En español

Análisis de la detección de colisiones en un entorno virtual para aplicaciones hápticas de asistencia quirúrgica

María Luisa Pinto Salamanca [1], José María Sabater Navarro [2], Jorge Sofrony Esmeral[3]

RESUMEN

Este trabajo presenta de forma resumida el uso de dos interfaces hápticas comercialmente disponibles para aplicaciones de asistencia quirúrgica en tareas de entrenamiento y simulación médica. Mediante la integración de *kits* de desarrollo con librerías *software* de código abierto como OpenGL y VCollide, se propone una solución a los problemas de pérdida de realismo táctil y detección de colisiones con objetos sólidos en un entorno virtual.

Con base en los resultados de algunos trabajos relacionados con la obtención, procesamiento y análisis de imágenes para la construcción de modelos, se desarrollan aplicaciones hápticas de interacción con sólidos rígidos de interés médico, que incluyen herramientas para la marcación de puntos y trayectorias sobre una superficie, y un algoritmo de reflexión de fuerzas para la simulación de interacción con órganos modelados por medio de técnicas hibridas (renderizado superficial y volumétrico). Se hace especial énfasis en la representación de los instrumentos virtuales quirúrgicos, integrando herramientas con geometría 3D.

Palabras claves: háptica, detección de colisiones, simulador médico.

Recibido: diciembre 7 de 2009
Aceptado: enero 24 de 2011

Introducción

Háptica es un área que estudia e investiga la interacción de la modalidad sensorial del tacto con un mundo virtual. Las interfaces hápticas son dispositivos bidireccionales que proporcionan sensaciones de fuerzas o tacto al operador a través de la misma interfaz con la que envía consignas al sistema remoto; son básicamente posicionadores de avanzadas prestaciones que permiten simular sensaciones táctiles gracias a la realimentación de fuerzas (Gómez, 2005).

[1] Ingeniera electrónica. M.Sc. en Ingeniería – Automatización Industrial, Universidad Nacional de Colombia. Profesora, Escuela de Ingeniería Electromecánica, Universidad Pedagógica y Tecnológica de Colombia Uptc. marialuisa.pinto@uptc.edu.co.

[2] PhD. en Ingeniería, Universidad Miguel Hernández. M.Sc. en Ingeniería Industrial, ETSII - Universidad Politecnica Valencia, España. Profesor Coordinador Virtual Reality and Robotics Lab. vr2, Universidad Miguel Hernández de Elche, España. j.sabater@umh.es.

[3] Ph.D. en Sistemas de Control, University Of Leicester, Reino Unido. M.Sc., Technology And Medicine. Profesor, Departamento de Ingeniería Mecánica y Mecatrónica, Universidad Nacional de Colombia. jsofronye@bt.unal.edu.co.

In English

Analysing collision detection in a virtual environment for haptic applications in surgery

María Luisa Pinto Salamanca[4], José María Sabater Navarro[5], Jorge Sofrony Esmeral[6]

ABSTRACT

This paper presents an analysis of two commercially-available haptic interfaces that can be used in surgical training and medical simulation. Integrating development kits with open source software libraries like OpenGL and VCollide led to proposing a solution to problems like loss of tactile ability and detecting collisions between objects in a virtual environment.

Haptic applications were based on results regarding capturing, processing and analysing images for building models for interaction with rigid bodies of medical interest. The application included tools for marking points and paths on a virtual surfaces and force reflection algorithms for simulating interactions with surface/volumetric 3D models. Immersion characteristics and the effect of virtual surgical instruments were analysed.

Keywords: haptics, collision detection, medical simulator.

Received: December 7th 2009
Accepted: January 24th 2011

Introduction

Haptics is a research area concentrating on studying the interactions of the sense of touch with virtual objects. Haptic interfaces are bi-directional devices transmitting the sensation of forces and/or contact to an operator at the same time as such operator sends commands to a remote area (i.e. they render advanced interactions allowing procedures requiring tactile sensations through force feedback to be simulated (Gomez, 2005).

[4] Electronic Engineering. M.Sc. in Engineering – Industrial Automation, Universidad Nacional de Colombia. Professor, School Electromechanical Engineering, Universidad Pedagógica y Tecnológica de Colombia Uptc. marialuisa.pinto@uptc.edu.co.

[5] PhD. in Engineering by the Universidad Miguel Hernández. M.Sc. in Industrial Engineer by the ETSII - Universidad Politecnica Valencia, Spain. Professor Coordinator, Virtual Reality and Robotics Lab. vr2, Universidad Miguel Hernández of Elche, Spain. j.sabater@umh.es.

[6] PhD. in Control Systems , University Of Leicester. M.Sc., Technology And Medicine. Professor, Department of Mechanical and Mechatronics Engineering, Universidad Nacional de Colombia. jsofronye@bt.unal.edu.co.

En español

De acuerdo a los resultados de selección de una interfaz háptica para aplicaciones de asistencia quirúrgica presentados en (Méndez, 2008), los *joysticks* hápticos Phantom Omni® y Novint Falcon™ son dispositivos de bajo costo con una serie de características electromecánicas que permiten una interacción apropiada en sistemas de teleoperación donde se requiere de ciertas habilidades manuales especiales.

La representación del efector final del dispositivo háptico en el entorno virtual se realiza mediante un solo punto con coordenadas en los ejes (x, y, z), denominado SCP (*surface contact point*), God-Object, HIP (*haptic interface point*) (Popescu, 1999), *proxy* (para el Phantom Omni) o *hdlTool* (para el Novint Falcon). A partir de este punto se efectúan las transformaciones de la geometría que representan la herramienta táctil o cursor háptico, el cálculo de colisiones y la realimentación de fuerzas (figura 1). Mientras no exista colisión entre el cursor háptico y una superficie (con la cual se desea interactuar en el entorno tridimensional), coincidirán las posiciones del efector final con el proxy, teniendo en cuenta las transformaciones necesarias para relacionarlo con la escena virtual; cuando exista un contacto entre el proxy y el sólido, el efector final podrá penetrar en el objeto, sin embargo el proxy se mantendrá en su superficie. La fuerza que se enviará entonces al dispositivo puede ser calculada siguiendo un modelo, por ejemplo masa-muelle, basado en la distancia existente entre el punto de contacto y el dispositivo háptico, o determinada por medio de una proporcionalidad con la superficie colisionada.

