EDUCACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Orizaba

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

OPCIÓN I.- TESIS

TRABAJO PROFESIONAL

"Biblioteca de Java para el desarrollo de interfaces gestuales con las manos utilizando LeapMotion"

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:

I.S.C. Ricardo Arellano Morales

DIRECTOR DE TESIS:

M.C. María Antonieta Abud Figueroa

CODIRECTOR DE TESIS:

Dra. Lisbeth Rodríguez Mazahua

ORIZABA, VERACRUZ, MÉXICO. MARZO 2022

Orizaba, Veracruz, 25/marzo/2022
Dependencia: **División de Estudios de Posgrado e Investigación**
Asunto: **Autorización de Impresión**
OPCION: **I**

C. RICARDO ARELLANO MORALES
Candidato a Grado de Maestro en:
SISTEMAS COMPUTACIONALES
P R E S E N T E.-

De acuerdo con el Reglamento de Titulación vigente de los Centros de Enseñanza Técnica Superior, dependiente de la Dirección General de Institutos Tecnológicos de la Secretaría de Educación Pública y habiendo cumplido con todas las indicaciones que la Comisión Revisora le hizo respecto a su Trabajo Profesional titulado:

" Biblioteca de Java para el desarrollo de interfaces gestuales con las manos utilizando LeapMotion"

comunico a Usted que este Departamento concede su autorización para que proceda a la impresión del mismo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
CIENCIA – TÉCNICA - CULTURA®

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE ORIZABA
DIV. DE EST. DE POSGRADO E INVEST.

DR. MARIO LEONCIO ARRIOJA RODRÍGUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN

OG-13-F06

Orizaba, Veracruz, 01/marzo/2022
Asunto: **Revisión de trabajo escrito**

C. MARIO LEONCIO ARRIOJA RODRÍGUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
P R E S E N T E.-

Los que suscriben, miembros del Jurado, han realizado la revisión de la Tesis del (la) C.

RICARDO ARELLANO MORALES

la cual lleva el título de:

"Biblioteca de Java para el desarrollo de interfaces gestuales con las manos utilizando LeapMotion"

y concluyen que se acepta.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
CIENCIA – TÉCNICA - CULTURA®

PRESIDENTE:	M.C. MA. ANTONIETA ABUD FIGUEROA	*FIRMA*
SECRETARIO:	DRA. LISBETH RODRÍGUEZ MAZAHUA	*FIRMA*
VOCAL:	DR. HILARIÓN MUÑOZ CONTRERAS	*FIRMA*
VOCAL SUP.:	DR. ULISES JUÁREZ MARTÍNEZ	*FIRMA*

TA-09-21

CARTA DE ORIGINALIDAD

En la ciudad de Orizaba, Veracruz, el día 25 de marzo del año 2022, el (la) que suscribe **Ricardo Arellano Morales,** alumno (a) del programa de **Maestría en Sistemas Computacionales** con número de control **M14010978**, manifiesta que es autor(a) del trabajo de tesis titulado **"Biblioteca de Java para el desarrollo de interfaces gestuales con las manos utilizando LeapMotion"** y declaro que el trabajo es original ya que sus contenidos son producto de mi directa contribución intelectual. Todos los datos y las referencias a materiales ya publicados están debidamente identificados con su respectivo crédito e incluidos en las notas bibliográficas y en las citas que se destacan como tal y, en los casos que así lo requieran, cuento con las debidas autorizaciones de quienes poseen los derechos patrimoniales. Por lo tanto, me hago responsable de cualquier litigio o reclamación relacionada con derechos de propiedad intelectual, exonerando de toda responsabilidad al Tecnológico Nacional de México / Instituto Tecnológico de Orizaba.

Ricardo Arellano Morales
Nombre y Firma

CARTA DE CESIÓN DE DERECHOS

En la ciudad de Orizaba, Veracruz, el día 25 del mes de marzo del año 2022, el (la) que suscribe **Ricardo Arellano Morales**, alumno (a) del programa de **Maestría en Sistemas Computacionales** con número de control **M14010978**, manifiesta que es autor(a) del trabajo de tesis bajo la dirección de María Antonieta Abud Figueroa y cede los derechos del trabajo de tesis titulado **"Biblioteca de Java para el desarrollo de interfaces gestuales con las manos utilizando LeapMotion"** al **Tecnológico Nacional de México / Instituto Tecnológico de Orizaba** para su difusión y divulgación, con fines académicos y de investigación.

Queda estrictamente prohibido reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del **Tecnológico Nacional de México / Instituto Tecnológico de Orizaba**. Este puede obtenerse escribiendo a la siguiente dirección: msc@orizaba.tecnm.mx. Si el permiso se otorga, cualquier usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Ricardo Arellano Morales

Nombre y Firma

A mi madre

Agradezco infinitamente todo lo que has hecho por mis hermanas y por mí, por estar con nosotros, por cuidarnos, apoyarnos y formarnos, te quiero mucho.

A mi padre

Agradezco infinitamente el compromiso, esfuerzo y sacrificio que, durante todo este tiempo realizaste para que tuviéramos la oportunidad de estudiar, sin tu apoyo no estaría aquí, te quiero mucho.

A mis hermanas

Su apoyo incondicional, comprensión y cariño han dejado huella en mí, me hacen ser mejor persona cada día, las quiero mucho.

Índice

Índice de figuras vi

Índice de tablas ix

Resumen x

Abstract xi

Introducción xii

Capítulo 1. Antecedentes 1

1.1 Marco Teórico 1

1.1.1 Interacción Humano Máquina (HMI) 1

1.1.2 Interacción Humano Computadora (HCI) 1

1.1.3 Interfaz de Usuario (UI) 1

1.1.4 Interfaz de Usuario Natural (NUI) 1

1.1.5 Computación Centrada en el Ser Humano (HCC) 2

1.1.6 Experiencia de Usuario 2

1.1.7 Interfaz Gestual 2

1.1.8 Reconocimiento Gestual 3

1.1.9 Diseño de Interacción 3

1.1.10 Técnicas de Interacción 3

1.1.11 Interacción en Tres Dimensiones 4

1.1.12 Gestos en el aire (*Mid-air gestures*) 4

1.1.13 Rastreo Óptico 4

1.1.14 Leap Motion 4

1.1.15 Reconocimiento de Patrones 5

1.2 Planteamiento del problema 5

1.3 Objetivo general y específico 5
1.3.1 Objetivo general 6
1.3.2 Objetivos específicos 6
1.4 Justificación 6
Capítulo 2. Estado de la práctica 7
2.1 Trabajos Relacionados 7
2.2 Análisis Comparativo 15
2.3 Propuesta de solución 20
2.3.1 Justificación de la solución 20
Capítulo 3. Aplicación de la Metodología 22
3.1 Descripción de la solución 22
3.1.1 Módulo de captura de imágenes 23
3.1.2 Módulo de configuración consumidor-gesto 23
3.1.3 Módulo de procesamiento de gesto-acciones 23
3.1.4 Módulo de integración con el desarrollo de aplicaciones 23
3.1.5 Módulo de interacción con usuario 23
3.1.6 Módulo de control 23
3.2 Marco de trabajo Scrum 23
3.2.1 Historias de usuario 24
3.2.2 Sprints 27
3.2.3 Sprint 1 29
3.2.3.1 HU_INV_01: Analizar las características del dispositivo Leap Motion 29
3.2.3.2 HU_INV_02: Analizar el kit de desarrollo de software del dispositivo Leap Motion 31

3.2.3.3 HU_DGC: Definir los gestos y comandos que se incorporarán a la biblioteca 33
3.2.3.3.1 Golpe suave con la mano 34
3.2.3.3.2 Golpe suave con los dedos 34
3.2.3.3.3 Pellizco 35
3.2.3.3.4 Pose con los dedos 35
3.2.3.3.5 Giro de mano 36
3.2.4 Sprint 2 36
3.2.4.1 HU_ARQ: Establecer una arquitectura que sirva de base al desarrollo de la biblioteca 36
3.2.4.1.1 Productor de eventos 37
3.2.4.1.2 Canal de eventos 38
3.2.4.1.3 Consumidor 38
3.2.4.1.4 Herramientas 38
3.2.5 Sprint 3 38
3.2.5.1 HU_CAP_IMG: Desarrollar módulo de captura de *frames* 39
3.2.5.2 HU_REC_GES: Desarrollar módulo de reconocimiento de gestos 42
3.2.6 Sprint 4 57
3.2.6.1 HU_CONF_CG: Desarrollar módulo de configuración consumidor-gesto. 57
3.2.7 Sprint 5 59
3.2.7.1 HU_PRO_GA: Desarrollar módulo de procesamiento de acciones 59
3.2.8 Sprint 6 61
3.2.8.1 HU_CONTROL: Desarrollar módulo de control 61
3.2.8.2 HU_INT_USU: Desarrollar módulo de interacción con usuario 62

3.2.9 Sprint 7 63
Capítulo 4. Resultados 65
4.1 Caso de estudio 65
4.1.1 Planteamiento del caso de estudio 65
4.1.2 Descripción 65
4.1.3 Objetivo general 66
4.1.4 Población atendida 66
4.1.5 Metodología 66
4.1.6 Desarrollo del caso de estudio 66
4.1.7 Implementación del caso de estudio 73
4.1.8 Resultado de la implementación del caso de estudio 75
Capítulo 5. Conclusiones y recomendaciones 79
5.1 Conclusiones 79
5.2 Recomendaciones 80
Productos académicos 81
Referencias 82

Índice de figuras

Figura 3.1 Arquitectura propuesta. 22
Figura 3.2 Representación del funcionamiento del dispositivo Leap Motion [22]. 29
Figura 3.3 Segmentos de la mano identificados por el dispositivo Leap Motion [23]. 30
Figura 3.4 Ejes de coordenadas del dispositivo Leap Motion [24]..... 30
Figura 3.5. Ejemplo de uso del dispositivo Leap Motion. 31
Figura 3.6 Clases provistas por el kit de desarrollo de aplicaciones. 32
Figura 3.7 Campo de visión disponible efectivo mediante la caja de interacción [24]. 33
Figura 3.8 Golpe suave con la mano [25]..... 34
Figura 3.9 Golpe suave con los dedos [25]. 34
Figura 3.10 Pellizco [25]. 35
Figura 3.11 Poses con los dedos [25]. 35
Figura 3.12 Giro de mano [25]..... 36
Figura 3.13. Arquitectura de biblioteca actualizada. 37
Figura 3.14. Diagrama de clases del módulo de captura de *frames.* 39
Figura 3.15. Interacción con la opción `offset` desactivada. 40
Figura 3.16. Interacción con la opción `offset` activada. 41
Figura 3.17. Atributos de la clase `TrackingData`. 41
Figura 3.18. Método `build_frame` de la clase `TrackingData`. 42
Figura 3.19 Ejemplos de gestos realizados con las manos [27]. 43
Figura 3.20 Funcionamiento del módulo de reconocimiento de gestos. 43
Figura 3.21 Ángulos formados por las falanges. 43
Figura 3.22. Características seleccionadas para describir un gesto. 44
Figura 3.23. Diagrama de clases del módulo de reconocimiento de gestos. 45
Figura 3.24. Herramienta de extracción de muestras. 45
Figura 3.25. Ejemplo de archivos de muestras generados. 46
Figura 3.26. Contenido de un archivo de muestra. 46

Figura 3.27. Secuencia de ángulos omega recuperados del dedo pulgar en el gesto Giro Mano..... 47
Figura 3.28. Aplicación del filtro digital Savitzky-Golay. 48
Figura 3.29. Valores seleccionados como parte del proceso de muestreo. 48
Figura 3.30. Función para entrenar modelo de clasificación. 49
Figura 3.31. Funciones para cargar los archivos de muestras para entramiento del modelo. 50
Figura 3.32. Método extract de la clase FrameExtractor. 52
Figura 3.33. Método finger_features de la clase FrameExtractor. 53
Figura 3.34. Métodos angle y valid_hand de la clase FrameExtractor. 53
Figura 3.35. Método `process_frame` de la clase `GestureRecognition`. 54
Figura 3.36. Método `sample_feature` de la clase `Sampler`. 55
Figura 3.37. Método `init` de la clase `Sampler`..... 55
Figura 3.38. Método `sample_sequence` de la clase `Sampler`. 56
Figura 3.39 Conexiones realizadas al módulo de configuración consumidor-gesto. . 57
Figura 3.40. Código requerido para iniciar el servidor de WebSocket. 58
Figura 3.41. Manejadores de consumidor y productor. 58
Figura 3.42. Manejador de la sección consumidor. 58
Figura 3.43. Manejador de la sección productor. 59
Figura 3.44. Diagrama de clases del módulo de procesamiento de acciones. 60
Figura 3.45. Clases del módulo de control. 61
Figura 3.46. Uso de las clases `ConfigBuilder` y `Config`. 62
Figura 3.47. Ventana del módulo de interacción con usuario. 62
Figura 3.48. Construcción de archivo JAR a través de Maven. 63
Figura 3.49. Ejemplo del uso de la biblioteca en un ambiente de desarrollo Java. ... 64
Figura 4.1. Ejercicio 1. 67
Figura 4.2. Ejercicio 2. 68
Figura 4.3. Ejercicio 3. 69
Figura 4.4. Ejercicio 4. 70
Figura 4.5. Ejercicio 5. 71

Figura 4.6. Nivel uno del Ejercicio 6. 72
Figura 4.7. Niveles adicionales del ejercicio 6. 72
Figura 4.8. Ejercicio en funcionamiento. 73
Figura 4.9. Participante uno del caso de estudio. 74
Figura 4.10. Participante dos del caso de estudio. 74
Figura 4.11. Criterios de evaluación de ejercicios. 77
Figura 4.12. Resultados de evaluación de los ejercicios del participante uno. 77
Figura 4.13. Resultados de evaluación de los ejercicios del participante dos. 78

Índice de tablas

Tabla 2.1. Análisis comparativo de artículos. 15
Tabla 2.2. Selección de tecnologías de información. 20
Tabla 3.1 Historia de usuario HU_INV_01: Analizar las características del dispositivo Leap Motion..... 24
Tabla 3.2 Historia de usuario HU_INV_02: Analizar el kit de desarrollo de software del dispositivo Leap Motion. 24
Tabla 3.3 Historia de usuario HU_DGC: Definir los gestos y comandos que se incorporarán a la biblioteca. 24
Tabla 3.4 Historia de usuario HU_ARQ: Establecer una arquitectura que sirva de base al desarrollo de la biblioteca. 25
Tabla 3.5. HU_CAP_IMG: Desarrollar módulo de captura de *frames*. 25
Tabla 3.6 Historia de usuario HU_REC_GES: Desarrollar módulo de reconocimiento de gestos. 25
Tabla 3.7 Historia de usuario HU_CONF_CG: Desarrollar módulo de configuración consumidor-gesto. 26
Tabla 3.8 Historia de usuario HU_PRO_GA: Desarrollar módulo de procesamiento de acciones. 26
Tabla 3.9 Historia de usuario HU_INT_USU: Desarrollar módulo de interacción con usuario. 26
Tabla 3.10 Historia de usuario HU_CONTROL: Desarrollar módulo de control. 27
Tabla 3.11 Historia de usuario HU_INT_DES: Desarrollar módulo de integración con el desarrollo de aplicaciones. 27
Tabla 3.12 *Sprints* e historias de usuario del proyecto. 27
Tabla 4.1. Tabla de resultados del cuestionario de usabilidad 76

Resumen

El dispositivo LeapMotion es una novedosa solución de hardware que ofrece datos de seguimiento de la mano a través de luz y un par de cámaras infrarrojas, se utiliza en varias áreas que se benefician en el uso de los movimientos de la mano, como en terapia de rehabilitación de las manos, la enseñanza del lenguaje de señas o en entornos de realidad virtual. En este trabajo se propone una biblioteca de software, que permitirá integrar los datos de seguimiento del dispositivo LeapMotion y las capacidades de reconocimiento de gestos, proporcionadas por un modelo de Aprendizaje Automático, en el lenguaje de programación Java. Se propone también una arquitectura dirigida por eventos para guiar el proceso de desarrollo de dicha biblioteca, la cual ofrece flexibilidad para utilizar herramientas especializadas en reconocimiento de gestos y Aprendizaje Automático. El protocolo WebSocket se utiliza para comunicar las partes internas de la arquitectura como un canal bidireccional de baja latencia. Como resultado, los desarrolladores de software utilizarán la biblioteca en sus proyectos y consumirán los datos proporcionados con poco esfuerzo.

Abstract

The LeapMotion device is a novel hardware solution that offers hand tracking data through light and a pair of infrared cameras, it is used in several areas that benefit in the use of hand movements, such as hand therapy rehabilitation, sign language teaching or in virtual reality environments. In this work, a software library is proposed, that allows to integrate the LeapMotion device tracking data and gesture recognition capabilities, provides by a Machine Learning model, into the Java programming language. An event driven architecture is proposed to guide the development process of such library, this architecture offers the flexibility to use tools that specialize in gesture recognition and Machine Learning. The WebSocket protocol is used to communicate the inner parts of the architecture as a low latency bidirectional channel. As a result, software developers will use the library in their projects to consume the provided data with little effort.

Introducción

Interactuar con una computadora en la actualidad es verdaderamente fácil si se compara con los primeros modelos, el avance y desarrollo de la Interacción Humano-Computadora es la base para diseñar y construir sistemas sencillos y eficaces. En un inicio la interacción se caracterizó por el uso de texto y comandos, después, se dio pauta para el desarrollo de interfaces gráficas que incluyeron el uso del puntero y nuevos aparatos para controlarlo. Pareciera que las interfaces gráficas de usuario serán utilizadas por mucho tiempo, de modo que una nueva generación, la interfaz natural de usuario (NUI) ha pasado desapercibida, en ella, se busca que la comunicación se realice a través de los movimientos del cuerpo o la voz.

Varios dispositivos de NUI han sido desarrollados, siendo LeapMotion uno de los más notables, por su accesibilidad y características que ofrece para el seguimiento del movimiento de las manos.

Este documento se compone de cinco capítulos, el primero se enfoca en dar a conocer los conceptos básicos del dominio del problema, el planteamiento, objetivo general y específicos, así como la justificación del proyecto. El capítulo dos se compone de una revisión de las investigaciones y publicaciones que se realizaron anteriormente, enfocadas en el uso del dispositivo Leap Motion, además de describir la propuesta de solución. En el capítulo tres se describe el desarrollo del proyecto, indicando la aplicación de la metodología, el capítulo cuatro contiene los resultados obtenidos durante el desarrollo de la tesis a través de un caso de estudio y en el capítulo cinco se exponen las conclusiones, así como las recomendaciones. Por último, se incluyen las secciones de anexos y referencias.

Capítulo 1. Antecedentes

En este capítulo se presentan los conceptos relacionados con el tema de investigación, se incluye el planteamiento del problema, objetivos (general y específicos) y la justificación.

1.1 Marco Teórico

En este capítulo se muestra las definiciones de los conceptos relacionados con el tema de investigación.

1.1.1 Interacción Humano Máquina (HMI)

La interacción humano-máquina (HMI) se refiere a la comunicación e interacción entre una persona y una máquina a través de una interfaz de usuario. Hoy en día, las interfaces de usuario naturales, como los gestos, han ganado una atención cada vez mayor, ya que permiten a los humanos controlar las máquinas mediante comportamientos naturales e intuitivos [1].

1.1.2 Interacción Humano Computadora (HCI)

El término Interacción Humano Computadora (HCI) es una disciplina preocupada por el diseño, evaluación e implementación de sistemas de cómputo interactivos para uso humano y por el estudio de los principales fenómenos que los rodean [2].

1.1.3 Interfaz de Usuario (UI)

El medio a través del cual tiene lugar la comunicación entre los usuarios y la computadora. La interfaz de usuario traduce las acciones y el estado (entradas) de un usuario en una representación que la computadora comprenda y sobre la que actúa, y traduce las acciones y el estado (salidas) de la computadora en una representación comprensible para los humanos y sobre la que actúan [3].

1.1.4 Interfaz de Usuario Natural (NUI)

Interfaz de Usuario Natural es un campo del diseño de interfaces donde las habilidades humanas se entrelazan con la tecnología. Una NUI se basa en que un usuario realice gestos relativamente naturales que se descubren rápidamente para controlar una aplicación o manipular el contenido en pantalla.

Es un área que se centra en las habilidades humanas, como la visión, el tacto, el habla, la escritura, el movimiento, así como la cognición, la creación y la exploración para replicar entornos del mundo real para optimizar la interacción entre objetos físicos y digitales [4].

1.1.5 Computación Centrada en el Ser Humano (HCC)

La computación centrada en el ser humano (HCC) es un campo emergente que tiene como objetivo cerrar las brechas existentes entre las diversas disciplinas involucradas con el diseño e implementación de sistemas informáticos que apoyan las actividades humanas. HCC tiene como objetivo integrar estrechamente las ciencias humanas (por ejemplo, sociales y cognitivas) y las ciencias de la computación (por ejemplo, HCI), procesamiento de señales, aprendizaje automático y computación ubicua) para el diseño de sistemas informáticos con un enfoque humano desde el principio hasta el fin. Este enfoque debe considerar los contextos personales, sociales y culturales en los que se implementan dichos sistemas. Más allá de ser un lugar de encuentro para las disciplinas existentes, HCC también apunta a cambiar radicalmente la computación con nuevas metodologías para diseñar y construir sistemas que apoyen y enriquezcan la vida de las personas [5].

1.1.6 Experiencia de Usuario

La experiencia de usuario es la alineación cuidadosa de los comportamientos, necesidades y habilidades humanas con el valor central entregado a través de un producto o servicio. Dependiendo del contexto, esta experiencia tiene componentes psicológicos, culturales, fisiológicos y emocionales, o una combinación de los cuatro.

Los expertos en usabilidad la reconocen en el contexto de la interfaz gráfica y el diseño web, como una forma de medir el grado de impacto en el usuario final [6].

1.1.7 Interfaz Gestual

Las interfaces gestuales tienen una gama mucho más amplia de acciones con las que manipular un sistema. Además de tener la capacidad de escribir, desplazarse, señalar y hacer clic, y realizar todas las demás interacciones estándar disponibles para los sistemas de escritorio, las interfaces gestuales aprovechan todo el cuerpo

para activar los comportamientos del sistema. El movimiento de un dedo permite iniciar un desplazamiento. El giro de una mano transforma una imagen. El movimiento de un brazo limpia los elementos de una pantalla. Una habitación cambia la temperatura cuando detecta que una persona ingresa [7].

1.1.8 Reconocimiento Gestual

El reconocimiento gestual es un campo activo de investigación que trata de integrar el canal gestual en la Interacción Humano Computadora. Tiene aplicaciones en el control de ambientes virtuales, pero también en la traducción de lenguajes, control remoto de robots o creación musical.

El reconocimiento de gestos humanos se inscribe en el marco más general de reconocimiento de patrones. En este marco, los sistemas consisten en dos procesos: la representación y el proceso de decisión. El proceso de representación convierte los datos numéricos sin procesar en una forma adaptada al proceso de decisión que luego clasifica los datos [8].

1.1.9 Diseño de Interacción

El diseño de interacción es la práctica de diseñar productos, entornos, sistemas y servicios digitales interactivos. Como muchas disciplinas del diseño, el diseño de interacción se preocupa por la forma. Sin embargo, ante todo, se centra en algo que las disciplinas de diseño tradicionales no suelen explorar: el diseño del comportamiento [9].

1.1.10 Técnicas de Interacción

Una técnica de interacción es un método que permite al usuario completar una tarea mediante una interfaz de usuario (UI). Una técnica de interacción incluye hardware (dispositivos de entrada/salida) y componentes de software. El componente de software de las técnicas de interacción es responsable de mapear la información del dispositivo (o dispositivos) de entrada en alguna acción dentro del sistema y de mapear la salida del sistema a un dispositivo (o dispositivos) de salida [3].

