Cómo citar este artículo: Pinto-Salamanca, M. L., Sofrony-Esmeral, J. I., & Jiménez, D. F. (2015). Detección de colisiones con librerías v-collide y physx para interacción virtual con interfaces hápticas. Rev.investig.desarro.innov, 5(2), 119-128.

DETECCIÓN DE COLISIONES CON LIBRERÍAS V-COLLIDE Y PHYSX PARA INTERACCIÓN VIRTUAL CON INTERFACES HÁPTICAS

COLLISION DETECTION WITH V-COLLIDE AND PHYSX LIBRARIES FOR VIRTUAL INTERACTION WITH HAPTIC INTERFACES

María Luisa Pinto-Salamanca[1]
Jorge Iván Sofrony-Esmeral[2]
Daniel Fernando Jiménez[3]

Recibido: octubre 22 de 2014
Aceptado: febrero 10 de 2015

Resumen

La integración de dispositivos hápticos en aplicaciones de entrenamiento virtual es una tarea compleja que requiere la integración de una variedad de librerías de software. Este trabajo se concentra en explorar la eficiencia de distintas librerías de detección de colisiones al utilizar modelado mixto (superficial/volumétrico) 3D de objetos en una escena virtual. Aplicando herramientas software de código abierto se desarrolla un entorno de interacción tridimensional con posibilidades de realimentación de fuerzas y de interacción entre componentes virtuales. Se presentan pruebas de interacción virtual con dos dispositivos hápticos, midiendo tiempos de inicialización, cuadros por segundo y consumo de RAM. Los resultados obtenidos argumentan la selección de la librería PhysX comparada con la librería V-Collide, para el análisis en la detección de colisiones en un entorno virtual con realimentación táctil.

Palabras clave: Detección de colisiones, háptica, realidad virtual.

Abstract

Integration of haptic devices in virtual training applications is a complex task that requires the integration of a variety of software libraries. This paper concentrates on exploring the performance of different collision detection algorithms that will allow the real-time manipulation of mixed (superficial/volumetric) 3D virtual models, and hence the geometric manipulation of these objects in a virtual scene. Using open source software tools, a three-dimensional environment was developed with the potential interaction of force feedback and interaction within virtual components. Virtual interaction tests are presented with two haptic devices, measuring initialization times, frame rates and consumption of RAM. The results allow selecting the PhysX library compared to the V-Collide library, for analysis of collision detection in a virtual environment with tactile feedback.

Key words: Collision detection, haptics, virtual reality.

[1] **Ingeniera Electrónica, Magíster en Ingeniería Automatización Industrial, Universidad Pedagógica y Tecnológica de Colombia, Colombia. E-mail: marialuisa.pinto@uptc.edu.co**
[2] **D. en Sistemas de Control, Departamento de Ingeniería Mecánica y Mecatrónica, Universidad Nacional de Colombia, Colombia. E-mail: jsofronye@bt.unal.edu.co**
[3] **Ingeniero Mecatrónico. Gerente Domoti S.A.S., Colombia. E-mail: dfjimenezto@gmail.com**

1. Introducción

Los sistemas de teleoperación permiten extender las capacidades sensoriales y de destreza humana para controlar la realización de tareas de un robot (llamado dispositivo esclavo) en entornos remotos reales o virtuales. En un sistema básico de teleoperación, el operador maneja un manipulador maestro para indicar las acciones que debe ejecutar el manipulador esclavo a través de algún tipo de interfaz, ciertos canales de comunicación y sistemas sensoriales.

La háptica estudia e investiga la combinación de la modalidad sensorial del tacto en un entorno de realidad virtual. Las interfaces hápticas, kinestésicas, o también llamadas interfaces de reflexión de fuerzas, son dispositivos bidireccionales diseñados para reflejar las fuerzas generadas en la simulación del entorno remoto (Ollero et al., 2006). La única diferencia entre los sistemas hápticos y los sistemas teleoperados se relaciona con el tipo de señal de realimentación; en las interfaces hápticas, la zona remota no existe físicamente, y es simulada vía software, mientras que en la teleoperación, la zona remota está compuesta por un robot manipulador y un conjunto de sensores (Sabater, 2003). La mayoría de dispositivos hápticos utilizan las manos, aprovechando la alta densidad de receptores táctiles de los dedos.

La renderización de fuerzas es el procedimiento para la transmisión de una sensación táctil a través de la aplicación de una serie de vectores de fuerza en el dispositivo háptico debida a la interacción y transformación dinámica entre los objetos virtuales de la escena y el elemento que representa a la interfaz. Los métodos para el cálculo de fuerzas varían dependiendo de la configuración física del dispositivo y el modelo de representación háptica que se haga en el entorno virtual, tomado por ejemplo en los brazos robóticos seriales, con respecto a un solo punto de contacto o Proxy conocido también como *háptic interface point* HIP (Ryden et al., 2011).

