UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SIMULACIÓN DE SENSACIÓN TÁCTIL EN UN SIMULADOR DE CIRUGÍA DE PRÓSTATA

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA

JORGE ROMMEL SANTIAGO ARCE

DIRECTOR DE TESIS
M.C. MIGUEL ÁNGEL PADILLA CASTAÑEDA

CO-DIRECTOR DE TESIS
DR. FERNANDO ARÁMBULA COSÍO

MÉXICO D.F. 2010

A
Diógenes Santiago
Beatriz Arce

Agradezco al grupo de Imágenes y Visualización del Centro de Ciencias Aplicadas y Desarrollo Tecnológico de la UNAM por haberme brindado la oportunidad de colaborar en sus proyectos. Al Dr. Jorge Márquez por su apoyo y valiosos consejos; al Dr. Fernando Arámbula por su tiempo, dedicación y compromiso en la revisión de este trabajo de Tesis; Al M en C. Miguel Ángel Padilla por haberme permitido participar en su investigación y abrir posibilidades de desarrollo personal y profesional.

Agradezco especialmente al Ingeniero Sergio Teodoro Vite por sus valiosas observaciones, recomendaciones y por la ayuda técnica que me brindó para la finalización de este trabajo.

Agradezco la DGAPA, con su programa de Fortalecimiento a la Docencia a través del Observatorio de Visualización de la UNAM, IXTLI, por brindarme los estímulos económicos necesarios para el desarrollo de mis actividades.

Agradezco a Maristmeña y a Tania, con quienes he compartido la emoción y el descubrimiento de crecer, por estar siempre ahí, en los momentos de tristeza y felicidad, los cuales, inefablemente nos llevan a ser mejores seres humanos.

Contenido

Introducción

Dentro de los sistemas de realidad virtual es posible desarrollar simuladores de cirugía, los cuales requieren interacción con el usuario y que el tiempo transcurrido entre el procesamiento de datos de entrada y la información de salida sea lo suficientemente pequeño para utilizarlos durante la simulación. En el campo de la medicina, los simuladores de cirugía aportan muchos beneficios tales como planeación de procedimientos quirúrgicos, asistencia médica, y entrenamiento de habilidades médicas para cirujanos, lo que permite un mejor conocimiento de los aspectos inherentes a cirugías particulares, disminuyendo el riesgo en la integridad y bienestar de los pacientes reales a los que el cirujano se enfrenta en su vida profesional.

Entre las características generales de un sistema de cirugía se encuentran aquellas en las que la salida del programa depende de las entradas que suministra el usuario, una de ellas es el tiempo en que se procesan los datos de entrada, el cual debe transcurrir lo suficientemente rápido para generar datos útiles al proceso. El realismo en el despliegue visual es deseable, lo que implica modelos gráficos de buena calidad generados por computadora o en su caso, procesados a partir de imágenes médicas, y se deben considerar señales auditivas que simulen el entorno del procedimiento de una cirugía real, así como la implementación de sensaciones táctiles para lograr una mayor sensación de inmersividad, es decir, crear un entorno en donde el usuario se sienta dentro del mismo, como si las interacciones que experimenta sean las reales.

Los primeros intereses en simuladores quirúrgicos se centraban en la navegación dentro de un entorno para visualizar un modelo gráfico que representaba una estructura anatómica, después, se desarrollaron algoritmos para la interacción de un modelo con una herramienta. Últimamente se ha tomado en cuenta la interacción con dispositivos capaces de generar sensaciones táctiles con el objetivo de incrementar el realismo en la simulación y así mejorar las habilidades de quien se entrena. Para ello, dichos dispositivos deben permitir un conjunto de movimientos deseados y los suficientes grados de libertad para lograr replicar o imitar la experiencia de tocar, manipular o percibir objetos dentro de un ambiente virtual o teleoperado [5].

Un problema con la simulación de sensaciones táctiles consiste en que los dispositivos comerciales que soportan un despliegue de fuerzas están enfocados en aplicaciones específicas como es el caso de las palancas de mando para videojuegos, por lo que, se han modificado para el desarrollo de otras aplicaciones como la teleoperación e interfaces de propósito general.

El sistema de simulación de cirugía de próstata, actualmente en desarrollo por parte del personal adscrito al Laboratorio de Imágenes y Visualización del Centro de Ciencias Aplicadas y Desarrollo Tecnológico, CCADET, de la UNAM, está compuesto por un módulo mecatrónico y por un módulo de *software* desarrollado con el IDE[1] de DevC++ [3]. El problema central para adaptar un dispositivo de realimentación de fuerzas comercial al sistema de simulación se enfoca en que las aplicaciones para aquel se programan en entornos distintos al de DevC++, tal es el caso de los productos de la compañía Sensable Technologies[25], los cuales utilizan el entorno de desarrollo de Visual Studio [17] y en consecuencia los procesos generados con dichos entornos son incompatibles por los recursos que utiliza cada uno, lo que imposibilita tener sensaciones táctiles en la simulación.

En este trabajo se propone una solución para el problema de la comunicación de procesos con recursos no compatibles, la cual consiste en crear las funciones que permitirán interactuar con la información que proporciona un dispositivo háptico, dentro del programa del sistema de simulación, y construir una aplicación que fungirá como servidor de datos, los cuales se utilizarán en el programa ejecutable del simulador de cirugía – el cliente – e interactuará para enviar de regreso los datos de despliegue de fuerza; para ello es preciso utilizar la funcionalidad del protocolo TCP/IP que hará posible la comunicación entre dichos procesos con base en una arquitectura cliente – servidor. Es necesaria también la utilización de la programación concurrente para procesar el ciclo de flujo del dispositivo háptico y de la conexión por medio de la apertura de canales de comunicación en el proceso servidor y en el proceso cliente.

Considerando lo anterior, el objetivo principal de este trabajo es el diseño de una interfaz (*software)* que permita la adaptación de un dispositivo háptico comercial –

[1] Entorno Integrado de Desarrollo (*Integrated Development Environment*).

interfaz mecatrónica basada en un sistema de servomotores controlados por software que transmite una fuerza resultante hacia un usuario – al Simulador de Cirugía de Próstata del CCADET. Un segundo objetivo, derivado del anterior es proveer sensaciones táctiles durante la simulación de tal manera que se puedan percibir como si se tocaran objetos reales.

El primer capítulo presenta en forma general el campo de estudio de la Cirugía Asistida por Computadora, trata de los simuladores virtuales aplicados a los procedimientos quirúrgicos, muestra algunos simuladores virtuales existentes así como la descripción detallada del Simulador de Cirugía de Próstata del CCADET y su aplicación principal.

El capítulo dos contiene la definición de Computación Háptica, área de estudio necesaria para desarrollar e implementar sensaciones táctiles. Después recurre al concepto de fuerza, una clasificación general en el ámbito de la sensación táctil y su relación con la rigidez, variable física que se utiliza en el despliegue de fuerzas por medio de un dispositivo háptico. Al final se describen las características del paquete de desarrollo de OpenHaptics para el dispositivo comercial Omni[2].

El capítulo tres aborda los conceptos relacionados con la programación concurrente, el protocolo de comunicación TCP/IP, muestra su uso para lograr la comunicación entre procesos no compatibles para conseguir la adaptación del dispositivo de realimentación de fuerzas y tener sensaciones táctiles durante la simulación quirúrgica.

En el capítulo cuatro se describe el sistema integrado con el dispositivo háptico como producto funcional y se presenta un análisis de los resultados obtenidos.

Por último, se presentan las conclusiones derivadas de este trabajo de tesis y se sugieren las actividades que pueden desarrollarse a futuro, a partir de esta investigación.

[2] Fabricado por SensAble Technologies, Inc..

Capítulo 1. Simuladores de cirugía

El campo de estudio de la Computación Gráfica es muy amplio. Diferentes disciplinas del conocimiento humano se combinan con técnicas de cómputo para producir aplicaciones que resuelven parte de sus problemas. En Medicina se logran métodos computacionales para modelar y estudiar propiedades físicas, diseñar modelos anatómicos, planear y ejecutar cirugías, diagnosticar, practicar cirugía a distancia, así como la educación y entrenamiento; que en forma genérica se les conoce como Cirugía Asistida por Computadora, CAS (Computer Asisted Surgery) [9].

El Diccionario de la Lengua Española define, en el ámbito de la tecnología, a un simulador como: “Aparato que reproduce el comportamiento de un sistema en determinadas condiciones, aplicado generalmente para el entrenamiento de quienes deben manejar dicho sistema” [22].

Los simuladores de cirugía son básicamente sistemas asistidos por computadora, desarrollados bajo esquemas de realidad virtual pura, o bien, en combinación con instrumentación y maniquíes, que representan la escena de un procedimiento quirúrgico en particular, con interacción en tiempo real, calidad gráfica de visualización, realimentación de fuerzas, y sonidos ambientales interactivos. Teodoro [33] describe varios casos de simuladores de cirugía y sistemas de realidad virtual aplicados a la medicina.

Delinguete [5] menciona tres generaciones de simuladores. La primera generación se enfocaba principalmente en la navegación tridimensional dentro de un sistema de visualización con base en un conjunto de datos anatómicos. La segunda generación se preocupó por modelar las respuestas físicas en una estructura anatómica, representada por modelos gráficos; en este aspecto, se interesaron por la cinemática ósea y por la deformación muscular de tejido humano como consecuencia de la interacción con una herramienta quirúrgica. La tercera generación tomaba en cuenta la naturaleza funcional de los órganos del cuerpo humano y su comportamiento en relación con otros órganos.

Para lograr una simulación avanzada, es importante considerar los distintos niveles en los que se desarrolla un procedimiento quirúrgico: los niveles geométrico, físico y fisiológico; así como la arquitectura básica de un sistema de simulación, la cual contempla tres módulos:

- Módulo de gráficos encargada de los cálculos de deformaciones.
- Módulo de cómputo de colisiones y de vectores de fuerza.
- Dispositivo de realimentación de fuerzas, el cual provee las posiciones de una herramienta quirúrgica y permite el despliegue de la resultante de fuerza correspondiente en ese instante de tiempo.

El sistema de simulación URO Mentor (véase figura 1), desarrollado por la compañía Simbionix [28], se enfoca en el entrenamiento para la endourología[1]. Este simulador cuenta con un módulo para la identificación anatómica y para el entrenamiento en la visualización. Provee una serie de herramientas para los procedimientos de esta cirugía en una variedad de casos con una anatomía y padecimientos representativos. Proporciona la simulación de cistoscopia[2] de ureteroscopia[3] y soporta la visualización de agentes de contraste que facilitan la navegación. No simula la resección transuretral de próstata.

Figura 1. *Simulador de endourología* [27].

[1] Conjunto de técnicas de mínima invasión, diagnósticas o terapéuticas, realizadas en el sistema genitourinario a través de los orificios naturales.
[2] Examen del interior de la vejiga urinaria.
[3] Examen del interior de la uretra.

1.1. Simuladores de cirugía de próstata.

La próstata es un órgano glandular del sistema genito – urinario masculino que se localiza en la profundidad de la pelvis, fija entre el pubis por delante, la vejiga por arriba, el recto por detrás y el piso pélvico por debajo (véase figura 2). A partir de los treinta años, el hombre experimenta el crecimiento gradual de dicho órgano hasta que produce trastornos en edades avanzadas. Uno de los padecimientos comunes, alrededor de los cincuenta años es la Hiperplasia Benigna de Próstata, que produce obstrucción e infección de las vías urinarias [1].

.

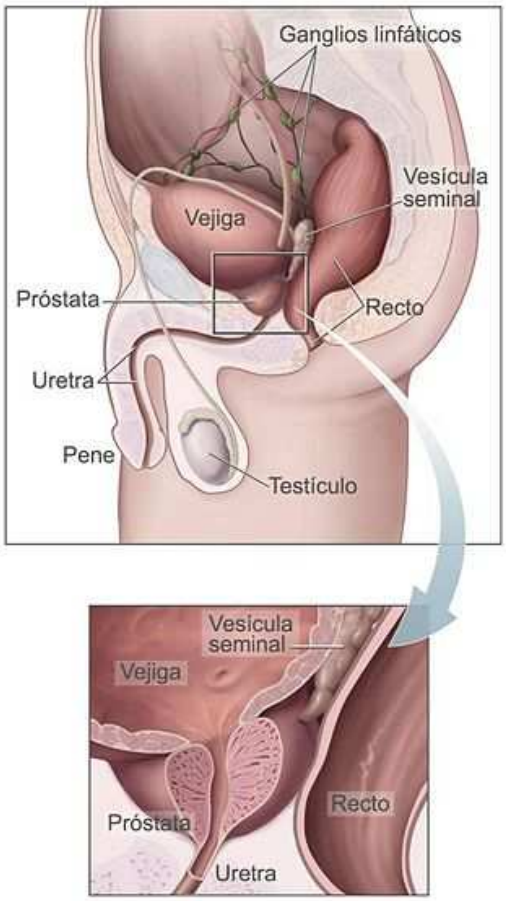

Figura 2. *Esquema de la próstata y su ubicación* [6].

El objetivo del tratamiento de la Hiperplasia Benigna de Próstata es la de “aliviar los síntomas, mejorar la calidad de vida y evitar la aparición de complicaciones” [1]. El procedimiento quirúrgico, considerado como estándar, para tratar este padecimiento es la resección transuretral de próstata, RTUP, que consiste en la “extirpación de tejido adenomatoso mediante su resección endouretral. En la RTU convencional se utiliza un

generador de corriente monopoloar, corrientes de alta frecuencia que emiten corrientes de corte puro y de electrocoagulación" [1], además de un instrumento llamado resectoscopio o resector para realizar los cortes de tejido y la coagulación de vasos rotos. En la figura 3 se muestra un resectoscopio de la firma Olympus. Cuenta con un lente por el cual el médico especialista observa el desarrollo de la cirugía. El cuerpo está formado por un cilindro llamado camisa, que se introduce vía uretra y que lleva en su interior el asa con la cual se realiza el corte y la coagulación de tejido crecido. Tiene también un sistema de drenado el cual facilita la intervención.

Figura 3. *Resectoscopio OES-Pro* [19].

Reyes [23] argumenta que el proceso de enseñanza de la RTUP implica una residencia en la especialidad de urología durante cuatro años, de acuerdo con los lineamientos del Sistema Nacional de Salud [23]. Dadas las dificultades que el aprendizaje de RTUP conlleva, se han desarrollado sistemas de simulación para el entrenamiento de las habilidades involucradas.

Para Robert Sweet [29], los modelos de simulación de entrenamiento de Resección Transuretral de Próstata son de gran importancia debido a que reducen considerablemente los costos de cirugía y los riesgos, entre los cuales menciona: la incontinencia urinaria, las heridas rectales, las heridas uretrales, las heridas en venas con pérdida de sangre, la disfunción eréctil, entre otros. "El procedimiento involucra la habilidad de trabajar en un pequeño espacio tridimensional mientras se recibe retroalimentación visual bidimensional, se requiere que el operador desarrolle habilidades visuales espaciales y motoras, manipular la herramienta, el asa de corte continua y simultáneamente mientras se administra la corriente eléctrica por medio de los pedales"[29].

El primer simulador de realidad virtual para la RTUP fue descrito por Lardennois en 1990 [29]. A partir de ahí se han desarrollado otros simuladores para entrenamiento de RTUP, como es el caso de “UW Virtual TURP Surgical Simulator” [32] desarrollado por el Department of Urology and Human Interface Technology Lab (HITL) de la Universidad de Washington, junto con el CREST (Center for Research in Education and Simulation Technologies) de la Universidad de Minnesota, el cual incluye los módulos de Identificación Anatómica, Corte, Coagulación, y Resección Avanzada, así como la retroalimentación de fuerzas mediante un sistema háptico (véase figura 4).

El módulo de identificación anatómica devuelve etiquetas verbales como respuesta a la interacción del usuario cuando toca alguna parte de los modelos gráficos. Esta parte prepara la RTUP mediante la simulación de una cistoscopia.

El módulo de Corte utiliza un pedal, el cual habilita el corte sobre la malla del modelo gráfico. Una variable de importancia es el tiempo en el cual se realizan los cortes.

El módulo de coagulación permite visualizar el sangrado dentro de la próstata con el fin de que el usuario accione un pedal de coagulación para cauterizar la herida. El tiempo es también importante ya que la escena se llena de sangre si no se cauterizan los vasos sanguíneos rápidamente.

El módulo de resección avanzada contempla el procedimiento quirúrgico completo para la resección transuretral de próstata.

Otro simulador de cirugía para tratar la Hiperplasia Benigna de Próstata fue desarrollado por el laboratorio MIMLab del Imperial College London [11]. Se trata de un sistema híbrido conformado por un módulo de realidad virtual, por un *phantom* o maniquí de material parecido al tejido humano de la próstata, el cual provee realimentación táctil, y por un registro para evaluar el desempeño del entrenamiento (véase figura 5). Este sistema cuenta con dos modalidades, la primera consiste en determinar el modelo gráfico de la próstata del paciente por medio de imagenología para logar un entrenamiento preoperatorio personalizado. La segunda realiza simplificaciones para entrenamiento general [7].

Figura 4. Vistas de las *etapas del simulador de cirugía de próstata desarrollado por la Universidad de Washington* [32].