En el contexto de una técnica de interacción, un objeto de interacción y su software de soporte a menudo se denominan "*widget*" [10].

1.1.11 Interacción en Tres Dimensiones

La interacción en tres dimensiones se define como la interacción humano-computadora en la que las tareas del usuario se realizan directamente en un contexto espacial 3D real o virtual. Los sistemas interactivos que muestran gráficos en 3D no implican necesariamente una interacción en 3D; por ejemplo, si un usuario recorre un modelo de un edificio en su computadora de escritorio eligiendo puntos de vista de un menú tradicional, no se produjo ninguna interacción 3D. Por otro lado, la interacción 3D no significa necesariamente que se utilicen dispositivos de entrada 3D; por ejemplo, en la misma aplicación, si el usuario hace clic en un objeto de destino para navegar a ese objeto, entonces la entrada del ratón 2D se traduce directamente a una ubicación virtual 3D, se considera que esto es una forma de interacción 3D [3].

1.1.12 Gestos en el aire (*Mid-air gestures*)

El término denota la interacción por parte del usuario con un sistema de cómputo mediante sus extremidades superiores. Dispositivos sensores transforman los movimientos de una o más extremidades superiores en un flujo de datos que es decodificada e interpretada por un algoritmo de computadora como un “gesto” [11].

1.1.13 Rastreo Óptico

El rastreo óptico es una tecnología que hace uso de información visual para rastrear al usuario. Existen diversas formas de lograr este cometido, la más común es hacer uso de una cámara de video que actúa como un ojo electrónico que observa al objeto o persona de interés. Las técnicas de visión por computadora se usan para determinar la posición del objeto basada en lo que la cámara ve. En algunos casos, dispositivos sensibles a la luz son utilizados en lugar de cámaras [12].

1.1.14 Leap Motion

Leap Motion es un dispositivo diseñado para detectar los gestos de la mano y posición de los dedos en aplicaciones de software interactivas; rastrea los brazos, muñecas y posiciones de las manos de hasta cuatro individuos. Cuenta con tres emisores LED (*Light-Emitting Diode*, Diodo Emisor de Luz) infrarrojos e incorpora dos cámaras. Utiliza coordenadas cartesianas para gestionar la detección de los

movimientos en un sistema de coordenadas diestro y la velocidad de seguimiento es de hasta 200 cuadros por segundo en un campo de visión de 150°, con aproximadamente ocho pies cúbicos de espacio 3D interactivo [2].

1.1.15 Reconocimiento de Patrones

El reconocimiento de patrones es un campo maduro y de rápido desarrollo; forma el núcleo de muchas otras disciplinas, como la visión por computadora, el procesamiento de imágenes, el diagnóstico clínico, la identificación de personas, el análisis de textos y documentos. Está estrechamente relacionado con el aprendizaje automático y también encuentra aplicaciones en áreas rápidamente emergentes como la biometría, la bioinformática, el análisis de datos multimedia y más recientemente, la ciencia de datos [13].

1.2 Planteamiento del problema

La investigación y el desarrollo de interfaces gestuales es abundante, las primeras soluciones planteadas requerían guantes caros u otro equipamiento intrusivo. En la medida que los sensores y microprocesadores son más potentes y baratos, es posible la construcción de interfaces capaces de recibir e interpretar los más variados gestos sin cables.

El dispositivo Leap Motion surgió como un intento de ofrecer un sensor de reconocimiento de gestos asequible, sobre el cual se han desarrollado diversas aplicaciones en los últimos años, sin embargo, no se tiene información sobre alguna biblioteca que facilite el desarrollo de aplicaciones controladas por interfaces gestuales.

Por esta razón en este trabajo de tesis se propuso construir una biblioteca para facilitar la fabricación de aplicaciones controladas por interfaces gestuales, mediante la ayuda del kit de desarrollo de Leap Motion y el lenguaje de programación Java.

1.3 Objetivo general y específico

A continuación, se describe el objetivo general y los objetivos específicos.

1.3.1 Objetivo general

Desarrollar una biblioteca en lenguaje Java para la realización de aplicaciones controladas por interfaces gestuales con las manos utilizando Leap Motion.

1.3.2 Objetivos específicos

- Revisar las características del dispositivo Leap Motion para identificar aquellas factibles de utilizar en el proyecto.
- Obtener y estudiar el SDK (*Software Development Kit*, Kit de Desarrollo de Software) del dispositivo Leap Motion.
- Definir los gestos y comandos que se incorporarán a la biblioteca y que servirán para desarrollar aplicaciones con interfaces gestuales.
- Establecer una arquitectura que sirva de base al desarrollo de la biblioteca.
- Diseñar y desarrollar la biblioteca para gestos.
- Probar la biblioteca en un caso de estudio.

1.4 Justificación

La realización de este proyecto permitirá que desarrolladores de aplicaciones controladas por interfaces gestuales se vean beneficiados, al tener como soporte una biblioteca sobre la cual construir software, los recursos valiosos como el tiempo y dinero podrán reducirse. Además de los beneficios antes mencionados este proyecto establecerá las bases para futuros trabajos de investigación, encaminados a implementar mejoras o incluir nuevas características.

Capítulo 2. Estado de la práctica

En este capítulo se describe un conjunto de trabajos relacionados con el uso de Leap Motion, su participación en una serie de soluciones y el estudio de sus capacidades físicas para la detección de gestos, se incluye un análisis comparativo de los trabajos mediante una tabla comparativa y para finalizar, se describen las conclusiones obtenidas a partir del análisis.

2.1 Trabajos Relacionados

Mantecón et al. [14] declararon que un gran número de trabajos dedicados al reconocimiento de gestos de la mano en tiempo real utilizan la información del esqueleto de la mano provista por el dispositivo Leap Motion, esto conlleva ciertas desventajas, entre las que destacan el restringido número de gestos que son reconocidos y el requerir de software propietario para acceder a esta funcionalidad.

Por tal motivo, en [14] se propuso un sistema de reconocimiento gestual de las manos en tiempo real basado en un dispositivo cercano a infrarrojo. Este nuevo sistema destaca el uso de información infrarroja, la cual se adquiere por el mismo dispositivo Leap Motion, además de emplear técnicas y algoritmos de segmentación, extracción y clasificación para el reconocimiento de gestos manuales.

Como resultado se logró mejorar el rendimiento y los puntajes en el reconocimiento de gestos individuales.

Gentile et al. [15] presentaron dos estudios relacionados con la interacción gestual sin contacto con computadoras de escritorio y portátiles. El primer estudio tuvo como propósito dar respuesta al problema de elegir el gesto correcto para una determinada acción, en este caso se enfocó en la acción de acercar o alejar elementos mientras se interactúa con un monitor de computadora. Se propuso un experimento en el cual participaron veinticinco personas, como resultado se obtuvo un conjunto de cien gestos manuales que fueron a su vez clasificados en un total de diez clases y de los cuales después de realizar un análisis se eligieron los dos más comunes.

El segundo estudio comparó los dos gestos seleccionados mediante un nuevo experimento, el objetivo fue permitir a los usuarios explorar y manipular modelos 3D, la interfaz muestra dos manos virtuales que reproducen las manos del usuario y cuyos movimientos son capturados por un dispositivo Leap Motion. Se pidió a cada participante que utilizara ambos gestos de acercamiento con dos niveles posibles. Cuando se validaron los resultados del estudio se descubrió una preferencia a uno de los gestos que resultó ser menos demandante mentalmente, más usable y ergonómico.

En [16] Boudjelthia et al. evaluaron el rendimiento de utilizar gestos manuales capturados por el dispositivo LMC (*Leap Motion Controller*, Controlador Leap Motion) en la navegación Web. Se desarrolló una extensión para el navegador Google Chrome con ayuda del lenguaje JavaScript y las bibliotecas provistas por Google Chrome y LeapJS (para realizar la conexión con el dispositivo Leap Motion). La extensión soporta cuatro posibles acciones: refrescar página, navegar entre pestañas, zoom y navegación entre el historial.

Se evaluó el uso de la extensión con una muestra de diecinueve participantes, los cuales se capacitaron en el uso del dispositivo LMC y la extensión. Se prepararon cuatro tareas que representan a cada una de las acciones soportadas, se midió el tiempo tomado en completar la tarea y el número de errores realizados. Una vez analizados los resultados se detectó que el rendimiento del dispositivo LMC fue menor en comparación al obtenido por el dispositivo ratón o *tracking pad*.

En [17] Cui y Sourin argumentaron que las herramientas usadas en el modelado se basan en el uso de dispositivos rastreadores 2D, como lo son el ratón o el *tracking pad*, la naturaleza de estos dispositivos comienza a generar problemas, ya que algunas de las operaciones esenciales en el modelado 3D deben descomponerse en una serie de pasos que un dispositivo 2D maneje de forma más apropiada.

El reemplazar los dispositivos rastreadores 2D con dispositivos ópticos rastreadores del movimiento de la manos permite obtener naturalidad al manipular elementos de interfaces 3D, aunado a la existencia de opciones comerciales a un costo accesible, por lo que se presenta como una solución prometedora aunque introduce nuevos

problemas como lo son: el nerviosismo en las manos (que suele pasar cuando no hay una superficie física de soporte), liberación de salto (que causa desplazamiento del objeto virtual al extender los dedos para liberarlo) y oclusión de la mano (que impide el rastreo preciso de la mano).

Cui y Sourin indagaron sobre cómo las interfaces de modelado 3D podrían mejorarse con el rastreo óptico de las manos y diseñaron un patrón de interacción con el cual los problemas antes mencionados se minimizaron.

Breslauer et al. indagaron en [11] sobre la importancia del desarrollo en la investigación de la HCI (Human-computer interaction, Interacción humano-computadora), remarcaron que su finalidad es lograr que la interacción entre una persona y una computadora sea tan natural como sea posible, de ahí que el uso de gestos y movimientos de la mano sea una forma conveniente y atractiva para lograr esta forma de comunicación, de forma que permita concebir una interfaz casi invisible, discreta y que permita al usuario dominar el uso de una computadora. Para detectar gestos y movimientos de las manos existen distintos tipos de dispositivos sensores: aquellos que deben estar físicamente adheridos al usuario, o que deban ser sostenidos por el usuario y aquellos que rastrean los movimientos y gestos sin algún contacto físico.

Los dispositivos de la segunda categoría presentan la ventaja de ofrecer a los usuarios mayor libertad de interacción, permitiendo movimientos naturales sin necesidad del uso de guantes o dispositivos *wearables*. El término común para este tipo de interacción sin contacto por parte de los usuarios se denomina “*mid-air interaction*” y “*mid-air gestures*”.

Con el objetivo de obtener una interfaz que sea percibida como natural, varios estudios revelaron que la interacción bimanual se percibe como más natural que la unimanual, utilizando ambas manos con una separación de funciones entre mano dominante y no dominante se logró que la interacción *mid-air* fuera más rápida que los dispositivos de entrada 2D para tareas de tipo 3D.

El problema más prominente con respecto a los sensores ópticos como Leap Motion con relación a la interacción bimanual *mid-air* es el limitado rango del campo de visión, es por ello por lo que Bresaluer et al. exploraron un método para expandir el rango de visión del sistema para la interacción bimanual *mid-air* usando más de un sensor. Debido a que el software de Leap Motion no permite utilizar dos dispositivos al mismo tiempo, se utilizó otra computadora para recopilar los datos del segundo dispositivo, los cuales se compartieron entre computadoras por medio de un enlace red. El programa encargado de la sincronización y operación de los dispositivos se construyó mediante el software Unity 3D, mediante este programa se logró mejorar el rango de detección del movimiento de las manos, esto permite ofrecer una mejor impresión acercándose a los principios que menciona la HCI. Las mejoras permitirán que se utilice en aplicaciones donde se hace uso intensivo de la entrada de datos mediante gestos manuales.

En [18] Müller et al. describieron las técnicas principales para interactuar con interfaces de usuario basadas en pantallas, el uso del ratón es históricamente predominante debido a que se concibió en las primeras etapas del desarrollo de las interfaces de usuario. Entre las ventajas del uso del ratón se encontró que ofrece al usuario efectos como "*mouseover*" y "*hover*", que tienen el propósito de proveer ayuda y fomentar al usuario el sentimiento de control al mostrar información adicional. Los usuarios aprenden más acerca de la funcionalidad de los elementos de la interfaz, como el significado de un ícono o la función de un botón.

Actualmente existe una gran cantidad de computadoras equipadas con pantallas táctiles, las cuales usualmente carecen de una técnica similar al "*mouseover*" que permitiría a los usuarios primerizos de una interfaz gráfica explorar de forma segura la aplicación sin tener que ser cautelosos de cometer algún error. Las restricciones de las interfaces táctiles provocaron la investigación y desarrollo de nuevas técnicas que permitan corregir estas limitaciones, Müller et al. desarrollaron una nueva técnica de interacción llamada "*Fich*" que dota de *tooltips* y efectos "*fingerover*" a las pantallas táctiles, enriqueciendo potencialmente la interacción de pantallas táctiles para un amplio rango de aplicaciones y sitios web. Como parte del desarrollo de

Fich, se diseñó y construyó una aplicación personalizada para consultar el clima instalada en un dispositivo Microsoft Surface y se condujo un extensivo estudio que contó con la participación de 42 personas. El diseño de la interfaz de la aplicación presentó diversos elementos (menús, listas desplegables, pestañas, botones y etiquetas) para mostrar la información del clima, así como las operaciones disponibles que se ofrecen al usuario. Varios elementos se dotaron con *tooltips* que mostraban información adicional. La aplicación permite el uso de las tres modalidades descritas anteriormente: *Mouse*, *Touch* y *Fich*, dependiendo del modo de interacción, los elementos de la interfaz se comportaron de distinta forma. El uso de la modalidad *Fich* permite utilizar el espacio al frente de la pantalla como entrada de usuario en conjunto con la entrada táctil, cuando el usuario mueve su dedo enfrente de la pantalla, el cursor lo sigue instantáneamente.

Para implementar la aplicación Web se utilizó HTML (*HyperText Markup Language*, Lenguaje de Marcado de Hipertexto), JavaScript y CSS (*Cascading Style Sheets*, Hoja de Estilos en Cascada) en conjunto con un servidor Flask escrito en Python el cual se encargó de capturar los datos de la interacción, además de administrar las distintas modalidades en las que se maneja la interfaz de usuario. Para rastrear el movimiento del dedo del usuario se usó un dispositivo Leap Motion, se utilizó el kit de desarrollo de software Leap Motion Desktop v2, así como el cliente oficial para JavaScript llamado LeapJS.

Se condujo un estudio en el que participaron 42 personas, el procedimiento consistió en una prueba de uso de la aplicación, se evaluaron las tres modalidades de uso y se pidió a los participantes que contestaran una serie de cuestionarios para conocer la experiencia que tuvieron con cada modalidad. Un aspecto que también se evaluó fue la capacidad que esta técnica tiene de proveer serendipia, esto es, realizar descubrimientos accidentales de nuevos elementos de la interfaz de usuario de la aplicación, debido a que esto es complicado de medir mediante un cuestionario, se realizó un análisis del uso de la modalidad *Fich* de cada participante de modo que se obtuviera un puntaje que a su vez se validó mediante una serie de preguntas realizadas al final de la prueba.

Los resultados del estudio mostraron que *Fich* es significativamente superior a los métodos tradicionales de entrada como el ratón o una pantalla táctil considerando la calidad hedónica de la experiencia de usuario, pero significativamente inferior considerando la calidad pragmática de la experiencia de usuario. Los resultados en la calidad pragmática son en parte atribuidos a las limitaciones de la tecnología actual de seguimiento del movimiento de las manos.

En [2] Bachmann et al. presentaron un estudio sobre el estado del arte de las técnicas de interacción humano-computadora (HCI) con un enfoque en el campo de la interacción en tercera dimensión, además de incluir una visión general de los dispositivos de interacción actualmente disponibles, sus aplicaciones y los métodos para el diseño y reconocimiento de gestos.

Se dio una especial atención a las interfaces basadas en el controlador Leap Motion (LMC) y los correspondientes métodos de diseño y reconocimiento de gestos. Las principales contribuciones del estudio fueron: La disponibilidad actual de dispositivos interactivos en tres dimensiones, los métodos para el diseño y reconocimiento de gestos, aplicaciones de uso, estado del arte en el diseño de interacción y evaluación de métodos para estos diseños.

En Schioppo et al. [19] argumentaron que el reconocimiento de gestos ganó atención de la comunidad científica y que se atribuye a su aplicación en campos como la robótica, la interacción humano-computadora, el reconocimiento de lenguaje de señas y el diseño de interfaces. También tiene éxito debido a dispositivos como el Leap Motion, una de las principales aplicaciones que utilizan este dispositivo y ha sido exitosas es el del reconocimiento de lenguaje de señas.

Los autores propusieron un enfoque para el reconocimiento de lenguaje de señas que hace uso de un *headset* de realidad virtual en conjunto con Leap Motion. Este enfoque utiliza las características de Leap Motion, capturando desde una vista egocentrista. Se analizó cuáles fueron las mejores características y se comprobó la eficacia del sistema propuesto con las veintiséis letras del alfabeto del lenguaje de señas americano en un ambiente virtual e inmersivo con la ayuda de una aplicación para aprender el lenguaje de señas. Las pruebas arrojaron resultados positivos, se

logró un porcentaje de certeza del 98.33% mediante la aplicación del algoritmo Random Forest y un 97.1% por medio del uso de una red neuronal prealimentada.

Guzsvinecz et al. investigaron en [20] sobre los dispositivos rastreadores de movimiento; detectaron que el problema con aquellos dispositivos que permiten el rastreo más preciso es que son costosos, por lo cual, muchas personas no podrían adquirirlos. Se propuso el uso de dispositivos sensores de bajo costo, como el sensor Kinect de Microsoft para el rastreo del movimiento del cuerpo completo por medio de cámaras de profundidad y el controlador Leap Motion (LMC) para el rastreo del movimiento de la mano para determinar si podrían sustituir a sus contrapartes más costosas, teniendo en cuenta su exactitud, precisión, precio y los algoritmos de reconocimiento de gestos existentes. Cuando se compararon los sensores, Guzsvinecz et al. llegaron a la conclusión de que los dispositivos Kinect y Leap Motion son opciones viables para reemplazar a sensores más costosos en la industria o en el mercado. El dispositivo LMC es más exacto, aunque no llegó a alcanzar los niveles especificados por su fabricante, su exactitud y precisión fueron mayores a los alcanzados por Kinect, esto en parte es debido a que tienen un enfoque de rastreo distinto. Su bajo costo los hace adecuados para su uso en múltiples campos de investigación, como la educación, rehabilitación, reconocimiento de gestos y entretenimiento. Mientras que no son dispositivos defectuosos, existen métodos de software que permiten mejorar sus capacidades.

En [21] Potter et al. presentaron un estudio exploratorio acerca de la idoneidad del controlador Leap Motion para reconocer el lenguaje de señas australiano (Auslan), poniendo a prueba las capacidades del dispositivo para reconocer las señas de Auslan realizadas en el campo de visión del controlador. La meta del proyecto es producir tecnología eficiente y asequible para las familias de jóvenes sordos y con problemas de audición mientras aprenden el lenguaje de señas, en especial Auslan.

La investigación realizada a la fecha en el área de reconocimiento de señas promueve el desarrollo de dispositivos entre los cuales se encuentra la tecnología de guantes, sensores de pulsera, cámaras 2D y 3D y la plataforma Kinect. Se aplicaron pruebas para establecer cómo funcionaba el controlador y para entender

la interacción básica, para el propósito del estudio se usó el alfabeto Auslan para probar la funcionalidad del controlador. Se eligió el alfabeto por la relativa facilidad de las señas individuales, así como por el diverso rango de movimientos involucrados en el alfabeto. Basado en la evaluación de las pruebas se concluyó que la API (*Application Programming Interface*, Interfaz de Programación de aplicaciones) que da soporte al dispositivo no está lista para interpretar el rango completo del lenguaje de señas, aunque funciona para reconocer señas básicas, sin embargo, no es apropiado para señas complejas especialmente aquellas que requieren contacto con la cara o cuerpo.

2.2 Análisis Comparativo

La Tabla 2.1 muestra un análisis comparativo de los artículos relacionados que se obtuvieron para este proyecto.

Tabla 2.1. Análisis comparativo de artículos.

Artículo	Problema	Contribución	Tecnología	Resultado	Estado
Mantecón et al. [14]	Se encontró que hay un número reducido de gestos de la mano que son reconocidos por el análisis de la información extraída del esqueleto de la mano.	Sistema de reconocimiento de gestos de las manos en tiempo real basado en un dispositivo cercano a infrarrojo.	Aprendizaje automático.	Se logró mejorar el rendimiento y el número de gestos manuales detectados por el dispositivo Leap Motion.	Finalizado.
Gentile et al. [15]	Elegir el gesto correcto para una determinada acción en una interfaz gráfica.	Estudio comparativo de gestos asociados a la acción de acercar/alejar.	Leap Motion	Se logró detectar un gesto manual para la acción de acercar/alejar menos demandante mentalmente, más usable y ergonómico.	Propuesta.

Tabla 2.1 Análisis comparativo de artículos (continuación).

Artículo	Problema	Contribución	Tecnología	Resultado	Estado
Boudjelthia et al. [16]	Obtener respuesta de si es viable el uso de gestos manuales para controlar la navegación.	Una extensión para el navegador Google Chrome conectada a la interfaz Leap Motion.	JavaScript.	Se detectó que el rendimiento fue menor comparándolo a la navegación realizada con un dispositivo ratón o *tracking pad*.	Finalizado.
Cui y Sourin [17]	Se encontró que los dispositivos rastreadores 2D (ratón, *tracking pad*) son ineficientes para realizar operaciones en ambientes 3D.	Se diseñó un patrón de interacción que hace uso de dispositivos ópticos rastreadores 3D de las manos, como el Leap Motion, para interactuar con ambientes 3D.	Leap Motion JavaScript	Al usar dispositivos de rastreo 3D se introdujeron nuevos problemas (nerviosismo en manos, oclusión, desplazamientos involuntarios), el patrón de interacción propuesto logró minimizar estos problemas.	Finalizado.

Tabla 2.1 Análisis comparativo de artículos (continuación).

Artículo	Problema	Contribución	Tecnología	Resultado	Estado
Breslauer et al. [11]	El rango del campo de visión del dispositivo Leap Motion es limitado para lograr realizar operaciones con dos manos, de forma que sea percibida como más natural.	Se investigó un método para ampliar el rango de visión en el eje horizontal, utilizando dos sensores Leap Motion.	Leap Motion Unity	Se logró mejorar el rango de detección del movimiento de las manos, de modo que se ofrece al usuario una mayor superficie de interacción en tres dimensiones.	Finalizado.
Müller et al. [18]	La falta de técnicas para implementar elementos de ayuda, como *tooltips*, en aplicaciones que hacen uso de interfaces táctiles.	Se desarrolló una técnica de interacción llamada “Fich” que dota de elementos de ayuda a las pantallas táctiles por medio del dispositivo Leap Motion.	Leap Motion HTML JavaScript CSS Python	Se diseñó y construyó una aplicación para ponerla a prueba mediante un estudio. Los resultados mostraron que Fich es superior en calidad hedónica a los métodos tradicionales de entrada, pero inferior a la calidad pragmática de estos.	Finalizado.

Tabla 2.1 Análisis comparativo de artículos (continuación).

