1-1-2002

Implementing haptic feedback in a projection screen virtual environment

Andrew George Fischer
Iowa State University

Implementing haptic feedback in a projection screen virtual environment

by

Andrew George Fischer

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Judy M. Vance (Major Professor)
Greg R. Luecke
Ashraf F. Bastawros

Iowa State University

Ames, Iowa

2002

Graduate College

Iowa State University

This is to certify that the master's thesis of

Andrew George Fischer

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES iv

CHAPTER 1. INTRODUCTION 1
- Motivation 2
- Thesis Organization 3

CHAPTER 2. HAPTICS IN VIRTUAL REALITY 5
- Haptics Background 5
- The PHANToM 6
- Commercial Haptic Devices 7
- Uses for Haptics Technology 10
- Improving Haptics Technology 12

CHAPTER 3. PROBLEM STATEMENT AND SOLUTIONS 16
- Problems to Overcome 16
- Solutions Presented 17
  - The PHANToM 18
  - Phantom Stand 19
  - Phantom Volume 21
  - Virtual Reality API 22

CHAPTER 4. STRUCTURE OF THE PROGRAM 24
- Hardware 24
- CavePhantomApp 26
- PhantomVolume Class 28
- PhantomDriver Class 30

CHAPTER 5. EXAMPLE APPLICATIONS 36
- NURBS Surface Example 36
  - NURBS Background 36
  - Example Program 37
- Virtual Assembly Example 40
  - Background 40
  - Program Structure 43
  - Assembly Example 45

CHAPTER 6. CONCLUSIONS AND FUTURE WORK 49
- Conclusions 49
- Future Work 51

BIBLIOGRAPHY 53

LIST OF FIGURES

Figure 2.1. SensAble Technologies' PHANToM haptic device 7

Figure 2.2. Immersion Corporation's CyberForce haptic device 8

Figure 2.3. Force Dimension's Delta Haptic Device 9

Figure 2.4. The Sarcos Dextrous Arm Master 9

Figure 3.1. The phantom stand 20

Figure 3.2. Computer model of the phantom stand 21

Figure 3.3. A phantom volume in the virtual environment 22

Figure 4.1. The C6 at Iowa State University 25

Figure 4.2. Connections to use the PHANToM in the C6 26

Figure 4.3. The PhantomDriver class's haptic scene graph structure 32

Figure 4.4. Communication between the main program and haptic process 34

Figure 5.1. A NURBS surface and its control points 38

Figure 5.2. The NURBS surface example application 39

Figure 5.3. Solid model of a rudder pedal assembly 41

Figure 5.4. Voxel model of a rudder pedal assembly 41

Figure 5.5. Point shell model of a rudder pedal assembly 42

Figure 5.6. The tangent-plane force model 43

Figure 5.7. Diagram of the virtual coupler scheme 45

Figure 5.8. Picture of the aircraft portions used in the virtual assembly example 46

Figure 5.9. Solid model of the aircraft portions for the virtual assembly example 47

Figure 5.10. The virtual assembly example application 48

CHAPTER 1

INTRODUCTION

Virtual reality, the immersion of a person in a realistic, computer-generated environment, is becoming a popular engineering design tool. The ability to interact with three-dimensional digital models by using natural human motions provides a unique interface between humans and computer models. Engineering design and prototyping already benefit from virtual reality in many ways, from prototype evaluation and virtual assembly to visualizing volumetric data sets (Ryken and Vance, 2000) (Gupta et al., 1997) (Avila and Sobierajski, 1996).

A key component of virtual reality (or VR) systems is the ability to immerse a participant in a computer generated virtual environment. Immersion refers to the sense of "being there" that a user feels in the virtual world; the greater the sense of immersion, the more real the virtual world appears (Pausch et al., 1997). The level of immersion provided by VR comes in many forms, from simple stereo vision on a desktop computer monitor to a multi-screen projection environment, complete with position tracking of the user(s) and surround sound.

The sense of immersion a user feels in a VR environment is related to the number of senses stimulated, such as sight and hearing (Burdea, 1996). However, most virtual reality systems are lacking in a key area of stimulation, namely some form of physical or haptic feedback. Haptics refers to the feeling of force, weight, roughness, or other physical resistance in a virtual environment (Burdea, 1996). The integration of haptics with virtual reality is a recent development. One device capable of providing haptic feedback in a virtual reality simulation is SensAble Technology's PHANToM. This device was designed for use in a desktop environment where the user either views the virtual display on a monitor or

through a head mounted display (HMD). This research examines the issues surrounding the use of the PHANToM in a projection screen synthetic environment.

Motivation

Developing a method to add force feedback to a projection screen virtual environment and investigate the benefits this adds to an engineering task, such as assembly methods prototyping, was one of the principle motivations for this work. Virtual assembly attempts to eliminate physical mockups in engineering design by simulating the assembly process in a virtual environment (McNeely et al., 1999). VR technology allows humans to interact directly with digital models. This can provide the ability to prototype assembly methods prior to part fabrications. Investigation of virtual assembly task times indicates that adding force feedback also increases efficiency (Burdea, 1999). Very similar to virtual assembly is virtual prototyping, where virtual reality is used to evaluate part designs for criteria such as ease of use by human operators. Again research shows the addition of haptic feedback significantly decreases task completion times (Volkov and Vance, 2001).

Developing a form of haptic feedback that could be applied to interactive shape design was another motivation for this research. Interactive shape design allows a designer to modify or deform a digital shape and immediately observe the changes. Virtual reality provides a three dimensional environment where such changes take place. Once shape and size are determined, parts may be assembled in the virtual environment to identify interference issues. The designer can continue to alter part geometry to resolve difficulties as they arise. The addition of analysis data such as stress to interactive shape design gives the designer another tool. As the part is changed to remedy interference issues, shape sensitivities are used to approximate the change in part stresses (Yeh and Vance, 1997). This way the user immediately sees all the effects of a design change. Performing this work in a virtual environment can greatly speed the design and optimization process, and the addition

of force feedback while manipulating or deforming a part could provide additional information to the user.

Bringing the PHANToM into a projection screen virtual environment presents several challenges. A way to support and move the PHANToM about the environment must be devised, the size differences between the haptic device's physical workspace and the virtual environment need to be worked around, and software to control the PHANToM in the environment has to be written.

Provided the program written to perform this PHANToM implementation is designed in an object-oriented fashion, a large number of applications could easily realize the benefits of haptic feedback in a projection-screen environment. In Burdea's review on robotics and virtual reality, he makes one of the best cases for the value of haptic feedback:

> "Being able to touch, feel, and manipulate objects in an environments, in addition to seeing (and hearing) them, provides a sense of immersion in the environment that is otherwise not possible. It is quite likely that much greater immersion in a virtual environment can be achieved by the synchronous operation of even a simple haptic interface with a visual and auditory display, than by large improvements in, say, the fidelity of the visual display alone." (Burdea, 1999)

Thesis Organization

Chapter 1 provides an introduction to, the objectives for, and the organization of this thesis. Chapter 2 presents a short history of haptic devices, the various current uses for haptic technology, ways the technology needs to improve, and the benefits of providing force feedback in simulations. The problems faced in this project and the solutions developed follow in Chapter 3. In Chapter 4 the program for integrating force feedback with the projection virtual environment is detailed. Chapter 5 demonstrates some example uses of the

program in shape exploration and a virtual assembly application. Finally, Chapter 6 presents conclusions and suggestions for future work.

CHAPTER 2

HAPTICS IN VIRTUAL REALITY

Virtual reality may be defined as "a system which provides real-time viewer-centered head-tracking perspective with a large angle of view, interactive control, and binocular display". (Cruz-Neira et al., 1993) In virtual reality, a user communicates and interacts with digital objects in a virtual environment through sensory channels such as vision, hearing, and touch. This level of immersion, when compared to the usual desktop computer, permits users to perform tasks that are impossible in the real world (Burdea, 1999). This thesis is concerned with the touch sensory channel of communication, often referred to as haptics.

Haptics Background

The word "haptic" comes from the Greek *haptesthai*, to touch or grab. Research shows that the sense of touch actually has two components, tactile and kinesthetic. The tactile sense refers to the actual touching of a surface with the finger and the sensing of roughness, temperature, etc. Kinesthetic or dynamic touch provides information about the physical properties of an object such as its weight, size, and position. While the tactile sense depends on nerve endings in the finger, the kinesthetic relies on the position of, and forces applied to, a user's hand and limbs (Perkowitz, 1999). Since the PHANToM only provides input to the dynamic or kinesthetic haptic sense, the word haptic will refer to only the kinesthetic sense throughout the remainder of this thesis.

Haptics grew out of the telerobotics research begun in the 1950's. In telerobotics, a person manipulates an arm-like mechanical device (the master) while another similar arm (the slave) mimics the motions. This permits the slave to do hazardous work in dangerous locations while the human operator works in relative safety. Forces felt by the slave arm are transferred back to the master arm through the haptic device. In virtual reality, the slave arm

is replaced by a digital representation of a hand or other manipulator and simulated forces in the virtual world are transferred back to the master arm (Burdea, 1996).

In order for a haptic device to provide force feedback to a user, some portion of the device has to be "grounded", or attached to a support. Commercially available haptic devices may be separated into two categories depending on whether they are grounded to the user's body or to a separate object in the real environment (Burdea, 2000). Body grounded haptic devices, by moving with the user, permit operation in a larger workspace. However, since the device must be large enough to apply significant force and still be carried, they are either fatiguing or they are limited to supplying forces to only the user's hand. As a result, most of the force-feedback haptic devices in use today are grounded to some other object such as a desk, floor, or ceiling (Burdea, 1996). This research focuses on using the PHANToM haptic device which is a desk-grounded three-degree of freedom force feedback mechanism.

The PHANToM

The PHANToM haptic device was developed at the Massachusetts Institute of Technology's Artificial Intelligence Laboratory by Thomas Massie and Kenneth Salisbury. The PHANToM, (Personal HAptic iNTerface Mechanism), was designed as a relatively low-cost device to provide the feeling of force interaction with virtual objects. (Massie and Salisbury, 1994) Today, despite a workspace volume of only 19x27x37 cm, the PHANToM's relative simplicity, ease of use, and commercial availability through SensAble Technologies, make it a popular haptic interface. The PHANToM may be programmed through a specialized software developer's toolkit called GHOST. A picture of the PHANToM appears in Figure 2.1.

Figure 2.1. SensAble Technologies' PHANToM haptic device

The GHOST software consists of a set of C++ objects that permit easy creation of a 3D haptic environment for the PHANToM. Using GHOST makes most of the low-level work required to operate the PHANToM transparent to the programmer. This lets the programmer concentrate on dealing with the geometry and properties of the haptic environment through a scene graph style structure. GHOST permits the user to assign surface properties to objects (roughness, compliance) as well as dynamic motion and physics. The software also creates a separate haptic process, which controls the PHANToM motors and keeps the force update rates of the PHANToM in the proper range. Finally, GHOST allows the programmer to expand the software capabilities to meet an individual's needs, providing great flexibility in an application (SensAble, 2001).

Commercial Haptic Devices

While a wide variety of specialized research oriented haptic devices exist, few are commercially available. A short description of some of the commercially available devices follows, excluding the aforementioned PHANToM.

The CyperGrasp, developed by Virtual Technologies and now marketed by Immersion, is a haptic glove, grounded at the user's wrist, which provides force feedback to each finger via an exoskeleton and cables. The actuators for each cable may be worn in a backpack, allowing the user free movement (Immersion, 2002). The CyberGrasp only

applies forces to the fingers so the weight and inertia of an object cannot be sensed. To remedy this, Immersion offers a CyberGrasp attachment called CyberForce. CyberForce is a robotic arm, similar in appearance to a PHANToM, that attaches to the CyberGrasp glove and provides whole-arm force feedback. Unfortunately, the CyberForce must be mounted to a separate object in the environment, limiting the user's mobility (Immersion, 2002). A picture of the CyberForce appears in Figure 2.2.