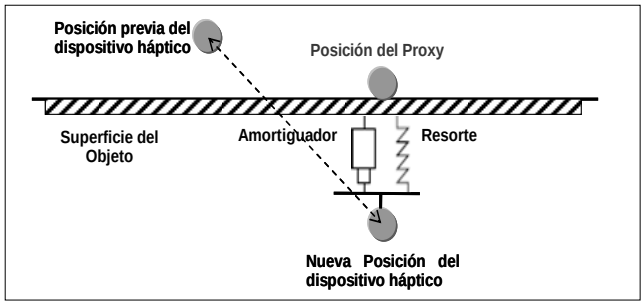

Figura 1. Relación del cursor o Proxy y la posición del dispositivo háptico (SensAble, 2005).

Este trabajo propone la integración de un conjunto de librerías para simular el contacto de un objeto sólido y una herramienta háptica representada también como un sólido en una escena virtual. Esto se realiza valiéndonos de un algoritmo para la detección de colisiones entre objetos rígidos y modelos superficiales triangulares cargados desde archivos *.obj, a partir del estudio de la librería VCollide y su integración con OpenGL® y HDAPI. Se generaliza, además, este procedimiento, implementándolo con el dispositivo Novint Falcon™ y su SDK HDAL (Novint, 2008), el cual no incluye algoritmo para la detección de colisiones. Las aplicaciones se desarrollaron sobre una plataforma Windows XP de 32-bits, con el compilador Microsoft® Visual Studio .Net 2005, en un PC con AMD Athlon™ 64 X2 Dual Core Processor TK57 1.90 GHz, 1.75 GB RAM y tarjeta gráfica NVIDIA GeForce 7000M 256 MB.

In English

According to (Mendez, 2008), haptic interfaces like Phantom Omni and Novint Falcon have specific electro-mechanical characteristics making them an appropriate choice for simulating procedures requiring some special manual ability.

The final mechanism effecter and its translation to a virtual space are done by mapping a single point having coordinate frame (x, y, z) in the axis labelled surface contact point (SCP), god-object, haptic interface point (HIP) (Popescu, 1999), Proxy (Phantom Omni) or hdTool (Novint Falcon). All geometric transformations are made regarding this point, or haptic cursor and aid virtual tool representation, collision detection and functionality force feedback (see Figure 1). If no collisions occur between the cursor and virtual objects then the desired position and the real position of the proxy coincide (after all geometric transformations have been applied). On the other hand, if collisions occur, then the physical final effecter can "enter" the object but the proxy has to remain on the surface of such virtual object. The force which will be fed-back to the haptic device is now calculated using such position discrepancy and may use different force interaction models to quantify this interaction (i.e. the mass-spring-damper used throughout this presentation). Alternatively, force-feedback signal intensity can be computed as a proportional value of the overall collision area; thus, if more area comes into contact, then signal intensity is higher and vice versa.

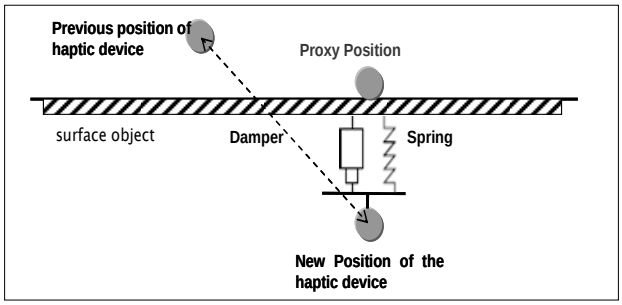

Figure 1. Relationship between proxy position that of the haptic device (SensAble, 2005)

This work proposed developing a set of software libraries to help simulating the interaction between a virtual solid object and a proxy (also represented in the visual scene as a geometrical solid). Collision detection algorithms had to be used so that rigid object contact and superficial triangular mesh models could be detected. VCollide is a publically-available algorithm for detectind *.obj files and easy integration with OpenGL and HDAPI. The same software architecture was used in this work for integrating Novint Falcon and its SDK HDAL (Novint, 2008), advancing this low-cost haptic device's functionalities. Developments were based on a Windows XP 32-bit platform, Microsoft Visual Studio.Net 2005, AMD Athlo 64 X2 Dual Core Processor TK57 1.90 GHz, 1.75 GB RAM y and an NVIDIA GeForce 7000M 256 MB graphical unit.

En español

Integración de los dispositivos hápticos

Descripción de equipos y software

Interfaz Phantom Omni®

El dispositivo Phantom Omni®, de SensAble Technologies, Inc. (http://www.sensable.com/haptic-phantom-omni.htm) (figura 2b), es una interfaz háptica tipo joystick de configuración serial con 6 GDL y realimentación de fuerzas nominales de 3.3N en tres ejes: x, y, z. El OpenHaptics® SDK es el kit de desarrollo, proporcionado con el dispositivo háptico Phantom Omni® en el OpenHaptics Toolkit v.2.0, que permite crear aplicaciones de software para el manejo de los dispositivo hápticos Phantom® de SensAble Technologies. Este SDK incluye el conjunto de librerías HDAPI (*Haptic Device API*), HLAPI (*Haptic Library API*), utilidades, controladores del dispositivo (*Phantom Device Drivers, PDD*) y ejemplos de códigos fuente; en este trabajo se usaron las librerías Open Haptics Academic Edition OHAE v2.0.

