Chapter 6
A VRPN Server for Haptic Devices Using OpenHaptics 3.0

Maria Cuevas-Rodriguez, Matthieu Poyade, Arcadio Reyes-Lecuona, and Luis Molina-Tanco

Abstract This chapter presents an implementation based on the Virtual Reality Peripheral Network (VRPN 07.30) to handle connectivity between Virtual Reality (VR) applications and SensAble® Technology Phantom Haptic Devices using the OpenHaptics 3.0 Haptic Library Application Programmable Interface (HLAPI). VRPN offers a client–server-based architecture to support network-transparent connectivity between VR applications and a set of physical interaction devices. In this context, VRPN provides a set of classes to handle various physical device types. The proposed implementation consists of (a) new VRPN classes that support connectivity between a haptic device server and VR applications, allow to specify arbitrary 3D object information to haptically render geometries, and report applied force, angle at contact point, Surface Contact Point (SCP), and Depth of Penetration (DOP) and (b) an upgrade of the Phantom dedicated VRPN class to handle haptic rendering using the OpenHaptics HLAPI to manage device state and force computation.

6.1 Introduction

During the last decade, Virtual Reality (VR) applications have included a wide variety of 3D interaction techniques and devices. Among them, the iteration between Virtual Environments (VE) and the user used to be through two senses: the sight and the hearing. Nevertheless, the touch sense is becoming more important and studied

M. Cuevas-Rodriguez (✉)
Departamento de Tecnología Electrónica, Universidad de Málaga, Malaga, Spain
e-mail: mariacuevasrodriguez@gmail.com

M. Poyade • A. Reyes-Lecuona • L. Molina-Tanco
Departamento de Tecnología Electrónica, Universidad de Málaga, Malaga, Spain
e-mail: matthieu.poyade@uma.es; areyes@uma.es; lmtanco@uma.es

V.M.R. Penichet et al. (eds.), *New Trends in Interaction, Virtual Reality and Modeling*, Human-Computer Interaction Series, DOI 10.1007/978-1-4471-5445-7_6,

in VR because it gives the user the possibility of interact and modify the environment. In this context, haptic devices play a really useful role in VR; they are very important as they have allowed a wider multimodality, beyond the traditional visual and auditory stimuli. In addition, haptic devices are bidirectional and allow a more natural interaction closing the perceptual-motor loop between the user and the VE [4]. For these reasons, many applications can benefit from using haptic devices to enhance interaction and providing a realistic force feedback to the user. Specifically, motor skill training applications need haptic interaction in order to be valid.

Haptic rendering is, however, very demanding in terms of computing power. Hence, for many applications a dedicated computer is required in order to provide realistic force feedback. This leads to the problem of connecting the computer or computers where the graphical rendering is being performed and the computer managing the haptic rendering [5]. There is an additional issue with manufacturers of haptic devices providing different programming interfaces. A standard way of efficiently accessing any haptic device through a local network connection would make it easier to use these devices for a wider range of applications.

These issues had to be tackled in the ManuVAR project. The ManuVAR project funded under the European Union's Seventh Framework Programme (NMP-CP-IP-211548) aimed to use virtual and augmented reality to develop an innovative technology platform and a framework to support high value manual work. One of the goals of this project was to provide a flexible platform where different technological elements and methodologies could be connected in a modular way. Therefore, some mechanism was needed for connecting haptic devices with a device-independent and network-transparent interface.

Exactly for these purposes, the Virtual Reality Peripheral Network (VRPN) was developed 10 years ago [7]. VRPN is an open source package, which provides a network architecture for connecting different interaction devices to a VR application. It has become a *de facto* standard for motion capture systems and tracker devices. However it is less popular for other kinds of devices. In its distribution, VRPN implements simple force feedback device classes for SensAble® products, with very limited functionality. For that reason, it is hoped that this work will help in contributing to VRPN becoming also a standard middleware for the integration of haptic devices in VR environments.

This chapter reports a new implementation of a VRPN server which provides the original tracker and button interfaces along with new and varied force device interfaces for SensAble® devices, using the OpenHaptics 3.0 Application Programmable Interface (API) [1], updating and extending the functionalities available in the current VRPN distribution (VRPN 07.30). This new VRPN server can manage the three different models of haptic devices manufactured by SensAble®: the Phantom® Onmi, Phantom® Desktop, and Phantom® Premium 3.0.