Artículo	Problema	Contribución	Tecnología	Resultado	Estado
Bachmann et al. [2]	Conocer el estado del arte de las técnicas de interacción humano-computadora, con énfasis en la interacción en tres dimensiones.	Se realizó un estudio que contribuye a dar una visión general de los dispositivos de interacción actualmente disponibles, sus aplicaciones y métodos para el reconocimiento de gestos.	Leap Motion	Se logró la revisión de métodos y técnicas de evaluación para la interacción en tres dimensiones, se espera que esto sirva como un recurso útil para los investigadores involucrados en el diseño e implementación de nuevos conceptos de HCI.	Finalizado.
Schioppo et al. [19]	Dar respuesta a la incógnita de si es posible realizar el reconocimiento de señas americano en un ambiente de realidad virtual.	Se montó un dispositivo que utilizó un *headset* y un dispositivo Leap Motion, junto con una aplicación para aprender el lenguaje de señas.	Leap Motion HTC Vive	Se logró un porcentaje de certeza para el reconocimiento de señas del 98.33% por medio del algoritmo de clasificación Random Forest y un 97.1% con ayuda de una red neuronal prealimentada.	Finalizado.

Tabla 2.1 Análisis comparativo de artículos (continuación).

Artículo	Problema	Contribución	Tecnología	Resultado	Estado
Guzsvinecz et al. [20]	Los dispositivos rastreadores de movimiento más precisos tienen la desventaja de ser costosos.	Se generó un estudio en el que se analizaron las características de los dispositivos Kinect y Leap Motion.	Kinect Leap Motion	Se llegó a la conclusión de que el bajo costo de ambos dispositivos los hace idóneos para su uso en múltiples campos de investigación, educación, rehabilitación, entre otros.	Finalizado.
Potter et al. [21]	Conocer la idoneidad del controlador Leap Motion para reconocer el lenguaje de señas australiano (Auslan).	La evaluación para determinar la viabilidad de uso del dispositivo para producir tecnología asequible para las familias de jóvenes sordos y con problemas de audición.	Leap Motion	Se llegó a la conclusión de que la API del dispositivo no está lista para interpretar el rango completo de señas de Auslan, funciona para reconocer señas básicas, pero no es apropiado para señas complejas que involucran el contacto con la cara o cuerpo.	Finalizado.

Después de realizar el análisis de los artículos previos, es posible identificar que existe un gran interés en la investigación y desarrollo de soluciones que emplean el dispositivo controlador Leap Motion, debido a su bajo costo, tamaño y funciones, es muy atractivo para un amplio rango de aplicaciones en diversos ámbitos como la interacción y modelado 3D, realidad virtual, la educación, la rehabilitación o el reconocimiento de lenguaje de señas. Al revisar el contenido de los trabajos relacionados, no se encontró alguna propuesta de alguna biblioteca que sirva de base para construir aplicaciones que utilicen interfaces gestuales a través del uso de Leap Motion, por lo cual se consideró viable el desarrollo del proyecto.

2.3 Propuesta de solución

La Tabla 2.2 presenta las tecnologías de información seleccionadas para emplearse en el desarrollo del proyecto.

Tabla 2.2. Selección de tecnologías de información.

Biblioteca para desarrollo de GUI en Java	IDE	Medio de almacenamiento	Metodología	Sistema de control de versiones
JavaFX	IntelliJ IDEA	XML	Scrum	Git

2.3.1 Justificación de la solución

La elección de la solución propuesta se basa en el análisis de las distintas tecnologías de la información, los aspectos de este análisis se discuten a continuación:

Biblioteca para desarrollo de GUI en Java: La elección de JavaFX se tomó con base en ciertas características que ofrece, como la flexibilidad para integrar el patrón MVC y el uso de documentos basados en XML para construir la interfaz de usuario, esto lo hace idóneo para el proyecto propuesto.

Entorno de desarrollo integrado: Se eligió a IntelliJ IDEA como entorno de desarrollo, las herramientas que provee están enfocadas a aumentar la productividad del desarrollador y soportar proyectos Java de distinta complejidad.

Medio de almacenamiento: Se decidió utilizar XML como medio de almacenamiento debido a su capacidad de adaptabilidad y extensibilidad al dominio que se plantea en este proyecto, el soporte a datos complejos es una ventaja que tiene sobre JSON, el cual solo tiene soporte a tipos de datos primitivos.

Metodología de desarrollo: Se optó por Scrum para guiar la construcción del proyecto, su enfoque iterativo e incremental se adapta bien para obtener software de calidad que cumpla con los requerimientos y que sea adapte a los posibles cambios.

Sistema de control de versiones: Git es el sistema de control de versiones por excelencia en la industria al momento de escribir este documento, se basa en un modelo de repositorio distribuido, es gratuito y soportado por los sistemas operativos más utilizados en la actualidad, además de tener una extensa documentación.

Capítulo 3. Aplicación de la Metodología

El presente capítulo tiene como objetivo presentar el desarrollo de la solución, siguiendo las pautas recomendadas en la metodología seleccionada, que en este caso fue Scrum.

3.1 Descripción de la solución

En la siguiente sección se realiza una descripción detallada de la solución propuesta.

A continuación, se describen los módulos propuestos de la arquitectura preliminar presentada en la Figura 3.1 para dar solución al problema planteado.

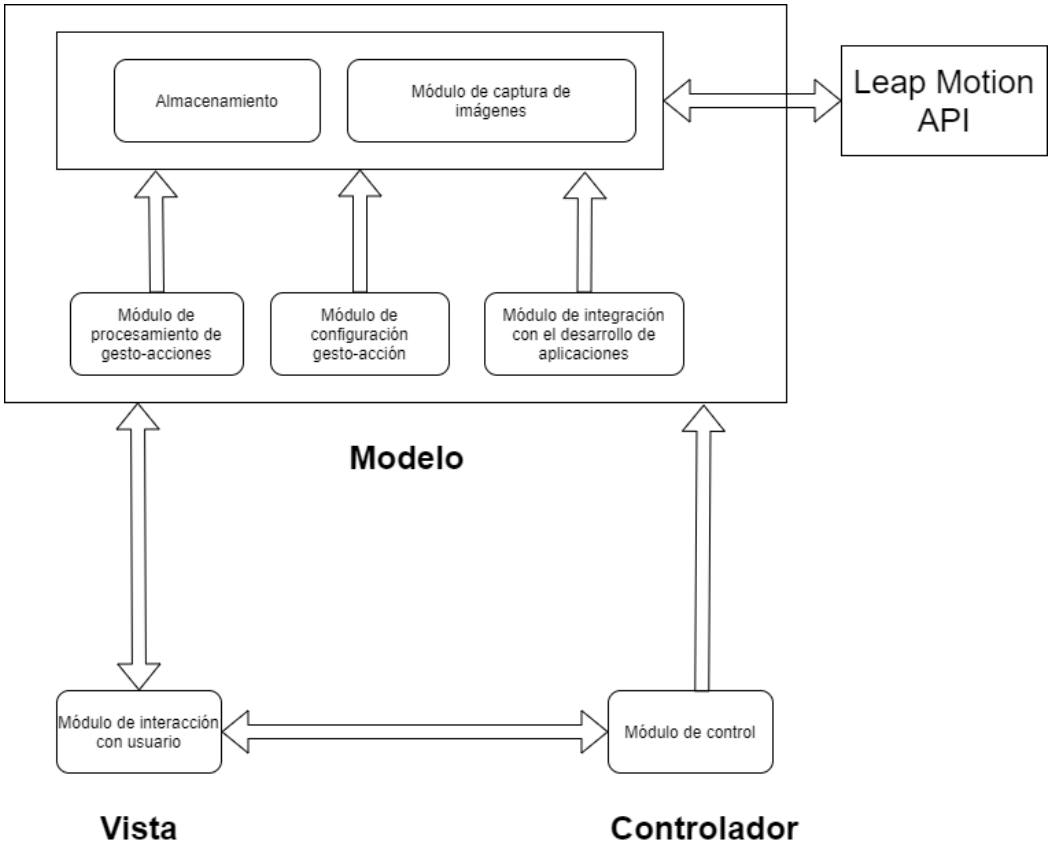


Figura 3.1 Arquitectura propuesta.

3.1.1 Módulo de captura de imágenes

Este módulo realiza la captura de imágenes correspondientes a gestos realizados con las manos a través del dispositivo Leap Motion, la comunicación se lleva a cabo a través de la API provista por el fabricante.

3.1.2 Módulo de configuración consumidor-gesto

Este módulo permite definir el gesto que está asociado a una acción en la aplicación. El resultado de esta configuración se almacena en un archivo, para su posterior integración en el desarrollo de una aplicación Java.

3.1.3 Módulo de procesamiento de gesto-acciones

Una vez realizada la configuración gesto-acción, el archivo resultante es analizado mediante este módulo, de tal forma que, apoyándose con el módulo de captura de imágenes, se compara de forma continua el gesto almacenado en el archivo con las imágenes capturadas por el dispositivo Leap Motion. Al haber una coincidencia, la acción correspondiente es ejecutada.

3.1.4 Módulo de integración con el desarrollo de aplicaciones

Este módulo se encarga de integrar las funciones ya definidas para utilizarlas en el desarrollo de una aplicación, de tal forma que el archivo ejecutable de la aplicación incluya todos los elementos necesarios para poner en marcha la funcionalidad deseada.

3.1.5 Módulo de interacción con usuario

Muestra al usuario una interfaz gráfica para interactuar con el módulo de control.

3.1.6 Módulo de control

El módulo de control se encarga de responder a las acciones que el usuario indica en la interfaz de usuario, sirviendo de intermediario con los distintos módulos que provee la biblioteca.

3.2 Marco de trabajo Scrum

En esta sección se describe la aplicación de la metodología Scrum, en las tablas 3.1 a 3.11 se describen las historias de usuario, las cuales guiaron el desarrollo del proyecto.

3.2.1 Historias de usuario

Tabla 3.1 Historia de usuario HU_INV_01: Analizar las características del dispositivo Leap Motion.

Código: HU_INV_01	**Nombre**: Analizar las características del dispositivo Leap Motion.
Tipo HU: Análisis	**Actor**: Usuario
HU Relacionadas:	
Descripción: Se requiere analizar las funciones que provee el dispositivo Leap Motion.	

Tabla 3.2 Historia de usuario HU_INV_02: Analizar el kit de desarrollo de software del dispositivo Leap Motion.

Código: HU_INV_02	**Nombre**: Analizar el kit de desarrollo de software del dispositivo Leap Motion.
Tipo HU: Análisis	**Actor**: Usuario
HU Relacionadas: HU_INV_01	
Descripción: Se pretende analizar el funcionamiento técnico del dispositivo y del kit de desarrollo de aplicaciones, ofrecido por el fabricante.	

Tabla 3.3 Historia de usuario HU_DGC: Definir los gestos y comandos que se incorporarán a la biblioteca.

Código: HU_DGC	**Nombre**: Definir los gestos y comandos que se incorporarán a la biblioteca.
Tipo HU: Análisis	**Actor**: Usuario
HU Relacionadas: HU_INV_02	
Descripción: Los gestos y comandos seleccionados servirán para desarrollar aplicaciones con interfaces gestuales.	

Tabla 3.4 Historia de usuario HU_ARQ: Establecer una arquitectura que sirva de base al desarrollo de la biblioteca.

Código: HU_ARQ	**Nombre**: Establecer una arquitectura que sirva de base al desarrollo de la biblioteca.
Tipo HU: Diseño	**Actor**: Usuario
HU Relacionadas: HU_INV_01, HU_INV_02, HU_DGC.	
Descripción: Con base en el análisis del dispositivo Leap Motion, se requiere diseñar una arquitectura que integre los módulos necesarios para llevar a cabo la detección y reacción a gestos.	

Tabla 3.5. HU_CAP_IMG: Desarrollar módulo de captura de *frames*.

Código: HU_CAP_IMG	**Nombre**: Desarrollar módulo de captura de *frames*.
Tipo HU: Funcional	**Actor**: Usuario
HU Relacionadas: HU_ARQ	
Descripción: Este módulo debe realizar la conexión con el dispositivo Leap Motion a través del API provista por el fabricante del dispositivo, para recuperar y comunicar los datos del seguimiento de las manos.	

Tabla 3.6 Historia de usuario HU_REC_GES: Desarrollar módulo de reconocimiento de gestos.

Código: HU_REC_GES	**Nombre**: Desarrollar módulo de reconocimiento de gestos.
Tipo HU: Funcional	**Actor**: Usuario
HU Relacionadas: HU_ARQ	
Descripción: Este módulo debe proveer un medio para detectar gestos realizados con las manos, utilizando los datos provistos por el dispositivo Leap Motion.	

Tabla 3.7 Historia de usuario HU_CONF_CG: Desarrollar módulo de configuración consumidor-gesto.

Código: HU_CONF_CG	**Nombre**: Desarrollar módulo de configuración consumidor-gesto.
Tipo HU: Funcional	**Actor**: Usuario
HU Relacionadas: HU_ARQ	
Descripción: Este módulo permitirá definir la asociación entre un gesto y un consumidor, en donde el último está interesado en la ocurrencia del primero.	

Tabla 3.8 Historia de usuario HU_PRO_GA: Desarrollar módulo de procesamiento de acciones.

Código: HU_PRO_GA	**Nombre**: Desarrollar módulo de procesamiento de acciones.
Tipo HU: Funcional	**Actor**: Usuario
HU Relacionadas: HU_ARQ	
Descripción: Este módulo se encarga de llevar a cabo una acción relevante en el contexto de una interfaz de usuario.	

Tabla 3.9 Historia de usuario HU_INT_USU: Desarrollar módulo de interacción con usuario.

Código: HU_INT_USU	**Nombre**: Desarrollar módulo de interacción con usuario.
Tipo HU: Funcional	**Actor**: Usuario
HU Relacionadas: HU_ARQ	
Descripción: Muestra al usuario una interfaz gráfica para interactuar con el módulo de control.	

Tabla 3.10 Historia de usuario HU_CONTROL: Desarrollar módulo de control.

Código: HU_CONTROL	**Nombre**: Desarrollar módulo de control.
Tipo HU: Funcional	**Actor**: Usuario
HU Relacionadas: HU_ARQ	
Descripción: El módulo de control se encargará de responder a las acciones que el usuario indique en la interfaz de usuario, sirviendo de intermediario con los distintos módulos que provee la biblioteca	

Tabla 3.11 Historia de usuario HU_INT_DES: Desarrollar módulo de integración con el desarrollo de aplicaciones.

Código: HU_INT_DES	**Nombre**: Desarrollar módulo de integración con el desarrollo de aplicaciones.
Tipo HU: Funcional	**Actor**: Usuario
HU Relacionadas: HU_ARQ	
Descripción: El módulo de integración es requerido para integrar las configuraciones, requerimientos o archivos necesarios para exportar las funciones de reconocimiento de gestos y ejecución de acciones.	

3.2.2 Sprints

Un *Sprint* es un periodo en el cual se trabaja para completar tareas establecidas. Los *Sprints* propuestos para realizar el proyecto se muestran en la tabla 3.12.

Tabla 3.12 *Sprints* e historias de usuario del proyecto.

Sprint	**Historias Planificadas**
Sprint 1	HU_INV_01: Analizar las características del dispositivo Leap Motion.
	HU_INV_02: Analizar el kit de desarrollo de software del dispositivo Leap Motion.

Sprint	**Historias Planificadas**
	HU_DGC: Definir los gestos y comandos que se incorporarán a la biblioteca.
Sprint 2	HU_ARQ: Establecer una arquitectura que sirva de base al desarrollo de la biblioteca.
Sprint 3	HU_CAP_IMG: Desarrollar módulo de captura de *frames*.
	HU_REC_GES: Desarrollar módulo de reconocimiento de gestos.
Sprint 4	HU_CONF_CG: Desarrollar módulo de configuración consumidor-gesto.
Sprint 5	HU_PRO_GA: Desarrollar módulo de procesamiento de acciones.
Sprint 6	HU_INT_USU: Desarrollar módulo de interacción con usuario.
	HU_CONTROL: Desarrollar módulo de control.
Sprint 7	HU_INT_DES: Desarrollar módulo de integración con el desarrollo de aplicaciones.

3.2.3 Sprint 1

Las tareas encomendadas a realizarse durante este Sprint fueron las siguientes:

- HU_INV_01: Analizar las características del dispositivo Leap Motion.
- HU_INV_02: Analizar el kit de desarrollo de software del dispositivo Leap Motion.
- HU_DGC: Definir los gestos y comandos que se incorporarán a la biblioteca.

3.2.3.1 HU_INV_01: Analizar las características del dispositivo Leap Motion

El dispositivo Leap Motion (o Controlador Leap Motion) es un dispositivo de hardware fabricado por UltraLeap, en un intento por ofrecer la tecnología de seguimiento de las manos a un costo accesible. Para lograrlo, el dispositivo emplea una serie de LEDs infrarrojos que iluminan de forma continua una zona que se extiende hasta 60 cm de profundidad en un campo de visión de 140º por 120º. Un par de cámaras infrarrojas capturan imágenes a una frecuencia que va de 20 a 200 veces por segundo. Por último, los datos de las imágenes son comunicados a la computadora a través de un puerto USB (*Universal Serial Bus*, Bus Universal en Serie). La Figura 3.2 muestra una representación del dispositivo en funcionamiento.

Figura 3.2 Representación del funcionamiento del dispositivo Leap Motion [22].

El software del dispositivo Leap Motion ofrece un modelo de datos que incluye información del seguimiento del movimiento de los brazos, manos y los huesos que dan forma a los dedos, entre los que se encuentran los metacarpianos y falanges (proximales, medias y distales). La Figura 3.3 muestra en un esquema las partes de la mano a las que da seguimiento.

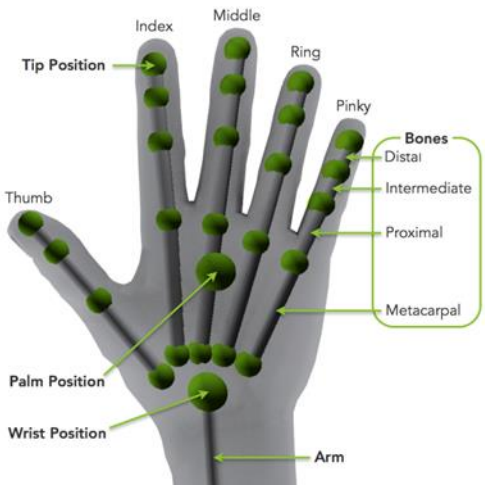


Figura 3.3 Segmentos de la mano identificados por el dispositivo Leap Motion [23].

El dispositivo provee los datos de rastreo mediante un sistema de coordenadas en tres dimensiones (Figura 3.4) o alternativamente con imágenes (sin procesar) de cada una de las cámaras. La Figura 3.5 muestra un ejemplo del dispositivo en funcionamiento.

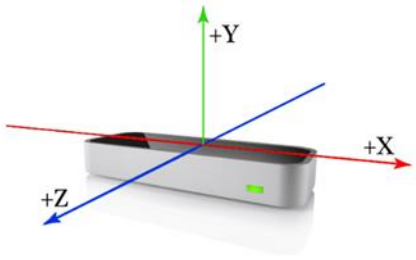


Figura 3.4 Ejes de coordenadas del dispositivo Leap Motion [24].

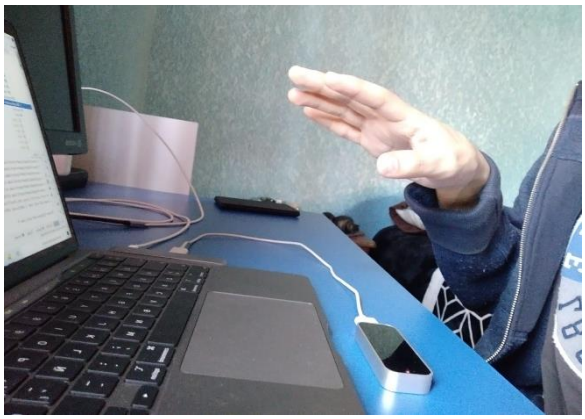


Figura 3.5. Ejemplo de uso del dispositivo Leap Motion.

3.2.3.2 HU_INV_02: Analizar el kit de desarrollo de software del dispositivo Leap Motion

El kit de desarrollo de aplicaciones está basado en la interfaz de desarrollo de aplicaciones escrita en el lenguaje de programación C, denominada LeapC y es utilizada para la construcción de los módulos de integración con los motores gráficos Unity y Unreal Engine. El historial de publicaciones del kit de desarrollo de aplicaciones reporta los lanzamientos de las versiones 2.x, 3.x, 4.x (código Orion) y 5.x (código Gemini). Además, ofrece un *binding* (mecanismo que permiten utilizar las funciones escritas en un lenguaje de programación, en este caso C, en un lenguaje distinto) para el lenguaje C# en las últimas versiones, y una serie de *legacy bindings* (limitado a las versiones 3.X y 2.X) para los lenguajes C++, Java, JavaScript, Python y Objective–C.

El kit de desarrollo de software emplea un esquema basado en clases para representar las distintas entidades que integran su modelo de seguimiento, además de una serie de clases de utilería que facilitan trabajar con el sistema de coordenadas de tres dimensiones. La Figura 3.6 presenta un esquema con las entidades que se ofrecen en el kit de desarrollo de software.

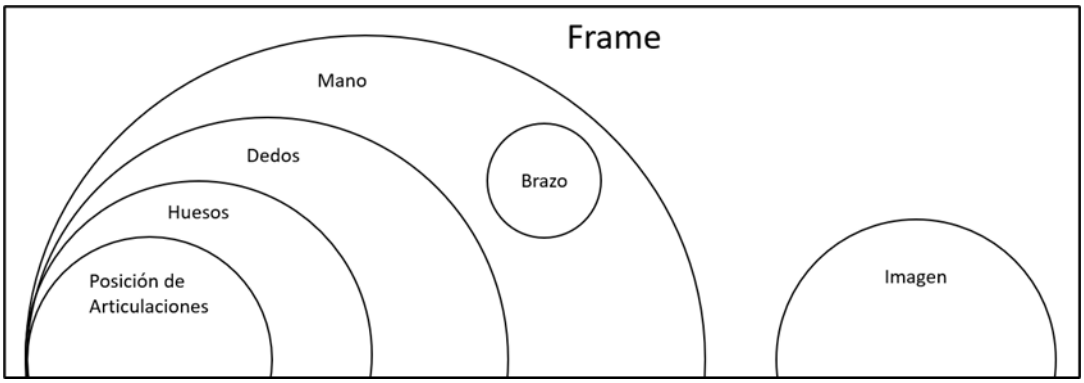


Figura 3.6 Clases provistas por el kit de desarrollo de aplicaciones.

La clase `Frame` (Marco) provee acceso a las demás entidades y es considerada como la raíz del modelo de datos. Una instancia de `Frame` es creada en cada intervalo de actualización por el dispositivo Leap Motion. El objeto mano describe la posición y orientación de la mano, realiza el seguimiento del movimiento entre *frames* y contiene una lista de dedos asociados con esa mano. Los objetos `Arm` (Brazo) describen la posición, dirección y orientación del brazo. Los objetos brazo solo son accedidos a través de un objeto `Mano`. Los objetos `Bone` (Hueso) representan la posición y orientación de un hueso. Los huesos rastreados incluyen a los metacarpianos y falanges de los dedos y pulgar. Los objetos `Image` (Imagen) proporcionan los datos del sensor sin procesar y la cuadrícula de calibración para las cámaras Leap Motion. La clase `Vector` describe puntos y direcciones, provee de una serie de funciones matemáticas para trabajar con vectores. La clase `Matrix` (Matriz) representa rotaciones y otras transformaciones devueltas por algunas funciones en el API.

Es posible identificar una región en forma de caja completamente dentro del campo de visión del controlador Leap Motion, esta abstracción permite que sea fácil mapear las posiciones registradas por el dispositivo a un sistema de coordenadas distinto, al proveer coordenadas normalizadas para manos, dedos y herramientas que se encuentren dentro de la caja. Dentro de la terminología de la documentación oficial de Leap Motion, se conoce como caja de interacción y es representada por la clase `InteractionBox`. La Figura 3.7 muestra un ejemplo del campo de visión del dispositivo Leap Motion y la caja de interacción.

