



UNIVERSIDAD PRIVADA BOLIVIANA
FACULTAD DE INGENIERÍA Y ARQUITECTURA
CARRERA DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

**PROTOTIPO DE SISTEMA COMPUTARIZADO PARA EL
ESTUDIO DEL DESARROLLO DE LA TÉCNICA BÁSICA
PIANÍSTICA**

PROYECTO DE GRADO

Presentado por: Miguel Ángel Carballo Arispe

Como requisito parcial para optar al título de:

LICENCIATURA EN INGENIERÍA DE SISTEMAS COMPUTACIONALES

Tutor: Mgr. Marcel Barrero Mendizábal

Cochabamba, Junio 2015

CONTENIDO

RESUMEN EJECUTIVO.....	x
ABSTRACT	xi
I ESTUDIO DE DIAGNOSTICO	1
1.1 Introducción	1
1.2 Técnica Pianística.....	2
1.2.1 Definición	2
1.2.2 Escuelas Pianísticas	2
1.2.3 Aspectos Principales	6
1.3 Antecedentes Tecnológicos Aplicados a la Técnica Pianística	6
1.4 Necesidades y Problemas Actuales del Aprendizaje de Piano	9
1.5 Objetivos	11
1.6 Objetivos Específicos	11
II MARCO TEORICO.....	13
2.1 Captura de Posiciones de la Mano.....	13
2.1.1 Sensor Infrarrojo	15
2.1.2 Rastreo de la mano con Leap Motion	16
2.1.3 Manejo de Datos de la Mano con Leap Motion	18
2.1.3.1 Clase Frame	18
2.1.3.2 Clase Hand	19
2.1.3.3 Clase Finger	21
2.1.3.4 Clase Bone	23
2.1.3.5 Clase Vector	24
2.2 Representación en 3D con Mallas Poligonales	25
2.2.1 Representación teórica y gráfica.....	25
2.2.2 Traslación	27
2.2.3 Escalamiento.....	28
2.2.4 Rotación	29
2.2.5 Proyección Ortogonal	30
2.2.6 Proyección en Perspectiva	30
2.3 Serialización de objetos en Java	32
2.3.1 Almacenamiento de objetos en fichero.....	32

2.4	Sonido MIDI en Java	33
2.4.1	Clase MidiSystem	34
2.4.2	Clase Synthesizer	34
2.5	Generación de Gráficos Estadísticos en Java con JFreeChart	35
2.5.1	Clase ChartPanel	35
2.5.2	Clase JFreeChart	36
2.5.3	Interfaz XYDataSet	36
2.6	Exportación de Datos con JExcelApi	37
III	ESTUDIO DE ALTERNATIVAS.....	39
3.1	Especificaciones técnicas iniciales.....	39
3.1.1	Especificaciones técnicas: Hardware de captura de datos de la mano	39
3.1.2	Especificaciones técnicas: Lenguaje de programación para análisis de datos, generación de gráficos, audio e interfaz de usuario	43
IV	DISEÑO DEL PROTOTIPO.....	49
4.1	Determinación de especificaciones.....	49
4.2	Diagramas de casos de uso del sistema.....	51
4.2.1	Módulo Audiovisual	52
4.2.2	Módulo de Modificación Virtual	53
4.2.3	Módulo de Manejo de Archivos	54
4.2.4	Módulo de Captura y Procesamiento de Datos.....	56
4.3	Diagrama de Paquetes	59
4.3.1	Coordinación General	61
4.3.2	Captura de datos	62
4.3.3	Representación Virtual	62
4.3.4	Producción de Sonido	63
4.3.5	Producción Visual – Interfaz Gráfica	63
4.3.6	Modificación Virtual	64
4.3.7	Manejo de Archivos.....	65
4.4	Diagrama de Clases del sistema	65
4.5	Estructura Física Anexa	67
4.5.1	Superficie de vidrio.....	67
4.5.2	Marco	68
4.5.3	Vara Vertical.....	70

4.5.4	Vara Horizontal	71
4.5.5	Ensamblado de las partes.....	71
4.6	Validación del prototipo en la evaluación de la técnica pianística	75
V	EVALUACIÓN FINANCIERA.....	77
VI	CONCLUSIONES.....	79
VII	RECOMENDACIONES.....	80
VIII	BIBLIOGRAFÍA	81
IX	ANEXOS.....	88
Anexo 1: Explicación de Términos Musicales		
Anexo 2: Mallas Poligonales en Java		
Anexo 3: Descripción de casos de uso		
Anexo 4: Manual de Usuario		
Anexo 5: Explicación de Diagrama de Clases		

LISTA DE TABLAS

Tabla N° 1: Indicadores y ponderaciones de las alternativas para hardware de captura de datos de la mano	42
Tabla N° 2: Indicadores y ponderaciones de las alternativas para lenguaje de programación	47
Tabla N° 3: Caso de uso Procesar Datos del Escenario Virtual	58
Tabla N° 4: Costos de desarrollo de prototipo.....	77
Tabla N° 5: Valor de frecuencias de las notas desde La0 hasta Do8.....	91
Tabla N° 6: Descripción de caso de uso Procesar Datos Visuales	100
Tabla N° 7: Descripción de caso de uso Procesar Datos Sonoros	101
Tabla N° 8: Descripción de caso de uso Ver Escenario Virtual	101
Tabla N° 9: Descripción de caso de uso Ver Panel de Ángulos	102
Tabla N° 10: Descripción de caso de uso Escuchar Teclado.....	102
Tabla N° 11: Descripción de caso de uso Modificar Visualización	103
Tabla N° 12: Descripción de caso de uso Modificar Teclado Virtual	104
Tabla N° 13: Descripción de caso de uso Grabar Manos	105
Tabla N° 14: Descripción de caso de uso Generar Archivo	106
Tabla N° 15: Descripción de caso de uso Cargar Archivo	107
Tabla N° 16: Descripción de caso de uso Reproducir Archivo	108
Tabla N° 17: Descripción de caso de uso Modificar Reproducción.....	109
Tabla N° 18: Descripción de caso de uso Generar Gráfico Estadístico.....	110
Tabla N° 19: Descripción de caso de uso Guardar Gráfico	111
Tabla N° 20: Descripción de caso de uso Exportar a Excel	112

LISTA DE FIGURAS

Figura N° 1: Posición Básica 1 - Escuela Francesa	3
Figura N° 2: Posición Básica 2 – Escuela Francesa	3
Figura N° 3: Posición Básica 1 – Escuela Alemana	4
Figura N° 4: Posición Básica 2 – Escuela Alemana	4
Figura N° 5: Posición Básica 1– Escuela Rusa	5
Figura N° 6: Posición Básica 2 – Escuela Rusa	5
Figura N° 7: Guía-mano de Kalkbrenner	7
Figura N° 8: Anuncio publicitario de Dactylion.....	7
Figura N° 9: Imagen de Chirogimnaste	7
Figura N° 10: Imagen de Digitorium.....	8
Figura N° 11: Imagen de manual Gymnasium for Musicians	8
Figura N° 12: Proyector infrarrojo, cámara RGB, cámara infrarroja	14
Figura N° 13: Componentes hardware de Leap Motion	14
Figura N° 14: Representación Esquemática de la relación de Profundidad-Desplazamiento	15
Figura N° 15: Imagen de patrón de motas de luz infrarroja proyectadas en objeto	16
Figura N° 16: Imagen resultante de análisis de profundidad.....	16
Figura N° 17: Rastreo de manos con Leap Motion	17
Figura N° 18: Longitud e intensidad de onda producida por Leap Motion.....	18
Figura N° 19: Ejemplo de uso de clases Controller y Frame en Java	19
Figura N° 20: Ejemplo de instanciación de clase Hand.....	19
Figura N° 21: Código para captura y muestra de datos Hand en Java.....	20
Figura N° 22: Output de muestra de datos Hand en Java	21
Figura N° 23: Representación gráfica de vectores obtenidos con el método direction	22

Figura N° 24: Ejemplo de uso del método tipPosition	22
Figura N° 25: Ejemplo de uso del método tipVelocity	22
Figura N° 26: Código para captura y muestra de datos Finger en Java.....	23
Figura N° 27: Output de muestra de datos Finger en Java	23
Figura N° 28: Código para captura y muestra de datos Bone en Java.....	24
Figura N° 29: Output de muestra de datos Finger en Java	24
Figura N° 30: Sistema de Coordenadas usado en Leap Motion	25
Figura N° 31: Representación gráfica de vértices y aristas	26
Figura N° 32: Figura representada con mallas poligonales	26
Figura N° 33: Antes y después de traslación de una figura	27
Figura N° 34: Antes y después aplicar escalamiento a una figura.....	29
Figura N° 35: Proyección de imagen en cámara oscura	30
Figura N° 36: Proyección de un punto en cámara oscura	31
Figura N° 37: Implementación de interfaz Serializable.....	32
Figura N° 38: Ejemplo de código para escritura de objetos serializados en fichero	33
Figura N° 39: Ejemplo de código para lectura de fichero que contiene objetos serializados	33
Figura N° 40: Extracto de código Java para generar nota Do4 por 3 segundos	35
Figura N° 41: Panel estadístico generado con JFreeChart.....	37
Figura N° 42: Sección de código para generar archivo Excel con JExcelApi.....	38
Figura N° 43: Sección de archivo Excel generado con JExcelApi.....	38
Figura N° 44: Diagrama de casos de uso del Módulo Audiovisual.....	52
Figura N° 45: Diagrama de casos de uso del Módulo de Modificación Virtual.....	53
Figura N° 46: Diagrama de casos de uso del Módulo de Manejo de Archivos	54
Figura N° 48: Diagrama de Paquetes del Sistema	60

Figura N° 49: Diagrama de Clases del Sistema.....	66
Figura N° 50: Partes de la estructura metálica sin ser ensambladas	67
Figura N° 51: Marco de soporte - Vista frontal	68
Figura N° 52: Marco de soporte - Vista superior.....	68
Figura N° 53: Borde derecho y sección del borde posterior del marco de soporte	69
Figura N° 54: Borde izquierdo del marco de soporte	69
Figura N° 55: Seguro de presión de la parte posterior del marco de soporte	70
Figura N° 56: Vara vertical.....	70
Figura N° 57: Vara horizontal	71
Figura N° 58: Vara vertical y horizontal ensambladas	72
Figura N° 59: Dispositivo de presión de la vara vertical ensamblado al borde izquierdo del marco	72
Figura N° 60: Partes metálicas de estructura física anexa ensambladas.....	73
Figura N° 61: Partes metálicas de estructura física anexa sujetado a una mesa	73
Figura N° 62: Prueba de prototipo	74
Figura N° 63: Prototipo siendo utilizado por un estudiante	74
Figura N° 64: Estudiante y profesor de piano utilizando el prototipo	75
Figura N° 65: Gráfico estadístico de dedos 3 y 4 de la mano derecha de persona con un mes de estudio de piano	76
Figura N° 66: Gráfico estadístico de dedos 3 y 4 de la mano derecha de una persona con un año de estudio de piano	76
Figura N° 67: Dos octavas de teclado con los nombres de las notas en cada tecla	89
Figura N° 68: Gráfico de Frecuencias en Hz y distancia de La0 en Semitonos	95
Figura N° 69: Diagrama de Clases de Mallas Poligonales	97

A mis padres

AGRADECIMIENTOS

Quiero agradecer a Mgr. Marcel Barrero por acompañarme en todo el proceso de la elaboración del presente proyecto y regalarme sus observaciones, críticas y facilidades.

También agradecer a Lic. Koichi Fujii por sus valiosas observaciones en el proyecto y, en general, por todas las enseñanzas de vida.

Finalmente agradecer a los docentes de piano de Academia Nacional de Música “Man Césped” por brindarme sus opiniones para la mejora del proyecto.

RESUMEN EJECUTIVO

El aprendizaje de la técnica pianística es una de las prioridades de los estudiantes de piano en el mundo. Sin embargo, actualmente no existen registros del movimiento de los dedos, exceptuando filmaciones (vídeos) que no brindan datos precisos ni modelos técnicos a seguir.

La enseñanza y el aprendizaje del instrumento mantienen todavía un ritmo tradicionalista, lo que da como resultado un retraso general de la comunidad pianística en el proceso de aprendizaje de la técnica básica.

Por otra parte, el desarrollo tecnológico actual ha generado opciones, tales como los dispositivos infrarrojos, que pueden registrar posiciones de las manos con precisión y en tiempo real.

En el presente proyecto final de grado se desarrolló un prototipo de sistema que tiene como propósito ayudar al estudio y la investigación del desarrollo de la técnica básica pianística.

El prototipo del sistema fue desarrollado en Java y captura los datos de la posición de las manos (dedos y muñecas) mediante un dispositivo Leap Motion, crea representaciones gráficas de los huesos de las manos simulando un escenario virtual en 3D utilizando mallas poligonales, implementa un teclado virtual (el cual produce sonido MIDI) y permite al usuario grabar secuencias de movimientos de manos en archivos para posteriormente reproducirlos y analizarlos. También, a partir de la información guardada, el sistema permite crear gráficos estadísticos mediante la biblioteca JFreeChart y puede exportar los datos a archivos Excel mediante la biblioteca JExcelApi, permitiendo de esta manera mostrarlos en un formato conocido y manejable por la comunidad.

ABSTRACT

Learning piano technique is one of the priorities of piano students in the world. However, now a days there are no records of the movement of fingers, except films (videos) that do not provide accurate data or technical role models.

Teaching and learning piano still maintain a old traditionalist manners, which results in a general delay of piano community in the process of learning the basic technique.

However, the current technological development has generated options such as infrared devices that can record hand positions accurately and in real time.

In this Final Project, it was made a prototype system that aims to help the study and research of the development of basic piano technique.

The prototype system was developed in Java and captures data of the position of the hands (fingers and wrists) by Leap Motion device, creates graphical representations of the bones of the hands simulating a virtual scenario in 3D using polygonal mesh, implements a virtual keyboard (which produces MIDI sound) and brings the option to record sequences of hand movements in files that enables future replays and analysis. Also, the system can create statistical charts of stored information using the JFreeChart library and can export data to Excel files by JExcelApi library, thereby enabling display the data in a manageable format known by the community.

I ESTUDIO DE DIAGNOSTICO

1.1 Introducción

En la interpretación pianística, el desarrollo de la técnica es uno de los aspectos más importantes y a la vez más polémicos que existen, ya que actualmente no está establecido un solo tipo de escuela ni enseñanza consensuada a nivel mundial.

La enseñanza de la técnica para la interpretación del piano ha sido, desde la creación del instrumento, en torno al año 1700, investigada y desarrollada de una generación a otra, gracias a escuelas formadas por maestros que pasaban su conocimiento, generado a partir su propia experiencia y sus propios métodos, a sus estudiantes y discípulos. Estos estudiantes posteriormente se convertían en maestros que, a su vez, se encargaban de desarrollar su propia técnica en base a los conocimientos iniciales adquiridos y en base a los cambios estructurales del piano de su época.

La tradición de enseñanza formal de piano, de maestro a discípulo, ha sido tomada por todas las generaciones hasta el día de hoy. Sin embargo, existen libros accesibles (entre métodos, tratados y partituras), que explican cómo desarrollar la técnica necesaria para la correcta ejecución de obras de piano, siendo todos, argumentados por personas estudiosas del tema, que incluyen pianistas de conciertos, maestros de piano o incluso médicos. Por otra parte, si bien los libros escritos hasta ahora proponen opciones eficientes de estudio técnico (gracias a conocimientos adquirido mediante práctica o estudio teórico de anatomía), no existe un sistema de registro de movimiento de los mismos.

Gracias a la tecnología desarrollada, se podría pensar que existen diferentes tipos de sistemas de medición, estudio y control del desarrollo técnico pianístico. Sin embargo, estos son prácticamente inexistentes. El manejo tradicional de la enseñanza del piano incluye métodos utilizados desde hace dos siglos y medio, que pueden ser estudiados con la tecnología actual para maximizar la eficiencia técnica en el estudio del piano, ya que el desarrollo técnico optimizado derivaría en una directa mejora en la calidad de interpretación de los pianistas.

El aprendizaje de la técnica de piano está atribuido, en la mayoría de los casos, a dos factores: La capacidad de enseñanza del maestro y la disciplina del estudiante. Siendo el

segundo el más importante e imprescindible para llegar a un nivel medio u avanzado. Sin embargo, existen varias preguntas entre todos los estudiantes, aún cuando esta disciplina existe: ¿Cómo se debe desarrollar la técnica básica? ¿Qué tipo de técnica tienen los pianistas avanzados? ¿Cómo se mueven sus dedos? ¿Qué tipo de movimientos realizan? ¿Es efectivo lo que se hace actualmente? ¿Es efectivo el tipo de técnica indicada si la forma de cada mano y su proporción son diferentes?

Se espera que cada maestro responda a las anteriores preguntas o guíe al estudiante para que tenga una respuesta propia. Pero, la realidad es que muchas veces no existen respuestas definidas, por lo que se tiene que realizar una investigación paciente basada en la intuición, teoría y el oído, cuando previamente otros pianistas (tanto actuales como de generaciones anteriores) ya han desarrollado la misma investigación con resultados óptimos, pero no han dejado registros científicos de cómo reproducirlos.

Para el desarrollo técnico correcto de piano, es necesario tener paciencia, detalle y creatividad para la resolución de problemas. Sin embargo cada estudiante no siempre visualiza todos los aspectos cuando realiza la práctica y posteriormente estos se convierten en deficiencias y limitaciones que llegan a frustrar al estudiante, llevando a la gran mayoría, generación tras generación, a un estado de fracaso. Cuando no se ven resultados, se pierde el entusiasmo, si se pierde el entusiasmo, no se practica y en consecuencia, no se logran mejoras, convirtiéndose todo lo anterior en un círculo vicioso.

1.2 Técnica Pianística

1.2.1 Definición

Entre las definiciones de “técnica” que tiene la Real Academia de la Lengua Española, están la “Habilidad para ejecutar cualquier cosa” y “conjunto de procedimientos especiales de una ciencia o arte” (Real Academia Española). De forma específica, la técnica pianística, según Chang, es “La habilidad de ejecutar millones de pasajes pianísticos diversos, por tanto, no es una destreza, sino un agregado de muchas habilidades” (Chang, 2008).

1.2.2 Escuelas Pianísticas

Las bases de la Técnica Pianística no están todavía consensuadas a nivel mundial. A lo largo de la historia se dividieron en escuelas que tuvieron una gran importancia hasta mitad

del siglo XX. Las escuelas pianísticas se dividían según los paradigmas de maestros del siglo XVIII y siglo XIX (Lanza, 1997).

Se puede entender a una escuela, como una “comunidad pianística”, cuyos miembros comparten ciertos usos, procedimientos o, simplemente, una determinada actitud hacia la ejecución pianística (Narejos, 1998).

Las escuelas más importantes son:

- La Escuela Francesa, que se enfoca principalmente a la articulación digital, haciendo actuar a los dedos como pequeños martillos, minimizando el movimiento de la muñeca. En la Figura N° 1 y N° 2 se muestra las posiciones básicas iniciales de la Escuela Francesa.



Figura N° 1: Posición Básica 1 - Escuela Francesa

Fuente: Elaboración propia



Figura N° 2: Posición Básica 2 – Escuela Francesa

Fuente: Elaboración propia

-
- La Escuela Alemana, centrada en la utilización del peso del brazo y bastante vinculada con la Escuela Rusa en la utilización de la muñeca. En la Figura N° 3 y N° 4 se muestra las posiciones básicas iniciales de la Escuela Alemana.



Figura N° 3: Posición Básica 1 – Escuela Alemana

Fuente: Elaboración propia



Figura N° 4: Posición Básica 2 – Escuela Alemana

Fuente: Elaboración propia

- La Escuela Rusa, caracterizada por dominios de matices extremos (fuerte-suave), por la base inicial de control con dedos planos y por el temperamento. En la Figura N° 5 y N° 6 se muestra las posiciones básicas iniciales de la Escuela Rusa.



Figura N° 5: Posición Básica 1– Escuela Rusa

Fuente: Elaboración propia



Figura N° 6: Posición Básica 2 – Escuela Rusa

Fuente: Elaboración propia

Las posiciones básicas son una muestra de las posiciones donde cada escuela inicia el entrenamiento técnico, lo que no quiere decir que una es mejor que otra. Las escuelas han ido convergiendo a un núcleo central un poco más homogéneo, la cual desde la segunda mitad del siglo XX es llamada la “Escuela Internacional”, basada en criterios más pragmáticos que tradicionales (Narejos, 1998).

1.2.3 Aspectos Principales

Entre todas las escuelas, existen algunos puntos comunes como la independización, exactitud de movimientos y velocidad de los dedos, que, actualmente, no suelen ser científicamente evaluados con precisión. Entre los aspectos principales trabajados para conseguir una buena técnica pianística están (Fujii, 2005):

- Forma de la mano y curvatura de dedos, que sirve como base para el inicio de la técnica pianística, siendo decisivo para los movimientos básicos.
- Independencia de dedos, ya que debido a la anatomía humana no existe una independencia natural, lo que hace a este aspecto muy importante para el desarrollo. Entiéndase independencia de dedos como la capacidad de realizar movimientos en los dedos, dentro del rango de movimientos naturales posibles, que no afecten o perturben de manera notoria a la capacidad motora de otro dedo en la misma posición de la mano.
- Movimiento general de la mano, que sirve como recurso en otros aspectos más detallados como fraseo, movimientos extendidos, pasajes de pulgar y peso en los dedos.

1.3 Antecedentes Tecnológicos Aplicados a la Técnica Pianística

Los primeros intentos conocidos de utilizar la tecnología como una herramienta para el desarrollo y el estudio de la técnica pianística, existen desde Friedrich Kalkbrenner (1784-1849) y Carl Czerny (1791 - 1857) que fue discípulo de Ludwig van Beethoven y maestro de Franz Liszt.

En la Figura N° 7 se puede ver una herramienta creada por Kalkbrenner para obligar a encontrar la calidad de sonido mediante la fortaleza de dedo sin intervención del antebrazo (Garcia, 2010). El uso de recursos de este tipo a lo largo del siglo XIX demuestra el interés de los pedagogos por la tecnología como herramienta en la enseñanza de la técnica pianística.

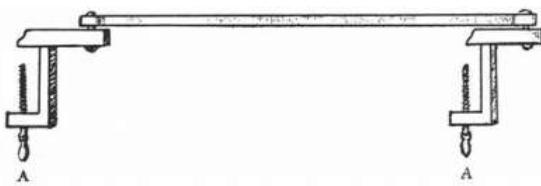


Figura N° 7: Guía-manos de Kalkbrenner

Fuente: (Garcia, 2010)

En la Figura N° 8, se puede ver el Dactylion, creado por el famoso pianista en su tiempo, compositor y fabricante de pianos Henri Herz (1803 - 1888). El objetivo de este aparato era poner una carga extra de trabajo al movimiento de bajada y de subida.



Figura N° 8: Anuncio publicitario de Dactylion

Fuente: (Garcia, 2010)

En las Figuras N° 9, N° 10 y N° 11 se pueden ver otros aparatos inventados en el siglo XIX y principios del siglo XX para ayudar en variedad de aspectos técnicos pianísticos.



Figura N° 9: Imagen de Chirogimmnaste

Fuente: (Garcia, 2010)

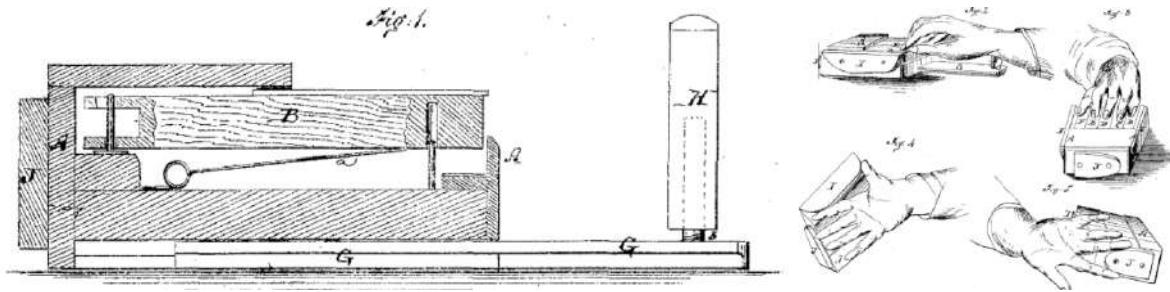


Figura N° 10: Imagen de Digitorium

Fuente: (Garcia, 2010)

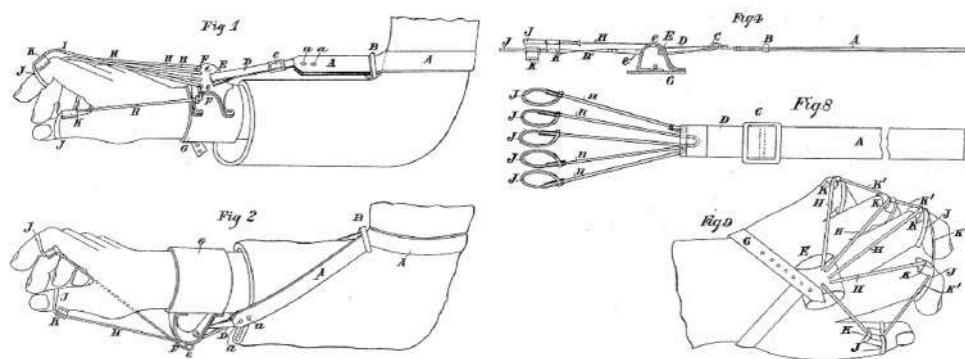


Figura N° 11: Imagen de manual Gymnasium for Musicians

Fuente: (Garcia, 2010)

Los aparatos mostrados anteriormente dejaron de ser utilizados casi en su totalidad debido a que, aplicados de manera inadecuada o de manera excesiva, llegaban a causar daños severos en los pianistas. Entre los casos más conocidos se puede citar al pianista y compositor Robert Schumann, que utilizó un artefacto de entrenamiento de dedo resultando dañado en el proceso (Worthen, 2007).

Sin embargo, los aparatos de entrenamiento técnico aplicados de manera correcta, aportaban beneficios a los que practicaban, siendo uno de ellos Franz Liszt, que se encuentra entre los pianistas más famosos debido a su técnica. Consiguió tal dominio de la técnica pianística, que es considerado, hasta ahora, un punto de perfección, más allá del cual no parece ser necesario, ni posible, ningún enriquecimiento esencial (Kaemper, 1968).

Dentro de la educación pianística, se diferencian dos posiciones: los detractores parciales de la técnica, que piensan que se toca fundamentalmente con el “corazón” y los que dan la

máxima importancia a la técnica, para que luego sirva a la parte interpretativa, como en el caso de Carl Czerny (Czerny, 1842).

Apoyando la idea de priorización técnica, existe un pensamiento de Chiantore, que expone de manera clara, la dependencia de la interpretación con la técnica:

“La moderna tendencia a escindir técnica y arte, denuncia una de las peculiaridades de nuestra cultura: la separación entre significado interior y forma exterior de una acción. Pero la actividad artística nos pone diariamente ante la necesidad de trascender semejante dualidad. En el caso de la música, la íntima unidad entre intención y realización, es decisiva y lo es aún más para los intérpretes que para los compositores.

En el momento de la ejecución, todo lo que hayamos aprendido, pensado o imaginado acerca de la obra pasa por el filtro de la materialidad de nuestra acción y de nuestra capacidad de entrar en sintonía con el instrumento. Y sin un dominio técnico a la altura de nuestra fantasía, las mejores intenciones nunca llegarán a concretarse” (Chiantore, 2001).

1.4 Necesidades y Problemas Actuales del Aprendizaje de Piano

En el desarrollo de la técnica pianística básica, que es el inicio del aprendizaje serio del instrumento, existen diversos factores que se deben tomar en cuenta para lograr un avance medio o avanzado en la ejecución del piano. Varios de estos factores dependen de la escuela (pianística), de la visión del profesor, del nivel y de la sensibilidad del estudiante de piano; que, siendo aislado parcialmente del conocimiento detallado de las escuelas existentes y de los avances técnicos de los demás pianistas, tiene que confiar en su propia intuición, la experiencia de su profesor y a una cantidad muy limitada de libros (a comparación de otras ramas de la música) que, normalmente, no son muy explicativos o tienen una visión muy parcializada de la técnica. Sin embargo desde el siglo XX, gracias a grabaciones y, posteriormente, filmaciones, se tiene una cantidad amplia de material para poder guiarse en algunos aspectos técnicos, pero ninguno llega a ser muy ilustrativo, pues, en la mayoría de los casos no se tiene una visión completa de los tipos de movimientos que realizan los dedos. Por lo tanto, no existen parámetros de evaluación (excepto el oído y en limitados casos la vista), ni registro de éstos, que incluya la posición, la velocidad de respuesta, ángulos utilizados de los dedos y sus tipos de movimiento.

Por otra parte, también se debe hacer énfasis en el hecho que se puede llegar a desarrollar la técnica de manera empírica o experimental hasta un grado avanzado. Entre los siglos XVII y XIX, la música se practicaba cotidianamente como algo común entre las personas europeas citadinas y la calidad de músicos era mayor en proporción a la población pianística. Sin embargo los estudios pianísticos no eran tomados como una carrera profesional que requerían un sistema educacional (Puelles López, 2004).

Las causas que llevan al estancamiento en el desarrollo de la técnica de piano son: el aislamiento de conocimientos técnicos, la falta de capacidad de transmisión de los conocimientos técnicos, falta de capacidad de recepción y asimilación de conocimientos técnicos, confusión de conceptos técnicos, divergencia de opiniones y contradicción de posiciones expertas sobre aspectos técnicos.

Otro problema existente de manera implícita es el modo de vida de la sociedad. Actualmente, la sociedad está erguida por sistemas de funcionamiento que buscan la facilidad de la vida cotidiana, disminuyendo en algunos aspectos la actividad del hombre, incluyendo escuchar con minuciosidad. La disminución de actividad intelectual se ve ligada a la capacidad de observación del humano por ser la principal desarrolladora de la misma, por lo tanto, se concluye que cuanto menos actividad intelectual, menos capacidad de observación y menos capacidad auditiva. Esto se ve relacionado a los músicos por la actividad intelectual, ya que un músico no puede persistir si no tiene capacidad intelectual ni capacidad auditiva, la que puede ser principalmente adquirida desde la infancia o niñez o por esfuerzo propio de la persona (Krakenberger, 2001).

Por lo tanto se concluye que la actual capacidad de los humanos está relacionada a la ideología y metodología de educación y es un factor de la disminución de músicos eficientes. Por esta razón, se debe buscar nuevas herramientas donde se aprovechen las tecnologías conocidas para optimizar el aprendizaje del piano que refuerzen los puntos débiles existentes en el proceso, ya que no existe ningún sistema tecnológico de evaluación de desarrollo de la técnica pianística en el mercado, como tal.

Existen prototipos de evaluación de movimientos de dedos, orientados a rehabilitación médica en la Universidad de Rochester, New York, que está incluido en un proyecto llamado “*Quantifying the Independence of Human Finger Movements*” (Hager-Ross &

Schieber, 2000), donde utilizan cámaras, como medio de captura de imágenes que, posteriormente, analizan para verificar la independencia de los dedos en pacientes.

La falta de registros científicos de los movimientos y las posiciones realizadas en la técnica de piano lleva actualmente un lento desarrollo a nivel mundial siendo porcentualmente muy pocos los capacitados que llegan a desarrollar una técnica avanzada ya sea gracias a su propio esfuerzo, paciencia, detallismo y/o mayor capacidad intuitiva y/o gracias a un docente altamente capacitado.

Se necesita proponer diversos mecanismos para evaluar y desarrollar la técnica pianística aprovechando los recursos tecnológicos existentes y, de esta manera realizar un estudio con registros y pruebas científicas. Esta tarea es extensa y requiere todo un conjunto de conocimientos tanto tecnológicos, anátomicos, artísticos y educativos para poder llevar a cabo todas las investigaciones. Pese a que es necesario todo un equipo de personas especialistas en múltiples disciplinas para establecer una rama de investigación considerable, también es necesario hacer pruebas y prototipos iniciales para definir las fortalezas que se pueden aprovechar y determinar las debilidades que deberían ser reforzadas en futuros proyectos destinados a este estudio.

1.5 Objetivos

3.1 Objetivo General

“Desarrollar un prototipo tecnológico que facilite el estudio del desarrollo técnico pianístico, haciendo uso de hardware que permita la captura de datos de las posiciones de la mano, para poder analizar sus movimientos en un entorno que simule un teclado virtual con sus características básicas, represente los componentes de las manos gráficamente y permita visualizar los datos con reportes y gráficos estadísticos”.

1.6 Objetivos Específicos

Para lograr el Objetivo General arriba propuesto, se han identificado los siguientes Objetivos Específicos:

1. Identificar la alternativa óptima para la captura de datos de las posiciones de la mano

-
-
2. Capturar datos de las posiciones de la mano y representarlos gráficamente en un ambiente 3D
 3. Crear un teclado virtual de mínimamente un intervalo de octava con sus características básicas: representación gráfica en 3D y producción de sonido
 4. Guardar los datos en archivos para estudiar el movimiento de las manos a posteriori
 5. Generar gráficos estadísticos en base a los datos de las posiciones de la mano

II MARCO TEORICO

En este capítulo se describen las herramientas y los fundamentos necesarios para el desarrollo del prototipo de sistema computarizado para el estudio del desarrollo de la técnica básica pianística.

Para exponer algunos principios y bases teóricas se utiliza el lenguaje Java, debido a que en el proyecto se trabajó con dicho lenguaje. Los motivos de esta elección están expuestos en el Capítulo III Estudio de Alternativas.

2.1 Captura de Posiciones de la Mano

La mano humana es una extremidad prensil con cinco dedos localizada al final del brazo, está compuesta de 27 huesos, además de varios músculos y ligamentos diferentes que permiten una gran cantidad de movimientos y destreza (Schmidt & Lanz, 2003). La captura de las posiciones de la mano con cualquier sistema de reconocimiento es, debido a su capacidad de movimientos, complicada. En general, cualquiera de los sistemas de reconocimiento tienen actualmente varias dificultades por superar, ya que la información de un mismo objeto puede variar de acuerdo al ambiente, a la distancia e incluso al ángulo de percepción, lo que marca sustancialmente los resultados dados por el reconocedor (Rodríguez, Pineda, & Sánchez, 2013).

El desarrollo de dispositivos de reconocimiento mediante ondas electromagnéticas en el rango infrarrojo para la interacción con la computadora, ha dado la posibilidad para que, de manera poco costosa, se pueda acceder a la tecnología necesaria para el desarrollo de un sistema basado en las capturas del movimiento de la mano. Dentro de estos dispositivos se encuentran el Kinect y el Leap Motion (Ramirez, 2012).

Kinect es un controlador creado originalmente para el entretenimiento. Permite a los usuarios interactuar con la consola, sin necesidad de contacto físico, gracias a los emisores y sensores infrarrojos que posee (Khoshelham, 2011).

En la Figura N° 12 se puede observar, los proyectores y cámaras que tiene un Kinect.

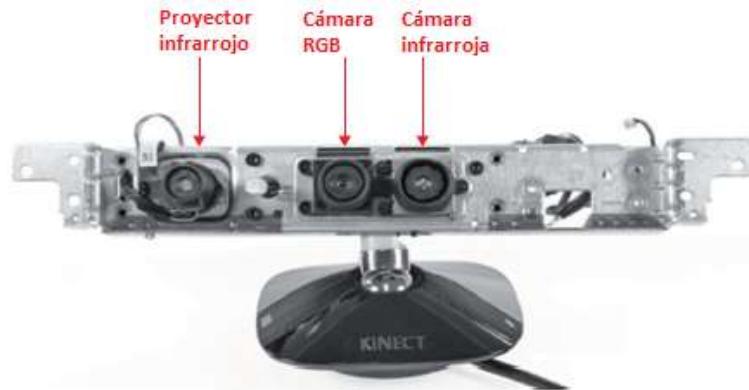


Figura N° 12: Proyector infrarrojo, cámara RGB, cámara infrarroja

Fuente: (Zhang, 2012)

Otro dispositivo que utiliza una tecnología similar es el Leap Motion. Mediante emisores y sensores infrarrojos (Ver Figura N° 13), rastrea el movimiento de la mano y los dedos. Está orientado a proveer un nuevo nivel de interacción con la computadora, de manera natural (Leap Motion).



Figura N° 13: Componentes hardware de Leap Motion

Fuente: (Gill)

2.1.1 Sensor Infrarrojo

El proyector o led infrarrojo, emite un haz principal, que se divide en múltiples puntos de luz infrarroja mediante una rejilla de difracción. Esto para crear un patrón proyectado sobre la escena. Este patrón es capturado por la cámara infrarroja y se correlaciona contra un patrón de referencia. El patrón de referencia se obtiene mediante la captura de un plano a una distancia conocida desde el sensor, que se guarda en la memoria del sensor. Cuando un punto de luz infrarroja se proyecta sobre un objeto a una distancia mayor o menor del plano de referencia, la posición del punto es desplazada en la dirección de la línea base entre el proyector y la perspectiva central de la cámara infrarroja. A continuación, en la Figura N° 14, se observa una imagen que esquematiza lo anteriormente explicado.

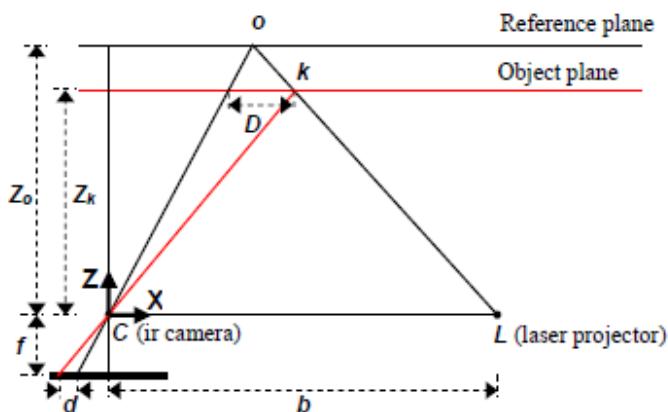


Figura N° 14: Representación Esquemática de la relación de Profundidad-Desplazamiento

Fuente: (Khoshelham, 2011)

Estos desplazamientos se miden para cada uno de los puntos proyectados, en un procedimiento de correlación de imágenes, que muestra la disparidad existente. A continuación, en la Figura N° 15 se puede observar la imagen del patrón de luz y la imagen resultante del procedimiento de correlación en la Figura N° 16.

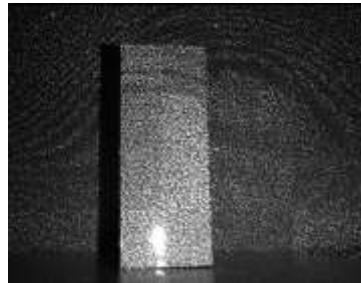


Figura N° 15: Imagen de patrón de motas de luz infrarroja proyectadas en objeto

Fuente: (Khoshelham, 2011)



Figura N° 16: Imagen resultante de análisis de profundidad

Fuente: (Khoshelham, 2011)

2.1.2 Rastreo de la mano con Leap Motion

Leap Motion ha sido creado de manera específica, para capturar el movimiento de la mano y de los dedos (Leap Motion). Su Kit de Desarrollo de Software o SDK (por su nombre en inglés: *Software Development Kit*) provee métodos que brindan directamente la opción de trabajar con el muestreo de la mano que ya ha sido elaborada y, además, brinda opciones para el manejo en diferentes lenguajes de programación: C++, C#, Objective-C, Java, JavaScript, Python (Leap Motion Inc., 2014).

En la Figura N° 17 se puede observar una captura de imagen de la aplicación de demostración, utilizada en el lanzamiento de Leap Motion. En esta imagen, se aprecia la representación de las manos en la computadora.

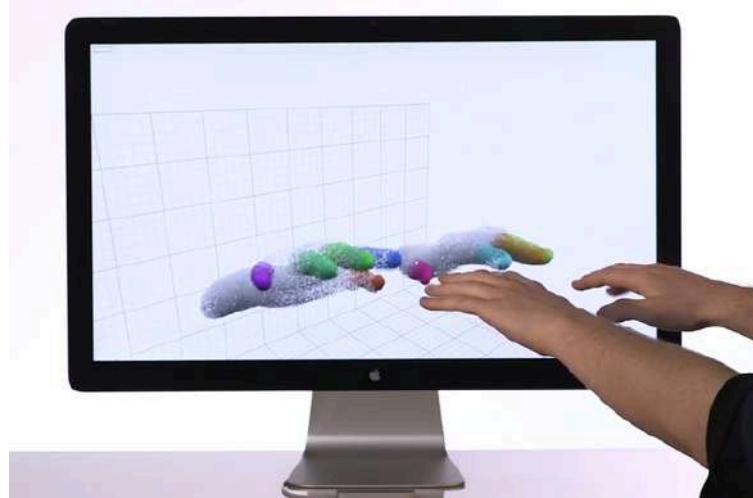


Figura N° 17: Rastreo de manos con Leap Motion

Fuente: (Gill)

Los procedimientos para realizar las muestras de las manos no son de carácter público y están directamente atribuidas a Leap Motion Inc. La fuente inicial de datos para la programación utilizando esta herramienta, es casi en su totalidad, el sitio web de la compañía, dedicado específicamente al desarrollo de aplicaciones (Leap Motion Inc., 2014).

Según el análisis de espectrometría que se realizó, se pudo conocer el tipo de ondas electromagnéticas que emite Leap Motion. El análisis se realizó con el propósito identificar los materiales que pueden ser útiles o perjudiciales en el diseño del prototipo, ya que se planificó utilizar una superficie de apoyo para las manos que no afecte o afecte en la menor medida posible a la luz infrarroja, tal como se explica en el Capítulo IV, Diseño del Prototipo.

En la Figura N° 18 se puede observar el resultado que se obtuvo mediante el software de análisis *Ocean Optics Spectra Suite* (SpectraSuite, 2012). Leap Motion trabaja con ondas electromagnéticas en el rango infrarrojo, es decir, longitudes de onda mayores a 750 nm (Frenzel, 2003), ya que las dos longitudes que proyecta son aproximadamente de 817 nm y de 863 nm. Por lo tanto, el vidrio común se puede considerar como material útil para servir como base de apoyo de las manos sin afectar a la captura de datos por Leap Motion debido a su grado de transparencia en el rango infrarrojo.

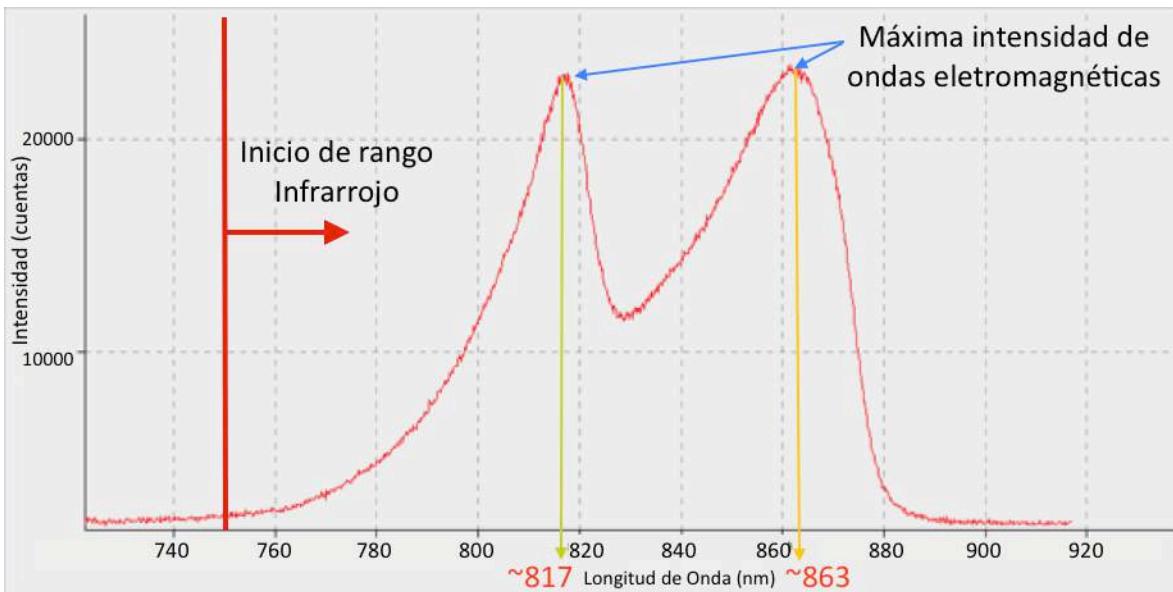


Figura N° 18: Longitud e intensidad de onda producida por Leap Motion

Fuente: Elaboración propia

2.1.3 Manejo de Datos de la Mano con Leap Motion

El dispositivo Leap Motion, como ya se mencionó anteriormente, reconoce automáticamente la posición de las manos y, mediante la importación de las bibliotecas que se distribuye mediante su página de desarrolladores y gracias a su API (*Application Programming Interface*), se pueden utilizar los datos que envía el dispositivo a la computadora en los lenguajes de programación preestablecidos.

2.1.3.1 Clase Frame

El Leap Motion API hace seguimiento a los datos de movimientos con una serie de capturas de escena llamados *frames*. Cada objeto *frame* contiene la información de cierto instante, incluyendo las manos (*Hands*), apuntadores (*Pointables*) que pueden ser dedos u objetos similares a dedos. 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* que es la interfaz principal del Leap Motion (Leap Motion Inc.).

A continuación la Figura N° 19 muestra un ejemplo de cómo extraer los datos de los objetos captados por el sistema de Leap Motion.

```

Controller controller = new Controller();
// wait until Controller.isConnected() evaluates to true
//...

Frame frame = controller.frame();
HandList hands = frame.hands();
PointableList pointables = frame.pointables();
FingerList fingers = frame.fingers();
ToolList tools = frame.tools();

```

Figura N° 19: Ejemplo de uso de clases Controller y Frame en Java

Fuente: ([Leap Motion Inc.](#))

2.1.3.2 Clase Hand

Leap Motion reconoce automáticamente la mano o varias manos y deriva el control de los datos a una clase Hand. La clase Hand es parte de una lista y, ésta, a su vez, de un Frame. Da acceso a atributos que dependen de él, como en el caso de los dedos o los apuntadores (*Pointables*) que puede llegar a ser cualquier objeto alargado (Leap Motion Inc). En la Figura N° 20 se pueden ver algunas de las características mencionadas: la inclusión en el Frame, la lista de objetos Hand y la asignación del primer miembro de la lista en variable firstHand.

```

Frame frame = controller.frame();
HandList hands = frame.hands();
Hand firstHand = hands.get(0);

```

Figura N° 20: Ejemplo de instanciación de clase Hand

Fuente: ([Leap Motion Inc.](#))

Un objeto Hand proporciona la información de la posición 3D, características y movimiento de la mano detectada y también de los dedos y objetos asociados a ella. Leap Motion puede detectar más de dos manos a la vez en el caso de que haya más de una persona con sus manos en el campo de visión. Para cada mano rastreada, se tienen esencialmente los siguientes atributos que contienen la información sobre la posición del centro de la palma, la velocidad de la palma, normal de la palma, la dirección de la mano,

posición de la muñeca, el centro de la esfera que forma y el radio de curvatura (López, 2013).

Hay que hacer notar que los Pointable se asocian con Hand, de forma que se accede a las variables de cada Pointable a través de un objeto Hand. Sin embargo, también puede existir algún Pointable no asociado a alguna mano, por estar parcialmente en el campo de rastreo del Leap Motion. A cada Pointable o Hand detectado se le asigna un identificador único, válido mientras el objeto esté siendo captado por Leap Motion. Sin embargo, si en algún momento el objeto deja de ser válido (por haber salido del campo de vista del Leap Motion o por haber sido oculto por otra mano o dedo) y posteriormente vuelve a ser válido, se asigna un nuevo identificador, de valor aleatorio. Es decir, el identificador ID sólo es válido desde el primer frame en que se rastrea hasta que desaparece (López, 2013).

En la Figura Nº 21 se puede observar un ejemplo de una implementación sencilla con datos de salida en texto mostrados en la Figura Nº 22.

```
Frame frame = controller.frame();
System.out.println("Frame ID: " + frame.id());
//Captura de manos
for(Hand hand : frame.hands()) {
    String handType = hand.isLeft() ? "Mano Izquierda" : "Mano Derecha";
    System.out.println(" " + handType + ", id: " + hand.id()
        + ", posicion de la palma: " + hand.palmPosition() +
        "\n      , posicion de la muñeca" + hand.wristPosition());

    // Captura vector normal y dirección de la mano
    Vector normal = hand.palmNormal();
    Vector direction = hand.direction();

    // Calcula ángulos de la mano pitch = X, roll = Y, and yaw = Z
    System.out.println(" pitch X: " + Math.toDegrees(direction.pitch()) + " grados, "
        + "roll Y: " + Math.toDegrees(normal.roll()) + " grados, "
        + "\n      yaw Z: " + Math.toDegrees(direction.yaw()) + " grados");
}
}
```

Figura Nº 21: Código para captura y muestra de datos Hand en Java

Fuente: Elaboración propia

```
Frame ID: 489125
Mano Izquierda, id: 190, posicion de la palma: (-7.9861, 83.0778, -26.803)
, posicion de la muñeca(-18.5165, 67.0824, 15.7644)
pitch X: 23.587764600504727 grados, roll Y: 11.891692316714238 grados,
yaw Z: 4.700078137241562 grados
```

```
Frame ID: 489126
Mano Izquierda, id: 190, posicion de la palma: (-8.06512, 82.2147, -25.7046)
, posicion de la muñeca(-18.7125, 65.8403, 16.6893)
pitch X: 24.057519398408814 grados, roll Y: 12.31359651684051 grados,
yaw Z: 4.858187198130425 grados
```

Figura N° 22: Output de muestra de datos Hand en Java

Fuente: Elaboración propia

2.1.3.3 Clase Finger

La clase `Finger`, representa al seguimiento o rastreo de un dedo. Los `Finger` son un tipo de `Pointable` que Leap Motion consigue rastrear. Estos objetos son atributos de `Frame` y de `Hand`.

Los dedos están asociados de forma permanente a una mano, por lo tanto el `ID` de `Finger` no va a variar mientras la mano o `Hand` no salga del campo de visión del Leap Motion. Debido a que es posible que los dedos se muevan dentro y fuera del área de reconocimiento, también es posible que el `ID` inicialmente asignado no sea el mismo.

Entre los métodos más útiles de la clase `Finger` está el método `direction` que devuelve un vector que expresa la dirección de la última falange del dedo (Leap Motion Inc.).

En la Figura N° 23 se puede observar la representación gráfica del vector de dirección de cada dedo.

Otro método muy útil es el `tipPosition`, devuelve un vector que representa la posición de la punta del dedo en unidad de milímetros, esta posición es calculada tomando en cuenta al origen mismo dentro desde el dispositivo del Leap Motion (Leap Motion Inc.).



Figura N° 23: Representación gráfica de vectores obtenidos con el método direction

Fuente: (Leap Motion Inc.)

En la Figura N° 24 se puede observar un vector currentPosition donde se almacena el dato de la mano mediante el método tipPosition de la clase Pointable de la que hereda la clase Finger.

```
Vector currentPosition = pointable.tipPosition();
```

Figura N° 24: Ejemplo de uso del método tipPosition

Fuente: (Leap Motion Inc.)

El método tipVelocity devuelve el vector que representa la velocidad de cambio de la posición de la palma en milímetros sobre segundo. En la Figura N° 25 se observa un ejemplo de la asignación de un vector llamado currentSpeed que almacena datos obtenidos mediante el método tipVelocity.

```
Vector currentSpeed = pointable.tipVelocity();
```

Figura N° 25: Ejemplo de uso del método tipVelocity

Fuente: (Leap Motion Inc.)

En la Figura N° 26 se puede observar un ejemplo de una implementación sencilla con datos de salida en texto mostrados en la Figura N° 27.

```

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 N° 26: Código para captura y muestra de datos Finger en Java

Fuente: Elaboración propia

```

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 N° 27: Output de muestra de datos Finger en Java

Fuente: Elaboración propia

2.1.3.4 Clase Bone

La clase Bone representa a un hueso de la mano que ha conseguido ser rastreado por el Leap Motion. Cada Finger (dedo) tiene 4 Bone (hueso) excepto el pulgar (Finger tipo Thumb). 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.

La clase Bone tiene métodos como direction (dirección), length (largo), width (ancho) que proveen información muy útil para poder realizar una representación gráfica de cada hueso (Leap Motion Inc.).

