Implementación de una API para la interacción del guante P5 con entornos de realidad virtual desarrollados en Java y Java 3D

Antonio José Escallón D.
aescallon@puj.edu.co

Ricardo Antonio Parra S.
rparra@puj.edu.co

Francisco Julián Herrera B.
fjherbo@hotmail.com

Andrés Adolfo Navarro Newball
anavarro@puj.edu.co

César Augusto Marín Tobón
camarin@emcali.net.co
Grupo Destino - Pontificia Universidad Javeriana – Cali, Colombia

Fecha de recepción: 22-1-2007 Fecha de selección: 8-10-2007 Fecha de aceptación: 30-8-2007

ABSTRACT

This paper presents a specific API's implementation process, this API allows the interaction of the P5 glove with virtual reality environments implemented in the programming language Java and its tool Java 3D. Also, an example program is implemented using the API. Then, a CPU and physical memory consumption tests is performed with the example. Finally, the conclusions obtained are specified.

KEY WORDS

API, P5 glove, programming language, virtual environments, Java, Java 3D.

RESUMEN

Este artículo presenta el proceso de implementación de una API (Application Programming Interface) que permite la interacción del guante P5 de Essential Reality[1] con un entorno virtual desarrollado en el lenguaje de programación Java y su librería Java 3D.[2] Por otra parte, se describe un ejemplo implementado, haciendo uso de la API en cuestión. Con base en este ejemplo se presentan los resultados de la ejecución de pruebas de requerimientos de recursos físicos como la CPU y memoria física. Finalmente, se especifican las conclusiones y resultados obtenidos.

PALABRAS CLAVE

API, Guante P5, Lenguaje de programación, Ambientes Virtuales, Java, Java3D

Clasificación Colciencias: Tipo 1

INTRODUCCIÓN

La realidad virtual se define como la ilusión de participación en un mundo sintético, más que la observación externa del mismo; adicionalmente, la realidad virtual es una experiencia inmersiva y multi – sensorial.[3] Al mismo tiempo, la interacción es un mecanismo para construir un diálogo desde varios dispositivos de control y aplicar ese diálogo a un sistema. Este mecanismo se divide en dos partes, a saber: primero, capturar las entradas del dispositivo e interpretarlas. Segundo, tomar la información significativa para el sistema y filtrar datos que no realizan acción. Adicionalmente, los propósitos de la interacción son traducir las acciones del usuario en cambios dentro del ambiente virtual, pasar comandos al ambiente y proveer información de entrada.[4]

Este artículo describe primero el guante P5 y el simulador WESST – OT (Web Environment For Surgical Skills Training in Otolaryngology);[5] luego explica el concepto de API, y continúa con la descripción del proceso de desarrollo de la API propuesta, que sirve para programar y configurar el guante P5 desde Java y Java 3D. Finalmente, explica las pruebas realizadas y las conclusiones. El API presentado fue desarrollado con el fin de implementar un mecanismo de interacción con WESST – OT .

EL GUANTE P5

El P5 (Figura 1) es un dispositivo en forma de guante, que posee las propiedades de rastreo remoto y censores curvos; además, genera en el usuario la impresión de interactuar con ambientes tridimensionales virtuales tales como juegos, sitios Web y software educacional; la Tabla 1 resume sus principales características.[1]

API

Una API (Aplication Programming Interface - Interfaz de Programación de Aplicaciones) es un protocolo de comunicación entre componentes de software y hardware; éste representa un método para conseguir abstracción en la interoperabilidad de la programación; así, uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general que beneficia a los programadores, les brinda funcionalidad y evita el trabajo de programar todo desde el principio.[6]

Figura 1. Guante P5

Tabla 1. Características del guante P5.

Característica	Valor
Peso	4.5 onzas
Grados de libertad	6 (X, Y, Z, yaw, pitch y roll) 0.125 pulgadas de resolución
Movilidad	Basada en sensores curvos y rastreo óptico y un receptor de control infrarrojo
Conexión	Puerto USB1.1
Sensores de los dedos	Medición independiente de los 5 dedos de 0 a 90 grados
Fuerzas de respuesta	No

WESST – OT[7]

WESST - OT es un entorno diseñado para ayudar en la práctica de las habilidades requeridas para la cirugía endoscópica en senos paranasales. Actualmente, la forma de interacción con este entorno requiere la utilización de un panel de control manejado por el ratón (Figura 2); así, el mecanismo de interacción con este entorno se constituye de una forma poco natural a los especialistas practicantes.