Interfaz Novint Falcon™

El Novint Falcon™ (figura 2 c) es un dispositivo con realimentación de fuerzas de configuración paralela, fabricado por Novint Technologies Inc., el cual permite la interacción en escenarios tridimensionales 3D. El Novint HDAL SDK es el kit de desarrollo para el dispositivo Novint Falcon y contiene toda la documentación y los archivos de software necesarios para desarrollar aplicaciones con el dispositivo háptico desde la capa de abstracción HDAL (*Haptic Device Abstraction Layer*). La versión utilizada en los experimentos desarrollados fue HDAL SDK v.2.1.3.

OpenGL API y VCollide

OpenGL® API es una interfaz de programación usada para creación de entornos virtuales 3D en tiempo real. Fue desarrollada por Silicon Graphics a partir de su biblioteca IRIS GL, y se considera actualmente como la API libre y portable más utilizada en la industria para el desarrollo de aplicaciones gráficas 2D y 3D (Hearn, 2006). Permite la generación de nuevos modelos o la interacción con modelos creados con otros paquetes gráficos a partir del manejo de ficheros WaveFront .obj y del uso de las librerías *gl, glu* y *glut* (Bradford, 2004) en lenguajes de programación como C++, C#, Java y Visual Basic. En el trabajo presentado se usaron las librerías OpenGL® API gl, glu, glut versión 3.7.6 para Windows®.

Por la facilidad que ofrece para trabajar con mallados triangulares, por el tipo de reporte de colisión y por registrar menor tiempo promedio de ejecución en comparación con otras librerías (Caselli, 2002; Muñoz-Moreno, 2004), se seleccionó VCollide para analizar la detección de colisiones en un entorno háptico. Esta librería fue desarrollada por el grupo GAMMA (*Geometric Algorithms for Modeling, Motion, and Animation*, http://www.cs.unc.edu/~geom/), de la Universidad de Carolina del Norte. Está escrita en C++ y fue diseñada para trabajar en ambientes que contienen un gran número de objetos geométricos formados por mallas de triángulos. Los experimentos de interacción háptica se desarrollaron con VCollide v.201, compilada con Microsoft Visual Studio® v.6.0 y v.8.0.

In English

Integrating haptic devices

Software and hardware

Phantom Omni

SensAble Technologies Inc. (http://www.sensable.com/haptic-phantom-omni.htm) makes the Phantom Omni haptic device (Figure 2.b) which has a 6 -DOF joystick-type serial configuration and 3.3N rated force feedback on the (x,y,z) axis. OpenHaptics SDK is provided by the manufacturer and is part of the OpenHaptics Toolkit v.2.0 which allows different software to be developed involving the human-machine interaction of a device and an operator. The SDK comes complete with a set of development tools and libraries such as HDAPI (Haptic Device API), HLAPI (Haptic Library API) utilities, Phantom Device drivers (PDD) and source code examples; OpenHaptics Academic Edition OHAE v2.0 libraries were used in this paper.

Novint Falcon

Novint Falcon (Figure 2.c) manufactured by Novint Technologies Inc is a force-feedback haptic device having parallel cinematic chain configuration. The Novint HDAL SDK contains the drivers and libraries needed for developing haptic environments; this is done from a haptic device abstraction layer (HDAL). HDAL SDK v.2.1.3 was used for all the developments presented in this paper.

OpenGL API and VCollide

OpenGL API is a programming interface used to create 3D virtual objects and interact with them in real time. It was developed by Sillicon Graphics based on their IRIS GL library and is considered to be the most used portable, open-source API in the industry for 2D and 3D applications (Hearn, 2006). It allows new models to be produced, or interaction with models created using other graphical suites exporting files in WaveFront *.obj extension. Interaction with objects is possible through *gl, glu* and *glut* libraries (Bradford, 2004). Programing can be done in C++, C#, Java or Visual Basic., OpenGL API gl, glu, glut v3.7.6 for Windows was used in the work presented here.

VCollide was selected as our collision detection algorithm as it offered the possibility of working with triangular mesh 3D objects, very complete collision log reports and has very high levels of performance (measured in time to detect) compared to other open-source libraries (Caselli, 2002; Muñoz-Moreno, 2004). The library was developed by Geometric Algorithms for Modeling, Motion, and Animation (GAMMA) (http://www.cs.unc.edu/~geom/) at the University of North Carolina. It is written in C++ and was optimized to work in highly congested environments with objects modelled as 3D triangular mesh surfaces. The experimental results were based on using VCollide v.201 compiled using Microsoft Visual Studio v.6.0 and v.8.0.

En español

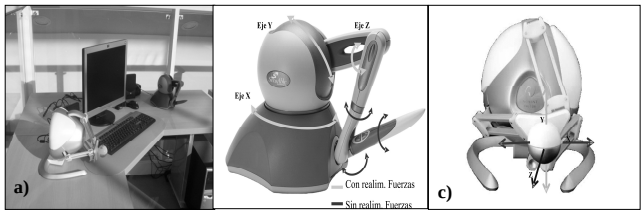

Figura 2. Interfaces hápticas usadas. a) Estación de trabajo. b) Interfaz Phantom Omni®. c) Interfaz Novint Falcon™

Integración OpenGL y VCollide

Teniendo en cuenta que tanto las librerías VCollide como OpenGL se basan en el lenguaje de programación C++, la integración para una aplicación de realidad virtual con objetos colisionables resulta una tarea sencilla. Para la integración de geometrías más elaboradas con OpenGL se utilizaron archivos WaveFront *.obj, que permiten una fácil interpretación con rutinas de C++. Estos archivos definen la geometría y otras propiedades del objeto, las cuales pueden ser desplegadas en visualizadores de prestaciones avanzadas. El archivo *.obj puede ser creado a partir de una exportación con dicho formato desde cualquier programa de modelado 3D.

