

# Whole-Hand Interaction with 3D Environments

M.R. Tremblay, C. Ullrich, D.H. Gomez, R. Carmel and J. Tian  
Virtual Technologies Inc.  
2175 Park Blvd., Palo Alto, CA 94306  
tremblay@virtex.com

## Abstract

*Graphical object manipulation in a computer-simulated environment holds much promise for increasing the productivity of users in a variety of occupations. Many of today's computer interfaces, such as mice and joysticks, are ill-suited for such a task and a better interaction paradigm is required. Once such approach is the use of a graphical hand which mimics the movement of the user's hand via an instrumented glove. While the concept of whole-hand interaction is intuitive and natural, its implementation is complex. This research note describe a VirtualHand® Toolkit being developed by researchers at Virtual Technologies, Inc. More specifically, it covers the ongoing development of a "Haptic Scene Graph," which can work in conjunction with a simulation's master scene graph. A demonstration of a functional implementation of the toolkit will be given at the workshop.*

## 1. Introduction

In the past year, computer users have witnessed a significant increase in 3D graphical capabilities. Computer games and engineering CAD software packages have been the driving force behind these advancements. Because of their inherent complexity, interfaces capable of interacting with these 3D environments have been slower to migrate from laboratories to the desktop. While the interfacing hardware has made some advances, the accompanying software required to interact with these environments has lagged behind. This is not surprising considering the complexity involved in physically modeling computer-generated environments so that they can provide real-time interactive capabilities. Nonetheless, companies producing some of the less-complex hardware interfaces have introduced Application Program Interfaces (API) and Software Development Kits (SDK) that facilitate the task of integrating their products into third-party applications.

One such company is Immersion Corp., which produces a line of force-feedback joysticks and has developed an API named I-Force™ [3] which helps developers simulate effects such as impacts, recoils, vibrations, springs, etc.

Another company is SensAble Technologies. SensAble markets a 3 degree-of-freedom (DOF) force-feedback stylus and an accompanying SDK named Ghost™ [4]. Ghost is an object-oriented toolkit that represents the haptic environment as a hierarchical collection of geometric objects (spheres, cubes, cylinders, etc.) and spatial effects (springs, dampers, vibrations, etc.)

Comparable software for whole-hand input interfaces has been slower to appear because of the inherent complexity of the whole-hand interaction paradigm, with its multiple constraints and degrees-of-freedom. Although whole-hand input devices present a greater challenge on the integration side, they also provide an extremely appealing advantage on the interaction side. What better way to manipulate virtual objects than to just "reach in and grab them?" For certain applications such as flight simulators and surgical simulations, joysticks and styluses can be very useful. In these instances, they provide realistic feedback at a level that cannot be attained with today's whole-hand interfaces. For many manipulation tasks however, they suffer from the inherent limitation of only providing a single point of contact. On the other hand, users wearing instrumented gloves, if properly implemented, can manipulate objects just as they would in the real world, i.e. with their hands. Of course, this type of interaction requires elaborate manipulation models which are more difficult to implement and require additional computing power.

Until recently, such software development efforts have resided in university and government laboratories. Examples include the work of Gomez et al. [2] at Rutgers University using the Rutgers Master II and the work of Coiffet et al. [1], at the Laboratoire de Robotique de Paris, using the LRP Hand Master. More recently, Luecke et al. have developed software to use an exoskeleton haptic device to apply virtual forces [3] and Thompson et al. [6] have used the Sarcos Dextrous Arm Master to perform direct parametric tracing on maneuverable NURBS models.

Commercially, two companies have introduced whole-hand software libraries: Virtual Technologies, Inc. offers its VirtualHand® software library for use with its