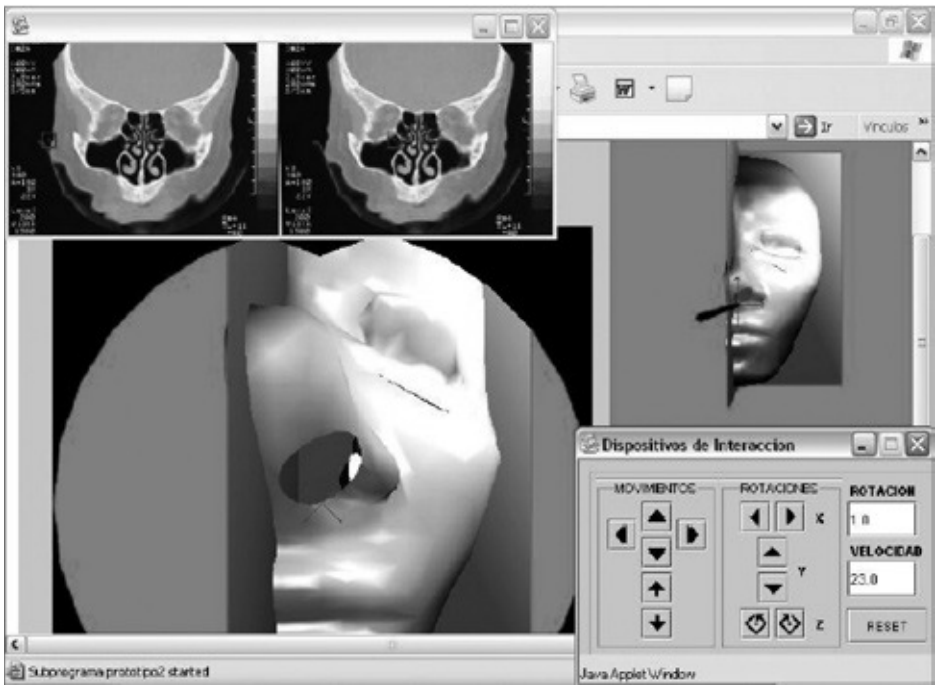

Figura 2. WESST - OT

DESARROLLO DE LA API

Actualmente, la API desarrollada posee la capacidad de inicializar el guante, de calcular su posición en el espacio y de procesar el doblamiento de los dedos; además, es compatible con la API JAVA 3D.[4] Para su desarrollo se tomó como base la librería dinámica "dll" que provee el fabricante del guante P5[1] y se desarrolló la API en la clase P5DLLw que actúa como un manejador intermediario entre la librería dinámica original y los programas que son implementados en Java y Java 3D.

La clase P5DLLw tiene los siguientes atributos:[4]

- M_nNumP5, mantiene el número de guantes conectados encontrados.
- ID, es el identificador asignado al guante
- m_fx, indica la posición en el eje x
- m_fy, indica la posición en el eje y
- m_fz, indica la posición en el eje z
- m_fyaw, indica el ángulo del guante cuando realiza el movimiento de la Figura 3a
- m_fpitch, indica el ángulo del guante cuando realiza el movimiento de la Figura 3b
- m_froll, indica el ángulo del guante cuando realiza el movimiento de la Figura 3c

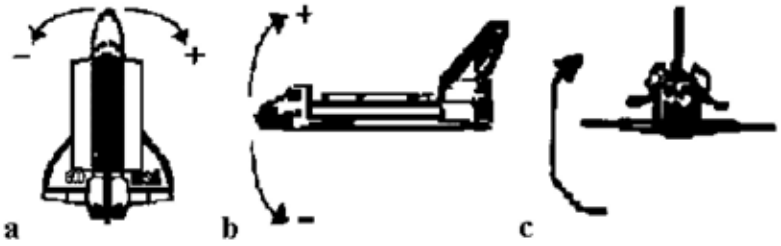

Figura 3. a) yaw, b) pitch c) roll

- ThumbValue, indexValue, middleValue, ringValue, pinkyValue, indican el nivel de doblamiento de los dedos pulgar, índice, corazón, anular y meñique respectivamente
- lastErrorCode, indica el último tipo de error ocurrido
- lastErrorDevice, indica el identificador del guante al que corresponde el último error

Además, la clase P5DLLw implementa los siguientes métodos:

- P5_Init(void) à boolean, reserva la memoria requerida por el guante cuando está conectado e inicializa sus variables. Retorna TRUE o FALSE, dependiendo de si el guante se pudo o no inicializar.
- P5_GetMouseState(int P5Id) à boolean, verifica en que estado se encuentra el guante cuyo identi-

ficador es P5Id. Retorna TRUE si el guante está funcionando como Mouse y FALSE si el guante está deshabilitado como Mouse.

