SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Tuxtla Gutiérrez

TRABAJO PROFESIONAL

COMO REQUISITO PARA OBTENER EL TITULO DE:

INGENIERO EN SISTEMAS COMPUTACIONALES

QUE PRESENTAN:

JOSÉ ANTONIO SÁNCHEZ ALFONSO
LEONARDO CABRERA GARCÍA

CON EL TEMA:

"SISTEMA DE SUPERFICIE AUDIOVISUAL CONTROLADO POR MULTIGESTOS KINESTÉSICOS"

MEDIANTE:

OPCION I
(TESIS)

TUXTLA GUTIERREZ, CHIAPAS **AGOSTO 2015**

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Tuxtla Gutiérrez

"2015, Año del Generalísimo José María Morelos Y Pavón"

DIRECCIÓN
SUBDIRECCIÓN ACADÉMICA
DIVISIÓN DE ESTUDIOS PROFESIONALES
Tuxtla Gutiérrez, Chiapas 02 de junio del 2015

OFICIO NUM. DEP-CT-614-2015

C. **JOSÉ ANTONIO SÁNCHEZ ALFONSO, LEONARDO CABRERA GARCÍA**
PASANTE DE LA CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES
EGRESADO DEL INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ.
P R E S E N T E.

Habiendo recibido la comunicación de su trabajo profesional por parte de los CC. M.C. AIDA GUILLERMNA COSSIO MARTÍNEZ, M.C. JOSÉ ALBERTO MORALES MANCILLA, DR. HÉCTOR GUERRA CRESPO, En el sentido que se encuentra satisfactorio el contenido del mismo como prueba escrita, **AUTORIZO** a Usted a que se proceda a la impresión del mencionado Trabajo denominado:

" SISTEMA DE SUPERFICIE AUDIOVISUAL CONTROLADO POR MULTIGESTOS KINESTÉSICOS."

Registrado mediante la opción:
I (TESIS PROFESIONAL)

A T E N T A M E N T E
"CIENCIA Y TECNOLOGÍA CON SENTIDO HUMANO"

Vo. Bo.

ING.JUAN JOSÉ ARREOLA ORDAZ
JEFE DEL DEPARTAMENTO DE LA DIVISION DE ESTUDIOS PROFESIONALES

M. en C. JOSÉ LUIS MÉNDEZ NAVARRO
DIRECTOR

C.c.p.- Departamento de Servicios Escolares
C.c.p.- Expediente
I'JLMN/I'JJAO/I'eeam

Secretaría de Educ. Pública
Instituto Tecnológico
de Tuxtla Gutiérrez,
Div. de Est. Profesionales

Agradecimientos

Agradezco a la M.C. Aida Guillermina Cossío Martínez que con gran profesionalismo asesoro este trabajo profesional.

Agradezco a mi Casa de Estudios, el Instituto Tecnológico de Tuxtla Gutiérrez por haberme educado y brindarme conocimientos y experiencias inolvidables.

Dedicatoria a...

Dios

Por haberme concedido la oportunidad
de vivir y de regalarme una segunda oportunidad, por guiarme en todo momento con su luz y darme la sabiduría necesaria para alcanzar una meta de muchas en mi vida.
De regalarme momentos extraordinarios y únicos y sobre todo por poner en mi camino a personas maravillosas que me apoyaron en todo momento y sobre todo de permitirme tener a mi lado a mis familia y seres queridos.

Mi madre Clary

Por el apoyo incondicional, por haberme traído al mundo, por haberme bendecido todo el tiempo y de haberme depositado toda tu confianza, porque gracias a ti madre mía éste logro fue posible .

Mis hermanos, Karen y Gerardo

Por sus consejos y llamadas de atención, por su cariño, compañía y apoyo, los quiero mucho, mis hermanitos.

Dedicatoria

A Dios:

Por darme la oportunidad de vivir gozando de buena salud, por fortalecer mi corazón e iluminar mi mente y por haber puesto en mi camino a aquellas personas que han sido mi soporte y compañía durante todo el periodo de estudio.

A mis padres:

Por el apoyo incondicional que siempre me han brindado, obsequiándome la mejor de las herencias y ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida.

Todo este trabajo ha sido posible gracias a ellos.

Mis hermanos:

Por sus consejos, darme ánimos para llegar hasta este momento, por estar conmigo y apoyarme siempre, los quiero mucho.