En los entornos de realidad virtual, en los que se interactúa con un dispositivo háptico, debe considerarse el análisis de la detección de colisiones entre objetos, de forma que permita realimentarse directamente al usuario una sensación de contacto y la aplicación de un modelo de renderización de fuerzas inmediato. La velocidad de reporte de colisión y el modelo de renderización de fuerzas, definen el nivel de inmersión de todo el sistema háptico.

Las librerías para la detección de colisiones son herramientas de software encargadas de comprobar, por medio de modelos computacionales, si los objetos que componen una escena gráfica entran o no en contacto, así como de determinar, en función de las características propias de cada modelo, qué consecuencias deberán tener tales contactos. Aunque son ampliamente utilizadas en los sistemas de simulación quirúrgica (Ruthenbeck & Reynolds, 2013), fue en el campo de los videojuegos donde se desarrollaron. Algunas librerías para detección de colisiones, disponibles y presentadas como paquetes de libre distribución implementadas en ANSIC o C++ son: ICollide, V-Collide, Rapid y Swift desarrolladas en la Universidad de Carolina del Norte (UNC); Solid, V-Clip, physX, entre otras.

El objetivo principal de los algoritmos para la detección de colisiones es calcular las interacciones geométricas entre los objetos, sin importar el número ni la complejidad que los objetos puedan tener. Tradicionalmente, han requerido una gran cantidad de pruebas de intersección geométrica, verificando si todos los polígonos que modelan la superficie de un objeto, intersectan la superficie de otro objeto, determinando de esta manera si dos objetos colisionan. La clave para la detección de colisiones en tiempo real, es el método que permita detectar más rápidamente cuando dos objetos no presentan colisión (Ramírez, 2005).

La mayoría de investigadores proponen algoritmos que ya han dado resultados efectivos y reducen el número de llamados para verificar la intersección entre dos primitivas geométricas. Estas técnicas plantean un tipo de volúmenes envolventes organizados en una estructura jerárquica y con ello evitan una verificación de los pares de primitivas geométricas directamente.

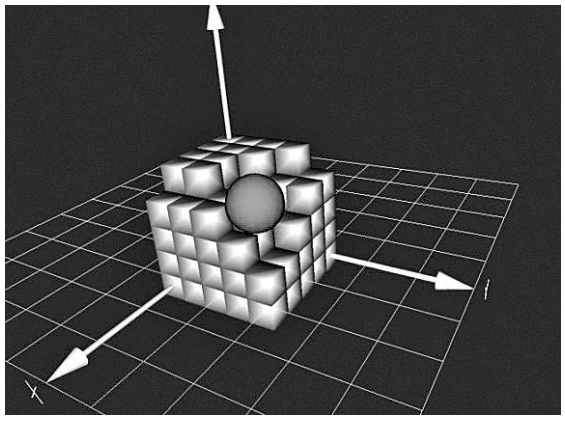
La detección de colisiones en un ambiente virtual con realimentación de fuerzas, se ha analizado previamente comparando el desempeño de librerías como Solid, Swift++ y PQP (Narber, 2010). Mediante el desarrollo de algunas aplicaciones software como herramienta de apoyo docente para la planeación de procedimientos TKR, Total Knee Replacement, se propuso la integración de un modelo mixto, superficial/volumétrico, para simular el maquinado de un hueso en tiempo real (Pinto et al., 2011); el modelo, basado en software de código abierto, incorporaba la tecnología háptica para aumentar las propiedades de inmersión en el desarrollo de simulaciones de cortes, interacción con eliminación de partículas y renderización de fuerzas en función de las superficies que entrarán en contacto. La aplicación inicial utilizó la librería V-Collide (2013) para realizar la detección de colisiones, sin embargo en la primera versión del simulador, se encontraron restricciones en términos de tiempos de inicialización de la aplicación y tiempos de renderizado gráfico y háptico, limitando así el número de objetos volumétricos que se pudieron utilizar en la escena al usar esa librería para detección de colisiones.