Figura 5. A la izquierda: *Ejecución del entrenamiento*. A la derecha: *Phantom usado en el sistema de simulación* [11].

1.2. El simulador RTUP del Laboratorio de Imágenes y Visualización CCADET – UNAM.

El Simulador de Cirugía de Próstata desarrollado por el grupo del Laboratorio de Imágenes y Visualización del Centro de Ciencias Aplicadas y Desarrollo Tecnológico de la UNAM, es un "conjunto de bloques funcionales que tienen como objetivo la simulación de condiciones reales en torno al procedimiento quirúrgico, haciendo una

abstracción de los elementos del mundo real dentro de un entrono virtual" [33], y cubre en su primera etapa una interfaz mecatrónica y un ambiente virtual.

El primer bloque consiste en un dispositivo mecatrónico que simula el instrumento de cirugía llamado resectoscopio y un protocolo de comunicación serial universal, USB (por sus siglas en inglés), entre dicho instrumento y el ambiente virtual. Su diseño actual contempla un juego de discos que permite los movimientos de giro sobre los tres ejes en un sistema de coordenadas cartesiano derecho y permite los grados de libertad involucrados. En la figura 6 se muestra el sistema mecatrónico, que en adelante se mencionará como la "Interfaz Mecatrónica".

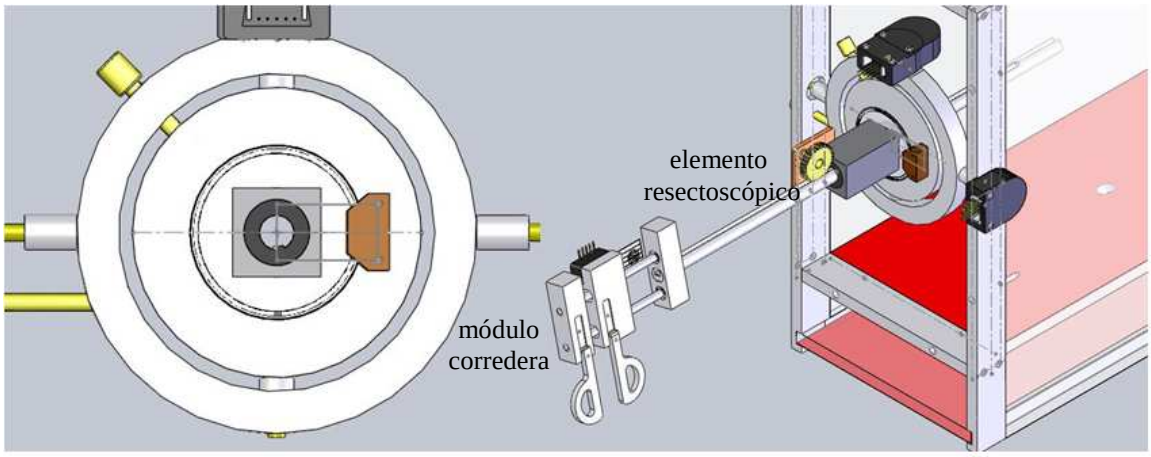

Figura 6. *A la izquierda: discos limitadores de movimiento. A la derecha: vista de la ineterfaz mecatrónica. El elemento resectoscópico simula la camisa, el módulo corredera hace las veces de asa de resección* [23].

El desplazamiento sobre el eje z es implementado por medio de una varilla que simula la camisa de la herramienta de resección y que cuenta en su extremo manipulable con un mecanismo que hace las veces del asa de corte del resectoscopio. Las lecturas de los datos necesarios para el movimiento implicado en el proceso quirúrgico se llevan a cabo mediante un sistema electrónico con sensores ópticos [23] y una tarjeta de adquisición de datos con comunicación serial universal, (véase figura 7). Esta información se envía al programa de simulación para tratarlos como coordenadas virtuales de posiciones de la herramienta. Es decir, para realizar un mapeo de coordenadas reales con las coordenadas del entorno gráfico.

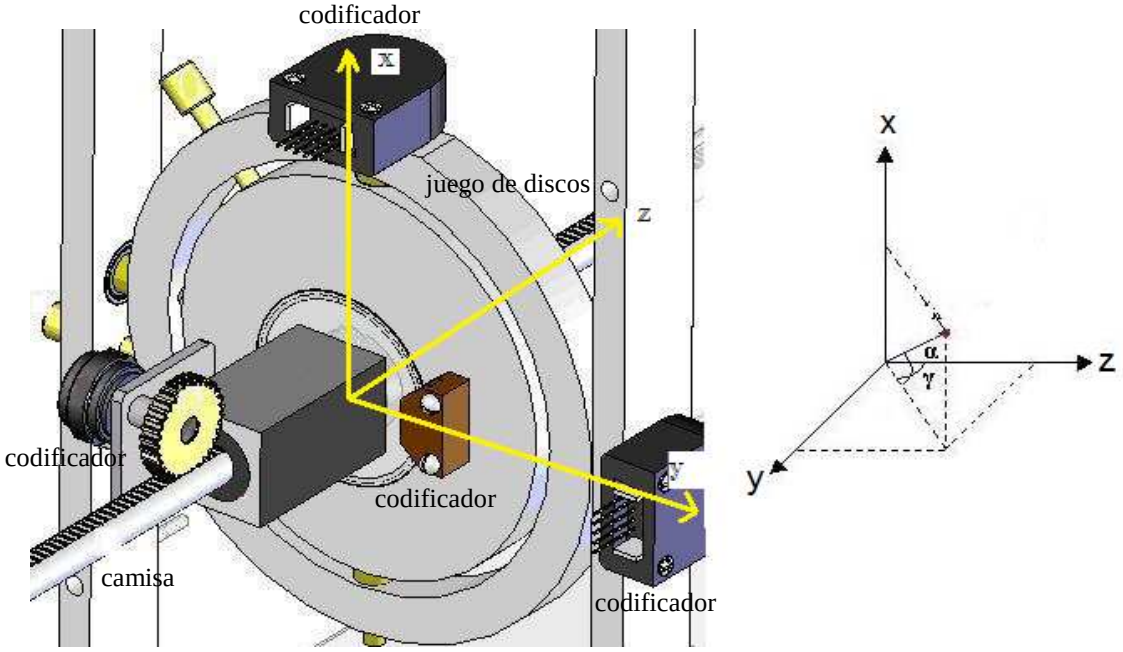

Figura 7. *Sistema de coordenadas del mecanismo* [23].

El segundo bloque se basa en un sistema de gráficos tridimensionales construido con la librería abierta de gráficos, OpenGL [20], y cuenta con un módulo de cálculos usados en algoritmos de deformación de mallas y algoritmos de detección de colisiones. Los modelos anatómicos están construídos a partir de imágenes médicas con algoritmos desarrollados en el Laboratorio de Imágenes y Visualización. En adelante, se hará referencia a este módulo como “Programa de Simulación”.

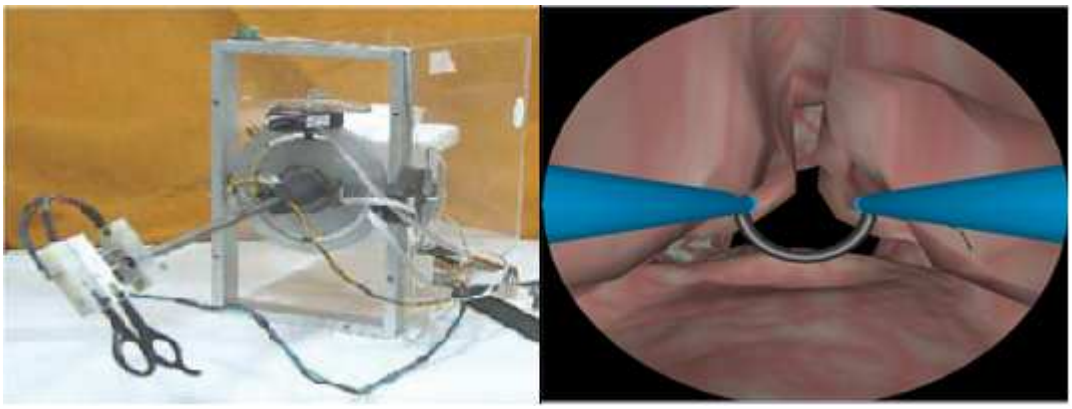

Figura 8. *Derecha: Interfaz mecatrónica. Izquierda: vista del despliegue gráfico.* Grupo de Micromecánica y Mecatrónica - Laboratorio de Imágenes y Visualización, CCADET, UNAM 2010.

El sistema de referencia que ocupa el Programa de Simulación es un sistema cartesiano derecho tridimensional. Se hace una correspondencia con los grados de libertad de la

Interfaz Mecatrónica para una correcta sincronía entre los movimientos en el espacio real y los gráficos en el espacio virtual.

El sistema completo, en adelante Simulador, interactúa con el usuario mediante la siguiente información:

- Desplazamiento absoluto. Corresponde con el movimiento sobre el eje z de la camisa o resectoscopio completo. En el ambiente virtual este movimiento simula el avance de la cámara a través de los modelos anatómicos.
- Desplazamiento relativo. Es el movimiento que realiza el asa de corte y depende de la posición de la camisa. Durante la simulación se visualiza este modelo gráfico.
- Ángulos sobre los ejes cartesianos. Movimientos para mover la camisa junto con el asa. En la navegación se visualiza como movimiento de la cámara situada al final del resectoscopio.
- Banderas para respuestas a colisiones. Estas banderas, dependiendo de su estado, habilitan el corte, la coagulación o bien la exploración de los modelos anatómicos.

En una segunda etapa, actualmente en desarrollo se trabaja en el mejoramiento de la calidad de visualización, adaptación de sonidos ambientales y de interacción, y la implementación de retroalimentación de fuerzas. En este sentido, Teodoro [33] sostiene: “Otra alternativa, actualmente en fase de desarrollo, para la representación del efecto de retroalimentación de fuerzas y el resectoscopio real, consiste en el uso de un dispositivo háptico”

Justificación.

La importancia de integrar un dispositivo háptico al Simulador es obtener un mayor realismo en la simulación del procedimiento quirúrgico en cuestión, ya que provee estimulación táctil, con lo que se logra un mejor entrenamiento para el usuario.

Delinguete [5] sostiene que existen al menos dos propósitos en la retroalimentación de fuerzas: el que desarrolla la cinestesia[4], la cual provee sensación de movimiento en el usuario; y el cognitivo, cuyo uso ayuda a distinguir texturas al examinar propiedades mecánicas; incrementando de esta manera las habilidades del usuario.

Así pues, en los capítulos siguientes se explican los conceptos necesarios para presentar un sistema que provea sensaciones táctiles debido al interés de contar con un Simulador lo más completo y realista posible que ayude al entrenamiento de la resección transuretral de próstata y se muestra el método que se utilizó para resolver los contratiempos que se presentaron durante la programación y el desarrollo de la integración del sistema.

[4] La Cinestesia es la percepción del equilibrio y de la posición de las partes del cuerpo.

Capítulo 2. Computación háptica

Háptico – háptica es un término que se refiere a la recepción de información por parte del cerebro a través del sentido del tacto. En la literatura relacionada se le conoce como *haptics*; Hannaford y Okamura [8] mencionan que la raíz es griega, de *haptestai*, lo relativo al sentido del tacto; Klatzky y Lederman [13] lo implican con la habilidad de sostener algo y manejan un sinónimo utilizado en el ámbito de la medicina, *stereognosis*.

El concepto se extiende hacia la manipulación por medio de interacciones en ambientes reales, virtuales o tele operados; ya sea mediante el ser humano, máquinas, o combinación de ambos, mediante dispositivos mecatrónicos que hacen posible la estimulación táctil:

- Háptica Humana. Es el estudio de la percepción y la manipulación a través del tacto mediante la cinestesia y los receptores cutáneos, que dan las sensaciones de fuerza, posición y contacto.
- Háptica Mecánica. Se basa en el diseño, construcción y uso de máquinas para reemplazar o bien, mejorar la manipulación por el sentido del tacto del ser humano.
- Computación Háptica. Conjunto de algoritmos y *software* asociado con la generación y despliegue de sensaciones táctiles en objetos virtuales.

Las disciplinas que contribuyen al desarrollo de esta área son la Biomecánica, Neurociencia, Psicofísica, Diseño y Control en Robótica, Modelado y Simulación, Matemáticas e Ingeniería de Software; y se puede aplicar a necesidades humanas como diseño de productos, entrenadores y simuladores de procedimientos quirúrgicos y rehabilitación; así como en el entretenimiento, educación, industria, milicia, tecnología asistida para discapacidad visual, videojuegos y arte, entre otras.

Se puede describir a la Computación Háptica como un área de investigación en técnicas y procesos asociados con la generación y despliegue de sensaciones y manipulaciones de objetos virtuales hacia un operador humano a través de un dispositivo transmisor de

fuerzas, es decir, que genera vectores en dirección y magnitud determinados o calculados previamente. Se maneja también, el término de imagen háptica o imagen táctil como sinónimo, de tal manera que puede considerarse también como dispositivo de despliegue de imágenes hápticas. Aun más, al tratarse de una imagen, es posible realizar un *render* de fuerzas. En el área de estudio del cómputo gráfico esto significa tomar la descripción geométrica de un objeto tridimensional o bidimensional para generar un objeto visible en un monitor [34]; haciendo una analogía, se habla de *render* háptico o *render* de fuerzas para contar con una descripción de la imagen táctil a partir de su geometría en el espacio.

2.1 Sistemas hápticos en CAS.

Las principales aplicaciones de la computación háptica a la medicina asistida por computadora se centran en la incorporación de sensaciones táctiles a sistemas robóticos para intervenciones quirúrgicas de mínima invasión para lograr la disminución de: fuerzas de contacto del cirujano, consumo de energía, tiempo de ejecución de la intervención y número de errores, entre otros aspectos.

En cuanto a sistemas de simulación, hay una amplia variedad de sistemas que se desarrollaron, en un principio, como ambientes de realidad virtual para entrenamiento y que después se mejoraron para asistir operaciones por medio de robots con efectores finales. Tal es el caso del Laparoscopic Impulse Engine, desarrollado por Immersion Corporation [21] cuyo dispositivo de realimentación de fuerzas puede rastrear posiciones con cinco grados de libertad y transmitir fuerzas con tres grados de libertad. Actualmente este motor es parte de una estación de trabajo de cirugía capaz de proveer cinco grados de libertad para despliegue de sensaciones hápticas [30].

Otros casos de simuladores de cirugía para entrenamiento que se mencionan en [30] han adaptado o modificado dispositivos hápticos comerciales para simular sensaciones táctiles en sus ambientes virtuales.

En el caso de la rehabilitación médica existen muchos sistemas de realidad virtual con realimentación de fuerzas como el desarrollado por la Johns Hopkins University para rehabilitación de personas con algún grado de daño cerebral [12].

2.2 Dispositivos Hápticos

Los sistemas de Realidad Virtual que soportan estimulación visual y auditiva para el usuario, proveen buena capacidad de interactuar con el mismo. El hecho de tocar, sentir y manipular objetos, acerca al usuario a sensaciones más reales. Esto se logra mediante interfaces hápticas, las cuales son implementadas por dispositivos físicos con sistemas de servo motores que permiten la interacción con el ser humano. Dichos dispositivos son capaces de generar imágenes táctiles como resultado de manipulaciones en el mundo virtual.

Las interfaces hápticas consisten en la experimentación de tocar, manipular y percibir un ambiente real a través de dispositivos mecánicos y el control mediante computación.

Los dispositivos hápticos deben ser capaces de proveer los movimientos que interesen al operador humano, esto supone un dispositivo que permita retroalimentación y los grados de libertad de movimiento necesarios para ello.

Los algoritmos de render de fuerzas se dividen en detección de colisiones y en respuestas a ellas, con base en posiciones y orientaciones referenciadas a un punto particular del dispositivo háptico, este punto es caracterizado por un efector final, parte del dispositivo háptico que el usuario manipula y sobre el cual se despliegan las resultantes como salida del sistema. Los vectores de fuerza, resultados de cálculos específicos son retroalimentados a través del efector final hacia el usuario. La secuencia de detección de colisiones y respuestas se ejecuta dentro de un ciclo a una frecuencia de un kilohertz aproximadamente para que se genere la sensación táctil en el cerebro del usuario y no se perciban vibraciones al interactuar con los objetos.

Un concepto de suma importancia al programar dispositivos hápticos es el ciclo principal, o *servo loop*. Se trata de una estructura de control de flujo de alta prioridad que se utiliza para el cálculo de fuerzas que se enviarán al dispositivo háptico. Como se mencionó anteriormente, para mantener un despliegue de fuerza coherente, es necesario mantener en ejecución este ciclo a una tasa de 1 kHz; para mantener la superioridad jerárquica de este ciclo, se implementa generalmente dentro de una estructura de control de flujo concurrente llamado hilo, y es éste precisamente el *servo loop* [26].

