INVESTIGACIÓN & DESARROLLO, No. **16**, Vol. **1**: 86 – 101 (2016)
ISSN 2518-4431

FiMAN: SISTEMA COMPUTARIZADO PARA ANÁLISIS DE MOVIMIENTOS DIGITALES

FiMAN: A COMPUTER SYSTEM FOR FINGER MOTION ANALYSIS

Miguel Carballo, Marcél Barrero y Alex Villazón
Centro de Investigaciones de Nuevas Tecnologías Informáticas (CINTI)
Universidad Privada Boliviana
avillazon@upb.edu
(Recibido el 10 abril 2016, aceptado para publicación el 10 junio 2016)

RESUMEN

La reciente tecnología de dispositivos y sensores infrarrojos han abierto nuevas posibilidades para el desarrollo de software relacionado con el área de análisis de movimientos corporales. Esta tecnología puede ser utilizada en áreas como la salud, la enseñanza, la realidad virtual y aumentada, control de robots y entretenimiento. En este artículo presentamos el desarrollo de FiMAN, un sistema que permite visualizar los movimientos detallados de las manos en tiempo real y en 3D utilizando el dispositivo infrarrojo Leap Motion para la captura de datos, permitiendo la realización de grabaciones y reproducciones de movimientos para su posterior análisis. Para su desarrollo fue necesaria la división modular de los componentes visuales, estructurales, analíticos, estadísticos y de sonido, además de la captura y manipulación de datos. Gracias a su diseño modular, se obtuvo un entorno genérico de desarrollo para aplicaciones especializadas en movimientos digitales. Como Estudio de Caso, se utilizó el sistema al área de la educación musical, específicamente a la enseñanza de la técnica del piano, para lo cual se extendió FiMAN con un teclado virtual y manejo de sonido. El uso real de FiMAN en este escenario, demostró su modularidad y extensibilidad a diferentes áreas relacionadas al análisis de movimientos digitales.

ABSTRACT

The introduction in the market of new devices and sensors with infrared technology has opened new opportunities for the development of specialized software for body motion analysis, in areas such as health, virtual and augmented reality, robot control and entertainment. In this article we present FiMAN, a computer system that allows capturing and visualizing detailed finger and hand motion in 3D and real time, using a Leap Motion sensor. FiMAN allows to record, store and replay finger motion for post-mortem analysis. We developed a modular system using components for visual, structural, analysis, statistical and sound management, which include also data capture a manipulation. Thanks to its modular design, FiMAN is a generic framework for the development of specialized motion analysis applications. As a Case Study, we extended FiMAN as a tool for musical education, namely to support teaching of piano techniques, through a virtual piano keyboard with a sound module. The real life usage of FiMAN, demonstrated its modularity and extensibility for its applicability in different areas related to motion analysis.

Keywords: Finger Motion Analysis, Sensor, Leap Motion, Virtual Keyboard.
Palabras Clave: Análisis Movimiento Digital, Sensor, Leap Motion, Teclado Virtual.

1. INTRODUCCIÓN

Durante los últimos años se ha desarrollado tecnología que permite el control a distancia a través del análisis de movimientos. Esta tecnología se ha aplicado, sobre todo al área del entretenimiento (video juegos), en los controles de consolas de juego como el Wii de Nintendo[1] y el Xbox de Microsoft con el sensor Kinect[2]. Sin embargo, más allá de las aplicaciones de carácter lúdico, aún no se ha aprovechado todo el potencial que esta tecnología puede tener en otras áreas con mayor impacto social. Recientemente, varios investigadores han empezado a utilizar estos sensores de movimiento en áreas como la salud [1], la enseñanza [2], la realidad virtual [3] y el control de robots con gestos [4].

En general, el análisis de movimiento requiere de dos componentes que pueden ser considerados por separado: el *hardware* (sensores que generan datos) y el *software* (programas que transformen los datos en información útil). Esto permite el uso de un mismo hardware para el desarrollo de software en diferentes áreas de investigación. Asimismo, siendo el hardware cada vez más accesible, tanto en disponibilidad como en costos, se tiene la oportunidad de desarrollar sistemas de software que permitan visualizar en tiempo real y en 3D, analizar, grabar y reproducir movimientos, para su aplicación en nuevas áreas de investigación o para el desarrollo de productos comerciales.

[1] http://www.wii.com/
[2] http://www.microsoft.com/en-us/kinectforwindows/develop/

Dentro del área investigación que estudia la interacción hombre-computador (Human-Computer Interaction – HCI), el análisis de movimientos de manos y dedos tiene gran importancia, por ser una de las formas más comunes de interacción con elementos reales o virtuales [5]. Uno de los sensores que permite ésta interacción es el Leap Motion[3], que es un dispositivo especializado en la captura de datos de manos y dedos. La tecnología Leap Motion, sin embargo, se limita al hardware del sensor, dejando a los programadores la responsabilidad de implementar aplicaciones especializadas [6].

Este artículo presenta el sistema FiMAN que permite visualizar los movimientos detallados de las manos en tiempo real y 3D capturados por un sensor Leap Motion. FiMAN permite además, grabar y reproducir los movimientos para su análisis posterior. Su diseño modular permite utilizarlo como una base extensible, donde se pueden añadir nuevos objetos o componentes de análisis; resultando un entorno genérico de desarrollo para aplicaciones especializadas en movimientos digitales. Ésta virtualización de movimientos en tiempo real permite crear la representación gráfica de movimientos de la mano desde diferentes ángulos y velocidades, lo que a su vez permite al usuario observar detalladamente los movimientos registrados desde la perspectiva y la velocidad que necesite.

Como Estudio de Caso, se ha extendido FiMAN al área de educación para analizar el movimiento de los dedos en las técnicas básicas de interpretación de piano, simulando un piano virtual. Esto permite a los maestros de piano contar con una herramienta tecnológica completamente nueva para evaluar las fortalezas y debilidades técnicas de los estudiantes en base al análisis de las representaciones gráficas y los datos de movimientos registrados por el sensor Leap Motion. Asimismo, la rápida detección de dificultades técnicas y en el planteamiento de nuevos métodos y ejercicios correctivos permite aumentar la eficiencia en el desarrollo de la técnica del estudiante. Si bien actualmente existen sistemas de análisis de movimiento para la dirección de orquesta [7] y evaluaciones preliminares de instrumentos musicales digitales o DMI (Digital Musical Instruments) basados en la tecnología de Leap Motion [8], no existen herramientas tecnológicas similares a FiMAN que permitan analizar los movimientos y posición de las manos y dedos para corregir y mejorar las técnicas de interpretación del piano, lo que en sí representa una innovación. FiMAN demuestra así su potencial para aplicarlo en otras áreas que requieran del análisis de movimiento digital, como ser la salud (recuperación en fisioterapia), responsabilidad social (aprendizaje de lenguaje de señas) u otras.

Este artículo está estructurado de la siguiente manera: La Sección 2 presenta la Arquitectura del Sistema donde se describe la interacción de cada uno de sus módulos. La Sección 2 muestra la implementación de FiMAN donde además se describe el hardware y software del sensor Leap Motion. En la Sección 3 se detalla el Estudio de Caso con la extensión de FiMAN para su aplicación en la educación musical. Finalmente, la Sección 4 concluye el artículo.

2. ARQUITECTURA DE FiMAN

FiMAN es un sistema que permite al usuario visualizar una representación gráfica los movimientos de las manos en la pantalla capturados mediante un sensor, además de procesar los datos para brindar al usuario opciones como almacenar los datos de movimientos para su posterior visualización, exportar los datos y mostrar gráficos estadísticos.

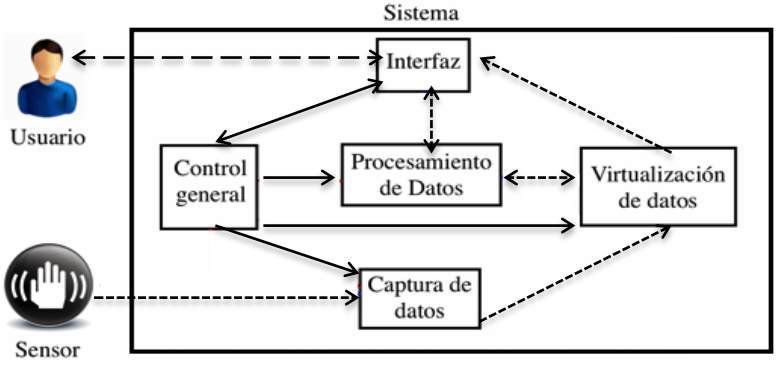

Figura 1 - Estructura general del Sistema FiMAN.