En este documento se explora la utilización de alternativas para la detección de colisiones, con el fin de mejorar los tiempos de procesamiento de ejecución y la aplicación futura en un sistema de teleoperación con dos paquetes de librerías que no habían sido comparadas directamente. Las pruebas que se presentan argumentan la selección de la librería PhysX (Geforce, 2010) para el análisis en la detección de colisiones con un modelo virtual mixto, mediante la comparación de desempeño con la librería V-Collide. El desempeño de cada librería es medida respecto a tasas de refresco visual (cuadros por segundo- FPS), tiempos de inicialización y consumo de memoria RAM. El modelo mixto para este caso, consiste en una matriz de cubos de tamaño variable cuyos componentes pueden aumentarse en cada eje coordenado, con la posibilidad de considerar además transformaciones geométricas (traslación, rotación y escalización) y eliminaciones de escena por la interacción con una herramienta virtual que representa un cursor háptico.

2. Materiales y métodos

2.1 Entorno y dispositivos hápticos

Para evaluar el desempeño del sistema y con el fin de generar un modelo volumétrico, se integraron superficies geométricas conocidas (cubos y esferas), lo que requirió desarrollar escenas virtuales compuestas por una matriz de cubos (matriz 3D) construidos a partir de la lectura de vértices (puntos con coordenadas x, y, z) desde archivos WaveFront *.obj (Wavefront's Advanced Visualizer), con la posibilidad de interacción entre los objetos tridimensionales y una interfaz háptica representada por un cursor tipo esfera como se ve en la figura 1, y la opción de desplazar cubos cuando se presenta un contacto háptico.

a. 

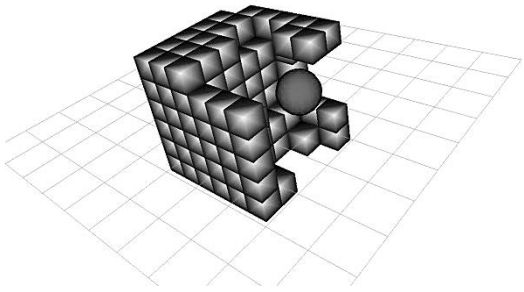
b. 

Figura 1. Escena virtual. a) Matriz de cubos. b) Interacción virtual.

Con el fin de determinar los parámetros de ejecución de cada escena, el tamaño y cantidad de cubos de la matriz 3D varió según las especificaciones del usuario. Por tanto, la cantidad de información a procesar en el sistema, fue proporcional al número de cubos, vértices y caras de cada matriz. Así, para un solo cubo se pudo contar con 6 caras cuadradas, 12 caras triangulares, y de 8 a 36 vértices según la lectura de cada cara. Si la matriz estaba compuesta por 10 cubos por cada lado (cada "arista"), la escena virtual debió actualizar constantemente la información referente a 1000 cubos, 6000 cuadrados, 12000 triángulos y de 8000 a 36000 vértices, ver figura 2.

Mediante un administrador de escenas se seleccionaron las opciones para el manejo de renderizado gráfico y los reportes de detección de colisiones. Las características y componentes de la aplicación fueron: Interacción con el usuario mediante QT v.4.1; renderizado gráfico con QGLViewer v.2.3.4; detección de colisiones con V-Collide v2.01 y Physx 2.8.3.21; integración de la interfaz háptica Phantom Omni® de SenSAble con Openhaptics™ Toolkit v. 2.0.

El simulador generado para las pruebas de comparación se basa en software de código abierto y está desarrollado en lenguaje C++, compilado con QtCreator 1.3.1 basado en Qt 4.6.1 (32 bit), S.O. Windows XP®, CPU Intel® Core(TM)2 Quad Q8200, 2.33GHz, 3GB de RAM.

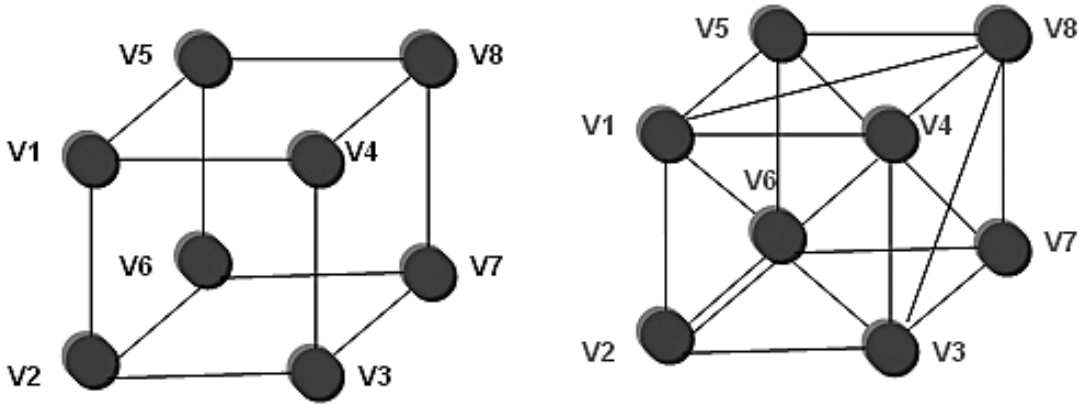


Figura 2. Escena virtual, identificación de vértices y caras de cada cubo.