Índice general

Capítulo 1 – Introducción

1.1 Antecedentes……...2

1.1.1 Antecedentes de la empresa...…….4

1.1.2 Misión……..……..….4

1.1.3 Visión……...……...4

1.1.4 Valores……..…...4

1.2 Planteamiento del problema……...…….5

1.3 Variables……..……....6

1.4 Hipótesis……..……....6

1.5 Objetivos……7

1.5.1 Objetivo general…….……7

1.5.2 Objetivos específicos…….…….……7

1.6 Justificación……8

1.7 Alcances y limitaciones……..……....9

1.7.1 Alcances…….……....9

1.7.2 Limitaciones……...…...10

Capítulo 2 - Metodología

2.1 Metodología para el proceso de investigación……..…….…..12

2.2 Metodología de desarrollo de software…….……13

2.1.1 Incremento 1: Interfaz distribución de funcionalidades básicas……..…….14

2.1.2 Incremento 2: Interacción con el usuario y el programa principal con el sensor Kinect..……...14

2.1.3 Incremento 3: Mejoramiento en la interacción Interfaz Gráfica-Usuario e implementación de funciones para el control de diapositivas a través de gestos kinestésicos……...14

2.1.4 Incremento 4: Funcionalidad de creación de proyectos personales para guardar diapositivas en formato .JPG y video en formato .MP4…….15

2.1.5 Incremento 5: Mejoramiento de la interfaz donde se visualizaran las diapositivas y videos e integración con la funcionalidad de creación de proyectos..……...15

2.1.6 Incremento 6: Mejoramiento de las funciones para interpretar gestos del usuario……....15

2.1.7 Incremento 7: Integración de animaciones durante el cambio de diapositivas……15

2.1.8 Incremento 8: Creación de herramientas interactivas para edición en tiempo de presentación……..……..16

2.1.9 Incremento 9: Integración de función para detectar apertura y cierre de las manos……..……17

2.1.10 Incremento 10: Mejoramiento en la obtención de coordenadas aplicando técnicas de filtrado……...17

Capítulo 3 – Estado del arte

3.1 Estado del arte…….…..19

3.1.1 La mesa mágica (The Magic Table)……...……..19

3.1.2 Pizarrón virtual de bajo costo multi-tacto (Virtual Board A low cost Multitouch)……...……...……..22

3.1.3 Pizarra interactiva eficiente de bajo costo (Low-Cost Efficiente Interactive Whiteboard)…….22

3.1.4 Neko, un tabletop basada en Kinect para manipulación de objetos virtuales 2D y 3D……..…….24

3.1.5 Interfaz de lenguaje natural usando Kinect.……..……....27

3.1.6 Virtual Blackboard, Colour and Human Gestures Motion Tracking ……28

Capítulo 4 – Marco teórico y conceptual

4.1 Marco teórico……...…….……....30

4.1.1 Tracking…….,...30

4.1.2 Filtro de Kalman……...…….30

4.1.3 Regresión y correlación…….……...30

4.2 Marco conceptual…….....32

4.2.1 Sensor Kinect…….32

4.2.2 Cámara de profundidad…….32

4.2.3 Reconstrucción de objetos…….33

4.2.4 Rastreo del esqueleto…….34

4.2.5 Biblioteca de desarrollo SDK……35

4.2.6 .Net Framework…….36

4.2.7 Visual C#……...36

4.2.8 WPF(Windows Presentation Fundation)...……...37

4.2.9 XAML……..……...……..37

Capítulo 5 – Técnicas de filtrado para la predicción y corrección de puntos durante el tracking

5.1 Filtración……...39

5.1.1 Filtro de partículas……..……..39

5.1.1.1 Inicialización…….39

5.1.1.2 Actualización……40

5.1.1.3 Estimación……40

5.1.1.4 Predicción…..……40

5.1.2 Filtrado bayesiano recursivo……..40

5.1.3 El método Monte Carlo……..41

5.1.4 Filtro de Kalman……42

5.1.4.1 El proceso a ser estimado…….42

5.1.4.2 El Algoritmo…….43

5.2 Selección de la técnica de filtración…….45

Capítulo 6 – Desarrollo del software "Sistema de superficie audiovisual controlado por multigestos kinestésicos"