En la **Figura 1** se muestra un esquema general del sistema, donde se puede observar los componentes más importantes del mismo interactuando entre sí, además de interactuar con el usuario y el sensor. Las flechas con línea punteada indican el flujo de los datos capturados con el sensor Leap Motion. Las flechas con línea continua indican el flujo de datos para coordinación de funcionamiento, petición de servicios y control general. La flecha con línea segmentada indica la interacción entre el usuario y el sistema, donde principalmente el usuario selecciona opciones del sistema y el

[3] https://www.leapmotion.com/product/desktop

este procede a mostrar imágenes o generar archivos. Los datos capturados del sensor son enviados al módulo de Virtualización de datos, los cuales son procesados para construir una representación en 3D de las manos. Los datos son luego enviados a un interfaz de visualización, con la cual el usuario puede interactuar gracias a un escenario virtual. El sistema es controlado por un módulo de Control General que se ocupa de coordinar las acciones entre los otros módulos.

2.1 Arquitectura General

La arquitectura general del sistema consta de 7 módulos con las siguientes funcionalidades:

- **Coordinación General:** Inicializa todo el sistema y coordina las interacciones de los otros módulos
- **Captura de Datos:** Utiliza un sistema de eventos para recibir datos del sensor Leap Motion
- **Interfaz Gráfica y Producción Visual:** Permite la visualización de objetos en un escenario virtual en 3D y la interacción del usuario con el sistema.
- **Representación Virtual de Componentes:** Permite la modelización de objetos que son visualizados por el sistema, que incluyen las manos y cualquier objeto definido por el usuario. La modelización se encuentra representada en su totalidad en un Escenario Virtual que contiene los datos de posición de las manos (dedos y muñecas).
- **Modificación Visual:** Permite la modificación interactiva de los parámetros de visualización (diferentes vistas para modificar el foco, ángulo y zoom)
- **Manejo de Escenarios Virtuales:** Define un formato propio para guardar en archivos y reproducir todos los movimientos con todos los detalles para su posterior utilización. Además, permite generar reportes estadísticos de análisis de cada mano y dedos de los diferentes ejes en 3D, i.e. coordenadas *(x,y,z)*, con respecto al tiempo en base a datos de Escenarios Virtuales previamente guardados.
- **Producción de Sonido:** Brinda la posibilidad de agregar una funcionalidad extra a sistemas desarrollados en base a los módulos del sistema básico: Permite asociar sonidos a los componentes visualizados con respecto a posiciones o movimientos.

Los módulos interactúan entre sí intercambiando datos y coordinando tareas. En la **Figura 2** se muestra un esquema de interacción entre los módulos, sensor Leap Motion y el usuario. Se puede observar que el módulo de Coordinación General interactúa con otros 3 módulos activos para coordinar su funcionamiento paralelo: módulo de Interfaz Gráfica y Producción Visual, módulo de Escenarios Virtuales y módulo de Captura de Datos.

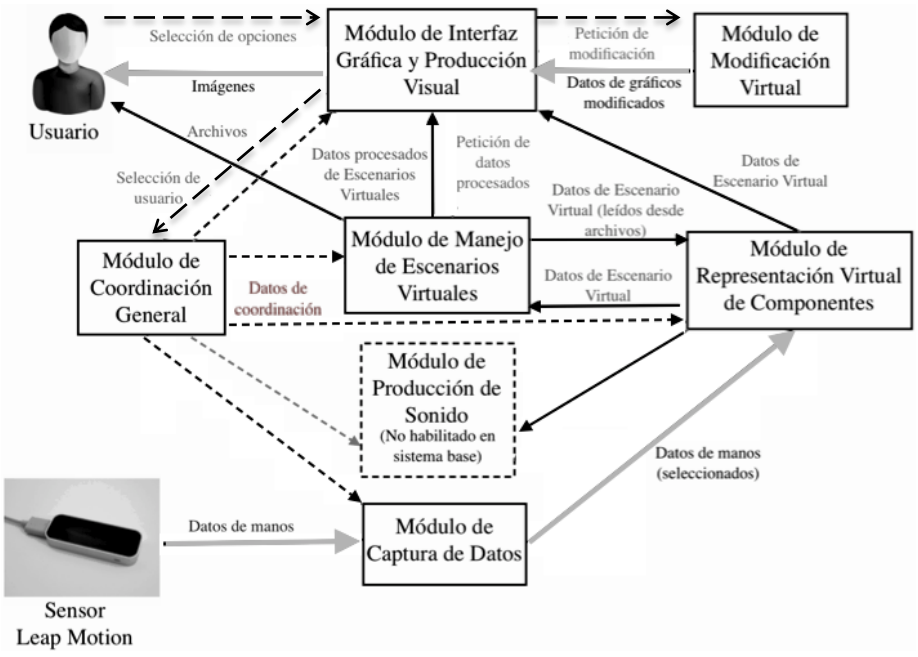

Figura 2 - Esquema de interacción general del sistema FiMAN.

Se puede observar que el sensor Leap Motion interactúa con el módulo de Captura el cual provee datos requeridos al módulo de Representación Virtual, desde donde los otros módulos del sistema que requieran podrán accederán a la información necesaria almacenada. Otra interacción importante es la del módulo de Interfaz Gráfica y Producción Visual, el cual requiere los datos de Representación Virtual para recrear gráficamente la modelización de objetos que se

encuentran en un Escenario Virtual. Por otra parte, el Módulo de Modificación Visual se relaciona directamente con el módulo de Interfaz Gráfica y Producción Visual para realizar los cambios visuales requeridos por el usuario. El módulo de Manejo de Escenarios Virtuales interactúa con los módulos de Interfaz Gráfica para la generación de datos estadísticos en base a Escenarios Virtuales guardados en archivos y con el módulo de Representación Virtual de Componentes para importar o exportar datos de Escenarios Virtuales a archivos.

Las líneas grises gruesas en la Figura 2 indican el flujo de datos de las manos y datos de representación gráfica, las líneas continuas normales indican que la interacción está relacionada directamente a los datos de Escenarios Virtuales, las líneas punteadas indican que la interacción está relacionada con datos de coordinación (iniciar, pausar, finalizar una tarea determinada de algún módulo) y las líneas segmentada indican que la interacción es producida de manera directa por intervención del usuario, tal como realizar una modificación virtual, seleccionar alguna opción como generar datos estadísticos, guardar datos en archivos o reproducir un archivo, entre otros.

El módulo de Producción de Sonido no está habilitado en el sistema básico, sin embargo, tiene las interfaces necesarias (*plugin*) para su extensión, donde interaccionaría con el módulo de Coordinación General y el módulo de Representación Virtual de Componentes produciendo sonidos en base a los datos de posición de los componentes del Escenario Virtual, e.g. se puede hacer que el sistema emita un sonido agudo (frecuencia alta dentro del rango de audición) cuando la punta de algún dedo llegue a una coordenada determinada (x,y,z).

2.2 Modelización del Software

El sistema FiMAN fue diseñado utilizando UML (Unified Modeling Language) [9] y se implementó cada módulo en un paquete según la siguiente relación (ver Figura 3):

- El módulo de Captura de Datos en el paquete ***Captura_Datos***
- El módulo de Representación Virtual de Componentes en el paquete ***Representacion_Virtual***
- El módulo de Interfaz Gráfica y Producción Visual en el paquete ***IG_Produccion_Visual***
- El módulo de Modificación Virtual en el paquete ***Modificacion_Virtual***
- El módulo de Manejo de Escenarios Virtuales en el paquete ***Manejo_Escenario***
- El módulo de Producción de Sonido en el paquete ***Produccion_Sonido***

La Figura 3 muestra el diagrama general de paquetes del sistema. Como se puede observar, existen relaciones de dependencia (flechas punteadas sin etiquetas), de unión (flechas punteadas con etiqueta "<<**merge**>>") y de acceso (flechas punteadas con etiqueta **"**<<**access**>>"), que determinan el espacio de trabajo (workspace) y el alcance de cada paquete.

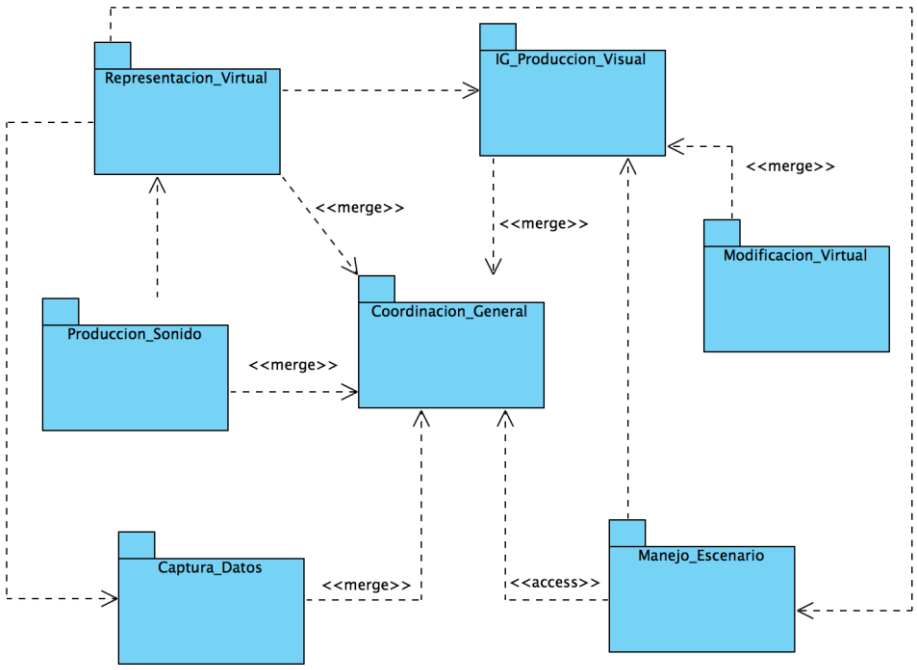

Figura 3 - Diagrama de Paquetes de FiMAN.