The remaining of the chapter is organized as follows: Sect. 6.2 gives a short overview of VRPN and SensAble® Software Developer's Toolkit (SDKs), Sect. 6.3 presents the main features of the proposed implementation, Sect. 6.4 briefly presents an application where it is being tested, and, finally, Sect. 6.5 summarizes giving some conclusions of this work.

6.2 Technical Background

Haptic technology in VR offers a 3D multimodal real-time sensory motor interaction paradigm that feedbacks force sensory information, leading to improve task performance and enhance the way users interact within VEs [4].

SensAble® Technologies is a developer of haptic interfaces that manufacture the Phantom® haptic devices since 1993 [6]. Different models of haptic devices are available as previously mentioned. Each of these devices is shaped as *stylus interactuators* and able to deliver force feedback and high degrees of maneuverability within VEs providing the same feeling experimented by touching a surface with the point of a pencil.

A C++ SDK to support haptic rendering for integration of the haptic interaction paradigm in VEs is provided by SensAble® Technologies. General Haptic Open Software Toolkit (GHOST) is a legacy API for Phantom® devices and currently superseded by the OpenHaptics Toolkit, an Open GL-based library. The OpenHaptics 3.0 toolkit presents a three-layer architecture: the Haptic Device API (HDAPI), the High-Level API (HLAPI), and the micro QuickHaptics API. HLAPI is a high-level API able to haptically render geometries stored into OpenGL's specific buffers. It offers several commands to set custom force effects (stiffness, damping, static and dynamic friction, viscosity, etc.) and handle the thread management required to support haptic rendering. HLAPI manages three different threads: the client thread (∼30 Hz) supports graphical rendering, the collision thread (∼100 Hz) supports collision detection, and the servo thread (∼1,000 Hz) handles the position and orientation of the haptic device and calculates forces.

The combination of a computer graphics engine and a force feedback haptic rendering engine results in a heavy computational load. An option to maintain performance is to execute the graphics and haptics rendering loops in separate computers [5]. VRPN offers a network-transparent architecture to handle connectivity between VR applications and physical interaction devices. VRPN provides a set of classes defining several *canonical interface types*. Each *canonical class* derives into a remote client interface class and a device server interface class. Both specify methods to be called from a remote client and the device server. Devices are mapped into one or several canonical interface types depending on the information reported. Generic interface types consist of Tracker, Button, Analog, Dial, and Force devices. In overall, a purely physical device dedicated class that inherits from one or several interface server types supports the device rendering.

At the time of writing, the currently available version of the standard distribution of VRPN is version 07.30. This version supports connectivity with Phantom® devices through the canonical classes vrpn_Tracker, vrpn_Button, and vrpn_ForceDevice [7]. These provide, respectively, the functionality of a Tracker that reports device position, orientation, velocity, and acceleration; Button that reports device buttons state; and Force Device that handles haptic parameters specifications and reports applied force and Surface Contact Point (SCP).

An additional class, vrpn_Phantom, is responsible of the communication between the haptic devices and the three canonical classes previously mentioned. It is the responsible of the haptic rendering for these specific devices which inherits from the Tracker server, Button server, and Force Device server dedicated classes.

6.3 Implementation

The main goal of this work is to make easier the management of haptic devices through the development of two new classes to be included in the last version of VRPN server. These classes provide innovative and useful functionalities allowing a haptically render of arbitrary geometries.

A new client has been also developed in order to test the described client–server architecture which allows the interaction with the VE. It is also in charge of creating the graphic scene and sending its characteristics to the server to reproduce the haptic scene.

6.3.1 New Classes Implemented

The proposed implementation consists of a new vrpn_ForceDevice class, named *vrpn_ForceDevice_Uma* (Fig. 6.1), and a new vrpn_Phantom class, *vrpn_Phantom_Uma* (Fig. 6.2).

The new vrpn_ForceDevice_Uma class, as the original one, supports connectivity between a haptic device server and VR remote client applications, reporting applied force and SCP. However, this new class allows specifying arbitrary 3D objects information to haptically render geometries and reports not only applied force and SCP but also angle at contact point, Depth of Penetration (DOP), and the identifier of the object that has been touched.

The new vrpn_Phantom_Uma class implements OpenHaptics HLAPI functionalities to manage device states and haptically render force effect models to provide force feedback. These two new classes have been implemented following the philosophy and structure tree of the original classes of VRPN. The vrpn_Phantom_Uma class inherits from the tracker and button canonical classes provided by VRPN and the new force device server class (vrpn_ForceDevice_Uma) as shown in Fig. 6.2.