6.1 Introducción al desarrollo del software "Sistema de superficie audiovisual controlado por multigestos kinestésicos"……..48

6.2 Incremento 1: Interfaz distribución de funcionalidades básicas……..……....48

6.2.1 Análisis……..48

6.2.2 Diseño……49

6.2.3 Código…..……..51

6.2.4 Prueba……51

6.3 Incremento 2: Interacción con el usuario y el programa principal con el sensor Kinect……51

6.3.1 Análisis……..52

6.3.2 Diseño……52

6.3.3 Código…..……..53

6.3.4 Prueba……56

6.4 Incremento 3: Mejoramiento en la interacción interfaz gráfica-usuario e implementación de funciones para el control de diapositivas a través de gestos kinestésicos……....58

6.4.1 Análisis……..58

6.4.2 Diseño……58

6.4.3 Código…..……..60

6.4.4 Prueba……61

6.5 Incremento 4: Funcionalidad de creación de proyectos personales para guardar diapositivas en formato .JPG y video en formato .MP4……..62

6.5.1 Análisis……..62

6.5.2 Diseño……63

6.5.3 Código…..……..70

6.5.4 Prueba……78

6.6 Incremento 5: Mejoramiento de la interfaz donde se visualizaran las diapositivas y videos e integración con la funcionalidad de creación de proyectos……...78

6.6.1 Análisis……..78

6.6.2 Diseño……80

6.6.3 Código…..……..82

6.6.4 Prueba……86

6.7 Incremento 6: Mejoramiento de las funciones para interpretar gestos del usuario…….....87

6.7.1 Análisis……..87

6.7.2 Diseño……87

6.7.3 Código…..……..89

6.7.4 Prueba……91

6.8 Incremento 7: Integración de animaciones durante el cambio de diapositivas……..92

6.8.1 Análisis…….92

6.8.2 Diseño……..….92

6.8.3 Código…..……..93

6.8.4 Prueba……93

6.9 Incremento 8: Creación de herramientas interactivas para edición en tiempo de presentación…….94

6.9.1 Análisis…….94

6.9.2 Diseño……94

6.9.3 Código…..……....96

6.9.4 Prueba……..106

6.10 Incremento 9: Integración de función para detectar apertura y cierre de las manos……...107

6.10.1 Análisis……....107

6.10.2 Diseño……..108

6.10.3 Código…..……109

6.10.4 Prueba……..113

6.11 Incremento 10: Mejoramiento en la obtención de coordenadas aplicando técnicas de filtradode Kalman....…….115

6.11.1 Análisis……115

6.11.2 Diseño……..115

6.11.3 Código…..……118

6.11.4 Prueba……..…122

Capítulo 7 – Implementación y pruebas de la primera versión del software

7.1 Resultados en la utilización del sistema de superficie audiovisual controlado por multigestos Kinestésicos……. 127

7.1.1 Interacción entre el usuario final y el sistema……...127

7.1.2 Interacción con la implementación del filtrado……127

7.2. Pruebas de graficación sin filtro y utilizando el filtro de Kalman……...128

Capítulo 8 – Conclusión.

8.1 Conclusión……..132

Capítulo 9 – Trabajos futuros.

9.1 Trabajos futuros…..…….135

9.1.1 Mejoramiento en la implementación de técnicas de filtrado……..……135

9.1.2 Kinect 2……...……135

Referencias...……..136

Anexos……138

CAPÍTULO I

INTRODUCCIÓN

CAPÍTULO I

1.1 Antecedentes

El sector educativo es donde se realizan actividades de retroalimentación visual, es decir una actividad donde el profesor utiliza una herramienta de apoyo (con mayor frecuencia el video proyector) para plasmar o simplemente mostrar conceptos o ideas de un tema en específico con la finalidad de facilitar el aprendizaje en sus espectadores y es a través de este método como se trasmite la información de una manera eficaz, teniendo como consecuencia la generación de conocimiento a terceras personas, pero en ocasiones las clases suelen ser poco interactivas para el público espectador debido a las formas tradicionales en las que se desempeñan, es decir que el expositor no se apoya de alguna herramienta interactiva con la cual pueda hacer más dinámica una presentación.