En caso del paquete ***Coordinacion_General*** tiene una relación de unión con todos los paquetes excepto ***Modificacion_Virtual*** y ***Manejo_Escenario***, ya que es el paquete principal del sistema desde donde se inicia y coordina todo el funcionamiento de los demás paquetes, por lo que puede acceder a todos ellos. En caso de la relación de dependencia, como es el de ***Representacion_Virtual*** y ***Captura_Datos,*** como su nombre indica, tendrá una dependencia entre los elementos de los paquetes**,** es decir que si se realiza una modificación en ***Captura_Datos*** afectará a ***Representación_Virtual***.

2. IMPLEMENTACIÓN DE FiMAN

Una vez la arquitectura de software modelizada como un sistema Orientado-Objeto, se seleccionó para la implementación de FiMAN el lenguaje de programación Java[4], principalmente por su naturaleza multiplataforma [10], por sus bibliotecas de desarrollo de imágenes y manejo de sonido. Para el desarrollo de aplicaciones, Leap Motion dispone de varios kits de desarrollo o SDK (Software Development Kit), con sus respectivos interfaces de programación o API (Application Programming Interface), en diversos lenguajes de programación, entre ellos Java.

A continuación se describe las características del hardware y software de Leap Motion además de todos los módulos implementados en el sistema.

2.1. El Sensor Leap Motion

El dispositivo Leap Motion es un sensor infrarrojo que ha sido creado por la empresa Leap Motion Inc. Tiene la capacidad de capturar la posición de la mano, específicamente del antebrazo, muñeca y dedos.

- **Especificaciones de Hardware:**

Este dispositivo tiene unas dimensiones de 75 mm de largo, 25 mm de ancho y 11 mm de alto[5]. Cuenta con dos cámaras, tres LEDs y un microcontrolador. Cada una de estas cámaras cuenta con un sensor monocromático, sensible a la luz infrarroja, con una longitud de onda de 850 nm[6]. Estos sensores pueden trabajar a una velocidad de hasta 200 fps (*frames per second*), dependiendo del rendimiento de la computadora conectada. Los LEDs se encargan de iluminar la zona de cobertura por inundación, trabajan en el espectro de luz infrarroja a una longitud de onda de 850 nm que es la misma a la que son sensibles los sensores ópticos. Además, este dispositivo cuenta con un controlador USB para que el ordenador pueda reconocerlo, este controlador es de alta velocidad y puede soportar USB 3.0.[7]. La zona de cobertura del dispositivo es una semiesfera de 61 cm de radio, esta zona depende del ángulo de visión de las lentes de las cámaras y de la intensidad máxima que puede entregar la conexión USB a los LEDs. Tanto el ángulo de visión horizontal de Leap Motion como el vertical son de 150º delimitando la zona de interacción[8].

- **API de Leap Motion:**

Leap Motion cuenta con una API que provee métodos que brindan directamente la opción de trabajar con los datos del muestreo de la mano. Los lenguajes de programación que pueden utilizar las Leap Motion API son: C++, C#, Objective-C, Java, JavaScript, Unity y Python[9].

En la API del dispositivo se define una zona de trabajo llamada "*Interaction Box*"[10] por un volumen de 110.55 mm de altura x 110.55 mm de ancho x 69.43 mm de profundidad, que varía sus dimensiones dependiendo de donde se encuentre el objeto a rastrear. Esta es la zona en la que se marca el centro del sistema de coordenadas cartesianas de Leap Motion. El sistema de coordenadas cartesianas tiene, desde la posición del usuario, el eje *y* con valores positivos hacia arriba, el *x* hacia la derecha y el *z* hacia el usuario.

El Leap Motion API obtiene los datos de las manos con una serie de capturas de escena llamados *frames*. Cada objeto *frame* contiene la información de cierto momento en el tiempo, esta información incluye datos de las manos (*Hands*). La cantidad de *frames* varía de acuerdo al uso que se está dando al dispositivo y puede variar desde 30 fps a 200 fps. *Frame* es un atributo de la clase *Controller*, la cual es la interfaz principal de Leap Motion.

En la Figura 2 se muestra un diagrama de clases (parcial) del API de Leap Motion donde se puede observar que la clase *Bone* compone (es una parte imprescindible, por definición) de *Finger* y este compone a *Hand*. Por otra parte, *Frame* es agregado (es una parte prescindible, puede o no existir) por *Hand* y, a su vez *Controller* es agregado por *Frame*.

[4] https://www.java.com/
[5] http://blog.leapmotion.com/designing-leap-motion-controller/
[6] https://developer.leapmotion.com/documentation/java/devguide/Leap_Overview.html
[7] La norma USB 3.0 permite velocidades de hasta de 480Mbps.
[8] http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/
[9] https://developer.leapmotion.com/documentation/
[10] https://developer.leapmotion.com/documentation/java/api/Leap.InteractionBox.html

Figura 4 - Diagrama de clases parcial de API de Leap Motion.

Las clases que contienen la información relevante de las capturas de manos son *Finger* y *Bone*, que representan a los dedos de las manos y a los huesos de las falanges respectivamente.

La clase *Finger* representa al seguimiento o rastreo de un dedo. Los objetos *Finger* son atributos de *Hand* y contienen datos como el tipo de dedo, longitud, ancho, velocidad y posición de punta de dedo.

En la Figura 5 se puede observar un segmento de código de una implementación con datos de salida en texto mostrados en la Figura 6. Este segmento de código muestra un bucle o ciclo donde se muestran los datos de cada *Finger* en un *Hand*. Entre los datos mostrados se encuentran el tipo de dedo (*type()*), el identificador (*id()*), largo del dedo (*length()*), ancho (*width()*), velocidad de la punta (*tipVelocity()*) en milímetros sobre segundo, para cada uno de sus componentes *x*, *y* y *z* obtenidos mediante los métodos *getX()*, *getY()* y *getZ()* respectivamente. Además, también se muestra los datos de la posición de la punta (obtenido mediante el método *tipPosition*()) en milímetros según los ejes *x, y, z*.

Los tipos de dedo (*type()*) están definidos en inglés y pueden ser *TYPE_THUMB*, *TYPE_INDEX*, *TYPE_MIDDLE*, *TYPE_RING* y *TYPE_PINKY* que corresponden los dedos pulgar, índice, medio, anular y meñique respectivamente.

```
for (Finger finger : hand.fingers()) {
    System.out.println("    " + finger.type() + ", id: " + finger.id()
                       + ", largo: " + finger.length()
                       + "mm, ancho: " + finger.width() + "mm");

    Vector velocidad = finger.tipVelocity();

    System.out.println("VELOCIDAD: x=" + velocidad.getX()+" y=" +
                         velocidad.getY()+" z =" + velocidad.getZ());

    System.out.println("POSICION DE PUNTA: " + finger.tipPosition());
}
```

Figura 5: Código para captura y muestra de datos *Finger* en Java

```
------------------------------------------------------------------------
Frame ID: 518860
    TYPE_THUMB, id: 1930, largo: 45.478523mm, ancho: 17.670841mm
VELOCIDAD: x=606.45386 y=84.60536 z =343.4027
POSICION DE PUNTA: (214.933, 115.036, 64.7597)
    TYPE_INDEX, id: 1931, largo: 51.317444mm, ancho: 16.879187mm
VELOCIDAD: x=668.93225 y=-164.91988 z =203.6142
POSICION DE PUNTA: (215.632, 103.52, 28.7524)
    TYPE_MIDDLE, id: 1932, largo: 58.472107mm, ancho: 16.577604mm
VELOCIDAD: x=706.37103 y=-243.59868 z =206.10104
POSICION DE PUNTA: (204.667, 94.7315, 15.0673)
    TYPE_RING, id: 1933, largo: 56.222492mm, ancho: 15.774642mm
VELOCIDAD: x=715.4694 y=-276.1854 z =205.87712
POSICION DE PUNTA: (188.16, 83.0184, 16.1457)
    TYPE_PINKY, id: 1934, largo: 44.077393mm, ancho: 14.01227mm
VELOCIDAD: x=712.41125 y=-272.2701 z =148.11781
POSICION DE PUNTA: (166.754, 78.2351, 19.4783)
------------------------------------------------------------------------
```

Figura 6 - Output de muestra de datos *Finger* en Java.