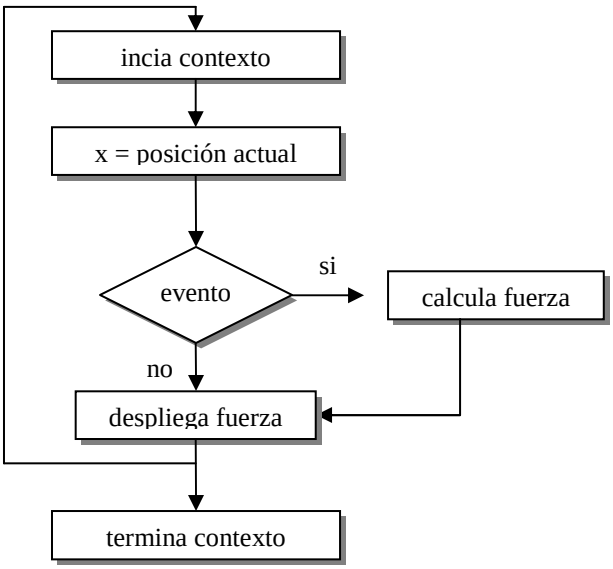

Figura 9. *Diagrama de flujo del servo loop*.

En la figura 9 se muestra un diagrama de flujo del *servo loop*. Inicia dentro de un contexto que define el alcance en el cual el estado del dispositivo háptico garantiza consistencia. Antes de cerrar el marco se deben completar las tareas: leer la posición actual, si hubo alguna interacción se calcula la fuerza y se despliega, en caso contrario sigue el flujo de control y se despliega la fuerza con valor nulo.

De acuerdo con Hannaford y Okamura [8] el *servo loop* consiste en tres pasos que se listan a continuación:

1. El dispositivo háptico reconoce la entrada de datos por medio de un operador o usuario, la cual puede ser la posición detectada sobre un efector final, una fuerza, o bien una actividad muscular.
2. La entrada es aplicada a un ambiente virtual o a un ambiente teleoperado.
3. La respuesta es transmitida mediante el efector final y visualizada en un monitor.

En el caso de un ambiente virtual, los datos recibidos por el dispositivo se procesan para obtener una serie de datos de interés para el operador, el cual obtiene como resultado del sistema un despliegue de fuerza.

Actualmente existen dispositivos hápticos comerciales. Las compañías principales, SensAble Technologies [25] e Immersion Corporation, ofrecen una variedad de productos con características específicas para diferentes aplicaciones. En la figura 10 se muestra el modelo Omni de SensAble Technologies, dispositivo con que cuenta el laboratorio de Imágenes y Visualización del CCADET de la UNAM, mismo que se utilizó para la simulación de sensaciones táctiles en el Programa de Simulación.

Figura 10. *Dispositivo Háptico Modelo Omni* [25].

Algunas de las características de este dispositivo háptico se muestran a continuación:

Especificación		*Sensable Technologies* PHANTOM Omni Developer Kit
Peso (kg)	1.8	
Grados de libertad (in)	6DOF	
Grados de libertad (out)	3DOF (x, y, z)	
Workspace (mm)	160 w x 120 h x 70 d	6.4 w x 4.8 h x 2.8 d in
Resolución 3D (mm)	0.009	
Rigidez (N/mm)	1.26x2.31x1.02 N/mm , 7.3 x 13.4 x 5.9 lbs/in	
Fuerza máxima (N)	3.3 N , 0.75 lbf	
Fuerza contínua (N) máxima (24 h)	0.88 N , 0.2 lbf	

Tabla 1. *Especificaciones generales del modelo Omni* [25].

El dispositivo Omni tiene las dimensiones que se muestran en la figura 11, y el sistema de referencia que se muestra en la figura 12.

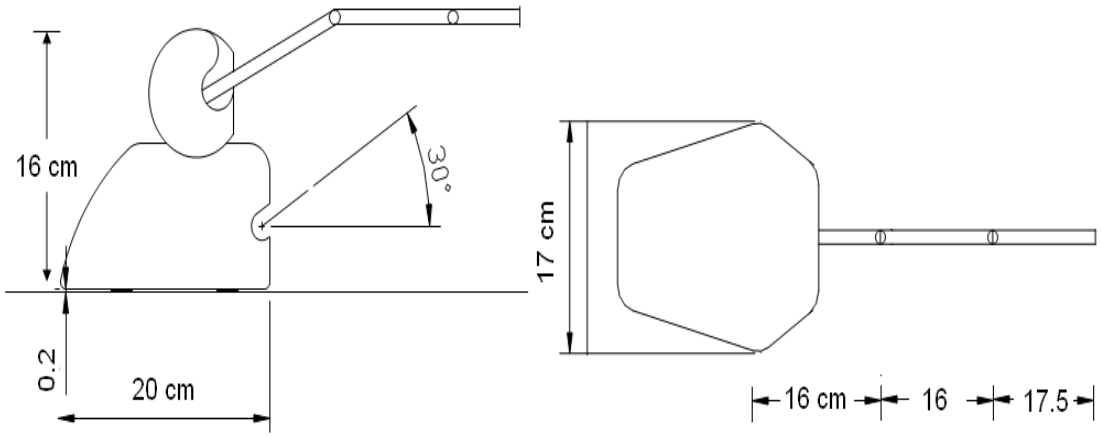

Figura 11. *Dimensiones del dispositivo háptico Omni*.

El origen del sistema de referencia del dispositivo Omni es un punto fijo dentro de su espacio de trabajo. En este modelo no es posible cambiar este punto por medio de las funciones programables del software de control. En modelos más sofisticados de la misma compañía si es posible hacerlo.

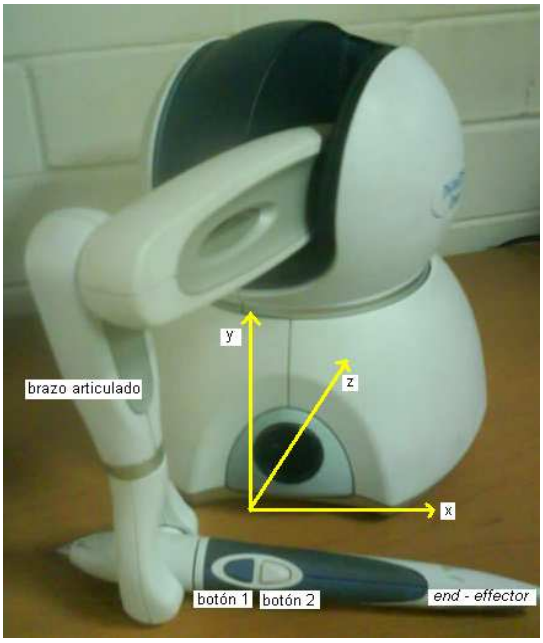

Figura 12. *Sistema de referencia del Omni*. Laboratorio de Imágenes y Visualización, CCADET, UNAM 2010.

2.3 Fuerza.

Los sistemas de realidad virtual tienen como principal reto proveer información hacia el cerebro humano para que éste lo interprete y se experimente una sensación de inmersividad lo más cercana a la realidad; para ello se debe buscar la manera de estimular a los sentidos mediante dispositivos electrónicos con características particulares. En el caso del sentido del tacto es complicado diseñar sensores táctiles que puedan estimular la piel y en general el sistema nervioso ya que es necesaria la interacción por medio de mecanismos físicos, los cuales deben adaptarse a alguna de las tres actividades táctiles básicas en los seres humanos: manipulación, exploración y respuesta (véase figura 13); relacionadas con tareas motoras finas, información de la superficie de materiales e impactos, respectivamente [4]. Aun más, estos mecanismos deben ser capaces de transmitir las respuestas obtenidas por medio de las interacciones antes descritas hacia el sistema psicomotriz del usuario para que sean percibidas como fuerzas y se tenga la sensación de que se ha tocado un objeto.

Debido a que el interés de adaptar el dispositivo háptico al simulador de cirugía de próstata es la sensación de tocar tejido, es necesario contar con un conjunto de fuerzas que estimule al sentido del tacto del operador en cada punto en donde la escena de la cirugía detecte colisión con la malla que representa la estructura anatómica, y que será desplegado por el Omni.

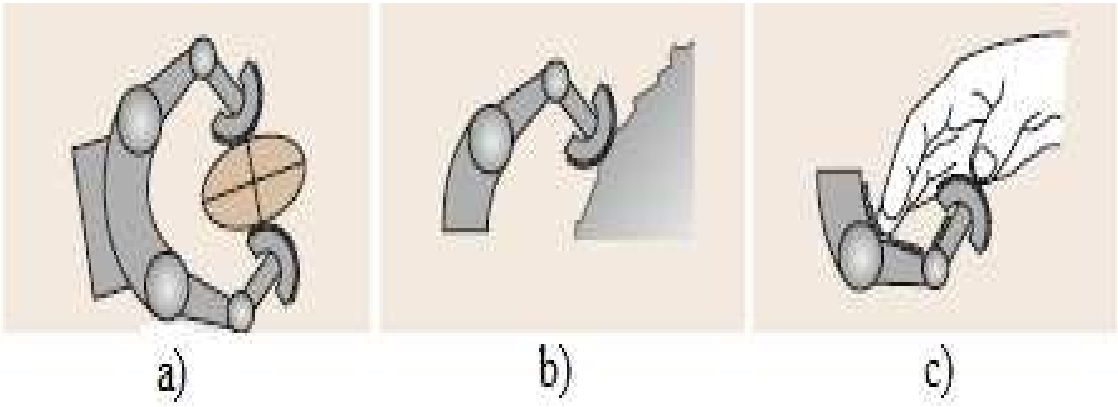

Figura 13. *Actividades táctiles básicas. A la izquierda: tareas motoras finas. En medio: exploración. A la derecha: impactos suaves* [4].

2.3.1 Conceptos generales

La fuerza, de acuerdo con la segunda ley de Newton, es:

$$\overline{F} = m\overline{a}$$

en donde,

$\overline{F}$ es el vector fuerza

m es la masa

$\overline{a}$ es el vector aceleración.

Esta ecuación indica que un cuerpo en estado de reposo o movimiento constante tiene un cambio en su estado inicial causada por una interacción externa a la cual se le conoce como fuerza.

2.3.2 Tipos de Fuerza (movimiento y tiempo)

La ejecución de un programa con un sistema de realimentación de imágenes táctiles puede considerarse como un *render* que produce un vector resultante de fuerza que se despliega en un dispositivo háptico. Para ello es necesario considerar los distintos tipos de fuerza que se pueden manejar, sabiendo, de antemano, que en un sentido estricto existen cuatro fuerzas fundamentales, la gravitacional, la electromagnética, la nuclear débil y la nuclear fuerte [27], sin embargo su estudio excede los límites de esta investigación. La unidad de salida para un dispositivo háptico es un vector de fuerza. Los tipos de fuerza que se pueden implementar en un dispositivo háptico son:

- Fuerza dependiente del movimiento: resortes, amortiguadores, fricción e inercia.
- Fuerza dependiente del tiempo: fuerza constante, fuerza periódica, impulsos y otras funciones dependientes del tiempo, por ejemplo las funciones lineales.
- Combinaciones de las anteriores.

La dependencia al movimiento implica que los cálculos están basados en los grados de libertad del dispositivo háptico. Un ejemplo de esta fuerza es el modelo de elongación de un resorte, descrito por la Ley de Hook [27]:

$$\bar{F} = k\bar{x}$$

En donde

k es una constante de rigidez, y

$\bar{x}$ es un vector de desplazamiento.

El resorte necesita una posición de anclaje y una posición final. La fuerza producida es conocida como fuerza de restitución del resorte.

Otros tipos de esta dependencia de movimiento que pueden ser simulados en un dispositivo háptico, se mencionan en la guía del programador de OpenHaptics [26]:

- Impulso. Es un vector de fuerza instantáneo. En un dispositivo háptico supone una imagen táctil en un intervalo pequeño de tiempo.
- Amortiguación. Un amortiguador reduce la vibración. La capacidad de un amortiguador es directamente proporcional a la velocidad del efector final, multiplicada por la constante de amortiguación. La resultante siempre es opuesta a la dirección de movimiento.
- Fricción. Existen varias formas de fricción que pueden simularse mediante un dispositivo háptico, en donde encontramos: fricción coulombica, fricción viscosa, fricción estática y fricción dinámica.
    - Fricción coulómbica. Es la que simplemente se opone a la dirección de movimiento con una magnitud constante. Este tipo de fuerza ayuda a crear transiciones cuando existe un cambio de dirección.
    - Fricción viscosa. Esta forma de fricción es similar a la anterior, en cuanto a que se calculan mediante un punto de anclaje. La diferencia radica en que la constante es menor y el valor del anclaje tiende a ser alto.

    - Fricción estática y dinámica. En este tercer tipo de fricción la fuerza se opone al movimiento lateral a lo largo de una superficie y su magnitud es proporcional al vector normal a la superficie en donde se aplica la fuerza.
- Inercia. Asociada al acto de mover una masa. Dada una trayectoria conocida, se calcula la fuerza resultante durante el movimiento mediante la ecuación de la Segunda Ley del Movimiento de Newton.

En cuanto a la dependencia al tiempo, los cálculos de fuerza están calculados en función del mismo. A este respecto la guía del programador de OpenHaptics [26] indica dos tipos: fuerza constante y fuerza periódica. No obstante, se debe aclarar que en general el dispositivo háptico Omni es capaz de manejar una gama de funciones simples dependientes del tiempo.

- La fuerza constante es aquella que utiliza la magnitud para compensaciones gravitacionales, por ejemplo, cuando el efector final identifica una pérdida o falta de peso, o bien se genera una sensación de incremento de peso.
- Una fuerza periódica se obtiene al aplicar una señal periódica en el tiempo. Esta fuerza está descrita por una constante de tiempo, la cual controla el periodo y la amplitud, lo que determina la intensidad de la fuerza en cada punto de la señal, aun más; la fuerza periódica requiere de un espacio geométrico definido para su despliegue, así, es posible hacer la simulación de tocar una línea, un cubo, una esfera y otras formas básicas.
- Otras fuerzas que no caen en las categorías anteriores como es el caso de aquellas basadas en funciones aleatorias.

Una variable de interés para efectos de este trabajo es la rigidez. La rigidez es la resistencia de un cuerpo elástico a la deformación por una fuerza aplicada. La dimensión de la rigidez está dada por unidad de fuerza sobre unidad de distancia. Si el cálculo de la fuerza se calcula mediante una aproximación de la ley de Hooke, esto implica que la rigidez, multiplicada por una distancia generará la fuerza de interés o bien el campo de fuerza deseado.

Un campo vectorial asocia un vector con un punto en el espacio [14], por lo que si tomamos un conjunto de vectores de fuerza y asociamos cada uno a un punto

perteneciente a una región en el espacio se obtiene un campo de fuerza. Si dicha región tiene una geometría definida por cuerpos concretos obtendremos una parte del espacio en donde interactuarán vectores de fuerza para definir una imagen táctil coherente.

2.4 Fuerza en el Simulador.

Como se mencionó en el capítulo 1, el bloque de gráficos del Simulador contiene un módulo de cálculos, en el cual, y entre otras variables de interés para la simulación, se determina el valor de una fuerza. En la figura 14 se muestra la estructura general del programa del simulador.

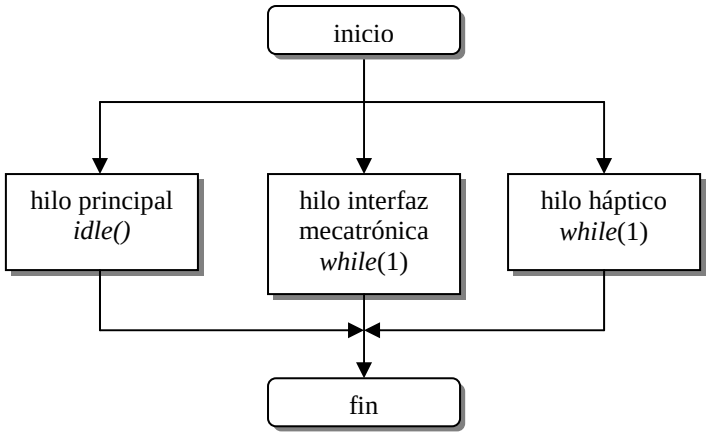

Figura 14. *Estructura general del bloque de gráficos del Simulador*.

Como puede observarse, la implementación del programa está basada en hilos de control. Un hilo es un tipo abstracto que representa el flujo de control dentro de un proceso como una secuencia ininterrumpida de instrucciones [24]. Dentro del hilo principal se encuentra una función encargada de habilitar la animación mientras que se realizan cálculos para obtener una simulación correcta. Es en aquella donde se procesan los datos relacionados con la obtención de la fuerza. Cabe mencionar que el Simulador utiliza el método de masas y resortes, el cual permite tener nodos ponderados conectados entre sí con propiedades de resortes, como herramienta para tener modelos de tejido deformable. De esta manera se tiene para el modelo anatómico una malla de tetraedros gráfica (véase figura 15), una malla que tiene asociados a sus vértices masas

y a sus aristas resortes, y la correspondencia entre ellas dentro de un medio viscoso. Los detalles del cálculo de fuerzas en el modelo gráfico de la próstata se reportaron en [21].

