



**UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERÍA**

ESCUELA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

**“Trabajo de grado previo a la obtención del Título de Ingeniero en Electrónica y
Telecomunicaciones”**

TRABAJO DE GRADUACIÓN

Título del proyecto:

**DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA INTERACTIVO MEDIANTE
TECNOLOGÍA KINECT V2.0 PARA DESARROLLAR LAS HABILIDADES
PSICOMOTRICES EN PERSONAS CON DISCAPACIDAD VISUAL.**

Autores:

Mayra Viviana Cujano Ortega
Catherine Iralda Vera Gaibor

Director:

Ing. Aníbal Llanga Mgs.

Riobamba – Ecuador

AÑO 2016

Los miembros del Tribunal de Graduación del proyecto de investigación de título: **DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA INTERACTIVO MEDIANTE TECNOLOGÍA KINECT V2.0 PARA DESARROLLAR LAS HABILIDADES PSICOMOTRICES EN PERSONAS CON DISCAPACIDAD VISUAL** presentado por: **Mayra Viviana Cujano Ortega, Catherine Iralda Vera Gaibor** y dirigida por: **Ingeniero Aníbal Llanga**.

Una vez escuchada la defensa oral y revisado el informe final del proyecto de investigación con fines de graduación escrito en la cual se ha constatado el cumplimiento de las observaciones realizadas, remite la presente para uso y custodia en la biblioteca de la Facultad de Ingeniería de la UNACH.

Para constancia de lo expuesto firman:

Ing. Paulina Vélez

Presidente del Tribunal



Firma

Ing. Aníbal Llanga

Director de Tesis



Firma

Ing. Juan Carlos Santillán

Miembro del Tribunal



Firma

CERTIFICACIÓN DEL TUTOR

Certifico que el presente trabajo de investigación previo a la obtención del grado de Ingeniero en ELECTRONICA Y TELECOMUNICACIONES. Con el tema: **“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA INTERACTIVO MEDIANTE TECNOLOGÍA KINECT V2.0 PARA DESARROLLAR LAS HABILIDADES PSICOMOTRICES EN PERSONAS CON DISCAPACIDAD VISUAL”** ha sido elaborado por las estudiantes **Mayra Viviana Cujano Ortega** y **Catherine Iralda Vera Gaibor**, el mismo que ha sido revisado y analizado en un cien por ciento con el asesoramiento permanente de mi persona en calidad de Tutor por lo que se encuentran aptas para su presentación y defensa respectiva.

Es todo cuanto puedo informar en honor de la verdad.



Ing. Aníbal Llanga
C.I 060293332-7

AUTORÍA DE LA INVESTIGACIÓN

La responsabilidad del contenido de este Proyecto de Graduación, nos corresponde exclusivamente a: **Mayra Viviana Cujano Ortega, Catherine Iralda Vera Gaibor e Ing. Aníbal Llanga Vargas**; y el patrimonio intelectual de la misma a la Universidad Nacional de Chimborazo.



Mayra Viviana Cujano Ortega
C.I. 060409934-1



Catherine Iralda Vera Gaibor
C.I. 020157183-3

AGRADECIMIENTO

Este presente trabajo es la muestra de haber llegado al final de nuestra vida universitaria.

Agradecemos a Dios que es un ser maravilloso y nos dio las fuerzas para cumplir nuestros objetivos.

A nuestras familias, en especial a nuestras madres que con sus consejos nos ayudaron a afrontar los retos que se han presentado a lo largo de nuestra carrera universitaria.

A nuestro tutor de tesis, Ing. Aníbal Llanga quien nos brindó su confianza y nos motivó a realizar un excelente trabajo.

A todos los docentes y amigos de la escuela de electrónica quienes nos han compartido sus enseñanzas y brindado su apoyo incondicional para cristalizar nuestra meta.

Mayra Cujano

Catherine Vera

DEDICATORIA

Dedico este trabajo principalmente a Dios, por haberme dado la vida y permitirme llegar a concluir mi carrera. A mi madre Martita por ser el pilar más importante y el principal cimiento para la construcción de mi vida profesional, que sentó en mí las bases de responsabilidad y deseos de superación. A mis hermanos Santi, Sebas y Erick puesto que en sus corazones siempre existió su aliento lo cual me sirvió para ponerme fuerte ante los problemas y superarlos de la mejor manera. A mi novio Jonathan quien estuvo siempre al pendiente de mí y supo llenarme de cariño y comprensión en los momentos más difíciles de mi vida. A Dalton, a mis abuelitos y a toda mi familia quienes me brindaron su confianza y han contribuido para el logro de mi objetivo.

Mayra Cujano

DEDICATORIA

Dedico este trabajo de manera especial a Iralda mi madre, por ser una mujer admirable ya que ha sido siempre un ejemplo de lucha, esfuerzo y superación personal.

A mi abuelita Amada que ha sabido aconsejarme y brindarme su apoyo y cariño incondicionalmente.

A mi querida tía Sarita que fue mi gran apoyo intelectual, y aunque no está presente la llevo siempre en mi corazón y jamás olvidaré las sabias palabras que compartía conmigo.

Y a mi familia quienes han estado presentes en todo momento.

Catherine Vera

ÍNDICE GENERAL

CERTIFICACIÓN DEL TUTOR	iii
ÍNDICE DE FIGURAS.....	xii
ÍNDICE DE TABLAS.....	xiv
RESUMEN.....	xv
ABSTRACT	xvi
INTRODUCCIÓN	1
CAPITULO I.....	2
1.FUNDAMENTACIÓN TEÓRICA.....	2
1.1.Medios empleados	2
1.1.1.Elementos Hardware	2
1.1.2.Elementos Software	3
1.2.Interfaz de usuario	3
1.3.Estado del arte	4
1.3.1.Kinect.....	4
1.3.1.1.Efecto Kinect.....	4
1.3.1.2.Aplicaciones del Sensor Kinect	6
1.3.2.Sistemas para discapacitados visuales	6
1.3.2.1.Funciones y tipos.....	6
1.4.Discapacidad visual.....	9
1.4.1.Terminología Histórica.	9
1.4.2.Aspectos Generales	9
1.4.3.Características Físicas y Psicológicas de personas con discapacidad visual.....	10
1.4.3.1.Características Psicomotrices en personas con discapacidad visual.....	10
1.4.4.Eschema Corporal en personas con discapacidad visual.	11
1.4.4.1.Características que presentan los ciegos y deficientes visuales	11
1.4.5.Actividades para desarrollar la psicomotricidad en personas con discapacidad visual.....	12
1.5.Hardware.....	13
1.5.1.Sensor Kinect v2.0 para Windows	13
1.5.1.1.Definición.....	13
1.5.1.2.Características	14

1.5.1.3.Especificaciones del Kinect	16
1.5.1.4.Consideraciones físicas del Sensor Kinect:.....	16
1.5.1.5.Consideraciones de audio y sonido del Sensor Kinect	19
1.5.1.6.Funcionamiento Básico	21
1.6.Ordenador.....	23
1.7.Software.....	23
1.7.1.Visual Studio.....	23
1.7.1.1.Introducción	23
1.7.1.2.Plataforma .NET	24
1.7.1.3.Microsoft Visual C#.....	25
1.7.2.Software Development Kit (SDK v2.0)	28
1.7.2.1.Skeletal Tracking	29
1.7.3.Speech Recognition.....	30
1.7.4.Base de Datos	33
1.7.4.1.SQL Server 2008.....	33
1.7.4.2.Crystal Reports.....	34
CAPITULO II	35
2.METODOLOGÍA	35
2.1.Tipo de estudio.....	35
2.2.Métodos, técnicas e instrumentos	35
2.2.1.Métodos	35
2.2.1.1.Analítico.....	35
2.2.1.2.Sintético.....	35
2.2.2.Técnicas	36
2.2.2.1.Observación.....	36
2.2.3.Instrumentos	36
2.3.Población y muestra	36
2.4.Hipótesis.....	37
2.5.Operacionalización de variables	38
2.6.Procedimientos	39
2.6.1.Identificación y estudio del sistema.	39
2.6.2.Selección y estudio de Hardware y Software.....	39
2.6.3.Etapas de la Aplicación.....	41
2.6.3.1.Interfaz entre usuario y máquina.....	41
2.6.3.2.Servidor de datos.....	42

2.6.3.3.Clientes.....	42
2.6.4.Programación	42
2.6.4.1.Diagrama de Flujo de la Programación.....	43
2.6.5.Recolección de datos a base de pruebas.....	45
2.7.Procesamiento y análisis	45
2.7.1.Inicio de KinectMove.....	45
2.7.2.Menú Principal de KinectMove	46
2.7.3.Actividades Psicomotrices	47
2.7.3.1.Actividad para desarrollar el Conocimiento Corporal	48
2.7.3.2.Actividad para desarrollar la Coordinación de Brazos	52
2.7.3.3.Actividades para desarrollar la coordinación de piernas.....	55
2.7.3.4.Actividades para desarrollar la Locomoción.....	57
2.7.3.5.Actividades para identificar la Posición.....	60
2.7.4.Registro y Reporte de Datos	61
2.8.Comprobación de la hipótesis.	64
2.8.1.Planteamiento de la hipótesis estadística.	64
2.8.2.Establecimiento de nivel de significancia.	64
2.8.3.Determinación del valor estadístico de prueba.	64
CAPITULO III.....	68
3.RESULTADOS.....	68
CAPITULO IV	73
4.DISCUSIÓN.....	73
CAPITULO V.....	75
5.CONCLUSIONES Y RECOMENDACIONES	75
5.1.Conclusiones	75
5.2.Recomendaciones	76
CAPITULO VI	77
6.PROPOSTA.....	77
6.1.Título de la propuesta	77
6.2.Introducción.....	77
6.3.Objetivos.....	78
6.3.1.Objetivo General	78
6.3.2.Objetivos Específicos.....	78

6.4.Fundamentación Científico – Técnico.....	78
6.5.Descripción de la propuesta	79
6.6.Diseño Organizacional	80
6.7.Monitoreo y Evaluación de la propuesta	80
7.BIBLIOGRAFÍA.....	81
8.ANEXOS	83

ÍNDICE DE FIGURAS

Figura 1. Lector de Pantalla para Invidentes.....	7
Figura 2. Prototipo de gafas electrónicas para personas no videntes.....	8
Figura 3. Anillo para leer texto	8
Figura 4. Sensor Kinect v2.0.....	14
Figura 5. Adaptador Kinect para Windows.....	14
Figura 6. Componentes del Sensor Kinect v2.0.....	15
Figura 7. Especificaciones del Sensor Kinect v2.0.....	16
Figura 8. Rango de profundidad.....	16
Figura 9. Ángulo de Visión.....	17
Figura 10. Seguimiento de esqueletos.....	17
Figura 11. Modelo del esqueleto completo	18
Figura 12. Modo Sentado.....	18
Figura 13. Entrada de Audio	19
Figura 14. Micrófonos.....	19
Figura 15. Umbral de sonido.....	20
Figura 16. Micrófono direccional	20
Figura 17. Fuente de localización	21
Figura 18. Crear un nuevo proyecto en C#	26
Figura 19. Crear un nuevo proyecto en C#	27
Figura 20. Crear un nuevo proyecto en C#	27
Figura 21. Agregar Referencia Microsoft.Kinect	28
Figura 22. Agregar Referencia Microsoft.Kinect	29
Figura 23. Joints del cuerpo humano	30
Figura 24. Agregar Librería Speech.Platform.....	32
Figura 25. Agregar Librería Speech.Platform.....	32
Figura 26. Diagrama de procedimientos	39
Figura 27. Diagrama esquemático del sistema interactivo.....	41
Figura 28. Flujograma de la programación, parte 1	43
Figura 29. Flujograma de la programación, parte 2.....	44
Figura 30. Pantalla Inicial	45
Figura 31. Pantalla Menú Principal.....	47
Figura 32. Flujograma Conocimiento Corporal.....	49
Figura 33. Pantalla Conocimiento Corporal.....	50
Figura 34. Flujograma Coordinación de brazos.....	53
Figura 35. Flujograma Coordinación de piernas.....	56
Figura 36. Flujograma Locomoción.....	59
Figura 37. Flujograma Posición	60
Figura 38. Conexión SQL – C#.....	62
Figura 39. SQL Server	62
Figura 40. Elemento Crystal Report.....	63
Figura 41. Resultado de comparación X^2 tabla con el X^2 calculado	67
Figura 42. Usuarios interactuando con KinectMove.....	68
Figura 43. Coordinación de brazos	69
Figura 44. Coordinación de piernas	70
Figura 45. Reconocimiento de posición.....	70
Figura 46. Servidor web.....	71
Figura 47. Reporte de datos del usuario.....	72

Figura 48. Esquema organizacional 80

ÍNDICE DE TABLAS

Tabla 1. Diferencia entre versiones del sensor.....	5
Tabla 2. Operacionalización de Variables	38
Tabla 3. Asignación de variables – coordinación de brazos	54
Tabla 4. Asignación de variables	55
Tabla 5. Numero de muestras.....	65
Tabla 6. Frecuencia obtenida	66
Tabla 7. Frecuencia esperada	66
Tabla 8. Tabla de resultados.....	66

RESUMEN

El proyecto de investigación “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA INTERACTIVO MEDIANTE TECNOLOGÍA KINECT V2.0 PARA DESARROLLAR LAS HABILIDADES PSICOMOTRICES EN PERSONAS CON DISCAPACIDAD VISUAL”, es desarrollado con la finalidad de ser un aporte para personas que presentan discapacidad visual.

El sistema interactivo está compuesto por elementos hardware es decir un ordenador Toshiba Core I5, 6 GB de memoria RAM y un sensor Kinect versión 2.0 con su respectivo adaptador para Windows. Además cuenta con elementos software como el SDK de Kinect, Visual Studio 2013 y paquetes necesarios para el reconocimiento de voz y audio, que al ser acoplados permiten que el sistema interactúe sin utilizar ningún dispositivo de mando.

El nombre que se da a este sistema interactivo es KinectMove el cual tiene cinco actividades para desarrollar la psicomotricidad, las mismas que han sido programadas en el compilador C# de Visual Studio, el usuario a través de comandos de voz inicializará la aplicación registrando sus datos como nombre, edad, sexo, permitiéndole seleccionar la actividad que desee realizar.

KinectMove además cuenta con un registro de datos remotos que son alojados en un servidor web, permitiéndole así a cualquier usuario observar el reporte que genera la aplicación. El almacenamiento de estos datos se lo hace utilizando el software SQL server 2008 ya que permite trabajar en modo cliente-servidor.

En conclusión KinectMove es capaz de reconocer al esqueleto del usuario en cualquier ambiente que desee realizar las actividades e interactuar con él por medio de comandos de voz y audio, para trabajos futuros se recomienda la utilización de filtros los cuales captan únicamente la voz del usuario, ya que el Kinect no es un sensor inteligente y tiende a confundir los datos que recibe



Dra. Janneth Caizaguano Mg

20 de Julio del 2016

ABSTRACT

The research project "DESIGN AND IMPLEMENTATION OF AN INTERACTIVE SYSTEM USING TECHNOLOGY KINECT V2.0 TO DEVELOP PSYCHOMOTOR SKILLS VISUALLY IMPAIRED PEOPLE", It is developed in order to be a contribution for people who have a visual impairment.

The interactive system is comprised of hardware elements that is a computer Toshiba Core I5, 6GB of RAM and a Kinect version 2.0 sensor with its own adapter for Windows. It also has software elements such as Kinect SDK, Visual Studio 2013 and packages needed for voice recognition and audio, which when coupled allow the system to interact without using any control device.

The name given to this interactive system is KinectMove which has five activities to develop psychomotor skills, the same that have been programmed in C # compiler of Visual Studio, the user through voice commands initialize the recording application data such as name, age, sex, allowing you to select the activity you want to perform.

Move Kinect also has a remote data record are hosted on a web server, allowing any user to view the report generated by the application. The storage of these data makes using the SQL Server 2008 software because it enables work in client-server mode.

In conclusion Move Kinect can recognize the skeleton of the user in any environment you want to perform the activities and interact with it through voice commands and audio, for future work is recommended using filters which capture only the user's voice, as the Kinect is not a smart sensor and tends to confuse the data it receives.



INTRODUCCIÓN

Las discapacidades físicas generan muchas dificultades para las personas que las padecen, las mismas que impiden su desenvolvimiento normal, limitándolos a través del tiempo de oportunidades laborales, acceso a espacios públicos e inclusive en la misma sociedad, para evitar esta problemática, el ser humano ha buscado formas de implementar maneras de ayudar y permitir a las personas discapacitadas equiparar sus oportunidades.

Con el avance de la tecnología el ser humano ha llegado, de cierto modo, a facilitar su estilo de vida, es por ello que viendo las bondades de la tecnología y limitantes de individuos con discapacidad visual, se ha promovido el desarrollo de un sistema interactivo que permita explotar las potencialidades que un individuo discapacitado no posee, para que con el tiempo realicen actividades que las permitan conocer el mundo que los rodea y mejorar su diario vivir.

Por los antecedentes expuestos anteriormente, la presente investigación centra su objeto de estudio en el diseño e implementación de un sistema interactivo utilizando el sensor Kinect v2.0 porque es muy importante el desarrollo de la psicomotricidad en personas con discapacidad visual para lograr un control y coordinación de sus movimientos.

El Capítulo I describe el enfoque teórico y la definición de los términos básicos necesarios para el desarrollo de la idea principal del funcionamiento del sistema interactivo. El Capítulo II detalla el Marco Metodológico que se utilizó en la investigación, así como las técnicas e instrumentos de la investigación empleada y los pasos que se siguieron para el desarrollo del proyecto. El Capítulo III muestra los resultados obtenidos al ejecutar la aplicación con personas que tienen discapacidad visual. El Capítulo IV plantea la discusión de la investigación. El Capítulo V expone las conclusiones y las recomendaciones, el Capítulo VI explica la propuesta y en la parte final de la investigación se presenta la Bibliografía y Anexos.

CAPITULO I

1. FUNDAMENTACIÓN TEÓRICA

1.1. Medios empleados

A continuación se describe brevemente los medios empleados, tanto hardware como software. Para la realización de este proyecto.

1.1.1. Elementos Hardware

Los elementos hardware utilizados son los detallados a continuación:

- a) **Ordenador:** Se ha utilizado un ordenador portátil tanto para el desarrollo de la aplicación, como para la elaboración de la documentación relacionada. El modelo utilizado ha sido un computador Toshiba con procesador Intel Core I5, 6 GB de memoria RAM y sistema operativo Windows 8.1.
- b) **Sensor Kinect v2.0:** En cuanto a los detalles del sensor, se verán en profundidad a lo largo del documento.
- c) **Adaptador Kinect para Windows:** Para que el sensor funcione correctamente en el ordenador, se debe disponer de un adaptador que conecte el Kinect a un puerto USB de la pc, así como a una alimentación externa. Esto es debido a que el puerto USB no proporciona la suficiente energía al sensor Kinect para que pueda hacer uso de todo su potencial.

1.1.2. Elementos Software

Los elementos software utilizados para el desarrollo del proyecto son los siguientes:

- a) **Kinect for Windows SDK 2.0:** Librería de desarrollo oficial de Microsoft para el desarrollo de aplicaciones para el sensor Kinect. Incluye drivers y documentación técnica para la implementación de aplicaciones, APIs de referencia y documentación para la programación.
- b) **Microsoft Visual Studio 2013:** Entorno de desarrollo completo para la programación de aplicaciones en Visual C# .NET, lenguaje de programación utilizado en el proyecto para realizar la aplicación.
- c) **Microsoft Office 2010:** Se ha utilizado Microsoft Office como herramienta para la elaboración del presente documento.

1.2. Interfaz de usuario

Dentro de cualquier sistema de un ordenador, una interfaz de usuario se considera el aspecto más importante, debido a que, para muchos usuarios la propia interfaz es el sistema. Es la parte del sistema que se puede ver, escuchar, e interactuar con ella. Los objetivos principales que hay que buscar a la hora de modelar una buena interfaz es conseguir que el usuario pueda trabajar con el ordenador o sistema de forma sencilla, productiva, y en algunos casos, entretenida. (Galitz.O, 2007)

La búsqueda del diseño de una buena interfaz de usuario ha dado pie a un campo de estudio propio llamado Interacción Persona-Ordenador (Human-Computer Interaction, HCI). El HCI es el estudio, planificación y diseño de cómo los humanos interactúan con los ordenadores para poder conseguir que las necesidades de esta persona sean satisfechas de la forma más efectiva posible.

Con estas ideas en mente, una interfaz de usuario queda definida como las partes físicas de un ordenador, así como su software, que una persona/usuario podrá ver, escuchar, comprender, interactuar y dirigir. Una buena interfaz de usuario deberá proveer de buenos mecanismos de entrada y salida de forma que se puedan satisfacer todas las necesidades del usuario, tomando en cuenta sus capacidades y limitaciones, de la forma más eficaz posible. (Trilles, 2012)

1.3. Estado del arte

En este punto se analizará brevemente las diferentes posibilidades encontradas para el desarrollo de este proyecto.

1.3.1. Kinect

1.3.1.1. Efecto Kinect

Kinect es una tecnología que permite a los usuarios interactuar y controlar una aplicación, sin la necesidad del contacto físico, mediante una interfaz natural de usuario que permite reconocer, gestos, comandos de voz, objetos e imágenes. El efecto Kinect es un fenómeno mundial al que se ha sumado, Ecuador, desarrollando diferentes aplicaciones de ayuda casera e innovadoras para el desarrollo de la comodidad de la ciudadanía en general.

La parte fundamental del éxito del Kinect es el software creado por Microsoft para la interpretación de la información obtenida por los diferentes sensores, cámara RGB, infrarrojos y micrófonos, aquí es donde Microsoft Research ha desempeñado un papel fundamental, e incluso hoy en día, investigando los más diversos usos para Kinect en colaboración con el SDK, ingresando muy fuerte en la tecnología y la ayuda de diferentes ámbitos sociales. (Quishpe & Ulloa, 2014)

El sensor Kinect mediante el software que incluye nos proporciona datos precisos de la posición de determinados puntos del cuerpo humano mediante grabación.

Estos puntos son los mismos que se utilizan en casi la mayoría de los métodos de evaluación de posición. (Herreros, 2013).

El sensor ha evolucionado de acuerdo a las necesidades de los usuarios, en consecuencia se creó el sensor Kinect versión 2, la diferencia con su predecesor es la nueva cámara principal, la cual captura movimientos e incorpora una cámara time – of – flight (TOF) de alta resolución, lo que permite obtener mayor precisión y resolución, el nuevo método de profundidad proporcionado por la cámara TOF permite tener tres veces más fidelidad, además de esto permite un 60 % de campo de visión más grande que con el Kinect versión 1.0, lo cual permite tener un espacio de mayor visión.

Una de las ventajas de la segunda versión del dispositivo, es el nuevo sensor de infrarrojo, el cual permite reconocer a los usuarios y objetos en condiciones de muy poca iluminación en las cuales para el ojo humano no sea visible. Kinect es una muestra del potencial científico y desarrollo técnico que brinda Microsoft a través de su equipo de investigación denominado Microsoft Research, el cual ha sido parte activa del desarrollo del sensor versión 2.0. (Quishpe & Ulloa, 2014)



	Versión 2	Versión 1
Sensor Kinect		
Rango de profundidad	0.5m – 4,5m	0.4m – 4.5m
Cámara de Color	1920x1080 @ 30 FPS	640x480 @ 30 FPS
Cámara de profundidad	512x424	320x240
Rango de Visión Hori.	70 grados	57 grados
Rango de Visión Ver.	60 grados	43 grados
Motor de inclinación	NO	SI
Uniones de esqueleto	25	20
Esqueleto reconocido	6	2
Reconocimiento facial	SI	SI
Gestos de la mano	SI	NO
Estándar USB	3.0	2.0

Tabla 1. Diferencia entre versiones del sensor

Fuente: Autoras

1.3.1.2. Aplicaciones del Sensor Kinect

Las aplicaciones del sensor Kinect son diversas a favor de la salud, ciencia, educación y comodidad de los seres humanos, las mismas son de gran ayuda para el desarrollo de la humanidad. Las aplicaciones más resaltadas son:

- Rehabilitación a los pacientes con problemas motrices utilizando una interfaz natural basada en Kinect, mediante un programa de ejercicios de coordinación de las extremidades superiores. (Mendoza, Sabino, & Márquez, 2015)
- Diseño e implementación de un dispositivo para rehabilitación de rodilla con envío de datos por RF. (Cabezas Manzano & Inca Balseca, 2014)
- Aplicación informática interactiva basada en reconocimiento de gestos para ser usada en música terapias de educación especial. (Escamilla & Suaza, 2013)
- Desarrollo de un sistema de aprendizaje interactivo para el área del idioma inglés con el soporte del Kinect de Microsoft- caso práctico para niños de 6 a 8 años en el centro educativo Ilinizas. (Peñaherrera, 2014)
- Análisis de la herramienta Kinect como recurso publicitario generadora de experiencias que motivan los deseos de compra o adquisición del consumidor de la ciudad de Quito. (Goetschel, 2014)

1.3.2. Sistemas para discapacitados visuales

1.3.2.1. Funciones y tipos

Los sistemas de asistencia para discapacitados son diversos, los cuales pueden mejorar la calidad de vida de personas con deficiencia visual,

proporcionando un estilo de vida independiente en el entorno. A continuación se detalla algunos de los sistemas de asistencia para discapacitados visuales.

- a) **MexVox Programa lector de pantalla para invidentes:** Es un software de libre distribución, creado para ser de ayuda a las personas con discapacidad visual. MexVox se comunica con el usuario mediante un sistema de voz que ayuda a la persona invidente a lograr un alto grado de independencia tanto en el estudio como en el trabajo. MexVox no solamente lee lo que aparece escrito en el monitor, sino que establece un diálogo amigable, a través de los programas específicos e interfaces adaptativas. (López, 2014)



Figura 1. Lector de Pantalla para Invidentes.

Fuente: (López, 2014)

- b) **Implementación de un prototipo de gafas electrónicas para personas no videntes:** En este proyecto se diseña y se implementa una interfaz visual para personas no videntes utilizando sensores de proximidad, utilizando el entorno de arduino, dado de fácil comprensión y maneja un lenguaje de programación ampliamente usado. Se realizó las pruebas en personas no videntes, tanto con el bastón blanco, como el dispositivo para comparar la efectividad de cada uno en un mismo ambiente y así determinar las ventajas y desventajas del sistema determinado. (Molina & Llanga, 2015).



Figura 2. Prototipo de gafas electrónicas para personas no videntes

Fuente: (Molina & Llanga, 2015)

- c) **Anillo para leer texto:** El anillo, desarrollado por investigadores del MIT Media Laboratory, utiliza un algoritmo creado especialmente para reconocer las palabras, que pasan a un programa que las lee en voz alta. A medida que la persona mueve el dedo por la página el aparato emite señales –bien sonidos o vibraciones- para evitar que se cambie de renglón sin darse cuenta. En su estado de desarrollo actual, el anillo debe estar conectado a un ordenador que es el que realiza la interpretación y la lectura del texto, pero sus creadores ya están desarrollando una versión que podría ejecutarse en un teléfono móvil. (Barbuzano, 2015)

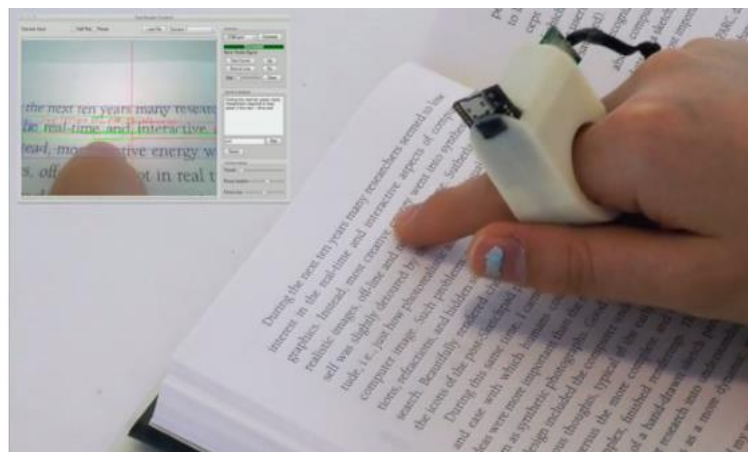


Figura 3. Anillo para leer texto

Fuente: (Barbuzano, 2015)

1.4. Discapacidad visual

1.4.1. Terminología Histórica.

Desde principios del Siglo XIX existe una falta de precisión en el empleo de términos referidos a personas con impedimentos visuales o ciegas. La inconsistencia en el uso por términos médicos, psicólogos y educadores puede ser características de actitudes profesionales o culturales, de diferentes conceptos entre las varias disciplinas y también de los roles divergentes que se asignan a cada disciplina. La lista de palabras que se incluye ilustra la variedad de términos utilizados durante los últimos cincuenta años para describir la disminución visual. (Ayala & Quito, 2012)

Ceguera Económica	Disminuido Visual
Ceguera Médica	Vidente Parcial
Ceguera Educacional	Defectuoso Parcial
Ceguera Congénita	Visión Residual
Ciego Adventicio	Limitado Visual
Ciego Legal	Visión Subnormal
Ciego Parcial	Impedido Visual
Ciego Vocacional	Incapacitado visual

1.4.2. Aspectos Generales

Cuando se habla de discapacidad visual, se está hablando de condiciones que limitan total o muy seriamente en el sentido de la vista de las personas. Las personas con discapacidad visual no pueden ver absolutamente nada, o en el mejor de los casos apoyados por servicios educativos, programas de rehabilitación visual, programas de estimulación visual y ayudas ópticas tienen menos grados de visión de lo normal. La discapacidad visual limita los desplazamientos de la vida

diaria, aprendizaje, el acceso y la participación del individuo en el mundo que lo rodea ya sea en la escuela, trabajo, entretenimiento, etc. (Bravo, 2014)

1.4.3. Características Físicas y Psicológicas de personas con discapacidad visual.

1.4.3.1. Características Psicomotrices en personas con discapacidad visual.

El desarrollo psicomotor es un proceso madurativo e intelectual, es la educación de movimiento de todo cuerpo movidos desde la parte neurofuncional y psíquica de todo ser humano, porque intervienen una serie de funciones mentales sensoriales que ayudan a las personas a tener un buen desarrollo psicomotriz.

Es importante y a la vez necesario reconocer que cualquier deficiencia en los seres vivos, ya sea de origen síquico o motriz, provoca alteraciones de locomoción, posición, coordinación, lo que manifiesta en la disminución de diferentes acciones motrices básicas como por ejemplo caminar, equilibrar, etc. A continuación se detalla las características psicomotrices que posee una persona con discapacidad visual. (Ayala & Quito, 2012)

- a) A mayor deficiencia o discapacidad visual, mayor dificultad en el aprendizaje y desarrollo psicomotor.
- b) El desarrollo psicomotor sigue las mismas fases pero a una velocidad más lenta en las personas con discapacidad visual.
- c) Los reflejos secundarios para caídas y diferentes apoyos puede tener un retraso en su aparición y esta en general debe ser estimulada.
- d) Dificultades en la manifestación de los motores básicos y las cualidades coordinativas.

1.4.4. Esquema Corporal en personas con discapacidad visual.

El Esquema Corporal: Es el conjunto de informaciones posturales, movimientos e impresiones visuales. En el niño ciego estas últimas fallan. Para Martínez y Núñez (1980), es la representación mental del propio cuerpo, de sus segmentos, de sus posibilidades de movimiento y de sus limitaciones espaciales. (Ayala & Quito, 2012)

Para autores como Picq y Vayer (1977) y Linares (1987), el esquema corporal está integrado por varios subconceptos psicomotores que tienen su propia identidad. Estos son:

- a) Conocimiento Corporal: es comprender que los distintos miembros del cuerpo pertenecen a un todo. Este se adquiere de forma evolutiva sobre los 3-5 años de edad.
- b) Conducta Respiratoria: según Martínez y Núñez (1980), la respiración está sometida a influencias corticales.
- c) Control Segmentario: concepto que permite evidenciar las posibilidades de independencia de los brazos en relación con el tronco, el control de los distintos segmentos y, además, el control emocional. Otros autores hablan de control tónico o tonicidad, pensando en que el músculo debe tener un tono necesario y adecuado para cada acción.

1.4.4.1. Características que presentan los ciegos y deficientes visuales

Las características presentadas por personas con este tipo de discapacidad son:

- a) Se retarda la toma de conciencia de su propio cuerpo por la ausencia de información visual exterior.
- b) Dificultades con la lateralidad, predominio funcional de un lado del cuerpo, (apoyar más una pierna que otra).
- c) Desajustes corporales, las personas ciegas adoptan ciertas posturas, las cuales les producen malformaciones, como pueden ser en muchos casos de columna.

1.4.5. Actividades para desarrollar la psicomotricidad en personas con discapacidad visual.

La Dra. Silvia Bravo Campoverde creó un programa de intervención motriz, a través de juegos para el desarrollo de la psicomotricidad en niños/as, es un recurso didáctico que facilita la intervención motriz.

Tomando como referencia este programa se creó una serie de actividades que ayudara a niños, niñas y adultos con discapacidad visual a desarrollar sus habilidades psicomotrices.

Actividad 1

Objetivo: Desarrollar el conocimiento corporal

Descripción de la actividad: Esta actividad estará dividida en dos partes:

Primera: Tocar la cabeza con la mano izquierda.

Segunda: Tocar su cintura con la mano derecha.

Actividad 2

Objetivo: Desarrollar coordinación de brazos

Descripción de la actividad: Levantar el brazo izquierdo al escuchar un número impar y levantar el brazo derecho al escuchar un número par.

Esta actividad estará dividida en dos ejercicios generados aleatoriamente:

Actividad 3

Objetivo: Desarrollar coordinación de piernas

Descripción de la actividad: Levantar la pierna correspondiente ya sea izquierda o derecha al escuchar las indicaciones proporcionadas por la máquina.

Actividad 4

Objetivo: Desarrollar locomoción.

Descripción de la actividad: Se realizará un conteo regresivo, el cual el usuario deberá desplazarse hacia la izquierda o derecha, cuando el conteo llegue a cero el usuario deberá unir sus pies.

Actividad 5

Objetivo: Identificar la posición

Descripción de la actividad: Detectar si el usuario está en la posición de parado o sentado.

1.5. Hardware

1.5.1. Sensor Kinect v2.0 para Windows

1.5.1.1. Definición

Kinect originalmente conocido por el nombre en clave «Project Natal», creado por Alex Kipman, desarrollado por Microsoft, y desde junio del 2011 para PC a través de Windows 7 y Windows 8 Kinect permite a los usuarios controlar e interactuar con el computador sin necesidad de tener contacto físico con un dispositivo de entrada; mediante una interfaz natural de usuario que reconoce gestos, comandos de voz, objetos e imágenes. (EcuRed, 2016)

1.5.1.2. Características

El sensor de Kinect es un equipo rectangular alargado, está diseñado para estar en posición horizontal. El dispositivo cuenta con una cámara RGB, un sensor de profundidad y cuatro micrófonos multi-array bidireccional, con lo que consigue capturar imágenes y movimientos de los cuerpos en 3D, además de ofrecer reconocimiento facial y aceptar comandos de voz. (Herrerros, 2013)



Figura 4. Sensor Kinect v2.0

Fuente: (Herrerros, 2013)

El sensor requiere un par de otros componentes para trabajar: el cubo y la fuente de alimentación, que acepta tres conexiones: el sensor USB 3.0, salida a PC, y el poder. La fuente de alimentación hace exactamente lo que su nombre implica: que suministra toda la potencia del sensor requiere para operar. Los cables de alimentación pueden variar según el país o la región, pero la propia fuente de alimentación compatible con voltajes de 100-240 voltios.