La clase *Bone* representa a un hueso de la mano rastreado por el sensor Leap Motion. Cada *Finger* (dedo) tiene 4 *Bone* (hueso falange) excepto el pulgar (*Finger* tipo *TYPE_THUMB*) que tiene 3. Los huesos se ordenan desde la base hasta la punta, indexado de 0 a 3. Además, el tipo de hueso (*type()*) puede ser utilizado para indexar un hueso específico anatómicamente.

2.2. Módulos de FiMAN

A continuación se explica la implementación de cada paquete correspondiente a cada módulo de FiMAN en Java.

- **Captura de Datos:**

El módulo de Captura de Datos funciona en un hilo de procesamiento (*Thread*) que es iniciado y supervisado por el módulo de Coordinación General y se encarga en almacenar los datos en el Módulo de Representación Virtual mientras se realiza la captura de los mismos a tiempo real utilizando la biblioteca de Leap Motion.

El sensor envía los datos mediante un cable USB a la computadora y, gracias a la biblioteca Java (**`LeapJava.jar`**) de Leap Motion, se pueden interpretar los mismos como objetos, seleccionándose en el Módulo de Captura, los datos de la mano para posteriormente pasarle los mismos al módulo de Representación Virtual.

En la Figura 7 se muestra un esquema de la interacción general entre el Módulo de Captura, el Módulo de Coordinación General, el sensor Leap Motion y el Módulo de Representación Virtual.

- **Representación Virtual:**

El Módulo de Representación Virtual está muy relacionado al Módulo de Captura de Datos, ya que todos sus datos provienen del mismo. Éste módulo principalmente se encarga de estructurar y almacenar momentáneamente los datos considerados necesarios para la representación gráfica y posterior reproducción en FiMAN.

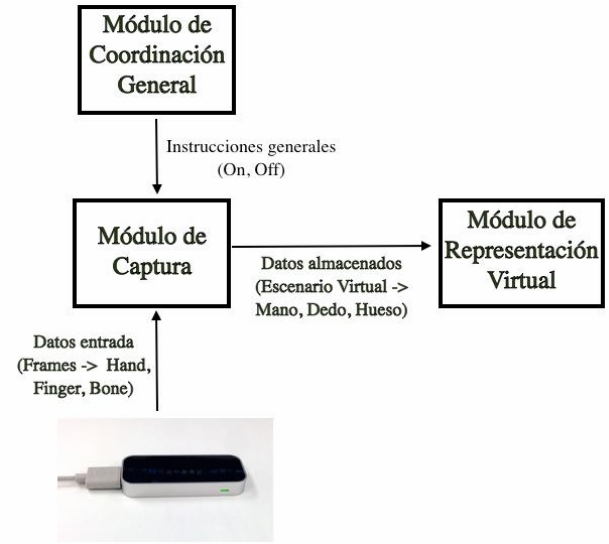

Figura 7 - Esquema de interacción entre sensor Leap Motion y módulos relacionados.

El módulo está compuesto por las clases que representan a un escenario virtual, que contiene a las estructuras de las manos. Cada mano está compuesta por los dedos y a su vez, éstos están compuestos por huesos (falanges) tal como se muestra en la Figura 8. Los componentes del escenario virtual contienen los datos de la posición en la que se encuentran (en milímetros), la velocidad de los dedos (en milímetros sobre segundo), además de otros datos relevantes para la representación gráfica.

La estructura de Representación Virtual corresponde en parte a la estructura que Leap Motion utiliza; donde una mano (*Hand*) está compuesta por dedos (*Finger*) que a su vez están compuestos por huesos falanges (*Bone*). Sin embargo, se utilizan clases propias para crear objetos sólo con la información útil requerida, tales como su posición inicial en el espacio, la velocidad del componente, posición de la muñeca, ángulos de inclinación de los huesos y los vectores base que representan la posición de un hueso.

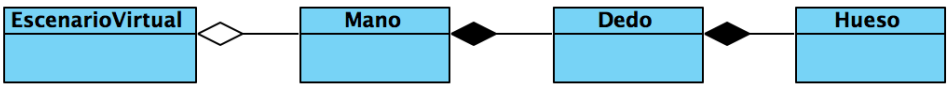

Figura 8 - Diagrama de clases simplificado del módulo de Representación Virtual.

La Representación Virtual depende del Módulo Captura (para actualizar los datos de la posición de un dedo, por ejemplo) y Manejo de Almacenamiento de Escenas (en grabación o reproducción).

- **Interfaz Gráfica y Producción Visual:**

La Interfaz Gráfica y Producción Visual está compuesta por las clases que se encargan de producir las representaciones gráficas de los dedos y muñecas que tienen datos almacenados en Representación Virtual. Estas clases han sido implementadas desde cero y no forman parte del API de Leap Motion.

Para realizar la representación gráfica de la mano se utilizan los datos de los huesos de los dedos capturados y la posición de la muñeca, que se dibujan mediante mallas poligonales [11].

En FiMAN, los componentes son representados tomando un sistema de coordenadas diferente al de Representación Virtual que tiene el mismo sistema de coordenadas del API de Leap Motion. El sistema de coordenadas de éste módulo se sitúa a 30 centímetros detrás del origen (tomando como punto de referencia la posición de la pantalla en la cual se proyecta la imagen) y 15 centímetros más bajo de la horizontal (tomando como punto de referencia el centro de la pantalla en la cual se proyecta la imagen) ya que de ésta manera se puede visualizar las mallas poligonales como un observador externo.

La biblioteca de visualización de mallas poligonales fue implementada de cero dentro del desarrollo de FiMAN a partir de principios matemáticos para la manipulación de objetos en 3 dimensiones [11]. Dicha biblioteca, no es específica a FiMAN, y puede ser utilizada en otros sistemas que requieran similar visualización.

En la Figura 9 se muestra el diagrama de clases utilizado en la biblioteca de visualización de mallas poligonales. Donde se define que, para que un objeto sea dibujado debe ser modelado con vértices y aristas, debiendo por lo tanto heredar de la clase *Malla*, que representa a una malla poligonal. La clase *Malla* está compuesta por una lista vértices y una lista de aristas, que son *VList* y *AList*, estas clases son listas doblemente enlazadas[11] de objetos *VNodo* y *ANodo*. A su vez, cada objeto *VNodo* y *ANodo* contiene a un vértice (atributo *vertice*) y una arista (atributo *arista*) respectivamente, además del siguiente *VNodo* o *ANodo* que componen una lista (atributo *next*). Por otra parte, cada objeto *Arista* es formado por dos vértices (atributos *v1* y *v2*).

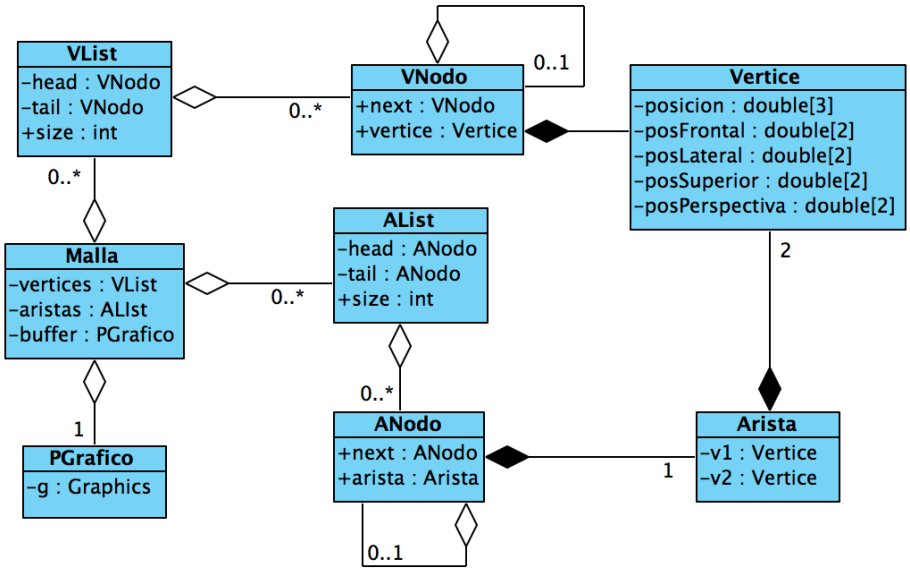

Figura 9 - Diagrama de clases de biblioteca de visualización de mallas poligonales.

Cada vértice contiene el dato de la posición en la que se encuentra (en un espacio 3 dimensiones) y además, contiene los datos de las posiciones respecto a unos marcos que muestran al espacio desde una posición frontal, lateral, superior y en perspectiva. Estos datos (contenidos en los atributos *posFrontal*, *posLateral*, *posSuperior* y *posPerspectiva*) están transformados a 2 dimensiones para poder ser dibujados en un *JFrame*[12] mediante la clase *PGrafico* que contiene a *Graphics*, la cual permite realizar gráficos que serán mostrados en la pantalla transformados a 2 dimensiones para poder

[11] Tipo de dato auto-referenciado compuesto por nodos que a su vez contienen un puntero a otro dato del mismo tipo. Cada nodo (excepto el último) enlaza con el nodo siguiente.