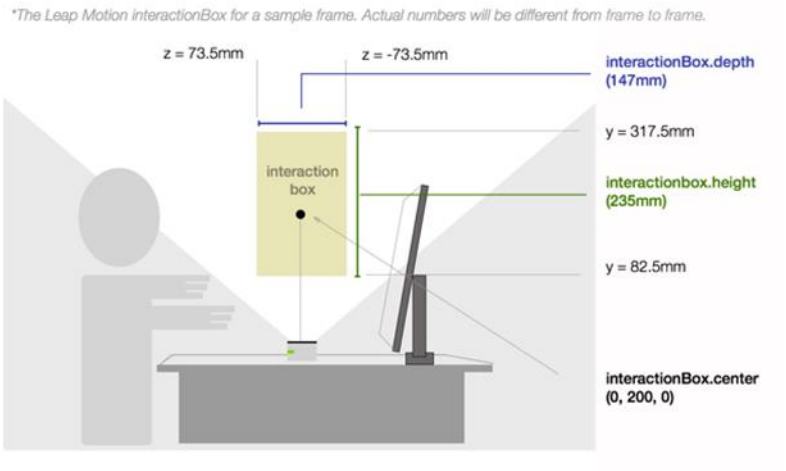


Figura 3.7 Campo de visión disponible efectivo mediante la caja de interacción [24].

3.2.3.3 HU_DGC: Definir los gestos y comandos que se incorporarán a la biblioteca

En esta sección se presentan los gestos y comandos que se incluirán en la biblioteca, los cuales se seleccionaron de un estudio publicado por el fabricante del dispositivo Leap Motion [25], en él se describieron los resultados de pruebas realizadas para lograr controlar los instrumentos de un automóvil mediante gestos realizados con las manos. Se tomaron en cuenta distintos aspectos a evaluar de cada uno de los siete gestos candidatos, los aspectos se describen a continuación.

- Usabilidad. Tiene que ver con cuan fácil es de aprender la forma y movimiento del gesto.
- Confiabilidad. Se refiere a la capacidad del dispositivo para interpretar correctamente el gesto.
- Confort. Se define como cuán difícil es de realizar un gesto, un mayor confort indica que un gesto se realiza fácilmente y sin esfuerzo.

Se eligieron cinco de los siete gestos propuestos en el estudio para integrarlos al funcionamiento de la biblioteca, se dio prioridad a aquellos que cuentan con un nivel mayor de confiabilidad. La selección se describe a continuación.

3.2.3.3.1 Golpe suave con la mano

En este gesto, el usuario dispone su mano horizontalmente y la mueve hacia abajo y hacia arriba. La Figura 3.8 representa el recorrido de la mano en este gesto.

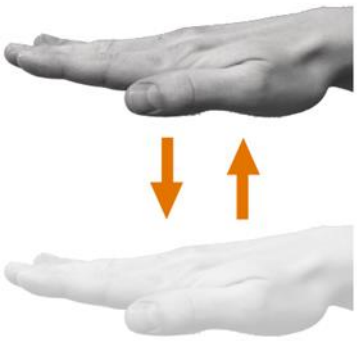


Figura 3.8 Golpe suave con la mano [25].

Algunos ejemplos de comandos con los que se recomienda usar son:

- Confirmar
- Encender/Apagar
- Pausar/Continuar

3.2.3.3.2 Golpe suave con los dedos

En este gesto, el brazo se mantiene fijo y únicamente los dedos se mueven hacia abajo y hacia arriba. La Figura 3.9 señala el movimiento que se debe realizar para completar este gesto.

Figura 3.9 Golpe suave con los dedos [25].

Requiere menos esfuerzo que el golpe suave con la mano y se recomienda utilizar con los comandos:

- Confirmar.

- Encender/Apagar.

3.2.3.3.3 Pellizco

Este gesto consiste en juntar el dedo índice con el pulgar, manteniendo los dedos restantes abiertos. La Figura 3.10 indica la pose recomendada para realizar este gesto.

Figura 3.10 Pellizco [25].

Es útil para:

- Ajustar valores (por ejemplo, para el volumen).
- Seleccionar una opción en un menú.

3.2.3.3.4 Pose con los dedos

En esta serie de gestos el usuario muestra una cantidad de gestos al dispositivo. La Figura 3.11 muestra las distintas poses disponibles en este gesto.

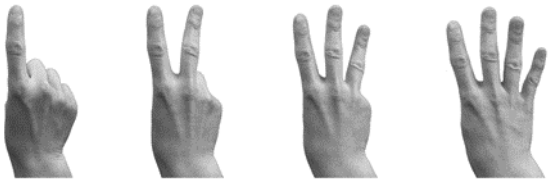


Figura 3.11 Poses con los dedos [25].

Se recomienda para realizar las siguientes acciones:

- Seleccionar un modo o característica.
- Elegir una cantidad de un ítem en particular.

3.2.3.3.5 Giro de mano

Con este gesto el usuario inicia con la mano abierta y la palma hacia abajo, el usuario gira la mano y luego la regresa a su posición inicial en un movimiento fluido. La Figura 3.12 muestra las tres partes que componen este gesto.

Figura 3.12 Giro de mano [25].

Este gesto podría ser utilizado para:

- Ir a una pantalla anterior.
- Cancelar una acción.

Con la adición de esta lista, se introduce a la biblioteca la funcionalidad de reconocimiento de gestos, que deberá identificar a través de los datos que provee el dispositivo Leap Motion cuando un gesto es realizado.

3.2.4 Sprint 2

La tarea asignada a este Sprint fue la siguiente:

- HU_ARQ: Establecer una arquitectura que sirva de base al desarrollo de la biblioteca.

3.2.4.1 HU_ARQ: Establecer una arquitectura que sirva de base al desarrollo de la biblioteca

Como parte de las funciones que debe cubrir la biblioteca y tomando en cuenta la facultad de realizar el reconocimiento del conjunto de gestos, se decidió utilizar herramientas especializadas en el tratamiento de este tipo de tareas. Dentro del amplio ecosistema de herramientas que implementan algoritmos de reconocimiento de patrones y aprendizaje automático, destaca el uso de la biblioteca de redes neuronales *Keras* y la plataforma *TensorFlow* escritas en el lenguaje de

programación *Python*. De forma que, para emplear dichas herramientas en el desarrollo de la biblioteca, se decidió buscar un patrón de arquitectura que permita implementar una parte de los módulos en Python y otra parte en el lenguaje Java.

Dentro de los distintos patrones de arquitectura de software, la arquitectura dirigida por eventos se posiciona como una arquitectura flexible y que ofrece bajo acoplamiento, de tal forma que cada uno de sus componentes tiene poco o nulo conocimiento del funcionamiento de otros componentes. Está enfocada en la producción, detección y reacción a eventos. En el dominio de este trabajo, la denominación de “evento” se asigna a la obtención de nuevos datos del dispositivo Leap Motion y a la detección de un gesto realizado con las manos.

Debido al cambio de patrón arquitectónico en el que se basa el desarrollo de la biblioteca, se actualizó la organización de los módulos presentados en la Figura 3.1 y se presentan adaptados a la arquitectura dirigida por eventos en la Figura 3.13.

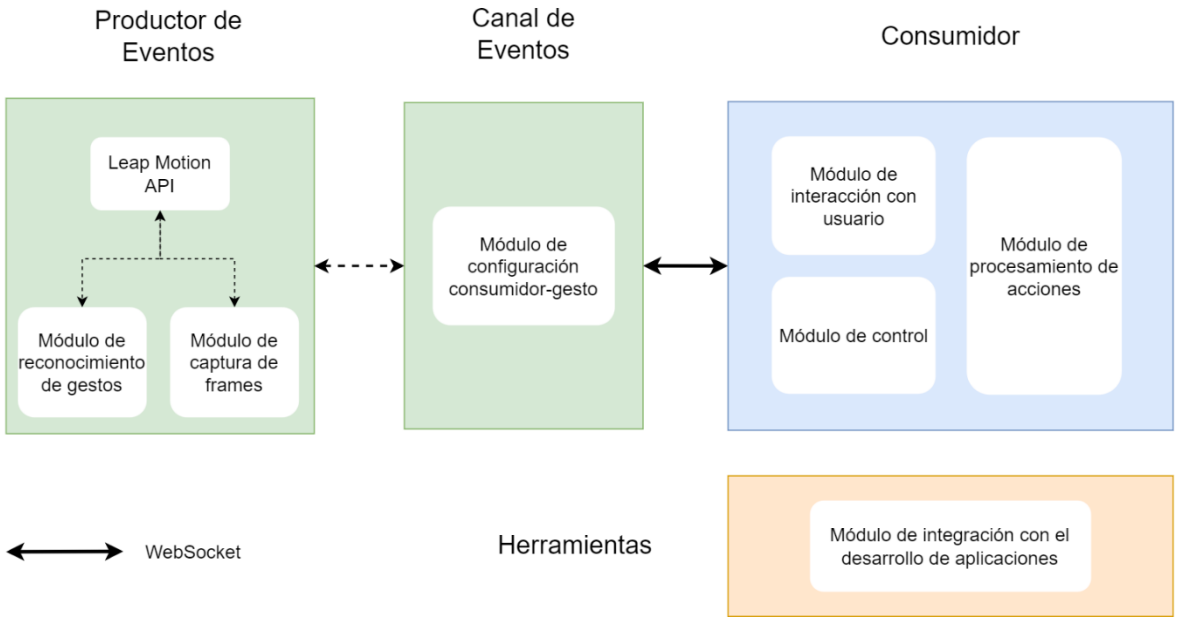


Figura 3.13. Arquitectura de biblioteca actualizada.

3.2.4.1.1 Productor de eventos

El productor de eventos se encarga de trabajar de forma cercana con el dispositivo Leap Motion a través de su interfaz de desarrollo de aplicaciones, de modo que

accede a los datos de seguimiento de las manos, los cuales son comunicados al módulo de reconocimiento de gestos y al canal de eventos.

3.2.4.1.2 Canal de eventos

Funciona como intermediario entre las capas productor de eventos y consumidor, de manera que ambos requieren establecer la comunicación con esta sección. El módulo de configuración consumidor-gesto tiene la responsabilidad de proveer a ambos extremos el medio de comunicación para intercambiar y consumir eventos.

3.2.4.1.3 Consumidor

La capa de consumidor tiene como propósito proveer a los desarrolladores de software las herramientas necesarias para consumir y enviar eventos. El módulo de control encapsula diversas funciones de configuración, relacionadas con los datos del seguimiento de las manos. El módulo de interacción con usuario ofrece una interfaz gráfica que permite ajustar, en tiempo de ejecución, los parámetros de configuración del módulo de control y, por último, el módulo de procesamiento de acciones tiene la responsabilidad de recibir las actualizaciones de eventos y de proporcionar un medio por el cual ejecuta una determinada acción mediante el uso de *callbacks* (código fuente que ejecutado en un tiempo dado).

3.2.4.1.4 Herramientas

La sección de herramientas abarca una serie de utilidades que se incluyen como parte adicional de la biblioteca. El módulo de integración con el desarrollo de aplicaciones provee funciones complementarias para facilitar la integración de la biblioteca en un ambiente de programación Java.

3.2.5 Sprint 3

La tarea asignada para este Sprint fue la siguiente:

- HU_CAP_IMG: Desarrollar módulo de captura de *frames*
- HU_REC_GES: Desarrollar módulo de reconocimiento de gestos.

3.2.5.1 HU_CAP_IMG: Desarrollar módulo de captura de *frames*

Este módulo involucra un conjunto de clases destinadas a la extracción y procesamiento de los datos de seguimiento del dispositivo Leap Motion, la Figura 3.14 muestra las clases involucradas y se describen a continuación.

La clase `TrackingData` encapsula los atributos y métodos necesarios para el procesamiento de los datos del seguimiento de las manos, proporcionados por el dispositivo Leap Motion a través de la clase `Frame`, presentada en la Figura 3.5.

Intervalo de actualización (`update_interval`), se refiere al periodo de tiempo que se deja pasar para extraer y procesar nuevos datos del dispositivo Leap Motion, un mayor valor corresponde a una ejecución más espaciada, lo que podría causar una interacción lenta. Por el contrario, un valor muy pequeño podría causar un mayor uso de recursos computacionales.

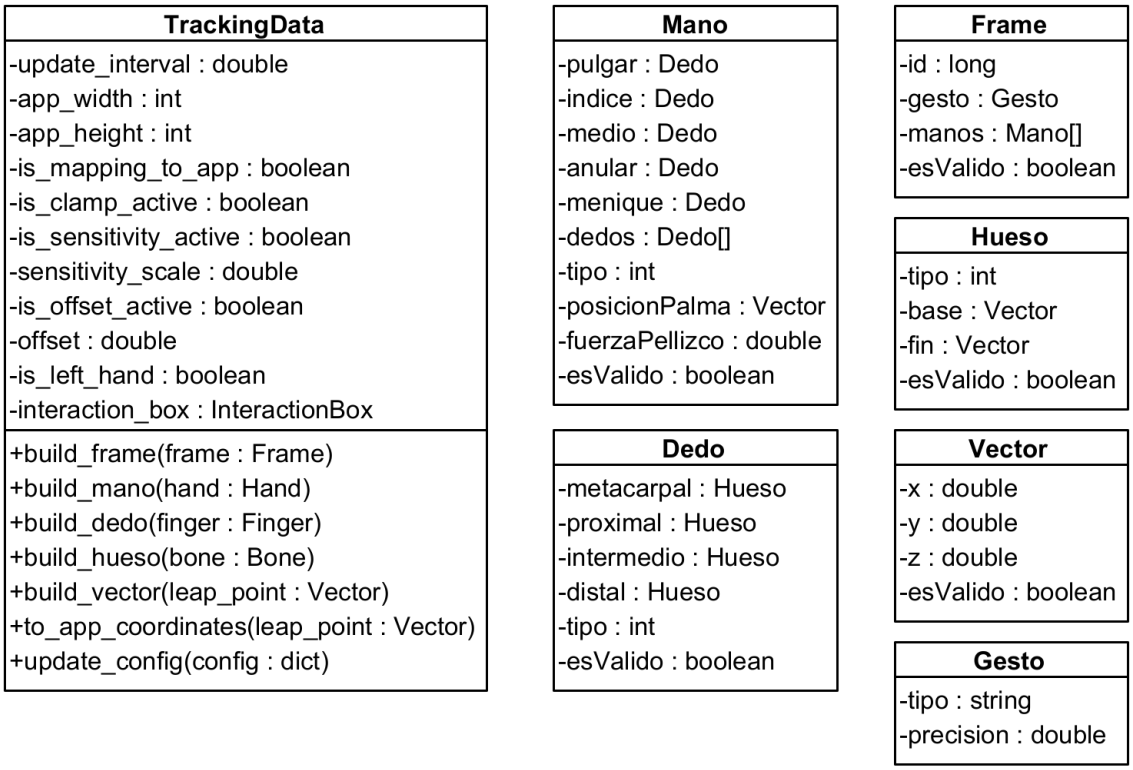


Figura 3.14. Diagrama de clases del módulo de captura de *frames.*

Las medidas de alto y ancho (`app_width, app_height`), son los valores que representan las dimensiones de la aplicación a la que se están mapeando las

coordenadas reportadas por el dispositivo Leap Motion, para activar o desactivar esta función se requiere actualizar el valor del atributo `is_mapping_to_app` a `True` o `False` respectivamente.

La función para restringir el movimiento al área de la aplicación es controlada por el atributo `is_clamp_active`, esto permite limitar las coordenadas de los distintos elementos de las manos a las medidas de alto y ancho, de tal forma que no desaparecen de la pantalla. Esto es útil para ofrecer retroalimentación a los usuarios sobre el área de interacción disponible.

La función de sensibilidad permite controlar el nivel de movimiento requerido para desplazarse de un punto al otro dentro del sistema de coordenadas, esta función es controlada a través de los atributos `is_sensitivity_active`, para activar o desactivar y `sensitivity_scale` para ajustar el nivel de sensibilidad.

La función para compensar el movimiento horizontal está relacionada con el sistema de coordenadas y el área cubierta por las manos. Cuando la opción está desactivada, a través del atributo `is_offset_active` con valor igual a `False`, para llegar al punto más a la derecha con la mano izquierda es necesario llevarla hasta la zona de la mano derecha y de forma análoga, para llegar al punto más a la izquierda con la mano derecha, se debe mover hasta la zona de la mano izquierda. Esta interacción logra causar incomodidad o sentirse poco natural, se representa en la Figura 3.15.

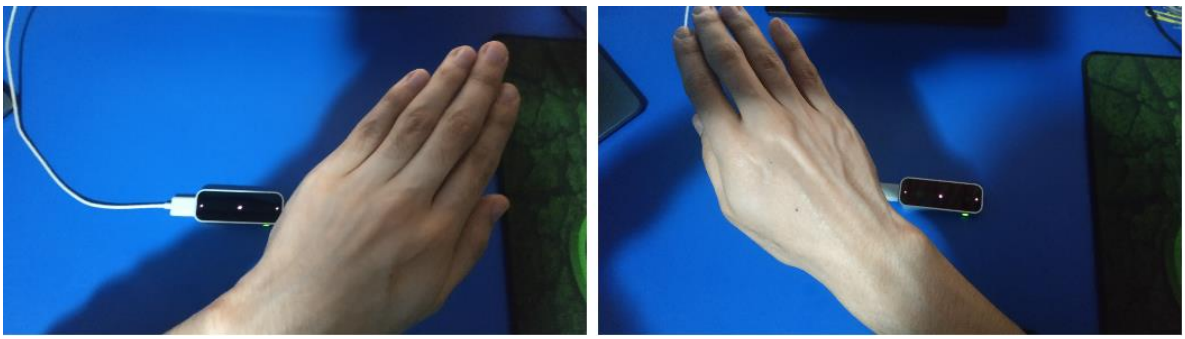


Figura 3.15. Interacción con la opción `offset` desactivada.

Al activar la opción, atributo `is_offset_active` con valor igual a `True`, el área de interacción descrita es recorrida hacia ambos lados, de tal forma que las manos descansen en su posición natural y no invaden el espacio de la otra. La Figura 3.16 representa un ejemplo de esta opción activada.

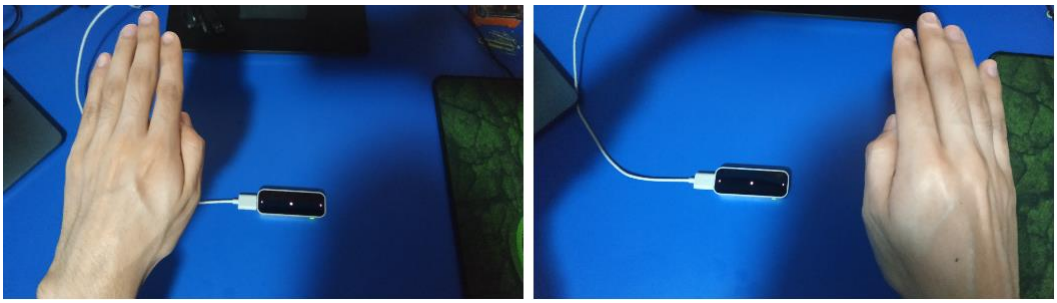


Figura 3.16. Interacción con la opción `offset` activada.

Los métodos que se ofrecen en la clase `TrackingData` están destinados a la creación de objetos de las clases `Mano`, `Dedo`, `Hueso`, `Gesto`, `Vector` y `Frame` a partir de objetos `Leap.Frame` del API del dispositivo Leap Motion.

Se ofrece el método `update_config` para realizar la actualización de las opciones previamente mencionadas. La Figura 3.17 muestra la definición de la clase `TrackingData` y sus atributos.

```
4   class TrackingData:
5       def __init__(self):
6
7           self.update_interval = 0.01
8           self.app_width = 800
9           self.app_height = 600
10          self.is_mapping_to_app = True
11
12          self.is_clamp_active = True
13
14          self.is_sensitivity_active = True
15          self.sensitivity_scale = 1.5
16
17          self.is_offset_active = True
18          self.offset = 0.5
19
20          self.is_left_hand = False
21
22          self.interaction_box = None
```

Figura 3.17. Atributos de la clase `TrackingData`.

La Figura 3.18 presenta la implementación del método `build_frame`, el cual se encarga de extraer los datos de objetos de la clase `Leap.Frame` y procesarlos para construir un nuevo objeto de la clase `Frame` presentada en la Figura 3.14.

```
28  def build_frame(self, frame : Leap.Frame):
29      app_frame = Frame()
30
31      if frame.is_valid:
32
33          self.interaction_box = frame.interaction_box
34
35          hands = frame.hands
36          manos = []
37          for hand in hands:
38              mano = self.build_mano(hand)
39              manos.append(mano)
40
41          app_frame.id = frame.id
42          app_frame.manos = manos
43          app_frame.esValido = True
44
45      else:
46          app_frame.esValido = False
47
48      return app_frame
```

Figura 3.18. Método `build_frame` de la clase `TrackingData`.

3.2.5.2 HU_REC_GES: Desarrollar módulo de reconocimiento de gestos

Continuando con los módulos de la capa productora de eventos, el módulo de reconocimiento de gestos emplea un flujo de trabajo de aprendizaje automático basado en el método propuesto por Avola et al. en [26], el cual sugiere el uso de los ángulos formados las falanges, las posiciones 3D de las puntas de los dedos y la palma de la mano como datos para identificar distintos gestos a través de una red neuronal. La Figura 3.19 muestra ejemplos de las distintas poses de la mano que componen a una serie de gestos.

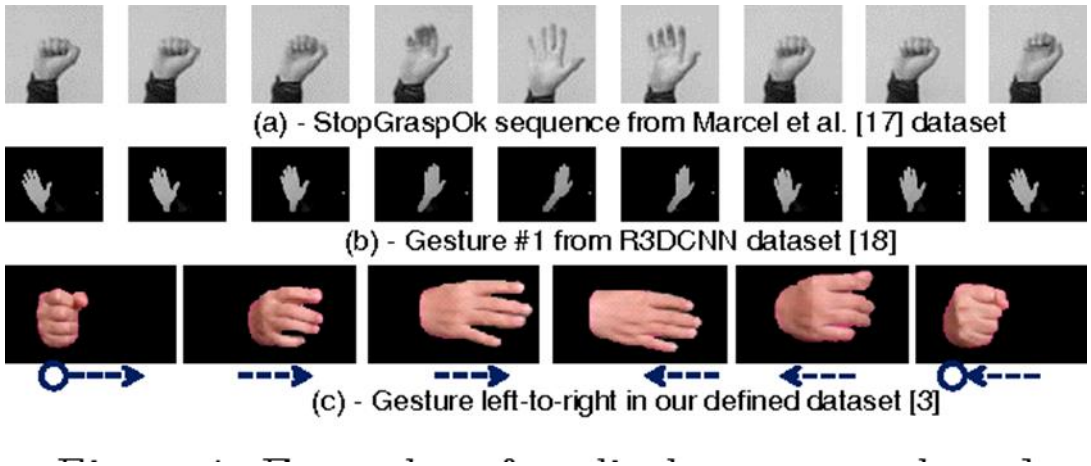


Figura 3.19 Ejemplos de gestos realizados con las manos [27].

La Figura 3.20 muestra la secuencia de pasos del flujo de trabajo para lograr realizar el reconocimiento de gestos.