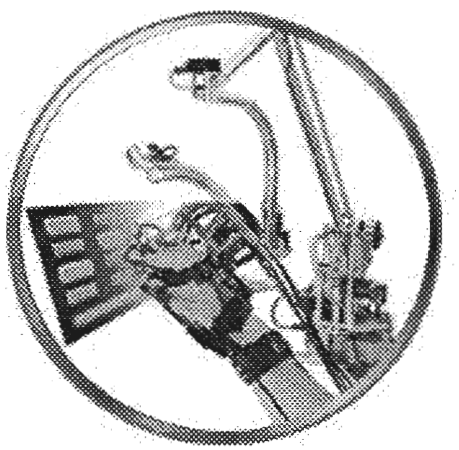

Figure 2.2. Immersion Corporation's CyberForce haptic device

The Force Dimension corporation offers a six degree of freedom force feedback mechanism called the Delta Haptic Device or DHD. The DHD is based on a patented Delta mechanism with three double-bar parallelogram linkages, allowing a 360mm OD by 200mm long cylindrical manipulator workspace. The Delta Haptic Device appears in Figure 2.2. Up to 25N of continuous force and 0.2Nm of torque may be applied to the user's hand (Grange et al., 2001).

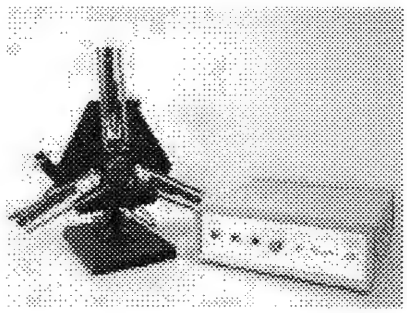

Figure 2.3. Force Dimension's Delta Haptic Device

Researchers at the University of Utah make extensive use of a Dextrous Arm Master from Sarcos (Figure 2.3) (Hollerbach et al., 1997). Essentially a large exoskeleton, the Dextrous Arm Master follows the user's arm movements and provides hydraulic powered force feedback. Figure 2.4 shows the Dextrous Arm Master. Disadvantages include the arm's large size, making it a stationary device, and its strength, which poses a potential safety threat to the user. Sarcos also offers the Utah/MIT Dextrous Hand, a haptic glove similar in appearance and functionality to the CyberGrasp (Sarcos, 2002).

Figure 2.4. The Sarcos Dextrous Arm Master

Uses for Haptic Technology

The early prototype stages of engineering design when models are made out of clay could greatly benefit from haptic technology. Currently automotive body shapes are first created in clay and then a CAD model is later developed. Obtaining a CAD model from a physical clay model is a time consuming and inexact process. To overcome this limitation, a "digital clay" program that uses the PHANToM to let designers sculpt clay models on the computer with a variety of tools has been developed (Perkowitz, 1999). SensAble's FreeForm modeling system, a digital clay sculpting software package for industrial designers, is the commercial version of this program. FreeForm modeling combines clay or foam's ease of use with the advantages of a digital model allowing for shorter product development times. SensAble's growing customer list for this product includes Ford, Honda, Toyota, Fisher-Price, and more (Hickey, 2001).

Medical surgery also benefits from haptic technology, especially surgical training. Surgical training simulators using one or more PHANToMs to represent surgical instruments not only provide feedback for simulating soft tissues but can also record the surgeons' actions to monitor their skill and progress (Salisbury and Srinivasan, 1997). Laparoscopic surgery, where the surgeon's work is done through small incisions in the skin and guided by images from a fiber optic camera, lends itself well to haptic simulators. Since surgeons perform the actual surgery watching a computer display, adding haptic feedback to simulator tools and using detailed digital models makes for a highly realistic simulation (Chen and Marcus, 1998). However, the modeling and computational difficulty of realistically simulating portions of virtual surgery, such as when soft organs interact with each other or deformations become very large, have yet to be fully overcome (Sorid, 2000).

Reseachers at The Boeing Company have integrated a six-degree of freedom PHANToM with their Voxmap PointShell (VPS) collision detection software to produce an impressive virtual assembly tool. VPS allows a polygonal model to be discretized into a

collection of voxels, which can be used as the basis for a very fast collision detection algorithm (McNeely, 2002). By combining VPS with the GHOST toolkit, dynamic manipulation of a detailed rigid object within an environment whose complexity is only limited by computer memory is achieved while maintaining haptic update rates (McNeely et al., 1999).

To investigate how accurately virtual assembly tasks simulate the actual assembly process, Rakesh Gupta developed a design for assembly analysis tool called the Virtual Environment for Design for Assembly or VEDA (Gupta et al., 1997). A VEDA user can see and manipulate several two-dimensional objects in the virtual world. Haptic force feedback using two PHANToMs, physically based modeling, accurate collision detection, and sound cues all provide a realistic virtual assembly experience. Testing with human subjects compared the task completion times for an actual assembly process and an identical virtual assembly process. Results showed VEDA assembly times correlated with actual assembly times as task difficulty increased, though all virtual task completion times were roughly twice the actual task times (Gupta et al., 1997).

Haptics technology is also being explored for use in data visualization, nano-manipulation, and as an aid to computer art. In their work on volume visualization, Avila and Sobierajski explore the use of a PHANToM with haptic feedback as an input and output device for exploring complex three-dimensional data. They report haptic integration gives a more intuitive method to understand and alter such data (Avila and Sobierajski, 1996). Researchers at the University of North Carolina at Chapel Hill are using haptics to explore extremely small objects with the nanoManipulator. This device combines virtual reality, a force-feedback stylus, and an atomic-force microscope (AFM) to let scientists observe and interact with molecular structures. The haptic device allows users to control the tip of the AFM and manipulate objects on the nanoscale surface (Guthold et al., 2000). For artistic work, researchers at the University of Utah demonstrate using a haptic device to paint texture

maps directly onto computer models. This allows the artist to use natural hand motions while digitally "painting" textures for 3D models (Johnson et al., 1999). Such an application could improve digital model creation for the gaming industry.

Improving Haptics Technology

Despite the many uses listed, the use of haptics in virtual environments has many limitations. One of the principle concerns with haptic feedback is the sampling or refresh rate necessary for realistic feeling. While computer graphics images only need to be refreshed 30 times a second (30 Hz) to appear smooth to the human eye, a person's skin can sense vibrations up to 1000 Hz (Salisbury and Srinivasan, 1997). This means a haptic device must have a control loop that updates nearly 1000 times a second to prevent the user from sensing unwanted vibrations. The order of magnitude difference between visual and haptic refresh rates limits the number and complexity of the models haptic displays can currently handle. To meet this requirement, the GHOST toolkit creates a separate haptic process, which runs at 1000 Hz, to control the PHANToM. If model complexity slows the simulation too much, the user is warned, and the application terminates (SensAble, 2001). To ensure these high refresh rates, the haptics process is often run on a separate processor or computer (Burdea, 2000). The faster the processor and the faster the connection between the haptics process and main simulation process, the more complex the haptic simulation can be. Thus improvements in microprocessor and network speeds are a key to advancing the use of haptics in virtual environments.

Another limitation of current haptic devices is the ability to interpret a variety of CAD model types. This limitation often makes some sort of model translation necessary to use a haptic device such as the PHANToM, since continuous surfaces, such as NURBS (Non-Uniform Rational B-Splines), have become the standard for representing geometry in computer aided industrial design (Hollerbach, 2000). Currently the GHOST toolkit is limited

to representing geometry with basic geometric shapes (cube, sphere, cylinder, etc.) and triangle polygon meshes for more general shapes (SensAble, 2001). To overcome this limitation, researchers at the University of Utah have developed an algorithm that permits direct haptic interaction with NURBS models. This algorithm was demonstrated with several haptic devices, including the PHANToM and a Sarcos Dextrous Arm Master (Thompson and Cohen, 1999).

Real-time deformation of the models used for haptic interaction, as in the simulation of crushing a rubber ball, is a difficult problem. Currently, no implementations exist that allow fast general deformation of arbitrary geometry with realistic haptic force feedback (Ramanathan and Metaxas, 2000). Typically, the Finite Element Method (FEM) is used to model the deformation and calculate the corresponding forces, as in some medical surgery simulators (Cotin and Delingette, 1998). However, finite element computations typically take too long to maintain haptic update rates. Thus the method is limited to using very simple models (Vuskovic et al., 2000), or approximated versions of the Finite Element Method (Weghorst et al., 1999). These deformations are especially critical in surgery simulation, where soft tissues need to be accurately represented (Frank et al., 2001). Adding the highly nonlinear properties of body organs to the simulation further increases the difficulty of accurate real time modeling (Sorid, 2000).

Modern haptic devices that permit force feedback to the user's entire arm, including elbow and shoulder joints, all suffer from some sort of workspace limitation. Smaller desktop devices such as the PHANToM constrain the user's hand to a few cubic feet of space or less, while devices that permit full arm motion are large and heavy enough that they must be in a fixed location for purposes of the simulation (Burdea, 1996). Such large haptic devices, such as the Sarcos Dextrous Arm Master, also pose a potential threat to the user because of their strength and inertia. Researchers at Iowa State University have worked toward overcoming some of the barriers to using such robotic arms by electro-magnetically

linking them to the user's hand (Luecke et al., 1997). A robotic arm with magnetic actuators uses sensors to track and closely follow the user's hand. Forces are applied through electro-magnetic fields to coils worn on the user's fingers. The user is freed from the inertia and friction of a large robotic arm while still benefiting from a large device workspace.

The limitations discussed above make integrating haptic feedback with large, fully immersive virtual environments a difficult task. Though most haptic devices are used with simple stereo monitors or head-mounted displays for visualization, these limit the number of participants in the simulation, restrict the user's movement, and in general don't provide the level of immersion that projection screen environments do. While work has been done that utilizes more specialized types of force feedback in such environments (Edwards, 1998), many challenges remain before general purpose commercial haptic feedback in projection screen virtual environments becomes truly practical.

Besides the technical limitations, there is a general lack of knowledge about how much workspace is necessary to effectively use a haptic device. Unless exact correlation between the user's hand and a virtual hand position is needed, there is the potential to scale the virtual haptic workspace, allowing a desktop haptic device to probe regions of varying size. Whether this affects the usefulness of the haptic feedback is not clear. As Hollerbach points out in his discussion of current haptics issues, "The influence of the size of the haptic workspace versus task requirements is not known" (Hollerbach, 2000).

This research seeks to add force feedback to a multiple projection screen environment by using a commercial haptic device. Two example applications, shape exploration and virtual assembly, will demonstrate force feedback in the virtual environment. By bringing a haptic device into the projection environment instead of the traditional desktop/HMD use, the benefits of a surround screen simulation, such as an increased sense of immersion and a more natural virtual environment, can be realized (Deisinger et al., 1997). In addition, scaling the PHANToM's working volume to match a defined portion of the projection environment's

much larger space will permit a qualitative look at the effects of differing workspace sizes on the use of haptic feedback for a defined task.

CHAPTER 3

PROBLEM STATEMENT AND SOLUTIONS

In order to integrate some form of haptic force feedback with a projection screen virtual environment, and to explore the benefits for engineering applications, several technical obstacles must be overcome. While an ideal solution would permit total freedom of user movement, direct interaction of the user's body with the virtual geometry, and no loss of immersion, such designs are unattainable with the current state of haptic technology. The solutions presented here attempt to compromise on the most difficult issues while still providing useful haptic interaction with the virtual environment.

Problems to Overcome

Though it has been stated that this research uses the PHANToM haptic device, other devices were researched and several objectives considered before the PHANToM was selected. While many haptic devices exist, most remain research tools designed for a particular task. They are limited in availability and versatility, and are often difficult to program. The device chosen for this research should be commercially available, adaptable to many different tasks, and easy to program. This will allow the finding of the research to be readily usable by many other researchers.