Dentro del Instituto Tecnológico de Tuxtla Gutiérrez, en su mayoría maestros y alumnos utilizan las proyecciones para exponer un tema, pero limitados a ir mostrando únicamente la secuencia del mismo sin poder interactuar de manera dinámica con su material de exposición, además se necesita portar un dispositivo que coadyuve en el avance de la presentación o en su defecto de una segunda persona que apoye al presentador.

Esta tesis presenta la implementación del proyecto "Pizarrón virtual multifuncional controlado a distancia por sensores de movimiento para la presentación de materiales audiovisuales" que se diseñó en la residencia profesional, mas la aplicación de técnicas de filtrado para facilitar el uso del sistema, permitiendo el despliegue y la interacción natural de un entorno virtual controlado por el usuario, proporciona un enfoque diferente para dar una exposición o una clase en particular. El Sistema de superficie audio visual controlado por multigestos kinestésicos, ayuda al ponente a poder manipular su material por sí solo únicamente con determinados gestos de las manos, además dicho software tiene funciones de zoom y subrayado así como poder trazar líneas, cuadrados y triángulos, cabe recalcar que dichas funciones se realizan únicamente con movimientos de las manos que son leídos con el sensor Kinect e interpretados por el programa principal y posteriormente los datos obtenidos con el dispositivo son tratados aplicando la técnica del filtrado de Kalman el cual actúa de forma similar a un predictor lineal donde a partir de una muestra, se estima la

siguiente y posteriormente se calcula el error cometido en dicha predicción, logrando obtener el mínimo error en el proceso del tracking o seguimiento del usuario que se encuentra utilizando el software, haciendo que su interacción mejore en un porcentaje aceptable.

Este proyecto será probado en el salón “D10” del edificio D1 de Ingeniería en Sistemas Computacionales en clase de la M.C. Aida Guillermina Cossío Martínez, asesora de la tesis.

1.1.1 Antecedentes de la empresa

El Instituto Tecnológico de Tuxtla Gutiérrez, se encuentra ubicado en Carretera Panamericana Km. 1080, cuenta con carreras de licenciaturas y dos posgrados.

La carrera de ingeniería en sistemas computacionales inicia actividades en enero de 1991, es acreditado por CONAIC desde el 17 de Diciembre de 2012. Cuenta con el cuerpo académico "Tecnología Computacional para el Desarrollo Regional, ITTUXG-CA-4" (desde 2011). Programa de titulación integral consolidado y en proceso la maestría en tecnologías de la información.

El área de esta carrera se encuentra ubicada en su mayoría en el Edificio D-1 planta alta, ofrece sus servicios a los alumnos en tiempo corrido de 8:00 a 21:00 horas. La carrera cuenta con sala Cisco, sala de juntas, laboratorios y aulas.

1.1.2 Misión

Formar de manera integral profesionales de excelencia en el campo de la ciencia y la tecnología con actitud emprendedora, respeto al medio ambiente y apego a los valores éticos.

1.1.3 Visión

Ser una institución de excelencia en la educación superior tecnológica del sureste, comprometida con el desarrollo socioeconómico sustentable de la región.

1.1.4 Valores

- El ser humano
- El espíritu de servicio
- El liderazgo
- El trabajo en equipo
- La calidad
- El alto desempeño

1.2 Planteamiento del problema

El proyecto “Pizarrón Virtual Multifuncional Controlado a Distancia Por Sensores de Movimiento para la Presentación de Materiales Audiovisuales”, realizado en la residencia profesional contempla el análisis, diseño, codificación y pruebas del desarrollo del sistema, en la última etapa se observa que existe poca precisión del tracking (seguimiento) del usuario con el sensor Kinect, durante el proceso de recolección de coordenadas de los Joints o Nodos (Cabeza, Mano Izquierda, Mano Derecha, Hombro Izquierdo, Hombro Derecho, Hombro Central, Cadera Central), en sus respectivos ejes X, Y y Z, se tienen variaciones o ruido que hacen la interacción poco fluida ocasionando errores de precisión al momento de realizar funciones dentro del software.

Teniendo el problema de las variaciones en las coordenadas hace que el sistema no cumpla con una buena interacción, ocasionando que al momento de estar realizando la presentación se tenga las siguientes deficiencias:

* Precisión para controlar las funciones de escritura a mano alzada, marca texto, subrayado y selección de cuadro de herramientas.

* Interpretación correcta de gestos para avanzar o retroceder diapositivas.