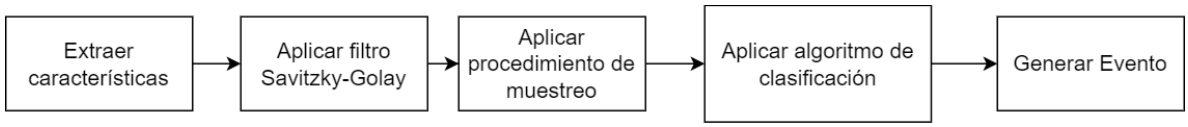


Figura 3.20 Funcionamiento del módulo de reconocimiento de gestos.

Extracción de características

La Figura 3.21 muestra como las falanges de la mano forman ángulos convexos de manera natural, lo cual permite lograr una gran variedad de movimientos que permiten realizar innumerables actividades.

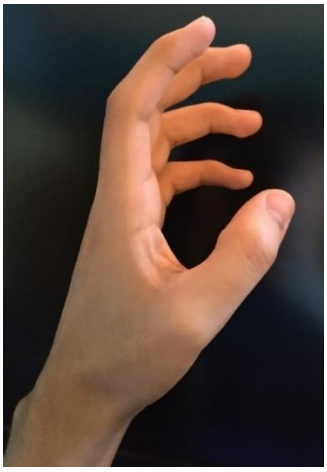


Figura 3.21 Ángulos formados por las falanges.

Los datos de entrada se conforman por los siguientes datos, marcados en la Figura 3.22.

- Naranja (posición 3D de las puntas de los dedos).
- Blanco (posición 3D del centro de la palma).
- Ángulos omega (ω) marcados con azul, están formados por la falange proximal y media, a excepción del pulgar que es formada por el hueso metacarpiano y la falange proximal. Se numeran del cero al cuatro iniciando con el pulgar.
- Ángulos beta (β) marcados con verde, están formados por las falanges media y distal, a excepción del pulgar que está formada por las falanges proximal y distal. Se numeran del cero al cuatro iniciando con el pulgar.
- Ángulos gamma (γ) marcados con amarillo, formados entre dedos consecutivos. Se toman en cuenta los dedos índice, medio, anular y meñique. Se numeran del 1 al 3 iniciando desde el ángulo formado por los dedos índice y medio.

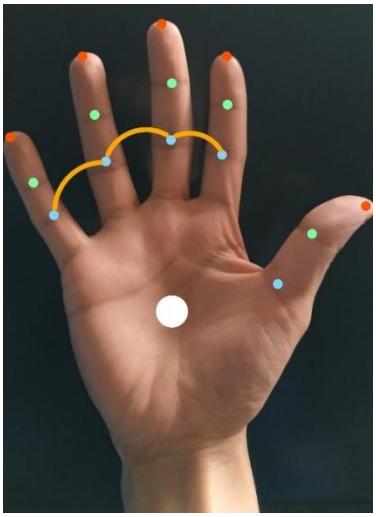


Figura 3.22. Características seleccionadas para describir un gesto.

La Figura 3.23 muestra las clases involucradas en el módulo de reconocimiento de gestos, cada nuevo objeto de la clase `Leap.Frame` recuperado del dispositivo Leap Motion es procesado por el método `process_frame` de la clase `GestureRecognition`.

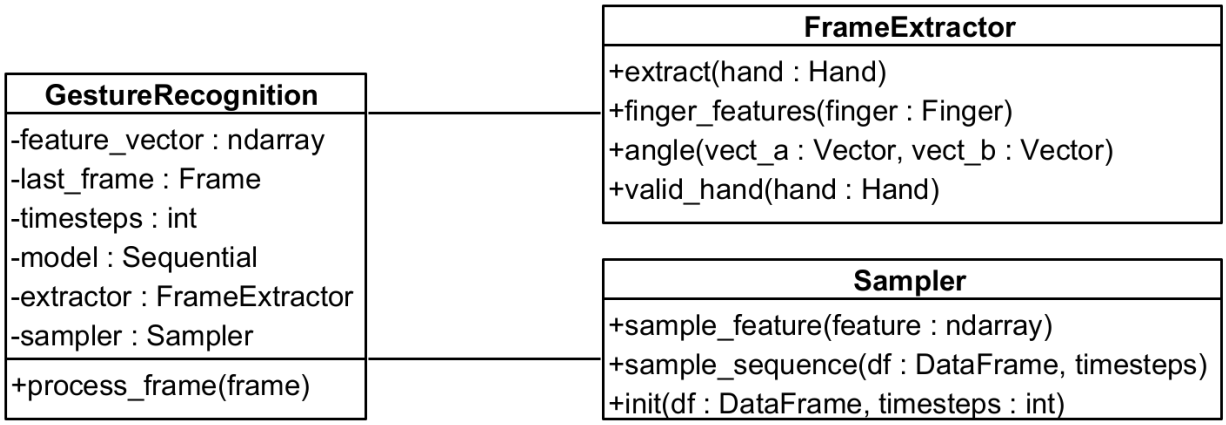


Figura 3.23. Diagrama de clases del módulo de reconocimiento de gestos.

La Figura 3.24 y Figura 3.25 muestran la herramienta para obtener las secuencias de muestras y los distintos archivos generados, respectivamente, para entrenar el modelo de aprendizaje automático.

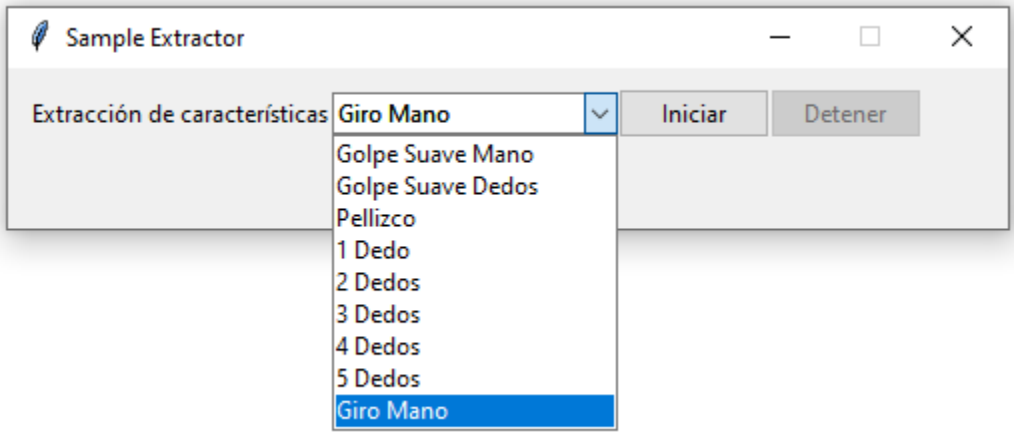


Figura 3.24. Herramienta de extracción de muestras.

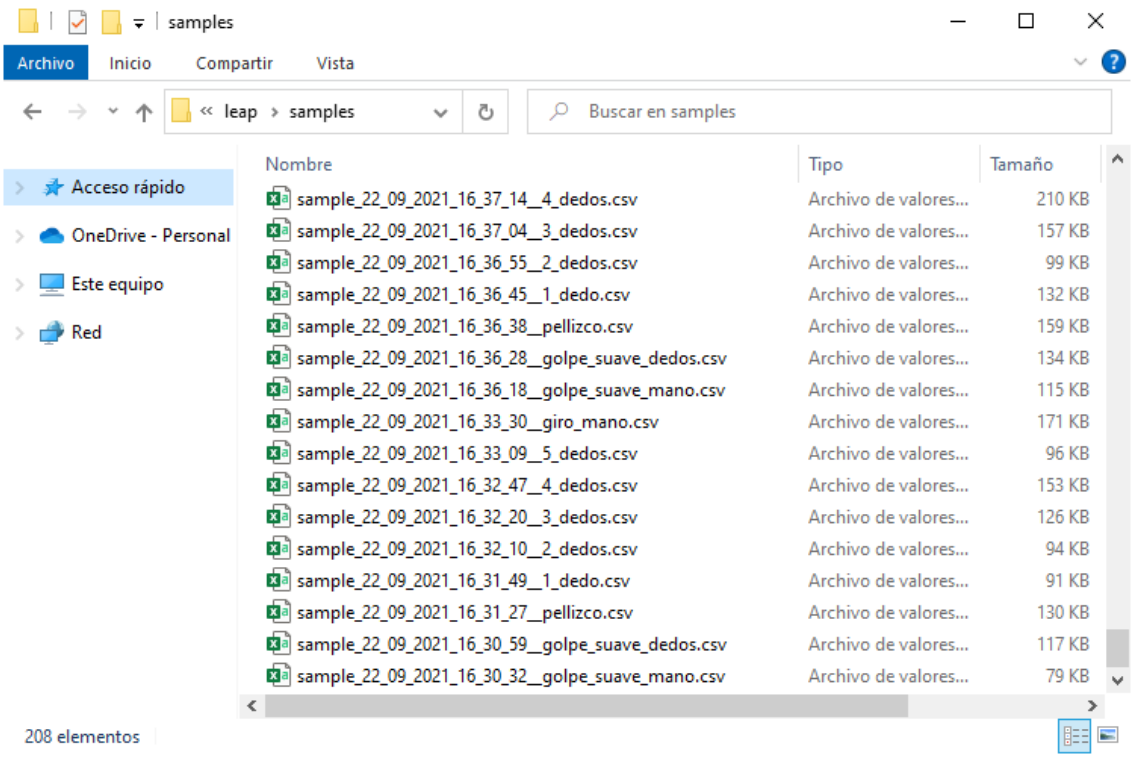
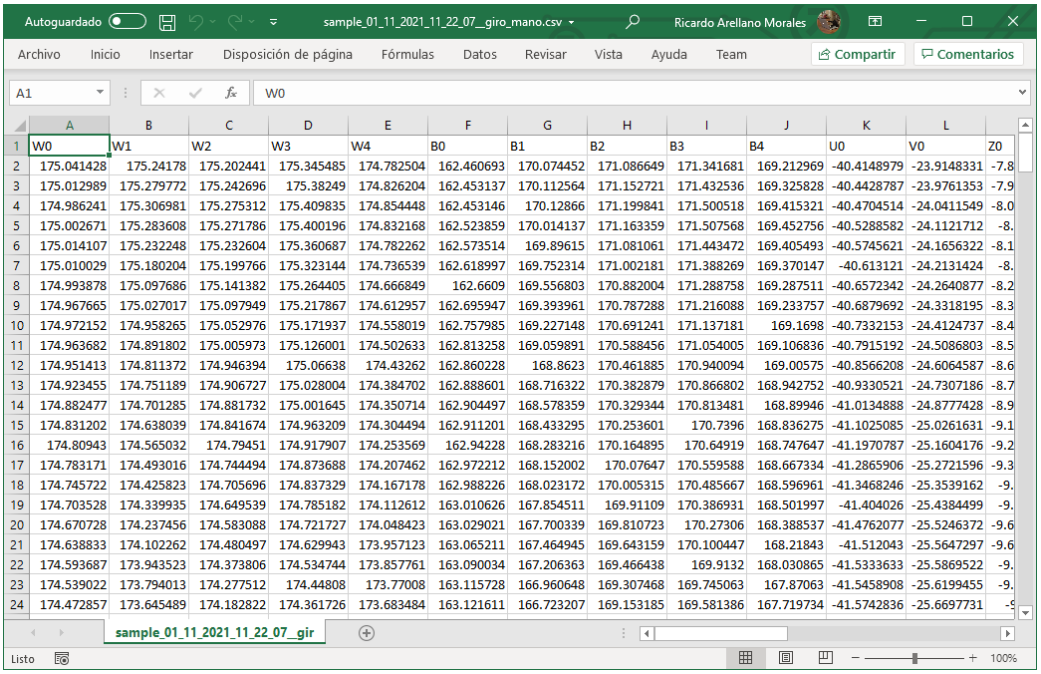


Figura 3.25. Ejemplo de archivos de muestras generados.

Las columnas del archivo mostrado en la Figura 3.26 representan a cada una de las características que describen a un gesto.



	A	B	C	D	E	F	G	H	I	J	K	L	
1	W0	W1	W2	W3	W4	B0	B1	B2	B3	B4	U0	V0	Z0
2	175.041428	175.24178	175.202441	175.345485	174.782504	162.460693	170.074452	171.086649	171.341681	169.212969	-40.4148979	-23.9148331	-7.8
3	175.012989	175.279772	175.242696	175.38249	174.826204	162.453137	170.112564	171.152721	171.432536	169.325828	-40.4428787	-23.9761353	-7.9
4	174.986241	175.306981	175.275312	175.409835	174.854448	162.453146	170.12866	171.199841	171.500518	169.415321	-40.4704514	-24.0411549	-8.0
5	175.002671	175.283608	175.271786	175.400196	174.832168	162.523859	170.014137	171.163359	171.507568	169.452756	-40.5288582	-24.1121712	-8.
6	175.014107	175.232248	175.232604	175.360687	174.782262	162.573514	169.89615	171.081061	171.443472	169.405493	-40.5745621	-24.1656322	-8.1
7	175.010029	175.180204	175.199766	175.323144	174.736539	162.618997	169.752314	171.002181	171.388269	169.370147	-40.613121	-24.2131424	-8.
8	174.993878	175.097686	175.141382	175.264405	174.666849	162.6609	169.556803	170.882004	171.288758	169.287511	-40.6572342	-24.2640877	-8.2
9	174.967665	175.027017	175.097949	175.217867	174.612957	162.695947	169.393961	170.787288	171.216088	169.233757	-40.6879692	-24.3318195	-8.3
10	174.972152	174.958265	175.052976	175.171937	174.558019	162.757985	169.227148	170.691241	171.137181	169.1698	-40.7332153	-24.4124737	-8.4
11	174.963682	174.891802	175.005973	175.126001	174.502633	162.813258	169.059891	170.588456	171.054005	169.106836	-40.7915192	-24.5086803	-8.5
12	174.951413	174.811372	174.946394	175.06638	174.43262	162.860228	168.8623	170.461885	170.940094	169.00575	-40.8566208	-24.6064587	-8.6
13	174.923455	174.751189	174.906727	175.028004	174.384702	162.888601	168.716322	170.382879	170.866802	168.942752	-40.9330521	-24.7307186	-8.7
14	174.882477	174.701285	174.881732	175.001645	174.350714	162.904497	168.578359	170.329344	170.813481	168.89946	-41.0134888	-24.8777428	-8.9
15	174.831202	174.638039	174.841674	174.963209	174.304494	162.911201	168.433295	170.253601	170.7396	168.836275	-41.1025085	-25.0261631	-9.1
16	174.80943	174.565032	174.79451	174.917907	174.253569	162.94228	168.283216	170.164895	170.64919	168.747647	-41.1970787	-25.1604176	-9.2
17	174.783171	174.493016	174.744494	174.873688	174.207462	162.972212	168.152002	170.07647	170.559588	168.667334	-41.2865906	-25.2721596	-9.3
18	174.745722	174.425823	174.705696	174.837329	174.167178	162.988226	168.023172	170.005315	170.485667	168.596961	-41.3468246	-25.3539162	-9.
19	174.703528	174.339935	174.649539	174.785182	174.112612	163.010626	167.854511	169.91109	170.386931	168.501997	-41.404026	-25.4384499	-9.
20	174.670728	174.237456	174.583088	174.721727	174.048423	163.029021	167.700339	169.810723	170.27306	168.388537	-41.4762077	-25.5246372	-9.6
21	174.638833	174.102262	174.480497	174.629943	173.957123	163.065211	167.464945	169.643159	170.100447	168.21843	-41.512043	-25.5647297	-9.6
22	174.593687	173.943523	174.373806	174.534744	173.857761	163.090034	167.206363	169.466438	169.9132	168.030865	-41.5333633	-25.5869522	-9.
23	174.539022	173.794013	174.277512	174.44808	173.77008	163.115728	166.960648	169.307468	169.745063	167.87063	-41.5458908	-25.6199455	-9.
24	174.472857	173.645489	174.182822	174.361726	173.683484	163.121611	166.723207	169.153185	169.581386	167.719734	-41.5742836	-25.6697731	-9

Figura 3.26. Contenido de un archivo de muestra.

La Figura 3.27 muestra la gráfica de los ángulos omega del pulgar al realizar el gesto Giro Mano.

Figura 3.27. Secuencia de ángulos omega recuperados del dedo pulgar en el gesto Giro Mano.

La Figura 3.28 muestra el resultado de aplicar el suavizado realizado a los datos obtenidos en el proceso de extracción de características mediante filtro digital Savitzky-Golay.

La Figura 3.29 muestra los distintos valores seleccionados para formar el nuevo conjunto de valores que será enviado al modelo de aprendizaje automático. Este procedimiento busca obtener un número fijo de valores (en este caso se utiliza el valor de cien) y se realiza con cada una de las características (columnas del archivo de muestra). Como criterio para seleccionar que valores formaran parte de la muestra, se ubican los puntos en los que varían de forma significativa los valores de los ángulos.

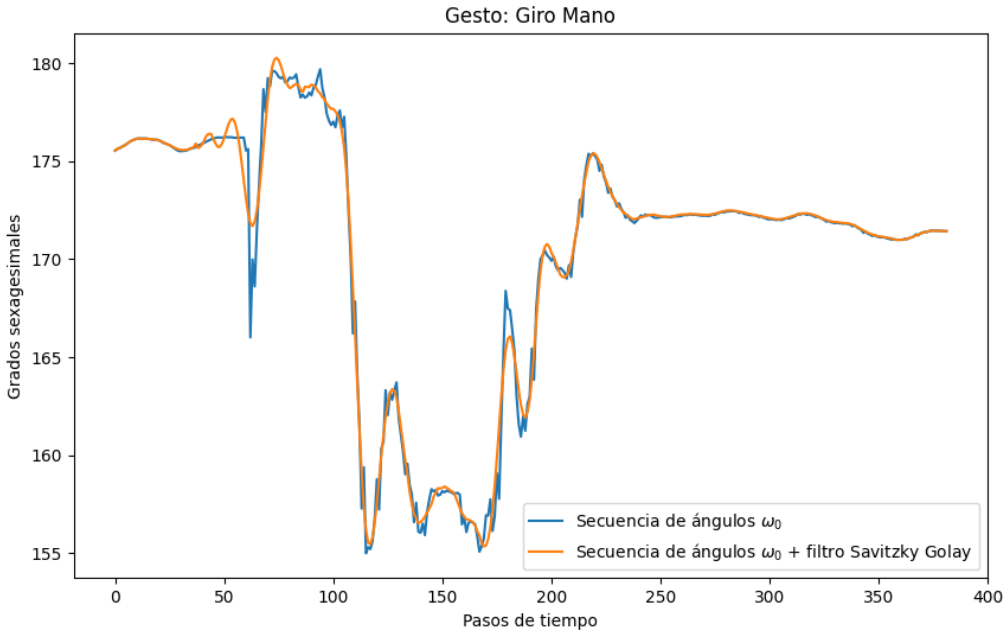


Figura 3.28. Aplicación del filtro digital Savitzky-Golay.

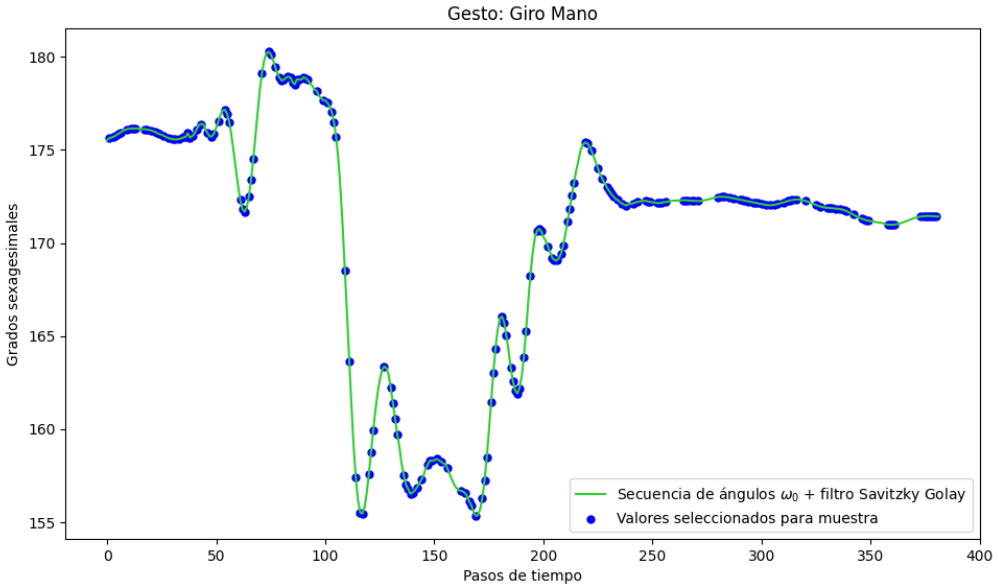


Figura 3.29. Valores seleccionados como parte del proceso de muestreo.

Clasificación

Una manera de trabajar con problemas de tipo secuencial es mediante RNNs (*Recurrent Neural Networks*, Redes Neuronales Recurrentes) y dentro de los modelos existentes, destaca el uso de LSTMs (*Long Short-Term Memory*).

Como último paso del flujo de trabajo, se envían las muestras al modelo de aprendizaje automático [28] implementado con la ayuda de las bibliotecas *Tensorflow* y *Keras*, el modelo resultante recibirá los datos de la muestra como entrada y determinará si existe o no una coincidencia con alguno de los gestos descritos en el *Sprint* 1. La Figura 3.30 muestra el código fuente para definir y entrenar el modelo de Aprendizaje Automático.

```
20  PATH_HOME = pathlib.Path.home() / "leap"
21
22  RESOURCES_PATH = pathlib.Path(__file__).parent / "resources"
23  DEFAULT_MODEL_NAME = "gr_model.h5"
24
25  def eval_model(train_x, train_y, test_x, test_y, time_instants):
26      # 1. Definir red
27      model = Sequential()
28      model.add(LSTM(time_instants, return_sequences=True,
29      input_shape=(time_instants,31)))
30      model.add(LSTM(time_instants, return_sequences=True))
31      model.add(LSTM(time_instants, return_sequences=True))
32      model.add(LSTM(time_instants))
33      model.add(Dense(9, activation='softmax'))
34
35      # 2. Compilar red
36      model.compile(optimizer=SGD(learning_rate=0.0001),
37      loss='categorical_crossentropy', metrics=['accuracy'])
38
39      # 3. Entrenar red
40      csv_logger = CSVLogger(RESOURCES_PATH / "training.log",
41      separator=",", append=False)
42
43      model.fit(train_x, train_y, epochs=1600, validation_split=0.2,
44      batch_size=9, verbose=1, callbacks=[csv_logger])
45
46      model.save(RESOURCES_PATH / DEFAULT_MODEL_NAME)
```

Figura 3.30. Función para entrenar modelo de clasificación.

La Figura 3.31 presenta la función `load_file` y `load_dataset` empleadas para cargar los archivos que contienen las muestras y que utilizaran para ser parte de los datos de entrenamiento del modelo.

```
49  # Recuperar datos para entrenamiento/predicción
50  def load_file(filepath):
51      dataframe = pd.read_csv(filepath)
52      return dataframe.values
53
54  def load_dataset(data_dir, time_instants):
55      files = []
56      classes = []
57      loaded_files = list()
58      sampler = Sampler()
59
60      for (_, _, filenames) in walk(data_dir):
61          files.extend(filenames)
62          random.shuffle(files)
63          break
64
65      for file_name in files:
66          print('Processing file ->', file_name)
67          class_gesture = file_name.split('__')[1]
68          class_key = class_gesture.replace('.csv', '')
69          class_id = LABEL_2_GESTURE_ID[class_key]
70          classes.append(class_id)
71          file_data = pd.read_csv((data_dir / file_name))
72          sample = sampler.init(file_data, time_instants)
73          loaded_files.append(sample)
74
75      x = np.stack(loaded_files)
76
77      onehot_encoder = OneHotEncoder(sparse=False)
78      int_encoded = np.array(classes)
79      int_encoded = int_encoded.reshape(len(int_encoded), 1)
80      y = onehot_encoder.fit_transform(int_encoded)
81
82      return x, y
```

Figura 3.31. Funciones para cargar los archivos de muestras para entramiento del modelo.