The necessity of interacting with different geometry types while maintaining haptics update rates (around 1000Hz) must be considered. Constructive solid geometry, parametric models, and other geometry formats should be "touchable" by the chosen haptic device to provide versatility. Haptic interaction with the geometry should take place at kilohertz update rates, since that is important for stable haptic perception and safety. There should be some way to monitor these update rates and stop the simulation if they cannot be maintained.

The physical size and available workspace of the haptic device must be considered. Any mechanism capable of providing force feedback to the user's whole arm throughout its entire range of motion while in a projection screen environment will be so physically large or encumbering that it would obscure the immersion sense provided by the virtual environment. A smaller haptic device, while more mobile and less intrusive to the user, will have a more limited workspace, preventing the user from interacting with all portions of the virtual environment. This potential disparity between the size of the virtual environment (roughly 10x10x10 ft) and the workspace of the haptic device needs to be addressed.

The software used to control the haptic device needs to integrate well with the software running the projection environment. Both programs should be flexible enough to allow the programmer to customize them, yet they should also abstract the operation of the hardware enough to avoid tedious programming. The number of necessary software changes to run different application types must be kept to a minimum.

Solutions Presented

In selecting a suitable haptic device availability, versatility and the quality of force feedback should be considered. Initially a body grounded haptic glove, such as Immersion Corporation's CyberGrasp (Immersion 2002) or Sarcos's Dextrous Hand (Sarcos 2002), might appear to be an excellent solution since it provides a large workspace with minimal obstruction to the user. Unfortunately, the lack of force feedback to the user's entire arm does not provide information such as the perceived weight and inertial properties of an object. In design applications, these forces are important. A larger whole arm force feedback mechanism such as the Sarcos Dextrous Arm Master (Nahvi, Nelson et al. 1998) is also unacceptable since it must be mounted to the ground and cannot be easily moved, thus restricting the user to a single location. For this research, SensAble Technology's three-degree of freedom PHANToM force feedback device was selected.

The PHANToM

Though the PHANToM's desktop device nature and limited workspace appear to be unsuitable for use in a projection screen environment, it has several advantages. Since it is one of the most successful commercial haptic devices (Perkowitz, 1999), the PHANToM is readily available. The low inertia and static friction of the device, combined with a peak resistance of 10N, gives a realistic portrayal of free space and stiff objects (Massie and Salisbury, 1994). Another benefit of using the PHANToM is its continuing development by SensAble. A more advanced six-degree of freedom model, which uses the same software, is also available. The potential exists to upgrade in the future with minimal changes to existing work.

The PHANToM is relatively easy to control with SensAble's GHOST software toolkit. GHOST provides a great deal of flexibility while protecting the programmer from the hardware control details. To ensure good force feedback, GHOST controls the PHANToM with a separate haptic process dedicated to maintaining haptic update rates. GHOST's object-oriented nature also makes it easy to use a single piece of code with several different applications.

Using GHOST allows the PHANToM to interact with many geometric primitives, as well as more generalized shapes represented by a mesh of points. Such a mesh may be obtained from solid geometry, parametric surfaces, or a simple data set, providing great flexibility. Use of the PHANToM directly on parametric NURBS surfaces has also been demonstrated, though it involves some lower level programming in addition to GHOST (Thompson and Cohen, 1999). The potential to use Boeing's VPS software with haptic feedback in a projection screen virtual environment for virtual assembly is another incentive for choosing the PHANToM.

Phantom Stand

Since the PHANToM is intended as a desktop device, physically placing it in the projection screen environment in a useful way presents some challenges. The PHANToM must be easily movable, so a user can freely position it, yet stable enough to effectively support the PHANToM when force feedback is applied. Its height should also be adjustable for users of varying size. To achieve this, a stand specifically to hold the PHANToM was designed and built. This stand rolls about on four castor wheels, which may be locked to keep the stand from moving when the PHANToM is in use. Stand height is adjustable from 28 to 42 inches to accommodate different users and postures. This also helps the PHANToM to be positioned out of the user's direct sight during use, to preserve a sense of immersion in the environment. Since knowing the orientation of the phantom stand may be desirable, magnetic tracking devices, as are typically used in projection screen environments, should be compatible with this stand. Since magnetic materials adversely affect the accuracy of such trackers, the phantom stand was constructed out of bonded PVC plastic and stainless steel hardware. Weight was also added in the base to prevent excessive force on the PHANToM from tipping the stand over. Figure 3.1 diagrams the phantom stand, while a 3D CAD model of the stand appears in Figure 3.2

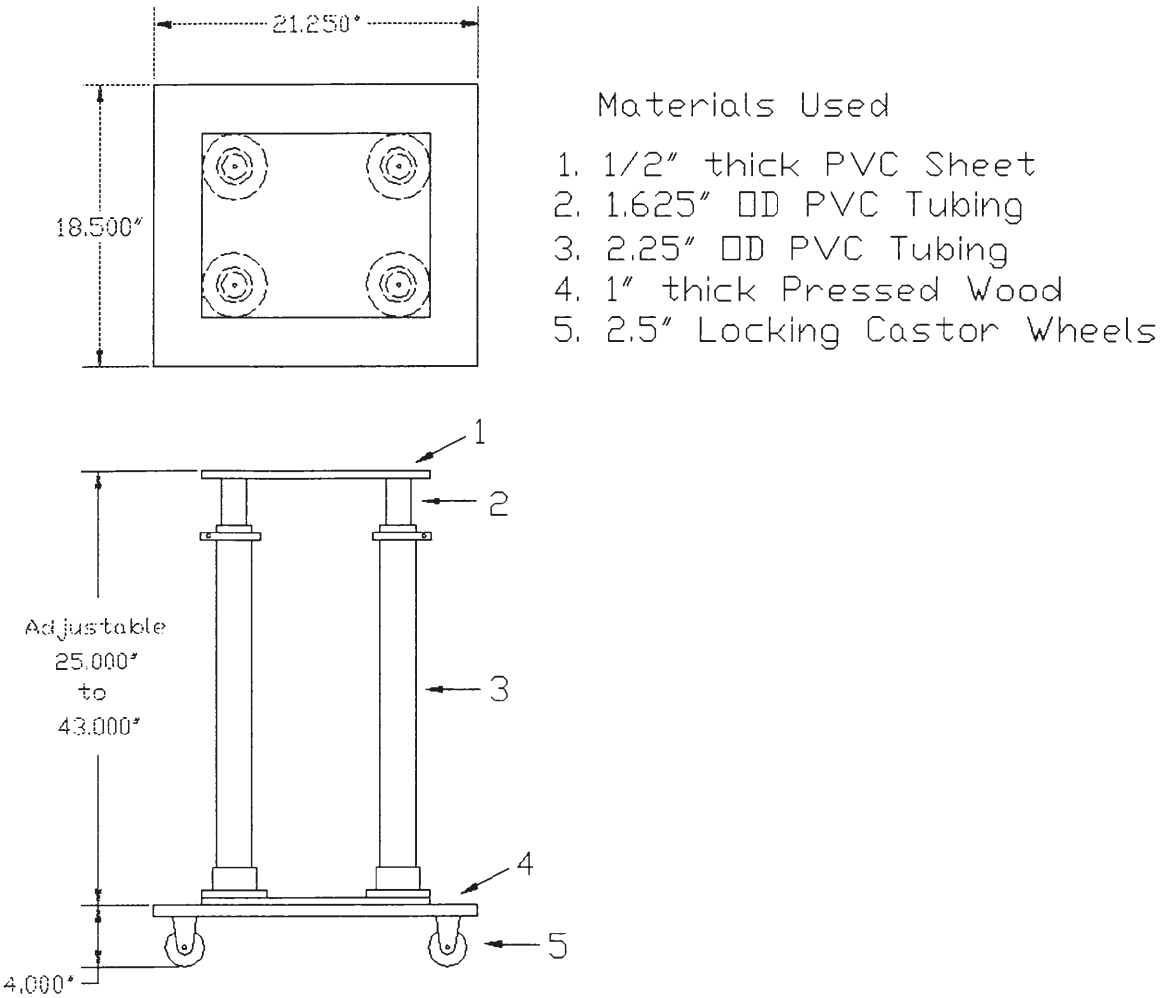

Figure 3.1. The phantom stand

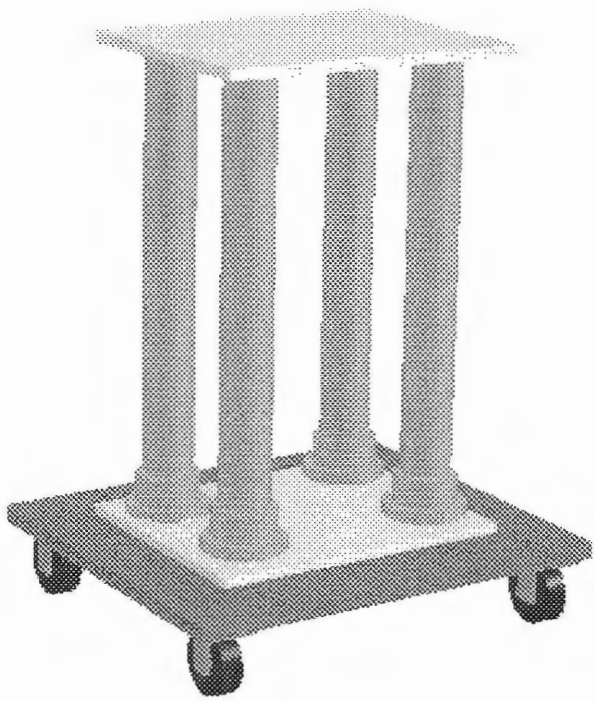

Figure 3.2. Computer model of the phantom stand

Phantom Volume

A major concern with the PHANToM is its limited workspace when compared to the volume of a projection screen environment. To address this issue, the concept of a phantom volume is presented. This is a user defined rectangular volume of space in the virtual environment that correlates motion of the PHANToM's physical endpoint to a virtual position in the environment. A user defines the volume by selecting two opposite corners in the virtual space with a wand. The known orientation of the phantom stand is used to orient the volume. To aid selection, the volume is dynamically drawn between the first point and the current wand position until a second point it chosen. Figure 3.3 shows a phantom volume in the virtual environment. When using the PHANToM, the virtual endpoint position is

confined to the phantom volume, just as the physical endpoint is confined by the PHANToM's physical workspace. Motion of the actual PHANToM is scaled to match motion of the virtual position. This way the limited PHANToM physical working volume can be matched and used over an arbitrarily large space in the virtual environment. Details on the implementation of this phantom volume appear in chapter 4.

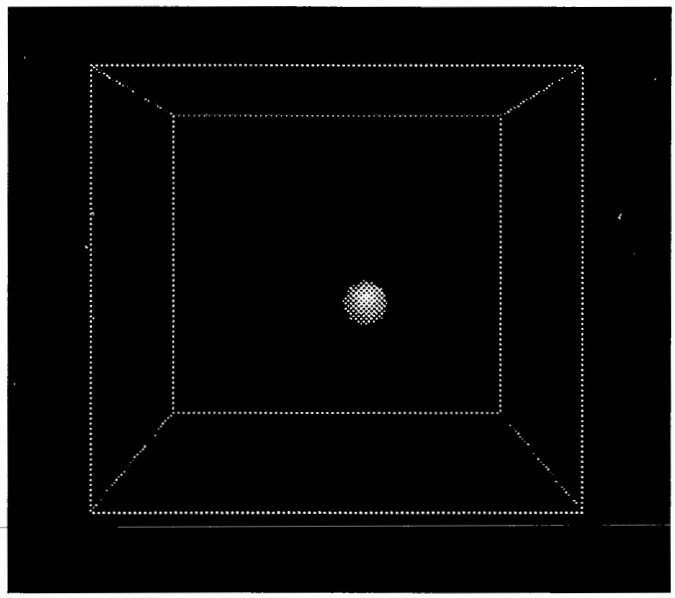

Figure 3.3. A phantom volume in the virtual environment

Virtual Reality API

An additional software package must be chosen to operate the virtual environments with the PHANToM. The software selected was vrJuggler, a complete framework for developing virtual reality applications that may be used on a variety of virtual reality devices, from head-mounted displays to multi-screen projection environments (Bierbaum et al., 2001). Since vrJuggler handles the hardware details and lets programmers focus on developing their applications, it makes an excellent complement to the GHOST software, which takes a similar approach to the operation of the PHANToM.