[12] La clase *JFrame* se utiliza como un contenedor de alto nivel de los componentes gráficos de una interfaz gráfica de usuario (https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html)

ser dibujados en un *JFrame*[13] mediante la clase *PGrafico* que contiene a *Graphics*, la cual permite realizar gráficos que serán mostrados en la pantalla[14].

En la Figura 10 se puede observar la representación de los dedos de la mano mediante mallas poligonales.

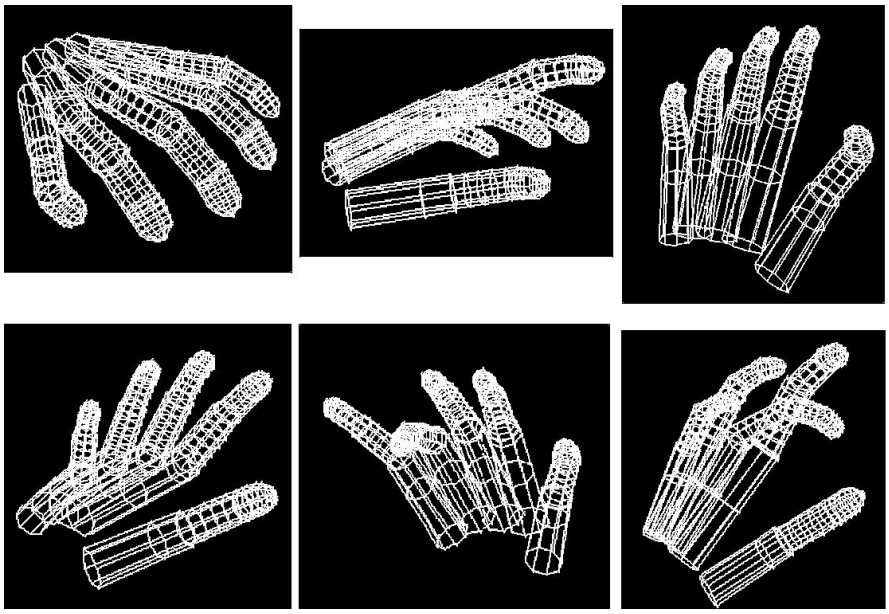

Figura 10 - Representación de los dedos mediante mallas poligonales en FiMAN.

Las representaciones gráficas pueden ser modificadas mediante las clases de Modificación Virtual y, a su vez Producción Visual puede modificar la configuración visual según los cambios que realice el usuario a través de la Interfaz Gráfica.

El módulo de Interfaz Gráfica y Producción Visual también están compuestos por la interfaz mediante la que el usuario interactúa con el sistema, es decir, los *frames* y paneles que contienen los botones, imágenes, *checkboxes*, barras de deslizamiento (*slide bar*), cuadros de dialogo y cuadros estadísticos, siendo los últimos implementados en base a la biblioteca JFreeChart[15].

- **Modificación Virtual:**

El módulo de Modificación Virtual está compuesto por las clases que se encargan de modificar el entorno virtual representado en la visualización a nivel gráfico.

La interacción del usuario con el módulo de Modificación Virtual en FiMAN se realiza mediante el teclado y la interfaz gráfica que activan los eventos para modificar la representación gráfica de mallas poligonales en la pantalla. El usuario realiza petición de modificación virtual al teclado (para rotar, acercar o alejar la imagen) o al módulo de Interfaz Gráfica y Producción Visual (para desactivar la representación gráfica de un componente), posteriormente las peticiones son remitidas al módulo de Modificación Virtual donde se procesan las peticiones y se realizan los cambios a los parámetros correspondientes dentro del módulo de Interfaz Gráfica y Producción Visual. Hay que hacer notar que el módulo de Representación Virtual no es afectado por modificaciones realizadas, ya que todos los cambios son a nivel de representación gráfica únicamente, sin realizar cambios a los datos dicho módulo que sigue enviando los datos del Escenario Virtual para ser representados gráficamente con los cambios realizados. En la Figura 11 se puede observar la interacción entre los componentes relacionados a la modificación virtual.

La modificación de visualización incluye los movimientos de rotación, acercamiento y alejamiento de la representación gráfica además de la activación o desactivación de componentes representados mediante mallas poligonales (para la habilitar o no habilitar el dibujo de los dedos y las muñecas).

[13] La clase *JFrame* se utiliza como un contenedor de alto nivel de los componentes gráficos de una interfaz gráfica de usuario (https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html)
[14] https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html
[15] http://www.jfree.org/jfreechart/

- **Manejo de Escenarios Virtuales:**

El módulo de Manejo de Escenarios Virtuales está compuesto por las clases que se encargan de generar archivos, leerlos y reproducirlos, crear gráficos estadísticos en base a archivos previamente guardados, así como también generar reportes en formato Excel mediante la biblioteca JExcel[16] en base a la información almacenada en Escenarios Virtuales.

En el sistema FiMAN, el usuario puede iniciar el proceso de grabación de movimiento de manos mediante la interfaz gráfica, lo que generará un hilo de ejecución (*Thread*) para capturar los datos de las manos del Escenario Virtual. Posteriormente, mediante el mecanismo de serialización de Java [12] se procede a guardar los datos en un archivo temporal, para luego dar la opción de guardar la grabación en un archivo. Éstos datos, que son captados por el Módulo de Captura de Datos, pueden ser almacenados en archivos con la extensión `.fim` (FiMAM file)[17].

Otras opciones de FiMAN permiten al usuario reproducir los archivos previamente guardados. En este caso, de manera inversa a lo que se realiza en la grabación, se genera un hilo de ejecución que procede a guardar los datos del archivo en el Escenario Virtual. De igual manera, se pueden generan gráficos estadísticos y el usuario tiene la opción de exportar los datos a Excel para un registro personal o para un análisis posterior. Estas opciones son habilitadas posteriormente al cargado de archivos `.fim` al sistema.

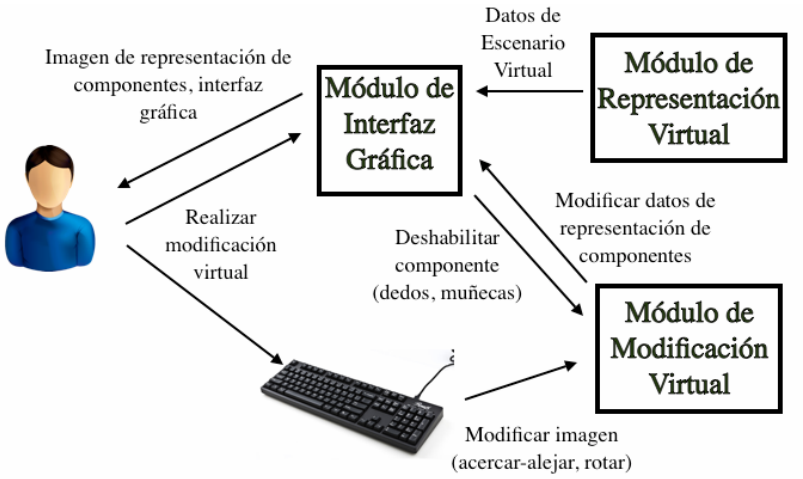

Figura 11 - Esquema de interacción de componentes relacionados a la modificación virtual.

- **Producción de Sonido:**

El módulo de Producción de Sonido fue desarrollado como un *plugin* del sistema y es el encargado de la activación de sonidos MIDI (Musical Instrument Digital Interface) . Esta funcionalidad fue implementada en base a la librería MIDI de Java[18] y consiste en un teclado (musical) virtual que reproduce sonidos correspondientes a las notas musicales activados mediante eventos.

MIDI es un protocolo estándar de comunicación para dispositivos musicales electrónicos, como los instrumentos de teclado electrónicos y computadoras personales. Los datos MIDI se pueden transmitir a través de cables especiales durante una actuación en vivo como también se pueden almacenar en un tipo estándar de archivo para la reproducción o la edición posterior[19].

Java, entre sus bibliotecas de clases, presenta un API que emula el comportamiento de un sintetizador MIDI mediante unas clases que están definidas en las bibliotecas de `javax.sound.midi`, donde existen secuenciadores, sintetizadores, transmisores (los relacionados con los puertos de entrada MIDI), receptores (los asociados con los puertos de salida MIDI), los datos de los archivos MIDI estándar y los datos de los archivos del banco de sonidos[20]. Las clases pertenecientes a este módulo corresponden a un hilo de proceso, al sintetizador (necesario para producir el sonido MIDI en Java), a las notas musicales para determinar el sonido a producir, la teclas y los estados de una tecla (activado y desactivado).

