UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

CONTROL DINÁMICO DE POWERPOINT MEDIANTE LA TECNOLOGÍA KINECT DE MICROSOFT

Reporte Técnico de Investigación presentado por:

José Juan Reyes Arzola 94633

Tania Abigail Martínez Chávez 93135

Requisito para la obtención del título de

INGENIERO EN SISTEMAS COMPUTACIONALES

Profesor Responsable: *M.C. Saúl González Campos*

Profesor Participante: Dra. Vianey Guadalupe Cruz Sánchez

Noviembre de 2013

Autorización de Impresión

Los abajo firmantes, miembros del comité evaluador autorizamos la impresión del proyecto de titulación

CONTROL DINÁMICO DE POWERPOINT MEDIANTE LA TECNOLOGÍA KINECT DE MICROSOFT

Elaborado por los alumnos:

José Juan Reyes Arzola 94633

Tania Abigail Martínez Chávez 93135

Fernando Estrada Saldaña	Saúl González Campos
Profesor de la Materia	Asesor Técnico

Declaración de Originalidad

Nosotros, José Juan Reyes Arzola y Tania Abigail Martínez Chávez declaramos que el material contenido en esta publicación fue generado con la revisión de los documentos que se mencionan en la sección de Referencias y que el Programa de Cómputo (Software) desarrollado es original y no ha sido copiado de ninguna otra fuente, ni ha sido usado para obtener otro título o reconocimiento en otra Institución de Educación Superior.

José Juan Reyes Arzola	Tania Abigail Martínez Chávez

Dedicatoria

Este proyecto va dedicado a nuestras familias, las cuales nos brindaron su apoyo incondicional, sin importar los problemas que se estuvieron suscitando en periodo de tiempo que estuvimos ocupados con nuestro proyecto y también gracias a nuestras familias por apoyarnos en el trascurso de nuestra vidas escolares, desde que iniciamos el preescolar y primaria, hasta estos momentos en los que estamos por culminar nuestras carreras profesionales.

También les dedicamos este proyecto a nuestros profesores los cuales son un motor muy importante para nuestro desarrollo educativo, social y hasta profesional. Gracias por su paciencia para enseñarnos los diferentes temas que requerimos en nuestro desarrollo profesional.

También muchas gracias a todos nuestros amigos los cuales nos apoyaron en a lo largo de los casi cinco años en la universidad, los cuales nos ayudaron a entender los temas que no entendíamos en diversas clase y nos apoyaron en los momentos buenos y malos de nuestra vida.

A todos ellos muchas gracias por su apoyo incondicional.

Agradecimientos

Queremos agradecer a nuestras familias por su apoyo a los largo de nuestra vida escolar y muy especialmente gracias por tenernos paciencia, estos días que la verdad no hemos tenido tiempo para convivir por estar desarrollando este proyecto.

También agradecemos a los profesores que tuvieron la paciencia de ayudarnos en los temas que no comprendimos, gracias por apoyarnos para aprender más de cada clase y muchas gracias por darnos proyectos difíciles que en un principio los veíamos inalcanzable, pero que en realidad solo nos ayudan a desarrollarnos profesionalmente.

Agradecemos a nuestros amigos por ayudarnos a llevar las clases más relajadas y con cordialidad.

Muchas gracias a todos ellos por su apoyo.

Índice

Lista de Figuras

Lista de Tablas

Introducción

Las presentaciones en *PowerPoint* son una herramienta fundamental para los alumnos y maestros, cada vez su uso es más frecuente en las escuelas, aunque existan otras herramientas alternas sigue siendo una de las herramientas más utilizadas para presentaciones.

El uso de *PowerPoint* ha causado la necesidad de los usuarios para manejar las presentaciones de manera más fluida y sencilla, cosa que ha causado que muchas personas busquen alternativas o traten de hacer más fácil el modo presentación, auxiliándose de dispositivos externos al programa, como controles remotos *bluetooth*, aplicaciones para celulares, mouse inalámbrico, etc.

El *KINECT de Microsoft* se ha convertido en una herramienta no solo para jugadores sino también para el uso de aplicaciones escolares, es por esto que el presente proyecto aplica para facilitar el uso de las presentaciones en *PowerPoint* en su modo presentación con la utilización de gestos y comandos de voz.

El presente proyecto se demuestra que es posible manejar de forma natural el modo presentación de *PowerPoint* por medio del sensor *KINECT* de *Microsoft*, de una forma cómoda y fácil. Ya que se utilizó el seguimiento del esqueleto para establecer el reconocimiento de gestos en ambas manos y se utilizó el arreglo de micrófonos para el reconocimiento de comandos de voz en español.

Se considera que la aplicación es agradable y cómoda, ya que las pruebas hechas no de muestran que la mayoría de las personas encuestadas, les agrado el sistema y no le parece difícil de utilizar.

El documento presenta los siguientes capítulos:

Capítulo 1.- En el primer capítulo se realizó la investigación para resolver el problema de la utilización de las presentaciones de *PowerPoint*, sin necesidad del algún dispositivo externo y manejar las presentaciones de una manera más sencilla.

Capítulo 2.- En el segundo capítulo se muestra teoría sobre el funcionamiento del *KINECT de Microsoft*, herramientas que se utilizaron, investigación sobre otros proyectos similares que utilizaron dicha tecnología.

Capítulo 3.- En el tercer capítulo se desarrolló la aplicación utilizando las herramientas mencionadas del capítulo anterior, se muestra la metodología y pasos utilizado en la creación de la aplicación.

Capítulo 4.- En el cuarto capítulo se realizó un cuestionario sobre los intereses que las personas puedan tener para utilizar la aplicación. También se hicieron pruebas con veintidós personas para saber el desempeño de la aplicación y si se procedería algún cambio, además se les hizo otro cuestionario en el cual se les pregunto si les había agradado el sistema, que les gustaría cambiarle y que no les gusto.

Capítulo 5.- En el quinto capítulo se respondieron las preguntas de investigación hechas en el primer capítulo de la investigación. Se juntaron las distintas opiniones de los usuarios para saber lo bueno y malo de la aplicación o si era re querible un cambio a la interfaz, también los resultados técnicos y generales al uso de la aplicación.

Capítulo 1. Planteamiento del problema

Mediante la realización del proyecto se han encontrado distintas aplicaciones que utilizan esta misma tecnología, los cuales se presentan en este capítulo.

Desde el lanzamiento de la tecnología *KINECT* de *Microsoft* han surgido muchos proyectos en todo el mundo con el objetivo de aprovechar esta tecnología para abarcar campos de desarrollo y no exclusivamente su uso en los videojuegos.

1.1 Antecedentes

Las presentaciones son importantes en todo el mundo, porque es la mejor forma de mostrar temas diversos ante una audiencia. Debido a esto, se puede encontrar en el mercado una gran variedad de herramientas para hacer presentaciones, entre las cuales se encuentran *Prezi*, *Impress* (de *Libre Office*) de las cuales la más conocida es el programa desarrollado por *Microsoft* conocido como *PowerPoint*, hacia el cual se enfoca el presente proyecto. [1]

Por otro lado la tecnología *KINECT*, la cual es un nuevo avance de *Microsoft*, facilita una interacción hombre-máquina de forma natural, es decir, se utiliza reconocimiento de voz y mando, gestos con las manos y seguimiento de movimientos del cuerpo. [2] Originalmente fue utilizado en la consola *Xbox* a finales del 2010, sin embargo las versiones actuales permiten la compatibilidad con el sistema *Windows*, ya que en junio del 2011 fue el lanzamiento del *KINECT* para *Windows.* [3] [4] Es en parte debido a dicho lanzamiento que los desarrolladores de todo el mundo intentan actualmente buscar áreas en donde aplicar esta tecnología.

A lo anterior se ha sumado la creación del *KINECT SDK* de *Windows* y de *frameworks* alternativos a este que facilitan la utilización correcta del *KINECT*. [4] Tal es el caso de *Kinesis.io*, un *framework* que sirve para la creación de aplicaciones nativas basadas en gestos utilizando *HTML* y *JavaScript.* Esto permite acceder a reconocimiento de gestos con la mano, seguimiento de articulaciones, saltos, movimientos corporal y reconocimiento de voz con palabras claves. Además brindan la herramienta de un simulador *KINECT* que permite probar aplicaciones a los desarrolladores que no tengan un sensor *KINECT* de *Microsoft*. [5]

OpenNI es un *framework* de código abierto para el desarrollo de aplicaciones *KINECT*, permite el manejo de comandos con gestos de voz y movimiento en tiempo real.

Actualmente está disponible para *Windows 7* y *Ubuntu 10.10*. y se considera una excelente alternativa a *Windows SDK*. [2] [6]

El *KINECT SDK* es el paquete de desarrollo de *Microsoft* para la realización de aplicaciones compatibles con esta tecnología, fue lanzado en Junio del 2011.
Este paquete de desarrollo se puede obtener de manera gratuita para estudiantes en el sitio de *Dreamspark*, así *Microsoft* facilita el desarrollo de aplicaciones a la comunidad estudiantil. [4]

Se han desarrollado muchos proyectos con esta tecnología pero los que más se relacionan con el proyecto son los siguientes:

Kinemote es una aplicación donde se utiliza el *KINECT* para sustituir los comandos del mouse por medio de gestos de movimiento compatible con *Windows 7*, desarrollado con el *framework OpenNI* de código abierto. [7]

WIN&I es una aplicación desarrollada por el equipo de *Evoluce*, donde éste sustituye, al igual que *Kinemote*, el ratón con gestos de movimiento utilizando el *KINECT* a un rango de siete metros. Esta aplicación funciona solamente en *Windows 7* y tiene un costo de 19.90 € (versión hogareña) y 39.90 € (versión empresarial). [8]

Kinesis es un proyecto que permite manejar presentaciones de documentos con el *KINECT* de *Microsoft*, en este se pueden visualizar diferentes tipos de archivos tales como documentos de *Microsoft Office*, imágenes y textos. Dos pantallas en donde una de ellas es para mostrar la presentación y la otra es para mostrarle al presentador el menú para manejar la presentación. La desventaja que tiene *Kinesis* es que requiere de muchos recursos en la computadora, pues necesita memoria *RAM* mayor a 2*GB* y un procesador con velocidad mayor a 2.66*GHz*. [9]

1.2 Definición del problema

Actualmente el manejo de las presentaciones consiste en manipularlas por medio de dispositivos *bluetooth* que se utilizan típicamente para encender, iniciar la presentación y controlar el cambio entre las diapositivas. Sin embargo, al no contar con estos dispositivos, el usuario debe manipular las presentaciones directamente, lo cual en ocasiones tiene como consecuencias: pérdidas de tiempo, de recursos y de la fluidez de la presentación.

1.3 Objetivos de la investigación

Desarrollar una interfaz para manipular presentaciones en *PowerPoint* utilizando la tecnología *KINECT* de *Microsoft.* Se pretende realizar un enlace entre *KINECT* y *PowerPoint* para controlar a distancia el movimiento de las diapositivas e interactuar con los objetos multimedia sin la necesidad de utilizar el teclado o el *mouse* y así crear una presentación más dinámica, mediante el desarrollo de una interfaz sencilla y amigable al usuario.

1.4 Preguntas de investigación

Las preguntas de investigación que surgieron en este proyecto son las siguientes:

- ¿Cómo se comunicará *Windows 8* con el *KINECT*?
- ¿Cómo se comunicará *Microsoft Office* con *KINECT?*
- ¿Qué herramientas se necesitarán para el desarrollo?
- ¿A qué distancia será posible manipular el programa?
- ¿Cómo se va a lograr que únicamente reconozca al presentador y no interfiera con la audiencia?
- ¿Qué limitaciones tiene el utilizar el modo "presentación" de *PowerPoint*?

1.5 Justificación de la investigación

La interfaz ayuda al manejo de presentaciones para así evitar problemas actuales, tales como el tener que cambiar manualmente de diapositiva o el utilizar dispositivos de *bluetooth* que tienen funciones limitadas, como el solo encender y cambiar de diapositivas. Así mismo, evita que los presentadores pierdan tiempo al tener que estar cambiando manualmente de diapositiva, sobre todo si se encuentran lejos del teclado o del *mouse*, de manera que puede favorecer la fluidez de la presentación de una manera significativa.

1.6 Limitaciones y delimitaciones de la investigación

Las limitaciones en la aplicación de este proyecto son:

Se limitará a computadoras con el sistema operativo *Windows 8* o superior.

Es necesario contar con el sensor *KINECT* de *Microsoft* y su adaptador de corriente.

Las delimitaciones:

Se decidirá en el transcurso de la investigación si se decide manipular los archivos de *Microsoft PowerPoint* para lograr una mayor funcionalidad en la interfaz o si solamente se manipula al *PowerPoint* en el modo “presentación”.

Se decidirá en el transcurso de la investigación si se programarán comandos de voz además de gestos de movimiento.

Capítulo 2. Marco Teórico

En este capítulo se muestra los conceptos utilizados en la realización del proyecto, el cual tiene el propósito de controlar las presentaciones de una manera más dinámica que en la actualidad, para permitir una interacción más natural entre el usuario y la interfaz.

2.1 *KINECT*

Es un sensor de movimiento *USB* (*Universal Serial Bus*) de *Microsoft* que revolucionó la manera de interactuar con los videojuegos, pues nos proporciona una interfaz natural, que permite el manejo del juego a distancia y sin la necesidad de utilizar algún dispositivo de mando, por medio de sencillos gestos, comandos de voz y movimientos, que originalmente fue lanzada para la consola *Xbox* en el 2010. [3,10,11,12]

Gracias al éxito que se obtuvo con su lanzamiento, *Microsoft* decidió lanzar *KINECT* para *Windows SDK* en junio de 2011, pero a hora para desarrolladores de todo el mundo. [4,3]

KINECT utiliza la tecnología de la empresa israelí *PrimeSense*, la cual permite generar imágenes de profundidad en tiempo real, además del color y datos de audio en una escena. También hace que *KINECT* funcione en diferentes condiciones de luminosidad como en habitaciones muy iluminadas o totalmente obscuras, y no se requiere que el usuario posea ningún dispositivo para detectarlo como en la tecnología de la competencia (*Sony PlayStation y Nintendo Wii Remote*). [10,13]

2.1.1 Componentes del *KINECT*

El *KINECT* es un dispositivo que está compuesto por un gran número de componentes y tecnologías que ofrece a los usuarios nuevas experiencias al involucrar sus diferentes sentidos. [14]

Por un lado *KINECT* tiene sensores que actúan como dispositivos de entrada, los cuales leen la información que se encuentra en el medio físico frente de ellos. Tiene cuatro micrófonos que combinados pueden filtrar ruido de fondo y detectar la posición de la persona en una habitación, estos están distribuidos de la siguiente manera: tres enfrente y uno a la izquierda (Figura 1, G-I), Oficialmente *KINECT SDK* es el único que puede acceder a ellos. También tiene una cámara estándar web en el centro (Figura 1, C). A su lado esta una cámara infrarroja y un acelerómetro dentro del dispositivo. [15,16,10]

Por otro lado *KINECT* tiene sensores que actúan como dispositivos de salida. Tiene un emisor laser (Figura 1, A) que se activa al conectarlo a un dispositivo y funciona en combinación con la cámara infrarroja (IR) para obtener la información de toda la habitación.

Otro componente es el indicador de *LED* (Figura 1, B) que puede servir para interactuar con el usuario al estar encendiendo y apagando. Para finalizar también tiene un motor que permite girar el *KINECT* treinta grados hacia arriba o hacia abajo. [15,10]

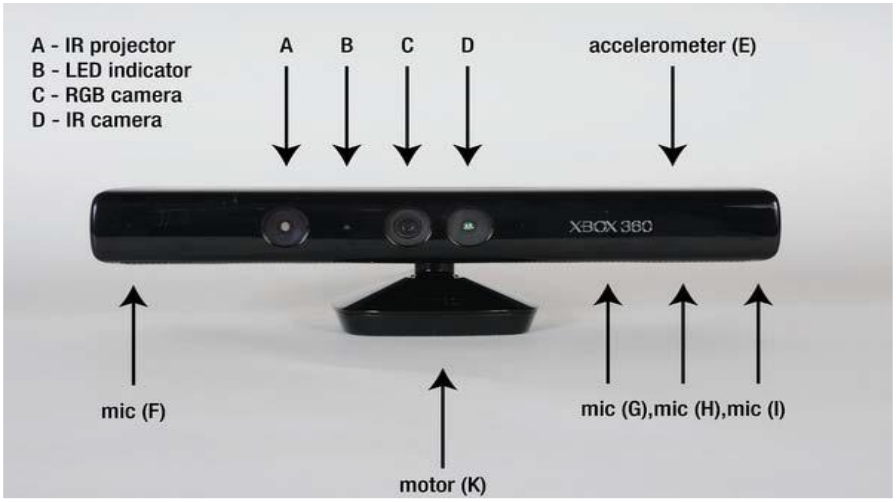


Figura 1: Sensor *KINECT* [15]

Gracias a todos estos elementos, el *KINECT* puede realizar las siguientes tareas: reconocimiento facial, de voz y seguimiento esquemático. [17]

2.1.2 Funcionamiento

Para poder entender el funcionamiento del *KINECT*, primero se tiene que entender cómo funciona la imagen de profundidad. Se compone de dos partes: el emisor de laser IR y una cámara de infrarrojos. El emisor de laser IR crea una nube de puntos de luz estructurada a 830 nm (nanómetros).

Estos puntos son registrados por la cámara y son comparados con patrones conocidos por el *KINECT*. Cualquier perturbación en la estructura se considera como una variación en la superficie las cuales son detectadas y permiten determinar qué tan profundos están los objetos a este proceso se le conoce como imagen de profundidad. (Figura 2). [18,19,20]

Figura 2: Nube de puntos de luz estructurada [14]

En este funcionamiento tiene las siguientes limitantes:

- La longitud de onda tiene que ser constante.
- La luz ambiental puede causar problemas.
- La distancia de visión está limitada a la resistencia del emisor.