En la ejecución del programa se completa el ciclo:

- Obtención de datos de la interfaz mecatrónica
- Detección de colisiones
- Cálculo de deformaciones
- Actualización de información para la simulación (la malla gráfica se actualiza a partir de la malla deformable)
- Dibujado de la escena

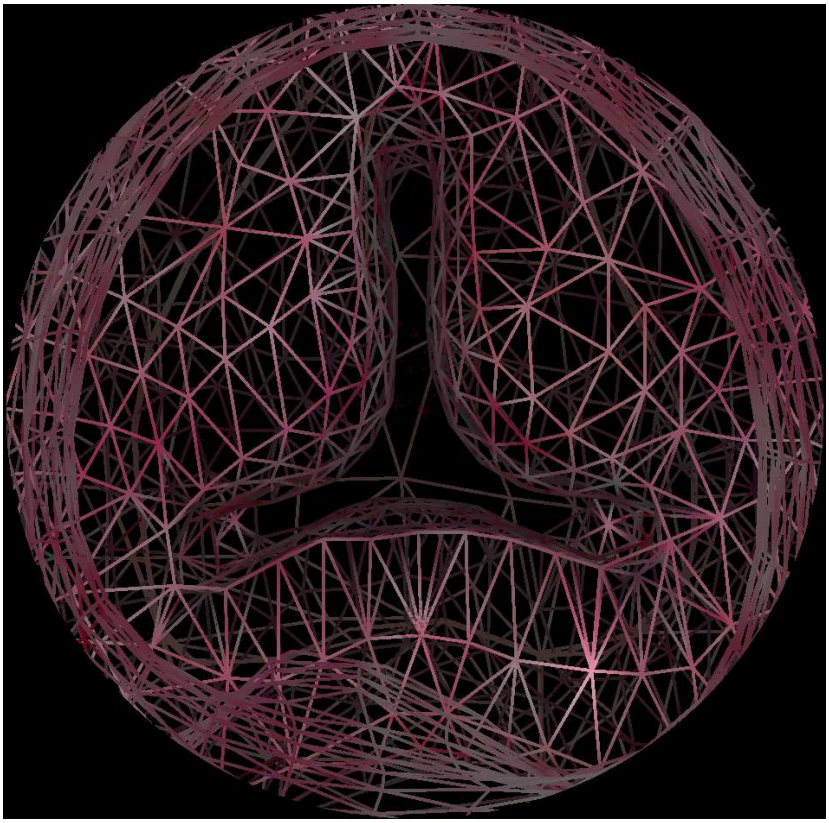

Figura 15. Malla gráfica del modelo de la próstata. Laboratorio de Imágenes y Visualización. CCADET, UNAM 2010.

La fuerza se calcula dentro del alcance en el que se determinan las propiedades de la malla deformable con el siguiente algoritmo:

Obtiene Rigidez

Obtiene Lista de vértices en colisión

Para todos los vértices:

Obtiene Lista de triángulos en colisión

Para todos los triángulos:

Obtiene la distancia del vértice al triángulo

Dmin = distancia menor

Obtiene normales de triángulos

*Penetración = normal del triángulo * Dmin*

*Fuerza = Rigidez * Penetración*

*Fuerza = Fuerza * Factor de escala*

Fuerza de reacción = Fuerza de reacción + Fuerza

Incrementa en uno número de fuerzas

Fuerza de reacción =Fuerza de reacción / número de fuerzas

Como nota importante hay que señalar que al interpretar la fuerza de reacción obtenida se deducen unidades de Fuerza, sin embargo, los datos que se procesan son ambiguos y no se tiene una forma de saber sus dimensiones claramente. Por ejemplo, la descripción geométrica del modelo gráfico de la próstata contiene vértices, definidos por vectores de tres componentes, los cuales no reflejan su cantidad física pero sí implican un módulo al asociarlos con un marco de referencia. Lo anterior tiene como consecuencia que se realicen cálculos con magnitudes que no reflejan su naturaleza dimensional. Se cuenta entonces con una magnitud y una dirección para el vector resultante.

2.5 Open Haptics

Open Haptics es un paquete de programación desarrollado por la compañía SensAble Technologies que permite la creación de aplicaciones de computación háptica por medio de dispositivos hápticos diseñados por la misma empresa.

Para obtener OpenHaptics es necesario ingresar a la página de la compañía en donde se descarga gratuitamente el paquete de programación, los controladores para el dispositivo háptico elegido y el paquete de utilerías para el desarrollo de programas de

ejemplo [25]. El sitio cuenta con un área de ayuda y se pueden consultar los requerimientos del sistema y plataformas que soportan OpenHaptics.

El desarrollo de programas en OpenHaptics está diseñado para implementarlos con VisualStudio, en lenguaje C++, las versiones de entornos de programación que lo soportan son VisualC++ 6, 7, y 8.

El paquete de programación está compuesto por:

- HDAPI, Haptic Device API
- HLAPI, Haptic Library API
- Utilities
- PDD, Phantom Device Drivers
- Código fuente de ejemplos
- Guía del programador y
- Referencia del API

HDAPI provee acceso a bajo nivel al dispositivo háptico. Permite el render de fuerzas directamente. Esta API tiene dos componentes principales, el dispositivo y el calendarizador. El dispositivo permite a cualquier dispositivo háptico ser usado con HDAPI. El calendarizador es un mecanismo de sincronización de tal manera que el hilo de la aplicación principal puede modificar el estado o información del servo – ciclo de una manera segura. Las llamadas al planificador permiten manejar comandos que se ejecutan en el servo – ciclo, generalmente mediante funciones de calendarización, las cuales se almacenan en un orden de prioridad, ya sea alto o bajo.

HLAPI está construido sobre HDAPI y está diseñado para un manejo más sencillo; soporta el despliegue de fuerzas a alto nivel. Permite el uso de código de OpenGL y simplifica la sincronización entre los hilos de gráficos y fuerzas. Permite especificar primitivas geométricas junto con las propiedades de materiales como la rigidez y la fricción.

Un concepto importante en la programación háptica es el *Proxy*. Un *Proxy* es una abstracción geométrica que realiza una acción en representación de otro. Su geometría es típicamente una esfera o colección de puntos. Cuando se refiere a un punto se le conoce como *Surface Contact Point* (SCP), es decir, el punto que indica la posición en el mundo virtual del dispositivo háptico, el cual se mueve en el mundo real.

2.5.1. Ambientes Hápticos.

Un ambiente háptico consta de dos partes: el entorno gráfico y el entorno háptico. Este último requiere de un sistema de coordenadas para definir el espacio físico (limitado por las dimensiones del dispositivo háptico) en el cual se desarrollará el despliegue de fuerza. La geometría definida por el conjunto de fuerzas desplegadas se mapea con la geometría del modelo gráfico para que exista una correspondencia lógica entre lo que el usuario ve y lo que siente.

La parte gráfica está soportada por alguna biblioteca de gráficos como OpenGL, la cual es una biblioteca abierta para graficación. Un aspecto importante que se debe considerar es que la tasa de refresco para el despliegue háptico es mayor que el refresco de despliegue gráfico. Una tasa aceptable para la parte visual es de veinticuatro a sesenta veces por segundo, mientras que una tasa para la parte háptica es de trescientos a mil veces por segundo, por lo que un ambiente háptico debe implementarse mediante procesos separados.[1]

2.5.2. Estructura básica de un programa en Open Haptics.

La siguiente figura (figura 16) muestra el diagrama de flujo de un programa con HDAPI [26]. El dispositivo háptico es inicializado, se habilita el despliegue de fuerzas y se hace el llamado a una función de calendarización, la cual se utiliza como mecanismo de sincronización para evitar conflictos al modificar los datos dentro de las estructuras concurrentes (hilos de control) que suceden durante la ejecución de la simulación. Entonces se inicializa el contexto en el cual se calculará la fuerza resultante de algún evento particular, generalmente una detección de colisiones, a partir de la posición

[1] Para ello se utiliza la concurrencia, tema que se explica en el siguiente capítulo.

actual del efector final. Se despliega la fuerza y se cierra el contexto. En caso de que el usuario decida terminar el programa se deshabilitan la función de calendarización y el dispositivo háptico.

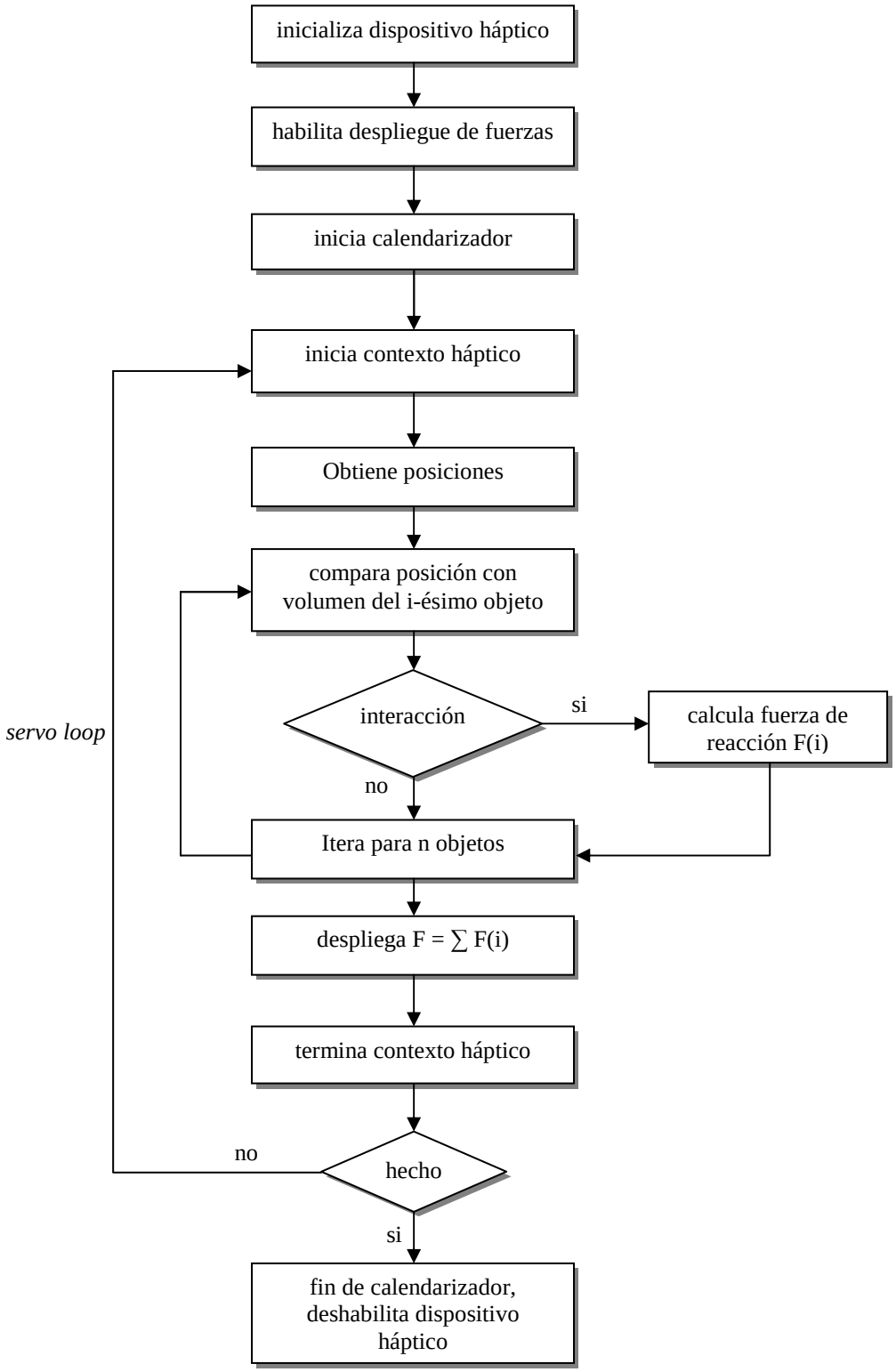

Figura 16. *Diagrama de flujo de un programa en HDAPI* [26].

Un programa sencillo con HDAPI se muestra a continuación. La aplicación crea un atractor hacia un punto cuando se pasa de cierta región esférica. El programa se llama HelloHapticsDevice.c[2].

```
#iclude <stdio.h>
#include <assert.h>

#if defined(WIN32)
# include <conio.h>
#else
# include "conio.h"
#endif

#include <HD/hd.h>
#include <HDU/hduError.h>
#include <HDU/hduVector.h>
```

HD y HDU son las librerías necesarias para las aplicaciones. En este caso se utilizan rutinas que manejan errores y operaciones diversas con vectores.

```
HDCallbackCode HDCALLBACK gravityWellCallback(void *data);
```

El uso de llamadas al calendarizador (*scheduler callback*) es el de accesar a variables modificadas por el hilo en el cual se ejecutan operaciones para el dispositivo háptico. Define operaciones que se ejecutarán en el planificador.

La función principal inicializa el dispositivo, inicia el calendarizador, crea una llamada para manejar las fuerzas de atracción, espera a que sea oprimida una tecla y sale de la aplicación.

```
HDErrorInfo error;
```

Define una estructura cuando se reporte un error

```
HDSchedulerHandle hGravityWell;
```

Es un manejador para las operaciones del planificador

```
HHD hHD = hdInitDevice(HD_DEFAULT_DEVICE);
   if (HD_DEVICE_ERROR(error = hdGetError()))
```

Verifica si ha ocurrido un error, busca en la pila si hay error

```
   {
      hduPrintError(stderr, &error, "Failed to initialize haptic device");
      fprintf(stderr, "\nPress any key to quit.\n");
```

[2] Copyright (c) 2004 SensAble Technologies, Inc. All rights reserved.

```
        getch();
        return -1;
    }

    printf("Hello Haptic Device!\n");
    printf("Found device model: %s.\n\n", hdGetString(HD_DEVICE_MODEL_TYPE));
```

Calendariza una operación en el *servo loop* y no espera a que sea completada, este proceso se asigna al manejador de operaciones del calendarizador. En este caso llama al atractor de fuerzas, sin datos de usuario y con una alta prioridad. Después habilita las fuerzas al dispositivo. Luego empieza el habilitador de fuerzas. Las llamadas asíncronas regresan inmediatamente después de planificarse, generalmente son las mejores para manejar el ciclo del háptico.

```
hGravityWell = hdScheduleAsynchronous(
        gravityWellCallback, 0,
        HD_MAX_SCHEDULER_PRIORITY);

    hdEnable(HD_FORCE_OUTPUT);
    hdStartScheduler();

{
        hduPrintError(stderr, &error, "Failed to start scheduler");
        fprintf(stderr, "\nPress any key to quit.\n");
        return -1;
    }
```

Esto no está dentro del contexto pero está en el *servo loop*, el contexto está en la función gravityWellCallback y ella misma es el *servo loop*.

```
    printf("Feel around for the gravity well...\n");
    printf("Press any key to quit.\n\n");
    while (!_kbhit())
    {
```

hdWaitForCompletion verifica si una llamada está calendarizada para su ejecución, el WAIT_CHECK devuelve la constante TRUE si continúa activo.

```
        if (!hdWaitForCompletion(hGravityWell, HD_WAIT_CHECK_STATUS))
        {
            fprintf(stderr, "Press any key to quit.\n");
            getch();
            break;
        }
    }
```

Lo siguiente es detener la planificación, limpiar las llamadas no calendarizadas y deshabilitar el dispositivo háptico.

```
    hdStopScheduler();
    hdUnschedule(hGravityWell);
    hdDisableDevice(hHD);
    return 0;
}
```

La siguiente función es la que simula el campo atractor de fuerza, el cual atrae al dispositivo a un centro de gravedad si se ha acercado cierto rango.

```
HDCallbackCode HDCALLBACK gravityWellCallback(void *data)
{
    const HDdouble kStiffness = 0.075; /* N/mm */
    const HDdouble kGravityWellInfluence = 40; /* mm */

    static const hduVector3Dd wellPos;

    HDErrorInfo error;
    hduVector3Dd position;          Define al vector de posición del efector final
    hduVector3Dd force;             Define al vector de fuerza
    hduVector3Dd positionTwell;     Define al vector


    HHD hHD = hdGetCurrentDevice();//reconoce al háptico actual
```

Comienza el contexto háptico.

```
    hdBeginFrame(hHD);

    /* Get the current position of the device. */
    hdGetDoublev(HD_CURRENT_POSITION, position);

    memset(force, 0, sizeof(hduVector3Dd));
```

Crea un vector desde la posición del dispositivo hacia el centro de gravedad.

```
    hduVecSubtract(positionTwell, wellPos, position);
```
función que resta vectores

Cálculo de la colisión.

```
    if (hduVecMagnitude(positionTwell) < kGravityWellInfluence)
    {
```

Ley de Hooke.

```
        hduVecScale(force, positionTwell, kStiffness);
    }
```

Envío de despliegue de fuerza.

```
    hdSetDoublev(HD_CURRENT_FORCE, force);

    hdEndFrame(hHD);

    if (HD_DEVICE_ERROR(error = hdGetError()))
    {
```

```
        hduPrintError(stderr, &error,
                      "Error detected while rendering gravity well\n");

        if (hduIsSchedulerError(&error))
        {
           return HD_CALLBACK_DONE;
        }
    }
    return HD_CALLBACK_CONTINUE;
}
```

En figura 17 se muestra la geometría del campo de fuerza resultante. El origen define vectores de posición, los cuales son detectados por el dispositivo háptico continuamente. La colisión se detecta cuando la magnitud del vector de posición es menor al radio de influencia definido. Si hay colisión, entonces la fuerza es calculada mediante la penetración por la rigidez, dato conocido, de la esfera virtual. La dirección de la fuerza se dirige hacia el centro de la esfera por lo que la sensación táctil es la de una región esférica de atracción si se cae dentro de su volumen.