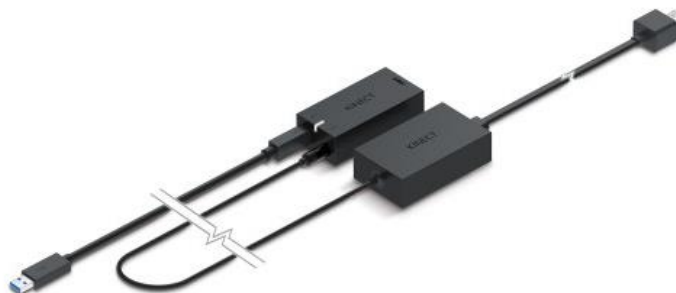


Figura 5. Adaptador Kinect para Windows

Fuente: (Herrerros, 2013)

Kinect para Windows es versátil. Se puede ver el movimiento de todo el cuerpo de las personas, así como pequeños gestos con las manos. Hasta seis personas pueden ser rastreadas en su conjunto. Como se muestra en la figura 6 el Sensor Kinect v2.0 para Windows tiene una cámara RGB (rojo-verde-azul) de vídeo a color, un emisor de infrarrojos y una cámara cuya profundidad es medida en resoluciones de milímetros, permite una amplia variedad de interacciones, pero no cualquier sensor tiene "puntos clave" y limitaciones. Teniendo esto en cuenta, se define sus objetivos y límites de la siguiente manera:

Límites físicos - las capacidades reales de lo que el sensor puede ver.

Puntos Clave - zonas donde las personas experimentan interacciones óptimas, dado que con frecuencia se tiene un rango de movimientos de sus brazos o piernas extendidas. (Corporation, 2014)



Figura 6. Componentes del Sensor Kinect v2.0

Fuente: (Corporation, 2014)

Para calcular las distancias entre un cuerpo y el sensor, el sensor emite un haz láser infrarrojo que proyecta un patrón de puntos sobre los cuerpos cuya distancia se determina. Una cámara infrarroja capta este patrón y por hardware calcula la profundidad de cada punto. El sensor de Kinect puede llegar a distinguir la profundidad de cada objeto con una resolución de 1 centímetro y las estimaciones de altura y anchura con una exactitud de 3 milímetros. (Herreros, 2013)

1.5.1.3. Especificaciones del Kinect

La figura 7 muestra las especificaciones del sensor Kinect v2.0 de Microsoft.

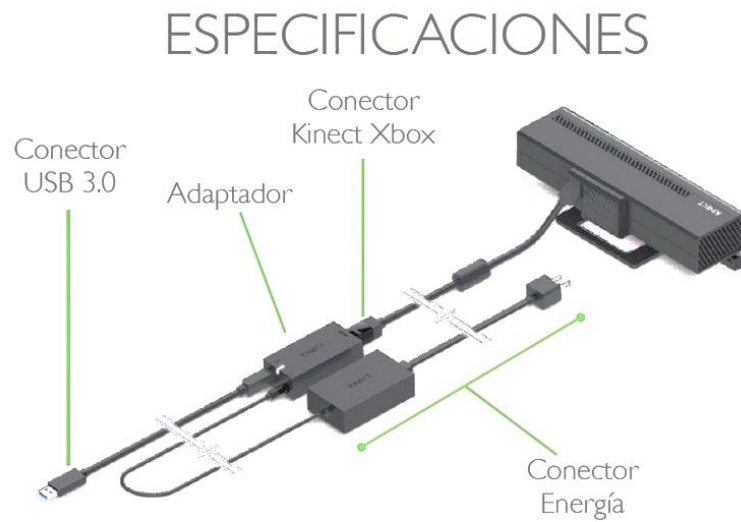


Figura 7. Especificaciones del Sensor Kinect v2.0

Fuente: Autoras

1.5.1.4. Consideraciones físicas del Sensor Kinect:

a) Cuerpo:

- Los límites físicos (Physical limits) : 0,5m a 4,5 m (defecto)
- Puntos de Precisión: 0,8 m a 3,5 m

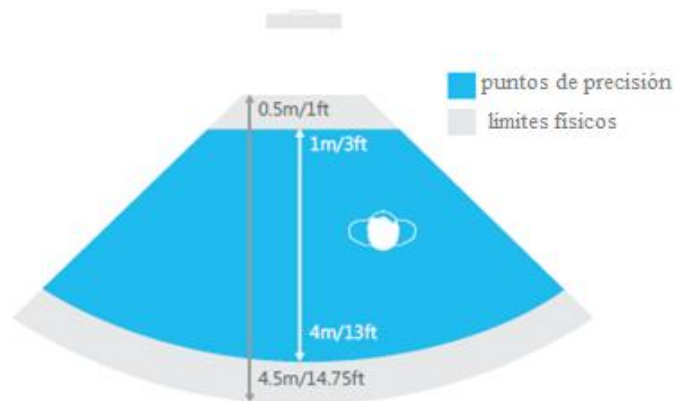


Figura 8. Rango de profundidad

Fuente: (Corporation, 2014)

b) El ángulo de visión (profundidad):

Horizontal: 70 grados

Vertical: 60 grados

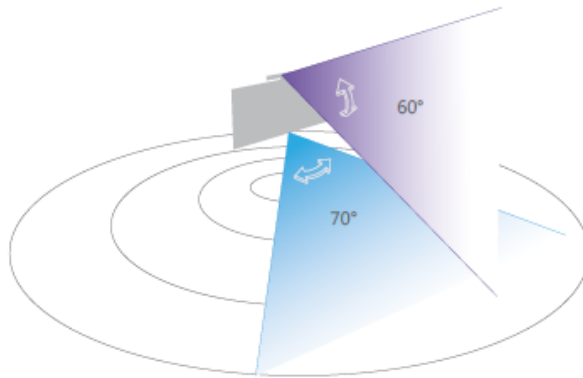


Figura 9. **Ángulo de Visión**

Fuente: (Corporation, 2014)

c) Seguimiento de Esqueletos:

Kinect v2.0 para Windows puede realizar un seguimiento de hasta seis personas dentro su punto de vista como esqueletos completos. Los esqueletos puede rastrearse si el usuario está de pie o sentado.

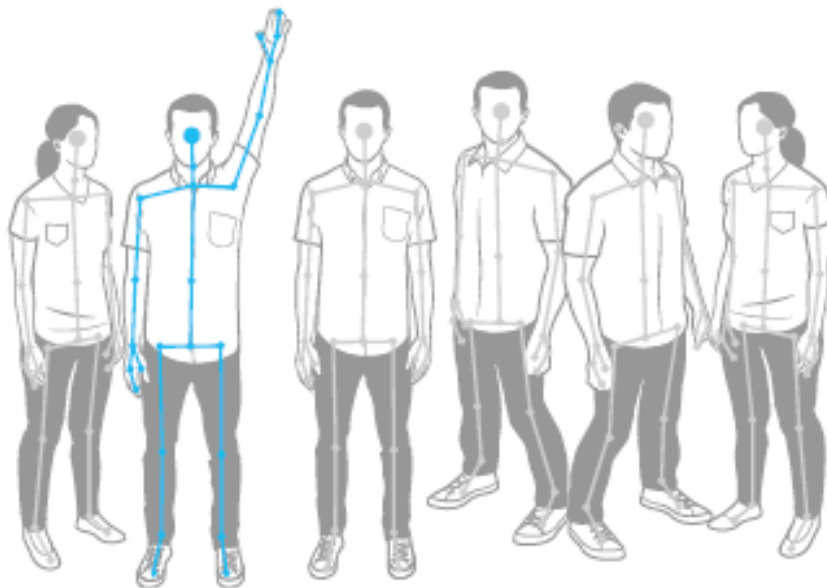


Figura 10. **Seguimiento de esqueletos**

Fuente: (Corporation, 2014)

d) Modelo del esqueleto completo:

Kinect para Windows puede realizar un seguimiento de los esqueletos en el modo de esqueleto completo por defecto con 25 articulaciones.



Figura 11. Modelo del esqueleto completo

Fuente: (Corporation, 2014)

e) Modo sentado.

Kinect para Windows también puede realizar un seguimiento de los esqueletos sentados con sólo las 10 articulaciones superiores.

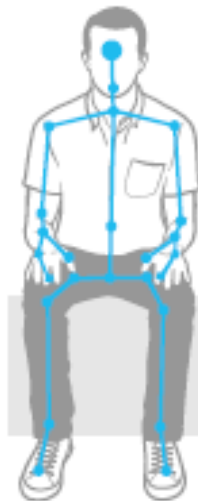


Figura 12. Modo Sentado

Fuente: (Corporation, 2014)

1.5.1.5. Consideraciones de audio y sonido del Sensor Kinect

Kinect para Windows es el único sensor que capta tanto la voz y el gesto, seguimiento de la cara y de pequeños movimientos de todo el cuerpo. El sensor tiene cuatro micrófonos que permiten su aplicación para responder a la entrada verbal, además de responder al movimiento.

a) Entrada de audio:

El Kinect detecta una entrada de audio de + y - 50 grados en la parte frontal del sensor.

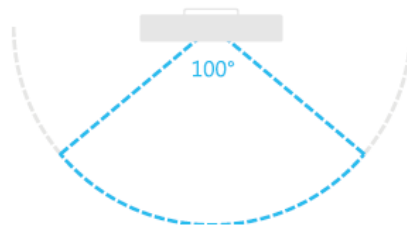


Figura 13. **Entrada de Audio**

Fuente: (Corporation, 2014)

b) Conjunto de Micrófonos:

El conjunto de micrófonos puede señalarse en incrementos de 5 grados en el rango de 180 grados. Esto puede ser usado para ser específico acerca de la dirección de los sonidos importantes, tales como una persona que habla, pero no va a eliminar por completo cualquier otro ruido ambiente.

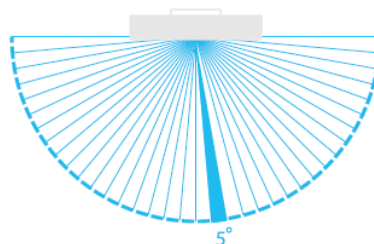


Figura 14. **Micrófonos**

Fuente: (Corporation, 2014)

c) Umbral de Sonido:

El conjunto de micrófonos puede cancelar 20 dB (decibelios) de ruido ambiente, lo que mejora la fidelidad de audio. Eso es aproximadamente el nivel de sonido de un susurro. (Kinect para Windows soporta la cancelación de sonido monofónico, pero no estereofónico.) El Sonido que viene desde detrás del sensor obtiene una supresión adicional de 6 dB basado en el diseño de la carcasa del micrófono.

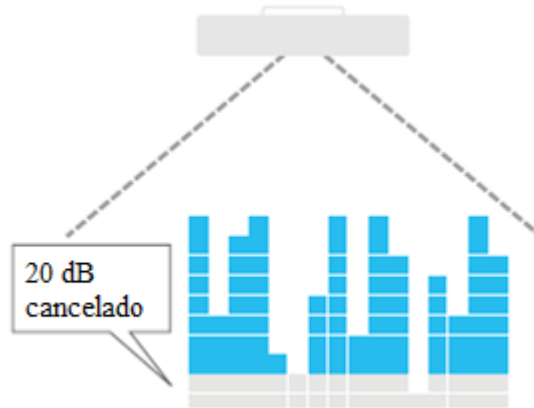


Figura 15. Umbral de sonido

Fuente: (Corporation, 2014)

d) Micrófono Direccional:

También puede dirigir mediante programación el conjunto de micrófonos - por ejemplo, hacia un lugar establecido, o después de un esqueleto como su seguimiento.

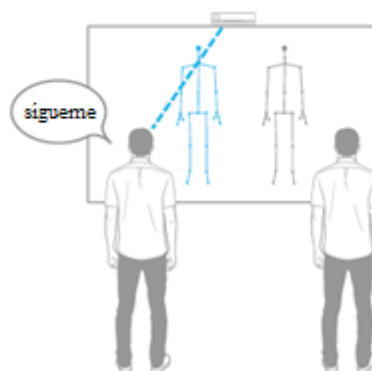


Figura 16. Micrófono direccional

Fuente: (Corporation, 2014)

e) Fuente más Fuerte de Localización

Por defecto, las pistas de Windows Kinect para la entrada de audio más fuerte.



Figura 17. Fuente de localización

Fuente: (Corporation, 2014)

1.5.1.6. Funcionamiento Básico

El funcionamiento básico de Kinect consta de tres partes diferenciadas. Por una parte, el reconocimiento de imágenes, por otro el reconocimiento de voz (Bunker, 2013)

a) Reconocimiento de imágenes

La configuración óptica de Kinect permite el reconocimiento de imágenes en tiempo real. La tecnología usada está disponible desde hace más de 15 años, por lo que no es altamente compleja. Microsoft, por su parte, ha conseguido algunos efectos y acciones que antes únicamente estaban disponibles a precios muy elevados. El sensor está compuesto de dos partes principales: un proyector y una cámara de infrarrojos VGA. La profundidad de los objetos es captada por la cámara gracias al rebote de los haces laser por el campo de juego, creando así un “campo de profundidad” que permite al sensor diferenciar entre los objetos estáticos de la sala y las personas utilizándolo. Este “campo de profundidad” consiste, básicamente, en que Kinect recibe este haz de luz como infrarrojos que

varían en mayor o menor grado de color dependiendo de la distancia a la que se encuentran del sistema. De este modo, los cuerpos aparecen como rojo, verde, etc, y los objetos más alejados aparecen en gris.

Con los datos obtenidos en esta imagen, el software aplica una serie de filtros para que Kinect pueda saber qué es una persona y qué no, basándose en una serie de directrices como “una persona tiene dos brazos y dos piernas”.

Una vez que se tiene la información ordenada, se identifican las partes del cuerpo y crea un esqueleto en movimiento. Kinect tiene unas 200 posturas precargadas, de manera que se puedan llenar los espacios en blanco en caso de que se realicen movimientos que obstruyan la visión de su esqueleto. El principal inconveniente que se puede encontrar, es que los dedos no se asignan de forma individual en el esqueleto, impidiendo con ello una serie de movimientos. (Fernandez, 2014)

b) Reconocimiento de voz

El principal problema del reconocimiento de voz en Kinect, es que tiene que ser sensible a las voces hasta 5 metros de distancia, además de que debe ser capaz de ignorar los ruidos ambientales y cualquier otro sonido de la propia voz. Para poder realizar esto, el equipo de Microsoft realizó una serie de pruebas en diferentes viviendas con 16 micrófonos. En estas pruebas se tomaron una serie de grabaciones de diferentes configuraciones, de forma que pudieran conseguir el mejor posicionamiento de los micrófonos en el sensor.

El resultado de estas pruebas proporcionó la configuración actual de los micrófonos en Kinect. Se trata de un array de cuatro micrófonos posicionados boca abajo, uno a la izquierda y tres en la parte derecha.

Este conjunto de micrófonos recoge las voces de la mejor manera posible. Aun así, necesita ayuda del software de la cámara. La unidad de procesamiento cancela el ruido, mientras que un software usa la cámara para calcular de donde proviene el sonido, de modo que crea una burbuja alrededor del usuario y consigue separar

la voz del usuario, omitiendo la del resto de personas que se encuentren alrededor. (Fernandez, 2014)

1.6. Ordenador

EL ordenador es una de las piezas más importantes y fundamentales para el desarrollo de aplicaciones con el sensor Kinect de Microsoft, el computador es el encargado de obtener el máximo provecho a través de los paquetes Windows: Software Development Kit (SDK) y las licencias para la elaboración del proyecto los cuales son compatibles con las aplicaciones generadas C++, o Visual Basic utilizando Visual Studio 2015.

Los requerimientos para el ordenador son: procesador de 64 bits de doble núcleo, USB 3.0 y 6GB de memoria RAM.

1.7. Software

1.7.1. Visual Studio

1.7.1.1. Introducción

Según Herbert Schildt Visual Studio es el ambiente de desarrollo integrado de Microsoft. Permite editar, compilar, ejecutar y depurar un programa, todo ello sin tener que abandonar el bien desarrollado ambiente. Visual Studio ofrece un medio conveniente y de gran ayuda para administrar los programas. Es mucho más efectivo para programas grandes, pero puede ser utilizado con gran éxito en programas pequeños.

Una característica importante de Visual Studio es la posibilidad de crear aplicaciones para versiones específicas de la plataforma. NET. Esta característica se denomina multitargeting y permite integrar la funcionalidad de múltiples diseñadores visuales. Visual Studio 2013 puede ser descargado a través del

siguiente link: <https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx>.

Microsoft ha estado prometiéndolo desde hace años que ofrecería un único entorno de desarrollo integrado (IDE) para todos sus lenguajes de programación. Ahora, este entorno está ya disponible y se incluye en Visual Studio.NET. Podrá utilizar Visual Studio.NET para desarrollar aplicaciones basadas en Visual Basic, y C#. Incluso, aún más interesante, podrá cargar decenas de otros tipos de archivos y utilizar el IDE como editor de recursos, un editor de imagen, un explorador HTML, etc. (Balena, 2003)

1.7.1.2. Plataforma .NET

Microsoft.NET extiende las ideas de Internet y sistema operativo haciendo del propio internet la base de un nuevo sistema operativo. En última instancia, esto permitirá a los desarrolladores crear programas que trasciendan los límites de los dispositivos y aprovechen por completo la conectividad de internet y sus aplicaciones.

Para ello proporciona una plataforma que incluye los siguientes componentes básicos:

Herramientas de programación para crear servicios Web XML con soporte multilingüe: Visual Studio.NET y .NET Framework (Ceballos, 2010)

a) Visual Studio.NET

Visual Studio .NET es un conjunto de herramientas orientadas al desarrollo de aplicaciones informáticas. Se puede construir aplicaciones de escritorio, para la Web o para dispositivos móviles, todas de gran escalabilidad y versatilidad. Así mismo, se pueden utilizar sus herramientas de diseño para desarrollar e implementar poderosas aplicaciones de negocios. (Gómez Jiménez, 2010)

b) **.NET Framework**

El Framework o marco de trabajo constituye la base de la plataforma .NET y denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entorno de ejecución distribuido. El Microsoft .NET Framework es un componente que provee soluciones pre codificadas para requerimientos comunes de los programas.

Las soluciones pre codificado que forman la biblioteca .NET, cubren un gran rango de necesidades de la programación de programas. El Framework incluye soluciones en áreas como: la interfaz de usuario, acceso a datos, conectividad a base de datos, criptografía, desarrollo de aplicaciones web, algoritmos numéricos y comunicación de redes. (Sánchez Flores, 2011).

El Framework nos provee la implementación de cuatro lenguajes compatibles con CLS (especificación de lenguaje común), junto con sus compiladores:

Microsoft Visual Basic .NET

Microsoft Visual C# .NET

Microsoft Visual F# .NET

Microsoft Visual C++ .NET

1.7.1.3. Microsoft Visual C#

Visual C# es un lenguaje orientado a objetos, sencillo pero eficaz y rápido, que permite que los programadores creen en gran variedad aplicaciones de diferentes tipos (móviles, Windows, web, etc.). Junto a .NET Framework, Visual C# permite el trabajo en todas la herramientas propias de ID que ofrece Visual Studio.

C# está directamente emparentado con C, C++ y Java, lo cual no es accidental. Estos son tres de los dos lenguajes de cómputo más utilizados en el mundo, C#

incluye características que soportan directamente elementos que constituyen de los componentes, como propiedades, métodos y eventos, sin embargo la capacidad de C# para trabajar en un ambiente seguro de lenguajes cruzados, es quizá su característica más importante en la orientación a componentes. (Schirt, 2010).

Para crear una aplicación en el entorno C#, primero se creará un proyecto de Windows Presentation Foundation (WPF) y una solución. Los pasos que se deben seguir son los siguientes:

1. Como se muestra en la figura 18 para crear un nuevo proyecto es necesario ubicarse en la barra de menús seleccionar File, New, Project.

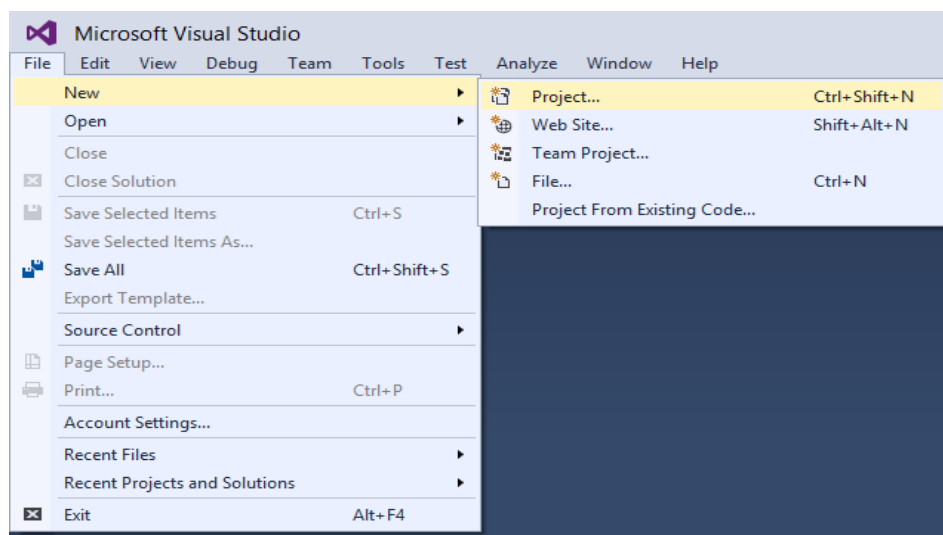


Figura 18. Crear un nuevo proyecto en C#

Fuente: Autoras

2. Se desglosara una ventana como se muestra en la figura 19. Seleccionar Installed, Templates, Visual C#, Windows, y elegir WPF Aplicación en el panel central. Asignar al proyecto el nombre KinectMove en la parte inferior del cuadro de diálogo New Project, asignar la ubicación del proyecto y presionar OK

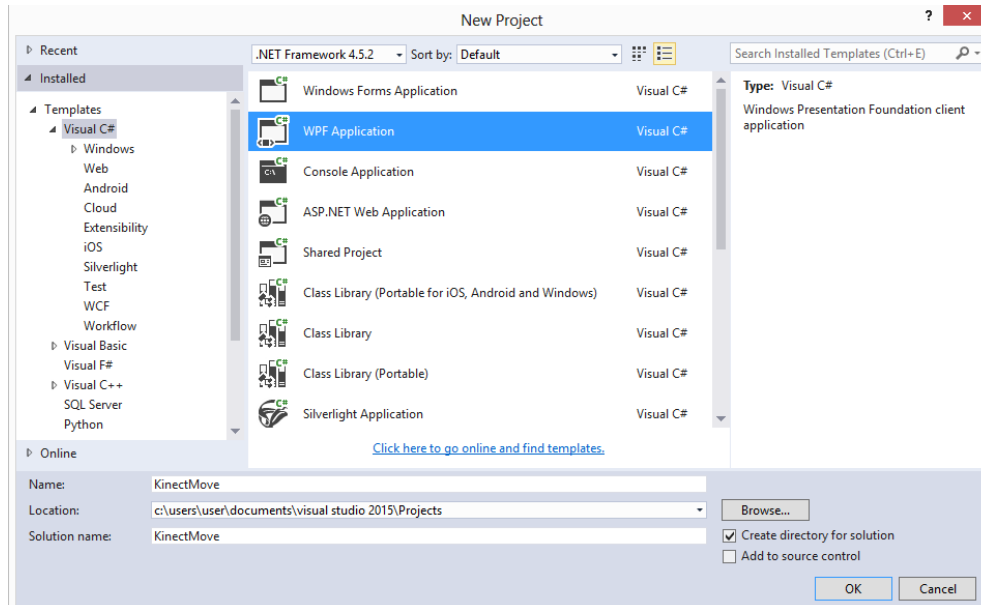


Figura 19. Crear un nuevo proyecto en C#

Fuente: Autoras

3. En la figura 20 se puede observar que el entorno genera automáticamente el esqueleto del programa

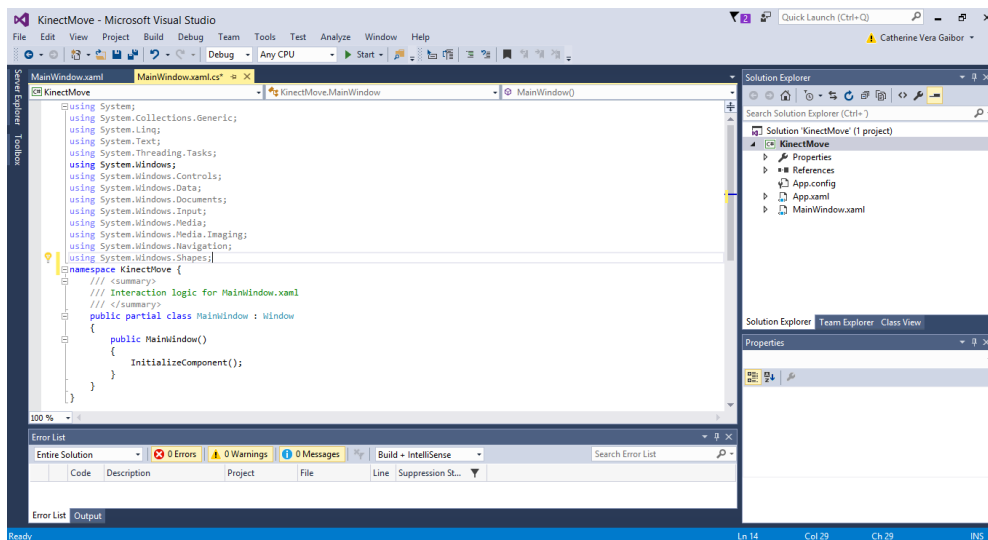


Figura 20. Crear un nuevo proyecto en C#

Fuente: Autoras

1.7.2. Software Development Kit (SDK v2.0)

El SDK proporciona herramientas y APIs, tanto nativas y administradas que se necesita para desarrollar aplicaciones compatibles con Kinect para Microsoft Windows. (Microsoft , 2016)

Con el link indicado a continuación se puede acceder al SDK v2.0: <https://www.microsoft.com/en-us/download/details.aspx?id=19988>.

El desarrollo de los controladores de Kinect para Windows es igual a muchas aplicaciones y dispositivos, el SDK de Kinect entrega todo el soporte para los controles de las características del Sensor, así como también librerías para los lenguajes compatibles de programación. (Quishpe & Ulloa, 2014)

Para poder hacer uso del SDK de Kinect es necesario instalar su librería en el entorno de Visual Studio, para ello se debe realizar lo siguiente:

1. Como se muestra en la figura 21 se debe dirigir a la barra de menú y seleccionar Project – Add Reference, la cual permitirá agregar una referencia de servicio:

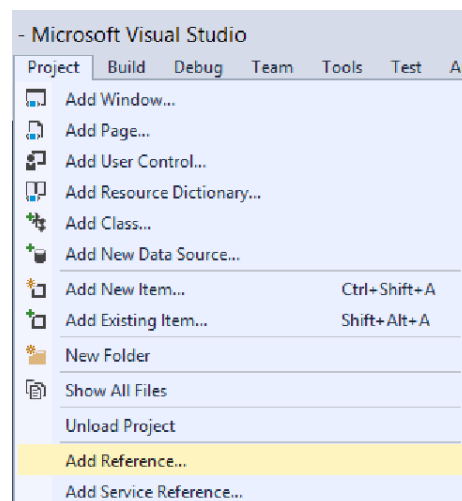


Figura 21. Agregar Referencia Microsoft.Kinect

Fuente: Autoras

- Elegir en el panel izquierdo **Assemblies, Extensions** y activar la opción **Microsoft.Kinect** en el panel central y dar clic en el botón ok para finalizar la instalación, cómo se muestra en la figura 22:

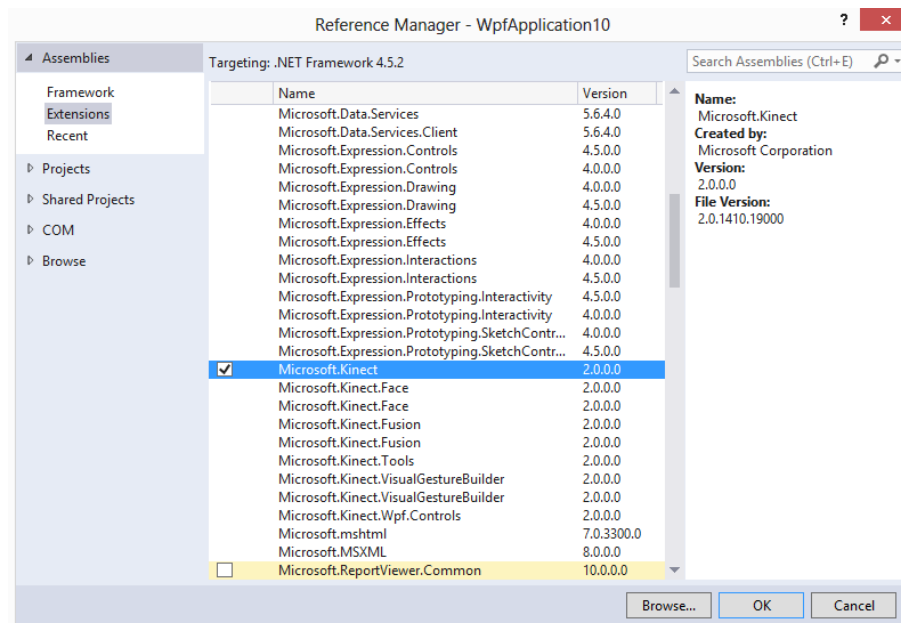


Figura 22. **Agregar Referencia Microsoft.Kinect**

Fuente: **Autoras**

Con la referencia agregada se añade el espacio de nombres junto con los demás usings del fichero para poder utilizar las clases y métodos del SDK.

1.7.2.1. Skeletal Tracking

Es la función más prometedora de Kinect, porque permite detectar la figura humana (esqueleto) cuando se está moviendo, estos movimientos son asociados a través de puntos o Joints los cuales grafican el esqueleto humano. Para la creación del cuerpo Kinect coge las siguientes 25 articulaciones.

Todo esto se logra debido a que el Skeletal Tracking tiene almacenado los datos de la imagen a color y los datos de la cámara de profundidad, lo que permite reconocer entre objetos y seres humanos ya que asocia los parámetros del cuerpo humano como extremidades superiores , articulaciones e incluso gestos ,para que de esta forma el cuerpo humano sea reconocido y detecte el movimiento de todas

las partes que conforman el esqueleto humano (brazos, manos , muñecas , brazos rodillas codos , cadera , etc. .)



Figura 23. Joints del cuerpo humano

Fuente: **Reto SDK de Kinect**

1.7.3. Speech Recognition.

El Kinect también puede trabajar como entrada de audio, lo cual permite reconocer comandos de voz dados por el usuario. El diccionario de la herramienta speech permite realizar el reconocimiento de palabras y frases en diferentes idiomas como: español, francés, japonés, y el original en inglés. (Quishpe & Ulloa, 2014)

Para que Kinect realice el reconocimiento de voz es necesario instalar diferentes librerías para obtener un diccionario de reconocimiento de palabras las cuales están en diferentes idiomas, las librerías y paquetes para el reconocimiento de voz son:

- **Kinect to Windows Runtime Language Pack**

Kinect to Windows Runtime Language Pack es el paquete que permite el reconocimiento del idioma con el cual se desee trabajar en el proyecto. Su descarga se la realiza a través del siguiente link: <https://www.microsoft.com/en-us/download/details.aspx?id=34809>.

Una vez obtenido este paquete basta con instalarlo y se puede acceder a sus librerías las cuales posteriormente deberán ser declaradas por el programador de acuerdo a sus requerimientos.

- **Microsoft Speech Platform**

La plataforma Microsoft Speech consiste en un tiempo de ejecución de aplicaciones que proporciona la funcionalidad de voz, una Application Programming Interface (API) para gestionar el tiempo de ejecución de Idiomas que permiten el reconocimiento de voz y síntesis de voz (text- to-speech o TTS) en un determinado idioma .

El uso de esta plataforma de voz de Microsoft, puede añadir la funcionalidad de reconocimiento de voz y de texto a voz (TTS) para mejorar la interacción de los usuarios con sus aplicaciones. Microsoft Speech puede ser descargado del siguiente link: <https://www.microsoft.com/en-us/download/details.aspx?id=27225>

Una vez instalado este paquete, para acceder a sus librerías también es necesario instalarlas en el entorno de trabajo del nuevo proyecto creado en C#.

Para esta instalación en la barra de menú dirigirse a **Project** la cual desplegará la opción **Add Reference** como se muestra en la sección 1.7.2, una vez hecho eso seguir los pasos indicados a continuación:

1. Como se observa en la figura 24 se debe legir en el panel izquierdo **Assemblies, Framework** y activar la opción System.Speech en el panel central.

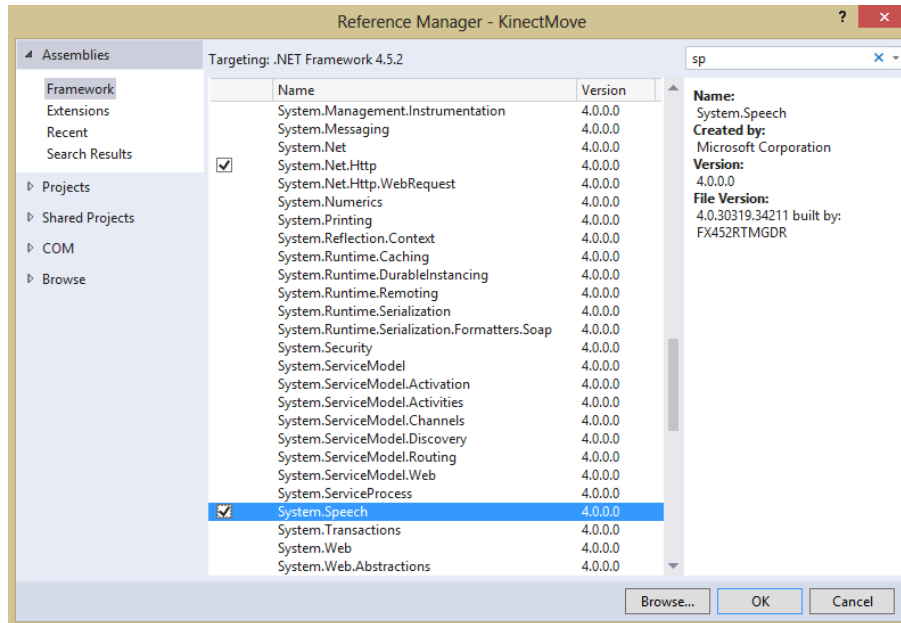


Figura 24. Agregar Librería Speech.Platform

Fuente: Autoras

- Al seleccionar la opción **Browse**, se desplegará la ventana mostrada en la figura 25, en la que se visualiza la librería que se debe añadir:

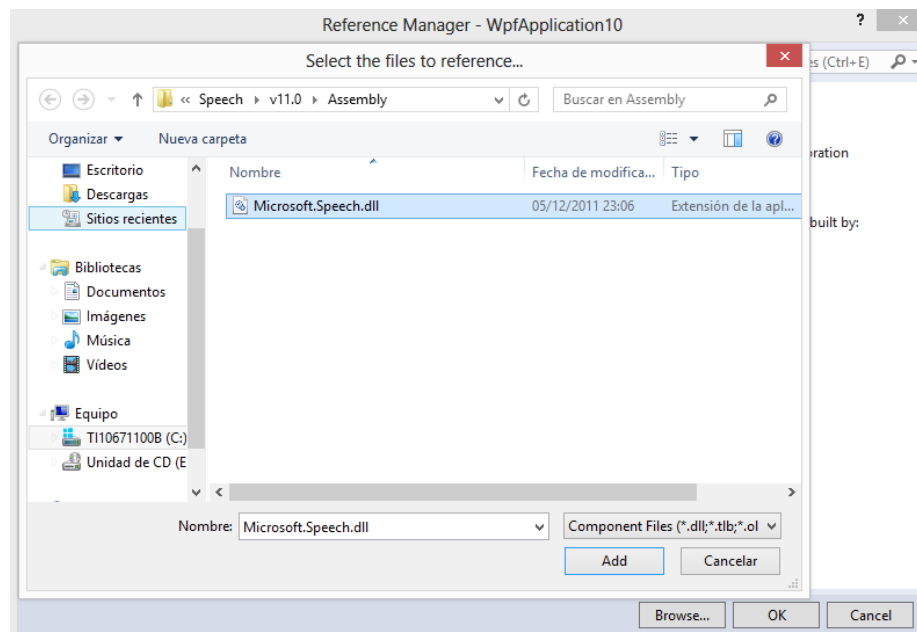


Figura 25. Agregar Librería Speech.Platform

Fuente: Autoras

1.7.4. Base de Datos

Su propio nombre sugiere que una base de datos es un lugar en el que es posible alojar información. Una base de datos puede contener mucho más que datos, configurándose en realidad en un objeto vital para la correcta gestión de la información. Una base de datos es el objeto lógico que sirve para el almacenamiento y recuperación de los datos desde un dispositivo, pero también debe asegurar la integridad de dichos datos basándose en reglas previamente definidas. La base de datos debe auto describirse, de tal forma que la información que contiene puede ser correctamente interpretada. En el caso particular de las bases de datos de SQL Server 2008 estas también almacenan funciones de usuario y permisos que permiten y describen los privilegios que tiene los usuarios sobre la información, así como módulos con código. NET que pueden llevar a cabo tareas completamente complejas. (Charte, 2009)

Una base de datos de Microsoft SQL Server 2008 está formada por un conjunto de tablas en las que se almacena datos estructurados. Una tabla contiene una colección de filas y columnas. Cada columna almacena un determinado tipo de información (números, fechas, nombres, etc.). Las tablas contienen diversos tipos de controles que garantizan la validez de los datos. En la creación de una nueva base de datos es muy necesaria la configuración de la estructura de almacenamiento de los datos. La arquitectura de almacenamiento de SQL Server 2008 distribuye la información de la base de datos en diversos archivos de datos y registro de transacciones. (Pérez, 2010)

1.7.4.1. SQL Server 2008

SQL Server 2008 es un elemento fundamental de la Plataforma de Datos de Microsoft, capaz de gestionar cualquier tipo de datos, en cualquier sitio y en cualquier momento. Le permite almacenar datos de documentos estructurados, semiestructurados o no estructurados como son las imágenes, música y archivos directamente dentro de la base de datos. SQL Server 2008 le ayuda a obtener más

rendimiento de los datos, poniendo a su disposición una amplia gama de servicios integrados como son consultas, búsquedas, sincronizaciones, informes y análisis. Sus datos pueden almacenarse y recuperarse desde sus servidores más potentes del Data Center hasta los desktops y dispositivos móviles, permitiéndole tener un mayor control sobre la información sin importar dónde se almacena físicamente.

