as well as with and without auditory substitution cues while measuring their performance and obtaining their subjective feedback. From this study, designers of VEs will gain knowledge of the suitability of haptics to VE tasks as well as that of using auditory cues to replace haptic feedback or to enhance it. These results could save designers considerable effort in creating VEs that are effective.

2. Procedures

This experiment will recruit 24 volunteers (12 males and 12 females) between the ages of 18 and 55 with normal or corrected to normal vision from the Virginia Tech Campus. Participants will be pre-screened out of the experiment if they have had extensive previous virtual environment experience. It is hoped to have funding from the department for paid participation ($10 an hour), otherwise participation will not be compensated.

The experimental design is a 2 x 2 within subject design, with the factors being haptics (with or without) and sound (with or without). The order of presentation of the conditions will be assigned randomly for each subject from a pre-determined list of orders generated from balanced latin squares. The location for the study will be the virtual environment lab (VEL), Whittemore 560. Upon arrival, the research goals, but not the hypothesis, will be explained to the participants and they will be asked to sign an informed consent form (a copy of which is appended to this document). A pre-questionnaire will then be administered to assess standard demographic information, including their previous computer experience. The next step is to administer the Guilford Zimmerman spatial orientation test (a paper and pencil test asking them to identify the orientation of a series of drawings). An initial familiarization and practice session lasting 6 minutes will follow the spatial test. This is given to habituate the participants to interacting with a VE and the PHANToM™. The participant will interact with the VE through a head mounted display (HMD) rendering stereo images, a PHANToM haptic interaction device, a mouse and

sounds through headphones on the HMD. The HMD, PHANToM™ and VE are shown in Figure 1.

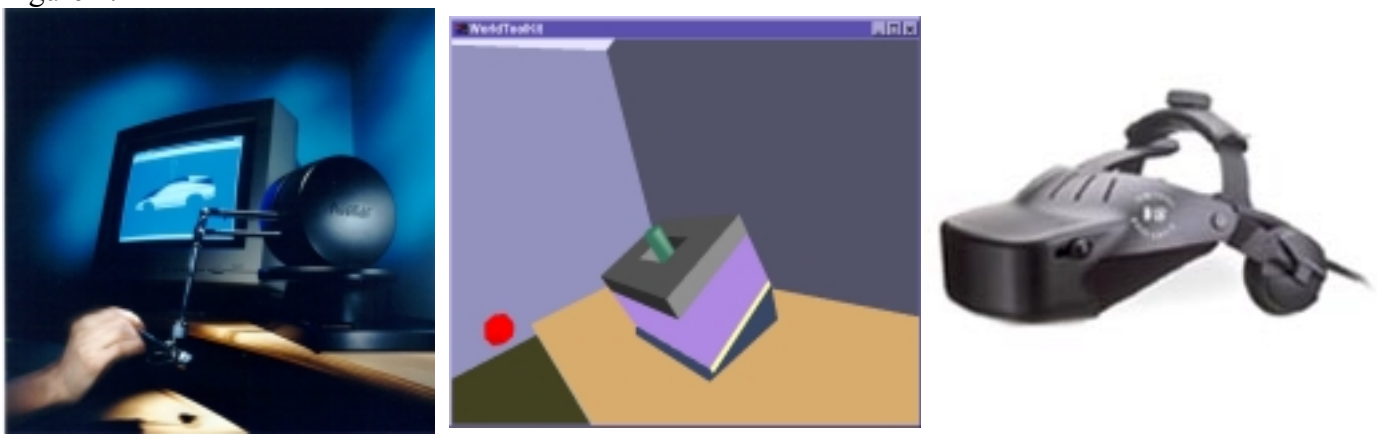


Figure 1: From left to right: PHANToM™ haptic interaction device (http://www.sensable.com/products/premium.htm), an image of the virtual environment and the head mounted display (http://www.virtualresearch.com/products/v8.htm).

After the familiarization session the participant will be told to take off the HMD and reminded that at any time they may stop the experiment if they do not feel comfortable. They will then begin the task. At the end of each condition they will be asked to take off the HMD and answer Likert-type ratings on the specific feedback condition they just underwent. At the end of all four conditions they will be also asked to fill out a subjective questionnaire including Likert-type rating scales and open-ended questions comparing the conditions. A copy of all questionnaires has been appended to this document. It is hoped that the relatively short exposure to the VE along with the rests provided by the questionnaires between each condition will avoid any post simulator sickness effects. The time sequence of these procedures is shown below in Table 1.

Table 1: Breakdown of Experiment Timing

Event	Estimated Time (minutes)
Pre-Questionnaire	5
Explanations and Informed Consent	5
Guilford Zimmerman Test	15
Practice and Familiarization	6
Briefing	2
4 Task Iterations + questionnaires	4 X 3 + 4 X 1.5 = 18 maximum
Post-Questionnaire	5
Total Participant Time	56

3. Risks and Benefits

There is the risk of simulator sickness, however many precautions have been built into the study to try to eliminate such an effect. These precautions include the intermittent use of the technology (only 3 minutes at a time), the limited total exposure (only 18 minutes maximum including familiarization and trials), and the inherent non-locomotive nature of the task (a common factor in creating simulator sickness). Participants will be encouraged to discontinue any usage of the equipment should they begin to feel ill at ease or nauseated in any way. The

benefit to the individual is in both contributing to the better design of virtual environments and also the opportunity to witness first-hand new technology currently being used in engineering and scientific research.

4. Confidentiality/Anonymity

The data for each participant will be numbered and no names will be used on the data collection forms or any other data recording medium. The subject/number coordination key will be kept for a period of one month after the study has been completed, after which it will be discarded. During this period the key will be kept in a secure place.

5. Informed Consent

All participants will receive two informed consent forms to be signed before beginning the study, one for them and one for our records. The consent form can also be found at the end of this document.

6. Biographical Sketches

- **Edwards, G.W.:**

A second year Masters student within the Human Factors Option of the ISE department who currently has 6 publications, two of which are in the area of virtual environments and human computer interaction. He has both participated in and run several user studies. His experience in running experiments has come from a virtual reality experiment that was run with 44 subjects during his undergraduate degree and three other small human-computer interaction studies involving about a dozen participants each. His participation has been in several small human computer interaction and human factors design studies as well as several large environmental and performance orientated physiological studies. In addition, his academic schooling in the area of research design and human factors have made him aware of many of the potential pitfalls inherent in user testing.

- **Barfield, W.**

Dr. Barfield is a Professor in the department of Industrial and Systems Engineering where he is also director of the Virtual Environment Laboratory. His areas of research include virtual and augmented reality, wearable computers, and force feedback displays. He is senior editor for "Presence: Teleoperators and Virtual Environments" and "Virtual Reality: Research, Development, and Applications". He has published 140 archival papers including five books.

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY

Informed Consent for Participants
of Investigative Projects

"Investigating the Use of Force Feedback and Auditory Substitutions in a Virtual Environment Task". Towards the M.S. Requirement in Human Factors Engineering

Investigators: Gregory W. Edwards and Dr. Woodrow Barfield

I. The Purpose of this Research/Project

There are currently many limitations with including the sense of touch (haptics) in virtual environments (VE), including limitations of the technology, high costs, difficult programming and sometimes large processing requirements. Given these problems, many designers of VEs are uncertain of whether or not to incorporate haptics into their applications or to use other more proven interaction cues such as sound. To determine how useful haptic and sound interaction cues are in general virtual environment usage we would like your help in giving us opinions and feedback on the technologies. As one of 24 participants in this study you will be helping to further our understanding of the utility of both haptic and sound cues in VEs and hopefully allow the creation of more useful VE applications.

II. Procedures

You will first be asked to fill out a questionnaire asking some basic questions about yourself and your computer experience. Then, to assess your spatial ability, you will be given a pencil and paper test. Following this you will begin a practice session with the equipment. The practice session allows you to become familiar with the equipment by giving you a minute and a half to use each of four different sets of feedback cues (related to sound and touch). In between each you may take a break if you wish. After the practice session you will be asked to take off the head-mounted display and the experimenter will brief you on the specific manipulation task you will have to perform. This task will be performed 4 times, each time with a different set of feedback cues (the ones you were exposed to in the practice session). For each task you will be timed on how long it takes and the number of mistaken collisions you make with the parts. After finishing each task you will be asked to take off the head mounted display and answer a set of questions on the feedback you received. After all four tasks are completed you will also be asked to fill out a final questionnaire. The entire session is not expected to take longer than an hour. If at any time you wish to stop the study, just mention this to the experimenter and the study will be stopped. You are under no obligation to complete it.

III. Risks

Although we have minimized the risk of harm by keeping the virtual environment exposures short and simple, there is still a chance that you may suffer from simulator sickness. The symptoms commonly associated with simulator sickness are temporary disorientation and

nausea. If you feel ill at ease at any time during the study you are encouraged to stop immediately, there will be no penalty for doing so.

IV. Benefits of this Project

This research will help us to determine the benefits of using haptics and sounds in virtual environment applications. In the future, this will allow us to create virtual environments that are easier to use and more helpful to the people that use them. No promise or guarantee is being made on the benefits that this will incur and none of these benefits are being promised to encourage you to participate.

V. Extent of Anonymity and Confidentiality

The data will be kept according to a subject number and will be kept and used solely for the purposes of the evaluation of the utility of the different feedback cues provided in the virtual environment. At no time will the researchers release the results of the study to anyone other than individuals working on the project without your written consent

In some situations, it may be necessary for an investigator to break confidentiality. If child abuse is known or strongly suspected, investigators are required to notify the appropriate authorities. If a subject is believed to be a threat to herself/himself or others, the investigator is required to notify the appropriate authorities.

VII. Freedom to Withdraw

You are free to withdraw from the study at any time without penalty. You are also free to not answer any questions or respond to experimental situations that you choose.

VIII. Approval of Research

This research project has been approved, as required, by the Institutional Review Board for Research Involving Human Subjects at Virginia Polytechnic Institute and State University.

IX. Subject's Responsibilities

I voluntarily agree to participate in this study. I have the following responsibilities:

- Giving honest answers to the questions asked.
- Giving honest opinions and feedback when asked for.
- Completing the tasks to the best of my abilities.

X. Subject's Permission

I have read and understand the Informed Consent and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent for participation in this project.

If I participate, I may withdraw at any time without penalty. I agree to abide by the rules of this project.

_____

Signature Date

Should I have any questions about this research or its conduct, I may contact:

An Investigator:	Gregory W. Edwards	231-9084
Faculty Advisor:	Dr. Maury Nussbaum	231-6053
Chair, IRB, Research Division:	H. T. Hurd	231-5281

Appendix B: Questionnaires

Haptics and Sounds in Virtual Environments
Pre-questionnaire

Please fill out the following questions to help us gather some basic information about yourself. Please note that if you have had **extensive** virtual environment experience you will not be able to participate in this study. Please alert the experimenter if this is the case.

Age:_____ Major:_____
Normal (or corrected to normal) vision? Yes / No Color Blindness? Yes / No

Current School Level (please circle one) :
1. Freshman
2. Sophomore
3. Junior
4. Senior
5. Master's
6. PhD
7. Other:_____
8.

1. Level of computer experience (please circle one):
Low Medium High

2. Ever had any Virtual Environment Experience? Yes / No
- If so, please briefly describe the type of experience, it's approximate length of time and frequency and how recently this occurred. E.g. Virtual hang-gliding video game, one time only, for about 5 minutes at an amusement park, last year sometime.

_____
_____

3. Ever played interactive video games that require reflexes and coordination (not just typing)? Yes / No

- If yes, do you still play them? Yes / No

- If yes, approximately how often do you play these video games? (please circle one)

1-2 hrs/week 3-4 hrs/week 5-6 hrs/week 7-8 hrs/week Over 8 hrs/week

- If no, approximately how long ago did you play them and with what frequency? (e.g. last year, about once a week for a couple of hours each time)

_____

_____

4. Have you ever used a force feedback device before (e.g. on a video game)? Yes / No

- If yes, please describe what type, when and how much you used the device:

_____

_____

_____

_____

Haptics and Sounds in Virtual Environments
Questionnaire

The following questions are to be completed after each trial and will be used to gather information on the benefit and overall effect of having the different feedback conditions in virtual environments.

Trial 1:

Please circle a number for each of the following questions relative to the specific feedback condition you just completed.

1. Overall reaction to the system was

   0 1 2 3 4 5 6 7 8 9

   Not favorable Very favorable

2. Found the interaction with the parts and the virtual environment

   0 1 2 3 4 5 6 7 8 9

   Not realistic Very realistic

3. Interacting with the parts and accomplishing the task was

   0 1 2 3 4 5 6 7 8 9

   Difficult Easy

4. Learning to use the system with the feedback provided was

   0 1 2 3 4 5 6 7 8 9

   Difficult Easy

5. The speed and continuity with which the simulation was updated was

   0 1 2 3 4 5 6 7 8 9

   Too slow Fast enough

6. How well you accomplished the task

   0 1 2 3 4 5 6 7 8 9

   Not well Very well

Trial 2:

Please circle a number for each of the following questions relative to the specific condition you just completed.

1. Overall reaction to the system was

0 1 2 3 4 5 6 7 8 9

Not favorable Very favorable

2. Found the interaction with the parts and the virtual environment

0 1 2 3 4 5 6 7 8 9

Not realistic Very realistic

3. Interacting with the parts and accomplishing the task was

0 1 2 3 4 5 6 7 8 9

Difficult Easy

4. Learning to use the system was

0 1 2 3 4 5 6 7 8 9

Difficult Easy

5. The speed and continuity with which the simulation was updated was

0 1 2 3 4 5 6 7 8 9

Too slow Fast enough

6. How well you accomplished the task

0 1 2 3 4 5 6 7 8 9

Not well Very well

Trial 3:

Please circle a number for each of the following questions relative to the specific condition you just completed.

1. Overall reaction to the system was

0 1 2 3 4 5 6 7 8 9

Not favorable Very favorable

2. Found the interaction with the parts and the virtual environment

0 1 2 3 4 5 6 7 8 9

Not realistic Very realistic

3. Interacting with the parts and accomplishing the task was

0 1 2 3 4 5 6 7 8 9

Difficult Easy

4. Learning to use the system was

0	1	2	3	4	5	6	7	8	9
Difficult									Easy

5. The speed and continuity with which the simulation was updated was

0	1	2	3	4	5	6	7	8	9
Too slow								Fast enough	

6. How well you accomplished the task

0	1	2	3	4	5	6	7	8	9
Not well									Very well

Trial 4:

Please circle a number for each of the following questions relative to the specific condition you just completed.

1. Overall reaction to the system was

0	1	2	3	4	5	6	7	8	9
Not favorable									Very favorable

2. Found the interaction with the parts and the virtual environment

0	1	2	3	4	5	6	7	8	9
Not realistic									Very realistic

3. Interacting with the parts and accomplishing the task was

0	1	2	3	4	5	6	7	8	9
Difficult									Easy

4. Learning to use the system was

0	1	2	3	4	5	6	7	8	9
Difficult									Easy

5. The speed and continuity with which the simulation was updated was

0	1	2	3	4	5	6	7	8	9
Too slow								Fast enough	

6. How well you accomplished the task

0	1	2	3	4	5	6	7	8	9
Not well									Very well

Haptics and Sounds in Virtual Environments
Post-questionnaire

Please answer the following questions comparing the different feedback conditions you received while doing the task.

1. Which condition did you like the best and the worst? (please circle the best and cross out your worst)

No feedback Sound only Haptics only Sound and Haptics

2. Rate how helpful each feedback condition was in terms of helping you to do the manipulation task quickly and accurately. (please check a box for each)

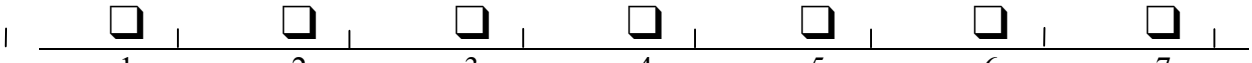
No feedback (just visual):

1	2	3	4	5	6	7
not helpful			moderately helpful			very helpful

Sound only :

1	2	3	4	5	6	7
not helpful			moderately helpful			very helpful

Haptics only:



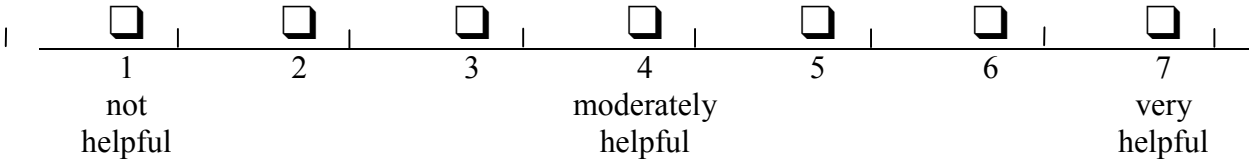
1	2	3	4	5	6	7
not helpful			moderately helpful			very helpful

Sound and Haptics:

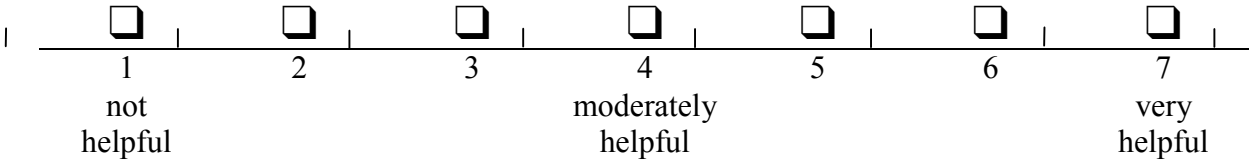
1	2	3	4	5	6	7
not helpful			moderately helpful			very helpful

3. Rate how helpful you think each feedback condition would be if you were an industrial engineer asked to use this system to evaluate a designer's first digital model of a system of parts. The evaluation would look at things such as how well the parts fit together, can be accessed for maintenance (pretend the barrier in the tasks you did was part of a fixture around the part that might exist in a real system), can be taken apart and put back together, etc. (please check a box for each condition)

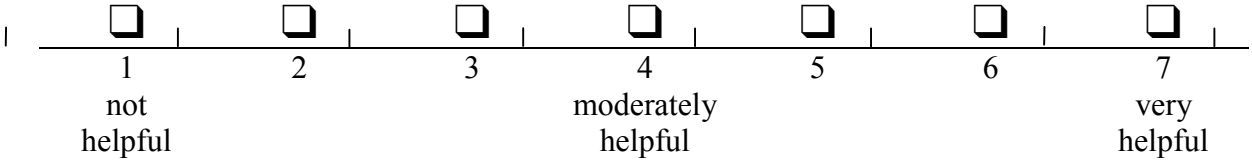
No feedback (just visual):



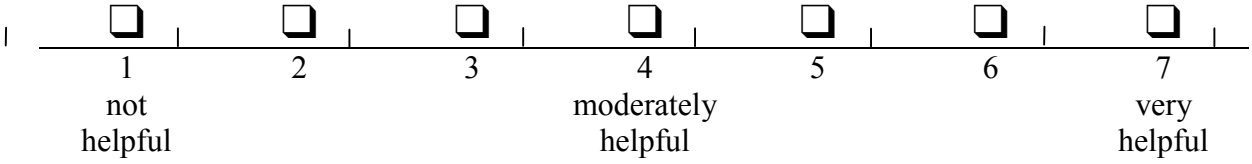
Sound only :



Haptics only:



Sound and Haptics:



4. Please comment on the feedback cues you received, how they helped you (if at all) and by how much. A feedback cue in this experiment refers to either a sound, the force feedback or the visual parts of the program that relayed information to you about the task you were doing (e.g. the sound of you hitting an object, the feeling of collision between two objects, an object flashing when you connect it to another).