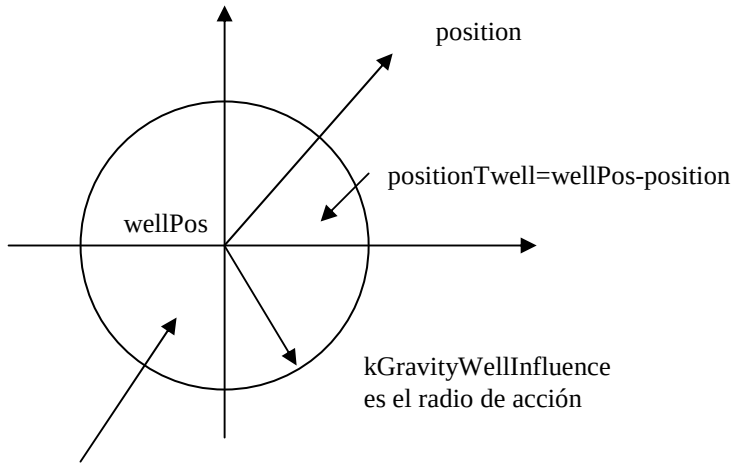

Figura 17. Geometría esférica del despliegue de vectores de fuerza.

Capítulo 3. Integración del sistema

En este capítulo se describen las tecnologías de computación utilizadas y el método que se desarrolló para integrar el dispositivo háptico al Simulador de Cirugía de Próstata tomando en cuenta la importancia que produce la distribución de cómputo sobre una red para formar aplicaciones nuevas. El modelo más utilizado para ello es la arquitectura cliente – servidor.

3.1 Protocolo de comunicación TCP/IP.

El Programa de Simulación, como se mencionó en el capítulo uno, fue escrito usando el IDE de Bloodshed Dev C++, los recursos que éste utiliza, y los que generan los programas escritos para Open Haptics, con Visual C++, producen un problema de incompatibilidad entre las bibliotecas de enlace dinámico[1] de ambos entornos de programación que se producen en su respectiva compilación, lo que provoca la imposibilidad de contar con la adaptación del dispositivo háptico Omni.

De esta manera se plantea una solución para la integración del dispositivo háptico al Simulador, que consiste en desarrollar un método para hacer que los procesos sean compatibles. Para ello se utilizó el protocolo TCP/IP mediante el uso de canales de comunicación llamados sockets en una arquitectura cliente – servidor.

El modelo de referencia de interconexión de sistemas abiertos, OSI[2], desarrollado por la IOS[3], está compuesto por siete capas, en cada una de las cuales, se definen los distintos elementos que intervienen en la comunicación de datos. Cada capa contiene un conjunto de servicios que proporciona a la capa siguiente y utiliza el conjunto de servicios que otorga la capa anterior. “Los servicios de una capa pueden verse como una interfaz de comunicación con la misma. Los protocolos, por su parte, definen la forma que tendrán las tramas de bits que intercambian las distintas capas y la manera en que se va a llevar a cabo dicha comunicación” [16].

[1] Dinamic Link Library, DLL.
[2] Open System Interconnection
[3] International Organization for Standarization

Las siete capas que forman al modelo OSI son:

1. Capa física. Provee la transmisión de datos, a nivel de señales en el medio.
2. Capa de enlace. Transfiere unidades de información, y verifica errores.
3. Capa de red. Entrega paquetes de información y se encarga del *routing*.
4. Capa de transporte. División de paquetes en unidades más pequeñas.
5. Capa de sesión. Establece sesiones de trabajo en máquinas remotas.
6. Capa de presentación. Revisa la sintaxis y semántica de los datos.
7. Capa de aplicación. Define los protocolos que usan las aplicaciones de usuario para intercambiar datos.

El protocolo TCP/IP pertenece a la capa de transporte en el modelo OSI, encargada de llevar confiablemente los datos recibidos de la capa de red hacia la capa de sesión. Es en esta etapa en donde es posible utilizar canales de comunicación para el envío remoto de datos.

Un esquema de la arquitectura del protocolo TCP/IP se muestra a continuación (figura 18).

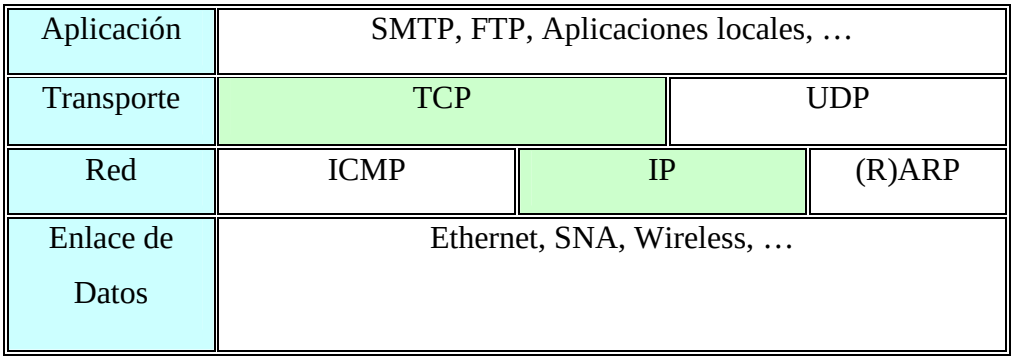

Figura 18. *Estructura del protocolo TCP/IP*.

El TCP, o Protocolo de Control de Transmisión, está definido en el Request For Comments[4], RFC – 793 [31], provee transporte orientado a conexión y presenta un flujo de bytes transparente para aplicaciones finales.

[4] Petición de Comentarios, serie de propuestas oficiales para protocolos de internet normalizada por el Internet Engineering Task Force, IETF.

Para establecer una conexión es necesario saber la dirección del Protocolo de Internet, IP, y el número de puerto del protocolo. La versión cuatro de IP, maneja un identificador entero de 32 bits, divididos en dos partes; la primera representa el número de red y la segunda representa el número de host. Un puerto está definido por un número entero de 16 bits, con lo cual resultan 2 elevado a la 16 puertos posibles, cada uno de los cuales se utiliza para un servicio o aplicación. La estructura de la conexión se puede esquematizar de la siguiente manera (figura 19):

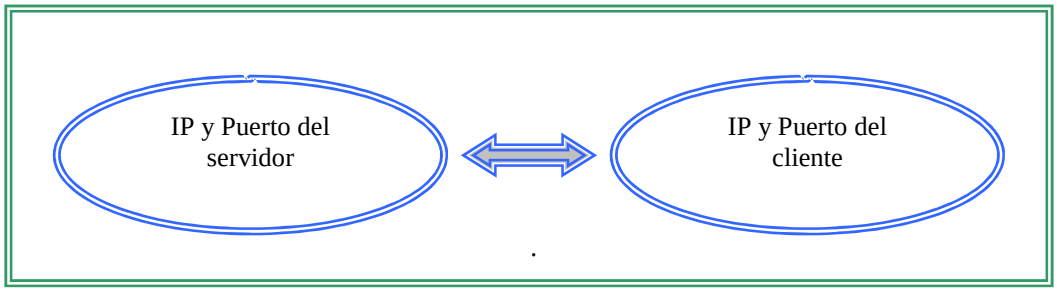

Figura 19. *Comunicación TCP/IP*

La figura 20 muestra en forma esquemática la solución para la comunicación de procesos mediante la apertura de canales, llamados sockets, entre los procesos cliente y servidor. Como se mencionó en la introducción, el cliente es el Programa de Simulación y el servidor es el programa que controla al dispositivo Omni :

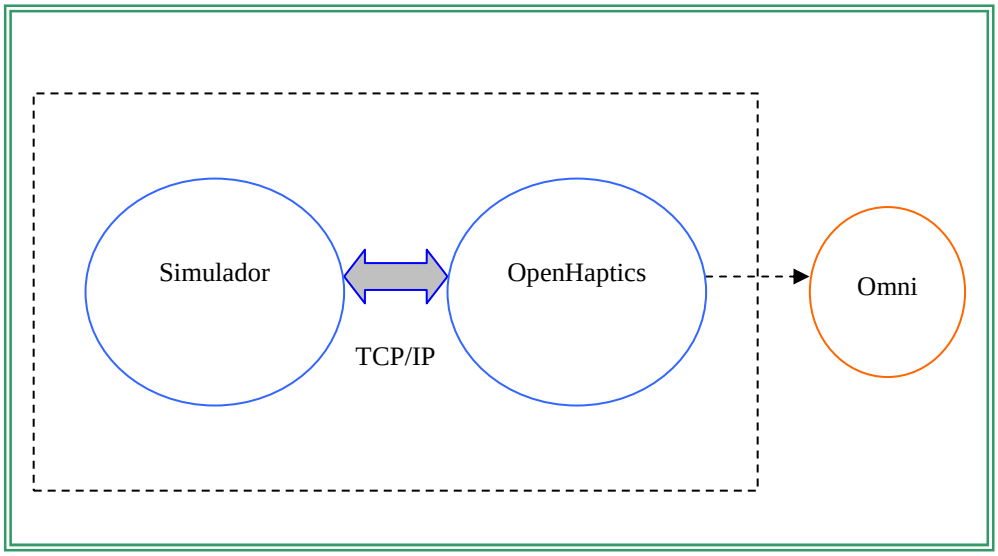

Figura 20. Comunicación *TCP/IP*.

3.1.1 Sockets.

Un *socket* es una interfaz de comunicación en una red de datos. Márquez García [16] dice: “la comunicación mediante sockets es una interfaz con la capa de transporte (nivel 4) de la jerarquía OSI” “a la hora de establecer una conexión mediante sockets, es necesario conocer la familia o dominio de la conexión y el tipo de conexión.”.

Una familia o dominio es un conjunto de *sockets* con características similares como es el caso de protocolos.

El tipo de conexión define el tipo de circuito que se va a emplear para establecer la comunicación, de tal forma que se pueden tener circuitos virtuales, orientados a conexión; y circuitos datagramas[5], no orientados a conexión. Los circuitos virtuales envían secuencialmente los datos mediante circuitos libres en la red de comunicación. La conexión mediante circuitos datagramas utiliza datos empaquetados, donde cada uno puede tener una ruta distinta, pero su integridad no es confiable.

El tipo de conexión elegido para conectar las aplicaciones del Simulador y del Dispositivo Háptico es por medio de circuitos virtuales, los cuales se implementan mediante el protocolo TCP/IP, ya que se requiere un flujo continuo de datos del servidor hacia el cliente.

La figura 21 muestra el diagrama de flujo del control básico para el manejo de *sockets*. El lado derecho corresponde al proceso cliente que en la solución propuesta estará dentro del Programa de Simulación, y el lado izquierdo al proceso servidor, mismo que toma datos del Omni y envía al cliente.

El proceso cliente crea un *socket*, solicita la conexión y envía – recibe datos. El proceso servidor crea un socket, informa o publica la dirección y su disponibilidad, empieza la escucha, que es la espera para una petición de conexión, acepta dicha conexión, y recibe – envía datos.

[5] Unidad básica de transferencia de datos, se compone básicamente de un encabezado y de algunos datos.

Cuando se requiere establecer una conexión, el Sistema Operativo debe conocer los siguientes datos para el uso de *sockets*:

1. Familia de protocolos.
2. Tipos de servicios.
3. Dirección IP local.
4. Puerto local.
5. Dirección IP remota.
6. Puerto remoto.

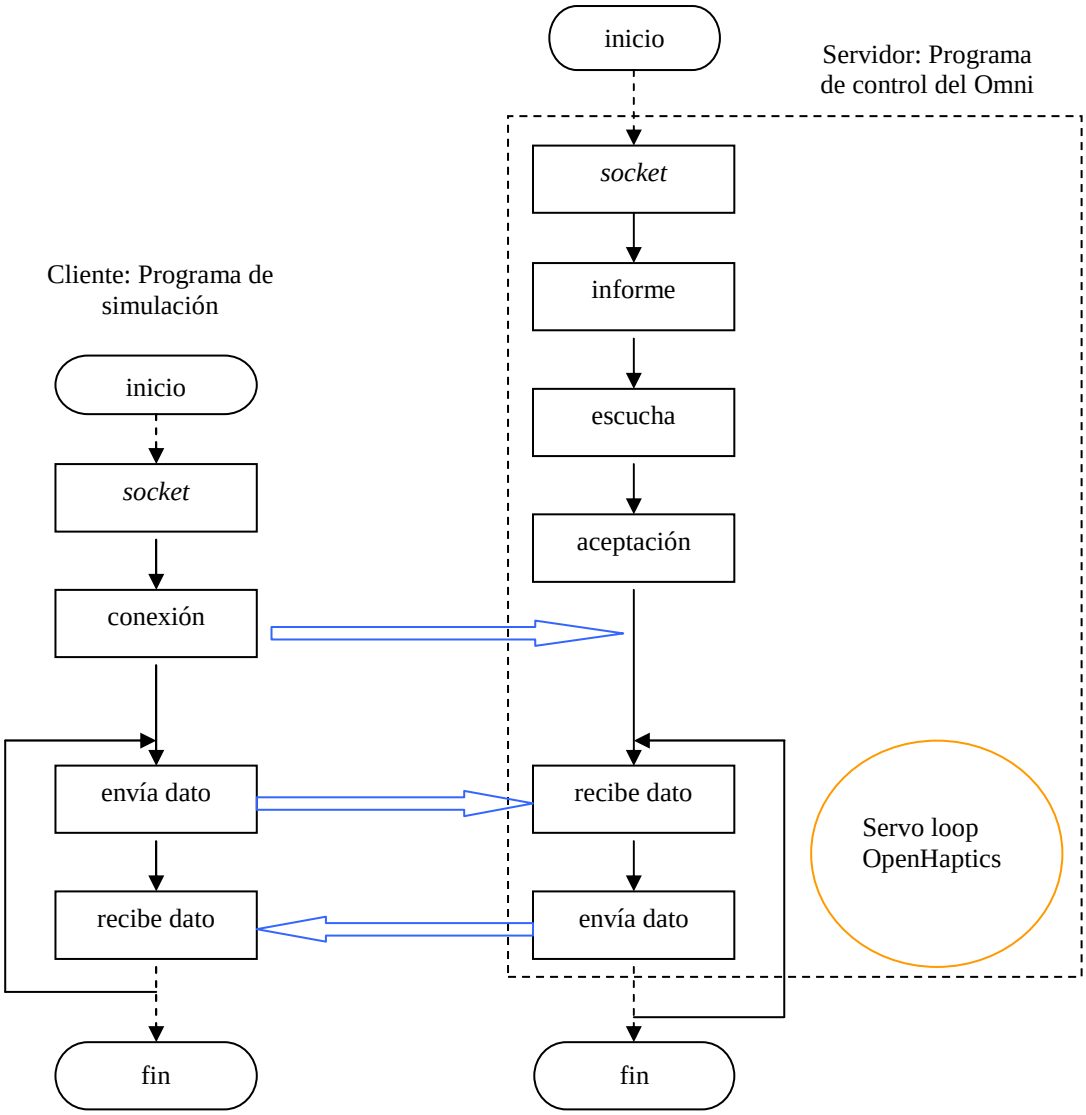

Figura 21. *Diagrama de flujo para manejo de canales de comunicación*.

3.1.2 Sockets en Windows

Para el manejo de canales de comunicación en Windows es necesario utilizar la librería que permite las llamadas a las funciones de *winsock*, WS2_32.lib, definidas en el archivo de encabezado <winsok.h>.

Los servidores y los clientes en Windows tienen comportamientos diferentes, sin embargo siguen el flujo básico de los *sockets*, los siguientes pasos ilustran la manera general en la que funciona cada uno (figura 22):

Servidor	Cliente
Inicializar *winsock*	Inicializar *winsock*
Crear un *socket*	Crear un *socket*
Publicar el *socket*	Conectar al servidor
Escuchar en el socket al cliente	Enviar y recibir datos
Aceptar la conexión desde el cliente	Desconectar
Recibir y enviar datos	
Desconectar	

Figura 22. *Control de flujo para sockets en Windows.*

3.2. Programación concurrente.

Con lo anterior se tiene comunicación entre los procesos, sin embargo, es necesario que dentro del cliente como del servidor existan procesos simultáneos para tener disponibles los datos que harán que la simulación tenga coherencia, es decir, se debe tener una manera para obtener dichos datos que se calculan dentro de ciclos infinitos. Para ello es necesaria la concurrencia de procesos.

La concurrencia es, de acuerdo con Robbins y su colaborador [24], la ejecución de dos programas en el mismo sistema de manera tal que comparten recursos[6] de

[6] Algunos recursos son procesador, memoria, dispositivos de entrada y salida, entre otros.

procesamiento y se entrelazan en el tiempo, y como consecuencia, el procesador parece ejecutar dichos programas simultáneamente.

Las entidades que hacen posible la concurrencia son conocidas como hilos de control de ejecución dentro de un programa, en este sentido Robbins [24] dice:

“Cuando se ejecuta un programa, la CPU utiliza el contador del programa para determinar qué instrucción es la siguiente por ejecutar. El flujo de instrucciones resultante recibe el nombre de hilo de ejecución del programa. Este es el flujo de control del proceso”.