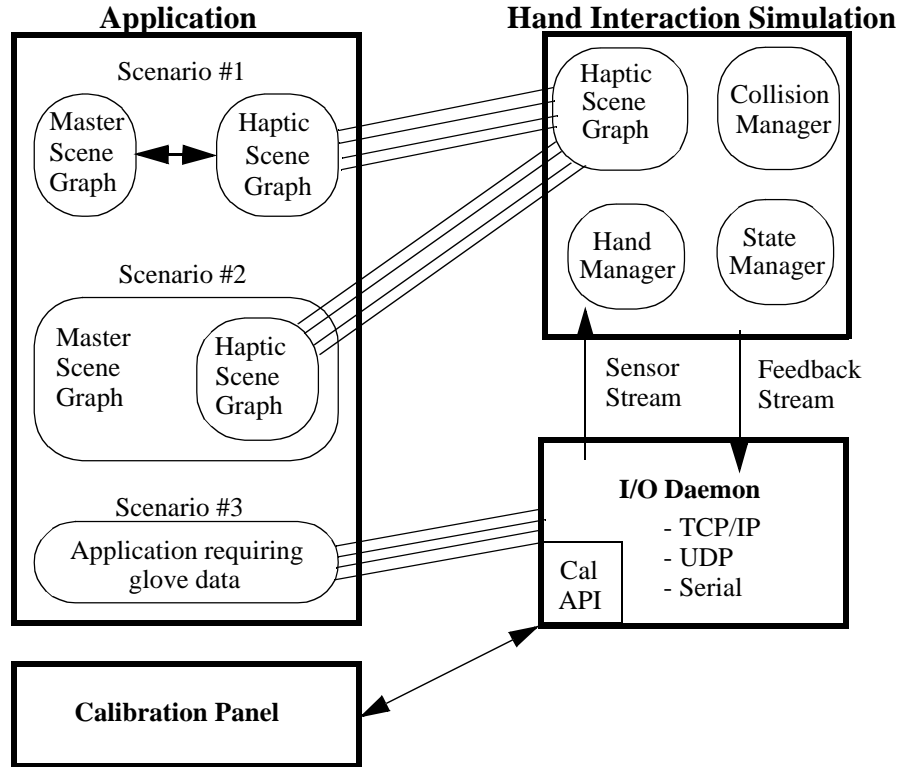


Figure 1: Overview of VirtualHand Toolkit

CyberGlove®, CyberTouch™ and CyberGrasp™ products. 5DT offers an API library for its Dataglove™. Both can display a graphical hand on the screen that mimics the glove wearer’s movements, but they do not provide manipulation capabilities; these still have to be programmed by the user. Additionally, only the VirtualHand library supports haptic feedback to the user.

This paper describes the VirtualHand Toolkit being developed by researchers at Virtual Technologies, Inc. More specifically, it covers the ongoing development of a *haptic scene graph*, which functions within a *hand interaction simulation*. This haptic scene graph interacts with a simulation’s master scene graph. A functional version of the toolkit will be demonstrated at the workshop.

## 2. VirtualHand Toolkit Overview

The VirtualHand Toolkit can be divided into the following:

- Hand Interaction Simulation (H.I.S.)
- I/O Daemon
- Calibration Panel

Additionally, to use the toolkit, the user must modify the application software such that a haptic scene graph may be constructed from the master scene graph. Figure 1 gives an overview of the VirtualHand toolkit. This paper will focus on the H.I.S.

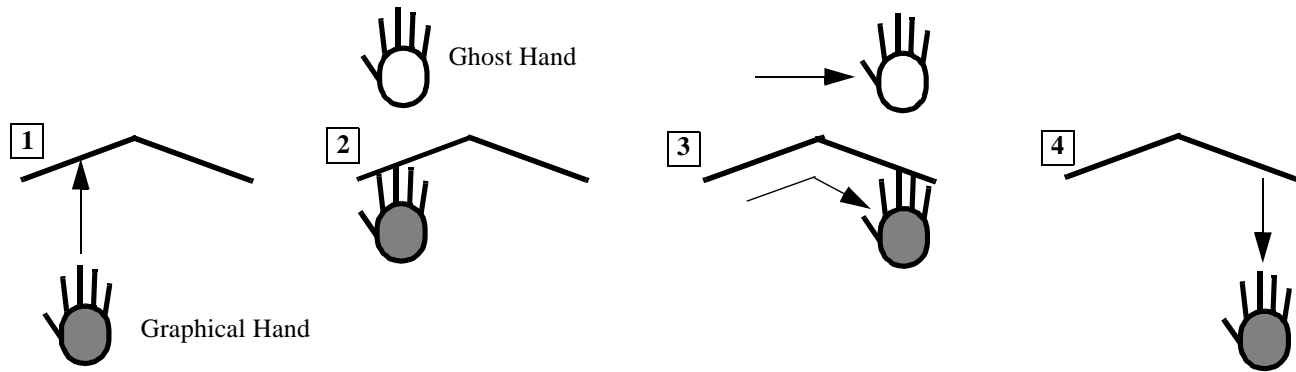
### 2.1 Hand Interaction Simulation (H.I.S.)