SQL Server 2008 le permite utilizar sus datos en aplicaciones a medida desarrolladas con Microsoft® .NET y Visual Studio y también desde su propia Arquitectura Orientada a Servicio (SOA) y los procesos empresariales empleando Microsoft® BizTalk® Server. Además, las personas que gestionan la información pueden acceder directamente a los datos con las herramientas que utilizan habitualmente como Microsoft® Office 2007. SQL Server 2008 le ofrece una plataforma de datos, fiable, productiva e inteligente para cubrir todas sus necesidades. (Microsoft, 2009)

1.7.4.2. Crystal Reports

Crystal Reports ha formado parte de Visual Studio desde 1993, y ahora es el estándar de elaboración de informes de Visual Studio. Se incluye en todas las copias de Visual Studio Professional y se integra directamente en el entorno de desarrollo. Crystal Reports para Visual Studio incorpora la posibilidad de crear contenido interactivo con calidad de presentación al entorno de Windows.

Con Crystal Reports para Visual Studio, puede crear informes complejos y profesionales en un programa basado en GUI. Después puede conectar el informe a casi todos los orígenes de base de datos, así como a datos proxy, como un conjunto de resultados (por ejemplo, un ADO.NET DataSet). Puede almacenar el informe en una aplicación Web o para Windows, con uno de los controles de visores de Crystal Reports para Visual Studio. La presentación de informes, tanto en clientes Windows como en HTML 3.2 ó 4.0, es muy interactiva y proporciona funciones como la profundización en gráficos, la exploración de informes y la búsqueda de texto. (Microsoft, Developer Network)

CAPITULO II

2. METODOLOGÍA

2.1. Tipo de estudio

Descriptivo – Aplicativo.- Con esta investigación se describe el problema partiendo de las características fundamentales de las personas con discapacidad visual y al mismo tiempo se requiere implementar un sistema que permita la participación directa entre usuario y aplicación.

2.2. Métodos, técnicas e instrumentos

2.2.1. Métodos

2.2.1.1. Analítico

Con esta metodología se puede entender en forma particular el funcionamiento de cada componente necesario para el desarrollo del sistema, además del software que manejará el mismo, y la forma de interactuar con sus componentes entre sí, para su correcto funcionamiento.

2.2.1.2. Sintético

Este método permitirá considerar las cualidades de los elementos y ayudará a buscar la mejor alternativa para una adecuada interacción entre Hardware y Software, con la finalidad de obtener como resultado un sistema eficiente.

2.2.2. Técnicas

2.2.2.1. Observación

Esta técnica consiste en la recolección de información que sea de apoyo para el desarrollo del proyecto, dando las pautas necesarias para el diseño e implementación del sistema interactivo mediante tecnología Kinect v2.0 para desarrollar las habilidades psicomotrices en personas con discapacidad visual.

2.2.3. Instrumentos

Los instrumentos necesarios son libros, folletos, archivos, páginas web, blogs, cursos, manuales, que son útiles para el diseño e implementación de este sistema.

2.3. Población y muestra

La población corresponde al total de ciclos que conforman el sistema, en los que se va a realizar pruebas de las actividades diseñadas con diferentes individuos que tienen discapacidad visual. Estas actividades serán las siguientes:

- Actividades de locomoción
- Actividades de posiciones
- Actividades para coordinación de piernas
- Actividades para coordinación de brazos
- Actividades de conocimiento corporal

Para determinar el tamaño de la muestra se aplica la siguiente ecuación:

$$n = \frac{Npq}{(N - 1) \frac{ME^2}{NC^2} + pq}$$

Dónde:

n = Tamaño de la muestra.

N = Tamaño del universo.

p = Probabilidad de ocurrencia (homogeneidad del fenómeno, porcentaje de respuestas fiables o confiables, generalmente $p = 0.5$).

q = Probabilidad de no ocurrencia $q = 1-p = 0.5$.

ME = Margen de error o precisión admisible con que se toma la muestra (generalmente se elige del 0,01 al 0.15), el más usual es 0.05.

NC = Nivel de confianza o exactitud con que se generaliza los resultados a la población Una forma de plantear ME y NC es, en porcentaje $ME+NC = 100\%$, es decir:

ME = 15% = 0.15; o sea el 85 % de confianza, **NC** = 1.44

ME = 10% = 0.10; o sea el 90 % de confianza, **NC** = 1.64

ME = 5% = 0.05; o sea el 95 % de confianza, **NC** = 1.96

ME = 1% = 0.01; o sea el 99 % de confianza, **NC** = 2.57 Para este estudio se ha escogido un margen de error de 5% (**ME=0.05**), con un nivel de confianza del 95% (**NC=1.96**).

Entonces:

N=6

p=0.5

q=0.5

ME=0.05

NC=1.96

$$n = \frac{(6)(0.5)(0.5)}{(6-1) \frac{(0.05^2)}{(1.96^2)} + (0.5)(0.5)} = 6$$

Al aplicar esta fórmula tenemos como resultado que la muestra es igual a la población.

2.4. Hipótesis

El diseño e implementación de un sistema interactivo mediante Tecnología Kinect v2.0 permitirá comprobar el funcionamiento de las actividades que desarrollen las personas con discapacidad visual.

2.5. Operacionalización de variables

Variables	Concepto	Dimensiones	Indicadores	Técnicas
Independiente: Sistema interactivo.	Un sistema interactivo es un sistema informático que se interrelaciona y depende de las acciones de un usuario para realizar una tarea, es decir, todo sistema en el que interactúan persona y máquina.	Aplicación con Kinect v2.0	Datos de movimientos Reconocimiento de extremidades.	Simulación para recolección de datos Observación
		Indicaciones	Audios.	Recolección de datos
		Visual Studio	Lenguaje de Programación	Software libre
Dependiente: Desarrollar habilidades psicomotrices	La psicomotricidad es una disciplina que, basándose en una concepción integral del sujeto, se ocupa de la interacción que se establece entre el conocimiento, la emoción, el movimiento y de su mayor validez para el desarrollo de la persona, de su corporeidad, así como de su capacidad para expresarse y relacionarse en el mundo que lo envuelve.	Actividades para desarrollo psicomotor	Actividades para desarrollar locomoción. Actividades para identificar la posición. Actividades para desarrollar coordinación de piernas. Actividades para desarrollar coordinación de brazos. Actividades para desarrollar conocimiento corporal.	Observación Evaluación Test
		Registro de datos.	Administración de la información	Evaluación

Tabla 2. Operacionalización de Variables

Fuente: Autoras

2.6. Procedimientos

Para el desarrollo de este sistema interactivo mediante tecnología Kinect v2.0, se requirió seguir una serie de pasos y etapas, que en conjunto, cumplirían con el objetivo planteado.

Cada etapa desempeña una función específica dentro del esquema global, y también sirve de apoyo para la siguiente etapa, haciendo de la aplicación implementada un sistema funcional.

A continuación se muestra el procedimiento que se realizará en el sistema.

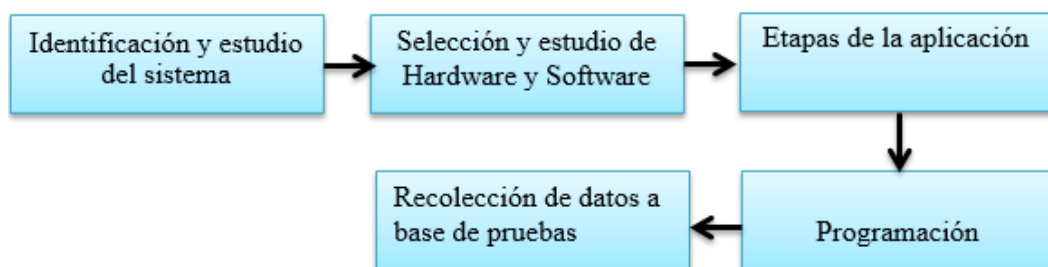


Figura 26. **Diagrama de procedimientos**

Fuente: Autoras

2.6.1. Identificación y estudio del sistema.

En la actualidad no existe un sistema que ayude a individuos con discapacidad visual a desarrollar satisfactoriamente sus habilidades psicomotrices. Las personas con discapacidad visual generalmente, no utilizan las estrategias adecuadas que les permitan dominar su cuerpo, ya que difícilmente pueden ejecutar actividades como: conocimiento corporal, locomoción, posición, coordinación de piernas y coordinación de brazos.

2.6.2. Selección y estudio de Hardware y Software

Se eligió utilizar un sensor de reconocimiento Kinect v2.0, puesto que es un dispositivo que permite el seguimiento del esqueleto humano en diferentes

posiciones, además de tener un costo accesible para su adquisición. Esto en contraste con el sensor Kinect v1.0 que también se estudió como una posible opción ya que su principal ventaja es que tiene un motor de inclinación, mientras que el sensor Kinect v2.0 no cuenta con uno y su inclinación se la debe realizar manualmente. Al final quedó descartado por su elevado costo y su muy limitado ángulo de inclinación. Además que este no puede reconocer el esqueleto en ambientes oscuros.

En lo que tiene que ver con el Software para el sistema se debe buscar un lenguaje de programación que brinde ventajas como la compatibilidad con el hardware escogido, que su manejo y complejidad no se dificultosa, es por ello que se optó por utilizar el lenguaje de programación Visual Studio con el compilador C#, para la integración de todos los elementos del sistema, por ser un software de distribución libre y de fácil acceso.

Es necesario la integración de referencias, clases y métodos para la comunicación con el sensor Kinect v2.0. Junto con la inclusión de las librerías, se diseñó el código para que el usuario pueda elegir las actividades que desea realizar, una vez diseñadas las actividades se procedió a añadir las instrucciones para que las personas las puedan realizar de manera correcta mediante señales de audio.

Para trabajar con las imágenes que representa el esqueleto del cuerpo humano se utiliza el SDK de Kinect permitiendo obtener los Joints, en coordenadas x-y-z, obteniendo uno a uno los datos de cada Joint que nos presenta el Kinect y mediante restas entre las posiciones de cada articulación, se hace cálculos para obtener la posición final que se requiere.

En lo referente al almacenamiento de datos se lo hace utilizando el software SQL Server 2008. Este software da la ventaja de almacenamiento y fácil manipulación de datos de una manera sencilla y rápida.

2.6.3. Etapas de la Aplicación

Como se muestra en la figura 27 el sistema interactivo está constituido principalmente por tres etapas, interconectados entre sí, como son la interfaz entre usuario y máquina que permite al usuario interactuar con la aplicación, el servidor de base de datos que almacenan la información y el reporte alojado en el servidor web para que los clientes accedan a la información.

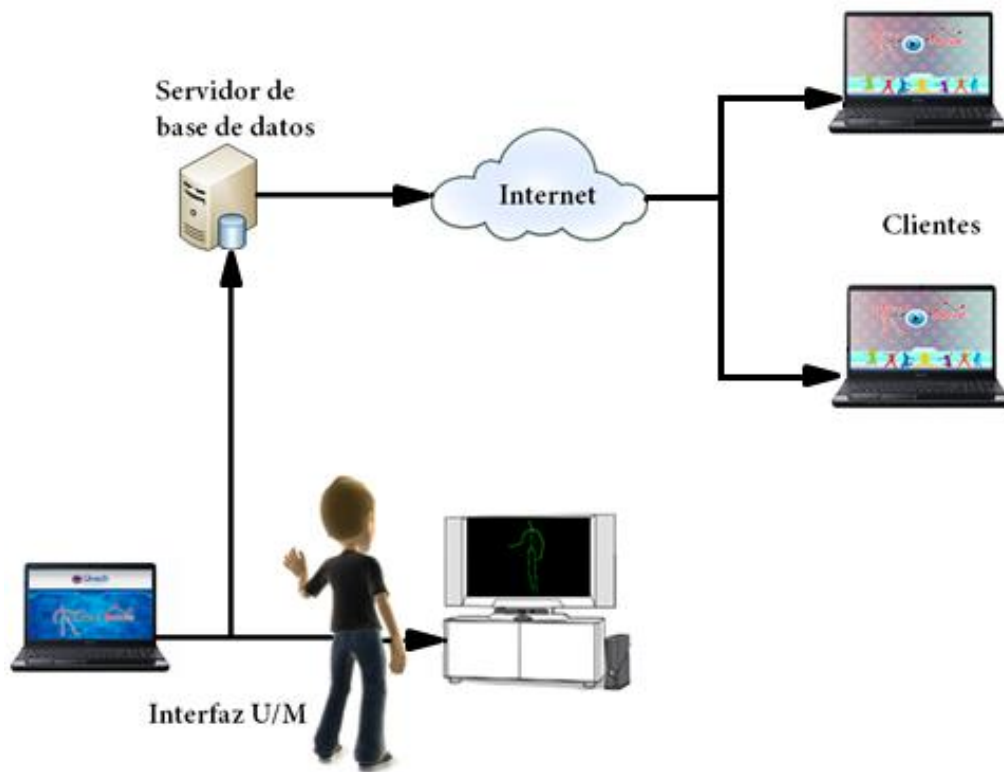


Figura 27. Diagrama esquemático del sistema interactivo

Fuente: Autoras

2.6.3.1. Interfaz entre usuario y máquina

Una vez que el sensor Kinect v2.0 detecte al usuario dará inicio la aplicación, en la cual la interfaz es reflejada mediante comandos de voz y señales de audio. Las señales de audio permitirán emitir instrucciones al usuario, para que este

escoja por medio de comandos de voz una opción o alternativa para el desarrollo de la aplicación.

KinectMove mediante comandos de audio dará un saludo de bienvenida a la aplicación, permitiendo así que el usuario ingrese a realizar las actividades o a la vez abandonar la misma.

El usuario deberá registrar sus datos como son nombre, edad y sexo antes de realizar cualquier actividad permitiéndole elegir y realizar 5 actividades a través de las instrucciones emitidas por el sensor, o la vez salir de la aplicación.

2.6.3.2. Servidor de datos

Se creara un registro de toda la información de los usuarios que hagan uso de la aplicación así como nombre, edad, sexo y aciertos o errores de las actividades ejecutadas.

2.6.3.3. Clientes

Los clientes o personas que requieran la información almacenada de la aplicación, la podrán revisar a través de un dominio.

2.6.4. Programación

El código de la programación del proyecto se lo detalla más adelante (ver Anexo 1) y se lo hace en bloques, como se muestra en el flujograma.

2.6.4.1. Diagrama de Flujo de la Programación

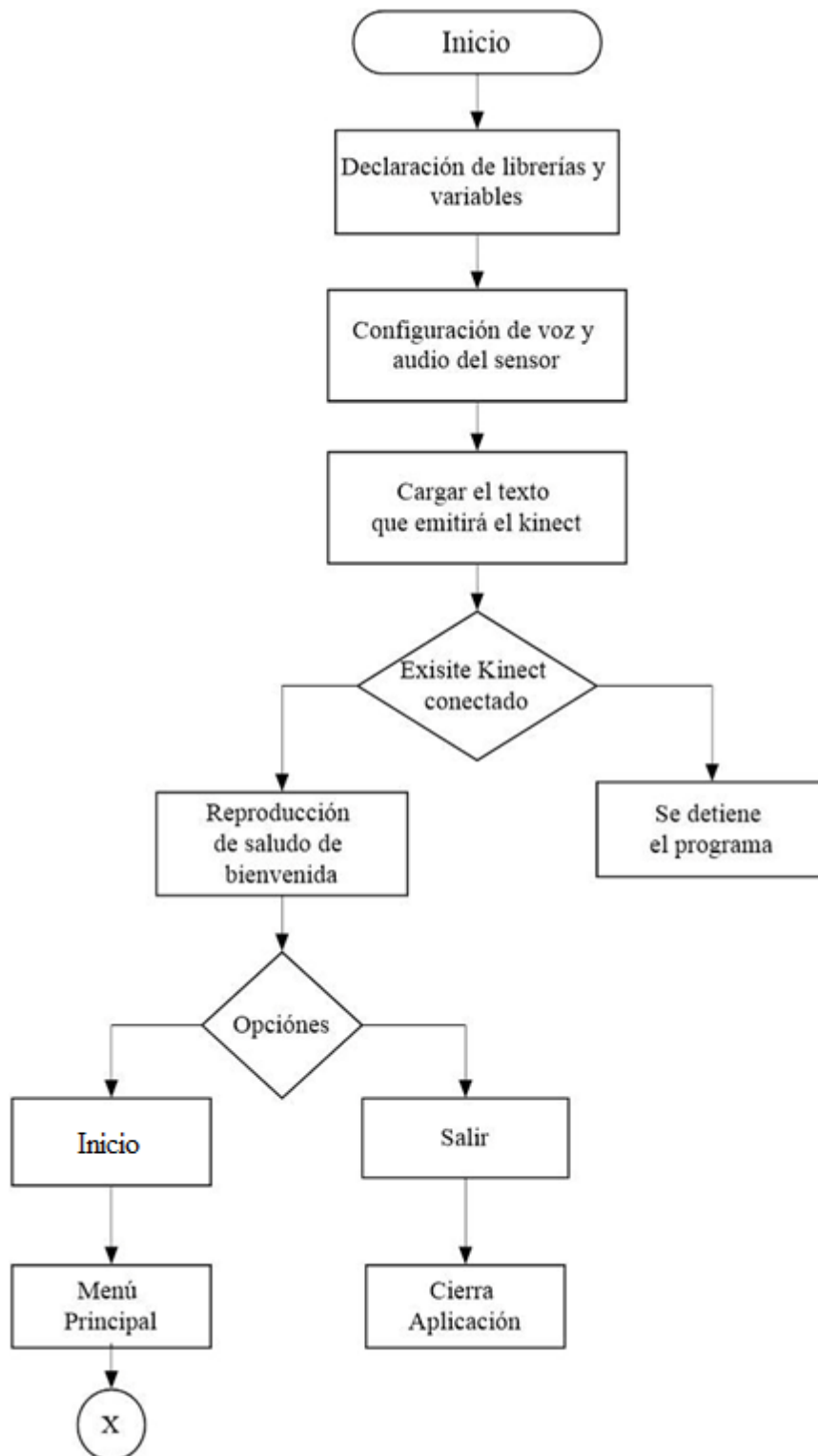


Figura 28. Flujograma de la programación, parte 1

Fuente: Autoras

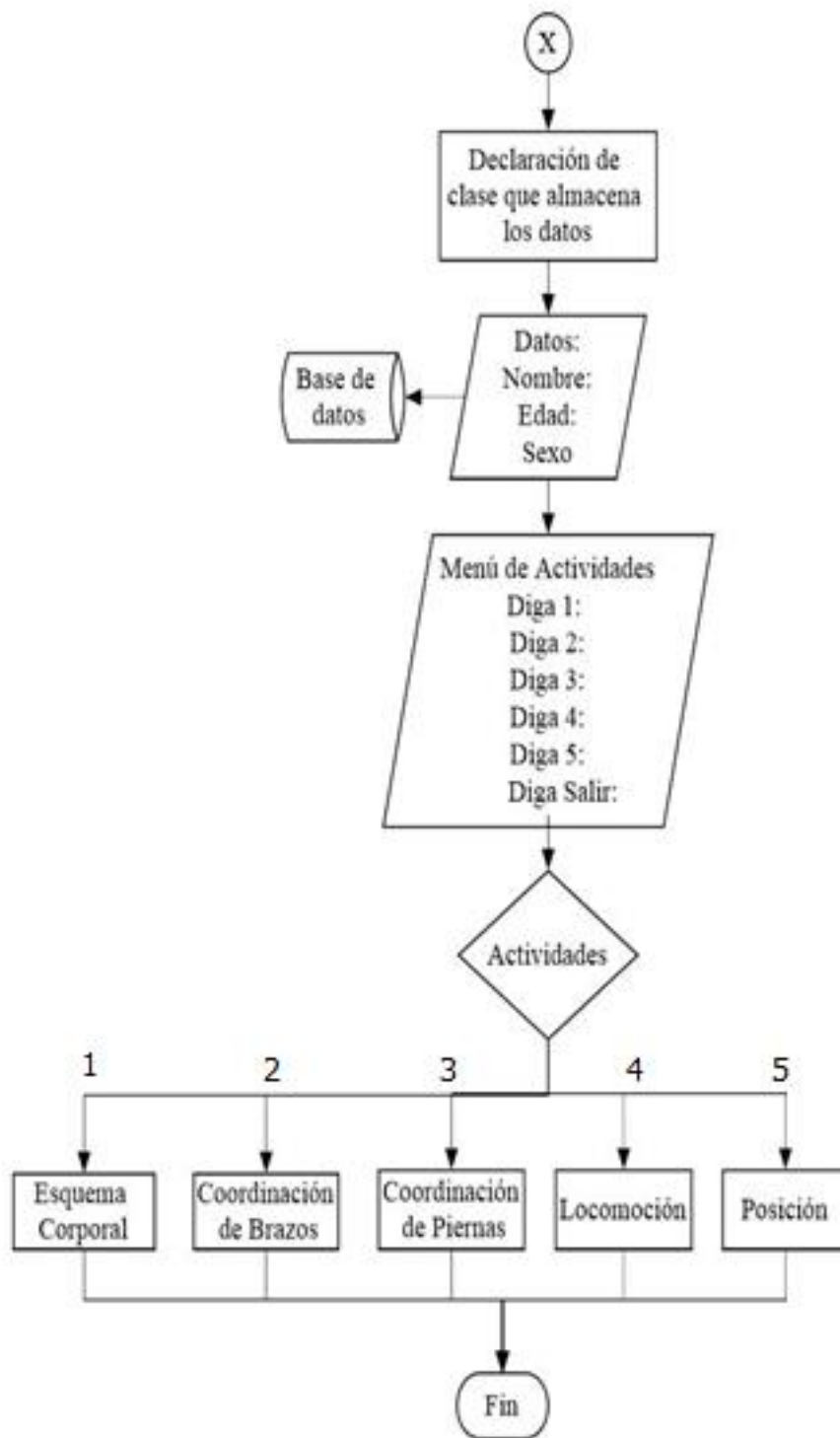


Figura 29. Flujograma de la programación, parte 2

Fuente: Autoras

2.6.5. Recolección de datos a base de pruebas

Para verificar la funcionabilidad de KinectMove se trabajó a base de pruebas con personas que simulan discapacidad visual, registrando y enviando los datos a un servidor web, permitiendo así comparar si los resultados son los mismos tanto en la aplicación como en el reporte generado.

2.7. Procesamiento y análisis

2.7.1. Inicio de KinectMove

Tal como se muestra en la figura 30 al inicializar la aplicación se desplegará una pantalla principal, o pantalla de bienvenida, en la cual el usuario con discapacidad visual deberá decir la palabra INICIO para acceder al menú principal del sistema, caso contrario dirá SALIR para abandonarlo.



Figura 30. Pantalla Inicial

Fuente: Autoras

Para inicializar, configurar o modificar un motor de síntesis de voz, crear mensajes, generar voz, responder a eventos, modificar características de voz y asignar el control del sensor Kinect se tomó las siguientes instrucciones:

```
using System.Speech.Recognition;
using System.Speech.Synthesis;
using Microsoft.Kinect;
```

Es importante realizar la declaración de clases, métodos, variables y objetos que cumplan una función específica. Esto se lo realiza dentro del programa principal que se genera al momento de crear un nuevo proyecto, el mismo que es nombrado por el programador.

Además se debe seleccionar un audio adecuado y de fácil recepción, para que el usuario pueda entender las instrucciones claramente, y la única reproducción de voz que se asemeja a la nuestra es "Microsoft Sabina Desktop" de origen Spanish MX.

2.7.2. Menú Principal de KinectMove

Una vez que el usuario haya seleccionado la opción inicio, el estará en la capacidad de seleccionar que actividad quiere realizar, pero antes de ello debe registrar sus información, los mismos que serán almacenados en una base de datos. En la figura 31 se observa claramente la ventana de menú principal, que contiene las actividades como: conocimiento corporal, coordinación de brazos, coordinación de piernas.

Se buscó una manera de proporcionar una infraestructura de tiempo de ejecución para la gestión de datos relacionales como objetos sin perder la capacidad de consultar, para esto se utilizó una clase proporcionada por el mismo lenguaje de programación llamado “**KinectMoveLinkToSQLDataContext**”, permitiendo que el seguimiento de los datos se realice de forma automática.

Para que se guarde los datos del usuario se debe tomar un tiempo de espera de 2 segundos entre cada registro para que el Kinect reciba correctamente la información, debido a la sensibilidad que tiene su micrófono. Es necesario

mencionar que si no se ha escogido unas de las opciones que el sistema indica, este se mantendrá en el menú principal hasta que sea una instrucción válida.



Figura 31. **Pantalla Menú Principal**

Fuente: **Autoras**

2.7.3. Actividades Psicomotrices

En la sección 1.4.5 se mencionó las actividades que ayudan a las personas con discapacidad visual a desarrollar sus habilidades psicomotrices.

En estas actividades es importante que el sensor active sus múltiples recursos como la cámara a color, el infrarrojo y la profundidad a través de:

```
MultiSourceFrameReader _reader;
```

Como se mencionó en la sección 1.7.2.1 el sensor Kinect v2.0 detecta 25 articulaciones más importantes del cuerpo humano, cada una de ellas trabaja en coordenadas $x - y - z$, y el sensor necesita tomar esos datos, es necesario trabajar con una matriz en la que se almacene esos valores para posteriormente realizar una resta que permita evaluar la posición actual del usuario.

Kinect captura los valores de las posiciones en las que se encuentran los Joints en coordenadas x y z, y la almacena en una variable llamada position3D. Para graficar el esqueleto se ha utilizado una clase obtenida de los demos del SDK de Kinect la cual se muestra en el Anexo 3. La instrucción para dibujar el esqueleto es la siguiente:

```
canvas.DrawSkeleton(body);
```

Como se va a trabajar con los Joints del cuerpo humano, se necesita utilizar varias instrucciones como:

```
BodyFrameReference.AcquireFrame://es el marco de referencia del cuerpo  
IList<Body> _bodies;//son variables que ayudan a dibujar el cuerpo  
foreach (var body in _bodies)//escanea todos los posibles cuerpos  
(body.IsTracked)//Verifica si el esqueleto es correcto
```

Este es un proyecto amplio y se necesitó dividir el programa principal en clases que son archivos independientes que permitió trabajar con varios proyectos al mismo tiempo, es decir solo se tenía que llamar a la clase sin tener que volver a crear el archivo de código fuente.

La clase con la que se trabaja para la detección de posturas es detected, dentro de la misma se llevan a cabo los cálculos y métodos que son utilizados para los ejercicios de psicomotricidad. Ver Anexo 2

En el anexo 5 se puede ver el gráfico del esqueleto en las diferentes actividades realizadas a lo largo del sistema.

2.7.3.1. Actividad para desarrollar el Conocimiento Corporal

Como se puede apreciar en la figura 32, el flujograma explica el funcionamiento y la secuencia a seguir de esta actividad, además se muestra la ventana que será presentada al usuario, el mismo que tendrá la oportunidad de

elegir entre dos opciones de ejercicios. El sensor dará indicaciones como: si el usuario está haciendo correctamente o no el ejercicio y de cuantas oportunidades tiene para realizar el mismo. Ver figura 33

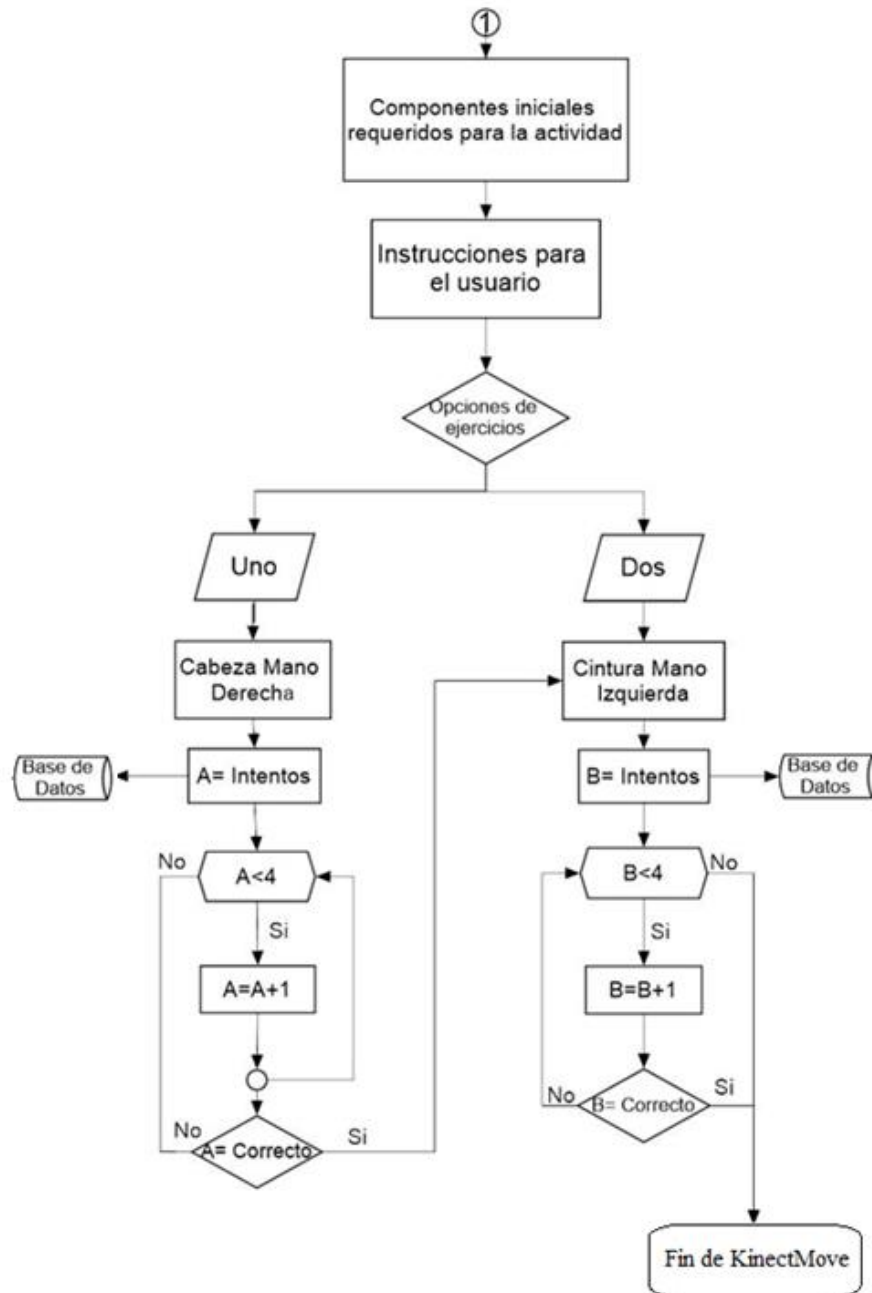


Figura 32. Flujograma Conocimiento Corporal

Fuente: Autoras



Figura 33. Pantalla Conocimiento Corporal

Fuente: Autoras

Opción 1:

Esta actividad permitirá que el usuario se toque la cabeza con la mano derecha, para ello el propio SDK de Kinect permite evaluar la posición de diferentes Joints que intervienen en la misma.

Para que la postura sea correcta se evaluó: head = cabeza; handRight = mano derecha en las 3 coordenadas. Para realizar la operación matemática se buscó la alternativa de cambiar el nombre a las variables debido a que los Joints tienen las mismas coordenadas es decir:

head.x = x

handRight.x = x 1

head.y = y

handRight.y = y 1

head.z = z

handRight.z = z 1

El sensor considerará que la mano está tocando la cabeza cuando la distancia entre las dos sea mayor que 0.06f, para obtener esa distancia se realizó una operación matemática, la misma que se detalla a continuación:

```

public bool HandOnHead(float x, float y, float z, float x1, float y1,
float z1)
{
float distance = (x1 - x) + (y1 - y) + (z1 - z);
if (Math.Abs(distance) > 0.06f)
{
derCab = 0;
return handHead = false;
}
else
return handHead = true;
}

```

Para tener mayor precisión en la detección de la postura se analizó los posibles errores que puedan presentarse con los Joints:

- Mano Izquierda en Cabeza.
- Mano Izquierda en Hombro.
- Mano Izquierda en Cintura.
- Mano Derecha en Hombro.
- Mano Derecha en Cintura.

Opción 2:

Esta actividad permitirá que el usuario se toque la cintura con la mano izquierda. De igual manera Kinect captura los valores de las posiciones en las que se encuentran los Joints que intervendrán en este ejercicio. Para que esta postura sea correcta se evaluó: hipLeft = cintura izquierda; handLeft = mano izquierda en las 3 coordenadas, se cambia el nombre a las variables:

hipLeft.x = x	handLeft.x = x1
hipLeft.y = y	handLeft.y = y1
hipLeft.z = z	handLeft.z = z1

La distancia entre la mano y la cintura debe ser mayor que 0.07f para que el sensor considere que la postura es correcta, se realiza la operación matemática detallada a continuación, para obtener esta distancia:

```
public bool HandOnHip(float x, float y, float z, float x1, float y1,
float z1)
{
float distance = (x1 - x) + (y1 - y) + (z1 - z);
if (Math.Abs(distance) > 0.07f)
{
accumulatorError++;
izqCintu = 0;
return false;
}
else
return true;
}
```

De igual forma se tomó en cuenta los posibles errores que puedan generarse con los Joints:

- Mano Derecha en Cabeza.
- Mano Izquierda en Cabeza.
- Mano Izquierda en Hombro.
- Mano Derecha en Hombro.
- Mano Derecha en Cintura.