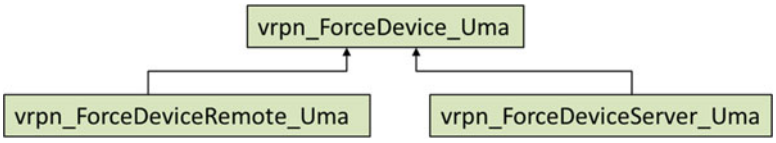

Fig. 6.1 VRPN class hierarchy for Force Device classes

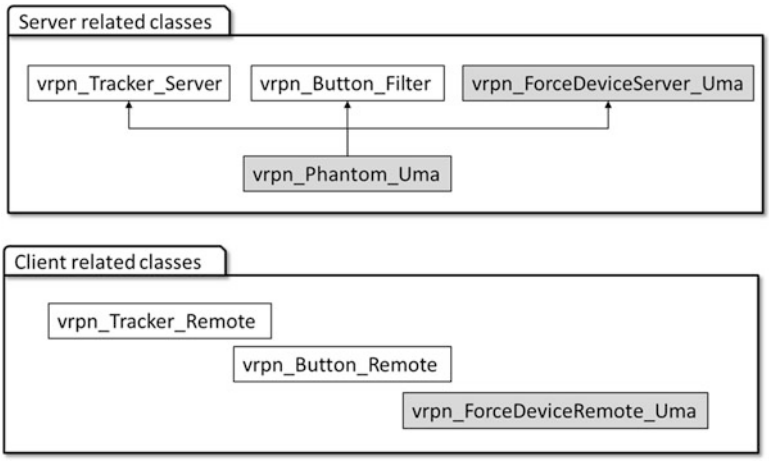

Fig. 6.2 Modifications to the VRPN

Table 6.1 New methods added to vrpn_ForceDeviceRemote_Uma interface

New methods	Description
setObjectNumber	Sets the number of objects to render
setVertex	Sets the vertex of an object
setTransformMatrix	Sets the transformation matrix for each object, which provides orientation, position, and scale of the object
setEffect	Sets the environmental effects to render. HLAPI provides four effects: constant force, spring, viscosity, and friction. For each one, gain, magnitude, frequency, duration, position, and the direction can be provided
startEffect	Indicates that the effect should begin
stopEffect	Indicates that the effect should finish
setHapticProperty	Sets the haptic properties of an object: stiffness, damping, static and dynamic friction, pop through, and mass
setEnvParameters	Sets the force effects which are used to generate ambient sensations: gravity and inertia
setTouchableFace	Sets the face of the object that will be haptically rendering: front, back, or both. This feature is the same for all objects
setWorkspace	Sets the work space, the space where the device is going to interact with the VE

The function of each class is summarized in the following paragraphs.

vrpn_ForceDevice_Uma. This class manages message type declaration and message encoding and decoding. This class increases the functionalities of the original vrpn_ForceDevice class, developing new message types and encoding and decoding functions related to implemented methods described in Tables 6.1 and 6.2 to respectively support connectivity to remote client and device server.

Table 6.2 Send methods within the new vrpn_Phantom_Uma interface

Send methods	Description
sendForce	Sends the applied force
sendDOP	Sends the Depth of Penetration
sendSCP	Sends the Surface Contact Point
sendIsTouching	Indicates if it is touching an object
sendTouchedObject	Sends the identified of the object that this is touching
sendAngle	Sends the angle at contact point

vrpn_ForceDeviceRemote_Uma. This class provides a set of new methods (Table 6.1) for the client application. These methods enable sending haptic parameters, defined in the client application, to be integrated with haptic rendering on the server side. Furthermore, vrpn_ForceDeviceRemote_Uma class implements a set of callback functions to receive haptic information messages from the server.

vrpn_ForceDeviceServer_Uma. It handles a set of callbacks to receive haptic parameter definition messages from the client decoding and forwarding the messages to vrpn_Phantom_Uma class.

vrpn_Phantom_Uma. The new vrpn_Phantom_Uma class has direct communication with the device using the OpenHaptics HLAPI. This class organizes within a set of structures, the received haptic parameters inherited from the vrpn_ForceDeviceServer_Uma class. vrpn_Phantom_Uma performs the haptic rendering providing force feedback to user through the phantom device. Moreover, vrpn_Phantom_Uma class provides force feedback data to the remote client application when any change happens (Fig. 6.3). To do so, a set of methods has been implemented as detailed in Table 6.2.