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Thank you for your participation!

Appendix C: Participant Instructions and Procedures

Investigating the Use of Force Feedback and Auditory Substitutions in a Virtual Environment Task
Instructions and Procedures Script

1. Ask them if they have ever had extensive virtual environment or haptic feedback usage before.

2. Have them fill out the informed consent form. Inform them that you will answer any questions they may have and that you will reiterate verbally a brief description of the study to them afterwards.

3. Verbal Description:

   "This study is aimed at investigating how useful it is to have sound and the sense of touch in a virtual environment. The first thing to do is to fill out a questionnaire and then take a spatial abilities test. After that, you will try a demo to get familiarized with using the PHANToM. Then you will be given some specific instructions and asked to complete a manipulation task in the virtual environment four times with a specifc feedback condition, involving sound and touch. During these tasks your performance will be monitored by the time it takes you and the number of times you bump into things. At the end of this set of trials you will be given a questionnaire asking you some questions specifically on the set you just completed. You will then be asked to complete another set of 4 trials with a different set of feedback and to answer a questionnaire again. There are a total of 4 sets of tasks that you will be asked to do and then you will be asked to fill out a final questionnaire. I'll give you more specific instructions on the task itself after you have filled out the questionnaire and the spatial abilities test. Please recall that if you begin to feel nauseous at any time you are encouraged to stop and withdraw from the study with no penalty. "

4. Fill out the pre-questionnaire.

5. Fill out the mental rotation task. Tell them they have 3 minutes for each part and that each part has 2 pages.

6. Now give instructions for the task and ask if they are right or left-handed.

   "The task will be to disassemble the following (show them the picture) system of parts and to replace the yellow part with a similar blue part that will be sitting on the workbench when you enter the virtual environment. To interact with the parts you will be using a phantom (show them) with a pen attachment on the end. The end of the pen will correspond to a small red ball in the virtual environment and all it's movements will be followed by the ball (just like a 3-d cursor with rotation). The phantom is capable of also exerting force when the program tells it to, allowing the program to simulate solid

objects in the environment. However, I ask that you be careful with the device as it was not designed to take large or sudden forces. So no sudden jerks or hard presses on things. To grab things place the cursor on a part and press the button on the pen. As long as you hold the button down you will be grabbing the part. Once you let go of the button the part will be released. Do not double-click, please only use single clicks. I will now let you try a demo for 1:30 minute to get used to using the phantom."

7. Use Dice demo and tell them they can grab the die.

8. "To disconnect the parts from one another just grab the top most part and begin moving it. Note that you cannot move an object if another object is on top of it. So you must always remove the topmost piece first. When taking apart the system, you may place the parts anywhere on the table and you may do so by dropping them if you wish. To re-attach parts place the part you are moving as close to its connected state as you can and then release (it will not connect until you release it). It does not have to be perfectly aligned for it to connect. But relatively close, use the whitish/gray lines on the parts to help align the front of the part. Note that you cannot drop a part into place, you must place it. Various things will happen when it connects depending on the feedback you are receiving, but you will always see the part flash. If it does not flash then it is not connected and you need to grab it again and re-position it. This may seem like a lot of information at first, which is why you will be asked to do the trial 4 times for each condition, during which you will have time to get used to the task."

   "You will do 4 sets of tasks like this. Each set of tasks has the same objective, but you will be given different types of feedback (i.e. sound and the sense of touch). During each trial your performance will be measured by the time it takes you to take apart the system, replace the part and put it back together AS WELL AS the number of times you hit parts against things when you are moving them around. So try to complete the task quickly while bumping parts around the least amount possible, this includes when you are lining up parts to be connected. Collisions are not counted when you drop the parts however, only when you are holding them. Also, when parts collide together they do not slide, they will get stuck. So when you make a collision and want to keep moving the part then back it up a bit and then move it. A last piece of advice is to be patient with the device sometimes. There is a lot of processing going on and sometimes the system is slow."

   "Do you have any questions?"

9. When putting on HMD for actual practice trials: Adjust nosepiece on HMD and back of headpiece.

10. Before beginning: "For the first trial you may explore the environment a bit before starting. The timer will only start when you grab a part. You may move the cursor around and feel things to get used to the environment before you begin. Once you grab the first part the timer will start though. Once you've put the whole thing back together I will press a key and set up the program to run again. Last of all, have fun, this is not meant to be a stressful experiment."

Appendix D: User Comments for each Feedback Condition

No Feedback (just visual)

- No sounds startling the person or forces inhibiting control
- Can concentrate more
- Calmer and more comfortable
- Hard to know if you bumped into things or how precise your movements were
- Less realistic
- Flashing of object when connected was the best feedback (augmented visual is good)
- Flashing helpful but sound and haptics better
- Could not discern edges in visual, loss of perspective

Sound Feedback

- Good for helping to detect collisions
- Less strenuous than force feedback
- Should have had a sound cue when the cursor hit something, not just when parts hit things. (more augmented sounds)
- Sound told you you were doing something wrong but not what
- Made them feel bad everytime they hit something (stressed them out more)
- Distracting, could concentrate better without them
- "The sound was very annoying!"
- Enhanced realism, but not very helpful
- A very familiar feedback
- Ok, but needed to be more realistic (i.e. real recorded sounds)
- Helped to know when connected
- Sound seemed to be off
- "Sound on the other hand was almost completely irrelevant when present in the absence of touch. It was quite distracting when I was trying to accomplish the task"
- "Sound worked well when I also had the force feedback, alone is was just annoying"
- Not that helpful just neat to be able to hear

Force Feedback

- Good for detecting errors
- Made controlling harder though (some bouncing around)
- Force told you more specifically what you were doing wrong (than sound)
- Helps to feel object when picking it up
- Wish parts could slide against one another, no sliding made it frustrating and distracting
- Not always correlated with the visual
- May lead to a nervous breakdown (stressful) if used every day
- Helpful in that it was similar to real world
- Put strain on the wrist
- "Useful if no sound was present"
- "Helped but made the task harder to control"
- "awesome and very realistic"
- "the tactile sense was tremendously helpful in completing the assigned task. It was as though I was actually moving the object"
- "Being able to feel if you were stuck on something was somewhat helpful but overall it was easier for me just by sight."
- The pieces would get stuck together too long
- Too much banging around made it more stressful

Sound and Force Feedback

- Too much feedback
- The more real (the more feedback provided) the more difficult it seemed. Probably because things seem easier on a screen than they are in real life.