En la aplicación se generaron, importaron y exportaron modelos de sólidos 3D con el programa de modelado Blender v.2.46 (www.blender.org), teniendo en cuenta que el modelo *.obj debía ser triangular para que se pudiera adicionar como objeto colisionable en VCollide. Para interpretar los archivos *.obj de cada objeto en la escena se desarrolló una clase que permitiera ordenar la información de caras y vértices, escalizar según el tamaño de cada objeto, calcular las normales de los triángulos, dibujar los sólidos y adicionarlos como nuevos objetos colisionables. Con el reporte de VCollide se rescató la información del par de objetos colisionados y los polígonos de solapamiento.

Experimentos de interacción háptica

La librería HDAPI proporciona un control directo del Phantom Omni®. Por medio de las funciones y utilidades ofrecidas se puede manejar la configuración del dispositivo háptico, comprobar su estado (posición, velocidad y fuerza del efector final), y generar fuerzas indicando la intensidad, dirección e instante de aplicación. La representación del cursor háptico se realiza mediante un sólido virtual teniendo en cuenta que las transformaciones del dispositivo son traslaciones y rotaciones efectuadas con respecto a la posición del proxy en la escena.

Estas consideraciones permitieron integrar las librerías VCollide, OpenGL y HDAPI de OpenHaptics SDK, haciendo que el vector de fuerza a aplicar dependiera de las colisiones detectadas entre los sólidos.

Con la inserción de mallados (de variados tamaños y número de vértices) obtenidos previamente por procesamiento de imágenes médicas (Tibamoso, 2009) se evaluó el desempeño de algoritmos implementando las librerías HDAPI y HLAPI de Open Haptics, VCollide, HDAL y OpenGL. La herramienta o cursor háptico se representó con modelos superficiales de un lápiz, un bisturí, una fresa y un martillo, intercambiables gracias a un menú de opciones. Los modelos superficiales, incluidos como archivos en formato WaveFront *.obj, contenían desde 528 hasta 1.794.041 polígonos triangulares. El análisis se concentró en comprobar el desempeño de:

In English

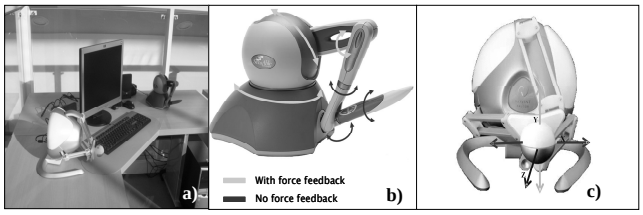

Figure 2. Haptic devices used. a) Work Station. b) Phantom Omni. c) Novint Falcon

Integrating OpenGL and VCollide

As VCollide and OpenGL use C++ programming language, their integration within a high level application was simple. For more complex scenarios, their interaction using VCollide was done using WaveFront *.obj files which allowed easy interpretation using C++ routines. These files defined an object's geometry and other properties which could be visualized. Files with *.obj extension can be generated using any commercially (or publically) available 3D-modelling software package.

3D models were generated, imported and exported in the application using open-source Blender v.2.46 (www.blender.org) 3D modelling software and objects were exported in *.obj format to add them as a collision-able objects by VCollide. A class object was developed to interpret *.obj files for haptic purposes to organise information about vertices and surfaces, to allow scaling objects, to calculate the normal vector associated with each triangle in the mesh, to draw objects and add them as collision-able objects. VCollide log reports were used and only the colliding objects' position was of interest.

Experiments and haptic interaction

HDAPI libraries provide direct control of Phantom Omni. The different functionalities and utilities allow configuring the device, sense joystick status (final effecter position, velocity and force) and generate forces by indicating its direction and intensity. The haptic cursor is represented as a virtual solid that takes into account the different geometric transformations that the object is subjected to, i.e. rotations and translations regarding the scenes coordinate frame.

These functionalities allowed easy integration of VCollide, OpenGL and OpenHaptics SDK HDAPI and rendered the force vector dependent on the collisions detected between objects.

Various sized medical images having several vertices (Tibamoso, 2009) were inserted and performance was analysed for different object properties using HDAPI and HLAPI from Open Haptics, VCollide, HDAL and OpenGL. The "tool" or haptic cursor was represented as a pencil, a scalpel, a drill and a hammer (changing them by using an options menu). The superficial mesh model inserted a range of between 528 to 1,794,041 triangular polygons. Analysis concentrated on determining performance by:

En español

1. Herramienta háptica con modelo superficial y librería HLAPI: interacción del dispositivo Phantom Omni (figura 3).

2. Herramienta háptica con modelado superficial y librerías HDAPI/ Vcollide: interacción del dispositivo Phantom Omni.

3. Herramienta háptica con modelado superficial y librerías HDAL/VCollide: interacción del dispositivo Novint Falcon.

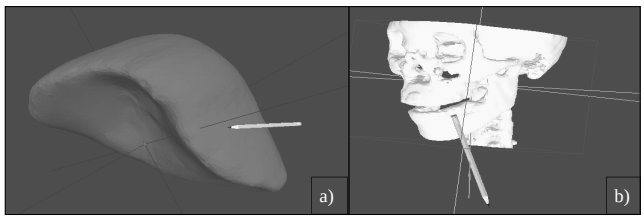

Figura 3. Interacción háptica con modelos superficiales sobre OpenGL. a) HLAPI. b) HDAL y VCollide.

Algoritmo de reflexión de fuerza por aproximación volumétrica

La renderización de fuerzas es el procedimiento por el cual se transmite una sensación táctil mediante la aplicación de una serie de vectores de fuerza a través del dispositivo háptico. Estos vectores de fuerza son actualizados según la interacción y transformación dinámica entre los objetos virtuales de la escena y la interfaz háptica. Una vez representado el dispositivo háptico en el entorno virtual, el cálculo y aplicación de los vectores de fuerzas se pueden dar en dependencia del movimiento, el tiempo, o una combinación de ambos. A partir de la información proporcionada en la detección de colisiones con VCollide se propuso un algoritmo que utiliza las librerías HDAPI y HDAL para permitir la aplicación de un vector de fuerzas con dirección y magnitud dependiente de la geometría y volumen de las partes colisionadas (Sabater, 2009).

