HAREVIR: A Methodology for Developing Virtual Reality Projects with Haptic Rendering

Esther Ortega-Mejía, Marva-Angélica Mora-Lumbreras and Alberto Portilla-Flores

Autonomous University of Tlaxcala
Calzada Apizaquito S/N, Apizaco, Tlaxcala, Mexico. C.P. 90300
{esther.ortega.m, marva.mora, alberto.portilla}@gmail.com

Abstract. This paper presents a methodological proposal, which is focused on the task of rendering and manipulation of virtual objects via kinesthetic. Our project involves two key areas: Virtual Reality and Haptic Rendering. Virtual Reality is related to the creation of virtual environments, considering physical and virtual aspects, levels of immersion and interaction with the user. Haptic rendering is used to provide kinesthetic stimuli in order to improve the man-machine interaction. We identify three main processes as a part of the methodology: i) selection of algorithms and methods for haptic rendering according to the project developed, ii) selection of languages and specialized software tools for virtual applications and iii) selection of haptic devices, all these considerations are key for developing a project using haptic rendering.

Keywords: Virtual Reality, Haptic Rendering, navigation.

1 Introduction

Currently, rendering of haptic cues is a challenge, mainly due to the bidirectional communication process between the user and the virtual project that must be provided. Haptic rendering allows the user to perceive through the skin (temperature and texture) and rendering through muscles, tendon and joints (position, velocity, acceleration, force and impedance). Due to such kind of complexity, these projects are limited to some cues. Haptic is still an incipient research area, then there is no enough information in literature related to the development of haptic rendering projects, so we consider imminent the creation of this proposal.

This paper presents a methodological proposal for developing robustness in projects using haptic rendering. The proposal has emphasis on different algorithms related to the haptic rendering. It was necessary to consider the state of the art of haptic rendering, to select the most relevant works, which have the necessary maturity to guide in the ad-hoc project development.

The rest of the paper is organized into the following sections: section 2 provides an over-view of haptic and virtual reality, section 3 describes the proposed methodology which presents a classification of haptic rendering algorithms, indicate the software used in the topic, also presents the main haptic devices, then in section 4 an analysis

about possible selection of algorithms is presented, finally the last section concludes this work.

2 Virtual Reality and Haptics

Virtual reality, in terms of functionality, is a simulation in which computer graphics are used to create a realistic-looking world [9]. A virtual environment (VE), specifically a computer-based simulated environment intends for its users to inhabit it and interact with it via avatars. These avatars are usually depicted as textual, two-dimensional or three-dimensional graphical representations, although other forms are also possible. Some virtual environments allow multiple users to participate simultaneously. The perception of these environments can be done via visible, audible, or tactile means.

Virtual reality considers four elements: virtual world, immersion, sensory feedback and interactivity: A virtual world is an artificial computer-generated environment in which users are able to interact with each other by means of characters and manipulating objects. Immersion is usually defined as the full sensory replacement by artificial means instead of being generated from the real world. The interaction between the system and the user should be natural. A good interaction helps to get a better immersion sense. Unlike traditional media such as video games, animation and other applications in virtual environment users can interact with systems, influence events in the world and receive a dynamic feedback.

To increase immersion in virtual environments different areas have been involved, such as haptic rendering, which focuses on making someone sense of touch or feeling the force using a haptic device [1]. The key features of a haptic device are: force range, degrees of freedom, workspace, etc. The devices allow users manipulate a virtual environment and their 3D objects.

3 HAREVIR Proposed Methodology

In this section a methodology for developing projects using virtual reality and haptic rendering is proposed: HAREVIR, (see Figure 1. Methodological proposal). Our methodology includes three important tasks: the selection of haptic rendering algorithms and methods, selection of specialized languages and selection of haptic devices. Next sections detail those tasks.

3.1 Algorithms and Methods for Haptic Rendering

While a haptic device provides interaction between the user and a given virtual environment, captures user positions, and others qualities [5], the algorithms related with haptic rendering generate kinesthetic stimuli in the users. They have two main tasks: compute the position and orientation of the virtual object manipulated by the user (avatar) and calculate the forces and torques that should be returned.