The only device-specific class is vrpn_Phantom_Uma. To integrate a different force feedback haptic device, a new class has to be implemented with the SDK of the specific device. The rest of the classes are device independent and can be reused, in the spirit of VRPN.

6.3.2 *Client–Server Communication*

Decoupling graphical and haptic rendering enables parting the asynchronous execution of both systems. On one hand, the client computes the graphical rendering and shows the graphic scene to the user, providing the auditory and visual stimuli. On the other hand, the server carries out the haptically render as it has mentioned in the previous section.

At the beginning of the communication, the client application sets the graphic scene and creates a vrpn connection to the VRPN server, then sets and sends haptic scene properties to force device server (see sequence diagram in Fig. 6.4), firstly by recovering geometry-based information about the objects deployed in the scene from graphics dedicated buffers using the setObjectNumber, setVertex, and

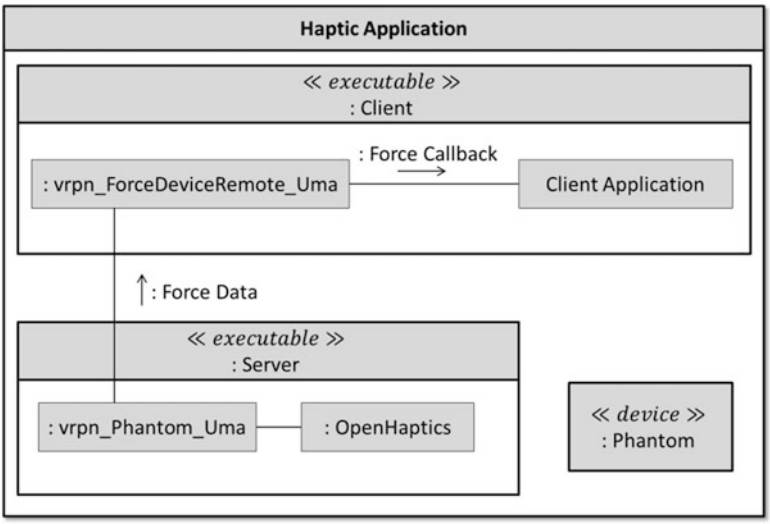

Fig. 6.3 The force feedback changes are sent from the Phantom Device to the Client Application through callback invocation

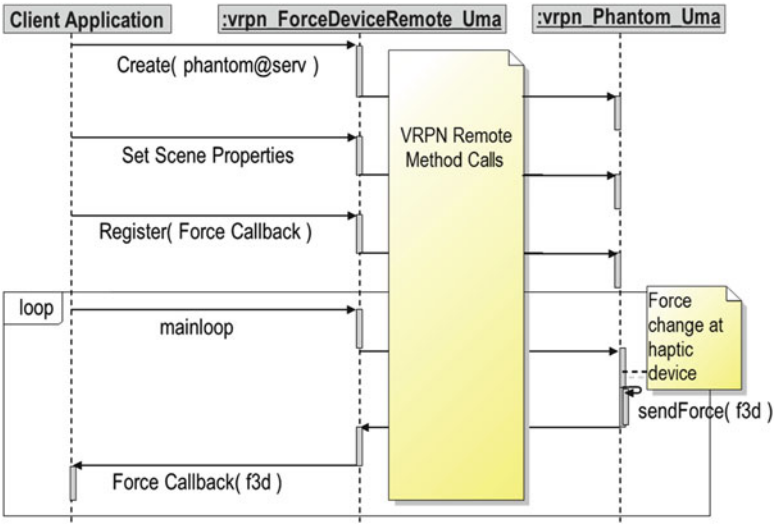

Fig. 6.4 Client–server communication between Client Application and Haptic Device

setTransformMatrix methods detailed in Table 6.1 and, secondly, by defining haptic effects and force model parameters using the methods setEffect, setHapticProperties, setEnvParameters, setTouchableFace, and setWorkspace. The client application starts the force effects and the haptic rendering loop and declares a set of callback functions to enable receiving phantom-based information from the server.

During user interaction within VEs, using the Phantom® Device, the client application receives Phantom-based information attached to tracker, button, and force device interfaces servers. The force-based information is named by the Force Callback function in Fig. 6.4 to simplify the scheme; however, all the supported methods are defined in Table 6.2.

6.4 Application

This implementation has been successfully tested in an industrial case within the framework of the ManuVAR project [2]. This consisted in the development of a training simulator for performing metallographic replicas (Fig. 6.5). The metallographic replica is a non-destructive inspection technique which requires following some specific steps, including a careful polishing of the surface where the replica is going to be taken. The simulator includes a wide variety of scenarios where different objects are rendered. Thanks to the server properties, allowing haptically render of any geometry, the haptic dimension was incorporated into the scene in a quick and easy way.