Appendix E: C++ Code for the VE

Haptics_main5.cpp: The Main file

```
/*******************************************************/
/* Haptic/Audio Industrial Maintenance Task v1.0         */
/* Greg Edwards, March 1, 2000                           */
/* Some Parts of Haptics Code is borrowed with many          */
/* thanks from Millersville University WTK-6DOF                  */
/* at http://cs.millersv.edu/haptics/index.html                  */
/*******************************************************/

#include "globals.h"  //some global variables I've defined
#include "wt.h" //for world tool kit
//#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include "wtcpp.h" //to use WTK C++ classes
#include "parts3.h" //has part class with model holder and data containers
#include "picking.h"  //has picking and moving functions
#include "collision4.h" //has collision detection and movement functions
#include "gravity.h"  //to implement falling of parts
#include <math.h>
#include "RigidManipulator.h"  //special manipulator subclass created to send
collision forces to phantom
                                                        //modified from phantom dice
demo.
#include "MovingPart.h"


/***HAPTICS STUFF***/
#include "GHOST.h"
bool haptics_on = false;
bool sound_on = false;
WtMovGeometry *phantom_cursor = NULL;
gstPHANToM *phantom;
#define PHANTOM_NAME       "Default PHANToM"
#define STIFF 1.0          /* This is a solid object */
#define NUM_SOUNDS  4

static gstPoint cursor_pos; /* This is the position of the PHANToM cursor. */
static gstScene *scene = NULL;
static gstSeparator *rootH;
static gstSeparator *hapticScene;
WtNodePath *phantom_cursor_path;
gstBoundaryCube *workspaceH;
WtNode* tempnode;  //used in set start position function
gstSeparator* haptic_parts[6];
gstTransformMatrix* rotationMatrix=NULL;  //for synchronizing rotations with picked
haptic object
gstPoint* translateVector = NULL;  //for synchronizing translations with the haptic
object
RigidManipulator* manip;  //manipulator to add collision and weight of object
MovingPart* movingobject;  //an overloaded dynamic haptic object that is set to moving
part to feel collisions

WtP3 haptic_pos[6];  //initial starting position and rotation
double haptic_rot[6];
```

```
float mass_array[6];

/*******************/
/***temp_objects***/
gstSeparator* temp_hpart;

/*** Objects ***/
Part* table=NULL;
Part* wall = NULL;
Part* wallb = NULL;
Part *current_parts[10];
WtNodePath *current_paths[6]; //paths for all connected objects
Part* space=NULL;
int maxindex=0;

/***************/

WtRoot *root=NULL;
WtMouse *mouse=NULL;
WtWindow *window =NULL;
WtWindow *window2 = NULL;
WtMotionLink *mouselink;
WtViewPoint *view=NULL;
WtUniverse *universe=NULL;
WtMtable* mtable = NULL;
unsigned char r,g,b;

/*** Collision Stuff ***/
Collision collision;
WtNode* inter_node=NULL;
WtNodePath* inter_path=NULL;
double maxdist = 0.0f;  //when two objects separated by more than this distance don't
check for collisions
WtP3* v_pos[6][40];  //holds up-to-date vertices of all objects
/************************/

/***WTUI stuff ******/

WTui *toplevel, *shell;
WTui *sub_num, *sub_label, *haptic, *haptic_label, *sound_label, *sound, *ok_button;
//char* subject_num = NULL;
//char subject_num[9];
/*******************/

/******SOUND*********/
WTsounddevice* sd;
WTsound* sounds[NUM_SOUNDS];
char filename[5];

/*********TIME and DATA COLLECTION*************/
//for data collection
char tbuffer [9];  //to hold the current time
ofstream* data = NULL;

//Function declarations
void init_haptics();

void start_haptics(void)
{
       /* Only initialize once. */
       static int first_time = TRUE;
       if (!first_time)
              return;
```

```
	first_time = FALSE;

	/*Initialize haptics and start haptics. */
	init_haptics();
	scene->startServoLoop();
}

//****************** Just Added for incorporating the Phantom *********************

//void get_haptics_pos(WtP3* cursor_pos, WtQ* cursor_rot)
void get_haptics_pos()
{
	scene->updateGraphics();
}

//************************
//**:::updatePhantomCB:::**
//************************
void updatePhantomCB(gstTransform *phantom, void *CBData, void *param)
{
		WTpq posq;
		WTp3 vec;
		WTm4 rotmatrix;
		float stylus_matrix[4][4];
		float newstylus_matrix[4][4];

		gstPoint SCP_WC = ((gstPHANToMGraphicsCBData *) CBData)->SCP_WC;

		// RWW added here 9/21/99 SCP
		vec[X] = SCP_WC[0];
	    vec[Y] = -SCP_WC[1];
	    vec[Z] = -SCP_WC[2];

		WTm4_init(rotmatrix);

		// get stylus matrix from Phantom device
		for (int i=0; i<4;i++)
		{
			for (int j=0;j<4;j++)
			{
				stylus_matrix[i][j] = ((gstPHANToMGraphicsCBData *)
CBData)->transform[i][j];
			}
		}

		//setup the rotation matrix - around x (180 deg)
		rotmatrix[1][1] = cos(PI);
		rotmatrix[1][2] = sin(PI);
		rotmatrix[2][1] = -sin(PI);
		rotmatrix[2][2] = cos(PI);

		//now do the rotation
		WTm4_multm4(stylus_matrix, rotmatrix, newstylus_matrix);
		WTm4_2pq(newstylus_matrix, &posq);
		phantom_cursor->SetOrientation(posq.q);
		phantom_cursor->SetTranslation(vec);

}

/*****************************************************************************/


//this function sets up the cursor for the Phantom
```

```
void setup_phantom_cursor(void)
{
       WtP3 pos;
       phantom_cursor = new WtMovGeometry(WTSPHERE, 0.6,4,6,FALSE,FALSE);
       phantom_cursor->SetRGB(255,0,0);
       root->AddChild(phantom_cursor);
       pos[X] = 0.0;
       pos[Y] = 15.0;
       pos[Z] = 11.0;
    phantom_cursor->SetTranslation(pos);
       phantom_cursor_path = new WtNodePath(phantom_cursor, root, 0);
       collision.set_phantom_geom(phantom_cursor);  //sets a pointer in collision to
be used there
}

void init_haptics()
{
       WtP3 t_pos;
       scene = new gstScene;
       rootH = new gstSeparator;
       hapticScene = new gstSeparator;
       gstVector xaxis(1,0,0);
       scene->setRoot(rootH);

       //prompt to reset object
       //cout<<"Place Phantom in reset position and press <ENTER> when ready"<<endl;
       //cin.get();

       // Create the phantom object.
       phantom = new gstPHANToM(PHANTOM_NAME, TRUE);
       if (!phantom || !phantom->getValidConstruction())
       {
              cerr << "Failed to create a valid PHANToM construction."<<endl;
              exit(-1);
       }
       phantom->setForceOutput(TRUE);

       rootH->addChild(phantom);
       // Create the phantom SCP object.
       gstPHANToM_SCP *phantomSCP = new gstPHANToM_SCP();
       // Tell the phantom object that the SCP is defined.
       phantom->setSCPNode(phantomSCP);
       // Set the graphics callback of the phantom
       phantom->setGraphicsCallback(updatePhantomCB,&cursor_pos);
       // Add the SCP to the root separator.
       rootH->addChild(phantomSCP);
       // Make workspace boundaries.
       workspaceH = new gstBoundaryCube();
       workspaceH->setWidth(WORKSPACE_SIZE*3);
       workspaceH->setLength(WORKSPACE_SIZE*1.7);
       workspaceH->setHeight(WORKSPACE_SIZE*1.2);
       workspaceH->setPosition(-1.0, -11.5, -35.0);
       workspaceH->setSurfaceKspring(STIFF);
       phantom->setBoundaryObj(workspaceH);
              // This contains the entire scene graph.
       hapticScene = new gstSeparator;


       // Add the boundary to the scene.
       hapticScene->addChild(workspaceH);
       rootH->addChild(hapticScene);

       if (haptics_on)
```

```
	{
		movingobject = new MovingPart;  //the dynamic haptic part used for
movement
		hapticScene->addChild(movingobject);
		//movingobject->set_collision(&collision);
		movingobject->setForce(collision.get_phantom_force());
		movingobject->setDepth(collision.get_depth());
		movingobject->setNormal(collision.get_normal());

		//Part1
		gstSeparator *part1h = gstReadVRMLFile("part1vrmltest2.wrl");
		hapticScene->addChild(part1h);

		//print out all errors while loading the file
		while (gstVRMLGetNumErrors() > 0)
		{
			gstVRMLError err = gstVRMLPopEarliestError();
			cout<<"Error in VRML file
"<<gstVRMLGetErrorTypeName(err.GetError())<<" ";
			cout<<err.GetMSG()<<" "<<"on line "<<err.GetLine()<<endl;
		}
		part1h->setTranslate(-4.7, -28.5, -37.2);
		part1h->rotate(xaxis,PI);
		haptic_parts[0] = part1h;  //set pointer in list

		//set initial haptic position and rotation for later connection use
		haptic_pos[0].Set(-4.7, -28.5, -37.2);
		haptic_rot[0] = PI;

		//Part2
		gstSeparator *part2h = new gstSeparator;
		//create the part now, first cube
		gstCube *part2cube1 = new gstCube;
		part2cube1->setHeight(2);
		part2cube1->setWidth(20.3);
		part2cube1->setLength(14.7);
		part2cube1->setTranslate(0,-1.2,0.4);

		//second cube
		gstCube *part2cube2 = new gstCube;
		part2cube2->setHeight(3.4);
		part2cube2->setWidth(12);
		part2cube2->setLength(9.2);
		part2cube2->setTranslate(0,1,0);

		part2h->addChild(part2cube1);
		part2h->addChild(part2cube2);

		part2h->setTranslate(-4.8, -26.8, -37);
		part2h->rotate(xaxis, PI/8.5);
		hapticScene->addChild(part2h);
		haptic_parts[1] = part2h;  //set pointer in list

		//set initial haptic position and rotation for later connection use
		haptic_pos[1].Set(-4.8, -26.8, -37);
		haptic_rot[1] = PI/8.5;

		//Part4
		gstSeparator *part3h = new gstSeparator;

		//create the part now, first cube, right side
		gstCube *part3cube1 = new gstCube;
		part3cube1->setHeight(9.6);
```

```
part3cube1->setWidth(3.1);
part3cube1->setLength(9);
part3cube1->setTranslate(4.6, 0, 0);

//second cube, left side
gstCube *part3cube2 = new gstCube;
part3cube2->setHeight(9.6);
part3cube2->setWidth(3.1);
part3cube2->setLength(9);
part3cube2->setTranslate(-4.6, 0, 0);

//third cube
gstCube *part3cube3 = new gstCube;
part3cube3->setHeight(9.6);
part3cube3->setWidth(6.8);
part3cube3->setLength(1.7);
part3cube3->setTranslate(0, 0, -3.46);

//fourth cube
gstCube *part3cube4 = new gstCube;
part3cube4->setHeight(9.6);
part3cube4->setWidth(6.8);
part3cube4->setLength(1.7);
part3cube4->setTranslate(0, 0, 3.46);

//put it all together under separator and move and rotate
part3h->addChild(part3cube1);
part3h->addChild(part3cube2);
part3h->addChild(part3cube3);
part3h->addChild(part3cube4);

part3h->setTranslate(-4.8, -20.7, -35);
part3h->rotate(xaxis, PI/8.5);

hapticScene->addChild(part3h);
haptic_parts[2] = part3h;  //set pointer in list

//set initial haptic position and rotation for later connection use
haptic_pos[2].Set(-4.8, -20.7, -35);
haptic_rot[2] = PI/8.5;

//part5
gstSeparator *part4h = new gstSeparator;
//create the part now, first cube, right side
gstCylinder *part4cyl = new gstCylinder;
part4cyl->setHeight(2.4);
part4cyl->setRadius(1.55);
part4h->addChild(part4cyl);
part4h->setTranslate(-4.8, -23.8, -36.0);
part4h->rotate(xaxis,PI/8.5);

hapticScene->addChild(part4h);
haptic_parts[3] = part4h;  //set pointer in list

//set initial haptic position and rotation for later connection use
haptic_pos[3].Set(-4.8, -23.8, -36.0);
haptic_rot[3] = PI/8.5;

//part7
gstSeparator *part5h = new gstSeparator;
//create the part now, first cube, right side
gstCylinder *part5cyl = new gstCylinder;
part5cyl->setHeight(9.54);
```

```
                part5cyl->setRadius(1.45);
                part5cyl->setSurfaceKspring(1);

                part5h->addChild(part5cyl);
                part5h->setTranslate(-4.8, -18.2, -33.7);
                part5h->rotate(xaxis,PI/8.5);
                hapticScene->addChild(part5h);
                haptic_parts[4] = part5h;  //set pointer in list

                //set initial haptic position and rotation for later connection use
                haptic_pos[4].Set(-4.8, -18.2, -33.7);
                haptic_rot[4] = PI/8.5;

                //Part8 (the replacement part).  No pointer in list until part replaced
                gstSeparator *part6h = new gstSeparator;
                //create the part now, first cube
                gstCube *part6cube1 = new gstCube;
                part6cube1->setHeight(2);
                part6cube1->setWidth(20.3);
                part6cube1->setLength(14.7);
                part6cube1->setTranslate(0,-1.2,0.4);

                //second cube
                gstCube *part6cube2 = new gstCube;
                part6cube2->setHeight(3.4);
                part6cube2->setWidth(14);
                part6cube2->setLength(10.2);
                part6cube2->setTranslate(0,1,0);

                part6h->addChild(part6cube1);
                part6h->addChild(part6cube2);

                part6h->setTranslate(30, -29.3, -38.5);
                //part8h->rotate(xaxis, PI/8.5);
                hapticScene->addChild(part6h);
                haptic_parts[5] = part6h;  //set pointer in list

                //table
                gstSeparator *tableh = new gstSeparator;
                gstCube *tablecube = new gstCube;
                tablecube->setHeight(1.8);
                tablecube->setWidth(140);
                tablecube->setLength(62);
                tablecube->setTranslate(0.15, -31.7, -34.7);
                //tablecube->rotate(xaxis,PI/8.5);

                tableh->addChild(tablecube);
                hapticScene->addChild(tableh);


                //wall
                gstSeparator *wallh = gstReadVRMLFile("wall1vrml2.wrl");
                hapticScene->addChild(wallh);

                //print out all errors while loading the file
                while (gstVRMLGetNumErrors() > 0)
                {
                       gstVRMLError err = gstVRMLPopEarliestError();
                       cout<<"Error in VRML file
"<<gstVRMLGetErrorTypeName(err.GetError())<<" ";
                       cout<<err.GetMSG()<<" "<<"on line "<<err.GetLine()<<endl;
                }
                wallh->setTranslate(-5, -32.5, -35);
```

```
		wallh->rotate(xaxis,PI);

		//second side of the wall
		//add haptic objects here
		gstSeparator *wallh2 = gstReadVRMLFile("wall2vrml2.wrl");
		hapticScene->addChild(wallh2);

		//print out all errors while loading the file
		while (gstVRMLGetNumErrors() > 0)
		{
			gstVRMLError err = gstVRMLPopEarliestError();
			cout<<"Error in VRML file
"<<gstVRMLGetErrorTypeName(err.GetError())<<" ";
			cout<<err.GetMSG()<<" "<<"on line "<<err.GetLine()<<endl;
		}
		wallh2->setTranslate(-5, -32.5, -35);
		wallh2->rotate(xaxis,PI);

		manip = new RigidManipulator;

	}  //end of if haptics is on statement
	else
	{
		//set object list to NULL...can check this to see if haptics is on
		for(int g=0;g<6;g++)
		{
			haptic_parts[g] = NULL;
		}
	}

	rotationMatrix = new gstTransformMatrix;
//setup collision object, part and paths already defined in load_system which is run
before this.
	collision.init(root, current_parts, current_paths, maxindex, maxdist, phantom,
haptic_parts, rotationMatrix, haptic_pos, haptic_rot, (WtMovable*)table,
(WtMovable*)wall, v_pos);  //initialize the collision object for collision detection
later on
}

void set_startpos(WtNodePath* path)
{
	WtP3 part_pos;
	WtQ part_q;
	WtP3 temp_pos;
	WtP3 local_pos;
	WtQ temp_q;
	WtPQ temp_frame;
	int i = 0;
	tempnode = path->GetNode(path->NumNodes()-1);
	i = ((Part*)tempnode)->get_index();

	//set last position in midpoint world coordinates
	path->GetTranslation(temp_pos);
	path->GetOrientation(temp_q);
	temp_frame.Set(temp_pos, temp_q);

	((WtGroup*)tempnode)->GetMidpoint(part_pos);
	((Part*)tempnode)->GetOrientation(part_q);

	part_pos.Local2WorldFrame(temp_frame, temp_pos);
	part_q.Local2WorldFrame(temp_frame, temp_q);

	((Part*)tempnode)->set_last_pos(temp_pos);
```

```
	((Part*)tempnode)->set_last_q(temp_q);

}



//This function loads the specific parts for the task to be done
void load_system1()
{
	//original positions are relative to the part before it (it's parent)...except
base
	WtP3 part1_1pos, part1_2pos, part1_3pos, part1_4pos, part1_5pos, part1_6pos;
	WtQ part1_1q, part1_2q, part1_3q, part1_4q, part1_5q, part1_6q;
	maxindex = 5;
	int index;
	int connect_to;
	WtP3 temp_pos;
	WtQ temp_q;
	WtPQ temp_frame;

	//set the starting/attached positions of the objects
	part1_1pos[X] = -5;
	part1_1pos[Y] = 32.5;
	part1_1pos[Z] = 35;
	part1_1q.Initialize();

	part1_2pos[X] = 0;
	part1_2pos[Y] = 0;
	part1_2pos[Z] = 0;
	part1_2q.Initialize();

	part1_3pos[X] = 0;
	part1_3pos[Y] = 0;
	part1_3pos[Z] = 0;
	part1_3q.Initialize();

	part1_4pos[X] = 0;
	part1_4pos[Y] = 0;
	part1_4pos[Z] = 0;
	part1_4q.Initialize();

	part1_5pos[X] = 0;
	part1_5pos[Y] = 0;
	part1_5pos[Z] = 0;
	part1_5q.Initialize();

	part1_6pos[X] = 30;
	part1_6pos[Y] = 34.1;
	part1_6pos[Z] = 35;
	part1_6q.Initialize();

	//create the object skeleton holders
	Part* part1_1, *part1_2, *part1_3, *part1_4, *part1_5,*part1_6;
	WtNodePath* part1_1path, *part1_2path, *part1_3path, *part1_4path,
*part1_5path,*part1_6path;

	index=0;  //the slanted base
	connect_to=-1;
	part1_1 = (Part*)MovNodeLoad("part1_1.nff", 1.0f);
	//index, connect_to, connectedposition translation, connectedposition rotation
	part1_1->set_all(index, connect_to, part1_1pos, part1_1q);  //leave rotation
weird for now
```

```
	root->AddChild(part1_1);
	part1_1->SetTranslation(part1_1pos);
	part1_1->SetOrientation(part1_1q);
	current_parts[index] = part1_1;
	//part1_1->SetName("Part1");
	//cout<<"made it to part1"<<endl;
	mass_array[index] = 0.5;
	//set max distance for collision detection to be the length of the biggest
object.
	maxdist = 20.1344f; //in scaled coordinates, since object scaled by 0.3

	index=1;  //the yellow base part
	connect_to=0;
	part1_2 = (Part*)MovNodeLoad("part1_2.nff", 1.0f);
	part1_2->set_all(index, connect_to, part1_2pos, part1_2q);
	part1_2->SetTranslation(part1_2pos);
	part1_2->SetOrientation(part1_2q);
	current_parts[index] = part1_2;
	//part1_2->SetName("Part2");
	mass_array[index] = 0.3;

	index=2;
	connect_to=1;  //the first hole part
	part1_3 = (Part*)MovNodeLoad("part1_4new.nff", 1.0f);
	part1_3->set_all(index, connect_to, part1_3pos, part1_3q);  //renumbered
parts...part4 now part3
	part1_3->SetTranslation(part1_3pos);
	part1_3->SetOrientation(part1_3q);
	current_parts[index] = part1_3;
	//part1_4->SetName("Part4");
	mass_array[index] = 0.3;

	index=3;  //the spring
	connect_to=1;
	part1_4 = (Part*)MovNodeLoad("part1_5a.nff", 1.0f);  //used to be part1_5a
	part1_4->set_all(index, connect_to, part1_4pos, part1_4q);  //leave rotation
weird for now, change back to 2,1 when spring
	part1_4->SetTranslation(part1_4pos);
	part1_4->SetOrientation(part1_4q);
	current_parts[index] = part1_4;
	//part1_5->SetName("Part5");
	mass_array[index] = 0.2;

	index=4;
	connect_to=3;  //the cork connects to the spring
	part1_5 = (Part*)MovNodeLoad("part1_7.nff", 1.0f);
	part1_5->set_all(index, connect_to, part1_5pos, part1_5q);  //leave rotation
weird for now, change back to 2,1 when spring
	part1_5->SetTranslation(part1_5pos);
	part1_5->SetOrientation(part1_5q);
	current_parts[index] = part1_5;
	//part1_7->SetName("Part7");
	mass_array[index] = 0.2;

	index=5;  //Will replace part 2 in array when switched.
	connect_to=0;
	part1_6 = (Part*)MovNodeLoad("part_fix.nff", 1.0f);
	root->AddChild(part1_6);
	part1_6->set_all(index, connect_to, part1_2pos, part1_2q);  //leave rotation
weird for now, change back to 2,1 when spring
	part1_6->SetTranslation(part1_6pos);
	part1_6->SetOrientation(part1_6q);
	part1_6->Rotation(0,-PI/9.6,0,WTFRAME_LOCAL);
```

```
	current_parts[index] = part1_6;
	//part1_8->SetName("Part8");
	part1_6->set_attached(false);  //not attached to anything
	part1_6->set_resting(true);                    //is resting on the table
	part1_6->set_last_pos(part1_6pos);  //don't usually have to do this since
initial position same as connecting position
	part1_6->set_last_q(part1_6q);          //which is set in the set_all function
	mass_array[index] = 0.3;

	//now connect all parts together
	for(int k=0; k<=maxindex-1; k++)  //"-1" because don't want to try to connect
part8
	{
		for(int l=0; l<=maxindex-1; l++)
		{
			if((current_parts[k])->get_index() == (current_parts[l])-
>get_connect_to())
			{
				(current_parts[k])->Attach((current_parts[l]),
WTNODE_APPEND);
			}
		}
	}


	//now create paths of all conected parts (for use later in intersection
testing)
	//do here in case parts have to be connected before paths can be made
	//Also have to have common ancestor node so that intersection testing will work
	index = 0;
	part1_1path = new WtNodePath(part1_1,root,0);
	current_paths[index] = part1_1path;  //list of pointers to all paths
	set_startpos(part1_1path);  //set last position in world coordinates

	index++;
	part1_2path = new WtNodePath(part1_2,root,0);
	current_paths[index] = part1_2path;  //list of pointers to all paths
	set_startpos(part1_2path);  //set last position in world coordinates

	index++;
	part1_3path = new WtNodePath(part1_3,root,0);
	current_paths[index] = part1_3path;  //list of pointers to all paths
	set_startpos(part1_3path);  //set last position in world coordinates

	index++;
	part1_4path = new WtNodePath(part1_4,root,0);
	current_paths[index] = part1_4path;  //list of pointers to all paths
	set_startpos(part1_4path);  //set last position in world coordinates

	index++;
	part1_5path = new WtNodePath(part1_5,root,0);
	current_paths[index] = part1_5path;  //list of pointers to all paths
	set_startpos(part1_5path);  //set last position in world coordinates

	index++;
	part1_6path = new WtNodePath(part1_6,root,0);
	current_paths[index] = part1_6path;  //list of pointers to all paths
	set_startpos(part1_6path);  //set last position in world coordinates

	for (int b=0;b<6;b++)
	{
		for (int c=0;c<40;c++)
		{
```

```
                v_pos[b][c] = NULL;
            }
        }


        //now load all vertices of objects into an array.  This makes accessing their
positions
        // in the collision algorithm faster since not constantly accessing functions
in scene graph loop.
        WTvertex* tempvertex = NULL;
        WtP3 tempvector;
        for (b=0;b<6;b++)
        {
            current_paths[b]->GetTranslation(temp_pos);  //needs to redefine
temp_frame for each object
            current_paths[b]->GetOrientation(temp_q);
            temp_frame.Set(temp_pos, temp_q);
            tempvertex = ((WtGeometry*)current_parts[b])->GetFirstVertex();
            for(int c=0;tempvertex !=NULL;c++)
            {
                ((WtGeometry*)current_parts[b])->GetVertexPosition(tempvertex,
tempvector);
                //convert to world frame
                v_pos[b][c] = new WtP3;
                tempvector.Local2WorldFrame(temp_frame, *v_pos[b][c]);  //put in
world coordinates
                tempvertex = WTvertex_next(tempvertex);
            }
        }
}

//determine which system to load
void load_models()
{
        setup_phantom_cursor();
        load_system1();
}

void FlipNormals(WtGeometry *geom)
{
        WTpoly *poly;

        geom->BeginEdit();
        poly = geom->GetFirstPoly();
        while (poly) {
            WTpoly_reversevertices(poly);
            poly = WTpoly_next(poly);
        }
        geom->EndEdit();
}

void build_scene()
{
        //add walls and table
            //create the room
            WtP3 space_pos;
            //space = new
WtMovGeometry(WTBLOCK,5*WORKSPACE_SIZE,(2.7f)*WORKSPACE_SIZE,(2.5f)*WORKSPACE_SIZE,FAL
SE);
            space = (Part*)new
WtMovGeometry(WTBLOCK,5*WORKSPACE_SIZE,(2.7f)*WORKSPACE_SIZE,(2.5f)*WORKSPACE_SIZE,FAL
SE);
            FlipNormals((WtGeometry*)space);
```

```
                ((WtMovGeometry*)space)->SetRGB(100, 100, 130);
                WTpoly *poly;
                ((WtMovGeometry*)space)->BeginEdit();
                poly = ((WtMovGeometry*)space)->GetFirstPoly();
                WTpoly_setrgb(poly, 180, 180, 250);
                poly=WTpoly_next(poly);
                WTpoly_setrgb(poly, 50, 50, 20);
                ((WtMovGeometry*)space)->EndEdit();
                root->AddChild(space);
                space_pos[X] = 0.0;
                space_pos[Y] = 0.0;
                space_pos[Z] = 35.0;
                ((WtMovGeometry*)space)->SetTranslation(space_pos);
                //space->Prebuild();
                //space->SetName("Space");
                current_parts[6] = space;
                space->set_index(6);
                //create the table
                WtP3 table_pos;
                table_pos[X] = 40;
                table_pos[Y] = 48;
                table_pos[Z] = -20;

                table=NULL;
                table = (Part*)MovNodeLoad("haptictable.nff", 1.0f);
                table->SetTranslation(table_pos);
                root->AddChild(table);
                //table->SetName("table");
                current_parts[7] = table;
                table->set_index(7);

                //the wall
                WtP3 wall_pos;
                wall_pos[X] = -5;
                wall_pos[Y] = 32.5;
                wall_pos[Z] = 35;

                wall=NULL;
                wall = (Part*)MovNodeLoad("wall.nff", 1.0f);
                wall->SetTranslation(wall_pos);
                root->AddChild(wall);
                //wall->SetName("wall");
                current_parts[8] = wall;
                wall->set_index(8);

                //a second wall with normals facing inwardly
                //the wall

                wallb=NULL;
                wallb = (Part*)MovNodeLoad("wallb.nff", 1.0f);
                wallb->SetTranslation(wall_pos);
                root->AddChild(wallb);
                //wallb->SetName("wallb");
                current_parts[9] = wallb;
                wallb->set_index(9);

                phantom_cursor->SetName("cursr");
}

inline void handle_key(int key)
{
        switch(key)
        {
```

```
	case 'q':
		(*data)<<"Total number of collisions:
"<<collision.get_hitcounter()<<endl;
		(*data)<<"*****End of Data Collection*****"<<endl;
		universe->Stop();
		break;
	default:
		break;
	}
}

static void ActionFn()
{
	if (phantom !=NULL)  //make sure that phantom is initialized first then start
doing actions
	{
		//remember that anything to be used on multiple run-throughs must be
static...
	static bool selected = FALSE;
	int key;


	//update position of phantom cursor with or without haptics
	scene->updateGraphics();

	//***** PICKING AN OBJECT *****
	if(phantom->getStylusSwitch())
	{
		if (!selected)  //if not currently selecting anything
		{
			inter_path = pick_object(root, phantom_cursor_path, maxindex,
current_parts, current_paths, selected, phantom, haptic_parts, rotationMatrix, manip,
movingobject, sound_on, sounds, mass_array);
			if (inter_path != NULL)
			{
				inter_node = inter_path->GetNode(inter_path->NumNodes()-1);
				current_paths[((Part*)inter_node)->get_index()] =
inter_path; //update path list if path changed as a result of attaching

				//write current time to file
				_strtime( tbuffer );  //gets current time
				(*data)<<"picked up: "<<tbuffer<<endl;
				//cout<<"Framerate is: "<<universe->FrameRate()<<endl;
			}
		}
	}
	//"selected" is set in the pick_object function call

	//****** COLLISION DETECTION *****
	if (selected) //if an object is selected and being moved around
		collision.manage_collision(inter_path);  //checks for collision and moves
appropriately if there is one


	//***** DROPPING OBJECT *****
	if(!phantom->getStylusSwitch() )  //if button is released check to make sure
nothing attached to cursor
	{	//...if still attached, detach it from cursor and check for attachments
with objects it is touching
		if (selected)  //and an object was selected
		{
		//***** ASSEMBLY CODE ******
		//if collision between objects as determined in collision detection
```

```
		//then check if parts can connect, if so then check if in position
		//if yes to both then attach and move object to correct final position
			inter_path = collision.check_attach(data);
			inter_node = inter_path->GetNode(inter_path->NumNodes()-1);
			//switch part 2 and 6 if part 6 is now being attached
			if (current_parts[5]->get_attached())  //if part 8 (fixed part) is
now attached then switch indices
			{
				//change part and path arrays now, note that these are only
local arrays to this object
				current_parts[5] = current_parts[1];
				current_paths[5] = current_paths[1];
				current_parts[1] = ((Part*)inter_node);
				current_paths[1] = inter_path;

				//now switch haptic array as well
				temp_hpart = haptic_parts[1];
				haptic_parts[1] = haptic_parts[5];
				haptic_parts[5] = temp_hpart;
			}
			else
				current_paths[((Part*)inter_node)->get_index()] =
inter_path; //update path list if path changed as a result of attaching

			selected = FALSE;
			//cout<<"Framerate is: "<<universe->FrameRate()<<endl;
		} //if selected loop
	}

	/****CHECK FOR FALLING****/
	if (!selected)
		checkfall(root, current_paths, current_parts, maxindex, haptic_parts,
table, wall, sound_on, sounds, v_pos);

	/************************/




/*********************************************************************/
	/*** Key board commands if needed...just add handle_key ***/
	key = WTkeyboard_getlastkey();
	if (key) handle_key(key);  /* interpret keypress */
	/*****************************************************/
	}  //end of if phantom initialized statement

}

//open sound device
bool open_sound_device()
{
	int num;
	sd = WTsounddevice_open(WTSOUNDDEVICE_WINMM,4,WTuniverse_getviewpoints());
	if (!sd)
	{
		WTerror("Couldn't initialize sound hardware\n");
		return(false);
	}
	else
	{
		cout<<"Sound is ok!!"<<endl;
		num = WTsounddevice_numplayable(sd);
		cout<<"The number of sounds playable at one time is: "<<num<<endl;
```

```
		WTsounddevice_setparam(sd, WTSOUNDDEVICE_ROLLOFF,1000.0f);
		return(true);
	}
}

//load all sounds
void load_all_sounds(void)
{
	cout<<"trying to load sounds"<<endl;
	sounds[0] = WTsound_load(sd, "hit2.wav");
	WTsound_setparam(sounds[0], WTSOUND_LOOPS,1.0f);
	WTsound_setparam(sounds[0], WTSOUND_PRIORITY, 0.5f);

	sounds[1] = WTsound_load(sd, "pickup.wav");
	WTsound_setparam(sounds[0], WTSOUND_LOOPS,1.0f);
	WTsound_setparam(sounds[0], WTSOUND_PRIORITY, 0.8f);

	sounds[2] = WTsound_load(sd, "letgo.wav");
	WTsound_setparam(sounds[0], WTSOUND_LOOPS,1.0f);
	WTsound_setparam(sounds[0], WTSOUND_PRIORITY, 0.8f);

	sounds[3] = WTsound_load(sd, "detach1.wav");
	WTsound_setparam(sounds[0], WTSOUND_LOOPS,1.0f);
	WTsound_setparam(sounds[0], WTSOUND_PRIORITY, 1.0f);
	//add other sounds here and set priorities
}


void getsub(WTui *pStruct, void *pData)
{
	//get subject number
	char* acounter = new char[2];
	acounter[0] = 'a';
	acounter[1] = '\0';
	char* subject_num = new char[6];
	subject_num = WTui_gettext(sub_num);
	cout<<"The subject number is:"<<subject_num<<endl;
	subject_num[5] = '\0';
	strcpy(filename, subject_num);

	//open data file and check for existing file
	strcat(filename, ".txt");
	cout<<"filename is "<<filename<<endl;
	data = new ofstream;
	//data = new ofstream(filename, ios::noreplace);
	data->open(filename, ios::noreplace);
	if(!*data)
	{
		cout<<"data file already existed"<<endl;
		universe->Stop();
	}


	//get sound and haptic flags
	int temp;
	temp = WTui_checkbuttonstate(haptic);
	haptics_on = temp;

	temp = WTui_checkbuttonstate(sound);
	sound_on = temp;

	cout<<"The sound state is : "<<sound_on<<endl;
	cout<<"The haptic state is :"<<haptics_on<<endl;
```

```
	cout<<"PRESS Q TO QUIT PROPERLY"<<endl;
	//add in code to close window
	WTui_delete(shell);
	start_haptics();  //this actually starts the phantom and only adds haptic
objects if haptics was selected in menu
	if (sound_on)
	{
		cout<<"Starting sounds"<<endl;
		sound_on = open_sound_device();
		if (sound_on)
			load_all_sounds();
	}
	collision.set_sound(sound_on, sounds);

	//insert parameter and subject data here
	(*data)<<"*****BEGINNING DATA FILE*****"<<endl
			<<"The sound is (1 = ON, 0 = OFF): "<<sound_on<<endl
			<<"The haptics are (1 = ON, 0 = OFF): "<<haptics_on<<endl<<endl;
}

void setup_params_window(int argc, char *argv[])
{
	int shell_width = 300;
	int shell_height = 320;
	//** Create toplevel widget **
	toplevel = WTui_init(&argc,argv);
	WTui_setscalefactor(1.0,1.0);

	//** Create the main form widget *
	shell =  WTuiform_new(toplevel, "Sound and Haptic Experiment",
				WTUIATT_LEFT, 20,
					WTUIATT_TOP, 20,
				WTUIATT_WIDTH, shell_width,
					WTUIATT_HEIGHT, shell_height, NULL);

	if (shell == NULL)
		printf("shell is NULL\n");

	//** Get the subject number **

	sub_label = WTuilabel_new(shell, "Subject Number (3 digits):", WTUI_TEXT,
WTUIATT_LEFT,10,WTUIATT_TOP,10,WTUIATT_WIDTH,175, WTUIATT_HEIGHT, 30, NULL);

	if (sub_label == NULL)
		printf("sub_label is NULL\n");

	sub_num = WTuitextfield_new(shell, "
",WTUIATT_LEFT,190,WTUIATT_TOP,8,WTUIATT_WIDTH,50, WTUIATT_HEIGHT, 20, NULL);

	if (sub_num == NULL)
		printf("sub_num is NULL\n");

	//check boxes for choosing haptics and sound options
	//haptics
	haptic = WTuicheckbutton_new(shell, "Check for Haptics",
WTUIATT_LEFT,10,WTUIATT_TOP,120,WTUIATT_WIDTH,150, WTUIATT_HEIGHT, 30,NULL);

	if (haptic == NULL)
		printf("haptic is NULL\n");

	//sound
	sound = WTuicheckbutton_new(shell, "Check for Sound",
WTUIATT_LEFT,10,WTUIATT_TOP,160,WTUIATT_WIDTH,150, WTUIATT_HEIGHT, 30,NULL);
```

```
	if (sound == NULL)
		printf("sound is NULL\n");

	ok_button = WTuipushbutton_new(shell, "OK",
WTUIATT_LEFT,110,WTUIATT_TOP,200,WTUIATT_WIDTH,40, WTUIATT_HEIGHT, 30, NULL);

	if (ok_button == NULL)
		printf("ok_button is NULL\n");

	WTui_setcallback(ok_button, WTUIEVENT_ACTIVATE, getsub, NULL);

}

void main(int argc, char *argv[])
{
	//variables
	short myevents[4];//will specify my event order to have sensor update before
actions
	WtP3 dir;
	WtAmbientLt *ambient;
	WtDirectedLt *directed, *directed1;
	WtP3 startpos;
	WtQ startq;
	RED = GREEN = BLUE = 0;

	//change this for stereo viewing
	universe->New(WTDISPLAY_DEFAULT,WTWINDOW_NOBORDER);

	root = universe->GetFirstRootNode();

	//Add stereo view parameters here
	view = universe->GetFirstViewPoint();

	window = universe->GetFirstWindow();
	window->SetBgRGB(RED, GREEN, BLUE);
	window->SetPosition(0,0,640,480);  //size of screen with HMD
	window->SetProjection(WTPROJECTION_ASYMMETRIC);

	//New Code for stereo window...test
	window2 = new WtWindow(641,0,640,480, WTWINDOW_NOBORDER);  //start second
window on right side
	window2->SetRootNode(root);
	window2->SetViewPoint(view);
	window2->SetEye(WTEYE_RIGHT);
	window2->SetProjection(WTPROJECTION_ASYMMETRIC);

	view->SetParallax(0.7f);
	view->SetConvDistance(4.0f);

	/* add lights */
	ambient = new WtAmbientLt(root);
	ambient->SetIntensity(.80f);

	directed = new WtDirectedLt(root);
	dir[0] = 0.68f;
	dir[1] = -0.48f;
	dir[2] = -0.58f;
	directed->SetDirection(dir);

	directed1 = new WtDirectedLt(root);
	dir[0] = -0.68f;
	dir[1] = 0.48f;
```

```
        dir[2] = -0.58f;
        directed1->SetDirection(dir);

        //Set Sensors
        mouse =new WtMouse;
        mouse->SetSensitivity(0.01*((float)(root->GetRadius())));
        //mouse->SetUpdateFn(WTmouse_drawcursor);
        //add Phantom declaration here
        mouse->SetUpdateFn(WTmouse_moveview2);
        //add PHANToM motionlink here

        setup_params_window(argc, argv);
        WTui_manage(shell);
        WTui_go(toplevel, 1);

        WTuniverse_setbgcolor(0x500);
        //Load Appropriate Models in beginning position for this simulation
        load_models();
        build_scene();

        window->ZoomViewPoint();
        window2->ZoomViewPoint();
        startpos[X] = 0.0f;
        startpos[Y] = -9.0f;
        startpos[Z] =-34.0f;
        view->SetPosition(startpos);
        view->Rotate(X,-0.15*PI,WTFRAME_VPOINT);

        //set universe event order and switch actions and sensor so actions after
sensor
        //the rest is the same

        universe->SetActions(ActionFn);
        myevents[0] = WTEVENT_OBJECTSENSOR;
        myevents[1] = WTEVENT_ACTIONS;
        myevents[2] = WTEVENT_TASKS;
        myevents[3] = WTEVENT_PATHS;
        universe->SetEventOrder(4,myevents);

        mouselink = new WtMotionLink(mouse, view, WTSOURCE_SENSOR, WTTARGET_VIEWPOINT);

        //remember to delete current parts before doing next system
        WtKeyboard::Open();

        universe->Ready();

        universe->Go();
        data->close();  //closes data file
        universe->Delete();
}
```