Cuando se detecta la colisión, la información del reporte de VCollide y la inicialización de la aplicación permiten conocer cuántos objetos sólidos han colisionado, así como cuáles, y cuantos triángulos se han solapado en cada sólido. El volumen de los sólidos incluidos en la escena se determina a partir de las coordenadas de los vértices que forman los polígonos triangulares de cada sección (figura 4). Por simplicidad del algoritmo, se calculó el volumen envolvente de una primitiva geométrica, que se seleccionó como un paralepípedo que reduce el cálculo matemático a la multiplicación de tres coordenadas (ancho, alto y profundidad) determinadas por la diferencia entre los componentes mínimos y máximos de cada eje coordenado. Esto permitió la aplicación de una técnica de caja envolvente alineada con el objeto (OBB), calculada por un algoritmo que acepta transformaciones de traslación y rotación del i-ésimo objeto ^{vs_i} de la siguiente manera:

1.Determinar $(x_{\min}, x_{\max}), (y_{\min}, y_{\max}), (z_{\min}, z_{\max})$

2. Calcular $x_v = x_{\max} - x_{\min}$, $y_v = y_{\max} - y_{\min}$, $z_v = z_{\max} - z_{\min}$

3.Transformar objeto, ir a *1)*

In English

1. Haptic tool with surface mesh model and HLAPI: interaction with Phantom Omni device (Figure 3);

2. Haptic tool with surface mesh model and HDAPI/ Vcollide: interaction with Phantom Omni device; and

3. Haptic tool with surface mesh model and HDAL/VCollide: interaction with Novint Falcon device.

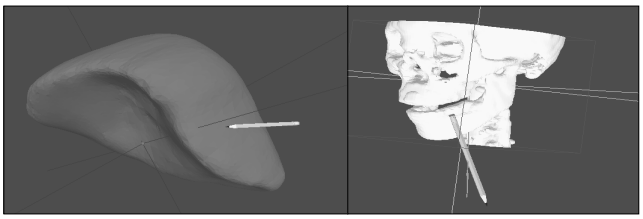

Figure 3. Interaction of haptic device and superficial model using OpenGL. a) HLAPI. b) HDAL and VCollide

Force reflecting algorithm using volumetric approximations

Rendering force-feedback is the main procedure by which the device can transmit tactile sensations to an operator. The force vector can be updated using different forms of interaction and has to take the proxy's dynamic geometrical transformations into account. Once the device has been integrated into the virtual scene, force vectors can be calculated and applied, depending on movement, time, position or a combination of these. Based on the information given by the collision detection algorithm (i.e. VCollide), a method was proposed that used HDAPI and HDAL libraries to render collided section geometry and volume-dependent force vector intensity and heading (Sabater, 2009).

Once a collision had been detected, the VCollide log report and scene set-up provided information about the collided sections, as well as how many triangles had been overlaid and the volume associated with such superimposition on each collided object. The volume of the objects inserted in the scene was determined on the coordinates for the vertices constituting each triangular section (Figure 4). For simplicity, volume was calculated using a geometric primitive shape circumscribing the triangle. This primitive was chosen as a parallelepiped so that mathematical calculations became reduced to multiplication on a three-coordinate axis, namely (width, height, depth), in turn being calculated by the difference between each coordinate frame's maximum and minimum components. This led to applying an object bounding-box algorithm that accepted translational and rotational transformations of the i-th object ^{vs_i} as follows:

1. Determining $(x_{\min}, x_{\max}), (y_{\min}, y_{\max}), (z_{\min}, z_{\max})$

2. Calculating $x_v = x_{\max} - x_{\min}$, $y_v = y_{\max} - y_{\min}$, $z_v = z_{\max} - z_{\min}$

3. Object transformation and "go to" step *1)*

En español

Para simular la interacción del sólido y el cursor háptico se supuso que el cálculo de la fuerza dependía de la información geométrica, vectorial y volumétrica de las partes del sólido cuyas caras han colisionado con la herramienta, utilizando información de los triángulos colisionados Tc_i, sus vértices vc_i, normales nc_i de la parte pc_i y volumen Voc_i correspondiente. Se planteó entonces la aplicación de un vector de fuerza con magnitud proporcional al volumen de la parte sólida tocada y con dirección dada por la sumatoria de las normales de cada una de las caras colisionadas en el sólido.

In English

It was proposed using vector, geometric and volumetric information to update the force feedback signal to simulate the interaction between virtual solids and haptic cursor. This information was retrieved via the VCollide log and useful data could be extracted in the form of the position for collided objects Tc_i, their vertices vc_i, normal vectors nc_i of section pc_i and corresponding volume Voc_i. It should be pointed out the information received constituted data regarding collisions between the proxy and virtual (collision-able) objects. Rendering a force vector having magnitude proportional to collided objects' volume and whose direction was mean normal vector of the collisions was thus proposed.

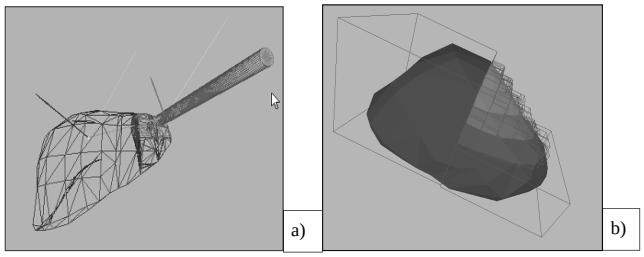

Figura 4. Representación de superficies. a) Mallado superficial de los objetos. b) Modelo OBB de cada objeto.

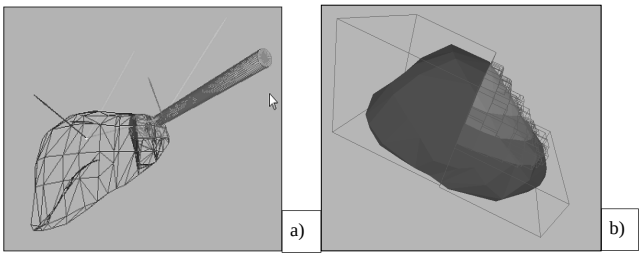

Figure 4. Representation of surfaces. a) surface mesh for an object. b) OBB for each object.