Se pretende atacar el problema implementando técnicas de filtrado y corrección que utilizan la predicción y correlación durante la recolección de coordenadas en tiempo de tracking, también se analizará una nueva función que optimiza la selección del cuadro de herramientas utilizando la apertura y cierre de las manos.

1.3 Variables

Las variables que se seleccionaron para la investigación de este software son:

1.- Precisión de tracking.

2.- Recolección de coordenadas.

3.- Variación de los Nodos o Joints de la clase Skeleton.

4.- Ruido.

5.- Predicción.

6.- Correlación.

Debido a que la selección de las variables se hacen acorde a la hipótesis. Se opta en decidir que la variable dependiente es la **precisión del tracking** o seguimiento del usuario, pues es el fenómeno que dio lugar a esta tesis. Ahora las demás variables son las independientes puesto que son las características que explican el problema.

1.4 Hipótesis

El sistema de superficie audiovisual controlado por multigestos Kinestésicos, se eficientizará en un 99% la precisión del tracking (seguimiento) del usuario durante el proceso de recolección de coordenadas de los Joints o Nodos de la clase Skeleton, filtrando las variaciones y ruidos mediante el uso de predicción y correlación de puntos espaciales.

1.5 Objetivos

1.5.1 Objetivo general

Implementar técnicas de filtrado que utilizan la predicción y correlación de puntos espaciales durante el tracking para recolección de coordenadas, con ello mejorar el programa "Pizarrón virtual multifuncional controlado a distancia por sensores de movimientos para la presentación de materiales audiovisuales", con función de apertura y cierre de las manos para el control de selección utilizando como sensor de movimientos un Kinect y como visualización de imágenes un video proyector.

1.5.2. Objetivos específicos

- Hacer búsqueda de algoritmos de corrección y correlación de puntos.

- Hacer búsqueda de librerías, bibliotecas o DLL´s para tratamiento digital de imágenes y utilerías del sensor Kinect e implementarlas en el proyecto o solución del software.

- Construir las funciones o clases necesarias para la corrección y correlación de puntos

- Construir las funciones para detección de apertura y cierre de las manos del usuario.

- Implementar las funciones o clases de filtrado y detección de apertura y cierre de manos en el programa principal.

- Generar el Release o Liberación del Software para las pruebas con usuarios.

- Crear el archivo de Instalación.

- Mostrar en la pantalla del sistema los movimientos de ambas manos del usuario.

- Calibrar el dispositivo para que solo reconozca los movimientos de la persona que este exponiendo.

- Avanzar la presentación con un movimiento específico de la mano derecha, mismo que será interpretado por el sensor.

- Retroceder la presentación con un movimiento específico de a mano izquierda.

- Simular la escritura, borrado y el zoom únicamente con los movimientos de las manos.

- Agregar diapositivas en formato JPG.

- Hacer barridos con ventanas dinámicas que se diseñaran y programarán para el pizarrón.

- Guardar las presentaciones (pizarrones) para que el usuario pueda ver lo que explico anteriormente.

1.6 Justificación

La creación de un software que sea capaz de interpretar los movimientos corporales del usuario por medio del sensor Kinect, ofrece una alternativa adecuada cuando se trata de manipulación remota de una presentación, con la utilización del sensor se pueden interpretar determinados movimientos (gestos), los cuales pueden ser utilizados para el control del material de exposición, al mismo tiempo de poder interactuar dinámicamente con funciones como: zoom (acercar, alejar), pintar y trazos de figuras geométricas.

La aplicación de filtrado a través de técnicas de predicción y correlación eficientizará la presentación pues permitirá la captación del movimiento de los Nodos o Joints quitando los ruidos y las variaciones que se producían en el inicio.

Alcanzando lo antes mencionado habrá un mejor dominio del material y una conexión intuitiva con la presentación audiovisual, gracias a las herramientas o utilerías utilizadas, en consecuencia el expositor ya no dependerá de terceras personas que le ayuden a controlar la presentación, estará enfocado a personas profesionales, docentes, alumnos y para todos aquellos que quieran impartir un tema ante un grupo.

1.7 Alcances y limitaciones

1.7.1. Alcances

- La implementación de los algoritmos de corrección durante el Tracking sirven para el filtrado y eliminación de ruido, mejorando la interacción del usuario con su material de exposición.