La longitud de onda es controlada por un dispositivo que mantiene el láser a una temperatura constante. [18]

La luz ambiental es la mayor limitante de los sensores de luz estructurada, por ello *KINECT* implementa un filtro IR de 830 nm (nanómetros) a lo largo de la cámara IR, para evitar la pérdida de información u obtener resultados falsos. A pesar utilizar estos dispositivos, es recomendable utilizar el *KINECT* en lugares cerrados, ya que no funciona bien en lugares iluminados con la luz del sol. [18,17]

2.1.3 Seguimiento de esqueleto

KINECT no se limita solo con la detección simple del cuerpo, ya que permite localizar las articulaciones y diferentes partes del cuerpo, a esto se le conoce como el seguimiento de esqueleto. Esta capacidad permite el crear aplicaciones interactivas utilizando cualquier tipo de gesto corporal con el controlador del *KINECT*. [14,16,19]

Proceso del seguimiento de esqueleto (Figura 3):

- El láser de infrarrojo lanza la nube de puntos de luz estructurada.
- *KINECT* crea el mapa de profundidad a partir de la detección de la nube por el sensor infrarrojo.

- Detecta el suelo y separa los objetos del fondo para encontrar el contorno humano.
- Se clasifica las partes humanas.
- Identifica las articulaciones.
- Recrea el esqueleto. [21]

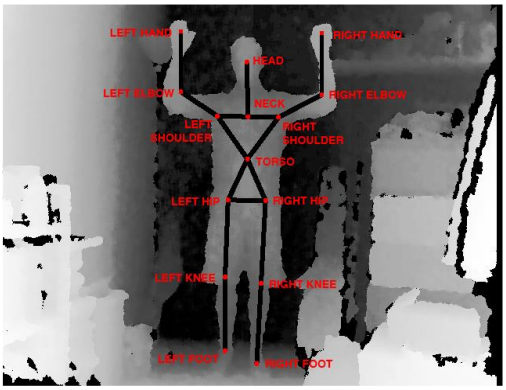


Figura 3: Seguimiento del esqueleto [14]

El esqueleto es un árbol grafico que consta de veinte puntos que corresponden a las diferentes articulaciones claves del cuerpo. Los puntos del esqueleto se expresan en coordenadas X, Y y Z que representan en metros. Como se muestran en la figura 4 donde el origen es el sensor *KINECT*. [22,23]

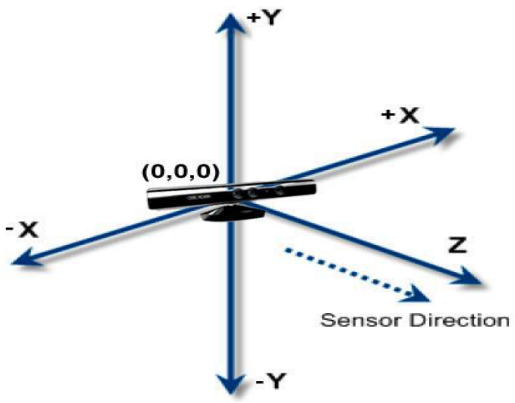


Figura 4: Sistemas de coordenadas de *KINECT*. [23]

KINECT puede rastrear a dos usuarios, al realizar seguimiento de esqueleto. La distancia recomendada del sensor al usuario es de aproximada 1,8 m para un usuario y para dos es de 2,5 m. y se requiere que el usuario tenga una altura mínima de 1m. [10]

2.1.4 Gestos

Es cualquier movimiento físico que un sistema digital puede percibir sin el uso de ningún dispositivo. Los gestos son fundamentales para el *KINECT,* son representaciones simples del mundo real para el manejo de interfaces. [24]

La tecnología de reconocimiento de gestos solo es la incorporación de información de que tan lejos o cerca está un objeto. [19]

2.1.5 Reconocimiento de voz

El proceso de reconocimiento de voz puede tener problemas con ruido del entorno en el que se pretende implementar. Para evitar este problemas se implementa el uso de una matriz de micrófonos y un procesador de señal digital para detentar la fuente de audio especifica. Además de la eliminación de ecos y ruidos.

El análisis de audio es una herramienta poderosa, pues acompañada por la visión del *KINECT* puede aumentar la cantidad de mando en una interfaz. [15,14]

2.2 Interfaz de usuario natural (*NUI*)

La Interfaz de Usuario Natural (*NUI* por sus siglas en inglés) se refiere a la interacción hombre-máquina de forma invisible para el usuario, donde los sentidos son los más importantes, en su mayoría la audición y la visión. [14]

La ventaja de usar *NUI* es que en poco tiempo el usuario pasa de ser inexperto a experto, pues el sistema se adapta a la forma de actuar del usuario. Es un término que abarca varias tecnologías tales como el reconocimiento de voz, *multitouch* e interfaces como *KINECT*. [14,16,24,2]

Las Interfaces Naturales se pueden dividir en dos grupos:

- Pantalla táctil. Como *iPad* y *Tablet*.
- Forma libre. Reconocimiento de gesto como *KINECT* o sensores. [24]

2.3 *Frameworks KINECT*

2.3.1 *KINECT* para *Windows SDK*

KINECT para *Windows SDK* es un conjunto de herramientas que permite crear aplicaciones que utilicen el *KINECT*. [23]

Fue desarrollado por *Microsoft*, lanzado el 16 Junio del 2011 la versión beta y al principio de 2012 lanza la versión comercial. Este paquete de desarrollo se puede obtener de manera gratuita para estudiantes en el sitio de *Dreamspark*. [4,25]

2.3.1.2 Arquitectura

La estructura interna de *KINECT* para *Windows SDK* se muestra en la figura 5, en donde se observa los componentes que *SDK* como la Interfaz de Usuario Natural, Interfaz de Programación de Aplicaciones (*API*) y los controladores del *KINECT*. [25]

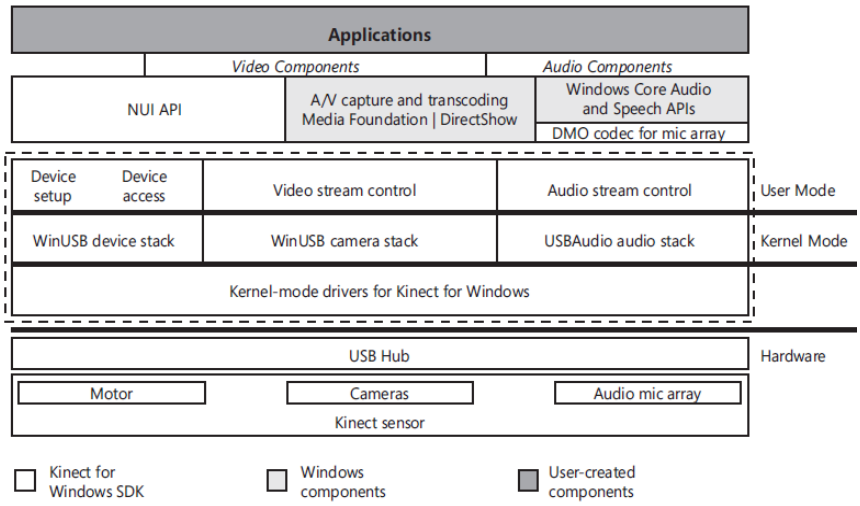


Figura 5: Arquitectura de *KINECT* para *Windows SDK* [25]

2.3.1.3 Capacidades

SDK permite controlar las siguientes capacidades del *KINECT*:

- **Controladores de *KINECT*:**
  - Permite la captura y procesar sonido de la *API* de audio de *Windows*
  - Permite captura y transmisión de datos de imagen y profundidad.
  - Además de permitir el uso de más de un *KINECT* en una aplicación.
  - Tiene una funcionalidad de detectar esqueleto desde 40 cm distancia. [23]

- ***NUI API:***

  Es un conjunto de *API* que obtienen los datos de sensor y controlan el dispositivo. Además de procesar los datos de imagen y profundidad para el seguimiento de esqueleto. [23]

- **Audio *API*:**

  Soporta y controla el sistema de micrófonos. [23]

2.3.1.4 Requisitos

Requisitos de *hardware*:

Para utilizar el sensor *KINECT* para *Windows* es necesario tener una *PC* con las siguientes características:

- *Windows 8*.
- 32 *bits*(x86) o 64 *bits*(x64)
- Procesador *dual-core* de 2,66 *GHz* o más rápido
- Dedicado *bus USB 2.0*
- 2 *GB* de *RAM* [26]

Requisitos de *software*:

- *Microsoft Visual Studio 2010 o 2012*
- *Microsoft .NET framework 4* [25]

2.3.2 *OpenNI*

OpenNI (Interacción Natural Abierto) es una organización sin fines de lucro, que tiene como objetivo certificar y promover la compatibilidad de Interacción Natural en dispositivos, aplicaciones y *middleware*. Para lograr este objetivo *OpenNI* a puesto a disposición de los desarrolladores un marco de código abierto, el cual proporciona un interfaz de programación de aplicaciones (*API*) con interacción natural, llamado *framework OpenNI*, el que se puede encontrar en la página oficial de *OpenNI*. [2,14]

Esta organización fue fundada en noviembre del 2010 por las compañías:

PrimeSense, es líder en el área de Interacción Natural (*NI*) y la profundidad 3D, además proporciona su tecnología de detección 3D al *KINECT*.

Willow Garage, laboratorio de investigación robótica donde desarrollan *hardware* y *software* de código abierto.

Side-Kick, empresa desarrolladora de juegos manejados por gestos.

Asus, proporciona hardware a *OpenNI* para promover las aplicaciones *NI*.

AppSide, ofrece plataformas de fácil acceso a juegos de movimiento controlado. [2,14,11]

2.3.2.1 Arquitectura de *framework OpenNI*

OpenNI se puede visualizar en la figura 6. La primera capa representa el software que utiliza el *API* de este *framework*. La capa de en medio representa *OpenNI*, que proporciona interfaces de comunicación para sensores y los componentes de *middleware*. La última capa muestra los dispositivos de *hardware* que capturan los datos y entornos. [2,14]

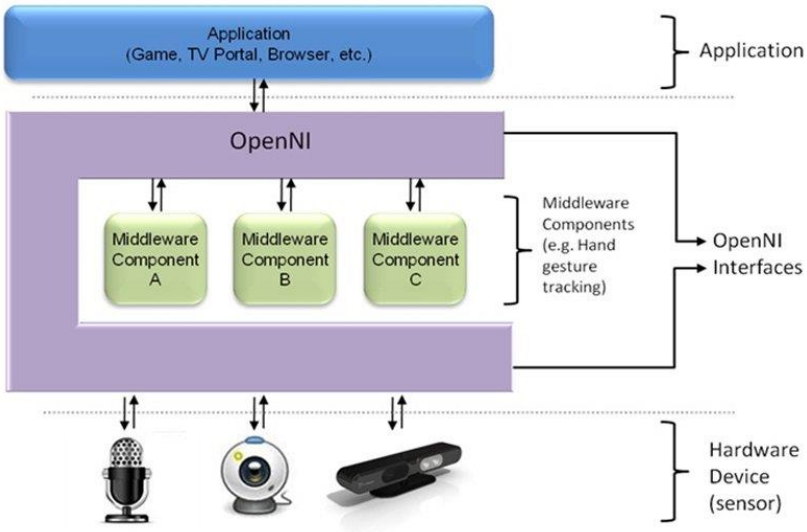


Figura 6. Arquitectura de *OpenNI* [2]

2.3.2.2 Módulos

Los módulos son los que proporciona la interfaz para manejar los componentes físicos y de *middleware*, los cuales se conocen como módulos, los que se usan para producir y procesar datos sensoriales.

Módulos de sensor:

- Sensor 3D
- Cámara *RGB*
- Cámara IR

- Dispositivos de audio

Módulos de *middleware*

- Análisis de cuerpo completo: en este se crean estructuras corporales como modelo de datos, describe articulaciones, orientación y centro de masa.
- Análisis de mano
- Gestos: identifica gestos predefinido y lanza alertas a la aplicación.
- Analizador de escenas: analiza el entorno para separar las figuras del fondo. [2]

2.3.2.3 Los nodos de producción

Los nodos de producción son los que productores de los datos para las aplicaciones *NI*, es decir son los generados en el mapa de profundidad. [14,2]

Ejemplo: aplicación de nodos de producción figura 7.

Figura 7: Identificación de gesto de mano.

Tipos de nodos de producción:

Los tipos de nodos de producción soportados en *OpenNI*:

- Los relacionados con los sensores: soportan dispositivos, generador de profundidad, de imágenes, IR y de audio. [14]
- Los relacionados con *Middleware*: soportan gestos generadores de alertas, analizador de escenas, generador de punto de mano, de usuario y jugador. [14]

2.3.2.4 Cadenas productivas:

Las cadenas productivas son aquellas que se generan al crear una secuencia de nodos dependientes el uno del otro. Esto puede suceder al momento de utilizar diferentes módulos simultáneamente registrando una solo aplicación [2]

2.3.2.5 Capacidades

OpenNI tiene claro que al trabajar con distintos dispositivos es necesario el tener un amplio listado de capacidades y de opciones de configuración para nodos de producción para soportar de forma satisfactoria. Sus capacidades son: [14]

- Segundo ángulo: permite cualquier generador de imágenes de profundidad, de imagen o de IR y trasformar los datos como si fueran capturados de otro lugar.
- Recortar: permite que el generador de imagen pueda seleccionar y recortar solo un área.
- *Frame sync*: permite que dos sensores sincronicen sus tramas al mismo tiempo.
- Espejo: Permite la duplicación de datos producidos por el generador.
- Pose de detección: permite reconocer una posición específica del usuario.
- Esqueleto: activa un generador de usuario para emitir los datos del esqueleto del usuario.
- Posición del usuario: permite optimizar la imagen de profundidad para un área específica de la escena.
- Estado de error: permite que un nodo de información se encuentre en error, es decir, que no esté funcionando correctamente.
- Bloque *Aware*: permite bloquear nodos fuera del límite del contexto.
- Mano que toca *Edge FOV*: es una alerta que se activa cuando la mano llega a los límites del campo de visión. [2]

2.3.2.6 Requisitos:

OpenNI está disponible en las siguientes plataformas:

- *Windows XP* y versiones posteriores, para *32-bit*
- *Linux Ubuntu 10.10* para x86 [2]

2.3.3 *Kinesis.io*

Kinesis.io, es un *framework* que sirve para la creación de aplicaciones nativas basadas en gestos utilizando *HTML* y *JavaScript.* Esto permite acceder a reconocimiento de gestos con la mano, seguimiento de articulaciones, saltos, movimientos corporal y reconocimiento de voz con palabras claves. [5]

Además nos brindan la herramienta de un simulador *KINECT* que permite probar aplicaciones a los desarrolladores que no tengan un sensor *KINECT* de *Microsoft*. Este

simulador está disponible para *Mac OS* y *Windows 7*, se puede descargar en la página oficial de *kinesis.io*. [5]

En este simulador se puede configurar las coordenadas de inicio X, Y, Z y seleccionar alguno de los gestos, que ya tiene configurados como se muestra en la figura 8 y su funcionamiento se muestra en figura 9. [5]

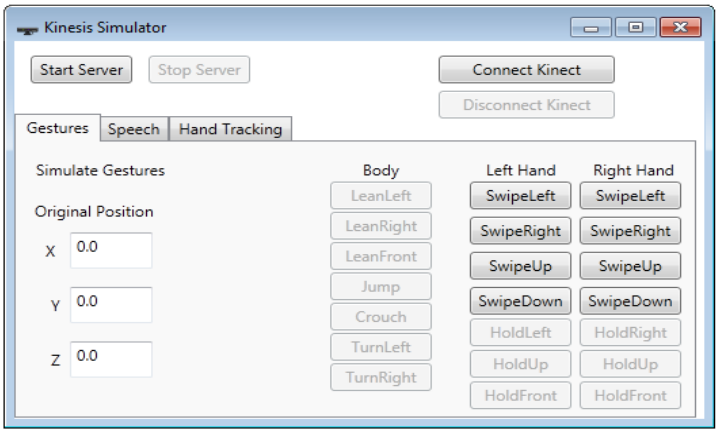


Figura 8: Ventana de configuración de Simulador *Kinesis*. [5]

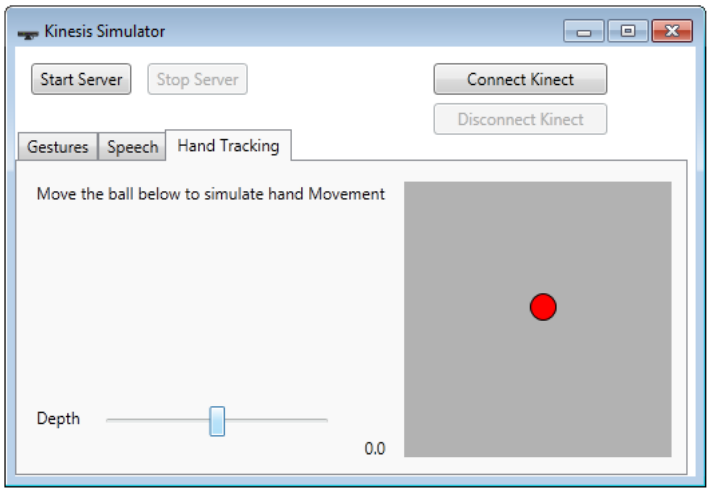


Figura 9: Ventana del Simulador *Kinesis*. El círculo rojo simula la posición de la mano. [5]

2.3.3.1 Requisitos

Para utilizar *Kinesis* es necesario cubrir los siguientes requisitos:

- *PC* con *Windows 7*
- *KINECT* para *Windows* o *XBOX 360.*
- *Kinesis SDK* [5]

2.4 Conclusión

A lo largo de este capítulo sea a mostrado los conceptos que son necesario para realizar el presente proyecto. Que herramientas se pueden utilizar para el desarrollo de la aplicación, qué características tiene el sensor *KINECT* y qué características tiene la interfaz natural con esta tecnología.