2.7.3.2. Actividad para desarrollar la Coordinación de Brazos

De igual manera en la figura 34 muestra el proceso de esta actividad, es decir los pasos que el usuario debe seguir al realizar el ejercicio, el mismo que cuenta con la oportunidad de levantar sus brazos, independientemente de las instrucciones del sistema.

Es necesario determinar los posibles errores que existan en la ejecución de los ejercicios, los cuales fueron:

- Levantar la mano equivocada
- No levantar las manos

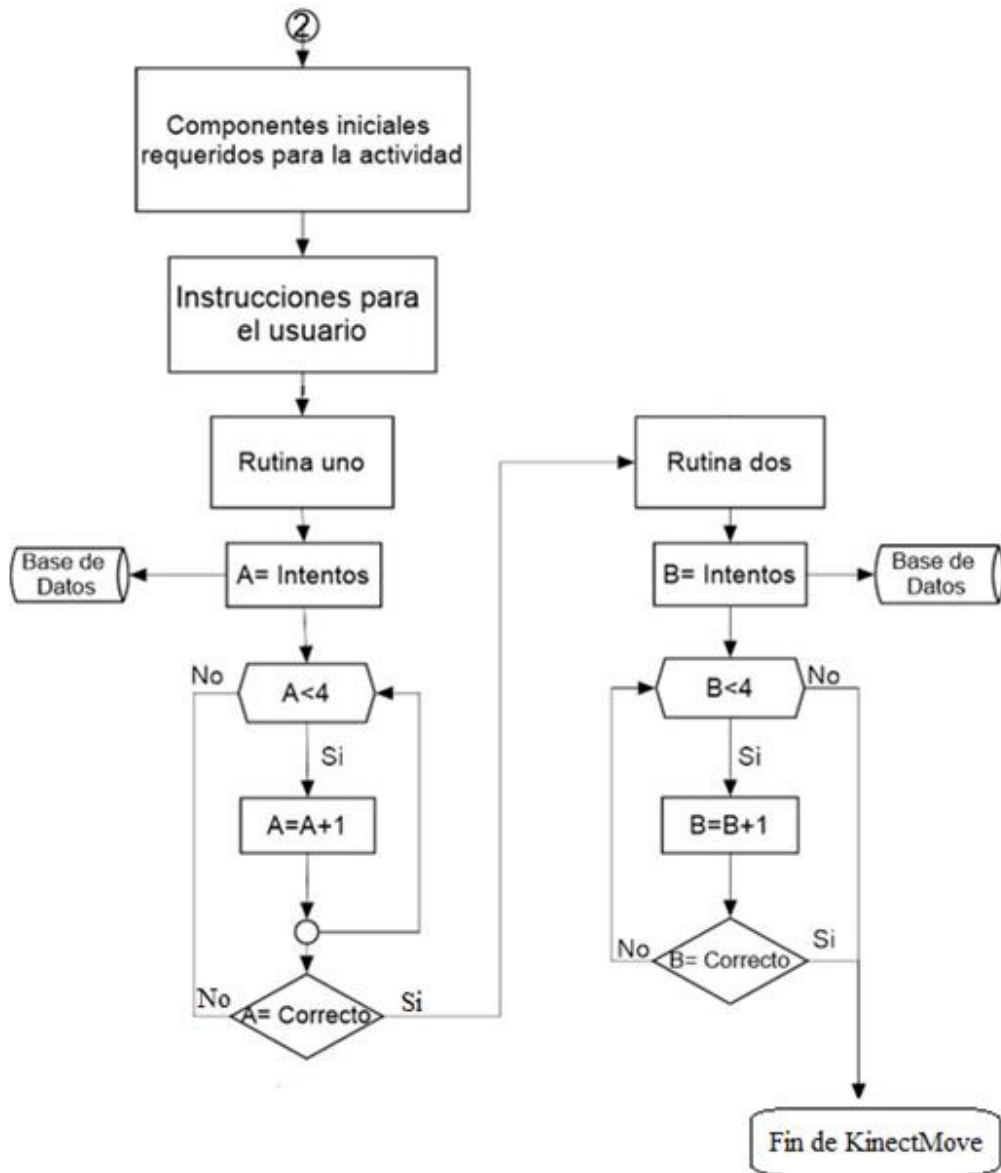


Figura 34. **Flujograma Coordinación de brazos**

Fuente: **Autoras**

El usuario deberá levantar el brazo izquierdo al escuchar un número par y el brazo derecho al escuchar un número impar tocar su cabeza con la mano derecha. Para esta actividad se diseñará una serie de números que se irán generando aleatoriamente, es decir se asignará un numero aleatorio del 1 al 10, para que el número sea par se realizará una división para 0, habrá un contador de pares que se generara hasta dos números seguidos y de no ser así se reiniciara el conteo. Los

parámetros a evaluar en estos ejercicios son: head = cabeza; handRight = mano derecha; handLeft = mano izquierda; shoulderLeft = hombro izquierdo; shoulderRight= hombro derecho; elbowLeft = codo izquierdo; elbowRight = codo derecho en las 3 coordenadas x y z.

Las variables fueron asignadas de la siguiente manera:

VARIABLES	
Número par mano derecha	Número impar mano izquierda
handRight.x = x	handLeft.x = x
elbowRight.x = x1	elbowLeft.x = x1
shoulderRight.x = x3	shoulderLeft.x = x3
head.y = y	head.y = y
elbowRight.y = y1	elbowLeft.y = y1

Tabla 3. Asignación de variables – coordinación de brazos

Fuente: Autoras

Se evalúa que los valores de la posición en la que se encuentran la cabeza, codo, mano y hombro sean mayores que 0.02f y al restar las posiciones entre la cabeza y el codo derecho el resultado sea mayor 0.02f.

```
// MANO DERECHA NUMERO PAR
public bool HandRightUp(float x, float x1, float x3, float y, float y1)
{
    if (distanceHandUp(x,x1,x3,y,y1)>0.02f && y-y1>0.02f)
    {
        handright = false;
        return false;
    }
    else
    {
        handright = true;
        return true;
    }
}
```

```

//MANO IZQUIERDA - NUMERO IMPAR
public bool HandLeftUp(float x, float x1, float x3, float y, float y1)
{
if (distanceHandUp(x,x1,x3,y,y1)>0.02f && y-y1>0.02f)
{
handleft = false;
return false;
}
else
{
handleft = true;
return true;
}
}
}

```

2.7.3.3. Actividades para desarrollar la coordinación de piernas

Como se puede observar en la figura 35 el proceso de esta actividad es que el usuario deberá levantar la pierna correspondiente de acuerdo a las instrucciones que le dé el sistema.

Los posibles errores que se consideraron cuando se hizo la programación de esta actividad son:

- Levantar la pierna equivocada.
- No levantar la pierna.

Para esta actividad se evalúa cuatro Joints que son kneeLeft = rodilla izquierda; ankleRight = tobillo derecho; kneeRight = rodilla derecha; ankleLeft = tobillo izquierdo. De igual manera se debe asignar cada una de las variables, como se muestra en la tabla 4

VARIABLES	
Pierna derecha	Pierna izquierda
kneeLeft.x = x	kneeRight.x = x
ankleRight.x = x1	ankleLeft.x = x1
kneeLeft.z = z	kneeRight.z = z
ankleRight.z = z1	ankleLeft.z = z1

Tabla 4. Asignación de variables

Fuente: Autoras

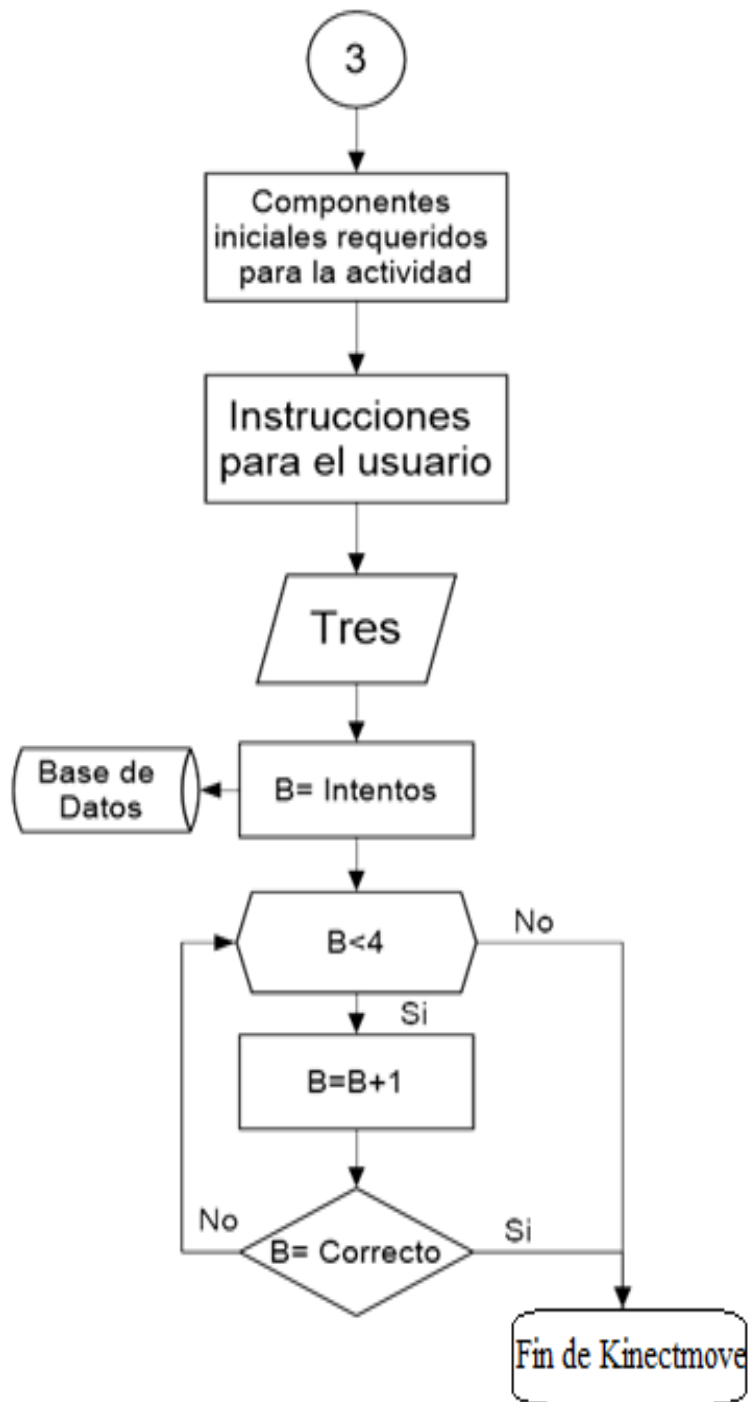


Figura 35. **Flujograma Coordinación de piernas**

Fuente: Autoras

Para que el sensor considere que la pierna se encuentra levantada se realizó una operación matemática entre las posiciones de la rodilla y el tobillo, esta operación da como resultado la distancia, la misma que debe ser mayor que 0.07f para la pierna derecha y 0.05f para la pierna izquierda.

```
//OPERACION PIERNA DERECHA
public bool rightLegUp(float x, float x1, float y, float y1, float z,
float z1)
{
if (Math.Abs((x - x1) + (z - z1)) > 0.07f)
{
rightLeg = false;
return false;
}
else
rightLeg = true;
return true;
}
```

```
//OPERACION PIERNA IZQUIERDA
public bool leftLegUp(float x, float x1, float y, float y1, float z,
float z1)
{
if (Math.Abs((x - x1) + (z - z1)) > 0.05f)
{
leftLeg = false;
return false;
}
else
leftLeg = true;
return true;
}
```

2.7.3.4. Actividades para desarrollar la Locomoción

En esta actividad se va a realizar un conteo descendente que inicializará en 15, en donde el usuario deberá desplazarse de izquierda a derecha, cuando el conteo llegue a cero el deberá unir sus pies. Ver flujograma del proceso en la figura 36.

Para evaluar si el resultado final es correcto se tomó los Joints (puntos) Footleft = Pie izquierdo, Footright = Pie derecho a cada posición se le ha asignado una variable.

footLeft.X = x	footRight.X = x1
footLeft.Y = y	footRight.Y = y1
footLeft.Z = z	footRight.Z = z1

Para que el sensor detecte que las piernas están unidas se hizo una operación matemática entre las posiciones del pie derecho y el pie izquierdo, esta operación da como resultado la distancia, la misma que debe ser mayor que 0.02f.

```
//OPERACION LOCOMOCIÓN
public bool foots(float x, float y, float z, float x1, float y1, float
z1)
{
float distance = (y1 - y)+ (z1 - z);
if (Math.Abs(distance) > 0.02f)
{
accumulatorError++;
piesFirmes = 0;
return piesCero = false;
}
else
return piesCero = true;
}
```

Para esta actividad se ha considerado tres casos de posibles errores que pueden cometer los usuarios al no escuchar ni comprender correctamente las instrucciones dadas por el sistema.

Por lo cual KinectMove tomará como postura incorrecta si el usuario:

- Se detiene antes de que el conteo regresivo finalice.
- No se detiene cuando el contador llegue a cero.
- No une los pies al momento en que el sistema llega a cero.

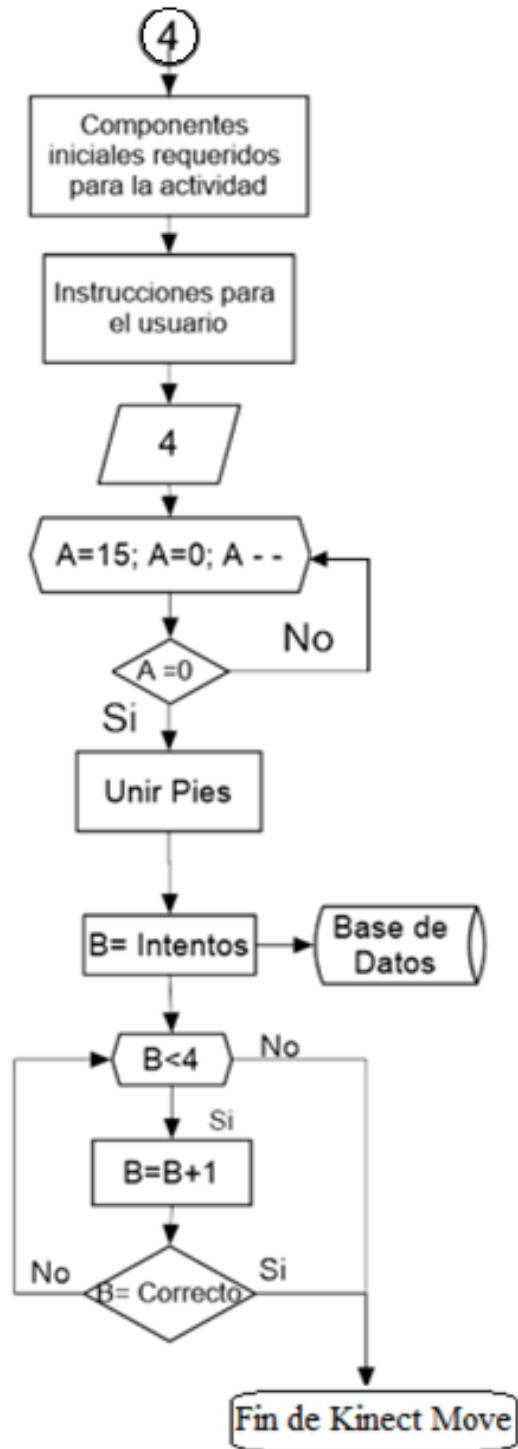


Figura 36. **Flujograma Locomoción**

Fuente: Autoras

2.7.3.5. Actividades para identificar la Posición

En esta actividad el sistema reconoce, captura, compara y evalúa los datos de los Joints de la cadera y rodilla izquierda en coordenadas x y z, para detectar si el usuario está en la posición de parado o sentado. Ver flujograma del proceso en la figura 37

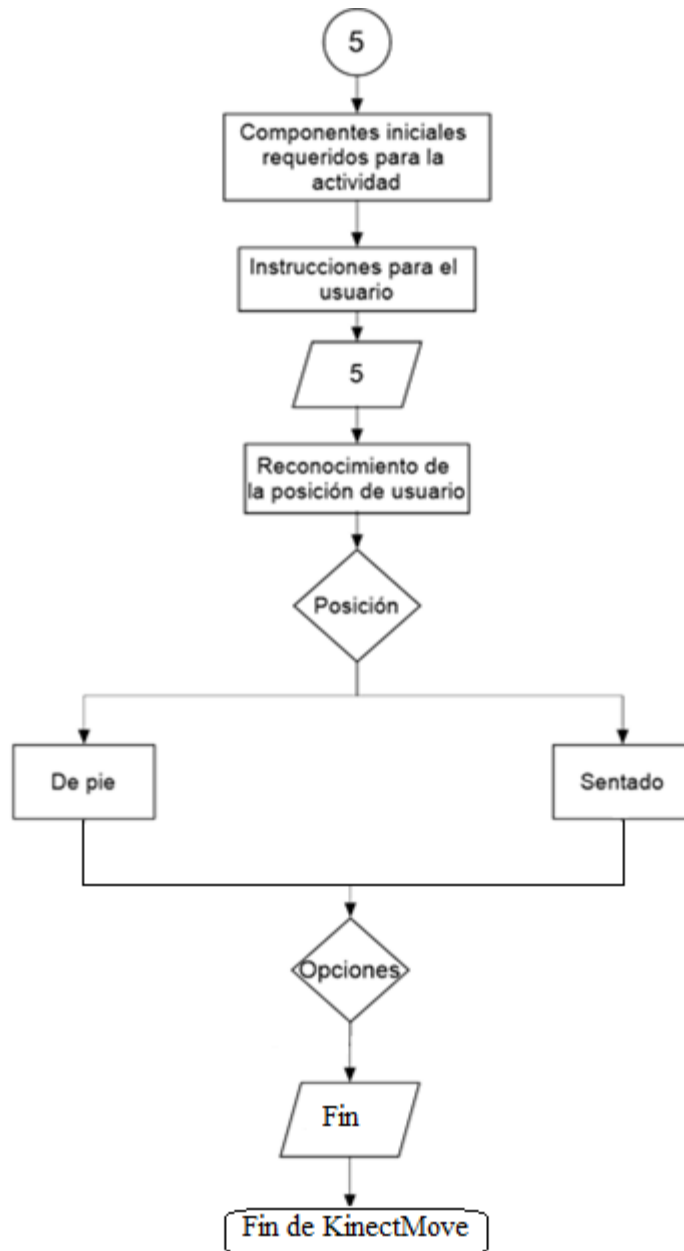


Figura 37. **Flujograma Posición**

Fuente: Autoras

Para realizar la operación matemática se le asignaron variables a los Joints que intervienen en la actividad, es decir:

hipLeft.X = x	kneeLeft.X = x1
hipLeft.Y = y	kneeLeft.Y = y1
hipLeft.Z = z	kneeLeft.Z = z1

El objetivo de asignar estos valores es detectar la posición de sentado, mediante una resta de Joints cadera y rodilla izquierda en coordenadas y (hipLeft.Y - kneeLeft.Y), obteniendo distancia $> 0.1f$. Si el sistema no detecta que el usuario está sentado querrá decir que está en la posición de parado.

```
// POSICIÓN SENTADO
public bool sitdown(float x, float y, float z, float x1, float y1, float
z1)
{
float distance = (y - y1);
if (Math.Abs(distance) > 0.1f)
{
accumulatorError++;
accumulator = 0;
return stand = false;
}
Else
return stand = true;
}
```

2.7.4. Registro y Reporte de Datos

El registro en SQL Server es una base que permite la creación flexible de vistas dinámicas de datos mediante consultas e informes. Como se muestra en la figura 39 se creó una tabla donde guarde el registro de cada participante que realicen las rutinas de actividades, utilizando una estructura de herramientas de cuadros y de texto.

Visual Studio y SQL son entornos muy diferentes por lo que es necesario crear una conexión, para que los datos se guarden en el servidor. Ver figura 38

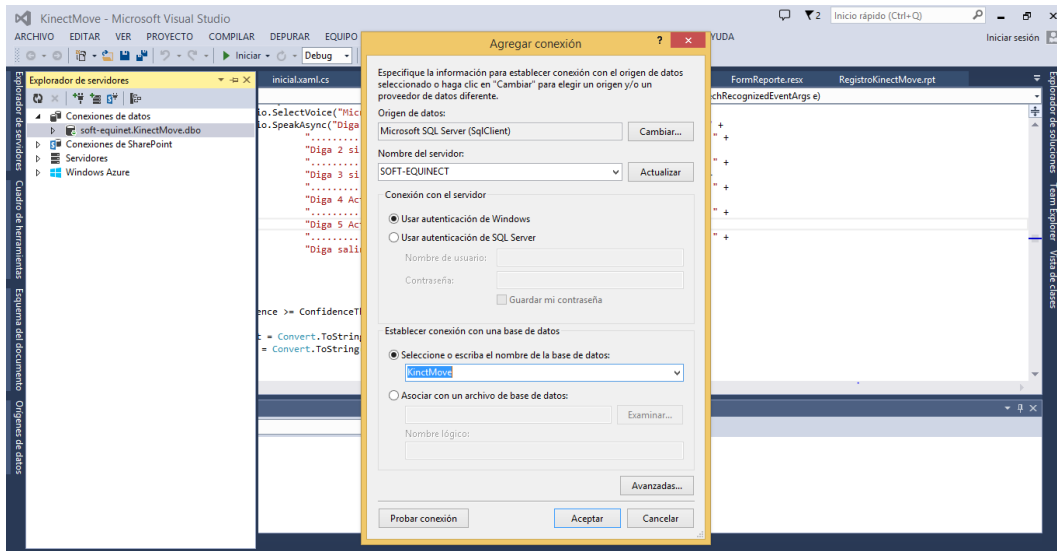


Figura 38. Conexión SQL – C#

Fuente: Autoras

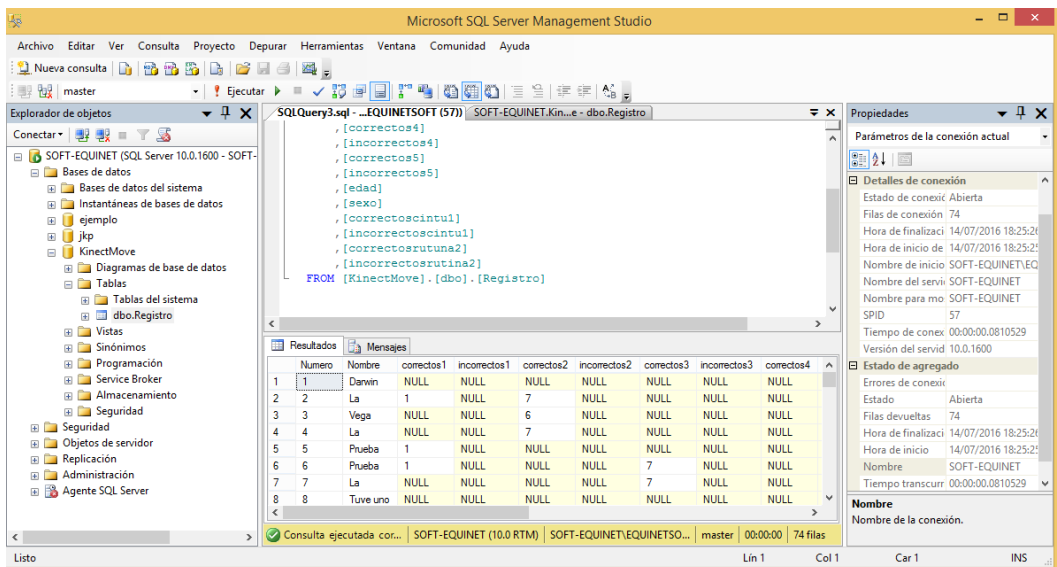


Figura 39. SQL Server

Fuente: Autoras

Para enviar los datos del usuario desde C# a SQL se trabajó con la clase Link to SQL, que sólo funciona con bases de datos de Microsoft SQL Server, para posteriormente utilizarlo y trabajar como un objeto, que contiene internamente una tabla de elementos para ver el registro del participante actual.

Cuando cada usuario diga la palabra SALIR, se crea una acción permitiendo así generar el reporte de sus datos. Para ello se utilizó la herramienta Crystal Reports que permite organizar y presentar la información la misma que será transformada y enviada al Hosting o servidor web en formato .pdf.

Crystal Reports es una herramienta externa de Visual, por lo que se requirió la herramienta Crystal report viewer que permite realizar la extracción de los datos, escogerlo los datos que corresponden y representarla en la base de datos.

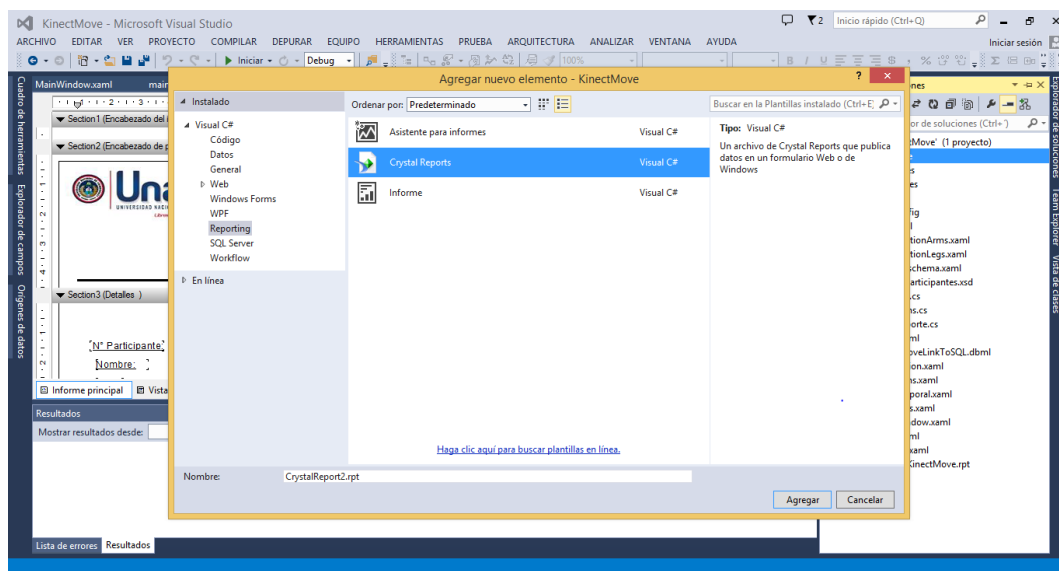


Figura 40. Elemento Crystal Report

Fuente: Autoras

Para exportar el reporte al sitio web se utiliza métodos y clases de parametrización como son: nombre de usuario, contraseña y ruta del host, es decir:

```
// Subir archivo a host
try
{
WebClient client = new WebClient();
client.Credentials = new NetworkCredential("ogmagp", "Ogmagp2017");
client.BaseAddress =
"ftp://107.180.36.178/kinectmove.com/reporte/reporte.pdf";
client.UploadFile("", "C:\\Users\\EQUINETSOFT\\Documents\\Visual Studio
2013\\Projects\\KinectMove\\KinectMove\\reporte\\reporte.pdf");
}
```

2.8. Comprobación de la hipótesis.

Para comprobar la hipótesis se utiliza el método estadístico Chi-Cuadrado, puesto que los valores de estas no son números sino categorías y accede a dos grados de posibilidad, alternativa la que se quiere comprobar y nula la que rechaza la hipótesis alternativa.

2.8.1. Planteamiento de la hipótesis estadística.

Hipótesis nula (H₀): El diseño e implementación de un sistema interactivo mediante Tecnología Kinect v2.0 no comprobará el funcionamiento de las actividades que desarrollen las personas con discapacidad visual.

Hipótesis alternativa (H₁): El diseño e implementación de un sistema interactivo mediante Tecnología Kinect v2.0 comprobará el funcionamiento de las actividades que desarrollen las personas con discapacidad visual.

2.8.2. Establecimiento de nivel de significancia.

Las pruebas se realizaron con un 95% de confiabilidad, es decir, se trabajó con un nivel de significancia de $\alpha=0.05$.

2.8.3. Determinación del valor estadístico de prueba.

Si el valor de CHI-CUADRADO es menor o igual que el Chi-Cuadrado crítico entonces se acepta la hipótesis nula, caso contrario se la rechaza.

$$X^2 \leq \text{Valor Crítico}$$

Para aceptar o rechazar esta hipótesis se tomó en cuenta el sistema interactivo, que determina si utilizando los comandos de audio y voz permiten determinar si funciona o no el reconocimiento de posturas en todos los ciclos del sistema. Por

lo que se requirió tomar un número de muestras de las pruebas realizadas con el sistema. Ver tabla 5

PRUEBAS DEL FUNCIONAMIENTO DEL SISTEMA

N° de muestras	CICLOS DEL SISTEMA					Registro de datos
	Actividad de conocimiento corporal	Actividad de conocimiento corporal	Actividad de conocimiento corporal	Actividad de conocimiento corporal	Actividad de conocimiento corporal	
1	NO	SI	SI	SI	SI	NO
2	NO	SI	SI	SI	SI	NO
3	NO	SI	SI	SI	NO	NO
4	SI	SI	SI	NO	SI	NO
5	SI	SI	SI	SI	SI	SI
6	SI	SI	SI	NO	SI	SI
7	NO	SI	SI	SI	SI	SI
8	NO	SI	NO	SI	SI	SI
9	NO	NO	NO	SI	SI	SI
10	NO	NO	NO	SI	SI	SI
11	SI	NO	SI	SI	SI	SI
12	SI	SI	SI	SI	SI	SI
13	SI	SI	SI	SI	SI	SI
14	SI	SI	SI	SI	SI	SI
15	SI	SI	SI	SI	SI	SI
16	SI	SI	SI	SI	SI	SI
17	SI	NO	SI	SI	SI	SI
18	NO	NO	SI	SI	SI	SI
19	NO	SI	SI	SI	SI	SI
20	NO	SI	SI	SI	SI	SI
21	SI	SI	SI	SI	SI	SI
22	SI	SI	SI	SI	SI	SI
23	SI	SI	SI	SI	SI	SI
24	SI	SI	SI	SI	SI	SI
25	SI	SI	SI	SI	SI	SI
26	SI	SI	SI	SI	SI	SI
27	SI	SI	SI	SI	SI	SI
28	SI	SI	SI	SI	SI	SI
29	SI	SI	SI	SI	SI	SI
30	SI	SI	SI	SI	SI	SI

Tabla 5. Numero de muestras

Fuente: Autoras

Para realizar el cálculo de χ^2 es necesario trabajar con la frecuencia obtenida, y frecuencia esperada los valores se han obtenido de la tabla 5

FRECUENCIA OBTENIDA

FUNCIONAMIENTO	CICLOS DEL SISTEMA						TOTAL
	1	2	3	4	5	6	
SI	20	25	27	28	29	26	125
NO	10	5	3	2	1	4	55
TOTAL	30	30	30	30	30	30	180

Tabla 6. Frecuencia obtenida

Fuente: Autoras

FRECUENCIA ESPERADA

FUNCIONAMIENTO	CICLOS DEL SISTEMA						TOTAL
	1	2	3	4	5	6	
SI	25.83	25.83	25.83	25.83	25.83	25.83	125
NO	4.17	4.17	4.17	4.17	4.17	4.17	55
TOTAL	30	30	30	30	30	30	180

Tabla 7. Frecuencia esperada

Fuente: Autoras

La tabla 8 muestra los valores obtenidos para la comprobación de la hipótesis.

Número de Filas	2
Número de Columnas	6
Valor calculado X^2	14.1677
Grados de libertad	5
Nivel de significancia	0.05
Probabilidad	0.95
Valor crítico	11,070

Tabla 8. Tabla de resultados

Fuente: Autoras

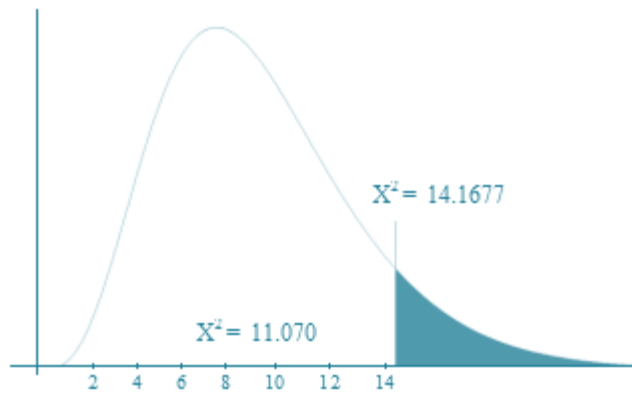


Figura 41. **Resultado de comparación X^2 tabla con el X^2 calculado**
 Fuente: Autoras

Como $x^2_{calculado} = 14.1677 > x^2_{tabla} = 11.070$ (Anexo 3) se rechaza H_0 y se concluye, al tener un nivel de significancia del 0.05 y aceptar la hipótesis alternativa, es decir:

El diseño e implementación de un sistema interactivo mediante Tecnología Kinect v2.0 permitirá comprobar el funcionamiento de las actividades que desarrollen las personas con discapacidad visual.

CAPITULO III

3. RESULTADOS

El estudio del estado de arte del sensor Kinect permitió realizar un análisis comparativo de las características entre las dos versiones existentes, mediante lo cual se aprovechó las propiedades mejoradas de la versión 2.0 para llevar a cabo la implementación de este proyecto.

Por la manera en que fueron desarrollados los algoritmos para las actividades que se ejecutarán en el sistema, se ha logrado tener una mayor precisión al momento de obtener los valores que recibe el sensor cuando realiza el reconocimiento de posiciones y reconocimiento de audio, siendo este el mayor inconveniente que posee el sistema ya que el sensor capta lo mínimo que escucha y guarda esa información.



Figura 42. Usuarios interactuando con KinectMove

Fuente: Autoras

Para constatar el funcionamiento adecuado del sistema interactivo se lo dividió en dos secciones.

Sección 1. Actividades psicomotrices

Por medio de pruebas realizadas con personas que simulan este tipo de discapacidad, se comprobó que cada actividad cumple con las instrucciones que emitan tanto el sensor como el usuario, teniendo un margen de error del 10% en el reconocimiento de articulaciones lo cual genera un retraso en la lectura de datos del sensor.

Instrucciones como selección de actividad, levantar los brazos, levantar la piernas, entre otras.



Figura 43. **Coordinación de brazos**

Fuente: **Autoras**



Figura 44. **Coordinación de piernas**

Fuente: **Autoras**

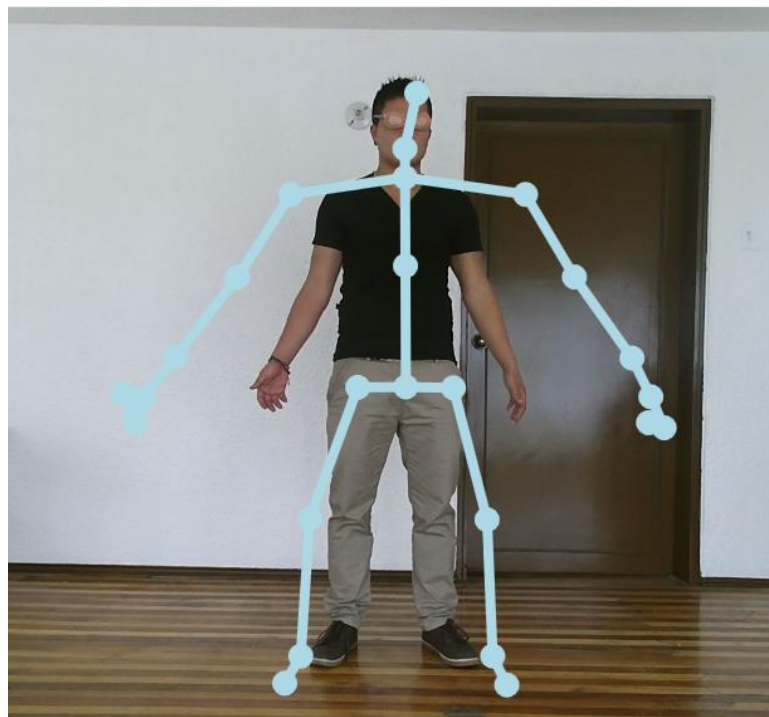


Figura 45. **Reconocimiento de posición**

Fuente: **Autoras**

Sección 2. Registro de datos de usuarios

Los datos de los usuarios son receptados por el sensor Kinect, almacenándose en la base de datos, permitiendo que las personas que requieran de esta información accedan a la misma sin ningún inconveniente, ya que este registro está alojado en un servidor web llamado kinectmove.com. Debido a que el sensor capta toda información de audio emitida en el ambiente, se produce un error al generar el reporte ya que algunas veces este recepta y graba datos erróneos.