- La apertura y cierre de las manos permiten al usuario aumentar la sensación de conexión con el material audiovisual.

- El pizarrón virtual puede controlar en su totalidad de manera remota, en un área entre 2 y 3 metros de frente al sensor Kinect.

- Controlar diapositivas que estén en formato JPG, para poder lograr esto, estas tienen que estar cargadas en un proyecto que se puede crear en el mismo programa.

- Adelantar y atrasar las diapositivas, haciendo gestos con los brazos, avanzar (gesto con la mano derecha) y atrasar (gesto con la mano izquierda).

- Zoom digital utilizando las dos manos posicionándolos frente del sensor y posicionando ambas manos en el centro del pecho desplazando hacia afuera logrando hacer la ampliación.

- Para el caso de reducir el Zoom, se tiene que hacer el mismo procedimiento en forma inversa como se explicó en el punto anterior.

- Para la función selección o clic utilizando la mano, la apertura y cierre debe ser frente al sensor.

1.7.2. Limitaciones

- La implementación de algoritmos que se basan en tratamiento digital de imágenes demandan altos recursos de la computadora por lo mismo los requerimientos de hardware aumentan.

- La detección de apertura y cierre de las manos funcionarán de manera correcta en ambientes controlados de iluminación.

- Los algoritmos para la mejora de tracking e interacción que se implementan no modifican la programación del sensor Kinect, solo mejoran la información que proporciona el sensor y en el programa principal donde se realiza el filtrado.

- Se necesita un sensor Kinect para detectar movimientos, un video proyector y una computadora de alto rendimiento para que pueda ejecutarse el programa sin problema alguno.

- El programa puede simular la escritura y borrado a mano, se limita a borrar todo lo que se ha pintado en el pizarrón, es decir que no se puede borrar por partes lo que sea escrito.

- El software solo pude funcionar en áreas donde no exista la exposición directa del sol, es decir que para el buen funcionamiento se necesita de un área con poca iluminación, se recomienda un aula o sala con cortinas, la interferencia directa con la luz solar respecto al sensor seguirán siendo un problema que no se puede solucionar al cien por ciento con los algoritmos que se programaron, debido a las especificaciones técnicas del fabricante del Kinect.

- No pueden utilizar dos personas el Pizarrón virtual, en caso de que se posicionen dos personas el programa proporcionara el mando a la que este más próxima al sensor Kinect.

CAPÍTULO II

METODOLOGÍA

CAPÍTULO II

2.1 Metodología para el proceso de investigación

La realización de este trabajo será bajo la metodología aplicada. Esta investigación recibe el nombre de "Investigación Práctica o Empírica", que se caracteriza porque busca la aplicación o utilización de los conocimientos adquiridos, a la vez que se adquieren otros, después implementar y sistematizar la practica basada en investigación.

La metodología que se siguió para el desarrollo del proyecto se llevó con forme a los siguientes pasos:

1.- Búsqueda, recopilación y análisis de información sujetas al desarrollo de pizarrones virtuales y controlador de imágenes a distancia utilizando el sensor Kinect.

2.- Búsqueda, recopilación y análisis de información del sensor Kinect a nivel Hardware y Bibliotecas de desarrollo (SDK´s) oficiales.

3.-Búsqueda de algoritmos, patrones, librerías y SDK´s para Procesamiento Digital de Imágenes.

4.- Búsqueda, Recopilación y Análisis de información que se centraran en la mejora o corrección de ruido durante el tracking que realiza el sensor Kinect hacia una persona.

5.- Diseño y construcción del Pizarrón virtual aplicando ingeniería de software con el modelo incremental.

6.- Realización de pruebas para la manipulación a distancia de las funciones del pizarrón virtual.

2.1 Metodología de desarrollo de software

Para el desarrollo de este Software se utilizó el modelo incremental el cual combina elementos del modelo lineal secuencial aplicados repetidamente con la filosofía interactiva de construcción de prototipos. Aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario.

Para la realización del software que dio lugar y fue objeto de investigación debido a la implementación de tracking, filtrado e interacción, entre otras características mencionadas en el capítulo anterior, se realizaron 10 incrementos funcionales. Cada incremento fue reutilizado desde el núcleo o algoritmo de obtención de puntos hasta el algoritmo de corrección.

La metodología se divide en: Incrementos y cada incremento consta de 4 partes.