Con este capítulo se vieron los componentes de *KINECT* que se utilizaron en la aplicación, tales como la cámara de detección de profundidad que es utilizado para el seguimiento del esqueleto y la detección de las manos. Y la matriz de micrófonos para utilizada para la detección de comandos de voz.

Se vieron las diferentes tecnologías que se pueden utilizar para el manejo del *KINECT,* tales como el *KINECT* para *Windows SDK* que es una de las mejores opciones que se tiene de para el manejo del *KINECT* en las aplicaciones, ya que para los desarrolladores que son estudiantes es gratuito.

Además no es de código abierto por lo que nadie puede modificar lo y con esto el presente proyecto no tendría problemas futuro. Y como esta librería es propiedad de *Microsoft* se tiene mucha más confianza y seguridad para usarlo en este proyecto.

Capítulo 3. Materiales y Métodos

En el presente capítulo, se expondrán los materiales y los métodos utilizados en este proyecto, que tiene como finalidad el desarrollar una aplicación de *software* que permita el manejo dinámico de una presentación *PowerPoint* mediante el *KINECT* de *Microsoft.*

3.1 Descripción del área de estudio

El proyecto es de desarrollo tecnológico en donde se implementa la tecnología *KINECT* de *Microsoft* para el manejo de presentaciones *PowerPoint*, por lo que se realizó una detallada investigación para el saber más acerca del funcionamiento del *KINECT*, los conceptos y características, además de herramientas para el manejo de *PowerPoint* y para el desarrollo de la aplicación. Desarrollo realizado en el lenguaje C#, utilizando las herramientas de desarrollo de *KINECT SDK*, utilizando sencillos y prácticos gestos programados.

Se realizó un estudio prospectivo, ya que se recolectó información de cómo se comporta la aplicación con forme se le hacen pruebas de escritorio, así para se tuvo en consideración la información recabada para hacer cambios en la aplicación, y hasta tener la aplicación terminada se procedió a hacer más pruebas en las cuales fueron realizadas por veintidós personas que no conocían la aplicación, en donde se levantaron datos acerca del comportamiento de la aplicación, datos que ayudaron a saber que más le faltaba a la aplicación y cambiar lo que no les agradó.

Y se realizó un estudio de tipo experimental, ya que es una aplicación prototipo, en la que se llevaron a cabo pruebas de escritorio para ir depurando los errores y la usabilidad de la aplicación, manipulando las variables tales como diferentes tipos de gestos y comandos de voz más agradables, hasta encontrar los más adecuados para el usuario.

3.2 Materiales

Materiales utilizados en el proyecto:

- *KINECT* de *Microsoft*
- Adaptador de corriente *USB* para *KINECT* de *Microsoft*
- Computadora personal con las siguientes especificaciones:

- *Windows 8*.
- 64 *bits*(x64).
- Procesador de 2,10 *GHz* o más rápido
- Dedicado *bus USB* 2.0
- 4 *GB* de *RAM*
- *Microsoft Visual Studio* 2012
- *KINECT SDK* 1.8
- *Microsoft .NET framework* 4.5
- *PowerPoint 2013*

3.3 Métodos

En este apartado se muestra los diferentes métodos utilizados, así como algunas características de la investigación.

El proyecto fue elaborado y probado en las instalaciones de Instituto de Ingeniería y Tecnología (IIT) en el periodo de agosto 2012 a octubre 2013. En donde participaron alumnos en las pruebas de la aplicación.

Dentro de esta investigación se desarrolló una aplicación en la cual los usuarios pudieron manipular las diapositivas de *PowerPoint*, mediante un interfaz de forma natural por medio del *KINECT* de *Microsoft*. Esta aplicación se realizó para resolver los inconvenientes que han tenido los expositores al manejar sus presentaciones, se tiene el control de las diapositivas en necesidad del teclado o algún dispositivo de *bluetooth*.

La aplicación se desarrolló mediante código en lenguaje C#, implementando las librerías de *KINECT SDK* para el manejo de *KINECT* de *Microsoft* y la de manejo de *PowerPoint.* Se tiene una interfaz de forma natural, la cual es manipulada por el usuario mediante gestos programados y comandos de voz.

En la aplicación se detecta y muestra al presentador por medio del *KINECT*, además se detectan los gestos y los comandos de voz que son requeridos para la manipulación de la presentación, gestos como la siguiente o anterior diapositiva, acercar, subir volumen, y comandos de voz, para abrir presentación, seleccionar, mudo, iniciar y detener videos multimedia, alejar y cerrar .

3.3.1 Metodología:

Sección donde se explican de forma amplia la metodología, pasos y herramientas utilizadas para la creación del presente proyecto.

3.3.3.1 Diseño:

El diseño que se aplicó consta de una interfaz intuitiva y simple, la cual contiene gestos programados y comandos de voz, los que son detectados por el *KINECT* de *Microsoft* y con ello la aplicación permite el manejo de las presentaciones, el diseño de esta aplicación es representado en el siguiente diagrama de flujo (Figura 10):

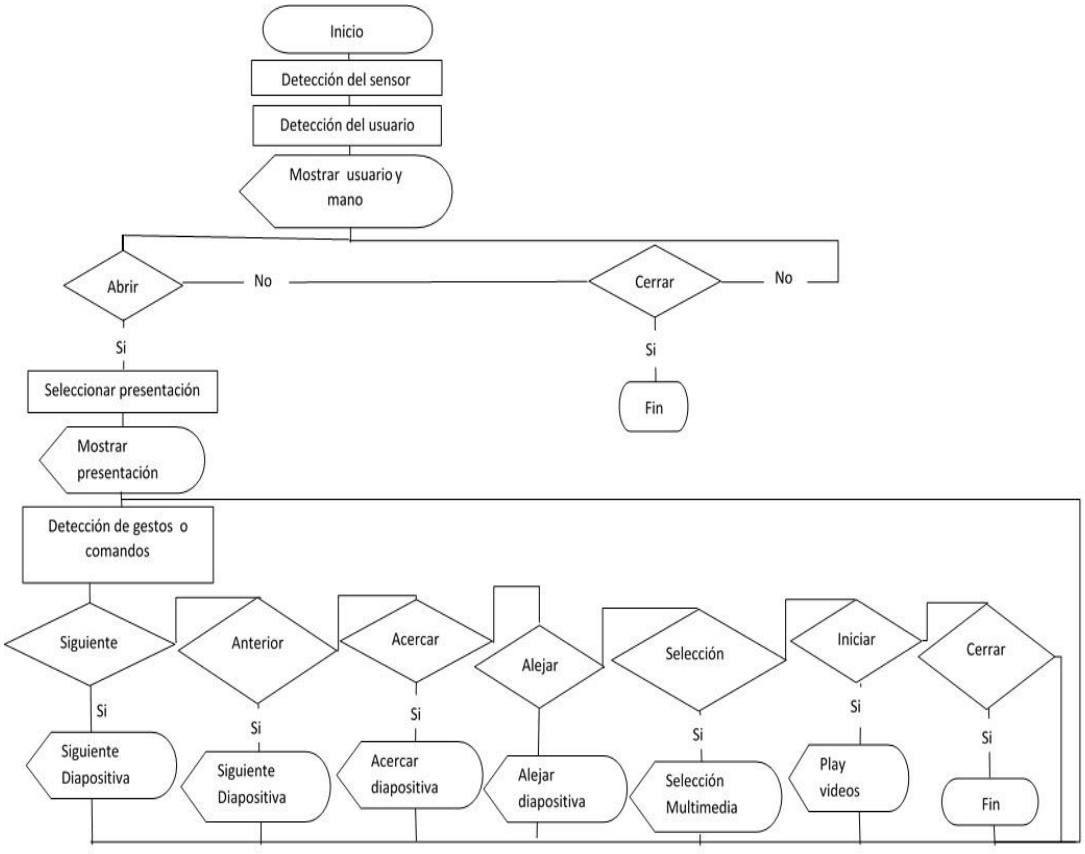


Figura 10: Diagrama de flujo de la aplicación.

La aplicación consta de una pantalla de inicio en la que se tienen un detector de sensor, un visor de usuario y dos botones *KINECT* , además del poder controlar este entorno con la mano del usuario por medio de *KINECT* . Esta pantalla se muestra en figura 11 y figura 12.

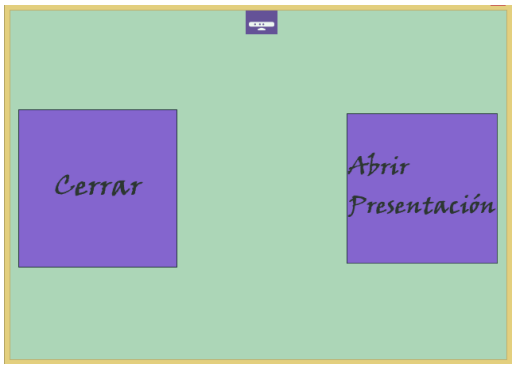


Figura 11: Pantalla de inicio 1.

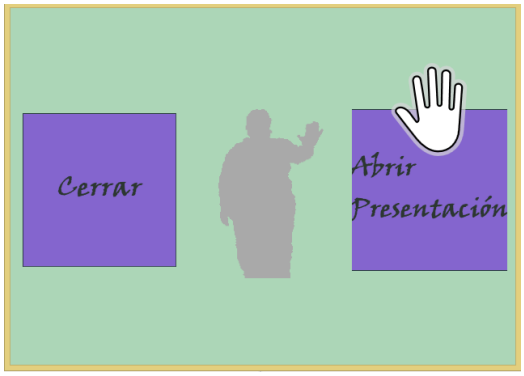


Figura 12: Pantalla de inicio 2.

Luego de presionar la opción “abrir presentación” se abre un cuadro de diálogo, en el cual se debe de seleccionar la presentación a utilizar (Figura 13).

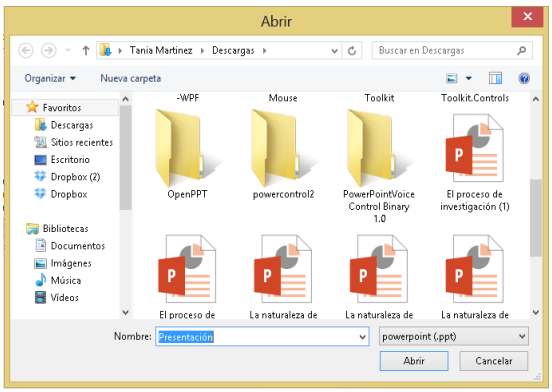


Figura 13: Cuadro de diálogo abrir.

Después de este proceso la presentación seleccionada se abre en modo presentación (Figura 14), la cual se puede manejar por medio de gestos y comandos de voz programados, los cuales nos permiten el manejo dinámico de la presentación.

Figura 14: *PowerPoint* modo presentación.

Los gestos y comandos de voz programados que contiene la aplicación son los siguientes:

- Abrir
- Siguiente diapositiva
- Anterior diapositiva
- Seleccionar componentes
- Iniciar multimedia
- Mudo
- Acercar
- Restaurar
- Alejar
- Cerrar

Gestos horizontales

Estos tipos de gestos detectan el movimiento de una mano en determinado sentido estando a la misma altura (Y), en movimiento en la coordenada X y a determinada velocidad.

Los gestos horizontales utilizados en el proyecto son los siguientes:

Gesto "siguiente diapositiva":

En este gesto el *KINECT* detecta el deslizamiento de la mano izquierda hacia la mano derecha y es utilizado en la aplicación para cambiar a la siguiente diapositiva de la presentación en curso. Es utilizado tal como se muestra en la figura 15.

Figura 15: Gesto “siguiente diapositiva”

Gesto “anterior diapositiva”:

Este gesto es detectado por el *KINECT* cuando se desliza la mano derecha hacia la mano izquierda. En la aplicación se utiliza este gesto para cambiar a la diapositiva anterior de la presentación, es utilizado tal como se muestra en la figura 16.

Figura 16: Gesto “anterior diapositiva”

Gestos verticales:

Estos tipos de gestos detectan el movimiento de una mano a determinado sentido estando a la misma distancia en X, pero moviéndose en la coordenada Y y a determinada velocidad.

Los gestos verticales utilizados en el proyecto son los siguientes:

Gesto "acercar":

Este gesto detecta el deslizamiento de la mano derecha de abajo a arriba, y es utilizado en la aplicación para hacer acercamientos en una diapositiva. Es utilizado tal como se muestra en la figura 17.

Figura 17: Gesto "acercar"

Gesto "subir volumen":

Este gesto detecta el deslizamiento de la mano izquierda de abajo a arriba, y es utilizado en la aplicación para subir el volumen de los elementos multimedia ya seleccionados en la diapositiva. Es utilizado tal como se muestra en la figura 18.

Figura 18: Gesto “subir volumen”

Comandos de voz:

Los otros gestos son por medio de voz, sus palabras claves son las siguientes: Abrir, seleccionar, iniciar, detener, bajar, restaurar, cerrar.

- Abrir: comando de voz utilizado después de seleccionar la presentación para selección en el cuadro de dialogo abrir presentación.
- Seleccionar: comando de voz utilizado para seleccionar elementos multimedia de la presentación.
- Iniciar: comando de voz utilizado para iniciar (*play*) los elementos multimedia ya seleccionados.
- Mudo: comando para poner en mudo el volumen del video en ejecución.
- Restaurar: comando de voz utilizado para restaurar el *zoom*, a tamaño normal.
- Detener: comando de voz utilizado para detener los elementos multimedia después de ser seleccionados.
- Cerrar: comando de voz utilizado para cerrar la aplicación.

A continuación se muestra de forma más clara el desarrollo de la aplicación.

3.3.3.4 Desarrollo:

Procedimiento para la elaboración de la aplicación, se utilizó una computadora con *Windows 8* y Microsoft *Visual Studio 2012*, el lenguaje de programación *C#* y se creó el proyecto de aplicación *Windows Presentation Foundation* (*WPF*). En dicho proyecto se agregaron las librerías correspondientes a *KINECT SDK* versión 1.8 tales *Microsoft.Kinect.Interaction*, además se agregó la librería para el manejo de *PowerPoint Microsoft.Office.Interop.powerpoint.*

En el diseño de la aplicación se presentan controles especiales de *Microsoft.Kinect* para las aplicaciones *WPF*, los cuales nos permiten diseño interactivo y amigable para los usuarios.

Controles de *KINECT* utilizados son:

Primero se agregó en el *MainView.xaml* la referencia de la figura 19:

```
xmlns:k="http://schemas.microsoft.com/kinect/2013"
```

Figura 19: Referencia de *KINECT*.

KinectRegion, esta es el área de la ventana en donde interactúan los elementos *KINECT*. El siguiente código es el necesario se muestra en la figura 20.

```
<k:KinectRegion Name="KinectRegion" Margin="0,-10,0,10">
```

Figura 20: *KinectRegion*.

Se agregó el control de usuario:

KinectSensorChooserUI, este control muestra el estado del sensor *KINECT*, es muy importante ya que muestra si el *KINECT* está funcionando al momento de ejecución de la aplicación, muestra si está conectado, desconectado, etc.

El código utilizado para implementar este control es el siguiente (figura 21).

```
<k:KinectSensorChooserUI HorizontalAlignment="Center"  VerticalAlignment="Top"
Name="sensorChooserUi" RenderTransformOrigin="1.1,0.662" Opacity="0.8" />
```

Figura 21: Código KinectSensorChooserUI.

Se inicializaron todas las funciones necesarias para el funcionamiento de los componentes del *KINECT*, así como controladores de eventos *KINECT* y el encendido del sensor, tal como se muestra en la figura 22:

```
private KinectSensorChooser sensor;
public MainWindow()
{  InitializeComponent();
this.Loaded += new RoutedEventHandler(MainWindow_Loaded);
this.sensor = new KinectSensorChooser();
this.sensor.KinectChanged += SensorChooserOnKinectChanged;
this.sensorChooserUi.KinectSensorChooser = this.sensor;
this.sensor.Start(); }
```

Figura 22: Código inicialización componentes.

La siguiente función permite activar la detección de datos de profundidad y del esqueleto según la posición del usuario, repetitivamente haciendo que en todo momento el *KINECT* esté detectando al usuario sin importar si está cerca o lejos del sensor, su codificación se muestra en la figura 23.

```
private void SensorChooserOnKinectChanged(object sender, KinectChangedEventArgs
args)
bool error = false;
 {  if (args.OldSensor != null)
    { try
          {args.OldSensor.DepthStream.Range = DepthRange.Default;
            args.OldSensor.SkeletonStream.EnableTrackingInNearRange = false;
            args.OldSensor.DepthStream.Disable();
            args.OldSensor.SkeletonStream.Disable();
          }
          catch (InvalidOperationException)
          {error = true; }
        }
```

```
        if (args.NewSensor != null)
       {try
{args.NewSensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
            args.NewSensor.SkeletonStream.Enable();
        try
            {args.NewSensor.DepthStream.Range = DepthRange.Near;
               args.NewSensor.SkeletonStream.EnableTrackingInNearRange = true;
            }
            catch (InvalidOperationException)
            {args.NewSensor.DepthStream.Range = DepthRange.Default;
              args.NewSensor.SkeletonStream.EnableTrackingInNearRange = false;
            }
          }
          catch (InvalidOperationException)
          {error = true; }
       } if (!error)
          kinectRegion.KinectSensor = args.NewSensor;
}
```

Figura 23: Código para activación de sensor.

Cuando se ejecuta la aplicación, el *KinectSensorChooserUI* indica los siguientes estados:

Estado "listo", esto sucede cuando el *KINECT* es correctamente conectado, se muestra en la figura 24.