- P5_SetMouseState(int P5Id, boolean state) à void, permite especificar el modo del guante con identificador P5Id, enviando el valor de la variable state como TRUE si se quiere usar el guante como Mouse o, de lo contrario, como FALSE.
- P5_GetLastError(void) à boolean, permite conocer el último error que ha ocurrido. Retorna TRUE si ha ocurrido un error o FALSE en caso contrario. Para conocer el código del error se debe ver el valor de la variable lastErrorCode y para saber el identificador del guante que lo generó se debe ver el valor de la variable lastErrorDevice.
- P5_BendInit(int P5Id) à void, Inicia el procesamiento de la información que proviene del doblamiento de los dedos a través del guante identificado con P5Id.
- P5_BendSetClickSensitivity(int finger, char value) à void, especifica el grado de sensibilidad para el dedo "finger", siendo 0: pulgar, 1: índice, 2: corazón, 3: anular y 4: meñique.
- P5_BendProcess(void) à void, procesa la información del doblamiento de los dedos.
- UpdateBendData(void) à void, actualiza las variables correspondientes al grado de doblamiento de cada dedo del guante. Normalmente, el valor de cada dedo varía entre 0-63, siendo 63 cuando el dedo está totalmente doblado.
- P5_MotionInit(int P5Id) à void, Inicia el procesamiento de la información que proviene del movimiento del guante identificado con P5Id.
- P5_MotionSetClipRegion(int xstart, int xend, int ystart, int yend, int zstart, int zend) à void, permite especificar los límites de movimiento del guante en los ejes x, y, z.
- P5_MotionProcess(void) à void, procesa la información del movimiento del guante.
- UpdateMotionData(void) à void, actualiza las variables correspondientes al movimiento del guante. El valor en cada eje varía entre xstart-xend, ystart-yend y zstart-zend.

Adicionalmente, se implementó una librería dinámica que actúa como intermediaria entre la librería original del guante P5 (P5.dll) y la clase implementada (P5DLLw); ésta hace uso de la interfaz JNI, que permite comunicar los lenguajes de programación Java y C++. La interfaz nativa de Java (JNI por sus siglas en inglés) es una herramienta que provee la plataforma Java. Las aplicaciones que utilizan JNI pueden incorporar código escrito en los lenguajes de programación C y C++, tanto como código escrito en el lenguaje de programación Java, es decir, JNI permite comunicar los lenguajes de programación C y C++ con Java.[8] La Figura 4 muestra la arquitectura de la API implementada en Java. Aquí se observa que la librería P5dll hace uso de los drivers del guante P5; además, JNI comunica la librería P5dll (en C++) con el API implementado

en la librería P5DLLw a través de un wrapper. Finalmente, la API puede ser llamada desde una aplicación en Java o Java 3D y es portable entre las plataformas Linux y Windows.

La API fue implementada en el lenguaje de programación Java. Este lenguaje fue seleccionado con el fin de asegurar la compatibilidad de la API con la tecnología utilizada en el Simulador de Otorrinolaringología. Así, para utilizar el guante P5 en aplicaciones implementadas en Java y ambientes virtuales implementados en Java 3D sólo se requiere ubicar la librería P5DLLw.dll en la misma carpeta de la librería original (que por defecto es la carpeta System32 dentro de la carpeta Windows) o en la misma carpeta de la aplicación que hace llamado a los métodos de la clase Java P5DLLw.

PRUEBAS Y RESULTADOS

Para facilitar la comprensión del funcionamiento de la API, se implementó un ejemplo en Java utilizando la API de Java 3D.[2] Este ejemplo está incluido en las demostraciones que trae el instalador de Java 3D[2] y fue modificado para que funcionara con el guante P5 a través de la API implementada en este proyecto. Esta aplicación consiste en manipular una mano de cuatro dedos que tiene tres estados posibles (abierta, cerrada y a medio cerrar) y cambia de forma de acuerdo con el nivel de doblamiento de los dedos a través del guante (Figura 5).