La Figura 3.32 presenta la definición e implementación del método `extract` de la clase `FrameExtractor`, este método recibe como parámetro un objeto de la clase `Leap.Hand` que se valida a través del método `valid_hand` (Figura 3.34), para continuar con la ejecución del método. Las características presentadas en la Figura 3.22 se extraen a través del método `finger_features` que se aplica a cada uno de los objetos `Leap.Finger` obtenidos del objeto mano. Los resultados de la operación anterior se recuperan y almacenan en un `array`.

El método `finger_features` presentado en la Figura 3.33 se encarga de procesar los objetos `Leap.Finger`, de tal manera que recupera las posiciones en tres dimensiones de las falanges (proximal, intermedio y distal) para obtener una serie de segmentos requeridos para obtener los ángulos omega y beta, a través del método `angle` presentado en la Figura 3.34.

La Figura 3.35 presenta la implementación del método `process_frame` de la clase `GestureRecognition`, este método recibe como parámetro un objeto `Leap.Frame` y se extraen los datos de la mano derecha, se realiza una valoración para determinar si existe movimiento en la mano (o del dedo índice), que indique la posibilidad de que el usuario esté realizando un gesto. Los valores del movimiento de la mano y velocidad del dedo índice se comparan con un valor de umbral previamente establecido.

```
13  def extract(self, hand : Hand):
14      if not self.valid_hand(hand):
15          return
16
17      fingers = list(
18      map(lambda f: hand.fingers.finger_type(f)[0], FINGER_LIST))
19      prox_bones = list(
20      map(lambda f: f.bone(Bone.TYPE_PROXIMAL), fingers))
21
22      gamma_values = []
23      for i in range(1, 4):
24          gamma_values.append(
25          self.angle(prox_bones[i].next_joint - hand.palm_position,
26          prox_bones[i+1].next_joint - hand.palm_position))
27
28      omega_vals = np.zeros(5)
29      beta_vals = np.zeros(5)
30      x_vals = np.zeros(6)
31      y_vals = np.zeros(6)
32      z_vals = np.zeros(6)
33
34      features = map(lambda f: self.finger_features(f), fingers)
35
36      for idx, feat in enumerate(features):
37          omega_vals[idx] = feat["omega"]
38          beta_vals[idx] = feat["beta"]
39          x_vals[idx] = feat["x"]
40          y_vals[idx] = feat["y"]
41          z_vals[idx] = feat["z"]
42
43      x_vals[5] = hand.palm_position.x
44      y_vals[5] = hand.palm_position.y
45      z_vals[5] = hand.palm_position.z
46
47      row_feature = np.concatenate([omega_vals, beta_vals, x_vals, y_vals,
48      z_vals, gamma_values])
49
50      return row_feature
```

Figura 3.32. Método extract de la clase FrameExtractor.

```
53  def finger_features(self, finger : Finger):
54      distal : Bone = finger.bone(Bone.TYPE_DISTAL)
55      proximal : Bone = finger.bone(Bone.TYPE_PROXIMAL)
56      inter : Bone = finger.bone(Bone.TYPE_INTERMEDIATE)
57
58      ba = proximal.prev_joint - proximal.next_joint
59      bc = inter.next_joint - inter.prev_joint
60
61      cb  = inter.prev_joint - inter.next_joint
62      cd = distal.next_joint - distal.prev_joint
63
64      omega_angle = self.angle(cb, cd)
65      beta_angle = self.angle(ba, bc)
66
67      features = {
68          "omega": omega_angle,
69          "beta": beta_angle,
70          "x": finger.tip_position.x,
71          "y": finger.tip_position.y,
72          "z": finger.tip_position.z
73      }
74
75      return features
```

Figura 3.33. Método finger_features de la clase FrameExtractor.

```
77  def angle(self, vect_a, vect_b):
78      return math.acos(
79          vect_a.dot(vect_b)/(vect_a.magnitude*vect_b.magnitude))*DEGREES_IN_RADIAN
80
81  def valid_hand(self, hand : Hand):
82      if hand.is_valid and hand.is_right:
83          for finger in hand.fingers:
84              if not finger.is_valid:
85                  return False
86              return True
87      return False
```

Figura 3.34. Métodos angle y valid_hand de la clase FrameExtractor.

```
24  def process_frame(self, frame : Frame):
25      hand = frame.hands.rightmost
26      if not hand.is_valid:
27          return None
28
29      umbral = math.sqrt(
30      math.pow(hand.rotation_angle(self.last_frame, Vector.x_axis),2) +
31      math.pow(hand.rotation_angle(self.last_frame, Vector.y_axis),2) +
32      math.pow(hand.rotation_angle(self.last_frame, Vector.z_axis),2))
33
34      index_finger_list = hand.fingers.finger_type(Finger.TYPE_INDEX)
35      index_finger = index_finger_list[0]
36      vector_velocity = index_finger.tip_velocity
37      tip_velocity = 0
38      message = None
39
40      if vector_velocity:
41          tip_velocity = vector_velocity.magnitude
42
43      if umbral > DEFAULT_THRESHOLD or tip_velocity > DEFAULT_FINGER_THRESHOLD:
44          feature_row = None
45          try:
46              feature_row = self.extractor.extract(hand)
47          except:
48              pass
49          if feature_row is not None:
50              self.feature_vector.append(feature_row)
51      else:
52          if len(self.feature_vector) > 100:
53              vector_sequence = np.stack(self.feature_vector)
54              dataframe = pd.DataFrame(vector_sequence, columns=DEFAULT_HEADER)
55
56              sample = self.sampler.init(dataframe, self.timesteps)
57              x = np.stack([sample])
58
59              prediction = self.model.predict(x)
60              max_value = np.amax(prediction, axis=-1)
61
62              if max_value[0] > 0.5:
63                  id_predicted_class = np.argmax(prediction, axis=-1)
64                  predicted_class = LABEL_GESTURE[id_predicted_class[0]]
65
66                  print(prediction, id_predicted_class, predicted_class)
67                  message = {'tipo':predicted_class, 'precision': str(max_value[0])}
68          self.feature_vector = []
69      return message
```

Figura 3.35. Método `process_frame` de la clase `GestureRecognition`.

La implementación de la clase `Sampler` se muestra en las Figura 3.36, Figura 3.38 y Figura 3.37. El método `sample_feature` se encarga de obtener los valores

máximo y mínimos relativos, que representan los puntos en los que existe un cambio significativo en los valores de la secuencia.

```
 7  class Sampler:
 8
 9      def sample_feature(self, feature: np.ndarray) -> np.ndarray:
10          maxima, _ = find_peaks(feature)
11          minima, _ = find_peaks(feature * -1)
12          return np.concatenate((minima, maxima))
```

Figura 3.36. Método `sample_feature` de la clase `Sampler`.

El método `init` (Figura 3.37) recibe como parámetro un objeto `DataFrame` de la biblioteca Pandas, el cual representa una secuencia de características obtenidas (con ayuda de la clase `FrameExtractor`) a lo largo de un periodo de tiempo, de tal forma que en esta secuencia se busca la coincidencia con uno de los gestos propuestos.

La Figura 3.38 presenta el método `sample_sequence`, que se ocupa de procesar cada una de las características del objeto `DataFrame`, es aquí donde se aplica el filtro digital Savitzky-Golay para eliminar el ruido causado por el temblor de las manos y es donde se emplea el método `sample_feature` presentado en la Figura 3.36.

```
54  def init(self, dataframe : pd.DataFrame, timesteps = 200):
55      sample_time_instants = self.sample_sequence(dataframe, timesteps)
56      sample_values = dataframe.iloc[sample_time_instants]
57      values = sample_values.fillna(0).values
58
59      if sample_values.shape[0] < timesteps:
60          size_aux = timesteps - sample_values.shape[0]
61          aux = np.zeros((size_aux, DEFAULT_NUMBER_FEATURES))
62          values = np.concatenate((values, aux))
63
64      return values
```

Figura 3.37. Método `init` de la clase `Sampler`.

```
16  def sample_sequence(self, df : pd.DataFrame, timesteps):
17      df_total_instants = df.shape[0]
18      time_instants = {}
19      total_time_instants = 0
20      for col in SAMPLE_FEATURES:
21          feature = savgol_filter(df[col], 51, 11)
22          feat_time_instants = self.sample_feature(feature)
23          time_instants[col] = feat_time_instants
24          total_time_instants += feat_time_instants.size
25
26      theta_set = list(time_instants.values())
27      sample = np.unique(np.concatenate(theta_set))
28
29      if sample.size > timesteps:
30          prop_ti = {}
31          with localcontext() as ctx:
32              ctx.rounding = ROUND_HALF_UP
33              for key, feature in time_instants.items():
34                  prop_sample = (Decimal(feature.size*timesteps)/
35                  Decimal(total_time_instants))
36
37                  prop_size = int(prop_sample.to_integral_value())
38                  prop_ti[key] = np.random.choice(feature, prop_size)
39
40          values = list(prop_ti.values())
41          values_cardinality = np.concatenate((values))
42          sample = np.unique(values_cardinality)
43
44          if sample.size < timesteps:
45              while True:
46                  remain = timesteps - sample.size
47                  rnd = np.random.choice(df_total_instants, remain)
48                  sample = np.unique(np.concatenate((sample, rnd)))
49                  if sample.size == timesteps:
50                      break
51
52      return sample
```

Figura 3.38. Método `sample_sequence` de la clase `Sampler`.

3.2.6 Sprint 4

La tarea planeada para este Sprint fue la siguiente:

- HU_CONF_CG: Desarrollar módulo de configuración consumidor-gesto.

3.2.6.1 HU_CONF_CG: Desarrollar módulo de configuración consumidor-gesto.

La capacidad para detectar cuando se realiza un gesto y distinguir de cual se trata, además de enviar constantemente los datos de seguimiento de las manos supone la creación de un flujo constante de información destinada a comunicarla a la sección de consumidor, el módulo de configuración consumidor-gesto se posiciona como el encargado de “escuchar” y notificar cuando nuevos eventos son detectados, de modo que los mensajes sean consumidos por él (los) consumidore(s). La Figura 3.39 presente un esquema lógico de la interconexión de los módulos de la sección productora de eventos y el módulo de configuración consumidor-gesto.

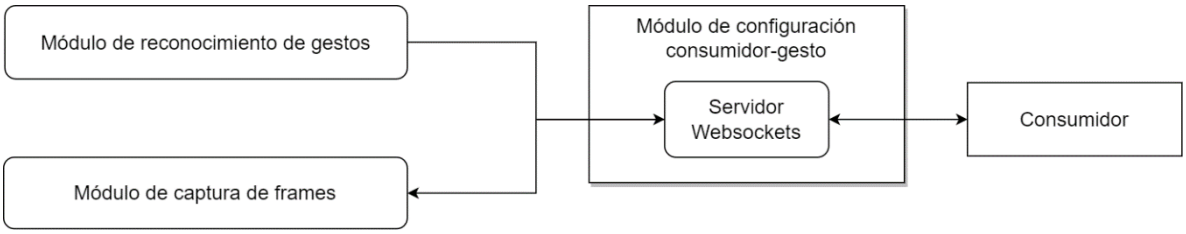


Figura 3.39 Conexiones realizadas al módulo de configuración consumidor-gesto.

El desarrollo del módulo se realizó con ayuda de la biblioteca websockets escrita en Python y que es implementada a través de la biblioteca asyncio incluida en el lenguaje y ampliamente utilizada para escribir código concurrente. La Figura 3.40, Figura 3.41, Figura 3.42 y Figura 3.43 muestran la integración de los módulos de reconocimiento de gestos y de captura de *frames* con el servidor de WebSockets.

```
77  async def main():
78      async with websockets.serve(handler, "localhost", 8765):
79          await asyncio.Future()
80
81  if __name__ == '__main__':
82      asyncio.run(main())
83
```

Figura 3.40. Código requerido para iniciar el servidor de WebSocket.

```
62  async def handler(websocket, path):
63      consumer_task = asyncio.create_task(consumer_handler(websocket,path))
64      producer_task = asyncio.create_task(producer_handler(websocket,path))
65      done, pending = await asyncio.wait(
66          [consumer_task, producer_task],
67          return_when=asyncio.FIRST_COMPLETED,
68      )
69      for task in pending:
70          task.cancel()
```

Figura 3.41. Manejadores de consumidor y productor.

```
49  async def consumer(message):
50      config = jsonpickle.decode(message)
51      tracking_data.update_config(config)
52      return
53
54
55  async def consumer_handler(websocket, path):
56      while True:
57          message = await websocket.recv()
58          print(message)
59          await consumer(message)
```

Figura 3.42. Manejador de la sección consumidor.

3.2.7 Sprint 5

La tarea planeada para este Sprint fue la siguiente:

- HU_PRO_GA: Desarrollar módulo de procesamiento de acciones.

3.2.7.1 HU_PRO_GA: Desarrollar módulo de procesamiento de acciones

El módulo de procesamiento de acciones se sitúa dentro de la sección de consumidor de la arquitectura, este módulo provee las funciones necesarias para realizar la conexión con el canal de eventos, puntualmente, con el módulo de configuración consumidor-gesto que se describió en el Sprint anterior.

```
15  controller = Leap.Controller()
16  tracking_data = TrackingData()
17  gesture_recognition = GestureRecognition()
18  stack = deque()
19
20  async def producer_handler(websocket, path):
21
22      lastProcessedFrameID = 0
23      gesture_recognition.last_frame = controller.frame()
24
25      while True:
26          currentID  = controller.frame().id
27
28          for history in range(0, currentID - lastProcessedFrameID):
29              frame = controller.frame(history)
30
31              if frame.is_valid:
32                  stack.append(frame)
33
34          lastProcessedFrameID = currentID
35
36          while len(stack) > 0:
37              frame = stack.pop()
38              app_frame = tracking_data.build_frame(frame)
39
40              message_gesture = gesture_recognition.process_frame(frame)
41              if message_gesture is not None:
42                  app_frame.gesto = message_gesture
43
44              data = jsonpickle.encode(app_frame, unpicklable=False)
45              await websocket.send(data)
46
47          await asyncio.sleep(tracking_data.update_interval)
```

Figura 3.43. Manejador de la sección productor.

Este módulo está escrito completamente en el lenguaje Java y abarca una serie de clases y una interfaz para llevar a cabo las responsabilidades que se le asignaron, la Figura 3.44 muestra las clases que integran este módulo.

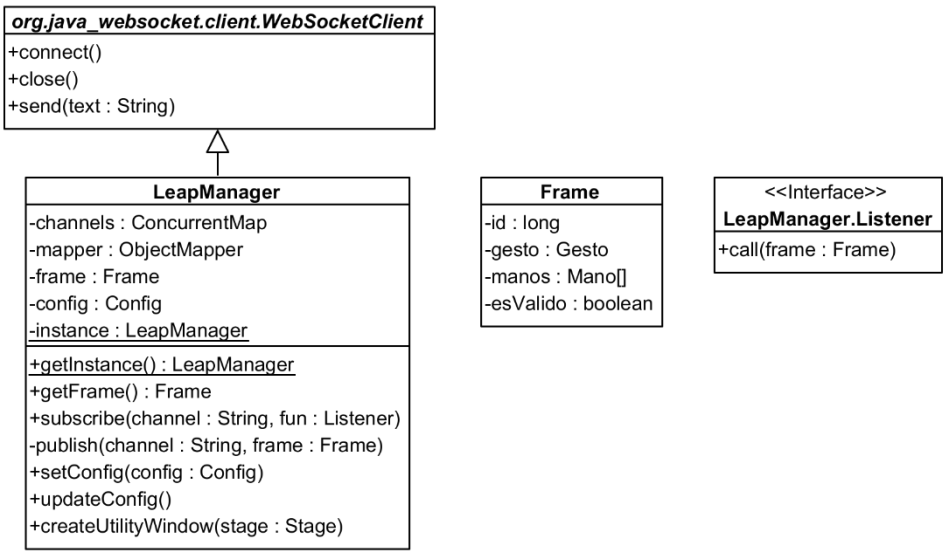


Figura 3.44. Diagrama de clases del módulo de procesamiento de acciones.

La clase `LeapManager` ofrece una serie de métodos para administrar la comunicación con el canal de eventos, es el caso de los métodos `connect` y `close`. El método subscribe encapsula un mecanismo mediante el cual se realiza la asociación entre un valor `String`, el cual identifica a un gesto y una implementación de la interfaz `LeapManager.Listener`, de manera que la ejecución de este método resulta en la generación de un mapa llave-valor.

La interfaz `LeapManager.Listener` funciona como un nuevo tipo de dato y define un método llamado `call`, el cual recibe como parámetro una instancia de la clase `Frame`, Es a través del método `call` que se ofrece una opción para interactuar y hacer uso de la información proveniente del canal de eventos. Una opción más es a través del método `getFrame` que devuelve el objeto `Frame` más reciente.

3.2.8 Sprint 6

Las tareas planeadas para este Sprint fueron las siguientes:

- HU_CONTROL: Desarrollar módulo de control.
- HU_INT_USU: Desarrollar módulo de interacción con usuario.

3.2.8.1 HU_CONTROL: Desarrollar módulo de control

El módulo de control está conformado por las clases `Config` y `ConfigBuilder`, presentadas en la Figura 3.45. La clase `Config` encapsula la configuración utilizada por el módulo de captura de *frames* y se actualiza a través del método `updateConfig` de la clase `LeapManager`, siempre y cuando haya sido agregada a través del método `setConfig`. Se ofrece la clase `ConfigBuilder` para construir de manera rápida objetos `Config`, se muestra un ejemplo del uso de ambas clases en la Figura 3.46.

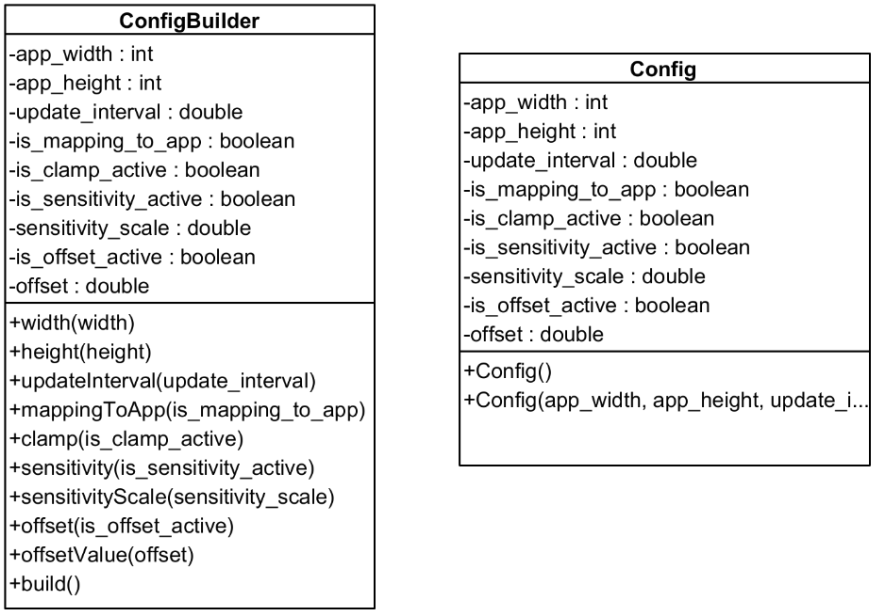


Figura 3.45. Clases del módulo de control.

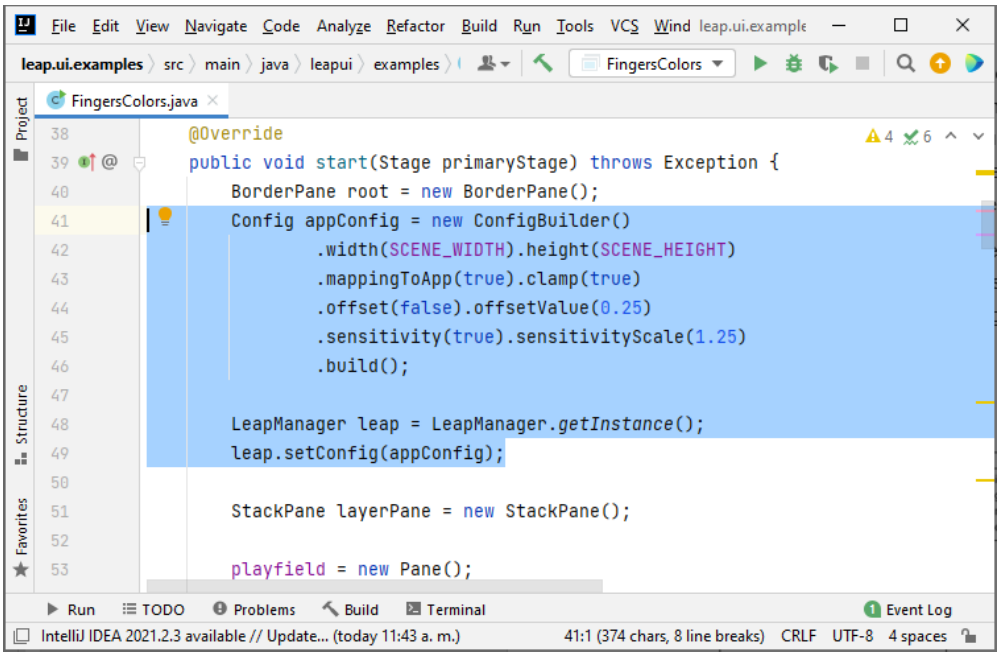


Figura 3.46. Uso de las clases `ConfigBuilder` y `Config`.

3.2.8.2 HU_INT_USU: Desarrollar módulo de interacción con usuario

Se ofrece una ventana de utilidad (Figura 3.47), con la que se controlan las configuraciones del módulo de captura de *frames*. De modo que será el desarrollador de software que utilice la biblioteca el encargado de coordinar si requiere o no usar esta ventana para ajustar los valores de configuración.

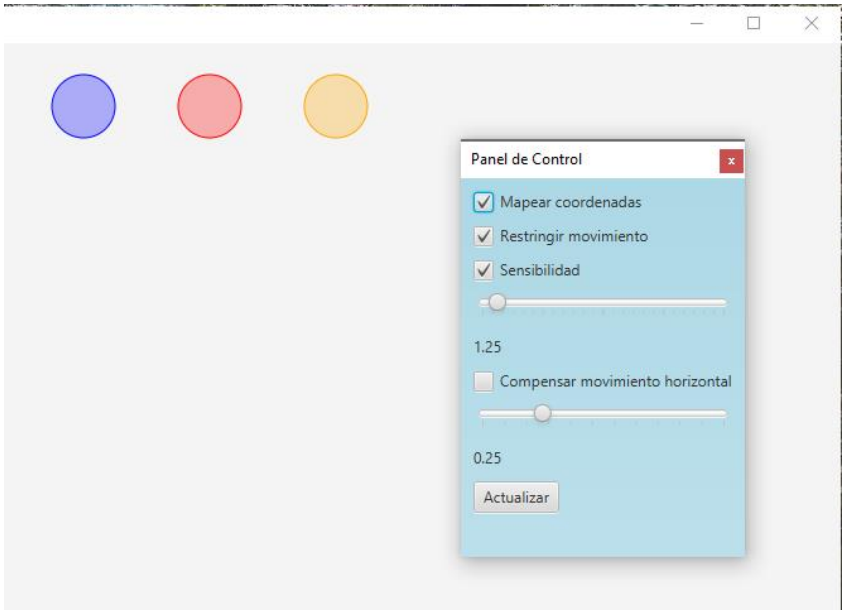


Figura 3.47. Ventana del módulo de interacción con usuario.