Ghost.h: Ghost Header Files

```
#include <fstream.h>
#include <gstBasic.h>
#include <gstBoundaryCube.h>
#include <gstButton.h>
#include <gstDynamic.h>
#include <gstNode.h>
#include <gstPHANToM.h>
#include <gstPHANToM_SCP.h>
#include <gstPolyMesh.h>
```

```
#include <gstPoint.h>
#include <gstScene.h>
#include <gstSeparator.h>
#include <gstSlider.h>
#include <gstTransformMatrix.h>
#include <gstRigidBody.h>
#include <gstVRML.h> //used to load the models in VRML 2 format
#include <ghostGLUTManager.h>  //used to view the VRML files temporarily

#include <gstCube.h>
#include <gstSphere.h>
```

Globals.h: global variables

```
unsigned char RED;
unsigned char BLUE;
unsigned char GREEN;
#define WORKSPACE_SIZE 35.0f
```

Parts3.h and Parts3.cpp: Graphical objects for parts

Parts3.h

```
#ifndef PARTS_H
#define PARTS_H

//==========================================================================
//   Filename : parts3.h
//      Written in April, 2000
//      By Greg Edwards
/*********************************************************************/
/* This is the base class of the moving parts in this simulation.   */
/* Parts are just Movable nodes with extra data variables for       */
/* position history, attachment information and current status      */
/* (whether attached or not and whether resting on anything or not. */
/*********************************************************************/
//==========================================================================

#include "wt.h"
#include <iostream.h>
#include "wtcpp.h"

struct part_position
{
       WtP3 connect_pos;  //will be in relative coordinates
       WtQ connect_q;
       WtP3 last_pos;  //will be in world coordinates
       WtQ last_q;
};

class Part: public WtMovable
{
private:
       int index;
       int connect_to;
       part_position pos;
       bool attached;
       bool resting;

public:
       void set_all(int i, int c, WtP3 pos, WtQ q);
       void set_index(int i);
```

```
	int get_index();
	void set_connect_to(int c);
	int get_connect_to();
	void set_connect_pos(WtP3 pos);
	WtP3 get_connect_pos();
	void set_connect_q(WtQ q);
	WtQ get_connect_q();
	void set_last_pos(WtP3 pos);
	WtP3 get_last_pos();
	void set_last_q(WtQ q);
	WtQ get_last_q();
	void set_attached(bool a);
	bool get_attached();
	void set_resting(bool r);
	bool get_resting();
};


#endif
```

Parts3.cpp

```
#include "parts3.h"


void Part::set_all(int i, int con_to, WtP3 apos, WtQ aq)
{
	index = i;
	connect_to = con_to;
	pos.connect_pos = apos;
	pos.connect_q = aq;
	pos.last_pos = apos;
	pos.last_q = aq;
	attached = true;  //starts out attached to something
	if (i == 0)
		attached = false;  //part 1 does not start out attached to
anything...it's the base, only has
	//things attached to it.
	resting = false; //only applicable if not attached, says whether in mid-air or
not
}

void Part::set_index(int i)
{
	index =i;
}

int Part::get_index()
{
	return index;
}

void Part::set_connect_to(int c)
{
	connect_to = c;
}

int Part::get_connect_to()
{
	return connect_to;
}

void Part::set_connect_pos(WtP3 apos)
```

```
{
	pos.connect_pos = apos;
}

WtP3 Part::get_connect_pos()
{
	return pos.connect_pos;
}

void Part::set_connect_q(WtQ aq)
{
	pos.connect_q = aq;
}

WtQ Part::get_connect_q()
{
	return pos.connect_q;
}

void Part::set_last_pos(WtP3 apos)
{
	pos.last_pos = apos;
}

WtP3 Part::get_last_pos()
{
	return pos.last_pos;
}

void Part::set_last_q(WtQ aq)
{
	pos.last_q = aq;
}

WtQ Part::get_last_q()
{
	return pos.last_q;
}

void Part::set_attached(bool a)
{
	attached = a;
}

bool Part::get_attached()
{
	return attached;
}

void Part::set_resting(bool r)
{
	resting = r;
}

bool Part::get_resting()
{
	return resting;
}
```

MovingPart.h and Movingpart.cpp: Dynamic haptic object used for haptic moving part

MovingPart.h

```
//==========================================================================
//   Filename : MovingPart.h
//      Based on Dice Demo RigidBodyDice class
//      Written on July 16th, 2000
//      By Greg Edwards
/*********************************************************************/
/* This is the declaration of a dynamic node. One                        */
/* dynamic node is used in the simulation and assigned                     */
/* to the currently moving part such that reaction force                 */
/* can be sent to the phantom from the movingpart.                         */
/* Collisions of the movingpart are first detected visually since   */
/* the program must be able to run without haptics as well, then    */
/* control is sent to the updateDynamics code of the dynamic node   */
/* for faster updates.  If did not have to worry about multiple     */
/* scenarios with and without haptics, I would have the haptic code */
/* detect the collision as well and have the haptics control the   */
/* graphics.
             */
/*********************************************************************/
//==========================================================================

#ifndef _MOVING_PART_H
#define _MOVING_PART_H

#include "collision4.h"

#define DEFAULT_MASS       0.2    //kg
#define      DEFAULT_DAMPING     1 //N/(mm/s)

#define GRAVITY     -2.0   //acceleration : N /kg (9.8 is too fast)

//*****************************************************
//                                Class: MovingPart
//*****************************************************

class MovingPart : public gstRigidBody
{
       public:
              MovingPart();
              virtual void updateDynamics();
              void fixRotationalDynamics();
              void setGravity(gstBoolean bEnable);
              gstBoolean isGravityOn() const
              {
                     return (gravity.norm() > 0);
              }

              void addForce(gstVector &force_WC);
              gstVector getCollisionForce_WC();
              //New function to get collision information from collision object
              /*void set_collision(Collision* c)
              {
                     collide = c;
              } */
              void setGrabMode(gstBoolean state)
              {
                     m_bInGrabMode = state;
              }
              void setForce(gstVector* f)
              {
                     force = f;
```

```
		}
		void setDepth(double* d)
		{
			depth = d;
		}
		void setNormal(gstVector* n)
		{
			normal = n;
		}

		void setbcollide(gstBoolean c)
		{
			b_collide = c;
		}

	protected:
		gstVector m_collisionTorque;
		gstVector m_collisionForce;
		//Collision* collide;
		gstBoolean m_bInGrabMode;
		gstBoolean b_collide;
		float m_nProportional;
		float m_nDifferential;
		gstVector* force;
		double* depth;
		gstVector* normal;

};

#endif
```

MovingPart.cpp

```
//=============================================================================
//   Filename : MovingPart.h
//      Based on Dice Demo RigidBodyDice class
//      Written on July 16th, 2000
//=============================================================================

#include "MovingPart.h"

MovingPart::MovingPart()
{
	setMass(DEFAULT_MASS);
	setDamping(DEFAULT_DAMPING);
	m_bInGrabMode = FALSE;
	b_collide = FALSE;
	m_nProportional = PROPORTIONAL;
	m_nDifferential = DIFFERENTIAL;
	force = NULL;
	depth = NULL;
	normal = NULL;

}


void MovingPart::setGravity(gstBoolean bEnable)
{
	if (bEnable)
	{
		gravity.init(0.0, GRAVITY, 0.0);
		addToDynamicList();
```

```
	}
	else
		gravity.init(0.0,0.0,0.0);
}

void MovingPart::addForce(gstVector &force_WC)
{
	gstVector newReactionForce = fromWorld(force_WC);
	reactionForce += newReactionForce;
	addToDynamicList();
}

gstVector MovingPart::getCollisionForce_WC()
{
	gstVector force_WC = toWorld(m_collisionForce);
	return force_WC;
}

//***************************************************************************
// Member Function : updateDynamics
// --------------------------------
// Uses the gstRigidBody parent class to provide the physics for movement and
// force reaction. This routine determines collisions with the bounding
// environment and applies the appropriate force
//***************************************************************************
void MovingPart::updateDynamics(void)
{
	gstVector move;
    static double local_depth = 0;
	double pointVelocity;
	static gstVector intern_force;
	gstVector current_position;
	static gstVector previous_position;
	static gstVector local_normal;


	//*************************************************************************
********
	//collision detection done by graphics alone is too slow.
	//this code will supplement the graphics version by trying to do some quick
tests
	//to see if the object is moving away from the collision or further in.
	//then setting the force appropriately instead of waiting for the graphics to
realize
	//that the collision is over (which takes too long).
	//*************************************************************************
********
	if (m_bInGrabMode)  //only add forces while being manipulated...don't want
forces accumulating when resting on something
	{
		if (force->norm() > 0)  //if hitting something
		{
			current_position  = getPosition_WC();  //get current position of
haptic object
			if (b_collide)  //if already in a collision then ignore graphics
and do some calcs
			{
				move = current_position - previous_position;
				local_depth += local_normal.dot(move);
				if (local_depth > 0)
				{
					pointVelocity = move.norm() / getDeltaT();  //norm()
gets magnitude
```

```
                                intern_force = (-1 * (local_normal)) *
(m_nProportional * local_depth +

      1.0/1000 * m_nDifferential * pointVelocity);
                        }
                        else
                                intern_force.init(0,0,0);
                  }
                  else  //just started colliding with object
                  {
                        //get initial depth and force here
                        local_depth = *depth;
                        local_normal = *normal;
                        intern_force = (-1 * (local_normal)) * (m_nProportional *
local_depth);
                        b_collide  = TRUE;
                  }

                  previous_position = current_position;
            }
            else  //not hittting anything
            {
                  b_collide = FALSE;
                  intern_force = *force;  //setting to zero
            }
            addForce(intern_force);

      }

    gstRigidBody::updateDynamics();
}

//***************************************************************************
// Member Function : fixTransformMatrix
// ---------------------------------
// The gstRigidBody class uses a quaternion to modify the orientation of
// the object. This quaternion must be updated to match the current transform
// matrix.
//***************************************************************************
void MovingPart::fixRotationalDynamics() {
    double rotationMatrix[3][3];

    // hack : need to set the current RigidBody quaternion to correspond with the
objTransf
    objTransf.getRotationMatrix(rotationMatrix);
    matrixToQuaternion(rotationMatrix, q);

    // set the angular inertia to be zero
    L.init(0.0,0.0,0.0);
}
```