Resultados

La detección de colisiones con VCollide contra HLAPI y HDAL

En la figura 5 se indican los tiempos de inicialización para las ejecuciones de cada dispositivo háptico en función del número de polígonos triangulares incluidos en cada modelo. Esta información evidencia que para cargar un objeto en la escena virtual, con 233.764 triángulos en su mallado superficial, con el dispositivo Novint Falcon (librerías HDAL y VCOLLIDE), se debieron esperar 25.342 ms antes de comenzar la aplicación, mientras que en la interfaz Phantom Omni (librerías HLAPI y HDAPI-VCOLLIDE) los tiempos de inicio para convertir los objetos en modelos colisionables fueron de 7.109 ms y 7.719 ms.

Experimental results

VCollide *cf* HLAPI/HDAL for collision detection

Figure 5 shows the set-up times for each device and a different number of triangular polygons. It can be observed that when inserting an object having 233,764 triangles, the Novint Falcon device (with HDAL and VCOLLIDE) user had to wait for about 25,342 ms before being able to use the application. The Phantom Omni device (with HLAPI and HDAPI-VCOLLIDE) had reduced set-up times (i.e. 7,719 ms stand-by time).

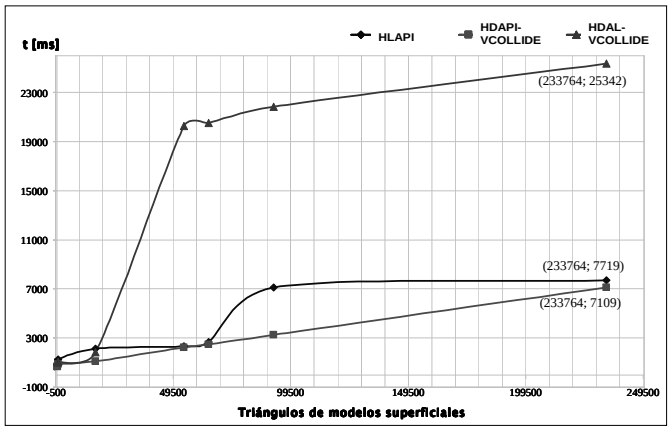

Figura 5. Relación de tiempos de inicialización de las aplicaciones hápticas.

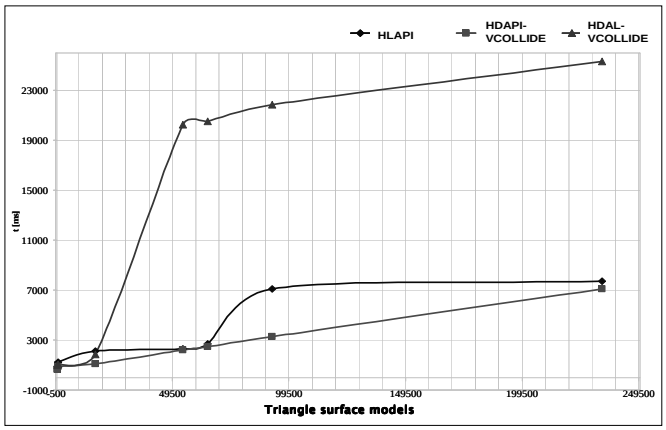

Figure 5. Haptic device set-up times

Al evaluar los componentes del SDK OpenHaptics en relación

OpenHaptics SDK had better initialization times using HDAPI-

En español

con el tiempo de inicialización, se obtuvo mejor respuesta usando HDAPI-VCollide. Sin embargo, el algoritmo de realimentación del HLAPI (que ya viene implementado en la misma librería) muestra una mejor respuesta en cuanto a la intensidad de la señal de fuerza y su variación al mover el cursor háptico sobre el sólido, permitiendo percibir una sensación táctil más suave y continua que con el HDAPI-VCollide. Pero esto sólo es cierto si la detección de colisiones se efectúa entre el proxy y un sólido, lo cual ocasiona que haya posibles inmersiones de la herramienta aun cuando se estén realimentando fuerzas.

Otro aspecto a considerar con el uso de las librerías HL fue su dependencia del tipo de renderizado gráfico para los polígonos que conforman el modelo superficial, cuyas variaciones visuales se obtuvieron por medio de la función *glPolygonMode* de OpenGL. El renderizado háptico con HL fue indetectable cuando los sólidos fueron visualizados a partir de puntos o líneas. En la tabla 1 se indican otros criterios de comparación.

Tabla 1. Comparación de aplicaciones con las librerías HLAPI, HDAPI-VCollide Y HDAL-VCollide

Dispositivo	Phantom Omni®		Novint Falcon™
Parámetro	HLAPI	HDAPI-VCollide	HDAL-VCollide
Sensación de inmersión táctil para el usuario	Alta	Alta	ALTA
Sensación de inmersión visual para el usuario	Alta	Alta	ALTA
Librería de detección de colisiones	HLAPI	VCollide	VCollide
Algoritmo de detección de colisiones	HLAPI	OBB's	OBB's
Modos de visualización con realimentación táctil	GL_FILL	GL_POINT, GL_LINE y GL_FILL	
Integración de archivos *obj	Posible		
Sincronización de hilos háptico y gráfico	Automática		
Actualización de transformaciones de escena	Automática	Realizable	Realizable
Algoritmos para cálculos de fuerza	Ya incluidos	Deben definirse	
Libertad de movimiento	Alta	Alta	Baja

Herramientas virtuales de exploración táctil en un modelo superficial

Con el uso de las funciones *callbacks* de las librerías HLAPI y HDAL se facilita el manejo de eventos de interés para la realimentación táctil: 1) detección de colisión de los objetos de la escena con el proxy o punto a partir del cual se dibuja el cursor háptico; 2) captura de entradas digitales por medio de los pulsadores con los que cuentan los dispositivos hápticos. Con estas funciones se implementaron herramientas de exploración táctil para la marcación de puntos y trayectorias sobre modelos superficiales de interés médico (figura 6a). Se incluyó la opción de almacenamiento de los puntos marcados en archivos de tipo texto plano.