Lira [15] plantea que para optar por la concurrencia en la solución de un problema es importante definir si se puede separar en problemas más sencillos y mutuamente independientes; entendiendo por independencia “la nula necesidad de esperar la finalización de un subproblema para la inicialización de otro”.

Considerando la definición anterior, se plantea entonces, utilizar la concurrencia para resolver el problema del flujo de control de ciclos infinitos en los programas cliente y servidor, es decir, implementar hilos de control que posibiliten la interacción entre los ciclos infinitos en cada parte de la arquitectura propuesta. En la figura 23 se muestra un esquema de la solución con comunicación y concurrencia entre los programas respectivos.

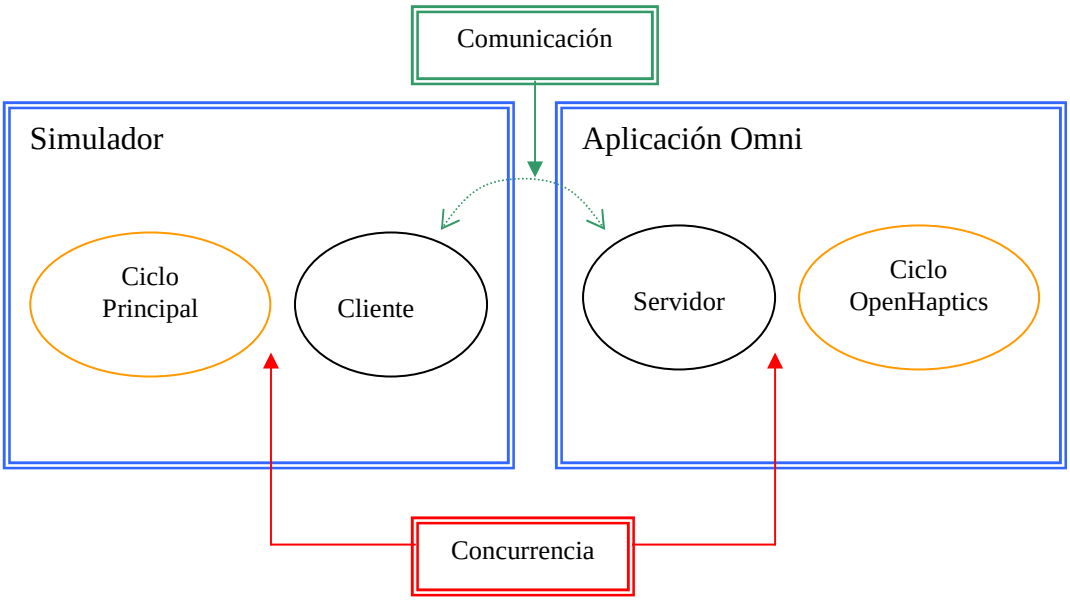

Figura 23. *Concurrencia entre procesos.*

3.2.1. Hilos Posix.

El IDE de Dev – C++ soporta la programación concurrente mediante la definición estándar para empleo de hilos de control, POSIX[7], por sus siglas en inglés. Este estándar fue desarrollado por el IEEE[8] con identificador 1003 para especificar la interfaz entre un sistema operativo y el usuario, de manera que se asegura la portabilidad entre distintas plataformas.

Es necesario asegurarse que la independencia entre procesos se implemente de forma segura, para ello, se utiliza la sincronización utilizando un objeto del Sistema Operativo llamado mutex. Este objeto provee un hilo con acceso mutuo exclusivo a un solo recurso. Cualquier hilo del proceso llamador puede especificar el manipulador del objeto mutex en una llamada a una función de espera. Las funciones de espera regresan cuando el estado del objeto específico es señalado. El estado de un objeto mutex es señalado cuando no pertenece a algún hilo. Si el estado del mutex está en “señalado”, un hilo en espera es candidato a poseerlo, el estado cambia a “no señalado” y la función de espera regresa. Una vez ocupado el mutex, el hilo que lo utilizó lo puede dejar libre mediante una función de liberación de mutex.

Los pasos básicos para tener hilos funcionales dentro de un programa son:

- Declarar una variable manejadora del hilo
- Crear un hilo (Llamar a la función que se implementa por el hilo)
- Al terminar su ejecución, reunir los hilos existentes en un hilo final

En el IDE de Dev – C++, la implementación POSIX se encuentra en el archivo de encabezado <pthread.h>; la variable manejadora debe ser de un tipo de dato “pthread”; esta variable se debe declarar en una función principal o “main”. Dentro del alcance de esta función se crea el hilo mediante el llamado a la función:

pthread_create(&mythread, NULL, thread_function, NULL)

[7] Portable Operating System Interface.
[8] Institute of Electrical and Electronics Engineers.

en donde, el primer parámetro es el nombre del manejador del hilo, por ejemplo, “mythread”; el segundo se usa para definir algunos atributos especiales para el hilo; el tercero es el nombre de la función que va a ser implementada por el hilo definido por el manejador y el cuarto argumento es un receptor de un conjunto arbitrario de datos. En el caso del segundo y cuarto argumento, si no son necesarios se declaran como NULL o 0.

Para reunir los hilos, tomando en cuenta que la función principal es, en sí un hilo, se utiliza la función

pthread_join (mythread, NULL);

en donde el primer argumento es el manejador del hilo creado, y el segundo es un apuntador que hace referencia a la localidad de memoria que se enviará a una función pthread_exit() para regresar un valor del hilo creado.

3.2.2. Hilos Windows en Visual C++.

Windows, por su parte, no maneja el estándar POSIX para el manejo de procesamiento concurrente, en su lugar, tiene desarrollada su propia implementación definida en el archivo de encabezado <windows.h>, mediante el uso de funciones para manejo de hilos del API de win32. La funcionalidad e implementación de los hilos de win32 es muy parecida a la implementación de los hilos POSIX, la página electrónica para soporte técnico de desarrolladores de Microsoft [17] lista como similitudes:

- Cada hilo tiene un punto de entrada. El nombre del punto de entrada debe ser especificado para una manipulación más sencilla.
- Cada hilo contiene un solo parámetro cuando es creado.
- Una función dentro de un hilo debe regresar un valor.
- Una función dentro de un hilo necesita usar parámetros y variables locales tanto como sea posible.

Dentro de la función principal se crea el hilo mediante la función:

```
a_thread = CreateThread(NULL, 0, hiloMensajero, NULL, 0, &a_threadID);
```

en donde, a_trhread es una variable de tipo DWORD y es la que manejará al hilo creado, de los seis parámetros que toma, el tercero es el nombre de la función que implementa la tarea que ejecutará el hilo, los otros no tienen relevancia para esta aplicación, sin embargo, deben declararse para el correcto funcionamiento de la función. El primer argumento es el tamaño de la pila que contendrá al hilo, el segundo es un atributo de seguridad, el cuarto es un valor opcional para manejo de información del usuario, el quinto es una bandera que indica la prioridad del hilo y el último apunta a la dirección que guarda el identificador del hilo.

La terminación del hilo se realiza de manera automática al llegar a la instrucción "return" del programa "main". Para una salida segura se utiliza la función:

```
GetExitCodeThread(a_thread, &thread_result);
```

en donde, el primer parámetro es el manejador del hilo creado y el segundo parámetro es el estado de terminación del hilo. Cabe mencionar que en la solución planteada, existen dos ciclos infinitos en el servidor: el que controla OpenHaptics y el que controla la comunicación; por lo que es necesario manejar accesos a memoria seguros. Es necesario entonces el uso de objetos de exclusión en Visual C++. La siguiente función abre el objeto mutex:

```
HANDLE hMutex = OpenMutex ( MUTEX_ALL_ACCESS, FALSE, NULL );
```

en donde, hMutex es el manejador del mutex, el primer parámetro especifica todos los derechos de acceso, el segundo indica si se heredará el manejador a subprocesos, en este caso no lo hará ya que no hay más procesos y el tercero es para datos del usuario. Para finalizar la exclusión mutua se llaman a las dos instrucciones:

```
ReleaseMutex(hMutex);
CloseHandle(hMutex);
```

las cuales, reciben el mutex. Esto dentro de la función del hilo. En la función "main" debe crearse el manejador de mutex con la siguiente función:

```
shared_mutex = CreateMutex ( NULL, TRUE, NULL );
```

en donde, shared_mutex es la variable que maneja el objeto. El primer argumento apunta a una estructura de seguridad para el hilo, el segundo fija al hilo como propietario del control al inicio del proceso y el tercero maneja datos del usuario. Para finalizar se hace un llamado a la función:

```
CloseHandle(shared_mutex);
```

En la cual se especifica como parámetro el objeto de exclusión que ha sido usado.

3.3 Pruebas preliminares.

Se escribió un programa para probar la funcionalidad de la solución propuesta. Dicho programa recibe la posición actual del dispositivo háptico y realiza el cálculo de un campo de fuerzas concéntricas, de geometría esférica y la manda al proceso remoto para desplegarse en el dispositivo, de tal manera que al manipular el efector final, se logra percibir una esfera que flota en el espacio.

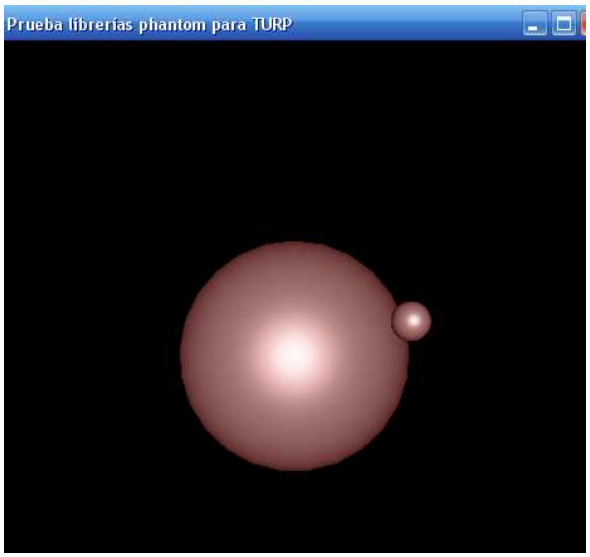

Figura 24. *Ejecución del programa de prueba*.

Este experimento de prueba consiste en un cliente y un servidor. El cliente se ocupa del despliegue gráfico de la escena, la cual está formada por una esfera de posición constante en el centro de la escena y por una esfera más pequeña que representa la punta del efector, la cual se mueve dentro de la escena; mediante un hilo de control recibe datos del dispositivo háptico a través de un *socket* servidor (véase figura 24). Este

cliente, calcula colisiones de geometría esférica en base a los datos obtenidos del servidor los cuales hacen mover a la esfera pequeña, de tal modo que visualmente y de manera táctil exista congruencia.

Una segunda prueba consistió en establecer la interacción del proceso servidor con el Programa de Simulación. Para tal efecto se configuró la IP en el cliente para procesamiento remoto, de tal manera que el servidor se ejecutó en una computadora y el cliente en otra. Se declararon las variables necesarias y se agregaron las funciones que habilitan la comunicación por canales. Se utilizó sólo la variable que manipula el movimiento en la dimensión de profundad en un sistema de referencia tridimensional (véase capítulo 2). Al ejecutar esta aplicación se observa en pantalla la escena principal con el modelo gráfico de la próstata y el movimiento del asa de resección que corresponde con los movimientos en el mundo real del usuario que mueve el efector final del dispositivo háptico.

3.4 Fijación de un origen común.

La determinación de un punto de inicio del dispositivo háptico presenta un problema al no tener la capacidad de reiniciar o fijar una referencia a cero, ya que cuenta con un origen definido (véase figura 26). Esto impacta en la visualización de la escena, en donde es posible que al iniciar la simulación, el efector final inicie en una posición que gráficamente corresponda a una no permitida o errónea, por ejemplo fuera de los modelos anatómicos. El origen del simulador se encuentra en una región próxima al modelo gráfico de la próstata, punto en el que se coloca la cámara para captar la escena que se muestra en la figura 26. Por ello se hace necesaria una forma de hacer corresponder el origen del Programa de Simulación con el origen del dispositivo háptico, considerando que se debe minimizar el procesamiento para que no afecte a la capacidad de desplegar una animación satisfactoria.

Una buena correspondencia entre los orígenes de ambos bloques del Simulador consiste en que al inicio de ésta, sea visualizada la escena mostrada en la figura 26 y que el brazo del dispositivo háptico se encuentre a la mitad de su extensión para poder explorar el modelo gráfico completo que consiste en uretra, próstata y vejiga, dado que el brazo es el que mueve la cámara para lograr los movimientos de visualización a la derecha, a la

izquierda, arriba y abajo. El movimiento absoluto, o del resectoscopio se controla por medio del teclado de la computadora mientras que el movimiento relativo o del asa de corte se controla por medio del brazo del háptico.

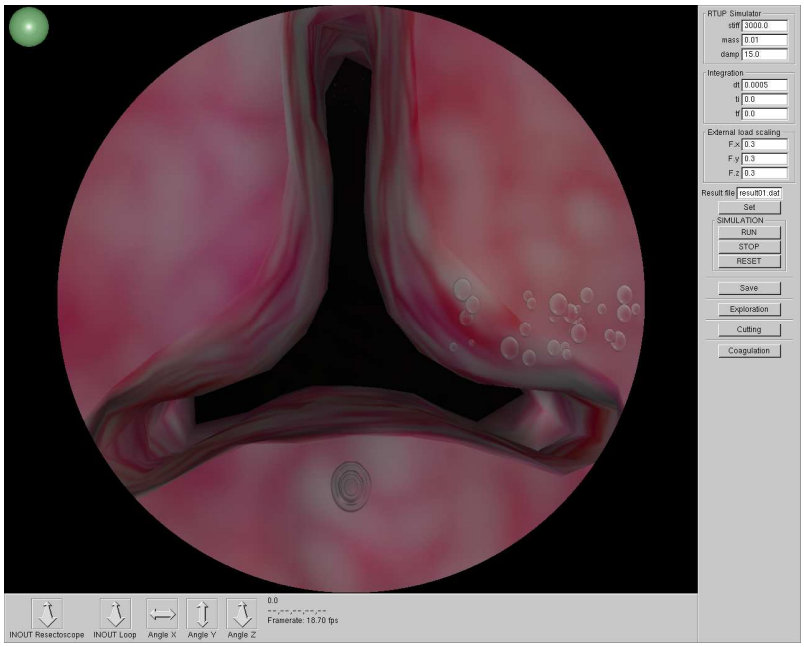

Figura 25. *Vista del origen en el bloque gráfico del Simulador*. Laboratorio de Imágenes y Visualización, CCADET, UNAM 2010.

El método que se utilizó para fijar el origen del háptico fue ejecutar un programa sencillo con despliegue a consola de las posiciones y los ángulos proporcionados por sus sensores en un formato de números flotantes. Con estos datos fue posible obtener una distancia de compensación para fijar una aproximación a cero para los ángulos y las posiciones.

En el caso de la distancia que recorre la herramienta se resuelve de la siguiente forma: Para z se toma la distancia más lejana del centro del efector, y la posición de reposo. Para la componente en *x* y la componente en *y* se toman las lecturas de la posición de reposo que arroja el programa de posicionamiento.

Con estos datos se determina un vector origen ($\overline{O}$) y un vector pivote ($\overline{piv}$), ambos constantes dependiendo de la posición actual del dispositivo. Cabe aclarar que el pivote

es un punto de articulación del disco que sostiene la herramienta o varilla que simula el resectoscopio. Se tiene entonces que:

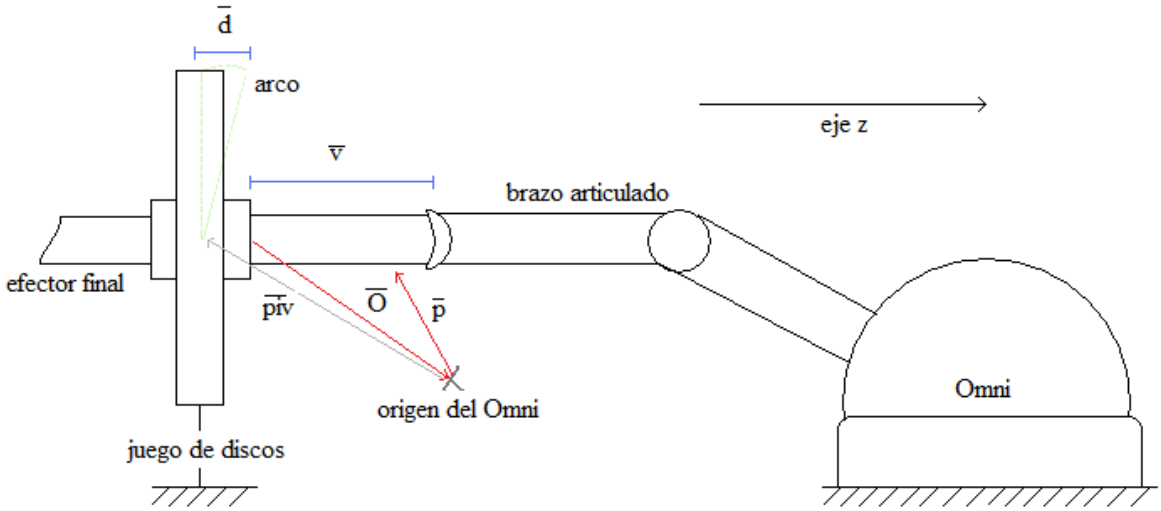

Figura 26. *|v|-|d| es la distancia en z.*

$$\bar{d} = \overline{piv} - \bar{O}$$

$$\bar{v} = \bar{p} - \bar{O}$$

Donde $\bar{d}$ es el vector del centro del disco al borde del cilindro que hace las veces de camisa del resectoscopio, $\bar{v}$ es el vector que representa la herramienta de resección y $\bar{p}$ es una posición dentro del espacio de trabajo.