En la Figura Nº 28 se puede observar un ejemplo de una implementación sencilla con datos de salida en texto mostrados en la Figura Nº 29.

```
for (Finger finger : hand.fingers()) {  
    for(Bone.Type boneType : Bone.Type.values()) {  
        Bone bone = finger.bone(boneType);  
        System.out.println("      " + bone.type()  
            + " hueso, inicio: " + bone.prevJoint()  
            + "\n              , fin: " + bone.nextJoint()  
            + "\n              , direccion: " + bone.direction());  
    }  
}
```

Figura Nº 28: Código para captura y muestra de datos Bone en Java

Fuente: Elaboración propia

```
Frame ID: 561398  
TYPE_METACARPAL hueso, inicio: (28.276, 78.1646, 84.8572)  
      , fin: (28.276, 78.1646, 84.8572)  
      , direccion: (0, 0, 0)  
TYPE_PROXIMAL hueso, inicio: (28.276, 78.1646, 84.8572)  
      , fin: (48.679, 80.7375, 45.3171)  
      , direccion: (-0.457791, -0.057729, 0.887183)  
TYPE_INTERMEDIATE hueso, inicio: (48.679, 80.7375, 45.3171)  
      , fin: (52.0858, 78.6142, 15.1411)  
      , direccion: (-0.111913, 0.0697478, 0.991267)  
TYPE_DISTAL hueso, inicio: (52.0858, 78.6142, 15.1411)  
      , fin: (38.3728, 72.4362, 0.635666)  
      , direccion: (0.656266, 0.295662, 0.694189)
```

Figura Nº 29: Output de muestra de datos Finger en Java

Fuente: Elaboración propia

2.1.3.5 Clase Vector

La clase Vector representa un vector matemático de tres componentes o un punto tal como una dirección o posición en el espacio tridimensional. El Leap Motion emplea el sistema de coordenadas cartesianas diestro. Los valores dados son en unidades de milímetros del mundo real. El origen se centra en el centro del controlador del Leap Motion. Los ejes X y Z se encuentran en el plano horizontal, con el eje X paralelo al borde largo del dispositivo. El eje Y es vertical, con valores positivos creciente hacia arriba (en contraste con la orientación hacia abajo de la mayoría de los gráficos por ordenador sistemas de

coordenadas). El eje Z tiene valores positivos que crecen cada vez que la distancia a la pantalla del ordenador aumenta. En la Figura N° 30 se puede observar la representación del sistema de coordenadas que se usa en el Leap Motion.

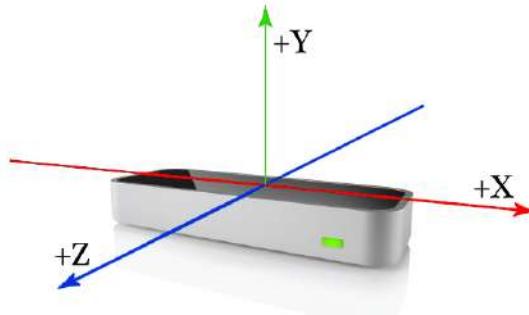


Figura N° 30: Sistema de Coordenadas usado en Leap Motion

Fuente: (Leap Motion Inc., 2014)

2.2 Representación en 3D con Mallas Poligonales

Una malla poligonal es un conjunto de vértices, aristas y caras que definen la forma de un objeto poliédrico en gráficos por ordenador en 3D y modelado de sólidos. Las caras por lo general constan de triángulos u otro polígono simple (Foley, Dam, Feiner, & Hughes, 1996)

La explicación de la representación en 3D con el lenguaje Java puede ser encontrada en Anexo 2.

2.2.1 Representación teórica y gráfica

Una malla poligonal se representa mediante una lista de vértices y una lista de aristas que se representan como la interconexión de vértices (Foley, Dam, Feiner, & Hughes, 1996). Lo anterior se puede representar de la siguiente manera, donde V es el conjunto de vértices, E es un conjunto de aristas y P es un polígono:

$$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), (x_4, y_4, z_4)) \quad (1)$$

$$E = (E_1, E_2, E_3, E_4) = ((V_1, V_2), (V_2, V_3), (V_3, V_4), (V_4, V_1)) \quad (2)$$

$$P = (V, E) \quad (3)$$

En (1) se puede observar que la componen cuatro puntos, que son los vértices. En (2) se definen las aristas, que a su vez están compuestas cada una por dos vértices y, finalmente en (3) se define un polígono simple como resultado compuesto por el conjunto V de vértices y E de aristas. Lo anterior se puede ver representado de manera gráfica en la Figura N° 31.

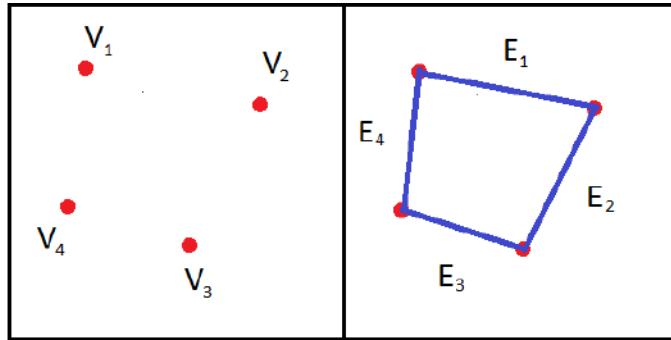


Figura N° 31: Representación gráfica de vértices y aristas

Fuente: Elaboración propia

Las mallas poligonales pueden representar el esquema de figuras 3D de manera simple, ya que no se toma en cuenta el comportamiento de la luz, lo que ahorra mucho tiempo en el procesamiento de la imagen y además se pueden realizar transformaciones geométricas como traslación, proyección y rotación (Foley, Dam, Feiner, & Hughes, 1996). En la Figura N° 32 se puede ver un ejemplo de una figura creada con mallas poligonales en Java utilizando herramientas de dibujo 2D.

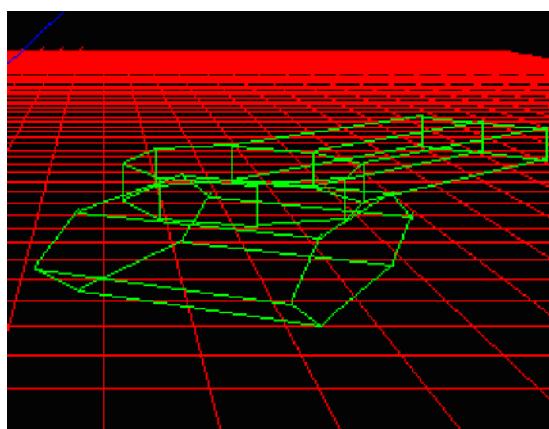


Figura N° 32: Figura representada con mallas poligonales

Fuente: Elaboración propia

2.2.2 Traslación

Para realizar la traslación de vértices a una nueva posición se suma la cantidad de traslación a los respectivas coordenadas. Para cada punto $P(x, y)$ a ser movido d_x unidades paralelos al eje x y d_y paralelo al eje y para llegar a ser el nuevo punto $P'(x', y')$, se puede escribir

$$x' = x + d_x \quad y' = y + d_y \quad (4)$$

Si se escribe en vectores columna, se puede definir P , P' y T

$$P = \begin{bmatrix} x \\ y \end{bmatrix}, P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \quad (5)$$

De esta manera (4) se puede escribir concisamente y generalizando para todas las dimensiones

$$P' = P + T \quad (6)$$

Se puede trasladar cualquier objeto representado por puntos aplicando la ecuación (6) para cada punto del objeto. Como cada línea tiene infinita cantidad de puntos, este proceso tomaría una infinita cantidad de tiempo en ser realizado. Sin embargo, se puede trasladar todos los puntos de una línea trasladando los puntos extremos y realizar un nuevo dibujo de línea entre los puntos trasladados (Foley, Dam, Feiner, & Hughes, 1996, p. 202). En la Figura N° 33 se muestra el efecto de trasladar una figura con el punto $(3, -4)$.

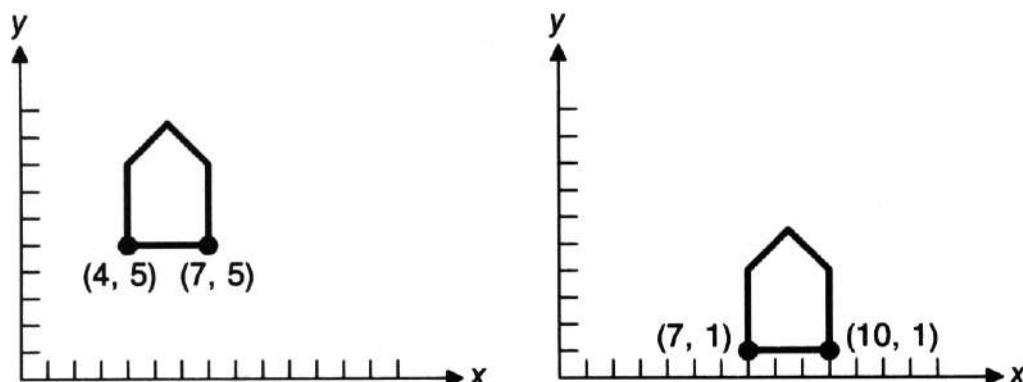


Figura N° 33: Antes y después de traslación de una figura

Fuente: (Foley, Dam, Feiner, & Hughes, 1996, p. 202)

2.2.3 Escalamiento

Los puntos de una figura pueden ser también escalados (aumentados y reducidos), por un coeficiente s_x en el eje x y por s_y en el eje y , para ser convertidos en nuevos puntos mediante multiplicaciones (Foley, Dam, Feiner, & Hughes, 1996, p. 202). Para cada punto $P(x, y)$ a ser escalado s_x unidades paralelos al eje x y s_y paralelo al eje y para llegar a ser el nuevo punto $P'(x', y')$, se puede escribir

$$x' = s_x \cdot x \quad y' = s_y \cdot y \quad (7)$$

Escrito de forma matricial, esto es equivalente a

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \quad (8)$$

Lo que se puede definir de forma generalizada para todas las dimensiones como

$$P' = S \cdot P \quad (9)$$

Siendo S una matriz cuadrada de dimensión n donde su diagonal principal contiene los coeficientes de escala para sus respectivos componentes.

Para que se realice una escala proporcional que produzca un efecto de alejamiento o acercamiento, los coeficientes de escala deben ser iguales. A esto se llama escalamiento uniforme.

En la Figura N° 34 se observa el escalamiento diferenciado con $s_x = \frac{1}{2}$ y $s_y = \frac{1}{4}$, es decir que no se usa el mismo valor para todos los coeficientes de escala.

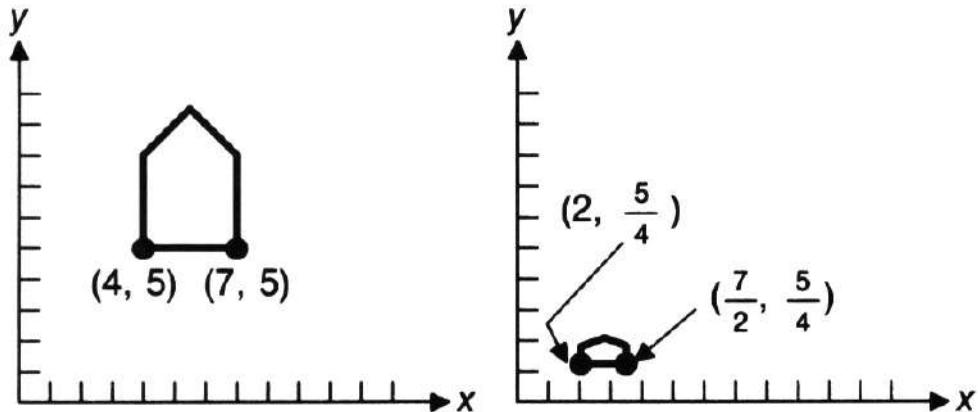


Figura N° 34: Antes y después aplicar escalamiento a una figura

Fuente: (Foley, Dam, Feiner, & Hughes, 1996, p. 202)

2.2.4 Rotación

Los puntos de una figura pueden ser rotados con un ángulo θ a partir del origen. Siendo el punto $P(x, y)$ a ser rotado un ángulo θ y llegar a ser el punto $P'(x', y')$, la rotación está definida por

$$x' = x \cdot \cos \theta - y \cdot \sin \theta, \quad y' = x \cdot \sin \theta - y \cdot \cos \theta \quad (10)$$

Escrito de forma matricial, lo anterior es equivalente a

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \quad (11)$$

Lo que se puede definir de forma generalizada para todas las dimensiones como

$$P' = R \cdot P \quad (12)$$

Como se puede deducir de (12), la rotación en 2D será a través de un plano, lo que hará que en 3D varíe la matriz R dependiendo el eje sobre el que se va a realizar la rotación, siendo tres las matrices posibles en R (Foley, Dam, Feiner, & Hughes, 1996, p. 215)

Rotación en el eje z

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (13)$$

Rotación en el eje x

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \quad (14)$$

Rotación en el eje y

$$R_z(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad (15)$$

2.2.5 Proyección Ortogonal

Una proyección es una representación gráfica de un objeto en una superficie, por lo tanto se realiza una conversión de la posición de los puntos de un objeto que están en 3D para asignarles un valor en 2D.

La proyección ortogonal es la representación más simple, ya que para calcular la posición en 2D, dependiendo en qué plano se quiere proyectar la figura, se procede a eliminar el tercer componente (Foley, Dam, Feiner, & Hughes, 1996, p. 234).

2.2.6 Proyección en Perspectiva

Para dar la sensación de profundidad se puede usar como concepto de las cámaras oscuras que, mediante la entrada de luz a un orificio proyectan una imagen de manera inversa y en perspectiva, tal como se representa en la Figura N° 35.

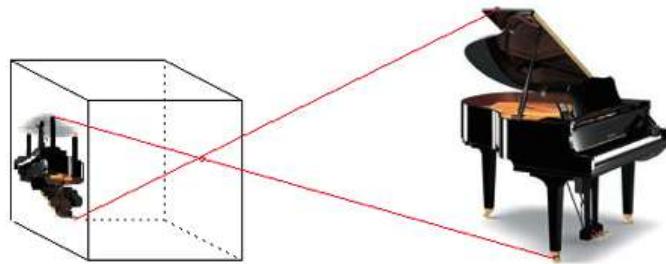


Figura N° 35: Proyección de imagen en cámara oscura

Fuente: Elaboración propia

La conversión de un punto 3D a un punto proyectado en perspectiva 2D se puede esquematizar en el dibujo que se muestra en la Figura N° 36

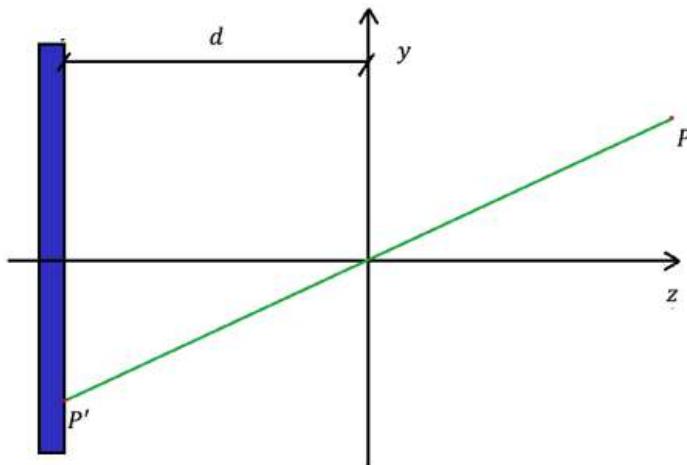


Figura N° 36: Proyección de un punto en cámara oscura

Fuente: Elaboración propia

Se puede notar que el esquema de la Figura N° 36 sirve también para los ejes x y z , donde la relación entre el negativo del componente y y x del punto P' y d es la misma que la relación del componente y y x del punto P y el componente z del punto P respectivamente, lo que genera la relación

$$\frac{-P'_y}{d} = \frac{P_y}{P_z}, \frac{P'_x}{d} = \frac{P_x}{P_z} \quad (16)$$

De donde se puede deducir que los componentes de P' , corrigiendo la inversión de imagen, son

$$P'_x = \frac{d P_x}{P_z}, P'_y = \frac{d P_y}{P_z}, P'_z = d \quad (17)$$

Graficando los puntos P' , tomando en cuenta los componentes en x y y , creados a partir de P según la ecuación (17), se obtiene la representación en 2D de una objeto en 3D donde d representa la proporción de cercanía o alejamiento del objeto.

2.3 Serialización de objetos en Java

La serialización de un objeto consiste en obtener una secuencia de bytes que represente el estado de dicho objeto. Esta secuencia puede utilizarse de varias maneras, puede enviarse a través de la red, guardarse en un fichero para su uso posterior y utilizarse para recomponer el objeto original.

Un objeto se puede serializar si implementa la interfaz `Serializable`. Esta interfaz no declara ninguna función miembro, se trata de una interfaz vacía.

El estado de un objeto viene dado, básicamente, por el estado de sus campos. Por lo tanto, serializar un objeto consiste en guardar el estado de sus campos. Si el objeto a serializar tiene campos que a su vez son objetos, se los debe serializar primero. Éste es un proceso recursivo que implica la serialización de todo un árbol de objetos. Además, también se almacena información relativa a dicho árbol, para poder llevar a cabo la reconstrucción del objeto serializado (IBM, 2012).

En la Figura N° 37 se puede observar un segmento de código donde se implementa la interfaz `Serializable` a la clase `claseEjemplo`.

```
public class claseEjemplo implements Serializable{
    public int d;
    public Integer e;
    String f;
}
```

Figura N° 37: Implementación de interfaz `Serializable`

Fuente: Elaboración propia

2.3.1 Almacenamiento de objetos en fichero

Es posible utilizar los mecanismos de serialización disponibles para serializar un objeto guardándolo en un fichero y para realizar el proceso inverso, recuperándolo desde el fichero. Esto se puede realizar gracias a las clases `ObjectInputStream` (mediante el método `readObject()`) y `ObjectOutputStream` (mediante el método `writeObject()`) que están especializados en la lectura y escritura de objetos.

En la Figura N° 38 se puede observar a un segmento de código donde se escriben dos objetos serializados Agenda en el fichero agenda.ser

```
Agenda a1 = new Agenda("Ana", "Martínez", "Fernández");
Agenda a2 = new Agenda("Ernesto", "García", "Pérez");
try{
    FileOutputStream fs = new FileOutputStream("agenda.ser");
    ObjectOutputStream os = new ObjectOutputStream(fs);
    os.writeObject(a1);
    os.writeObject(a2);
    os.close();
}catch(FileNotFoundException e){
    e.printStackTrace();
}catch(IOException e){
    e.printStackTrace();
}
```

Figura N° 38: Ejemplo de código para escritura de objetos serializados en fichero

Fuente: Elaboración propia

En la Figura N° 39 se puede observar a un segmento de código donde se leen objetos serializados Agenda desde fichero agenda.ser.

```
try{
    FileInputStream fis = new FileInputStream("agenda.ser");
    ObjectInputStream ois = new ObjectInputStream(fis);
    a1 = (Agenda) ois.readObject();
    a2 = (Agenda) ois.readObject();
    ois.close();
}catch(FileNotFoundException e){
    e.printStackTrace();
}catch(IOException e){
    e.printStackTrace();
}catch(ClassNotFoundException e){
    e.printStackTrace();
}
```

Figura N° 39: Ejemplo de código para lectura de fichero que contiene objetos serializados

Fuente: Elaboración propia

2.4 Sonido MIDI en Java

El MIDI (*Musical Instrument Digital Interface*) 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 (Oracle, 2014).

Java entre sus bibliotecas de clases presenta un API (*Aplication Interface*) 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 (Oracle, 2014).

2.4.1 Clase `MidiSystem`

La clase `MidiSystem` proporciona acceso a los recursos del sistema MIDI instalados, incluyendo dispositivos como sintetizadores, secuenciadores y entrada MIDI y puertos de salida. Una aplicación MIDI simple, normalmente comienza invocando uno o más métodos `MidiSystem` para conocer qué dispositivos están instalados y obtener un acceso a los necesarios en esa aplicación. El método `GetSynthesizer`, por ejemplo, devuelve el sintetizador de la computadora por defecto.

2.4.2 Clase `Synthesizer`

La clase `Synthesizer` es la representación de un sintetizador. La generación de sonido ocurre cuando uno de los objetos `Synthesizer` mediante alguno de sus `MIDI Channel` (canal MIDI), recibe un mensaje `NoteOn`. Los `Synthesizer` aceptan receptores, a través de los cuales los eventos MIDI pueden ser asignados. En esos casos, el `Synthesizer` responde enviando un mensaje correspondiente al `MIDI Channel` apropiado o mediante el procesamiento del evento, si es que éste no es uno de los mensajes de canal MIDI (Oracle, 2014).

La clase `Synthesizer` tiene métodos para obtener los canales MIDI para la generación de sonido y el banco de sonidos (donde se encuentran los sonidos de instrumentos MIDI que pueden ser reproducidos). En la Figura N° 40 se puede observar el código para la creación de sonido MIDI correspondiente a la nota Do4 (ver Anexo 1) por 3 segundos.

```

//Componentes MIDI
Synthesizer synth = null;
MidiChannel channels[];
try
{
    synth = MidiSystem.getSynthesizer();
    synth.open();
} catch(Exception e) { e.printStackTrace();}

channels = synth.getChannels();
Soundbank bank = synth.getDefaultSoundbank();
synth.loadAllInstruments(bank);
channels[1].noteOn(60, 50);
sleep(3000);
channels[1].noteOff(60);

```

Figura N° 40: Extracto de código Java para generar nota Do4 por 3 segundos

Fuente: Elaboración propia

Las notas en MIDI tienen un valor numérico predefinido. La nota Do4 que también es llamada Do Central tiene un valor de 60 (Oracle, 2014). Cada semitono equivale al valor numérico de 1, por lo tanto, el Re4 tendrá el valor numérico de 62 ya que esta nota es superior al Do4 por dos semitonos (ver Anexo 1).

2.5 Generación de Gráficos Estadísticos en Java con JFreeChart

La biblioteca JFreeChart permite generar gráficos estadísticos de alta calidad. Entre sus características incluye una API documentada, generación de varios tipos de gráficos, generación de imágenes como archivos de salida incluyendo JPG y PNG y otros tipos de archivo como PDF, EPS y SVG. La biblioteca JFreeChart es *open source*, software libre y es distribuido bajo los términos GNU Lesser General Public License (LGPL) que permite su uso en aplicaciones propietarias (JFree, 2014).

2.5.1 Clase ChartPanel

La clase ChartPanel Es un componente GUI de Swing para mostrar un objeto JFreeChart. Se encuentra en la biblioteca JFreeChart y es una herencia del JPanel de Java. Contiene un Chart (tablero) donde están los datos a ser dibujados.

Si un `ChartPanel` recibe la notificación de que el `Chart` que contiene sufre cualquier cambio en alguno de sus componentes, el gráfico se vuelve a dibujar automáticamente (JFree).

2.5.2 Clase `JFreeChart`

La clase `JFreeChart` Es una clase que ha sido implementada utilizando las API de Java 2D. La versión actual soporta gráficos de barras, gráficos de líneas, gráficos circulares y gráficos xy (incluidos los datos de series de tiempo).

La clase `JFreeChart` coordina varias acciones para dibujar un gráfico en un dispositivo de gráficos 2D de Java: una lista de títulos de la tabla o `Chart` (que a menudo incluye la leyenda de la tabla), un gráfico diagramado o `Plot` y un conjunto de datos (un `Plot` a su vez gestiona un eje de dominio y un eje de rango). Además, se puede usar la clase `ChartFactory` que contiene métodos estáticos para la creación de gráficos prediseñados pasando, entre otros parámetros, a objetos cuyas clases implementen la interfaz `XYDataSet`.

Para que `JFreeChart` sea visualizado, se debe usar un `ChartPanel` para mostrarlo en una GUI (JFree).

2.5.3 Interfaz `XYDataSet`

`XYDataSet` es una interfaz a través de la cual los datos de la forma (x, y) pueden ser almacenados, extraídos y modificados. Tiene métodos específicos para cada acción, siendo los más relevantes los `get` y los `set`.

La clase `TimeSeriesCollection` implementa la interfaz `XYDataSet` y es muy útil para almacenar tipos de datos relacionados con el tiempo, ya sean milisegundos, segundos, minutos, horas y años. (JFree).

En la Figura Nº 41 se puede observar una imagen generada con la biblioteca `JFreeChart` utilizando un `JPanel` y como almacenamiento previo, un `TimeSeriesCollection`.

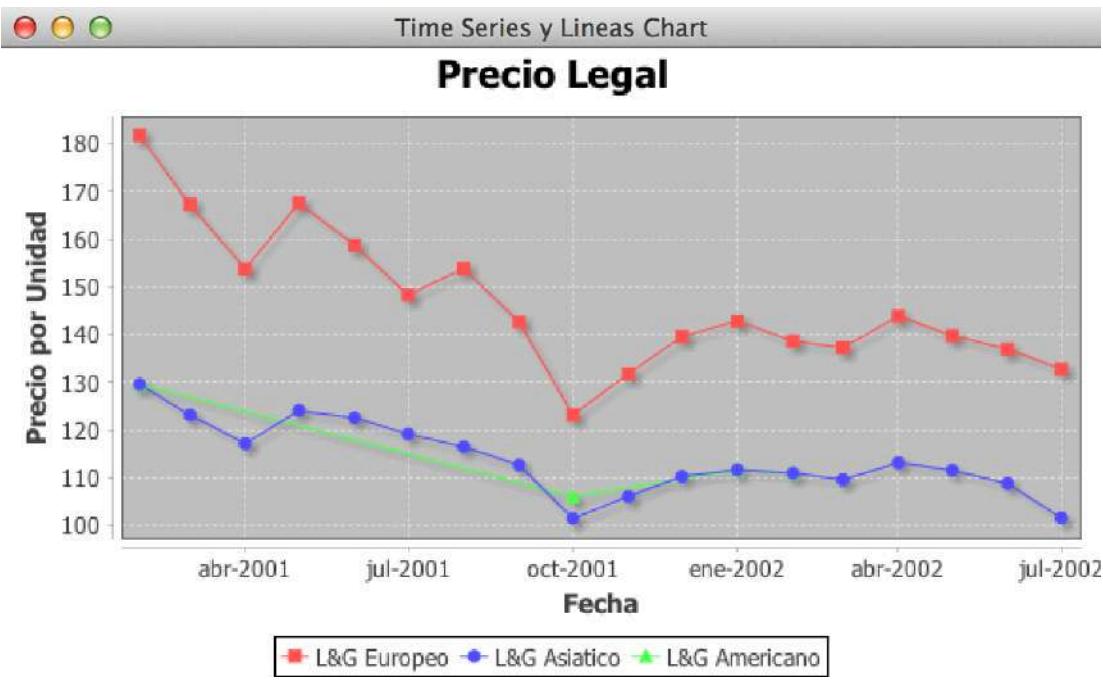


Figura N° 41: Panel estadístico generado con JFreeChart

Fuente: Elaboración propia

2.6 Exportación de Datos con JExcelApi

JExcelApi es un API *open source* creado en Java que tiene herramientas que permiten leer, escribir y modificar hojas de cálculo Excel de forma dinámica. La biblioteca jxl tiene clases como Workbook, Sheet, Cell que representan a una Hoja de Cálculo, Tabla, y Celdas respectivamente.

Para la creación de un nuevo archivo en Excel se debe instanciar una clase WritableWorkbook y pasar como parámetro File, la dirección y el nombre donde se creará el archivo. Posteriormente se debe crear un WritableSheet mediante el método createSheet del WritableWorkbook creado. Después, ya se podrá añadir celdas con su contenido mediante el método addCell del WritableSheet. Las celdas serán creadas automáticamente cuando se pasan los datos a ser almacenados en el parámetro de addCell. Se pueden pasar como parámetros los datos Label, DateTime, Boolean y Number . Finalmente se debe escribir los datos con el método write y posteriormente cerrar el WritableWorkbook con el método close (Quickly Java).

En la Figura N° 42 se puede observar una sección de código para la creación de un archivo de Excel con 4 celdas escritas con diferente tipos de dato.

```
File exlFile = new File("/usuarios/miguel/escritorio");
WritableWorkbook writableWorkbook = Workbook.createWorkbook(exlFile);
WritableSheet writableSheet = writableWorkbook.createSheet("Sheet1", 0);

Label label = new Label(0, 0, "Label (String)");
DateTime date = new DateTime(1, 0, new Date());
Boolean bool = new Boolean(2, 0, true);
Number num = new Number(3, 0, 9.99);

writableSheet.addCell(label);
writableSheet.addCell(date);
writableSheet.addCell(bool);
writableSheet.addCell(num);

writableWorkbook.write();
writableWorkbook.close();
```

Figura N° 42: Sección de código para generar archivo Excel con JExcelApi

Fuente: Elaboración propia

En la Figura N° 43 se puede observar la sección del archivo Excel generado por el código de la Figura N° 42.

	A	B	C	D
1	Label (String)	8/15/2012	TRUE	9.99
2				

Figura N° 43: Sección de archivo Excel generado con JExcelApi

Fuente: Elaboración propia

III ESTUDIO DE ALTERNATIVAS

En esta sección se procederá a determinar las especificaciones del prototipo, se identificarán las alternativas comerciales, se definirán los criterios de evaluación y finalmente se hará una selección de las alternativas óptimas.

3.1 Especificaciones técnicas iniciales

Acorde a los problemas y las necesidades encontrados, las especificaciones que debe tener el prototipo de sistema se describirán en esta sección.

El sistema debe poder capturar las posiciones de las manos (dedos y muñeca) y enviar la información a una computadora con una velocidad media de 30 capturas por segundo mediante algún hardware. A la vez, el sistema debe facilitar la visualización del movimiento de las manos en una pantalla mediante imágenes que representen el espacio 3D y tener una interfaz gráfica para que el usuario pueda interactuar con el sistema mediante el mouse y el teclado. Además, el sistema debe poder simular un teclado musical virtual simple con una extensión de al menos una octava (ver Anexo 1), para que se pueda tener un entorno básico de práctica.

Por otra parte, el sistema debe poder grabar las posiciones capturadas para posteriores visualizaciones, realizar gráficos estadísticos basados en la información obtenida y exportar los datos a formato de archivos Excel.

Todo lo anterior debe poder funcionar en diferentes computadoras sin tener la necesidad de reformular todo el sistema, por lo tanto, debe poder ser en esencia y en la medida de lo posible, independiente del sistema operativo.

3.1.1 Especificaciones técnicas: Hardware de captura de datos de la mano

Para realizar la captura de los datos de la mano se debe contar con un hardware que:

- Permite captar la posición de los dedos y la muñeca
- Pueda enviar los datos en tiempo real a la computadora
- Tenga *Software Development Kit (SDK)* y documentación
- Sea accesible

Los dispositivos hardware que cumplen con los requisitos y que existen en el mercado actualmente son: Kinect y Leap Motion.

Kinect es un controlador de juego desarrollado por Microsoft para la video consola Xbox 360 lanzado en noviembre del año 2010. Actualmente se encuentra en la segunda versión (v2) y tiene un SDK que se puede descargar de la página de Microsoft. También existe una versión de código abierto en Linux llamado *libfreenect* creado con ingeniería inversa que permite el uso de la cámara RGB y las funciones de profundidad (Openkinect, 2012).

El dispositivo Kinect es muy conocido gracias a que es accesorio de Xbox 360. Está orientado principalmente a ser un sensor de movimientos corporales en la rama de entretenimiento, aunque también varios proyectos de investigación han ido utilizando el Kinect en ramas de educación y medicina. En octubre de 2014, Microsoft ha mostrado en su página de investigaciones, *Microsoft Research*, sus recientes avances de captura de la mano con el Kinect en el proyecto de investigación *Handpose: Fully Articulated Hand Tracking* (Microsoft, 2015), la publicación formal fue realizada en abril de 2015. De manera no formal, existen desarrolladores que utilizando *libfreenect* han podido rastrear los dedos y capturar los datos de sus posiciones.

Dentro de las características del Kinect es que el dispositivo puede rastrear desde los 1.2 metros hasta 3.5 metros de profundidad, alcanza una velocidad de captura de 30 fps de manera constante y además del sistema de captura de imagen y profundidad, tiene un sistema de captura de sonido (Microsoft).

Por otra parte, Leap Motion es un dispositivo de reciente lanzamiento (año 2012) no muy conocido, sin embargo tiene una comunidad de desarrolladores que va creciendo. El dispositivo está orientado a la captura de datos de las posiciones de los dedos y muñecas para proveer un nuevo nivel de interacción con la computadora gracias a los movimientos de las manos. Su SDK está disponible para seis lenguajes de programación en su página oficial, al igual que la documentación donde se pueden observar varios ejemplos cortos, explicaciones del funcionamiento del código y las características del dispositivo conectado a la computadora (Leap Motion).

Dentro de las características del Leap Motion se puede resaltar la profundidad de alcance de rastreo que es de aproximadamente 0.5 metros a 0.8 metros y su velocidad variable de captura de datos.

La lógica de funcionamiento no ha sido revelada por la compañía, pero se puede observar que la velocidad máxima de captura es de alrededor de 180 fps y una velocidad mínima de 20 fps dependiendo de la cantidad de movimiento, aumentando en caso de la existencia de mucho movimiento y reduciendo en caso de poco o nulo movimiento de las manos (Leap Motion Inc., 2014).

En cuanto a la accesibilidad del Kinect y del Leap Motion, se ha podido observar que sobre todo Kinect tiene una gran demanda en el mundo pero no tiene alcance en Bolivia. Cabe aclarar que el dispositivo periférico de Xbox, aunque es un Kinect, no es compatible con una computadora, es decir, se requiere la compra del Kinect para PC. Por lo tanto, cualquiera de los dispositivos deben ser traídos de otros países para poder desarrollar el sistema propuesto. En cuanto al costo de ambos, es accesible debido al carácter de entretenimiento de ambos dispositivos, aproximadamente es de 200 USD el Kinect (Microsoft, 2015) y 100 USD el Leap Motion (Leap Motion) según ambas páginas web oficiales.

La Tabla N° 1 muestra diferentes indicadores significativos y ponderaciones para las alternativas a analizarse. La Tabla N° 1 ayuda a establecer la opción más adecuada para el desarrollo del sistema, tomando en cuenta una ponderación de 1 a 5, donde 1 es el menor y 5 el máximo posible.

Como se puede observar en la Tabla N° 1, Leap Motion es el dispositivo hardware que cumple mejor con los indicadores establecidos

Tabla N° 1: Indicadores y ponderaciones de las alternativas para hardware de captura de datos de la mano

Indicador/ Alternativa	Detalles Kinect	Ponderación Kinect	Detalles Leap Motion	Ponderación Leap Motion
Captura datos de posición de los dedos y la muñeca	Se puede capturar la posición de las manos, pero no datos de dedos ni muñeca. Existen herramientas externas al proyecto original Kinect que sí brindan esa posibilidad	3	Está orientado a la captura de datos de los dedos y de las muñecas.	5
Envío de datos en tiempo real a la computadora	Realiza capturas equivalentes a 30 fps de manera constante	2	Realiza capturas desde 20 fps hasta 180 fps dependiendo del movimiento de la mano	5
Presenta SDK y documentación	Tiene SDK y documentación abundante, además de una comunidad amplia de desarrolladores y empresas importantes que impulsan las investigaciones	5	Tiene SDK y documentación brindada sobre todo por la misma organización. Desde el año 2014 se ha ido ampliando la comunidad de desarrolladores	3
Accesibilidad	Está ampliamente difundido y tiene un precio promedio de 120 USD. Sin embargo, en Bolivia no existe a la venta como producto individual, sino, como parte de Xbox de Microsoft	3	Los centros de venta principalmente se encuentran en EEUU. El precio promedio es de 100 USD.	3
TOTAL		13		16

Fuente: Elaboración propia

3.1.2 Especificaciones técnicas: Lenguaje de programación para análisis de datos, generación de gráficos, audio e interfaz de usuario

Ya que el dispositivo de hardware seleccionado es Leap Motion, las opciones de lenguajes de programación y plataformas que pueden ser utilizadas de manera directa debido al desarrollo de bibliotecas por la propia empresa (Leap Motion), son:

- C++
- Unity
- Objective-C
- Python
- Java
- JavaScript

Para elegir un lenguaje de programación, se dio prioridad a las herramientas nativas y a bibliotecas que sean de organizaciones que presenten una documentación y soporte de su creación. Se sabe que en los lenguajes de programación se brindan diversas posibilidades o facilidades, pero existen lenguajes que, en determinados aspectos son más amplios que otros (por el desarrollo de sus bibliotecas) o se consiguen los resultados de manera más rápida (lo que también varía dependiendo de la experiencia de la persona), por lo tanto, aunque existen varios lenguajes donde se puede alcanzar un resultado equivalente, también se tomará en cuenta la experiencia que se tiene en el manejo del lenguaje.

Como está previsto que el sistema funcione como aplicación de escritorio, debido a las características ya mencionadas en el inicio del capítulo, JavaScript quedaría descartado debido a que no se va a utilizar una de sus principales fortalezas, manejo de páginas web. De igual manera, Objective-C no podría ser una opción válida debido a que está orientado casi exclusivamente a sistema operativo Mac OS e iOS (Apple, 2014), contrariando a uno de los requerimientos iniciales.

El lenguaje de programación C++ fue diseñado a mediados de los años 80, como una extensión del lenguaje C con mecanismos que permiten la manipulación de objetos además de la programación estructurada, por lo que es considerado un lenguaje multiparadigma (cplusplus). El lenguaje C++ aunque es muy amplio y permite un mayor control en los programas generados, no permite crear interfaces gráficas de una forma sencilla

(comparando con otros lenguajes más recientes). Actualmente existe la opción de usar Visual C++ que está disponible en el IDE Visual Studio 2013 de Microsoft (Microsoft, 2014), sin embargo, limitaría su uso al sistema operativo Windows. Por otra parte, tampoco tiene una biblioteca nativa para la generación de sonido MIDI, pero existen bibliotecas externas como el `midifile` con documentación y ejemplos para su uso (`midifile`). De igual manera, existen bibliotecas externas para la generación de cuadros estadísticos y exportación de datos a archivos Excel (como el LibXL) (LibXL).

Unity, según la propia explicación de su página web, es una plataforma de desarrollo flexible y poderosa para crear juegos y experiencias interactivos 3D y 2D multiplataforma (Unity). Es un ecosistema completo para todo aquel que busque desarrollar un negocio a partir de la creación de contenido de alta gama y conectarse con sus jugadores . Sin embargo, aunque visualmente es una plataforma muy amplia, no existe una biblioteca nativa para generación de sonido MIDI. Existe una librería externa creada por *Foriero Studio*, que actualmente se encuentra a la venta (Foriero, 2012).

Python, es un lenguaje de programación interpretado que da mucha importancia a que la sintaxis favorezca un código legible. Es multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional (Python). Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma, por lo que brindaría la flexibilidad requerida por el prototipo.

Python tiene una amplia comunidad de desarrolladores y documentación, además de bibliotecas que brindan al lenguaje versatilidad en varios aspectos. Sin embargo, no tiene una biblioteca para el manejo MIDI, por lo que existen bibliotecas externas, no oficiales, que aporta la comunidad para el uso, advirtiendo que el funcionamiento no es perfecto. Un ejemplo de lo mencionado anteriormente es el `python-midi` (`ensemblерobot`).

Por otra parte, existen organizaciones que se han dedicado al desarrollo de bibliotecas gráficas para Python, como el VPython, que brindan herramientas que facilitan el dibujo de objetos en 3D (VPython).

Java, es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener la menor cantidad posible de dependencias de la implementación, lo cual permite que los desarrolladores de aplicaciones escriban el

programa una vez y lo ejecuten en cualquier dispositivo, lo que según los requerimientos iniciales del prototipo sería lo más recomendado. Como Java corre en la JVM (*Java Virtual Machine*), entonces se puede utilizar el mismo programa en diferentes sistemas operativos que tengan JVM (Oracle), dando la flexibilidad requerida para la creación del prototipo.

Java también tiene de manera nativa la biblioteca `javax.sound.midi` que permite el manejo de sintetizadores y canales MIDI para la producción de sonido (Oracle, 2014). Además existen foros sobre el tema, ejemplos de aplicaciones funcionales y variedad de documentación en línea aparte de la otorgada por Oracle.

Otra de las características que se puede resaltar de Java es su variedad de bibliotecas para poder realizar la construcción de interfaces gráficas simples, biblioteca para realizar dibujos en 2D, proyecto Java 3D que utiliza OpenGL y Direct3D para proveer gráficos en 3D (Java3D), biblioteca para exportar datos a Excel de manera simple como el `JExcelApi` y bibliotecas para generar gráficos estadísticos con `JFreeChart`. Todas las bibliotecas mencionadas tienen documentación, foros de desarrolladores, organizaciones dedicadas a las mejoras y a la corrección de errores, además de ejemplos en aplicaciones funcionales.

Cabe aclarar que si bien existen bibliotecas para Java que brindan opciones para dibujo en 3D (como el Java3D que corre sobre OpenGL), éstas no son totalmente compatibles actualmente con Java, ya que la biblioteca JOGL (*Java OpenGL*) es la que permite acceder a OpenGL mediante programación en Java y, desde el año 2009, Sun (que pasó a pertenecer a Oracle el año 2010) dejó de mantener a JOGL, creando incompatibilidades en las bibliotecas (Oracle).

Debido a los problemas de compatibilidad en diversas bibliotecas de gráficos 3D externas, también se plantea la solución de utilizar una biblioteca propia, creada en la materia de Infografía. En esta materia se aprendieron las bases necesarias para la recreación de imágenes en 3D mediante mallas poligonales, que si bien son muy simples a comparación de los gráficos actuales, cumplirían con el objetivo de recrear un ambiente virtual 3D. Como la mencionada biblioteca fue hecha en Java, se tendría una gran ventaja en la experiencia, conocimiento del lenguaje y en la manejabilidad de la biblioteca respecto a otro lenguaje u otra biblioteca.

Para realizar la ponderación, se debe tomar en cuenta que el lenguaje de programación elegido debe brindar facilidades para:

- Generar gráficos que representen a las manos
- Generar sonido MIDI
- Crear una interfaz simple
- Realizar gráficos estadísticos
- Exportar datos a archivos Excel
- Crear el sistema compatible en distintos sistemas operativos

La Tabla N° 2 muestra diferentes indicadores significativos y ponderaciones para las alternativas a analizarse. La Tabla N° 2 ayuda a establecer la opción más adecuada para el desarrollo del sistema, tomando en cuenta una ponderación de 1 a 5, donde 1 es el menor y 5 el máximo posible.

Como se puede observar en la Tabla N° 2, Java es el lenguaje de programación que cumple mejor con los indicadores establecidos.

Tabla N° 2: Indicadores y ponderaciones de las alternativas para lenguaje de programación

Indicador/ Alternativa	Detalles C++	Ponderación C++	Detalles Unity	Ponderación Unity	Detalles Python	Ponderación Python	Detalles Java	Ponderación Java
Generación de gráficos 3D	Difícil de producir imágenes 3D, existen opciones novedosas que restringen su uso a Microsoft	1	Plataforma dedicada a la producción de gráficos 3D	5	Tiene bibliotecas externas como el VPython que brindan herramientas para imágenes 3D	4	Herramientas de mallas poligonales de proyectos propios anteriores. Se tiene conocimiento y control completo.	4
Generación de sonido MIDI	Biblioteca externa midi file provee herramientas de manejo MIDI	3	Biblioteca externa creada por Foriero Studio, tiene un costo elevado	2	Biblioteca externa que todavía no funciona de manera estable	2	Biblioteca propia de Java para manejo de MIDI	5
Herramientas para creación de interfaz de usuario	Existen herramientas, pero su manejo es complicado a comparación de las otras opciones	2	Herramientas propias que facilitan el desarrollo de interfaces gráficas	5	Biblioteca TKinter que brinda varias herramientas para el desarrollo de interfaces gráficas	4	Bibliotecas propias que facilitan el desarrollo de interfaz gráfica	4
Herramientas para creación de gráficos estadísticos	Bibliotecas externas, de uso complejo.	2	Biblioteca externa Graph Maker con costo	3	Varias bibliotecas externas para creación de gráficos estadísticos, ejemplo: CairoPlot	4	Biblioteca JFreeChart que facilita el desarrollo de gráficos estadísticos	4

Indicador/ Alternativa	Detalles C++	Ponderación C++	Detalles Unity	Ponderación Unity	Detalles Python	Ponderación Python	Detalles Java	Ponderación Java
Exportación de datos a Excel	Biblioteca LibXL con costo	2	No se conoce ninguna herramienta de exportación de datos a Excel	1	Bibliotecas creadas por Simplistix para manejo de Excel con Python	4	Biblioteca JExcel que proporciona herramientas de manejo de datos en archivos Excel	4
Compatibilidad con sistemas operativos	Multiplataforma	5	Multiplataforma	5	Multiplataforma	5	Multiplataforma	5
Conocimiento del lenguaje	Básico	2	Ninguno	1	Básico	2	Medio-Avanzado	4
Cantidad y accesibilidad de documentación encontrada en investigación de alternativas	Poca información en los temas necesarios para desarrollo del prototipo	1	Abundante información en algunos de los aspectos investigados	3	Abundante documentación base, foros y ejemplos disponibles en todos los aspectos investigados	5	Abundante documentación base, foros y ejemplos disponibles en todos los aspectos investigados	5
TOTAL		18		25		30		35

Fuente: Elaboración propia

IV DISEÑO DEL PROTOTIPO

En este capítulo se describirá el diseño que se planteó para el desarrollo del prototipo.

Para realizar el software del prototipo, se utilizó principalmente RUP (*Rational Unified Process*), pero no se lo aplicó en su totalidad debido a que los roles (cliente, jefe de proyecto, diseñador, programador, examinador) eran compartidos por una sola persona y esto permitía un mayor flujo de información sin ser necesario un registro completo detallado, especificación de división de tareas, división de equipos y otros aspectos propio del trabajo de equipo. Sin embargo, en esta sección se mostrarán y explicarán los diagramas de casos de uso y diagramas de clases, siendo los más relevantes para el desarrollo del prototipo.

Es necesario recalcar que la toma de requerimientos o especificaciones se hizo principalmente por planteamiento de mi persona con la colaboración de otros docentes de piano de la Academia Nacional de Música Man Césped posterior a una Solicitud a Expertos.

4.1 Determinación de especificaciones

De manera general, las especificaciones se expusieron en el título 3.1. Sin embargo, aunque la parte principal e imprescindible del proyecto es el diseño del sistema de software, se vio que para maximizar el funcionamiento del sistema propuesto, era necesario diseñar una estructura física o armazón de apoyo para el dispositivo de captura. Por esta razón, la determinación de especificaciones se divide en dos: especificaciones del sistema y especificaciones de estructura física anexa.

Las especificaciones del sistema son:

- El sistema debe facilitar la visualización del movimiento de las manos mediante representaciones gráfica simple en 3D
- El sistema debe mostrar los datos de los ángulos de contacto e inclinación de los dedos del usuario
- El sistema debe tener un teclado virtual de, al menos, una octava de extensión
- El sistema debe representar al teclado virtual en el espacio virtual de manera visible

-
- El teclado virtual debe poder tocarse, es decir, producir sonido cuando el sistema detecta que un dedo llega a tocar (estar en la posición) de una tecla
 - El sistema debe tener la opción de mover el teclado virtual en los ejes x , y , z del espacio virtual, para facilitar la calibración de la posición en el momento de ajuste de la estructura física anexa
 - El sistema debe tener la opción de desactivar Teclado Virtual, anulando de esta manera el Teclado Virtual del Escenario Virtual y por consiguiente cualquier tipo de sonido y representación gráfica de éste.
 - El sistema debe permitir al usuario reposicionar el punto de visualización de la representación gráfica 3D mediante rotación, acercamiento y alejamiento para mejorar la visualización de los objetos en el espacio virtual mediante el teclado
 - El sistema debe permitir al usuario activar y desactivar, de manera independiente, los componentes (dedos, muñecas, teclado) en la representación gráfica 3D
 - El sistema debe tener implementada la opción de grabar las posiciones de las manos del usuario en archivos.
 - El sistema debe tener implementada la opción de reproducir los archivos que contienen los datos de las posiciones de las manos, lo que implica que la reproducción se debe poder visualizar con la representación gráfica 3D previamente implementada.
 - El sistema debe brindar al usuario la facilidad de cambiar las velocidades de reproducción de un archivo. El usuario debe poder acelerar la reproducción (multiplicando la velocidad por 2, 4, 8 y 16 mínimamente) y ralentizarla (dividiendo la velocidad entre 2, 4, 8 y 16 mínimamente) en cualquier momento
 - El sistema debe brindar al usuario la posibilidad de pausar o parar la reproducción en cualquier momento
 - El sistema debe poder generar gráficos estadísticos de los datos (posiciones en los ejes x , y , z respecto al tiempo además de la velocidad en y respecto al tiempo) de los dedos y las muñecas a partir de archivos grabados previamente
 - El sistema debe tener incluida la opción de guardar los gráficos estadísticos como imágenes

-
- El sistema debe tener la opción de exportar los datos (posiciones en los ejes x , y , z respecto al tiempo además de la velocidad en y respecto al tiempo) a un archivo Excel para poder ver los datos numéricos obtenidos en las capturas

Las especificaciones de la estructura física anexa son:

- Tener una superficie de apoyo para las manos del usuario, en un área de un lado similar al largo de las teclas del piano y otro lado del rango medio de captura del dispositivo
- Servir de apoyo al dispositivo de captura de datos (en este caso Leap Motion) por debajo de la superficie de apoyo de las manos del usuario
- Tener la opción de regular la altura del apoyo del dispositivo de captura
- Evitar acumular el calor producido por el dispositivo de captura de datos
- Afectar lo menos posible a la captura de datos por parte del dispositivo de captura, por lo tanto, el material de la superficie de apoyo para las manos del usuario debe ser el más transparente posible para el dispositivo de captura
- Sujetarse a mesas que tengan el grosor de hasta 4 centímetros en el borde
- El armazón de la estructura física anexa debe ser de un material resistente que sea accesible conseguir por el precio y la calidad.

4.2 Diagramas de casos de uso del sistema

En esta sección se expondrán los diagramas casos de uso que fueron generados a partir de las especificaciones del sistema indicadas anteriormente. La descripción de los casos de uso se encuentran en el Anexo 3.

Debido al tamaño del sistema, se realizó una subdivisión en 4 módulos:

- Módulo Audio Visual
- Módulo de Modificación Virtual
- Módulo de Manejo de Archivos
- Módulo de Captura y Procesamiento de Datos

4.2.1 Módulo Audiovisual

El Módulo Audiovisual se encarga de presentar mediante imágenes y sonidos los datos que se generan en el Módulo de Captura y Procesamiento de Datos. En la Figura N° 44 se puede observar el diagrama de casos de uso del Módulo Audiovisual, que tiene por actores al Usuario y al Módulo de Captura y Procesamiento de Datos.

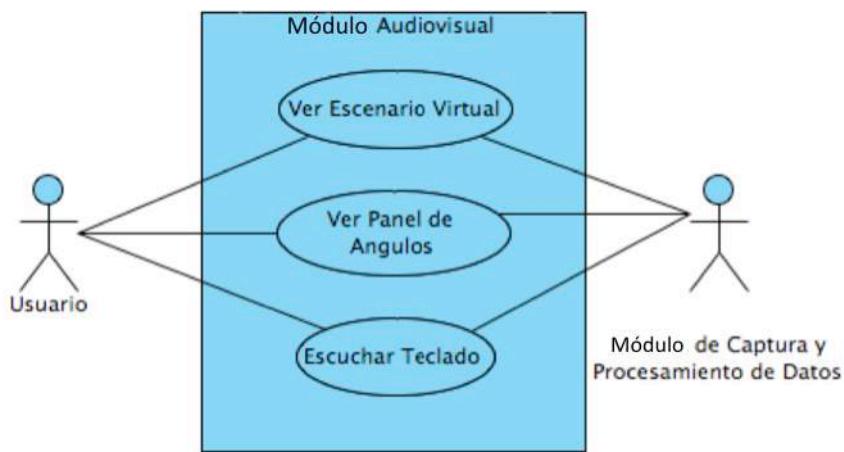


Figura N° 44: Diagrama de casos de uso del Módulo Audiovisual

Fuente: Elaboración propia

El caso de uso Ver Escenario Virtual se refiere al procedimiento necesario que el módulo realiza para mostrar el Escenario Virtual representado en gráficos 3D al Usuario, según los datos que existan en el módulo de Captura y Procesamiento de Datos.

El caso de uso Ver Panel de Ángulos se refiere al procedimiento necesario que el módulo realiza para mostrar el Panel de Ángulos (que contienen los datos de los ángulos de los dedos) al Usuario. Estos datos son brindados por el Módulo de Captura y Procesamiento de Datos.

El caso de uso Escuchar Teclado se refiere a los procedimientos que realiza el módulo para producir el sonido del Teclado Virtual que será captado auditivamente por el Usuario. La producción del sonido depende de los datos obtenidos del Módulo de Captura y Procesamiento de Datos.

Ambos casos de uso son requeridos de manera permanente por el Usuario desde el inicio del sistema hasta el final de su uso, por lo tanto, el Módulo de Captura y Procesamiento de Datos estará interactuando permanentemente con el Módulo Audiovisual.

4.2.2 Módulo de Modificación Virtual

El Módulo de Modificación Virtual se encarga de modificar el entorno virtual según lo que el Usuario defina mediante la interfaz gráfica o mediante la pulsación de una tecla. De esta manera, el entorno virtual puede ser visualizado desde diferente perspectiva y el teclado virtual puede ser movido, activado y desactivado en el Escenario Virtual.

En la Figura N° 45 se puede observar el diagrama de casos de uso del Módulo de Modificación Virtual, que tiene por actores al Usuario y al Módulo de Captura y Procesamiento de Datos.