3.2.9 Sprint 7

La tarea planeada para este Sprint fue la siguiente:

- HU_INT_DES: Desarrollar módulo de integración con el desarrollo de aplicaciones.

El objetivo de la biblioteca es que sea factible integrarse en un ambiente de programación Java de forma fácil y rápida, para lograrlo se propone exportar las clases presentadas en los módulos de procesamiento de acciones, control e interacción con usuario a un archivo JAR (Java Archive). Este archivo contiene los archivos de las clases, así como las dependencias utilizadas.

Para generar el archivo JAR se utilizó la herramienta de construcción de proyectos Maven. En la Figura 3.48 se muestra el uso de la fase `install` del ciclo de vida estándar de Maven, la cual es descrita por la documentación oficial de Maven como: “instalar el paquete en el repositorio local, para utilizarlo como dependencia en otros proyectos a nivel local”.

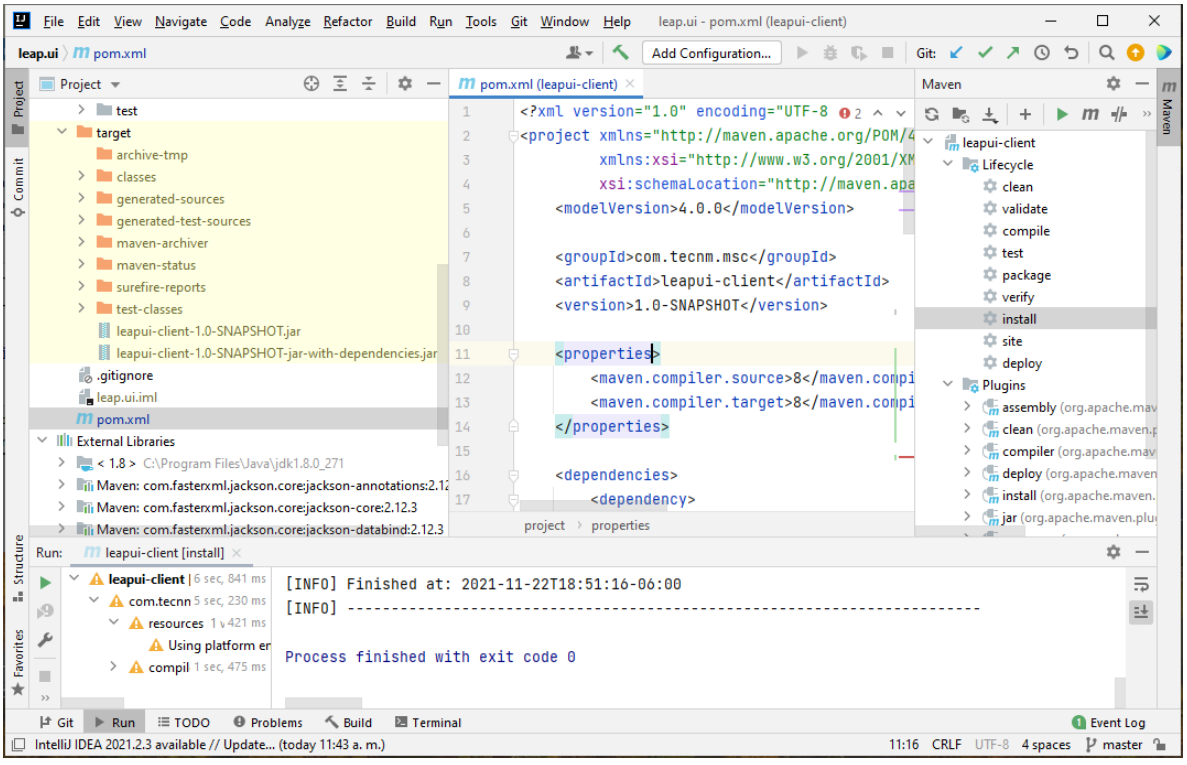


Figura 3.48. Construcción de archivo JAR a través de Maven.

El archivo JAR generado se importa en proyectos Java, como por ejemplo, en la Figura 3.49 se muestra un ejemplo de un proyecto que utiliza las clases del paquete `mx.tecnm.ito.depi.msc.leapui` a través de instrucciones `import`.

El nombre LeapUI se asigna a la biblioteca desarrollada, para identificarla de forma práctica.

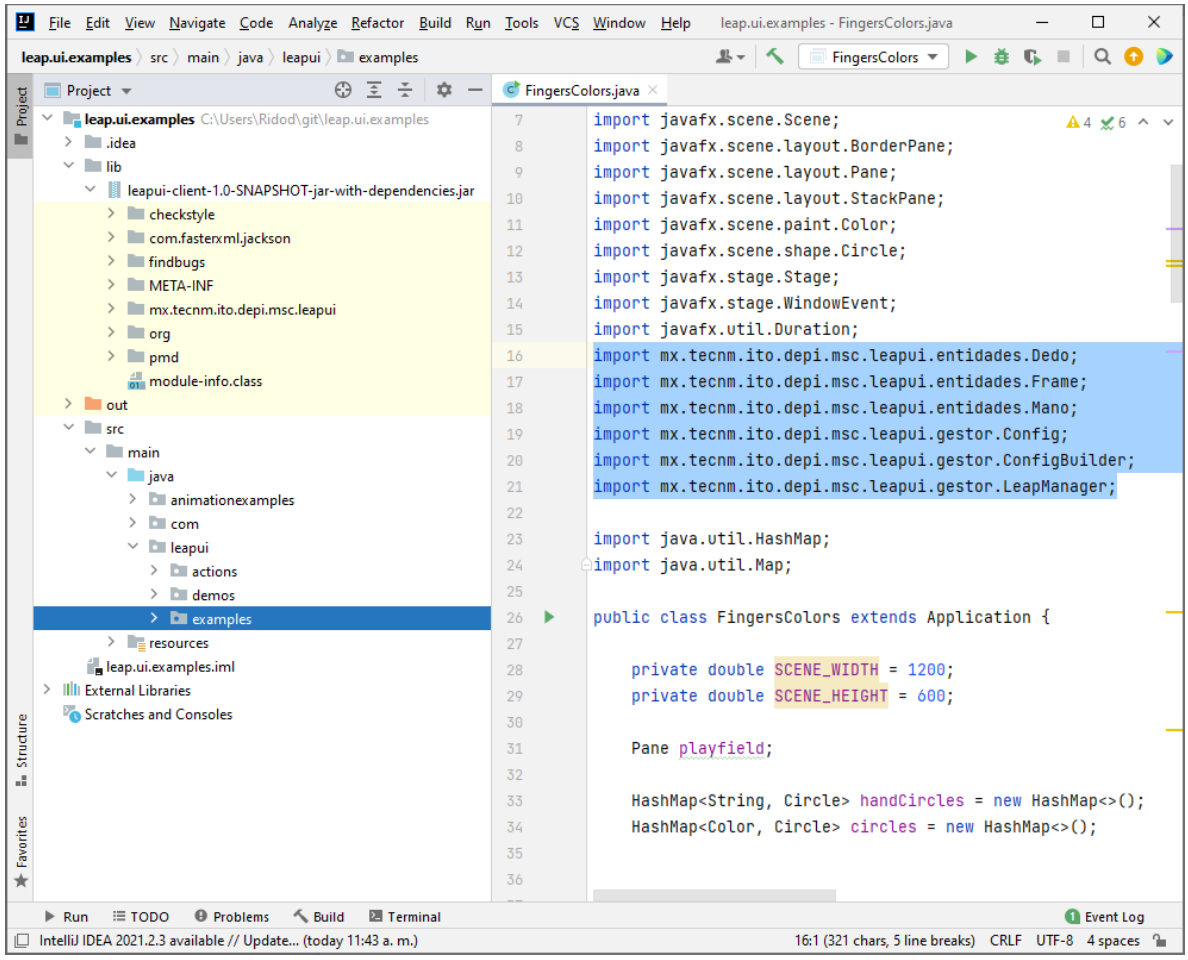


Figura 3.49. Ejemplo del uso de la biblioteca en un ambiente de desarrollo Java.

Capítulo 4. Resultados

En este capítulo se muestran los resultados de la implementación de la biblioteca, partiendo por el caso de estudio.

4.1 Caso de estudio

Esta sección describe la aplicación del caso de estudio, partiendo por su planteamiento, seguido del desarrollo, implementación y los resultados obtenidos.

4.1.1 Planteamiento del caso de estudio

El Centro de Entrenamiento y Educación Especial es una entidad del Instituto de Psicología y Educación de la Universidad Veracruzana (CEEEORI), ubicado en la ciudad de Orizaba, Veracruz. Dentro de sus actividades se encuentra la identificación y solución de problemas que la población infantil de la región de Orizaba presenta, a través de la aplicación del conocimiento generado de la investigación.

El programa de modificación de conducta es uno de los cuatro programas que componen al CEEEORI y busca servir a la población estudiantil aplicando principios conductuales que guíen su desarrollo psicoeducativo.

Se contactó a la Mtra. Guadalupe Gaytán Romero, coordinadora del centro, para plantear el desarrollo de una aplicación de software que utilice las capacidades del dispositivo Leap Motion a través de la biblioteca de software expuesta en este trabajo.

4.1.2 Descripción

Se desarrolló una aplicación de software titulada: “MFRP: atención, motricidad fina, percepción visual, noción espacial y resolución de problemas para niños a través del dispositivo LeapMotion”, controlada por el movimiento de las manos, a través de dispositivo Leap Motion. La aplicación se compuso de seis ejercicios enfocados en ejercitar las capacidades de atención, motricidad fina, percepción visual, noción espacial y resolución de problemas. Se realizaron pruebas de validación y retroalimentación.

4.1.3 Objetivo general

El objetivo general es integrar la aplicación “MFRP: atención, motricidad fina, percepción visual, noción espacial y resolución de problemas para niños a través del dispositivo LeapMotion” en el programa de nociones básicas, subprograma del programa de modificación de conducta del CEEEORI, para servir a los estudiantes en los aspectos ya mencionados.

4.1.4 Población atendida

La población atendida está conformada por niños con un rango de edad de 5 a 7 años.

4.1.5 Metodología

La metodología empleada en el caso de estudio se compone de las siguientes secciones.

1. Análisis y diseño de prototipo de aplicación
2. Desarrollo de aplicación
3. Pruebas con la población estudiantil
4. Rediseño y adecuaciones
5. Aplicación de cuestionario de retroalimentación

4.1.6 Desarrollo del caso de estudio

Se realizó el análisis para determinar los ejercicios que podrían aplicarse como actividad en el programa de nociones básicas del CEEEORI, tomando en cuenta algunos aspectos que debería incluirse como parte de la interacción, entre los que se encuentran:

- Atención
- Motricidad fina
- Percepción visual
- Noción espacial
- Resolución de problemas

La interacción se dio exclusivamente a través del movimiento de las manos, no se permitió utilizar el puntero o el teclado para completar las actividades. El dispositivo Leap Motion, apoyado de la biblioteca de software, fue utilizado para completas los ejercicios.

El desarrollo de la aplicación se llevó a cabo con la tecnología para interfaces gráficas JavaFX del lenguaje de programación Java y los datos del movimiento de las manos fueron provistos por la arquitectura presentada en el Capítulo 3. A continuación, se describe el desarrollo de los ejercicios diseñados para ser aplicados como actividades.

Ejercicio 1

El ejercicio uno tiene como propósito presentar el funcionamiento del dispositivo Leap Motion, al iniciar se muestra una figura esférica de un tamaño establecido y conforme el usuario acerca uno o más dedos con el dedo pulgar (formando una pinza), la figura cambia de tamaño. Si la pinza se cierra, el círculo crece a su máximo tamaño y, por el contrario, si la palma de la mano está abierta, la figura disminuye su tamaño. La interfaz de este ejercicio se muestra en la Figura 4.1.

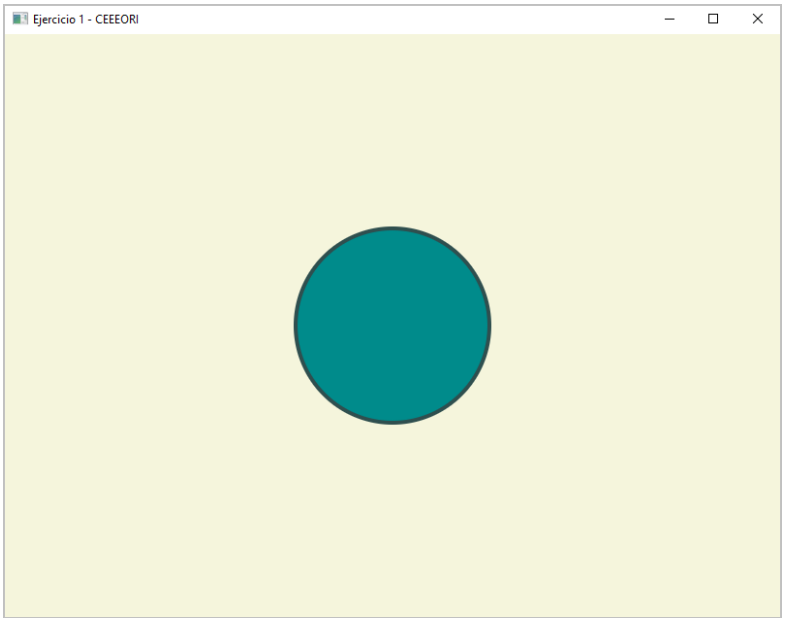


Figura 4.1. Ejercicio 1.

Ejercicio 2

Este ejercicio (Figura 4.2) está enfocado en mostrar, así como el ejercicio 1, el funcionamiento del dispositivo Leap Motion. En esta ocasión se muestran en pantalla figuras circulares que representan cada uno de los dedos de las manos, diferenciando los pulgares con las figuras de mayor tamaño. Las figuras emulan el movimiento de las manos y podrá observarse a través de la pantalla.

Este ejercicio tiene propósito de capturar la atención y percepción visual del usuario.

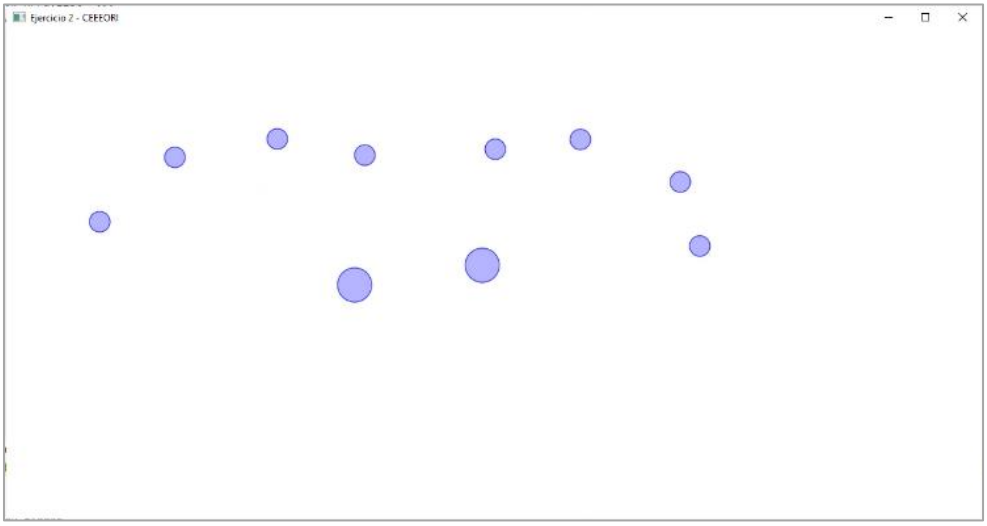


Figura 4.2. Ejercicio 2.

Ejercicio 3

Al inicio, en la parte superior de la pantalla se muestran cinco círculos fijos, cada uno con un color de relleno diferente (Figura 4.3). Por otra parte, en el centro de la pantalla se muestra una serie de diez círculos sin color de relleno, los cuales representan a cada uno de los dedos de las manos. Se busca que el usuario, a través del movimiento de sus manos (captado por el dispositivo Leap Motion) controle la posición de las figuras y las lleve hasta los círculos de colores. Al momento de tocar alguno de los círculos fijos, la figura cambiará su color de relleno

por el color correspondiente al círculo fijo que tocó. El ejercicio cuenta con los siguientes objetivos que deberán completarse.

1. El usuario debe cambiar el relleno de todas las figuras que representan los dedos de las manos, sin importar el color elegido.
2. El usuario debe elegir dos colores, uno para el relleno las figuras que representan los dedos de la mano izquierda y el otro para los de la mano derecha. Una vez elegidos los colores, deberá cambiar el relleno de los círculos ya descritos.
3. El usuario debe cambiar el relleno de los círculos que representan los dedos de la mano izquierda con el color verde y los de la mano derecha con naranja.

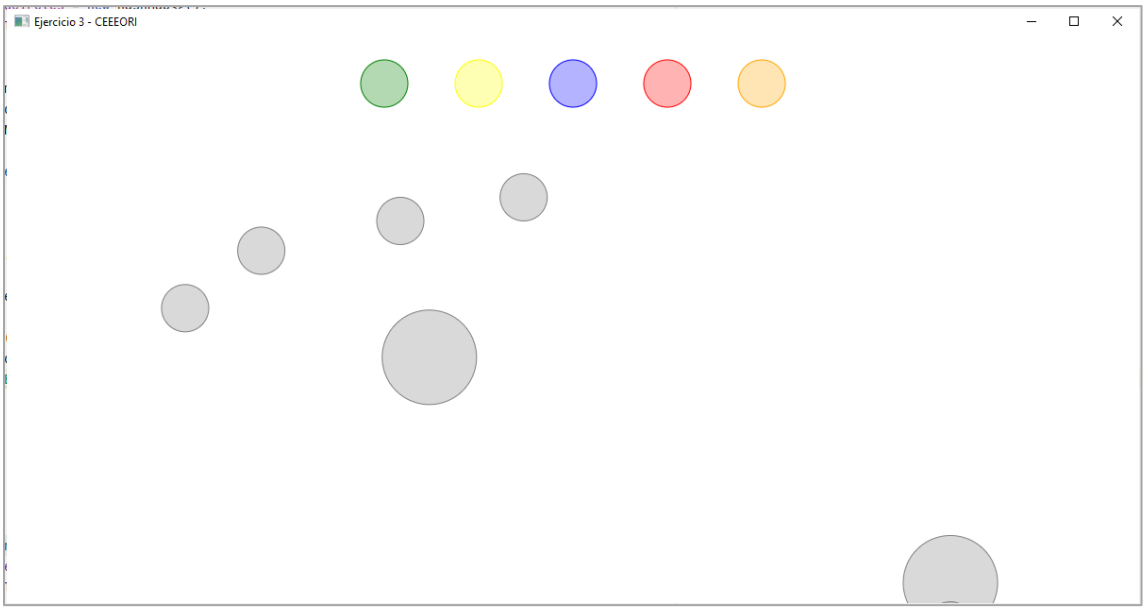


Figura 4.3. Ejercicio 3.

Ejercicio 4

El ejercicio inicia con un conjunto de figuras ubicadas en la parte superior de la pantalla (Figura 4.4), se busca trasladar a la zona correspondiente cada una de las figuras, a través del movimiento de la mano (ya sea izquierda o derecha). Para mover las piezas, se propone el uso de una figura especial, que emula la función “arrastrar y soltar” comúnmente realizada con el ratón o touchpad. Para controlar el

mecanismo de selección y “arrastre” de las figuras, se propone el uso de la pinza cerrada, presentado en el Ejercicio 1 y para liberar el movimiento de la figura (“soltar”), la pinza hecha con la mano debe abrirse. El objetivo del ejercicio consiste en colocar los triángulos en la zona izquierda y los círculos en la zona derecha.

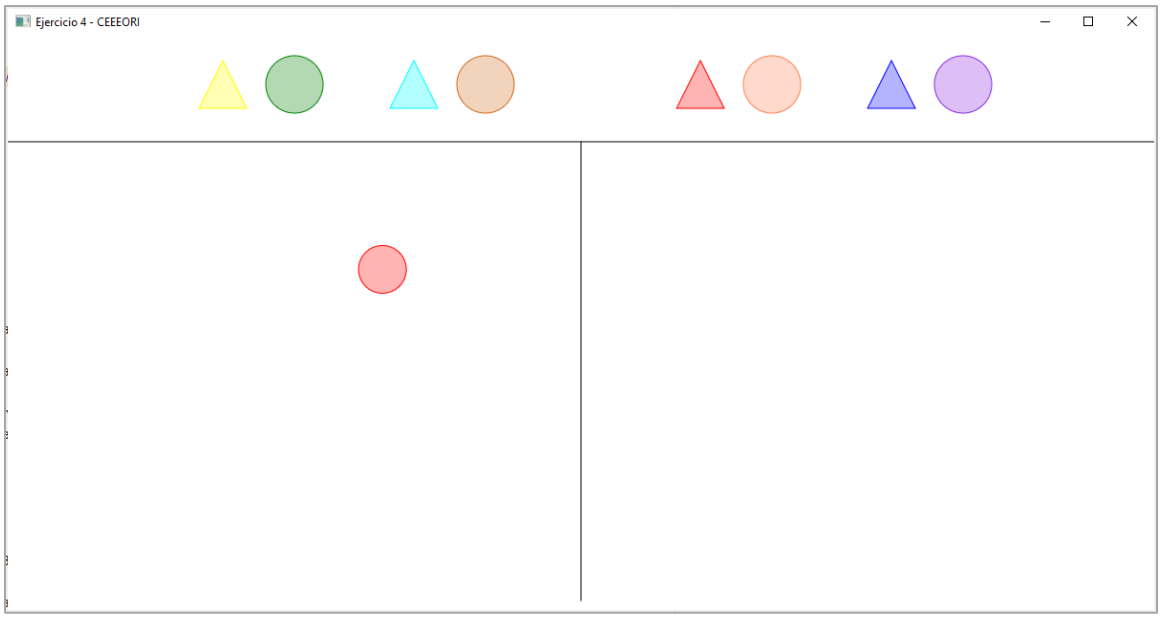


Figura 4.4. Ejercicio 4.

Ejercicio 5

Este ejercicio presenta tres imágenes de objetos (Figura 4.5), las cuales se ubican arriba de la imagen de una mesa en el centro de la pantalla. Las imágenes podrán ser movidas a través de un puntero, similar al presentado en Ejercicio 4. A los costados, en el lugar de los círculos, se propone colocar imágenes de sillas, lo cual resulta en la simulación de un comedor. Este ejercicio consiste en realizar las siguientes actividades:

1. Colocar una figura arriba de la silla derecha.
2. Colocar una figura debajo de la silla izquierda.
3. Colocar la figura restante debajo de la mesa.