De la figura 22 se tiene entonces que la distancia recorrida de la herramienta es:

$$dist = \left|\bar{v}\right| - \left|\bar{d}\right|$$

Dato que es el desplazamiento relativo en el Simulador, es decir el movimiento del asa de corte. Con esto se logra un mapeo coherente entre ambos orígenes.

3.5 Integración háptica al Simulador TURP.

La adaptación del dispositivo háptico al sistema de simulación de cirugía de próstata consiste en los siguientes pasos:

1. Definir el cliente: Programa del Simulador (DevC++).
2. Definir el servidor: Programa servidor (VisualC++).
3. Crear hilos de control en ambos procesos.
4. Crear canales de comunicación en ambos procesos.
5. Establecer la comunicación.
6. Manejar las funciones para envío y recepción de datos en ambos procesos dentro de los hilos creados.

3.5.1 El cliente.

Dentro del Programa de Simulación se realiza la creación de un conjunto de funciones. Este módulo se encarga de:

- Configurar el canal de comunicación Tcp/Ip
- Inicializar la conexión del socket
- Establecer la comunicación
- Recibir datos de posiciones del servidor.
- Poner a disposición estos datos para uso de las funciones del simulador.
- Enviar al servidor vectores de fuerza calculados en el simulador.

En estructura del código fuente del Programa de Simulación se encuentra un módulo de inicialización de parámetros. Se agregan las líneas que configuran la dirección y el puerto de la máquina que ejecuta el proceso cliente:

ipAddress[] = 127.0.0.1

portConnect[] = 5000

En el hilo de control “threadMain” se llama a la función que fija el estado inicial del dispositivo háptico. En este mismo “main” se inicializa el hilo que creará a su vez el proceso para la comunicación con el dispositivo háptico Omni (véase figura 27).

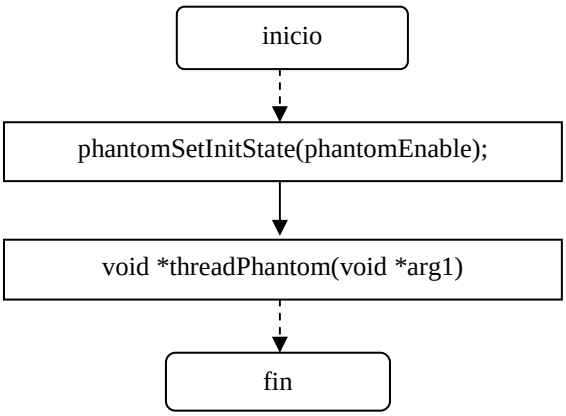

Figura 27. *Creación del hilo para sensaciones táctiles.*

En las figuras 28 y 29 se muestra el funcionamiento del cliente. En el hilo de control “threadPhantom” se declaran variables para posiciones de la herramienta gráfica, ángulos, distancia por defecto y estructuras de datos definidas para la recepción de datos y envío de información. Si el estado del dispositivo háptico se detecta como habilitado, entonces comienza el ciclo infinito. Se llaman a las funciones para fijar la fuerza, envíar datos, recibir datos, obtener desplazamiento sobre el eje z, obtener posición y obtener ángulos.

Las estructuras de datos especifican el formato de la variable que enviará el vector de fuerza al proceso servidor para su despliegue. Se habilita el envío y recepción de datos entre los procesos cliente y servidor, y se obtienen los datos de posición, ángulos y distancia del dispositivo háptico.

A continuación se utiliza el objeto de exclusión mutua para actualizar los datos recibidos del Omni que se guardan en la estructura de datos que calcula las colisiones. Se obtiene el estado de los botones del háptico, de tal manera que se presentan tres posibilidades: La primera es que ninguno de los dos botones esté presionado; para lo cual, el simulador consiste sólo en la exploración del tejido, la segunda es que el botón

número 1 esté presionado con lo que se habilita el corte de tejido; y la tercera forma es que el segundo botón esté presionado para habilitar la coagulación.

La fuerza resultante calculada en las rutinas del Programa de Simulación (véase página 27) es asignada al tipo de dato que guarda las tres componentes del vector que se enviará al dispositivo háptico para su despliegue. Este dato es normalizado para para proteger al dispositivo háptico de datos que salen fuera de su resolución y de esta manera evitarle daños.

La fuerza final de despliegue fue determinada a partir de su sensado por parte de un especialista urólogo[9], quien, con base en pruebas aceptó un rango para estas magnitudes. Esto se realizó mediante una aplicación consistente solamente en el despliegue táctil de fuerzas con la rigidez en incremento. El dato que aprobó el especialista urólogo fue el producido por una rigidez de 0.01 N/mm, que, depediendo de la distancia recorrida produce una sensación parecida a la de una intervención real. La terminación de la aplicación se lleva a cabo cuando el cliente o bien el servidor termina su ejecución.

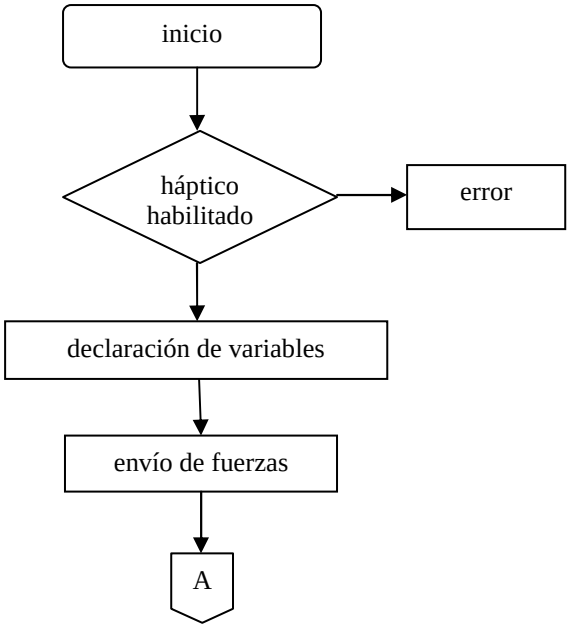

Figura 28. *Flujo de control del cliente (continúa en página siguiente).*

[9] Dr. Sergio Durán Ortiz (http://www.inr.gob.mx/cu261.htm) Asesor médico del proyecto del Simulador.

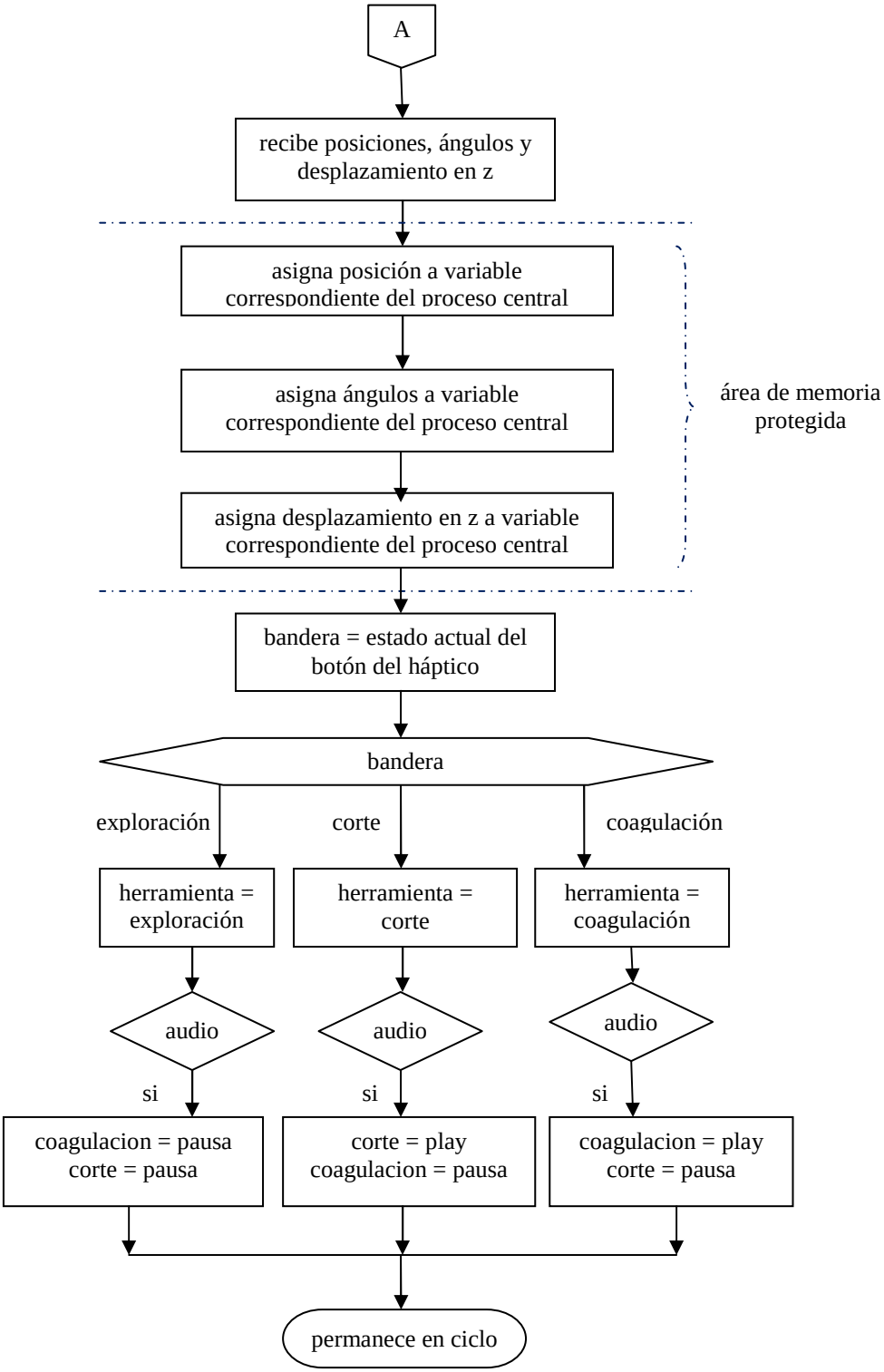

Figura 29. *Flujo de control del cliente (continuación).*

3.4.2. El servidor.

En el caso del programa servidor, se desarrolló la aplicación que se encarga de:

- Configurar el canal de comunicación TCP/IP
- Activar el *servo loop* del dispositivo háptico.
- Recepción de datos correspondientes a vectores de fuerza.
- Habilitar el despliegue de la imagen táctil en el dispositivo háptico.
- Enviar al cliente las posiciones registradas en cada instante.

En la figura 30 se muestra el diagrama de flujo de la aplicación. Inicia con los archivos de encabezado necesarios para el programa. Se declara el prototipo de función para OpenHaptics en donde se calendariza el despliegue de fuerzas en el Omni, donde se determinan las posiciones y los ángulos del dispositivo háptico. En la función principal se declara el "hilo mensajero" y el mutex que va a controlar al hilo principal (es decir la función main) con el hilo mensajero. Esta función principal contiene las declaraciones e inicializaciones de OpenHaptics y el llamado a la calendarización del *servo loop* para render de fuerzas.

La primera función paralela, "hiloMensajero" es la que implementa la conexión, envía y recibe datos. Se crea el *socket* que va a ser el que manipule la escucha de peticiones de conexión. Configura el protocolo tcp por el puerto 5000, u otro que no se ocupe en aplicaciones de red. A continuación se publica la dirección, se atienden las peticiones de conexión y se crea un manejador para aceptar las conexiones. Una vez aceptada la conexión se implementa el ciclo infinito para enviar los datos del dispositivo háptico que se guardan en una variable global. La función de envío determina el tamaño de esta variable y envía el flujo de datos como una cadena de bits.

Al mismo tiempo se inicializa el dispositivo háptico. Si se detecta un error, la aplicación termina con un mensaje en pantalla.

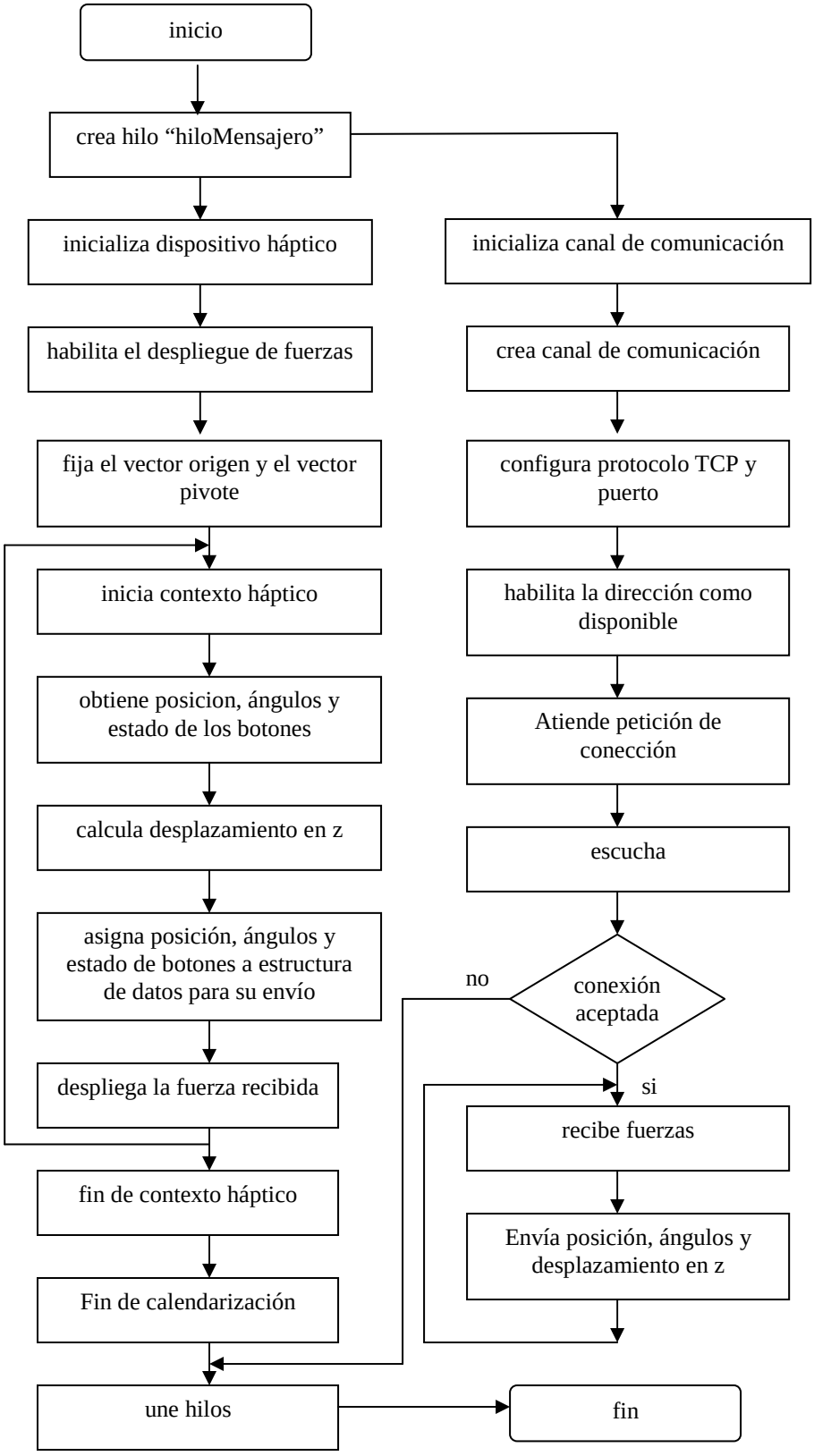

Figura 30. *Flujo de control del servidor*.

Se habilita el despliegue de fuerza, configura el calendarizador para el *servo loop*. Llama a la función que implementa el contexto háptico con la prioridad de ejecución que se toma por omisión, ya que no hay otras funciones calendarizadas. A continuación se enciende el calendarizador con su control de errores.

Por último, la función calendarizada provee un espacio para obtener posiciones, ángulos de la cadena de articulaciones y estados de los botones del dispositivo háptico.

Se reúnen los hilos y termina la aplicación.

Capítulo 4. Versión activa y resultados

En este capítulo se presentan los resultados del desempeño de la versión activa, es decir, con realimentación táctil, del Simulador RTUP del CCADET de la UNAM.

4.1 Modalidades

Se presentan dos modalidades integradas que sustituyen a la Interfaz Mecatrónica en su totalidad. El sistema completo (véase figura 31) consiste en el bloque de gráficos en conjunto con el bloque háptico formado por el servidor y el dispositivo de despliegue de fuerza. Los elementos físicos de interacción son los que constituyen al háptico, es decir, dos botones, el efector final y el brazo. A continuación se describen dos formas de configurar estos elementos.