[16] https://www.teamdev.com/jexcel
[17] Se utiliza un tipo de fichero específico a FiMAN con la representación de las escenas completas.
[18] https://docs.oracle.com/javase/7/docs/api/javax/sound/midi/package-summary.html
[19] http://docs.oracle.com/javase/tutorial/sound/overview-MIDI.html
[20] http://docs.oracle.com/javase/tutorial/sound/accessing-MIDI.html

- **Coordinación General:**

El módulo de Coordinación General es el encargado de controlar el funcionamiento del sistema mediante permisos que sirven para iniciar o finalizar *Threads* de los módulos del sistema. El sistema FiMAN funciona con 4 *Threads* ejecutándose de manera concurrente. Esos *Thread* son regulados (iniciados o detenidos) por Coordinación General y corresponden a las clases que son encargadas de manejar los procesos de los módulos Producción Visual, Producción de Sonido, Captura de Datos y Manejo de Escenarios Virtuales (para la funcionalidad de grabación y reproducción).

El módulo contiene un registro los *Threads* activos del sistema por lo que, constantemente sirve como verificador para que todos los procesos funcionen de manera coordinada y accedan sin problemas a los diferentes objetos en tiempo de ejecución.

La Figura 12 muestra el sistema de base completo de FiMAN, que incluye todas las funcionalidades para realizar análisis de movimientos de las manos de manera visual, en tiempo real y además brinda al usuario las diferentes opciones de manejo del sistema: grabación, reproducción, exportación de datos, cargado de archivos, producción de gráficos estadísticos, activación y desactivación de representaciones gráficas, además de un visualizador de datos de posiciones de la punta de dedos y ángulos de dedos respecto al plano horizontal y vertical.

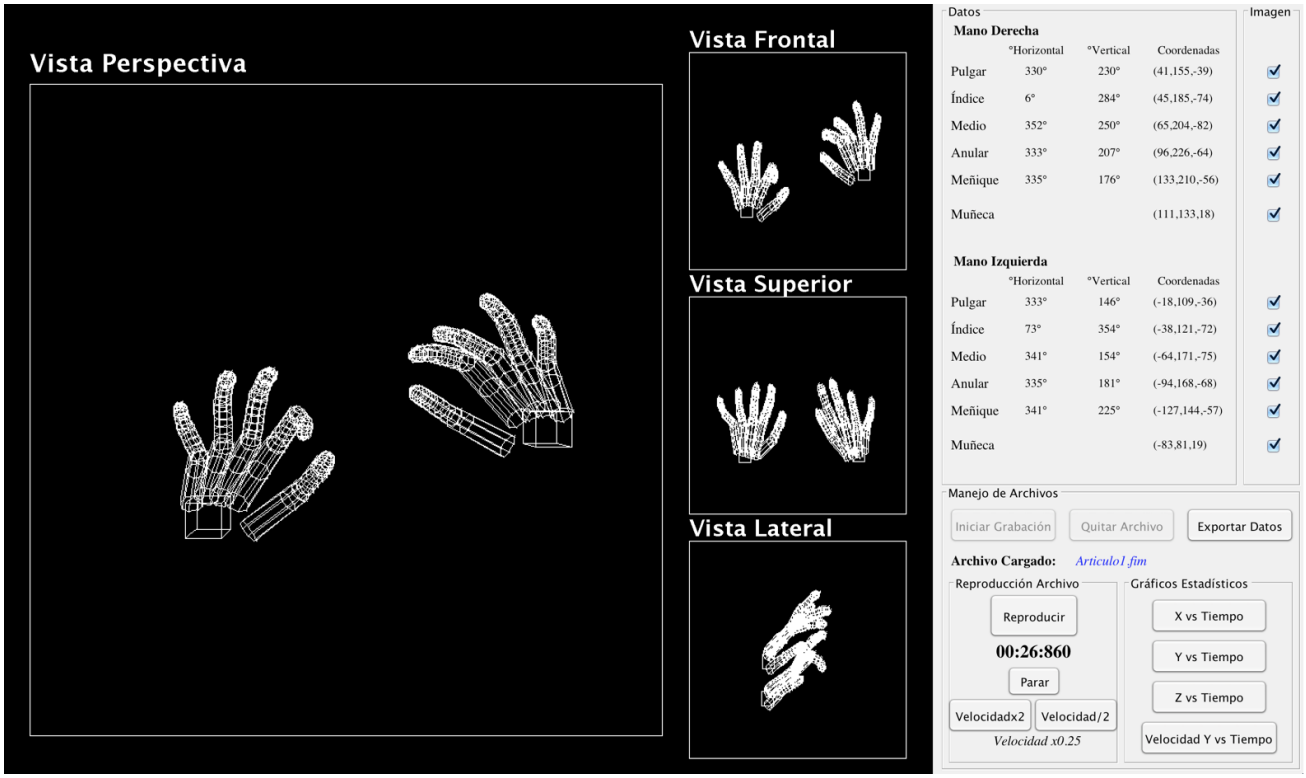

Figura 12 - Captura de pantalla de FiMAN en funcionamiento.

La implementación completa de FiMAN incluye 53 clases Java y más de 7700 líneas de código fuente (Lines of Code – LoC), y el programa fue compilado y empaquetado en un archivo ejecutable multiplataforma para su fácil despliegue y ejecución, requiriendo únicamente el entorno de ejecución Java (Java Runtime Environment – JRE)[21].

3. ESTUDIO DE CASO: EXTENSIÓN DE FiMAN PARA SU APLICACIÓN A LA EDUCACIÓN MUSICAL

Una de las características más importantes de FiMAN es su modularidad, lo que permite desarrollar nuevas aplicaciones basadas en los datos capturados por un sensor Leap Motion. Para demostrar dicha modularidad, se implementó una extensión para la educación musical de técnicas de piano. Para esto se definieron algunos requerimientos específicos al Estudio de Caso mediante una consulta a expertos maestros de piano de la Academia Nacional de Música "Man Césped" de la ciudad de Cochabamba, Bolivia.

[21] http://www.oracle.com/technetwork/java/javase/

En este Estudio de Caso se muestra el potencial del sistema desarrollado, puesto que permite la inclusión de nuevos componentes (teclado virtual de un piano), el desarrollo de análisis específicos (posicionamiento de la mano), la inclusión de interacciones entre componentes virtuales (importancia de la posición de la yema del dedo con respecto a una tecla y la velocidad al tocar) y el análisis posterior que puede realizar el maestro para indicar de manera visual al estudiante, cómo corregir sus errores desde varios puntos de perspectiva y a diferentes velocidades.

3.1. Soporte Físico

Para poder determinar el rango infrarrojo exacto en el que trabaja el sensor Leap Motion, se realizaron medidas con un espectrómetro. En la Figura 13 se puede observar el resultado que se obtuvo mediante el espectrómetro OceanOptics STS-VIS[22] con el software de análisis *Ocean Optics Spectra Suite*[23]. El Leap Motion trabaja con ondas electromagnéticas en el rango infrarrojo, es decir, longitudes de onda mayores a 750 nm. Los 3 LEDs del Leap Motion emiten luz infrarroja tienen sus máximos en aproximadamente de 817 nm y de 863 nm. Las 2 cámaras infrarrojas del Leap Motion capturan los movimientos con longitudes de onda de 850 nm[24]. Como se observa en la Figura 11, la intensidad para 850 nm. es suficiente para que las cámaras puedan capturar el movimiento.

Para la simulación del teclado virtual, se diseñó un soporte físico de referencia que permite al usuario sentir una superficie donde se represente al teclado virtual del piano. Se utilizó como superficie una placa de vidrio común, puesto que este material tiene una absorción mínima en estas longitudes de onda con respecto a otros materiales (plástico, plexiglás, resina, etc.).

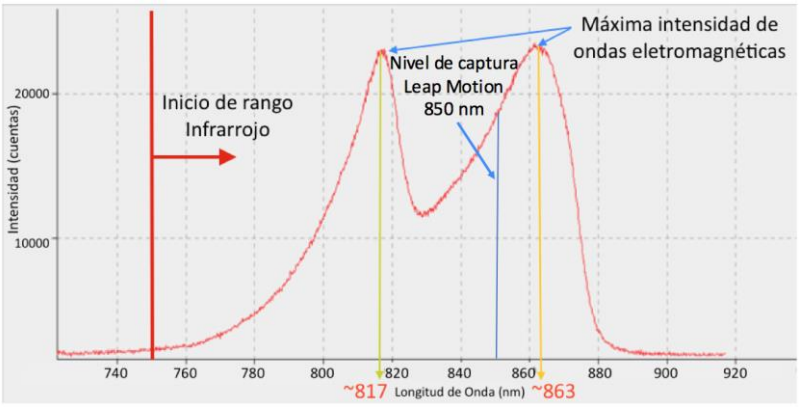

Figura 13 - Longitud e intensidad de onda producida por Leap Motion.