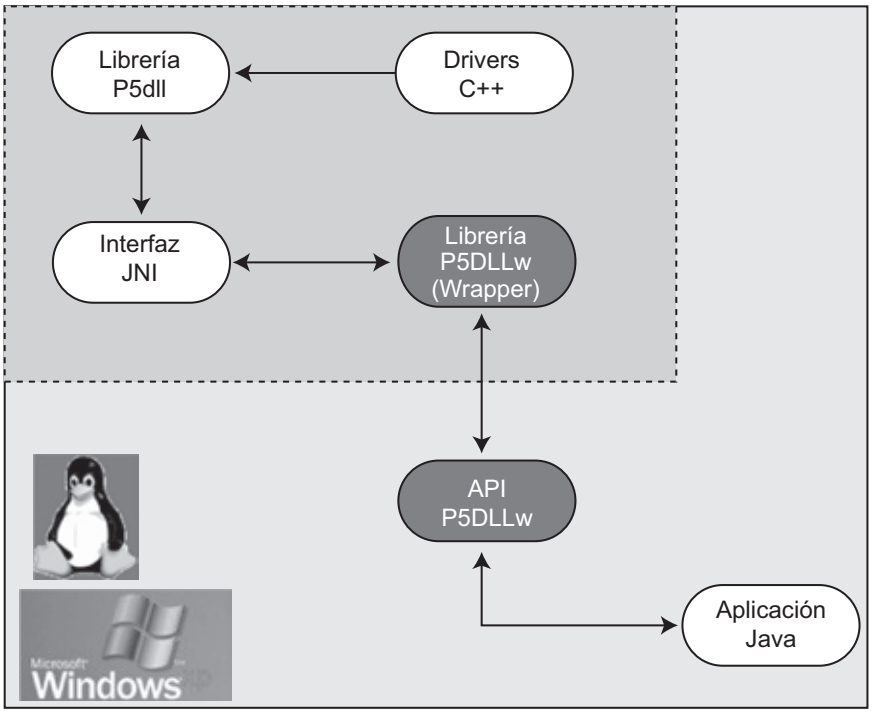

Figura 4. Arquitectura planteada[4]

Figura 5. Imagen del ejemplo

Como complemento, se hizo una prueba de carga. Java posee una máquina virtual que se encarga de interpretar y ejecutar las instrucciones generadas por el compilador, haciendo que la carga de éste cree un retraso adicional al retraso en renderizar o cargar los objetos tridimensionales en Java 3D. Esta prueba se realizó con el objetivo de tomar datos para determinar el porcentaje de carga de CPU y de memoria RAM ocasionada por una aplicación implementada en Java utilizando la API de Java 3D que usa el guante P5. En la Tabla 2 se aprecian los resultados de las pruebas de carga del ejemplo antes de la modificación que permite el uso del guante P5, y en la Tabla 3 se observa la misma prueba realizada a la misma aplicación con las modificaciones necesarias para usar el guante P5.[4]

Los resultados obtenidos fueron:

Tabla 2. Prueba de carga del ejemplo sin usar las funciones de la API

	Especificaciones técnicas	Carga de CPU (%)	Carga de memoria física (%)
Computador A	Windows XP Procesador Athlon XP de 2,0 GHz 512 MB de RAM	92,38	50,02
Computador B	Windows XP Procesador Pentium IV de 1,4 GHz 256 MB de RAM	89,12	76,23

Tabla 3. Prueba de carga del ejemplo usando la API

	Especificaciones técnicas	Carga de CPU (%)	Carga de memoria física (%)
Computador A	Windows XP Procesador Athlon XP de 2,0 GHz 512 MB de RAM	94,68	58,21
Computador B	Windows XP Procesador Pentium IV de 1,4 GHz256 MB de RAM	98,16	80,51

Se puede observar que se presentó un aumento significativo en los resultados de las pruebas de carga realizadas antes y después de utilizar la API en el ejemplo. Este aumento en la carga puede ser atribuido al ciclo que se ocupa de inspeccionar los cambios en el estado del guante P5 durante la ejecución de la aplicación. Aquí los valores obtenidos fueron tomados poco tiempo después de iniciadas las aplicaciones con el fin de esperar hasta que el sistema se estabilizara.

CONCLUSIONES

El trabajo realizado puede aportar al mejoramiento del grado de inmersión de aplicaciones en las que se presenten entornos virtuales al usuario final (por ejemplo, WESST – OT), pues permite la interacción directa de la mano del usuario y va más allá del ratón. Este es un paso inicial para que en el futuro se utilice un guante que brinde realimentación táctil. La API en Java permite que el desarrollador utilice Java 3D como lenguaje opcional para la creación de objetos tridimensionales.