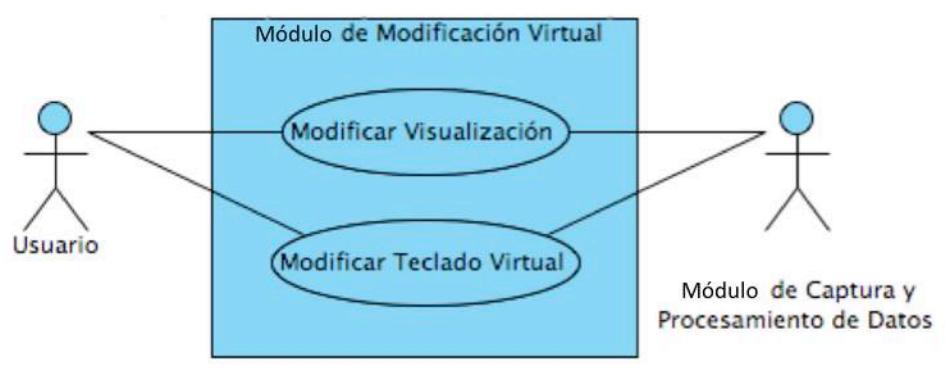


Figura N° 45: Diagrama de casos de uso del Módulo de Modificación Virtual

Fuente: Elaboración propia

El caso de uso Modificar Visualización se refiere al conjunto de procesos que el módulo realiza cuando el Usuario cambia el modo de visualización (rotando, acercando, alejando) del Escenario Virtual representado en gráficos 3D por el Módulo Audiovisual mediante el teclado de la computadora o bien, cuando el Usuario active o desactive los componentes visuales (dedos, muñecas, teclado) mediante la interfaz gráfica. Las modificaciones que se realizan en los procesos afectarán al Módulo de Captura y Procesamiento de Datos, que a su vez, afectan al Módulo Audiovisual.

El caso de uso Modificar Teclado Virtual se refiere al conjunto de procedimientos que el módulo realiza cuando el Usuario mueve, activa o desactiva el teclado virtual mediante la

interfaz gráfica. Estas acciones envían las modificaciones al Módulo de Captura y Procesamiento de Datos.

4.2.3 Módulo de Manejo de Archivos

El Módulo de Manejo de Archivos se encarga de los procedimientos relacionados con la producción y lectura de archivos. La fuente de datos del sistema son los archivos previamente grabados por el Módulo de Captura y Procesamiento de Datos.

En la Figura N° 46 se puede observar el diagrama de casos de uso del Módulo de Manejo de Archivos que tiene por actores al Usuario y al Módulo de Captura y Procesamiento de Datos.

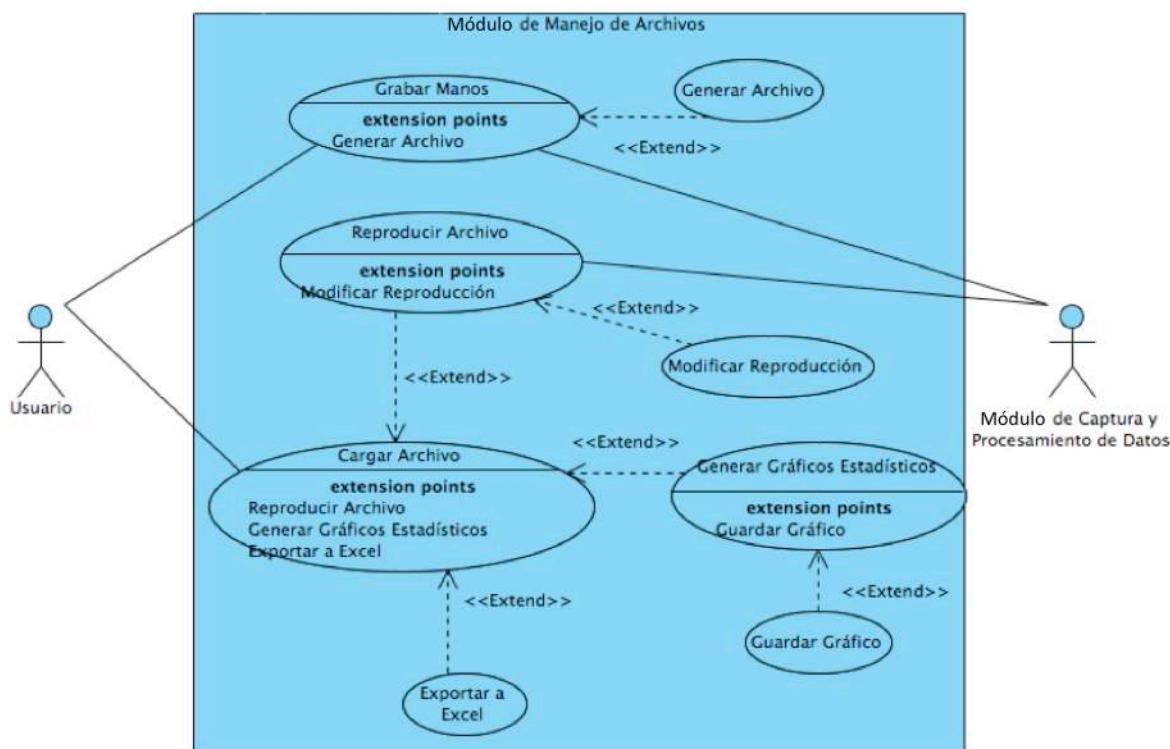


Figura N° 46: Diagrama de casos de uso del Módulo de Manejo de Archivos

Fuente: Elaboración propia

El caso de uso Grabar Manos es el conjunto de procedimientos que el módulo realiza cuando el Usuario activa la opción Grabar en la interfaz gráfica, haciendo que este módulo capture los datos del Módulo de Captura y Procesamiento de Datos almacenándolos temporalmente. Posteriormente si el Usuario lo requiere, se utiliza el caso de uso Generar

Archivo, donde se guardan los datos (antes temporales) en un archivo definido por el Usuario.

El caso de uso Generar Archivo se refiere al conjunto de procedimientos que el módulo realiza cuando el Usuario acepta guardar el archivo con los datos previamente grabados. Dentro de los procedimientos están: asignar nombre y ubicación del archivo a guardar y confirmación de la acción.

El caso de uso Cargar Archivo es el conjunto de procedimientos que realiza el módulo para proceder a la lectura de un archivo cuando el Usuario activa la opción de Cargar Archivo en la interfaz gráfica. Este conjunto de procedimientos es necesario para poder reproducir un archivo, generar gráficos estadísticos y exportar los datos a Excel, por lo tanto, ninguna de éstas opciones estará habilitada en la interfaz gráfica en caso de que no se realice el cargado de archivo previamente. Cada uno de estos procedimientos está descrito en sus respectivos casos de uso.

El caso de uso Cargar Archivo también contiene los procedimientos necesarios para Quitar Archivo posterior a su uso (en caso de que se haya activado la reproducción, generación de gráficos estadísticos o exportación a Excel) para poder cargar otro archivo.

El caso de uso Reproducir Archivo se refiere al conjunto de procedimientos que el módulo realiza cuando el Usuario activa la opción de Reproducir mediante la interfaz gráfica. Como se mencionó anteriormente, de manera previa debe cargarse un archivo para poder reproducirlo y mientras está siendo reproducido, el Usuario puede modificar la reproducción. Esta opción es descrita en el caso de uso Modificar Reproducción. Mientras se reproduzca el archivo, se mandarán los datos al Módulo de Captura y Procesamiento de Datos, lo que hará que el Módulo Audiovisual proceda a mostrar la reproducción al Usuario.

El caso de uso Modificar Reproducción se refiere al conjunto de procedimientos que realiza el sistema cuando el Usuario activa una modificación a la reproducción desde la interfaz gráfica. Las modificaciones pueden ser: Velocidadx2, Velocidad/2, Pausar y Parar. Cada una de estas modificaciones tiene su propio procedimiento y sólo podrá ser activado mientras la reproducción está en curso. La modificación Velocidadx2 duplicará la velocidad de reproducción, la Velocidad/2 la reducirá a la mitad, la modificación Pausar hará que se pause la reproducción y Parar hará que la reproducción finalice.

El caso de uso de uso Generar Gráficos Estadísticos se refiere al conjunto de procedimientos que realiza el módulo cuando el Usuario activa alguna de las 4 opciones para generar gráficos estadísticos: X vs Tiempo, Y vs Tiempo, Z vs Tiempo y Velocidad Y vs Tiempo. Estos gráficos estadísticos estarán generados a partir de los datos de las posiciones y la velocidad de los elementos de la mano (dedos y muñeca) respecto al tiempo obtenidos del archivo cargado. Una vez generado el gráfico, el Usuario tendrá la opción de guardarla como imagen. Los procedimientos requeridos para la acción mencionada corresponden al caso de uso Guardar Gráfico.

El caso de uso Guardar Gráfico, incluye los procedimientos necesarios para que el módulo guarde los gráficos estadísticos generados en formato de imagen cuando el Usuario, mediante la interfaz gráfica, seleccione la opción de Guardar Imagen. Entre los procedimientos se encuentran: asignar nombre y ubicación del archivo a guardar y confirmación de la realización de la acción.

El caso de uso Exportar a Excel incluye los procedimientos necesarios para que el módulo genere un archivo Excel cuando el Usuario seleccione la opción de Exportar Archivo. Entre los procedimientos se encuentran: seleccionar resolución y filtro de datos a ser exportados, confirmar la selección, asignar nombre y ubicación del archivo a guardar y, finalmente, confirmación de la acción.

4.2.4 Módulo de Captura y Procesamiento de Datos

El Módulo de Captura y Procesamiento de Datos se encarga de los procedimientos relacionados con la captura y el posterior procesamiento de datos, siendo la parte principal del sistema.

Este módulo interactúa de varias maneras con los demás módulos, además del Usuario y el Leap Motion:

- Sirve como fuente de datos para el Módulo Audiovisual
- El Módulo de Modificación Virtual, activa operaciones que llegan a modificar los datos del sistema
- El Módulo de Manejo de Archivos, para la generación o reproducción de archivos
- Recibe los datos del Leap Motion

- Interactúa directamente con el Usuario para el inicio y el final de sus procedimientos.

En la Figura N° 47 se puede observar el diagrama de casos de uso del Módulo de Captura y Procesamiento de Datos que tiene por actores a todos los demás módulos además del Usuario y el dispositivo de captura Leap Motion.

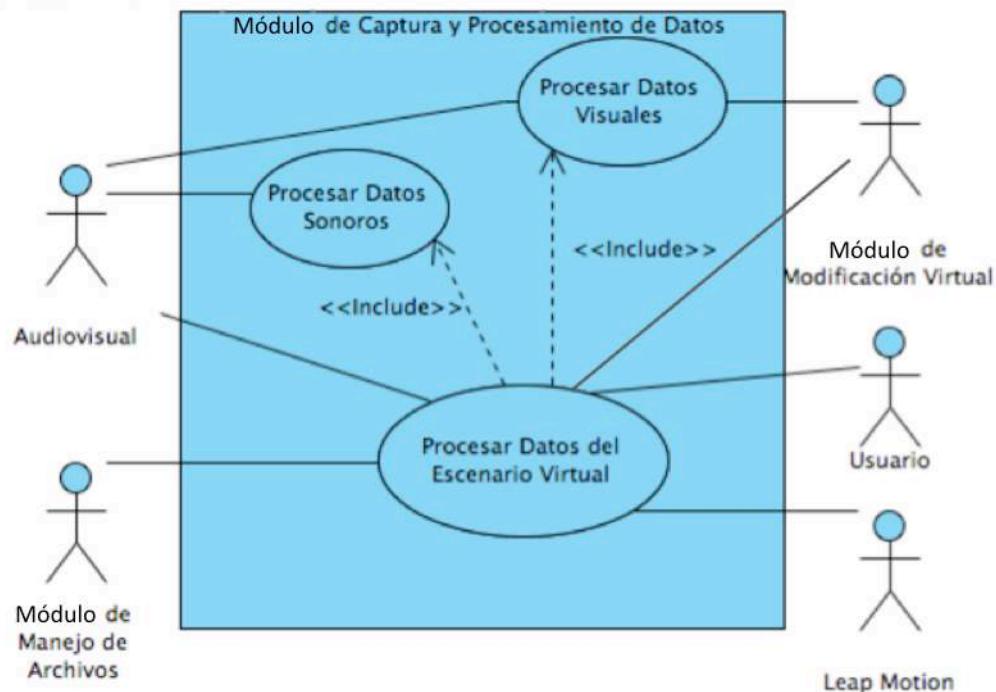


Figura N° 47: Diagrama de casos de uso del Módulo de Captura y Procesamiento de Datos

Fuente: Elaboración propia

El caso de uso Procesar Datos de Escenario Virtual es el conjunto de procesos que el sistema realiza para gestionar y controlar los datos en el Escenario Virtual. El Usuario inicia el funcionamiento del sistema general y el actor Leap Motion por defecto comenzará enviar los datos, que serán procesados en el caso de uso Procesar Datos de Escenario Virtual para su posterior registro en el Escenario Virtual. Sin embargo, como alternativa, el caso de uso Procesar Datos de Escenario Virtual puede no requerir el caso de uso Captura de Datos en caso de que se utilice el Módulo de Manejo de Archivos (cuando exista la reproducción de archivos).

Para procesar los datos sonoros y visuales, el caso de uso Procesar Datos del Escenario Virtual incluye los casos de uso Procesar Datos Sonoros y Procesar Datos Visuales. Entre

los procedimientos también se encuentra el procedimiento para mandar la información sobre los ángulos de los dedos al Sistema Audiovisual.

Entre las secuencias opcionales del caso de uso están:

- La reproducción de archivos, donde el Módulo de Manejo de Archivos manda los datos de un archivo para representarlo en el Escenario Virtual.
- La grabación, donde el Módulo de Manejo de Archivos captura los datos para guardarlos en un archivo.
- La modificación, donde el Módulo de Modificación Virtual procede a cambiar la posición del teclado dentro del Escenario Virtual.

En la Tabla N° 3 se puede observar la descripción del caso de uso Procesar Datos del Escenario Virtual

Tabla N° 3: Caso de uso Procesar Datos del Escenario Virtual

Caso de Uso:	Procesar Datos del Escenario Virtual	
Actores:	Usuario, Leap Motion, Módulo de Modificación Virtual, Módulo Audiovisual, Módulo de Manejo de Archivos	
Descripción	Es el conjunto de procesos que el sistema realiza para gestionar y controlar los datos de entrada y salida en el Escenario Virtual	
Precondición		
Flujo Principal	Paso	Acciones
	1	Módulo de Captura y Procesamiento de Datos crea Espacio Virtual y Reproducción Gráfica por defecto
	2	Leap Motion manda datos capturados
	3	Módulo de Captura y Procesamiento de Datos guarda los datos de Leap Motion en Escenario Virtual
	4	Módulo de Captura y Procesamiento de Datos utiliza caso de uso Procesar Datos Visuales
	5	Módulo de Captura y Procesamiento de Datos utiliza caso de uso Procesar Datos Sonoros
	6	Módulo de Captura y Procesamiento de Datos manda los datos de ángulos de dedos a Módulo Audiovisual
	7	Se repite desde Paso 3 hasta que Usuario indique la finalización de sistema

	Paso	Acciones
Flujo Opcional	2A.1	Módulo de Manejo de Archivos manda datos (reproducción de archivos)
	2A.2	Módulo de Captura y Procesamiento de Datos guarda datos en Escenario Virtual
	2A.3	Retorna a Paso 4 de Flujo Principal
	3A.1	Módulo de Modificación Virtual manda datos para modificar Escenario Virtual (modificar Teclado Virtual)
	3A.2	Módulo de Captura y Procesamiento modifica Escenario Virtual (específicamente el Teclado Virtual)
	3A.3	Retorna a Paso 4 de Flujo Principal
	3B.1	Módulo de Manejo de Archivos pide datos (para grabación de archivos)
	3B.2	Módulo de Captura y Procesamiento de Datos manda datos a Módulo de Manejo de Archivos
	3B.3	Retorna a Paso 4 de Flujo Principal
	Poscondiciones	No existen

Fuente: Elaboración propia

El caso de uso Procesar Datos Sonoros es el conjunto de procedimientos que se encargan de verificar si existen datos que produzcan sonido para enviar la información al Módulo Audiovisual.

El caso de uso Procesar Datos Visuales es el conjunto de procedimientos que se encargan de procesar los datos del Escenario Virtual para poder crear la representación gráfica de los elementos requeridos por el Usuario en el Módulo Audiovisual. El Módulo de Modificaciones Virtuales también puede realizar operaciones de cambio en la Representación Gráfica (rotación, traslación, acercamiento o alejamiento).

4.3 Diagrama de Paquetes

En esta sección se expondrá el diagrama de paquetes que fue elaborado tomando en cuenta los requerimientos y los casos de uso anteriormente descritos. El diagrama de paquetes se puede observar en la Figura N° 48.

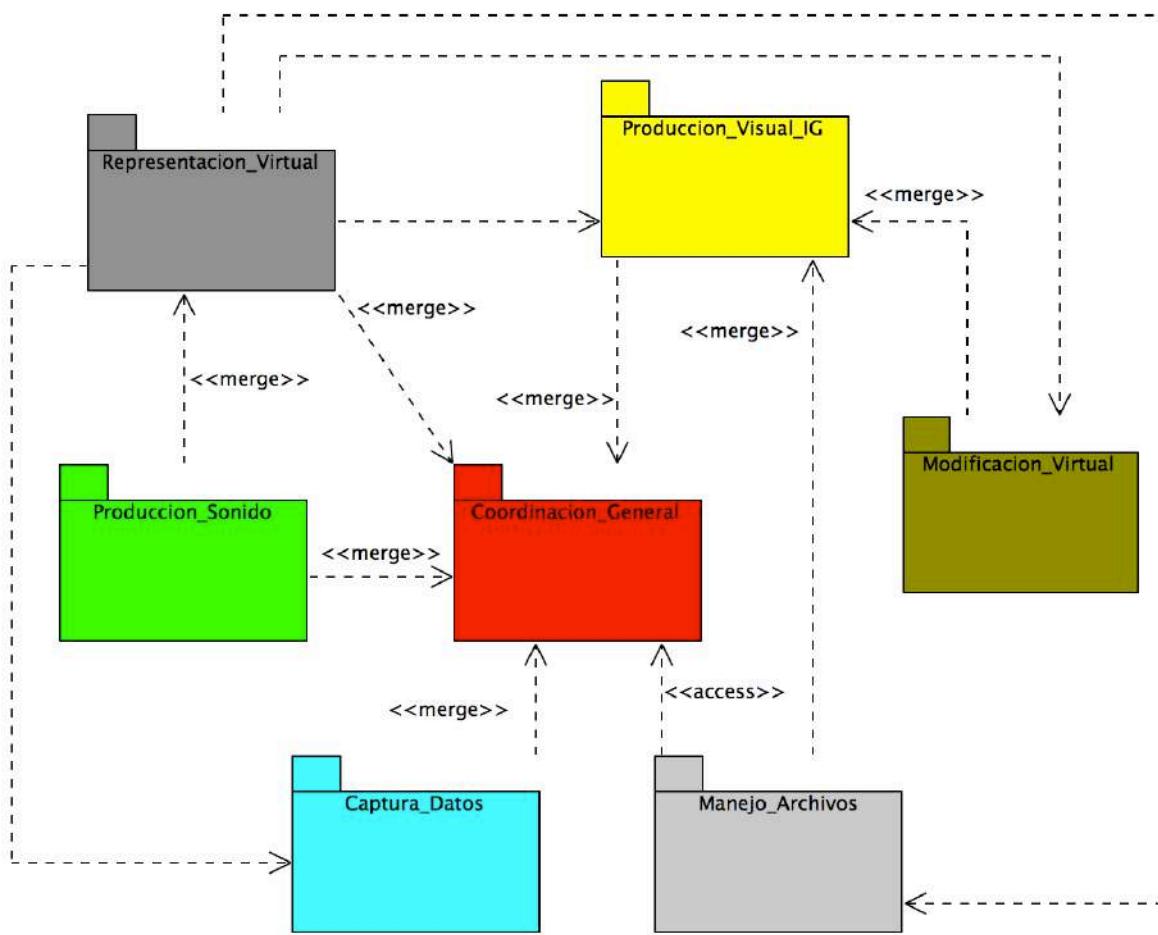


Figura N° 48: Diagrama de Paquetes del Sistema

Fuente: Elaboración propia

El sistema está compuesto por módulos que realizan procedimientos determinados a cumplir los requerimientos establecidos. Cada módulo corresponde a un paquete desarrollado:

- Captura de Datos (paquete **Captura_Datos**), contiene las clases que están relacionadas con la captura de datos mediante el Leap Motion
- Representación Virtual (paquete **Representacion_Virtual**), contiene las clases que representan a los componentes del escenario virtual que van a ser representados gráficamente, es decir, manos y teclado
- Producción de Sonido (paquete **Produccion_Sonido**), contiene las clases que están relacionadas con la producción de sonido MIDI en el sistema

-
- Producción Visual – Interfaz Gráfica (paquete `Produccion_Visual_IG`), contiene las clases que generan la interfaz gráfica y también las intervienen en la generación de gráficos mediante Mallas Poligonales y representan a las clases (abstracción de la mano y el teclado) de Representación Virtual
 - Modificación Virtual (paquete `Modificacion_Virtual`), contiene las clases que están relacionadas con la modificación de gráficos y con modificación de la posición del teclado virtual
 - Manejo de Archivos (paquete `Manejo_Archivos`), contiene las clases que sirven para realizar grabaciones, reproducciones, generar gráficos estadísticos y exportar datos en Excel
 - Coordinación General (paquete `Coordinacion_General`), compuesta por la clase que inicia el funcionamiento del sistema y de la clase que regula y coordina los procesos que el sistema puede realizar

El funcionamiento inicial por defecto del sistema incluye a la Producción Visual – Interfaz gráfica, Producción de Sonido, Captura de Datos, Representación Virtual de Teclado y Manos, además de la Coordinación general. Los módulos Modificación Virtual y Manejo de Archivos son activados posteriormente por el usuario de manera opcional.

4.3.1 Coordinación General

El sistema inicia con las clases de Coordinación General que, como se puede observar en la Figura N° 48, tiene una relación *merge* con los paquetes correspondientes a los módulos de Representación Virtual, Producción de Sonido, Producción Visual y Captura de Datos los cuales funcionan desde el principio del sistema, ya que son precisamente instanciados desde Coordinación General.

El sistema funciona con 5 hilos de ejecución (ejecutándose de manera concurrente un máximo de 4 al mismo tiempo). Esos hilos son regulados (iniciados o detenidos) por Coordinación General. Corresponden a clases (que heredan de la clase `Thread`) de los módulos Producción Visual, Producción de Sonido, Captura de Datos y Manejo de Archivos (para la grabación y la reproducción).

El paquete contiene un registro de qué hilos de ejecución del sistema están funcionando por lo que, constantemente sirve como verificador (*scheduler*) para que todos los procesos funcionen según lo previsto.

4.3.2 Captura de datos

La Captura de Datos está compuesta por las clases que se encargan de recibir los datos del dispositivo Leap Motion. Las clases utilizan varios métodos de la biblioteca de Leap Motion y pasan los datos obtenidos al módulo de Representación Virtual.

Las coordenadas recibidas están en milímetros, la velocidad en milímetros sobre segundo y corresponden a un sistema de coordenadas predefinido por Leap Motion, donde el origen es el centro del dispositivo.

La captura de datos existe desde el inicio del sistema y el único momento donde no es utilizado, es cuando se está reproduciendo un archivo.

Para mayor información sobre los conceptos y el funcionamiento de las clases de Captura de Datos, ver el Anexo 5, capítulo 1.

4.3.3 Representación Virtual

La Representación Virtual de Teclado y Manos está compuesta por las clases que representan el Escenario Virtual, que contiene a las estructuras de las manos y el teclado. La mano está compuesta por los dedos y a su vez, éstos están compuestos por huesos (falanges). Por otra parte, el teclado está compuesto por teclas. 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.

Además de las clases que representan los componentes que conforman la estructura de la mano, también contiene a las clases que representan al teclado, el cual a su vez es formado por teclas.

La Representación Virtual depende del módulo Modificaciones Virtual, ya que puede variar la posición del Teclado. También depende de las modificaciones que realice Captura de Datos (cambiar los datos de un dedo, por ejemplo) y Manejo de Archivos (en grabación o reproducción).

Para mayor información sobre los conceptos y el funcionamiento de las clases de Representación Virtual, ver Anexo 5, capítulo 2.

4.3.4 Producción de Sonido

La Producción de Sonido está compuesta por las clases que se encargan de producir el sonido del Teclado Virtual.

Estas clases representan al proceso que estará funcionando todo el momento (Hilo de sonido), al sintetizador (necesario para producir el sonido MIDI en java), a las notas musicales para determinar el sonido a producir, la parte musical de la tecla relacionada con la representación virtual de la tecla (de Representación Virtual) y el estado de la misma (“presionada” o “suelta”).

Para saber si una tecla es presionada por un dedo se verifica la posición de la punta de todos los dedos en la Representación Virtual y se evalúa si efectivamente está “presionando” una tecla, es decir, si está en el área de toque de la tecla virtual.

Para mayor información sobre los conceptos y el funcionamiento de las clases de Producción de Sonido, ver Anexo 5, capítulo 3.

4.3.5 Producción Visual – Interfaz Gráfica

La Producción Visual – Interfaz Gráfica está compuesta por las clases que se encargan de producir las representaciones gráficas de los dedos, muñecas y teclas que tienen datos almacenados en Representación Virtual (en Escenario Virtual), además de 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*) y cuadros

de dialogo. Corresponde a una clase que hereda de Thread y es encargado de que la producción visual funcione desde el inicio hasta el fin de la ejecución.

En este módulo se determina cómo va a ser representado cada componente del Escenario Virtual (del módulo Representación Visual), es decir, cómo van a ser representados los huesos de los dedos, la muñeca y las teclas mediante Mallas Poligonales.

Es necesario recalcar que en Producción Visual se representa gráficamente a los componentes tomando un sistema de coordenadas diferente al de Representación Virtual, que 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 mejor a las Mallas Poligonales.

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

Para mayor información sobre los conceptos y el funcionamiento de las clases de Producción Visual – Interfaz Gráfica, ver Anexo 5, capítulo 4.

4.3.6 Modificación Virtual

La Modificación Virtual está compuesta por las clases que se encargan de modificar el entorno virtual representado en la visualización (nivel gráfico, relacionado con Producción Visual) y mover el Teclado Virtual (nivel de estructura, relacionado con Representación Virtual).

La modificación gráfica incluye los movimientos de rotación, acercamiento y alejamiento de la representación gráfica además de la activación o desactivación (mediante la Interfaz Gráfica) de elementos representados (dedos, teclado y muñeca). En cambio, la modificación a nivel de estructura sólo incluye la funcionalidad de mover el Teclado Virtual.

Para mayor información sobre los conceptos y el funcionamiento de las clases de Modificación Virtual, ver Anexo 5, capítulo 5.

4.3.7 Manejo de Archivos

El Manejo de Archivos 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) de los mismos.

El usuario puede, mediante Interfaz Gráfica, iniciar el proceso de grabar el movimiento de sus manos, lo cual hará que se genere un hilo de ejecución (un Thread) para capturar los datos de las manos (de la Estructura Virtual) y, mediante la serialización, guardar los datos en un archivo temporal, para posteriormente dar la opción al usuario de guardar la grabación en un archivo en la dirección que requiera. Estos datos, que corresponden a unas sesión de trabajo, se almacenan en un archivo que tiene la extensión *.ptf* (piano technique file).

También el usuario puede, mediante Interfaz Gráfica, reproducir los archivos previamente guardados. En este caso, al igual que en la grabación, se genera un hilo de ejecución pero para copiar los datos del archivo a la Estructura Virtual.

De igual manera, se generan gráficos estadísticos y el usuario tiene la opción de exportar los datos a Excel para un registro personal o para un análisis posterior. Esto puede hacerse mediante Interfaz Gráfica. Para que esta opción se habilite, el usuario debe haber cargado previamente el archivo *.ptf* correspondiente a su sesión de trabajo..

Para mayor información sobre los conceptos y el funcionamiento de las clases de Manejo de Archivos, ver el Anexo 5, capítulo 6.

4.4 Diagrama de Clases del sistema

En esta sección se mostrará el diagrama de clases que se realizó para el desarrollo del prototipo. El diagrama de clases anteriormente mencionado se puede observar a en la Figura N° 49 y como se puede constatar, el diagrama de clases es bastante extenso, sin embargo, mantiene la lógica del Diagrama de Paquetes de la Figura N° 48.

Para mayor información sobre los conceptos y el funcionamiento de las clases del sistema, ver Anexo 5.

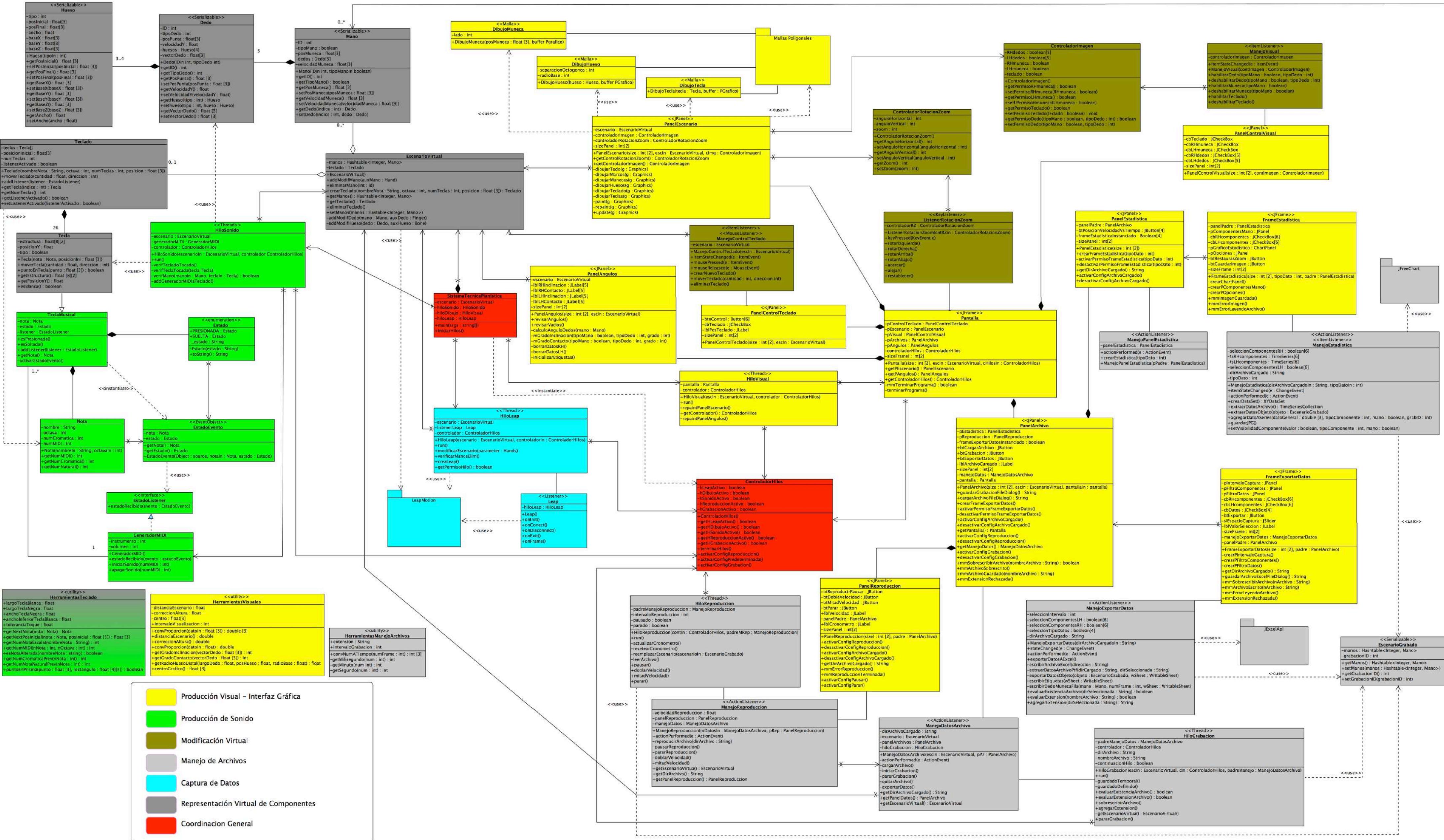


Figura N° 49: Diagrama de Clases del Sistema

4.5 Estructura Física Anexa

Para cumplir los requerimientos de la estructura física anexa, ésta tiene un 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.

En la Figura N° 50 se puede observar las partes de la estructura metálica sin ser ensambladas.



Figura N° 50: Partes de la estructura metálica sin ser ensambladas

Fuente: Elaboración propia

4.5.1 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 según diversas 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.

4.5.2 Marco

Es una estructura metálica que sujeta la superficie de vidrio en 3 lados, sujeta a la vara vertical y a su vez, tiene prensas o seguros de presión para sujetarse a mobiliario (mesa, pupitre, escritorio).

Presenta en el borde interno una forma en L para brindar soporte a la superficie de vidrio. El lado izquierdo está soldado a una vara en forma de T invertida que sirve como riel a la vara vertical para facilitar el desplazamiento y seguridad de sujeción.

En la Figura N° 51 se puede observar al marco de soporte desde una posición frontal.



Figura N° 51: Marco de soporte - Vista frontal

Fuente: Elaboración propia

En la Figura N° 52 se puede observar al marco de soporte desde una posición superior.



Figura N° 52: Marco de soporte - Vista superior

Fuente: Elaboración propia

En la Figura N° 53 se puede observar el borde derecho y una sección del borde posterior del marco.



Figura N° 53: Borde derecho y sección del borde posterior del marco de soporte

Fuente: Elaboración propia

En la Figura N° 54 se puede observar el lado izquierdo del marco.



Figura N° 54: Borde izquierdo del marco de soporte

Fuente: Elaboración propia

Las prensas que se encuentran en la parte posterior del marco le permiten sujetarse a superficies de hasta 5 cm de grosor con longitud de sujeción de 3 a 4,5 cm.

En la Figura N° 55 se puede observar un seguro de presión de la parte posterior del marco.



Figura N° 55: Seguro de presión de la parte posterior del marco de soporte

Fuente: Elaboración propia

4.5.3 Vara Vertical

Para que el Leap Motion se encuentre debajo de la superficie de vidrio y para que la distancia entre éste y la superficie pueda ser regulable, se utilizaron 2 varas de fierro, una vertical y otra horizontal.

La vara vertical es de 18 cm de longitud y en la parte superior presenta un dispositivo de presión que se ajusta al lado izquierdo del marco (en forma de T invertida).

En la Figura N° 56 se puede observar a la vara vertical.



Figura N° 56: Vara vertical

Fuente: Elaboración propia

4.5.4 Vara Horizontal

La vara horizontal es de 23 cm de longitud, en el extremo izquierdo tiene un dispositivo de presión que se ajusta con la vara vertical y, en el extremo derecho (últimos 8,3 cm) está soldado a una superficie que soporta al Leap Motion.

El soporte del Leap Motion es de aluminio debido a sus propiedades de disipación de calor y tiene una medida de 8,5 cm por 4 cm con bordes verticales alrededor (para evitar el desplazamiento accidental del Leap Motion) de 0,7 cm de altura exceptuando en el lado izquierdo ya que se deja un espacio para el cable del dispositivo.

En la Figura N° 57 se puede observar la vara horizontal.

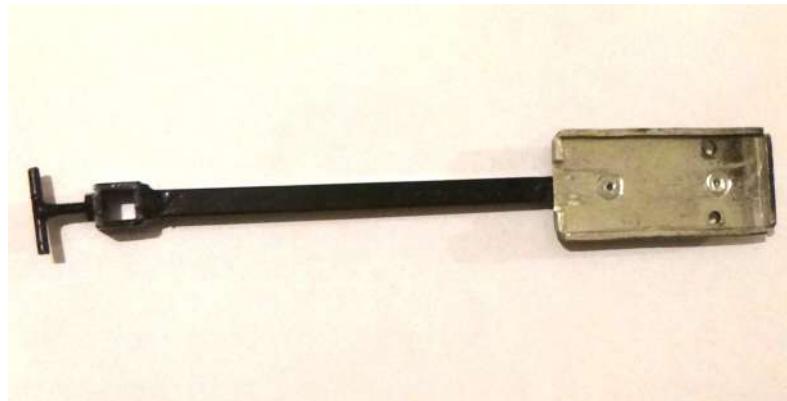


Figura N° 57: Vara horizontal

Fuente: Elaboración propia

4.5.5 Ensamblado de las partes

El marco se sujeta de la mesa mediante dos prensas soldadas a los extremos de la parte posterior del marco, la vara vertical se coloca en el borde izquierdo del marco que funciona como riel (lo cual sirve para la regulación horizontal del Leap Motion), la vara horizontal es colocada en la vara vertical mediante el dispositivo de presión (lo cual sirve para la regulación vertical del Leap Motion) y finalmente, el vidrio es puesto en el marco.

En la Figura N° 58 se puede observar imágenes las varas vertical y horizontal ensambladas.

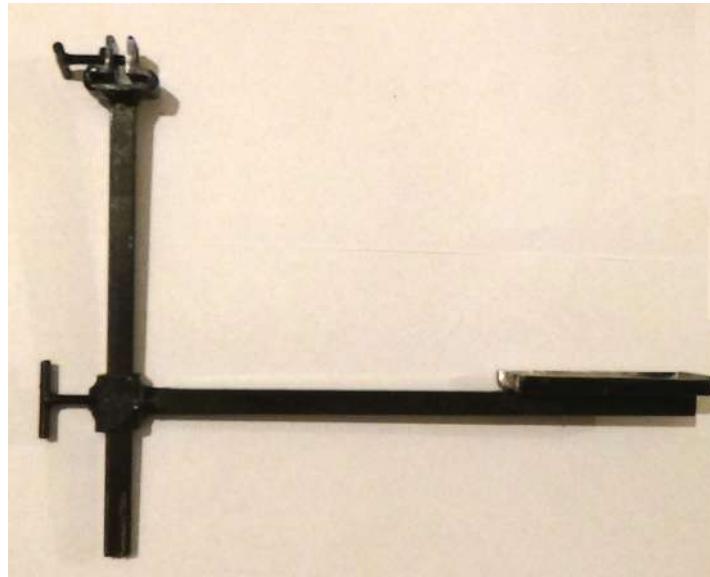


Figura N° 58: Vara vertical y horizontal ensambladas

Fuente: Elaboración propia

En la Figura N° 59 se puede observar la parte superior (dispositivo de presión) de la vara vertical ensamblado al borde izquierdo del marco.



Figura N° 59: Dispositivo de presión de la vara vertical ensamblado al borde izquierdo del marco

Fuente: Elaboración propia

En la Figura N° 60 se puede observar a las partes metálicas estructura física anexa ensambladas.



Figura N° 60: Partes metálicas de estructura física anexa ensambladas

Fuente: Elaboración propia

En la Figura N° 61 se puede observar a la estructura física anexa sujetada a una mesa.



Figura N° 61: Partes metálicas de estructura física anexa sujetado a una mesa

Fuente: Elaboración propia

En la Figura N° 62 y Figura N° 63 se puede observar a la estructura física anexa siendo utilizada en el prototipo de sistema de evaluación del desarrollo de la técnica básica pianística.

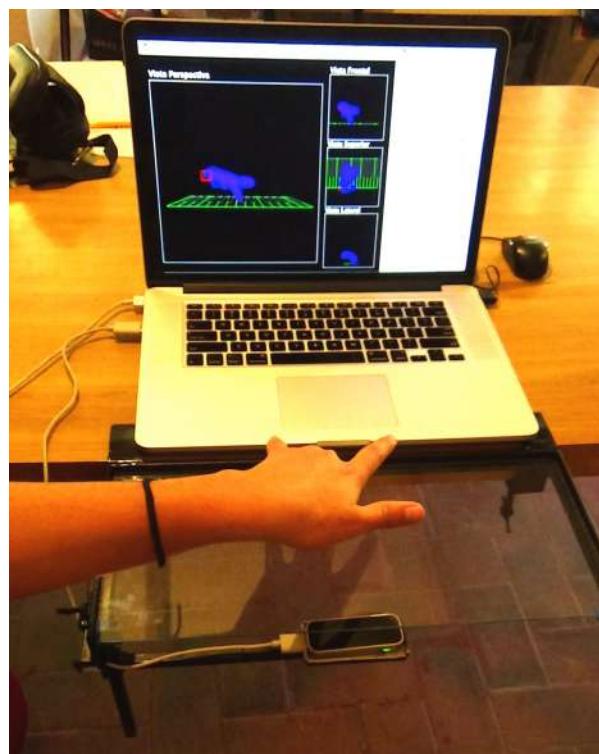


Figura N° 62: Prueba de prototipo

Fuente: Elaboración propia



Figura N° 63: Prototipo siendo utilizado por un estudiante

Fuente: Elaboración propia

4.6 Validación del prototipo en la evaluación de la técnica pianística

Una de las formas más efectivas de evaluar la capacidad técnica de los dedos son los trinos. Los trinos consisten en repetir dos notas de manera alternada, esto son principalmente utilizados como ejercicios de independencia en los primeros meses de aprendizaje formal de piano.

En la Figura N° 64 se puede observar a un estudiante de piano que con la guía de su profesor está utilizando el prototipo para evaluar el desarrollo de su técnica mediante los trinos.



Figura N° 64: Estudiante y profesor de piano utilizando el prototipo

Fuente: Elaboración propia

En la Figura N° 65 se puede observar el gráfico estadístico (posición y vs tiempo) de trinos con los dedos 3 y 4 de la mano derecha, realizados por una persona nueva en el estudio del piano (1 mes).

En la Figura N° 66 se puede observar el gráfico estadístico (posición y vs tiempo) de trinos con el dedos 3 y 4 de la mano derecha, realizados por una persona que comenzó sus estudios de piano hace 1 año.

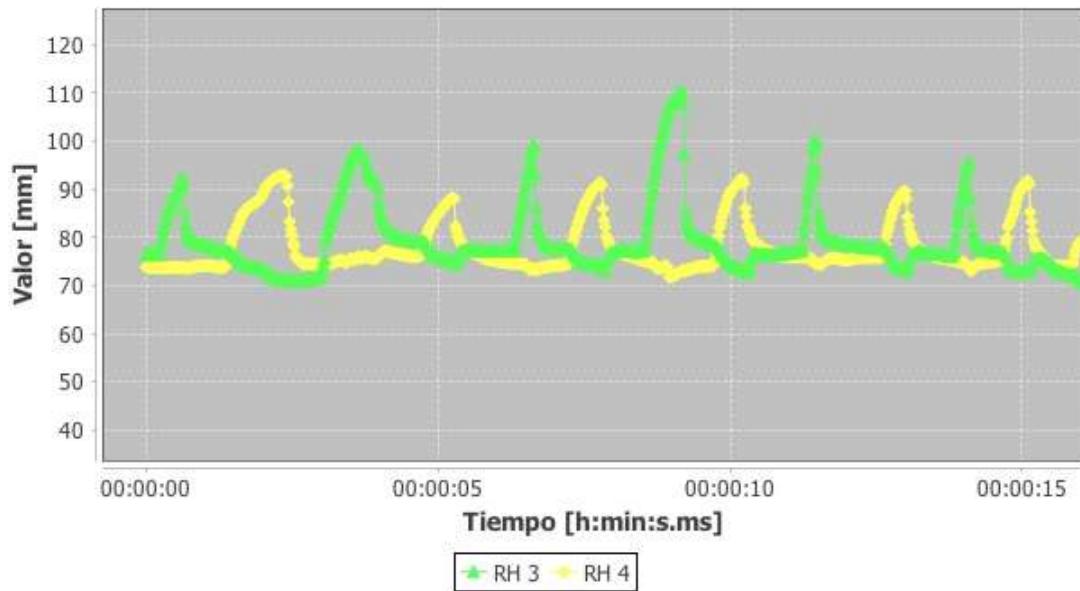


Figura N° 65: Gráfico estadístico de dedos 3 y 4 de la mano derecha de persona con un mes de estudio de piano

Fuente: Elaboración propia

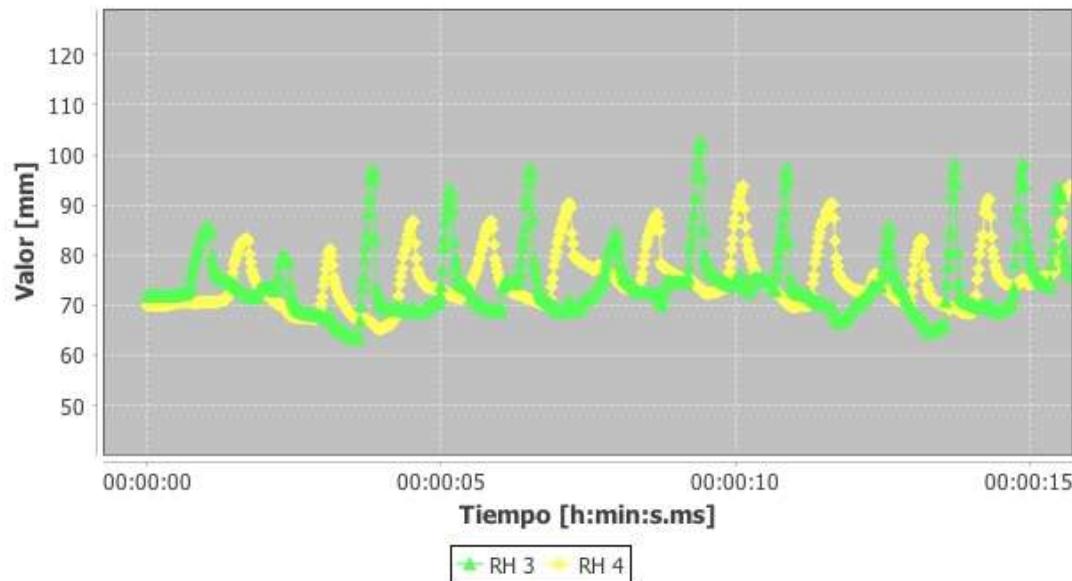


Figura N° 66: Gráfico estadístico de dedos 3 y 4 de la mano derecha de una persona con un año de estudio de piano

Fuente: Elaboración propia

Como se puede observar, el prototipo refleja el desarrollo de la técnica pianística en el aspecto de independencia de dedos evaluado con los trinos y sirve como referencia a los estudiantes para que puedan visualizar sus mejoras a corto y mediano plazo. Esto puede servir para establecer metas y solucionar dificultades técnicas en diferentes aspectos.

V EVALUACIÓN FINANCIERA

En la presente sección se realizará la evaluación financiera del proyecto de Prototipo de Sistema Computarizado para el Estudio de la Técnica Básica Pianística.

Debido a que, actualmente no existe ningún artefacto/sistema tecnológico de evaluación de desarrollo de la técnica pianística en el mercado como tal, no se puede realizar una comparación de los costos.

Los costos que necesarios para el desarrollo del prototipo se encuentran ilustrados en la Tabla N° 4.

Tabla N° 4: Costos de desarrollo de prototipo

Recurso	Unidad de Medida	Precio Unitario (Bs/UM)	Cantidad Total Requerida	Monto (Bs)
Proyectista	Hora	30	1120	33600
Computadora-Internet	Hora	3	746	2238
Leap Motion	Unidad	700	1	700
Estructura Física Anexa	Unidad	300	1	300
Material de escritorio	Paquete	60	1	60
Total				36898

Fuente: Elaboración propia

Tomando en cuenta que el prototipo es funcional, pero pueden ser requeridas ampliaciones de funcionalidad o adaptaciones, se puede deducir que el prototipo está finalizado en un 85% por lo que, completando su desarrollo, la inversión total sería de 43409 Bolivianos.

Debido a que la orientación del proyecto es investigativo, artístico y el enfoque de población es la sociedad pianística, que tiene mayor presencia en Europa, Norteamérica y Asia, se puede prever que, en caso de que el producto llegue a ser comercializado, los interesados principalmente serían instituciones educativas musicales para realizar

investigaciones propias o empresas de software que deseen adaptar la idea original del prototipo a otros sectores como medicina, educación, comunicación o entretenimiento, lo cual conllevaría pagar todos los gastos de desarrollo y un contrato de remuneración por porcentaje de ganancia que requerirían estudios completos de mercado potencial.

VI CONCLUSIONES

Debido a la inexistencia de investigación en la técnica básica pianística creados con dispositivos tecnológicos que faciliten el desarrollo de aprendizaje, se construyó un prototipo de sistema utilizando el Leap Motion, el lenguaje Java, las bibliotecas JExcel, JFreeChart y representación en Mallas Poligonales.

El prototipo fue construido tomando en cuenta los requerimientos esenciales desde el punto de vista técnico pianístico y anatómico, siendo importante la experiencia y conocimiento propio en el área.

Previo a la creación del prototipo, se hizo una valoración del hardware para realizar la captura de datos de los dedos y el lenguaje de programación para el desarrollo del software que cumpla los requerimientos establecidos.

Se realizó el diseño de software, el cual fue dividido en siete módulos, siendo éstos el módulo de captura de datos, representación virtual de componentes, producción visual, producción de sonido, modificación virtual, manejo de archivos y coordinación general. Cada módulo, si bien trabaja en coordinación con las demás, brinda una funcionalidad específica (cinco de ellos mediante hilos de ejecución).

El sistema presenta al usuario la posibilidad de visualizar el movimiento de sus manos desde diferentes perspectivas, elegir los dedos o muñecas que desea ver, guardar los datos de la posición de las manos en archivos para visualizar posteriormente, exportar los datos y finalmente, crear gráficos estadísticos. Lo anterior deriva en la facilidad del usuario para analizar el tipo de técnica básica que tiene, plantear nuevas formas de estudio, plantear objetivos y tener una visión más exacta del tipo de movimiento que realizan sus dedos.

Para la fácil comprensión de los desarrolladores posteriores se elaboró el diagrama de clase y las explicaciones del mismo (en Diseño del Prototipo y Anexos 5), exponiendo la información acerca del diseño y manejo del prototipo.

El prototipo desarrollado es innovador en el área y, sus diferentes secciones o conceptos pueden ser utilizados para la ampliación y la mejora de funcionalidades así como también, pueden ser utilizados en proyectos en otras áreas como medicina, educación, comunicación o entretenimiento.

VII RECOMENDACIONES

El prototipo ha sido construido como una herramienta inicial de investigación de la técnica pianística desde un punto de vista más científico y no ha sido considerado el hecho de evaluar si alguna técnica es correcta o incorrecta, ya que no se tienen datos ni muestras previas suficientes. Sin embargo, a futuro, el proyecto puede ser ampliado mediante inteligencia artificial para realizar una evaluación completa de efectividad de movimientos y calidad técnica, tomando muestras de pianistas profesionales con diferentes medidas de manos y con diferentes tipos de escuelas pianísticas.

Se debe considerar que, como la presente propuesta todavía tiene en gran medida un lenguaje considerado científico para la persona media, puede llegar ser poco entendible por un usuario que se encuentra en el área de aprendizaje de piano y que no tenga conocimientos básicos en coordenadas o ángulos. Por esta razón, se recomienda que, en futuros proyectos que sean destinados más al área educativa que al área de investigación de desarrollo de técnica pianística, se realice una adaptación con un lenguaje simple y se utilicen más elementos gráficos que ayuden a la comprensión.

Por otra parte, el módulo gráfico del prototipo ha sido desarrollado utilizando una biblioteca propia de gráficos que facilitaron en gran medida la realización del proyecto, llegando a cumplir con los requerimientos. Sin embargo, se pueden realizar mejoras en gráficos en la misma biblioteca o reemplazar por otras bibliotecas especializadas en gráficos 3D que, si bien son más completas, el tiempo de desarrollo podía ser extendido.

Para el desarrollo del prototipo fue utilizado un dispositivo Leap Motion que brinda un alcance equivalente a un intervalo de dos octavas del teclado. En futuros proyectos se pueden utilizar varios dispositivos Leap Motion sincronizados para tener un teclado de 8 octavas (equivalente a un piano real), construyendo así un teclado virtual completo.

En el teclado virtual del prototipo el volumen del sonido (amplitud de onda) no varía según el toque de dedo, ya que era necesario un estudio previo de la relación del volumen de sonido con la velocidad de contacto del dedo con una tecla virtual. En futuros proyectos, se podría ampliar la calidad el teclado dándole sensibilidad de toque considerando la velocidad de los dedos, lo que brindaría mucho más realismo auditivo al teclado virtual.