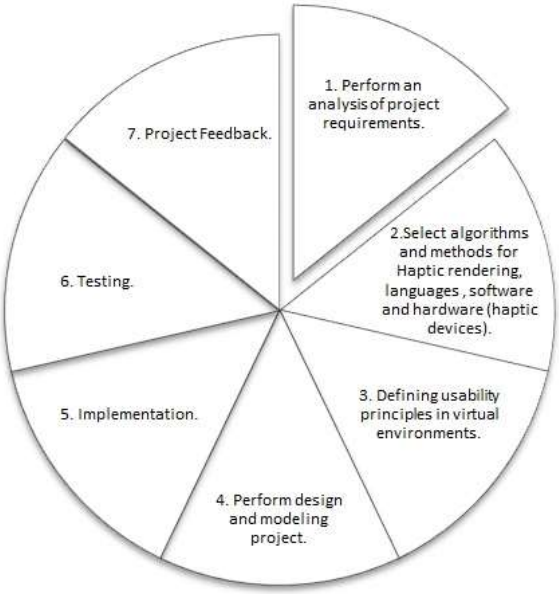

Fig. 1. Methodological proposal.

Most haptic rendering methods divide the process of calculating the collision force and torque in three stages: collision detection, collision response and control algorithms. In the first stage, the position of the haptic device and virtual environment information is considered to check any collisions. In the second stage, if there were a collision, information would be taken, the ideal position of avatar and the ideal strength of interaction between the avatar and the environment would be calculated. In the third stage, a force returns to the user, which approximates the ideal force according to the capacity of the haptic device (see Figure 2).

After having a requirement analysis of the project, it is possible to select a specific haptic rendering algorithm. It is important to considerate that there are simple and complex virtual environments. The algorithms have different approaches; the Figure 3 shows several haptic rendering algorithms considered by HAREVIR.

Penalty methods

Penalty methods calculate the collision force based on the amount of interpenetration of the virtual objects in a collision, either by using a model of elastic contact, viscoelastic or a different one.

The elastic model calculates the forces and torques collision, in the simplest case.

This method calculates the magnitude of the penetration p, the normal collision direction n and the contact point cp, where cp is the center of inertia of the avatar and K represents the virtual stiffness contact (see equation 1).

$$F = Kpn, M = (cp - cg)F \qquad (1)$$

Penalty methods are divided into two main types: polygonal methods and voxel-based methods.

Polygonal methods

These methods use polygonal representations, usually triangle mesh to define the surface of the virtual objects, and the collision response calculated from the

information associated with the detected collision triangles. Once detected the triangles, the position information of its vertices and the normal vector of each triangle it is calculated the normal direction, penetration and a point of contact to obtain a force and torque representative of the collision.

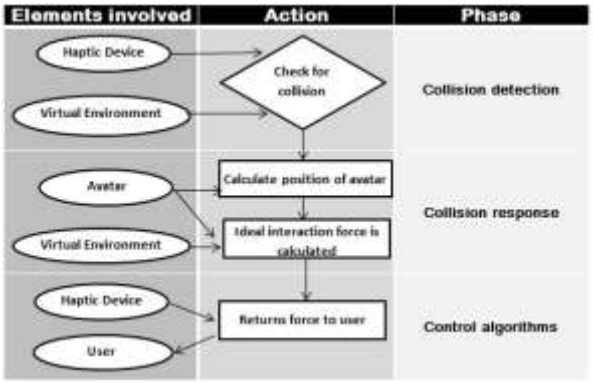

Fig. 2. Stages of forces and torques collision calculation.

The most basic representation for a volume is the classic voxel array in which each discrete spatial location has a one-bit label indicating the presence or absence of material, in volume haptic rendering technique. Sometimes it uses additional physical properties like stiffness, color and density during the voxel representation [8].