Finally, the solutions listed above need to be combined and implemented in a computer application to be useful. This controlling program must create the virtual environment, display all relevant geometry, generate the phantom volume, translate models from the virtual environment to the haptic workspace, activate the PHANToM, and manage the application to which haptics are applied. The next section discusses the details of this PHANToM application.

CHAPTER 4

STRUCTURE OF THE PROGRAM

The application presented here is designed to provide a general framework for using the PHANToM in a projection screen virtual environment. Since the program code is object oriented, a variety of applications could use portions of this program to add haptic feedback to an existing application. Written in the C++ programming language, this application uses vrJuggler for controlling the virtual environment, GHOST for driving the PHANToM, and OpenGL for displaying the virtual world. The controlling program is constructed from three main classes: the CavePhantomApp, the PhantomVolume, and the PhantomDriver. First in this chapter is a description of the hardware used to integrate the PHANToM into a projection screen environment. Next is a discussion of the components of the three main program classes.

Hardware

In this research, the PHANToM was implemented in the C6, located at Iowa State University's Virtual Reality Application Center. The C6 is a 10ft by 10ft by 10ft six-sided projection screen virtual environment capable of immersing the user in a virtual world. Figure 4.1 shows the exterior of the C6 environment.

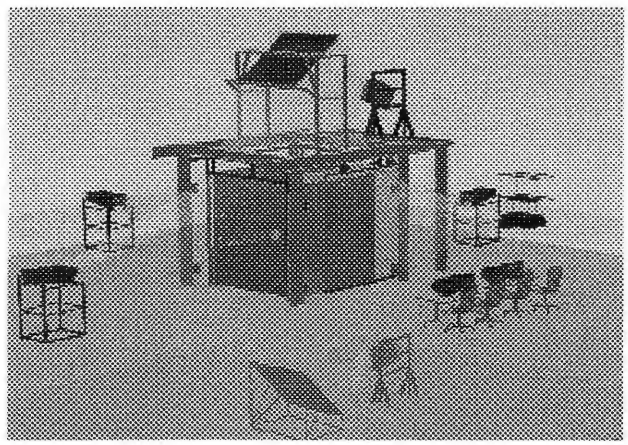

Figure 4.1. The C6 at Iowa State University

An SGI Onyx2 InfiniteReality2 Monster computer with six graphics displays, 24 processors, and 12 gigabytes of memory drives the C6. The six graphics displays or "pipes" connect to six BarcoReality projectors, one for each rear-projected screen. An Ascension Technologies Motionstar magnetic tracking system provides information on the location and orientation of objects inside the C6. To produce stereo, each wall projector rapidly switches between the left eye and right eye viewpoint displays while the users wear wireless CrystalEyes shutter glasses that alternately block the image from reaching the incorrect eye. One pair of glasses is position tracked to determine a reference point for drawing the virtual environment. This results in the display of a stereo three-dimensional image. A wireless position-tracked wand with several buttons allows a user inside the C6 to interact with the virtual environment (VRAC, 2001).

Actually using the PHANToM inside the C6 requires it be placed on and secured to the phantom stand. The stand may then be moved about inside the C6 to a suitable location where the stand wheels are locked and the height adjusted for use. The control box that drives the PHANToM is placed outside the environment and a cable connects the two devices. This control box is also connected directly to the Onyx2 via another cable and a specialized computer card. Figure 4.2 details these connections. A magnetic tracker is placed on the side of the phantom stand to obtain information about its orientation.

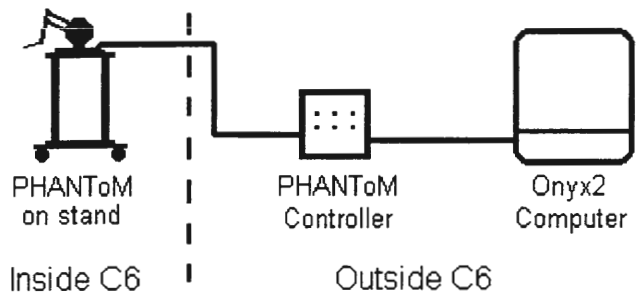

Figure 4.2. Connections to use the PHANToM in the C6

CavePhantomApp

The main application class that starts and controls the whole program is called the CavePhantomApp. The only part of the program dealing directly with vrJuggler, the CavePhantomApp generates the virtual environment, handles user actions and movement, displays and controls the menu system, and launches other portions of the application.

On startup, the CavePhantomApp declares an instance of each program class and sets all necessary parameters. Several OpenGL calls are made to properly display the virtual world. All geometry needed for the demonstration applications is also loaded at this time and displayed at various locations in the virtual world. A group of menus is then created and placed in front of the user's face. Since the user's head location and orientation are tracked, the menus follow the user and remain easily visible. The user has the ability to hide the menus by pressing a wand button.

These menus represent all the options the user has while in the virtual environment including: navigation, building and manipulating the phantom volume, starting a demonstration application, resetting all parameters, and exiting the program. The user chooses between these menu options with buttons on the wand. To avoid an excessive number of options appearing at any one time, sub-menus for the phantom volume and the demonstration applications exist and may be accessed from the appropriate choice on the

main menu. Since the menus are generated from a separate class, it's easy to alter and expand them.

While physically walking about in the C6 moves the user through the virtual world, the range of travel is limited by the 10ft by 10ft floor of the C6. To fully explore the virtual environment, a form of navigation is needed. This navigation essentially moves the entire world around the user's coordinate system, creating the impression of traveling through the environment. The CavePhantomApp's *navigate* option allows the user to "fly" through the world by pointing the wand in the direction of desired travel, including up and down. Buttons on the wand provide a way to increase and decrease speed, as well as immediately stop.

Once the user is sufficiently close to the desired geometry, a phantom volume may be drawn with the *phantom volume* menu option. The user inputs two opposite corners for the volume with the wand, and the CavePhantomApp passes these wand positions and the phantom stand's orientation to the PhantomVolume class. The completed volume may be translated about the environment, scaled to fit a larger space than what the user could reach, or deleted altogether.

Once a phantom volume exists, navigation is disabled to avoid the possibility of navigating about while the PHANToM is in use. Allowing navigation with a phantom volume present would force a choice between leaving the volume in place relative to the world coordinate system or moving it about with the user's coordinate system. Either option would be undesirable, since the former would cause a loss of alignment between the volume and the actual PHANToM orientation. The latter might necessitate moving world geometry through the volume while the PHANToM is in use, an operation that would produce confusing haptics and may not always be possible at haptic update rates.

With a phantom volume created and properly positioned, the user may choose to start a demonstration with the *use phantom* menu option. Selecting a particular application from

the *use phantom* sub-menu causes the CavePhantomApp to obtain the necessary information from the PhantomVolume class and pass it to the PhantomDriver class with instructions to setup haptics for the particular demonstration application. Any other specific parameters are also passed along at this time.

The remaining menu options are *reset* and *exit*. *Reset* is similar to restarting the application. The user is placed at the center of the world, if a phantom volume exists it is destroyed, and any changes made to the scene geometry are undone. *Exit* simply quits the entire application.

PhantomVolume Class

The PhantomVolume class contains the information about and functions for the phantom volume. This volume defines the region of virtual space the PHANToM's actual motion will be mapped to. Geometry located inside the phantom volume will be capable of interaction with the PHANToM.

Building the phantom volume requires three pieces of information: two points in space and a rotation value. The two points, P1 in equation 4.1 and P2 in equation 4.2, are selected by the user's wand and define opposite points of the volume. These points are stored as four-dimensional vectors to permit transformation with a 4x4 transformation matrix. The CavePhantomApp passes these points to the PhantomVolume class. The rotation value, theta θ, represents the orientation of the phantom stand in the physical world. Since the stand can only rotate around the vertical axis, its orientation can be represented by the degrees of rotation away from the front wall of the C6. The user should ensure the phantom stand is in the proper location and its wheels locked before defining the volume, as the first point of the volume and the degrees of rotation are passed to the PhantomVolume class together. Failure to secure the stand could result in the motion of the PHANToM in the volume not corresponding to its physical motion.

$$P1 = \begin{bmatrix} P1x \\ P1y \\ P1z \\ 1 \end{bmatrix} \tag{4.1}$$

$$P2 = \begin{bmatrix} P2x \\ P2y \\ P2z \\ 1 \end{bmatrix} \tag{4.2}$$

The PhantomVolume's position and dimensions are stored as a transformation matrix and a point vector. The transformation matrix, TM in equation 4.3, contains both the rotation of the volume about the vertical axis and a translation to the point P1. By inverting the transformation matrix and multiplying that inverse by the second selection point as seen in equation 4.4, the point vector, PV in equation 4.5, is obtained. As a result, the phantom volume exists in its own coordinate system as an axis aligned box with the origin and the point vector at opposite corners.

$$TM = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & P1x \\ 0 & 1 & 0 & P1y \\ -\sin(\theta) & 0 & \cos(\theta) & P1z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

$$PV = TM^{-1} P2 \tag{4.4}$$

$$PV = \begin{bmatrix} P2x\cos(\theta) - P2z\sin(\theta) + P1z\sin(\theta) - P1x\cos(\theta) \\ P2y - P1y \\ P2x\sin(\theta) + P2z\cos(\theta) - P1z\cos(\theta) - P1x\sin(\theta) \\ 1 \end{bmatrix} \tag{4.5}$$

This matrix/vector representation of the phantom volume makes translating and scaling easy. To translate the volume, the program adds the desired translations in x, y, and z to the P1x, P1y, and P1z values of the transformation matrix. Scaling the volume is accomplished by multiplying the x, y, and z values of the point vector by a scalar value. Positioning the virtual PHANToM position to match the phantom volume is also simplified, since the transformation matrix and point vector can be used directly by GHOST in the PhantomDriver.

PhantomDriver Class

The PhantomDriver is responsible for starting and controlling the haptic process that drives the PHANToM. It combines information from the phantom volume with properties specific to the selected demonstration application and uses the GHOST toolkit to build the haptic environment.

Calling the PhantomDriver class to begin a specific demonstration application requires three pieces of information: 1. The location and dimensions of the phantom volume object in world coordinates. 2. The size of workspace the actual PHANToM device will be allowed to move within. 3. Any parameters specific to the current application.

All necessary information about the phantom volume's position and size can be obtained from the volume's transformation matrix and point vector. The translations and orientations for one corner of the volume are stored in the transformation matrix, while the three values in the point vector represent the size of the volume along the x, y, and z-axis. When the PhantomDriver is called, the transformation matrix and point vector are obtained from the PhantomVolume class by the CavePhantomApp and passed as arguments to the PhantomDriver.

The PHANToM device's physical workspace is a box that limits the travel of the PHANToM's physical endpoint. Defining the size of this workspace is required by GHOST,

and may depend on the specific PHANToM model used, the type of application, and the user's preference. In the PhantomDriver class this workspace size is represented by the *max_workspace_size* parameter. In some cases it is useful to confine the PHANToM endpoint to a box of modest size, ensuring that the user doesn't run out of device travel or collide the endpoint with the bulk of the physical PHANToM. At other times it may be desirable to make the physical workspace limits much larger, allowing the PHANToM free movement throughout its entire range of travel. For the bulk of this research, the *max_workspace_size* is set to 120.0 millimeters, which prevents the PHANToM endpoint from colliding with the physical device or the phantom stand.