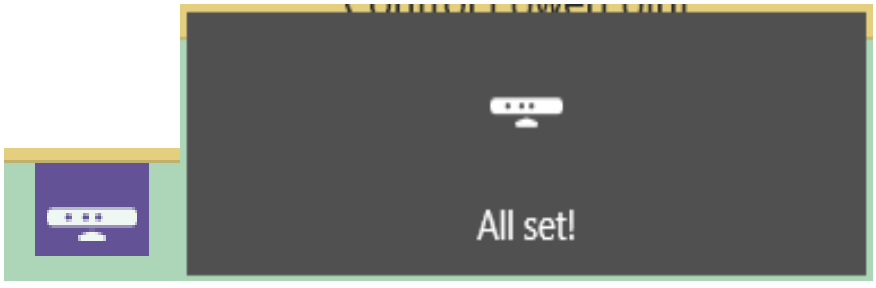


Figura 24: *KinectSensorChooserUI* en estado conectado.

Estado *KINECT* "necesario", este estado sucede cuando el *KINECT* no está conectado correctamente la computadora, se muestra en la figura 25.

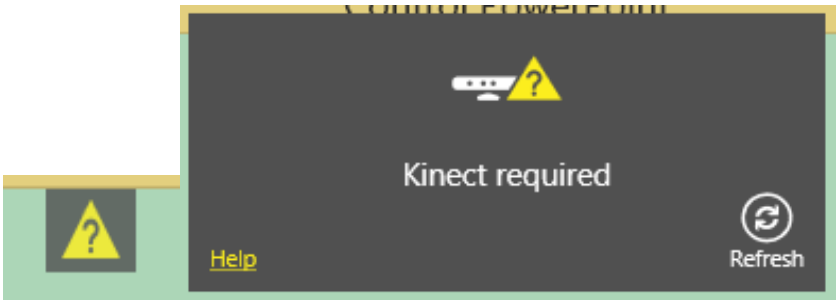


Figura 25: *KinectSensorChooserUI* en estado *KINECT* necesario.

Estado "inicializar", este estado sucede inmediatamente que se conecta el *KINECT*, se muestra en la figura 26.

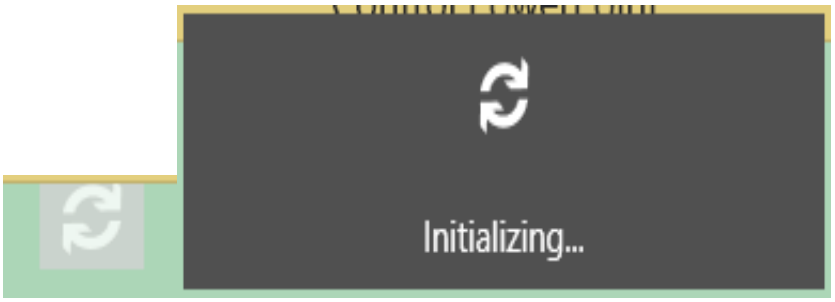


Figura 26: *KinectSensorChooserUI* en estado inicializar.

Estado "cable de alimentación desconectado", se produce cuando el cable de alimentación de corriente del *KINECT* está desconectado, se muestra en la figura 27.

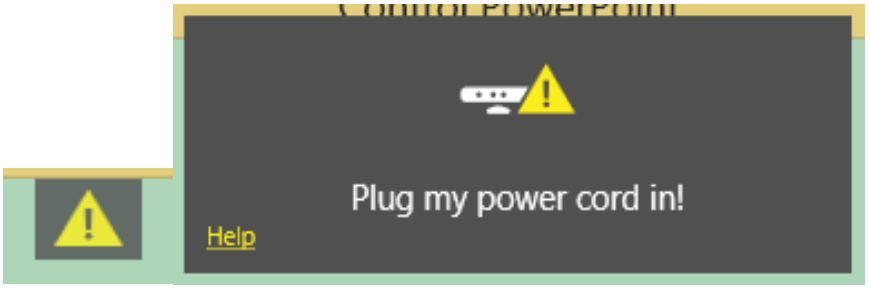


Figura 27: *KinectSensorChooserUI* en estado cable de alimentación desconectado.

KinectUserViewer, control *KINECT* que permite mostrar al usuario por medio de pixeles de profundidad del usuario, su funcionamiento se puede observar en la figura 28.

Figura 28: *KinectUserViewer*.

El siguiente código es el utilizado para agregar el *KinectUserViewer* a la ventana de diseño. Su código se muestra en la figura 29.

```
<k:KinectUserViewer k:KinectRegion.KinectRegion="{Binding
ElementName=kinectRegion}" Margin="144,45,152,44" />
```

Figura 29: *KinectUserViewer.*

KinectTileButton, son los botones utilizados en aplicaciones *KINECT* los cuales son semejantes a botones de inicio *Windows 8 ,* se muestra en la figura 30.

Figura 30: *KinectTileButton*

El código utilizado para agregar el botón cerrar es figura 31:

```
<k:KinectTileButton x:Name="Botoncerrar" Click="Buttoncerrar"
VerticalAlignment="Bottom"  FontSize="36" FontFamily="Viner Hand ITC"
FontWeight="Bold" Width="200" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" Foreground="#FF0C0F0F"
Height="200">Cerrar</k:KinectTileButton>
```

Figura 31: Código botón cerrar.

El funcionamiento de este botón es el siguiente figura 32:

```
private void Buttoncerrar(object sender, RoutedEventArgs e)
        {sensor.Stop();
        this.Close();}
```

Figura 32: Código función cerrar.

También se utilizó el botón abrir diapositiva como se muestra en la figura 33:

Figura 33: *KinectTileButton* abrir

Este botón abre un cuadro de dialogo, en el cual se deberá seleccionar la presentación y si se desea se abrirá la presentación.

El siguiente código es el utilizado para este botón figura 34:

```
private void Buttonabrir(object sender, RoutedEventArgs e)
    {
      Microsoft.Win32.OpenFileDialog Dialogo = new
      Microsoft.Win32.OpenFileDialog();
      Dialogo.FileName = "Presentación";
      Dialogo.DefaultExt = ".ppt";
      Dialogo.Filter = "powerpoint (.ppt)|*.pptx;*.ppt";
      Nullable<bool> result = Dialogo.ShowDialog();
      if (result == true)
      {
        FileName = Dialogo.FileName;
        PowerPoint.Application application = new PowerPoint.Application();
         PowerPoint.Presentation presentation =
```

```
            application.Presentations.Open2007(FileName,
            Microsoft.Office.Core.MsoTriState.msoTrue,
            Microsoft.Office.Core.MsoTriState.msoFalse,
            Microsoft.Office.Core.MsoTriState.msoFalse,
            Microsoft.Office.Core.MsoTriState.msoTrue);
             PowerPoint.SlideShowSettings sst = presentation.SlideShowSettings;
             sst.ShowType =
Microsoft.Office.Interop.PowerPoint.PpSlideShowType.ppShowTypeSpeaker;
             sst.Run();
             this.Botonabrir.IsEnabled = false;
          }
       }
```

Figura 34: Código para abrir cuadro de diálogo.

Otros componentes utilizados en la aplicación son los siguientes:

3.3.3.2 Gestos programados:

Los gestos son aquellos movimientos consecutivos de partes del cuerpo que hacen una determinada acción, así como saludar, brincar, etc. En esta aplicación se utilizaron los gestos programados llamados deslizar (*swipe*), los cuales detectan los movimientos de manos en forma de vectores, ubicados en el espacio X, Y, Z.

Dichos vectores son detectados según la coordenada en la que se haga el movimiento y si se tiene un uniforme movimiento tanto en la misma altura de mano y velocidad, además de estar en la dirección correcta, ya que se tiene gestos de derecha-izquierda, gesto de izquierda-derecha y arriba-abajo.

Para la realización de estos gestos fue necesario utilizar el seguimiento de esqueleto, el cual es un algoritmo que logra identificar las partes de las personas que están en el campo de visión del *KINECT*, por medio de este algoritmo se obtendrá las articulaciones (*Joints*) y su posición en el espacio. Lo cual permite programar la forma de

detectar los movimientos repetitivos en un determinado intervalo de tiempo y obtener los gestos que se deseen.

En seguida se muestra el código utilizado para detectar el esqueleto del usuario en la aplicación, para detectar la posición de las manos (figura 35).

```
void sensor_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs
e)      {using (var skeletonFrame = e.OpenSkeletonFrame())
           {if (skeletonFrame == null)
            return;
           if (skeletons == null ||
            skeletons.Length != skeletonFrame.SkeletonArrayLength)
            {skeletons = new Skeleton[skeletonFrame.SkeletonArrayLength]; }
             skeletonFrame.CopySkeletonDataTo(skeletons); }
             Skeleton closestSkeleton = skeletons.Where(s => s.TrackingState ==
          SkeletonTrackingState.Tracked)
          .OrderBy(s => s.Position.Z * Math.Abs(s.Position.X))
          .FirstOrDefault();
          if (closestSkeleton == null)   return;
          var head = closestSkeleton.Joints[JointType.Head];
          var rightHand = closestSkeleton.Joints[JointType.HandRight];
          var leftHand = closestSkeleton.Joints[JointType.HandLeft];
          var leftShoulder = closestSkeleton.Joints[JointType.ShoulderLeft];
          var rightShoulder = closestSkeleton.Joints[JointType.ShoulderRight];
          if (head.TrackingState == JointTrackingState.NotTracked ||
          rightHand.TrackingState == JointTrackingState.NotTracked ||
          leftHand.TrackingState == JointTrackingState.NotTracked ||
          rightShoulder.TrackingState == JointTrackingState.NotTracked ||
          leftShoulder.TrackingState ==   JointTrackingState.NotTracked)
         {return; }
          GestoAvanceIzquierda(head, rightHand, leftHand);
          GestoAvanceDerecha(head, rightHand, leftHand);
```

```
        GestoAvanceArriba(head, rightHand, leftHand);
        GestoAvanceArribaIzq(head, rightHand, leftHand);
        GestoAvanceAbajo(head, rightHand, leftHand);
        GestoAvanceAbajoIzq(head, rightHand, leftHand);
}
```

Figura 35: Código esqueleto.

Después de este algoritmo, se utilizaron listas tipo Vector 3 para almacenar las posiciones que obtienen en cada tipo de gesto y otra lista para identificar los tipos de gestos ya identificados. Se declararon los valores flotantes X, Y, Z, para representar los valores de las posiciones de la mano como si fueran coordenadas.

Después se declararon los gestos y funciones utilizadas para identificarlos, además de las variables utilizadas para la detección del movimiento (figura 36).

```
        public struct Vector3
            {public float X;
              public float Y;
              public float Z;
              public DateTime date; }
         public enum Gesture
            {None,
              SwipeLeft,
              SwipeRight,
              SwipeUp,
              SwipeUpLeft,
               SwipeDown,
              SwipeDownLeft,}
            List<Vector3> positionListRight = new List<Vector3>();
            List<Vector3> positionListLeft = new List<Vector3>();
            List<Vector3> positionListUp = new List<Vector3>();
            List<Vector3> positionListUpLeft = new List<Vector3>();
            List<Vector3> positionListDown = new List<Vector3>();
```

```
List<Vector3> positionListDownLeft = new List<Vector3>();
List<Gesture> gestureAcceptedList = new List<Gesture>();
const float SwipeMinimalLength = 0.4f;
const float SwipeMaximalHeight = 0.2f;
const int SwipeMinimalDuration = 200;
const int SwipeMaximalDuration = 1500;
DateTime lastGestureDate = DateTime.Now;
int MinimalPeriodBetweenGestures = 0;
bool gestoAdelanteActivo = false;
bool gestoAtrasActivo = false;
bool gestoArribaActivo = false;
bool gestoArribaIzqActivo = false;
bool gestoAbajoActivo = false;
bool gestoAbajoIzqActivo = false;
```

Figura 36: Código variables y vectores.

Programación de gestos:

Gestos de mano derecha

Gesto derecha-izquierda

Es donde se detecta el movimiento de mano derecha a dirección de mano izquierda. Este gesto fue programado de la siguiente manera:

Se definió una función que detecta los movimientos realizados en la mano derecha en coordenadas X, su funcionamiento es recorrer la lista de posiciones (arreglo de posiciones), e ir comprobando si existe un conjunto de posiciones que puedan ser un gesto, comprueba si el conjunto de posiciones se encuentran a la misma altura y dirección correcta.

Luego de esto se comprueba si el movimiento es de longitud y velocidad adecuada. Y finalmente se establece si hubo o no un gesto de este tipo.

Dicha función se muestra en la figura 37.

```
bool SwipeRightToLeft()
  {int start = 0; for (int index = 0; index < positionListLeft.Count - 1; index++)
   {if ((Math.Abs(positionListLeft[0].Y - positionListLeft[index].Y) >
SwipeMaximalHeight) || (positionListLeft[index+1].X - positionListLeft[index].X <
-0.02f)) {  start = index; }if ((Math.Abs(positionListLeft[index].X -
positionListLeft[start].X) > SwipeMinimalLength)) {
                double totalMilliseconds = (positionListLeft[index].date -
positionListLeft[start].date).TotalMilliseconds;if (totalMilliseconds >=
SwipeMinimalDuration && totalMilliseconds <= SwipeMaximalDuration)
{
                  if (DateTime.Now.Subtract(lastGestureDate).TotalMilliseconds >
MinimalPeriodBetweenGestures)
     {
 gestureAcceptedList.Add(Gesture.SwipeRight);
 lastGestureDate = DateTime.Now;
                    positionListLeft.Clear();
                    return true;   }
           }
           }
           } return false;
}
```

Figura 37: Código función derecha a izquierda.

Función para definir las variables utilizadas en la detección del gesto, además de identificar si es este el gesto seleccionado.

Se utilizan comparaciones de las variables booleanas, que es la variable de retorno de esta función para detectar si no son gestos ya detectados en otra función y luego manda llamar la función *SwipeRightToLeft*, se compara si es verdadero o falso, para especificar qué operación hacer, que en este caso es presionar la tecla *right*, el código se muestra en la figura 38.

```
private void GestoAvanceAdelante(Joint head, Joint rightHand, Joint leftHand)
      {if (!gestoAtrasActivo && !gestoAdelanteActivo)
         {if (!gestoAtrasActivo && !gestoArribaActivo)
            {if (!gestoAtrasActivo && !gestoArribaIzqActivo)
              { if (!gestoAtrasActivo && !gestoAbajoActivo)
                  {if (!gestoAtrasActivo && !gestoAbajoIzqActivo)
                     {positionListRight.Add(new Vector3()
                   {X = rightHand.Position.X,
                     Y = rightHand.Position.Y,
                     Z = rightHand.Position.Z,
                     date = DateTime.Now});
                   if (SwipeRightToLeft())
                   {gestoAdelanteActivo = true;
                    System.Windows.Forms.SendKeys.SendWait("{Right}"); }
                   if (positionListRight.Count() > 20)
                   {positionListRight.RemoveAt(0); }
                }
              }
         }   else
         {gestoAdelanteActivo = false; }
      }
```

Figura 38: Código detección movimiento derecha a izquierda.

Gesto de mano derecha-arriba

En este gesto se detecta el movimiento de la mano derecha, posicionada en la coordenada Y, desde abajo hacia arriba. Este gesto fue programado de la siguiente manera:

Al igual que el gesto anterior se definió una función que detecta los movimientos realizados en la mano derecha en coordenadas Y, pero en dirección de abajo a arriba, su funcionamiento es recorrer la lista de posiciones (arreglo de posiciones), e ir

comprobando si existe un conjunto de posiciones que puedan ser un gesto, comprueba si el conjunto de posiciones se encuentran a la misma posición en X y dirección correcta.

Luego de esto se comprueba si el movimiento es de longitud y velocidad adecuada. Finalmente se establece si hubo o no un gesto de este tipo. Dicha función se muestra en la figura 39.

```
bool SwipeUp()
      {int start = 0;
         for (int index = 0; index < positionListUp.Count - 1; index++)
         {if ((Math.Abs(positionListUp[0].X - positionListUp[index].X) >
SwipeMinimalLength) || (positionListUp[index+1].Y - positionListUp[index].Y > -
0.04f))
 {start = index; }
if ((Math.Abs(positionListUp[index].Y - positionListUp[start].Y) >
SwipeMaximalHeight))
            {double totalMilliseconds = (positionListUp[index].date -
positionListUp[start].date).TotalMilliseconds;
 if (totalMilliseconds >= SwipeMinimalDuration && totalMilliseconds <=
SwipeMaximalDuration)
               {if (DateTime.Now.Subtract(lastGestureDate).TotalMilliseconds >
MinimalPeriodBetweenGestures)
                  { gestureAcceptedList.Add(Gesture.SwipeUp);
                     lastGestureDate = DateTime.Now;
                     positionListUp.Clear();
                     return true; }
               }
            }
         }return false;
 }
```

Figura 39: Código función derecha dirección arriba.

Función “detección de gesto”:

Función que se utiliza para definir las variables utilizadas en la detección del gesto, además de identificar si es este el gesto seleccionado. Al igual que el gesto anterior utiliza varias comparaciones para determinar que sea éste el gesto adecuado y finalmente manda llamar la función *SwipeUptoDown,* para determinar si éste es el gesto correcto y ejecutar las instrucciones correspondientes que en este caso es presionar tecla “+” que significa “acercar”, el código se muestra en la figura 40.

```
private void GestoAvanceArriba(Joint head, Joint rightHand, Joint leftHand)
      {if (!gestoArribaActivo && !gestoAtrasActivo)
         {if (!gestoArribaActivo && !gestoAdelanteActivo)
            {if (!gestoArribaActivo && !gestoArribaIzqActivo)
               { if (!gestoArribaActivo && !gestoAbajoActivo)
                  {if (!gestoArribaActivo && !gestoAbajoIzqActivo)
                     {positionListUp.Add(new Vector3()
                     {X = rightHand.Position.X,
                     Y = rightHand.Position.Y,
                     Z = rightHand.Position.Z,
                     date = DateTime.Now
                  });
                  if (SwipeUp())
                  {gestoArribaActivo = true;
                     System.Windows.Forms.SendKeys.SendWait("{+}");}
                  if (positionListUp.Count() > 20)
                  {positionListUp.RemoveAt(0); }
               }
            }
         }
         else
         {gestoArribaActivo = false; }
      }
```

Figura 40: Código detección movimiento derecha dirección arriba abajo.