VIII BIBLIOGRAFÍA

1. Chang, C. (2008). *Fundamentos del Estudio del Piano*. Booksurge.
2. Real Academia Española. (s.f.). *Diccionario Usual*. (Real Academia Española) Recuperado el 1 de Abril de 2014, de Buscador de Real Academia Española: buscon.rae.es/drae/srv/search?val=t%E9cnica
3. Narejos, A. (1998). Teoría y práctica de la ejecución pianística. *I*.
4. Garcia, R. L. (2010). Representaciones Gráficas del sonido: Una herramienta para el análisis de la interpretación pianística. (65).
5. Kaemper, G. (1968). Techniques pianistiques: l'évolution de la technologie pianistique.
6. Zhang, Z. (2012). Microsoft kinect sensor and its effects. *19* (2).
7. Khoshelham, K. (2011). Accuracy analysis of kinect depth data. *38* (5).
8. Oikomidis, I., Antonis, A., & Kyriazis, N. (2011). Efficient model-based 3D tracking of hand articulations using Kinect.
9. Hager-Ross, C., & Schieber, M. (2000). Quantifying the independence of human finger movements: comparisons of digits, hands and movement frequencies. *20* (22).
10. Leap Motion. (s.f.). *Product*. Recuperado el 1 de Mayo de 2014, de Sitio web de Leap Motion: www.leapmotion.com/product
11. Gill, D. (s.f.). *IT Services. Solutions Technologies*. Recuperado el 1 de Mayo de 2014, de Documentos de IT Services, Solutions Technologies: http://www.pdsit.net/wp-content/uploads/2013/10/Dan-Gill_PDS-IT-Deck-1.pdf
12. Leap Motion Inc. (2014). *Developer Leap Motion*. Recuperado el 1 de Mayo de 2014, de Developer Portal: <http://developer.leapmotion.com>
13. Leap Motion Inc. (s.f.). *Tracking Hands, Fingers, and Tools*. Recuperado el 1 de Mayo de 2014, de Documentación: https://developer.leapmotion.com/documentation/java/devguide/Leap_Tracking.html
14. Leap Motion Inc. (s.f.). *Hand*. Recuperado el 1 de Mayo de 2014, de Documentación de Hand: <https://developer.leapmotion.com/documentation/java/api/Leap.Hand.html>

-
-
15. Leap Motion Inc. (s.f.). *Finger*. Recuperado el 1 de Mayo de 2014, de Documentación de Finger: <https://developer.leapmotion.com/documentation/java/api/Leap.Finger.html>
 16. Fujii, K. (2005). *La Ciencia y la Técnica Fundamental de Piano*. Cochabamba.
 17. Leap Motion Inc. (s.f.). *Frame*. (Leap Motion Inc.) Recuperado el 8 de Mayo de 2014, de Documentación de Frame:
<https://developer.leapmotion.com/documentation/python/api/Leap.Frame.html>
 18. Hillel, M., & Olson, L. F. (s.f.). *Piano for the Developing Musician*. Recuperado el 5 de Mayo de 2014, de Hand: http://www.pdmpiano.org/prelim_p004_1_hand.html
 19. Last, J. (1985). *The Young Pianist*. Oxford University Press.
 20. Walden Pond. (s.f.). *American School of Piano Teaching*. Recuperado el 5 de Mayo de 2014, de Fingering: <http://www.americanpianoschool.com/college3.html>
 21. Ákos, J., Harcos, P., Karoly, R., & Fazekas, G. (2005). Analysis of finger-tapping movement. *141* (1).
 22. Swift, A. (1997). *An Introduction to MIDI*. Recuperado el 12 de mayo de 2014, de http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/aps2/
 23. Oracle. (s.f.). *Documentation Package javax.sound.midi*. (Oracle) Recuperado el 13 de Mayo de 2014, de [http://docs.oracle.com/javase/7/docs/api\(javax/sound/midi/package-summary.html](http://docs.oracle.com/javase/7/docs/api(javax/sound/midi/package-summary.html)
 24. Berg, R. (2007). *MIDI Keyboard GUI*. Recuperado el 1 de Mayo de 2014, de <http://www.raymondberg.com/paste/mp>
 25. Worthen, J. (2007). *Robert Schumann: Life and Death of a Musician*. Yale: Yale University Press.
 26. Czerny, C. (1842). *Recollections from my Life*. (E. Sanders, Trad.) The Musical Quarterly.
 27. Lanza, S. (1997). Nacimiento del piano romántico y la escuela moderna de técnica pianística en España durante la primera mitad del siglo XIX. *Edades: revista de historia* .

-
-
28. Krakenberger, J. (2001). *Rol de la Música en la Formación del Ser Humano*. (R. C. MusicaClasicayMusicos.com, Productor) Recuperado el 9 de 12 de 2014, de Musica Clasica y Musicos: <http://www.musicaclassicaymusicos.com/rol-de-la-musica.html>
29. Schmidt, H.-M., & Lanz, U. (2003). *Surgical Anatomy of the Hand*. EEUU.
30. Rodríguez, C. P., Pineda, J. A., & Sánchez, D. (2013). Prototipo traductor de señales manuales a texto legible, utilizando Kinect. *Revista Avances*, 10 (2).
31. Ramirez, E. (Julio de 2012). *3D BODY SCANNING*. Recuperado el 12 de 12 de 2014, de Archivos de Centro de Computación Gráfica:
<http://lcg.ciens.ucv.ve/~esmitt/files/3dscanning.pdf>
32. Frenzel, L. L. (2003). *Sistemas electrónicos de comunicaciones* (3ra Edición ed.). Mexico D.F., México: Alfaomega.
33. Oracle. (s.f.). *Conozca más sobre la tecnología Java*. Recuperado el 6 de 1 de 2015, de Java: <https://www.java.com/es/about/>
34. López, J. T. (2013). *Reconocimiento e interpretación de gestos con dispositivo Leap*. Trabajo Fin de Máster , Escuela de Ingeniería y Arquitectura, Universidad de Zaragoza, Departamento de Informática e Ingeniería de Sistemas, Zaragoza.
35. Leap Motion Inc. (s.f.). *Bone*. Recuperado el 8 de 1 de 2015, de Documentación de Bone:
https://developer.leapmotion.com/documentation/java/api/Leap.Bone.html#javaclasscom_1_leapmotion_1_leap_1_bone
36. Tobler, R., & Maierhofer, S. (2006). *A Mesh Data Structure for Rendering and Subdivision*. VRVis Research Center, Viena.
37. Foley, J. D., Dam, A. v., Feiner, S. K., & Hughes, J. F. (1996). *Computer Graphics Principles and practice* (2da Edicion ed.). (I. E. Board, Ed.) Addison-Wesley Publishing Company.
38. Oracle. (2014). *Overview of the MIDI Package*. Recuperado el 9 de 2 de 2015, de Oracle Java Documentation: <http://docs.oracle.com/javase/tutorial/sound/overview-MIDI.html>

-
-
39. Oracle. (2014). *Accessing MIDI System Resources*. Recuperado el 8 de 1 de 2015, de Oracle Java Documentation: <http://docs.oracle.com/javase/tutorial/sound/accessing-MIDI.html>
40. Oracle. (2014). *Interface Synthesizer*. Recuperado el 10 de 1 de 2015, de Java Documentation:
<http://docs.oracle.com/javase/7/docs/api/javax/sound/midi/Synthesizer.html>
41. Oracle. (2014). *Interface MidiChannel*. Recuperado el 10 de 1 de 2015, de Java Documentation:
<http://docs.oracle.com/javase/7/docs/api/javax/sound/midi/MidiChannel.html>
42. JFree. (2014). *Welcome to JFreeChart*. (Object Refinery Limited) Recuperado el 10 de 1 de 2015, de JFreeChart: <http://www.jfree.org/jfreechart/>
43. JFree. (s.f.). *Class ChartPanel*. Recuperado el 10 de 1 de 2015, de JFree Documentation:
<http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/ChartPanel.html>
44. JFree. (s.f.). *Class JFreeChart*. Recuperado el 10 de 1 de 2015, de JFree Documentation JFreeChart: <http://www.jfree.org/jfreechart/api/javadoc/>
45. JFree. (s.f.). *Interface Dataset*. Recuperado el 10 de 1 de 2015, de JFree Documentation:
<http://www.jfree.org/jfreechart/api/javadoc/org/jfree/data/general/Dataset.html>
46. Quicly Java. (s.f.). *Quickly Java* . Recuperado el 10 de 1 de 2015, de Write to Excel in Java Using JExcel API: <http://www.quicklyjava.com/write-to-excel-in-java/>
47. Chiantore, L. (2001). *Historia de la técnica pianística*. Madrid: Alianza Música.
48. Puelles López, J. (2004). *Hausmusik: Musica Clasica y Musicos*. (R. C. MusicaClasicayMusicos.com, Productor) Recuperado el 7 de 12 de 2014, de Musica Clasica y Musicos: <http://www.musicaclasicaymusicos.com/>
49. Openkinect. (7 de Marzo de 2012). *Main Page Openkinect*. Recuperado el 12 de 12 de 2014, de Openkinect Project: http://openkinect.org/wiki/Main_Page

-
-
50. Microsoft. (2015). *Microsoft Research*. Recuperado el 3 de 1 de 2015, de Fully Articulated Hand Tracking: <http://research.microsoft.com/en-us/projects/handpose/>
 51. Microsoft. (s.f.). *Kinect for Windows*. Recuperado el 3 de 1 de 2015, de Kinect for Windows Features: <http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx>
 52. Microsoft. (2015). *Buy the Kinect Hardware*. Recuperado el 5 de 1 de 2015, de Get Kinect: <http://www.microsoft.com/en-us/kinectforwindows/purchase/default.aspx>
 53. Leap Motion . (s.f.). *Leap Motion SDK and Plugin Documentation*. Recuperado el 20 de 6 de 2014, de Developer Portal:
<https://developer.leapmotion.com/documentation/index.html?proglang=current>
 54. Apple. (17 de Septiembre de 2014). *Programming with Objective C*. Recuperado el 21 de 10 de 2014, de Mac Developer Library:
<https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>
 55. cplusplus. (s.f.). *A Brief Description*. Obtenido de Cplusplus:
<http://wwwcplusplus.com/info/description/>
 56. Microsoft. (2014). *Visual C++ resources*. Recuperado el 6 de 1 de 2015, de Visual Studio: <https://msdn.microsoft.com/en-us/vstudio/hh386302.aspx>
 57. midifile. (s.f.). *C++ library for parsing Standard MIDI Files*. Recuperado el 1 de 7 de 2014, de Midifile: <http://midifile.sapp.org>
 58. LibXL. (s.f.). *excel library for developers*. Recuperado el 6 de 7 de 2014, de LibXL:
<http://libxl.com>
 59. Unity. (s.f.). *La mejor plataforma para crear juegos*. Recuperado el 9 de 6 de 2014, de Unity: <https://unity3d.com/es/unity>
 60. Foriero. (2012). *MIDI Unified*. Recuperado el 1 de 7 de 2014, de Studio Foriero:
<http://studio.foriero.com/products/midi/midi.php>
 61. Python. (s.f.). *About Python*. Recuperado el 3 de 1 de 2015, de Python:
<https://www.python.org/about/>

-
-
62. ensemblerobot . (s.f.). *Python MIDI*. Recuperado el 7 de Julio de 2014, de GitHub:
<https://github.com/vishnubob/python-midi>
63. VPython. (s.f.). *VPython 3D Programming for Ordinary Mortals*. (B. Sherwood, Productor) Recuperado el 10 de Julio de 2014, de Sitio Web de VPython:
<http://vpython.org>
64. Oracle. (s.f.). *Conozca más sobre la tecnología Java*. Recuperado el 1 de Julio de 2014, de Sitio web de Java: <https://www.java.com/es/about/>
65. Java3D. (s.f.). *Website de Java3D*. Recuperado el 3 de Agosto de 2014, de Project Java.net: <https://java3d.java.net>
66. Oracle. (s.f.). *JDK-7124557 : [macosx] Java3D HelloUniverse doesn't start*. Obtenido de Java Bug Database: http://bugs.java.com/view_bug.do?bug_id=7124557
67. The University Chicago Medicine. (s.f.). *Anatomía de la Mano*. Recuperado el 12 de 2 de 2015, de The University Chicago Medical Center:
<http://www.uchospitals.edu/online-library/content=S04195>
68. Samper, E. (16 de Diciembre de 2006). *¿Por qué crujen los nudillos?* Recuperado el 12 de Febrero de 2015, de MedTempus, Blog de medicina y salud:
<http://medtempus.com/archives/por-que-crujen-los-nudillos/>
69. Leap Motion. (s.f.). *API Overview*. Recuperado el 17 de 2 de 2015, de Developer Portal Leap Motion:
https://developer.leapmotion.com/documentation/java/devguide/Leap_Overview.html
70. Leap Motion. (s.f.). *Coordinate System*. Recuperado el 28 de Marzo de 2015, de Leap Motion Developer Portal:
https://developer.leapmotion.com/documentation/java/devguide/Leap_Coordinate_Mapping.html
71. Chantore, L. (2001). *Historia de la técnica pianística*. Madrid: Alianza Música.
72. SpectraSuite. (Mayo de 2012). *Installation and Operation Manual*. Recuperado el 5 de Marzo de 2015, de OceanOptics: <http://oceanoptics.com/wp-content/uploads/SpectraSuite.pdf>

73. IBM. (10 de Diciembre de 2012). *IBM developerWorks*. Recuperado el 4 de Noviembre de 2014, de Introducción a la programación Java, parte 2: Construcciones para aplicaciones del mundo real:
<http://www.ibm.com/developerworks/ssa/java/tutorials/j-introtojava2/section11.html>

IX ANEXOS

Anexo 1: Explicación de Términos Musicales

Nota Musical

Una nota musical es un sonido determinado por una vibración regular producido por un cuerpo sonoro (normalmente cuerdas, membranas o labios) de afinación definida. Físicamente, el sonido de una nota musical tiene una frecuencia principal de vibración y varias frecuencias de vibraciones secundarias denominadas *frecuencias armónicas* que determinan el llamado “color” o “timbre” del sonido, las cuales son frecuencias múltiplo de la frecuencia principal.

Existen siete nombres de notas musicales en la notación occidental: do, re, mi, fa, sol, la y si. Es necesario aclarar que el nombre de las notas en la notación anglosajona corresponde a las letras C, D, E, F, G, A, B respectivamente.

Cada una de las notas puede ser alterada, es decir, su frecuencia puede ser modificada (aumentada o disminuida) con las *alteraciones musicales*. Éstas suben o bajan un *semitono* (ver definición de Semitono) de la nota a la cual están alterando.

Las alteraciones musicales que suben un semitono o bajan un semitono a la nota son, el *sostenido* representado con el símbolo \sharp y el *bemol* representado con el símbolo \flat . Debido a la distribución de las notas, existen notas alteradas que son *enarmónicas* de otras notas, es decir, tienen diferente nombre pero tienen la misma frecuencia.

Las notas naturales y las notas alteradas forman una secuencia repetida, llamada *escala cromática*, donde las notas están distanciadas por semitonos (ver definición de Semitono). La secuencia es: do, do \sharp o re \flat , re, re \sharp o mi \flat , mi, fa, fa \sharp o sol \flat , sol, sol \sharp o la \flat , la, la \sharp o si \flat , si. Esta secuencia se repite desde las frecuencias de menor valor (notas graves o bajas) hasta las de mayor valor (notas agudas o altas), cada ciclo de repetición es un intervalo de octava (ver definición de Octava) donde la primera nota del ciclo es la última del ciclo anterior.

El teclado del piano es uno de los lugares donde se puede ver de manera gráfica la secuencia de las notas. En la Figura N° 67 se puede observar a una sección (2 octavas) del

teclado del piano con los nombres de las notas en cada tecla, las notas graves se encuentran a la izquierda y las notas agudas a la derecha.



Figura N° 67: Dos octavas de teclado con los nombres de las notas en cada tecla

Fuente: Elaboración propia

Para identificar las notas en diferentes octavas se utiliza el *índice acústico científico*, el cual es un índice registral que asigna el número 0 a la octava (por lo que es llamado también número de octava) que comienza con la nota más grave del órgano, el do0 correspondiente a 16,3516 Hz.

El patrón de afinación definido internacionalmente es el de la nota La4 que corresponde a una frecuencia de vibración de 440 Hz.

Intervalo

Intervalo es la distancia entre dos notas, pueden ser medidas cuantitativamente (en grados o notas naturales) o cualitativamente (en tonos y semitonos). Por ejemplo, desde la nota do a la nota re existe un intervalo cuantitativo de 2da (segunda) y, por otra parte, un intervalo cualitativo de 1 tono.

Uno de los intervalos más utilizados es el de octava, es decir, es un intervalo que empieza y termina con notas del mismo nombre correspondiente dos frecuencias diferentes. Este intervalo está compuesto por 12 semitonos.

Tono y semitono

El tono es un intervalo musical que es igual a un sexto de octava. La mitad de un tono es un semitono y equivale a un doceavo de octava (ver Definición de frecuencia de una nota musical).

Definición de la frecuencia de una nota musical

Cada nota tiene una frecuencia principal matemáticamente definida y regular, según el sistema de afinación promediada. Como se mencionó anteriormente, el intervalo de octava está compuesto por 12 semitonos y, cada nota que dista de otra en un intervalo de octava tiene el doble de frecuencia (y tendrá el mismo nombre). Por lo tanto, si la frecuencia de una nota es F_0 , su octava tendrá la frecuencia F_{12} equivalente a dos veces F_0 , tal como se muestra en la siguiente ecuación

$$F_{12} = 2 F_0 \quad (18)$$

Cada frecuencia de nota F_{n+1} que se encuentra un semitono superior de otra F_n , matemáticamente tienen una relación

$$F_{n+1} = x F_n \quad (19)$$

Siguiendo la secuencia lógica, se puede saber que

$$F_{12} = x F_{11}, \quad F_{11} = x F_{10}, \quad F_{10} = x F_9, \dots, \quad F_1 = x F_0 \quad (20)$$

Por lo tanto, reemplazando las equivalencias de la ecuación (20) se tiene que

$$F_{12} = x^{12} F_0 \quad (21)$$

Tomando en cuenta (18) y (21) se puede deducir

$$x^{12} = 2 \quad (22)$$

Escrito de otra manera

$$x = \sqrt[12]{2} \quad (23)$$

Por lo tanto, cada nota con la frecuencia F_{n+1} que es un semitono superior a F_n estará definida por

$$F_{n+1} = \sqrt[12]{2} F_n \quad (24)$$

Siendo i el número de semitonos superiores, lo anterior se puede generalizar como

$$F_{n+i} = (\sqrt[12]{2})^i F_n \quad (25)$$

En la Tabla Nº 5 se puede observar el valor de las frecuencias de todas las notas (de la escala cromática) desde la nota La0 hasta la nota Do8, que es el registro de un piano estándar.

Tabla Nº 5: Valor de frecuencias de las notas desde La0 hasta Do8

Distancia de La0 en semitonos	Frecuencia en Hz	Nombre de Nota	Índice Acústico Científico
0	27,5	La	0
1	29,13523509	La#/Si♭	0
2	30,86770633	Si	0
3	32,70319566	Do	1
4	34,64782887	Do#/Re♭	1
5	36,70809599	Re	1
6	38,89087297	Re#/Mi♭	1
7	41,20344461	Mi	1
8	43,65352893	Fa	1
9	46,24930284	Fa#/Sol	1
10	48,9994295	Sol	1
11	51,9130872	Sol#/La♭	1
12	55	La	1
13	58,27047019	La#/Si♭	1
14	61,73541266	Si	1

Distancia de La0 en semitonos	Frecuencia en Hz	Nombre de Nota	Índice Acústico Científico
15	65,40639133	Do	2
16	69,29565774	Do#/Re♭	2
17	73,41619198	Re	2
18	77,78174593	Re#/Mi♭	2
19	82,40688923	Mi	2
20	87,30705786	Fa	2
21	92,49860568	Fa#/Sol♭	2
22	97,998859	Sol	2
23	103,8261744	Sol#/La♭	2
24	110	La	2
25	116,5409404	La#/Si♭	2
26	123,4708253	Si	2
27	130,8127827	Do	3
28	138,5913155	Do#/Re♭	3
29	146,832384	Re	3
30	155,5634919	Re#/Mi♭	3
31	164,8137785	Mi	3
32	174,6141157	Fa	3
33	184,9972114	Fa#/Sol♭	3
34	195,997718	Sol	3
35	207,6523488	Sol#/La♭	3
36	220	La	3
37	233,0818808	La#/Si♭	3
38	246,9416506	Si	3
39	261,6255653	Do	4

Distancia de La0 en semitonos	Frecuencia en Hz	Nombre de Nota	Índice Acústico Científico
40	277,182631	Do#/Re♭	4
41	293,6647679	Re	4
42	311,1269837	Re#/Mi♭	4
43	329,6275569	Mi	4
44	349,2282314	Fa	4
45	369,9944227	Fa#/Solb	4
46	391,995436	Sol	4
47	415,3046976	Sol#/Lab	4
48	440	La	4
49	466,1637615	La#/Sib	4
50	493,8833013	Si	4
51	523,2511306	Do	5
52	554,365262	Do#/Reb	5
53	587,3295358	Re	5
54	622,2539674	Re#/Mib	5
55	659,2551138	Mi	5
56	698,4564629	Fa	5
57	739,9888454	Fa#/Solb	5
58	783,990872	Sol	5
59	830,6093952	Sol#/Lab	5
60	880	La	5
61	932,327523	La#/Sib	5
62	987,7666025	Si	5
63	1046,502261	Do	6
64	1108,730524	Do#/Reb	6

Distancia de La0 en semitonos	Frecuencia en Hz	Nombre de Nota	Índice Acústico Científico
65	1174,659072	Re	6
66	1244,507935	Re#/Mib	6
67	1318,510228	Mi	6
68	1396,912926	Fa	6
69	1479,977691	Fa#/Sol♭	6
70	1567,981744	Sol	6
71	1661,21879	Sol#/La♭	6
72	1760	La	6
73	1864,655046	La#/Si♭	6
74	1975,533205	Si	6
75	2093,004522	Do	7
76	2217,461048	Do#/Re♭	7
77	2349,318143	Re	7
78	2489,01587	Re#/Mi♭	7
79	2637,020455	Mi	7
80	2793,825851	Fa	7
81	2959,955382	Fa#/Sol♭	7
82	3135,963488	Sol	7
83	3322,437581	Sol#/La♭	7
84	3520	La	7
85	3729,310092	La#/Si♭	7
86	3951,06641	Si	7
87	4186,009045	Do	8

Fuente: Elaboración propia

En la Figura N° 68 se puede observar la gráfica resultante según los valores de Frecuencia en Hz y Distancia de La0 en Semitonos de la Tabla N° 5.

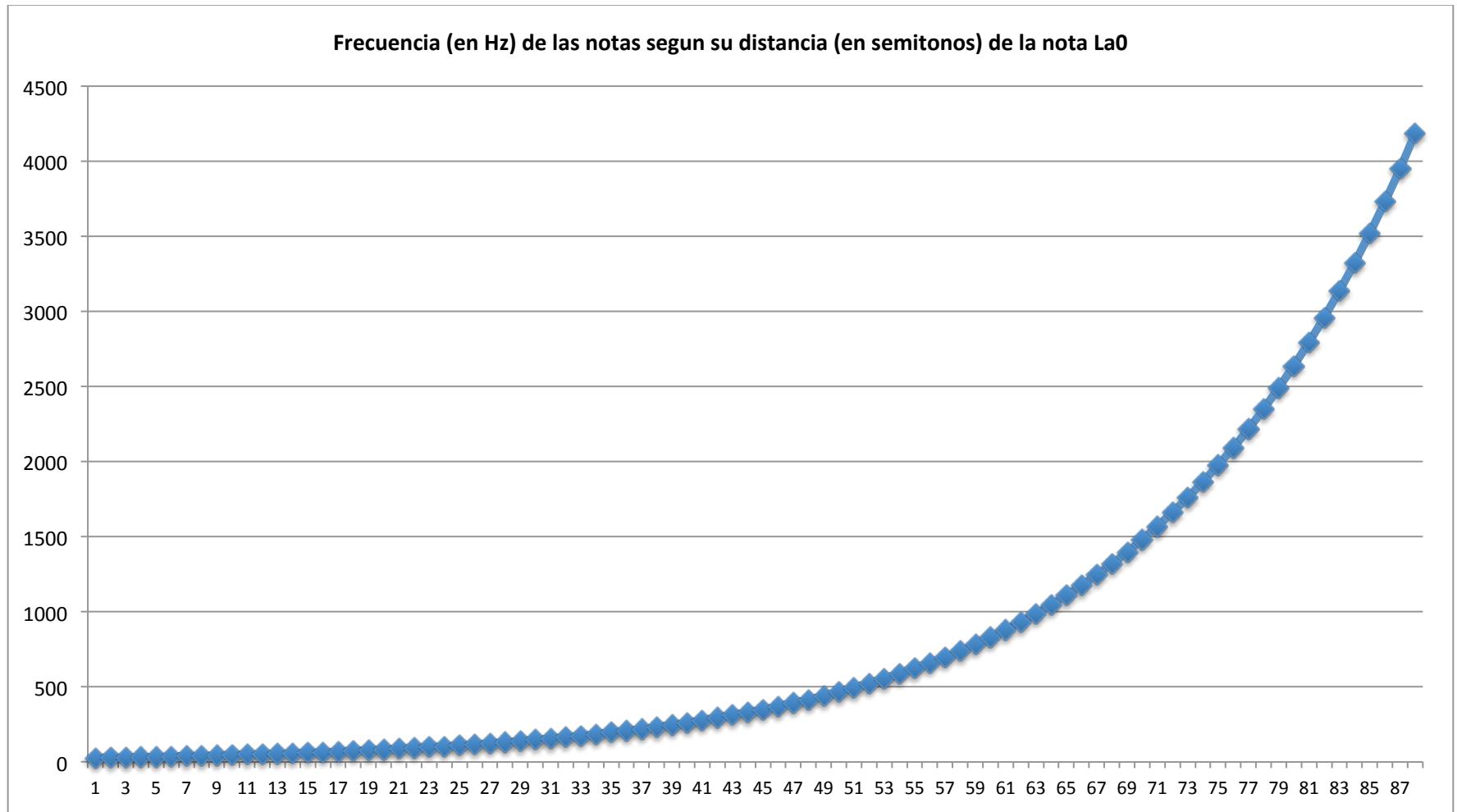


Figura N° 68: Gráfico de Frecuencias en Hz y distancia de La0 en Semitonos

Fuente: Elaboración propia

Anexo 2: Mallas Poligonales en Java

El proyecto de Mallas Poligonales fue realizado en la materia de Infografía de la Universidad Privada Boliviana y fue adaptado para el presente proyecto.

El proyecto original consistía en la representación gráfica de objetos tridimensionales utilizando la teoría de Mallas Poligonales con el lenguaje Java. Cada objeto debía ser graficado en cuatro ventanas en la pantalla, éstas debían contener el gráfico del objeto mostrado desde la vista frontal, superior, lateral y en perspectiva. Además, el proyecto debía permitir mover el o los objetos desde el teclado.

La teoría de Mallas Poligonales se encuentra explicado en el Marco Teórico y es aplicada en el lenguaje Java junto con las bibliotecas de dibujo `java.awt` para la creación de las clases que dibujan la representación de objetos con Mallas Poligonales.

Diagrama de Clases

En la Figura N° 69 se puede observar el diagrama de clases de Mallas Poligonales.

Para que un objeto sea dibujado debe ser modelado con vértices y aristas, por lo tanto debe heredar de la clase `Malla`, que representa a una Malla Poligonal.

`Malla` está compuesta por vértices y aristas, que son `VList` y `AList` respectivamente, éstas clases son listas de objetos `VNodo` y `ANodo`. 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`). A su vez, cada objeto `Arista` es formada por dos vértices (atributos `v1` y `v2`).

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 las ventanas que muestran al espacio desde una posición frontal, lateral, superior y en perspectiva. Éstos datos (contenidos en los atributos `posFrontal`, `posLateral`, `posSuperior` y `posPerspectiva`) están transformados a 2 dimensiones para poder ser dibujados en un `JFrame`.

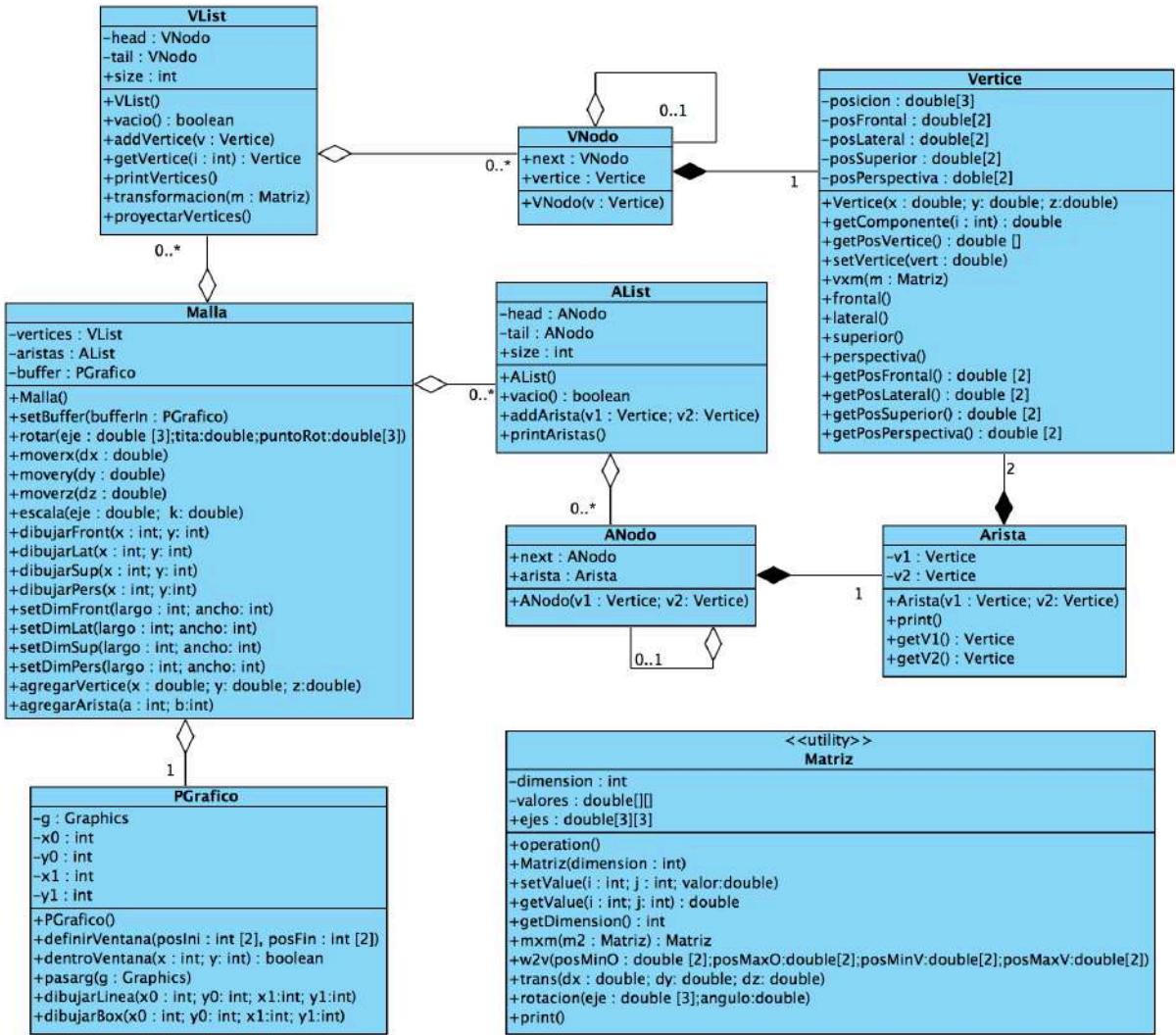


Figura N° 69: Diagrama de Clases de Mallas Poligonales

Fuente: Elaboración propia

Modelamiento de un objeto con Mallas Poligonales

Cada objeto que es modelado con Mallas Poligonales, hereda de la clase **Malla**. Se utilizan los métodos **agregarVertice** para agregar los vértices necesarios y, posteriormente se utiliza el método **agregarArista** para crear las aristas (conexiones entre vértices) necesarios para modelar el objeto.

Cuando se utiliza el método **agregarVertice** de **Malla**, se utiliza a su vez el método **addVertice** de **VList** para que éste instancie un **VNodo** y le asigne el vértice a su atributo **vertice**. Cuando se crea un vértice, es necesario tener el dato de posición ya que

`Vertice` tiene en su constructor los parámetro `x`, `y` y `z` (de tipo `double`), los cuales son almacenados en el atributo `posicion`.

Cuando se utiliza el método `agregarArista` de `Malla`, se utiliza a su vez el método `addArista` donde es necesario proporcionar como parámetros dos vértices. El método `agregarArista` utiliza el método `getVertice` de `VLista` (donde se necesita un `int` que indique la posición del vértice en la lista) para asignar los vértices requeridos a `ALista`.

Movimiento de Mallas Poligonales

Para que una Malla Poligonal se mueva, se hace que todos los vértices que lo componen realicen el movimiento en la dirección requerida. El movimiento se realiza mediante los métodos `moverx`, `movey` o `moverz` que, modifican la posición (atributo `posición` de `Vertice`) en cada uno de los vértices, los cuales se encuentran en el atributo `vertices` de la clase `Malla`.

Para realizar la rotación de una Malla Poligonal, se utiliza el método `rotar` de la clase `Malla` que, realiza los cálculos de la matriz (en los métodos de `mxm` de `Matriz`) en base a los datos del ángulo de rotación y el eje que son dados en el parámetro del método. Posteriormente, se utiliza el método `vxm` de cada `Vertice` (del atributo `vertices` de `Malla`) para realizar los cambios requeridos en la posición (atributo `posición`).

El efecto de movimiento de una cámara de visualización de Mallas Poligonales se puede conseguir mediante el movimiento de Mallas Poligonales, ya sea rotación o traslación.

Dibujo de Mallas Poligonales

Para realizar los dibujos en todas las ventanas requeridas, se utiliza el método `dibujarFront`, `dibujarLat`, `dibujarSup` y `dibujarPers` que, a su vez utilizan el método `proyectarVertices` en cada uno de los vértices (objetos `Vertice`) de las aristas del atributo `aristas`.

El método `proyectarVertices` hace que cada vértice sea proyectado en coordenadas en 2D utilizando los métodos frontal, lateral, superior y perspectiva que, asignan un valor a los atributos `posFrontal`, `posLateral` y `posPerspectiva` de cada `Vertice`

según la posición visual requerida y un escalamiento con el método `w2v` de `Matriz`, el cual ajusta la imagen según las ventanas (posiciones en 2D de los vértices) y los parámetros asignados de rango de visibilidad.

Posteriormente, los métodos `dibujarFront`, `dibujarLat`, `dibujarSup` y `dibujarPers`, mediante el atributo `g` (tipo `Graphic`) de la clase `PGrafico` realiza el dibujo de una línea (mediante el método `dibujarLinea`) entre la posición de los vértices de cada arista, la cual es evaluada (en caso de que se salga de la posición de una ventana) gracias a que se realiza una definición de tamaño de cada ventana mediante los métodos `setDimFront`, `setDimLat`, `setDimSup`, `setDimPers`.

Anexo 3: Descripción de casos de uso

Tabla N° 6: Descripción de caso de uso Procesar Datos Visuales

Caso de Uso:	Procesar Datos Visuales	
Actores:	Módulo de Modificación Virtual, Sistema Audiovisual	
Descripción	Procedimientos que se encargan de procesar los datos del Espacio Virtual para poder crear la representación gráfica de los elementos existentes.	
Precondición	El caso de uso Procesar Datos del Escenario Virtual requiere el presente caso de uso	
Flujo Principal	Paso	Acciones
	1	Módulo de Captura y Procesamiento de Datos verifica la lista de objetos a ser representados gráficamente en el Módulo de Modificación Virtual
	2	Módulo Captura y Procesamiento de Datos crea la estructura gráfica de los elementos existentes en el Espacio Virtual
	3	Módulo Audiovisual recibe los datos de gráficos
Flujo Secundario	2A.1	Módulo de Modificación Virtual realiza cambio en la Representación Gráfica (rotación, traslación, acercamiento o alejamiento).
	2A.2	Módulo de Captura y Procesamiento de Datos realiza cambios predeterminados de posición de objetos requeridos.
	2A.3	Vuelve a Paso 2 de Flujo Principal
Poscondiciones	El Módulo Audiovisual recibe datos de los gráficos a ser dibujados	

Fuente: Elaboración propia

Tabla N° 7: Descripción de caso de uso Procesar Datos Sonoros

Caso de Uso:	Procesar Datos Sonoros	
Actores:	Módulo Audiovisual	
Descripción	Procedimientos que se encargan de verificar si existen datos que produzcan sonido para enviar la información al Módulo Audiovisual	
Precondición	El caso de uso Procesar Datos del Escenario Virtual requiere el presente caso de uso	
Flujo Principal	Paso	Acciones
	1	Módulo de Captura y Procesamiento de Datos evalúa si los elementos del Espacio Virtual producen sonido (si los dedos tocan alguna tecla)
	2	En caso de que Módulo de Captura y Procesamiento de Datos encuentre elementos del Espacio Virtual que produzcan sonido, envía los datos de la tecla o teclas al Módulo Audiovisual
	3	Módulo Audiovisual recibe los datos para emitir sonido
Flujo Secundario	2A.1	Módulo de Captura y Procesamiento de Datos no encuentra elementos del Espacio Virtual que produzcan sonido.
Poscondiciones	El Módulo Audiovisual recibe datos de las notas a ser producidas	

Fuente: Elaboración propia

Tabla N° 8: Descripción de caso de uso Ver Escenario Virtual

Caso de Uso:	Ver Escenario Virtual	
Actores:	Usuario, Módulo de Captura-Procesamiento de Datos	
Descripción	Conjunto de procedimientos necesario que, el Módulo realiza para mostrar el Escenario Virtual representado en gráficos 3D al Usuario.	
Precondición	El Módulo Audiovisual recibe datos de los gráficos a ser dibujados	
Flujo Principal	Paso	Acciones
	1	Módulo Audiovisual crea la representación gráfica según los datos de gráficos a ser dibujados
	2	Módulo Audiovisual muestra a Usuario la representación gráfica en la pantalla
	El Usuario debe ver la representación gráfica de los elementos del Espacio Virtual en la pantalla	

Fuente: Elaboración propia

Tabla N° 9: Descripción de caso de uso Ver Panel de Ángulos

Caso de Uso:	Ver Panel de Ángulos	
Actores:	Usuario, Módulo de Captura-Procesamiento de Datos	
Descripción	Conjunto de procedimientos que el Módulo realiza para mostrar el Panel de Ángulos (que contienen los datos de los ángulos de los dedos) al Usuario.	
Precondición	El Módulo Audiovisual recibe datos de los ángulos para ser mostrados en la pantalla	
Flujo Principal	Paso	Acciones
	1	Módulo Audiovisual crea la representación gráficas(Panel de Ángulos) según los datos los datos de ángulos recibidos
	2	Módulo Audiovisual muestra a Usuario el Panel de Ángulos en la pantalla
Poscondiciones	El Usuario debe ver los datos de ángulos de los dedos en la pantalla (Panel de Ángulos)	

Fuente: Elaboración propia

Tabla N° 10: Descripción de caso de uso Escuchar Teclado

Caso de Uso:	Escuchar Teclado	
Actores:	Usuario, Módulo de Captura-Procesamiento de Datos	
Descripción	Procedimiento que realiza el Módulo para producir el sonido del Teclado Virtual que será captado auditivamente por el Usuario	
Precondicion	El Módulo Audiovisual recibe datos de las teclas para producir sonido	
Flujo Principal	Paso	Acciones
	1	El Módulo Audiovisual define las notas MIDI a ser emitidas según los datos enviados del Módulo de Captura-Procesamiento de Datos
	2	El Módulo Audiovisual emite sonido de la o las notas para que el Usuario escuche
Poscondiciones	El Usuario debe escuchar el sonido de las teclas tocadas en el Escenario Virtual	

Fuente: Elaboración propia

Tabla N° 11: Descripción de caso de uso Modificar Visualización

Caso de Uso:	Modificar Visualización	
Actores:	Usuario, Módulo de Captura-Procesamiento de Datos	
Descripción	Proceso que el Módulo realiza cuando el Usuario cambia el modo de visualización (rotando, acercando, alejando) o cambia el permiso de visualización de algún elemento del Escenario Virtual representado en gráficos 3D por el Módulo Audiovisual	
Precondición		
Flujo Principal	Paso	Acciones
	1	El Usuario activa al Módulo (mediante el teclado o la interfaz gráfica) para realizar alguna modificación de visualización
	2	El Módulo de Modificación manda al Módulo de Captura – Procesamiento de Datos los datos (de rotación, acercamiento y alejamiento) para que realice las modificaciones.
	3	El Módulo de Modificación manda al Módulo de Captura – Procesamiento de Datos los datos de permiso de visualización de los elementos del Escenario Virtual, para que realice las modificaciones.
Poscondiciones	El Usuario debe poder percibir los cambios requeridos en la pantalla	

Fuente: Elaboración propia

Tabla N° 12: Descripción de caso de uso Modificar Teclado Virtual

Caso de Uso:	Modificar Teclado Virtual	
Actores:	Usuario, Módulo de Captura-Procesamiento de Datos	
Descripción	Proceso que el Módulo realiza cuando el Usuario mueve, activa o desactiva el teclado.	
Precondición		
Flujo Principal	Paso	Acciones
	1	El Usuario mueve el teclado
Flujo Secundario	2	El Módulo de Modificación manda al Módulo de Captura – Procesamiento de Datos los datos de movimiento para que realice las modificaciones en el Escenario Virtual
	2A.1	El Usuario activa el teclado
	2A.2	El Módulo de Modificación hace que el Módulo de Captura – Procesamiento de Datos cree un nuevo Teclado Virtual en el Escenario Virtual
	2B.1	El Usuario desactiva el teclado
Poscondiciones	2B.2	El Módulo de Modificación hace que el Módulo de Captura – Procesamiento de Datos elimine el teclado del Escenario Virtual
	El Usuario debe poder percibir los cambios requeridos mediante el Módulo Audiovisual	

Fuente: Elaboración propia

Tabla N° 13: Descripción de caso de uso Grabar Manos

Caso de Uso:	Grabar Manos	
Actores:	Usuario, Módulo de Captura-Procesamiento de Datos	
Descripción	Proceso que el Módulo realiza cuando el Usuario activa la opción Grabar en la interfaz gráfica, haciendo que el Módulo capture los datos del Módulo de Captura y Procesamiento de Datos almacenándolos en un archivo	
Precondición		
Flujo Principal	Paso	Acciones
	1	El Usuario activa la opción de Grabar
	2	El Módulo de Manejo de Archivos crea un archivo temporal
	3	El Módulo de Manejo de Archivos escribe los datos del Escenario Virtual del Módulo de Captura-Procesamiento de Datos
	4	Se repite el paso 3 hasta que el usuario decida dejar de grabar
	5	El Módulo de Manejo de Archivos elimina el archivo temporal
Flujo Secundario	4A.1	El Usuario elige la opción de guardar la grabación
	4A.2	El Módulo de Manejo de Archivos utiliza el caso de uso Generar Archivo
Poscondiciones	Si se sigue el Flujo Secundario, un archivo que guarda los datos de una grabación de las manos es generado.	

Fuente: Elaboración propia

Tabla N° 14: Descripción de caso de uso Generar Archivo

Caso de Uso:	Generar Archivo	
Actores:	Usuario	
Descripción	Procedimiento que el Módulo realiza cuando el usuario decide guardar el archivo con los datos previamente grabados	
Precondición	El Usuario elige la opción de guardar la grabación	
Flujo Principal	Paso	Acciones
	1	El Módulo de Manejo de Archivos pregunta al Usuario dónde y con qué nombre guardar el archivo de la grabación
	2	El Usuario elige la ubicación y el nombre de archivo
	3	El Módulo de Manejo de Archivos copia el archivo temporal de la grabación a la ubicación elegida por el Usuario y le asigna el nombre elegido por el Usuario
Poscondiciones	Existencia de un archivo que contiene los datos de la grabación generado en la ubicación y el nombre elegidos por el Usuario.	

Fuente: Elaboración propia

Tabla N° 15: Descripción de caso de uso Cargar Archivo

Caso de Uso:	Cargar Archivo	
Actores:	Usuario	
Descripción	Procedimiento que realiza el Módulo para proceder a la lectura de un archivo cuando el Usuario activa la opción de Cargar Archivo	
Precondición		
Flujo Principal	Paso	Acciones
	1	El Usuario activa la opción de Cargar Archivo
	2	El Módulo de Manejo de Archivos pregunta al Usuario qué archivo debe cargar al Módulo
	3	El Usuario elige el archivo a ser cargado
	4	El Módulo de Manejo de Archivos mantiene cargado el archivo seleccionado por el Usuario hasta que éste (el Usuario) decida quitarlo
Flujo Secundario	4A.1	El Usuario elige reproducir el archivo
	4A.2	El Módulo de Manejo de Archivos utiliza el caso de uso Reproducir Archivo
	4B.1	El Usuario elige generar gráficos estadísticos en base a los datos del archivo cargado
	4B.2	El Módulo de Manejo de Archivos utiliza el caso de uso Generar Gráficos Estadísticos
	4C.1	El Usuario elige exportar datos a un archivo Excel
	4C.2	El Módulo de Manejo de Archivos utiliza el caso de uso Exportar a Excel
Poscondiciones	El Usuario puede realizar las acciones de reproducción de archivo, generación de gráficos estadísticos y exportación a archivo Excel con éxito mientras el archivo está cargado	

Fuente: Elaboración propia

Tabla N° 16: Descripción de caso de uso Reproducir Archivo

Caso de Uso:	Reproducir Archivo	
Actores:	Usuario, Módulo de Captura-Procesamiento de Datos	
Descripción	Procedimiento que el Módulo realiza cuando el Usuario activa la opción de Reproducir. Mientras se reproduzca el archivo, se mandan los datos al Módulo de Captura y Procesamiento de Datos lo que hace que el Módulo Audiovisual proceda a mostrar al Usuario la representación de los datos de las manos almacenadas en el archivo cargado.	
Precondición	Un archivo que contenga los datos de las manos debe estar cargado al sistema	
Flujo Principal	Paso	Acciones
	1	El Usuario elige la opción de reproducir el archivo cargado
	2	El Módulo de Manejo de Archivos lee los datos de las manos almacenados en el archivo
	3	El Módulo de Manejo de Archivos manda los datos de las manos al Módulo de Captura-Procesamiento de Datos para que los copie en el Escenario Virtual
	4	Se repite desde 2 hasta que el Usuario decida dejar de reproducir el archivo o termine la lectura de todos los datos del archivo
Flujo Secundario	3A.1	El Usuario modifica la reproducción normal
	3A.2	El Módulo de Manejo de Archivos utiliza el caso de uso Modificar Reproducción
Poscondiciones	El Usuario pudo observar la representación de las manos mediante el Módulo Audiovisual que se encontraban almacenados en el archivo cargado	

Fuente: Elaboración propia

Tabla N° 17: Descripción de caso de uso Modificar Reproducción

Caso de Uso:	Modificar Reproducción	
Actores:	Usuario, Módulo de Captura-Procesamiento de Datos	
Descripción	Procedimiento que realiza el Módulo cuando el Usuario activa una modificación a la reproducción. Las modificaciones pueden ser: Velocidadx2, Velocidad/2, Pausar y Parar	
Precondición	El Usuario modifica la reproducción	
Flujo Principal	Paso	Acciones
	1	El Módulo de Manejo de Archivos modifica el tiempo de lectura del archivo o lo reinicia según la modificación requerida por el Usuario
	2	El Módulo de Manejo de Archivos informa al Módulo de Captura –Procesamiento de Datos sobre los cambios realizados
	3	El Módulo de Captura-Procesamiento de Datos recibe la información sobre los cambios realizados (lo que hace que el Módulo Audiovisual muestre los cambios)
Poscondiciones	El Usuario pudo observar las modificaciones requeridas mediante el Módulo Audiovisual	

Fuente: Elaboración propia

Tabla N° 18: Descripción de caso de uso Generar Gráfico Estadístico

Caso de Uso:	Generar Gráfico Estadístico	
Actores:	Usuario	
Descripción	Procedimiento que realiza el Módulo cuando el Usuario activa alguno de las 4 opciones para generar gráficos estadísticos: X vs Tiempo, Y vs Tiempo, Z vs Tiempo y Velocidad Y vs Tiempo. Estos gráficos estadísticos estarán generados a partir de los datos de las posiciones y la velocidad de los elementos de la mano (dedos y muñeca) respecto al tiempo obtenidos del archivo cargado	
Precondición	Un archivo que contenga los datos de las manos debe estar cargado al sistema	
Flujo Principal	Paso	Acciones
	1	El Usuario elige una de las 4 opciones (X vs Tiempo, Y vs Tiempo, Z vs Tiempo y Velocidad Y vs Tiempo) para generar gráficos estadísticos
	2	El Módulo de Manejo de Archivos lee los datos de las manos almacenados en el archivo según la opción elegida por el Usuario
	3	El Módulo de Manejo de Archivos crea el gráfico en base a la lectura de los datos leídos
	4	El Módulo de Manejo de Archivos muestra el gráfico generado al Usuario
	5	El Usuario elige salir de la ventana de visualización del gráfico estadístico
Flujo Secundario	5A.1	El Usuario elige la opción de guardar gráfico estadístico como imagen
	5A.2	El Módulo de Manejo de Archivos utiliza el caso de uso Guardar Gráfico
Poscondiciones	El Usuario pudo observar el gráfico estadístico requerido	

Fuente: Elaboración propia

Tabla N° 19: Descripción de caso de uso Guardar Gráfico

Caso de Uso:	Guardar Gráfico	
Actores:	Usuario	
Descripción	Procedimiento que el Módulo realiza para guardar los gráficos estadísticos generados en formato de imagen cuando el Usuario selecciona la opción de Guardar Imagen.	
Precondición	El Usuario elige la opción de guardar gráfico estadístico como imagen	
Flujo Principal	Paso	Acciones
	1	El Módulo de Manejo de Archivos pregunta al Usuario dónde y con qué nombre guardar el archivo de la imagen del gráfico
	2	El Usuario elige la ubicación y el nombre de archivo
	3	El Módulo de Manejo de Archivos crea el archivo de imagen y lo guarda en el lugar y con el nombre asignado por el Usuario
Poscondiciones	Existencia de un archivo de imagen que es generado con el nombre y la ubicación elegidos por el Usuario	

Fuente: Elaboración propia

Tabla N° 20: Descripción de caso de uso Exportar a Excel

Caso de Uso:	Exportar a Excel	
Actores:	Usuario	
Descripción	Procedimiento que el Módulo realiza para generar un archivo Excel con las datos leídos del archivo cargado cuando el Usuario mediante la interfaz gráfica, seleccione la opción de Exportar Archivo.	
Precondición	Un archivo que contenga los datos de las manos debe estar cargado al sistema	
Flujo Principal	Paso	Acciones
	1	El Usuario elige la opción de Exportar a Excel
	2	El Módulo muestra las opciones de exportación de datos: el tipo de dato a ser exportado (posiciones y velocidad) de qué elementos (dedos y muñecas)
	3	El Usuario elige la configuración de exportación requerida y selecciona la opción de exportar en archivo Excel
	4	El Módulo de Manejo de Archivos pregunta al Usuario dónde y con qué nombre guardar el archivo de la imagen del gráfico
	5	El Usuario elige la ubicación y el nombre de archivo
	6	El Módulo de Manejo de Archivos crea el archivo de Excel
	7	El Módulo de Manejo de Archivos lee la información del archivo cargado y escribe en el nuevo archivo Excel según la configuración elegida por el Usuario
Poscondiciones	Existencia de un archivo Excel que es generado con el nombre y la ubicación elegidos por el Usuario. Éste archivo contiene los datos sobre las posiciones o velocidades de los dedos o las muñecas requeridas por el Usuario.	

Fuente: Elaboración propia

Anexo 4: Manual de Usuario

Manual de Usuario

**SISTEMA COMPUTARIZADO PARA EL ESTUDIO DEL
DESARROLLO DE LA TÉCNICA BÁSICA PIANÍSTICA**

CONTENIDO

1	Descripción general	1
2	Requisitos y recomendaciones	1
3	Iniciar el Software	1
4	Configuración inicial del Teclado Virtual.....	3
5	Opciones de movimiento del Teclado Virtual.....	4
6	Eliminar/Crear el Teclado Virtual.....	4
7	Cambiar el permiso de dibujo de los dedos, muñecas y teclado	5
8	Cambio de posición de la visualización: Rotación y Zoom.....	6
9	Grabación en archivo <i>.ptf</i>	8
10	Cargado de archivo <i>.ptf</i> al sistema	9
11	Reproducción de archivo <i>.ptf</i>	9
12	Generación, manejo de imagen y exportación de gráficos estadísticos	10
13	Exportación de datos de un archivo <i>.ptf</i> a archivo Excel	11
14	Uso de estructura física anexa	12

INDICE DE ILUSTRACIONES

Ilustración N° 1: Pantalla principal del sistema	2
Ilustración N° 2: Vista externa de la pantalla, manos y Leap Motion	2
Ilustración N° 3: Panel Control de Teclado	3
Ilustración N° 4: Vista Perspectiva del teclado virtual en tres posiciones mientras se mueve hacia la derecha.....	4
Ilustración N° 5: Panel Control de Imagen	5
Ilustración N° 6: Vista Perspectiva con la representación gráfica del teclado, dedos y muñecas	5
Ilustración N° 7: Panel Control de Imagen con dedos 2 y 4 de la mano derecha y teclado desactivados.....	6
Ilustración N° 8: Vista perspectiva con gráficos sin los dedos 2 y 4 de la mano derecha y sin teclado	6
Ilustración N° 9: Rotación de imagen a derecha e izquierda visto en perspectiva	7
Ilustración N° 10: Aumento y disminución de zoom visto en perspectiva	7
Ilustración N° 11: Panel Manejo de Archivos	8
Ilustración N° 12: Ventana de guardado de archivo .ptf.....	8
Ilustración N° 13: Ventana de cargado de archivo .ptf.....	9
Ilustración N° 14: Panel Gráficos Estadísticos	10
Ilustración N° 15: Cuadro Estadístico Y vs Tiempo.....	11
Ilustración N° 16: Ventana de Exportación de Datos a Excel	12
Ilustración N° 17: Partes metálicas de la estructura física anexa.....	12
Ilustración N° 18: Vara vertical y horizontal ensambladas.....	13
Ilustración N° 19: Estructura metálica ensamblada	13
Ilustración N° 20: Estructura metálica ajustada al borde de una mesa	14

1 Descripción general

Es un sistema que sirve como una herramienta en el estudio del desarrollo de la técnica pianística básica.