Since the phantom volume isn't constrained to be a cube, it may be different sizes in the x, y, and z directions, provided all are non-zero. In the PhantomDriver, the physical workspace confining the PHANToM endpoint is created to match the form of the phantom volume, so a single scale value suffices to match the motion of the PHANToM's physical endpoint to the virtual world. Thus the *max_workspace_size* scales to the largest dimension of the phantom volume, while the two other workspace sizes are smaller in proportion to the other dimensions of the phantom volume. These phantom volume dimensions are obtained from the point vector which is passed to the PhantomDriver from the PhantomVolume class. Dividing the *max_workspace_size* by the largest dimension of the point vector produces the world coordinate system to haptic coordinate system scale value called *world_haptic_scale*. Equation 4.6 shows the computation of *world_haptic_scale*.

$$world\_haptic\_scale = \frac{max\_workspace\_size}{PV[largest\_dimension]} \tag{4.6}$$

The PhantomDriver's next step is to construct any haptic geometry required by the application. This geometry may consist of simple primitives and/or polygonal meshes, as

these are the geometry types GHOST can use. Geometry may be passed to the PhantomDriver class from the CavePhantomApp or read in from a file.

Once the Phantom Driver has the workspace set up and the geometry loaded, it uses GHOST to build the haptic scene graph. A diagram of this scene graph appears in Figure 4.2. Below the name of each scene graph node is the type of GHOST object representing that node. Note that since the haptic geometry may be any of several different GHOST objects, the specific type isn't listed.

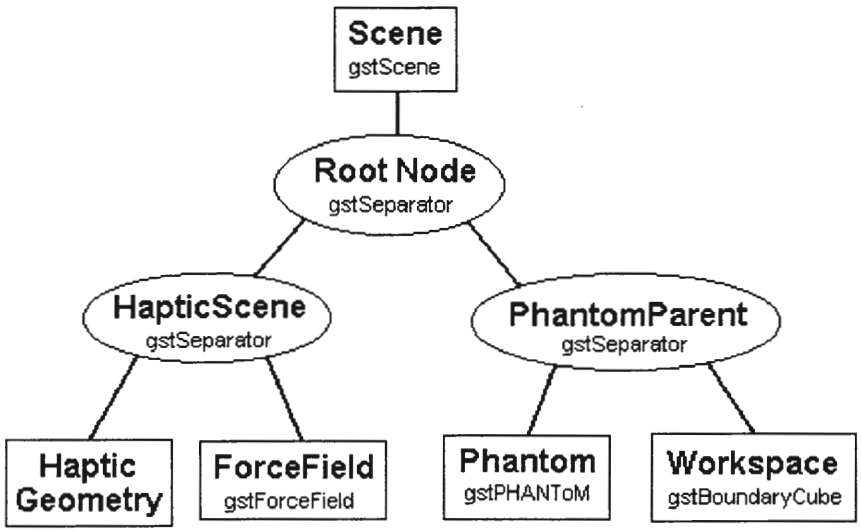

Figure 4.3. The PhantomDriver class's haptic scene graph structure

The Scene and Root Node form the basis for the haptic scene graph and are required by GHOST. Both the HapticScene and the PhantomParent are separators that allow certain actions to be performed on their members without affecting the rest of the scene. The HapticScene contains all haptic geometry in its many forms and a ForceField object, which is used by certain applications. The PhantomParent separator includes the actual PHANToM object (labeled Phantom), which contains the physical location of the PHANToM device, and the Workspace that defines the physical bounds for the PHANToM endpoint.

A key feature of the PhantomDriver's scene graph is the placement of the Phantom and the Workspace objects into the PhantomParent separator. This allows the position of the PHANToM device and its workspace to be translated and rotated in the haptic space to match the position of the phantom volume simply by applying the proper transformations to the PhantomParent, since moving the PhantomParent separator moves both the PHANToM and it's Workspace.

The PhantomParent must be translated to the center of the phantom volume so that when the PHANToM is initialized to the center of its physical workspace, the virtual PHANToM position is initialized to the center of the phantom volume. This is accomplished by dividing the x, y, and z components of the phantom volume's point vector by two, to obtain the center of the volume in its own coordinate system, and then transforming it by the transform matrix to obtain the volume's center in world coordinates. The resulting center vector, which gives the center point of the phantom volume in world coordinates, is placed into the translation section of the transform matrix to produce the alignment matrix (AM) in equation 4.7. Since the transform matrix already contained the phantom volume's rotation θ, applying the alignment matrix to the PhantomParent aligns the PHANToM and it's Workspace with the phantom volume.

$$AM = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & \frac{1}{2}(P2x + P1x) \\ 0 & 1 & 0 & \frac{1}{2}(P2y + P1y) \\ -\sin(\theta) & 0 & \cos(\theta) & \frac{1}{2}(P2z + P1z) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

Scaling of the PHANToM's movement in its workspace to match the phantom volume is actually accomplished by applying a scale function (provided by GHOST) to scale the HapticScene node by the *world_haptic_scale* value. This appears unintuitive, as it would seem simpler to scale the PhantomParent instead, but scaling a separator with the Phantom

object beneath it produces unreliable results, including oscillations of the PHANToM. Since the Phantom can only deal with the Haptic Geometry through the HapticScene separator, scaling the HapticScene still produces the desired result. The only additional step required with this approach is scaling the position of the PHANToM endpoint to world coordinates if it is ever accessed directly by another class. This is accomplished by using a PhantomDriver function, which applies the proper scaling of 1/*world_haptic_scale*, to obtain the PHANToM endpoint position in world coordinates.

With the PHANToM positioned to match the phantom volume and all geometry scaled accordingly, the Phantom Driver initializes the PHANToM and starts the haptic servo loop process. This servo loop runs at roughly 1000 Hz and is responsible for keeping track of the position of the PHANToM and forces applied to the PHANToM to ensure smooth haptic feedback (SensAble, 2001). As seen in Figure 4.3, GHOST sets up this loop as a separate process and handles communication between the user's application and the haptic servo loop. In the CavePhantomApp program this communication between processes occurs once every graphics frame, roughly 30 times a second.

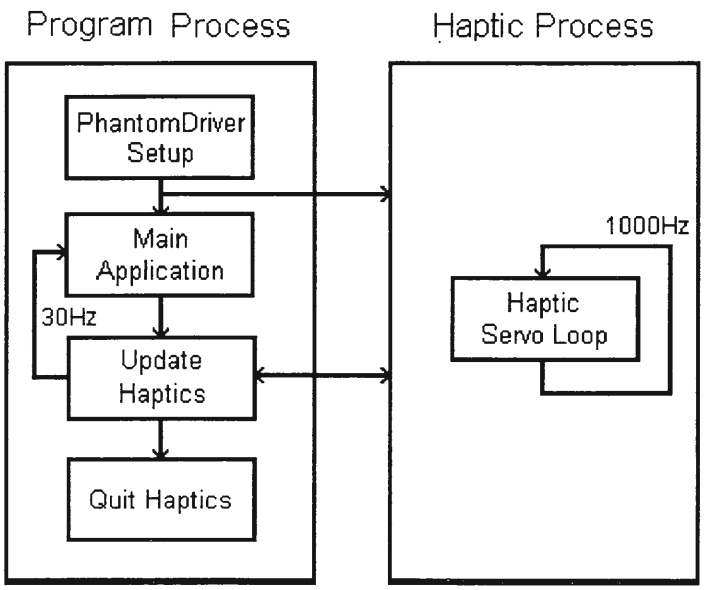

Figure 4.4. Communication between the main program and haptic process

Once the haptic process is running, any changes or commands given to the haptic simulation are communicated through the PhantomDriver program to the servo loop. These commands may include changing the scene geometry, applying a specific force to the PHANToM endpoint, or halting the haptic process when the application ends.

The classes described here all operate through the CavePhantomApp to run applications utilizing the PHANToM in a projection screen virtual environment. Two example programs, exploring a simple NURBS surface and performing a virtual assembly project, are discussed in the following chapter.

CHAPTER 5

EXAMPLE APPLICATIONS

The two example programs presented in this chapter each have a different purpose. The first, a NURBS surface that may be felt with the PHANToM, is a simple demonstration of using the phantom volume on a portion of geometry in the virtual world. The second is a virtual assembly application that allows the user to insert a rudder pedal assembly into the lower front portion of an aircraft. This task provides a good example of how haptic feedback and a projection screen virtual environment could assist in the evaluation of assembly tasks during the design process.

NURBS Surface Example

Since NURBS have become the standard for representing curves and surfaces in computer aided design (Piegl and Tiller, 1997), developing a way to haptically interact with NURBS is an important step in adding force feedback to the virtual design process. This example demonstrates how to use the GHOST software to create the haptic representation of an existing NURBS surface from its equations. The geometry is then displayed in the virtual environment where the user may interact with any portion of the surface.

NURBS Background

A NURBS surface is defined by a group of parametric piecewise polynomial basis functions and a collection of weighted control points (Piegl and Tiller, 1997). The location of a point on a NURBS surface, $S(u,v)$, is a function of the sum of basis functions $N_i(u)$ and $N_j(v)$ and control points $P_{i,j}$ with weights $w_{i,j}$, as seen in equation 5.1. Here u and v are parameters ranging from 0 to 1 inclusive. With the control points, weights and basis functions known, a point on the surface may be computed as a function of parametric values

u and v. This permits a complex surface to be stored as a group of control points, weights, and the corresponding basis functions instead of a large number of surface data points.

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_i(u)N_j(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_i(u)N_j(v)w_{i,j}} \tag{5.1}$$

Unfortunately, the GHOST software doesn't permit direct interaction with a NURBS surface from the mathematical representation. To avoid additional low level programming of the PHANToM, an intermediate surface model, composed of many small triangles, must exist for the PHANToM to interact with. This is analogous to the problem of rendering a NURBS surface in computer graphics. Since graphics hardware can only draw points, lines, and polygons, smooth surfaces must be approximated with many small polygons before they can be displayed (Woo et al., 1999). In the NURBS example application this approximation process is handled by the NurbSurface program class.

Example Program

The NurbsSurface class is the portion of the program responsible for generating and displaying a NURBS surface that may interact with the PHANToM. When the CavePhantomApp starts, an instance of the NurbSurface class is declared and a surface created. This surface is generated from a predefined square grid of equally weighted control points in the x-z plane with varying height or y-values. Figure 5.1 shows a simple NURBS surface and its corresponding control points as small spheres. Since supplying only the control points isn't enough information to calculate a unique NURBS surface, additional information is needed about the basis functions. This information is provided by two knot

vectors, one for each parameter, which determine the exact form of the basis functions. In this example application these knot vectors are already defined.

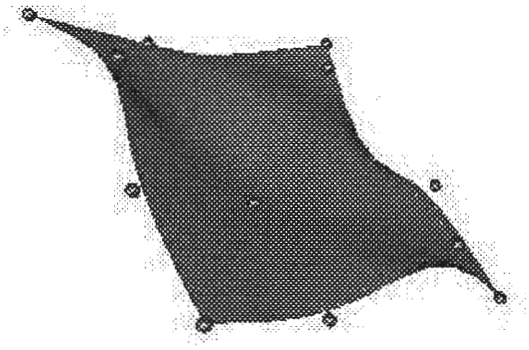

Figure 5.1. A NURBS surface and its control points

To render this surface, equation 5.1 is solved repeatedly for several u and v values ranging from 0 to 1 to obtain a set of $S(u,v)$ values. The $S(u,v)$ values represent points on the surface and are used to draw a series of triangles representing the surface. Using the PHANToM requires a similar list of triangles representing the actual surface. However, the number of triangles GHOST can haptically render and still maintain kilohertz update rates is much smaller than the number of triangles a modern graphics display can draw with at least 30Hz update rates. This difference may require the list of triangles used for the haptic surface to be smaller, and therefore each triangle larger and the surface approximation coarser, than those used in graphical rendering. The list of haptic surface points is generated and stored by the NurbsSurface class when the surface is created.