RigidManipulator.h and RigidManipulator.cpp: Used to attach haptic object to PHANToM

Rigidmanipulator.h

```
//==============================================================================
//     This file was modified from the Sensable dice demo.  It's purpose is
// to move the haptic object and allow reaction forces to be sent to the
// the phantom from the dynamic node (movingpart.h)
//
```

```
//   Filename : RigidManipulator.h
// Written by : Brandon Itkowitz (bitkowitz@sensable.com) and Greg Edwards
//    Project : Dice Demo
//     Module : Rigid Body Manipulator
// -----------------------------------------------------------------------------

#ifndef RIGID_MANIPULATOR_H
#define RIGID_MANIPULATOR_H

#include "collision4.h"
#include <gstManipulator.h>
#include <gstSeparator.h>
#include "MovingPart.h"

//*****************************************************************************
//                             C o n s t a n t s
//*****************************************************************************

#define EXPONENTIAL_RISE_TC         0.5   // exponential rise time constant

#define MAX_GAIN_SCALE              0.75  // scale back the max gain to avoid buzzing

#define MAX_TORQUE                  5.0
#define TORQUE_MAGNIFICATION_X      1.5
#define TORQUE_MAGNIFICATION_Y      1.5
#define TORQUE_MAGNIFICATION_Z      1.0

//*****************************************************************************
//                  Class : R i g i d M a n i p u l a t o r
//*****************************************************************************
class RigidManipulator : public gstManipulator
{

public:

    RigidManipulator() {
        m_bGrabbed = FALSE;
    }

    ~RigidManipulator(){};

    virtual gstVector calcManipulatorForce(void *PHANToM) {
       if (PHANToM) {}     // To remove compiler warning
       return gstVector(0.0,0.0,0.0);
    }

    virtual gstVector calcManipulatorForce(void *PHANToM, gstVector &torque);

    // don't disable phantom touch of the manipulated node
    virtual void start() {
        active = TRUE;
       }

    gstBoolean isGrabbed() {
        return m_bGrabbed;
    }

protected:

    gstBoolean m_bGrabbed;
       gstVector offset;

    float absValue(float value) {
```

```
        return ((value < 0) ? -value : value);
    }

};

#endif
```

RigidManipulator.cpp

```
//==============================================================================
// Modified version of dice demo manipulator.  This version is only used to
// send forces to the Phantom (no motion control).
// Greg Edwards, July 11, 2000
//==============================================================================

#include "RigidManipulator.h"

//******************************************************************************
// Member Function : calcManipulatorForce
// --------------------------------------
// Modify the transform matrix of the dynamic separator, affecting the
// translation and orientation of the die.
//******************************************************************************
gstVector RigidManipulator::calcManipulatorForce(void *PHANToM, gstVector &torque)
{
    gstPHANToM *myPhantom = (gstPHANToM *) PHANToM;
    MovingPart *dynamicNode = (MovingPart *) manipNode;
       gstVector reactionForce;
    gstVector tempVector;
    float distance;
    static double timer;    // the timer for performing exponential rise
    gstPoint transformAngles;
    static gstTransformMatrix transformMatrix, rotationMatrix;
    if (active && !m_bGrabbed && myPhantom->getStylusSwitch()) {
        m_bGrabbed = TRUE;
        timer = 0;  // reset the timer
             dynamicNode->setGrabMode(TRUE);
             dynamicNode->setGravity(TRUE);
             dynamicNode->setbcollide(FALSE);
             offset = dynamicNode->getPosition_WC() - myPhantom->getPosition_WC();
//gets original grabbing offset
        // setup the rotation matrix
        gstTransformMatrix phantomTransform, objectTransform;
        myPhantom->getTransformMatrix().getRotationMatrix(phantomTransform);
        dynamicNode->getTransformMatrix().getRotationMatrix(objectTransform);

        phantomTransform.inverse();
        mulM(rotationMatrix, objectTransform, phantomTransform);
    }

    if (active && m_bGrabbed && !myPhantom->getStylusSwitch()) {
        myPhantom->stopManipulator();
        dynamicNode->setGrabMode(FALSE);
             dynamicNode->setGravity(FALSE);
        m_bGrabbed = FALSE;
    }

    if (!active || !m_bGrabbed) {
        torque.init(0.0, 0.0, 0.0);
        return gstVector(0.0, 0.0, 0.0);
    }
```

```
    //-------------------------------------------------------------------------
    // reactionForce is result of a spring force between the current PHANToM location
    // and location of the dynamicNode
    tempVector = (dynamicNode->getPosition_WC() - myPhantom->getPosition_WC()) -
offset;
       distance = tempVector.norm();
    reactionForce = tempVector.normalize() * (myPhantom->getMaxGain() * MAX_GAIN_SCALE
* distance);

    if (timer < 5 * EXPONENTIAL_RISE_TC)
        reactionForce *= (1.0 - exp(-timer / EXPONENTIAL_RISE_TC));

    // add this force to the dynamicNode
    dynamicNode->addForce(-reactionForce);
    //-------------------------------------------------------------------------
    // rotationally align the part with the phantom
    mulM(transformMatrix, rotationMatrix, myPhantom->getTransformMatrix());
    transformMatrix.getRotationAngles(transformAngles);

    dynamicNode->setRotateDynamic(gstVector(1.0, 0.0, 0.0), transformAngles[0]);
    dynamicNode->rotateDynamic(gstVector(0.0, 1.0, 0.0), transformAngles[1]);
    dynamicNode->rotateDynamic(gstVector(0.0, 0.0, 1.0), transformAngles[2]);

    // make sure the change in rotation is in sync with the block dynamics
    dynamicNode->fixRotationalDynamics();

    // increment the time counter
    if (timer < EXPONENTIAL_RISE_TC * 5)
        timer += myPhantom->getDeltaT();

    return reactionForce;
}
```

Collision4.h and Collision7.cpp: Graphical collision detection

Collision4.h

```
#ifndef _COLLISION_
#define _COLLISION_

//==========================================================================
//   Filename : collision4.h
//      Based on Dice Demo RigidBodyDice class
//      Written in August, 2000
/*************************************************************/
/* THe basic algorithm uses ray casting from the vertices of the    */
/* moving object as well as from all objects in the scene. Any rays */
/* that collide closer than the planned motion create a collision.  */
/* The smallest ray collision indicates the first vertex to hit.    */
/* Movement after is the left over vector projected onto the plane  */
/* of the polygon that was hit.  A second bounding box and polygon  */
/* intersection combination test is also done since the rays miss   */
/* some collisions.  To save time, this extra test is only done on  */
/* essential parts...i.e. the ones it connects to.                  */
/* Special thanks to Rory Jackson BSc., Software Developer for      */
/* Logicom Virtual Worlds, for the ray shooting from the vertices   */
/* idea     .                                                       */
/*************************************************************/
//==========================================================================

#include "wt.h" //for world tool kit
#include <fstream.h>
```

```
#include <string.h>
#include "wtcpp.h" //to use WTK C++ classes
#include "parts3.h" //has part class with model holder and data containers
#include <math.h>
#include <gstBasic.h>
#include <gstSeparator.h>
#include <gstPHANToM.h>
#include <gstNode.h>
#include <gstDynamic.h>  //not sure if needed
#include <gstConstraintEffect.h> //not sure if needed
#include <gstRigidBody.h>  //not sure of needed
#include <gstTransformMatrix.h>

#define PROPORTIONAL 1.5;
#define DIFFERENTIAL 3.0;

class Collision
{
	private:
		//calculation variables
		double speed;
		double depth;
		WtP3 wtkforce;  //used to set in WTK before gst
		gstVector force;
		WtP3 intersect_normal;
		int ind;
		int count;
		WtP3 dir;
		WtP3 tempvector;
		WtNode* i_node;
		WtP3 midpoint;
		WtQ midq;
		WtP3 temp_pos;
		WtQ temp_q;
		WtPQ temp_frame;
		WtPQ temp_frame2;
		WtPQ inter_frame; //reference frame for the object in hand
		WtPQ col_frame;  //reference frame for the part to be connected to
		WtQ current_q;  // current orientation for this time frame
		WtP3 current_pos;
		WtP3 previous_pos;  //to keep track of last position of the moving part
		WtQ previous_q;
		WtP3 scene_pos;
		WtP3 movevector;  //to keep track of the movement of the part over time
		double movedistance;
		double moveturn;
		WtP3 vert_pos[40];  //holds vertex positions for object before casting
rays from them
		WtP3* v_pos[6][40];  //pointers to vertex positions of all the objects
		int numvertices;  //keeps track of number of vertices in current object
being traversed
		float *distance; //temporary variable used to get each ray casting
collision distance
		WTpoly* poly;
		WTpoly* hitp;
		WtNode* tempn;
		WtNode* hitn;
		//char* tempname;
		//char* internname;
		WtP3 contact_pos;
		WtP3 transvec;  //translation vector of phantom position (parent of
moving node...moving parent moves child)
		WtQ contact_q;
```

```
            //made it a separate entity from the part because it crashed when added
to part for some reason
            //the part class or the struct would not accept a third WtP3.  Kept
giving Node Type 0 not updated, or Fatal parent
            //link error.
            WtP3 localpos;  //for phantom
            WtQ localq;
            gstPoint angles[6];
            int lasthit;  //records index of last hit to compare to index of current
hit (curhit)
            int curhit;  //if the same then don't replay sound or record a new hit

            //variables received from main function
            WtRoot* r;
            WtNodePath *i_path;
            Part* c_parts[6];
            WtNodePath* c_paths[6];
            WtMovable* tab;
            WtMovable* wal;
            int maxindex;
            double maxdist;
            WtMovable* parent;
            WtNodePath* parent_path;
            //variables to hold results of collision detection
            double min_d;
            WtP3 minvert_p;
            WtNodePath* hitpath;
            WtNodePath* hitpath_array[10];
            WtNodePath* temp_hitpath;
            int normal_negate;

            gstPHANToM *phantom;
            gstSeparator* haptic_parts[6];
            gstTransformMatrix* rotationMatrix, objectTransform, transformMatrix;
//for synchronizing rotation of chosen object
            gstVector g_normal;

            WtP3 haptic_p[6];  //holds initial haptic object position and rotation
for later check_attach() use
            double haptic_r[6];
            bool hit;

            WtP3 local;
            WtP3 local_current;
            //sound stuff
            bool snd_on;
            WTsound* snd[4];
            int hitcount;


      public:
            Collision();
            ~Collision();
            void init(WtRoot* rt, Part* cu_parts[], WtNodePath* cu_paths[], int
mxindex, double mxdist, gstPHANToM* phantom, gstSeparator* haptic_parts[],
gstTransformMatrix* rm, WtP3 h_p[], double h_r[], WtMovable* t, WtMovable* w, WtP3*
ve_pos[][40]);
            void manage_collision(WtNodePath* inter_path);
            void ray_check();
            void check_position();
            void slide();
            WtNodePath* check_attach(ofstream* d);
            void set_position(WtNodePath*);
```

```
            void set_localpos(WtP3 pos);
            WtP3 get_localpos();
            void set_localq(WtQ q);
            WtQ get_localq();
            gstPoint get_previous_hrotation(int i);
            void set_previous_hrotation(gstPoint a, int i);
            void set_phantom_geom(WtMovable*);
            gstVector* get_phantom_force();
            gstVector* get_normal();
            double* get_depth();
            void set_sound(bool s_on, WTsound* s[]);
            int get_hitcounter();  //returns the total number of collisions to date

};

#endif
```