Voxel-based methods

Virtual environment is divided with mesh container boxes, in several researches, called voxels, to simplify the calculation of collisions and penetrations between two virtual objects.

Polygonal representations are replaced by two different structures, named Voxmap-PointShell. The first, Voxmap, is a partitioning spatial structure voxels based replacing the geometrical model of the virtual scene. The second, PointShell, is a cloud of points defining the surface of the avatar associated with each data point. Both facilitate the calculation of the normal and the resulting penetration in avatar collisions with objects of the scene.

Constraints-based methods

Constraint-based methods do not use interpenetration forces in rigid objects to calculate the collision response. These methods use the virtual coupling and restrict decoupled virtual object on the surface of the obstacles, guiding their dynamics by restrictions. An example is the God-object method, this algorithm locates a virtual object minimizing the distance between it and the position of the haptic device at every moment under restrictions [8].

Impulse-based methods

Impulse-based methods are less quoted in the references. They restitute forces in impulse form in collision events. Continuously collisions are treated like series of small collisions, in every of them impulses are applied to avoid penetration between objects. They produce visually acceptable results, but not very convenient in haptics.

Degrees of freedom

Another classification of haptic rendering methods is based on the degrees of freedom on which they can act. When they only act on the translations of the virtual object they are called methods of three degrees of freedom (3DOF), but if they also act on the turns of the object they are considered methods of six degrees of freedom (6DOF). For example the haptic rendering algorithm of Three Degrees of Freedom (3DOF) [1] takes as input the position and orientation of the haptic device and transforms them to the reference system of 3D dataset.

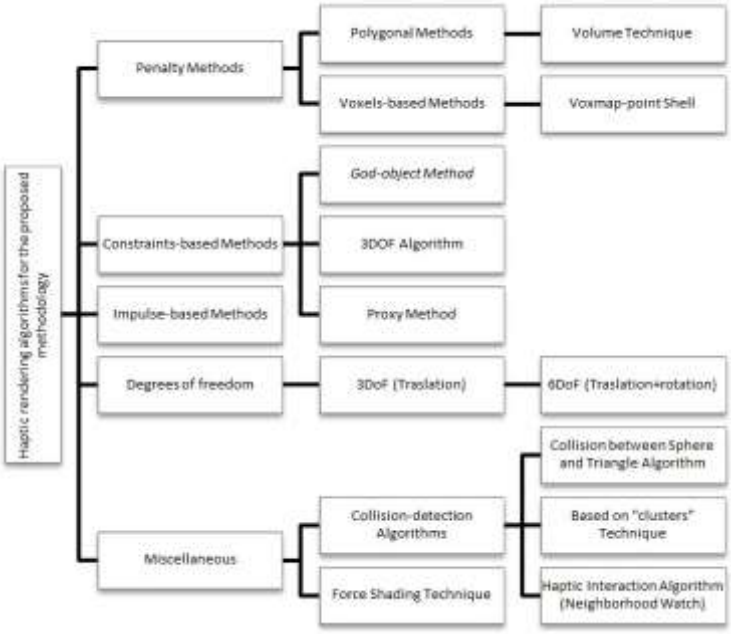

Fig. 3. Classification of Haptic Rendering Algorithms.

3.2 Languages and Tools

Several tools, libraries and languages support the development of haptic applications. They interact with haptic devices and even reveal implementation details of certain device function features such as HDAL, then they are described briefly some of them.

Object and model generation

Generation of models and objects is performed by a wide range of specialized software. Examples of these programs are: Blender is an open source suite to create 3D animations, 3DStudioMax is useful for creating virtual objects and scenes, Panda3D is a game engine and OpenGL is a multi-platform library that allows developing 3D graphic applications. Some features of the software are: professional support in modeling and animation, varied file formats allowed, simulations and high quality sculptures creation, etc.

Game engines

Today there are a variety of game engines for creating applications with a wide diversity of languages some of which include their own Software Development Kit (SDK). Two of them are: Unity, a game development engine to create interactive 3D content and Panda3D that is a framework for 3D rendering and game development [7].