La arquitectura planteada facilita el mejoramiento de la API y la implementación de nuevas librerías dinámicas que soporten el uso del guante P5 en múltiples sistemas operativos. Para esto se debe reemplazar la librería dinámica P5DLLw.dll por la librería dinámica correspondiente, de acuerdo con el sistema operativo en el que se establezca.

Los ejemplos indicados facilitan la comprensión del uso de la API en aplicaciones implementadas con Java y Java 3D. En dichos ejemplos se utilizan la mayoría de los métodos o funciones de la API.

El uso del guante P5 como mecanismo de interacción no se ha incorporado al simulador WESST-OT todavía. Sin embargo, en el futuro se espera eliminar el panel de control (Figura 2) para dar cabida a una visualización en tres dimensiones de un endoscopio virtual que pueda ser manipulado por el guante y que permita interactuar con la anatomía desplegada.

BIBLIOGRAFÍA

1. P5 Glove. http://www.vrealities.com/P5.html. visitado 2006.
2. D Selman. Java 3D Programming. Manning Publications. 2002.
3. R.A. Earnshaw, M.A. Gigante, H. Jones. Virtual Reality Systems. Academic Press, 1993.
4. A.J. Escallón, R.A. Parra. Implementación de una API para la interacción del guante P5 con entornos de realidad virtual implementados en Java y Java 3D. Trabajo de grado para Ingeniería de sistemas y computación. Pontificia Universidad Javeriana Cali, 2006.
5. C.J. Hernández, Análisis, Diseño e Implementación del Prototipo de un Entorno de Práctica de Habilidades Quirúrgicas en Otorrinolaringología, Tesis de Pregrado, Pontificia Universidad Javeriana, 2004.
6. API. http://www.whatis.com. Visitado 2006.
7. A.A. Navarro Newball, C.J. Hernández, J.A. Vélez, L.E. Múnera, G.B. García, C.A. Gamboa, A.J. Reyes. Virtual Surgical Telesimulations in Otolaryngology. Studies in Health Technology

and Informatics. Vol. 111. pp 353 – 355. IOS Press, 2005.

8. E. Gabrilovich. JNI- C++ Integration made easy: Extremely versatile interfaces like the java jni also tend to be extremely cumbersome, as a rule. The authors have found a way to break that rule. Volumen 19 Número 1, C/C++ Users Journal p10-21, 2001.

CURRÍCULOS

Antonio José Escallón D. Ingeniero de Sistemas y Computación egresado de la Pontificia Universidad Javeriana en Cali. Perteneció al grupo de investigación *Destino*, de la Facultad de Ingeniería durante el desarrollo de su trabajo de grado.

Ricardo Antonio Parra S. Ingeniero de Sistemas y Computación, egresado de la Pontificia Universidad Javeriana en Cali. Perteneció al grupo de investigación *Destino*, de la Facultad de Ingeniería durante el desarrollo de su trabajo de grado.

Francisco J. Herrera Botero. Ingeniero de Sistemas y Computación egresado de la Pontificia Universidad Javeriana en Cali, tiene una especialización en Redes y Comunicación (CCNA CISCO) de la Pontificia Universidad Javeriana - Cali. Se desempeñó como Asistente de Investigación (Programa Jóvenes Investigadores de Colciencias) en la Pontificia Universidad Javeriana – Cali. Además es fundador y gerente de la empresa de base tecnológica Soluciones Virtuales. Actualmente hace parte del grupo de investigación *Destino* y es profesor hora cátedra de la Universidad ICESI.

Andrés A. Navarro Newball. Ingeniero de Sistemas y Computación egresado de la Pontificia Universidad Javeriana en Cali, tiene un MSc in Computer Graphics and Virtual Environments de la Universidad de Hull en Inglaterra y una especialización en Redes y Comunicación de la Universidad Icesi en Cali. Además, formó parte Centro de Telemedicina de Colombia y fue profesor hora cátedra de la Universidad Icesi. Actualmente se desempeña como profesor de la Pontificia Universidad Javeriana – Cali, donde coordina el grupo de investigación *Destino* y está realizando estudios de PhD en la Universidad de Otago, Nueva Zelanda.

César Augusto Marín Tobón. Ingeniero Electrónico de la Pontificia Universidad Javeriana – Cali; se desempeñó como miembro importante del proyecto T@L@MED de la Universidad Santiago de Cali; actualmente realiza estudios de Maestría en la Universidad Politécnica de Valencia-España.