Once the user has defined a phantom volume around some portion of the NURBS surface in the virtual environment, the NURBS demo may be selected from the menu options. Starting the NURBS example causes the CavePhantomApp to pass the PhantomDriver a pointer to the list of haptic surface points. The PhantomDriver uses a GHOST routine to traverse this point list and generate a triPolyMesh geometry object. The

triPolyMesh is the most general geometry type available to GHOST. It represents a collection of triangles the PHANToM can touch. This triPolyMesh object is inserted into the haptic scene graph under the HapticScene separator node (see Figure 4.3) and the PHANToM's servo loop is started. The user may now explore the NURBS surface, as in Figure 5.2, until the example is terminated.

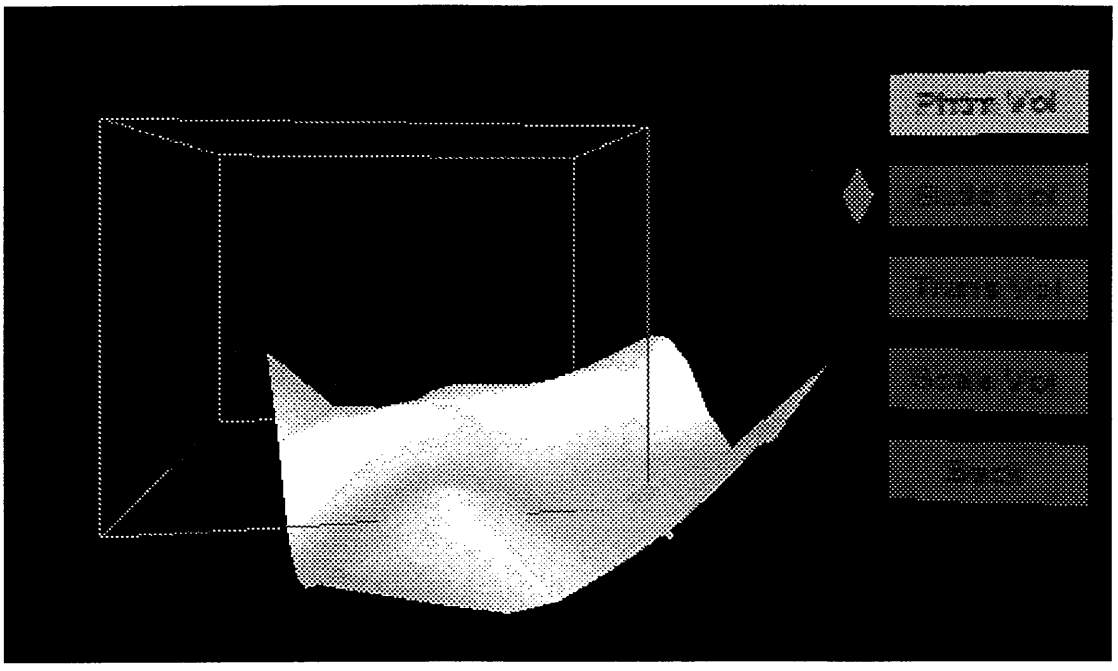

Figure 5.2. The NURBS surface example application

The NURBS surface example provides a simple demonstration of defining a phantom volume around certain geometry in the virtual environment and haptically interacting with that geometry. A technique for transforming a NURBS surface into a GHOST haptic mesh is also explained. Using this program lets the user develop a feel for using the PHANToM in the C6 while exploring the effects of large differences between the physical PHANToM workspace and the virtual phantom volume working space. The next example program provides a more powerful look at the benefits of haptic feedback in the projection screen virtual environment.

Virtual Assembly Example

Virtual assembly seeks to reduce or eliminate the need for physical mockups in the design process (McNeely et al., 1999). Since part assembly and maintenance are physical tasks, force feedback is an essential part of the virtual assembly process. This example application demonstrates a design for assembly task that would benefit from force feedback in the projection screen virtual environment: the installation of a rudder pedal assembly into the lower front portion of a small aircraft.

Background

Much of this example follows work done by researchers at Boeing who used a 6 degree of freedom PHANToM, their Voxmap PointShell (VPS) collision detection software, and physically based modeling to create a virtual assembly tool (McNeely et al., 1999). Functions from the VPS software library and some of the physically based modeling functions are used in this example. These were integrated into a separate object, the VPSpbm class, which interacts with the CavePhantomApp and controls the virtual assembly example.

The Voxmap PointShell software was developed as a fast collision detection method between complex objects. VPS is especially useful for haptics and other applications where speed is critical at the expense of some accuracy (McNeely, 2002). VPS discretizes polygonal objects into a set of voxels: small filled cubic regions of space. The size of these voxels determines the level of accuracy to which a collision may be detected. In the example presented here the voxel size is on the order of 0.5 inches. This is acceptable in assembly tasks, since it is common to provide at least this much clearance in part removal to allow for tools and the worker's hands (McNeely et al., 1999). Figures 5.3 and 5.4 below show the model of a rudder pedal assembly and its voxel representation in VPS.

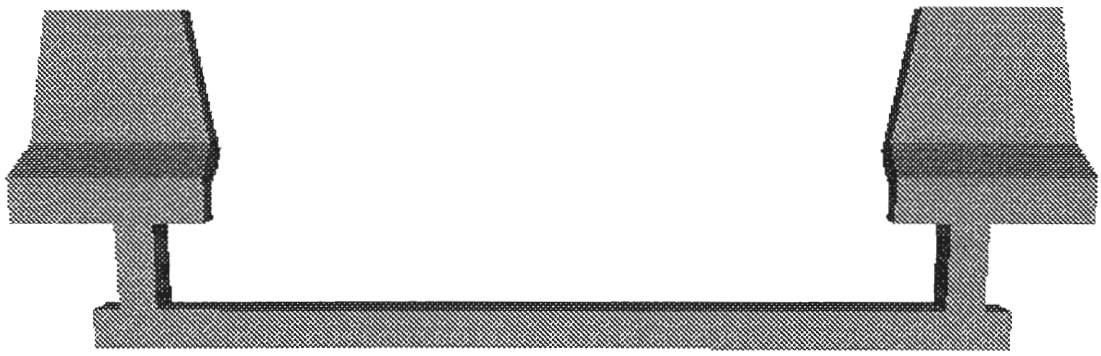

Figure 5.3. Solid model of a rudder pedal assembly

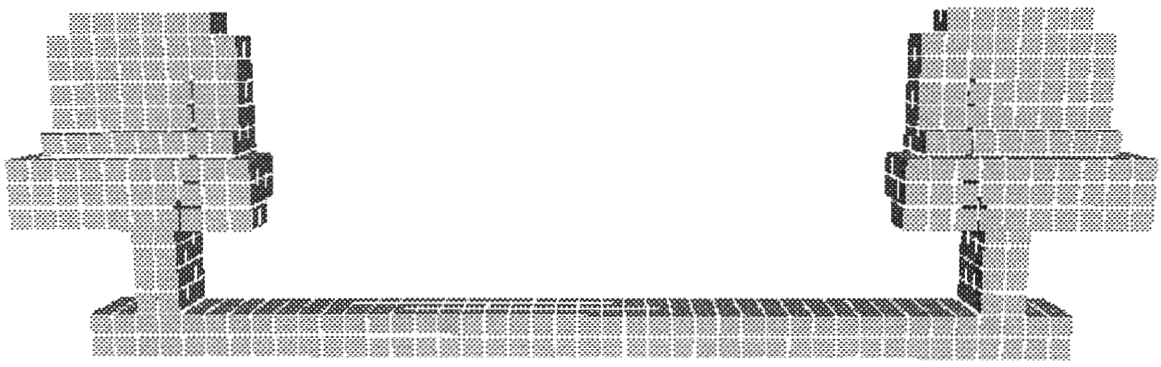

Figure 5.4. Voxel model of a rudder pedal assembly

To produce the forces and torques necessary for haptic feedback, some form of physically based modeling and a force model must be combined with the VPS software. This is accomplished by making a distinction between static and dynamic objects in the haptic environment and applying a special force model. The dynamic object is free to be manipulated by the haptic device, while the static object consists of everything else in the environment.

In VPS, a tangent-plane force model is used to calculate the contact forces. The voxel representation of the dynamic haptic object is used to produce a separate point shell

representation. This point shell is generated by assembling a shell of points on the surface of the dynamic object and inward surface normals for each point (McNeely et al., 1999). The associated pointshell for the rudder pedal assembly appears in Figure 5.5.

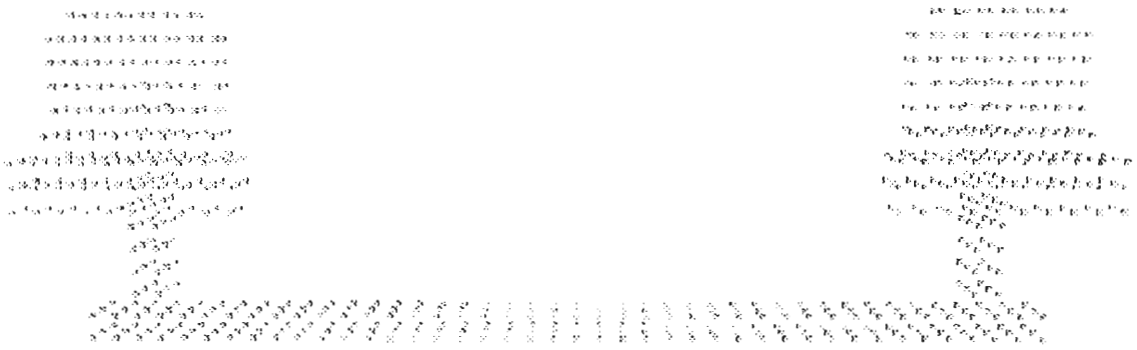

Figure 5.5. Point shell model of a rudder pedal assembly

When a point on the dynamic object's pointshell penetrates a static object's voxel, a depth of penetration is calculated. This depth represents how far the point has gone past the voxel's tangent plane which is a plane passing through the voxel center with a normal equal to that of the penetrating point (see Figure 5.6). The force **F** on the dynamic object's point is then equal to the penetration depth **d** times some stiffness value K via Hooke's law (equation 5.2). (McNeely et al., 1999). Here **F** and **d** are in the direction of the penetrating point's normal.

$$\mathbf{F} = K\,\mathbf{d} \tag{5.2}$$

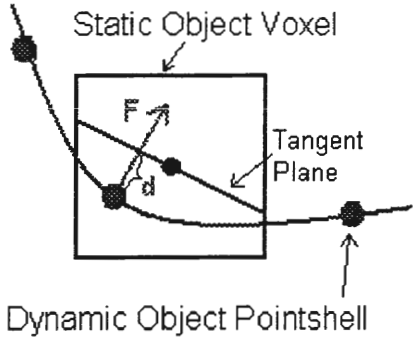

Figure 5.6. The tangent-plane force model

By summing the vector force contributions from all point-voxel penetrations for the dynamic object, a net force and torque may be calculated and returned to the haptic device (McNeely et al., 1999). Providing values for the mass and moment of inertia of the dynamic object permits realistic physically based modeling to be performed. One problem with the tangent plane method is that penetration of the models will likely occur before a resistive force is generated (McNeely et al., 1999). To prevent this undesirable result, two layers of additional voxels are placed on the outside of the static model. Offsetting the force generation by this amount prevents any possibility of model penetration, though it adds a small amount of additional inaccuracy to the collisions.