Gestos de mano izquierda

Gesto de mano izquierda-derecha

Es donde se detecta el movimiento de mano izquierda a dirección de mano derecha. Éste gesto fue programado de la siguiente manera:

En éste gesto se utilizó el mismo procedimiento que en el gesto "derecha a izquierda", solo que se cambiaron las variables X, Y, Z del vector de lista de posiciones, se cambiaron de *rightHand.Position.*X a X = *leftHand.Position*.X, es decir las variables se cambian de detección mano derecha a mano izquierda, además se cambió el sentido de detección por "de mano izquierda" a "mano derecha" (ver figura 41).

```
bool SwipeLeftToRight()
      {int start = 0;
         for (int index = 0; index < positionListRight.Count - 1; index++)
         {if ((Math.Abs(positionListRight[0].Y - positionListRight[index].Y) >
SwipeMaximalHeight) || (positionListRight[index+1].X -
positionListRight[index].X > -0.02f))
{start = index; }
if ((Math.Abs(positionListRight[index].X - positionListRight[start].X) >
SwipeMinimalLength))
{double totalMilliseconds = (positionListRight[index].date -
positionListRight[start].date).TotalMilliseconds;
 if (totalMilliseconds >= SwipeMinimalDuration && totalMilliseconds <=
SwipeMaximalDuration) {
  if (DateTime.Now.Subtract(lastGestureDate).TotalMilliseconds >
MinimalPeriodBetweenGestures)
                  { gestureAcceptedList.Add(Gesture.SwipeLeft);
                     lastGestureDate = DateTime.Now;
                     positionListRight.Clear();
                     return true; }
               }
            }
         }return false;
      }
```

Figura 41: Código función izquierda a derecha.

Función para detectar si se utilizó este gesto:
Función que igual que las anteriores define las variables utilizadas en la detección del gesto, además de identificar si es éste el gesto seleccionado. Al igual que los gestos anteriores utiliza varias comparaciones para determinar que sea éste el gesto adecuado.

Finalmente manda llamar la función *SwipeLeftToRight,* y determina si es el gesto correcto y ejecuta la instrucción correspondiente que en éste caso es presionar la tecla "*left*", que significa presionar la tecla izquierda para cambiar a la siguiente diapositiva de la presentación, su código se muestra en la figura 42.

```
private void GestoAvanceAtras(Joint head, Joint rightHand, Joint leftHand)
      {if (!gestoAdelanteActivo && !gestoAtrasActivo)
         {if (!gestoAdelanteActivo && !gestoArribaActivo)
            {if (!gestoAdelanteActivo && !gestoArribaIzqActivo)
                  {if (!gestoAdelanteActivo && !gestoAbajoActivo)
                  {if (!gestoAdelanteActivo && !gestoAbajoIzqActivo)
                   {positionListLeft.Add(new Vector3()
                   {X = leftHand.Position.X,
                      Y = leftHand.Position.Y,
                      Z = leftHand.Position.Z,
                      date = DateTime.Now});
                  if (SwipeLeftToRight())
                  {gestoAtrasActivo = true;
                     System.Windows.Forms.SendKeys.SendWait("{Left}");}
                  if (positionListLeft.Count() > 20)
                  {positionListLeft.RemoveAt(0); }
               }
            }
         }else
         {gestoAtrasActivo = false; }
      }
```

Figura 42: Código detección movimiento izquierda a derecha.

Gesto de mano izquierda-arriba

En este gesto se detecta el movimiento de la mano izquierda, posicionada en la coordenada Y, desde arriba hacia abajo. Este gesto fue programado de la siguiente manera: Al igual que el gesto anterior se definió una función que detecta los movimientos realizados en la mano derecha en coordenadas Y, pero en dirección de abajo hacia a arriba, su funcionamiento es recorrer la lista de posiciones (arreglo de posiciones), e ir comprobando si existe un conjunto de posiciones que puedan ser un gesto, comprueba si el conjunto de posiciones se encuentran a la misma posición en X y dirección correcta. Dicha función se muestra en la figura 43.

```
bool SwipeUptoDownLeft()
      {int start = 0;
         for (int index = 0; index < positionListUpLeft.Count - 1; index++)
         {if ((Math.Abs(positionListUpLeft[0].X - positionListUpLeft[index].X) >
SwipeMinimalLength) || (positionListUpLeft[index+1].Y -
positionListUpLeft[index].Y > -0.04f)) {start = index; }
if ((Math.Abs(positionListUpLeft[index].Y - positionListUpLeft[start].Y) >
SwipeMaximalHeight))
 {double totalMilliseconds = (positionListUpLeft[index].date -
positionListUpLeft[start].date).TotalMilliseconds;
  if (totalMilliseconds >= SwipeMinimalDuration && totalMilliseconds <=
SwipeMaximalDuration)
  {if (DateTime.Now.Subtract(lastGestureDate).TotalMilliseconds >
MinimalPeriodBetweenGestures)
                  {gestureAcceptedList.Add(Gesture.SwipeUpLeft);
                     lastGestureDate = DateTime.Now;
                     positionListUpLeft.Clear();
                     return true; }
               }
            }
         }return false;
      }
```

Figura 43: Código función izquierda de arriba.

Función detectora de gestos:

Esta función es como las anteriores de este tipo, que como resultado detectan si se utilizó este gesto y ejecuta las instrucciones deseadas para el gesto, que en este caso es cambiar subir el volumen de los elementos multimedia, presionando las teclas *alt* y la tecla flecha arriba.

El código se muestra en la figura 44.

```
private void GestoAvanceArribaIzq(Joint head, Joint rightHand, Joint leftHand)
      {if (!gestoArribaIzqActivo && !gestoAtrasActivo)
         {if (!gestoArribaIzqActivo && !gestoAdelanteActivo)
            {if (!gestoArribaIzqActivo && !gestoArribaIzqActivo)
               {if (!gestoArribaIzqActivo && !gestoAbajoActivo)
                  {if (!gestoArribaIzqActivo && !gestoAbajoIzqActivo)
                     {positionListUpLeft.Add(new Vector3()
                  {
                  X = leftHand.Position.X,
                     Y = leftHand.Position.Y,
                     Z = leftHand.Position.Z,
                     date = DateTime.Now
          });
                  if (SwipeUptoDownLeft())
                  {gestoArribaIzqActivo = true;
                     System.Windows.Forms.SendKeys.SendWait("%(up)");}
                  if (positionListUpLeft.Count() > 20)
                  {positionListUpLeft.RemoveAt(0); }
               }
            }
          }
         else
          {
   gestoArribaIzqActivo = false;
       }
     }
```

Figura 44: Código detección movimiento izquierda arriba.

3.3.3.3 Comandos de voz:

Se utilizó comando de voz en lenguaje español México, en este método se utilizan palabras claves, que activan eventos tales como seleccionar elemento multimedia, iniciar, detener, *enter* (abrir), alejar, restaurar, mudo volumen y cerrar aplicación.

Primero se inicializaron los eventos a utilizar para inicializar el *KINECT*, esto se muestra en la figura 45.

```
void MainWindow_Loaded(object sender, RoutedEventArgs e)
      {sen = KinectSensor.KinectSensors.FirstOrDefault();
        sen.SkeletonFrameReady += new
EventHandler<SkeletonFrameReadyEventArgs>(sensor_SkeletonFrameReady);
        conectaActiva();}
```

Figura 45: *Main Windows*.

Se estableció la función "conecta activa" que nos permite activar los componentes del *KINECT* que se utilizan, tales como el paquete de idioma utilizado, el algoritmo de reconocimiento y los comandos a utilizar, como se muestra en la figura 46.

```
        void conectaActiva()
            {if (KinectSensor.KinectSensors.Count > 0)
               {if (this._sensor == null)
                  {this._sensor = KinectSensor.KinectSensors[0];
                     if (this._sensor != null)
                     {try
                        {this._sensor.Start();
                           _sensor.ElevationAngle = 3;
                         }
                        catch (Exception ex)
                        {statusK.Text = ex.Message.ToString();
                         }
                        RecognizerInfo ri = obtenerLP();
                        if (ri != null)
                        {this.speechengine = new SpeechRecognitionEngine(ri.Id);
                           var opciones = new Choices();
```

```
                            opciones.Add("uno", "Cerrar");
                             opciones.Add("dos", "Abrir");
                             opciones.Add("tres", "Acercar");
                             opciones.Add("cuatro", "Alejar");
                             opciones.Add("cinco", "Detener");
                             opciones.Add("seis", "Iniciar");
                             opciones.Add("siete", "Seleccionar");
                             opciones.Add("ocho", "Restaurar");
                             opciones.Add("nueve", "Mudo");
            var grammarb = new GrammarBuilder { Culture = ri.Culture };
            grammarb.Append(opciones);
             var grammar = new Grammar(grammarb); grammar
             this.speechengine.LoadGrammar(grammar);
             speechengine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(speechengine_SpeechRecognized);
speechengine.SetInputToAudioStream(_sensor.AudioSource.Start(), new
SpeechAudioFormatInfo(EncodingFormat.Pcm, 16000, 16, 1, 32000, 2, null));
                             speechengine.RecognizeAsync(RecognizeMode.Multiple);
                          }
                        }
                      }
                    }
                }private RecognizerInfo obtenerLP()
                {foreach (RecognizerInfo recognizer in
SpeechRecognitionEngine.InstalledRecognizers())
                    {string value;
                       recognizer.AdditionalInfo.TryGetValue("Kinect", out value);
                      if ("True".Equals(value, StringComparison.OrdinalIgnoreCase) &&
"es-MX".Equals(recognizer.Culture.Name, StringComparison.OrdinalIgnoreCase))
                       {return recognizer;
                       } }return null;}
```

Figura 46: función conecta activa.

Luego de establecer los comandos a utilizar, se tiene la siguiente función que permite identificar el comando requerido en la aplicación. En esta función se ejecutarán las instrucciones requeridas para cada comando. Se muestra en la figura 47.

```
void speechengine_SpeechRecognized(object sender, SpeechRecognizedEventArgs
e)
                { const double igualdad = 0.5;
                   if (e.Result.Confidence > igualdad)
                   { switch (e.Result.Words[0].Text)
                      { case  "Cerrar":
                         PowerPoint.Application application = new
                   PowerPoint.Application();
                    application.Quit();
                   sensor.Stop();
                     this.Close();
                     break;
               case "Abrir":
                System.Windows.Forms.SendKeys.SendWait("{Enter}");
                   break;
                  case "Acercar":
                   System.Windows.Forms.SendKeys.SendWait("{+}");
                   break;
                    case "Alejar":
                    System.Windows.Forms.SendKeys.SendWait("{-}");
                   break;
                   case "Detener":
                   System.Windows.Forms.SendKeys.SendWait("%(q)");
                   break;
                   case "Iniciar":
                   System.Windows.Forms.SendKeys.SendWait("%(c)");
                   break;
                   case "Seleccionar":
                   System.Windows.Forms.SendKeys.SendWait("{TAB}");
                   break;
```

```
case "Restaurar":
System.Windows.Forms.SendKeys.SendWait("{ESC}");
break;
case "Mudo":
System.Windows.Forms.SendKeys.SendWait("%{u}");
break; }
  }
 }
      void kinectSensorChooser_KinectSensorChanged(object sender,
DependencyPropertyChangedEventArgs e)
                {KinectSensor sensor = (KinectSensor)e.NewValue;
                        if (sensor == null)
                        {return; }
TransformSmoothParameters parameters = new TransformSmoothParameters();
                        parameters.Smoothing = 0.7f;
                        parameters.Correction = 0.3f;
                        parameters.Prediction = 0.4f;
                        parameters.JitterRadius = 1.0f;
                        parameters.MaxDeviationRadius = 0.5f;
sensor.SkeletonStream.Enable(parameters);
sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
sensor.DepthStream.Enable(DepthImageFormat.Resolution320x240Fps30);
                }
}
}
```

Figura 47: función *SpeechRecognized*

3.3.2 Selección de variables:

Las variables seleccionadas se dividen en: conceptos, indicadores, variables y valores, en donde los conceptos son las variables principales utilizadas a lo largo del proyecto. Los indicadores hacen referencia a la usabilidad de la aplicación específicamente, las variables son las requeridas en la aplicación para su funcionamiento.

Los valores son los utilizados para calificar los aspectos de la aplicación al momento de probarla. Dichas variables son mostradas en la tabla 1.

Conceptos	Indicadores	Variables	Valores
NUI	Fácil de usar	Sensor	Si
KINECT	Presentación dinámica	Coordenada X,Y,Z	No
Gestos	Confiable	Vector 3	Muy interesante
Comandos de voz		Mano izquierda	Interesante
Esqueleto		Mano derecha	No interesante
Articulación		Cabeza	
Imagen de profundidad			
Diapositivas			

Tabla 1. Conceptos, indicadores, variables y valores.

3.3.3 Procedimiento:

En este apartado se muestran los procedimientos utilizados en la investigación, los tipos de instrumentos aplicados para obtener la información requerida.

Primero se recolectaron datos de personas ajenas al proyecto, a las cuales se les hicieron cuestionarios para saber si les agradaría el contar con una herramienta para *PowerPoint*, como esta aplicación y saber si era o no viable el proyecto, esta encuesta se muestra en el anexo I.

En este proyecto se consideró tomar los datos desde usuarios, ya que se consideró hacer pruebas de la aplicación y luego practicarles un cuestionario en la que se obtuvieran datos del comportamiento de la aplicación. La población en la que se basa la investigación es grande por lo que se decidió establecer solo una pequeña muestra a la cual se aplicó los cuestionarios. Dicho cuestionario contiene preguntas cerradas y concretas, en donde el usuario respondió fácilmente, se muestra en el anexo II.

También se hicieron casos de estudio para examinar el comportamiento de la aplicación con el usuario, y se detectaron los posibles errores que se podrían tener dentro de ella. Se especifica el comportamiento de la aplicación en un entorno de escritorio normal, en donde el investigador trata de plasmar todos los entornos que se puedan presentar en un entorno real, utilizando los esquemas de casos de uso tal como se muestra en el anexo III.

3.3.4 Recolección de información:

En este apartado se muestran los métodos utilizados para la recolección de datos de la presente investigación.

Se aplicaron los cuestionarios mencionados anteriormente, al tenerlos ya contestados se separaron por respuestas y se contaron cada una de las respuestas de cada pregunta, teniendo como resultado el total de respuestas positivas y negativas. Teniendo estos resultados se llenó una tabla por cada pregunta donde se muestran las respuestas posibles y el número de personas que los respondió.

Finalmente la información recolectada fue registrada en tablas y gráficas de *Word 2013*, las cuales se muestran en el capítulo 4.

3.3.5 Aspectos éticos:

En este apartado se muestran los aspectos éticos considerados para la investigación y desarrollo del proyecto.

En primer lugar se ha considerado enfocar el presente proyecto a la comunidad estudiantil, para utilización con estudiantes de la universidad. Ya que es una población que en su mayoría utiliza el programa *PowerPoint* y a la que le agradaría mucho el utilizar esta aplicación.

Los aspectos éticos que se respetaron en el proceso de desarrollo de este proyecto es el solo utilizar ejemplos y artículos de programación en donde el autor permita utilizarlos como referencia. Se utilizan códigos y ejemplos de *Microsoft*, libros y artículos tecnológicos, los cuales permiten su libre uso. Las personas involucradas en la presente investigación dieron su total consentimiento para exponer los resultados de las encuestas, críticas y observaciones.

Capítulo 4. Resultados de la investigación

Dentro del presente capítulo se muestran los resultados obtenidos dentro de la investigación, los cuales fueron obtenidos de los cuestionarios realizados después de probar la aplicación desarrollada en el proyecto y a personas que no sabían nada acerca de ella.

4.1 Presentación de resultados

En la presente investigación se utilizaron dos tipos de cuestionarios, el primer cuestionario fue respondido por personas que no conocían la aplicación y el segundo cuestionario fue realizado por personas que utilizaron la aplicación, a las cuales se les dio una pequeña sección de prueba en la que se les mostró el funcionamiento de la aplicación, tal como se muestra en la figura 48.

Figura 48: Prueba de la aplicación

El primer cuestionario consta de cinco preguntas las cuales se enfocan en qué es lo que opinan de las presentaciones en forma general, qué tanto les gustaría que se desarrollara una aplicación como la presente y qué opinarían de ella. De esta manera saber si la presente aplicación sería aceptada o no en el ámbito escolar. Este cuestionario se muestra en el anexo I.

La segunda encuesta fue aplicada después de que las personas probaron la aplicación tal como se muestra en la figura 49, ésta consta de diez preguntas las cuales se enfocan en saber qué tanto les agrado el sistema, qué tan difícil fue su utilización, qué tan cómodo fue, si lo recomendarían, qué no les gusto y qué tanto les gusto, etcétera. Dicho cuestionario se encuentra en el anexo II.

Figura 49: Prueba de la aplicación

A continuación se presentará el análisis de los resultados obtenidos en las encuestas antes mencionadas.

4.2 Análisis e interpretación de resultados

Los resultados en las encuestas son analizados en las siguientes tablas y gráficas.

Cuestionario 1:
Este cuestionario fue aplicado a cincuenta estudiantes del Instituto de Ingeniería y Tecnología (IIT).

1. **¿Utilizas el *PowerPoint*?**

Esta pregunta muestra la importancia que tiene el manejo de *PowerPoint* en la universidad, en la que la mayoría de los encuestados respondió que sí lo utilizaba, tal como se muestra en la tabla 2 y figura 50.

Respuestas recibidas

Si	No
47	3

Tabla 2. Respuestas de pregunta 1 cuestionario 1.