2.1.1 Entorno de renderizado gráfico con QGLViewer

Qt es un entorno de trabajo basado en C++ para el desarrollo de software en plataforma cruzada, que a partir de un conjunto de widgets, "controles" en la terminología del sistema operativo Windows, se encarga de proporcionar las funciones estándar de un interfaz o GUI. QGLViewer es una librería basada en C++ que facilita la integración de Qt con visores de OpenGL, con las funcionalidades típicas de un entorno tridimensional como movimiento de cámaras usando el ratón, definición de sistemas de referencia, integración de teclado, animación de escenas, capturas de pantalla, entre otros (QGLViewer, 2010).

2.1.2 Detección de colisiones

V-Collide: Por la facilidad que ofrece para trabajar con mallados triangulares, por el tipo de reporte de colisión y por registrar un menor tiempo promedio de ejecución en comparación con otras librerías según (Reggiani, 2002) y (Moreno et al., 2002), se seleccionó V-Collide para analizar el problema de 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.

PhysX: Es un kit de desarrollo diseñado para llevar a cabo cálculos físicos muy complejos, desarrollado por Nvidia® e integrado en sus chips gráficos más recientes. Es una solución multifuncional GPU/PPU que ya ha sido previamente utilizado en aplicaciones de asistencia quirúrgica con realimentación táctil (Maciel et al., 2009), (Chan & Choi, 2009). Las pruebas presentadas en este trabajo fueron realizadas en la CPU, ya que solo interesaba probar la eficiencia del algoritmo de detección de colisiones y no del hardware.

2.1.3 Integración de dispositivos Hápticos

Phantom Omni®: es un dispositivo de SensAble Technologies, Inc. (ver figura 3a), es una interfaz háptica tipo joystick de configuración serial con 6 grados de libertad (GDL) y realimentación de fuerzas nominales de 3.3N en tres ejes, por ejemplo x, y, z. El OpenHaptics SDK es un kit de desarrollo proporcionado con el dispositivo háptico Phantom Omni®, que contiene el OpenHaptics Toolkit con el que se pueden crear aplicaciones de software para el manejo de los dispositivos hápticos Phantom® de SensAble Technologies. En este trabajo se usaron las librerías Open Haptics Academic Edition OHAE v2.0 ya integrada en otros trabajos también con PhysX (Chan & Choi, 2009).

Novint Falcon™: es un dispositivo fabricado por Novint Technologies Inc., ver figura 3b, para interacción en escenarios 3D, con realimentación de fuerzas. Se ha diseñado inicialmente para aplicaciones de entretenimiento, videojuegos o sustitución de periféricos como el ratón o el joystick y por su bajo costo se ha convertido en el pionero en la categoría de productos hápticos para el mercado de consumo. Este dispositivo háptico es un esclavo de arquitectura paralela, con un espacio de trabajo de 4" x 4" x 4", capaz de soportar fuerzas mayores a 2 lbs y una resolución de posición mayor a 400 dpi (Novint, 2014).

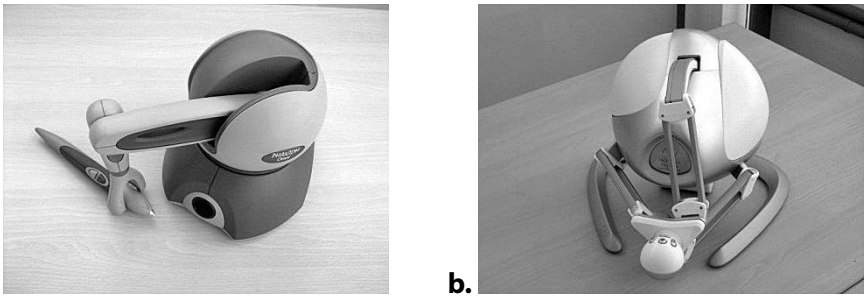


Figura 3. Interfaces hápticas usadas. a) Interfaz Phantom Omni®. *b) Interfaz* Novint Falcon™

El NOVINT HDAL SDK o Kit de desarrollo para el dispositivo Novint Falcon™, contiene toda la documentación y los archivos de software necesario para desarrollar aplicaciones con el dispositivo háptico desde la capa de abstracción HDAL (Haptic Device Abstraction Layer) con el lenguaje de programación C y C++, ya que la última versión contiene completa compatibilidad con C. Para su

uso, se asume que los controladores del Novint Falcon™ y del puerto UBS ya han sido instalados, y ya se han incluido los archivos DLLs, NOVINT*.BIN, y HDAL en los proyectos de aplicación. Los ejemplos presentados en el SDK se han desarrollado para trabajar en entorno de programación gráfica DirectX® y OpenGL®.