El sistema permite saber la posición y velocidad de los dedos y las muñecas, genera un teclado virtual de dos octavas que puede ser tocado según la posición que se encuentran los dedos, produce el sonido del teclado virtual, muestra representaciones gráficas sencillas con Mallas Poligonales de los componentes (dedos, muñecas y teclado) en la pantalla, realiza grabaciones del movimiento de los dedos y las muñecas, genera archivos con los datos que pueden ser posteriormente reproducidos en el sistema, genera gráficos estadísticos y también puede exportar los datos obtenidos a archivos Excel.

2 Requisitos y recomendaciones

Para que el sistema funcione, es necesario:

- Un dispositivo Leap Motion y su cable de conexión USB
- Java 7 o superior instalado
- Carpeta de proyecto

Se recomienda el uso de la Estructura Física Anexa (ver Uso de la Estructura Física Anexa).

3 Iniciar el Software

Para iniciar el software debe abrir la Terminal o cmd (en Windows) y escribir:

```
java -jar dir_carpeta_de_proyecto/SistemaPianoTecnica.jar
```

Donde `dir_carpeta_de_proyecto` es la dirección de la carpeta del proyecto y, a continuación debe pulsar la tecla Enter.

El sistema comenzará a funcionar y le mostrará la pantalla principal, tal como se puede observar se puede observar en la Ilustración Nº 1.

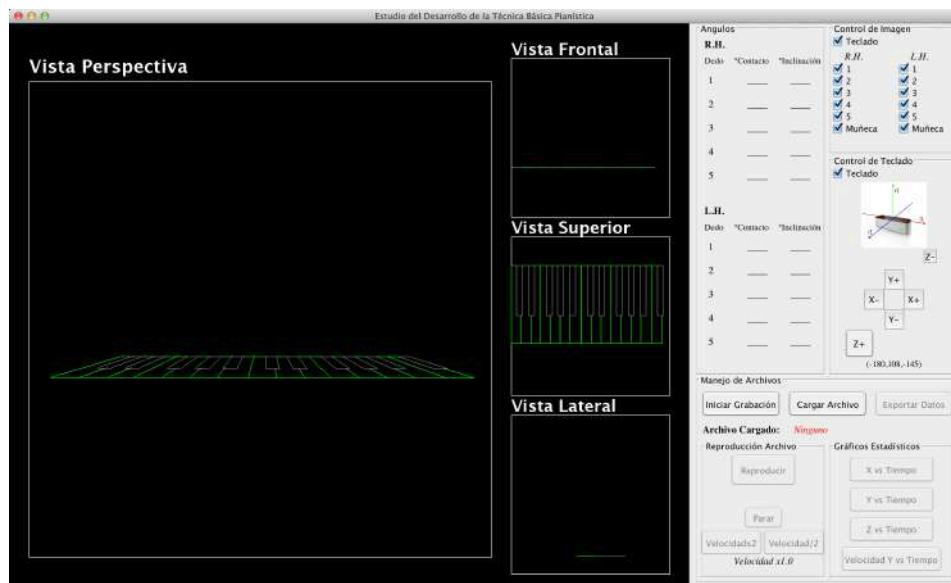


Ilustración N° 1: Pantalla principal del sistema

Conecte el dispositivo Leap Motion a la computadora mediante su cables USB. Si el dispositivo Leap Motion funciona correctamente debe reconocer sus manos y, el sistema debe representarlos gráficamente, como se puede observar en la Ilustración N° 2.

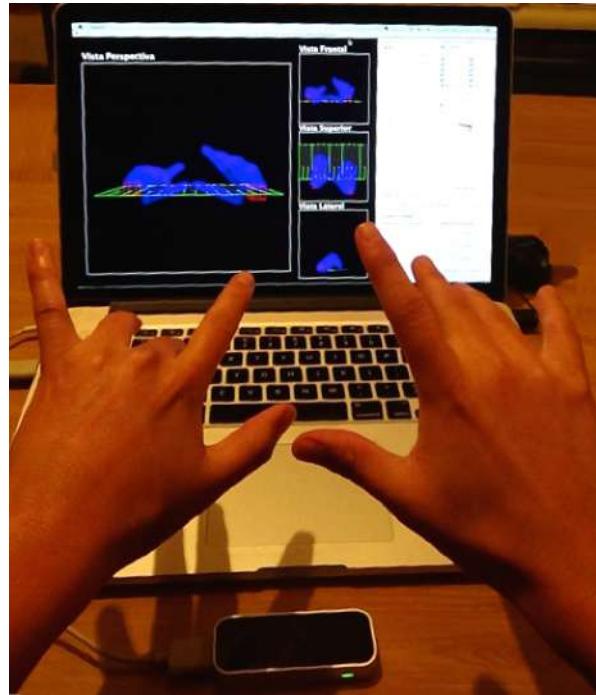


Ilustración N° 2: Vista externa de la pantalla, manos y Leap Motion

4 Configuración inicial del Teclado Virtual

La configuración inicial del sistema tiene incluido al Teclado Virtual. Éste puede ser tocado con los dedos moviendo a la posición de las teclas representadas gráficamente en la pantalla de su computadora, según la captura de posiciones del dispositivo Leap Motion.

Si la punta de un dedo llega a la posición de una tecla perteneciente al Teclado Virtual, el sistema producirá el sonido de la nota de la respectiva tecla (según la afinación promediada internacional).

La posición (coordenadas) del Teclado Virtual puede observarse en la parte inferior central del Panel de Control de Teclado. La posición inicial por defecto es (-180, 108, -45) donde cada uno de los componentes representa la posición (en milímetros) según al eje x, y y z respectivamente, siendo el origen el dispositivo Leap Motion.

En la Ilustración N° 3 se puede observar el Panel Control de Teclado que, también contiene un gráfico donde se indica la posición de los ejes respecto al dispositivo Leap Motion.

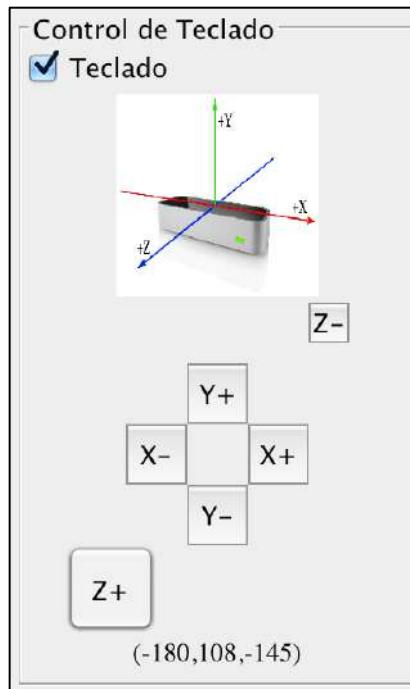


Ilustración N° 3: Panel Control de Teclado

5 Opciones de movimiento del Teclado Virtual

Para mover el Teclado Virtual debe pulsar los botones de dirección que se encuentran en el Panel Control de Teclado de la pantalla.

- Para mover el teclado a la derecha, haga clic en el botón x+
- Para mover el teclado a la izquierda, haga clic en el botón x-
- Para mover el teclado hacia delante, haga clic en el botón z+
- Para mover el teclado hacia atrás, haga clic en el botón z-
- Para mover el teclado hacia arriba, haga clic en el botón y+
- Para mover el teclado hacia abajo, haga clic en el botón y-

En la Ilustración N° 4 se puede observar el movimiento (hacia el lado derecho) del teclado en el cuadro de Vista en Perspectiva.

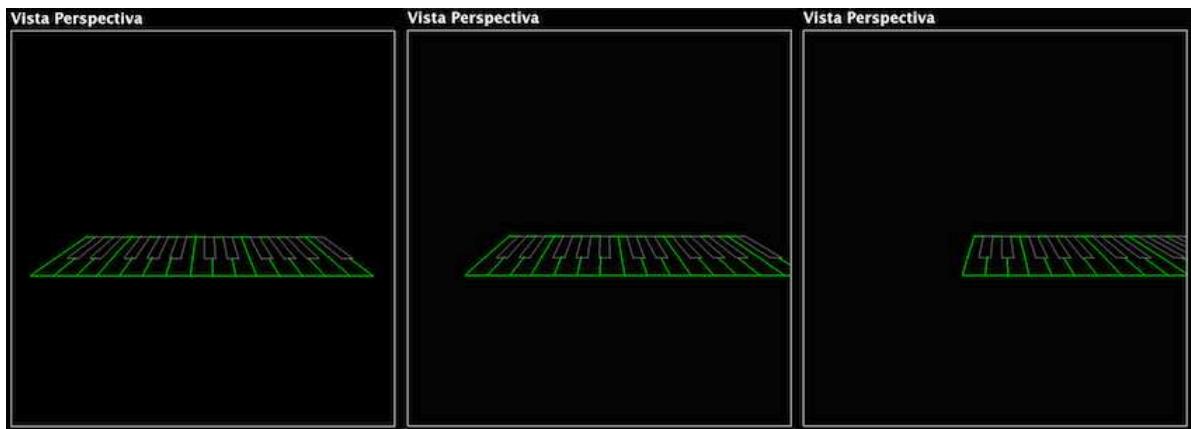


Ilustración N° 4: Vista Perspectiva del teclado virtual en tres posiciones mientras se mueve hacia la derecha

6 Eliminar/Crear el Teclado Virtual

Para eliminar el Teclado Virtual, desactive el *checkbox* Teclado y, para crear el Teclado Virtual (con su configuración inicial) active el *checkbox* Teclado del Panel Control de Teclado (Ver Ilustración N° 3).

7 Cambiar el permiso de dibujo de los dedos, muñecas y teclado

Para hacer que un componente (dedos, muñecas o teclado) sea representado o no sea representado gráficamente en la pantalla, debe activar o desactivar respectivamente su *checkbox* del Panel Control de Imagen, el cual se puede observar en la Ilustración N° 5.



Ilustración N° 5: Panel Control de Imagen

En la Ilustración N° 6 se puede observar al cuadro de Vista Perspectiva con el dibujo de ambas manos de la pantalla. En la Ilustración N° 7 se puede observar al Panel Control de Imagen con los permisos de dibujo de los dedos 2 y 4 de la mano derecha y el teclado desactivados, lo cual se puede observar en la Ilustración N° 8.

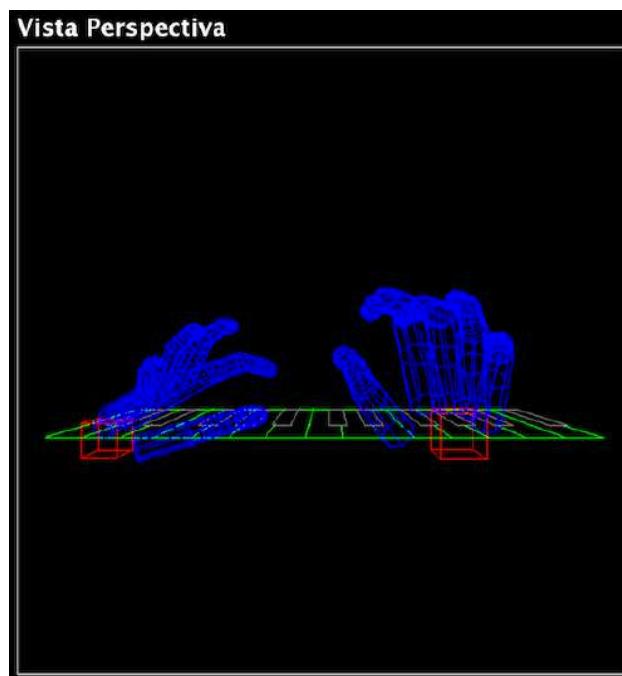


Ilustración N° 6: Vista Perspectiva con la representación gráfica del teclado, dedos y muñecas



Ilustración N° 7: Panel Control de Imagen con dedos 2 y 4 de la mano derecha y teclado desactivados

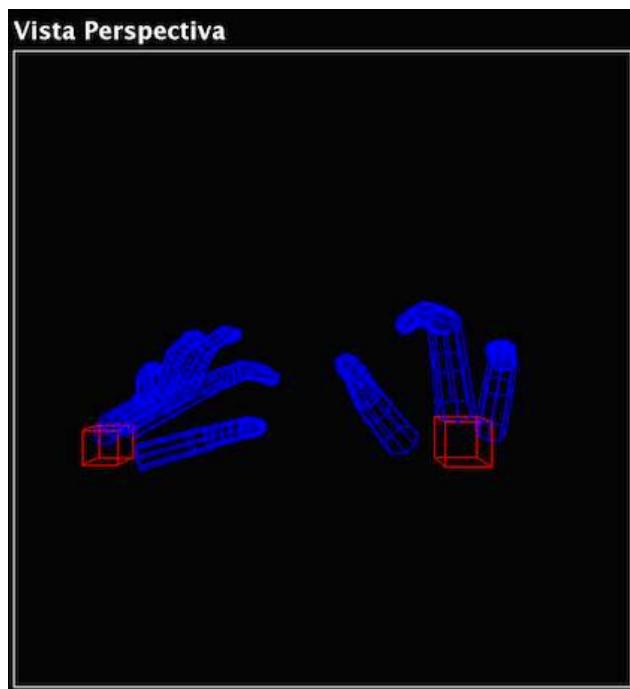


Ilustración N° 8: Vista perspectiva con gráficos sin los dedos 2 y 4 de la mano derecha y sin teclado

8 Cambio de posición de la visualización: Rotación y Zoom.

Para realizar un movimiento de rotación de la visualización de las imágenes, pulse la tecla de dirección derecha, izquierda, arriba y abajo del teclado según lo requiera. Mientras mantenga presionada la tecla la imagen rotará en la dirección seleccionada.

En la Ilustración N° 9 se puede observar la rotación de aproximadamente 45° de la imagen mostrada en la Ilustración N° 6 hacia el lado derecho e izquierdo.

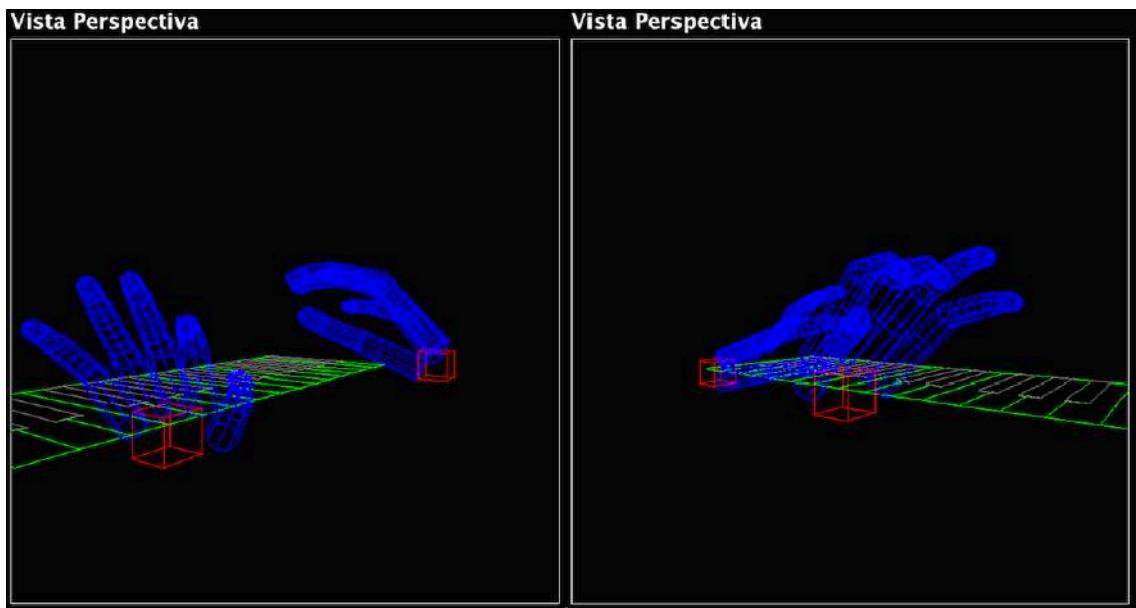


Ilustración N° 9: Rotación de imagen a derecha e izquierda visto en perspectiva

Para aumentar el zoom de la imagen, presione la tecla W y, para disminuirlo presione la tecla S.

En la Ilustración N° 10 se puede observar las imágenes del aumento y la disminución de zoom de la imagen de la Ilustración N° 6.

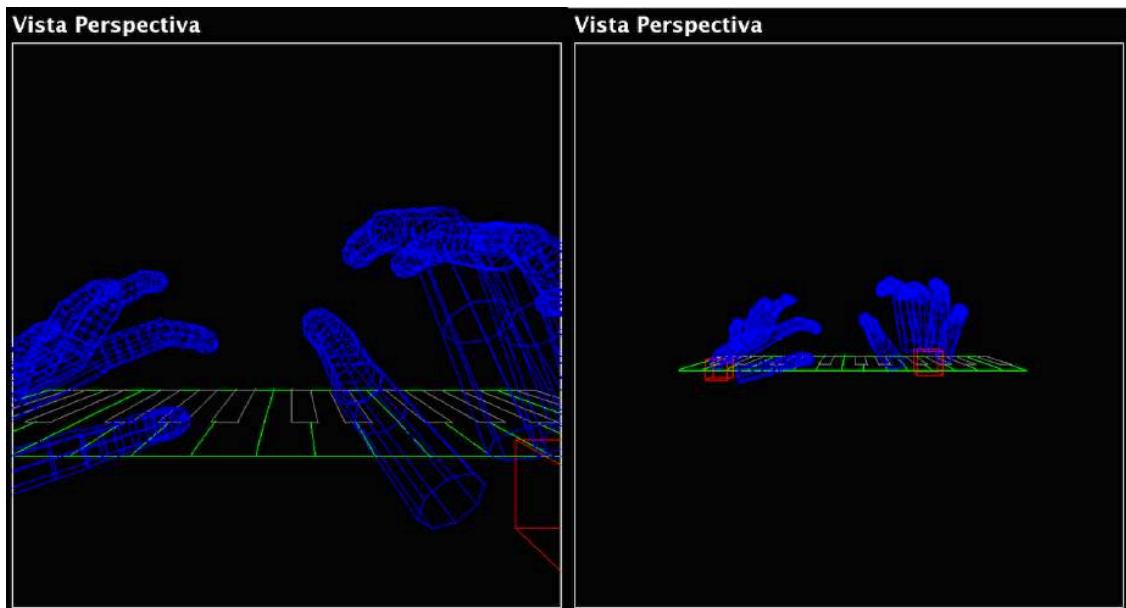


Ilustración N° 10: Aumento y disminución de zoom visto en perspectiva

Para restaurar los valores predeterminados de las imágenes (anular rotación y zoom) presione la tecla R.

9 Grabación en archivo .ptf

Para grabar una secuencia de movimiento de dedos y muñecas, haga clic en el botón “Iniciar Grabación” del Panel Manejo de Archivos (ver Ilustración N° 11)



Ilustración N° 11: Panel Manejo de Archivos

Para parar la grabación, haga clic en el botón “Parar Grabación” del Panel Manejo de Archivos. Posteriormente el sistema le mostrará una ventana (ver la Ilustración N° 12) para elegir la dirección de guardado de archivo .ptf que contiene la grabación.

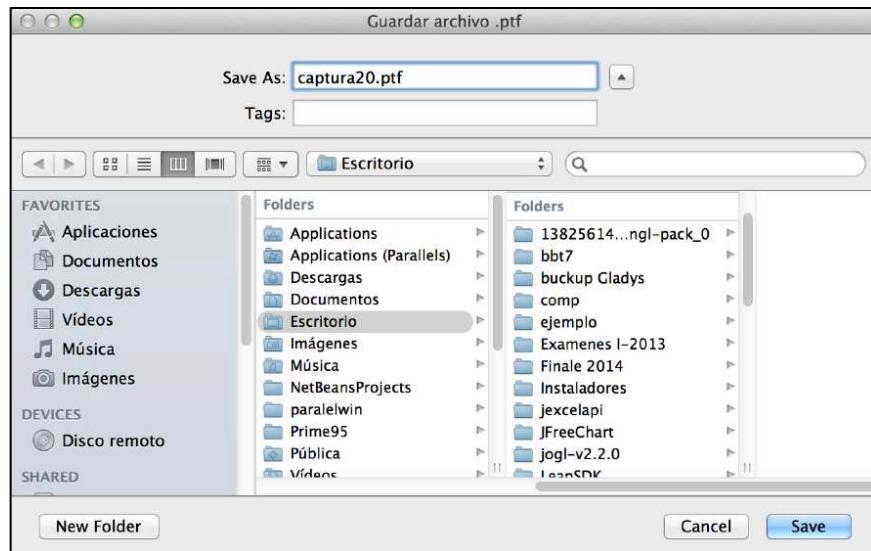


Ilustración N° 12: Ventana de guardado de archivo .ptf

10 Cargado de archivo .ptf al sistema

Para poder reproducir un archivo, generar gráficos estadísticos y exportar datos a Excel se necesita realizar el cargado de archivo .ptf al sistema.

Para cargar el archivo haga clic en el botón Cargar Archivo del panel Manejo de Archivos (ver Ilustración N° 11). Posteriormente el sistema le mostrará una ventana (ver la Ilustración N° 13) para que elija la dirección del archivo .ptf que desea cargar al sistema.

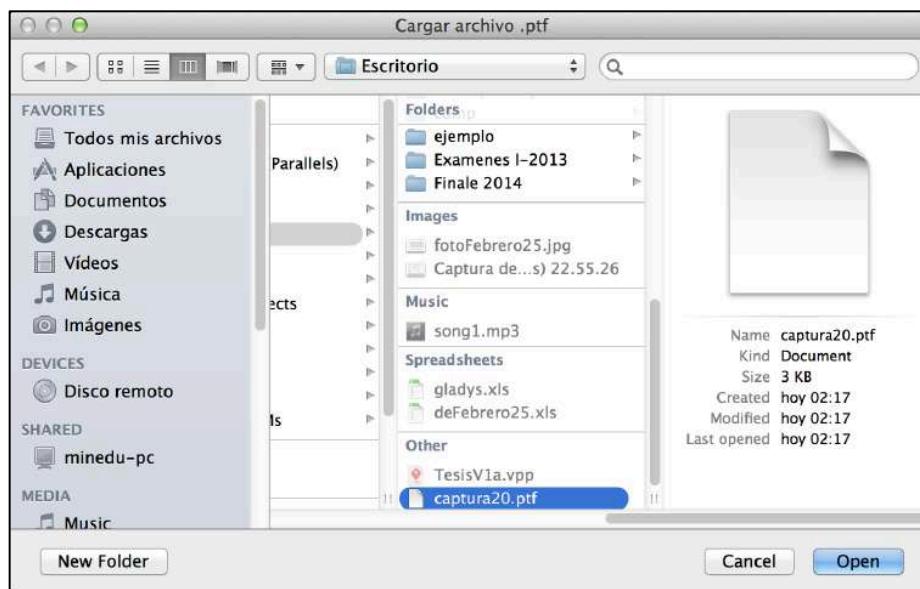


Ilustración N° 13: Ventana de cargado de archivo .ptf

11 Reproducción de archivo .ptf

Para reproducir un archivo .ptf que contiene la grabación de secuencia de movimientos de los dedos y la muñeca, debe cargar el archivo al sistema. (ver Cargado de archivo .ptf al sistema).

Para reproducir un archivo .ptf (previamente cargado al sistema) haga clic en el botón “Reproducir” (ver Ilustración N° 11) del Panel Manejo de Archivos. El sistema comenzará a reproducir el archivo, representando la posición de los dedos y la muñeca gráficamente tal y como se estaría capturando el movimiento mediante el dispositivo Leap Motion en el momento.

Para pausar la reproducción haga clic en el botón “Pausar” del Panel Manejo de Archivos.

Para parar la reproducción haga clic en el botón “Parar” del Panel Manejo de Archivos.

Para duplicar la velocidad de reproducción haga clic en el botón “Velocidad x2” del Panel Manejo de Archivos. El límite de velocidad es 16 veces la velocidad original.

Para reducir a la mitad la velocidad de reproducción haga clic en el botón “Velocidad/2” del Panel Manejo de Archivos.

12 Generación, manejo de imagen y exportación de gráficos estadísticos

Para poder generar gráficos estadísticos debe cargar el archivo *.ptf* al sistema. (ver Cargado de archivo *.ptf* al sistema).

Para generar un gráfico estadístico, haga clic uno de los botones del Panel Gráficos Estadísticos (ver Ilustración N° 14) según los datos que quiera utilizar para la construcción del gráfico estadístico.



Ilustración N° 14: Panel Gráficos Estadísticos

Una vez que se genere el gráfico estadístico (ejemplo en Ilustración N° 15), puede realizar la selección de los componentes que desea que sean vistos gráficamente activando o desactivando los *checkboxes* del lado izquierdo del panel. En la parte inferior de la ventana se indica los colores que representan a los componentes.

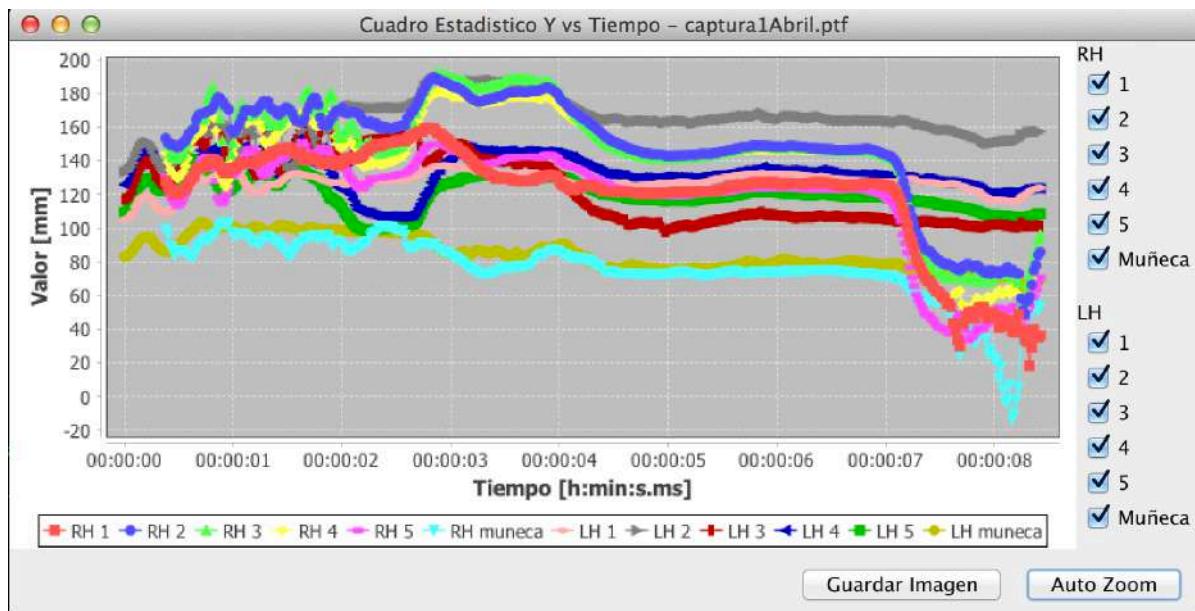


Ilustración N° 15: Cuadro Estadístico Y vs Tiempo

Para realizar un acercamiento a alguna sección del panel, utilice el mouse para seleccionar el área que desea visualizar. El sistema realizará un zoom a la sección seleccionada, el cual puede ser revertido haciendo clic en el botón “Auto Zoom”.

Para exportar el panel como un archivo de imagen en formato JPG haga clic en el botón “Guardar Imagen”. El sistema le mostrará una ventana para que elija la dirección para guardar del archivo JPG.

13 Exportación de datos de un archivo .ptf a archivo Excel

Para poder exportar datos a un archivo Excel debe cargar un archivo *.ptf* al sistema. (ver Cargado de archivo *.ptf* al sistema) y, posteriormente, hacer clic en botón Exportar a Excel (Ver Ilustración N° 11).

El sistema le mostrará la Ventana de Exportación de Datos a Excel (ver Ilustración N° 16) donde puede elegir la resolución de exportación de datos (intervalo de datos desde 20 milisegundos hasta 1 segundo), componentes y tipos de datos de éstos a ser exportados.



Ilustración N° 16: Ventana de Exportación de Datos a Excel

Para confirmar la exportación, haga clic en el botón Exportar a Excel. El sistema le mostrará una ventana para elegir la dirección y el nombre del archivo Excel, para que éste sea creado con los datos elegidos del archivo .ptf cargado.

14 Uso de estructura física anexa

La estructura está compuesta de 3 partes metálicas: el marco, la vara vertical y la vara horizontal (ver Ilustración N° 17) además de una superficie de vidrio.



Ilustración N° 17: Partes metálicas de la estructura física anexa

La estructura sirve para dar mayor comodidad en la captura de datos de las manos mediante el Leap Motion, brindando una superficie de apoyo de las manos sin interferir con el funcionamiento regular de todo el sistema.

Para armar la estructura, ajustar la vara vertical a la horizontal tal como se muestra en la Ilustración N° 18 y asegurar el tornillo del extremo de la vara horizontal.

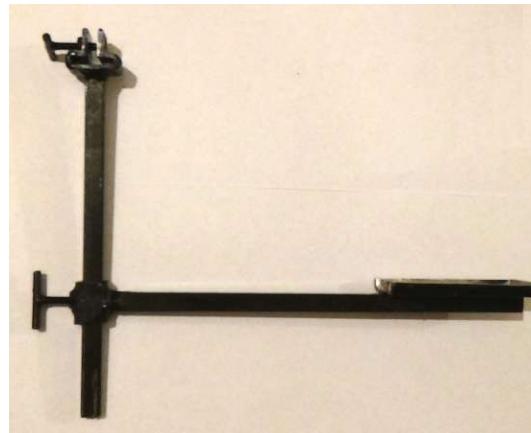


Ilustración N° 18: Vara vertical y horizontal ensambladas

Posteriormente ensamblar la parte superior de la vara vertical (en forma de T invertida) en el lado izquierdo del marco tal como se muestra en la Ilustración N° 19 y asegurar el tornillo del extremo superior de la vara vertical



Ilustración N° 19: Estructura metálica ensamblada

Una vez que tenga armada la estructura metálica, puede ajustar el borde posterior del marco a la superficie de una mesa o escritorio de hasta 4 centímetros de grosor gracias a las prensas del marco tal como se muestra en la Ilustración N° 20.



Ilustración N° 20: Estructura metálica ajustada al borde de una mesa

Cuando la estructura metálica está asegurada en una superficie, proceda a colocar el vidrio en el marco y realice los ajustes de posición que requieran las varas vertical y horizontal. Posteriormente coloque el dispositivo Leap Motion en el extremo derecho de la vara horizontal (en la superficie rectangular de aluminio) que sirve como soporte del dispositivo y, en caso necesitar ajustes de posición, proceda a regular las varas vertical y horizontal según sus requerimientos.

En la Ilustración N° 21 se puede observar la estructura física anexa armada y siendo utilizada junto al sistema

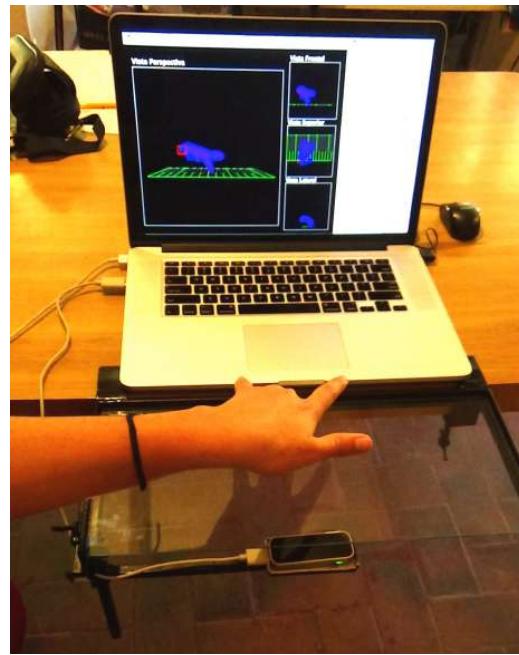


Ilustración N° 21: Estructura física anexa siendo utilizada

Se recomienda que el dispositivo Leap Motion esté a una distancia de aproximadamente 8 centímetros de la superficie de vidrio.

Una vez que tenga regulada la posición del Leap Motion, puede regular la posición del Teclado Virtual en el Panel Control de Teclado (ver Opciones de movimiento del Teclado Virtual)

Debe tomar en cuenta que tanto el vidrio como el Leap Motion pueden ensuciarse y en consecuencia puede disminuir la efectividad de captura de datos de las manos.

Anexo 5: Explicación de Diagrama de Clases

**EXPLICACIÓN DE DIAGRAMA DE CLASES DEL PROTIPO
DE SISTEMA COMPUTARIZADO PARA EL ESTUDIO
DEL DESARROLLO DE LA TÉCNICA BÁSICA
PIANÍSTICA**

CONTENIDO

I Captura de datos	1
II Representación Virtual	2
2.1 Representación virtual de la manos, dedos y huesos	3
2.2 Representación virtual del teclado	8
III Producción de Sonido.....	15
3.1 Funcionamiento de GeneradorMIDI.....	15
3.2 Funcionamiento de Nota	16
3.3 Activación de sonido mediante dedo	17
IV Producción Visual.....	22
4.1 Composición general y funcionamiento de Pantalla.....	22
4.2 Composición general y funcionamiento de PanelAngulos	25
4.3 Composición general de PanelEscenario	27
4.4 Conversión del sistema de coordenadas de representación virtual al visual.....	29
4.5 Creación de DibujoHueso	30
4.6 Creación de DibujoMuneca	37
4.7 Creación de DibujoTecla	38
4.8 Funcionamiento de PanelEscenario	39
4.9 Composición y funcionamiento de PanelControlTeclado	40
4.10 Composición y funcionamiento de PanelControlVisual	41
4.11 Composición de PanelArchivo	42
4.12 Funcionamiento de btGrabacion	43
4.13 Funcionamiento de btCargarArchivo.....	47
4.14 Funcionamiento de btExportarDatos.....	49
4.15 Composición y funcionamiento de FrameExportarDatos	49
4.16 Composición y funcionamiento de PanelEstadistica	54
4.17 Composición y funcionamiento de FrameEstadistica	55
4.18 Composición y funcionamiento de PanelReproduccion.....	57
V Modificación Virtual.....	62
5.1 Control de dibujo de los componentes de Escenario Virtual	62

5.2	Movimiento, eliminación y creación de Teclado	63
5.3	Movimiento de cámara de visualización	65
VI	Manejo de Archivos.....	70
6.1	Grabación de archivos	70
6.2	Reproducción de archivos	71
6.3	Generación de cuadros estadísticos	73
6.4	Exportación de datos a archivo Excel	74
VII	Coordinación General.....	76

INDICE DE FIGURAS

Figura N° 1: Representación de eje de coordenadas en la Representación Virtual	2
Figura N° 2: Imagen de mano derecha e izquierda y sus partes a ser representadas	4
Figura N° 3: Anatomía de la mano	5
Figura N° 4: Tipos de hueso según representación en clase Hueso.....	6
Figura N° 5: Representación gráfica de los vectores base de los huesos.....	7
Figura N° 6: Estructura de intervalo de séptima (<i>do a si</i>) del teclado	10
Figura N° 7: Representación de las teclas blanca y negra, compuesta de vértices y aristas	11
Figura N° 8: Numeración cromática y numeración natural de las teclas.....	14
Figura N° 9: Representación gráfica de la estructura (valores de los vértices) de la tecla <i>sol1</i>	14
Figura N° 10: Representación gráfica de un dedo tocando tecla – Vista lateral.....	17
Figura N° 11: Representación gráfica de un dedo tocando tecla – Vista superior	18
Figura N° 12: Representación gráfica de un dedo tocando tecla – Vista diagonal.....	18
Figura N° 13: Representación gráfica de espacio de tolerancia – Tecla Negra.....	19
Figura N° 14: Representación gráfica de espacio de tolerancia – Tecla Blanca	20
Figura N° 15: Composición gráfica de Pantalla	23
Figura N° 16: Mensaje de confirmación de salida de la aplicación.....	24
Figura N° 17: Imagen de PanelAngulos mostrando ángulos de ambas manos	26
Figura N° 18: Imagen de PanelAngulos mostrando ángulos de la mano derecha.....	27
Figura N° 19: Composición gráfica de PanelEscenario.....	28
Figura N° 20: Sistema de Coordenadas Visual y de Sistema de Coordenadas Representación Virtual	29
Figura N° 21: Traslación de matriz base a posición de inicio de Hueso – Definición de primer vértice	31

Figura Nº 22: Proceso de definición de vértices desde Vértice 2 a Vértice 7	32
Figura Nº 23: Fin de primer ciclo y comienzo de segundo ciclo de definición de vértices	32
Figura Nº 24: Representación de vértices de DibujoHueso	33
Figura Nº 25: Representación de vértices y aristas de octógonos de DibujoHueso	33
Figura Nº 26: Representación de todos los vértices y aristas de DibujoHueso	34
Figura Nº 27: Dibujo que representa a un hueso proximal mediante mallas poligonales ..	34
Figura Nº 28: Imagen de paraboloide en la aplicación <i>Grapher</i>	35
Figura Nº 29: Esquema de representación del hueso distal en 2D	35
Figura Nº 30: Dibujo que representa a un hueso distal mediante mallas poligonales	36
Figura Nº 31: Dibujo que representa a un dedo índice mediante mallas poligonales	37
Figura Nº 32: Representación de los dedos de la mano izquierda en seis posiciones	37
Figura Nº 33: Representación gráfica de la muñeca	38
Figura Nº 34: Representación gráfica de un teclado	39
Figura Nº 35: Imagen de PanelControlTeclado	41
Figura Nº 36: Imagen de PanelControlVisual	41
Figura Nº 37: Composición gráfica de PanelArchivo	43
Figura Nº 38: Imagen de PanelArchivo con la configuración de grabación	44
Figura Nº 39: Imagen de PanelArchivo con la configuración por defecto	44
Figura Nº 40: Imagen del FileDialog utilizado en método guardarGrabacionFileDialog de la clase PanelArchivo	45
Figura Nº 41: Imagen de MessageDialog utilizado en método mmArchivoGuardado de la clase PanelArchivo	46
Figura Nº 42: Imagen de OptionDialog utilizado en método mmArchivoGuardado de la clase PanelArchivo	46

Figura Nº 43: Imagen de MessageDialog utilizado en método mmExtensionRechazada de la clase PanelArchivo.....	47
Figura Nº 44: Imagen de PanelArchivo con la configuración de archivo cargado.....	48
Figura Nº 45: Imagen del FileDialog utilizado en método cargarArchivoFileDialog de la clase PanelArchivo	48
Figura Nº 46: Imagen de MessageDialog utilizado en método mmArchivoCargado de la clase PanelArchivo	49
Figura Nº 47: Imagen de FrameExportarDatos.....	51
Figura Nº 48: Imagen del FileDialog utilizado en método guardarArchivoExcelFileDialog de la clase FrameExportarDatos.....	51
Figura Nº 49: Imagen de MessageDialog utilizado en método mmArchivoEscrito de la clase FrameExportarDatos	52
Figura Nº 50: Imagen de OptionDialog utilizado en método mmSobrescribirArchivo de la clase FrameExportarDatos	52
Figura Nº 51: Imagen de MessageDialog utilizado en método mmExtensionRechazada de la clase FrameExportarDatos	53
Figura Nº 52: Imagen de MessageDialog utilizado en método mmErrorLeyendoArchivo de la clase FrameExportarDatos.....	53
Figura Nº 53: Imagen de PanelEstadistica	54
Figura Nº 54: Imagen de FrameEstadistica	56
Figura Nº 55: Imagen de PanelReproduccion con la configuración de reproducción	59
Figura Nº 56: Imagen de PanelReproduccion con la configuración de pausa	59
Figura Nº 57: Imagen de PanelReproduccion con la configuración por defecto	60
Figura Nº 58: Imagen de MessageDialog utilizado en método mmReproduccionTerminada de la clase PanelReproduccion.....	60

Figura N° 59: Imagen de MessageDialog utilizado en método mmErrorReproduccion de la clase PanelReproduccion	61
Figura N° 60: Imagen de PanelControlVisual con algunos componentes deshabilitados	62
Figura N° 61: Imagen de PanelEscenario con algunos componentes deshabilitados ..	63
Figura N° 62: Vista Perspectiva PanelEscenario con teclado virtual en tres posiciones mientras se mueve en el eje x en sentido positivo	64
Figura N° 63: Vista Perspectiva de PanelEscenario sin movimiento de cámara de visualización	66
Figura N° 64: Vista Perspectiva de PanelEscenario con rotación a la izquierda y rotación a la derecha	67
Figura N° 65: Vista Perspectiva de PanelEscenario con rotación hacia arriba y rotación hacia abajo	67
Figura N° 66: Vista Perspectiva de PanelEscenario con acercamiento y alejamiento ..	69
Figura N° 67: Captura de pantalla al contenido de un archivo Excel generado con el sistema	75

I CAPTURA DE DATOS

De manera predeterminada, el HiloLeap comienza a capturar los datos enviados por el dispositivo Leap Motion. Para realizar esta acción, el HiloLeap tiene al atributo Leap, que es a su vez una clase que hereda de la clase Listener propio de la biblioteca de Leap Motion.

La clase Leap sobrescribe los métodos OnInit, onConnect, onDisconnect, onExit y onFrame de la clase Listener, de esta manera, se disponen de los datos capturados de las manos, representados en la clase Hands en cada una de las capturas, representados en los Frame de la biblioteca del Leap Motion.

El método modificarEscenario de la clase hiloLeap se encarga de cambiar los datos que se encuentran en el EscenarioVirtual cada vez que recibe una nueva captura del Listener, utilizando los métodos addModifMano del EscenarioVirtual, pasándole como parámetro objetos Hand.

Previo a realizar cambios en los objetos Mano y Dedo ya existentes (que tengan el mismo parámetro ID) se verifica si el dispositivo Leap Motion los reconoce internamente como objetos capturados previamente u objetos nuevos, verificando así si algún objeto anterior ha sido eliminado (por ejemplo, cuando la mano sale del rango de captura del Leap Motion) procediendo a borrar los objetos inexistentes de EscenarioVirtual en el método verifManosElim.

La captura de datos va a funcionar siempre que el método getPermisoHilo, que utiliza el método getHLeapActivo del parámetro controlador (de HiloLeap), el cual es un ControladorHilos devuelva el valor true. Cabe aclarar que dicho método va a devolver false mientras se reproduzca un archivo, es decir, siempre que el módulo de Manejo de Archivos, específicamente HiloReproduccion haya usado el método activarConfigReproduccion del ControladorHilos. Esta configuración se mantendrá hasta que se utilice el método activarConfigPredeterminada del ControladorHilos y en caso de que se realice una grabación por parte del módulo Manejo de Archivos, no afectará al módulo de Captura de Datos.

II REPRESENTACIÓN VIRTUAL

Los elementos que se representan virtualmente son las manos y el teclado. Estos elementos son representados en el sistema por las clases Mano y Teclado respectivamente y también son atributos de la clase EscenarioVirtual.

El EscenarioVirtual es la clase principal del sistema, debido a que contiene los objetos que son representados gráficamente por el módulo de Producción Visual, sus objetos son necesarios para determinar si el módulo de Producción de Sonido va a producir algún sonido, contiene el objeto que representa teclado virtual que el módulo de Modificación Virtual puede cambiar y finalmente, los datos de sus objetos sirven para realizar la producción de archivos en el módulo Manejo de Archivos.

Las objetos Mano y Teclado, los cuales están en el EscenarioVirtual, tienen entre sus datos, la posición en la que se encuentran en el espacio tridimensional. Los datos de las posiciones son vectores de tamaño 3 tipo float donde cada uno representa a la posición en milímetros respecto al eje x, eje y al eje z respectivamente.

Se toma como punto de origen el centro de la superficie superior del dispositivo Leap Motion y se toma como ejes de coordenadas a los mismos ejes predefinidos por Leap Motion. En la Figura N° 1 se puede observar una representación del eje de coordenadas utilizado en la Representación Virtual.

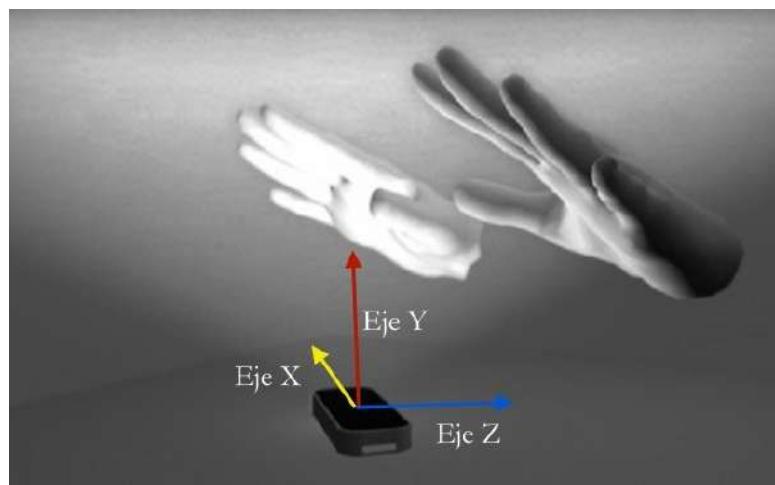


Figura N° 1: Representación de eje de coordenadas en la Representación Virtual

Fuente: (Leap Motion)

2.1 Representación virtual de la manos, dedos y huesos

Los objetos Mano son creados a partir de los datos de objetos Hand (propios de la biblioteca de Leap Motion) que el módulo Captura de Datos pasa como parámetro. El método encargado de la creación de las manos addModifMano del EscenarioVirtual.

El método addModifMano utiliza el constructor de Mano utilizando los parámetros ID que es el identificador de la mano predeterminado por el Leap Motion y el tipoMano que es true en caso de que la mano sea derecha y false en caso de que la mano sea izquierda. Otros métodos utilizados para la creación de Mano son los métodos setPosMuneca para la posición de la muñeca y setVelocidadMuneca para la velocidad de la muñeca.

Además, para completar la creación de la mano, el método addModifMano utiliza el método addModifDedo y éste a su vez addModifHueso internamente para completar la estructura de la mano, pasando como parámetros objetos Finger y Bone (de la biblioteca de Leap Motion) respectivamente.

El método addModifDedo utiliza un objeto Finger (de la biblioteca de Leap Motion) para crear un objeto Dedo que, recibe como parámetros en su constructor un ID y tipoDedo, ambos tipo int. El ID es, al igual que el caso de la mano, un identificador predeterminado por el Leap Motion. Por otra parte el tipoDedo es un número que identifica a cada tipo de dedo según la notación internacional de piano, siendo en ambas manos 1 el pulgar, 2 el índice, 3 el medio, 4 el anular y 5 el meñique. El número tipoDedo es asignado según los tipos de Finger que maneja Leap Motion, los cuales son: TYPE_THUMB, TYPE_INDEX, TYPE_MIDDLE, TYPE_RING y TYPE_PINKY.

En la Figura N° 2 se puede observar la mano derecha y la mano izquierda, las partes que van a ser representadas en Mano y la numeración de los dedos.

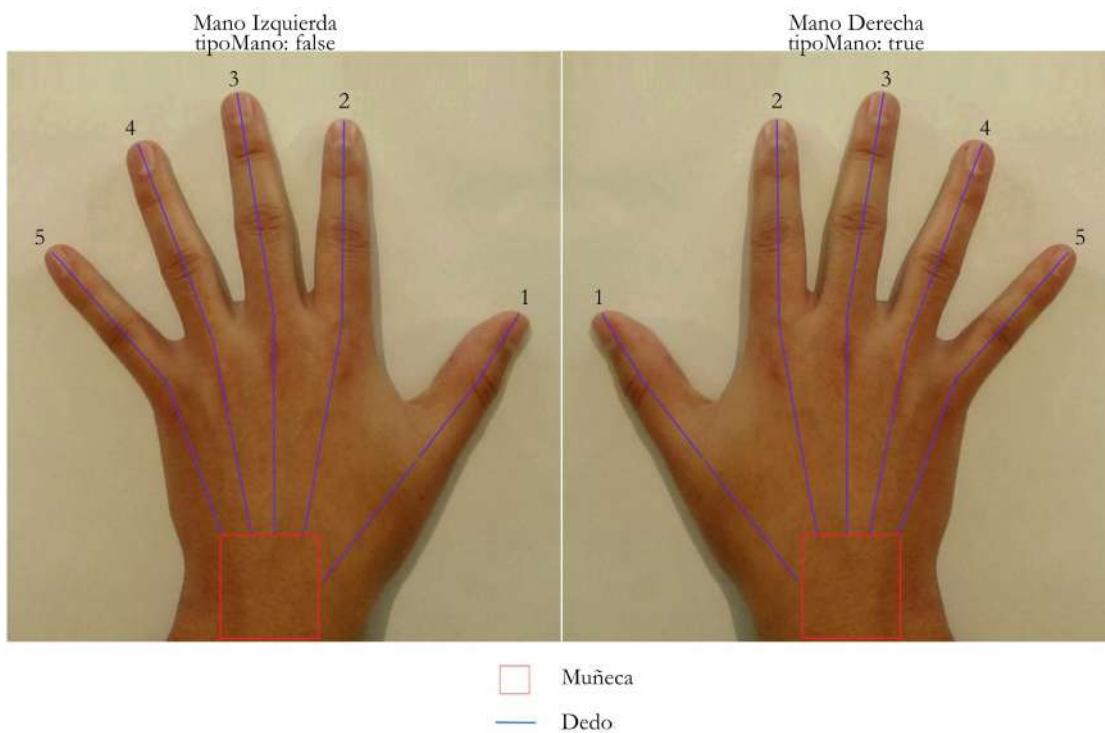


Figura N° 2: Imagen de mano derecha e izquierda y sus partes a ser representadas

Fuente: Elaboración propia

Para completar los datos del Dedo, se utilizan los métodos:

- `setPosPunta` donde se pasa como parámetro la posición de la punta del dedo (que se obtiene gracias al método `tipPosition` de la clase `Finger`) y se guarda en el atributo `PosPunta`
- `setVelocidad` donde el parámetro es la velocidad de la punta de dedos (que se obtiene mediante el método `tipVelocity` de `Finger`) y se guarda en el atributo `velocidadY`
- `setVectorDedo` donde el parámetro es la dirección de la punta del dedo en vector normalizado (`float`) del hueso distal, desde la base hasta la punta (que se obtiene mediante el método `direction`) y se almacena en el atributo `vectorDedo`.

Para agregar los objetos Hueso, se utiliza el método `addModifHueso`, que crea y agrega un Hueso al Dedo a partir de la información de un objeto Bone (de la biblioteca de Leap Motion).

El Hueso es creado pasando como parámetro tipo (tipo de dato int) que representa el tipo de Hueso. Éste es asignado del 1 al 4 respectivamente a partir de los tipos de Bone que existen en el Leap Motion: TYPE_METACARPAL, TYPE_PROXIMAL, TYPE_INTERMEDIATE y TYPE_DISTAL.

Es necesario recalcar que el Dedo con el tipoDedo igual a 1 (representación del dedo pulgar) no tiene el Hueso con el tipo igual a 1 (representación del hueso metacarpo).

En la Figura N° 3 se puede observar la anatomía de la mano y en la Figura N° 4 se puede observar la asignación equivalente de los tipos de hueso que serán representados.



Figura N° 3: Anatomía de la mano

Fuente: (The University Chicago Medicine)



Figura N° 4: Tipos de hueso según representación en clase Hueso

Fuente: Elaboración propia y (Samper, 2006)

Los datos de los objeto `Hueso` son completados gracias a los métodos `setPosInicial` y `setPosFinal`, que asignan las coordenadas del inicio y el fin del hueso representando a los atributos `posInicial` y a `posFinal` respectivamente, tomando en cuenta que el inicio es el extremo del hueso más cercano a la muñeca. El método `setAncho` que asigna el ancho del hueso al atributo `ancho` y finalmente, a los métodos `setBaseX`, `setBaseY` y `setBaseZ` que contienen las coordenadas en *arrays* de `float` que representan la direccionalidad de los huesos como un vector normalizado, siendo cada uno de ellos el vector que representa a un eje coordenado local tomando en cuenta que:

- El vector base X es perpendicular al eje longitudinal del hueso y sale del lado derecho del dedo
- El vector base Y es perpendicular al eje longitudinal del hueso, sale de la parte superior e inferior del dedo siendo más positivo en la dirección hacia arriba
- El vector base Z está lineado con el eje longitudinal del hueso y es más positiva hacia la base del dedo

Los vectores base sirven para recrear visualmente los dedos de manera más elaborada, ya que brindan los datos para poder determinar los vértices de las mallas poligonales según la posición del hueso.