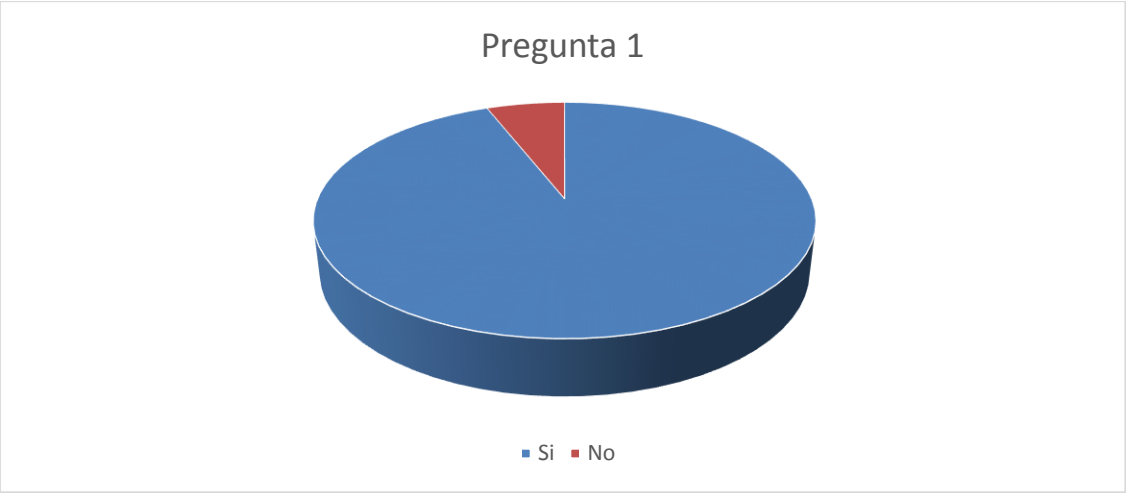


Figura 50: Gráfica resultados pregunta 1

2. ¿Crees que es aburrido?

Esta pregunta muestra que las personas encuestadas consideran que no es aburrido, pero un treinta por ciento considera que si lo es.

Por lo que se puede concluir que la mayoría de las exposiciones no son aburridas. Los resultados se muestran en la tabla 3 y en la figura 51.

Respuestas recibidas

Si	No
15	35

Tabla 3. Respuestas de pregunta 2 cuestionario 1.

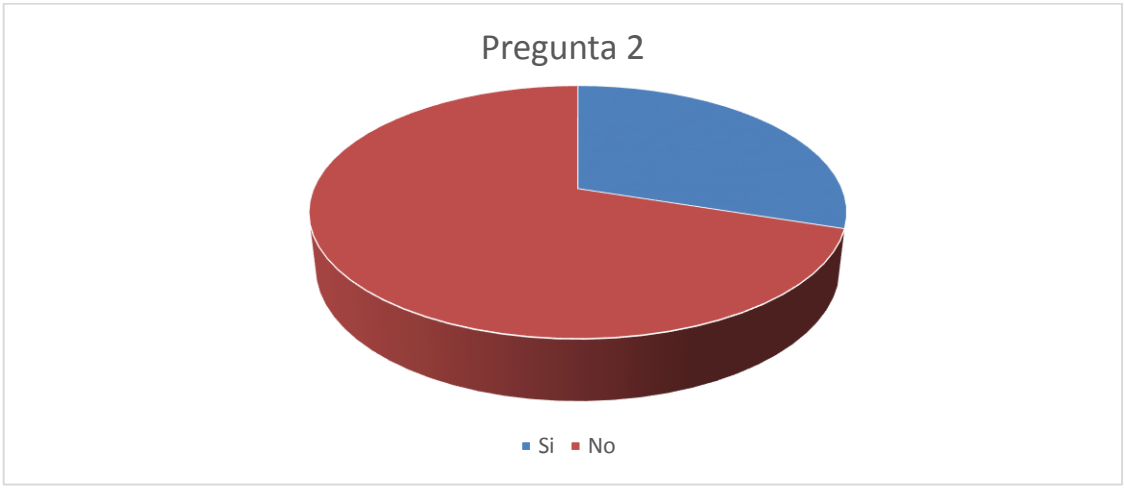


Figura 51: Gráfica resultados pregunta 2

3. ¿Utilizas dispositivos inalámbricos para tus presentaciones?

En esta pregunta se muestra que hay muchas personas que utilizan aparatos inalámbricos para el control de presentaciones, pero un sesenta por ciento de las personas encuestadas no los utilizan. Tal como se muestra en las tabla 4 y figura 52.

Respuestas recibidas

Si	No
20	30

Tabla 4. Respuestas de pregunta 3 cuestionario 1.

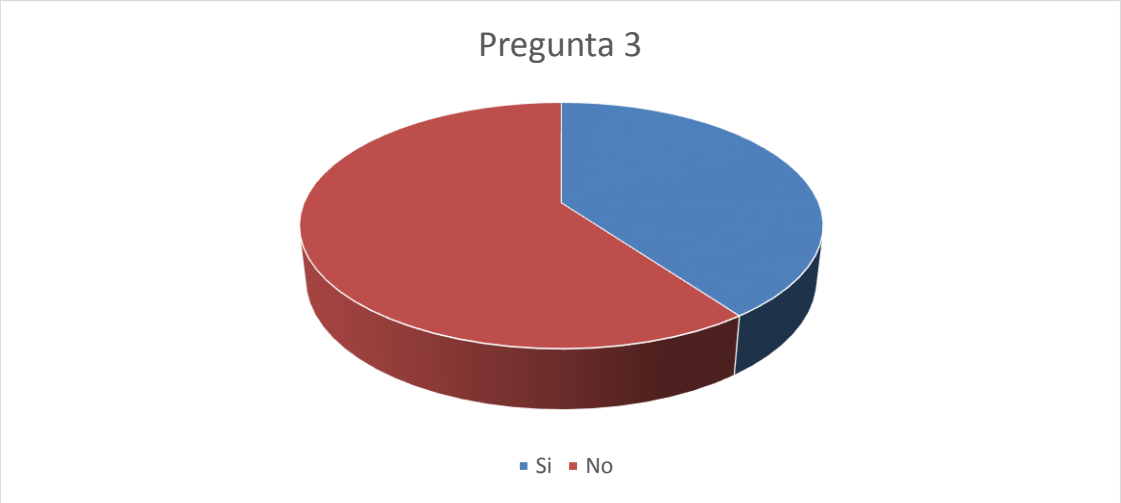


Figura 52: Gráfica resultados pregunta 3

4. ¿Te gustaría manejar la presentación sin necesidad de ningún dispositivo?

En esta pregunta se muestra que a las personas encuestadas si le gustaría el manejar sus presentaciones sin tener que utilizar un dispositivo inalámbrico. Que es muy probable que la aplicación sea aceptada en la población encuestada. Tal como se muestra en la tabla 5 y figura 53.

Respuestas recibidas

Si	No
44	6

Tabla 5. Respuestas de pregunta 4 cuestionario 1.

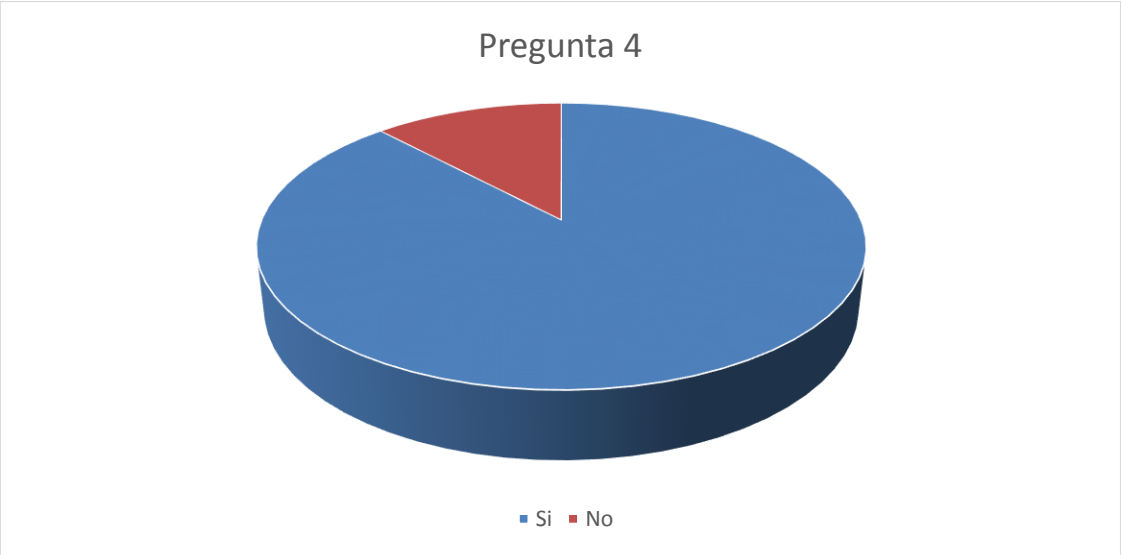


Figura 53: Gráfica resultados pregunta 4

5. ¿Crees que el *KINECT* se pueda utilizar para manejar *PowerPoint*?

Esta pregunta demuestra que las personas creen que si es posible utilizar *KINECT* para el manejo de *PowerPoint (*tabla 6 y figura 54).

Respuestas recibidas

Si	No
39	11

Tabla 6. Respuestas de pregunta 5 cuestionario 1.

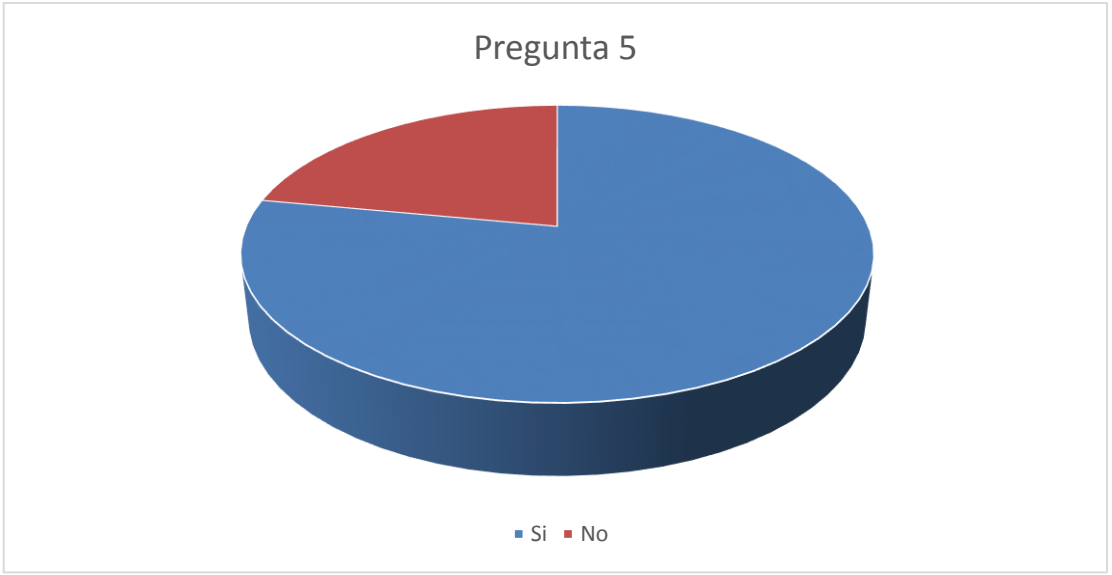


Figura 54: Gráfica resultados pregunta 5

Conclusión final cuestionario 1:

Como se creía en el principio de la investigación, *PowerPoint* es utilizado en la mayoría de las presentaciones en la universidad, por lo que la presente aplicación debe de tener éxito en el ámbito universitario, a pesar de que solo a un treinta por ciento de los encuestados no les parecen aburridas las presentaciones, a la mayoría de la población encuestada le gustaría manejar sus presentaciones si ningún dispositivo inalámbrico y le gustaría utilizar el *KINECT* para ello, utilizando una aplicación como la presente, a pesar de que muchos de los encuestados ya utilizan dispositivos inalámbricos.

Por lo que se concluyó que es factible la elaboración de la presente aplicación y que realmente sería aceptada por los estudiantes del IIT.

Cuestionario 2:

Este cuestionario fue aplicado a veintidós estudiantes del Instituto de Ingeniería y Tecnología (IIT) después de haber probado la aplicación.

1. **¿Qué te pareció el sistema, tu primera impresión?**

Los resultados arrojan que a las personas encuestadas si les gustó la aplicación; a la mayoría les pareció muy interesante. Tal como se demuestra en la tabla 7 y figura 55.

Respuestas recibidas

Muy interesante	Interesante	No me interesa
17	5	0

Tabla 7. Respuestas de pregunta 1 cuestionario 2.

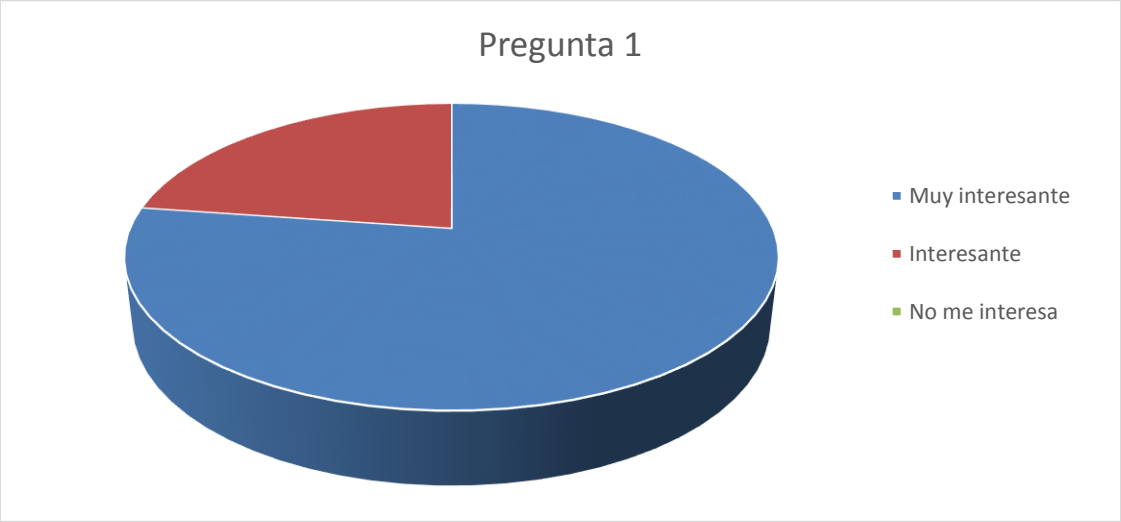


Figura 55: Gráfica resultados pregunta 1

2. ¿Fue cómodo el sistema?

A la mayoría le pareció cómodo el manejo de la aplicación, solo a 2 personas les pareció un poco incómodo, pues se cansaron un poco al hacer los gestos (tabla 8 y figura 56). Respuestas recibidas.

Si	Un poco	No
20	2	0

Tabla 8. Respuestas de pregunta 2 cuestionario 2.

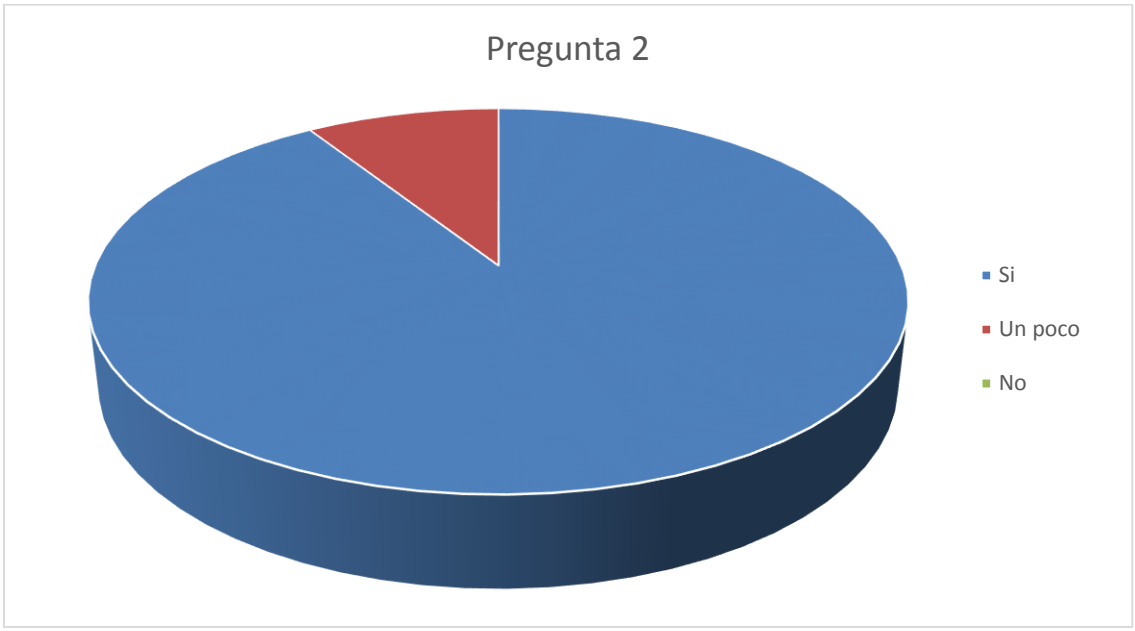


Figura 56: Gráfica resultados pregunta 2

3. ¿Fue cansado utilizarlo?

Esto nos demuestra que la aplicación puede llegar a ser un poco cansada, aunque a la mayoría no le molestó, como se ve en la tabla 9 y figura 57.

Si	Un poco	No
1	5	16

Tabla 9. Respuestas de pregunta 3 cuestionario 2.