Program Structure

The VPSpbm class uses VPS, the physically based modeling method just described, and GHOST to create a virtual assembly example with haptic feedback. An instance of the VPSpbm class is created when the CavePhantomApp starts.

VPSpbm automatically loads the geometry files necessary for the example and voxelizes them with a series of VPS routines. Since the voxelization process can be time consuming depending on the geometry complexity and voxel size, it is performed before the immersive portion of the application starts to avoid leaving the user waiting in the virtual

environment. Currently only geometry files in the stereolithography (.stl) file format may be loaded. VPSpbm also handles displaying the loaded geometry in the virtual environment.

Once the user has defined a phantom volume around the relevant geometry and selects the *assembly demo* menu option, VPSpbm chooses the dynamic object, by default as the first geometry file loaded. In this application the first model is a rudder pedal assembly. Physical properties are applied to the dynamic object and VPS prepares to detect collisions between it and the static environment.

With the dynamic object ready, VPSpbm uses a PhantomDriver routine to start the haptic servo loop. While the GHOST scene graph structure now follows Figure 4.2, a key difference is the lack of any defined haptic geometry, as far as GHOST is concerned. Instead, VPSpbm queries GHOST for the PHANToM's endpoint position, performs the collision detection and physically based modeling to calculate the forces on the PHANToM endpoint, and applies those forces directly to the PHANToM through the ForceField object.

The dynamic object geometry is now attached to the virtual PHANToM endpoint position, represented by a small sphere. This attachment isn't rigid, however, it is modeled through a spring-damper connection between the PHANToM endpoint and the object. Such a connection is referred to as the "virtual coupler" scheme (Figure 5.7). The sphere represents the PHANToM endpoint. This helps prevent excessive force from being applied to the haptic device and improves haptic stability. The values for this spring-damper coupler are predefined in the VPSpbm. Values selected for this application are the same as those provided in the Boeing example (McNeely et al., 1999).

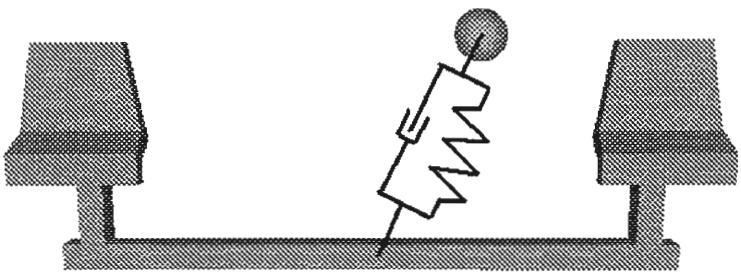

Figure 5.7. Diagram of the virtual coupler scheme

A key feature of the VPSpbm program is the addition of collision detection and physical modeling into the haptics servo loop while maintaining roughly a 1000Hz update rate. As the user moves the dynamic model about the environment, VPS keeps track of any collisions between geometry. Physical modeling provides the forces and torques on the model based on collisions as well as the inertial properties of the dynamic object. The VPSpbm class applies these forces directly to the PHANToM. The result is a fast and versatile way to manipulate a complex part through a large group of objects with haptic feedback.

Assembly Example

This virtual assembly example is designed to be similar to the type of problem a design engineer might face. The goal is to determine if an aircraft rudder pedal assembly can be inserted into its location below the instrument panel and next to the firewall without removing any control cables. Such removal of the rudder pedal assembly must take place with certain frequency for aircraft maintenance, and removing control cables can be a time consuming task.

The parts being modeled resemble those of a 1968 Cessna 177 Cardinal, a single-engine four-person aircraft. A picture of the area of the aircraft that was modeled appears in Figure 5.8. The rudder pedal group consists of two nearly identical assemblies located next

to the firewall. The CAD geometry representing these parts is shown in Figure 5.9. The geometry was created using IronCAD, a solid geometry modeling package, and exported as a series of stereolithography files.

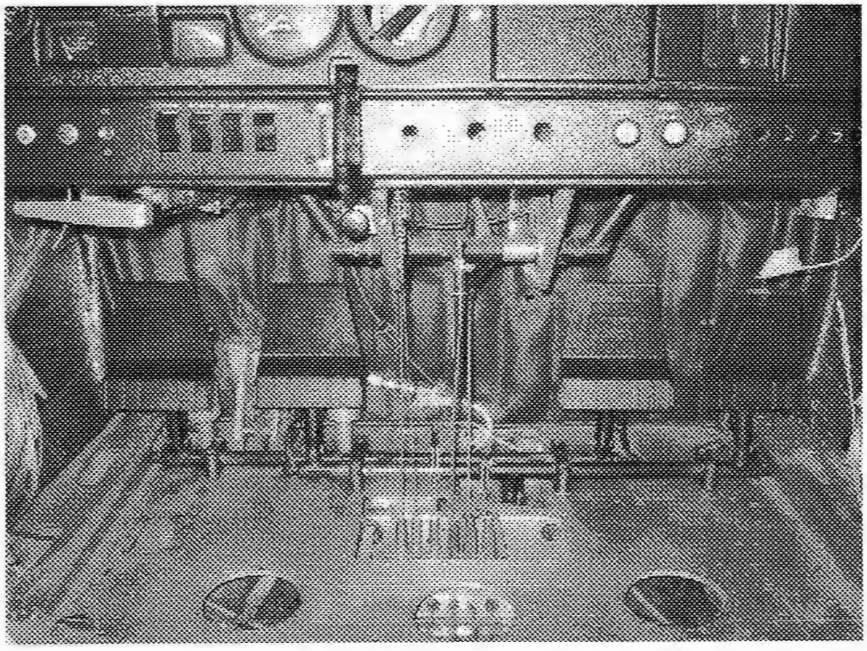

Figure 5.8. Picture of the aircraft portions used in the virtual assembly example

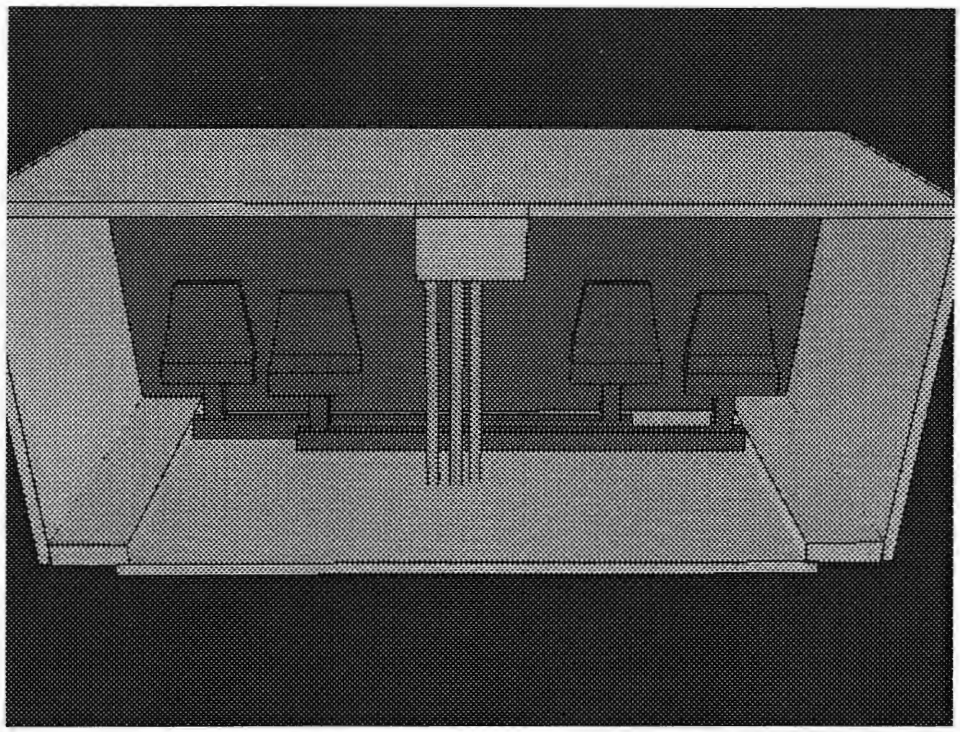

Figure 5.9. Solid model of the aircraft portions for the virtual assembly example

Implementing haptic feedback in a projection screen virtual environment shows several benefits in this example. It is fast and intuitive for a user to determine if the rudder pedal assembly may be inserted into the proper location around the control cables, which run vertically from the center of the cabin floor to the base of the instrument panel. Placing the pedals requires the assembly to be turned on its side, placed partway under the panel, twisted around the cables, and finally set upright. Manipulating the pedal assembly with the PHANToM and its haptic feedback is far easier than attempting to use the traditional keyboard and mouse approach.

The projection screen virtual environment allows any interference issues to be quickly spotted and explored. A user can actually place his/her head inside the geometry to get a better understanding of assembly fit. There is also a benefit for collaboration, since several designers could be in the same environment studying any problems that arise and ways to resolve them. Figure 5.10 shows the virtual assembly demonstration in use.

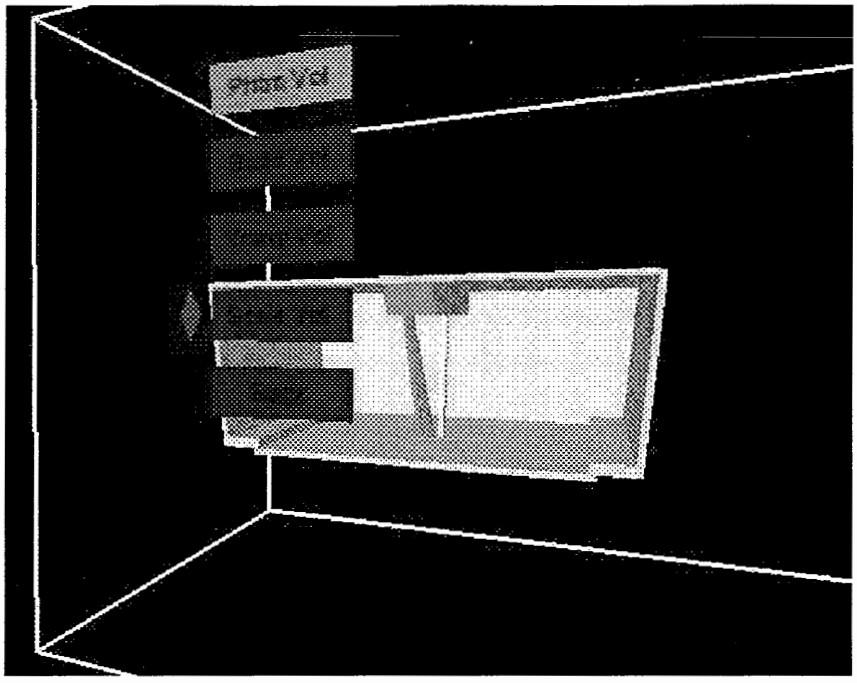

Figure 5.10. The virtual assembly example application

Overall, integrating force feedback and a projection screen virtual environment provides many benefits to the virtual assembly process. In addition to the rudder pedal assembly shown, the application could easily accommodate other geometry in order to prototype other assembly applications. The VPSpbm class permits existing CAD models to be loaded with a minimal amount of geometry translation and simplification.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This research achieved the stated goal of integrating haptic force feedback into a multi-screen projection screen virtual environment by placing a three-degree of freedom PHANToM haptic device into the C6. The application behind this integration combined several software packages including vrJuggler, GHOST, and VPS to explore the benefits haptic feedback can provide to various tasks. The concept of a phantom volume that mapped a portion of the virtual world to the motion of the actual PHANToM device was presented. A movable stand was designed and built to support the PHANToM and track its orientation in the virtual environment. Two example applications, loading a NURBS surface that interacts with the PHANToM and a virtual assembly task, demonstrated some uses for haptic feedback in the virtual environment.

Conclusions

After using this application and experimenting with the examples, some conclusions can be drawn about how force feedback and the projection screen virtual environment aid the user in different tasks.