El soporte físico está conformado por la superficie de vidrio y una estructura metálica, formada por un marco, una vara vertical y vara horizontal, las cuales están unidas entre sí mediante prensas o seguros de presión. La Figura 11 muestra el soporte físico ensamblado.

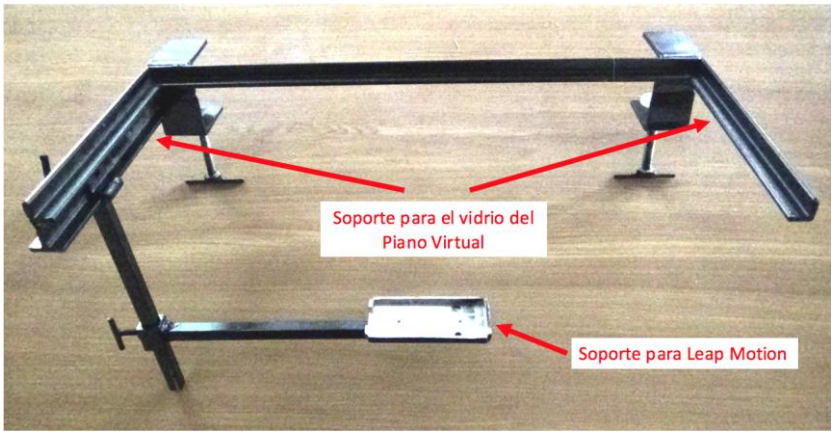

Figura 14 - Soporte físico ensamblado con soporte para el Leap Motion.

[22] http://oceanoptics.com/product/sts-vis-microspectrometer/
[23] http://oceanoptics.com/support/software-downloads/
[24] http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/

La superficie de vidrio es una sección de vidrio de 2 mm de espesor, 38,5 cm de largo y 17 cm de ancho que funciona como superficie soporte a las manos del usuario y a la vez, permite al Leap Motion captar la posición de la mano ya que es un material que no interfiere o interfiere en muy poca medida con el infrarrojo.

El largo y el ancho fueron definidos en base a pruebas de alcance del Leap Motion, a la medida de 2 octavas del piano (33 cm) y la longitud de las teclas blancas del piano (14.7 cm) aumentando 2,3 cm de tolerancia y de sujeción al marco.

3.2. Diseño de Teclado Virtual

Para la representación del teclado virtual se desarrolló un modelo propio, tomando como parámetro referencias reales de la composición de las teclas (blancas y negras) siguiendo el Índice Acústico Científico [13]. La Figura 11 muestra los valores de los ocho vértices (V_1 ,V_2, ... V_8) de la tecla 'sol' tomando como referencia principal las coordenadas *(x,z)*[25] correspondientes al vértice V_1; donde la longitud N de las teclas negras es 9,5 cm., la longitud B de las blancas es 14,7 cm., S_n y S_b corresponden al grosor de las teclas negras y blancas respectivamente.

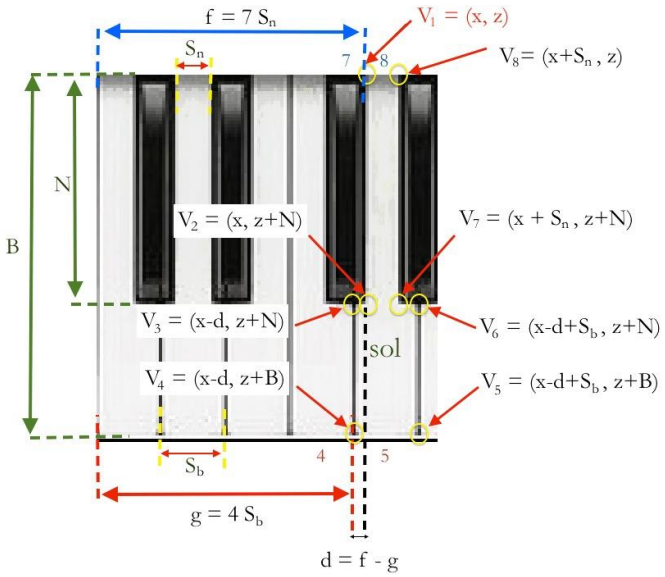

Figura 15 - Representación gráfica de la estructura (valores de los vértices) de la tecla 'sol'.

Con los vértices identificados, el sistema permite calibrar el soporte físico de vidrio para identificar los vértices de la manera más realista posible. Para esto se determina la posición y el ángulo de contacto con un espacio y profundidad de tolerancia de toque, como muestra la Figura 16.

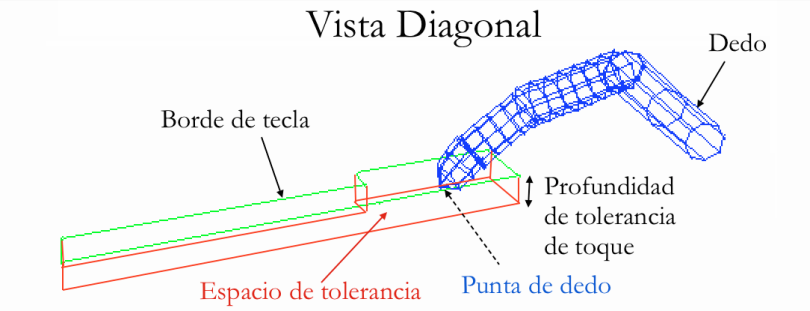

Figura 16 - Virtualización de las teclas de piano y su interacción con los dedos.

La parte funcional del teclado virtual fue desarrollado en base al módulo de sonido de FiMAN que proporciona las herramientas necesarias para la activación del sonido correspondiente a cada tecla. Una particularidad es el manejo de la profundidad de tolerancia de toque, puesto que se desea que el piano virtual sea lo más realista posible, controlando la intensidad del sonido dependiendo la posición de la punta del dedo, con respecto al lugar de la tecla tocada (espacio de

[25] El componente *x* corresponde a la cantidad de milímetros en dirección positiva derecha y *z* en dirección positiva al usuario teniendo como origen y punto de referencia al sensor Leap Motion

tolerancia) y la velocidad de apoyo sobre la tecla virtual. Esto permite equilibrar la falta de una tecla real, con la intensidad del sonido generado.

3.3. Uso real del sistema

Se realizaron pruebas reales con estudiantes de piano individuales para analizar técnicas de piano (ver Figura 17) con un resultado óptimo.

Entre las posibilidades de uso de FiMAN para el análisis de movimiento digital, la verificación de movimientos de dedo respecto al tiempo tiene una especial utilidad al momento de evaluar el desarrollo de la técnica pianística de un estudiante.

Figura 17 - Estudiante y maestro de piano utilizando el FiMAN extendido.

Una de las formas de evaluar la técnica básica consiste en tocar "trinos" (alternación de dos notas repetidas de manera cíclica) con dos dedos contiguos realizando movimientos de subida y bajada sin variar la posición horizontal. Se puede realizar la evaluación de los dedos mediante trinos con cualquier combinación posible de dedos contiguos. Sin embargo existe mayor dificultad de movimiento con algunos dedos, siendo los dedos medio y anular una de las combinaciones más complicadas para los principiantes debido a la poca utilización independiente de estos dedos en la vida diaria, ya que casi siempre se mueven al mismo tiempo y en la misma dirección, lo que representa una dificultad natural anatómica a ser superada.

Los trinos, en el caso ideal, deberían ser veloces y ser regulares en el tiempo (presentar el mismo lapso de tiempo entre la ejecución de una nota y otra, en todos los ciclos). La velocidad y regularidad de los trinos demuestra un control de los dedos implicados por parte del ejecutante, como también el desarrollo muscular requerido para avanzar técnicamente y tocar obras más difíciles. Esta evaluación actualmente se la realiza de manera auditiva, es decir, a criterio del maestro y del estudiante, quienes ajustan los parámetros de evaluación según su percepción.

La extensión desarrollada en el Estudio de Caso, permite grabar los movimientos de los dedos tocando trinos en dos teclas del teclado virtual en un lapso de tiempo determinado por el usuario y, posteriormente generar gráficos estadísticos de los movimientos verticales realizados (posiciones de la punta de dedos en eje *y*), dando la opción de activar o desactivar la muestra de datos de cualquier dedo en dicho gráfico.

En la Figura 18 y la Figura 19 se pueden observar los gráficos de la captura de dos grabaciones realizadas durante 15 segundos por estudiantes de piano con un mes y un año de aprendizaje respectivamente.

En las imágenes se observan los datos de la posición vertical (en milímetros) de la punta de los dedos medio y anular de la mano derecha respecto al tiempo. La notación utilizada "RH 3" y "RH 4" corresponden a la notación internacional de piano utilizada en la actualidad, donde "RH" es mano derecha (abreviación de "Right Hand") y los números 3 y 4 corresponden a los dedos medio y anular respectivamente.