1.- Análisis

2.- Diseño

3.- Código

4.- Prueba

Los incrementos deben cumplir con los requerimientos que se necesitan acorde al diagrama de casos de uso (Ver figura 2.1)

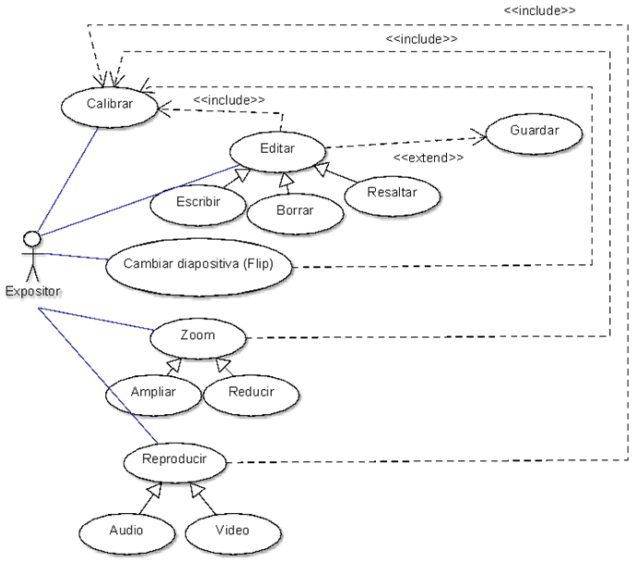

Figura 2.1: Diagrama de casos de usos del sistema.

2.1.1 Incremento 1: Interfaz distribución de funcionalidades básicas

El primer incremento se centra en un diseño rápido de la interfaz o UI con la que el usuario interactúa y la implementación del SDK para utilizar funciones básicas:

1.- Inicializar tracking del sensor Kinect.

2.- Visualizar en los 3 rectángulos la posición de un Joint (Mano derecha), el Skeleton completo, las cámaras RGB y profundidad.

2.1.2 Incremento 2: Interacción con el usuario y el programa principal con el sensor Kinect

Aquí se llega a la interacción con el usuario y objetos pintados en la pantalla principal, también se obtuvieron tres puntos específicos del Skeleton: Mano Derecha, Mano Izquierda y Cabeza en sus respectivos ejes X y Y posteriormente se visualizaron en la pantalla para analizar el movimiento de las manos y cabeza.

2.1.3 Incremento 3: Mejoramiento en la interacción interfaz gráfica-usuario e implementación de funciones para el control de diapositivas a través de gestos kinestésicos

Esta fase es donde se obtiene una interacción avanzada entre el usuario y el material de exposición, dichas funciones son:

1.- Visualización de diapositivas estáticas que se leen en un directorio predefinido la computadora.

2.- Interacción con la diapositiva utilizando movimientos de la mano derecha o izquierda.

2.1.4 Incremento 4: Funcionalidad de creación de proyectos personales para guardar diapositivas en formato .JPG y video en formato .MP4

La creación de proyectos tiene funcionalidad para que el usuario pueda agregar diapositivas máximo 99 en total y videos con límite de 3, dándole así la posibilidad de modificar su proyecto y poder exponer con la interacción del sensor Kinect, el proyecto se guarda con una extensión que solo el programa puede interpretar y todos los archivos agregados se guardan en una ubicación del disco duro ordenados por carpetas.

Este incremento aún está separado con la interacción del usuario con el proyecto.

2.1.5 Incremento 5: Mejoramiento de la interfaz donde se visualizan las diapositivas y videos e integración con la funcionalidad de creación de proyectos

El mejoramiento de la interfaz de presentación es un requerimiento importante debido a que en ella se visualizan las diapositivas en tamaño completo acorde a la resolución de la pantalla de la computadora, así mismo se integra la funcionalidad de creación de proyectos para que las diapositivas puedan ser leídas por la interfaz interpretando el archivo donde se configuro previamente.

2.1.6 Incremento 6: Mejoramiento de las funciones para interpretar gestos del usuario

En este incremento se hizo el mejoramiento de las funciones para interpretar los gestos del usuario, se modifican las funciones básicas y se agregan más Joints para tener una mejor interacción de coordenadas,

2.1.7 Incremento 7: Integración de animaciones durante el cambio de diapositivas

Las animaciones son un requerimiento para darle una mejor presentación a las diapositivas al momento de exponer, se implementó una animación para el cambio de diapositiva, requirió una reestructura de la interfaz a nivel código XAML.