Figura 46. Servidor web

Fuente: Autoras

REGISTRO DE DATOS

N° Participante: 1
Nombre: anita
Edad: 34
Sexo: femenino

<u>Actividad</u>	<u>Correctos</u>	<u>Incorrectos</u>
<u>Esquema Corporal</u>		
Primera Actividad	1	2
Segunda Actividad		
<u>Coordinación de Brazos</u>		
Primera Rutina		
Segunda Rutina		
<u>Coordinación de Piernas</u>		
Locomoción		

Figura 47. Reporte de datos del usuario

Fuente: Autoras

CAPITULO IV

4. DISCUSIÓN

El sistema interactivo mediante tecnología Kinect v2.0 permite que las personas con discapacidad visual realicen actividades que les ayude a utilizar las estrategias adecuadas para dominar su cuerpo, procurando mejorar su calidad de vida y que las mismas sean menos dependientes manejando el desarrollo de su psicomotricidad como elemento fundamental para su movilidad.

Dado el continuo avance de la tecnología, el sensor Kinect v2.0 por las características que posee permite la interacción con un ordenador sin la necesidad de utilizar el teclado, mouse, o de manipular un sistema periférico de entrada de datos, permitiendo interactuar con aplicaciones mediante los movimientos gestuales de las manos, comandos de voz o el propio cuerpo como forma de comunicación.

Cabe mencionar que en la actualidad existen varias aplicaciones que emplean el sensor Kinect, por ejemplo en el sistema educativo que ha sido aplicado para optimizar y mejorar la concentración de los niños en el aprendizaje del idioma inglés, en el área de salud es utilizado como complemento de rehabilitación de lesiones, traumatismos y desarrollo de motricidad en personas que han sufrido daños cerebrales, por último empresas y centros comerciales lo han tomado como recurso publicitario tratando de generar experiencias que alteren o motiven el deseo de compra del consumidor, pero estas no son enfocadas para personas con este tipo de discapacidad.

El presente trabajo tiene como propósito implementar un sistema en donde no haya restricción de estatura, género y edad para los usuarios, la ventaja principal que tiene KinectMove es que al momento de realizar la interpretación de datos no los confunde y se lo puede utilizar en cualquier sitio u espacio requerido.

El sistema KinectMove toma datos de acuerdo a las coordenadas x-y-z de las articulaciones para la interpretación de posiciones lo cual difiere en cuanto al proyecto Diseño e implementación de un dispositivo para rehabilitación de rodilla con envío de datos por RF que tiene un limitado método de receptar los datos del Kinect ya que ha sido realizado a través de captura de imágenes, esto no es muy recomendable porque el sensor va a grabar la imagen obtenida y no reconocerá a otro usuario que posea características diferentes, de la misma manera se tiende a confundir a las personas con objetos por lo tanto el dispositivo requiere que el lugar en donde se lo vaya a utilizar este siempre despejado, también la posición en la que realice el ejercicio debe ser siempre la misma.

CAPITULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

Al terminar la presente investigación se concluye lo siguiente:

- Mediante el estudio del estado de arte del sensor Kinect se pudo aprovechar las características mejoradas con las que cuenta la versión 2.0, brindando ventajas y oportunidades al momento de realizar la programación.
- El sistema es capaz de reconocer al esqueleto del usuario en cualquier ambiente que desee realizar las actividades e interactuar con él por medio de comandos de voz y audio.
- Con la ayuda de métodos y clases que contiene el lenguaje de programación C# se logra que el sensor capture los datos de las coordenadas x-y-z en las que se encuentra los puntos del esqueleto del usuario.
- Kinect no es un sensor inteligente lo cual dificulta la interacción por comandos de voz entre el usuario y el sistema, ya que recepta el mínimo sonido que pueda producirse en el ambiente de trabajo.
- Una vez finalizada la implementación del proyecto se realizaron pruebas y mediante el reporte del registro de datos se verificó el correcto funcionamiento de las actividades psicomotrices, y buscar posibles deficiencias o limitaciones que pueda causarle contratiempos al usuario.

5.2. Recomendaciones

Luego de haber realizado las conclusiones de la investigación se proponen las siguientes recomendaciones:

- El sensor Kinect v2.0 es muy sensible al momento de recibir los comandos de voz y tiende a confundir el ruido con datos de entrada, por ello se recomienda restringir estos datos en la programación, para que la información que este recibe sea la correcta.
- Realizar varias pruebas para definir las coordenadas con las cuales es mejor trabajar, para el reconocimiento de las posturas y así evitar el cruce de articulaciones.
- Para tener un mejor control de la programación se trabaja con métodos y clases predefinidos de Visual Studio, permitiendo así reducir errores de compilación.
- Revisar los demos del SDK de Kinect v2.0 para tener una guía de programación ya que es muy útil porque contiene información de paquetes y elementos requeridos por el sensor.
- Para aplicaciones posteriores en las que empleen el sensor Kinect con utilización de comandos de voz sería útil la implementación de filtros los cuales capten únicamente la voz del usuario.

CAPITULO VI

6. PROPUESTA

6.1. Título de la propuesta

SISTEMA INTERACTIVO MEDIANTE TECNOLOGÍA KINECT V2.0 PARA DESARROLLAR LAS HABILIDADES PSICOMOTRICES EN PERSONAS CON DISCAPACIDAD VISUAL

6.2. Introducción

En el país, el desarrollo de las tecnologías ha permitido crear nuevas plataformas que permite satisfacer varias necesidades, pero al mismo tiempo que evoluciona la tecnología se va generando diferentes inconvenientes en la sociedad, es así el caso de las personas con discapacidad visual. .

Gracias a la incorporación de estas tecnologías y herramientas han permitido enfocarse en realizar un sistema interactivo con el sensor Kinect v2.0 para ayudar a las personas con discapacidad visual ya que es muy importante el desarrollo de su psicomotricidad porque necesitan un control y coordinación de sus movimientos para que puedan desenvolverse sin ninguna dificultad y así mejorar su estilo y calidad de vida.

Este sistema interactivo utiliza los sensores de movimiento de Kinect para leer las coordenadas del cuerpo, determinar la postura del usuario y corregir sus movimientos mediante instrucciones de voz. El software permite ejecutar cinco actividades que ayuden a desarrollar las habilidades psicomotrices, cada actividad

cuenta con comandos diferentes de voz que sirven para ajustar la postura de los brazos y piernas de las personas con discapacidad visual.

6.3. Objetivos

6.3.1. Objetivo General

Sistema interactivo mediante Tecnología Kinect v2.0 para desarrollar las habilidades psicomotrices en personas con discapacidad visual.

6.3.2. Objetivos Específicos

- Estudiar el estado del arte de aplicaciones Kinect v1.0 y v2.0, por sus similares características de programación.
- Desarrollar un algoritmo de reconocimiento de gestos e interfaz auditiva, mediante el software de programación para Kinect v2.0.
- Programar actividades para desarrollar las habilidades psicomotrices en personas con discapacidad visual.
- Realizar pruebas y evaluar el funcionamiento del proyecto con registro de datos remotos.

6.4. Fundamentación Científico – Técnico

El presente proyecto fundamenta su funcionamiento en dos cosas principales, la primera que es el sistema de detección que lo hace el sensor Kinect y la segunda que es el control de toda la aplicación que lo hace el software Visual Studio con el compilador C#.

El sensor Kinect, se encarga de la detección del cuerpo del usuario y los movimientos que él haga, principalmente las actividades de psicomotricidad, pues

el objetivo del proyecto es que puedan ser desarrolladas por personas con discapacidad visual.

Visual Studio, es el lenguaje de programación que da la facilidad y ventajas de poder programar e interactuar con la interfaces gráficas, instrumento que vamos a utilizar permanentemente en las actividades.

6.5. Descripción de la propuesta

El presente proyecto que se pone a beneficio de las personas con discapacidad visual, tiene como finalidad generar actividades, la detección, el control, y el envío de datos que ayudan a desarrollar las habilidades psicomotrices.

El primer paso es la detección del cuerpo humano del usuario discapacitado y principalmente la detección de las articulaciones, con esto se puede lograr que la animación tenga sentido y sobre todo tenga éxito.

Luego está lo referente al control de actividades, pues esto hace que el usuario haga de manera correcta y completa el ejercicio, pues si no es así el sensor detectará que el movimiento está mal hecho generando una alerta de error a través de comando de audio en el cual indicará que solamente tiene tres oportunidades.

También está el almacenamiento de datos del usuario, como son el nombre, edad y sexo, son los que se almacenará en la aplicación gracias al software SQL Server, estos datos servirán para verificar si está en uso la aplicación.

Finalmente está la parte del envío del registro de datos, es decir la transmisión del reporte de las actividades realizadas por el usuario, que estarán alojados en un servidor web que es el servicio que provee a los usuarios de Internet un sistema para poder almacenar información, para utilizar las funciones de acceso web por navegador debe tener asociado un dominio en este caso www.kinectmove.com.

6.6. Diseño Organizacional

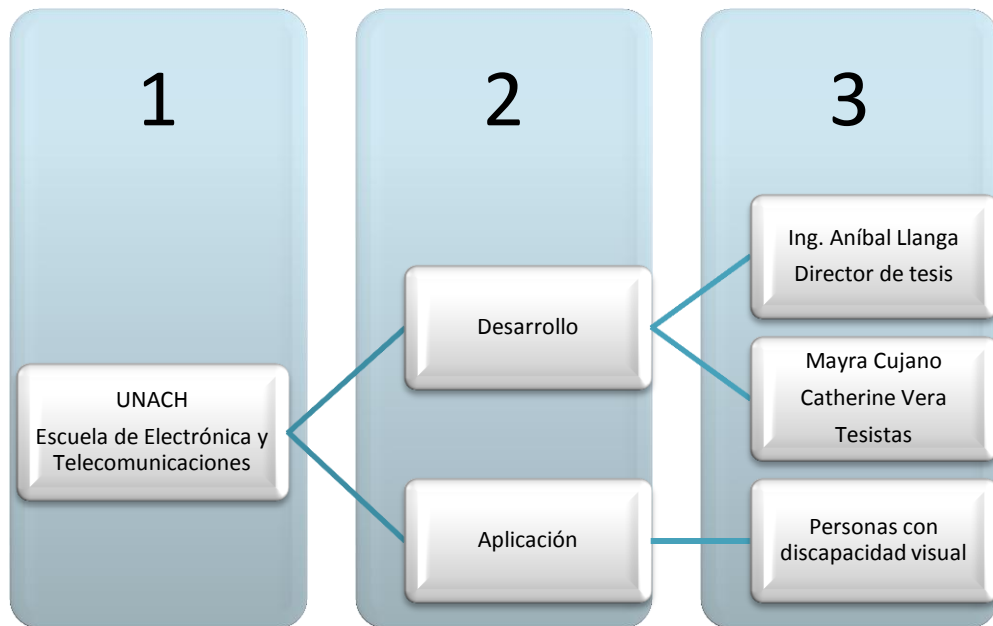


Figura 48. **Esquema organizacional**

Fuente: **Autoras**

6.7. Monitoreo y Evaluación de la propuesta

El monitoreo y la evaluación de la propuesta se la realizará a través de más pruebas con otras personas para determinar si el uso de la aplicación interactiva es o no relevante.

Para una correcta evaluación del sistema se documentará cada prueba realizada, de forma que, al revisar el control y la confiabilidad de funcionamiento del mismo, se pueda establecer las limitaciones que presenta y poder corregirlas.

7. BIBLIOGRAFÍA

- [1] Ayala, A., & Quito, L. (2012). Desarrollo del sentido del equilibrio como factor para el mejoramiento de la condición física de los no videntes de la sociedad de no videntes del Azuay (sonva). Cuenca.
- [2] Balena, F. (2003). Programacion avanzada con Microsoft Visual Basic .NET. España: McGraw-Hill.
- [3] Barbuzano, J. (2015). OpenMind. 5 ejemplos de tecnología para personas ciegas: más allá del Braille.
- [4] Bravo, S. (2014). Programa de Intervencion Motriz para el desarrollo de la Psicomotricidad gruesa de niños(as) de educacion inicial con discapacidad visual, de la escuela municipal de ciegos " Cuatro de Enero" de la ciudad de Guayaquil. Guayaquil .
- [5] Cabezas Manzano, R. V., & Inca Balseca, D. (2014). Diseño e Implementación de un dispositivo para la rehabilitación de rodilla con envío de datos por RF. Riobamba.
- [6] Ceballos, J. (2010). Visual Basic. Mexico: Alfaomega.
- [7] Charre, F. (2009). SQL Server 2008. Madrid: ANAYA.
- [8] Corporation, M. (2014). Human Interface Guidelines v2.0.
- [9] Escamilla, A., & Suaza, M. (2013). Aplicación informática interactiva basada en reconocimiento de gestos para ser usada en música terapias de educación especial. Medellín.
- [10] Fernandez, Z. (2014). Control de Software Educativo mediante Kinect de Microsoft. Madrid.
- [11] Galitz, O. W. (2007). The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques (Vol. III). IN: Indianapolis Wiley.
- [12] Goetschel, E. I. (2014). Análisis de la herramienta Kinect como recurso publicitario generadora de experiencias que motivan los deseos de compra o adquisición del consumidor de la ciudad de Quito. Quito.
- [13] Gómez Jiménez, E. (2010). Aplicaciones con Visual Basic .NET. Mexico D.F: Alfaomega.
- [14] González, A. (2010). Programación de base de datos con C#. México D.F.: Alfaomega Grupo Editor.
- [15] Herreros, M. (2013). Evaluación ergonómica en tiempo real mediante sensores de profundidad de bajo coste (Kinect) . Valencia.
- [16] López, M. P. (2014). Discapacidad Online. MexVox Programa lector de pantalla para invidentes.
- [17] Mendoza, D., Sabino, M., & Márquez, D. (2015). INTERFAZ NATURAL PARA REHABILITACIÓN MOTRIZ. Estado de Hidalgo.
- [18] Molina, A., & Llanga, A. (2015). Diseño e implementacion de un prototipo de gafas electronicas para personas no videntes. Riobamba.
- [19] Peñaherrera, L. (2014). DESARROLLO DE UN SISTEMA DE APRENDIZAJE INTERACTIVO PARA EL ÁREA DEL IDIOMA INGLÉS CON EL SOPORTE DEL KINECT DE MICROSOFT- CASO PRÁCTICO PARA NIÑOS DE 6 A 8 AÑOS EN EL CENTRO EDUCATIVO ILINIZAS. Latacunga.
- [20] Pérez, C. (2010). SQL Server. Mexico: Alfaomega.

- [21] Quishpe, F., & Ulloa, D. (2014). Implementación del modelo de una habitación con un sistema de asistencia para discapacitados, mediante el uso de un sensor de interfaz natural de usuario Kinect de Microsoft. Quito.
- [22] Sánchez Flores, C. (2011). Conociendo lo Nuevo de Visual C# 2010 y Framework 4.0. Lima - Perú: Empresa Editora Macro E.I.R.L.
- [23] Schirt, H. (2010). C# 3.0. México: Mc Graw Hill.
- [24] Trilles, E. (2012). DESARROLLO DE INTERFACES DE USUARIO NATURALES CON KINECT. Valencia.
- [25] Bunker. (septiembre de 2013). Obtenido de <http://www.t3.com/features/exclusive-how-does-microsoft-xbox-kinect-work>
- [26] EcuRed. (2016). Project Natal. Obtenido de http://www.ecured.cu/Project_Natal
- [27] Microsoft . (2016). Developer Network. Recuperado el 2016, de <https://msdn.microsoft.com/en-us/library/dn799271.aspx>
- [28] Microsoft. (2009). Microsoft SQL Server 2008 R2 Enterprise Edition DVD Español. Obtenido de <http://www.intercambiosvirtuales.org/software/microsoft-sql-server-2008-r2-enterprise-edition-dvd-espanol>
- [29] Microsoft. (s.f.). Developer Network. Obtenido de [https://msdn.microsoft.com/es-es/library/ms225593\(v=vs.90\).aspx](https://msdn.microsoft.com/es-es/library/ms225593(v=vs.90).aspx)

8. ANEXOS

ANEXO 1 DESARROLLO DEL CÓDIGO EN C# PANTALLA INICIAL

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Threading;
using System.Speech.Synthesis;
using System.Speech.Recognition;
using System.Speech.AudioFormat;
using System.Runtime.InteropServices;
using Microsoft.Kinect;
using System.IO;

/*
using Microsoft.Speech.AudioFormat;
using Microsoft.Speech.Synthesis;

using Microsoft.Speech.Recognition;
*/
namespace KinectMove
{
    /// <summary>
    /// Lógica de interacción para inicial.xaml
    /// </summary>
    public partial class inicial : Window
    {
        //private KinectAudioStream convertStream = null;
        //Esta variable sera la encargada de representar a nuestro dispositivo
        Kinect y es la que ejecutara
        //algunas de las acciones del hardware
        KinectSensor _sensor;
        //KinectAudioStream convertStream = null;
        List<Span> recognitionSpans;
        MainWindow menu = new MainWindow();
        //menu menu = new menu();
        SpeechRecognitionEngine speechEngine = null;
        int c = 0;

        public inicial()
        {
            InitializeComponent();
        }
    }
}
```

```

SpeechSynthesizer synthesizer = new SpeechSynthesizer();
synthesizer.Volume = 100; // 0...100
synthesizer.Rate = -2; // -10...10
synthesizer.SpeakAsyncCancelAll();
synthesizer.SelectVoice("Microsoft Sabina Desktop");
synthesizer.SpeakAsync("Bienvenidos al sistema interactivo KinectMub,
repita Inicio para acceder al sistema, o SALIR para abandonar la
aplicación");
loadprogressbar();
}

private void loadprogressbar()
{
Duration dur = new Duration(TimeSpan.FromSeconds(10));
DoubleAnimation dblani = new DoubleAnimation(200.0, dur);
pb1.BeginAnimation(ProgressBar.ValueProperty, dblani);
c++;
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
//Elimina Reporte del servidor local despues de subir a host
string[] ficherosCarpeta =
Directory.GetFiles(@"C:\\Users\\EQUINETSOFT\\Documents\\Visual Studio
2013\\Projects\\KinectMove\\KinectMove\\reporte");
foreach (string ficheroActual in ficherosCarpeta)
File.Delete(ficheroActual);
//
_sensor = KinectSensor.Default();
_sensor.Open();
var audioBeamList = _sensor.AudioSource.AudioBeams;
var audioStream = audioBeamList[0].OpenInputStream();

RecognizerInfo ri = GetKinectRecognizer();//establece reconocimiento desde
kinect
speechEngine = new SpeechRecognitionEngine(ri.Id);

// Create a grammar definition ...
speechEngine.LoadGrammar(new DictationGrammar());
speechEngine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(SpeechRecognized);

speechEngine.SetInputToDefaultAudioDevice();
speechEngine.RecognizeAsync(RecognizeMode.Multiple);
}
private void SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
// Speech utterance confidence below which we treat speech as if it hadn't
been heard
const double ConfidenceThreshold = 0.3;
string recognitionWord;

test.Content = Convert.ToString(e.Result.Text);
if (e.Result.Confidence >= ConfidenceThreshold)
{

recognitionWord = Convert.ToString(e.Result.Text);
switch (recognitionWord)
{
case "Inicio":

```

```

menu.ShowDialog();
this.Close();
//txmesagge.Content = recognitionWord;
break;
case "inició":
menu.ShowDialog();
this.Close();
//txmesagge.Content = recognitionWord;
break;
case "inicia":
menu.ShowDialog();
this.Close();
//txmesagge.Content = recognitionWord;
break;
case "Inicio":
menu.ShowDialog();
this.Close();
//txmesagge.Content = recognitionWord;
break;
case "iniciar":
menu.ShowDialog();
this.Close();
//txmesagge.Content = recognitionWord;
break;
case "inicio":
menu.ShowDialog();
this.Close();
//txmesagge.Content = recognitionWord;
break;
case "Inicia":
menu.ShowDialog();
this.Close();
//txmesagge.Content = recognitionWord;
break;
case "iniciara":
menu.ShowDialog();
this.Close();
//txmesagge.Content = recognitionWord;
break;
case "salir":
Application.Current.Shutdown();
break;

} }
}

private static RecognizerInfo GetKinectRecognizer()
{
IEnumerable<RecognizerInfo> recognizers;
try
{
//Verifica speech instalado
recognizers = SpeechRecognitionEngine.InstalledRecognizers();
}
catch (COMException)
{
return null;
}
foreach (RecognizerInfo recognizer in
SpeechRecognitionEngine.InstalledRecognizers())

```

```

{
string value;
recognizer.AdditionalInfo.TryGetValue("Kinect", out value); // especificar
de donde obtiene el audio
return recognizer;
}

return null;
}
private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
//Si la variable sensor no es nula, es decir que ya fue inicializada,
procederemos a detener la entrada de datos de audio y detener el
dispositivo
/*if (null != this.convertStream)
{
this.convertStream.SpeechActive = false;
}*/

if (null != this.speechEngine)
{
this.speechEngine.SpeechRecognized -= this.SpeechRecognized;
this.speechEngine.RecognizeAsyncStop();
}
if (this._sensor != null)
{
this._sensor.Close();
this._sensor = null;
}
}

private void btnIngresar_Click(object sender, RoutedEventArgs e)
{
menu.ShowDialog();
this.Close();
}

private void btnSalir_Click(object sender, RoutedEventArgs e)
{
Application.Current.Shutdown();
} }
}

```

MENÚ PRINCIPAL

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

using System.Speech.Synthesis;
using System.Speech.Recognition;
using System.Speech.AudioFormat;
using System.Runtime.InteropServices;
using System.Windows.Threading;
using Microsoft.Kinect;

namespace KinectMove
{
    /// <summary>
    /// Lógica de interacción para MainWindow.xaml
    /// </summary>

    public partial class MainWindow : Window
    {
        KinectMoveLinkToSQLDataContext dc;

        mainCorporal corporalMenu;
        corporalschema corporal;
        coordinationArms arms;
        SpeechSynthesizer synthesizer1 = new SpeechSynthesizer();
        public bool firstTime = true;
        KinectSensor _sensor;
        SpeechRecognitionEngine speechEngine = null;
        public string recognitionWord;
        string name, year, gender;
        public int c = 0;
        public int count = 1, nextCount = 0;
        //public int ok = 0, incorrect = 0;
        bool existCount;
        public bool allError, returned = false;

        public int time = 2;
        public DispatcherTimer Timer, Timer1, Timer2;
        public bool on = true, ismain = true;
        public string year1, gender1;
        public MainWindow()
        {

            InitializeComponent();
```

```

dc = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);

//Cuando la base este vacia
var count = dc.Registro.OrderByDescending(w => w.Numero).FirstOrDefault();
if (count == null)
{
// si es nueva base solo instancia la tabla
existCount = false;

Registro newRegistro = new Registro();

}
else
{
//Si ya existen registros toma el ultimo valor de la columna registros
nextCount = count.Numero;
}

}
public void Timer_Tick(object sender, EventArgs e)
{
if (time >= 0)
{
on = false;
SpeechSynthesizer synthesizerConteo = new SpeechSynthesizer();
synthesizerConteo.SelectVoice("Microsoft Sabina Desktop");
synthesizerConteo.Volume = 100; // 0...100
synthesizerConteo.Rate = -4; // -10...10
//synthesizerConteo.SpeakAsync(time.ToString());
time--;
}
else
{
//para definir lapso de nueva palabra
if (Timer1 != null)
{
Timer.Stop();
on = true;
}
if (Timer1 != null)
{
Timer1.Stop();
on = true;
}
if (Timer2 != null)
{
Timer2.Stop();
on = true;
} }
}

private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
if (null != this.speechEngine)
{

```



```

this.speechEngine.SpeechRecognized -= this.SpeechRecognized;
this.speechEngine.RecognizeAsyncStop();
}
if (this._sensor != null)
{
this._sensor.Close();
this._sensor = null;
}
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
//edad, sexo, nombre
if (firstTime)
{
synthesizer1.Volume = 100; // 0...100
synthesizer1.Rate = -4; // -10...10

synthesizer1.SelectVoice("Microsoft Sabina Desktop");
synthesizer1.Speak("Menú principal." +
"Por favor diga su nombre");
Timer = new DispatcherTimer();
Timer.Interval = new TimeSpan(0, 0, 1);
Timer.Tick += Timer_Tick;
Timer.Start();
}

_sensor = KinectSensor.Default();
_sensor.Open();
var audioBeamList = _sensor.AudioSource.AudioBeams;
var audioStream = audioBeamList[0].OpenInputStream();

RecognizerInfo ri = GetKinectRecognizer();
speechEngine = new SpeechRecognitionEngine(ri.Id);

// Create a grammar definition ...
speechEngine.LoadGrammar(new DictationGrammar());
speechEngine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(SpeechRecognized);
speechEngine.SetInputToDefaultAudioDevice();
speechEngine.RecognizeAsync(RecognizeMode.Multiple);
}
private void SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
// Speech utterance confidence below which we treat speech as if it hadn't
// been heard
const double ConfidenceThreshold = 0.3;

recognitionWord = Convert.ToString(e.Result.Text);
//aquí imprimir cadena nombreobjeto.content = recognitionWord;
if (c == 0 && firstTime && recognitionWord != "Y"
&& recognitionWord != "uno" && recognitionWord != "dos"
&& recognitionWord != "tres" && recognitionWord != "cuatro"
&& recognitionWord != "cinco" && recognitionWord != "salir"
&& recognitionWord != "reporte")//on: tiempo que tiene para decir nombre
edad sexo.
{
//Recepcion Nombre

```

```

name = recognitionWord;
dc = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
Registro newRegistro = new Registro();
var countRegister = dc.Registro.OrderByDescending(w =>
w.Numero).FirstOrDefault();// consulta a tabla registro para obtener ultimo
registro.
if (countRegister != null && countRegister.Numero != 0)
{
nextCount = countRegister.Numero + 1;
}
if (countRegister == null)
{
nextCount = count;
}
//asignando datos desde aplicacion a base de datos.
newRegistro.Numero = nextCount;
newRegistro.Nombre = name;
dc.Registro.InsertOnSubmit(newRegistro);
dc.SubmitChanges();//Grabar nuevo participante

recognitionWord = "";

synthesizer1.Volume = 100; // 0...100
synthesizer1.Rate = -4; // -10...10
synthesizer1.SelectVoice("Microsoft Sabina Desktop");
synthesizer1.Speak("Por favor diga su edad");
Timer1 = new DispatcherTimer();
Timer1.Interval = new TimeSpan(0, 0, 1);
Timer1.Tick += Timer_Tick;
Timer1.Start();

c++;
}
if (on)
{
recognitionWord = Convert.ToString(e.Result.Text);//captura edad despues
que se cumple 2 segundos para que en esta variable no se llene con la
palabra reconocida anterior(nombre)
on = false;

}

if (c == 1 && firstTime && recognitionWord != ""
)
{
int caracteres = recognitionWord.Length;
if (caracteres >= 10)
{
year1 = recognitionWord.Substring(0, 9);
}

year = year1;
dc = new KinectMoveLinkToSQLDataContext();
var countRegister = dc.Registro.Where(w => w.Numero ==
nextCount).FirstOrDefault();//seleccionar registro
countRegister.edad = year;
dc.SubmitChanges();
recognitionWord = "";

```

```

synthesizer1.Volume = 100; // 0...100
synthesizer1.Rate = -4; // -10...10
synthesizer1.SelectVoice("Microsoft Sabina Desktop");
synthesizer1.Speak("Por favor diga su sexo");

Timer2 = new DispatcherTimer();
Timer2.Interval = new TimeSpan(0, 0, 1);
Timer2.Tick += Timer_Tick;
Timer2.Start();
c++;
}

if (on)
{
recognitionWord = Convert.ToString(e.Result.Text);
on = false;
}

if (c == 2 && firstTime && recognitionWord != "" && recognitionWord != "uno"
&& recognitionWord != "dos"
&& recognitionWord != "tres" && recognitionWord != "cuatro"
&& recognitionWord != "cinco" && recognitionWord != "salir"
&& recognitionWord != "reporte")
{
int caracteres = recognitionWord.Length;
if (caracteres >= 10)
{
gender1 = recognitionWord.Substring(0, 9);
}

gender = gender1;
dc = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);

var countRegister = dc.Registro.Where(w => w.Numero ==
nextCount).FirstOrDefault();//seleccionar registro
countRegister.sexo = gender;

dc.SubmitChanges();
recognitionWord = "";
c++;
}
if (c == 3 && firstTime)
{
//synthesizer1.SpeakAsyncCancelAll();
SpeechSynthesizer synthesizerInicio = new SpeechSynthesizer();
synthesizerInicio.SelectVoice("Microsoft Sabina Desktop");
synthesizerInicio.SpeakAsync("Diga 1 si desea Actividades para desarrollar
conocimiento corporal." +
"....."
" +
"Diga 2 si desea Actividades para desarrollar la Coordinación de brazos" +
"....."
" +
"Diga 3 si desea Actividades para desarrollar la Coordinación de piernas" +
"....."
" +

```

```

"Diga 4 Actividad para desarrollar la locomoción" +
"....." +
"Diga 5 Actividad para desarrollar la posición" +
"....." +
"Diga salir si desea abandonar la aplicación");
C++;
}

//Acceso a ventanas
if (e.Result.Confidence >= ConfidenceThreshold)//Validar que reconocio una
palabra
{
reconoce.Content = Convert.ToString(e.Result.Text);
recognitionWord = Convert.ToString(e.Result.Text);// asigna en variable lo
que reconocio
if (ismain)
{
switch (recognitionWord)
{
case "uno":
if (c == 4)
{
//corporalschema corporalMenu = new corporalschema();
mainCorporal corporalMenu = new mainCorporal();
firTime = false;
ismain = false;
recognitionWord = "";
corporalMenu.Show();//.ShowDialog();
this.Close();
}

break;

case "un":
if (c == 4)
{
//corporalschema corporalMenu = new corporalschema();
mainCorporal corporalMenu = new mainCorporal();
firTime = false;
ismain = false;
recognitionWord = "";
corporalMenu.Show();
this.Close();
}

break;
case "dos":
if (c == 4)
{
//coordinationArms arm = new coordinationArms();
mainArms arm = new mainArms();
//c = 1;
firTime = false;
ismain = false;
recognitionWord = "";
arm.Show();
this.Close();
}
}
}

```

```

}
break;

case "tres":
if (c == 4)
{
coordinationLegs legs = new coordinationLegs();
//c = 1;
firstTime = false;
ismain = false;
recognitionWord = "";
legs.ShowDialog();
}
break;

case "iii":
if (c == 4)
{
coordinationLegs legs1 = new coordinationLegs();
firstTime = false;
ismain = false;
recognitionWord = "";
legs1.ShowDialog();
}
break;

case "cuatro":
if (c == 4)
{
locomotion loc = new locomotion();
firstTime = false;
ismain = false;
recognitionWord = "";
loc.ShowDialog();
}
break;

case "cuadro":
if (c == 4)
{
locomotion loc = new locomotion();
firstTime = false;
ismain = false;
recognitionWord = "";
loc.ShowDialog();
}

break;
case "iv":
if (c == 4)
{
locomotion loc1 = new locomotion();
firstTime = false;
ismain = false;
recognitionWord = "";
loc1.ShowDialog();
}
}

```

```

break;

case "cinco":
if (c == 4)
{
position pos = new position();
firstTime = false;
ismain = false;
recognitionWord = "";
pos.ShowDialog();
}

break;

case "v":
if (c == 4)
{
position pos1 = new position();
firstTime = false;
ismain = false;
recognitionWord = "";
pos1.ShowDialog();
}
break;

case "salir":
FormReporte rep = new FormReporte();
rep.Show();
SpeechSynthesizer msgsalir = new SpeechSynthesizer();
msgsalir.SelectVoice("Microsoft Sabina Desktop");
msgsalir.Speak("Fin de la aplicación");
//Application.Current.Shutdown();
break;
case "Salir":
FormReporte rep1 = new FormReporte();
rep1.Show();
SpeechSynthesizer msgsalir1 = new SpeechSynthesizer();
msgsalir1.SelectVoice("Microsoft Sabina Desktop");
msgsalir1.Speak("Fin de la aplicación");
Application.Current.Shutdown();
break;
case "seis":
if (c == 4)
{
SpeechSynthesizer synthesizerErrMsg = new SpeechSynthesizer();
synthesizerErrMsg.SelectVoice("Microsoft Sabina Desktop");
synthesizerErrMsg.Speak("Error, solo puede elegir opciones del uno al cinco");
}

break;
case "siete":
if (c == 4)
{
SpeechSynthesizer synthesizerErrMsg1 = new SpeechSynthesizer();
synthesizerErrMsg1.SelectVoice("Microsoft Sabina Desktop");
synthesizerErrMsg1.Speak("Error, solo puede elegir opciones del uno al cinco");
}
}

```

```

break;
case "ocho":
if (c == 4)
{
SpeechSynthesizer synthesizerErrMsg2 = new SpeechSynthesizer();
synthesizerErrMsg2.SelectVoice("Microsoft Sabina Desktop");
synthesizerErrMsg2.Speak("Error, solo puede elegir opciones del uno al
cinco");
}
break;
case "nueve":
if (c == 4)
{
SpeechSynthesizer synthesizerErrMsg3 = new SpeechSynthesizer();
synthesizerErrMsg3.SelectVoice("Microsoft Sabina Desktop");
synthesizerErrMsg3.Speak("Error, solo puede elegir opciones del uno al
cinco");
}
break;
case "diez":
if (c == 4)
{
SpeechSynthesizer synthesizerErrMsg4 = new SpeechSynthesizer();
synthesizerErrMsg4.SelectVoice("Microsoft Sabina Desktop");
synthesizerErrMsg4.Speak("Error, solo puede elegir opciones del uno al
cinco");
}
break;
} } }
}
private static RecognizerInfo GetKinectRecognizer()
{
IEnumerable<RecognizerInfo> recognizers;
try
{
recognizers = SpeechRecognitionEngine.InstalledRecognizers();
}
catch (COMException)
{
return null;
}
foreach (RecognizerInfo recognizer in
SpeechRecognitionEngine.InstalledRecognizers())
{
string value;
recognizer.AdditionalInfo.TryGetValue("Kinect", out value);
return recognizer;
}

return null;
}

private void btnCorporal_Click(object sender, RoutedEventArgs e)
{
//corporalschema corporalMenu = new corporalschema();
mainCorporal corporalMenu = new mainCorporal();
corporalMenu.Show();
this.Close();
synthesizer1.Dispose();
firTime = false;
}

```

```

}

private void btnBrazos_Click(object sender, RoutedEventArgs e)
{
    coordinationArms arms = new coordinationArms();
    //coordinationArms arm = new coordinationArms();
    arms.Show();
    synthesizer1.Dispose();
    firstime = false;
}

private void btnSalir_Click(object sender, RoutedEventArgs e)
{
    FormReporte rep = new FormReporte();
    rep.Show();
    SpeechSynthesizer msgsalir = new SpeechSynthesizer();
    msgsalir.SelectVoice("Microsoft Sabina Desktop");
    msgsalir.Speak("Fin de la aplicación");
    Application.Current.Shutdown();
}

private void btnPiernas_Click(object sender, RoutedEventArgs e)
{
    coordinationLegs legs = new coordinationLegs();
    firstime = false;
    ismain = false;
    recognitionWord = "";
    legs.ShowDialog();
}

private void DataKinect_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
}

private void btnLocomocion_Click(object sender, RoutedEventArgs e)
{
    locomotion loc = new locomotion();
    firstime = false;
    ismain = false;
    recognitionWord = "";
    loc.ShowDialog();
}

private void btnMenu_Click(object sender, RoutedEventArgs e)
{
    MainWindow m = new MainWindow();
    m.Show();
    this.Close();
    m.firstime = false;
}

private void Button_Click(object sender, RoutedEventArgs e)
{
    FormReporte rep = new FormReporte();
    rep.Show();
    c = 1;
    firstime = false;
}