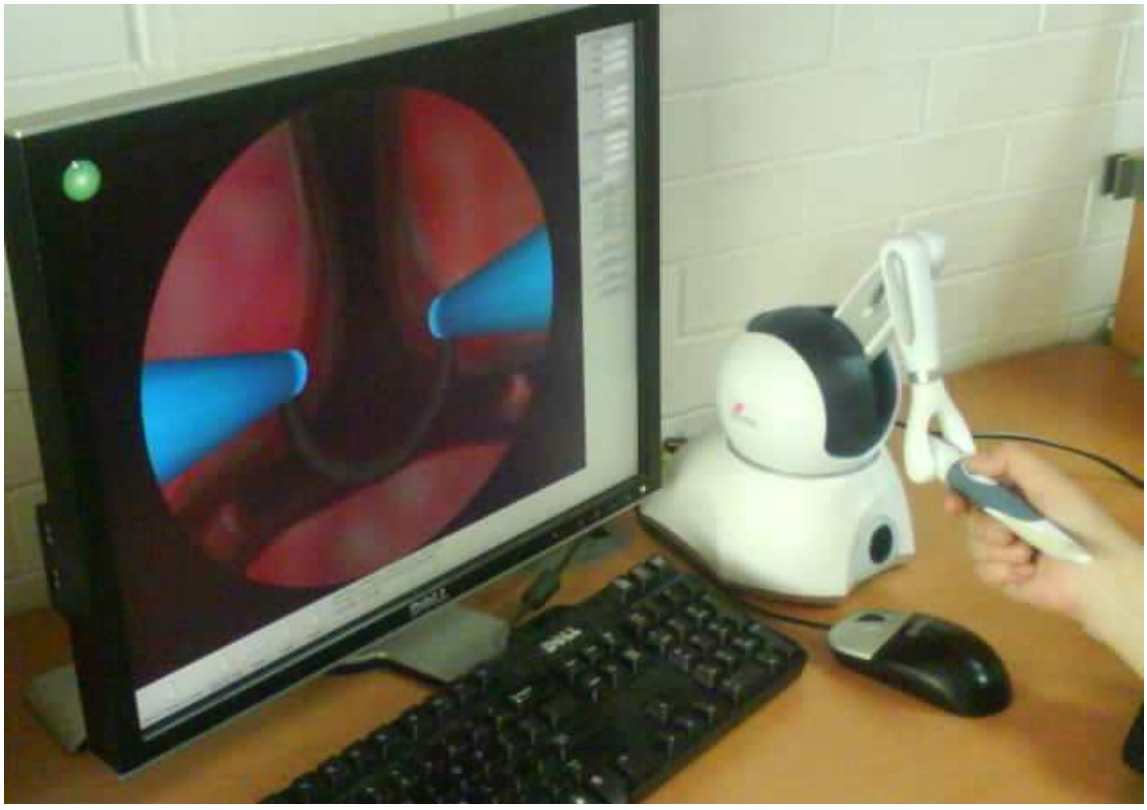

Figura 31. Simulación con el Omni. Laboratorio de Imágenes y Visualización, CCADET, UNAM 2010.

4.1.1 Omni con avance relativo

La primera opción para la configuración de los elementos del Omni se ocupa de la siguiente manera:

- Botón 1 presionado: bandera de corte
- Botón 2 presionado: bandera de coagulación
- Botones sin presionar: movimiento relativo, bandera de exploración
- Teclado de la computadora: movimiento absoluto
- Brazo del Omni: Asa de resección.

Esta modalidad permite una simulación aceptable. Entre las ventajas se puede mencionar: provee la sensación de tocar el tejido de la próstata, permite los grados de libertad necesarios para la navegación en el sistema virtual, estos movimientos corresponden a los visualizados en el Simulador. Sin embargo presenta las siguientes desventajas: Solo permite el movimiento relativo, el efector final no es similar al resectoscopio real, la navegación resulta difícil.

4.1.2 Omni con avance relativo y absoluto

La segunda opción consiste en la siguiente combinación de elementos:

- Botón 1 presionado: bandera de corte
- Botón 2 presionado: bandera de coagulación
- Ambos botones presionados: movimiento absoluto
- Botones sin presionar: movimiento relativo, bandera de exploración

Las características de esta modalidad son similares a las de la anterior. En este caso se tiene la ventaja de que ambos movimientos son proporcionados por el Omni, sin embargo la navegación resulta más complicada. En la tabla 2 se comparan las ventajas y desventajas de estas dos versiones:

	versiones	
	4.2.1	4.2.2
ventajas	- Provee sensación táctil - Soporta grados de libertad necesarios. - Movimientos coherentes con el sistema visual	- Provee sensación táctil - Soporta grados de libertad necesarios. - Movimientos coherentes con el sistema visual - Movimientos relativo y absoluto
desventajas	- Solo tiene movimiento relativo - El efector final no es similar al resectoscopio real - Navegación difícil	- El efector final no es similar al resectoscopio real - Navegación muy difícil

Tabla 2. *Comparación entre las versiones del sistema activo.*

Otro aspecto de interés es el análisis de los cuadros por segundo que resultan de la ejecución de la simulación, ya que una buena animación contiene 24 cuadros por segundo. A continuación se presentan los datos obtenidos (cuadros por segundo) de ejecutar la simulación en el orden en que se realiza una cirugía normal, es decir, la exploración de modelos anatómicos, el corte y coagulación del modelo que representa el tejido de la próstata. Se tomaron veinte muestras para cada caso durante el uso del sistema en cuatro casos posibles[1]:

a) simulación sin interfaz mecatrónica

Uso de CPU: 51%			Programa en memoria: 7200 kB						
19.70	17.72	18.70	18.81	18.72	18.70	18.43	17.46	17.72	17.19
16.93	16.75	17.73	18.15	18.83	18.72	17.79	18.81	17.73	18.72

Tabla 3. *Cuadros por segundo, simulación sin interfaz mecatrónica.*

b) simulación con interfaz USB

Uso de CPU: 58%			Programa en memoria: 7556kB						
19.78	15.28	13.90	15.30	15.02	16.73	15.05	15.52	15.07	15.05
16.73	16.24	14.73	15.52	14.76	14.55	15.07	15.52	14.76	14.12

Tabla 4. *Cuadros por segundo, simulación con interfaz USB.*

[1] Experimento realizado con un procesador AMD Dual Core x64, 3GHz, con 4 GB en RAM.

c) simulación con dispositivo Omni.

Uso de CPU: 96%			Programa en memoria: 9200 kB						
10.20	10.51	9.14	9.14	9.70	10.20	9.55	8.11	9.41	8.47
9.45	9.55	10.52	9.84	9.55	10.25	9.42	10.06	10.77	10.36

Tabla 5. *Cuadros por segundo, simulación con dispositivo Omni.*

d) simulación con dispositivo Omni en procesamiento remoto.

Uso de CPU: 58%			Programa en memoria: 7556 kB						
18.70	18.15	18.16	17.21	17.72	17.44	16.70	18.15	18.72	15.75
17.46	18.15	17.89	18.70	17.19	15.05	18.15	16.73	18.43	16.95

Tabla 6. *Cuadros por segundo, simulación con dispositivo Omni en procesamiento remoto.*

Casos	a)	b)	c)	d)
Promedio de cuadros por segundo (cps)	18.17	15.44	9.70	17.57
cps obtenidos/cps ideal	0.76	0.64	0.40	0.73

Tabla 7. *Comparción de cuadros por segundo para los casos a), b), c) y d).*

Observando las tablas 3 a la 7 se puede afirmar que el mejor caso para utilizar la versión activa del simulador con el dispositivo háptico Omni es utilizar el cliente en una computadora mientras que el servidor se debe procesar en otra, o bien en procesamiento paralelo. La modalidad factible es la que proporciona sólo el movimiento relativo debido a su mejor facilidad en su manipulación.

Conclusiones y trabajo a futuro

5.1 Conclusiones.

- Mediante la revisión de las referencias bibliográficas se verifica que la simulación de sensaciones táctiles en ambientes virtuales es posible mediante dispositivos mecatrónicos que soportan el despliegue de imágenes hápticas. Existen dos posibilidades para su integración, la primera, es que sean diseñados para propósitos específicos y la segunda es que se utilicen dispositivos hápticos comerciales.

- Se comprobó también que la comunicación entre procesos que utilizan recursos mutuamente incompatibles puede realizarse mediante el uso de canales de comunicación y que la interacción entre dos procesos que ejecutan ciclos infinitos se resuelve mediante la programación concurrente, si se cuenta con un solo procesador, o bien con la programación en paralelo si se cuentan con dos o más procesadores; ya que maneja los procesos de una forma independiente y los accesos a los valores de las variables en memoria se maneja en forma segura, con lo cual pueden estar los dos procesos en la ejecución de un mismo sistema.

- La integración de sensaciones hápticas al simulador de cirugía de próstata fue implementada exitosamente utilizando las tecnologías de programación desarrolladas y explicadas en este trabajo de tesis. La metodología utilizada fue el crear canales de comunicación dentro de hilos de control para lograr la interacción entre los datos provenientes de distintas fuentes, cada una con características diferentes.

- Se logró simular las habilidades básicas de una resección transuretral de próstata tal como la exploración de los modelos anatómicos de vejiga, uretra y próstata, así como la sensación de tocar tejido suave.

5.2 Trabajo a futuro.

Como investigaciones y aplicaciones posteriores, a partir de este trabajo, se pueden mencionar:

- La adaptación del háptico al bloque mecatrónico consistente en el acoplamiento del efector final hacia la punta del resectoscopio de la Interfaz Mecatrónica con la finalidad de que las fuerzas desplegadas sean transmitidas por medio de la camisa hacia el usuario. Así, el dispositivo mecatrónico proporcionará la ergonomía y las posiciones de la herramienta, mientras que el dispositivo háptico desplegará la imagen táctil de los modelos anatómicos.

- El desarrollo de una metodología que permita evaluar la importancia de tener sensaciones táctiles durante la simulación en base a utilizar la versión activa y la versión pasiva del simulador.

- El mejoramiento del bloque mecatrónico, de tal manera que contemple un sistema de realimentación de fuerzas, de tal manera que se tenga una versión activa sin dependencia de equipos comerciales.

- Mejorar la navegación de tal forma que los movimientos que se realizan en la simulación correspondan con los que se presentan en una cirugía real, es decir, crear un modelo gráfico parecido a la anatomía real de la uretra y restringir los movimientos desde la entrada de la uretra.

- La simulación de entrenamiento para otros procedimientos quirúrgicos de urología, así como la construcción de simuladores de entrenamiento para otras especialidades como es el caso de la ginecología.

Referencias bibliográficas

[1] Alonso Prieto M., Pérez Romero N., Silmi Moyano A., "Hiperplasia Benigna de próstata" en AAVV, *Libro del Residente de Urología*, al Cuidado de J. Castiñeiras Fernández, Madrid, España, 2007, pp. 997 – 999, 1001, 1003, 1007, 1008.

[2] Altamirano del Monte, Felipe, *Interfaz Mecatrónica para un Simulador de Cirugía de Próstata*, Tesis de Maestría en Ingeniería Eléctrica – Instrumentación, UNAM – Programa de Maestría y Doctorado, México, 2007.

[3] Colin Laplace. Bloodshed Dev-C++ [en línea]. [Consulta: febrero de 2009]. Disponible en: <http://www.bloodshed.net/dev/devcpp.html>.

[4] Cutkosky Mark R., Howe Robert D., Provancher William R., "Force and Tactile Sensors" en AAVV, *Springer Handbook of Robotics*, al cuidado de Bruno Siciliano y Ossana Khatib, Stürtz Gimbh, Würzburg. 2008, p 456.

[5] Delingette Hervé, "Toward Realistic Soft-Tissue Modeling in Medical Simulator", *Proceedings of the IEEE*, vol 86, no. 3, marzo 1998, pp. 512 – 516.

[6] Fundación Wikimedia. Próstata.jpg [en línea]. [Consulta: febrero de 2010]. Dsiponible en: <http://es.wikipedia.org/wiki/archivo:Prostata.jpg>.

[7] Gomes Paula, Barret Adrian, Timoney Anthony, Davies Brian., "A Computer-Assisted Training/Monitoring System form TURP Structure and Design". *IEEE Transactions on information technology in biomedicine*, vol 3, no. 4, December 1999, pp. 242, 243.

[8] Hannaford Blake, Okamura Allison M., "Haptics" en AAVV, *Springer Handbook of Robotics*, al cuidado de Bruno Siciliano y Ossana Khatib, Stürtz Gimbh, Würzburg. 2008, pp. 719 – 721, 727,728, 731.

[9] Hearn Donald, Baker Pauline M.,Computer Graphics with OpenGL. Third Edition. Pearson – Prentice Hall USA, 2004, p 32.

[10] Immersion Corporation. Immersion [en línea]. [Consulta: enero de 2009]. Disponible en: <http://www.immersion.com/>.

[11] Imperial College London. TURP-TM Training system for Endoscopic Soft Tissue Surgery [en línea]. [Consulta: enero de 2009]. Disponible en: <http://www.imperial.ac.uk/mechatronicsinmedicine/research/turptrainer>.

[12] John Hopkins University. Research/Rehabilitation – Haptics [en línea]. [Consulta: mayo de 2010]. Disponible en: <https://haptics.lcsr.jhu.edu/Research/Rehab>.

[13] Klatzky R.L., Lederman S.J., "Haptic Perception", *Encyclopedia of Cognitive Science*, MacMillan Press, p. 508.

[14] Leithold Louis,El cálculo con Geometría Analítica. Quinta Edición. Editorial Harla.México 1987, p1455.

[15] Lira Berra, Erick, *Implementación de un modelo deformable de masas y resortes en una GPU para un simulador de cirugía de próstata,* Tesis de Licenciatura en Ingeniería en Computación, UNAM – Facultad de Ingeniería, Méixico, 2008, p. 30.

[16] Márquez García, José Manuel.,Unix. Programación avanzada. RA-MA Editorial, España, 1993. p 331 – 334.

[17] Microsoft Corporation. Visual Studio [en línea]. [Consulta: febrero de 2009]. Disponible en: <http://msdn.microsoft.com/es-es/vstudio/default.aspx>.

[18] Network Programming, Linux Socket [en línea]. [Consulta: junio de 2009]. Disponible en: <http://www.tenouk.com/Module39.html>.

[19] Olympus America Inc. Resectoscopios [en línea]. [Consulta: febrero de 2009]. Disponible en: <http://www.olympusmexico.com.mx/spanish/msg/msg_product_detail_esp.asp?D=4&s=10&c=46&g=253>.

[20] OpenGL.org. The Industry Standard for High Performance Graphics [en línea]. [Consulta: noviembre de 2008]. Disponible en: <http://www.opengl.org>.

[21] Padilla Castañeda M. A., Arámbula Cosío F., "Deformable model of the prostate for TURP surgery simulation", *Computers and Graphics*, no. 28, 2004, pp. 767 – 777.

[22] Real Academia Española. Diccionario de la lengua española – Vigésima segunda edición [en línea]. [Consulta: diciembre de 2008]. Disponible en: <http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=simulador>.

[23] Reyes Ramírez, Bartolomé, *Validación de un simulador para entrenamiento en cirugía de próstata,* Tesis de Maestría en Ingeniería Eléctrica, Instrumentación, UNAM – Facultad de Ingeniería, México, 2010, pp. 13, 18, 27, 28.

[24] Robbins Kay, Robbins Steven,Unix Programación práctica. Guía para la concurrencia, la comunicación y los multihilos., Prentice Hall Hispanoamericana, México 1997, p 5, 10.

[25] Sensable Technologies Inc. Sensable [en línea]. [Consulta: marzo de 2009]. Disponible en: <http://www.sensable.com/>.

[26] SensAble Technologies, Inc,Programmer's Guide, Open Haptics Toolkit Version 2.0. USA, 2005, pp. 2-2, 2-3.

[27] Serway Raymond A, Jewett John W,Physics for Scientists and Engineers. 6th Edition. Brooks/Cole, Cenage Learning, USA, 2006, p 113, 190.

[28] Simbionix. Uro Mentor TM [en línea]. [Consulta: marzo de 2009]. Disponible en: <http://www.simbionix.com/URO_Mentot.html>.

[29] Sweet Robert M., “Review of Traners for Transuretrhal Resection of the Prostate Skills”, *Journal of Endourology*, vol 21, no.3, march 2007, pp. 280 – 282.

[30] Tavakoli, M., Patel, R. V., Moallem, M., Aziminejad, A., Haptics for Teleoperated Surgical Robotics Systems, World Scientific Publishing, Singapore, 2008, pp 9, 10, 11.

[31] The Internet Engineering Task Force. IETF [en línea]. [Consulta: marzo de 2009]. Disponible en: <http://www.ietf.org/>.

[32] The University of Washington. UW Virtual TURP Surgical Simulator [en línea]. [Consulta: diciembre de 2008]. Disponible en: <http://www.hitl.washington.edu/people/tmk/TURPsim/>.

[33] Teodoro Vite, Sergio, *Modelado de un ambiente virtual para un sistema de simulación de cirugía de próstata*, Tesis de Licenciatura en Ingeniería en Computación, UNAM – Facultad de Ingeniería – CCADET, México, 2008, pp. III, 30.

[34] Wright, Richard Jr, Lipchak Benjamin, Haemel Nicholas,OpenGL Superbible. 4a Edition, Comprehensive Tutorial and Reference. Addison Wesley, USA, 2007, p 14.