1. Haptic feedback makes manipulating a complex part through confined spaces faster and more intuitive than using a standard keyboard and mouse approach. Attempting to perform the rudder pedal insertion example with a standard computer interface requires several different commands to manipulate the assembly into its proper location.
2. Being able to touch objects with the PHANToM provides additional information about the geometric structure, which may not immediately be noticeable visually.

Touching the NURBS surface provided feedback to the user on the actual shape of the surface.

3. The projection screen virtual environment makes interference issues encountered in a particular task, such as virtual assembly, easy to find and remedy. Complex structures are more easily understood when the user can look inside the geometry and observe any portion.
4. Since several people may observe the virtual environment and share the PHANToM in the same simulation, the projection screen environment enhances collaboration between users. All users in the environment can easily "feel" the benefits of haptic feedback by taking turns using the PHANToM.
5. For the examples presented in this work, the differences in workspace size between the physical PHANToM and the virtual phantom volume appear relatively unimportant. In both the NURBS surface and the virtual assembly application, the haptic feedback provides additional information about the geometry and/or task, even though the haptic feedback is scaled.

While the addition of haptics to the virtual environment improved the ability of users to interact with digital models for the examples presented, there are some guidelines that should be followed to unsure a high quality force feedback experience in a virtual environment. These guidelines include:

1. Avoid positioning much of the phantom in a location that does not align with the user's viewpoint. Looking in a direction different from the PHANToM's position while using it seems to make the haptics confusing as the user becomes unsure about which direction to move the PHANToM endpoint.
2. Avoid making the phantom volume excessively large, as this reduces the quality of the force feedback. This may be due in part to the fact that the user must look in different directions as the virtual PHANToM endpoint moves about the

phantom volume and in part to the high sensitivity of the virtual endpoint as it moves through space.

3. Avoid positioning the phantom stand in the direct line of the user's sight. Making the stand just high enough for the PHANToM to be comfortably reached and positioning it off to the right (or left) side of the user seems to work well.
4. Keep the PHANToM and its stand outside of the virtual geometry. Seeing the actual PHANToM within the virtual objects degrades the user's sense of immersion.

Future Work

Though this research succeeded in implementing haptic feedback into the projection screen environment, there are several areas where improvements could be made. Improvements to the CavePhantomApp program software would make it more powerful and general, while changes to the hardware would result in more realistic haptic feedback and a better sense of immersion. Some suggestions for future improvement appear below.

1. The NURBS surface portion of the program should be changed to load a more standard geometry format so that models created in a CAD package could be exported and loaded into a haptic form using GHOST geometry types.
2. Some type of model deformation, such as deforming NURBS geometry with some mathematical model for force feedback, would be another step toward adding haptics to interactive shape design. Edwards presents similar work in his research on force feedback with dynamic models (Edwards, 1998), so the potential exists to combine such work with GHOST and the PHANToM.
3. Using the virtual assembly example program makes a strong case for adding a six degree of freedom haptic device that can apply torques to the user's hand. Currently the user can rotate the object about three axes, but the lack of torque

feedback can be confusing. Since GHOST makes integrating a six degree of freedom PHANToM into the existing application straightforward, only a small amount of reprogramming would be necessary.

4. Though scaling the small PHANToM physical workspace to a larger volume in the environment works well for the applications presented, using a haptic device which has a larger physical workspace, such as Immersion's CyberForce, should be considered. Increasing the range of motion to the level of the user's arm motion may make the haptics even more convincing. Changing haptic devices, however, may require redesigning or reconsidering the phantom stand and the use of GHOST.
5. In this work the haptic servo loop runs as a separate process on the SGI Onxy2 computer controlling the simulation. By implementing the haptic process on a separate dedicated haptic computer it may be possible to handle more complex haptic geometry without falling below the necessary kilohertz update rates (Chen and Marcus, 1998). It would simplify the hardware, since the PHANToM and its dedicated computer could be connected to the Onyx2 through a standard network instead of using special hardware to directly connecting the PHANToM to the Onyx2.

Making the improvements listed above would result in better immersion and greater flexibility with the haptic feedback in a projection screen virtual environment. Even though haptic technology needs many improvements before it reaches the immersive level of current graphics displays, adding force feedback to a simulation still provides additional information to the user that can make certain engineering tasks easier and more efficient.

BIBLIOGRAPHY

Avila, R. S. and L. M. Sobierajski (1996). A Haptic Interaction Method for Volume Visualization. Visualization '96, San Francisco, CA, IEEE.

Bierbaum, A., C. Just, P. Hartling, K. Meinert, A. Baker, C. Cruz-Neira (2001). VR Juggler: A Virtual Platform for Virtual Reality Application Development. Virtual Reality, 2001, Yokohama, Japan, IEEE.

Burdea, G. C. (1996). Force and Touch Feedback for Virtual Reality. New York, John Wiley & Sons, INC.

Burdea, G. C. (1999). "Invited Review: The Synergy Between Virtual Reality and Robotics." IEEE Transactions of Robotics an Automation **15**(3): 400-410.

Burdea, G. C. (2000). Haptics Issues in Virtual Environments. CGI 2000, Geneva, Switzerland, IEEE.

Chen, E. and B. Marcus (1998). "Force Feedback for Surgical Simulation." Proceedings of the IEEE **86**(3): 524-530.

Cotin, S. and H. Delingette (1998). Real-time Surgery Simulation with Haptic Feedback using Finite Elements. IEEE International Conference on Robotics & Automation, Leuven, Belgium, IEEE.

Cruz-Neira, C., D. J. Sandin, T. DeFanti (1993). Surround-screen projection-based virtual reality: the design and implementation of the CAVE. International Conference on Computer Graphics and Interactive Techniques, ACM Press.

Deisinger, J., C. Cruz-Neira, O. Riedel, J. Symanzik (1997). "The Effect of Different VIewing Devices for the Sense of Presence and Immersion in Virtual Environments: A Comparison of Stereoprojections Based on Monitors, HMD, and Screen." Advances in Human Factors/Ergonomics **21B**: 881-884.

Edwards, J. C. (1998). Interactive synthetic environments with force feedback. Mechanical Engineering. Ames, IA, Iowa State University.

Frank, A. O., A. Twombly, T. Barth, J. Smith (2001). Finite Element Methods for real-time Haptic Feedback of Soft-Tissue Models in Virtual Reality Simulators. IEEE Virtual Reality, Yokohama, Japan, IEEE.

Grange, S., F. Conti, P. Rouiller, P. Helmer, C. Baur (2001). The Delta Haptic Device. Mecatronics 2001, Besancon, France.

Gupta, R., T. Sheridan, D. Whitney (1997). "Experiments Using Multimodal Virtual Environments in Design for Assembly Analysis." Presence **6**(3): 318-338.

Guthold, M., M. Falvo, W. Matthews, S. Paulson, S. Washburn, D. Erie, R. Superfine, F. Brooks, R. Taylor (2000). "Controlled Manipulation of Molecular Samples with the nanoManipulator." IEEE ASME Transactions on Mechatronics **5**(2): 189-198.

Hickey, T. (2001). SensAble Advances Its Touch-Enabled Modeling Technology for Industrial Designers. CAD/CAM, CAE: Survey, Review and Buyer's Guide. **13:** 1-4.

Hollerbach, J. M. (2000). Some Current Issues in Haptics Research. IEEE International Conference on Robotics & Automation, San Francisco, CA.

Hollerbach, J. M., E. Cohen, W. Thompson, R. Freier, D. Johnson, A. Nahvi, D. Nelson, T. Thompson (1997). Haptic Interfacing for Virtual Prototyping of Mechanical CAD Designs. ASME Design Engineering Technical Conferences, Sacramento, California.

Immersion (2002). CyberForce, Immersion Corporation. (Date Accessed: March 25, 2002) http://www.immersion.com/products/3d/interaction/cybergrasp.shtml

Immersion (2002). CyberGrasp, Immersion Corporation. (Date Accessed: April 15, 2002) http://www.immersion.com/products/3d.interaction/cyberforce.shtml

Johnson, D., T. V. Thompson, M. Kaplan, D. Nelson, E. Cohen (1999). Painting Textures with a Haptic Interface. IEEE Virtual Reality Annual Symposium, Houston, TX, IEEE.

Luecke, G. R., Y.-H. Chai, J. Edwards (1997). "Force Interactions in the Synthetic Environment Using the ISU Force Reflecting Exoskeleton." Computers and Graphics **21**(4): 431-442.

Massie, T. and J. K. Salisbury (1994). The PHANTOM haptic Interface: A Device for Probing Virtual Objects. ASME Symposium on Haptic Interfaces for Virtual Environments, Chicago, IL, ASME.

McNeely, B. (2002). Introduction to VPS, The Boeing Company. (Document with software) **2002**.

McNeely, W. A., K. D. Puterbaugh, J. Troy (1999). Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling. 26th Annual Conference on Computer Graphics and Interactive Techniques.

Nahvi, A., D. D. Nelson, J. M. Hollerbach, D. Johnson (1998). Haptic Manipulation of Virtual Mechanisms from Mechanical CAD Designs. IEEE International Conference on Robotics and Automation, Leuven, Belgium.

Pausch, R., D. Proffitt, G. William (1997). Quantifying Immersion in Virtual Reality. 24th annual conference on Computer graphics and interactive techniques, ACM Press.

Perkowitz, S. (1999). Feeling is Believing. New Scientist. **163:** 34-7.

Piegl, L. and W. Tiller (1997). The NURBS Book. Berlin, Springer.

Ramanathan, R. and D. Metaxas (2000). Dynamic Deformable Models for Enhanced Haptic Rendering in Virtual Environments. IEEE Virtual Reality, New Brunswick, NJ, IEEE.

Ryken, M. J. and J. M. Vance (2000). "Applying virtual reality techniques to the interactive stress analysis of a tractor lift arm." Finite Elements in Analysis and Design **35**: 141--155.

Salisbury, J. K. and M. A. Srinivasan (1997). "Phantom-Based Haptic Interaction with Virtual Objects." IEEE Computer Graphics and Applications **17**(5): 6-10.

Sarcos (2002). Utah/MIT Dextrous Hand Master, Sarcos. (Date Accessed: April 16, 2002) http://www.sarcos.com/interspec_ut_mithand.html

SensAble (2001). The GHOST SDK Programmer's Guide, SensAble Technologies, Inc. **2002**.

Sorid, D. (2000). The Virtual Surgeon. IEEE Spectrum. **37:** 26-31.

Thompson, T. V. and E. Cohen (1999). Direct Haptic Rendering of Complex Trimmed Nurbs Models. ASME Dynamic Systems and Control Division, Nashville, Tennessee, ASME.

Volkov, S. and J. Vance (2001). "Effectiveness of Haptic Sensation for the Evaluation of Virtual Prototypes." Journal of Computing and Information Science in Engineering **1**.

VRAC (Virtual Reality Applications Center) (2001). About the C6, Iowa State University. (Date Accessed: February 26, 2002) www.vrac.iastate.edu/about/facilities/c6

Vuskovic, V., M. Kauer, G. Szekely, M. Reidy (2000). Realistic Force Feedback for Virtual Reality Based Diagnostic Surgery Simulators. IEEE International Conference on Robotics & Automation, San Francisco, CA, IEEE.

Weghorst, S., J. Berkley, M. Ganter, H. Gladstone, G. Raugi, D. Berg (1999). Real-Time Finite Element Modelling with Haptic Support. ASME Design Engineering Technical Conferences, Las Vegas, ASME.

Woo, M., J. Neider, T. Davis, D. Schreiner (1999). OpenGL Programming Guide. Reading, Massachusetts, Addison Wesley.

Yeh, T.-P. and J. Vance (1997). "Combining MSC/NASTRAN, sensitivity methods, and virtual reality to facilitate interactive design." Finite Elements in Analysis and Design **26**: 161-169.