```



```
}  
  
private void btnPosicion_Click(object sender, RoutedEventArgs e)  
{  
    position pos = new position();  
    firstTime = false;  
    ismain = false;  
    recognitionWord = "";  
    pos.ShowDialog();  
} }  
}
```

CONOCIMIENTO CORPORAL

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

using Microsoft.Kinect;
using System.Speech.Synthesis;
using System.Speech.Recognition;
using System.Speech.AudioFormat;
using System.Runtime.InteropServices;
using System.Windows.Threading;

namespace KinectMove
{
    /// <summary>
    /// Lógica de interacción para corporalschema.xaml
    /// </summary>
    public partial class corporalschema : Window
    {
        CameraMode _mode = CameraMode.Color;

        bool _displayBody = false;
        KinectMoveLinkToSQLDataContext db;
        KinectSensor _sensor;
        //MainWindow menu = new MainWindow();
        MultiSourceFrameReader _reader;
        IList<Body> _bodies;
        detected detected = new detected();
        public int answerNum = 1;
        int errorNum = 1;
        int errorNumCintura = 1;
        public int ok = 0, incorrect = 0, okcintu = 0 , incorrectcintu =0 ;
        MainWindow m;
        SpeechRecognitionEngine speechEngine = null;
        public bool answer = false;
        public string recognitionWord;
        public bool answerOn = true;
        public bool answerNumFlag = false;
        public bool bottonTwo = false;

        enum CameraMode
        {
            Color,
            Depth,
            Infrared
        }
    }
}
```

```

public struct position3D
{
public float X;
public float Y;
public float Z;
}

SpeechSynthesizer synthesizer1 = new SpeechSynthesizer();

public corporalSchema()
{
InitializeComponent();
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
synthesizer1.Volume = 100; // 0...100
synthesizer1.Rate = -4; // -10...10
synthesizer1.SpeakAsyncCancelAll();

//synthesizer1.Dispose();
_sensor = KinectSensor.GetDefault();

if (_sensor != null)
{
_sensor.Open();

//reconocimiento de voz

var audioBeamList = _sensor.AudioSource.AudioBeams;
var audioStream = audioBeamList[0].OpenInputStream();

RecognizerInfo ri = GetKinectRecognizer();
speechEngine = new SpeechRecognitionEngine(ri.Id);

// Create a grammar definition ...
speechEngine.LoadGrammar(new DictationGrammar());
speechEngine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(SpeechRecognized);
speechEngine.SetInputToDefaultAudioDevice();
speechEngine.RecognizeAsync(RecognizeMode.Multiple);
//fin reconocimiento de voz
synthesizer1.SelectVoice("Microsoft Sabina Desktop");
_reader = _sensor.OpenMultiSourceFrameReader(FrameSourceTypes.Color |
FrameSourceTypes.Depth | FrameSourceTypes.Infrared |
FrameSourceTypes.Body);
_reader.MultiSourceFrameArrived += Reader_MultiSourceFrameArrived;
}
}

private void SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
// Speech utterance confidence below which we treat speech as if it hadn't
been heard
const double ConfidenceThreshold = 0.3;

if (e.Result.Confidence >= ConfidenceThreshold)
{

recognitionWord = Convert.ToString(e.Result.Text);
}
}

```

```

//txtRecognition.Content = recognitionWord;//Imprime la palabra que
reconoce el sensor
switch (recognitionWord)
{
case "salir":
Application.Current.Shutdown();
answer = true;
break;

case "dos":
if (answerNum == 2)
{
answerNumFlag = true;
}
break;

case "Dos":
if (answerNum == 2)
{
answerNumFlag = true;
}
break;

} }
}

private static RecognizerInfo GetKinectRecognizer()
{
IEnumerable<RecognizerInfo> recognizers;
try
{
recognizers = SpeechRecognitionEngine.InstalledRecognizers();
}
catch (COMException)
{
return null;
}
foreach (RecognizerInfo recognizer in
SpeechRecognitionEngine.InstalledRecognizers())
{
string value;
recognizer.AdditionalInfo.TryGetValue("Kinect", out value);
return recognizer;
}
return null;
}
void Reader_MultiSourceFrameArrived(object sender,
MultiSourceFrameArrivedEventArgs e)
{
var reference = e.FrameReference.AcquireFrame();
// Color
using (var frame = reference.ColorFrameReference.AcquireFrame())
{
if (frame != null)
{
if (_mode == CameraMode.Color)
{
camera.Source = frame.ToBitmap();
} }
}
}
}

```

```

// Depth
using (var frame = reference.DepthFrameReference.AcquireFrame())
{
    if (frame != null)
    {
        if (_mode == CameraMode.Depth)
        {
            camera.Source = frame.ToBitmap();
        }
    }
}

// Infrared
using (var frame = reference.InfraredFrameReference.AcquireFrame())
{
    if (frame != null)
    {
        if (_mode == CameraMode.Infrared)
        {
            camera.Source = frame.ToBitmap();
        }
    }
}

//Body
//Obtenemos los datos del esqueleto
using (var frame = reference.BodyFrameReference.AcquireFrame())
{

    if (frame != null)
    {
        canvas.Children.Clear();

        _bodies = new Body[frame.BodyFrameSource.BodyCount];

        frame.GetAndRefreshBodyData(_bodies);

        foreach (var body in _bodies)//escaneamos todos los posibles esqueletos
        {

            if (body.IsTracked)//Verificamos si el esqueleto es correcto antes de
            realizar cualquier operacion
            { //ya que de lo contrario el sensor no podra trabajar con normalidad
            if (answerNum == 3)
            {
                answerNum++;
                //Actualizacion de aciertos en base de datos
                m = new MainWindow();
                db = new
                KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
                onnectionString);
                var updateOk = db.Registro.Where(w => w.Numero ==
                m.nextCount).FirstOrDefault();
                okcintu = okcintu + 1;
                updateOk.correctoscintu1 = okcintu;
                db.SubmitChanges();
                //
                synthesizer1.SelectVoice("Microsoft Sabina Desktop");
                synthesizer1.Speak("Actividad finalizada");
                FormReporte rep = new FormReporte();
                rep.Show();//llama al reporte
            }
            }
        }
    }
}

```

```

SpeechSynthesizer msgsalir = new SpeechSynthesizer();
msgsalir.SelectVoice("Microsoft Sabina Desktop");
msgsalir.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion
}

//cuando ya exedio el numero de intentos para la primera actividad
if (errorNum == 4)
{
m = new MainWindow();
//Actualización numero errores
db = new KinectMoveLinkToSQLDataContext();

var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();//seleccionar registro
incorrect = incorrect + 1;
updateError.incorrectos1 = incorrect;
db.SubmitChanges();

//
SpeechSynthesizer synthesizer1 = new SpeechSynthesizer();
synthesizer1.SelectVoice("Microsoft Sabina Desktop");
synthesizer1.Speak("Ah exedido el número de intentos.");
errorNum++;

FormReporte rep = new FormReporte();
rep.Show();//llama reporte
SpeechSynthesizer msgsalir = new SpeechSynthesizer();
msgsalir.SelectVoice("Microsoft Sabina Desktop");
msgsalir.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion
}

//cuando ya exedio el numero de intentos para la segunda actividad
if (errorNumCintura == 4)
{
errorNumCintura++;
//synthesizer1.SpeakAsyncCancelAll();
SpeechSynthesizer mesaggeExit = new SpeechSynthesizer();
m = new MainWindow();

//synthesizer1.SpeakAsyncCancelAll();

//Actualización numero errores
db = new KinectMoveLinkToSQLDataContext();

var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();//seleccionar registro
incorrectcintu = incorrectcintu + 1;
updateError.incorrectoscintu1 = incorrectcintu;
db.SubmitChanges();

//
mesaggeExit.SelectVoice("Microsoft Sabina Desktop");
mesaggeExit.Speak("Ah exedido el número de intentos");
}

FormReporte rep = new FormReporte();
rep.Show();//llama reporte

```

```

SpeechSynthesizer msgsalir = new SpeechSynthesizer();
msgsalir.SelectVoice("Microsoft Sabina Desktop");
msgsalir.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion

switch (errorNum)
{
case 1:
if ((answerOn && answerNum == 1) && bottonTwo == false)
{
synthesizer1.SpeakAsync("Para realizar este ejercicio usted tendrá tres
oportunidades:");
synthesizer1.Speak("Primer Intento, Toque su cabeza con la mano derecha");
answerOn = false;
}
break;

case 2:
if ((answerOn && answerNum == 1) && bottonTwo == false)
{
synthesizer1.Speak("Segundo Intento, Toque su cabeza con la mano derecha");
answerOn = false;
//Actualización numero errores
db = new KinectMoveLinkToSQLDataContext();
m = new MainWindow();
var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();//seleccionar registro
incorrect = incorrect + 1;
updateError.incorrectos1 = incorrect;
db.SubmitChanges();
//
}
break;
case 3:
//synthesizer1.SpeakAsyncCancelAll();
if ((answerOn && answerNum == 1) && bottonTwo == false)
{
synthesizer1.Speak("Tercer y último intento, Toque su cabeza con la mano
derecha");
answerOn = false;
//Actualización numero errores
db = new KinectMoveLinkToSQLDataContext();
m = new MainWindow();
var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();//seleccionar registro
incorrect = incorrect + 1;
updateError.incorrectos1 = incorrect;
db.SubmitChanges();
//
}
break;
}
switch (errorNumCintura)
{
case 1:
//synthesizer1.SpeakAsyncCancelAll();
if ((answerOn && answerNum == 2 && answerNumFlag) || bottonTwo && answerOn)
{
synthesizer1.Speak("Para realizar este ejercicio usted tendrá tres
oportunidades:");

```

```

synthesizer1.Speak("Primer Intento, Toque su cintura con la mano
izquierda");
answerOn = false;
}

break;

case 2:
//synthesizer1.SpeakAsyncCancelAll();
if ((answerOn && answerNum == 2 && answerNumFlag) || bottonTwo && answerOn)
{
synthesizer1.Speak("Segundo Intento, Toque su cintura con la mano
izquierda");
answerOn = false;
//Actualización numero errores
db = new KinectMoveLinkToSQLDataContext();
m = new MainWindow();
var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrectcintu = incorrectcintu + 1;
updateError.incorrectoscintu1 = incorrectcintu;
db.SubmitChanges();
//
}
break;
case 3:
//synthesizer1.SpeakAsyncCancelAll();
if ((answerOn && answerNum == 2 && answerNumFlag) || bottonTwo && answerNum
== 2)
{
synthesizer1.Speak("Tercer y último intento, Toque su cintura con la mano
izquierda");
answerOn = false;
//Actualización numero errores
db = new KinectMoveLinkToSQLDataContext();
m = new MainWindow();
var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrectcintu = incorrectcintu + 1;
updateError.incorrectoscintu1 = incorrectcintu;
db.SubmitChanges();
}
break;

}
//Obtener posiciones de las partes del cuerpo

position3D head = new position3D();
head.X = body.Joints[JointType.Head].Position.X;
head.Y = body.Joints[JointType.Head].Position.Y;
head.Z = body.Joints[JointType.Head].Position.Z;

position3D handRight = new position3D();
handRight.X = body.Joints[JointType.HandRight].Position.X;
handRight.Y = body.Joints[JointType.HandRight].Position.Y;
handRight.Z = body.Joints[JointType.HandRight].Position.Z;

position3D handLeft = new position3D();
handLeft.X = body.Joints[JointType.HandLeft].Position.X;

```



```

handLeft.Y = body.Joints[JointType.HandLeft].Position.Y;
handLeft.Z = body.Joints[JointType.HandLeft].Position.Z;

position3D hipLeft = new position3D();
hipLeft.X = body.Joints[JointType.HipLeft].Position.X;
hipLeft.Y = body.Joints[JointType.HipLeft].Position.Y;
hipLeft.Z = body.Joints[JointType.HipLeft].Position.Z;

position3D hipRight = new position3D();
hipRight.X = body.Joints[JointType.HipRight].Position.X;
hipRight.Y = body.Joints[JointType.HipRight].Position.Y;
hipRight.Z = body.Joints[JointType.HipRight].Position.Z;

position3D hipCenter = new position3D();
hipCenter.X = body.Joints[JointType.SpineMid].Position.X;
hipCenter.Y = body.Joints[JointType.SpineMid].Position.Y;
hipCenter.Z = body.Joints[JointType.SpineMid].Position.Z;

position3D shoulderLeft = new position3D();
shoulderLeft.X = body.Joints[JointType.ShoulderLeft].Position.X;
shoulderLeft.Y = body.Joints[JointType.ShoulderLeft].Position.Y;
shoulderLeft.Z = body.Joints[JointType.ShoulderLeft].Position.Z;

position3D shoulderRight = new position3D();
shoulderRight.X = body.Joints[JointType.ShoulderRight].Position.X;
shoulderRight.Y = body.Joints[JointType.ShoulderRight].Position.Y;
shoulderRight.Z = body.Joints[JointType.ShoulderRight].Position.Z;
if (errorNum <= 3)
{

if (answerNum == 1 && _sensor != null && bottonTwo == false)// &&
detected.accumulator != 0 )
{
//Detección de errores

//ManoIzquierda Cabeza
if (detected.HandLeftOnHead(head.X, head.Y, head.Z, handLeft.X, handLeft.Y,
handLeft.Z))
{
//this.txtcount.Content = "mano Izquierda Cab " + detected.izqCab;//imprime
contador
detected.PostureDetector(KinectMove.Posture.ManoIzquierdaCabeza);
if (detected.izqCab > 17)
{
SpeechSynthesizer synthesizetest1 = new SpeechSynthesizer();
synthesizetest1.SelectVoice("Microsoft Sabina Desktop");
detected.izqCab = 0;
synthesizetest1.Speak("postura incorrecta");
errorNum++;
answer = false;
answerOn = true;
}
}

//ManoIzquierda Hombro
if (detected.HandLeftOnShoulderL(handLeft.X, handLeft.Y, handLeft.Z,
shoulderLeft.X, shoulderLeft.Y, shoulderLeft.Z))
{
//this.txtcount.Content = "hombro Izquierdo " + detected.izqHomb;//imprime
contador

```

```

detected.PostureDetector(KinectMove.Posture.ManoIzquierdaHombro);
if (detected.izqHomb > 17)
{
SpeechSynthesizer synthesizetest1 = new SpeechSynthesizer();
synthesizetest1.SelectVoice("Microsoft Sabina Desktop");
synthesizetest1.Speak("postura incorrecta");
detected.izqHomb = 0;
errorNum++;
answer = false;
answerOn = true;
}
}

//ManoIzquierda Cintura
if (detected.HandOnHip(handLeft.X, handLeft.Y, handLeft.Z, hipLeft.X,
hipLeft.Y, hipLeft.Z))
{
//this.txtcount.Content = "mano IZq Cintura " + detected.izqCintu;
detected.PostureDetector(KinectMove.Posture.ManoIzquierdaCintura);
if (detected.izqCintu > 17)
{
SpeechSynthesizer synthesizetest1 = new SpeechSynthesizer();
synthesizetest1.SelectVoice("Microsoft Sabina Desktop");
synthesizetest1.Speak("postura incorrecta");
detected.izqCintu = 0;
errorNum++;
answer = false;
answerOn = true;
}
}

//ManoDerecha Hombro
if (detected.HandRightOnShoulderR(handRight.X, handRight.Y, handRight.Z,
shoulderRight.X, shoulderRight.Y, shoulderRight.Z))
{
//this.txtcount.Content = "hombro derecho " + detected.derHomb;//imprime
contador
detected.PostureDetector(KinectMove.Posture.ManoDerechaHombro);
if (detected.derHomb > 17)
{
SpeechSynthesizer synthesizetest1 = new SpeechSynthesizer();
synthesizetest1.SelectVoice("Microsoft Sabina Desktop");
synthesizetest1.Speak("postura incorrecta");
detected.derHomb = 0;
errorNum++;
answer = false;
answerOn = true;
}
}

//ManoDerecha Cintura
if (detected.HandRightOnHipR(handRight.X, handRight.Y, handRight.Z,
hipCenter.X, hipCenter.Y, hipCenter.Z))
{
//this.txtcount.Content = "mano derecha Cintura " +
detected.derCintu;//imprime contador
detected.PostureDetector(KinectMove.Posture.ManoDerechaCintura);
if (detected.derCintu > 17)
{
SpeechSynthesizer synthesizetest1 = new SpeechSynthesizer();

```

```

synthesizetest1.SelectVoice("Microsoft Sabina Desktop");
synthesizetest1.Speak("postura incorrecta");
detected.derCintu = 0;
errorNum++;
answer = false;
answerOn = true;
}
}
//Fin Posturas Incorrectas
//En caso que sea la postura correcta
//ManoDerecha Cabeza
if (detected.HandOnHead(head.X, head.Y, head.Z, handRight.X, handRight.Y,
handRight.Z))
{
//this.txtcount.Content = detected.derCab;//imprime contador
detected.PostureDetector(KinectMove.Posture.ManoDerechaCabeza);
if (detected.derCab > 17)
{
synthesizer1.SpeakAsyncCancelAll();
synthesizer1.SelectVoice("Microsoft Sabina Desktop");
synthesizer1.Speak("Correcto mano en cabeza");
detected.derCab = 0;
answerNum = answerNum + 1;
answer = false;

if (answerNum == 2)
{
//Actualizacion de aciertos en base de datos
m = new MainWindow();
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
var updateOk = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
ok = ok + 1;
updateOk.correctos1 = ok;
db.SubmitChanges();
//
synthesizer1.SelectVoice("Microsoft Sabina Desktop");
synthesizer1.SpeakAsync("Diga dos para continuar con el siguiente
ejercicio");
answerOn = true;
} }
} //fin postura correcta
}
}
if (errorNumCintura <= 3 && answerOn == false)
{
SpeechSynthesizer synthesizetest12 = new SpeechSynthesizer();
//synthesizetest12.Speak("mensaje: answerNum"+ answerNum + "answerFlag"+
answerNumFlag);

if ((answerNum == 2 && answerNumFlag) || bottonTwo && answerNum <= 3)
{
//POstura correcta ManoIzquierda Cintura
if (detected.HandOnHip(handLeft.X, handLeft.Y, handLeft.Z, hipLeft.X,
hipLeft.Y, hipLeft.Z))
{
//this.txtcount.Content = "mano IZq Cintura " + detected.izqCintu;//imprime
contador

```

```

detected.PostureDetector(KinectMove.Posture.ManoIzquierdaCintura);
if (detected.izqCintu > 17)
{
SpeechSynthesizer synthesizetest123 = new SpeechSynthesizer();
synthesizetest123.SelectVoice("Microsoft Sabina Desktop");
synthesizetest123.Speak("Correcto, Mano izquierda en cintura");
detected.izqCintu = 0;
answerOn = true;
if (bottonTwo)
{
answerNum = 3;
}
else
{
answerNum++;
} }
}

//Detección de errores
//ManoDerecha Cabeza
if (detected.HandOnHead(head.X, head.Y, head.Z, handRight.X, handRight.Y,
handRight.Z))
{
//this.txtcount.Content = detected.derCab;//imprime contador
detected.PostureDetector(KinectMove.Posture.ManoDerechaCabeza);
if (detected.derCab > 17)
{
//synthesizer1.SpeakAsyncCancelAll();
synthesizer1.SelectVoice("Microsoft Sabina Desktop");
synthesizer1.Speak("Incorrecto");
detected.derCab = 0;
errorNumCintura++;
answer = false;
answerOn = true;
}
}

//ManoIzquierda Cabeza
if (detected.HandLeftOnHead(head.X, head.Y, head.Z, handLeft.X, handLeft.Y,
handLeft.Z))
{
//this.txtcount.Content = "mano Izquierda Cab " + detected.izqCab;//imprime
contador
detected.PostureDetector(KinectMove.Posture.ManoIzquierdaCabeza);
if (detected.izqCab > 17)
{
SpeechSynthesizer synthesizetest1 = new SpeechSynthesizer();
synthesizetest1.SelectVoice("Microsoft Sabina Desktop");
synthesizetest1.Speak("postura incorrecta");
detected.izqCab = 0;
errorNumCintura++;
answer = false;
answerOn = true;
}
}

//ManoIzquierda Hombro

```

```

if (detected.HandLeftOnShoulderL(handLeft.X, handLeft.Y, handLeft.Z,
shoulderLeft.X, shoulderLeft.Y, shoulderLeft.Z))
{
//this.txtcount.Content = "hombro Izquierdo " + detected.izqHomb;//imprime
contador
detected.PostureDetector(KinectMove.Posture.ManoIzquierdaHombro);
if (detected.izqHomb > 17)
{
SpeechSynthesizer synthesizetest1 = new SpeechSynthesizer();
synthesizetest1.SelectVoice("Microsoft Sabina Desktop");
synthesizetest1.Speak("postura incorrecta");
detected.izqHomb = 0;
errorNumCintura++;
answer = false;
answerOn = true;
}
}

//ManoDerecha Hombro
if (detected.HandRightOnShoulderR(handRight.X, handRight.Y, handRight.Z,
shoulderRight.X, shoulderRight.Y, shoulderRight.Z))
{
//this.txtcount.Content = "hombro derecho " + detected.derHomb;//imprime
contador
detected.PostureDetector(KinectMove.Posture.ManoDerechaHombro);
if (detected.derHomb > 17)
{
SpeechSynthesizer synthesizetest1 = new SpeechSynthesizer();
synthesizetest1.SelectVoice("Microsoft Sabina Desktop");
synthesizetest1.Speak("postura incorrecta");
detected.derHomb = 0;
errorNumCintura++;
answer = false;
answerOn = true;
}
}

//ManoDerecha Cintura
if (detected.HandRightOnHipR(handRight.X, handRight.Y, handRight.Z,
hipCenter.X, hipCenter.Y, hipCenter.Z))
{
//this.txtcount.Content = "mano derecha Cintura " +
detected.derCintu;//imprime contador
detected.PostureDetector(KinectMove.Posture.ManoDerechaCintura);
if (detected.derCintu > 17)
{
SpeechSynthesizer synthesizetest1 = new SpeechSynthesizer();
synthesizetest1.SelectVoice("Microsoft Sabina Desktop");
synthesizetest1.Speak("postura incorrecta");
detected.derCintu = 0;
errorNumCintura++;
answer = false;
answerOn = true;
} } }
}
canvas.DrawSkeleton(body);

} } } }
}
private void Window_Closed(object sender, EventArgs e)

```

```

{
//Si la variable sensor no es nula, es decir que ya fue inicializada,
procederemos a detener la entrada de datos de audio y detener el
dispositivo
if (this._sensor != null)
{
this._sensor.Close();
this._sensor = null;
}
}

private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
//this.Hide();
if (this._sensor != null)
{
this._sensor.Close();
this._sensor = null;
}
//e.Cancel = true;
}

private void bntBody_Click(object sender, RoutedEventArgs e)
{
_displayBody = !_displayBody;
} }
}

```

COORDINACION DE BRAZOS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

using Microsoft.Kinect;
using System.Speech.Synthesis;
using System.Threading;
using System.Speech.Recognition;
using System.Speech.AudioFormat;
using System.Runtime.InteropServices;
using System.Windows.Threading;

namespace KinectMove
{
    /// <summary>
    /// Lógica de interacción para coordinationArms.xaml
    /// </summary>
    public partial class coordinationArms : Window
    {
        public struct position3D
        {
            public float X;
            public float Y;
            public float Z;
        }

        CameraMode _mode = CameraMode.Color;
        enum CameraMode
        {
            Color,
            Depth,
            Infrared
        }
        KinectMoveLinkToSQLDataContext db;
        KinectSensor _sensor;
        MultiSourceFrameReader _reader;
        IList<Body> _bodies;
        detected detected = new detected();
        Random r = new Random();
        int answerNum = 1, order = 0;
        int errorNum = 1;
        int errorNumSegundaActv = 1;
        int[] numeros = new int[6];
        int[] numeros2 = new int[6];
        SpeechRecognitionEngine speechEngine = null;
        public bool answer = false;
        public string recognitionWord;
```

```

public bool answerOn = true, answeOn2 = false;
public bool answerNumFlag = false;
public bool bottonTwo = false;
public int error, acierto;
public int ok = 0, incorrect = 0, okrut2 =0, incorrectrut2 =0;
MainWindow m;

SpeechSynthesizer synthesizer1 = new SpeechSynthesizer();

public coordinationArms()
{
this.DataContext = this;
InitializeComponent();
int i = 0, cp =0, cim=0, cp1 =0, cim1=0;
int j = 0;
while ( i < 6 )
{

numeros[i] = r.Next(1, 10);
if (numeros[i]%2 == 0)
{
cp++;
if (cp < 2)
{
i++;
}
else
{
cp = 0;
}
}
else
{
cim++;
if (cim < 2)
{
i++;
}
}
else
{
cim = 0;
} }
}
//

while ( j < 6 )
{

numeros2[j] = r.Next(1, 10);
if (numeros2[j]%2 == 0)
{
cp1++;
if (cp1 < 2)
{
j++;
}
}
else
{
cp1 = 0;
}
}
}

```



```

}
else
{
cim1++;
if (cim1 < 2)
{
j++;
}
else
{
cim1 = 0;
} }
}
for (int s = 0; s < 6; s++)
{
test.Items.Add(numeros[s]);
test1.Items.Add(numeros2[s]);
}
}

private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
//Si la variable sensor no es nula, es decir que ya fue inicializada,
procederemos a detener la entrada de datos de audio y detener el
dispositivo
if (this._sensor != null)
{
this._sensor.Close();
this._sensor = null;
}
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
synthesizer1.Volume = 100; // 0...100
synthesizer1.Rate = -2; // -10...10
synthesizer1.SpeakAsyncCancelAll();

//synthesizer1.Dispose();
_sensor = KinectSensor.GetDefault();

if (_sensor != null)
{
_sensor.Open();

//reconocimiento de voz

var audioBeamList = _sensor.AudioSource.AudioBeams;
var audioStream = audioBeamList[0].OpenInputStream();

RecognizerInfo ri = GetKinectRecognizer();
speechEngine = new SpeechRecognitionEngine(ri.Id);

// Create a grammar definition ...
speechEngine.LoadGrammar(new DictationGrammar());
speechEngine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(SpeechRecognized);
speechEngine.SetInputToDefaultAudioDevice();

```

```

speechEngine.RecognizeAsync(RecognizeMode.Multiple);
//fin reconocimiento de voz
synthesizer1.SelectVoice("Microsoft Sabina Desktop");

_reader = _sensor.OpenMultiSourceFrameReader(FrameSourceTypes.Color |
FrameSourceTypes.Depth | FrameSourceTypes.Infrared |
FrameSourceTypes.Body);
_reader.MultiSourceFrameArrived += Reader_MultiSourceFrameArrived;

}
}
private void SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
// Speech utterance confidence below which we treat speech as if it hadn't
been heard
const double ConfidenceThreshold = 0.3;

if (e.Result.Confidence >= ConfidenceThreshold)
{

recognitionWord = Convert.ToString(e.Result.Text);
this.texto.Content = recognitionWord;
switch (recognitionWord)
{

case "dos":
if (answerNum == 2)
{
answerNumFlag = true;
}
break;

case "Dos":
if (answerNum == 2)
{
answerNumFlag = true;
}
break;

this.Close();

break;

} }
}
private static RecognizerInfo GetKinectRecognizer()
{
IEnumerable<RecognizerInfo> recognizers;
try
{
recognizers = SpeechRecognitionEngine.InstalledRecognizers();
}
catch (COMException)
{
return null;
}
foreach (RecognizerInfo recognizer in
SpeechRecognitionEngine.InstalledRecognizers())
{

```

```

string value;
recognizer.AdditionalInfo.TryGetValue("Kinect", out value);
return recognizer;
}

return null;
}
void Reader_MultiSourceFrameArrived(object sender,
MultiSourceFrameArrivedEventArgs e)
{
var reference = e.FrameReference.AcquireFrame();

// Color
using (var frame = reference.ColorFrameReference.AcquireFrame())
{
if (frame != null)
{
if (_mode == CameraMode.Color)
{
camera.Source = frame.ToBitmap();
} }
}

// Depth
using (var frame = reference.DepthFrameReference.AcquireFrame())
{
if (frame != null)
{
if (_mode == CameraMode.Depth)
{
camera.Source = frame.ToBitmap();
} }
}

// Infrared
using (var frame = reference.InfraredFrameReference.AcquireFrame())
{
if (frame != null)
{
if (_mode == CameraMode.Infrared)
{
camera.Source = frame.ToBitmap();
} }
}

// Body
//Obtenemos los datos del esqueleto
using (var frame = reference.BodyFrameReference.AcquireFrame())
{
if (frame != null)
{
canvas.Children.Clear();

_bodies = new Body[frame.BodyFrameSource.BodyCount];

frame.GetAndRefreshBodyData(_bodies);

foreach (var body in _bodies)//escaneamos todos los posibles esqueletos
{

```

```

if (body.IsTracked)//Verificamos si el esqueleto es correcto antes de
realizar cualquier operacion ya que de lo contrario el sensor no podra
trabajar con normalidad
{
//Obtener posiciones de las partes del cuerpo

//obtenemos la posición con respecto al sensor. Con las posiciones de las
distintas partes podremos detectar posturas

position3D head = new position3D();
head.X = body.Joints[JointType.Head].Position.X;
head.Y = body.Joints[JointType.Head].Position.Y;
head.Z = body.Joints[JointType.Head].Position.Z;

position3D handLeft = new position3D();
handLeft.X = body.Joints[JointType.HandLeft].Position.X;
handLeft.Y = body.Joints[JointType.HandLeft].Position.Y;
handLeft.Z = body.Joints[JointType.HandLeft].Position.Z;

position3D shoulderLeft = new position3D();
shoulderLeft.X = body.Joints[JointType.ShoulderLeft].Position.X;
shoulderLeft.Y = body.Joints[JointType.ShoulderLeft].Position.Y;
shoulderLeft.Z = body.Joints[JointType.ShoulderLeft].Position.Z;

position3D handRight = new position3D();
//handRight.X = body.JointOrientations[JointType.HandRight].Orientation.X;
handRight.X = body.Joints[JointType.HandRight].Position.X;
handRight.Y = body.Joints[JointType.HandRight].Position.Y;
handRight.Z = body.Joints[JointType.HandRight].Position.Z;

position3D elbowRight = new position3D();
elbowRight.X = body.Joints[JointType.ElbowRight].Position.X;
elbowRight.Y = body.Joints[JointType.ElbowRight].Position.Y;
elbowRight.Z = body.Joints[JointType.ElbowRight].Position.Z;

position3D elbowLeft = new position3D();
elbowLeft.X = body.Joints[JointType.ElbowLeft].Position.X;
elbowLeft.Y = body.Joints[JointType.ElbowLeft].Position.Y;
elbowLeft.Z = body.Joints[JointType.ElbowLeft].Position.Z;

position3D shouldlerRight = new position3D();
shouldlerRight.X = body.Joints[JointType.ShoulderRight].Position.X;
shouldlerRight.Y = body.Joints[JointType.ShoulderRight].Position.Y;
shouldlerRight.Z = body.Joints[JointType.ShoulderRight].Position.Z;

if (answerNum == 2 && order == 6)
{
//synthesizer1.SpeakAsyncCancelAll();
SpeechSynthesizer segundaAct = new SpeechSynthesizer();
segundaAct.SelectVoice("Microsoft Sabina Desktop");
segundaAct.SpeakAsync("Correcto, Diga dos para continuar con el siguiente
ejercicio");
answeOn2 = true;
order = 0;
//Actualizacion de aciertos en base de datos
m = new MainWindow();
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);

```

```

var updateOk = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
ok = ok + 1;
updateOk.correctos2 = ok;
db.SubmitChanges();
}

if (answerNum == 3 && order == 6)
{
answeOn2 = true;
order = 0;
//Actualizacion de aciertos en base de datos
m = new MainWindow();
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
var updateOk = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
okrut2 = okrut2 + 1;
updateOk.correctosrutuna2 = okrut2;
db.SubmitChanges();
//
SpeechSynthesizer synthFin = new SpeechSynthesizer();
synthFin.SelectVoice("Microsoft Sabina Desktop");
synthFin.Speak("Correcto, Actividad finalizada");

FormReporte rep = new FormReporte();
rep.Show();//llama reporte
SpeechSynthesizer msgsalir = new SpeechSynthesizer();
msgsalir.SelectVoice("Microsoft Sabina Desktop");
msgsalir.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion
}
if (errorNum == 4)
{
errorNum++;
//synthesizer1.SpeakAsyncCancelAll();

//synthesizer1.SpeakAsyncCancelAll();
MainWindow m = new MainWindow();

//Actualización errores en base de datos
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
m.allError = true;
var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrect = incorrect + 1;
updateError.incorrectos2 = incorrect.ToString();
db.SubmitChanges();
//

synthesizer1 = new SpeechSynthesizer();
synthesizer1.SelectVoice("Microsoft Sabina Desktop");
synthesizer1.Speak("Ah exedido el número de intentos.");

FormReporte rep = new FormReporte();
rep.Show();//llama reporte
SpeechSynthesizer msgsalir = new SpeechSynthesizer();

```

```

msgsalir.SelectVoice("Microsoft Sabina Desktop");
msgsalir.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion
}
if (errorNumSegundaActv == 4)
{
    MainWindow m = new MainWindow();
    m.firsTime = false;
    m.allError = true;
    m.Show();
    db = new
    KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
    onnectionString);
    var updateError = db.Registro.Where(w => w.Numero ==
    m.nextCount).FirstOrDefault();
    incorrectrut2 = incorrectrut2 + 1;
    updateError.incorrectosrutina2 = incorrectrut2;
    db.SubmitChanges();
    //
    synthesizer1.Speak("Ah exedido el número de intentos.");
    FormReporte rep = new FormReporte();
    rep.Show();//llama reporte
    SpeechSynthesizer msgsalir = new SpeechSynthesizer();
    msgsalir.SelectVoice("Microsoft Sabina Desktop");
    msgsalir.Speak("Fin de la aplicación");
    Application.Current.Shutdown();//cerrar toda la aplicacion
}

switch (errorNum)
{
    case 1:
        SpeechSynthesizer synthesizer2 = new SpeechSynthesizer();
        synthesizer2.SelectVoice("Microsoft Sabina Desktop");
        if ((answerOn && answerNum == 1) && bottonTwo == false)// && order == 0)
        {

            synthesizer2.SpeakAsync("Para realizar este ejercicio usted tendrá tres
            oportuidades:");
            synthesizer2.Speak("Primer Intento, Usted deberá levantar el brazo
            izquierdo al escuchar un número impar" +
            "y el brazo derecho al escuchar un numero par. El primer número es:" +
            numeros[order].ToString());
            answerOn = false;
        }

        break;

    case 2:
        //synthesizer1.SpeakAsyncCancelAll();
        if ((answerOn && answerNum == 1) && bottonTwo == false)
        {
            SpeechSynthesizer synthesizer3 = new SpeechSynthesizer();
            synthesizer3.SelectVoice("Microsoft Sabina Desktop");
            synthesizer3.Speak("Segundo Intento, Usted deberá levantar el brazo
            izquierdo al escuchar un número impar" +
            "y el brazo derecho al escuchar un numero par. El primer número es:" +
            numeros[order].ToString());
            answerOn = false;
        }
}