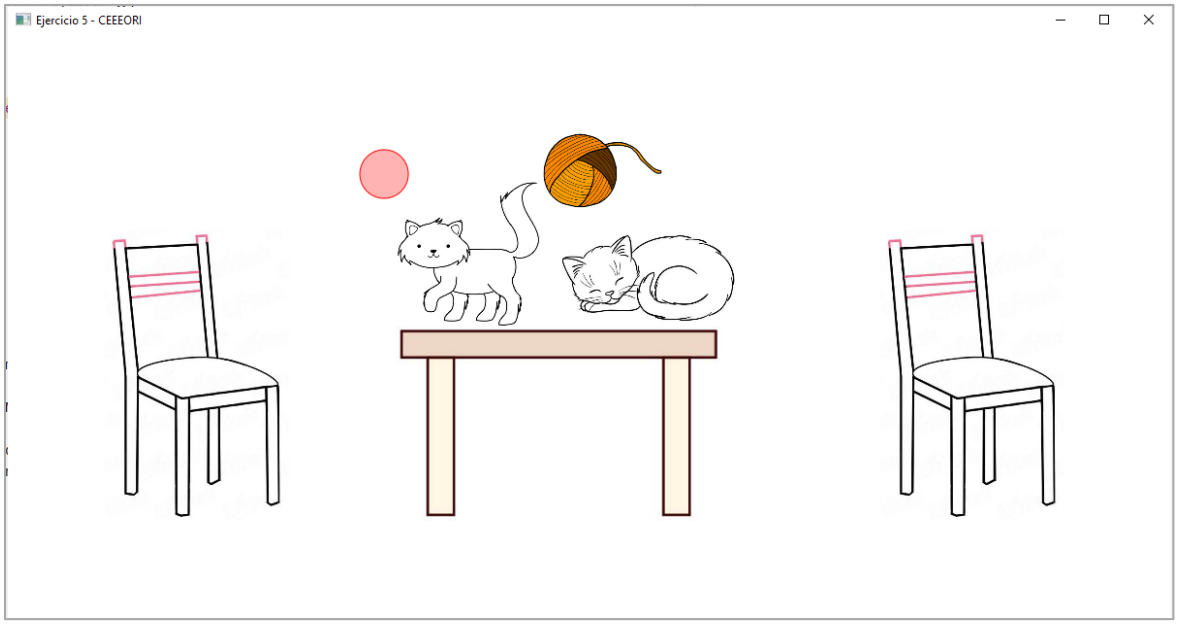


Figura 4.5. Ejercicio 5.

Ejercicio 6

Este ejercicio consiste en presentar un área delimitada, la cual contiene una serie de figuras (rectángulos y cuadrados) de distintos tamaños, dispuestos en diferentes ubicaciones (Figura 4.6). Las figuras solo se mueven en una dirección, ya sea horizontal o vertical, y no podrán sobreponerse, por lo cual se crean delimitaciones entre las piezas, que no podrán cruzar. El objetivo es mover las piezas de color gris, para liberar el camino de la pieza de color azul y llevar hasta el área de color verde, con lo cual finalizará el nivel y dará paso al siguiente (Figura 4.7). Se considera crear una serie de niveles, los cuales presenten distintos grados de dificultad, con base en la disposición y número de piezas.

Para mover las piezas, se empleará un puntero dotado con el mecanismo de “arrastrar y soltar” presentado en Ejercicio 4.

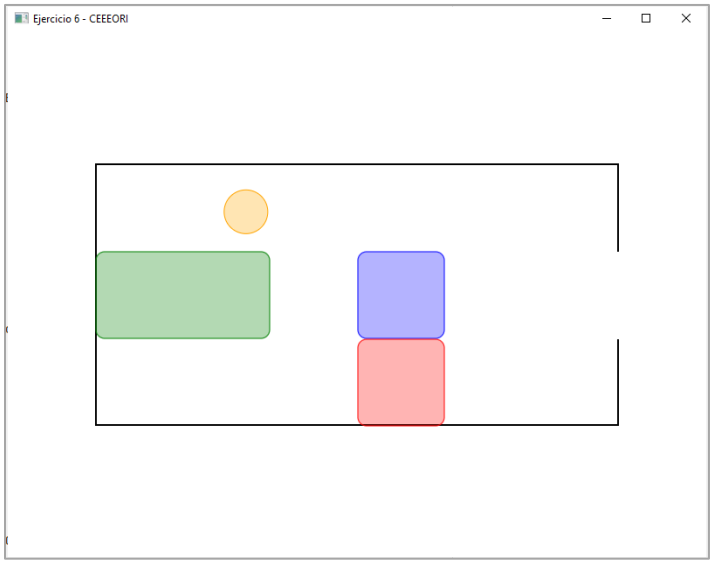


Figura 4.6. Nivel uno del Ejercicio 6.

Figura 4.7. Niveles adicionales del ejercicio 6.

4.1.7 Implementación del caso de estudio

La implementación del caso de estudio se llevó a cabo en las instalaciones del Centro de Entrenamiento y Educación Especial, contó con la participación de un alumno del centro y un niño no adjunto al centro.

La configuración utilizada para trabajar en la aplicación del caso de estudio fue la siguiente (Figura 4.8):

- Procesador Intel Core i5 1038NG7 @ 2.00GHz
- 16 GB de memoria RAM.
- Windows 10 64 bits
- Dispositivo Leap Motion
- Java 8
- Python 3.8
- Biblioteca LeapUI.

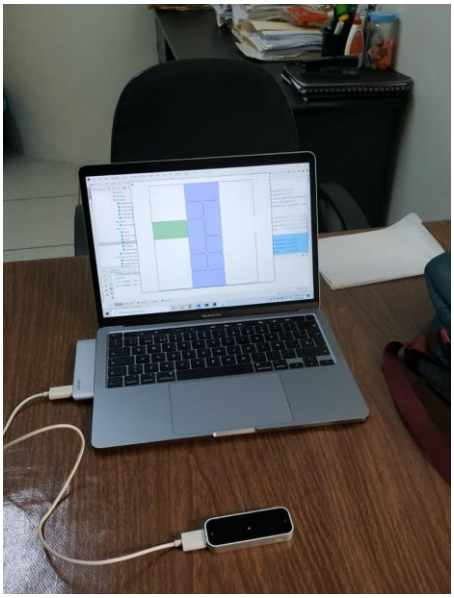


Figura 4.8. Ejercicio en funcionamiento.

En la Figura 4.9 y Figura 4.10 se aprecia a los participantes del caso de uso.

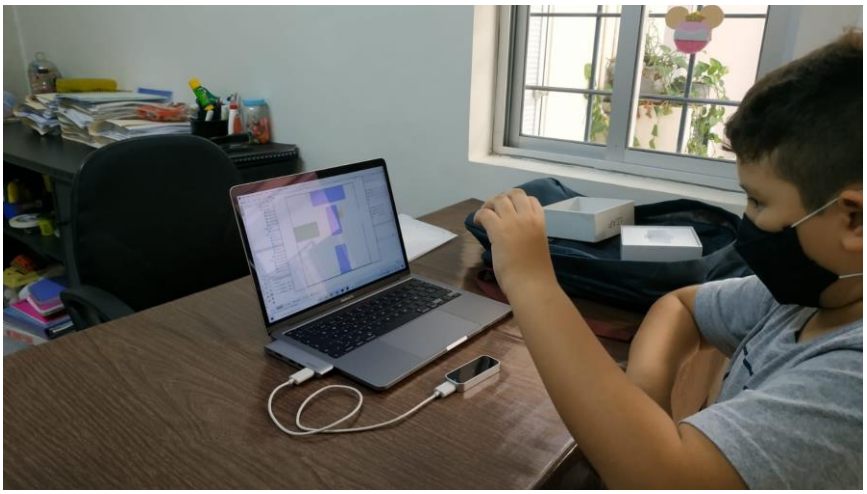


Figura 4.9. Participante uno del caso de estudio.

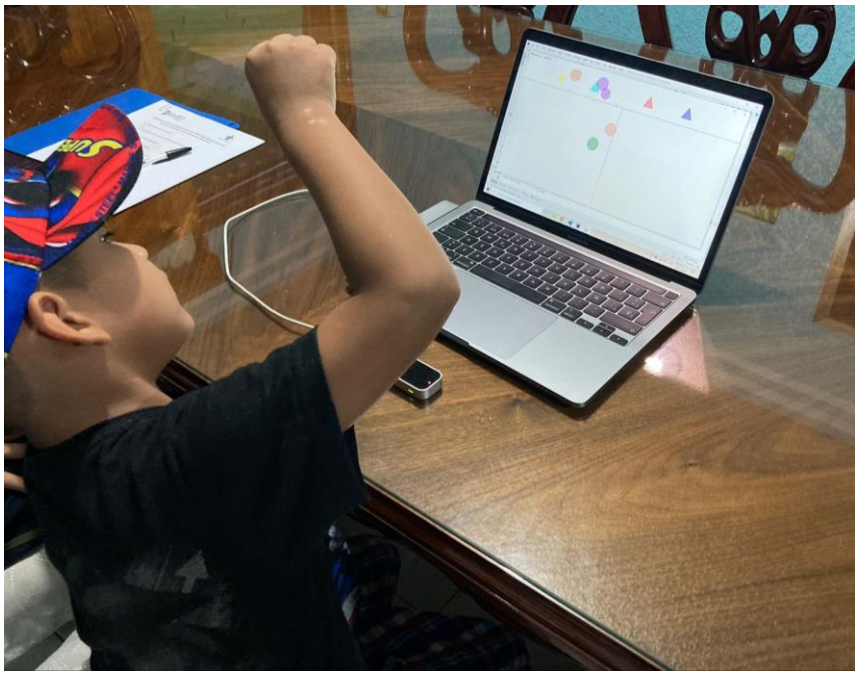


Figura 4.10. Participante dos del caso de estudio.

4.1.8 Resultado de la implementación del caso de estudio

Se aplicó un cuestionario para conocer la opinión general e individual de los ejercicios realizados por los participantes. Las preguntas que componen el cuestionario se muestran a continuación.

Preguntas del cuestionario de usabilidad

1. ¿Sabes utilizar una computadora?
   Si

   No
2. ¿Se te hizo fácil aprender cómo se usa el dispositivo Leap Motion?
   Si

   No
3. ¿Los ejercicios se te hicieron agradables?
   Si

   No

   Algunos
4. ¿Los ejercicios te parecieron entendibles?
   Si

   No
5. ¿Los ejercicios te parecieron aburridos o emocionantes?
   Aburridos

   Emocionantes
6. ¿La interacción se te hizo rápida o lenta?
   Rápida

   Lenta
7. ¿Te generó cansancio usar el dispositivo Leap Motion?
   Si

   No
8. ¿Qué ejercicio te llamó más la atención?
9. ¿Qué fue lo que te gustó de los ejercicios?
10. ¿Qué fue lo que no te gustó de los ejercicios?
11. Tienes algún comentario o sugerencia.

La Tabla 4.1 muestra los resultados obtenidos del cuestionario de usabilidad, las respuestas a las preguntas ocho, nueve, diez y once se presentan a continuación.

Tabla 4.1. Tabla de resultados del cuestionario de usabilidad

No.	Pregunta/Respuesta	Si	No
1	¿Sabes utilizar una computadora?	2	0
2	¿Se te hizo fácil aprender cómo se usa el dispositivo Leap Motion?	2	0
3	¿Los ejercicios se te hicieron agradables?	2	0
4	¿Los ejercicios te parecieron entendibles?	2	0
7	¿Te generó cansancio usar el dispositivo Leap Motion?	2	0
		Aburridos	Emocionantes
5	¿Los ejercicios te parecieron aburridos o emocionantes?	0	2
		Rápida	Lenta
6	¿La interacción se te hizo rápida o lenta?	0	2

El participante número uno contestó que el ejercicio que más le llamó la atención fue el ejercicio 3, le gustó interactuar con los colores. Lo que más le gustó de los ejercicios fue mover las figuras con la mano. Lo que no le gustó fue que la función de arrastrar y soltar no le hiciera caso.

El participante número dos contestó que el ejercicio que más le llamó la atención fue el ejercicio 5 y que lo que más le gustó fueron las imágenes y figuras. Las preguntas diez y once no tuvieron respuesta.

La evaluación individual se realizó a través de siete criterios, los cuales se tomaron del cuestionario de experiencia de usuario, el cual tiene como propósito evaluar la calidad de productos interactivos y se presentan en la Figura 4.11.

Figura 4.11. Criterios de evaluación de ejercicios.

A continuación, en la Figura 4.12 y Figura 4.13 se muestran los resultados obtenidos en las evaluaciones realizadas por los participantes.

		Ejercicio					
Valor mínimo (0)	**Valor máximo (5)**	**Uno**	**Dos**	**Tres**	**Cuatro**	**Cinco**	**Seis**
Complicado	**Fácil**	4	5	5	1	5	1
Aburrido	**Emocionante**	3	5	5	5	1	4
Rápido	**Lento**	5	5	1	5	2	4
Fácil de aprender	**Difícil de aprender**	1	1	1	2	1	1
Bueno	**Malo**	2	1	1	1	1	3
Incómodo	**Cómodo**	1	5	5	1	3	4
Claro	**Confuso**	2	1	5	1	5	2

Figura 4.12. Resultados de evaluación de los ejercicios del participante uno.

		Ejercicio					
Valor mínimo (0)	Valor máximo (5)	Uno	Dos	Tres	Cuatro	Cinco	Seis
Complicado	Fácil	5	5	5	4	2	1
Aburrido	Emocionante	4	5	5	4	4	1
Rápido	Lento	1	2	1	2	5	5
Fácil de aprender	Difícil de aprender	1	1	1	2	2	5
Bueno	Malo	1	1	2	1	2	4
Incómodo	Cómodo	4	4	5	2	4	1
Claro	Confuso	1	1	1	2	2	1

Figura 4.13. Resultados de evaluación de los ejercicios del participante dos.

Capítulo 5. Conclusiones y recomendaciones

A continuación, se presenta las conclusiones con base en el desarrollo y el caso de estudio presentados en los capítulos previos de este documento, además se presenta una serie de recomendaciones para mejorar y expandir la funcionalidad de la biblioteca.

5.1 Conclusiones

Se logró implementar los módulos que integran la biblioteca, de forma inicial se planteó la idea de desarrollar todos los módulos en el lenguaje de programación Java, aunque la dificultad que introduce la función de reconocimiento de gestos hizo que se replanteara la idea y se decidió adicionar al desarrollo el lenguaje de programación Python y sus diversas herramientas para tratar con tareas de aprendizaje automático.

Emplear Scrum como metodología de desarrollo ayudó a que los cambios en el proyecto no fueran catastróficos y permitió la transición hacia un nuevo patrón de arquitectura fácilmente.

No se necesitó almacenar archivos adicionales en el módulo de reconocimiento de gestos, debido a que el modelo entrenado integra en un solo archivo todos los datos necesarios para funcionar, además, los datos del proceso de extracción y muestreo se almacenan en tiempo de ejecución en la memoria principal del sistema de cómputo donde se ejecuta la biblioteca.

Implementar la arquitectura a través del patrón dirigido por eventos permite utilizar herramientas de distintos lenguajes de programación, además de dar pauta para implementar la sección de consumidor en distintos lenguajes de programación.

Se desarrolló una aplicación de software titulada: “MFRP: atención, motricidad fina, percepción visual, noción espacial y resolución de problemas para niños a través del dispositivo LeapMotion”. La aplicación se integra por seis ejercicios, enfocados en ejercitar las capacidades de atención, motricidad fina, percepción visual, noción espacial y resolución de problemas. Las pruebas de validación, retroalimentación y evaluación realizadas dieron a conocer que la interacción a través de las manos es

atractiva y fácil de aprender para los niños, aunque en algunas situaciones es poco precisa, lo que genera frustración en el niño.

La integración del módulo de reconocimiento de gestos no está limitado a los gestos descritos en la biblioteca, es posible utilizarlo de tal forma que reconozca nuevos gestos, aunque previamente deba entrenarse un nuevo modelo.

5.2 Recomendaciones

Utilizar por largos periodos de tiempo el dispositivo LeapMotion causa fatiga en manos y brazos, por lo que se recomienda utilizarlo en sesiones cuya duración no supere los 40 minutos".

La arquitectura permite mejorar y extender el modelo de aprendizaje automático empleado en el módulo de reconocimiento de gestos, la herramienta de extracción de muestras podrá emplearse para obtener un número mayor de secuencias de gestos realizados, para posteriormente entrenar el modelo con el *script* incluido en la biblioteca.

El fabricante del dispositivo Leap Motion ha publicado, al momento de escribir este documento, la versión 5.2.0 (Gemini) del SDK. Esta versión ofrece mejoras en la precisión y robustes de la detección del movimiento de las manos, además de mejorar el rendimiento a través de un amplio rango de tamaño de manos.

Emplear la nueva versión del API escrita en el lenguaje de programación C, en los módulos de reconocimiento de gestos y captura de *frames* podría traer como beneficio un incremento en la velocidad de ejecución, además de aquellos incluidos con la actualización.

Productos académicos

Ricardo Arellano Morales, María Antonieta Abud Figueroa, Lisbeth Rodríguez Mazahua, Hilarión Muñoz Contreras and Ulises Juárez Martínez.

Architecture for the integration of the Leap Motion device into application development

Estado: Presentado.

Referencias

[1] Q. Ke, J. Liu, M. Bennamoun, S. An, F. Sohel, y F. Boussaid, «Chapter 5 - Computer Vision for Human–Machine Interaction», en *Computer Vision for Assistive Healthcare*, M. Leo y G. M. Farinella, Eds. Academic Press, 2018, pp. 127-145. doi: 10.1016/B978-0-12-813445-0.00005-8.

[2] D. Bachmann, F. Weichert, y G. Rinkenauer, «Review of Three-Dimensional Human-Computer Interaction with Focus on the Leap Motion Controller», *Sensors*, vol. 18, n.º 7, Art. n.º 7, jul. 2018, doi: 10.3390/s18072194.

[3] J. J. L. Jr, E. Kruijff, R. P. McMahan, D. Bowman, y I. P. Poupyrev, *3D User Interfaces: Theory and Practice*. Addison-Wesley Professional, 2017.

[4] K.-L. Kramer, «Chapter 2 - Approaches to A Sustainable User Experience», en *User Experience in the Age of Sustainability*, K.-L. Kramer, Ed. Boston: Morgan Kaufmann, 2012, pp. 31-68. doi: 10.1016/B978-0-12-387795-6.00002-0.

[5] N. Sebe, «Human-centered Computing», en *Handbook of Ambient Intelligence and Smart Environments*, H. Nakashima, H. Aghajan, y J. C. Augusto, Eds. Boston, MA: Springer US, 2010, pp. 349-370. doi: 10.1007/978-0-387-93808-0_13.

[6] V. D. Goodman, «5 - Inside the mind of the user: qualitative approaches to understanding user experience in library settings», en *Qualitative Research and the Modern Library*, V. D. Goodman, Ed. Chandos Publishing, 2011, pp. 107-124. doi: 10.1016/B978-1-84334-644-9.50005-6.

[7] D. Saffer, *Designing Gestural Interfaces: Touchscreens and Interactive Devices*. O'Reilly Media, Inc., 2008.

[8] T. Allevard, E. Benoit, y L. Foulloy, «Hand posture recognition with the fuzzy glove», en *Modern Information Processing*, B. Bouchon-Meunier, G. Coletti, y R. R. Yager, Eds. Amsterdam: Elsevier Science, 2006, pp. 417-427. doi: 10.1016/B978-044452075-3/50035-2.

[9] A. Cooper, R. Reimann, D. Cronin, y C. Noessel, *About Face: The Essentials of Interaction Design*. John Wiley & Sons, 2014.

[10] R. Hartson y P. S. Pyla, «Chapter 1 - Introduction», en *The UX Book*, R. Hartson y P. S. Pyla, Eds. Boston: Morgan Kaufmann, 2012, pp. 1-46. doi: 10.1016/B978-0-12-385241-0.00001-4.

[11] N. Breslauer, I. Galić, M. Kukec, y I. Samardžić, «Leap Motion Sensor for Natural User Interface», *Tehnički vjesnik*, vol. 26, n.º 2, pp. 560-565, abr. 2019, doi: 10.17559/TV-20181012093055.

[12] W. R. Sherman y A. B. Craig, «CHAPTER 3 - Interface to the Virtual World — Input», en *Understanding Virtual Reality*, W. R. Sherman y A. B. Craig, Eds. San Francisco: Morgan Kaufmann, 2003, pp. 75-112. doi: 10.1016/B978-155860353-0/50004-5.

[13] I. Zoppis, G. Mauri, y R. Dondi, «Kernel Machines: Applications», en *Encyclopedia of Bioinformatics and Computational Biology*, S. Ranganathan, M. Gribskov, K. Nakai, y C. Schönbach, Eds. Oxford: Academic Press, 2019, pp. 511-518. doi: 10.1016/B978-0-12-809633-8.20343-9.

[14] T. Mantecón, C. R. del-Blanco, F. Jaureguizar, y N. García, «A real-time gesture recognition system using near-infrared imagery», *PLOS ONE*, vol. 14, n.º 10, p. e0223320, oct. 2019, doi: 10.1371/journal.pone.0223320.

[15] V. Gentile, D. Fundarò, y S. Sorce, «Elicitation and evaluation of zoom gestures for touchless interaction with desktop displays», en *Proceedings of the 8th ACM International Symposium on Pervasive Displays*, Palermo, Italy, jun. 2019, pp. 1-7. doi: 10.1145/3321335.3324934.

[16] A. Boudjelthia *et al.*, «Enabling Mid-air Browser Interaction with Leap Motion», en *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, Singapore, Singapore, oct. 2018, pp. 335-338. doi: 10.1145/3267305.3267580.

[17] J. Cui y A. Sourin, «Interactive shape modeling using leap motion controller», en *SIGGRAPH Asia 2017 Technical Briefs*, Bangkok, Thailand, nov. 2017, pp. 1-4. doi: 10.1145/3145749.3149437.

[18] J. Müller *et al.*, «Mouse, touch, or fich: comparing traditional input modalities to a novel pre-touch technique», en *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia*, Pisa, Italy, nov. 2019, pp. 1-7. doi: 10.1145/3365610.3365622.

[19] J. Schioppo, Z. Meyer, D. Fabiano, y S. Canavan, «Sign Language Recognition: Learning American Sign Language in a Virtual Environment», en *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, Glasgow, Scotland Uk, may 2019, pp. 1-6. doi: 10.1145/3290607.3313025.

[20] T. Guzsvinecz, V. Szucs, y C. Sik-Lanyi, «Suitability of the Kinect Sensor and Leap Motion Controller—A Literature Review», *Sensors*, vol. 19, n.º 5, Art. n.º 5, ene. 2019, doi: 10.3390/s19051072.

[21] L. E. Potter, J. Araullo, y L. Carter, «The Leap Motion controller: a view on sign language», en *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, Adelaide, Australia, nov. 2013, pp. 175-178. doi: 10.1145/2541016.2541072.

[22] «API Overview — Leap Motion Python SDK v3.2 Beta documentation». https://developer-archive.leapmotion.com/documentation/python/devguide/Leap_Overview.html (accedido nov. 23, 2021).

[23] «hand-hierarchy», *Leap Motion Blog*. http://blog.leapmotion.com/wp-content/uploads/2014/08/hand-hierarchy.png (accedido mar. 10, 2021).

[24] «Coordinate Systems — Leap Motion Python SDK v3.2 Beta documentation». https://developer-archive.leapmotion.com/documentation/python/devguide/Leap_Coordinate_Mapping.html (accedido nov. 23, 2021).

[25] «Design guide for gesture control in automotive HMI | Ultraleap». https://www.ultraleap.com/company/news/blog/gesture-control-automotive-hmi/ (accedido mar. 10, 2021).

[26] D. Avola, M. Bernardi, L. Cinque, G. L. Foresti, y C. Massaroni, «Exploiting Recurrent Neural Networks and Leap Motion Controller for the Recognition of Sign Language and Semaphoric Hand Gestures», *IEEE Transactions on Multimedia*, vol. 21, n.º 1, pp. 234-245, ene. 2019, doi: 10.1109/TMM.2018.2856094.

[27] H. Doan, H. Vu, y T.-H. Tran, «Dynamic hand gesture recognition from cyclical hand pattern», *undefined*, 2017, Accedido: may 24, 2021. [En línea]. Disponible en: /paper/Dynamic-hand-gesture-recognition-from-cyclical-hand-Doan-Vu/3640ef638cc1c8c8e29be78ab17eda458b086972

[28] S. Hochreiter y J. Schmidhuber, «Long Short-term Memory», *Neural computation*, vol. 9, pp. 1735-80, dic. 1997, doi: 10.1162/neco.1997.9.8.1735.