Languages

General purpose languages such as C++, C#, Java and Visual Basic and interpreted programming languages such as Python (that through scripts makes Blender extendible) are commonly used to develop haptic applications.

Libraries and plugins

Some libraries detect whether objects in a virtual environment are colliding with each other or with devices, there are commercial and other free distribution, such as ICollide, Vcollide, Rapid, etc. [4]. They are designed to work with specific models and algorithms. Others as H5H, which is a plugin to provide haptic web effects, HAPI (Haptic API) provide haptic effects, H3DAPI uses OpenGL to handle graphics and haptic scenes and CHAI3D that is an open source set of C++ libraries for haptic rendering [6].

HDAL

The Haptic Device Abstraction Layer (HDAL) provides a uniform interface to all supported type devices. The API provides means for selecting a device, initializing it, reading its state, etc. This includes implementing a callback function that HDAL will cause to be executed at a 1 KHz rate, to achieve the required haptic fidelity [2].

For the development of this proposal it is used: C++ programming language, OpenGL to model some objects, Panda3D to generate interaction between the 3D model and the user, and to control the haptic device it was considered HDAL interface.

3.3 Haptic Devices

A haptic device stimulates tactically the operator, this information is received through the skin by pressure of a certain area, through twists and movements supported in a virtual environment. In [4] they are classified according to their portability in: desktop, fixed and portable.

Desktop

Desktop devices provide information about embossing, texture and even temperature of the virtual environment. They are used like a joystick and are subdivided into spherical, cartesian, parallel, serial and made out of cables. Some examples are: Novint Falcon, Phantom Omni, Excalibur (Cartesian), MantisFlyer (made out of cables), etc.

Fixed

Fixed devices reproduce the movements of the operator in the remote environment through an anthropomorphic robot with similar operator arms. In this classification fixed exoskeletons and robot arms are found, for example, the Master Arm, Haptic Master, etc.

Portable

In Portable devices the user can support the whole weight of the interface. They can be exoskeletons hold to operator and others like gloves, for example, CyberGrasp and Rutgers Master II.

Novint Falcon Haptic Device

The Novint Falcon desk haptic device provides information about embossing, texture and feedback force. It is a commercial haptic device to develop video games. It has three degrees of freedom, a removable end-effector, a USB interface and bears three Newtons of force. It is a parallel robot with three arms joined by an end effector. It is considered as a manipulator robot. Its dynamic model is shown in Equation 2.

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \qquad (2)$$

Where $(q,\dot{q}) \in R^{2n}$ are vectors representing the position and velocity, n is the number of degrees of freedom, $H(q) \in R^{nxn}$ is a matrix meaning robot inertial forces with $H(q) = H(q)^T$ defined as positive, $C(q,\dot{q}) \in R^{nxn}$ is a matrix that symbolizes the Coriolis force, $g(q) \in R^n$ is the vector of gravitational torque, $\tau \in R^n$ is a vector which forms the entrance wall of a joint [3].

Today we can see affordable devices, compatible with different languages, easy to use, give the user a more immersive experience and a realistic sense of touch, whose applications are in several areas: education, telemedicine, robotics, surgical training, etc.

3.4. Analysis

Phases and activities of the methodology

Some activities of the first two stages analyzed in this paper, are presented here, which could be applied in concordance with the needs of any project (see Figure 4. Activities to do in each phase).

User requirements	Select algorithms and methods for Haptic rendering	Languages, libraries and software.	Hardware (haptic devices).
•Defining geometry objects. •Planning moves allowed. •Analysing the haptic device suitability. •Proposing forces rendering procedure. •Devising proxy relation to the position of the haptic device. •Preparing sensory channels to be worked on. •Selection of modeling software, libraries and language.	•Setting type of objects to be worked on. •Calculating position and orientation of objects. •Checking the necessary level of haptic feedback (calculation of forces and torques). •Detecting collisions. •Calculating collision response (calculate the surface normal and the point of contact, time response).	•Designing the virtual environment. •Designing graphic rendering (convert geometric pattern on an image). •Designing proxy relation to the position of the haptic device. •Checking haptic device compatibility. •Creating or setting the initialization routines of the haptic device with libraries. •Programming chosen algorithms.	•Reviewing all the technical specifications (degrees of freedom, maximum weight to bear, etc). •Make reading device status (position, velocity, buttons, etc). •Considering the borne work area. •Analyzing the kinematics of the device. •Creating or setting the device dynamic model. •Transforming the device coordinates to the end user .