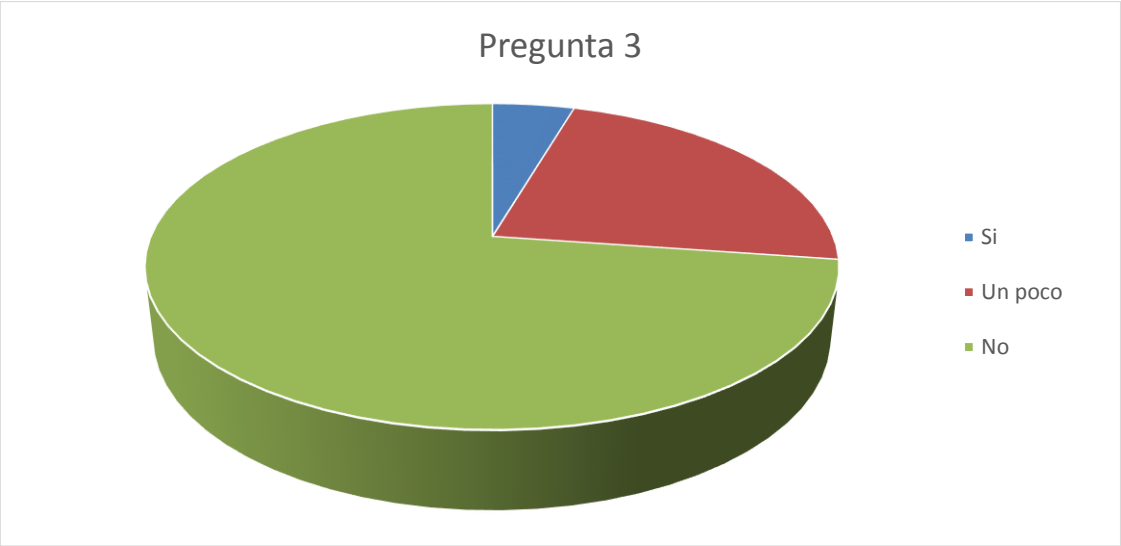


Figura 57: Gráfica resultados pregunta 3

4. ¿El sistema hacia lo que debía?

La gran mayoría concluyó en que sí hacia lo que debía hacer al momento de estar haciendo los gestos y ordenando los comandos (tabla 10 y figura 58).

Respuestas recibidas

Si	Un poco	No
21	1	0

Tabla 10. Respuestas de pregunta 4 cuestionario 2.

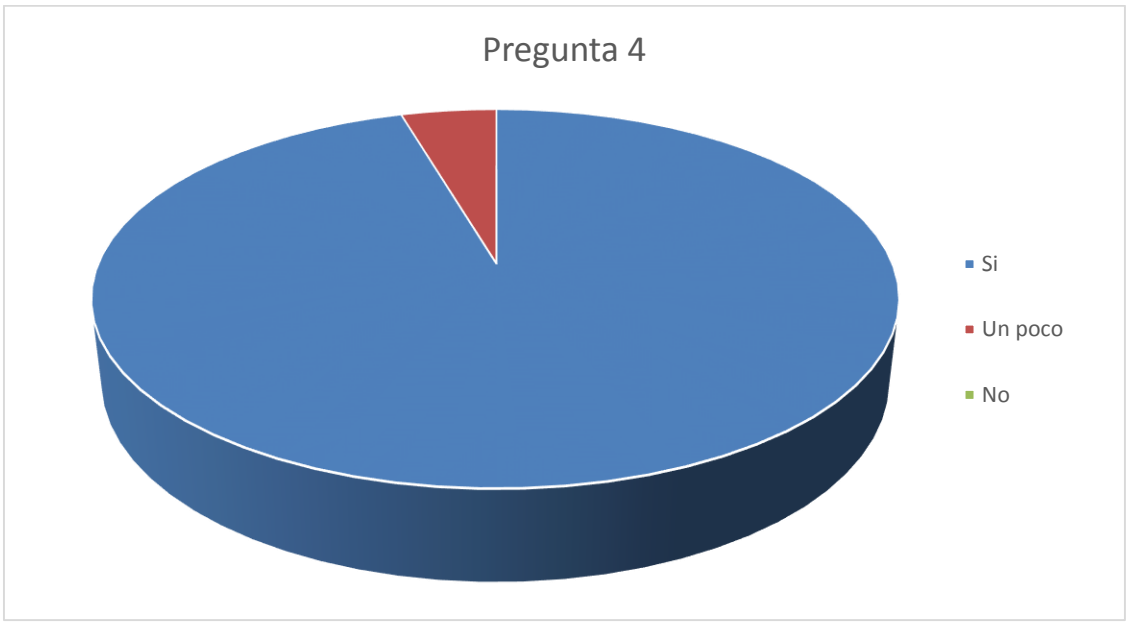


Figura 58: Gráfica resultados pregunta 4

5. ¿El sistema fue fácil de usar?

La gran mayoría concluyo en que la aplicación sí es fácil de utilizar, que los comandos y gestos son sencillos (tabla 11 y figura 59).

Respuestas recibidas

Si	Un poco	No
21	1	0

Tabla 11. Respuestas de pregunta 5 cuestionario 2.

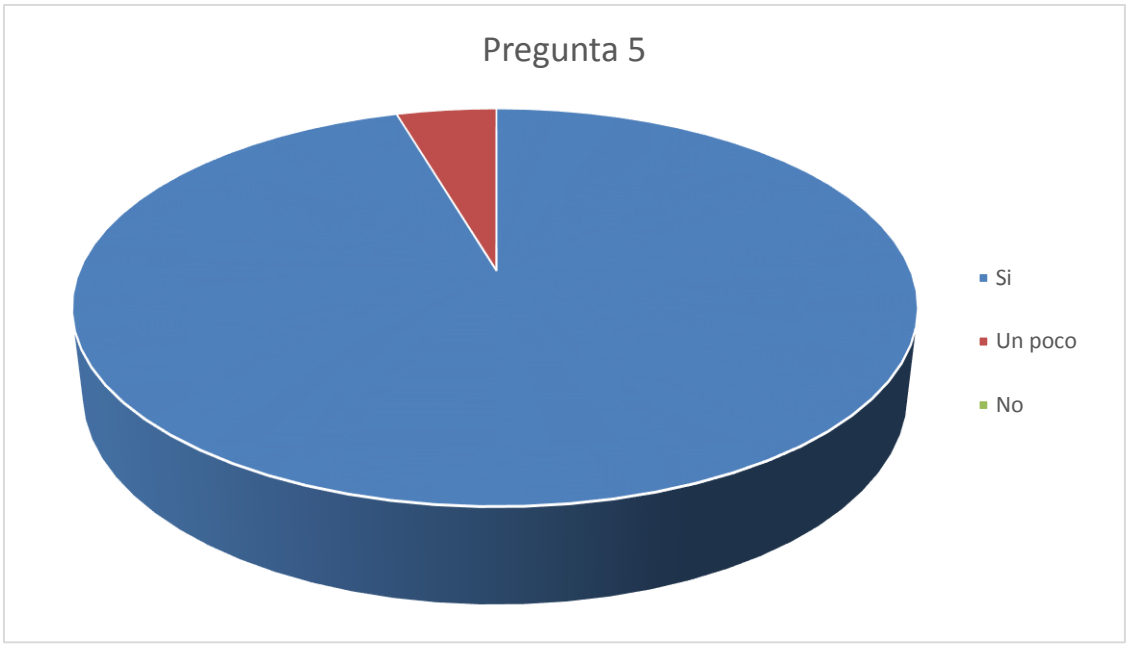


Figura 59: Gráfica resultados pregunta 5

6. ¿Lo utilizarías?

La mayoría quiere utilizar esta aplicación en sus presentaciones escolares (tabla 12 y figura 60).

Respuestas recibidas

Si	Tal vez	No
21	0	1

Tabla 12. Respuestas de pregunta 6 cuestionario 2.

Figura 60: Gráfica resultados pregunta 6

7. ¿Recomendarías los sistemas?

La mayoría recomendaría el sistema ya que les gustó y les agrada la interfaz (tabla 13 y figura 61).

Respuestas recibidas

Si	No se	No
21	1	0

Tabla 13. Respuestas de pregunta 7 cuestionario 2.

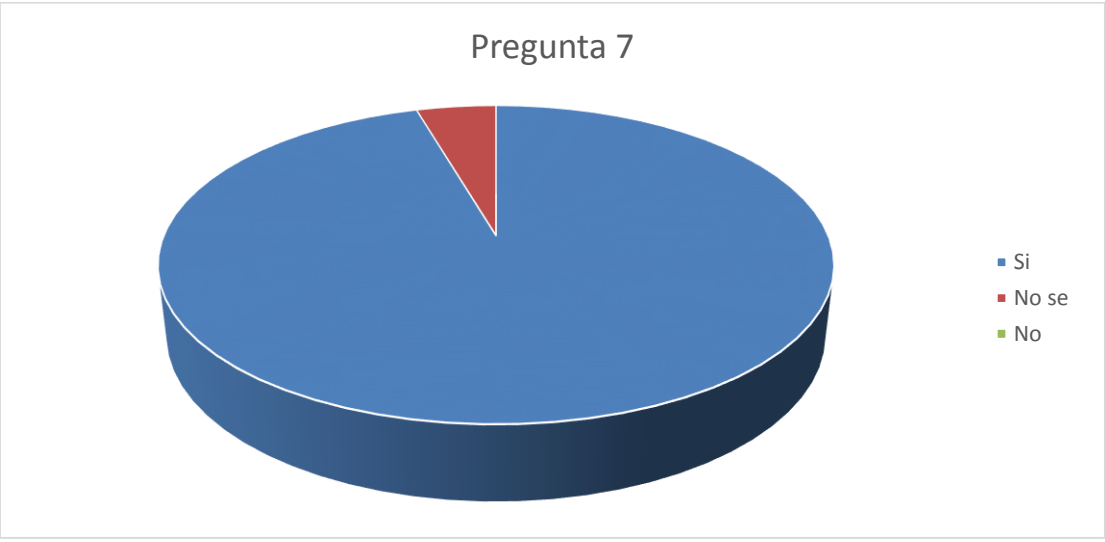


Figura 61: Gráfica resultados pregunta 7

8. ¿Qué le cambiarías o mejorarías al sistema?

A la mayoría no les agradaron los comandos de voz, quieren que sean más eficaces al estar exponiendo, a otros no les agradó el color de la aplicación, a la minoría no le agradó que es necesario estar un poco lejos del *KINECT*. Y a muchos si les gustó todo (tabla 14 y figura 62).

Respuestas recibidas

Comandos de voz	Calibrar el sensor	Gesto	Interfaz(color)	Nada
6	2	2	5	7

Tabla 14. Respuestas de pregunta 8 cuestionario 2.

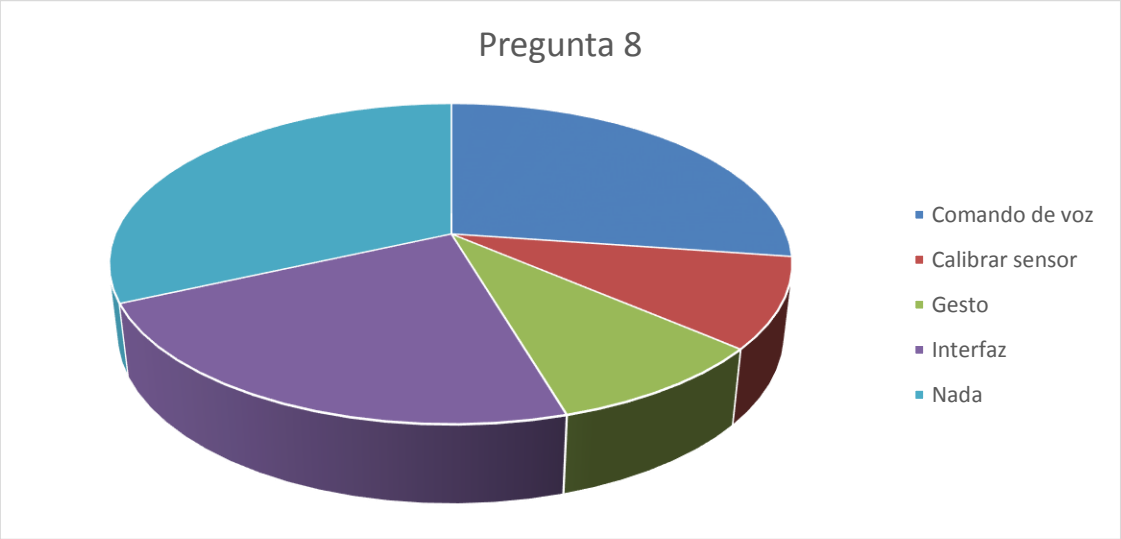


Figura 62: Gráfica resultados pregunta 8

9. ¿Qué fue lo que más te gustó del sistema?

Los aspectos que más les gustaron a los encuestados fueron los comandos de voz y los gestos, ya que les agrado el poder manejar con su propio cuerpo la presentación, tal como se muestra en la tabla 15 y figura 63.

Respuestas recibidas

Todo	Comandos de voz	Concepto futurístico	Salir en la aplicación	Gesto
10	6	1	1	4

Tabla 15. Respuestas de pregunta 9 cuestionario 2.

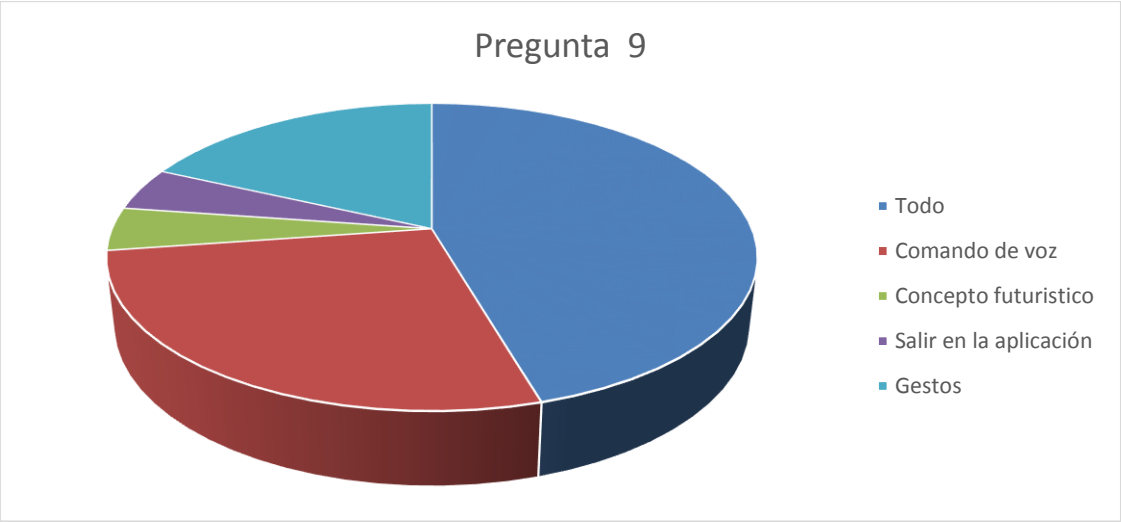


Figura 63: Gráfica resultados pregunta 9

10. ¿Qué fue lo que menos te gustó del sistema?

A la mayoría les gustó todo, solo un veintisiete porciento tuvo problemas para que les detectara el comando de voz, ya que los decían en un volumen algo bajo. Y a algunos les gustaría cambiar el color de la aplicación (tabla 16 y figura 64)

Respuestas recibidas

Interfaz inicial	Comandos de voz	No entendí	Ajustes	Nada
3	6	3	1	9

Tabla 16. Respuestas de pregunta 10 cuestionario 2.

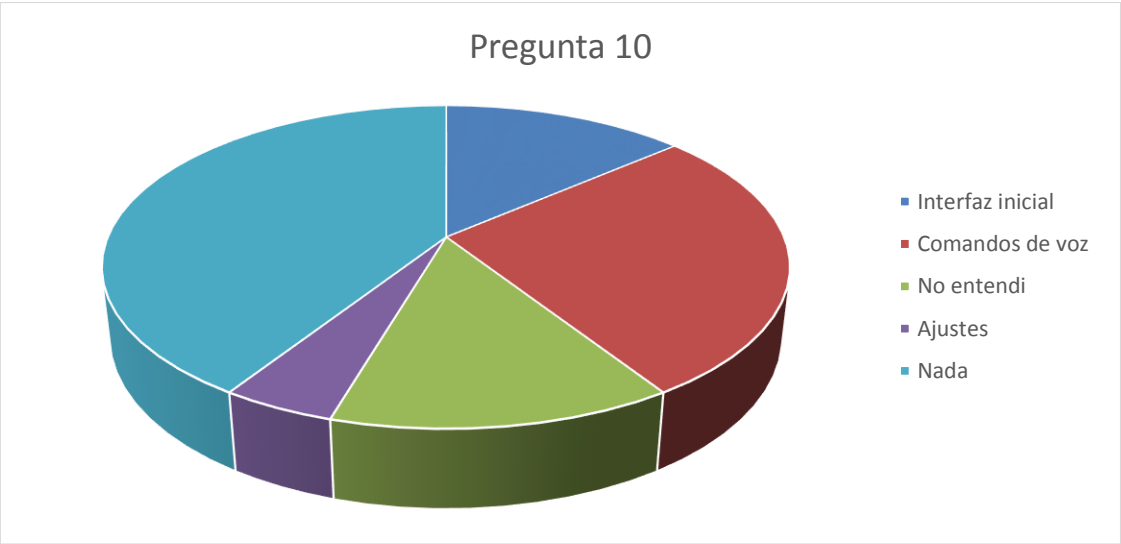


Figura 64: Gráfica resultados pregunta 10

Conclusión final cuestionario 2:

La aplicación logró satisfacer a la mayoría de los usuarios con los que se realizaron las pruebas, una de las cuestiones que se dieron a notar, fue que la distancia del uso de la aplicación varía conforme a la altura del usuario.

En sí los cambios que se sugirieron fueron menores, relacionadas con el cambio de los colores de la interfaz, pero en general los resultados fueron satisfactorios.