En la Figura N° 5 se puede observar la representación gráfica de los vectores base en dos huesos del dedo 1 y tres huesos del dedo 2 de la mano izquierda

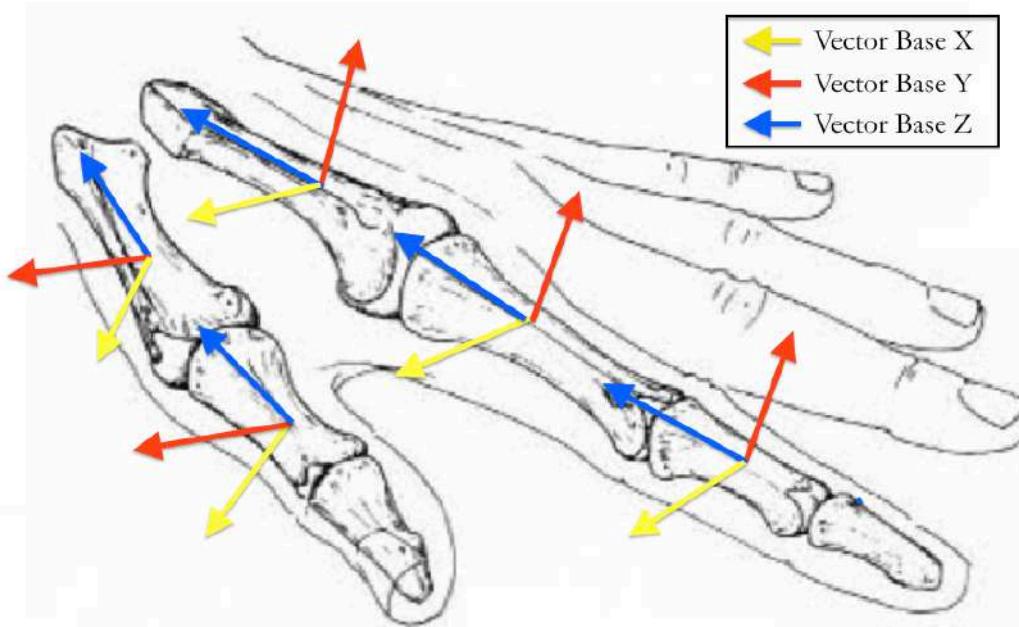


Figura N° 5: Representación gráfica de los vectores base de los huesos

Fuente: Elaboración propia

Una vez que se haya terminado de construir y completar los datos del objeto Hueso, se agrega al Dedo y éste a su vez, a la Mano.

En caso de que se detecte que el ID es el mismo en un objeto Mano (que ya existe) en EscenarioVirtual y un objeto Hand que el Leap Motion captura, se modifican los datos de Mano sin eliminarlo como objeto.

En caso de que el ID de un objeto Mano no se encuentre en ninguno de los objetos Hand se procede a eliminar el objeto Mano con el método eliminarMano, ya que quiere decir que Leap Motion ya no ha capturado la misma mano. Esto puede no ser cierto (que realmente sea otra mano) en todos los casos, pero evita que se acumulen objetos Mano en el EscenarioVirtual.

2.2 Representación virtual del teclado

Para poder entender la representación virtual del teclado, se recomienda leer previamente el Anexo 1, que brindará el conocimiento necesario de los términos utilizados en esta sección.

Debido a que el teclado tiene una parte física tangible y otra parte sonora o musical, la representación virtual está muy ligada al módulo de Producción de Sonido. Por esta razón, la descripción de algunos aspectos de la representación virtual del teclado será complementada en la Producción de Sonido.

Para cumplir con los requerimientos del proyecto, se desarrolló un teclado virtual que es representado en la clase Teclado. El Teclado está como atributo en el EscenarioVirtual y es creado mediante el método `crearTeclado`, que tiene como parámetros el nombre de la nota más baja con la que comienza el teclado, el número de octava a la que pertenece la nota, el número de teclas que tendrá el teclado y la posición del teclado en el espacio tridimensional (punto de inicio superior de la tecla con la nota más baja del teclado). Los datos de la posición inicial corresponden al atributo `posicionInicial` y, el número de teclas a `numTeclas` de la clase Teclado.

Debido al tiempo de desarrollo del proyecto no se pudo completar la interfaz de variación de las notas del teclado (ya que no era un requerimiento, sino una mejora), lo que proporcionaba la opción de modificar el tamaño del teclado al usuario. Sin embargo, los métodos utilizados internamente son los mismos, con la diferencia que se pasan parámetros preestablecidos en el EscenarioVirtual, los cuales son los siguientes: como nota de inicio se establece la nota *do*, octava 3, 26 teclas y, como posición inicial, el punto $(-180, 108, -145)$ que puede ser modificado posteriormente moviendo el teclado virtual (ver movimiento, eliminación y creación de teclado).

El teclado (del presente proyecto) de manera predeterminada está compuesto 26 por teclas, desde la nota *Do3* hasta la nota *Do#5*.

La tecla está representada por dos clases, la clase Tecla y la clase TeclaMusical, siendo la primera una clase heredera de la segunda. La Tecla corresponde a la representación física tangible de la superficie de una tecla real y la TeclaMusical representa a la parte sonora y funcional de la tecla.

Para determinar la estructura de las teclas, se tomó en cuenta la distribución estándar de los teclados de los pianos, donde se verificó que:

- Cada intervalo de séptima (desde la tecla correspondiente a la nota *do* hasta la nota *si*) mide 16,5 cm
- Cada intervalo de séptima tiene 12 teclas, 7 blancas y 5 negras
- El ancho de las teclas en la parte inferior es 1/7 parte del intervalo de séptima
- El ancho de las teclas en la parte superior es 1/12 parte del intervalo de séptima, excepto en el caso de la tecla correspondiente a la nota *mi*, que tiene un grosor de $\frac{3}{7}$ menos $\frac{4}{12}$ del intervalo de séptima, para compensar el desajuste entre la división entre 12 y 7 del dicho intervalo
- El ancho de las teclas negras se mantiene constante desde el inicio al fin
- El ancho de las teclas blancas varía, por lo tanto, cada tecla blanca tiene una forma diferente dependiendo de la disposición de las teclas negras
- El largo de las teclas negras es 9,5 cm.
- El largo de las teclas blancas es 14,7 cm

Además, se determinó que, para realizar la creación de un teclado virtual básico no es necesario cambiar la altura de las teclas blancas y negras, debido a que se tiene sólo una superficie regular de apoyo de los dedos y la técnica básica no requiere la distinción de altura para las notas alteradas.

En la Figura N° 6 se puede observar la estructura de un intervalo de séptima del teclado (secuencia que se repite a lo largo de un instrumento de teclado) con las medidas anteriormente determinadas, siendo S el largo físico de un intervalo de séptima en el teclado, B el largo de las teclas blancas y N el largo de las teclas negras.

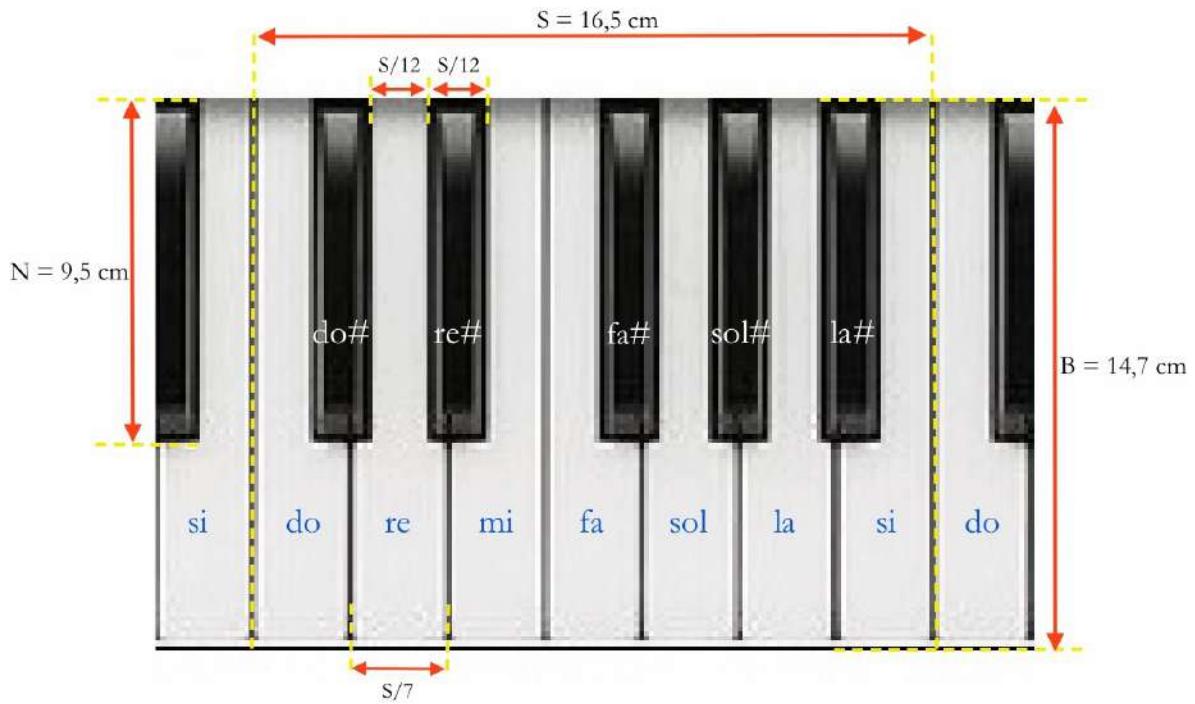


Figura N° 6: Estructura de intervalo de séptima (do a si) del teclado

Fuente: Elaboración propia

Por lo tanto, para representar una tecla, se determinó que:

- El número máximo de vértices necesarios para representar a una tecla blanca es 8, de tal manera que si se llegan a unir mediante líneas, forman el borde de la tecla.
- El número de vértices necesarios para representar a una tecla negra es 4, debido a su forma regular.

En la Figura N° 7 se puede observar la representación (aproximada) de una tecla blanca y una tecla negra, siendo V_x y A_x los vértices y las aristas respectivamente.

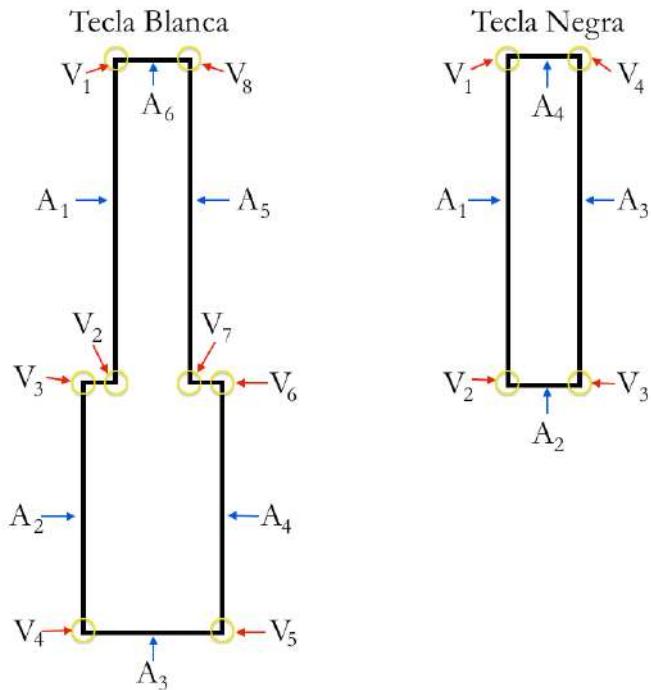


Figura N° 7: Representación de las teclas blanca y negra, compuesta de vértices y aristas

Fuente: Elaboración propia

Debido al tipo de representación planteado, la clase Tecla tiene como atributos:

- La estructura como matriz de 8 por 2 en tipo *float* cuando la tecla es blanca y 4 por 2 cuando la tecla es negra. Estos datos representan los vértices (en los ejes *x* y *z*) y sirven para representar gráficamente a las teclas y para evaluar si un dedo toca a una tecla.
- La *posicionY* que representa la posición respecto al eje *y* (vertical) en la que se encuentra la tecla con un *float*, debido a que todas las teclas están a la misma altura, es decir en la misma posición respecto al eje *y*
- El *tipo* que es tipo *boolean*, es *true* en caso de las teclas blancas y *false* en caso de las teclas negras.

Como consecuencia, para crear las teclas que conforman el teclado se utilizan los siguientes parámetros en la constructor de Teclado:

- Parámetros *nombreNota* y *octava*, son los que permiten determinar la nota inicial más grave, desde la que, gracias al método *getNextNota* de la clase utilitaria *HerramientasTeclado* que devuelve la nota siguiente (la nota más alta

inmediata), se puede tener todas las notas requeridas para la creación de las teclas respectivas

- Parámetro `numTeclas`, que sirve para determinar el número de teclas que va a tener el teclado. Ésta información posteriormente se queda almacenada en el atributo `numTeclas`, que internamente funciona como el límite numérico de repetición de ciclo para creación de las teclas, además de ser indicador de índice máximo cuando se requiere alguna tecla del teclado (a la que puede acceder mediante el método `getNumTeclas`)
- Parámetro `posición`, que sirve para determinar la posición del primer vértice de la tecla más baja del teclado. Gracias al método `getNextPosInicial` de la clase utilitaria `HerramientasTeclado`, se puede determinar la posición inicial de la siguiente tecla y se pueden crear las teclas iterativamente

Por otra parte, los objetos `Tecla` tienen, en su propio constructor (de la clase `Tecla`) los parámetros `nota` (de la clase `Nota`), y `la posicionIni`, que es un vector de tamaño 3 de tipo `float`.

- El parámetro `Nota` del constructor de `Tecla` es necesario para la creación de la parte musical de la tecla, para determinar qué tipo de tecla es (blanca o negra) y para determinar la posición de sus vértices en `estructura` (los cuales son necesarios para la correcta representación gráfica).
- El parámetro `PosicionIni` es necesario para establecer la posición del primer vértice de la tecla y construir a partir de éste dato la estructura de la tecla

La estructura de la tecla negra (que se puede ver en la Figura N° 7) es definida tomando en cuenta los siguientes puntos:

- El vértice V_2 está definido por, el largo de las tecla negra (dato que se encuentra en la clase utilitaria `HerramientasTeclado`, en el atributo `largoTeclaNegra`) sumado al componente z del vértice V_1
- El vértice V_3 está definido por, el largo y el ancho de la tecla negra (el dato del ancho se encuentra en la clase utilitaria `HerramientasTeclado`, en el atributo

`anchoTeclaNegra)` sumados a los componentes z y x del vértice V_1 respectivamente

- El vértice V_4 está definido por, el ancho de la tecla negra sumado al componente x del vértice V_1

La estructura de la tecla blanca (que se puede ver en la Figura N° 7) es definida tomando en cuenta los siguientes puntos:

- Los componentes z de los vértices V_2, V_3, V_6 y V_7 son iguales al largo de la tecla negra sumado al componente z del vértice V_1
- Los componentes z de los vértices V_4 y V_5 son iguales al largo de la tecla blanca sumado al componente z del vértice V_1
- El componente z del vértice V_8 es igual al componente z del vértice V_1
- El componente x del vértice V_2 es igual al componente x del vértice V_1
- Los componentes x de los vértices V_3 y V_4 son iguales al componente x del vértice V_1 restados por d , siendo d igual a: la multiplicación del número de la nota anterior en la escala (que se obtiene con el método `getNumCromaticaPrev`) por el ancho de la tecla negra (dato que se encuentra en el atributo `anchoTeclaNegra` de la clase utilitaria `HerramientasTeclado`) restados por, el número natural de la nota anterior (que se obtiene con el método `getNumNaturalPrev`) multiplicado por el ancho de la tecla blanca (dato que se encuentra en la clase utilitaria `HerramientasTeclado`, en el atributo `anchoTeclaBlanca`).
- Los componentes x de los vértices V_5 y V_6 son iguales al ancho de la tecla blanca sumado al componente x del vértice V_3
- Los componentes x de los vértices V_7 y V_8 son iguales al ancho de la tecla blanca sumado al componente x del vértice V_1 , excepto en el caso de la tecla correspondiente a la nota mi , donde serán iguales al componente x del vértice V_5

En la Figura N° 8 se puede observar una sección del teclado con los números correspondientes a la numeración de la escala y la numeración natural, además del nombre de la nota de la tecla.

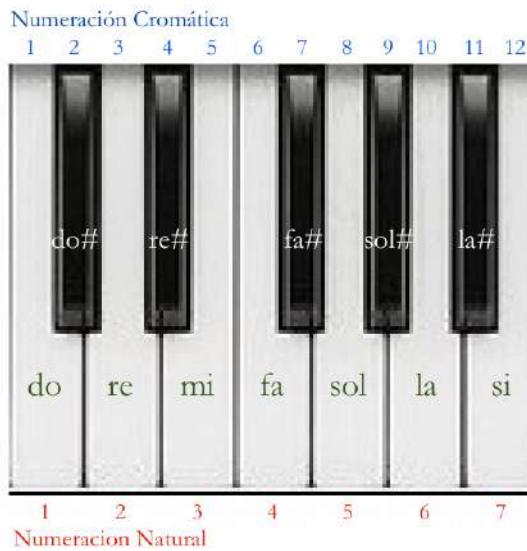


Figura N° 8: Numeración cromática y numeración natural de las teclas

Fuente: Elaboración propia

En la Figura N° 9 se puede observar la representación gráfica de los puntos utilizados para definir la estructura de la tecla blanca correspondiente a la nota *sol*, es decir, la asignación de valores (componentes x y z de los vértices) a su estructura.

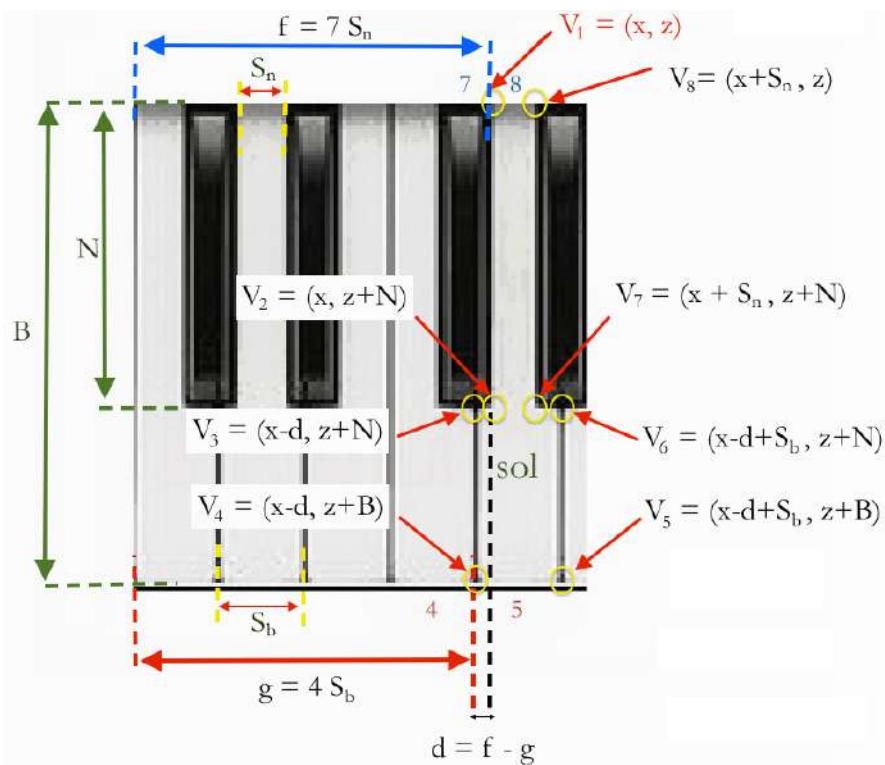


Figura N° 9: Representación gráfica de la estructura (valores de los vértices) de la tecla sol

Fuente: Elaboración propia

III PRODUCCIÓN DE SONIDO

La producción de sonido es principalmente gestionado por HiloSonido, que tiene dos funciones:

- Asignar el GeneradorMIDI al teclado (que a su vez lo asignará a todas sus teclas) cuando un teclado es creado
- Verificar si los dedos tocan o sueltan las teclas, para hacer que cada tecla involucrada mande el respectivo evento a GeneradorMIDI (mediante los métodos `esPresionada` y `esSoltada` de la clase TeclaMusical)

Un HiloSonido es creado cuando el software inicia (e inmediatamente es iniciado el método `run`) y el método `run` termina su ejecución cuando el método `getSonidoActivo` de ControladorHilos devuelve el valor `false`, lo cual ocurre al finalizar el programa.

En el constructor de HiloSonido se crea un GeneradorMIDI, el que HiloSonido mediante el método `addGeneradorMIDIaTeclado`, asigna a Teclado (que está dentro de EscenarioVirtual). Como un teclado puede ser eliminado y creado de nuevo, el método `addGeneradorMIDIaTeclado` será usado cada vez que el usuario cree un nuevo teclado (en el módulo de Modificación Virtual).

3.1 Funcionamiento de GeneradorMIDI

El GeneradorMIDI implementa la interfaz EstadoListener y trabaja con EstadoEvento, que contiene en sus atributos: nota (objeto Nota) que debe ser detenida o reproducida por el generador y, el estado (objeto Estado), que puede ser PRESIONADA o SUELTA.

Cuando GeneradorMIDI recibe un evento (objeto EstadoEvento) quiere decir que una tecla ha sido tocada por la punta de un dedo (estado PRESIONADA), o bien, la punta del dedo ha dejado de estar en la tecla (estado SUELTA).

Si el estado es PRESIONADA, GeneradorMIDI produce el sonido de la nota mediante el método `iniciarSonido` (internamente se utiliza el método `noteOn` propio de la biblioteca

MIDI de Java), que tiene como parámetro de entrada el número MIDI que debe producir. El número MIDI de una nota está en el atributo numMIDI de Nota.

Si el estado es SUELTA, GeneradorMIDI deja de producir el sonido mediante el método apagarSonido (internamente se utiliza el método noteOff propio de la biblioteca MIDI de Java) que, al igual que iniciarSonido, tiene como parámetro de entrada al número MIDI de la nota.

3.2 Funcionamiento de Nota

La clase Nota representa una nota musical y contiene la información que llega a reproducirse en el GeneradorMIDI.

La clase Nota tiene como atributos:

- nombre que es el nombre de la nota tipo String. Puede ser un nombre de nota perteneciente a la escala natural: *do*, *re*, *mi*, *fa*, *sol*, *la*, *si*, o caso contrario, el nombre de una nota alterada, es decir, *do#*, *re#*, *fa#*, *sol#* o *la#*
- octava que es el número de la octava tipo int. El número corresponde al índice acústico científico (ver Anexo 1)
- numCromatica que es el número de nota en la escala cromática tipo int. El número es el correspondiente a la nota según la disposición de semitonos que conforman la escala cromática. En la Tabla Nº 4 se puede observar la asignación de número de escala cromática a las notas

Tabla Nº 1: Numeración cromática

NOTA	do	do#	re	re#	mi	fa	fa#	sol	sol#	la	la#	si
Nº	1	2	3	4	5	6	7	8	9	10	11	12

Fuente: Elaboración propia

- numMIDI que es el número de la nota según la numeración de notas de la biblioteca MIDI de Java (`javax.sound.midi`). El número se utiliza en la producción de

sonido donde la nota *do4* es la que corresponde al número 60. Para calcular el número MIDI de cada una de las notas, se utilizan los datos del número según la escala cromática (del 1 al 12) y el número de la octava. A continuación se muestra la ecuación (1) utilizada el cálculo, siendo N_{MIDI} el número MIDI, N_{octava} el número de la octava y $N_{cromática}$ el número de la nota en la escala cromática.

$$N_{MIDI} = 60 + 12(N_{octava} - 4) + (N_{cromática} - 1) \quad (1)$$

3.3 Activación de sonido mediante dedo

El sistema produce un sonido cuando la posición de la punta de un dedo toca la superficie de una tecla y se mantiene dentro de un espacio de tolerancia, es decir, cuando el atributo *posPunta* de la clase *Dedo* está dentro del espacio que tiene, como ancho, el borde de la tecla (vértices que se encuentran en estructura de *Tecla*) y como profundidad, el atributo *toleranciaToque* (de la clase *HerramientasTeclado*).

En las siguientes figuras se puede observar la representación gráfica de un dedo tocando una tecla, el espacio de tolerancia, la profundidad de tolerancia de toque, la superficie de la tecla y los bordes de la tecla. Las Figuras N° 10, 11 y 12 corresponden a las vistas lateral, superior y diagonal respectivamente.

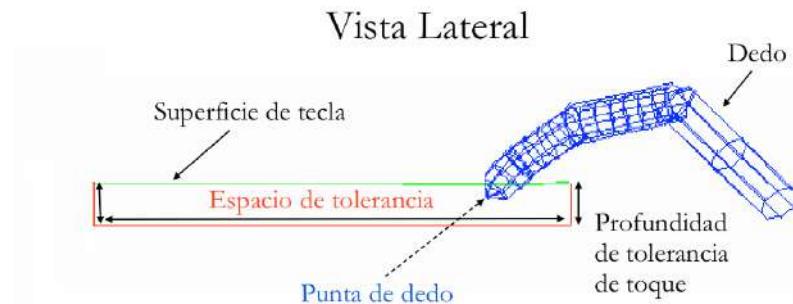


Figura N° 10: Representación gráfica de un dedo tocando tecla – Vista lateral

Fuente: Elaboración propia

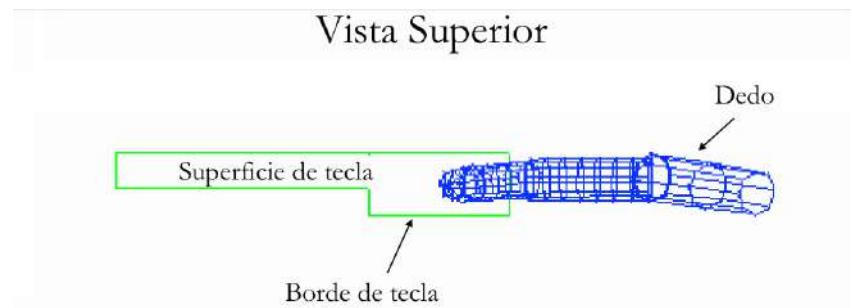


Figura N° 11: Representación gráfica de un dedo tocando tecla – Vista superior

Fuente: Elaboración propia

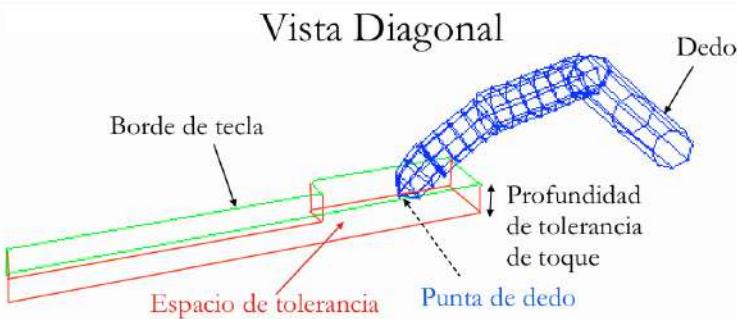


Figura N° 12: Representación gráfica de un dedo tocando tecla – Vista diagonal

Fuente: Elaboración propia

Para evaluar si la punta del dedo (`posPunta de la clase Dedo`) está en el espacio de tolerancia, se utiliza el método `verifTecladoTocado` de la clase `HiloSonido`, que internamente revisa individualmente si cada tecla (todos los objetos `Tecla` de `Teclado`) es presionada o soltada mediante el método `verifTeclaTocada`.

El método `verifTeclaTocada`, a su vez, revisa si la posición de todos los dedos de las manos del escenario virtual están en el espacio de tolerancia de la tecla mediante el método `verifMano` que, devuelve un valor boolean. Si `verifMano` es `true`, el método `verifTeclaTocada` utiliza el método `esPresionada` de `Tecla` para iniciar con la producción de sonido, si es `false`, utiliza el método `esSoltada`. El método `esPresionada` sólo puede ser usada si el estado (parámetro `estado`) de la tecla previamente es `SUELTA`, de manera equivalente, el método `esSoltada` sólo puede ser usado si el estado de la tecla previamente es `PRESIONADA`.

El método `verifMano` utiliza internamente el método `puntoEnTecla` de `Tecla` para determinar si algún o algunas puntas de dedo de la mano (`posPunta de Dedo`) se encuentran

en el espacio de tolerancia de toque de la tecla. Para determinar si una tecla es presionada, el espacio de tolerancia es delimitado por los bordes de los prismas rectangulares que forman el vértice de las teclas y la profundidad de tolerancia de toque.

En caso de las teclas negras, existe un solo prisma rectangular, en función a los vértices V_1, V_2, V_3, V_4 de la tecla y la profundidad de tolerancia de toque Q , que sirve como límite de evaluación de la punta del dedo. En la Figura N° 13 se puede observar la representación gráfica del espacio de tolerancia (un prisma) de una teclas negra modelo.

Tecla Negra

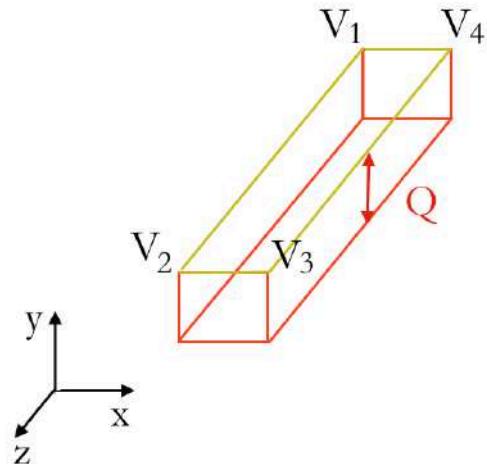


Figura N° 13: Representación gráfica de espacio de tolerancia – Tecla Negra

Fuente: Elaboración propia

En caso de las teclas blancas, existen dos prismas rectangulares, uno en función de los vértices V_1, V_2, V_7, V_8 de la tecla y la profundidad de tolerancia de toque Q y otro en función de V_3, V_4, V_5, V_6 de la tecla y la profundidad de tolerancia de toque Q . En la Figura N° 14 se puede observar la representación gráfica del espacio de tolerancia (dos prismas) de una tecla blanca modelo.

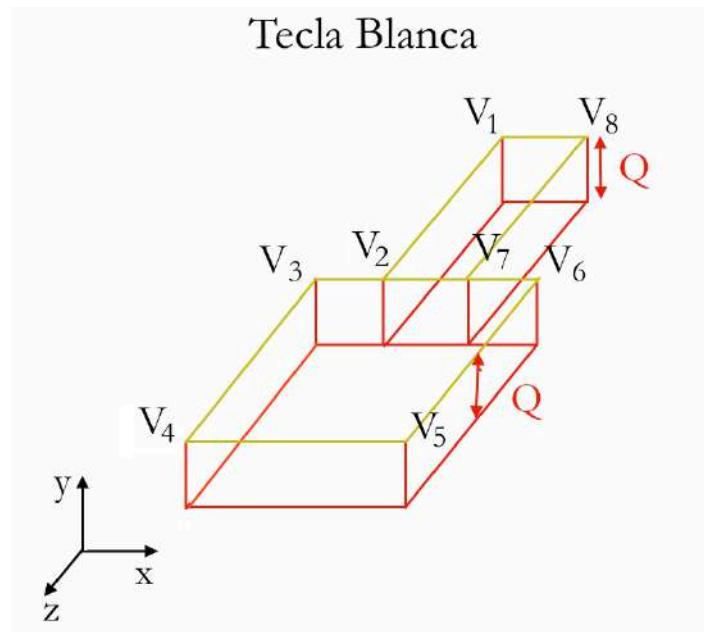


Figura N° 14: Representación gráfica de espacio de tolerancia – Tecla Blanca

Fuente: Elaboración propia

Debido a que se usan prismas creados a partir de los puntos de los vértices para evaluar el punto de una tecla, el método `puntoEnTecla` internamente usa el método `puntoEnPrisma` (de la clase utilitaria `HerramientasTeclado`), donde se utilizan los vértices de la tecla que conforman la parte superior del prisma. Cabe aclarar que la profundidad de tolerancia de toque Q no es un parámetro, ya que en la misma clase `HerramientasTeclado` se encuentra el atributo `toleranciaToque`.

Por lo tanto, para que un punto P se encuentre dentro del prisma, con vértices de la cara superior V_1, V_2, V_3 y V_4 (como se muestra en la Figura N° 13), siendo (x_n, y_n, z_n) los componentes del vértice V_n , (x_p, y_p, z_p) los componentes de P y siendo Q la profundidad de tolerancia de toque, debe cumplir con (2), (3) y (4). Debe considerarse que la punta del dedo P es la coordenada que representa la posición de la punta del dedo, por lo tanto, sólo se podrá encontrar dentro de un prisma a la vez.

$$x_4 \leq x_p \leq x_1 \quad (2)$$

$$(y_1 - Q) \leq y_p \leq y_1 \quad (3)$$

$$z_1 \leq x_p \leq z_2 \quad (4)$$

Como se puede ver en (2), (3) y (4), no se necesita la posición de V_3 debido a que, no existe la posibilidad que el usuario pueda hacer que el teclado virtual realice un movimiento de rotación, en consecuencia, el largo de las teclas siempre serán paralelas al eje z y perpendiculares al eje y y al eje x del espacio virtual, por lo tanto, los vértices V_1, V_2, V_4 contienen los datos necesarios para determinar si un punto está dentro del espacio de tolerancia.

IV PRODUCCIÓN VISUAL

La producción visual es iniciada y gestionada principalmente por `HiloVisual`, que contiene a `Pantalla` (`JFrame` principal del sistema). `HiloVisual` es instanciado en `SistemaTecnicaPianistica`, donde el método `run` del `HiloVisual` es ejecutado dentro del método `main` para activar todo el módulo de Producción Visual del sistema.

La clase `HiloVisual`, tiene como atributo, además de `pantalla` (de la clase `Pantalla`), el `controlador` (que es un objeto de la clase `ControladorHilos`), el cual le es asignado como parámetro en su constructor. El controlador sirve para coordinar el funcionamiento (inicio y final) de otras clases que heredan de la clase `Thread`.

4.1 Composición general y funcionamiento de `Pantalla`

La clase `Pantalla` contiene a las clases que heredan de `JPanel`, las cuales se pueden ver en el funcionamiento regular del sistema. Estas clases, clasificadas por su funcionalidad, pueden servir para visualización de datos (con las que el usuario no puede interactuar directamente, sólo visualizarlas) y las clases que sirven como interfaz activa (con las que el usuario interactúa mediante el *mouse*).

En la Figura N° 15 se puede observar la composición gráfica de `Pantalla`.

Las clases que sirven para la visualización de datos y componen `Pantalla` como atributos son:

1. `PanelAngulos`, donde se muestran los ángulos de la inclinación de los dedos (objetos dedo) que se encuentran en el escenario virtual
2. `PanelEscenario`, donde se muestran las representaciones gráficas de los componentes (manos y teclado) del escenario virtual desde diferentes puntos de visualización.

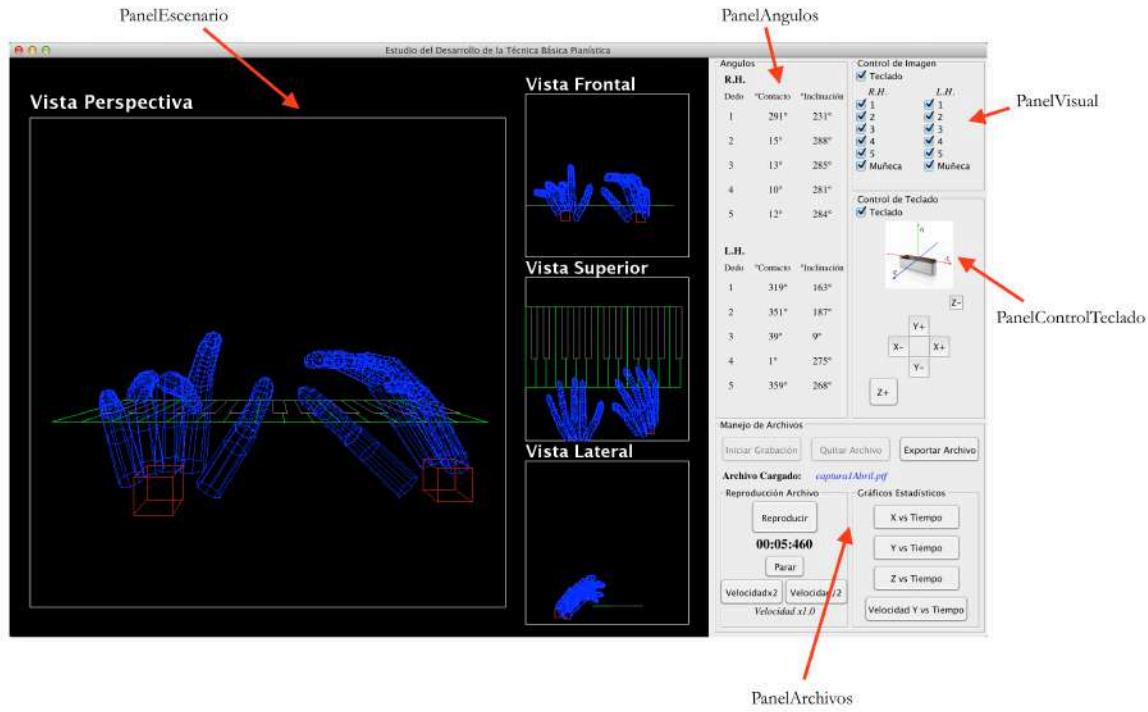


Figura N° 15: Composición gráfica de Pantalla

Fuente: Elaboración propia

Las clases que sirven como interfaz activa de usuario y componen Pantalla como atributos son:

1. PanelControlTeclado mediante el que, el usuario interactúa con el módulo de Modificación Virtual para modificar la posición del teclado (haciendo clic en botones del panel), además de activar o desactivar al mismo mediante un *checkbox*. La clase ManejoControlTeclado es su *listener* de eventos y es la que ejecuta los métodos accionados por el usuario para que se cumplan los cambios requeridos, en este caso, en el teclado del escenario virtual.
2. PanelControlVisual mediante el que, el usuario interactúa con el módulo de Modificación Virtual para activar o desactivar (a través de *checkboxes* del panel) la creación de las representaciones gráficas de los componentes (manos y teclado) que se encuentran en el escenario virtual. La clase ManejoVisual es su *listener* de eventos y es la que ejecuta los métodos accionados por el usuario para que se cumplan los cambios requeridos en la visualización de componentes.

-
3. PanelArchivo mediante el que, el usuario interactúa con el módulo de Manejo de Archivos. Tiene como *listener* a la clase ManejoDatosArchivos (del módulo Manejo de Archivos), la cual ejecuta los métodos accionados por el usuario para que se grabe, cargue o exporte los datos de un archivo.
 4. PanelDatos contiene los botones que permiten al usuario accionar la carga de archivos (btCargarArchivo), grabación de archivos (btGrabacion) y exportación de datos de archivos a formato Excel (btExportarDatos). Contiene además, el PanelReproduccion y el PanelEstadística.

El funcionamiento de la pantalla es visible desde el inicio del programa y finaliza al terminar el mismo, cuando el usuario cierra el JFrame general (Pantalla), momento en el que se activa el método mmTerminarPrograma, que despliega un mensaje para confirmar la salida del usuario.

En la Figura N° 16 se puede observar el mensaje de confirmación de salida de la aplicación desplegado por el método mmTerminarPrograma.

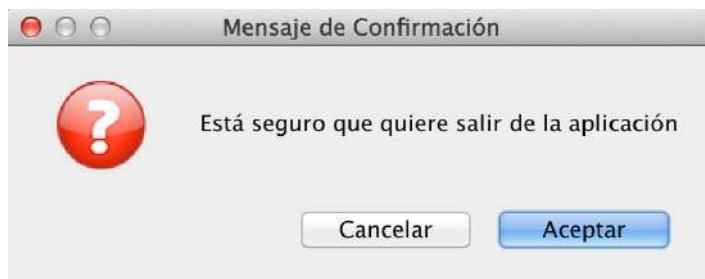


Figura N° 16: Mensaje de confirmación de salida de la aplicación

Fuente: Elaboración propia

Cuando el usuario elige la opción Aceptar, el método terminarPrograma finaliza la ejecución del programa terminando la ejecución de todos los hilos mediante el método terminarHilos de ControladorHilos.

Si el usuario elige la opción Cancelar, la aplicación sigue funcionando con normalidad.

4.2 Composición general y funcionamiento de *PanelAngulos*

La clase *PanelAngulos* hereda de *JPanel* y se encarga de mostrar los ángulos de inclinación (respecto a una superficie vertical) y de contacto (respecto a una superficie horizontal) de los dedos.

PanelAngulos está compuesta por cuatro vectores de *JLabel*, cada uno con 5 *JLabel* que representan los datos de cada dedo. Cada vector de *JLabel* muestran la información de:

- Los ángulos de inclinación de los dedos de la mano derecha. Están en el atributo *lblRHInclinacion*.
- Los ángulos de contacto de los dedos de la mano derecha. Están en el atributo *lblRHContacto*.
- Los ángulos de inclinación de los dedos de la mano izquierda. Están en el atributo *lblLHInclinacion*.
- Los ángulos de contacto de los dedos de la mano derecha. Están en el atributo *lblLHContacto*.

En la Figura N° 17 se puede observar al *PanelAngulos* y los *JLabel* que se encuentran en los atributos mostrando la información de ángulos de los dedos. Debe considerarse que, la anotación R.H. y L.H. (abreviación de Right Hand y Left Hand respectivamente), son las abreviaciones oficiales internacionalmente para nombrar la mano derecha e izquierda dentro de la escritura musical.

Para que *PanelAngulos* muestre los ángulos, el *HiloVisual* ejecuta en su método *run* el método *revisarAngulos* y *revisarVacios* 5 veces por segundo.

El método *revisarAngulos* se encarga de pasar los datos de las primeras manos (derecha e izquierda) que encuentre en el *EscenarioVirtual* al método *calculoAnguloDedos*, donde se revisa todos los dedos de las manos y se calculan los ángulos de inclinación y contacto mediante los métodos *getGradoInclinación* y *getGradoContacto* (ambos de la clase utilitaria *HerramientasVisuales*) que devuelven en número *int* los grados de inclinación y contacto a partir de los parámetros que se obtienen con el método *getVectorDedo* de la clase *Dedo*.

Angulos		
R.H.		
Dedo	°Contacto	°Inclinación
1	327°	229°
2	24°	293°
3	44°	303°
4	34°	296°
5	7°	277°
L.H.		
Dedo	°Contacto	°Inclinación
1	345°	138°
2	358°	96°
3	22°	42°
4	28°	38°
5	18°	44°

Figura N° 17: Imagen de PanelAngulos mostrando ángulos de ambas manos

Fuente: Elaboración propia

En el método `calculoAnguloDedos`, posterior a revisar todos los dedos de las manos y calcular los ángulos de inclinación y contacto, se muestran los datos obtenidos mediante los métodos `mGradoInclinacion` y `mGradoContacto` que rellenan (según los parámetros de entrada tipo de dedo y tipo de mano) los correspondientes `JLabel` de `PanelAngulos`.

El método `revisarVacios` revisará el escenario virtual y, en caso de que no se encuentre ninguna mano derecha e izquierda registrada en `EscenarioVirtual`, se utilizarán los métodos `borrarRH` y `borrarLH` respectivamente, para asignar “ ” en el texto de cada uno de los `JLabel` correspondientes.

En la Figura N° 18 se puede observar a `PanelAngulos` mostrando los ángulos de la mano derecha (Mano con `tipoMano` igual a `true`), ya que no se encuentra ninguna mano izquierda (Mano con `tipoMano` igual a `false`) en el `EscenarioVirtual`, en ese momento.

Angulos			
R.H.	Dedo	°Contacto	°Inclinación
	1	342°	250°
	2	32°	354°
	3	30°	355°
	4	41°	348°
	5	38°	342°

L.H.	Dedo	°Contacto	°Inclinación
	1	—	—
	2	—	—
	3	—	—
	4	—	—
	5	—	—

Figura N° 18: Imagen de PanelAngulos mostrando ángulos de la mano derecha

Fuente: Elaboración propia

4.3 Composición general de PanelEscenario

La clase PanelEscenario hereda de JPanel, se encarga de mostrar la representación gráfica de los elementos que se encuentran en EscenarioVirtual y utiliza clases que pertenecen a Mallas Poligonales para realizar la representación gráfica.

Está compuesta por 4 cuadros de visualización, los que son dibujados a partir de los métodos de Mallas Poligonales (ver Anexo 2), éstas son las vistas frontal, superior, lateral y perspectiva.

En la Figura N° 19 se puede observar la composición gráfica de PanelEscenario

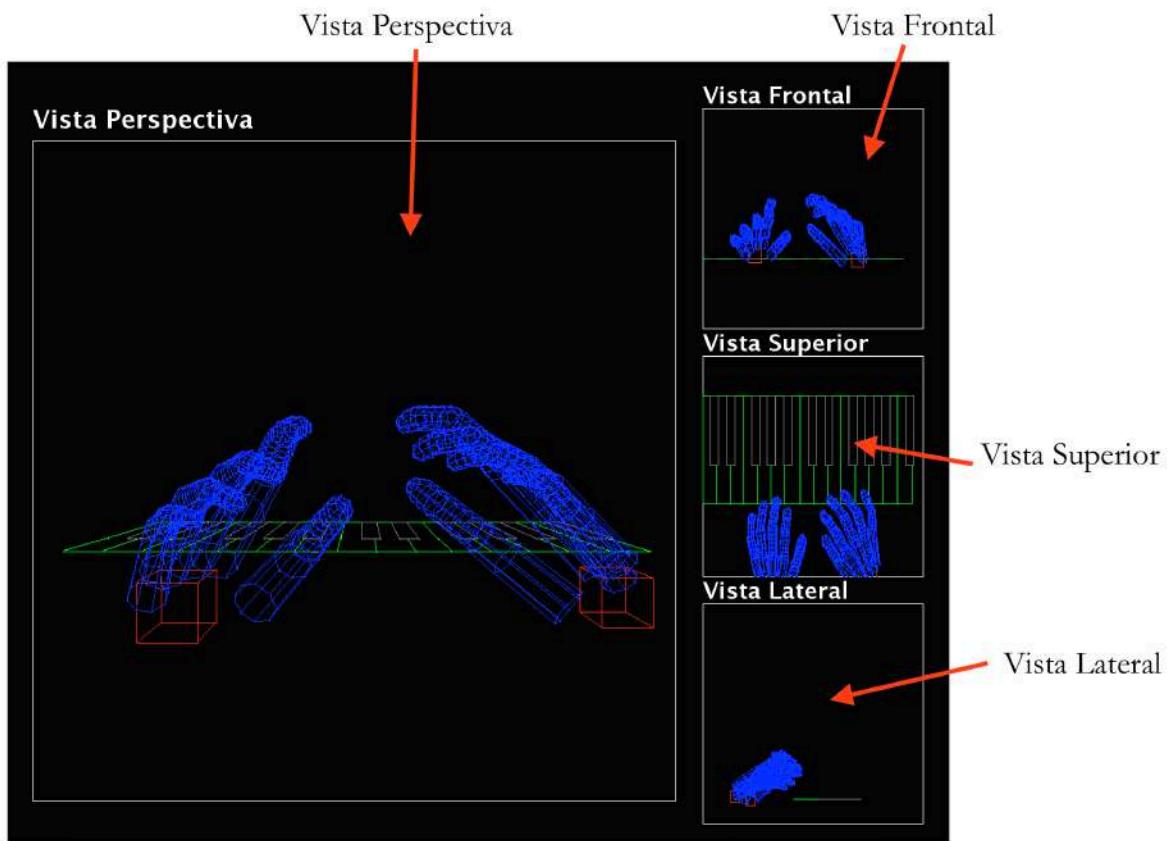


Figura N° 19: Composición gráfica de PanelEscenario

Fuente: Elaboración propia

Los tipos de gráficos que se realizan son:

- Dibujo de los huesos del dedo, representado por la clase `DibujoHueso`.
- Dibujo de la muñeca, representado por la clase `DibujoMuneca`.
- Dibujo de las teclas, representado por la clase `DibujoTecla`

Cada una de las clases que representan a los dibujos heredan de la clase `Malla` (de `MallasPoligonales`) y, por consecuencia, se componen de aristas y vértices.

Los vértices y las aristas se crean según los datos de posición del hueso, de la muñeca o de la tecla, sin embargo, internamente se utiliza el método `convProporcion` de la clase utilitaria `HerramientasVisuales`.

4.4 Conversión del sistema de coordenadas de representación virtual al visual

Para realizar la representación gráfica, se utiliza la idea de cámara de visualización (de Mallas Poligonales a partir de la que se sacan otras visualizaciones), la cual tiene su posición en el origen del sistema de coordenadas visual que, difiere del sistema de coordenadas de representación virtual (utilizado en el módulo de representación virtual).

En la Figura N° 20 se puede observar una representación gráfica de los sistemas de coordenadas visual y el sistema de coordenadas de Representación Virtual respecto del Escenario Virtual.

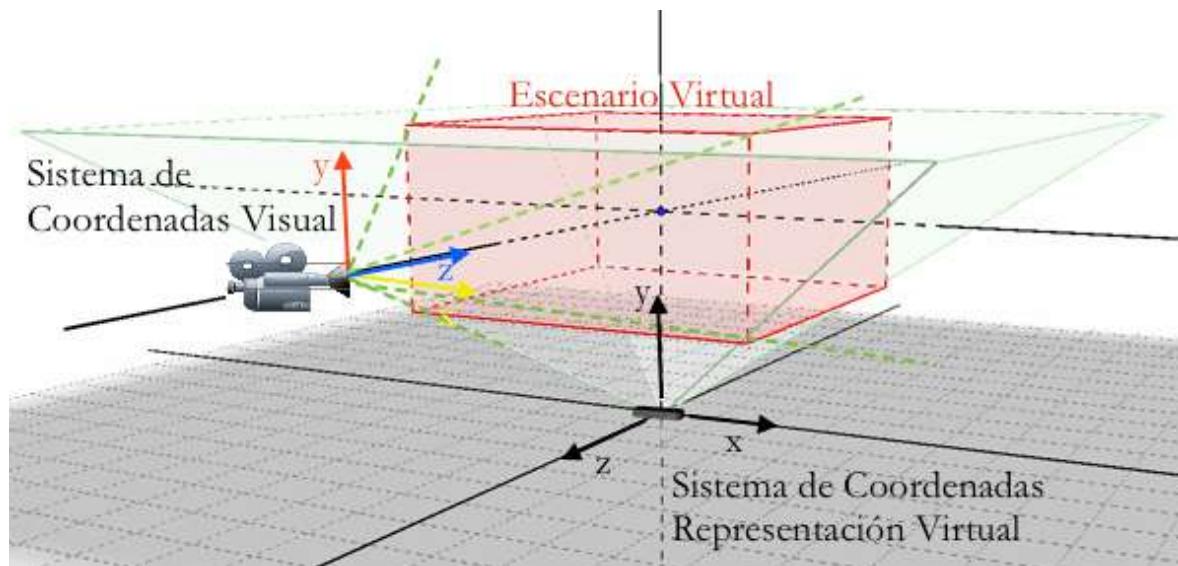


Figura N° 20: Sistema de Coordenadas Visual y de Sistema de Coordenadas Representación Virtual

Fuente: (Leap Motion) y Elaboración propia

Para realizar la conversión del sistema de coordenadas de representación virtual al sistema de coordenadas visual (método `convProporcion`) se resta 150 del componente y , se multiplica por -1 al componente z y se suma 300. Finalmente, se dividen los tres componentes entre 10.

La conversión fue determinada tomando en cuenta que, se deben restar 15 centímetros de la altura del Leap Motion (150 unidades en el Sistema de Coordenadas de Representación Virtual) para poder visualizar la representación gráfica desde un punto elevado, es decir, un componente que se encuentre en la altura de 15 centímetros en el eje y del Sistema de

Coordenadas Representación Virtual estará al nivel 0 del eje *y* del Sistema de Coordenadas Visual. También se tomó en cuenta que la altura de reconocimiento confiable por parte de Leap Motion es de 7 a 18 centímetros aproximadamente, por lo cual, estando el Sistema de Coordenadas Visual elevado 15 centímetros, facilita la visualización en la mayoría de las situaciones.

La multiplicación por -1 del valor del componente *z* de cada punto de objeto representado es debido a que el Sistema de Coordenadas de Representación Virtual respeta el sistema de coordenadas del SDK de Leap Motion y, por el contrario, el módulo de Mallas Poligonales que representa gráficamente a los objetos y que ha sido desarrollado por separado, tiene el eje *z* en dirección contraria.

La suma de 300 unidades (30 centímetros) al componente *z* de cada punto de objeto representado es debido a que el Sistema de Coordenadas de Representación Virtual se tiene que encontrar a una distancia similar a la distancia horizontal (eje *z*) de las manos a los ojos, para poder simular la distancia que existe entre la visión real de los ojos a las manos y el punto de captura de la cámara de visualización a la representación gráfica en pantalla.