Con el uso de unos objetos de programación específicos, se generó una interfaz para la identificación, control de los dispositivos hápticos y su integración con el entorno de QGLView. Esto, a partir del reconocimiento de conexión, transformaciones del espacio de trabajo, lectura de vectores de posición (además de ángulos para el caso del Phantom Omni) y aplicación de señales de fuerza. Si no se encontraba conectada ninguna interfaz háptica, se simuló la presencia de un cursor háptico virtual con el movimiento de ratón despreciando o manteniendo constante el valor de posición en la coordenada z.

2.2 Metodología

Teniendo como variables el número de cubos por cada lado de la matriz y la escalización en los tres ejes coordenados, se plantearon pruebas para la medición de parámetros de ejecución según la librería utilizada (V-Collide o Physx). Los cubos componentes de la matriz se generaron a partir de la lectura de un archivo WaveFront *.obj exportado desde un programa de renderizado 3D, con mediciones unitarias en cada eje y organización triangular para sus caras (12 caras por cubo). El cursor háptico utilizado también se generó con la lectura de información de una esfera de radio unitario: vértices y caras obtenidas a partir de un archivo WaveFront *.obj.

La aplicación desarrollada eliminaba cualquier componente que entrara en contacto con el cursor háptico, de tal forma que una vez fuera detectado un contacto virtual entre la esfera y alguno de los cubos componentes de la matriz, el cubo colisionado era retirado de la escena virtual y de la matriz de objetos colisionables. Esto se hizo con el fin de generar funciones futuras para la simulación de maquinado en un modelo mixto a partir de la eliminación de partículas.

Se plantearon cuatro casos de estudio de acuerdo a la interfaz háptica utilizada y el uso de una librería específica para la detección de colisiones, con cada ejercicio definido como:

Falcon-VC: librería V-Collide y dispositivo Novint Flacon™

Phantom-VC: librería V-Collide y dispositivo Phantom Omni®

Falcon-Physx: librería physX y dispositivo Novint Flacon™

Phantom-Physx: librería physX y dispositivo Phantom Omni®

Para evaluar el comportamiento de la aplicación según el manejo de vértices con valores decimales, se consideraron las escalizaciones con el aumento o disminución del tamaño de cada cubo y por defecto de la matriz 3D con cubos de 0.5, 1, 1.5 y 2 unidades por lado. Cada caso fue probado 56 veces bajo un modelo determinístico, considerando cuatro casos de estudio según la interfaz háptica y la librería seleccionada, así como un barrido desde n=1 hasta n=14, siendo *n* el número total de cubos por cada lado de la matriz 3D. La cota superior impuesta a *n* se debió a que los tiempos de ejecución medidos con V-Collide fueron muy grandes para matrices con más de 14 cubos por lado.

Para el renderizado háptico, una vez detectadas las colisiones entre objetos se aplicó un vector de fuerzas definido por la ecuación 1:

$$\vec{F}_s = -Kx \qquad (1)$$

Con K determinada de acuerdo a parámetros de estabilidad háptica. Para la interfaz Phantom Omni, se definió un K=0.2 como constante de rea-

limentación de fuerza. La variable x correspondió a la diferencia entre la posición actual del cursor háptico y la posición anterior capturada por la detección de colisiones.

Las variables utilizadas para medir el desempeño de las librerías de detección de colisiones fueron: tiempo de inicialización requerido para la lectura y renderización de los componentes de la matriz 3D; tasa de refresco de imagen o cuadros por segundo con y sin reporte de colisión; y por último, consumo de memoria RAM una vez fueron cargados todos los objetos virtuales de la escena.

3. Resultados y discusión

3.1 Tiempo de Inicialización

El tiempo de inicialización es un factor importante debido a que permite medir la duración de reinicios de la aplicación, así como reconfiguraciones del modelo volumétrico de forma transparente. Se midieron los tiempos de inicialización en cada prueba cambiando la librería, la escala de la matriz 3d y el dispositivo háptico.