Fig. 4. Activities to do in each phase.

Selection of tools, languages, devices and objects

In this section a table of possible choices of algorithms, tools and languages is presented, as well as some aspects to consider for the use of algorithms according to the type of interaction and Virtual Reality project you want to develop.

The first is an effective rendering technique with mesh models of objects, but it fails in the use of point cloud based models. Second, it renders smooth objects and it is used to create the illusion of a smoothly curved shape. The third allows working with thin objects, but it is invalid to represent applications where there are twists. The fourth is capable of rendering Monge surfaces represented by a non-uniform point cloud data. And the fifth helps to prevent that collision method detect a high number of different contacts at the time of decomposing convex primitive geometric models (see Table 1. Selection of tools for different types of projects).

The use of algorithms, languages, tools, devices and objects depend on the parameters set for any project and after a whole review the most appropriate tools and languages can be chosen.

Table 1. Selection of tools for different type of projects.

NO	ALGORITHM	LANGUAGE AND TOOLS	DEVICE	OBJECTS
1	God-Object	C++, Java, C#, CHAI3D	PHANTOM Omni / Falcon	Point, plane
2	Force Shading	Python/ Panda3D	PHANTOM Omni	Polyhedral meshes
3	Virtual Proxy	C++, Java, CHAI3D	PHANTOM Omni / Falcon	Spherical object
4	Haptic rendering on different scales	OpenGL, C++, HAPI library	Falcon	Point cloud based 3D models
5	Based on clusters Technique	C++, Java, C#	PHANTOM Omni SARCOS Dexterous	Convex objects

4. Conclusions

In this article it was presented the use of a methodology that will provide support and guidance in developing focused haptic rendering applications, as well as, a classification of a set of algorithms. Three main processes have been identified: Selection of algorithms and methods for haptic rendering according to the project developed, selection of languages and specialized software tools for virtual applications and selection of haptic devices, all these considerations are key to the development of a dedicated project in haptic rendering. The detection of the key elements was achieved after a thorough research on haptic rendering and virtual reality topics, as well as, the experience developed during the doctoral project Haptic Rendering for Navigation and Manipulation of Virtual Objects.

References

1. Corenthy L., Martín J.S.: Volume Haptic Rendering with Dynamically Extracted Isosurface. IEEE Haptics Symposium, Vancouver (2012)
2. Haptic Device Abstraction Layer (HDAL) Programmer's Guide. Novint Technologies Incorporated, Albuquerque (2008)
3. Gamboa J., Sepulveda G.: Análisis Biomecánico y Anatómico para el Modelado Dinámico y Simulador Virtual del Tejido Blando con Dispositivos Hápticos, Veracruz (2009)
4. Ambrosio M., Ramírez D.: Desarrollo de interfaces hápticas en medios virtuales, Universidad Simón Bolívar, Venezuela (2011)
5. Goldáraz D., García M.: Diseño de un algoritmo de navegación háptica. Universidad Rey Juan Carlos, España, (2010)
6. CHAI3D. www.chai3d.org/concept.html
7. PANDA3D. www.panda3d.org
8. Sreeni K.G., Priyadarshini K.: Haptic Rendering of Cultural Heritage Objects at Different Scales, Vision and Image Processing Laboratory, Department of Electrical Engineering, Indian Institute of Technology Bombay (2012)
9. Burdea, G. and Coiffet, P.: Virtual Reality Technology, 2nd ed. John Wiley & Sons (2003)