Collision7.cpp

```
#include "collision4.h"

const int DTolerance = 4;  //distance tolerance for connecting
const float QTolerance = 0.15f; // orientation/(quaternion component) tolerance for
connecting
WtTask *start_flash = NULL;

//Note that all changes in motion are applied to the phantom cursor.  This changes the
position of the object
//since the object is attached to the cursor.  All calculations however are based on
the position of the object.
//This is why current and previous position are kept for the object (for
calculations), but the local position
// variables are kept for the cursor (for repositioning).

//task function used as a callback for flashing two connecting objects with a
different color (used for visual cues of attaching/detaching)
void flash(void* node)
{
       static int v_count = 0;
       static int matid[30];
       static int matid2[30];
       int z = 0;
       static WTpoly* temp_poly = NULL;
       static WtNode* temp_node = NULL;
       static WTpoly* first_poly = NULL;
       static WTpoly* first_poly2 = NULL;
       switch(v_count)
       {
       case 0:
              //get original color for part
              temp_poly = ((WtGeometry*)node)->GetFirstPoly();
              first_poly = temp_poly;
              while(temp_poly != NULL)
              {
                     matid[z] = WTpoly_getmatid(temp_poly);
                     temp_poly = WTpoly_next(temp_poly);
                     z++;
              }
              //set new color
              ((WtGeometry*)node)->SetRGB(255,255,255);
```

```
            //now get original colors for part connecting to
            z = 0;
            temp_node = ((WtMovable*)node)->GetParent(0);
            temp_poly = ((WtGeometry*)temp_node)->GetFirstPoly();
            first_poly2 = temp_poly;
            while(temp_poly != NULL)
            {
                  matid2[z] = WTpoly_getmatid(temp_poly);
                  temp_poly = WTpoly_next(temp_poly);
                  z++;
            }
            ((WtGeometry*)temp_node)->SetRGB(255,255,255);
            v_count++;
            break;
      case 2:
            //for first object
            temp_poly = first_poly;
            while(temp_poly != NULL)
            {
                  WTpoly_setmatid(temp_poly, matid[z]);  //a is reset to zero with
iteration of action loop so this is ok
                  temp_poly = WTpoly_next(temp_poly);
                  z++;
            }
            //for second object
            z = 0;
            temp_poly = first_poly2;
            while(temp_poly != NULL)
            {
                  WTpoly_setmatid(temp_poly, matid2[z]);  //a is reset to zero with
iteration of action loop so this is ok
                  temp_poly = WTpoly_next(temp_poly);
                  z++;
            }
            v_count++;
            break;
      case 4:
            ((WtGeometry*)node)->SetRGB(255,255,255);
            ((WtGeometry*)temp_node)->SetRGB(255,255,255);
            v_count++;
            break;
      case 6:
            //for first object
            temp_poly = first_poly;;
            while(temp_poly != NULL)
            {
                  WTpoly_setmatid(temp_poly, matid[z]);  //a is reset to zero with
iteration of action loop so this is ok
                  temp_poly = WTpoly_next(temp_poly);
                  z++;
            }
            //for second object
            z = 0;
            temp_poly = first_poly2;
            while(temp_poly != NULL)
            {
                  WTpoly_setmatid(temp_poly, matid2[z]);  //a is reset to zero with
iteration of action loop so this is ok
                  temp_poly = WTpoly_next(temp_poly);
                  z++;
            }
            v_count++;
            break;
```

```
	case 8:
		v_count = 0;
		delete start_flash;  //and remove task here as well
		start_flash = NULL;
		break;
	default:
		v_count++;
	} //end of switch
}

Collision::Collision()
{
	//variable used to negate normal when using polygon surface of other objects
	//want it to be negative for other object polygons and positive for same object
polygons
	normal_negate = -1;
	g_normal.init(0,0,0);
	tab = NULL;
	wal = NULL;
	ind = 0;  //index of part
	speed = 0;
	depth = 0;
	i_node = NULL;
	movedistance = 0.0f;
	moveturn = 0.0f;
	numvertices = 0;
	distance = new float(0.0f);
	poly = NULL;
	hitp = NULL;
	tempn = NULL;
	hitn = NULL;
	//tempname = new char[8];
	//tempname[0] = NULL;
	//intername=new char[8];
	//intername[0]=NULL;
	parent = NULL;
	parent_path = NULL;
	force.init(0,0,0);

	i_path = NULL;

	for (int i=0; i<10;i++)
	{
		hitpath_array[i] = NULL;
	}

	//variables to hold results of collision check
	min_d = 0;
	hitpath = NULL;
	temp_hitpath = NULL;

	temp_pos.Initialize();
	temp_q.Initialize();
	intersect_normal.Initialize();

	localpos.Initialize();
	localq.Initialize();
	inter_frame.Initialize();
	col_frame.Initialize();
	temp_frame.Initialize();
	transvec.Initialize();
	for (int j=0;j<6;j++)
	{
```

```
		angles[j].init(0,0,0);
	}

	for (j=0;j<4;j++)
	{
		snd[j] = NULL;
	}
	snd_on = false;
	start_flash = NULL;
	lasthit = 99;
	curhit = 99;
}

Collision::~Collision()
{

}

void Collision::set_sound(bool s_on, WTsound* s[])
{
	snd_on = s_on;
	for (int i=0; i<4;i++)
	{
		snd[i] = s[i];
	}
}

int Collision::get_hitcounter()
{
	return hitcount;
}

void Collision::init(WtRoot* rt, Part* cu_parts[], WtNodePath* cu_paths[], int
mxindex, double mxdist, gstPHANToM* p, gstSeparator* h_parts[], gstTransformMatrix*
rm, WtP3 h_p[], double h_r[], WtMovable* t, WtMovable* w, WtP3* ve_pos[][40])
{
	hitcount =0;
	wal = w;
	tab = t;
	phantom = p;
	rotationMatrix = rm;
	r = rt;
	for (int i=0; i<6;i++)
	{
		c_parts[i] = cu_parts[i];
		c_paths[i] = cu_paths[i];
	}

	for(int g=0;g<6;g++)
	{
		haptic_parts[g] = h_parts[g];
		haptic_p[g] = h_p[g];
		haptic_r[g] = h_r[g];
	}

	maxindex = mxindex;
	maxdist = mxdist;

	//setup array of pointers to all vertex positions.  Actual objects reside in
the haptics_main function.

	for (int b=0;b<6;b++)
	{
```

```
		for (int c=0;c<40;c++)
		{
			v_pos[b][c] = ve_pos[b][c];
		}
	}

}

//positions of the phantom cursor!!  The object is attached to the cursor and will
move with it.

void Collision::set_localpos(WtP3 apos)
{
	localpos = apos;
}

WtP3 Collision::get_localpos()
{
	return localpos;
}

void Collision::set_localq(WtQ aq)
{
	localq = aq;
}

WtQ Collision::get_localq()
{
	return localq;
}


//create similar function for haptic

gstPoint Collision::get_previous_hrotation(int i)
{
	return angles[i];
}

void Collision::set_previous_hrotation(gstPoint a, int i)
{
	angles[i] = a;
}


void Collision::set_phantom_geom(WtMovable* pg)
{
	parent = pg;
	parent_path = new WtNodePath(parent,r,0);
}

void Collision::manage_collision(WtNodePath* inter_path)
{
	hit = false;
	depth = 0;

	//set variables in local object to be used by functions.  inter_path is path of
selected object
	hitpath = NULL; //set to NULL at beginning...if it gets set then there was a
collision
	i_path = inter_path;
	i_node = i_path->GetNode(i_path->NumNodes()-1);
	ind = ((Part*)i_node)->get_index();
```

```
	c_paths[ind] = i_path; //update path list if path changed as a result of
attaching

	//get current position for movement vector in world coordinates...need midpoint
for this
	((WtGroup*)i_node)->GetMidpoint(midpoint);
	((Part*)i_node)->GetOrientation(midq);
	i_path->GetTranslation(temp_pos);
	i_path->GetOrientation(temp_q);
	temp_frame.Set(temp_pos, temp_q);
	midpoint.Local2WorldFrame(temp_frame, current_pos);
	midq.Local2WorldFrame(temp_frame, current_q);

	previous_pos = ((Part*)i_node)->get_last_pos();
	previous_q = ((Part*)i_node)->get_last_q();

	ray_check();  //only does ray casts
	if (hitpath !=NULL)
		slide();
	else
	{
		//if no collision use transvec that was set in ray check...just check for
size
		if (transvec.Mag() > 4)
		{
			//make maximum length of 4
			transvec.Norm();
			transvec = transvec * 4.0f;
			//now reset current position to show this reduction in the
translation vector
			current_pos = previous_pos + transvec;  //current_pos is used in
the set_position function
			//this is no longer exact since transvec is for parent and
applying it to child here
			//close enough though since only using for collision detection ray
directions
		}
	}
	check_position();
	//set position and make current position the previous position for the next
iteration
	set_position(i_path);

}


inline void Collision::ray_check()
{
	min_d = 0;
	hitp = NULL;
	minvert_p.Initialize();  //not used now but looked like it might be useful
later, vertex position
	hitn = NULL;
	unsigned char red, green, blue;

	//get previous position for movement vector, which is stored in world
coordinates
	movevector = current_pos - previous_pos;  //get the movement vector
	dir = movevector;
	dir.Norm();  //vector for ray casts
	moveturn = current_q.GetAngle() - previous_q.GetAngle(); //get the change in
rotation in radians
```

```
	movedistance = movevector.Mag();
	min_d = movedistance;  //set for ray casts to have something to initially
compare to
	//if collision distance greater than movedistance then no collision in this
timestep

	//add angle checks too...|| moveturn != 0 and have a movement vector for pure
rotation!
	if (movedistance !=0 || moveturn !=0)  //if no movement then don't do any
checking
	{
		//get last local position
		local = this->get_localpos();

		//now set object back to last position so can send rays from it and so
other objects will
		//send rays to it in the correct position.
		//if the first time then local position will be zero and should be
updated to a value:
		if ((local[X] == 0) && (local[Y] == 0) && (local[Z] == 0))//hasn't been
set yet
		{
			((WtMovable*)parent)->GetTranslation(local);
		}

		((WtMovable*)parent)->GetTranslation(local_current);  //get current
position in local coordinates for later use
		//***********************************
		transvec = local_current - local;  //setting translation vector with
phantom position
		//***********************************
		((WtMovable*)parent)->SetTranslation(local);


		//*****STEP 1 check ray casts
************************************************
		//* send out raycasts from each vertex of the selected object in the
direction of *
		//* the movement vector to check for collisions
*

	//*************************************************************************
****

		if (movedistance !=0)  //if moving and not rotating only then do ray
casts
		{
				i_path->GetTranslation(temp_pos);  //needs to redefine
temp_frame since it has changed since beginning of function
				i_path->GetOrientation(temp_q);
				temp_frame.Set(temp_pos, temp_q);

				WTvertex* tempvertex;
				tempvertex = ((WtGeometry*)i_node)->GetFirstVertex();
				numvertices = 0;
				while (tempvertex !=NULL)
				{
					//get vertex position
					((WtGeometry*)i_node)->GetVertexPosition(tempvertex,
tempvector);
					//convert to world frame
					tempvector.Local2WorldFrame(temp_frame,
*v_pos[ind][numvertices]);
```

```
                                numvertices++;
                                tempvertex = WTvertex_next(tempvertex);
                        }

                    //now cast rays from vertices of moving object and record the
smallest collision potential
                        for(int i = 0; i<numvertices;i++)
                        {
                                WtNodePath* temppath=new WtNodePath;
                                //had to create object since rayintersect won't take
a null path
                                //vert_pos must be in root coordinates
                                poly = r->RayIntersect(dir, *v_pos[ind][i], distance,
&temppath);
                                if (poly != NULL)
                                {
                                        tempn = temppath->GetNode(temppath-
>NumNodes()-1);
                                        WTpoly_getrgb(poly,&red,&green,&blue);
                                        if (((Part*)tempn)->get_index() != ind &&
!(red ==255 && green ==0 && blue==0)) // if not itself or the cursor then count as hit
                                        {
                                                if (*distance < min_d)  //if ray
collision distance smaller than mindistance
                                                {
                                                        min_d = *distance;  //set
necessary information for collision reaction
                                                        minvert_p = *v_pos[ind][i];
                                                        hitp = poly;
                                                        if (hitpath!=NULL)
                                                                delete hitpath;  //get rid
of old object to free up memory and not have stranded object
                                                        hitpath = temppath;
                                                        hitn = hitpath->GetNode(hitpath-
>NumNodes()-1);
                                                        curhit = ((Part*)hitn)-
>get_index();  //get index of object being hit
                                                        //cout<<"***The closest hit node
when rays from moving object is "<<hitn->GetName()<<endl;
                                                        normal_negate = -1;
                                                }
                                        }
                                }
                                else
                                        delete temppath;

                        }

                        //*****************************STEP
2*****************************************
                        //* send out raycasts from each vertex of the objects in
the scene in the inverse *
                        //* direction of the movement vector to check for
collisions with their vertices. *

      //*************************************************************************
****
                        for (i=maxindex; i>=0; i--)  //only checking for collision
with parts at the moment
                        {
                                //to reduce the number of objects to test only do for
those that are within
                                //possible hitting distance
```

```
                                    c_paths[i]->GetTranslation(temp_pos);
                                    c_paths[i]->GetOrientation(temp_q);
                                    temp_frame.Set(temp_pos, temp_q);
                                    if ((i != ind) && (c_paths[i] !=NULL))
                                    {
                                          ((WtGroup*)c_parts[i])->GetMidpoint(midpoint);

      midpoint.Local2WorldFrame(temp_frame,scene_pos);

                                          tempvector = current_pos - scene_pos;
                                          *distance= (float)(tempvector.Mag());

                                          if (*distance < maxdist)
                                          {
                    //now cast rays from vertices of stationary object in opposite
direction and record the smallest collision potential
                                                for(int k=0;v_pos[i][k] != NULL;k++)
                                                {
                                                      //CHANGED THIS LINE FROM null TO
NEW NODEPATH
                                                      WtNodePath* temppath=new
WtNodePath;
                                                      poly = r->RayIntersect(-1*dir,
*v_pos[i][k], distance, &temppath);
                                                      if (poly !=NULL) //if an object
gets hit
                                                      {
                                                            //check to see if it is
the moving object
                                                            tempn = temppath-
>GetNode(temppath->NumNodes()-1);
                                                            if (((Part*)tempn)-
>get_index() == ind) // if it is the moving object then                    {
                                                            {
                                                                  if (*distance <
min_d)  //if ray collision distance smaller than mindistance
                                                                  {
                                                                        min_d =
*distance;  //set necessary information for collision reaction
                                                                        //minvert_p =
vert_pos[k];
                                                                        minvert_p =
*v_pos[i][k];
                                                                        hitp = poly;
                                                                        hitpath =
c_paths[i];
                                                                        hitn =
c_parts[i];
                                                                        curhit = i;
//get index of object being hit
                                                                        normal_negate
= 1;
                                                                  }
                                                            }
                                                      }
                                                      else
                                                            delete temppath;
                                                }
                                          }  //end of if distance smaller than
maxdistance
                                    }     //end of if not itself
                              }  //end of for loop iterating through parts
```

```
                    } //end of section that is done only for translation (if
movedistance !=0)

       }  //end of if movedistance is not 0
       //cout<<"End of collision detection"<<endl;

}

//slides the object along the polygon of the hit object
//first calculates the contact point of the closest vertex and then
//projects the remaining vector onto the polygon plane.  At the end of this
//function, current position will be ideal projected position after slide and
//previous position will be set to contact position.  Thus, after this
//a set position must be done and then another collision check for the slide portion.

inline void Collision::slide()
{

       WtP3 normal;
       WTp3 temp; //old C format to get data from polygons
       WtP3 remain;
       WtP3 projection;

       WTpoly_getnormal(hitp, temp);

       //put into C++ format
       normal[X] = temp[X];
       normal[Y] = temp[Y];
       normal[Z] = temp[Z];
       normal = normal_negate * normal;  //want inwards pointing normal.
       g_normal.init(normal[X], -normal[Y], -normal[Z]);
       //now get the contact position by normalizing the movement vector and
multiplying
       //by the hit distance
       contact_pos = movevector;
       contact_pos.Norm();

       if (min_d < 0.01)
              min_d = 0;
       else
              min_d = min_d *0.8;

       contact_pos = min_d * contact_pos;  //multiply by the contact distance
       contact_pos = contact_pos + previous_pos;  //this is now the contact position
of the midpoint
       //for now let contact rotation just be the previous rotation
       contact_q = previous_q;

       //Now get the projection of the remaining vector onto the face of the polygon
       //equation = remainder vector - proj of remainder vector onto normal of polygon
       //actually we're getting a projection vector onto a plane parallel to the face
of the
       //polygon and passing through the midpoint of the moving object.

       //remainder from midpoint at contact position to end of vector
       remain = current_pos - contact_pos;
       depth = Dot(remain,normal);
       //projection onto plane parallel to polygon, but passing through midpoint of
moving object
       projection = remain - (depth * normal);

       //modified from 1
       if (projection.Mag() > 1)
```

```
		projection.Norm();

	//now set final position of midpoint after slide
	current_pos = projection + contact_pos;
	//for testing without slide: ***************
	//current_pos = contact_pos;
	//*****************************
	transvec = current_pos - previous_pos;
	if (haptic_parts[ind] != NULL)
		force.init(1,1,1);

	if (curhit != lasthit  && curhit != 99)  //if this is an original hit
	{
		hitcount++;  //increases hit counter
		if(snd_on)
		{
			if (!WTsound_isplaying(snd[1]) && !WTsound_isplaying(snd[2]) &&
!WTsound_isplaying(snd[3]))
			{
				WTsound_stop(snd[0]);
				WTsound_play(snd[0]);
			}
		}
	}
}

//Ray casts not perfect so check with polygon intersection for specific blocks for
each object

inline void Collision::check_position()
{
	temp_hitpath = NULL;
	count=0;
	((WtMovable*)parent)->Translate(transvec, WTFRAME_PARENT);  //move to new
potential place
	//these only check for those parts that the rays will not always detect
	int m = 0;
	switch(ind)
	{
	case 5:
		temp_hitpath = i_path->IntersectNode(c_parts[0], 0);  //only checking
with part 1 at the moment
		if(temp_hitpath !=NULL)  //if it collides with the node being checked
		{
			hitpath_array[count] = new WtNodePath(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1),r,0) ;
			count++;
		}
		break;

	case 4:  //check with parts 3 and 4
		for (m=3; m>=2; m--)  //only checking for collision with parts 6 and 5 at
the moment
		{
			temp_hitpath = i_path->IntersectNode(c_parts[m], 0);  //only
checking with other parts at the moment
			if(temp_hitpath !=NULL)  //if it collides with the node being
checked
			{
				hitpath_array[count] = new WtNodePath(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1),r,0) ;
				count++;
			}
```

```
            }
            break;
/*     case 5:
            temp_hitpath = i_path->IntersectNode(c_parts[2], 0);  //only checking
with part 3 at the moment
            if(temp_hitpath !=NULL)  //if it collides with the node being checked
            {
                  hitpath_array[count] = new WtNodePath(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1),r,0) ;
                  count++;
            }
            break; */
      case 3:
            for (m=2; m>=1; m--)  //only checking for collision with parts 3 and 2 at
the moment
            {
                  temp_hitpath = i_path->IntersectNode(c_parts[m], 0);  //only
checking with other parts at the moment
                  if(temp_hitpath !=NULL)  //if it collides with the node being
checked
                  {
                        hitpath_array[count] = new WtNodePath(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1),r,0) ;
                        count++;
                  }
            }
            break;
      case 2:
            for (m=4; m>=1; m--)  //only checking for collision with parts 5,4 and 2
at the moment
            {
                  if (m ==2)
                        m--;
                  temp_hitpath = i_path->IntersectNode(c_parts[m], 0);  //only
checking with part 5 and 2 at the moment
                  if(temp_hitpath !=NULL)  //if it collides with the node being
checked
                  {
                        hitpath_array[count] = new WtNodePath(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1),r,0) ;
                        count++;
                  }
            }
            break;
/*     case 2:
            for (m=4; m>=1; m--)  //only checking for collision with parts 5,4 and 2
at the moment
            {
                  if (m ==2)
                        m--;
                  temp_hitpath = i_path->IntersectNode(c_parts[m], 0);  //only
checking with part 2 at the moment
                  if(temp_hitpath !=NULL)  //if it collides with the node being
checked
                  {
                        hitpath_array[count] = new WtNodePath(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1),r,0) ;
                        count++;
                  }
            }
            break; */
      case 1:
```

```
            temp_hitpath = i_path->IntersectNode(c_parts[0], 0);  //only checking
with part 1 at the moment
            if(temp_hitpath !=NULL)  //if it collides with the node being checked
            {
                  hitpath_array[count] = new WtNodePath(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1),r,0) ;
                  count++;
            }
            break;
      case 0:
            break;
      default:
            break;
      }

            //add wall and table at end of the whole switch!!
            //wall
            temp_hitpath = i_path->IntersectNode(wal, 0);  //change part to wall
pointer
            if(temp_hitpath !=NULL)  //if it collides with the node being checked
            {
                  hitpath_array[count] = new WtNodePath(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1),r,0) ;
                  count++;
            }

            temp_hitpath = i_path->IntersectNode(tab, 0);  //change part to table
pointer
            if(temp_hitpath !=NULL)  //if it collides with the node being checked
            {
                  hitpath_array[count] = new WtNodePath(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1),r,0) ;
                  count++;
            }
//**************STEP 2: Polygon intersection with bounding box collisions

            if (count != 0)   //if there is a bounding box collision check to see if
polygon intersection
            {
                  //now check for collision
                  for (m=0; m < count; m++)  //only checking for collision with
parts at the moment
                  {
                        if(i_path->IntersectPoly(hitpath_array[m]))  //if it
collides with the node being checked
                        {
                              //there may not be a node hit yet, so can't use this
                              //the node hit may not be the predominant node hit,
it just takes the first one hit
                              //when starting from the largest index downwards.
                              //hitn = hitpath_array[m]->GetNode(hitpath_array[m]-
>NumNodes()-1);
                              hit = true;  //hit something
                              if (depth ==0)
                              {
                                    hitn = hitpath_array[m]-
>GetNode(hitpath_array[m]->NumNodes()-1);
                                    curhit = ((Part*)hitn)->get_index();  //get
index of object being hit
                                    if (curhit != lasthit && curhit != 99 ) //if
there was no slide before this, and not the same as last hit
            if (curhit != lasthit  && curhit != 99)  //if this is an original hit
                                    {
```

```
                                                hitcount++;
                                                if(snd_on)
                                                {
                                                      if (!WTsound_isplaying(snd[1]) &&
!WTsound_isplaying(snd[2]) && !WTsound_isplaying(snd[3]))
                                                      {
                                                            WTsound_stop(snd[0]);
                                                            WTsound_play(snd[0]);
                                                      }
                                                }
                                          }
                                    }
                                    m = count+1;  //stops the for loop

                              }
                        }
                  } //end of if bounding box detected anything (count !=0)

                  for (int a=0;a<count;a++)
                  {
                        if (hitpath !=NULL)
                        {
                              if (((Part*)(hitpath_array[a]->GetNode(hitpath_array[a]-
>NumNodes()-1)))->get_index() != ((Part*)(hitpath->GetNode(hitpath->NumNodes()-1)))-
>get_index())
                                    delete hitpath_array[a];
                        }
                        else
                              delete hitpath_array[a];
                        hitpath_array[a] = NULL;
                  }
}


//set current position and then record it as the previous position for the next
iteration of
//the simulation
inline void Collision::set_position(WtNodePath* path)
{
      //all in terms of the midpoint. Need to get back in terms of the normal
translation coordinates.

      WtNode* node = NULL;

      node = path->GetNode(path->NumNodes()-1);

      if (hit == true)  //if collides with something put it back to previous position
      {
            temp_q = this->get_localq();  //if there was a collision then set to last
rotation
            if ((temp_q[X] == 0) && (temp_q[Y] == 0) && (temp_q[Z] == 0) &&(temp_q[W]
== 1))//hasn't been set yet
                  ((WtMovable*)parent)->GetOrientation(temp_q);

            ((WtMovable*)parent)->SetOrientation(temp_q);

            //do the same for the haptic object too
            if (haptic_parts[ind] != NULL)
            {

                  //set force to be sent to Phantom if not already set by ray
collision slide function
                  if (depth == 0)
```

```
                    {
                           //cout<<"Rays didn't detect collision"<<endl;
                           if (force.distToOrigin() == 0)  //this occurs when no
previous collision and rays didn't detect this one
                           {
                                  temp_pos = movevector;
                                  temp_pos.Norm();
                                  force.init(1, 1, 1);  //just need to set to A value
for movingPart to take over
                                  depth = 1.5;  //set standard depth for use with
MovingPart
                                  //temp_pos = temp_pos; //inward pointing normal sent
to MovingPart
                                  g_normal.init(temp_pos[X], -temp_pos[Y], -
temp_pos[Z]);  //set standard normal as well
                           }
                           //else leave the same as before
                    }
             }
             //add negative translation here to put it back to where it was before
check position
                    ((WtMovable*)parent)->SetTranslation(local);
                    current_pos = previous_pos;
                    current_q = previous_q;
                    //haptic object left where it was, no need to update it.
      }
      else  //if no collision
      {
             //if no collision keep rotation and translation the way they have been
updated.
             if(haptic_parts[ind] != NULL)  //if no collision and haptic objects, then
rotate them freely
             {
                    //Now worry about forces
                    if (depth == 0)
                    {
                           force.init(0,0,0);  //otherwise (depth != 0) the force is
kept as set in slide function
                           curhit = 99;  //no hits this time around
                    }
             }
      }
      ((WtGroup*)i_node)->GetMidpoint(midpoint);
      i_path->GetTranslation(temp_pos);
      i_path->GetOrientation(temp_q);
      temp_frame.Set(temp_pos, temp_q);
      midpoint.Local2WorldFrame(temp_frame, temp_pos);

      //***********SETTING NEW LAST POSITION FOR NEXT ITERATION
      ((WtMovable*)parent)->GetTranslation(temp_pos);
      ((WtMovable*)parent)->GetOrientation(temp_q);
      this->set_localpos(temp_pos);  //in local coordinates
      this->set_localq(temp_q);  //in local coordinates

      ((Part*)node)->set_last_pos(current_pos);  //in world coordinates for
calculations
      ((Part*)node)->set_last_q(current_q);

      transvec.Initialize();
      lasthit = curhit;
}
```

```
WtNodePath* Collision::check_attach(ofstream* d)
{
	gstSeparator* gstparent1; //Temporary for testing
	gstSeparator* gstparent2; //Temporary for testing
	WtP3 inter_pos_local;
	WtQ inter_q_local;
	WtP3 ideal_pos;
	WtQ ideal_q;
	WtP3 attach_dist;
	WtQ attach_q;
	WtM4 trans;
	gstPoint transformAngles;
	gstPoint temp_point;
	gstSeparator* temp_hpart = NULL;
	char tbuf[9];
	WtP3 adjustvect(0.0,-0.3,0.0); //adjustment vector

	depth = 0; //set depth of penetration to zero
	//detach part from cursor and put in real world coordinates
	i_path->GetTransform(trans);  //get accumulated position
	i_node->Remove();
	r->AddChild(i_node);  //attach to the root so still visible
	((WtMovable*)i_node)->SetTransform(trans);  //set to position
	delete i_path; //delete old path
	i_path = new WtNodePath(i_node,r,0);  //make new path
	c_paths[ind] = i_path; //update list
//the manipulator strategy
	if (haptic_parts[ind] != NULL)
	{
		phantom->stopManipulator();
		gstparent1 = (gstSeparator*)haptic_parts[ind]->getParent();
		gstparent1->removeChild(haptic_parts[ind]);  //remove from dynamic
parent
		gstparent2 = (gstSeparator*)gstparent1->getParent(); //put back on
hapticscene
		gstparent2->addChild(haptic_parts[ind]);

		//set to position
		temp_point.init(current_pos[X], -current_pos[Y], -current_pos[Z]);
		haptic_parts[ind]->setPosition_WC(temp_point);

		//set rotation
		mulM(transformMatrix, *rotationMatrix, phantom->getTransformMatrix());
		transformMatrix.getRotationAngles(transformAngles);  //set to current
		haptic_parts[ind]->setRotate(gstVector(1.0, 0.0, 0.0),
transformAngles[0]);
		haptic_parts[ind]->rotate(gstVector(0.0, 1.0, 0.0), transformAngles[1]);
		haptic_parts[ind]->rotate(gstVector(0.0, 0.0, 1.0), transformAngles[2]);

		haptic_parts[ind]->touchableByPHANToM(TRUE);
	}
	if (snd_on)
	{
		WTsound_stop(snd[0]);
		WTsound_play(snd[2]);
	}

	//see if part in a position to attach to something
	if (hitpath !=NULL) //if there was a collision
	{
		if (((Part*)i_node)->get_connect_to() == ((Part*)hitn)->get_index())

		{
```

```
                    hitpath->GetOrientation(temp_q);
                    hitpath->GetTranslation(temp_pos);
                    col_frame.Set(temp_pos,temp_q);
                    i_path->GetOrientation(temp_q);
                    i_path->GetTranslation(temp_pos);
                    inter_frame.Set(temp_pos,temp_q);

                    temp_pos.World2LocalFrame(col_frame,inter_pos_local);  //gets
local position
                    temp_q.World2LocalFrame(col_frame,inter_q_local);  //gets local
orientation

                    //none of this takes into account the rotation of the parent node

                    //gets the ideal position relative to the parent coordinate frame
                    ideal_pos = ((Part*)i_node)->get_connect_pos();
                    //now see how close part is to ideal position
                    //cout<<"The translated position is "<<inter_pos_local[X]<<"
"<<inter_pos_local[Y]<<" "<<inter_pos_local[Z]<<endl;
                    //cout<<"The ideal position is "<<ideal_pos[X]<<"
"<<ideal_pos[Y]<<" "<<ideal_pos[Z]<<endl;
                    //first get distance in coordinates relative to parent

                    //check to see how close to ideal position
                    attach_dist[X] = inter_pos_local[X] - ideal_pos[X];
                    attach_dist[Y] = inter_pos_local[Y] - ideal_pos[Y];
                    attach_dist[Z] = inter_pos_local[Z] - ideal_pos[Z];

                    //now check for orientation
                    ideal_q = ((Part*)i_node)->get_connect_q();
                    //cout<<"The translated orientation is "<<inter_q_local[X]<<"
"<<inter_q_local[Y]<<" "<<inter_q_local[Z]<<" "<<inter_q_local[W]<<endl;
                    //cout<<"The ideal orientation is "<<ideal_q[X]<<"
"<<ideal_q[Y]<<" "<<ideal_q[Z]<<" "<<ideal_q[W]<<endl;
                    attach_q[X] = inter_q_local[X] - ideal_q[X];
                    attach_q[Y] = inter_q_local[Y] - ideal_q[Y];
                    attach_q[Z] = inter_q_local[Z] - ideal_q[Z];
                    attach_q[W] = inter_q_local[W] - ideal_q[W];

                    //if close enough then attach and move to ideal position
                    //set back to 1 after testing
                    if (fabs(attach_dist[X])<DTolerance &&
fabs(attach_dist[Y])<DTolerance && fabs(attach_dist[Z])<DTolerance &&
fabs(attach_q[X])<QTolerance && fabs(attach_q[Y])<QTolerance &&
fabs(attach_q[Z])<QTolerance && fabs(attach_q[W])<QTolerance)
                    {
                           cout<<"distance smaller than "<<DTolerance<<endl;
                           cout<<"Orientation difference smaller than
"<<QTolerance<<endl;

                           i_node->Remove();
                           ((Part*)hitn)->Attach(i_node, WTNODE_APPEND);
                           ((WtMovable*)i_node)->SetTranslation(ideal_pos);
                           ((WtMovable*)i_node)->SetOrientation(ideal_q);
                           ((Part*)i_node)->set_attached(true);  //set flag
                           delete i_path; //delete old path
                           i_path = new WtNodePath(i_node,r,0);  //make new path
                           c_paths[ind] = i_path; //update path list if path changed
as a result of attaching

                           if (snd_on)
                           {
                                  WTsound_stop(snd[0]);     //stop hit sound
```

```
                                    WTsound_stop(snd[2]);   //stop letgo sound
                                    WTsound_play(snd[3]);  //play attachment sound
                             }
                             if (ind == 5) //if this is the replacement part then switch
it to part2's index now so others
                             {                              //can attach to it
                                    ((Part*)i_node)->set_index(1);
                                    c_parts[1]->set_index(5);  //change original part 2's
index to be 5 now
                                    //change part and path arrays now, note that these
are only local arrays to this object
                                    c_parts[5] = c_parts[1];
                                    c_paths[5] = c_paths[1];
                                    c_parts[1] = ((Part*)i_node);
                                    c_paths[1] = i_path;

                                    //now switch haptic array as well
                                    temp_hpart = haptic_parts[1];
                                    haptic_parts[1] = haptic_parts[5];
                                    haptic_parts[5] = temp_hpart;
                                    ind = 1;
                             }
                             if (haptic_parts[ind] != NULL)  //if haptics is turned on
then set haptic object position
                             {
                                    haptic_parts[ind]->setTranslate(haptic_p[ind][X],
haptic_p[ind][Y], haptic_p[ind][Z]);
                                    haptic_parts[ind]->setRotate(gstVector(1.0, 0.0,
0.0), haptic_r[ind]);

                             }
                             if (start_flash !=NULL)
                             {
                                    delete start_flash;
                                    start_flash = NULL;
                             }

                             //put visual flashing here for both parts (two flashes if I
can)
                             start_flash = new WtTask(i_node, flash, 1.0f);
                             //record the attachment
                             _strtime( tbuf );  //gets current time
                             (*d)<<"Attached: "<<tbuf<<endl;//gets current time
                      }
              /*     else if (ind ==5)  //if not connected and hitting something (i.e.
gravity won't be done) then adjust part 5's position
                      {
                             ((Part*)i_node)->Translate(adjustvect, WTFRAME_PARENT);
                      } */

              } //seing if part can be connected loop
       } //if there was a collision loop

       //set parameters for next iteration of collision check
       //in world coordinates
       ((WtGroup*)i_node)->GetMidpoint(midpoint);
       ((Part*)i_node)->GetOrientation(midq);
       i_path->GetTranslation(temp_pos);
       i_path->GetOrientation(temp_q);
       temp_frame.Set(temp_pos, temp_q);
       midpoint.Local2WorldFrame(temp_frame, current_pos);
       midq.Local2WorldFrame(temp_frame, current_q);
```

```
	((Part*)i_node)->set_last_pos(current_pos);  //in world coordinates for
calculations
	((Part*)i_node)->set_last_q(current_q);
	//update vertex position array
	WTvertex* tempvertex;
	tempvertex = ((WtGeometry*)i_node)->GetFirstVertex();
	numvertices = 0;
	while (tempvertex !=NULL)
	{
	//get vertex position
		((WtGeometry*)i_node)->GetVertexPosition(tempvertex, tempvector);
		//convert to world frame
		tempvector.Local2WorldFrame(temp_frame, *v_pos[ind][numvertices]);
		numvertices++;
		tempvertex = WTvertex_next(tempvertex);
	}

	//set local position of cursor to zero...since will not pick up object with
cursor in exact same place next time
	temp_pos.Initialize();
	temp_q.Initialize();
	transformAngles.init(0,0,0);

	this->set_localpos(temp_pos);
	this->set_localq(temp_q);

	//set force to zero while not being picked
	force.init(0,0,0);
	return i_path;  //return pointer to new path so can update the list and main
pointer in main function
	//the local path object will remain for duration of program since collision
object does.

}

gstVector* Collision::get_phantom_force()
{
	return &force;
}

gstVector* Collision::get_normal()
{
	return &g_normal;
}

double* Collision::get_depth()
{
	return &depth;
}
```