```

```

//Actualizacion errores en base de datos
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
m = new MainWindow();

var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrect = incorrect + 1;
updateError.incorrectos2 = incorrect.ToString();
db.SubmitChanges();
//
}
break;
case 3:
//synthesizer1.SpeakAsyncCancelAll();
if ((answerOn && answerNum == 1) && bottonTwo == false)
{
SpeechSynthesizer synthesizer4 = new SpeechSynthesizer();
synthesizer4.SelectVoice("Microsoft Sabina Desktop");
synthesizer4.Speak("Tercer y último intento, Usted deberá levantar el brazo
izquierdo al escuchar un número impar" +
"y el brazo derecho al escuchar un numero par. El primer número es:" +
numeros[order].ToString());
answerOn = false;
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
m = new MainWindow();

var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrect = incorrect + 1;
updateError.incorrectos2 = incorrect.ToString();
db.SubmitChanges();
}
break;
}
if (answerNumFlag || bottonTwo)
{
switch (errorNumSegundaActv)
{
case 1:
//synthesizer1.SpeakAsyncCancelAll();
if ((answeOn2 && answerNum == 2) || bottonTwo)// && order == 0)
{
SpeechSynthesizer synthesizer2 = new SpeechSynthesizer();
synthesizer2.SelectVoice("Microsoft Sabina Desktop");
synthesizer2.SpeakAsync("Para realizar este ejercicio usted tendrá tres
oportunidades:");
synthesizer2.Speak("Primer Intento, Usted deberá levantar el brazo
izquierdo al escuchar un número impar" +
"y el brazo derecho al escuchar un numero par. El primer número es:" +
numeros2[order].ToString());
answeOn2 = false;
bottonTwo = false;
}
}
break;
}

```

```

case 2:
//synthesizer1.SpeakAsyncCancelAll();
if ((answeOn2 && answerNum == 2) || bottonTwo)
{
SpeechSynthesizer synthesizer3 = new SpeechSynthesizer();
synthesizer3.SelectVoice("Microsoft Sabina Desktop");
synthesizer3.Speak("Segundo Intento, Usted deberá levantar el brazo
izquierdo al escuchar un número impar" +
"y el brazo derecho al escuchar un numero par. El primer número es:" +
numeros2[order].ToString());
answeOn2 = false;
bottonTwo = false;

db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
m = new MainWindow();

var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrectrut2 = incorrectrut2 + 1;
updateError.incorrectosrutina2 = incorrectrut2;
db.SubmitChanges();
}
break;
case 3:
//synthesizer1.SpeakAsyncCancelAll();
if ((answeOn2 && answerNum == 2) || bottonTwo)
{
SpeechSynthesizer synthesizer4 = new SpeechSynthesizer();
synthesizer4.SelectVoice("Microsoft Sabina Desktop");
synthesizer4.Speak("Tercer y último intento, Usted deberá levantar el brazo
izquierdo al escuchar un número impar" +
"y el brazo derecho al escuchar un numero par. El primer número es:" +
numeros2[order].ToString());
answeOn2 = false;
bottonTwo = false;
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
m = new MainWindow();

var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrectrut2 = incorrectrut2 + 1;
updateError.incorrectosrutina2= incorrectrut2;
db.SubmitChanges();
}
break;
}
}

if (errorNum <= 3 && order <=5)
{
this.txterr.Content = errorNum;

if (answerNum == 1 && _sensor != null && order <=6)

{

```



```

if ((numeros[order] % 2) == 0) // si es un numero par
{
if (detected.HandRightUp(handRight.X, elbowRight.X, shouldlerRight.X,
head.Y, elbowRight.Y))
{
//this.txtcount.Content = detected.accumulator; //imprime contador
detected.PostureDetector(KinectMove.Posture.ManoDerechaArriba);
if (detected.accumulator > 7)
{

//synthesizer1.SpeakAsyncCancelAll();
SpeechSynthesizer synthesizerMesagge = new SpeechSynthesizer();
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");
if (order<=6)
{
order++;

if (order<=5)
{
SpeechSynthesizer synthesizerMesagge3 = new SpeechSynthesizer();
synthesizerMesagge3.SelectVoice("Microsoft Sabina Desktop");
synthesizerMesagge3.Speak("Correcto , levante el brazo que corresponde al
siguiente número. ");
synthesizerMesagge3.Speak(numeros[order].ToString());
detected.accumulator = 0;

//Actualizacion de aciertos en base de datos
m = new MainWindow();
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
var updateOk = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
ok = ok + 1;
updateOk.correctos2 = ok;
db.SubmitChanges();
}
else
{
answerNum++;
SpeechSynthesizer synthesizerMesagge2 = new SpeechSynthesizer();

} } }
}

if (detected.HandLefttUp(handLeft.X, elbowLeft.X, shoulderLeft.X, head.Y,
elbowLeft.Y))
{
//this.txtcount.Content = detected.accumulator; //imprime contador
detected.PostureDetector(KinectMove.Posture.ManoIzquierdaArriba);
if (detected.accumulator > 7)
{
SpeechSynthesizer synthesizerMesagge = new SpeechSynthesizer();
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");
synthesizerMesagge.Speak("Error se debia levantar el brazo derecho");
detected.accumulator = 0;
errorNum++;
answerOn = true;
} }

```

```

}
if (order<6 && ((numeros[order] % 2) != 0))
// si es un número impar

{

if (detected.HandLeftUp(handLeft.X, elbowLeft.X, shoulderLeft.X, head.Y,
elbowLeft.Y))// //Se evalua que la mano, el codo y el hombro estén
alineados
//en el eje de las X, y la cabeza con el codo en el eje de las Y
{
//this.txtcount.Content = detected.accumulator;//imprime contador
detected.PostureDetector(KinectMove.Posture.ManoIzquierdaArriba);
if (detected.accumulator > 7)
{
SpeechSynthesizer synthesizerMesagge = new SpeechSynthesizer();
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");

if (order <= 5)
{
order++;
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");

if (order<=5)
{
synthesizerMesagge.Speak("Correcto , levante el brazo que corresponde al
siguiente número.");
detected.accumulator = 0;

synthesizerMesagge.Speak(numeros[order].ToString());
}
else
{
synthesizerMesagge.Speak("Correcto ");
answerNum++;
}
}
//Actualizacion de aciertos en base de datos
m = new MainWindow();
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
var updateOk = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
ok = ok + 1;
updateOk.correctos2 = ok;
db.SubmitChanges();
} }
}

if (detected.HandRightUp(handRight.X, elbowRight.X, shouldlerRight.X,
head.Y, elbowRight.Y)) //Se evalua que la mano, el codo y el hombro estén
alineados
//en el eje de las X, y la cabeza con el codo en el eje de las Y
{
//this.txtcount.Content = detected.accumulator;//imprime contador
detected.PostureDetector(KinectMove.Posture.ManoDerechaArriba);
if (detected.accumulator >= 7)
{

```

```

SpeechSynthesizer synthesizerMesagge = new SpeechSynthesizer();
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");
synthesizerMesagge.Speak("Error se debia levantar el brazo IZQUIERDO");
detected.accumulator = 0;
errorNum++;
answerOn = true;

} } } }
}

if (errorNumSegundaActv <= 3 && order <= 5)
{
if ((answerNum == 2 && answerNumFlag) || bottonTwo)
{
if ((numeros2[order] % 2) != 0 && order <=5)// SI ES NUMERO IMPAR
{

if (detected.HandRightUp(handRight.X, elbowRight.X, shouldlerRight.X,
head.Y, elbowRight.Y)) //Se evalua que la mano, el codo y el hombro estén
alineados
//en el eje de las X, y la cabeza con el codo en el eje de las Y
{
//this.txtcount.Content = detected.accumulator;//imprime contador
detected.PostureDetector(KinectMove.Posture.ManoDerechaArriba);
if (detected.accumulator >= 7)
{
SpeechSynthesizer synthesizerMesagge = new SpeechSynthesizer();
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");
synthesizerMesagge.Speak("Error se debia levantar el brazo IZQUIERDO");
detected.accumulator = 0;
errorNumSegundaActv++;
answeOn2 = true;
}
}

if (detected.HandLefttUp(handLeft.X, elbowLeft.X, shoulderLeft.X, head.Y,
elbowLeft.Y))// //Se evalua que la mano, el codo y el hombro estén
alineados
//en el eje de las X, y la cabeza con el codo en el eje de las Y
{
//this.txtcount.Content = detected.accumulator;//imprime contador
detected.PostureDetector(KinectMove.Posture.ManoIzquierdaArriba);
if (detected.accumulator > 7)
{
SpeechSynthesizer synthesizerMesagge = new SpeechSynthesizer();
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");
if (order <= 6)
{
order++;

if (order <= 5)
{

SpeechSynthesizer synthesizerMesagge3 = new SpeechSynthesizer();
synthesizerMesagge3.SelectVoice("Microsoft Sabina Desktop");
synthesizerMesagge3.Speak("Correcto , levante el brazo que corresponde al
siguiente número.");
synthesizerMesagge3.Speak(numeros2[order].ToString());

detected.accumulator = 0;

```

```

//Actualizacion de aciertos en base de datos
m = new MainWindow();
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
var updateOk = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
okrut2 = okrut2 + 1;
updateOk.correctosrutuna2 = okrut2;
db.SubmitChanges();
}
else
{
answerNum++;
SpeechSynthesizer synthesizerMesagge2 = new SpeechSynthesizer();
} } } }
}

if (order <=5 && (numeros2[order] % 2) == 0)// SI ES NUMERO PAR
{
if (detected.HandLeftUp(handLeft.X, elbowLeft.X, shoulderLeft.X, head.Y,
elbowLeft.Y))
{
//this.txtcount.Content = detected.accumulator;//imprime contador
detected.PostureDetector(KinectMove.Posture.ManoIzquierdaArriba);
if (detected.accumulator > 7)
{
SpeechSynthesizer synthesizerMesagge = new SpeechSynthesizer();
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");
synthesizerMesagge.Speak("Error se debia levantar el brazo derecho");
detected.accumulator = 0;
errorNumSegundaActv++;
answeOn2 = true;
}
}
if (detected.HandRightUp(handRight.X, elbowRight.X, shouldlerRight.X,
head.Y, elbowRight.Y))
{
//this.txtcount.Content = detected.accumulator;//imprime contador
detected.PostureDetector(KinectMove.Posture.ManoDerechaArriba);
if (detected.accumulator > 7)
{
//synthesizer1.SpeakAsyncCancelAll();
SpeechSynthesizer synthesizerMesagge = new SpeechSynthesizer();
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");
if (order <= 6)
{
order++;
}
}
}

if (order <= 5)
{
SpeechSynthesizer synthesizerMesagge3 = new SpeechSynthesizer();
synthesizerMesagge3.SelectVoice("Microsoft Sabina Desktop");
synthesizerMesagge3.Speak("Correcto , levante el brazo que corresponde al
siguiente número.");
synthesizerMesagge3.Speak(numeros2[order].ToString());
detected.accumulator = 0;
//Actualizacion de aciertos en base de datos
m = new MainWindow();

```

```

db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
var updateOk = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
okrut2 = okrut2 + 1;
updateOk.correctosrutuna2 = okrut2;
db.SubmitChanges();
}
else
{
answerNum++;
SpeechSynthesizer synthesizerMesagge2 = new SpeechSynthesizer();

} } } } } }
}
//
canvas.DrawSkeleton(body);
//
} } } }
}

private void Window_Closed(object sender, EventArgs e)
{
//Si la variable sensor no es nula, es decir que ya fue inicializada,
procederemos a detener la entrada de datos de audio y detener el
dispositivo
if (this._sensor != null)
{
this._sensor.Close();
this._sensor = null;
}
}

private void btnSalir_Click(object sender, RoutedEventArgs e)
{
MainWindow m = new MainWindow();
m.Show();
this.Close();
answer = true;
} }
}

```

COORDINACION DE PIERNAS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

using Microsoft.Kinect;
using System.Speech.Synthesis;
using System.Threading;
using System.Speech.Recognition;
using System.Speech.AudioFormat;
using System.Runtime.InteropServices;
using System.Windows.Threading;

namespace KinectMove
{
    /// <summary>
    /// Lógica de interacción para mainLegs.xaml
    /// </summary>
    public partial class coordinationLegs : Window
    {
        CameraMode _mode = CameraMode.Color;
        public struct position3D
        {
            public float X;
            public float Y;
            public float Z;
        }

        KinectMoveLinkToSQLDataContext db;
        KinectSensor _sensor;
        MultiSourceFrameReader _reader;
        IList<Body> _bodies;
        detected detected = new detected();
        Random r = new Random();
        Random obj = new Random();
        string posibles = "IDIDIDIDIDID";

        char letra;
        int longitudnuevacadena = 6;
        public string nuevacadena = "";
        int answerNum = 1, order = 0;
        int errorNum = 1;
        int errorNumSegundaActv = 1;
        int[] numeros = new int[6];
        string[] letras = new string[6];
        SpeechRecognitionEngine speechEngine = null;
        public bool answer = false;
    }
}
```

```

public string recognitionWord;
public bool answerOn = true;
public bool answerNumFlag = false;
public bool bottonTwo = false;
SpeechSynthesizer synthesizer1 = new SpeechSynthesizer();
public int time = 16;
public DispatcherTimer Timer, Timer1, Timer2;
public bool go = false;
public int error, acierto;
public int ok = 0, incorrect = 0;
MainWindow m;
public bool actual = true;

enum CameraMode
{
    Color,
    Depth,
    Infrared
}
public coordinationLegs()
{

InitializeComponent();
int longitud = posibles.Length;
for (int i = 0; i < longitudnuevacadena; i++)
{
    letra = posibles[obj.Next(longitud)];
    nuevacadena += letra.ToString();
    letras[i] = letra.ToString();
}

}
public void Timer_Tick(object sender, EventArgs e)
{
    if (time >= 0)
    {
        go = false;
        SpeechSynthesizer synthesizerConteo = new SpeechSynthesizer();
        synthesizerConteo.SelectVoice("Microsoft Sabina Desktop");
        synthesizerConteo.Volume = 100; // 0...100
        synthesizerConteo.Rate = -4; // -10...10
        synthesizerConteo.SpeakAsync(time.ToString());
        time--;
    }
    else
    {
        Timer.Stop();
        go = true;
    }
}

private void Window_Closed(object sender, EventArgs e)
{

}

private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
    if (this._sensor != null)

```

```

{
this._sensor.Close();
this._sensor = null;
}
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
synthesizer1.Volume = 100; // 0...100
synthesizer1.Rate = -2; // -10...10
synthesizer1.SpeakAsyncCancelAll();

//synthesizer1.Dispose();
_sensor = KinectSensor.GetDefault();

if (_sensor != null)
{
_sensor.Open();

//reconocimiento de voz

var audioBeamList = _sensor.AudioSource.AudioBeams;
var audioStream = audioBeamList[0].OpenInputStream();

RecognizerInfo ri = GetKinectRecognizer();
speechEngine = new SpeechRecognitionEngine(ri.Id);

// Create a grammar definition ...
speechEngine.LoadGrammar(new DictationGrammar());
speechEngine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(SpeechRecognized);
speechEngine.SetInputToDefaultAudioDevice();
speechEngine.RecognizeAsync(RecognizeMode.Multiple);
//fin reconocimiento de voz
synthesizer1.SelectVoice("Microsoft Sabina Desktop");

_reader = _sensor.OpenMultiSourceFrameReader(FrameSourceTypes.Color |
FrameSourceTypes.Depth | FrameSourceTypes.Infrared |
FrameSourceTypes.Body);
_reader.MultiSourceFrameArrived += Reader_MultiSourceFrameArrived;
}
}
private void SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
// Speech utterance confidence below which we treat speech as if it hadn't
been heard
const double ConfidenceThreshold = 0.3;

if (e.Result.Confidence >= ConfidenceThreshold)
{

recognitionWord = Convert.ToString(e.Result.Text);
this.texto.Content = recognitionWord;
//txtRecognition.Content = recognitionWord;
switch (recognitionWord)
{
case "dos":
if (answerNum == 2)
{
answerNumFlag = true;

```



```

}
break;

} }
}
private static RecognizerInfo GetKinectRecognizer()
{
IEnumerable<RecognizerInfo> recognizers;
try
{
recognizers = SpeechRecognitionEngine.InstalledRecognizers();
}
catch (COMException)
{
return null;
}
foreach (RecognizerInfo recognizer in
SpeechRecognitionEngine.InstalledRecognizers())
{
string value;
recognizer.AdditionalInfo.TryGetValue("Kinect", out value);
return recognizer;
}

return null;
}

private void btnSalir_Click(object sender, RoutedEventArgs e)
{
MainWindow menu = new MainWindow();
menu.Show();
this.Close();
}
//

void Reader_MultiSourceFrameArrived(object sender,
MultiSourceFrameArrivedEventArgs e)
{
var reference = e.FrameReference.AcquireFrame();
bool dataReceived = false;
//interacion con extensions
// Color
using (var frame = reference.ColorFrameReference.AcquireFrame())
{
if (frame != null)
{
if (_mode == CameraMode.Color)
{
camera.Source = frame.ToBitmap();
} }
}

// Depth
using (var frame = reference.DepthFrameReference.AcquireFrame())
{
if (frame != null)
{
if (_mode == CameraMode.Depth)
{
camera.Source = frame.ToBitmap();
}
}
}
}

```

```

} }
}

// Infrared
using (var frame = reference.InfraredFrameReference.AcquireFrame())
{
if (frame != null)
{
if (_mode == CameraMode.Infrared)
{
camera.Source = frame.ToBitmap();
} }
}
////fin interacion con extensiones
// Body
//Obtenemos los datos del esqueleto
using (var frame = reference.BodyFrameReference.AcquireFrame())
{
if (frame != null)
{
canvas.Children.Clear();

_bodies = new Body[frame.BodyFrameSource.BodyCount];

frame.GetAndRefreshBodyData(_bodies);

foreach (var body in _bodies)//escaneamos todos los posibles esqueletos
{

if (body.IsTracked)//Verificamos si el esqueleto es correcto antes de
realizar cualquier operacion ya que de lo contrario el sensor no podra
trabajar con normalidad
{
//Obtener posiciones de las partes del cuerpo

//con respecto al sensor. Con las posiciones de las distintas partes
podremos detectar posturas

position3D hipLeft = new position3D();
hipLeft.X = body.Joints[JointType.HipLeft].Position.X;
hipLeft.Y = body.Joints[JointType.HipLeft].Position.Y;
hipLeft.Z = body.Joints[JointType.HipLeft].Position.Z;

position3D hipRight = new position3D();
hipRight.X = body.Joints[JointType.HipRight].Position.X;
hipRight.Y = body.Joints[JointType.HipRight].Position.Y;
hipRight.Z = body.Joints[JointType.HipRight].Position.Z;

position3D ankleLeft = new position3D();
ankleLeft.X = body.Joints[JointType.AnkleLeft].Position.X;
ankleLeft.Y = body.Joints[JointType.AnkleLeft].Position.Y;
ankleLeft.Z = body.Joints[JointType.AnkleLeft].Position.Z;

position3D ankleRight = new position3D();
ankleRight.X = body.Joints[JointType.AnkleRight].Position.X;
ankleRight.Y = body.Joints[JointType.AnkleRight].Position.Y;
ankleRight.Z = body.Joints[JointType.AnkleRight].Position.Z;

position3D kneeLeft = new position3D();
kneeLeft.X = body.Joints[JointType.KneeLeft].Position.X;

```

```

kneeLeft.Y = body.Joints[JointType.KneeLeft].Position.Y;
kneeLeft.Z = body.Joints[JointType.KneeLeft].Position.Z;

position3D kneeRight = new position3D();

kneeRight.X = body.Joints[JointType.KneeRight].Position.X;
kneeRight.Y = body.Joints[JointType.KneeRight].Position.Y;
kneeRight.Z = body.Joints[JointType.KneeRight].Position.Z;

position3D footLeft = new position3D();
footLeft.X = body.Joints[JointType.FootLeft].Position.X;
footLeft.Y = body.Joints[JointType.FootLeft].Position.Y;
footLeft.Z = body.Joints[JointType.FootLeft].Position.Z;

position3D footRight = new position3D();
footRight.X = body.Joints[JointType.FootRight].Position.X;
footRight.Y = body.Joints[JointType.FootRight].Position.Y;
footRight.Z = body.Joints[JointType.FootRight].Position.Z;

if (order == 6)
{
order++;
answerOn = true;
order = 0;

//Actualizacion de aciertos en base de datos
m = new MainWindow();
db = new KinectMoveLinkToSQLDataContext();
var updateOk = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
ok = ok + 1;
updateOk.correctos3 = ok;
db.SubmitChanges();
//
synthesizer1 = new SpeechSynthesizer();
synthesizer1.SelectVoice("Microsoft Sabina Desktop");
synthesizer1.Speak("Correcto, Actividad finalizada");
actual = false;
FormReporte rep = new FormReporte();
rep.Show();//llama reporte
SpeechSynthesizer msgsalir = new SpeechSynthesizer();
msgsalir.SelectVoice("Microsoft Sabina Desktop");
msgsalir.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion
}
if (errorNum == 4)
{

MainWindow m = new MainWindow();

//Actualización errores en base de datos
db = new KinectMoveLinkToSQLDataContext();
var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrect = incorrect + 1;
updateError.incorrectos3 = incorrect;
db.SubmitChanges();
//
synthesizer1.SelectVoice("Microsoft Sabina Desktop");

```

```

synthesizer1.Speak("Ah excedido el número de intentos.");
actual = false;

FormReporte rep = new FormReporte();
rep.Show();//llama reporte
SpeechSynthesizer msgsalir = new SpeechSynthesizer();
msgsalir.SelectVoice("Microsoft Sabina Desktop");
msgsalir.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion
}

switch (errorNum)
{
case 1:
SpeechSynthesizer synthesizer2 = new SpeechSynthesizer();
synthesizer2.SelectVoice("Microsoft Sabina Desktop");
synthesizer2.Rate = -4;
synthesizer2.SpeakAsyncCancelAll();
if ((answerOn && answerNum == 1) && bottonTwo == false && actual)
{

synthesizer2.SpeakAsync("Esta actividad le ayudara a desarrollar su
coordinación de piernas." +
"Para realizar este ejercicio usted tendrá tres oportunidades:");
synthesizer2.Speak("Primer Intento, Cuando escuche pierna izquierda," +
"tiene que levantar la pierna izquierda, i"+
"Cuando escuche pierna derecha." +
"tiene que levantar la pierna derecha" +
"La primera instrucción es:");
if (letras[0] == "D")
{
synthesizer2.Speak("Levante su pierna derecha");
}
else
{
synthesizer2.Speak("Levante su pierna izquierda");
}
answerOn = false;
}

break;

case 2:
//synthesizer1.SpeakAsyncCancelAll();
if ((answerOn && answerNum == 1) && bottonTwo == false)
{
SpeechSynthesizer synthesizer3 = new SpeechSynthesizer();
synthesizer3.Rate = -4;
synthesizer3.SelectVoice("Microsoft Sabina Desktop");
synthesizer3.Speak("Segundo Intento, Cuando escuche pierna izquierda," +
"tiene que levantar la pierna izquierda, i" +
"Cuando escuche pierna derecha." +
"tiene que levantar la pierna derecha" +
"La primera instrucción es:");
if (letras[order] == "D")
{
synthesizer3.Speak("Levante su pierna derecha");
}
else

```

```

{
synthesizer3.Speak("Levante su pierna izquierda");
}
answerOn = false;
//Actualización errores en base de datos
db = new KinectMoveLinkToSQLDataContext();
//m.allError = true;
var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrect = incorrect + 1;
updateError.incorrectos3 = incorrect;
db.SubmitChanges();
//
}
break;
case 3:
//synthesizer1.SpeakAsyncCancelAll();
if ((answerOn && answerNum == 1) && bottonTwo == false)
{
SpeechSynthesizer synthesizer4 = new SpeechSynthesizer();
synthesizer4.Rate = -4;
synthesizer4.SelectVoice("Microsoft Sabina Desktop");
synthesizer4.Speak("Tercer Intento, Cuando escuche pierna izquierda," +
"tiene que levantar la pierna izquierda, y" +
"cuando escuche pierna derecha." +
"tiene que levantar la pierna derecha" +
"La primera instrucción es:");
if (letras[order] == "D")
{
synthesizer4.Speak("Levante su pierna derecha");
}
else
{
synthesizer4.Speak("Levante su pierna izquierda");
}
answerOn = false;
//Actualización errores en base de datos
db = new KinectMoveLinkToSQLDataContext();
//m.allError = true;
var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrect = incorrect + 1;
updateError.incorrectos3 = incorrect;
db.SubmitChanges();
//
}
break;
}
if (errorNum <= 3)
{
//this.txterr.Content = errorNum;

if ( _sensor != null&& order <=5)

{

if (letras[order] == "D")
{
if (detected.rightLegUp(kneeLeft.X, ankleRight.X, kneeLeft.Y, ankleRight.Y,
kneeLeft.Z, ankleRight.Z))// //tobillo derecho alineado a rodilla izquierda

```

```

{
//this.txtcount.Content = detected.accumulator;//imprime contador
detected.PostureDetector(KinectMove.Posture.PiernaDerechaArriba);
if (detected.accumulator > 7)
{
SpeechSynthesizer synthesizerMesagge = new SpeechSynthesizer();
order++;
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");
synthesizerMesagge.Speak("Correcto, pierna derecha");
detected.accumulator = 0;
if (order < 6)
{
synthesizerMesagge.Speak("La siguiente instrucción es. ");
if (letras[order] == "D")
{
synthesizerMesagge.Speak("Levante su pierna derecha");
}
else
{
synthesizerMesagge.Speak("Levante su pierna izquierda");
}
}
}

//Actualizacion de aciertos en base de datos
m = new MainWindow();
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
var updateOk = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
ok = ok + 1;
updateOk.correctos3 = ok;
db.SubmitChanges();
//
}
}
if (detected.leftLegUp(kneeRight.X, ankleLeft.X, kneeRight.Y, ankleLeft.Y,
kneeRight.Z, ankleLeft.Z))// //tobillo derecho alineado a rodilla izquierda
{
//this.txtcount.Content = detected.accumulator;//imprime contador
detected.PostureDetector(KinectMove.Posture.PiernaIzquierdaArriba);
if (detected.accumulator > 7)
{
SpeechSynthesizer synthesizerMesagge = new SpeechSynthesizer();
errorNum++;
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");
synthesizerMesagge.Speak("Incorrecto, se debia levantar la pierna
derecha");
detected.accumulator = 0;
answerOn = true;
} }
}

if (order < 6 && letras[order] == "I")
{
if (detected.leftLegUp(kneeRight.X, ankleLeft.X, kneeRight.Y, ankleLeft.Y,
kneeRight.Z, ankleLeft.Z))// //tobillo derecho alineado a rodilla izquierda
{
//this.txtcount.Content = detected.accumulator;//imprime contador
detected.PostureDetector(KinectMove.Posture.PiernaIzquierdaArriba);

```

```

if (detected.accumulator > 7)
{
SpeechSynthesizer synthesizerMessage = new SpeechSynthesizer();
synthesizerMessage.SelectVoice("Microsoft Sabina Desktop");
order++;

synthesizerMessage.Speak("Correcto, pierna izquierda");
detected.accumulator=0;
synthesizerMessage.Speak("La siguiente instruccion es. ");
if (order<6 && letras[order]=="D")
{
synthesizerMessage.Speak("Levante su pierna Derecha");
}
else
{
synthesizerMessage.Speak("Levante su pierna izquierda");
}
//Actualizacion de aciertos en base de datos
m = new MainWindow();
db = new KinectMoveLinkToSQLDataContext();
var updateOk = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
ok = ok + 1;
updateOk.correctos3 = ok;
db.SubmitChanges();
//
}
}
if (detected.rightLegUp(kneeLeft.X, ankleRight.X, kneeLeft.Y, ankleRight.Y,
kneeLeft.Z, ankleRight.Z))// //tobillo derecho alineado a rodilla izquierda
{
//this.txtcount.Content = detected.accumulator;//imprime contador
detected.PostureDetector(KinectMove.Posture.PiernaDerechaArriba);
if (detected.accumulator > 7)
{
SpeechSynthesizer synthesizerMessage = new SpeechSynthesizer();
errorNum++;
synthesizerMessage.SelectVoice("Microsoft Sabina Desktop");
synthesizerMessage.Speak("Incorrecto, se debia levantar la pierna
izquierda");
detected.accumulator = 0;
answerOn = true;
} } } }
}
//
canvas.DrawSkeleton(body);//extensions
//
} }
} //Fin frame!null
} //Fin using
} //Fin multiSourceFrameArrived
}
}

```

DESARROLLO DE LOCOMOCION

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

using Microsoft.Kinect;
using System.Speech.Synthesis;
using System.Threading;
using System.Speech.Recognition;
using System.Speech.AudioFormat;
using System.Runtime.InteropServices;
using System.Windows.Threading;
namespace KinectMove
{
    /// <summary>
    /// Lógica de interacción para locomotion.xaml
    /// </summary>
    public partial class locomotion : Window
    {
        CameraMode _mode = CameraMode.Color;

enum CameraMode
{
    Color,
    Depth,
    Infrared
}
        public struct position3D
        {
            public float X;
            public float Y;
            public float Z;
        }

        KinectMoveLinkToSQLDataContext db;
        KinectSensor _sensor;
        MultiSourceFrameReader _reader;
        IList<Body> _bodies;
        detected detected = new detected();
        int answerNum = 1;
        int errorNum = 1, initial = 1;
        SpeechRecognitionEngine speechEngine = null;
        public bool answer = false;
        public string recognitionWord;
        public bool answerOn = true, answeOn2 = false;
        public bool answerNumFlag = false;
        public bool bottonTwo = false;
    }
}
```



```

public int error, acierto, countError = 1;
public int ok = 0, incorrect = 0;
MainWindow m;
public bool go = false, goes = false, pieFirmes = false, manoFirmes =
false;
public int time = 15;
public DispatcherTimer Timer, Timer1, Timer2;
public bool on = true;
public bool non = false;
public locomotion()
{

InitializeComponent();
}
public void Timer_Tick(object sender, EventArgs e)
{
if (time >= -5)
{
if (time == 0)
{
goes = true;
}
go = false;
SpeechSynthesizer synthesizerConteo = new SpeechSynthesizer();
synthesizerConteo.SelectVoice("Microsoft Sabina Desktop");
synthesizerConteo.Volume = 100; // 0...100
synthesizerConteo.Rate = -4; // -10...10
if (time >= 0)
{
synthesizerConteo.SpeakAsync(time.ToString());
}
time--;
if (time <= 0 && time >= -3)//
{
go = true;
}
if (time < -3 && time >= -5)
{
non = true;
go = false;
}
}
else
{

Timer.Stop();
}
}

private void Window_Closed(object sender, EventArgs e)
{
if (this._sensor != null)
{
this._sensor.Close();
this._sensor = null;
}
}

private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)

```

```

{
if (this._sensor != null)
{
this._sensor.Close();
this._sensor = null;
}
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
SpeechSynthesizer synthesizerLoc = new SpeechSynthesizer();
synthesizerLoc.SelectVoice("Microsoft Sabina Desktop");
synthesizerLoc.Volume = 100; // 0...100
synthesizerLoc.Rate = -4; // -10...10
synthesizerLoc.SpeakAsyncCancelAll();

_sensor = KinectSensor.Default();

if (_sensor != null)
{
_sensor.Open();

//reconocimiento de voz
var audioBeamList = _sensor.AudioSource.AudioBeams;
var audioStream = audioBeamList[0].OpenInputStream();

RecognizerInfo ri = GetKinectRecognizer();
speechEngine = new SpeechRecognitionEngine(ri.Id);

// Create a grammar definition ...
speechEngine.LoadGrammar(new DictationGrammar());
speechEngine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(SpeechRecognized);
speechEngine.SetInputToDefaultAudioDevice();
speechEngine.RecognizeAsync(RecognizeMode.Multiple);
//fin reconocimiento de voz
synthesizerLoc.SelectVoice("Microsoft Sabina Desktop");

_reader = _sensor.OpenMultiSourceFrameReader(FrameSourceTypes.Color |
FrameSourceTypes.Depth | FrameSourceTypes.Infrared |
FrameSourceTypes.Body);
_reader.MultiSourceFrameArrived += Reader_MultiSourceFrameArrived;

}
}

void Reader_MultiSourceFrameArrived(object sender,
MultiSourceFrameArrivedEventArgs e)
{
var reference = e.FrameReference.AcquireFrame();

// Color
using (var frame = reference.ColorFrameReference.AcquireFrame())
{
if (frame != null)
{
if (_mode == CameraMode.Color)
{
camera.Source = frame.ToBitmap();
} }
}
}
}

```

```

// Depth
using (var frame = reference.DepthFrameReference.AcquireFrame())
{
    if (frame != null)
    {
        if (_mode == CameraMode.Depth)
        {
            camera.Source = frame.ToBitmap();
        } }
    }

// Infrared
using (var frame = reference.InfraredFrameReference.AcquireFrame())
{
    if (frame != null)
    {
        if (_mode == CameraMode.Infrared)
        {
            camera.Source = frame.ToBitmap();
        } }
    }

// Body
//Obtenemos los datos del esqueleto
using (var frame = reference.BodyFrameReference.AcquireFrame())
{

    if (frame != null)
    {
        canvas.Children.Clear();

        _bodies = new Body[frame.BodyFrameSource.BodyCount];

        frame.GetAndRefreshBodyData(_bodies);

        foreach (var body in _bodies)//escaneamos todos los posibles esqueletos
        {

            if (body.IsTracked)//Verificamos si el esqueleto es correcto antes de
            realizar cualquier operacion ya que de lo contrario el sensor no podra
            trabajar con normalidad
            {
                //Obtener posiciones de las partes del cuerpo

                position3D hipLeft = new position3D();
                hipLeft.X = body.Joints[JointType.HipLeft].Position.X;
                hipLeft.Y = body.Joints[JointType.HipLeft].Position.Y;
                hipLeft.Z = body.Joints[JointType.HipLeft].Position.Z;

                position3D hipRight = new position3D();
                hipRight.X = body.Joints[JointType.HipRight].Position.X;
                hipRight.Y = body.Joints[JointType.HipRight].Position.Y;
                hipRight.Z = body.Joints[JointType.HipRight].Position.Z;

                position3D handLeft = new position3D();
                handLeft.X = body.Joints[JointType.HandLeft].Position.X;
                handLeft.Y = body.Joints[JointType.HandLeft].Position.Y;
                handLeft.Z = body.Joints[JointType.HandLeft].Position.Z;
            }
        }
    }
}

```

```

position3D footLeft = new position3D();
footLeft.X = body.Joints[JointType.FootLeft].Position.X;
footLeft.Y = body.Joints[JointType.FootLeft].Position.Y;
footLeft.Z = body.Joints[JointType.FootLeft].Position.Z;

position3D footRight = new position3D();
footRight.X = body.Joints[JointType.FootRight].Position.X;
footRight.Y = body.Joints[JointType.FootRight].Position.Y;
footRight.Z = body.Joints[JointType.FootRight].Position.Z;

position3D kneeLeft = new position3D();
kneeLeft.X = body.Joints[JointType.KneeLeft].Position.X;
kneeLeft.Y = body.Joints[JointType.KneeLeft].Position.Y;
kneeLeft.Z = body.Joints[JointType.KneeLeft].Position.Z;

position3D kneeRight = new position3D();

kneeRight.X = body.Joints[JointType.KneeRight].Position.X;
kneeRight.Y = body.Joints[JointType.KneeRight].Position.Y;
kneeRight.Z = body.Joints[JointType.KneeRight].Position.Z;

if (answerNum == 2)
{
//Actualizacion de aciertos en base de datos

m = new MainWindow();
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
var updateOk = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
ok = ok + 1;
updateOk.correctos4 = ok;
db.SubmitChanges();
//
SpeechSynthesizer fin = new SpeechSynthesizer();
fin.Volume = 100; // 0...100
fin.Rate = -4;
fin.SelectVoice("Microsoft Sabina Desktop");
fin.Speak("Actividad Finalizada");

answerNum++;

FormReporte rep = new FormReporte();
rep.Show();//llama reporte
SpeechSynthesizer msgsalir = new SpeechSynthesizer();
msgsalir.SelectVoice("Microsoft Sabina Desktop");
msgsalir.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion
}
if (errorNum == 4)
{
MainWindow m = new MainWindow();

//Actualizacion de errores en base de datos.
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);