4.5 Creación de DibujoHueso

La clase `DibujoHueso` tiene en su constructor los parámetros de `hueso` (tipo `Hueso`) y un `buffer` (de la clase `PGrafico`, se encuentra en `Mallas Poligonales`) necesario para realizar el dibujo.

Los vértices de la clase `DibujoHueso` se definen de la siguiente manera:

1. En caso de que el hueso sea tipo 1 (hueso metacarpo) de cualquier dedo o tipo 2 (proximal) del pulgar se define al atributo `separacionOctagonos` con el valor 30 (que representa 30 milímetros) y con el valor 6 (equivale a 6 milímetros) en caso de otro tipo de hueso.
2. Se define el valor de `radioBase` como la mitad del ancho hueso (atributo `ancho` que se obtiene mediante el método `getAncho` de `Hueso` dividido entre 2)
3. Se define una matriz base con los datos de los atributos `baseX`, `baseY`, `baseZ` de `Hueso`. Cabe recalcar que la `baseZ` que se usa en `DibujoHueso` se multiplica por -1 debido a que todo el módulo de Producción Visual utiliza el eje *z* al revés del

módulo Representación Virtual), se transporta la matriz base al punto de inicio del hueso (atributo posInicial de Hueso) y se definen los vértices según la separacionOctogonos, es decir, se transporta la matriz base a la posición de inicio del hueso, luego se transporta la matriz base cada 30 milímetros o cada 6 milímetros (dependiendo del valor de separacionOctogonos) y finalmente, se transporta la matriz base a la posición del final de hueso (posFinal de Hueso)

4. En cada distancia de valor separacionOctogonos se hace rotar 45 grados a la matriz 7 veces, de esta manera se consigue formar los vértices de un octógono alrededor del centro del dedo, cada uno a una distancia de valor radioBase.
5. Se procede a juntar los vértices con las aristas. Cada vértice de octógono se junta con el siguiente vértice y también con su equivalente en posición en el octógono anterior.

En la Figura N° 21 se puede observar una representación gráfica de la traslación de la matriz base a la posición de inicio del hueso y la definición del primer vértice.

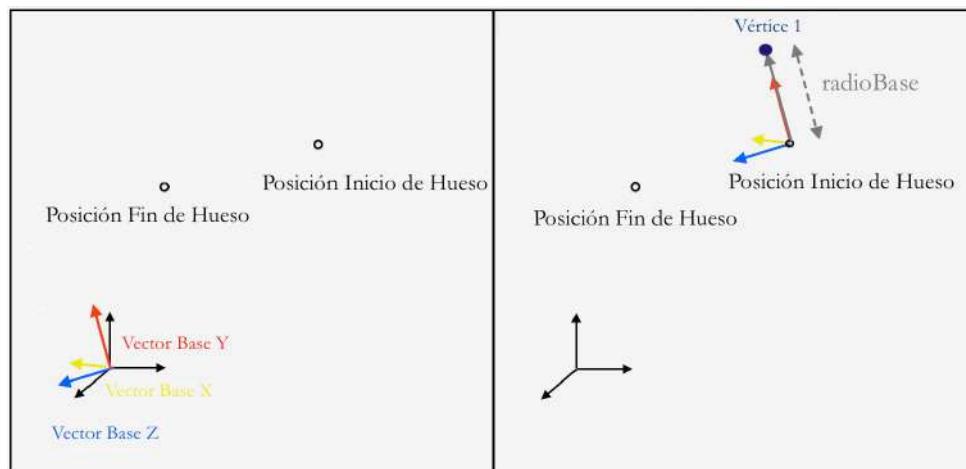


Figura N° 21: Traslación de matriz base a posición de inicio de Hueso – Definición de primer vértice

Fuente: Elaboración propia

En la Figura N° 22 se puede observar la representación gráfica del proceso para la definición de vértices en el primer ciclo a partir del segundo vértice y, en la imagen izquierda de la Figura N° 23 se puede observar la representación gráfica del final del primer ciclo de definición de vértices.

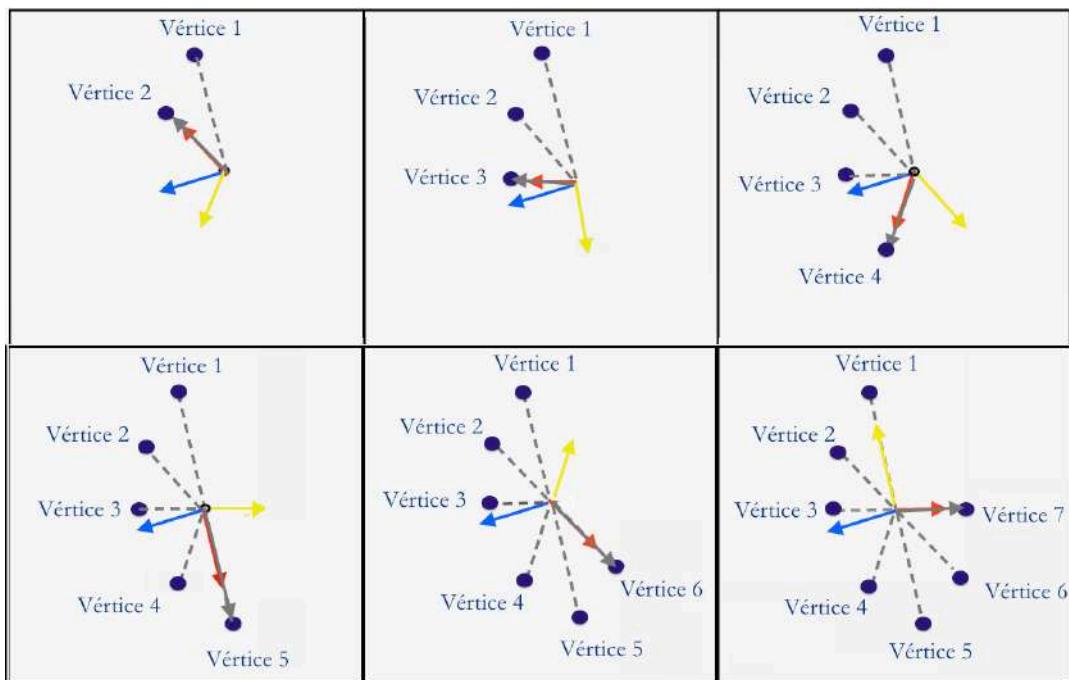


Figura N° 22: Proceso de definición de vértices desde Vértice 2 a Vértice 7

Fuente: Elaboración propia

En la imagen derecha de la Figura N° 23 se puede observar la representación del comienzo del segundo ciclo, es decir, la definición del noveno vértice, cuya posición se encuentra más cerca de la posición de la terminación del hueso (`posFinal` de Hueso).

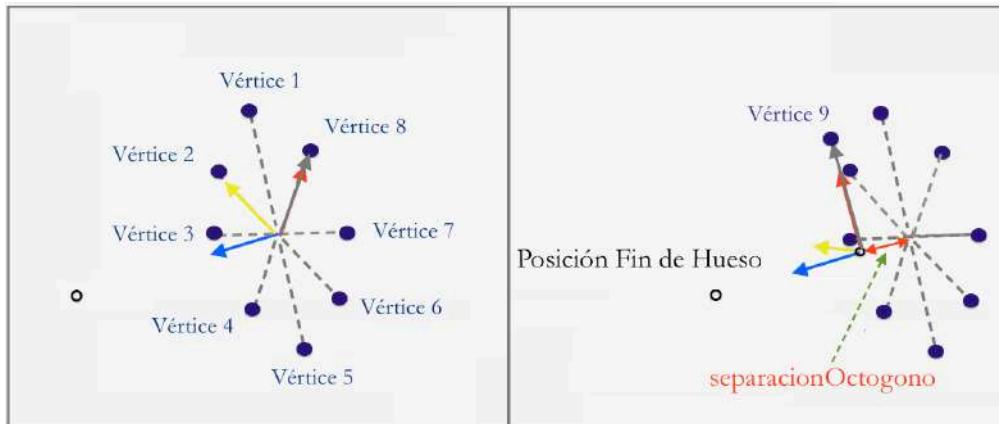


Figura N° 23: Fin de primer ciclo y comienzo de segundo ciclo de definición de vértices

Fuente: Elaboración propia

En la Figura N° 24 se puede observar la representación gráfica de un hueso modelo con todos los vértices definidos, desde la posición inicial del hueso (`posInicial`) hasta la posición final (`posFinal`).

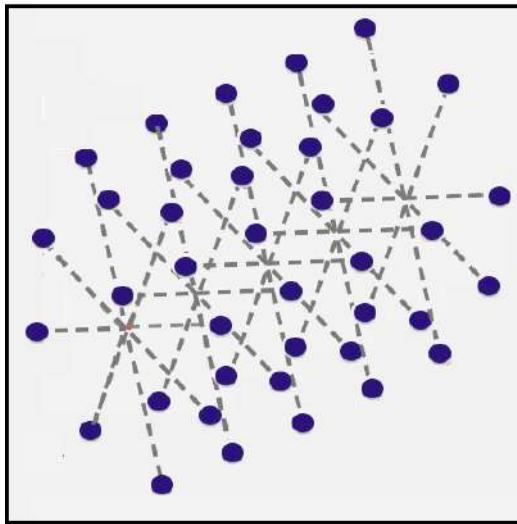


Figura N° 24: Representación de vértices de DibujoHueso

Fuente: Elaboración propia

En la Figura N° 25 se puede observar la representación gráfica de la definición de aristas dentro de los octógonos que contiene un hueso.

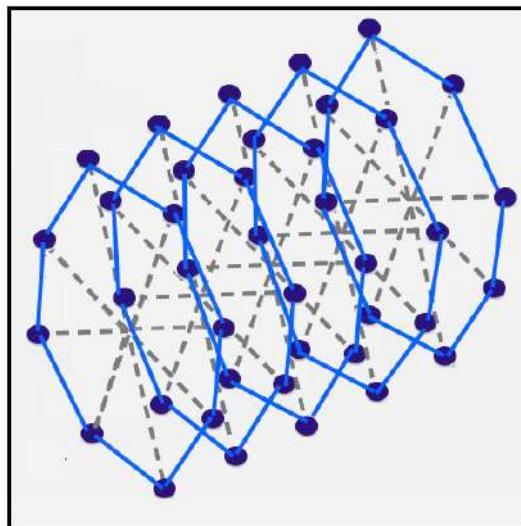


Figura N° 25: Representación de vértices y aristas de octógonos de DibujoHueso

Fuente: Elaboración propia

En la Figura N° 26 se puede observar la representación gráfica de la definición de todas las aristas entre los vértices de diferentes octógonos que componen el dibujo del hueso.

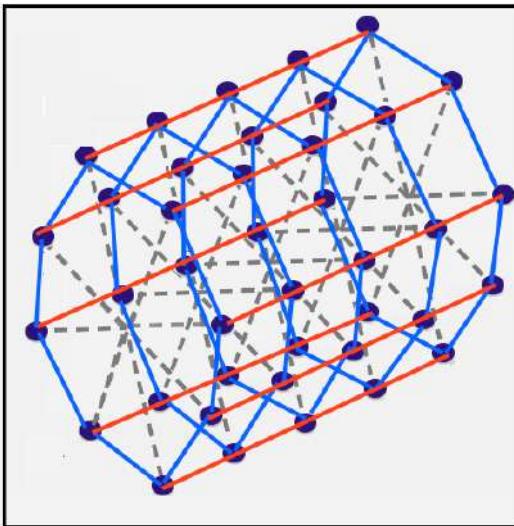


Figura N° 26: Representación de todos los vértices y aristas de DibujoHueso

Fuente: Elaboración propia

En la Figura N° 27 se puede observar la imagen que representa a un hueso proximal realizado con mallas poligonales.

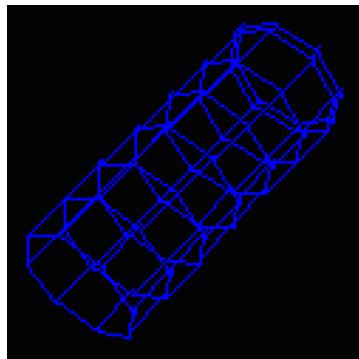


Figura N° 27: Dibujo que representa a un hueso proximal mediante mallas poligonales

Fuente: Elaboración propia

En caso de que DibujoHueso represente a un hueso distal (Hueso con el atributo tipo igual a 4) se asignan la posición de los vértices de diferente manera.

Como la punta del dedo corresponde a una forma aproximada a un paraboloide, se definió la separación entre octógono con un valor de 3 milímetros (separacionOctogonos de valor 3) para conseguir que se aproxime más a la forma y, que el radio del hueso disminuya en relación a la distancia de la punta (el radio es menor cuando se encuentra más cerca de la punta).

En la Figura N° 28 se puede observar una aproximación a paraboloides hecho con la aplicación *Grapher*, donde se ve que es una aproximación aceptable para representar la punta de los dedos.

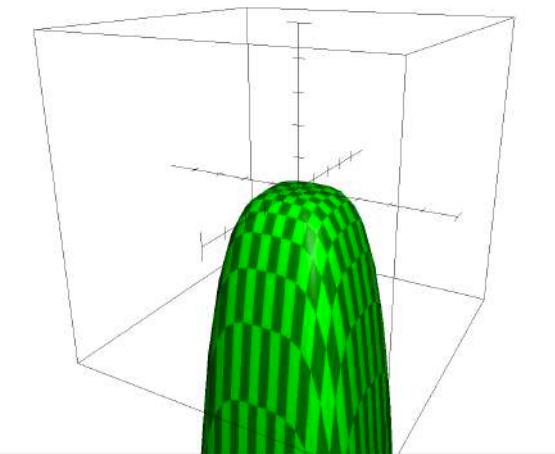


Figura N° 28: Imagen de paraboloides en la aplicación *Grapher*

Fuente: Elaboración propia

La ecuación que se utiliza para generar la imagen de la Figura N° 28 es

$$z = -(x^4 + y^4) \quad (5)$$

Para determinar el radio en caso de la representación de un hueso distal, se toma en cuenta que el radio r está en función de la longitud del hueso representado por A y el radio de la base del hueso representado por B representados en la Figura N° 29.

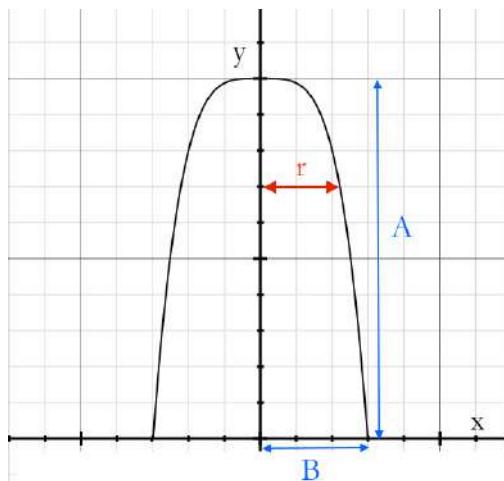


Figura N° 29: Esquema de representación del hueso distal en 2D

Fuente: Elaboración propia

Como se puede ver en la Figura N° 29, la función es de grado 4 que, gráficamente representa a una forma aproximada a una parábola convexa.

El valor de y a la altura B en x será 0 y, el valor de y en 0 de x será A , por lo que se puede deducir que el componente y de los puntos de la gráfica corresponden a las ecuaciones (6) y (7).

$$y = -Cx^4 + A \quad (6)$$

$$0 = -CB^4 + A \quad (7)$$

Por lo tanto, la ecuación que servirá para determinar el valor y será

$$y = A\left(-\frac{x^4}{B^4} + 1\right) \quad (8)$$

El método que se encarga de realizar los cálculos de posición de y , la ecuación (8), es `getRadioHuesoDistal` de la clase utilitaria `HerramientasVisuales`.

En la Figura N° 30 se puede observar el dibujo que representa a un hueso distal.

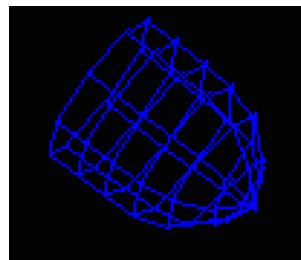


Figura N° 30: Dibujo que representa a un hueso distal mediante mallas poligonales

Fuente: Elaboración propia

Como la mano está compuesta principalmente de dedos y, los dedos a su vez están compuestos por huesos, cuando se dibujan todos los huesos que existen en el Escenario Virtual, se puede observar la forma de la mano.

En la Figura N° 31 se puede observar el dibujo completo de un dedo índice (Dedo con `tipoDedo` igual a 2) compuesto por cuatro huesos.

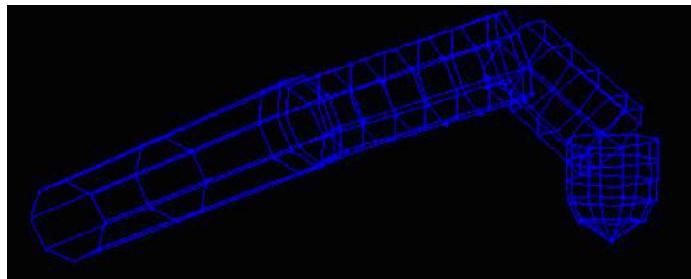


Figura N° 31: Dibujo que representa a un dedo índice mediante mallas poligonales

Fuente: Elaboración propia

En la Figura N° 32 se puede observar la representación de los dedos de la mano izquierda en seis diferentes posiciones.

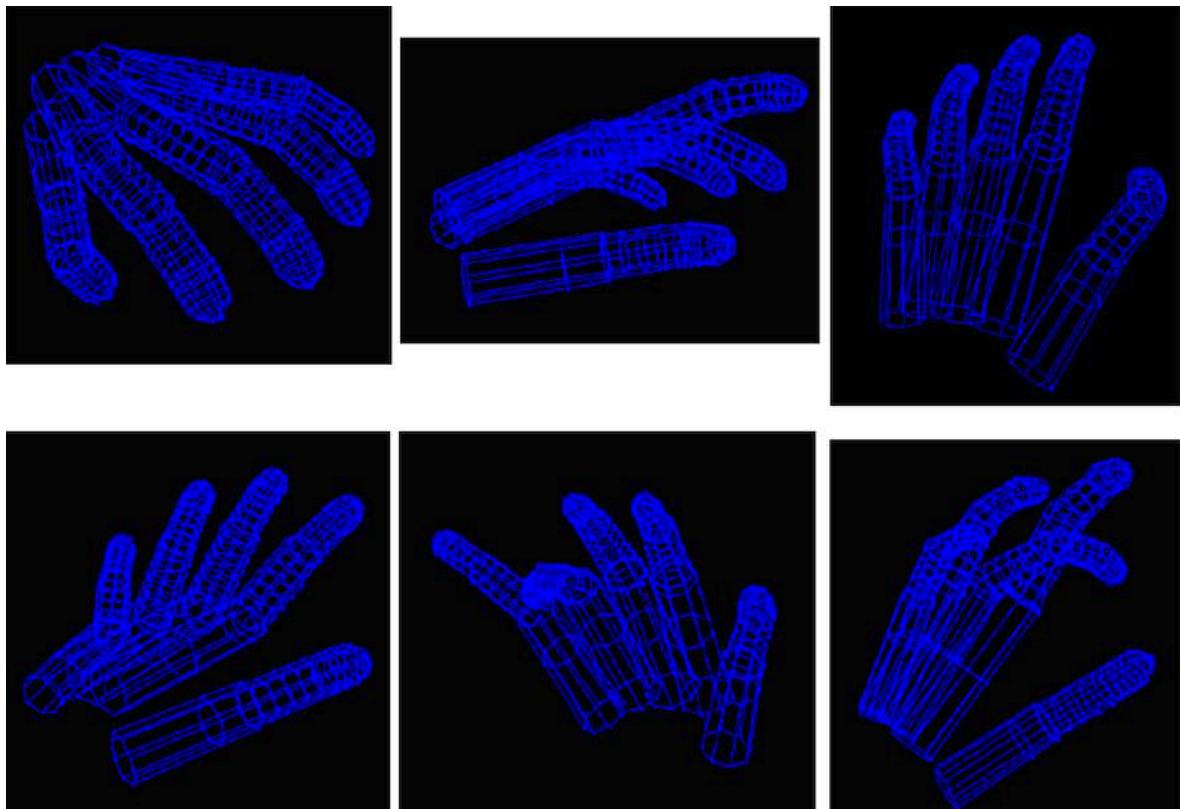


Figura N° 32: Representación de los dedos de la mano izquierda en seis posiciones

Fuente: Elaboración propia

4.6 Creación de DibujoMuneca

La clase DibujoMuneca representa gráficamente a la muñeca y tiene en su constructor los parámetros de entrada `posMuneca` (dato de la posición de la muñeca en el Escenario Virtual) y un `buffer` (de la clase PGrafico) necesario para realizar el dibujo.

El dibujo de la muñeca está predefinido y tiene la forma de un cubo de 2 centímetros a cada lado (atributo `lado`), siendo el centro de éste la posición dada en el parámetro del constructor.

Para representar la forma cúbica, `DibujoMuneca` tiene 8 vértices y 12 aristas que son creados en su constructor.

En la Figura N° 33 se puede observar el dibujo que representa a una muñeca.

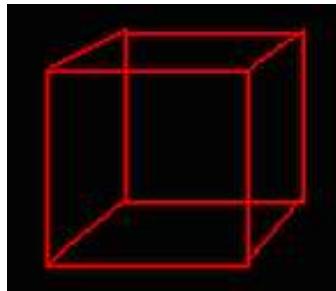


Figura N° 33: Representación gráfica de la muñeca

Fuente: Elaboración propia

4.7 Creación de DibujoTecla

La clase `DibujoTecla` representa gráficamente a una tecla (objeto `Tecla`) y tiene en su constructor los parámetros de entrada `tecla` (el objeto `Tecla` que representa gráficamente) y un `buffer` (de la clase `PGrafico`) necesario para realizar el dibujo.

El dibujo de la tecla será diferente en caso de las teclas blancas (tipo con valor `true`) y teclas negras (tipo con valor `false`). Si la tecla es blanca, el número de vértices predefinidos para el dibujo será 8, si la tecla es negra, el número de vértices predefinidos será 4.

Los datos de los vértices que se utilizan para graficar la tecla se encuentran en el atributo `estructura` de `Tecla`. Para registrar cada vértice se obtienen los datos (el número variará en caso de tecla blanca y tecla negra) y se convierten al eje de coordenadas visual con al método `convProporcion`. Posteriormente, se crean las aristas con los vértices existentes.

En la Figura N° 34 se puede observar la representación de 26 teclas que componen el teclado.

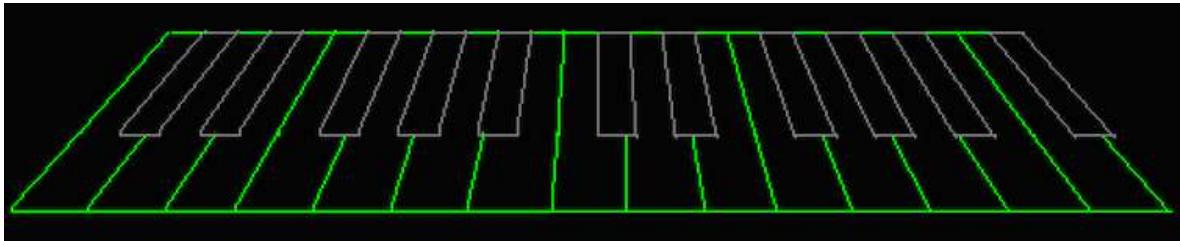


Figura N° 34: Representación gráfica de un teclado

Fuente: Elaboración propia

4.8 Funcionamiento de PanelEscenario

El constructor de PanelEscenario tiene entre sus parámetros a cImg (de la clase ControladorImagen que se le asigna al atributo controladorImagen) y a escIn (de la clase EscenarioVirtual que se le asigna al atributo escenario).

El atributo controladorImagen contiene los datos sobre los permisos de dibujo de los componentes que existen en el Escenario Virtual, es decir, se verifica el permiso de cada componente para poder ser dibujado. Las verificaciones se realizan mediante los métodos getPermisoRHMuñeca, getPermisoLHMuñeca, getPermisoTeclado y getPermisoDedo de la clase ControladorImagen. La activación o desactivación de los permisos se realizan en el PanelControlVisual (módulo de Producción Visual) y se registran en ControladorImagen (módulo de Modificación Virtual).

Como PanelEscenario contiene las imágenes que deben dar la ilusión de animación, el HiloVisual se encarga de activar su método update, que a su vez llama a repaint, cada 40 milisegundos (25 veces por segundo) lo que, permite dar el efecto deseado de animación.

Internamente, los métodos paint y repaint utilizan el método dibujarTodo y le pasan como parámetro un Graphic (que es una clase que permite acceder a métodos de dibujo en Java) y que los objetos Malla (u objetos de clases herederas de ésta) utilizan para crear PGrafico, que sirve como un buffer para realizar gráficos (dibujar líneas, evaluar posiciones, definir posición de ventanas de visualización).

El método dibujarTodo utiliza los métodos dibujarMarcos, dibujarMuñecas, dibujarHuesos y dibujarTeclado (que utiliza internamente el método

dibujarTecla) para realizar los dibujos de los marcos (letras y bordes de cuadros de visualización), muñecas, huesos de los dedos y las teclas respectivamente.

Los métodos dibujarMunecas, dibujarHuesos y dibujarTeclado utilizan las clases DibujoMuneca, DibujoHueso y DibujoTeclas respectivamente, pasando la información de los componentes de EscenarioVirtual a ser representados gráficamente (posición de la muñeca, huesos de los dedos y teclas) en los parámetros, además de PGrafico (que utiliza métodos de Graphic) que sirve para realizar los gráficos de Mallas Poligonales en Java.

Internamente, métodos dibujarMunecas, dibujarHuesos y dibujarTeclado también utilizan las clases ControladorImagen (que se mencionó previamente) y ControladorRotacionZoom, ambos pertenecientes al módulo Modificación Virtual, para verificar la posición de las imágenes respecto a la cámara de visualización (ver Anexo 2) y verificar el permiso de dibujo de cada componente del Escenario Virtual a ser representado.

4.9 Composición y funcionamiento de PanelControlTeclado

La clase PanelControlTeclado hereda de JPanel, contiene:

- Un *checkbox* (atributo cbTeclado) para habilitar o deshabilitar el teclado que utiliza los métodos crearNuevoTeclado y eliminarTeclado de la clase ManejoControlTeclado (del módulo Modificación Virtual) respectivamente.
- Seis botones (atributo btnControl), dos por cada eje, ya que representan a las direcciones posibles de movimiento. Estos botones, al ser presionados por el usuario, activan el método moverTeclado de la clase ManejoControlTeclado (del módulo Modificación Virtual) para mover la posición del teclado.
- Una imagen que indica la posición positiva de los ejes respecto al Leap Motion como ayuda de ubicación espacial al usuario
- Una etiqueta (atributo lblPosTeclado) que muestra la posición (los 3 componentes) del teclado virtual (posición de inicio de la primera tecla a la izquierda) si es que existe alguno en el Escenario Virtual.

En la Figura N° 35 se puede observar la imagen de PanelControlTeclado.

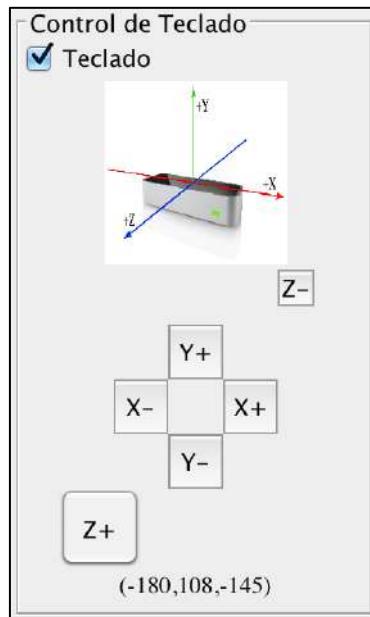


Figura N° 35: Imagen de PanelControlTeclado

Fuente: Elaboración propia

4.10 Composición y funcionamiento de PanelControlVisual

La clase PanelControlVisual hereda de JPanel, contiene 13 checkboxes para habilitar o deshabilitar el permiso de dibujo de los componentes en el PanelEscenario. Cada checkbox representa a un elemento del Escenario Virtual, siendo 10 los checkboxes asignados a los 10 dedos de ambas manos, 2 a las muñecas de ambas manos y 1 al teclado.

En la Figura N° 36 se puede observar la imagen de PanelControlVisual.



Figura N° 36: Imagen de PanelControlVisual

Fuente: Elaboración propia

Cuando el usuario habilita los *checkboxes*, se activan los métodos (dependiendo cual sea el componente) `habilitarDedo`, `habilitarMuneca` o `habilitarTeclado` de la clase `ManejoVisual` (del módulo Modificación Virtual). Estos métodos hacen que se habilite el permiso de dibujo (registrados en `ControladorImagen`) de los componentes seleccionados para poder mostrarlos en `PanelEscenario`.

Cuando el usuario deshabilita los *checkboxes*, se usan los métodos `deshabilitarDedo`, `habilitarMuneca` o `deshabilitarTeclado` de la clase `ManejoVisual`. Estos métodos hacen que se deshabilite el permiso de dibujo (registrados en `ControladorImagen`) de los componentes seleccionados, por lo tanto, éstos no se muestran en `PanelEscenario`.

4.11 Composición de PanelArchivo

La clase `PanelArchivos` hereda de `JPanel` y se compone de:

- Un botón (atributo `btGrabacion` de clase `JButton`) que permite al usuario iniciar o parar una grabación.
- Un botón (atributo `btCargarArchivo` de clase `JButton`) que permite al usuario cargar un archivo (para su reproducción, exportación a archivo Excel o generación de cuadros estadísticos) o quitar el archivo previamente cargado.
- Un botón (atributo `btExportarDatos` de clase `JButton`) que permite al usuario exportar los datos a formato Excel. Este botón es activado solamente cuando un archivo es previamente cargado.
- Una etiqueta (atributo `lblArchivoCargado` de clase `JLabel`) que muestra al usuario el nombre del archivo cargado (si es que algún archivo ha sido cargado previamente).
- Un panel (atributo `pReproducción`, de la clase `PanelReproduccion`) que contiene los botones y etiquetas que permiten al usuario reproducir, pausar, parar, aumentar o disminuir la velocidad de reproducción y visualizar datos relevantes de la reproducción de un archivo (tiempo transcurrido y velocidad de reproducción).
- Un panel (atributo `pEstadistica`, de la clase `PanelEstadistica`) que contiene los botones que permiten al usuario crear los cuadros estadísticos.

En la Figura N° 37 se puede observar la composición gráfica de PanelArchivo.

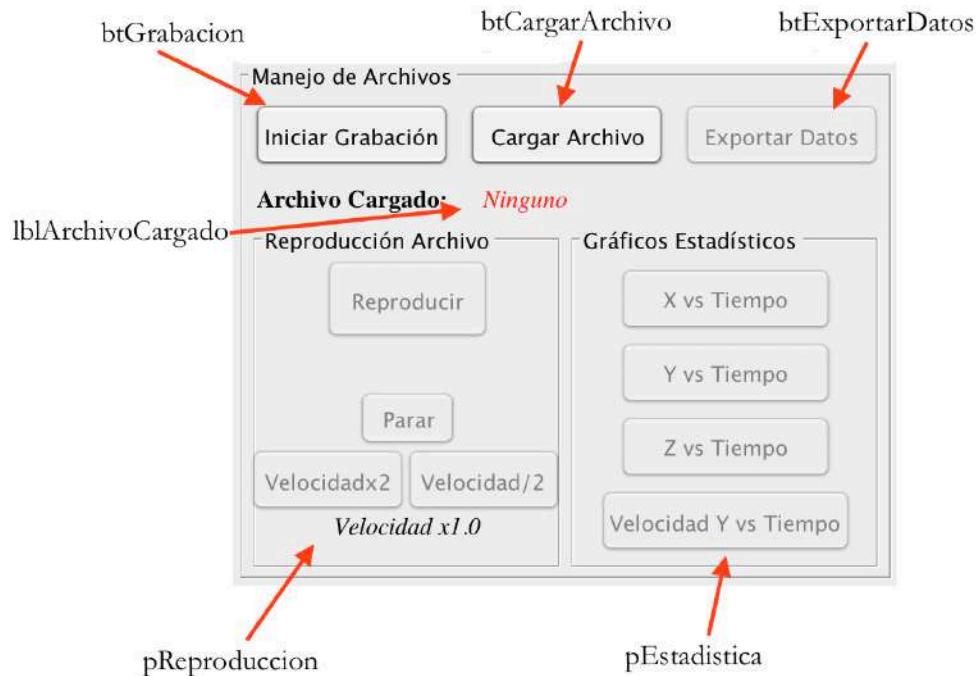


Figura N° 37: Composición gráfica de PanelArchivo

Fuente: Elaboración propia

4.12 Funcionamiento de btGrabacion

El botón de grabación (atributo btGrabacion) puede contener dos tipos de texto, “Iniciar Grabación” y “Parar Grabación”.

Al iniciar el programa, por defecto está habilitado el texto “Iniciar Grabación”, lo que brinda al usuario la posibilidad de iniciar la grabación haciendo clic en el botón.

Si el usuario hace clic en btGrabacion cuando contiene el texto “Iniciar Grabación”, activa el método iniciarGrabacion de la clase ManejoDatoArchivo (del módulo Manejo de Archivos) que procede a iniciar la grabación de los datos que existen en el Escenario Virtual. Además, se utiliza el método activarConfigGrabacion que hará inaccesibles algunos botones y hará accesible otros, también cambiará el texto del propio btGrabacion a “Parar Grabación”.

En la Figura N° 38 se puede observar a PanelArchivo con la configuración de grabación (realizado en método activarConfigGrabacion) y en la Figura N° 39 se puede observar

a PanelArchivo sin la configuración de grabación (realizado con el método desactivarConfigGrabacion), que también es la configuración por defecto.



Figura N° 38: Imagen de PanelArchivo con la configuración de grabación

Fuente: Elaboración propia



Figura N° 39: Imagen de PanelArchivo con la configuración por defecto

Fuente: Elaboración propia

Si el usuario hace clic en btGrabacion cuando contiene el texto “Parar Grabación”, activa el método pararGrabacion de la clase ManejoDatoArchivo que procede a parar la grabación que estaba realizando, éste, a su vez, utiliza el método guardarGrabacionFileDialog que muestra al usuario una ventana (FileDialog) para elegir la ubicación y el nombre del archivo a ser guardado con los datos de la grabación.

En la Figura Nº 40 se puede observar al `FileDialog` utilizado en el método `guardarGrabacionFileDialog`

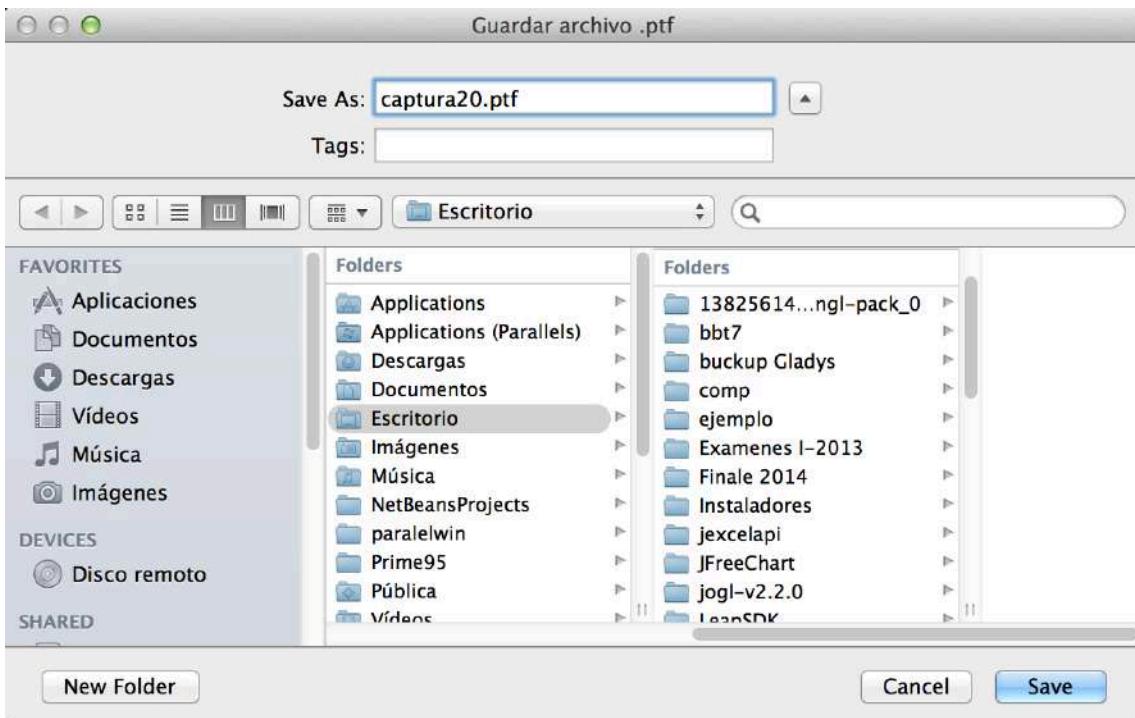


Figura Nº 40: Imagen del `FileDialog` utilizado en método `guardarGrabacionFileDialog` de la clase `PanelArchivo`

Fuente: Elaboración propia

En caso que se realice el guardado del archivo correctamente, se utiliza el método `mmArchivoGuardado` que muestra al usuario una ventana (`MessageDialog`) que avisa que la operación de guardado ha sido realizada correctamente.

La extensión de guardado de archivos es `.ptf`, definido así por las letras iniciales de *Piano Technique File* (traducido del Inglés: Archivo de Técnica de Piano) como un método de reconocimiento rápido de formato por parte del sistema. Actualmente la extensión `.ptf` es utilizada también en los archivos de Scrapbook Flair Template de Aurora Digital Imaging, PSP Theme File de Sony y Pro Tools 7 Session File de Avid Technology.

En la Figura Nº 41 se puede observar el `MessageDialog` que utiliza el método `mmArchivoGuardado` cuando un archivo de nombre “`captura20.ptf`” ha sido guardado correctamente.

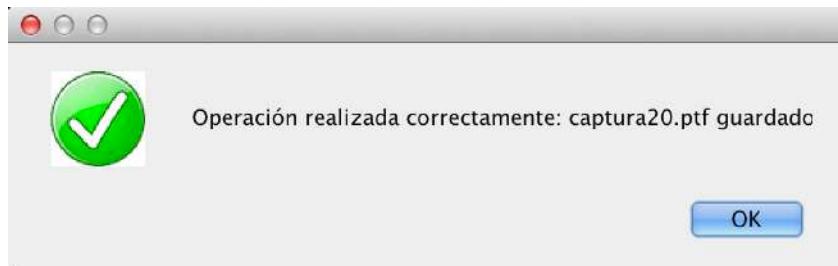


Figura N° 41: Imagen de `MessageDialog` utilizado en método `mmArchivoGuardado` de la clase `PanelArchivo`

Fuente: Elaboración propia

En caso que el usuario utilice un nombre de archivo que ya existe, el método `evaluarExistenciaArchivo` de la clase `HiloGrabacion` (módulo Manejo de Archivos) utiliza el método `mmSobrescribirArchivo`, que muestra un `OptionDialog`, dando la opción de confirmar la acción de sobrescribir del archivo o cancelar dicha acción.

En la Figura N° 42 se puede observar el `OptionDialog` de confirmación utilizado en `mmSobrescribirArchivo`, el archivo tiene el nombre “*captura20.ptf*”.

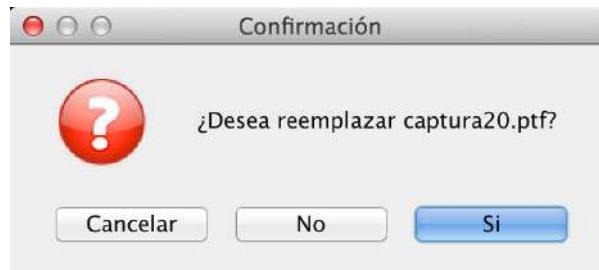


Figura N° 42: Imagen de `OptionDialog` utilizado en método `mmArchivoGuardado` de la clase `PanelArchivo`

Fuente: Elaboración propia

En caso de que la extensión sea cambiada y se coloque otro tipo de extensión (por defecto es tipo *.ptf* y agregado internamente en caso que no se coloque ninguna extensión), el método `evaluarExtensionArchivo` de la clase `HiloGrabacion` (módulo Manejo de Archivos) utiliza el método `mmExtensionRechazada`, que muestra un `MessageDialog` informando al usuario que la extensión colocada es inválida y, posteriormente vuelve a mostrar el `FileDialog` de `guardarGrabacionFileDialog`.

En la Figura N° 43 se puede observar el `MessageDialog` que informa al usuario que la extensión debe ser “`.ptf`”.

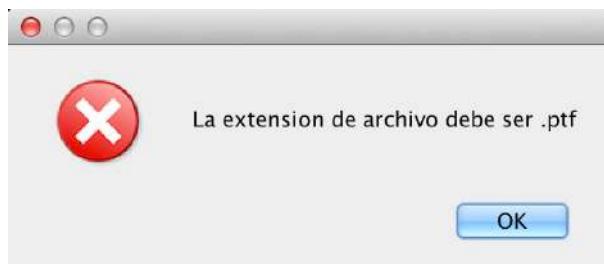


Figura N° 43: Imagen de `MessageDialog` utilizado en método `mmExtensionRechazada` de la clase `PanelArchivo`

Fuente: Elaboración propia

Al finalizar la grabación, se utiliza el método `desactivarConfigGrabacion` que revertirá los cambios realizados en `activarConfigGrabacion`.

4.13 Funcionamiento de `btCargarArchivo`

El botón de cargado de archivo (atributo `btCargarArchivo`) puede contener dos tipos de texto, “Cargar Archivo” y “Quitar Archivo”.

Al iniciar el programa, por defecto está habilitado el texto “Cargar Archivo”, lo que brinda al usuario la posibilidad de cargar un archivo (para su reproducción, generación de cuadros estadísticos o exportación de datos a Excel) haciendo clic en el botón.

Si el usuario hace clic en `btCargarArchivo` cuando contiene el texto “Cargar Archivo” inicia el método `cargarArchivo` de la clase `ManejoDatoArchivo` (del módulo Manejo de Archivos) que, a su vez, utiliza el método `cargarArchivoFileDialog` (que utiliza un `FileDialog` con filtro para archivos de extensión “`.ptf`” para que el usuario elija un archivo de extensión válida para cargar al sistema) y el método `activarConfigArchivoCargado` que hará inaccesibles algunos botones y etiquetas y, hará accesible otros (activa los métodos del mismo nombre en `PanelReproduccion` y `PanelEstadistica`), también cambiará el texto del propio `btGrabacion` a “Quitar Archivo”.

En la Figura N° 44 se puede observar a PanelArchivo con la configuración de archivo cargado (realizado en método activarConfigArchivoCargado), en este caso de nombre “*captura1Abril.ptf*”.



Figura N° 44: Imagen de PanelArchivo con la configuración de archivo cargado

Fuente: Elaboración propia

En la Figura N° 45 se puede observar al FileDialog utilizado en el método cargarArchivoFileDialog.

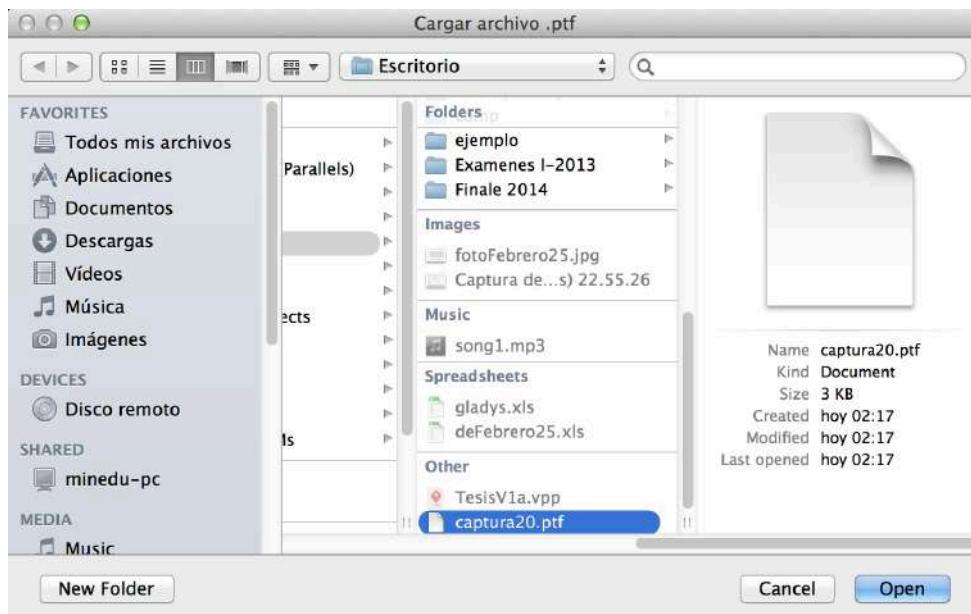


Figura N° 45: Imagen del FileDialog utilizado en método cargarArchivoFileDialog de la clase PanelArchivo

Fuente: Elaboración propia

Cuando un archivo es cargado correctamente, se utiliza el método `mmArchivoCargado` que muestra al usuario una ventana (`MessageDialog`) que avisa que la operación de cargado de archivo ha sido realizado correctamente.

En la Figura N° 46 se puede observar el `MessageDialog` que utiliza el método `mmArchivoCargado` cuando un archivo de nombre “*captura20.ptf*” ha sido cargado correctamente.



Figura N° 46: Imagen de `MessageDialog` utilizado en método `mmArchivoCargado` de la clase `PanelArchivo`

Fuente: Elaboración propia

4.14 Funcionamiento de `btExportarDatos`

El botón de exportación de datos (atributo `btExportarDatos`) es habilitado cuando se carga un archivo, como se puede ver en el funcionamiento de `btCargarArchivo`.

Cuando el usuario hace clic en `btExportarDatos`, se utiliza el método `exportarDatos` de la clase `ManejoDatosArchivo`, que a su vez utiliza el método `crearFrameExportarDatos` donde se instancia la clase `FrameExportarDatos`. También se utiliza el método `desactivarPermisoFrameDatos` (que hace que el atributo `frameExportarDatosInstanciado` sea `false` para evitar que se realicen varias instancias de `FrameExportarDatos`) y se desactiva el botón `btCargarArchivo` hasta que se cierre el objeto de la clase `FrameExportarDatos`.

4.15 Composición y funcionamiento de `FrameExportarDatos`

La clase `FrameExportarDatos` hereda de la clase `JFrame`, contiene un botón y tres paneles (creados con los métodos `crearPIntervaloCaptura`, `crearPFiltroComponentes` y `crearPFiltroDato` utilizados en el constructor).

-
- Panel de intervalo de captura de datos (atributo pIntervaloCaptura). Se utiliza para mostrar al Usuario las opciones y para que éste elija el espacio entre capturas de datos, siendo el intervalo predeterminado de 20 milisegundos, con un mínimo espacio de 20 milisegundos entre captura y un máximo de 1 segundo. Para que el Usuario elija las opciones de intervalo de captura, se utiliza a s1EspacioCaptura (de la clase JSlider) y, para mostrar al usuario el valor de la selección que realiza, se utiliza a lblValorSeleccion (de la clase JLabel).
 - Panel de filtro de componentes (pFiltroComponentes). Contiene 12 checkboxes (de la clase JCheckBox) que, corresponden a los 10 dedos de las manos y a las 2 muñecas, siendo agrupados por tipo de mano en los atributos cbRHComponentes y en cbLHComponentes. El usuario puede activar y desactivar cada checkbox (por defecto están todos activados) para que se realice o no se realice la exportación de los datos correspondientes a cada componente.
 - Panel de filtro de datos (atributo pFiltroDatos). Contiene 4 checkboxes (de la clase JCheckBox y se encuentran en el atributo cbDatos) que, corresponden a los 4 tipos de datos que se registran en un archivo: la posición respecto a los 3 ejes y la velocidad en el eje y. El usuario puede activar o desactivar cada checkbox (por defecto están todos activados) para que se realice o no la exportación de los tipos de dato seleccionados.
 - Botón exportar a Excel (atributo btExportar, de la clase JButton) que, cuando el usuario hace clic, se activa el método exportarDatosAExcel de la clase ManejoExportarDatos (módulo de Manejo de Archivos). Éste método realiza el procedimiento para crear un archivo Excel con los datos seleccionados en los paneles pIntervaloCaptura, pFiltroComponentes y pFiltroDatos.

En la Figura N° 47 se puede observar al de FrameExportarDatos utilizado para exportar datos del archivo de nombre “*captura1Abril.ptf*”.

El método exportarDatosAExcel de la clase ManejoExportarDatos, utiliza el método guardarArchivoExcelFileDialog que, muestra al usuario un FileDialog para que elija dónde y con qué nombre guardar el archivo Excel.

En la Figura Nº 48 se puede observar al FileDialog utilizado en guardarArchivoExcelFileDialog.



Figura Nº 47: Imagen de FrameExportarDatos

Fuente: Elaboración propia

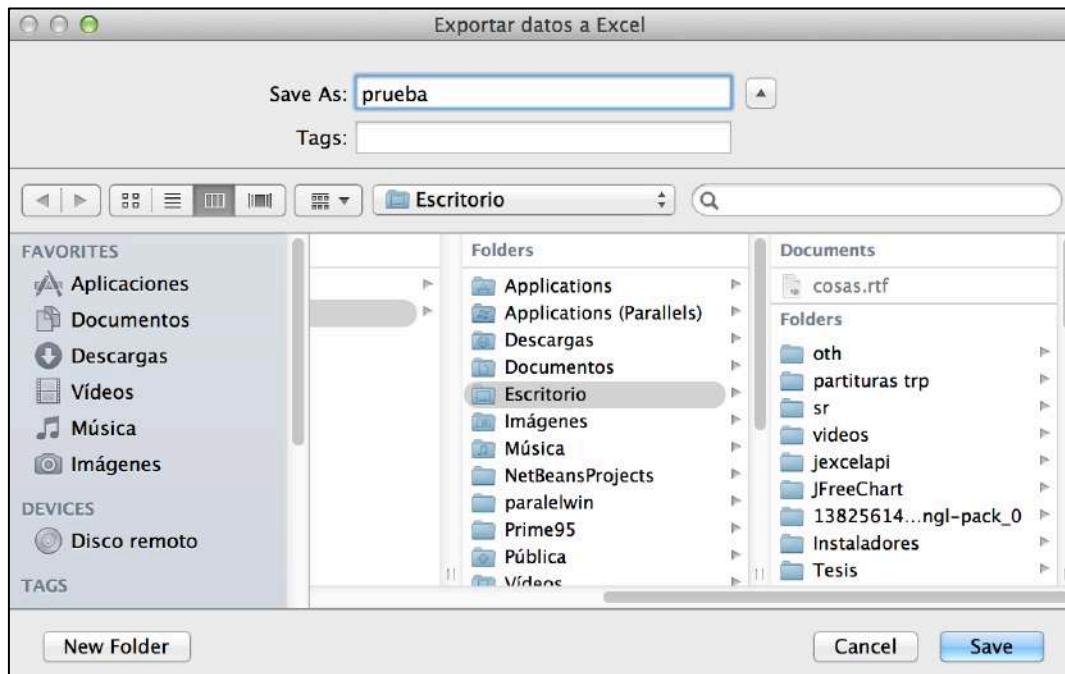


Figura Nº 48: Imagen del FileDialog utilizado en método guardarArchivoExcelFileDialog de la clase FrameExportarDatos

Fuente: Elaboración propia

En caso que se realice la creación del archivo Excel con éxito, se utiliza el método `mmArchivoEscrito` que muestra al usuario una ventana (`MessageDialog`), la cual avisa que la operación de creación de archivo Excel y la exportación de datos ha sido realizada correctamente.

En la Figura Nº 49 se puede observar el `MessageDialog` que utiliza el método `mmArchivoEscrito` cuando un archivo de nombre “`pruebaExportar.xls`” ha sido correctamente creado y escrito con los datos de exportación.

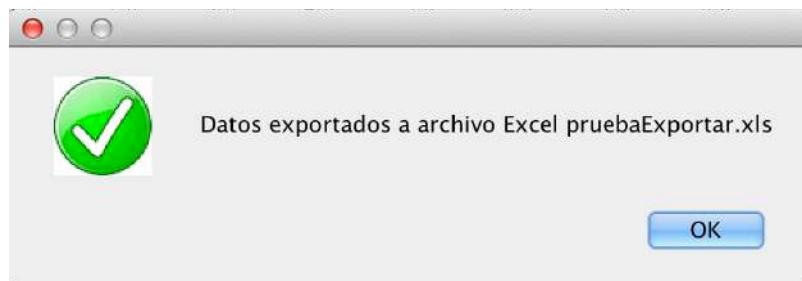


Figura Nº 49: Imagen de `MessageDialog` utilizado en método `mmArchivoEscrito` de la clase `FrameExportarDatos`

Fuente: Elaboración propia

En caso que el usuario utilice un nombre de archivo que ya existe, el método `evaluarExistenciaArchivo` de la clase `ManejoExportarDatos` (módulo `Manejo de Archivos`) utiliza el método `mmSobrescribirArchivo`, que muestra un `OptionDialog`, el cual da la opción de confirmar la acción de sobrescribir el archivo o cancelar dicha acción.