picking.h and picking.cpp: Picking object algorithms

picking.h

```
#ifndef _PICKING_
#define _PICKING_

//==========================================================================
//   Filename : picking.h
//      Based on Dice Demo RigidBodyDice class
//      Written in April, 2000
//   Modified in August 2000
```

```
//      By Greg Edwards
/*********************************************************************/
/* This file contains the function that detaches a part from another*/
/* is appropriate and attaches the part to the cursor.                          */
/* If haptics are turned on then a rigidmanipulator in combination  */
/* with a dynamic node is used to allow haptic forces to be updated */
/*********************************************************************/
//=============================================================================

#include "wt.h" //for world tool kit
#include <iostream.h>
#include <string.h>
#include "wtcpp.h" //to use WTK C++ classes
#include "parts3.h" //has part class with model holder and data containers
#include <math.h>
#include <gstSeparator.h>
#include <gstPHANToM.h>
#include <gstRotateManipulator.h>
#include <gstTransformMatrix.h>
#include <gstPoint.h>
#include "RigidManipulator.h"
#include <gstRigidBody.h>
#include "MovingPart.h"

WtNodePath* pick_object(WtRoot* r, WtNodePath* phantom_path, int maxindex, Part*
current_parts[], WtNodePath* current_paths[], bool& picked, gstPHANToM* phantom,
gstSeparator* haptic_parts[], gstTransformMatrix* rm, RigidManipulator* manip,
MovingPart* mp, bool s_on, WTsound* s[], float m[]);

#endif
```

picking.cpp

```
#include "picking.h"

WtTask* start_blink = NULL;


void blink(void* node)
{
       static int visual_count = 0;
       static int matid[30];
       int z = 0;
       static WTpoly* temp_poly = NULL;
       static WTpoly* first_poly = NULL;
       //cout<<"Number of polygons: "<<((WtGeometry*)node)->NumPolys()<<endl;
       switch(visual_count)
       {
       case 0:
              //get original color for part
              temp_poly = ((WtGeometry*)node)->GetFirstPoly();
              first_poly = temp_poly;
              while(temp_poly != NULL)
              {
                     matid[z] = WTpoly_getmatid(temp_poly);
                     temp_poly = WTpoly_next(temp_poly);
                     z++;
              }
              //set new color
              ((WtGeometry*)node)->SetRGB(255,255,255);
              visual_count++;
              break;
       case 2:
```

```
			//for first object
			temp_poly = first_poly;
			while(temp_poly != NULL)
			{
				WTpoly_setmatid(temp_poly, matid[z]);  //a is reset to zero with
iteration of action loop so this is ok
				temp_poly = WTpoly_next(temp_poly);
				z++;
			}
			visual_count++;
			break;
		case 4:
			visual_count = 0;
			delete start_blink;  //and remove task here as well
			start_blink = NULL;
			break;
		default:
			visual_count++;
		} //end switch
}


WtNodePath* pick_object(WtRoot* r, WtNodePath* phantom_path, int maxindex, Part*
c_parts[], WtNodePath* c_paths[], bool& picked, gstPHANToM* phantom, gstSeparator*
haptic_parts[], gstTransformMatrix* rotationMatrix, RigidManipulator* manipulator,
MovingPart* movingobj, bool snd_on, WTsound* snd[], float m_array[])
{

		int numattach = 0;
		WtP3 i_pos;
		WtNode* i_node = NULL;
		WtNodePath* i_path = NULL;
		picked = FALSE;
		WtMovable* phantom_geom = (WtMovable*)phantom_path->GetNode(phantom_path-
>NumNodes()-1);
		WtP3 temp_pos;
		WtQ temp_q;
		WtPQ node_frame;
		WtP3 midpoint;
		WtQ midq;
		WtPQ phantom_frame;
		phantom_path->GetTranslation(temp_pos);
		phantom_path->GetOrientation(temp_q);
		phantom_frame.Set(temp_pos, temp_q);
		WtP3 current_pos;
		WtQ current_q;
		WtM4 trans;
		gstSeparator* parent;

		//haptic variables
		gstTransformMatrix phantomTransform, objectTransform;

		for (int m=maxindex; m >= 0; m--)  //only allowed to select parts at the moment
		{
			if (((WtMovable*)c_parts[m])->NumAttachments()==0)  //if the part has no
attachments on it
			{
				if(phantom_path->IntersectPoly(c_paths[m]))  //if the phantom
cursor collides with the node being checked
				{
					i_path = c_paths[m];
					i_node = i_path->GetNode(i_path->NumNodes()-1);
					int ind = ((Part*)i_node)->get_index();
```

```
                        if(ind <=5 && ind != 0)  //checks to see if it is a
part...modified from 7 to 5 when parts changed
                        {  //Insert moving code here
                              if (((Part*)i_node)->get_attached())  //if attached
to something then find it and detach it
                              {
                                    for(int k=0; k<=maxindex; k++)  //search for
connections
                                    {
                                          if(((Part*)i_node)->get_connect_to() ==
c_parts[k]->get_index())
                                          {
                                                //if connected need to get
attachment number of child (know child but not number :(
                                                numattach = c_parts[k]-
>NumAttachments();
                                                for (int j=0;j<numattach;j++)
//cycle through attachments until get right attachment number
                                                {
                                                      if(((Part*)(c_parts[k]-
>GetAttachment(j)))->get_index()==ind)
                                                      {
                                                            //couldn't figure
out going directly to another reference frame
                                                            //so going to root
then other frame

                                                            i_path-
>GetTransform(trans);  //get accumulated position
                                                            (c_parts[k])-
>Detach(j);  //detach the part
                                                            r-
>AddChild(i_node);  //attach to the root so still visible

      ((WtMovable*)i_node)->SetTransform(trans);  //set to position
                                                            delete i_path;
//delete old path
                                                            i_path = new
WtNodePath(i_node,r,0);  //make new path

                                                            //now attach to
cursor
                                                            i_path-
>GetOrientation(temp_q);
                                                            i_path-
>GetTranslation(temp_pos);

      temp_pos.World2LocalFrame(phantom_frame,current_pos);  //gets local position

      temp_q.World2LocalFrame(phantom_frame,current_q);  //gets local orientation

                                                            i_node->Remove();
                                                            ((Part*)i_node)-
>set_attached(false);  //set flag that not attached
                                                            phantom_geom-
>Attach(i_node, WTNODE_APPEND);  //attach to the phantom cursor
                                                            //set position

      ((WtMovable*)i_node)->SetTranslation(current_pos);

      ((WtMovable*)i_node)->SetOrientation(current_q);
```

```
                                                                    j=numattach+1;
//leave for loop
                                                                    //cout<<"Just
detached from parent part"<<endl;
                                                                    delete i_path;
//delete old path
                                                                    i_path = new
WtNodePath(i_node,r,0);  //make new path
       c_paths[((Part*)i_node)->get_index()] = i_path; //update list
                                                                    //NOW DO THE SAME
FOR HAPTICS...IF HAPTICS ARE ON
                                                                    if (haptic_parts[m]
!= NULL) //if haptics is on and haptic objects defined
                                                                    {
                                                                           //make it not
touchable by phantom while moving
       haptic_parts[m]->touchableByPHANToM(FALSE);

       //haptic_parts[m]->getCumulativeTransformMatrix(objectTransform);
                                                                    parent =
(gstSeparator*)haptic_parts[m]->getParent();
                                                                    parent-
>removeChild(haptic_parts[m]);
                                                                    movingobj-
>addChild(haptic_parts[m]);
                                                                    //set the
mass of the object
                                                                    movingobj-
>setMass(m_array[m]);
       //haptic_parts[m]->setTransformMatrix(objectTransform);
                                                                    manipulator-
>setNode(movingobj);
                                                                    phantom-
>setManipulator(manipulator);
                                                                    phantom-
>startManipulator();
                                                                    // setup the
correctional rotation matrix
                                                                    phantom-
>getTransformMatrix().getRotationMatrix(phantomTransform);
       haptic_parts[m]->getTransformMatrix().getRotationMatrix(objectTransform);

       phantomTransform.inverse();
       mulM(*rotationMatrix, objectTransform, phantomTransform);  //correction
rotation matrix
                                                                    }
                                                                    if (snd_on)
                                                                    {
       WTsound_stop(snd[0]);      //stop hit sound
       WTsound_stop(snd[2]);   //stop letgo sound
       WTsound_play(snd[3]);  //play detachment sound
```

```
                                                        }
                                                    }
                                                }  //end of for loop
                                                k=maxindex+1; //once found
attachment and disconnected, don't look anymore, exit from k-for loop
                                        }  //end of if
                                    } //end of for
                                }  //end of if attached to something
                                else  //not attached to another part but still want
to pick it up
                                {
                                    //put code here to attach to the cursor from
root
                                    ((WtMovable*)i_node)-
>GetTranslation(midpoint);
                                    ((WtMovable*)i_node)->GetOrientation(midq);

                                    midpoint.World2LocalFrame(phantom_frame,
current_pos);
                                    midq.World2LocalFrame(phantom_frame,
current_q);

                                    i_node->Remove();
                                    phantom_geom->Attach(i_node, WTNODE_APPEND);
//attach to the phantom cursor
                                    //set position
                                    ((WtMovable*)i_node)-
>SetTranslation(current_pos);
                                    ((WtMovable*)i_node)-
>SetOrientation(current_q);

                                    delete i_path; //delete old path
                                    i_path = new WtNodePath(i_node,r,0);  //make
new path
                                    c_paths[((Part*)i_node)->get_index()] =
i_path; //update list

                                    if (haptic_parts[m] != NULL) //if haptics is
on and haptic objects defined
                                    {

                                        //make it not touchable by phantom
while moving
                                        haptic_parts[m]-
>touchableByPHANToM(FALSE);

                                        parent =
(gstSeparator*)haptic_parts[m]->getParent();
                                        parent->removeChild(haptic_parts[m]);
                                        movingobj->addChild(haptic_parts[m]);
                                        movingobj->setMass(m_array[m]);
                                        //haptic_parts[m]-
>setTransformMatrix(objectTransform);
                                        manipulator->setNode(movingobj);
                                        phantom->setManipulator(manipulator);
                                        phantom->startManipulator();


                                        // setup the correctional rotation
matrix
                                        phantom-
>getTransformMatrix().getRotationMatrix(phantomTransform);
```

```
                                                haptic_parts[m]-
>getTransformMatrix().getRotationMatrix(objectTransform);

                                                phantomTransform.inverse();
                                                mulM(*rotationMatrix, objectTransform,
phantomTransform);  //correction rotation matrix

                                        }
                                }
                                picked = TRUE;
                                ((Part*)i_node)->set_resting(false);  //set flag that
not resting on anything

                                if (start_blink !=NULL)
                                {
                                        delete start_blink;
                                        start_blink = NULL;
                                }
                                start_blink = new WtTask(i_node, blink, 1.0f);

                                if (snd_on)
                                {
                                        if (!WTsound_isplaying(snd[3]))
                                        {
                                                WTsound_stop(snd[0]);
                                                WTsound_play(snd[1]);
                                        }
                                }

                        }  //end of if it is a part
                        m = -1;  //if found a hit then stop the for loop by setting
the index
                } //end of if hit something
            } //end of if for dissasemble code
      } //end of intersection for loop.
      if (i_path ==NULL)
      {
            picked=FALSE;
            cout<<"No Part under cursor"<<endl;
      }
      return i_path;
}
```