La figura 6b presenta la imagen de la aplicación de integración OpenGL, VCollide y HDAPI. En este caso se apreció el efecto del algoritmo de fuerzas propuesto; la dirección del vector de fuerza (flecha negra) es correspondiente con la sumatoria de la dirección de las normales y la magnitud cambia con respecto al volumen de la superficie tocada con la herramienta háptica. Aunque la función del cálculo de volumen que se realizó no es del todo confiable para modelos superficiales con cavidades, lo cual implicaría un mayor costo computacional, se pudo concluir experimentalmente que a menor tamaño de los objetos colisionables, mejor será la aproximación volumétrica y el vector de fuerza aplicado.

In English

VCollide than with its original firmware. Nonetheless, force feedback using HLAPI (which is the standard manufacturer's firmware) had better results in terms of intensity and orientation when the dynamic cursor moved along a surface, allowing smoother, more continuous tactile sensations than when HDAPI-VCollide was used. This was only the case when the proxy were represented as a single point; tactile sensation was acceptable if the proxy was a solid, but visual immersion became lost as the objects became superimposed on each other and it seemed as if the proxy was entering the solid.

Another aspect t needing to be considered was graphical rendering and its dependence on the number and type of polygons used to generate the virtual objects and their surface mesh representation. Visual variations were measured using the *glPolygonMode* function in OpenGL. As mentioned previously, haptic rendering was transparent if the solids were represented as a collection of points or lines. Table 1 shows the other comparison criteria.

Table 1. Comparing application performance using HLAPI, HDAPI-VCollide Y HDAL-VCollide

Device	Phantom Omni		Novint Falcon
Parameter	HLAPI	HDAPI-VCOLLIDE	HDAL-VCOLLIDE
Tactile sensation immersion	High	High	High
Visual sensation immersion	High	High	High
Collision detection library	HLAPI	VCollide	VCollide
Collision detection algorithm	HLAPI	OBB's	OBB's
Visualization modes with tactile feedback	GL_FILL	GL_POINT, GL_LINE y GL_FILL	
Integrating *obj files	Possible		
Synchronising visual/haptic threads	Automatic		
Transformation scene update	Automatic	Possible	Possible
Force reflecting algorithm	Included	Must define	
Freedom of movement	High	High	Low

Tactile exploration tools for surface mesh models

Using existing callback functions in HLAPI and HDAL, it was possible to manipulate the device through events of interest and tactile feedback. Some events of interest were collision detection between the proxy and objects in the virtual scenario and digital input available in the device. These events were used to implement tactile exploration tools marking points and trajectories on the virtual solid, placing special emphasis on objects of medical interest (Figure 6). An output report was added which mapped all marked point and trajectories.

Figure 6 (b) presents the integration results including the use of OpenGL, VCollide and HDAPI. This image shows the proposed algorithm's mechanism; the arrow represents force vector direction and its intensity was proportional to contact volume. Although contact volume estimation was imprecise (in particular when dealing with non-convex geometries), it was possible to reduce such discrepancy at the expense of higher computational cost. The experimental results led to concluding that volumetric approximation, and thus the feeling of immersion, became more precise as an object become smaller; this enhanced force reflection properties and made the system appear "more real".

En español

Conclusiones

Las ventajas de programar el dispositivo Phantom Omni con la

librería HLAPI son notorias en cuanto a la realimentación táctil de forma continua y a la inserción previa de algoritmos de renderización de fuerza. Sin embargo, se encuentran considerables pérdidas de inmersión visual y realismo de la aplicación, ya que la detección de colisiones se realiza únicamente con respecto a un punto, lo cual puede ser indeseado en un simulador médico. Con el uso de VCollide para la detección de colisiones se superan estos inconvenientes, pero se deben generar algoritmos para la identificación de triángulos y vértices de los modelos superficiales, la detección de posición del efector final y la aplicación de fuerzas con la integración de las librerías HDAPI para el dispositivo Phantom Omni y HDAL para el Novint Falcon.

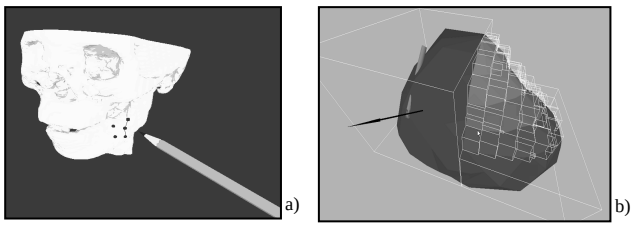

Figura 6. Aplicaciones. a) Uso de callbacks de HL para manejo de eventos. b) Simulaciones del algoritmo de fuerza

En el algoritmo para la generación de la escena gráfica mediante OpenGL, con el uso de OpenHaptics o HDAL, la sincronización de los hilos háptico y gráfico es transparente para el usuario. No obstante, las aplicaciones de HLAPI en las que se cargaron modelos superficiales con más de 230.000 caras triangulares provocaron lentitud en la ejecución gráfica. Las actualizaciones de la escena (ubicación de puntos de vista, ejecución de transformaciones del cursor háptico, acercamientos, etc.) son automáticas con HLAPI, pero la realimentación táctil solamente se presenta cuando los sólidos son visualizados como modelos superficiales completos (GL-FILL), es decir, que se pierde la información táctil cuando se visualizan los objetos por sus vértices o por las líneas que componen el mallado (GL-POINT, GL-LINE y GL-FILL). Esta situación es imperceptible con el uso de HDAPI o HDAL.