En la Figura Nº 50 se puede observar el `OptionDialog` de confirmación utilizado en `mmSobrescribirArchivo`, el archivo tiene el nombre “`pruebaExportar.xls`”.



Figura Nº 50: Imagen de `OptionDialog` utilizado en método `mmSobrescribirArchivo` de la clase `FrameExportarDatos`

Fuente: Elaboración propia

En caso de que la extensión sea cambiada y se coloque otro tipo de extensión (por defecto es tipo `.xls` y se agregado internamente en caso que no se coloque ninguna extensión), el método `evaluarExtensionArchivo` de la clase `ManejoExportarDatos` (módulo Manejo de Archivos) utiliza el método `mmExtensionRechazada`, que muestra un `MessageDialog` informando al usuario que la extensión colocada es inválida y, posteriormente vuelve a mostrar al `FileDialog` de `guardarArchivoExcelFileDialog`.

En la Figura N° 51 se puede observar el `MessageDialog` que informa al usuario que la extensión debe ser “`.xls`”.



Figura N° 51: Imagen de `MessageDialog` utilizado en método `mmExtensionRechazada` de la clase `FrameExportarDatos`

Fuente: Elaboración propia

En caso de que se produzca algún error de lectura del archivo `.ptf`, se le notifica al usuario que ocurrió un error mediante un `MessageDialog`, generado por el método `mmErrorLeyendoArchivo`.

En la Figura N° 52 se puede observar el `MessageDialog` generado en el método `mmErrorLeyendoArchivo`.

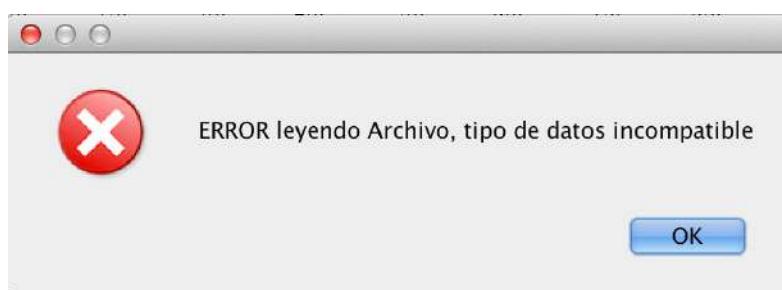


Figura N° 52: Imagen de `MessageDialog` utilizado en método `mmErrorLeyendoArchivo` de la clase `FrameExportarDatos`

Fuente: Elaboración propia

Cuando FrameExportarArchivo realiza el proceso de exportación de datos a Excel y notifica al usuario del éxito, en caso de que el usuario cierre FrameExportarArchivo o en caso de que exista un error de lectura de archivo, se utiliza el método cerrarFrame.

El método cerrarFrame utiliza activarPermisoFrameExportarDatos de PanelArchivo, que habilita el botón btCargarArchivo y activa el permiso (frameExportarDatosInstanciado con valor true) para poder instanciar un nuevo FrameExportarArchivo.

4.16 Composición y funcionamiento de PanelEstadistica

La clase PanelEstadistica hereda de JPanel y dentro del sistema, su instanciación se encuentra en un objeto PanelArchivo como atributo pEstadistica.

Contiene 4 botones que, permiten al usuario crear los gráficos estadísticos utilizando los datos de un archivo cargado previamente haciendo clic en alguno de éstos. Los botones se encuentran en el atributo btPosicionVelocidadVsTiempo (es un vector de cuatro JButton) y, cuando el usuario hace clic en alguno de éstos, utilizan el método crearFrameEstadistica.

Los botones de PanelEstadistica son habilitados con el método activarConfigArchivoCargado sólo cuando un archivo ha sido cargado previamente y, son deshabilitados con el método desactivarConfigArchivoCargado cuando no hay ningún archivo cargado.

En la Figura N° 53 se puede observar a PanelEstadistica.



Figura N° 53: Imagen de PanelEstadistica

Fuente: Elaboración propia

El método `crearFrameEstadistica` instancia un `FrameEstadistica` indicándole en un parámetro (`tipoDato tipo int`) el tipo de dato que debe tomar en cuenta para crear el gráfico estadístico. Los tipos de dato que se toman en cuenta para crear el `FrameEstadistica` son los valores de x , y , z de las posiciones y la velocidad en y de los 10 dedos y 2 muñecas almacenados en un archivo `.ptf`.

Para regular la instanciaciones de los `FrameEstadistica` se utiliza el parámetro `frameEstadisticaInstanciado`, que es vector de cuatro `boolean`, donde cada uno contiene el valor `true` si un `FrameEstadistica` está instanciado y `false` en caso contrario. De esta manera, se puede instanciar un tipo de `FrameEstadistica` sólo una vez al mismo tiempo.

4.17 Composición y funcionamiento de `FrameEstadistica`

La clase `FrameEstadistica` hereda de la clase `JFrame`. Sirve para mostrar los gráficos estadísticos y brinda opciones sobre manejo de gráficos (exportación de imagen, acercamiento, alejamiento de imagen) al usuario. Contiene elementos (botones, *checkboxes* y gráficos) dentro de tres paneles:

- Panel de gráfico estadístico (atributo `pGraficoEstadistico`). Contiene el gráfico estadístico creado a partir del tipo de datos que selecciona el usuario en el botón que genera un `FrameEstadistica`. Se crea en el método `crearChartPanel`, utilizado en el constructor de `FrameEstadistica`, en el que interviene el método `crearDataSet` de la clase `ManejoEstadistica` (del módulo Manejo de Archivos) donde asigna (y utiliza otros métodos del módulo Manejo de Archivos) los datos leídos de un archivo `.ptf` a un `DataSet` necesario para la creación de un `ChartPanel`.
- Panel de selección de componentes (atributo `pComponentesMano`). Se crea en el método `crearPComponentesMano` que se utiliza en el constructor de `FrameEstadistica`. Contiene 12 *checkboxes* (con valor `true` por defecto) que representan a los 10 dedos de la manos y a las 2 muñecas, divididas por el tipo de mano en los atributos `cbRHComponentes` y `cbLHComponentes`. Cuando se cambia el valor de un *checkbox* se utiliza el método `setVisibilidad` de la clase

ManejoEstadistica (del módulo Manejo de Archivos), que hace visible o invisible (según el valor del parámetro `valor` tipo boolean) los dibujos (línea en el gráfico en el panel `pGraficoEstadistico`) del componente al cual corresponde el *checkbox*.

- Panel de opciones (atributo `pOpciones`). Se crea en el método `crearPOpciones` que se utiliza en el constructor de `FrameEstadistica`. Contiene los botones `btRestaurarZoom` y `btGuardarImagen` que, respectivamente, brindan al usuario la opción de restaurar el zoom por defecto del gráfico estadístico (en el que el usuario puede realizar un zoom gracias a las propiedades de los `ChartPanel` de la biblioteca `JFreeChart`) usando el método `restoreAutoBounds` (de la clase `ChartPanel`) y guardar en un JPG la imagen de gráfico estadístico obtenido usando el método `guardarJPG` de la clase `ManejoEstadistica` (del módulo Manejo de Archivos) mediante un clic.

En la Figura N° 54 se puede observar a un `FrameEstadistica`.

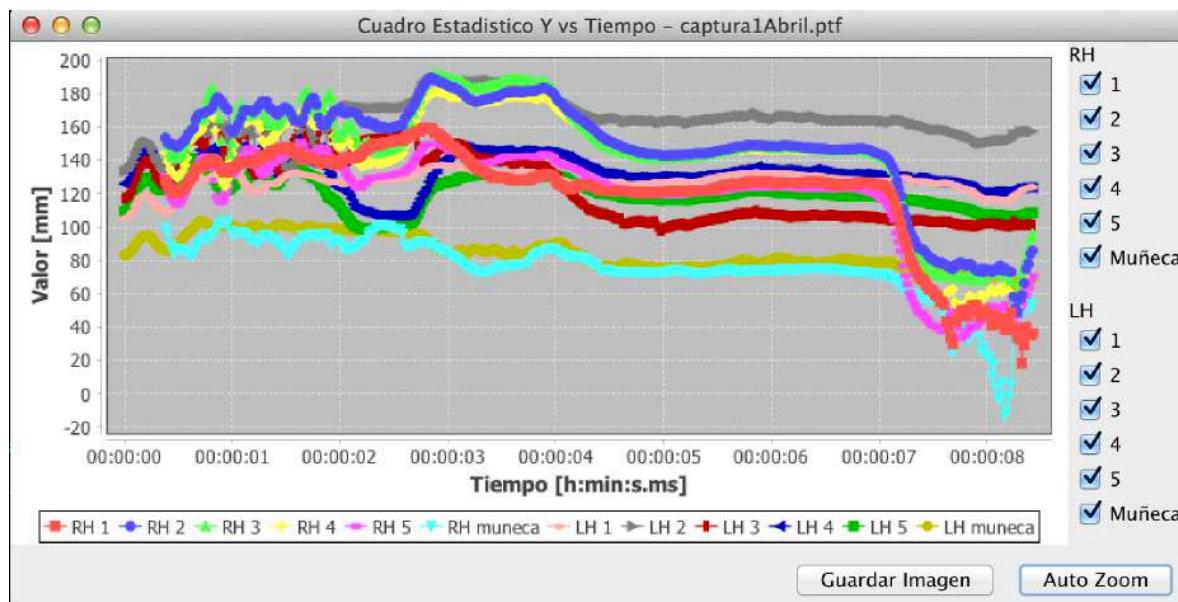


Figura N° 54: Imagen de `FrameEstadistica`

Fuente: Elaboración propia

En el caso de que se produzca un error al crear un `FrameEstadistica` (debido a error de lectura de archivo) se utiliza el método `mmErrorLeyendoArchivo` para mostrar un

MessageDialog al usuario comunicando el error. En la Figura Nº 52 se puede observar el MessageDialog mostrado al usuario con el método `mmErrorLeyendoArchivo`.

Cuando el usuario hace clic en `btRestaurarZoom` (de texto “Auto Zoom”) hace que se restaure el nivel de zoom predeterminado de la imagen (mediante el método `restoreAutoBounds`).

Si el usuario hace clic en `btGuardarImagen` (de texto “Guardar Imagen”) se utiliza el método `guardarJPG` de la clase `ManejoEstadistica` que, muestra un `JFileChooser` para que el usuario elija el nombre y la dirección del archivo a ser guardado. Si el archivo es guardado correctamente, se utiliza el método `mmImagenGuardada`, caso contrario, se utiliza el método `mmErrorImagen`.

4.18 Composición y funcionamiento de PanelReproduccion

La clase `PanelReproduccion` hereda de la clase `JPanel`. Contiene los botones y etiquetas que permiten reproducir, pausar y parar el archivo cargado, también modificar la velocidad de reproducción y visualizar los datos de cronómetro y velocidad de reproducción. Sus componentes son habilitados (mediante el método `activarConfigArchivoCargado`) cuando existe un archivo cargado.

Los botones (objetos instanciados de `JButton`) que contiene `PanelReproduccion` son:

- Botón de reproducción y pausa (atributo `btReproducirPausar`). Permite al usuario iniciar (haciendo clic a éste) la reproducción y pausarla mediante los métodos `reproducirArchivo` y `pausarReproducción` de la clase `ManejoReproduccion` (del módulo Manejo de Archivos) respectivamente.
- Botón de doble velocidad (atributo `btDobleVelocidad`). Permite al usuario doblar la velocidad de reproducción (haciendo clic a éste) mediante el método `doblarVelocidad` de la clase `ManejoReproduccion`.
- Botón de mitad de velocidad (atributo `btMitadVelocidad`). Permite al usuario reducir la velocidad de reproducción a la mitad (haciendo clic a éste) mediante el método `mitadVelocidad` de la clase `ManejoReproduccion`.

-
- Botón de parar reproducción (atributo `btParar`). Permite al usuario parar (haciendo clic en éste) la reproducción mediante el método `pararReproduccion` de la clase `ManejoReproduccion`.

Las etiquetas (objetos instanciados de `JLabel`) que contiene `PanelReproduccion` son:

- Etiqueta de cronómetro (atributo `lblCronometro`). Permite al usuario visualizar un cronómetro que indica el tiempo registrado en la grabación mientras se reproduce un archivo. El texto de `lblCronometro` actualiza su contenido (según el proceso de reproducción) mediante los métodos `actualizarCronometro` y `resetearCronometro` de la clase `HiloReproduccion` (del módulo `Manejo de Archivos`). El tiempo es mostrado en el orden de minutos, segundos y milisegundos, separados por el símbolo dos puntos (:).
- Etiqueta de velocidad de reproducción (atributo `lblVelocidad`). Permite al usuario visualizar la velocidad de reproducción con relación a la velocidad original. El texto de `lblVelocidad` por defecto es “Velocidad x1.0” y cambia cuando el usuario hace clic en `btMitadVelocidad` o `btDobleVelocidad` gracias a los métodos `mitadVelocidad` y `doblarVelocidad` de la clase `ManejoReproduccion`.

Cuando el usuario inicia la reproducción de un archivo, se utiliza el método `reproducirArchivo` de la clase `ManejoReproduccion`, éste a su vez, utiliza el método `activarConfigReproduccion` para cambiar la configuración del panel a configuración de reproducción, donde se realizan cambios en la habilitación de botones (incluyendo el `btCargarArchivo` de `PanelArchivo`), el `lblCronómetro` y en el texto de `btReproducirPausar` que cambia a “Pausar”.

En la Figura N° 55 se puede observar a `PanelReproduccion` con la configuración de reproducción (realizado con el método `activarConfigReproduccion`) y con la velocidad de reproducción duplicada, mostrado en `lblVelocidad` el texto “*Velocidad x2.0*”.



Figura N° 55: Imagen de `PanelReproduccion` con la configuración de reproducción

Fuente: Elaboración propia

Cuando el usuario pausa la reproducción de un archivo, se utiliza el método `pausarReproduccion` de la clase `ManejoReproduccion` que (además de utilizar otros métodos) utiliza a su vez el método `activarConfigPausar` para cambiar la configuración del panel a configuración de pausa, donde se realizan cambios en la habilitación de botones el `lblCronómetro` y en el texto de `btReproducirPausar` que cambia a “Reproducir”.

En a Figura N° 56 se puede observar a `PanelReproduccion` con la configuración de pausa (método `activarConfigPausar`)



Figura N° 56: Imagen de `PanelReproduccion` con la configuración de pausa

Fuente: Elaboración propia

Cuando el usuario para la reproducción de un archivo, se utiliza `pararReproduccion` de la clase `ManejoReproduccion`, éste a su vez, utiliza el método

desactivarConfigRerproduccion para cambiar a la configuración por defecto (anterior a la reproducción, cuando un archivo es cargado), donde se realizan cambios en la habilitación de botones, el lblCronometro (con el método resetarCronometro de la clase HiloReproduccion) y el texto de btReproducirPausar que cambia a “Reproducir”.

En la Figura N° 57 se puede observar a PanelReproduccion con la configuración de por defecto (método desactivarConfigReproduccion).

Cuando la reproducción de un archivo termina, se utiliza el método mmReproduccionTerminada que muestra al usuario una ventana (MessageDialog), la cual avisa que la reproducción ha terminado.

En la Figura N° 58 se puede observar el MessageDialog que utiliza el método mmReproduccionTerminada.



Figura N° 57: Imagen de PanelReproduccion con la configuración por defecto

Fuente: Elaboración propia



Figura N° 58: Imagen de MessageDialog utilizado en método mmReproduccionTerminada de la clase PanelReproduccion

Fuente: Elaboración propia

En caso de que se produzca algún error en la reproducción, se le notifica al usuario que ocurrió un error mediante un `MessageDialog`, generado por el método `mmErrorReproduccion`.

En la Figura Nº 59 se puede observar el `MessageDialog` que utiliza el método `mmErrorReproduccion` cuando hay error reproduciendo el archivo “*captura25febrero.ptf*”

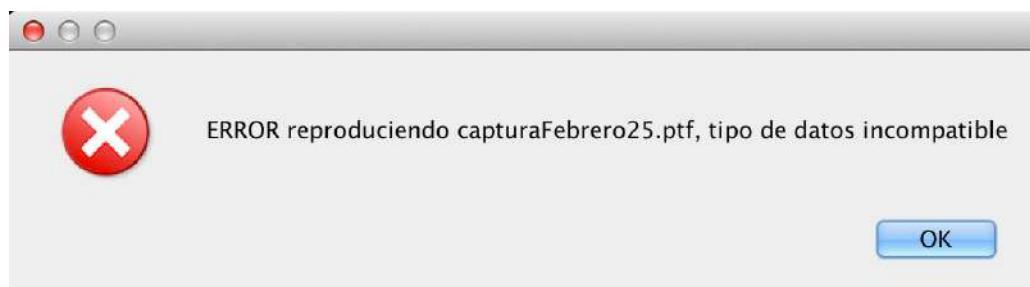


Figura Nº 59: Imagen de `MessageDialog` utilizado en método `mmErrorReproduccion` de la clase `PanelReproduccion`

Fuente: Elaboración propia

V MODIFICACIÓN VIRTUAL

El módulo Modificación Virtual se encarga de realizar modificaciones a la Producción Visual y a la Representación Virtual de Componentes, e interviene específicamente en tres acciones: el control de dibujo del teclado, los dedos y la muñeca del Escenario Virtual, el movimiento del teclado virtual y el movimiento (rotación y zoom) de la cámara de visualización.

5.1 Control de dibujo de los componentes de Escenario Virtual

Todos los componentes dibujables (dedos, muñeca, teclas) por defecto están habilitados para ser dibujados, pero cuando el usuario deshabilita un *checkbox* correspondiente a un componente, la clase ManejoVisual (que es el ItemListener del PanelControlVisual) modifica al objeto ControladorImagen (contiene el registro de los componentes habilitados para el dibujo en sus atributos) que utiliza PanelEscenario para determinar si va a realizar o no los dibujos de cada componente.

Las modificaciones del ControladorImagen se realizan con los métodos habilitarDedo, habilitarMuneca, habilitarTeclado para realizar la habilitación de componentes y, los métodos deshabilitarDedo, deshabilitarMuneca, deshabilitarTeclado para deshabilitarlos.

En la Figura Nº 60 se puede observar un PanelControlVisual con todos los componentes de la mano izquierda y con los dedos 1, 3, 5 y la muñeca de la mano derecha habilitados y, en la Figura Nº 61 se puede observar PanelEscenario con las representaciones gráficas de los componentes seleccionados.



Figura Nº 60: Imagen de PanelControlVisual con algunos componentes deshabilitados

Fuente: Elaboración propia

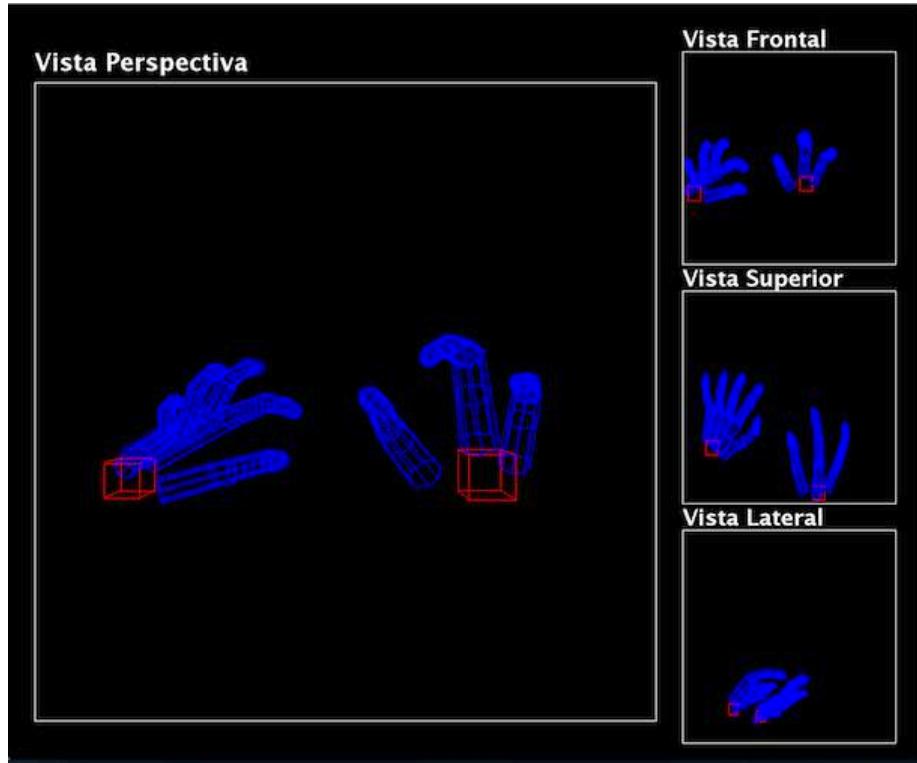


Figura N° 61: Imagen de `PanelEscenario` con algunos componentes deshabilitados

Fuente: Elaboración propia

5.2 Movimiento, eliminación y creación de Teclado

Cuando el usuario presiona un botón de `PanelControlTeclado`, se utiliza el método `moverTeclado` de la clase `ManejoControlTeclado` (`ItemListener` y el `MouseListener` de `PanelControlTeclado`) que modifica la posición del teclado mediante el método `moverTeclado` de `Teclado` (que se encuentra en `EscenarioVirtual`) según el botón que el usuario haya presionado.

En la Figura N° 62 se puede observar tres imágenes en Vista Perspectiva del `PanelEscenario` mostrando al teclado virtual en tres posiciones mientras se mueve en el eje *x* en sentido positivo.

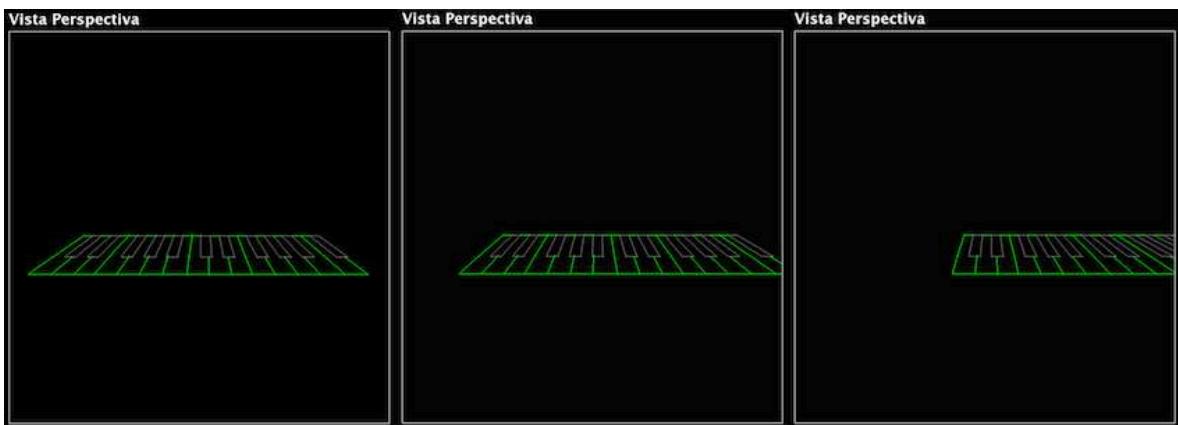


Figura N° 62: Vista Perspectiva PanelEscenario con teclado virtual en tres posiciones mientras se mueve en el eje x en sentido positivo

Fuente: Elaboración propia

Cuando el usuario deshabilita el *checkbox* cbTeclado de PanelControlTeclado, se utiliza el método eliminarTeclado de la clase ManejoControlTeclado (que a su vez utiliza el eliminarTeclado de EscenarioVirtual) y hace que el atributo teclado de EscenarioVirtual sea null.

La diferencia de deshabilitar o habilitar el *checkbox* de dibujo del teclado en PanelControlVisual y deshabilitar o habilitar el *checkbox* de teclado en PanelControlTeclado es, que el primero sólo afecta a la representación gráfica del teclado en PanelEscenario sin afectar al teclado virtual en el EscenarioVirtual, por lo tanto, el sistema sigue evaluando la posición de los dedos para generar sonido en caso de contacto con el teclado virtual. Mientras que, el *checkbox* de teclado en PanelControlTeclado, elimina el teclado virtual del Escenario Virtual, por lo tanto, afectará a la representación gráfica de teclado en PanelEscenario y a la generación de sonido.

Cuando el usuario habilita el *checkbox* cbTeclado de PanelControlTeclado, se utiliza el método crearTeclado de la clase ManejoControlTeclado (que a su vez utiliza el crearTeclado de EscenarioVirtual) y hace que se cree un nuevo teclado en la posición inicial por defecto, es decir, el punto (-180, 108, -145) según el sistema de coordenadas de Representación Virtual.

5.3 Movimiento de cámara de visualización

Para que el Usuario pueda ver la representación gráfica en PanelEscenario desde varios puntos del espacio, se habilitaron 4 teclas para conseguir el efecto de rotación de la cámara de visualización y dos teclas para producir el efecto de alejamiento y acercamiento (zoom).

Para conseguir el efecto de rotación de la cámara de visualización hacia la derecha, izquierda, arriba y debajo de su posición, se hace rotar en el sentido contrario todas las mallas poligonales del sistema (DibujoTecla, DibujoHueso y DibujoMuneca) alrededor de un punto (definido en atributo centro de la clase utilitaria HerramientasVisuales).

Los métodos de la clase ListenerRotacionZoom que intervienen en el efecto de rotación de los gráficos son rotarIzquierda, rotarDerecha, rotarArriba y rotarAbajo, que internamente, modifican los valores de los atributos anguloVertical o anguloHorizontal de la clase ControladorRotacionZoom, los cuales son utilizados en los métodos dibujarMunecas, dibujarHueso y dibujarTeclado de PanelEscenario para hacer que los gráficos sean rotados (mediante el método rotar de la clase Malla de Mallas Poligonales) según éos valores de rotación.

Por lo tanto, cuando el Usuario presiona cualquiera de las teclas de dirección, se activa un evento que, el método keyPressed de ListenerRotacionZoom (KeyListener de Pantalla) permite gestionar. Éste utiliza a su vez los métodos rotarIzquierda, rotarDerecha, rotarArriba y rotarAbajo previamente mencionados para realizar el efecto de rotación de la cámara de visualización.

Cuando el usuario presiona alguna de las 4 teclas de dirección, los métodos son utilizados de la siguiente manera:

- Tecla de dirección arriba, representado por la constante VK_UP perteneciente a la clase KeyEvent en Java, utiliza el método rotarArriba que disminuye el valor de anguloVertical de la clase controladorRotacionZoom.
- Tecla de dirección abajo, representado por la constante VK_DOWN a la clase KeyEvent en Java, utiliza el método rotarAbajo que aumenta el valor de anguloVertical de la clase controladorRotacionZoom.

-
- Tecla de dirección derecha, representado por la constante `VK_RIGHT` perteneciente a la clase `KeyEvent` en Java, utiliza el método `rotarDerecha` que disminuye el valor de `anguloHorizontal` de la clase `controladorRotacionZoom`.
 - Tecla de dirección izquierda, representado por la constante `VK_LEFT` perteneciente a la clase `KeyEvent` en Java, utiliza el método `rotarIzquierda` que aumenta el valor de `anguloHorizontal` de la clase `controladorRotacionZoom`.

En las Figura N° 63 al N° 65 se pueden observar los gráficos en Vista Perspectiva de `PanelEscenario` utilizando las teclas de dirección. Las imágenes son: gráficos sin alteración (Figura N° 63), con rotación de aproximadamente 45° a la derecha (Figura N° 64, derecha), izquierda (Figura N° 64, izquierda), arriba (Figura N° 65, izquierda) y abajo (Figura N° 65, derecha).

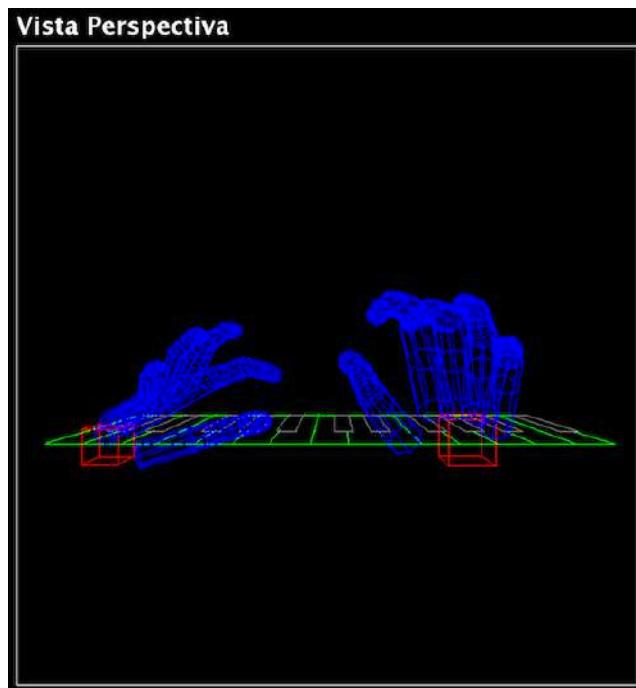


Figura N° 63: Vista Perspectiva de `PanelEscenario` sin movimiento de cámara de visualización

Fuente: Elaboración propia

Para conseguir el efecto de alejamiento y acercamiento de la cámara de visualización (efecto zoom) se mueve el centro de las mallas poligonales, haciendo que, en caso de que el usuario

quiera acercar la imagen, aumente el valor del componente z del centro y, en caso de que se quiera alejar la imagen, disminuya el valor.

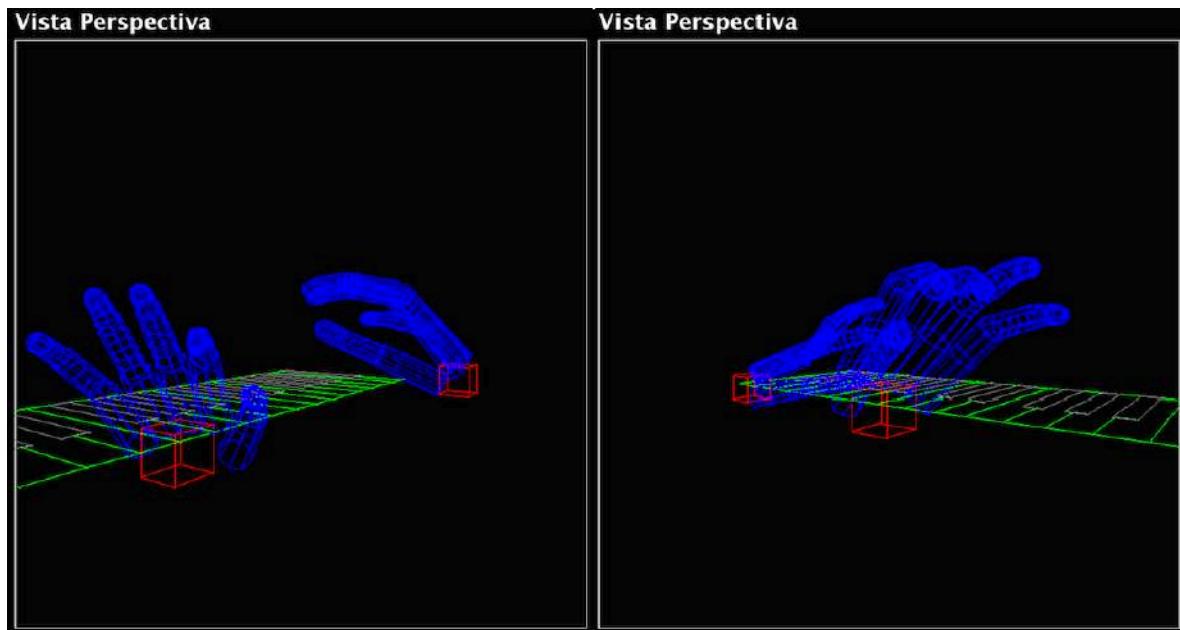


Figura N° 64: Vista Perspectiva de PanelEscenario con rotación a la izquierda y rotación a la derecha

Fuente: Elaboración propia

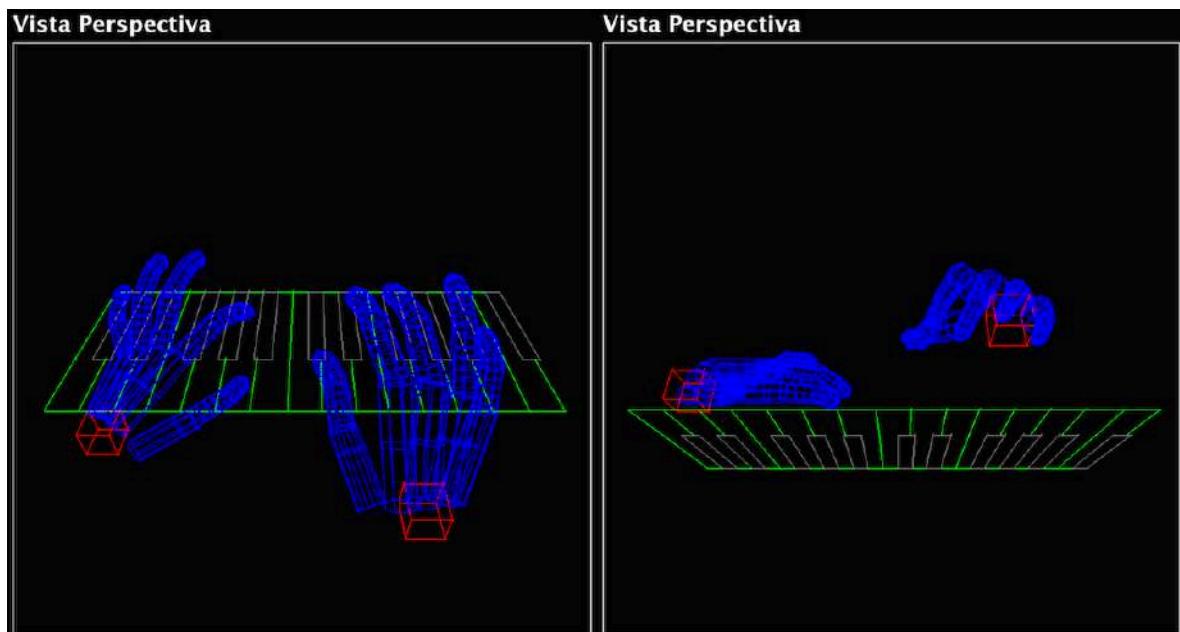


Figura N° 65: Vista Perspectiva de PanelEscenario con rotación hacia arriba y rotación hacia abajo

Fuente: Elaboración propia

Los métodos de la clase ListenerRotacionZoom que intervienen en el efecto de acercamiento y alejamiento de los gráficos son acercar y alejar, que internamente modifican los valores del atributo zoom de ControladorRotacionZoom, el cual es utilizado en los métodos dibujarMunecas, dibujarHueso y dibujarTeclado de PanelEscenario para hacer que los gráficos se acerquen o alejen (mediante el método moverZ de la clase Malla de Mallas Poligonales).

Cuando el usuario presiona alguna de las 2 teclas, S o W, los métodos son utilizados de la siguiente manera:

- Tecla S, representado por la constante VK_S perteneciente a la clase KeyEvent en Java, utiliza el método alejar que hace que el componente z del centro de las mallas poligonales aumente su valor, por lo tanto hace que el centro recorra en dirección del eje z positivo según el sistema de coordenadas utilizado en gráficos.
- Tecla W, representado por la constante VK_W perteneciente a la clase KeyEvent en Java, utiliza el método acercar que hace que el componente z del centro de las mallas poligonales disminuya su valor, por lo tanto hace que el centro recorra en dirección del eje z negativo según el sistema de coordenadas utilizado en gráficos.

En la Figura N° 63 se pueden observar los gráficos en Vista Perspectiva de PanelEscenario sin efectos (original) y en la Figura N° 66 se pueden observar con efecto de acercamiento (al presionar la tecla S) y efecto de alejamiento (al presionar la tecla W).

Se puede realizar el efecto de acercamiento de la cámara de visualización hasta un máximo 13 centímetros de la posición original y, se puede realizar el efecto de alejamiento de cámara de visualización 13 centímetros de la posición original.

El usuario puede restablecer la vista original (sin efecto de acercamiento, alejamiento o rotación) presionando la tecla R que, mediante el método restablecer, hará que los valores de los atributos anguloVertical, anguloHorizontal y zoom de la clase ControladorRotacionZoom sean restablecidos a sus valores iniciales.

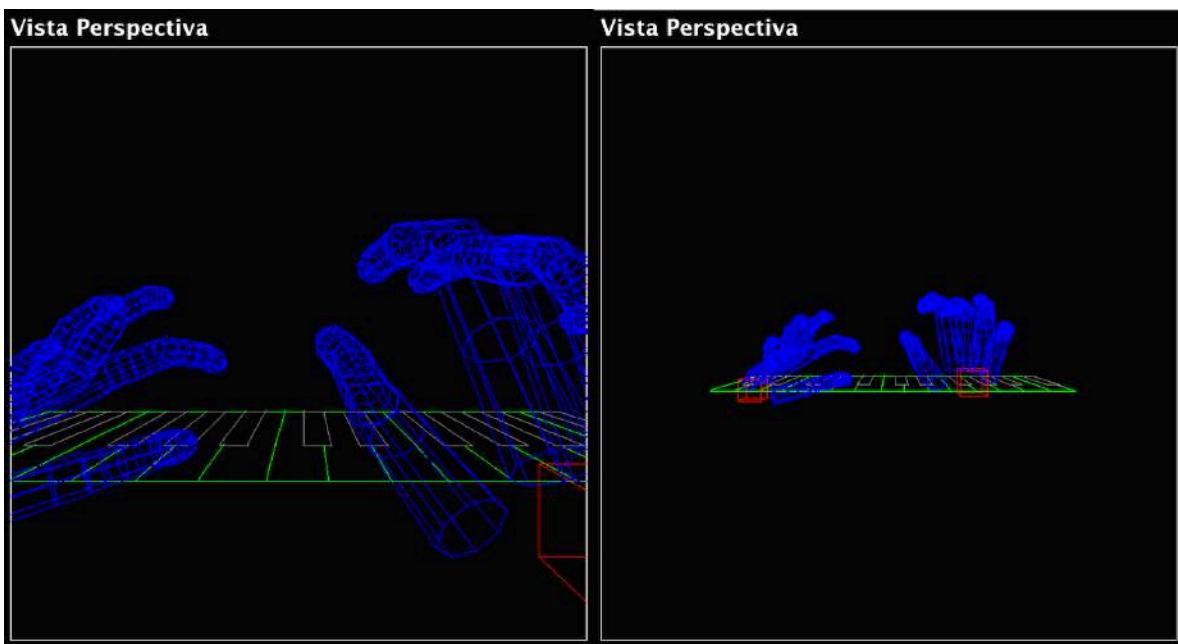


Figura N° 66: Vista Perspectiva de Panel Escenario con acercamiento y alejamiento

Fuente: Elaboración propia

VI MANEJO DE ARCHIVOS

Para poder reproducir los movimientos de las manos y analizar sus movimientos se procedió a crear un sistema de guardado de archivos, éstos tienen la extensión *.ptf* y son creados mediante la grabación de archivos.

Un archivo puede ser leído para reproducir los movimientos en el sistema, para generar cuadros estadísticos y para exportar sus datos a un archivo Excel.

6.1 Grabación de archivos

Cuando el usuario comienza a grabar (haciendo clic en el botón btGrabacion de PanelArchivo), se utiliza el método iniciarGrabacion del método ManejoDatosArchivo (ActionListener de PanelArchivo) que, a su vez, utiliza el método run de HiloGrabacion (el cual hereda de Thread).

En el método run se utiliza el método guardadoTemporal para registrar los datos existentes en el atributo manos de EscenarioVirtual (mediante el método getManos) en un archivo temporal.

En el método guardadoTemporal, el valor del atributo manos de EscenarioVirtual (Hashtable de objetos Mano) es clonado al atributo manos de objetos EscenarioGrabado y éstos son escritos en el archivo temporal “aux.temp” cada 20 milisegundos (determinado en el atributo intervaloGrabacion de la clase utilitaria HerramientasManejoArchivos) mediante el método writeObject de la clase ObjectOutputStream.

Para guardar los datos en un archivo, fue necesario implementar la interfaz Serializable en Mano, Dedo y Huésped para que puedan almacenarse en objetos EscenarioGrabado (que también implementa la interfaz Serializable).

Una vez que el usuario para la grabación (haciendo clic en btGrabacion de PanelArchivo), se utiliza el método pararGrabacion de la clase ManejoDatosArchivo que, a su vez, utiliza el método pararGrabacion del HiloGrabacion y hace que finalice el ciclo de guardado temporal (modificando el atributo

continuacionHilo) y se procede a utilizar el método guardarGrabacionFileDialog de la clase PanelArchivo (para que el usuario puede elegir el nombre y el lugar de guardado del archivo).

El método guardarGrabacionFileDialog utiliza los métodos EvaluarExistenciaArchivo, EvaluarExtensionArchivo y agregarExtensionArchivo para evaluar la existencia de archivos con el mismo nombre, evaluar si la extensión de archivo es correcto y agregar la extensión en caso de que el nombre no tenga alguna extensión (ver Funcionamiento de btGuardar).

Una vez que se determine la dirección de guardado de archivo, se utiliza el método guardadoDefinido que procede a copiar el archivo temporal “aux.temp” a la dirección determinada y posteriormente, borra el archivo temporal, terminando así la ejecución del método run de HiloGrabacion.

6.2 Reproducción de archivos

Cuando el usuario comienza a reproducir archivos (haciendo clic en el botón btReproducirPausar de PanelReproduccion), se utiliza el método reproducirArchivo de la clase ManejoReproduccion (ActionListener de PanelReproduccion) que, a su vez, utiliza el método run de HiloReproduccion (el cual hereda de Thread) y activa la configuración de reproducción en PanelReproduccion (con el método activarConfigReproduccion).

En el método run se utiliza el método leerArchivo donde se procede a leer los datos del archivo cargado previamente (la dirección es conseguida mediante los métodos getArchivo de ManejoReproduccion y ManejoDatosArchivo). La lectura de datos se realiza con el método readObject de la clase ObjectInputStream, de donde se obtienen objetos EscenarioGrabado cada 20 milisegundos (determinado en el atributo intervaloGrabacion de la clase utilitaria HerramientasManejoArchivos).

Cada vez que se realiza la lectura de un objeto EscenarioVirtual, se utiliza el método reemplazarEscenario donde se procede a clonar el atributo mano de EscenarioGrabado al atributo mano de EscenarioVirtual, esto hará que el módulo

de Producción Visual represente gráficamente los componentes que estaban almacenados en el archivo.

El método leerArchivo además utiliza el método actualizarCronómetro que actualiza el lblCronometro de PanelReproduccion. Los datos del cronómetro son obtenidos con el método convNumATiempo (de la clase utilitaria HerramientasManejoArchivo) a partir del atributo grabacionID de los objetos EscenarioGrabado.

Cuando el usuario pausa la reproducción, se utiliza el método pausarReproduccion de la clase ManejoReproduccion, que a su vez utiliza el método pausar de la clase HiloGrabacion, el cual establece el valor del atributo pausado en true, haciendo que no se realice ninguna acción de reemplazo del método leerArchivo. Además, se activa la configuración de pausa en PanelReproduccion (con el método activarConfigPausar).

Cuando el usuario dobla la velocidad, se utiliza el método doblarVelocidad de la clase ManejoReproduccion, que a su vez utiliza el método doblarVelocidad de la clase HiloGrabacion modificando el valor de intervaloReproduccion a la mitad del valor anterior y asignando el nuevo valor de velocidad al lblVelocidad.

Cuando el usuario reduce a la mitad la velocidad, se utiliza el método mitadVelocidad de la clase ManejoReproduccion, que a su vez utiliza el método mitadVelocidad de la clase HiloGrabacion modificando el valor de intervaloReproduccion al doble del valor anterior y asignando el nuevo valor de velocidad al lblVelocidad.

Cuando el usuario para la reproducción, se utiliza el método pararReproduccion de clase ManejoReproducción que a su vez utiliza el método pararGrabacion de HiloGrabacion, donde se asigna al cronómetro (lblCronometro de PanelReproduccion) los valores iniciales (método resetearCronometro), se activa la configuración por defecto en PanelReproduccion (con el método activarConfigParar) y se procede a establecer el valor del atributo parado en true, haciendo que se termine la ejecución del método run.

Cuando una reproducción finaliza o cuando existe un error, se utilizan los métodos mmReproduccionTerminada y mmErrorReproduccion (ambos de PanelReproduccion) respectivamente para informar al usuario. (ver Composición y funcionamiento de PanelReproduccion).

6.3 Generación de cuadros estadísticos

Cuando el usuario hace clic a un botón de PanelEstadistica se utiliza el método crearEstadística de la clase ManejoPanelEstadistica (ActionListener de PanelEstadistica) que instancia FrameEstadistica (ver Composición y funcionamiento de FrameEstadistica) el cual, a su vez, en su constructor instancia ManejoEstadistica y le asigna como su ActionListener e ItemListener.

Cuando se instancia FrameEstadistica, se llenan los datos de pGraficoEstadistico (panel que contiene el gráfico estadístico) con el método crearDataSet de FrameEstadistica, éste a su vez utiliza el método extraerDatosArchivo para realizar la lectura de los datos del Archivo teniendo como filtro el tipo de dato (posición en x , y , z o velocidad en y) en el parámetro tipoDatoIn de su constructor.

El método extraerDatosArchivo lee los datos del archivo (cuya dirección es conseguida del parámetro dirArchivoCargadoIn) con el método ExtraerDatosObjeto, que a su vez, para leer los datos utiliza el método readObject de la clase ObjectInputStream, de donde se obtienen objetos EscenarioGrabado y se procede a agregar los datos relevantes a los DataSet con el método agregarDatoASeries.

Los datos de los objetos Mano (de los EscenarioGrabado guardados en un archivo .ptf) que se agregan a DataSet para crear los gráficos estadísticos son (dependiendo del valor del parámetro tipoDatoIn) los componentes de posPunta de Dedo, posMuneca de Mano o velocidadMuneca de Mano.

Cuando el usuario desmarca o marca un *checkbox* (alguno de cbRHcomponentes o cbLHcomponentes) del panel pComponentesMano, se utiliza el método setVisibilidadComponente que, a su vez utiliza el método setSeriesVisible

(con parámetro `true` o `false`) de la clase `AbstractRenderer` que permite hacer visible o invisible la serie de datos graficada, la cual pertenece a los dedos o las muñecas, dependiendo del *checkbox* que inicia el evento.

Si el usuario hace clic en el botón `btRestaurarZoom` se utiliza el método `restoreAutoBounds` de la clase `ChartPanel` para restaurar el zoom predeterminado en un panel de estadística, en este caso, el atributo `pGraficoEstadístico`.

Cuando el usuario hace clic en el botón `btGuardarImagen` se utiliza el método `guardarJPG` que utiliza un `JFileChooser` que muestra al usuario una interfaz donde puede elegir el nombre y la ubicación del archivo para posteriormente guardarlo. En caso que el archivo se guarde correctamente, se utiliza el método `mmImagenGuardada` y, en caso de que ocurra un error utilizará el método `mmErrorImagen`.

6.4 Exportación de datos a archivo Excel

Cuando el usuario hace clic en el botón `btExportarDatos` del panel `PanelArchivo`, se utiliza el método `crearFrameExportarDatos` que, entre otras cosas (como el control de instanciación) instancia `FrameExportarDatos`.

`FrameExportarDatos` es una clase que hereda de `JFrame` y muestra las opciones que tiene un usuario para elegir los datos a ser exportados (ver Composición y funcionamiento de `FrameExportarDatos`).

Cuando el usuario hace clic en el botón `btExportar`, se utiliza el método `ExportarDatosAExcel` de la clase `ManejoExportarDatos` (que es el `ActionListener` de `FrameExportarDatos`) que, a su vez utiliza el método `guardarArchivoExcelFileDialog` para determinar el nombre y la dirección a ser guardados. Para controlar la extensión válida, agregar extensión y archivos existentes (para sobrescribirlos) se utilizan los métodos `evaluarExtension`, `agregarExtension` y `evaluarExistenciaArchivo` respectivamente (ver Composición y funcionamiento de `FrameExportarDatos`).

Cuando se define el nombre y la dirección para generar el archivo Excel, se utiliza el método `escribirArchivoExcel` que utilizará el método `extraerDatosArchivoPtf` para leer los datos del archivo cargado.

El método `extraerDatosArchivoPtf` crea un archivo Excel (con la biblioteca `JExcelApi`) y procede a colocar las cabeceras de las columnas (según los datos elegidos por el usuario en `FrameExportarDatos`) con el método `EscribirEtiquetas`. Posteriormente, cada vez que realiza la lectura de un objeto `EscenarioGrabado` (utilizando el método `readObject` de la clase `ObjectInputStream`), utiliza el método `exportarDatosObjeto` para seleccionar los datos del objeto `EscenarioGrabado` (según los datos elegidos por el usuario en `FrameExportarDatos`) y utiliza el método `escribirDedoMuneca` para escribir los datos en cada fila del archivo Excel.

Cuando se termina el proceso de escritura del archivo Excel correctamente, se utiliza el método `mmArchivoEscrito` para mostrar un `MessageDialog` al usuario, informando que el archivo Excel ha sido escrito correctamente. En caso de que exista algún error en el proceso, se utiliza el método `mmErrorLeyendoArchivo` para mostrar un `MessageDialog` al usuario, informando que existió un error.

En la Figura N° 67 se puede observar una captura de pantalla al contenido de un archivo Excel generado.

	A	B	C	D	E	F	G	H	I
1	Minuto	Segundo	Milisegundo	Mano	Componente	Posicion X [mm]	Posicion Y [mm]	Posicion Z [mm]	Velocidad Y [mm/s]
2	0	0	0	RH	Dedo 3	47	177	32	96
3	0	0	0	RH	Dedo 2	29	175	49	92
4	0	0	0	RH	Dedo 1	27	158	87	105
5	0	0	0	RH	Dedo 5	96	154	37	53
6	0	0	0	RH	Dedo 4	72	173	30	0
7	0	0	0	RH	Muneca	95	116	138	114
8	0	0	500	RH	Dedo 3	42	200	19	27
9	0	0	500	RH	Dedo 2	21	205	36	228
10	0	0	500	RH	Dedo 1	12	178	74	48
11	0	0	500	RH	Dedo 5	90	176	30	46
12	0	0	500	RH	Dedo 4	66	187	20	-30
13	0	0	500	RH	Muneca	86	145	128	99
14	0	1	0	RH	Dedo 3	22	196	25	-18
15	0	1	0	RH	Dedo 2	2	193	41	-18
16	0	1	0	RH	Dedo 1	-6	179	79	-16
17	0	1	0	RH	Dedo 5	78	189	38	-4
18	0	1	0	RH	Dedo 4	50	203	29	-1

Figura N° 67: Captura de pantalla al contenido de un archivo Excel generado con el sistema

Fuente: Elaboración propia

VII COORDINACIÓN GENERAL

Para el desarrollo general del sistema, se utilizan las clases *SistemaTecnicaPianistica* y *ControladorHilos*, que sirven para iniciar el sistema, finalizar el sistema y controlar el funcionamiento de los hilos, dependiendo la configuración que requiera el usuario.

La clase *SistemaTecnicaPianistica* es la que contiene el método *main*, por lo tanto es la que inicia el sistema con la configuración predeterminada utilizando los métodos *run* de *HiloSonido*, *HiloDibujo* e *HiloLeap*.

La clase *ControladorHilos* se encarga de controlar el funcionamiento de los hilos, contiene la información sobre el estado de funcionamiento de los hilos en sus atributos. Cuando el usuario inicia una reproducción o grabación, cambia la configuración predeterminada del sistema y deshabilita o habilita los hilos necesarios según la configuración que se requiera.

Cuando el usuario inicia la reproducción de un archivo, se utiliza el método *activarConfigReproduccion* que, cambia el valor del atributo correspondiente al hilo del Leap Motion (*hLeapActivo*) a false, lo cual hace que el Leap Motion no capture los datos (ni realice ningún cambio al Escenario Virtual). Además, cambia el valor del atributo correspondiente al hilo de reproducción (*hReproduccionActivo*) a true, lo que hace que los objetos del Escenario Virtual sean copiados de un archivo *.ptf* (proceso realizado en *HiloReproduccion*).

Cuando la reproducción finaliza o el usuario lo termina, se procede a restaurar la configuración predeterminada con el método *activarConfigPredeterminada*.

Cuando el usuario inicia una grabación, se utiliza el método *activarConfigGrabacion* que, cambia el valor del atributo correspondiente al hilo de grabación (*hGrabacionActivo*) a true, lo que hace que (hasta que el usuario finalice el proceso) los objetos del Escenario Virtual sean copiados a un archivo temporal (proceso realizado en *HiloGrabacion*).

Al igual que en la reproducción, cuando la grabación finaliza o el usuario lo termina, se procede a restaurar la configuración predeterminada con el método activarConfigPredeterminada.

Cuando el usuario finaliza el sistema desde la interfaz gráfica, el objeto instanciado de la clase Pantalla (JFrame principal del sistema) del módulo Producción Visual utiliza el método terminarPrograma, que a su vez utiliza el método terminarHilos de ControladorHilos para que todos los hilos terminen su ejecución y el sistema finalice.