Como se muestra en la Figura 18 y Figura 19, se puede observar la diferencia entre de capacidad motriz en los dedos medio y anular en un lapso de 15 segundos. Se aprecia que el estudiante con un mes de aprendizaje de piano realiza 6 ciclos de subida y bajada de dedos, a diferencia de los 11 ciclos que realiza un estudiante de piano con un año de práctica, además se puede apreciar que el tiempo que el estudiante tarda en levantar cada dedo es, en el primer caso, más pronunciado, formando un arco en la gráfica, debido a que los músculos extensores de sus dedos tienen mucha

menor capacidad que sus músculos flexores lo que, a diferencia del segundo caso, no se aprecia de manera evidente. Viendo la gráfica resultante, el maestro de piano puede plantearse la posibilidad de brindar al primer estudiante la práctica de ejercicios adicionales para aumentar su capacidad de extensión de los dedos medio y anular de la mano derecha. Asimismo, la reproducción de los movimientos con visualización de la posición de la mano y los dedos, en 3D y a diferentes velocidades y con diferentes perspectivas, permiten al maestro visualizar en detalle la evolución de los estudiantes, y dar guías para corregir la posición de las manos y los dedos si es necesario.

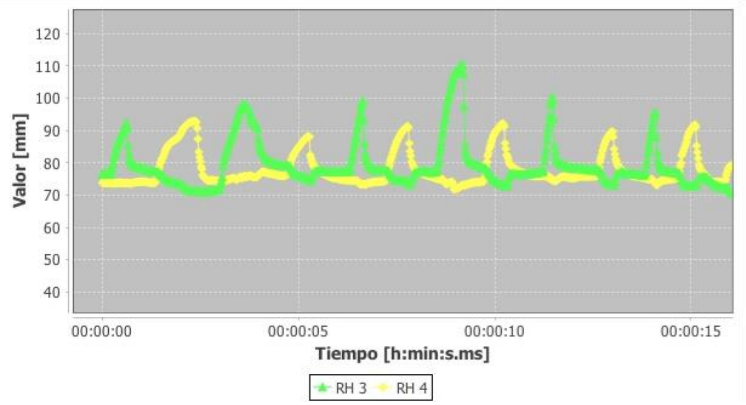

Figura 17 - Captura de movimiento realizando un trino con los dedos medio y anular de la mano derecha de una persona con un mes de estudio de piano.

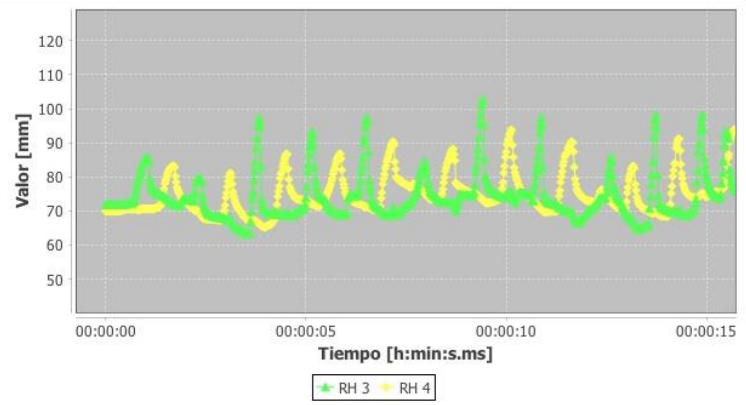

Figura 18 - Captura de movimiento realizando un trino con los dedos medio y anular de la mano derecha de una persona con un año de estudio de piano.

4. CONCLUSIÓN Y DESARROLLO FUTURO

En este artículo se presentó el desarrollo de FiMAN, un Sistema Computarizado para Análisis de Movimientos Digitales. FiMAN utiliza un sensor Leap Motion para capturar datos de los movimientos de las manos, y genera las representaciones en 3D y en tiempo real para su análisis inmediato o posterior. Todo el software de FiMAN fue desarrollado de cero, de manera modular y utilizando únicamente el interface de programación de Leap Motion para acceder a datos en bruto. Fueron descritos los diferentes módulos de software y la interacción entre ellos para manejar tanto el flujo de datos como el flujo de control relacionado a las interacciones del usuario. Para demostrar la modularidad de FiMAN, se desarrolló una extensión para el área de educación musical, a través de la modelización de un piano virtual. La extensión incluye un *pluggin* para la generación de sonidos de acuerdo a los movimientos que el usuario realiza, con respecto a la posición de las teclas virtuales. Se demostró la utilidad de FiMAN en un uso real con estudiantes de piano, donde el maestro pudo realizar un análisis de los movimientos y de la posición de las manos y dedos de los estudiantes, para así poder hacer un seguimiento de la evolución de la técnica pianística de sus estudiantes. Actualmente no existen herramientas tecnológicas similares a FiMAN para el análisis de movimientos y posición de las manos y dedos para la corrección de técnicas de interpretación de piano, lo que muestra el carácter innovador de FiMAN.

Las perspectivas de desarrollo futuro incluyen el uso de varios sensores, puesto que en algunos casos, el uso de un sensor puede ser limitante, esto se debe a que algunas partes de las manos no son captadas correctamente (normalmente por superposición de objetos en el rango del haz infrarrojo), lo que puede resultar en análisis más complejos o erróneos.

Asimismo el análisis de datos requiere un desarrollo adicional, ya que la cantidad de datos recolectados puede ser muy grande. En el sistema de base de FiMAN, se limitó dicha producción de datos a través de filtros donde se seleccionan las partes de interés de la mano.

De manera general podemos concluir que FiMAN es una base para el desarrollo futuro de aplicaciones en varios dominios que requieren análisis de movimientos, como la salud (recuperación en fisioterapia), responsabilidad social (aprendizaje de lenguaje de señas), la realidad virtual aumentada, el control de robots, el entretenimiento y otros. Este desarrollo futuro puede ser realizado gracias a la modularidad del sistema, el cual brinda flexibilidad y adaptabilidad.

5. AGRADECIMIENTOS

Los autores agradecen a Koichi Fujii, director de la Academia Nacional de Música "Man Cesped" de Cochabamba, Bolivia, el permitir probar el sistema FiMAN con estudiantes de piano de la Academia.

6. BIBLIOGRAFÍA

[1] Y.-J. Chang, S.-F. Chen, and J.-D. Huang, "A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities," *Res. Dev. Disabil.*, vol. 32, no. 6, pp. 2566–2570, 2011.

[2] I. J. Garcia Agenjo, "Uso de Leap Motion en juegos didácticos para niños con necesidades educativas especiales," Universidad Politécnica de Madrid, Madrid, Spain, 2015.

[3] I.-C. Chung, C.-Y. Huang, S.-C. Yeh, W.-C. Chiang, and M.-H. Tseng, "Developing Kinect Games Integrated with Virtual Reality on Activities of Daily Living for Children with Developmental Delay," in *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, I.-C. Chung, H.-C. Chao, D.-J. Deng, and J. J. Park, Eds. Springer Netherlands, 2014, pp. 1091–1097.

[4] D. Bassily, C. Georgoulas, J. Guettler, T. Linner, and T. Bock, "Intuitive and Adaptive Robotic Arm Manipulation using the Leap Motion Controller," in *ISR/Robotik 2014. 41st International Symposium on Robotics*, 2014, pp. 1–7.

[5] Y. Wu and T. S. Huang, "Human hand modeling, analysis and animation in the context of HCI," in *International Conference on Image Processing. ICIP'99*, 1999, vol. 3, pp. 6–10.

[6] D. A. Alfaro Vives and D. M. Velarde Robles, "Desarrollo de aplicaciones con Leap Motion," Universidad Peruana de Ciencias Aplicadas - UPC, Lima, Perú, 2015.

[7] J. H. Fonteles, É. S. Silva, and M. A. Formico Rodrigues, "Gesture-Driven Interaction Using the Leap Motion to Conduct a 3D Particle System: Evaluation and Analysis of an Orchestral Performance," *SBC J. Interact. Syst.*, vol. 6, no. 2, pp. 11–21, 2015.

[8] E. S. Silva, A. J. O. De Abreu, H. P. De Almeida, V. Teichrieb, and G. L. Ramalho, "A Preliminary Evaluation of the Leap Motion Sensor as Controller of New Digital Musical Instruments," Recife, Brasil, 2013.

[9] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*, 2nd Editio. Addison Wesley, 2005.

[10] J. Gosling, B. Joy, G. L. Steele, and G. Bracha, *The Java Language Specification, Third Edition*. Addison-Wesley, 2005.

[11] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice in C*, 2nd ed. Addison-Wesley, 2004.

[12] L. Opyrchal and A. Prakash, "Efficient Object Serialization in Java," in *Workshops on Electronic Commerce and Web-based Applications/Middleware. 19th IEEE International Conference on Distributed Computing Systems.*, 1999, pp. 96–101.

[13] R. W. Young, "Terminology for Logarithmic Frequency Units," *J. Acoust. Soc. Am.*, vol. 11, no. 1, p. 134, 1939.