De las pruebas realizadas, independientemente de la escala de la matriz, se obtuvo un reporte similar entre el tiempo de inicialización para cada librería. En la figura 4 se presenta un caso de análisis con escala unitaria, en la que se aprecia una diferencia considerable entre el uso de la librería V-Collide y PhysX, siendo esta última la de mejor comportamiento para el cálculo y reporte de colisiones, independientemente de la cantidad de cubos por lado que se agreguen a la matriz y del dispositivo háptico utilizado. Sin embargo, las diferencias en tiempos de inicialización del simulador virtual, son altamente notorias cuando el volumen de información de los objetos virtuales crece, así, mientras que PhysX-Phantom se requieren cerca de 86ms para cargar una matriz de 2744 cubos, 14 por cada lado, usando V-Collide con el mismo dispositivo háptico el tiempo de inicio supera los 3 minutos, 191828ms, no mostrado en la gráfica debido a su crecimiento exponencial. En cuanto al tiempo de inicialización, la mejor opción para interactuar en un escenario virtual háptico consiste en calcular y reportar las colisiones con PhysX interactuando con la interfaz háptica Phantom Omni.

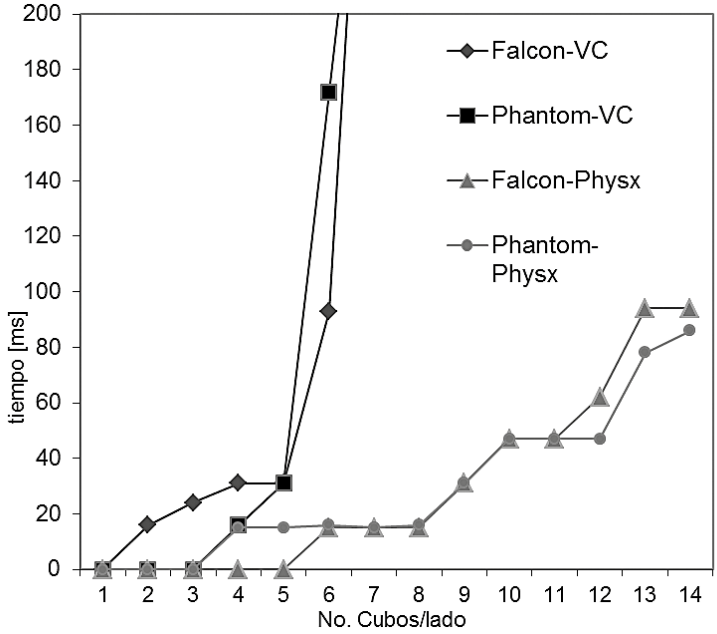


Figura 4. Tiempo inicialización de la aplicación, prueba sin escala.

3.2 Cuadros por segundo

Después de cargar los objetos virtuales de cada prueba en la escena, a través de un reporte en un archivo de texto, se leyeron las tasas de refresco de imagen, la cual se estableció en cuadros por segundo o FPS, por sus siglas en inglés. En cada caso se consideró el cálculo de FPS, con y sin reporte de colisión, con el fin de analizar problemas de retardo en la actualización gráfica cuando los componentes de la matriz entraban en contacto con el cursor háptico.

En la figura 5 se presenta un resumen de casos de estudio; en todas las pruebas realizadas, las mejores respuestas se dan al interactuar con el dispositivo Phantom Omni, detectando y reportando colisiones con PhysX ya que con esta librería se obtuvieron diferencias máximas de 10Hz aprox., al aumentar el tamaño de la matriz; mientras que con V-Collide se encontró una reducción en la tasa de refresco, cuando se interactúa con tamaños de cubos no enteros, lo cual no garantiza una ejecución continua de la aplicación gráfica, produciendo como consecuencia la pérdida de sincronización con el hilo háptico.