En las simulaciones hechas para aplicaciones de asistencia quirúrgica se consideró que los objetos de la escena virtual (tanto órganos como herramientas del instrumental quirúrgico) fueron rígidos (geometría constante) y atómicos (topología constante), esto es, no se podían dividir, ni seccionar. En dichas aplicaciones estos objetos sólo experimentaron rotaciones y traslaciones, las escalizaciones se realizaron previamente al cargar la información sobre los polígonos de objetos. Sin embargo, para lograr que estas simulaciones puedan ser utilizadas con éxito en el campo quirúrgico, se deben incluir algoritmos de deformación en un campo tridimensional que permitan analizar órganos como objetos deformables, además de herramientas del instrumental quirúrgico especiales, como la aguja y el hilo de sutura, y lograr un equilibrio entre el costo computacional de los algoritmos y el realismo para simular estas situaciones.

In English

Conclusions

The advantages of using Phantom Omni and HLAPI (provided by the manufacturer) were noticeable in terms of force reflection and tactile sensation; the joystick had smooth dynamic transitions. Nonetheless, a considerable loss of visual immersion was present and the application's realism could become lost as a consequence. It was thought that the library should compromise visual immersion to obtain high tactile immersion and only use a single point representation for the proxy. Consequently, such software architecture is undesirable for medical simulators. By contrast, using VColide requires a more complex architecture including algorithms for identifying surface models' triangles and vertices, transforming and identifying the proxy's position and rendering forces using volumetric contact information. This requires integrating HDAPI (for Phantom Omni devices) and HDAL (for Novint Falcon devices) with VCollide.

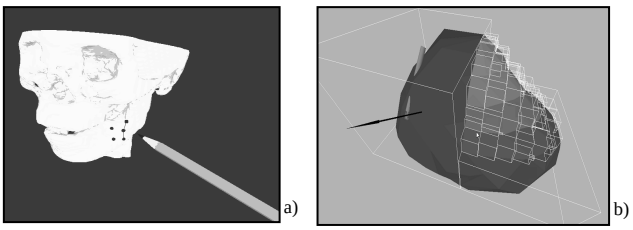

Figure 6. Applications. a) Using HL callback for event handling. b) Force algorithm simulations

The graphical rendering algorithm was transparent when OpenGL was used, with either OpenHaptics or HDAL libraries. This was also true for synchronising haptic and graphic threads. Nonetheless, HLAPI applications had reduced performance when superficial models having more than 230,000 triangles were loaded, especially in graphical rendering. Scene updates (points of view, haptic cursor transformations and zoom) were automatic with HLAPI, but tactile feedback was only possible for full surface models (GL-FILL); in other words, tactile information when objects were visualized using vertices or lines composing a surface mesh model (GL-POINT, GL-LINE and GL-FILL). This was not the case when using HDAPI or HDAL.

Simulation focused on exploring functionalities within the context of medical training applications. Objects considered were thus organs or some type of surgical tool. Objects were assumed to be rigid (constant geometry) and atomic (constant topology); in other words objects could not be divided and their shape did not change upon contact. Simulations only considered rotational and translational transformations; no on-line scaling was performed. Some of the immediate advances that must be made to use this type of technology in a medical training scenario would be real-time elasticity simulation algorithms allowing 3D objects to be represented as being deformable, details modelling special surgical tools such as needles and suture thread, leading to an attractive balance between computational cost and immersion in specific situations.

Referencias / References

Bradford, J., Using OpenGL & GLUT in Visual Studio .NET, 2003. http://csf11.acs.uwosh.edu/cs371/visualstudio/index.html ,2004.

Caselli, S., Mazolli, M., Reggiani, M., A experimental evaluation of colision detection packages for robot motion planning. Procedings of the IEEE/RSJ Intl., Conference of Intelligent Robots and Systems EPFL. Lausanne, Switzerland, 2002.

Gómez, J.M., Muñoz, V., Dominguez, F., Serón, J., Sistema experimental de Tele-Cirugía., Revista eSalud, Vol. 1, No 2, Abril-Junio, 2005.

Hearn, D., Baker, M., Gráficos por computadora con OpenGL., PEARSON Prentice Hall, Madrid. 3 Edition. 2006.

Méndez, L., Pinto, M., Sofrony, J., Definición y Selección de una Interfaz Háptica para Aplicaciones Preliminares de Asistencia Quirúrgica., Memorias del III Encuentro Nacional de Investigación en Posgrados – ENIP 2008, Universidad Nacional de Colombia, Mayo, 2008.

Novint Technologies, Inc., Haptic Device Abstraction Layer (HDAL)., Programmer's Guide HDAL SDK VERSION 2.1.3 Agosto, 2008.

Muñoz-Moreno, E., Rodríguez, S., Vilora, A., Lamata, P., Martín, M.A., Luis-García, R., Aja, S., Gómez, E., Alberola, C., Detección de colisiones. Un problema clave en la simulación quirúrgica., Informática y Salud (I+S) vol. 48, Octubre, 2004, pp. 23-35.

Popescu, V., Burdea, G., Bouzit, M. Virtual Reality Simulation Modeling for a Haptic Glove., Computer Animation, 1999. Proceedings. Geneva, Switzerland, 1999, pp.195.

Sabater, J., Pinto, M., Saltaren, R., Sofrony, J., Azorin, J.M., Perez, C., Badesa, J., Force reflecting algorithm in tumour resection simulation procedures., CARS 2009 - Proceedings of the 23rd International Congress and Exhibition, Berlin, Germany, June 23 - 27, 2009, pp. 138-140.

SensAble Technologies, Inc., OpenHaptics Toolkit v. 2.0., Programmer's Guide, 2005, pp. 5-3 5-4.

Tibamoso G., Romero, E., Segmentación y Reconstrucción Simultánea del Volumen del Hígado a partir de imágenes de Tomografía Axial Computarizada, Descripción de proyecto., Universidad Nacional de Colombia, http://www.bioingenium.unal.edu.co/pagpro.php?idp=3Dhigado&lang=es&linea=2, 2009.