```

```

var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrect = incorrect + 1;
updateError.incorrectos4 = incorrect;
db.SubmitChanges();
//
SpeechSynthesizer toterror = new SpeechSynthesizer();
toterror.Volume = 100; // 0...100
toterror.Rate = -4;
toterror.SelectVoice("Microsoft Sabina Desktop");
toterror.Speak("Ah exedido el número de intentos.");
errorNum++;

FormReporte rep = new FormReporte();
rep.Show();//llama reporte
SpeechSynthesizer msgsalir = new SpeechSynthesizer();
msgsalir.SelectVoice("Microsoft Sabina Desktop");
msgsalir.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion
}

switch (errorNum)
{
case 1:

if (answerOn && answerNum == 1)
{
if (initial == 1)
{
SpeechSynthesizer synthesizerPrim = new SpeechSynthesizer();
synthesizerPrim.SelectVoice("Microsoft Sabina Desktop");
synthesizerPrim.Speak("Primer Intento, Al escuchar el conteo regresivo,
debe desplazarse hacia la " +
"izquierda y derecha. Cuando el conteo llegue a cero debe unir sus pies");
initial++;
Timer = new DispatcherTimer();
Timer.Interval = new TimeSpan(0, 0, 1);
Timer.Tick += Timer_Tick;
Timer.Start();
answerOn = false;

}
}

break;

case 2:
//synthesizer1.SpeakAsyncCancelAll();
if (answerOn && answerNum == 1)
{
if (initial == 1)
{
SpeechSynthesizer synthesizerPrim = new SpeechSynthesizer();
synthesizerPrim.SelectVoice("Microsoft Sabina Desktop");
synthesizerPrim.Speak("Segundo Intento, Al escuchar el conteo regresivo,
debe desplazarse hacia la " +
"izquierda y derecha. Cuando el conteo llegue a cero debe unir sus pies");
initial++;
Timer1 = new DispatcherTimer();
Timer1.Interval = new TimeSpan(0, 0, 1);
}
}
}

```

```

Timer1.Tick += Timer_Tick;
Timer1.Start();
on = true;
non = false;
countError = 0;
//Actualizacion errores en base de datos
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
m = new MainWindow();
var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrect = incorrect + 1;
updateError.incorrectos4 = incorrect;
db.SubmitChanges();
answerOn = false;
}

}
break;

case 3:
//synthesizer1.SpeakAsyncCancelAll();
if (answerOn && answerNum == 1)
{

if (initial == 1)
{
SpeechSynthesizer synthesizerPrim = new SpeechSynthesizer();
synthesizerPrim.SelectVoice("Microsoft Sabina Desktop");
synthesizerPrim.Speak("Tercer y último Intento, Al escuchar el conteo
regresivo, debe desplazarse hacia la " +
"izquierda y derecha. Cuando el conteo llegue a cero debe unir sus dos
pies");
initial++;
Timer2 = new DispatcherTimer();
Timer2.Interval = new TimeSpan(0, 0, 1);
Timer2.Tick += Timer_Tick;
Timer2.Start();
on = true;
non = false;
countError = 0;
//Actualizacion errores en base de datos
db = new
KinectMoveLinkToSQLDataContext();//(Properties.Settings.Default.KinectMoveC
onnectionString);
m = new MainWindow();

var updateError = db.Registro.Where(w => w.Numero ==
m.nextCount).FirstOrDefault();
incorrect = incorrect + 1;
updateError.incorrectos4 = incorrect;
db.SubmitChanges();
//
answerOn = false;
}
}
break;

}

```

```

pies.Content = detected.piesFirmes;
if (errorNum <= 3 && answerNum == 1)
{
if (_sensor != null)
{

if (go)// && time>=-1)
{

pies.Content = detected.piesFirmes;
//count.Content = detected.accumulatorError;
//pies unidos
//detected.foots(kneeLeft.X, kneeLeft.Y, kneeLeft.Z, kneeRight.X,
kneeRight.Y, kneeRight.Z);
detected.foots(footLeft.X, footLeft.Y, footLeft.Z, footRight.X,
footRight.Y, footRight.Z);
detected.PostureDetector(KinectMove.Posture.PiesUnidos);
//if (detected.piesFirmes == 17)
if (detected.piesFirmes > 17)
{
SpeechSynthesizer synthesizerResp = new SpeechSynthesizer();
synthesizerResp.SelectVoice("Microsoft Sabina Desktop");
synthesizerResp.Speak("Correcto");
detected.piesFirmes = 0;
on = false;
answerNum++;
countError = 0;
//pieFirmes = true;
}
else
{
countError++;
}

}
//si va NON
if (non)// && time>=-1)
{
//pies unidos
//detected.foots(kneeLeft.X, kneeLeft.Y, kneeLeft.Z, kneeRight.X,
kneeRight.Y, kneeRight.Z);
detected.foots(footLeft.X, footLeft.Y, footLeft.Z, footRight.X,
footRight.Y, footRight.Z);
//if (detected.piesCero && on)
//{
detected.PostureDetector(KinectMove.Posture.PiesUnidos);
//if (detected.piesFirmes == 17)
if (detected.piesFirmes > 17)
{
SpeechSynthesizer synthesizerResp = new SpeechSynthesizer();
synthesizerResp.SelectVoice("Microsoft Sabina Desktop");
synthesizerResp.Speak("incorrecto");
if (Timer1 != null)
{
Timer1.Stop();
}
if (Timer2 != null)
{
Timer2.Stop();
}
}
}
}
}

```

```

if (Timer != null)
{
Timer.Stop();
}
detected.piesFirmes = 0;
on = false;
time = 15;
initial = 1;
errorNum++;
//pieFirmes = true;
answerOn = true;
}
if (detected.accumulatorError > 16)
{
SpeechSynthesizer synthesizerResp = new SpeechSynthesizer();
synthesizerResp.SelectVoice("Microsoft Sabina Desktop");
synthesizerResp.Speak("incorrecto");
if (Timer1 != null)
{
Timer1.Stop();
}
if (Timer2 != null)
{
Timer2.Stop();
}
if (Timer != null)
{
Timer.Stop();
}
detected.piesFirmes = 0;
on = false;
time = 15;
initial = 1;
errorNum++;
answerOn = true;
//pieFirmes = true;
}
}

//si va
if (go == false && time >= 0 && time < 13)
{

//pies unidos
detected.foots(footLeft.X, footLeft.Y, footLeft.Z, footRight.X,
footRight.Y, footRight.Z);
//if (detected.piesCero)
//{
detected.PostureDetector(KinectMove.Posture.PiesUnidos);
if (detected.piesFirmes > 17)
{
SpeechSynthesizer synthesizerResp = new SpeechSynthesizer();
synthesizerResp.SelectVoice("Microsoft Sabina Desktop");
if (errorNum >= 3)
{
synthesizerResp.Speak("Incorrecto");
detected.piesFirmes = 0;
}
}
else
{

```



```

synthesizerResp.Speak("Incorrecto");
detected.piesFirmes = 0;
}
if (Timer1 != null)
{
Timer1.Stop();
}
if (Timer2 != null)
{
Timer2.Stop();
}
if (Timer != null)
{
Timer.Stop();
}

time = 15;
initial = 1;
errorNum++;
answerOn = true;
} } }
}
canvas.DrawSkeleton(body);
} } } }
}
private void SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
// Speech utterance confidence below which we treat speech as if it hadn't
been heard
const double ConfidenceThreshold = 0.3;
if (e.Result.Confidence >= ConfidenceThreshold)
{

recognitionWord = Convert.ToString(e.Result.Text);
this.texto.Content = recognitionWord;
//txtRecognition.Content = recognitionWord;
switch (recognitionWord)
{
case "salir":
Application.Current.Shutdown();
break;

case "dos":
if (answerNum == 2)
{
answerNumFlag = true;
}
break;

} }
}

private static RecognizerInfo GetKinectRecognizer()
{
IEnumerable<RecognizerInfo> recognizers;
try
{
recognizers = SpeechRecognitionEngine.InstalledRecognizers();
}
catch (COMException)

```

```
{
return null;
}
foreach (RecognizerInfo recognizer in
SpeechRecognitionEngine.InstalledRecognizers())
{
string value;
recognizer.AdditionalInfo.TryGetValue("Kinect", out value);
return recognizer;
}
return null;

} }
}
```

IDENTIFICAR LA POSICIÓN

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

using Microsoft.Kinect;
using System.Speech.Synthesis;
using System.Threading;
using System.Speech.Recognition;
using System.Speech.AudioFormat;
using System.Runtime.InteropServices;

namespace KinectMove
{
    /// <summary>
    /// Lógica de interacción para position.xaml
    /// </summary>
    public partial class position : Window
    {
        CameraMode _mode = CameraMode.Color;

        enum CameraMode
        {
            Color,
            Depth,
            Infrared
        }

        public struct position3D
        {
            public float X;
            public float Y;
            public float Z;
        }

        KinectMoveLinkToSQLDataContext db;
        KinectSensor _sensor;
        MultiSourceFrameReader _reader;
        IList<Body> _bodies;
        detected detected = new detected();
        int answerNum = 1, order = 0;
        int errorNum = 1;
        SpeechRecognitionEngine speechEngine = null;
        public bool answer = false;
        public string recognitionWord;
        public bool answerOn = true, answeOn2 = false;
        public bool answerNumFlag = false;
        public bool bottonTwo = false;
    }
}
```

```

public int error, acierto;
public int ok = 0, incorrect = 0;
MainWindow m;
public position()
{

InitializeComponent();
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
SpeechSynthesizer synthesizerLoc = new SpeechSynthesizer();
synthesizerLoc.SelectVoice("Microsoft Sabina Desktop");
synthesizerLoc.Volume = 100; // 0...100
synthesizerLoc.Rate = -4; // -10...10
synthesizerLoc.SpeakAsyncCancelAll();

_sensor = KinectSensor.Default();

if (_sensor != null)
{
_sensor.Open();

//reconocimiento de voz

var audioBeamList = _sensor.AudioSource.AudioBeams;
var audioStream = audioBeamList[0].OpenInputStream();

RecognizerInfo ri = GetKinectRecognizer();
speechEngine = new SpeechRecognitionEngine(ri.Id);

// Create a grammar definition ...
speechEngine.LoadGrammar(new DictationGrammar());
speechEngine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(SpeechRecognized);
speechEngine.SetInputToDefaultAudioDevice();
speechEngine.RecognizeAsync(RecognizeMode.Multiple);
//fin reconocimiento de voz
synthesizerLoc.SelectVoice("Microsoft Sabina Desktop");

_reader = _sensor.OpenMultiSourceFrameReader(FrameSourceTypes.Color |
FrameSourceTypes.Depth | FrameSourceTypes.Infrared |
FrameSourceTypes.Body);
_reader.MultiSourceFrameArrived += Reader_MultiSourceFrameArrived;

}
}
private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
if (this._sensor != null)
{
this._sensor.Close();
this._sensor = null;
}
}
void Reader_MultiSourceFrameArrived(object sender,
MultiSourceFrameArrivedEventArgs e)
{

```

```

var reference = e.FrameReference.AcquireFrame();
bool dataReceived = false;

// Color
using (var frame = reference.ColorFrameReference.AcquireFrame())
{
    if (frame != null)
    {
        if (_mode == CameraMode.Color)
        {
            camera.Source = frame.ToBitmap();
        }
    }
}

// Depth
using (var frame = reference.DepthFrameReference.AcquireFrame())
{
    if (frame != null)
    {
        if (_mode == CameraMode.Depth)
        {
            camera.Source = frame.ToBitmap();
        }
    }
}

// Infrared
using (var frame = reference.InfraredFrameReference.AcquireFrame())
{
    if (frame != null)
    {
        if (_mode == CameraMode.Infrared)
        {
            camera.Source = frame.ToBitmap();
        }
    }
}

// Body
//Obtenemos los datos del esqueleto
using (var frame = reference.BodyFrameReference.AcquireFrame())
{
    if (frame != null)
    {
        canvas.Children.Clear();

        _bodies = new Body[frame.BodyFrameSource.BodyCount];

        frame.GetAndRefreshBodyData(_bodies);

        foreach (var body in _bodies)//escaneamos todos los posibles esqueletos
        {

            if (body.IsTracked)//Verificamos si el esqueleto es correcto antes de
            realizar cualquier operacion ya que de lo contrario el sensor no podra
            trabajar con normalidad
            {
                //Obtener posiciones de las partes del cuerpo

                position3D hipLeft = new position3D();
                hipLeft.X = body.Joints[JointType.HipLeft].Position.X;
            }
        }
    }
}

```

```

hipLeft.Y = body.Joints[JointType.HipLeft].Position.Y;
hipLeft.Z = body.Joints[JointType.HipLeft].Position.Z;

position3D hipRight = new position3D();
hipRight.X = body.Joints[JointType.HipRight].Position.X;
hipRight.Y = body.Joints[JointType.HipRight].Position.Y;
hipRight.Z = body.Joints[JointType.HipRight].Position.Z;

position3D kneeLeft = new position3D();
kneeLeft.X = body.Joints[JointType.KneeLeft].Position.X;
kneeLeft.Y = body.Joints[JointType.KneeLeft].Position.Y;
kneeLeft.Z = body.Joints[JointType.KneeLeft].Position.Z;

position3D kneeRight = new position3D();

kneeRight.X = body.Joints[JointType.KneeRight].Position.X;
kneeRight.Y = body.Joints[JointType.KneeRight].Position.Y;
kneeRight.Z = body.Joints[JointType.KneeRight].Position.Z;

switch (errorNum)
{
case 1:
SpeechSynthesizer synthesizer2 = new SpeechSynthesizer();
synthesizer2.Volume = 100; // 0...100
synthesizer2.Rate = -4;
synthesizer2.SelectVoice("Microsoft Sabina Desktop");
synthesizer2.Speak("Esta actividad le ayudara a identificar su posición, diga FIN cuando desee terminar esta actividad");
answerOn = false;
errorNum++;
break;
}

if (answerNum == 1 && _sensor != null)
{

if (detected.standUp(hipLeft.X, hipLeft.Y, hipLeft.Z, kneeLeft.X,
kneeLeft.Y, kneeLeft.Z))//,
{
detected.PostureDetector(KinectMove.Posture.Sentado);
if (detected.accumulator > 17)
{
SpeechSynthesizer synthesizerMesagge2 = new SpeechSynthesizer();
synthesizerMesagge2.SelectVoice("Microsoft Sabina Desktop");
synthesizerMesagge2.Volume = 100; // 0...100
synthesizerMesagge2.Rate = -4;
synthesizerMesagge2.Speak("sentado");
detected.accumulator = 0;
}
}
else
{
if (detected.accumulatorError > 17)
{
SpeechSynthesizer synthesizerMesagge = new SpeechSynthesizer();
synthesizerMesagge.SelectVoice("Microsoft Sabina Desktop");
synthesizerMesagge.Volume = 100; // 0...100
synthesizerMesagge.Rate = -4;
synthesizerMesagge.Speak("parado");
}
}
}
}

```

```

detected.accumulatorError = -50;
} }
}
//
canvas.DrawSkeleton(body);
//
} } } }
}

private void SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
// Speech utterance confidence below which we treat speech as if it hadn't
been heard
const double ConfidenceThreshold = 0.3;

if (e.Result.Confidence >= ConfidenceThreshold)
{

recognitionWord = Convert.ToString(e.Result.Text);
//txtRecognition.Content = recognitionWord;
switch (recognitionWord)
{

case "fin":

FormReporte rep = new FormReporte();
rep.Show();//llama reporte
SpeechSynthesizer msgsalir = new SpeechSynthesizer();
msgsalir.SelectVoice("Microsoft Sabina Desktop");
msgsalir.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion
break;

case "FIN":

//m.ShowDialog();
FormReporte rep1 = new FormReporte();
rep1.Show();//llama reporte
SpeechSynthesizer msgsalir1 = new SpeechSynthesizer();
msgsalir1.SelectVoice("Microsoft Sabina Desktop");
msgsalir1.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion
break;

case "final":

FormReporte rep2 = new FormReporte();
rep2.Show();//llama reporte
SpeechSynthesizer msgsalir2 = new SpeechSynthesizer();
msgsalir2.SelectVoice("Microsoft Sabina Desktop");
msgsalir2.Speak("Fin de la aplicación");
Application.Current.Shutdown();//cerrar toda la aplicacion
break;
} }
}
private static RecognizerInfo GetKinectRecognizer()
{
IEnumerable<RecognizerInfo> recognizers;
try

```

```

{
recognizers = SpeechRecognitionEngine.InstalledRecognizers();
}
catch (COMException)
{
return null;
}
foreach (RecognizerInfo recognizer in
SpeechRecognitionEngine.InstalledRecognizers())
{
string value;
recognizer.AdditionalInfo.TryGetValue("Kinect", out value);
return recognizer;
}

return null;
}

private void Window_Closed(object sender, EventArgs e)
{
if (this._sensor != null)
{
this._sensor.Close();
this._sensor = null;
} } }
}

```


ANEXO 2

CLASE DETECTED

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Microsoft.Kinect;

namespace KinectMove
{
    public enum Posture
    {
        Ninguna,
        ManoDerechaCabeza,
        ManoIzquierdaCabeza,
        ManoDerechaCintura,
        ManoDerechaHombro,
        ManoIzquierdaHombro,
        ManoIzquierdaCintura,
        ManoDerechaArriba,
        ManoIzquierdaArriba,
        PiernaIzquierdaArriba,
        PiernaDerechaArriba,
        Sentado,
        Parado,
        PiesUnidos,
        PiesSeparados,
        ManosUnidas
    }

    class detected
    {
        //corporalschema coporal = new corporalschema();
        const int PostureDetectionNumber = 18;
        public int accumulator = 0, accumulatorError = 0;
        public int izqCab, izqHomb, izqCintu = 0;
        public int derCab, derHomb, derCintu = 0;
        public int piesFirmes, manosFirmes, errorpiesFirmes = 0;
        Posture postureInDetection = Posture.Ninguna;
        Posture previousPosture = Posture.Ninguna;
        public bool handleft, handright, handHead, handHip, handHipR,
        detectionPosture, rightLeg, leftLeg, sitdown, stand, piesCero, pieslejos,
        manosCero = false;
        Random r = new Random();
        //int errorNum = 1;
        int[] numeros = new int[3];

        public bool PostureDetector(Posture posture)
        {
            {
                if (postureInDetection != posture)
                {
                    accumulator = 0;
                    accumulatorError = 0;
                }
            }
        }
    }
}
```

```

izqCab = 0;
izqHomb = 0;
izqCintu = 0;
derCab = 0;
derHomb = 0;
derCintu = 0;
manosFirmes = 0;
piesFirmes = 0;
errorpiesFirmes = 0;
//accumulatorError++;
postureInDetection = posture;
detectionPosture = false;
return false;
}
if (accumulator < PostureDetectionNumber)
{
piesFirmes++;
errorpiesFirmes++;
manosFirmes++;
accumulator++;
izqCab++;
izqHomb++;
izqCintu++;
derCab++;
derHomb++;
derCintu++;
//accumulatorError = 0;
detectionPosture = false;//test
//detectionPosture = true;
return false;
}
if (posture != previousPosture)
{
errorpiesFirmes++;
previousPosture = posture;
accumulator++;
piesFirmes++;
manosFirmes++;
izqCab++;
izqHomb++;
izqCintu++;
derCab++;
derHomb++;
derCintu++;
//accumulatorError = 0;
//accumulator = 0;
detectionPosture = true;
return true;
}
else
accumulator = 0;
izqCab = 0;
izqHomb = 0;
izqCintu = 0;
derCab = 0;
derHomb = 0;
derCintu = 0;
manosFirmes = 0;
accumulatorError = 0;
piesFirmes = 0;

```

```

errorpiesFirmes = 0;
detectionPosture = false;
return false;
//coporal.txtcount.Content = accumulator;
}

public bool PostureDetector2(Posture posture)
{
if (postureInDetection != posture)
{
accumulatorError = 0;
accumulator = 0;
izqCab = 0;
izqHomb = 0;
izqCintu = 0;
derCab = 0;
derHomb = 0;
derCintu = 0;
manosFirmes = 0;
piesFirmes = 0;
//accumulatorError++;
postureInDetection = posture;
detectionPosture = false;
return false;
}
if (accumulator < PostureDetectionNumber)
{
piesFirmes++;
manosFirmes++;
accumulator++;
izqCab++;
izqHomb++;
izqCintu++;
derCab++;
derHomb++;
derCintu++;
detectionPosture = false;//test
return false;
}
if (posture != previousPosture)
{
previousPosture = posture;
accumulator++;
piesFirmes++;
manosFirmes++;
izqCab++;
izqHomb++;
izqCintu++;
derCab++;
derHomb++;
derCintu++;
detectionPosture = true;
return true;
}
else
accumulator = 0;
izqCab = 0;
izqHomb = 0;
izqCintu = 0;
derCab = 0;

```

```

derHomb = 0;
derCintu = 0;
manosFirmes = 0;
piesFirmes = 0;
accumulatorError = 0;
detectionPosture = false;
return false;
}

public bool sit(float xd, float yd, float zd, float x1d, float y1d, float
z1d)
{
float distance = (xd - x1d);
if (Math.Abs(distance) > 0.07f)
{
accumulatorError++;
accumulator = 0;
return sitdown = false;
}

else
return sitdown = true;
}

public bool standUp(float x, float y, float z, float x1, float y1, float
z1)
{
float distance = (y - y1);
if (Math.Abs(distance) > 0.1f)
{
accumulatorError++;
accumulator = 0;
return stand = false;
}

else
return stand = true;
}

public bool foots(float x, float y, float z, float x1, float y1, float z1)
{
float distance = (y1 - y) + (z1 - z);
if (Math.Abs(distance) > 0.02f)
{
accumulatorError++;
piesFirmes = 0;
return piesCero = false;
}
else
return piesCero = true;
}

public bool errorfoots(float x, float y, float z, float x1, float y1, float
z1)
{
float distance = (x - x1);
if (Math.Abs(distance) > 0.05f)
{
accumulatorError++;
errorpiesFirmes = 0;
return pieslejos = false;
}
}

```

```

else
return pieslejos = true;
}
public bool hands(float x, float y, float z, float x1, float y1, float z1)
{
float distance = (x1 - x) + (y1 - y) + (z1 - z);
if (Math.Abs(distance) > 0.07f)
{
accumulatorError++;
manosFirmes = 0;
return manosCero = false;
}
else
return manosCero = true;
}

float distanceHandOnHead(float x, float y, float z, float x1, float y1,
float z1)
{
if ((x < 0 && y < 0 && z < 0 && x1 < 0 && y1 < 0 && z1 < 0) || (x > 0 && y
> 0 && z > 0 && x1 > 0 && y1 > 0 && z1 > 0))
return Math.Abs((x1 - x) + (y1 - y) + (z1 - z));
else
return Math.Abs((x1 + x) + (y1 + y) + (z1 + z));
}
public bool HandOnHead(float x, float y, float z, float x1, float y1, float
z1)
{
float distance = (x1 - x) + (y1 - y) + (z1 - z);
if (Math.Abs(distance) > 0.06f)
{
derCab = 0;
return handHead = false;
}
else
return handHead = true;
}
public bool HandLeftOnHead(float x, float y, float z, float x1, float y1,
float z1)
{
float distance = (x1 - x) + (y1 - y) + (z1 - z);
if (Math.Abs(distance) > 0.08f)
{
izqCab = 0;
return false;
}
else
return true;
}

public bool HandLeftOnShoulderL(float x, float y, float z, float x1, float
y1, float z1)
{
float distance = (x1 - x) + (y1 - y) + (z1 - z);
if (Math.Abs(distance) > 0.07f)
{
izqHomb = 0;
return false;
}
}

```

```

else
return true;
}

public bool HandOnHip(float x, float y, float z, float x1, float y1, float
z1)
{
float distance = (x1 - x) + (y1 - y) + (z1 - z);
if (Math.Abs(distance) > 0.07f)
{
accumulatorError++;
izqCintu = 0;
return false;
}
else
return true;
}

public bool HandRightOnShoulderR(float x, float y, float z, float x1, float
y1, float z1)
{
float distance = (x1 - x) + (y1 - y) + (z1 - z);
if (Math.Abs(distance) > 0.03f)
{
derHomb = 0;
return false;
}

else
return true;
}

public bool HandRightOnHipR(float x, float y, float z, float x1, float y1,
float z1)
{
float distance = (x1 - x) + (y1 - y) + (z1 - z);
if (Math.Abs(distance) > 0.07f)
{
accumulatorError++;
derCintu = 0;
return handHipR = false;
}
else
return handHipR = true;
}
public bool HandRightUp(float x, float x1, float x3, float y, float y1)
{
if (distanceHandUp(x,x1,x3,y,y1)>0.02f && y-y1>0.02f)
{
handright = false;
return false;
}
else
{
handright = true;
return true;
}
}
}

```

```

public bool HandLefttUp(float x, float x1, float x3, float y, float y1)
{
if (distanceHandUp(x, x1, x3, y, y1) > 0.02f && y - y1 > 0.02f)
{
handleleft = false;
return false;

}
else
handleleft = true;
return true;

public bool rightLegUp(float x, float x1, float y, float y1, float z, float
z1)
{
if (Math.Abs((x - x1) + (z - z1)) > 0.05f)

{
rightLeg = false;

return false;

}
else
rightLeg = true;
return true;

}

public bool leftLegUp(float x, float x1, float y, float y1, float z, float
z1)
{
if (Math.Abs((x - x1) + (z1 - z)) > 0.05f)
{
leftLeg = false;
return false;

}
else
leftLeg = true;
return true;
}

float distanceLegUp( float y, float y1)
{
if ((y < 0 && y1 < 0) || (y > 0 && y1 > 0))
return Math.Abs(y - y1);
else
return Math.Abs(y + y1);
}

float distanceHandUp(float right, float left, float x3, float y, float y1)
{
if ((right < 0 && left < 0 && x3 < 0 && y < 0 && y1 < 0) || (right > 0 &&
left > 0 && x3 > 0 && y > 0 && y1 > 0))
return Math.Abs(right - left - x3);
else
return Math.Abs(right + left + x3);
} }
}

```

ANEXO 4

CLASE EXTENSIONS

```
using Microsoft.Kinect;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace KinectMove
{
    public static class Extensions
    {
        #region Camera

        public static ImageSource ToBitmap(this ColorFrame frame)
        {
            int width = frame.FrameDescription.Width;
            int height = frame.FrameDescription.Height;
            PixelFormat format = PixelFormats.Bgr32;

            byte[] pixels = new byte[width * height * ((format.BitsPerPixel + 7) / 8)];

            if (frame.RawColorImageFormat == ColorImageFormat.Bgra)
            {
                frame.CopyRawFrameDataToArray(pixels);
            }
            else
            {
                frame.CopyConvertedFrameDataToArray(pixels, ColorImageFormat.Bgra);
            }

            int stride = width * format.BitsPerPixel / 8;

            return BitmapSource.Create(width, height, 96, 96, format, null, pixels,
                stride);
        }

        public static ImageSource ToBitmap(this DepthFrame frame)
        {
            int width = frame.FrameDescription.Width;
            int height = frame.FrameDescription.Height;
            PixelFormat format = PixelFormats.Bgr32;

            ushort minDepth = frame.DepthMinReliableDistance;
            ushort maxDepth = frame.DepthMaxReliableDistance;

            ushort[] pixelData = new ushort[width * height];
            byte[] pixels = new byte[width * height * (format.BitsPerPixel + 7) / 8];

            frame.CopyFrameDataToArray(pixelData);

            int colorIndex = 0;
            for (int depthIndex = 0; depthIndex < pixelData.Length; ++depthIndex)
```



```

{
ushort depth = pixelData[depthIndex];

byte intensity = (byte)(depth >= minDepth && depth <= maxDepth ? depth :
0);

pixels[colorIndex++] = intensity; // Blue
pixels[colorIndex++] = intensity; // Green
pixels[colorIndex++] = intensity; // Red

++colorIndex;
}

int stride = width * format.BitsPerPixel / 8;

return BitmapSource.Create(width, height, 96, 96, format, null, pixels,
stride);
}

public static ImageSource ToBitmap(this InfraredFrame frame)
{
int width = frame.FrameDescription.Width;
int height = frame.FrameDescription.Height;
PixelFormat format = PixelFormats.Bgr32;

ushort[] frameData = new ushort[width * height];
byte[] pixels = new byte[width * height * (format.BitsPerPixel + 7) / 8];

frame.CopyFrameDataToArray(frameData);

int colorIndex = 0;
for (int infraredIndex = 0; infraredIndex < frameData.Length;
infraredIndex++)
{
ushort ir = frameData[infraredIndex];

byte intensity = (byte)(ir >> 7);

pixels[colorIndex++] = (byte)(intensity / 1); // Blue
pixels[colorIndex++] = (byte)(intensity / 1); // Green
pixels[colorIndex++] = (byte)(intensity / 0.4); // Red

colorIndex++;
}

int stride = width * format.BitsPerPixel / 8;

return BitmapSource.Create(width, height, 96, 96, format, null, pixels,
stride);
}

#endregion

#region Body

public static Joint ScaleTo(this Joint joint, double width, double height,
float skeletonMaxX, float skeletonMaxY)
{
joint.Position = new CameraSpacePoint
{

```

```

X = Scale(width, skeletonMaxX, joint.Position.X),
Y = Scale(height, skeletonMaxY, -joint.Position.Y),
Z = joint.Position.Z
};

return joint;
}

public static Joint ScaleTo(this Joint joint, double width, double height)
{
return ScaleTo(joint, width, height, 1.0f, 1.0f);
}

private static float Scale(double maxPixel, double maxSkeleton, float
position)
{
float value = (float)((((maxPixel / maxSkeleton) / 2) * position) +
(maxPixel / 2));

if (value > maxPixel)
{
return (float)maxPixel;
}

if (value < 0)
{
return 0;
}

return value;
}

#endregion

#region Drawing

public static void DrawSkeleton(this Canvas canvas, Body body)
{
if (body == null) return;

foreach (Joint joint in body.Joints.Values)
{
canvas.DrawPoint(joint);
}

canvas.DrawLine(body.Joints[JointType.Head], body.Joints[JointType.Neck]);
canvas.DrawLine(body.Joints[JointType.Neck],
body.Joints[JointType.SpineShoulder]);
canvas.DrawLine(body.Joints[JointType.SpineShoulder],
body.Joints[JointType.ShoulderLeft]);
canvas.DrawLine(body.Joints[JointType.SpineShoulder],
body.Joints[JointType.ShoulderRight]);
canvas.DrawLine(body.Joints[JointType.SpineShoulder],
body.Joints[JointType.SpineMid]);
canvas.DrawLine(body.Joints[JointType.ShoulderLeft],
body.Joints[JointType.ElbowLeft]);
canvas.DrawLine(body.Joints[JointType.ShoulderRight],
body.Joints[JointType.ElbowRight]);
canvas.DrawLine(body.Joints[JointType.ElbowLeft],
body.Joints[JointType.WristLeft]);
}
}

```

```

canvas.DrawLine(body.Joints[JointType.ElbowRight],
body.Joints[JointType.WristRight]);
canvas.DrawLine(body.Joints[JointType.WristLeft],
body.Joints[JointType.HandLeft]);
canvas.DrawLine(body.Joints[JointType.WristRight],
body.Joints[JointType.HandRight]);
canvas.DrawLine(body.Joints[JointType.HandLeft],
body.Joints[JointType.HandTipLeft]);
canvas.DrawLine(body.Joints[JointType.HandRight],
body.Joints[JointType.HandTipRight]);
canvas.DrawLine(body.Joints[JointType.HandTipLeft],
body.Joints[JointType.ThumbLeft]);
canvas.DrawLine(body.Joints[JointType.HandTipRight],
body.Joints[JointType.ThumbRight]);
canvas.DrawLine(body.Joints[JointType.SpineMid],
body.Joints[JointType.SpineBase]);
canvas.DrawLine(body.Joints[JointType.SpineBase],
body.Joints[JointType.HipLeft]);
canvas.DrawLine(body.Joints[JointType.SpineBase],
body.Joints[JointType.HipRight]);
canvas.DrawLine(body.Joints[JointType.HipLeft],
body.Joints[JointType.KneeLeft]);
canvas.DrawLine(body.Joints[JointType.HipRight],
body.Joints[JointType.KneeRight]);
canvas.DrawLine(body.Joints[JointType.KneeLeft],
body.Joints[JointType.AnkleLeft]);
canvas.DrawLine(body.Joints[JointType.KneeRight],
body.Joints[JointType.AnkleRight]);
canvas.DrawLine(body.Joints[JointType.AnkleLeft],
body.Joints[JointType.FootLeft]);
canvas.DrawLine(body.Joints[JointType.AnkleRight],
body.Joints[JointType.FootRight]);
}

public static void DrawPoint(this Canvas canvas, Joint joint)
{
if (joint.TrackingState == TrackingState.NotTracked) return;

joint = joint.ScaleTo(canvas.ActualWidth, canvas.ActualHeight);

Ellipse ellipse = new Ellipse
{
Width = 20,
Height = 20,
Fill = new SolidColorBrush(Colors.LightBlue)
};

Canvas.SetLeft(ellipse, joint.Position.X - ellipse.Width / 2);
Canvas.SetTop(ellipse, joint.Position.Y - ellipse.Height / 2);

canvas.Children.Add(ellipse);
}

public static void DrawLine(this Canvas canvas, Joint first, Joint second)
{
if (first.TrackingState == TrackingState.NotTracked || second.TrackingState
== TrackingState.NotTracked) return;

first = first.ScaleTo(canvas.ActualWidth, canvas.ActualHeight);
second = second.ScaleTo(canvas.ActualWidth, canvas.ActualHeight);

```

```
Line line = new Line
{
    X1 = first.Position.X,
    Y1 = first.Position.Y,
    X2 = second.Position.X,
    Y2 = second.Position.Y,
    StrokeThickness = 8,
    Stroke = new SolidColorBrush(Colors.LightBlue)
};

canvas.Children.Add(line);
}

#endregion
}
}
```

**ANEXO 4
TABLA CHI²**

Grados de libertad	Probabilidad										
	0,95	0,90	0,80	0,70	0,50	0,30	0,20	0,10	0,05	0,01	0,001
1	0,004	0,02	0,06	0,15	0,46	1,07	1,64	2,71	3,84	6,64	10,83
2	0,10	0,21	0,45	0,71	1,39	2,41	3,22	4,60	5,99	9,21	13,82
3	0,35	0,58	1,01	1,42	2,37	3,66	4,64	6,25	7,82	11,34	16,27
4	0,71	1,06	1,65	2,20	3,36	4,88	5,99	7,78	9,49	13,28	18,47
5	1,14	1,61	2,34	3,00	4,35	6,06	7,29	9,24	11,07	15,09	20,52
6	1,63	2,20	3,07	3,83	5,35	7,23	8,56	10,64	12,59	16,81	22,46
7	2,17	2,83	3,82	4,67	6,35	8,38	9,80	12,02	14,07	18,48	24,32
8	2,73	3,49	4,59	5,53	7,34	9,52	11,03	13,36	15,51	20,09	26,12
9	3,32	4,17	5,38	6,39	8,34	10,66	12,24	14,68	16,92	21,67	27,88
10	3,94	4,86	6,18	7,27	9,34	11,78	13,44	15,99	18,31	23,21	29,59
	No significativo								Significativo		

**ANEXO 5
ACTIVIDADES REALIZADAS CON KINECTMOVE**



Inicio y Selección de Actividades



Conocimiento Corporal - Mano derecha en Cabeza



Coordinación de Brazos – Numero par



Coordinación de Brazos – Número impar



Coordinación de Piernas - Pierna izquierda



Locomoción – Piernas juntas



Identificación de Posición – Parado



Identificación de Posición – Sentado



Actividades erróneas