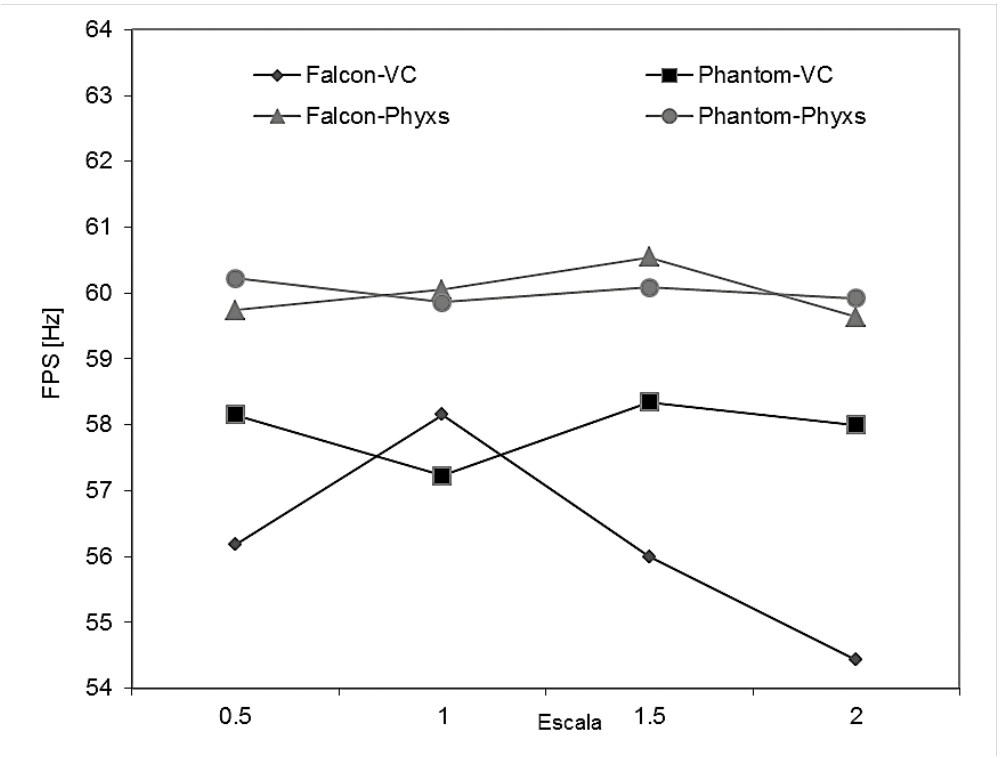


Figura 5. Comportamiento promedio de las tasas de refresco.

3.3 Consumo de memoria RAM

En cada una de las pruebas realizadas, una vez cargados los cubos componentes de la matriz, se tomaron datos del consumo de memoria RAM del sistema al interactuar con una interfaz háptica, procurando que todos los experimentos se realizaran bajo las condiciones similares del software al ejecutar los mismos programas en el sistema operativo. Se observó que el mínimo consumo de memoria RAM se dio con la aplicación de la librería V-Collide y el dispositivo háptico Novint Falcon según se aprecia en la figura 6, aunque las pruebas con la librería PhysX no registraron valo-

res de consumo críticos que pusieran en peligro la aplicación futura en un sistema de entrenamiento virtual, considerando las alternativas comerciales que actualmente se ofrecen para la memoria de datos en un sistema operativo. Con el aumento de los cubos no se presentaron variaciones bruscas en el uso de RAM, sino como era esperado, un consumo proporcional al volumen de información manejada en cada escena virtual.

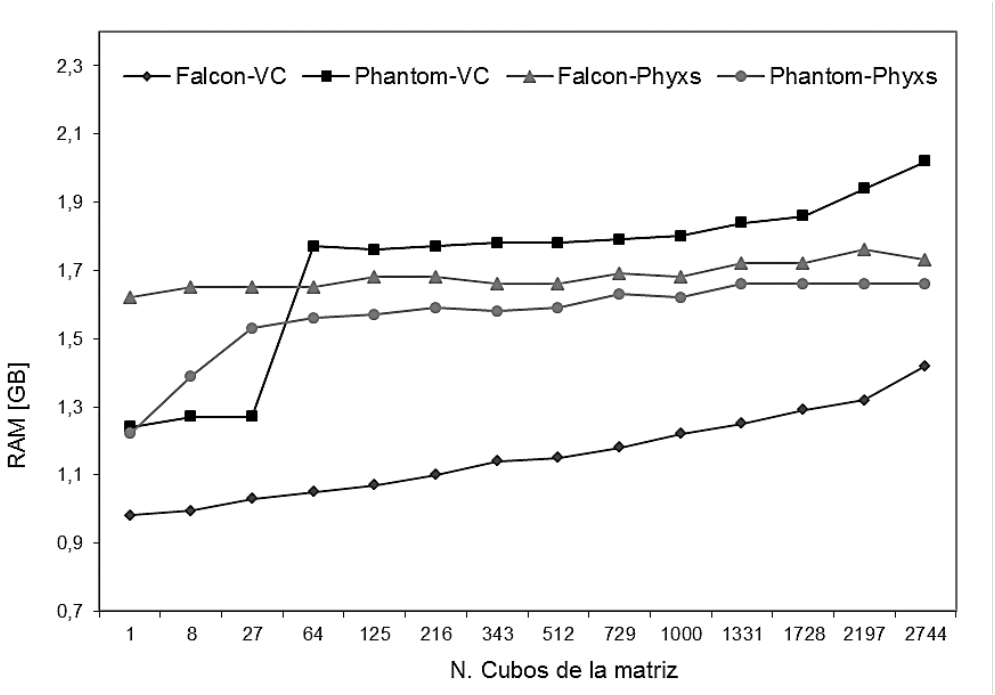


Figura 6. Consumo de memoria RAM en función del tamaño de objetos en la escena.