This is the main loop that controls the hand’s interaction with objects. It can be a separate process running on the host computer or, alternatively, it can run on the CyberGrasp controller. It consists of four major components:

- Haptic Scene Graph
- Collision Manager
- State Manager
- Hand Manager

The main part of the H.I.S. is the *haptic scene graph*, which contains a node for each of the objects in the “hand scene.” The hand scene is defined as all the objects that fall within a pre-determined radius of the graphical hand, i.e. within its *bounding sphere*. Each node in the scene graph has a bounding sphere associated with it, therefore the bounding sphere of the base node of the hand can be used to determine the hand scene. There are exceptions to this rule: If a linked object enters the hand scene, then all the other objects it is linked to, whether inside the bounding sphere or not, automatically are included in the hand scene. The same holds true for unlinked objects that are in contact with an object in the hand scene, as long as they remain in contact.

The *collision manager* is responsible for navigating the scene graph and determining which dynamic nodes are



**Figure 3:** Graphical hand following constraints as it tries to track the ghost hand.

colliding with one another. Static nodes are ignored by the collision manager. Currently, sphere/sphere, sphere/box and sphere/plane collisions are supported. Future planned “shape primitives” include: cylinders, cones, tori and ellipsoids. It should be noted that for haptic feedback, determining that two objects are colliding is not sufficient. The simulation must compute the depth of penetration of the fingers into the object in order to compute the feedback force, based on a stiffness model.

In this approach, the hand itself is part of the haptic scene graph and consists of multiple dynamic nodes. In the current implementation, each of the phalanges and the palm contain multiple dynamic sphere nodes. In the future, phalanges could be replaced with cylinder nodes. It should be noted that the nodes in the haptic scene graph do not necessarily correspond to the nodes in the graphics scene graph. For example, a complex shape could be rendered “as is” graphically, but from a haptic point of view, it would be represented by a bounding box.

The *state manager* is responsible for updating the state of each of the nodes in the haptic scene graph. At each time step, it traverses the tree, assigning new states (position, velocity, etc.) to the dynamic nodes.

Finally, the *hand manager* is responsible for dealing with the specifics of hand interaction. It is the process that sets the environment constraints, i.e. ensures that the graphical fingers don’t penetrate the virtual object, that a hand doesn’t go through a wall, etc. This is achieved by using two separate hand models: the *ghost hand* and the *graphical hand*. The ghost hand corresponds exactly to what is being measured by the instrumented glove and the spatial trackers. The graphical hand is constrained by the objects in the virtual environment. When no objects are encountered, the two hands overlap completely and have the same configuration and location. When constraints are encountered, the correspondence ceases and the two are linked via an elaborate network of virtual springs and

dashpots. This ensures that when the user’s real hand passes through a virtual wall and re-enters elsewhere, the graphical hand will be waiting at the appropriate location such that the two overlap again. This is illustrated in Figure 3. In step 1, the graphical hand is approaching a wall. In step 2, the user’s hand passes through the wall, but the graphical hand stops at the constraint. In step 3, the user’s hand moves sideways while the graphical hand tracks the wall. In step 4, the user’s hand backs away from the constraint and both hand models overlap.

## 2.2 Application Software

Typically, applications using the VirtualHand Toolkit fall within three scenarios:

### SCENARIO #1

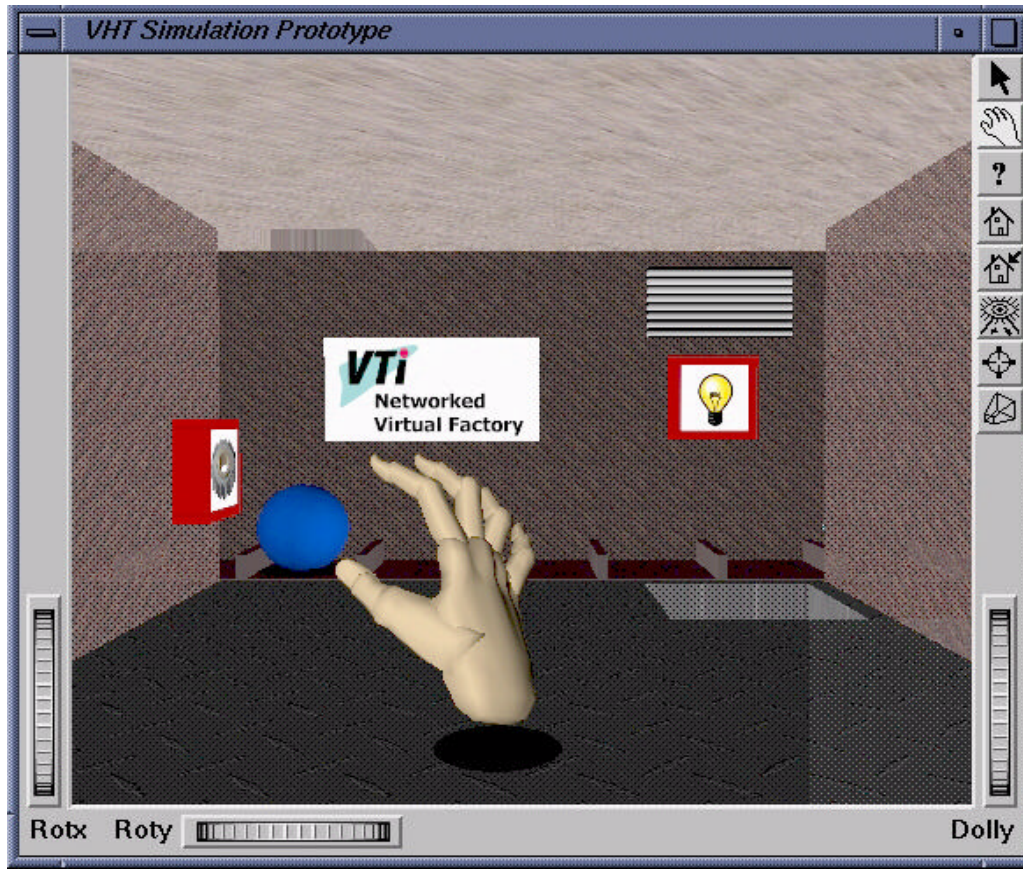
The toolkit is being used with a third party software package such as the world building toolkits provided by Sense8 Corp., Division Inc. and Deneb Robotics Inc. In this case, the applications already have a master scene graph and data must be extracted from it to construct a haptic scene graph residing on the host. This haptic scene graph will then be copied over to the H.I.S. and the two are updated asynchronously. This is done because the simulation running on the host will typically run at a much slower rate than the simulation running on the dedicated CyberGrasp controller. A high servo rate is desirable for high-fidelity feedback.

### SCENARIO #2

An application that is “haptic scene graph aware” is being written from the ground up using OpenGL, OpenInventor, etc. In this case, the haptic scene graph can be directly incorporated into the master scene graph.

### SCENARIO #3

In some instances, the users might want to bypass the haptic scene graph altogether and write their own hand interaction simulation. In these instances, they can



**Figure 2:** CyberFactory demonstration using VirtualHand Toolkit.

communicate directly with the I/O daemon to read data from the devices and control them directly.

### 3. Demonstration

A demonstration which illustrates the functionality of the current incarnation of the VirtualHand Toolkit will be presented at the workshop. The demonstration has been created using OpenInventor and consists of a simulation of a virtual CyberFactory (Figure 2). The user can interact with the CyberFactory using a CyberGrasp force-feedback device. The CyberFactory contains a conveyor belt, multiple types of spheres that can be manipulated (pulsing, breaking, exploding and compliant) and some push-buttons that start/stop the conveyor belt and toggle a light. Full dynamics are implemented for the spheres and the conveyor belt, and the user can feel the spheres in his/her hand while manipulating them and also feel the resistance of the buttons being pressed.

### References

- [1] P. Coiffet, M. Bouzit and G. Burdea, "The LRP Dextrous Hand Master," *VR Systems Fall 1993 Conference*, Sig Advanced Applications, New York City, October, 1993
- [2] D. H. Gomez, G. Burdea and N. Langrana, "Integration of the Rutgers Master II in a Virtual Reality Simulation," *1995 IEEE Virtual Reality Annual International Symposium*, pp. 198-202, San Francisco, CA, 1995
- [3] G. R. Luecke, Y. H. Chai, J.A. Winkler and J. C. Edwards, "An Exoskeleton Manipulator for Application of Electro-Magnetic Virtual forces," *Proceedings of the 1996 ASME Dynamics Systems and Control Division*, pp.489-494, Atlanta, GA, Nov. 17-22, 1996
- [4] L. Rosenberg, "A Force Feedback Programming Primer - For PC Gaming Peripherals Supporting I-Force 2.0 and Direct X 5.0," San Jose, CA, 1997
- [5] SensAble Technologies, "*GHOST* Software Developer's Toolkit - Programmer's Guide Version 1.1," Cambridge, MA 1996
- [6] T. V. Thompson II, D. D. Nelson, E. Cohen and J. Hollerbach, "Maneuverable Nurbs Models within a Haptic Virtual Environment," *Proceedings of the 1997 ASME Dynamics Systems and Control Division*, pp.37-44, Dallas, TX, Nov. 16-21, 1997