Capítulo 5. Discusiones, conclusiones y recomendaciones

En este capítulo se describen las discusiones, conclusiones y las recomendaciones relacionadas con el presente proyecto.

5.1 Con respecto a las preguntas de investigación

Las preguntas de investigación planteadas en el capítulo 1, fueron respondidas una por una conforme fue desarrollada la investigación. Sus respuestas se muestran a continuación.

- ¿Cómo se comunicará *Windows 8* con el *KINECT*?

  Mediantes el uso de las herramientas de *Microsoft* como *Visual Studio 2012* y el *Kinect SDK.*

- ¿Cómo se comunicará *Microsoft Office* con *KINECT?*

  Utilizando la librería *Microsoft.Office.Interop.powerpoint.* y mandando llamar funciones de esta librería por medio de los gestos y comandos de voz, reconocidos por el *Kinect SDK* , además de la utilización de comandos de teclas llamadas por gestos y comandos también.

- ¿Qué herramientas se necesitarán para el desarrollo?

  Computadora con las siguientes especificaciones: 64 *bits*(x64), procesador de 2,10 *GHz* o más rápido, 4 *GB* de *RAM.*, con *Windows* 8, *Visual Studio* 2012, *Microsoft Office* 2013, *Microsoft .NET framework* 4.5y *Kinect SDK* 1.8, además del *KINECT* de *Microsoft* y su adaptador de corriente.

- ¿A qué distancia será posible manipular el programa?

  Distancia recomendada 2.5 Metros (Varía conforme la estatura)

- ¿Cómo se va a lograr que únicamente reconozca al presentador y no interfiera con la audiencia?

No hubo problema dado que inicialmente escanea al presentador y detecta su esqueleto, si después interfiere otra persona, hay una interrupción momentánea pero el sensor vuelve a escanear rápidamente al presentador.

- ¿Qué limitaciones tiene el utilizar el modo "presentación" de *PowerPoint*?

  No poder editar diapositivas.

5.2 Con respecto al objetivo de la investigación

En base a los resultados presentados se puede concluir que se logró hacer el enlace entre el *KINECT de Microsoft* y *PowerPoint*, para alcanzar nuestro objetivo que es el controlar a distancia el movimiento de las diapositivas e interactuar con los objetos multimedia sin la necesidad de medios externos, utilizando comandos de voz y gestos, que son fáciles y sencillos de utilizar.

5.3 Recomendaciones para futuras investigaciones

Lograr el uso de algún gesto para usar el subrayado de texto habilitando el mouse y hacer un gesto más dinámico para el uso del *Zoom*, así como también poder utilizar otras librerías diferentes a *KINECT SDK* para el reconocimiento de más gestos como el uso de figuras geométricas o reconocimiento de los dedos para que la aplicación sea más dinámica.

Poder agregar sincronización de la altura del usuario con el sensor *KINECT* para que no haya cuestiones de distancia al momento de utilizar la aplicación.

Se recomienda el implementar la aplicación con otras librerías diferentes a *KINECT SDK*, y utilizar otros tipos de gestos.

Se recomendaría el implementar esta aplicación en otro tipo de programas, tal como en un visualizador de imagen o en otros programas de *Microsoft Office*.

Referencias

[1] GENBETA. [Online]. http://www.genbeta.com/ofimatica/doce-alternativas-a-powerpoint-para-realizar-tus-presentaciones

[2] OpenNI. (2011) OpenNI TM. [Online]. http://www.openni.org//

[3] Microsoft Corporation. (2011) Microsoft Careers JobsBlog: Experience Microsoft. [Online]. http://microsoftjobsblog.com/Tags/Kinect%20for%20Windows

[4] Microsoft Corporation. (2012) Microsoft Careers. [Online]. http://www.microsoft-careers.com/go/Kinect-for-Windows/308299/?utm_campaign=k4wDirect

[5] Kinesis.io. KINESIS.IO. [Online]. http://kinesis.io/

[6] Muy Linux. (2010, Dec.) Muy Computer. [Online]. http://www.muylinux.com/2010/12/17/openni-el-driver-open-source-oficial-de-kinect/

[7] KinEnemote. (2012, septiembre) KinEmote Gesture Driven Control. [Online]. http://www.kinemote.net/

[8] EVOLUCE AG. (2011) Evoluce Website. [Online]. http://www.evoluce.com/_win-and-i/en/software/overview/index.php

[9] ISDC. KINESIS. [Online]. http://code.google.com/p/kinesis/

[10] Bryab Blanchard, Cory Walker, Julio Montero, Azlap Tripathy, Ricardo Gutierrez Maged Kamel. (2011) healthgeographics. [Online]. http://www.ij-healthgeographics.com/content/10/1/45

[11] PrimeSense. (2011) PrimeSense Natural Interaction. [Online]. http://www.primesense.com/en/making-history

[12] (2011, julio) ieeexplore. [Online]. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5975266&isnumber=5948287

[13] Julian Dibbell. (2011, Mayo) PrimeSense Natural Interaction. [Online]. http://www.primesense.com/en/component/content/article/5-news/25-gestural-interfaces-controlling-computers-with-our-bodies

[14] Mattia Avancini, "Using Kinect to emulate an Interactive," UNIVERSITÀ DEGLI STUDI DI TRENTO, Tesis 2011.

[15] Jonathan Hall, Phoenix Perry Sean Kean, *"Meet the Kinect, An Introduction toProgramming Natural User Interfaces"*.: Apress, 2011.

[16] James Ashley Jarrett Webb, "Beginning Kinect Programming with the Microsoft Kinect SDK," 2012.

[17] Microsoft. (2012) xbox. [Online]. http://www.xbox.com/es-ES/Kinect/GetStarted

[18] Nicolas Burrus, Florian Echtler, Daniel Herrera, Matt Parker Jeff Kramer, *Hacking the Kinect*, TECHNOLOGY IN ACTION, Ed., 2012.

[19] Yi Li. (2012) ieeexplore. [Online]. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6269439

[20] Dmitriy Vatolin, Yury Berdnikov Sergey Matyunin. (2011) ieeexplore. [Online]. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5877202&isnumber=5877142

[21] Daniel Yuste Padilla, "Sistema de reconocimiento de gestos para un visor de imagenes," Universitat Politècnica de Catalunya (UPC), 2012.

[22] Janusz Konrad, Prakash Ishwar Kam Lai. ieeexplore. [Online]. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6202484

[23] David de la Fuente Garrido. (2012) [Online]. http://upcommons.upc.edu/pfc/bitstream/2099.1/15334/1/memoria.pdf

[24] Adrian Sanchez Cuervo. [Online]. http://eprints.ucm.es/13737/1/Adri%C3%A1n_S%C3%A1nchez_Cuervo_-_TFM.pdf

[25] David Catuhe, *programming with the Kinect for Windows Software Development Kit*.: Microsoft Press, 2012.

[26] Microsoft. (2012) KINECT for Windows. [Online]. http://www.microsoft.com/en-us/kinectforwindows/Develop/New.aspx

Apéndices

Anexo I: Cuestionario 1.

Instrucción> Conteste las siguientes preguntas tachando la respuesta.

1. **¿Utilizas el *PowerPoint*?**

Si No

2. **¿Crees que es aburrido?**

Si No

3. **¿Utilizas dispositivos inalámbricos para tus presentaciones?**

Si No

4. **¿Te gustaría manejar la presentación si necesidad de ningún dispositivo?**

Si No

5. **¿Crees que el *KINECT* se pueda utilizar para manejar *PowerPoint*?**

Si No

Anexo II: Cuestionario 2.

Instrucción> Conteste las siguientes preguntas tachando la respuesta.

1. **¿Qué te pareció el sistema, tu primera impresión?**

Muy interesante Interesante No me interesa

2. **¿Fue cómodo el sistema?**

Si Un poco No

3. **¿Fue cansado utilizarlo?**

Si Un poco No

4. **¿El sistema hacia lo que debía?**

Si Un poco No

5. **¿El sistema fue fácil de usar?**

Si Un poco No

6. **¿Lo utilizarías?**

Si Tal vez No

7. **¿Recomendarías los sistemas?**

Si No se No

8. **¿Qué le cambiarias o mejorarías al sistema?**

9. **¿Qué fue lo que más te gusto del sistema?**

10. **¿Qué fue lo que menos te gusto del sistema?**

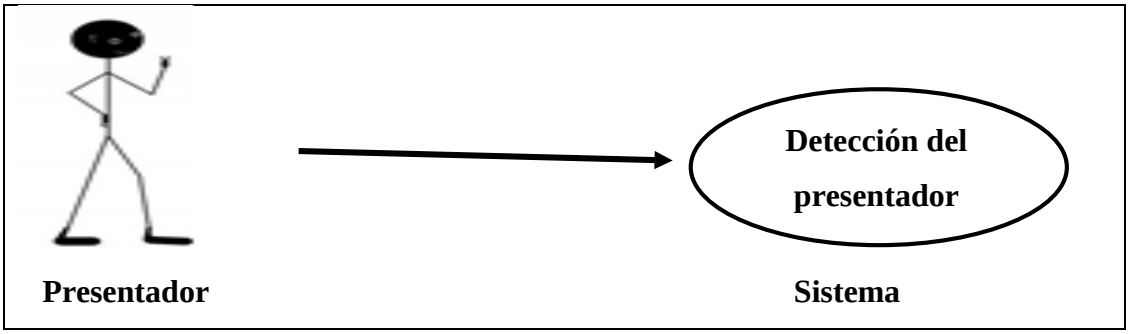
Anexo III: Casos de uso

Casos de uso: Detección del presentador

Actor: Presentador

Descripción: En este caso de uso un presentador quiere ser detectado por el sensor *KINECT* para manejar su presentación.

Diagrama de casos de uso:



Inicio: Este caso de uso se inicia cuando el presentador inicializa la aplicación.

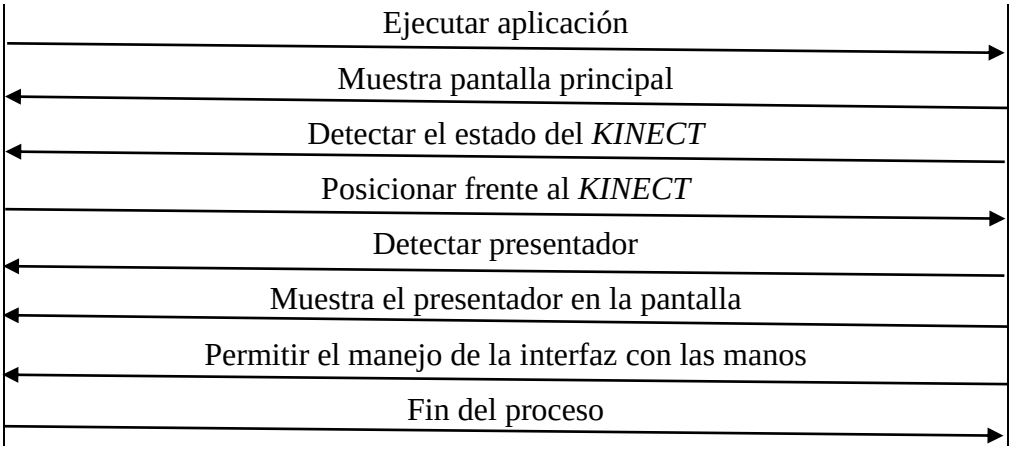
Precondiciones: El presentador debe de ejecutar la aplicación.

Actor: presentador	**Sistema: caso feliz**
1.Ejecutar la aplicación	2.Muestra pantalla principal
	3.Detecta el estado del *KINECT*
4.Posicionarse frente al *KINECT*	5.Detecta al presentador
	6.Muestra al presentador en pantalla
	7.Muestra la mano del presentador
	8.Permite manejar la aplicación con las manos
9.Finaliza proceso	
Caso alterno	
3.1 *KINECT* no conectado 3.2 *KINECT* ocupado 6.1 Presentador no detectado	

Post-condiciones: El presentador tendrá la opción de manipular con su cuerpo la pantalla principal, donde se puede dar clic en los botones abrir diapositivas o cerrar.

Diagrama de la caja negra:

Presentador **Sistema**

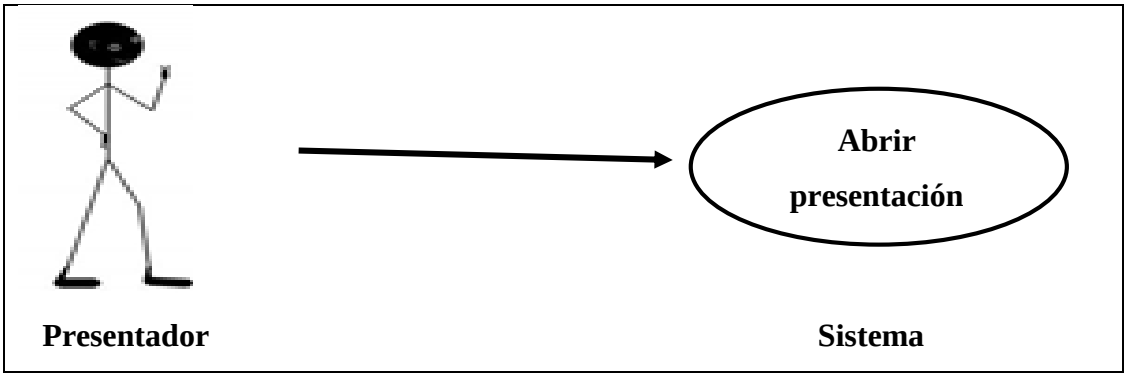


Casos de uso: Abrir presentación

Actor: Presentador

Descripción: En este caso de uso el presentador quiere abrir una presentación *PowerPoint.*

Diagrama de casos de uso:



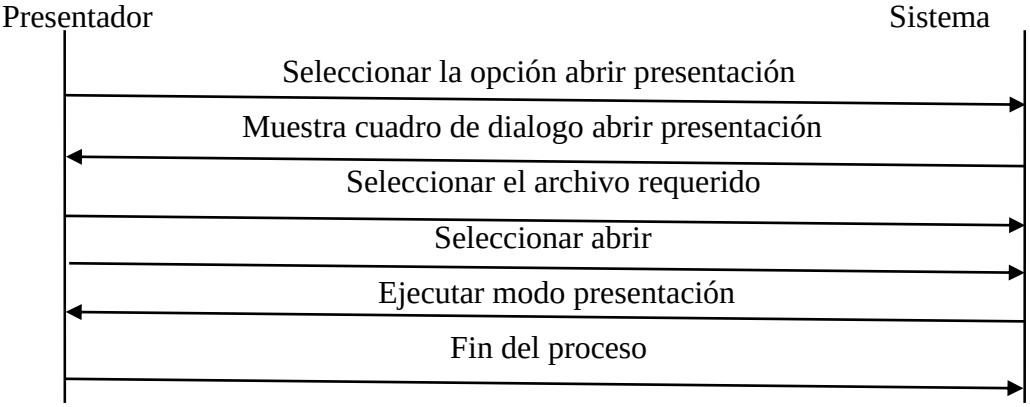
Inicio: Este caso de uso se inicia cuando el presentador selecciona la opción abrir presentación de la pantalla de inicio.

Precondiciones: El presentador debe de estar dentro de la aplicación y haber sido detectado por el *KINECT*.

Actor: presentador	**Sistema: caso feliz**
1.Selecionar la opción abrir presentación 3.Seleccionar el archivo requerido 5.Seleccionar abrir	2.Muestra cuadro de dialogo abrir presentación 4.Ejecutar modo presentación
Caso alterno	
3.1 No seleccionar archivo 5.1 Seleccionar cancelar	

Post-condiciones:

El presentador ya tendrá acceso al modo presentación del archivo seleccionado y podrá manejarlo con los gestos y comandos de voz deseados.

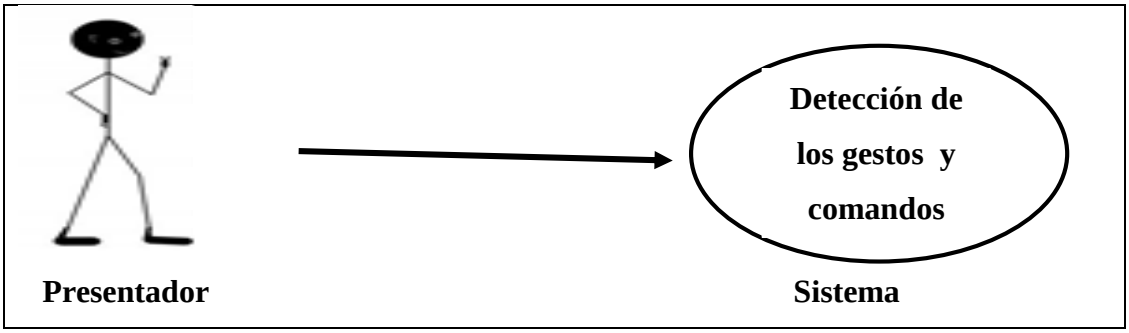


Casos de uso: Detección de los gestos y comandos de voz

Actor: Presentador

Descripción: En este caso de uso un presentador quiere dirigir su presentación por medio de gestos y comandos de voz

Diagrama de casos de uso:



Inicio: Este caso de uso se inicia cuando el presentador ya finalizó el caso de uso anterior, y tiene abierta la presentación.

Precondiciones: El presentador debe de tener la presentación abierta.

Actor: presentador	**Sistema: caso feliz**
1. Seleccionar gesto o comando de voz 3. Seleccionar gesto o comando de voz 4.Seleccionar comando cerrar	2. Ejecutar la acción que se seleccionó. 4. Ejecutar la acción que se seleccionó. 5.Cerrar la aplicación
Caso alterno	
4.1 Seleccionar gesto o comando de voz 5.1 Ejecutar la acción que se seleccionó.	

Post-condiciones:

El presentador podrá manejar la presentación con mediante los gestos y comandos de voz deseado, hasta finalizar su presentación con el comando de voz cerrar.