4. Conclusiones

Los resultados obtenidos argumentan la selección de la librería PhysX para el análisis en la detección de colisiones con un modelo mixto, mediante la comparación de desempeño con la librería V-Collide, en cuanto a tasas de refresco visual, tiempos de inicialización y consumo de memoria RAM.

De los casos de estudio, evaluando los parámetros de ejecución definidos, la mejor opción para interactuar en el escenario virtual de un sistema de teleoperación consiste en calcular y reportar las colisiones con PhysX interactuando con la interfaz háptica Phantom Omni. Esta selección permite además aprovechar la definición de propiedades físicas para la interacción de los objetos tridimensionales y la generación de efectos de contacto teniendo en cuenta velocidad, aceleración de cursor y otras características ya ofrecidas por la librería.

Agradecimientos

Este trabajo es resultado de los proyectos SGI 1299 y SGI 956, financiados por la Universidad Pedagógica y Tecnológica de Colombia. También se soporta en el proyecto: Modelado de tejidos blandos en un simulador háptico para entrenamiento de operaciones TKR y THR, financiado por el Banco Santander y desarrollado en la Universidad Nacional de Colombia. Los autores expresan su agradecimiento a José María Sabater, del Virtual Reality and Robotics Lab., Universidad Miguel Hernández de Elche, España.

Referencias

Chan, L., & Choi K. (2009). Integrating PhysX and OpenHaptics: Efficient force feedback generation using physics engine and haptic devices. Pervasive Computing (JCPC), 2009 Joint Conferences on. Publisher IEEE. 853 – 858. Doi: 10.1109/JCPC.2009.5420068.

Geforce. (2010). Technologies: PhysX. Extraído el 1 de octubre de 2010 de http://www.geforce.com/hardware/technology/physx

Maciel, A., Halic, T., Lu, Z., Nedel, L. P., & De, S. (2009), Using the PhysX engine for physics-based virtual surgery with force feedback. Int. J. Med. Robotics Comput. Assist. Surg., 5: 341–353. doi: 10.1002/rcs.266.

Moreno, E., & Rodríguez, S. (2002). Detección de colisiones. Un problema clave en la simulación quirúrgica. Especial: Tecnología de Simulación y Planificación.

Narber, C., & Duric, Z. (2010). Analysis of collision detection algorithms in haptic environments. Haptic Audio-Visual Environments and Games (HAVE), IEEE International Symposium on. 1-4. Doi: 10.1109/HAVE.2010.5623969.

Novint Technologies Inc. (2014). Falcon Technical Specifications. Extraído el 1 de febrero de 2014 de http://www.novint.com/index.php/novintxio/41

Ollero, A., García, A., & Gómez M. (2006).Teleoperación y Telerrobótica. PEARSON Prentice Hall, CEA Comité Español de Automática, Madrid, España.

Pinto, M., Sabater, J., Sofrony, J., Garcia, N., Azorín, J., & Pérez, C. (2011). Sistema de entrenamiento para operaciones de reemplazo total de rodilla. Revista TRAUMA. 22(3).

QGLViewer Features (2010). Extraído el 10 de septiembre de 2010 de http://www.libqglviewer.com/features.html

Ryden, F., Kosari S., & Chizeck, H. (2011). Proxy method for fast haptic rendering from time varying point clouds. IEEE/RSJ International Conference on Intelligent Robots and Systems. 2614 – 2619, doi: 10.1109/IROS.2011.6094673.

Ramírez, M. (2005). Estudio e implementación de un algoritmo de detección de colisiones basado en esferas. Cap. 2. PhD Thesis, Universidad de las Américas Puebla, Puebla, México. Maestría en Ciencias con Especialidad en Ingeniería en Sistemas Computacionales.

Reggiani, M., Caselli, S., & Mazolli, M. (2002). An experimental evaluation of colision detection packages for robot motion planning. Proceedings of the IEEE/RSJ Intl. Conference of Intelligent Robots and Systems EPFL.

Ruthenbeck, G. & Reynolds, J. (2013). Virtual reality surgical simulator software development tools. Journal of Simulation. 7, 101–108. doi:10.1057/jos.2012.22.

Sabater, J. M. (2003). Desarrollo de una Interfaz Kinestésica Paralela y Experimentación en Control de Sistemas Hápticos y Teleoperados. PhD thesis, Universidad Miguel Hernández, Elche, Alicante España.

V-COLLIDE (2013). Collision Detection for Arbitrary Polygonal Objects. Extraído el 8 de junio de 2013 de http://gamma.cs.unc.edu/V-COLLIDE/