Within ManuVAR, a distributed platform has been developed to integrate different applications where the connection of any interaction device is location transparent; furthermore, the computational cost of running these applications is allocated in various machines.

The proposed implementation using VRPN has also been successfully used in an experiment where a high number of people tested the simulator to determine which feedback is most appropriate in that training tool. Everybody agreed with the natural and real-world feeling due to the haptic sensation.

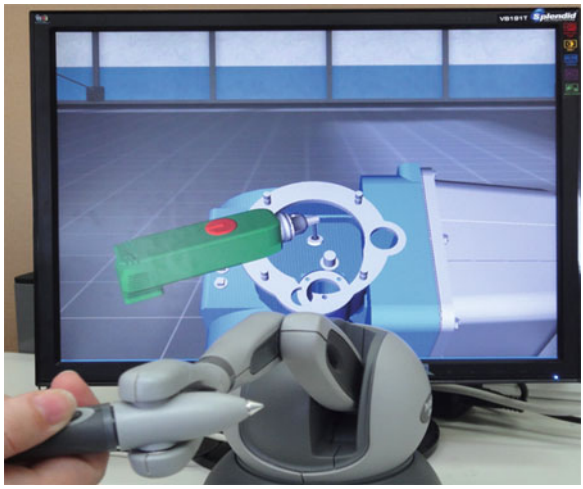

Fig. 6.5 Application case using the Phantom-based VRPN server

An experimental study carried out at the University of Nottingham investigated the design of augmented feedback for improving virtual reality haptic training in the performance of a complex inspection task in a real manufacturing case study [3].

6.5 Conclusion

In this chapter a new VRPN server implementation of force feedback for SensAble® haptic devices is introduced and described. These new contributions include some features, not developed in the current distributions of VRPN. Furthermore, the server provides an easy and efficient way to introduce the haptic dimension in a virtual scene due to the new developed interfaces that offer the possibility of introducing any kind of geometry. Haptic Phantom® device contacts with the virtual geometry and much information about its interaction is reported by the VRPN server.

The work has been developed within the ManuVAR project, where a modular, flexible, and location transparent architecture required a distributed connection of VR devices. All the features described in this chapter have been introduced in that project providing the required results.

These implementations could be the beginning of a new trend of standardizing the haptic device integration in virtual environments using VRPN thanks to the simplicity and modularity of its development.

Acknowledgment The abovementioned research has received funding from the European Commission's Seventh Framework Programme FP7/2007–2013 under grant agreement 211548 "ManuVAR."

References

1. Itkowitz, B., Handley, J., & Zhu, W. (2005). The openHaptics toolkit: A library for adding 3D touch navigation and haptic to graphics applications. In *Proceedings of the Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (pp. 590–591).
2. Krassi, B., D'Cruz, M., & Vink, P. (2010). ManuVAR: A framework for improving manual work through virtual an augmented reality. In *Proceedings of the AHFE 3rd International Conference on Applied Human Factors and Ergonomics, AHFE*, Miami, Florida, USA, 17–20 July, 2010.
3. Langley, A., Sharlples, S., D'Cruz, M., Patel, H., Poyade, M., Reyes-Lecuona, A., & Molina-Tanco, L. Impact of multimodal feedback on VR training for manufacturing manual work. In *Proceedings of the Tenth International Conference on Manufacturing Research* (Vol. 1, pp. 219–224). Birmingham, UK, September 2012.
4. Mac Lean, K., & Hayward, V. (2008). Do it yourself haptics: Part ii [tutorial]. *Robotics & Automation Magazine, IEEE, 15*(1), 104–119.

5. Mark, W., Randolph, S., Finch, M., Vanverth, J., Taylor, I., & Russell, M. (1996). Adding force feedback to graph ics systems: Issues and solutions. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (pp. 447–452). ACM, SIGGRAPH 96.
6. Massie, T., & Salisbury, J. (1994). The phantom haptic interface: A device for probing virtual objects. In *Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (Vol. 55, pp. 295–300). Chicago, IL.
7. Taylor, I., Russell, M., Hudson, T., Seeger, A., Weber, H., Juliano, J., & Helser, A. (2001) VRPN: a device- independent, network-transparent VR peripheral system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (pp. 55–61). ACM. VRST 01, November 15–17, 2001, Banff, Alberta, Canada.