gravity.h and gravity.cpp: Implements gravity on both graphical and haptic objects

gravity.h

```
#ifndef _GRAVITY_
#define _GRAVITY_

//===========================================================================
//   Filename : collision4.h
//      Based on Dice Demo RigidBodyDice class
//      Written in August, 2000
/*****************************************************************/
/* THe basic algorithm is the same as the collision code.           */
/* only now the movement vector is always pointing down with a      */
/* standard movement.
            */
/*****************************************************************/
//===========================================================================
```

```
#include "wt.h" //for world tool kit
#include <iostream.h>
#include <string.h>
#include "wtcpp.h" //to use WTK C++ classes
#include "parts3.h" //has part class with model holder and data containers
#include <math.h>
#include "collision4.h" //has collision detection and movement functions

void checkfall(WtRoot* rt, WtNodePath* c_paths[], Part* c_parts[], int max,
gstSeparator* haptic_parts[], WtMovable* t, WtMovable* w, bool s_on, WTsound* s[],
WtP3* ve_pos[][40]);

#endif
```

gravity.cpp

```
#include "gravity.h"

const float fall_dist = 3.0f;

void checkfall(WtRoot* rt, WtNodePath* c_paths[], Part* c_parts[], int max,
gstSeparator* haptic_parts[], WtMovable* tab, WtMovable* wal, bool snd_on, WTsound*
snd[], WtP3* ve_pos[][40])
{

       //go through all parts that are detached and not on something (set flag
       //that object has hit something and turn it off again once picked).

       //get vertices of the part
       unsigned char red,green,blue;
       float smallest;
       WtP3 temp_pos;
       WtQ temp_q;
       WtP3 midpoint;
       WtQ midq;
       WtPQ temp_frame;
       WTvertex* tempvertex=NULL;  //to get vertex temporarily
       WtP3 tempvector;  //to get position of vertices temporarily
       int numvertices=0;
       float* distance = new float(0.0f);
       WtP3 vert_pos[40];
       WtP3 movevector(0.0, 1.0, 0.0); //pointing straight down
       WtP3 adjustvect(0.0,-0.8,0.0); //adjustment vector
       WTpoly* poly=NULL;
       WtNode* tempn = NULL;
       WtP3 transvec(0.0, fall_dist, 0.0);  //standard down vector for translation,
change y component if too close
       WtNode* node = NULL;
       WtNodePath* temp_hitpath = NULL;
       int c = 0;
       WtNodePath* hitpath_array[10];
       for (int b=0;b<10;b++)
       {
              hitpath_array[b] = NULL;
       }

       for (int a = 0; a<= max; a++)  //loop for all parts
       {
              node = c_paths[a]->GetNode(c_paths[a]->NumNodes()-1);
              //check if detached and in mid-air
              if (!((Part*)node)->get_attached() && !((Part*)node)->get_resting())
              {
```

```
                    if (haptic_parts[a] != NULL) //if haptics turned on then make sure
can't hit it while falling
                           haptic_parts[a]->touchableByPHANToM(FALSE);  //come back to
this if more time permits :)
                    //***START WITH RAY CHECK ********************

                    c_paths[a]->GetTranslation(temp_pos);
                    c_paths[a]->GetOrientation(temp_q);
                    temp_frame.Set(temp_pos, temp_q);

                    tempvertex = ((WtGeometry*)node)->GetFirstVertex();
                    numvertices = 0;
                    while (tempvertex !=NULL)
                    {
                    //get vertex position
                    ((WtGeometry*)node)->GetVertexPosition(tempvertex, tempvector);
                    //convert to world frame
                    //tempvector.Local2WorldFrame(temp_frame, vert_pos[numvertices]);
                    tempvector.Local2WorldFrame(temp_frame, *ve_pos[a][numvertices]);
                    numvertices++;
                    tempvertex = WTvertex_next(tempvertex);
                    }
                    smallest = fall_dist;  //set initial comparing distance to
standard fall distance
                    //now cast rays from vertices of object and record the smallest
downwards collision potential
                    for(int i = 0; i<numvertices;i++)
                    {
                           WtNodePath* temppath=new WtNodePath;

                           //had to create object since rayintersect won't take a null
path
                           //vert_pos must be in root coordinates
                           //poly = rt->RayIntersect(movevector, vert_pos[i],
distance, &temppath);
                           poly = rt->RayIntersect(movevector, *ve_pos[a][i],
distance, &temppath);
                           if (poly != NULL)
                           {
                                  tempn = temppath->GetNode(temppath->NumNodes()-1);
                                  //strncpy(tempname,((WtMovable*)tempn)->GetName(),5);
                                  //tempname[7] = '\0';
                                  WTpoly_getrgb(poly,&red,&green,&blue);
                                  if (((Part*)tempn)->get_index() != a && !(red ==255
&& green ==0 && blue==0)) // if not itself or the cursor then count as hit
                                  {
                                         if (*distance < smallest)  //if ray collision
distance smaller than normal fall distance
                                         {
                                                //cout<<"Smallest is: "<<smallest<<"
distance is: "<<*distance<<endl;
                                                //fall by this smaller amount
                                                smallest = *distance;
                                                transvec[Y] = smallest;
                                                ((Part*)node)->set_resting(true);
                                                if (haptic_parts[a] != NULL) //if
haptics turned on
                                                       haptic_parts[a]-
>touchableByPHANToM(TRUE);
                                                //put sound here

                                                if(snd_on)
                                                {
```

```
                                                WTsound_stop(snd[0]);
                                                WTsound_play(snd[0]);
                                        }

                                }
                                //otherwise keep distance the same and
transvec at default
                        }
                }
                else
                        delete temppath;
        }
        ((Part*)node)->Translate(transvec, WTFRAME_PARENT);
        if (haptic_parts[a] !=NULL) //if haptics are turned on
                haptic_parts[a]->translate(transvec[X], -transvec[Y], -
transvec[Z]);

        //***NOW CHECK WITH POLYGON INTERSECTION SINCE RAYS MISS THINGS
SOMETIMES (NEED RAYS FOR DISTANCE THOUGH)
        //now check for collision
        for (int m=max; m>=0; m--)  //only checking for collision with
parts at the moment
        {
                if (m != ((Part*)node)->get_index())  //don't check for
collision with itself
                {
                        //not sure if should use root as ancestor or
attachment parent
                        //if(inter_path->IntersectBBox(check_path))
                        temp_hitpath = c_paths[a]->IntersectNode(c_parts[m],
0);  //only checking with other parts at the moment
                        if(temp_hitpath !=NULL)  //if it collides with the
node being checked
                        {
                                bool same = false;
                                //check if this node is already in our list
                                for (int a=0;a<c;a++)
                                {
                                        if (hitpath_array[a] != NULL)
                                        {
                                                if (((Part*)(hitpath_array[a]-
>GetNode(hitpath_array[a]->NumNodes()-1)))->get_index() == ((Part*)(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1)))->get_index())
                                                {
                                                        same = true; //found this
node already in list
                                                        a = c+1; //to exit for
loop
                                                }
                                        }

                                }
                                if(same == false)
                                {
                                        hitpath_array[c] = new
WtNodePath(temp_hitpath->GetNode(temp_hitpath->NumNodes()-1),rt,0) ;
                                        c++;
                                }

                        }

                }
        }
```

```
                    //also check for collision with table and wall
                    //table
                    temp_hitpath = c_paths[a]->IntersectNode(tab, 0);  //change part
to table pointer
                    if(temp_hitpath !=NULL)  //if it collides with the node being
checked
                    {
                          hitpath_array[c] = new WtNodePath(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1),rt,0) ;
                          c++;
                    }
                    //wall
                    temp_hitpath = c_paths[a]->IntersectNode(wal, 0);  //change part
to wall pointer
                    if(temp_hitpath !=NULL)  //if it collides with the node being
checked
                    {
                          hitpath_array[c] = new WtNodePath(temp_hitpath-
>GetNode(temp_hitpath->NumNodes()-1),rt,0) ;
                          c++;
                    }
      //**************STEP 2: Polygon intersection with bounding box collisions
                    if (c != 0)   //if there is a bounding box collision check to see
if polygon intersection
                    {
                          //now check for collision
                          for (m=0; m < c; m++)  //only checking for collision with
parts at the moment
                          {
                                if(c_paths[a]->IntersectPoly(hitpath_array[m]))  //if
it collides with the node being checked
                                {
                                      ((Part*)node)->Translate(-1*transvec,
WTFRAME_PARENT);  //move back to previous position
                                      m = c+1;  //stops the for loop
                                      ((Part*)node)->set_resting(true);
                                      if (haptic_parts[a] !=NULL) //if haptics are
turned on
                                      {
                                            haptic_parts[a]->translate(-
transvec[X], transvec[Y], transvec[Z]); //move back to previous position
                                            haptic_parts[a]-
>touchableByPHANToM(TRUE);
                                      }
                                      //put sound here

                                      if(snd_on)
                                      {
                                            WTsound_stop(snd[0]);
                                            WTsound_play(snd[0]);
                                      }

                                }
                          }
                    } //end of if bounding box detected anything (count !=0)

                    for (int j=0;j<c;j++)
                    {
                          delete hitpath_array[j];
                          hitpath_array[j] = NULL;
                    }
```

```
                    //set variables for recording position
                    ((WtGroup*)node)->GetMidpoint(midpoint);
                    ((Part*)node)->GetOrientation(midq);
                    midpoint.Local2WorldFrame(temp_frame, temp_pos);
                    midq.Local2WorldFrame(temp_frame, temp_q);
                    ((Part*)node)->set_last_pos(temp_pos);  //in world coordinates for
calculations
                    ((Part*)node)->set_last_q(temp_q);
                    ((Part*)node)->GetTranslation(temp_pos);
                    ((Part*)node)->GetOrientation(temp_q);

                    transvec[Y] = fall_dist; //set back to default in case it was
changed
                    /*if (((Part*)node)->get_resting()  && ((Part*)node)->get_index()
== 5)  //if part 6 just started resting
                           ((Part*)node)->Translate(adjustvect, WTFRAME_PARENT);
//then adjust graphics up a bit to make picking up easier */

             }  //if detached and not resting on anything
      }  //loop for all parts
}
```

Vita

Gregory W. Edwards

BASc Systems Design Engineering,
Candidate for M.S. Human Factors Engineering, November 2000

Skills Summary

- Experience in running small user tests in academic setting (for classes, honours thesis & Masters thesis).
- Experience in researching topics related to HCI, virtual environments and human factors.
- Proficient in many computer applications, languages and operating systems.

Education

Honours Bachelor of Applied Science, Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada, May 1998.

Candidate for Masters of Science, Human Factors Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, United States of America, Aug. 1998 - present.

Computer Experience:

Languages: C++, Visual Basic, Sense8 World Up and World Toolkit, GHOST Haptic Libraries, HTML, Performer and CAVE libraries for graphics programming.

Software: Standard PC and Mac Operating Systems and applications, Unix.

Hardware: Various experience from computer support jobs (mostly PC equipment), various VR equipment (HMDs, trackers,PHANToM,etc.).

Relevant Courses:

- Software Design: Borland C++ & Visual Basic
- Man Machine Systems Design
- Human Factors System Design
- Research Design
- Engineering Psychology
- Cognition
- Usability Engineering
- Training Systems Design

Publications:

Ducharme, M.B., Edwards, G. and J. Frim (In Press). Comparison of two rewarming techniques with a thermal water manikin. Proceedings of The 8th International Conference on Environmental Ergomonics, San Diego, CA, USA, 18 - 23 Oct., 1998 (in Press).

Ducharme, M.B. and G. Edwards. (In Press) The comparison of three field rewarming systems using a thermal water manikin. DCIEM Report #99-XX, 1999 (In Press).

Eaton, D., Edwards, G. and M.B. Ducharme (1999). Evaluation of a diving heating system using a thermal water manikin. DCIEM Report #TR 1999-004, 1999.

Edwards, G.W., Thompson, J. and C.G. MacGregor. (1998). Investigating the Role of Guided Tours in the Learning of Virtual Environments. Proceedings of the 30th Annual Conference of the Human Factors Association of Canada (pp.3-9). Toronto, Ontario: Human Factors Association of Canada. **Tied for Best Undergraduate Paper of 1998.**

Nash, E.B., Edwards, G.W., Thompson, J.A. and W. Barfield. (2000). A review of presence and performance in virtual environments. International Journal of Human-Computer Interaction, Vol. 12(1), p.1-41.

Work Experience

Jan. '94 - April '94: ***Computer Support***
Information Management Practice
Consulting and Audit Canada
Ottawa, Ontario

- Gathered and Analyzed Information from Distributors for Consultant Projects.
- Updated, Translated and Modified Customer Reports and FoxPro Help databases.

Sept. '94 - Dec. '94 ***Technical Support***
& May '95 - Aug. '95: ***MIS, Ontario Teachers' Pension Plan Board***
North York, Ont.

- Assessed and Provided Technology Solutions to Individual Needs of Users and Managers.
- Setup, Documented and Managed a Remote Access Device Allowing Managers to Work from Home.
- Created a Database to Keep Track of Software Licensing.

Jan. '96 - Apr. '96: ***Business Systems Analyst***
IT Systems, Newbridge Networks
Kanata, Ont.

- Provided Business Systems Support, Needs Assessment and Technology Implementations.
- Organized and Lead, from Start to Finish, an Audit and Repair Project on their MRP System.
- Analyzed Task of Finished Goods Inventory Tracking and Provided System.

Sept. '96 - Dec. '96: ***Research Assistant***
May '97 - Aug. '97 ***HPP Sector, Defence and Civil Institute of Environmental Medicine***
May '98 - Aug. '98 ***Toronto, Ont.***

- Researched and Assisted in the Prototyping of a Portable Rewarming System for Hypothermia Victims.
- Created a Thermal Water Manikin for Testing of Rewarming Prototypes.
- Conducted Rewarming Comparison Experiments and Reported Results in Technical Reports.
- Presented Findings to the Search and Rescue Community as well as the DCIEM Institute.

Aug. '98 - May '99: ***Part-Time Teaching Assistant***
ISE Dept., VPI&SU
Blacksburg, VA.

- Graded Class Material.
- Occasionally Taught Class.
- Developed and Maintained Web Page.

June '99 – June '00: ***Part-time Research Assistant under Dr. Ron Kriz***
Laboratory for Scientific Visual Analysis, VPI&SU
Blacksburg, VA.

- Part of development team creating of a collaborative software tool for virtual environment visualization.
  URLs: http://www.sv.vt.edu/future/cave/software/ccc/ & http://www.sv.vt.edu/future/cave/software/cccatom/
- Presented application at SuperComputing 1999.
- Programmed in C++ and used Performer, CAVE and CAVERN libraries.