2.1.8 Incremento 8: Creación de herramientas interactivas para edición en tiempo de presentación

La programación de las herramientas son fundamentales debido a la necesidad de poder utilizar un apuntador, zoom digital de la diapositiva, marca texto, escribir a mano alzada, insertar figuras geométricas o reproducir videos, las herramientas específicas son:

1.- Lápiz: Programado para la escritura a mano libre o mano alzada.

2.- Marca texto: Útil al momento de resaltar algún texto o párrafo.

3.- Video: Reproduce una secuencia de videos que se agregaron al momento de crear el proyecto.

4.- Trazar Cuadro: Función para encerrar un objeto dentro de la diapositiva.

5.- Limpiar o Borrador: Limpia toda la pantalla y deja la diapositiva con su contenido original.

6.- Circulo: Función para crear un círculo a partir de puntos de referencia inicial y final.

7.- Trazar línea: Tiene la funcionalidad de trazar una línea recta definiendo el punto inicial y punto final.

8.- Encerrar: Se asemeja con la funcionalidad de marca texto, ideal para encerrar párrafos completos, se define con un punto inicial y punto final.

9.- Triángulo: Función para dibujar un triángulo de diferente Angulo definiendo tres puntos de referencia.

10.- Herramienta de Zoom digital: esta función tiene la finalidad de ampliar la diapositiva que se esté visualizando, el usuario solo necesita hacer un gesto para poder ejecutar la función.

Todas las herramientas descritas se utilizan con gestos del usuario y se controlan a distancia.

2.1.9 Incremento 9: Integración de función para detectar apertura y cierre de las manos

La función de detección apertura o cierre de las manos proporciona mejor precisión al momento de realizar acciones básicas como selección o escritura a mano libre, en este incremento se implementaran bibliotecas de interacción con las que cuenta el SDK del Kinect.

2.1.10 Incremento 10: Mejoramiento en la obtención de coordenadas aplicando técnicas de filtrado

Este incremento es el más complejo y motivó la investigación dados los problemas encontrados con el sensor durante el tracking bajo ambientes no controlados, para la mejora se han implementado técnicas de filtrado donde se aplica predicción y correlación en los Joints: Mano derecha y Mano izquierda, se implementa la biblioteca de Emgu CV especial para procesamiento de imágenes.

En total se hicieron diez incrementos y cada uno enfocado a funciones distintas, con forme se pedían o encontrabanse implementaban a cada uno, al final se unificó todo el proyecto: obteniendo el prototipo de software "Sistema de Superficie Audiovisual Controlado por Multigestos Kinestécicos".

CAPÍTULO III

ESTADO DEL ARTE

CAPÍTULO III

3.1 Estado del arte

Para la realización de la presente tesis se hicieron búsquedas exhaustivas en diversas publicaciones, artículos y tesis de posgrados nacionales e internacionales relacionados al desarrollo o construcción de pizarrones virtuales.

Diversas tesis o publicaciones se centran en el análisis de los datos recopilados por el sensor Kinect e incorporan programación de alto nivel con la ayuda del SDK, o de alguna otra librería disponible para la programación del sensor. Algunos trabajos que aquí se mencionan solo lograron cubrir una parte del proyecto que es la abstracción de las coordenadas pero no desarrollaron en completa funcionalidad las interfaces, enseguida se repasan los últimos avances tecnológicos relacionados y se enfatiza en aquellos proyectos similares a éste.

3.1.1 La mesa mágica (the magic table)

La mesa mágica es un pizarrón avanzado el cual sirve como apoyo en reuniones creativas. La mesa mágica usa la visión por computadora para la digitalización y especialmente la organización de los textos y dibujos en la superficie. La información digital se organiza a través de la manipulación de símbolos (pequeños discos de plástico).

La interpretación consiste en gestos rápidos y fáciles de aprender que soportan múltiples usuarios simultáneos reconociendo las dos manos de cada uno. La aplicación tiene dos principales componentes: el modelo de color en función del Token Tracker y el escáner basado en técnicas de mosaicos [1].

3.1.1.1 Sistema de la mesa mágica

Los principales componentes de la mesa mágica es una calibración de la pizarra con sus lápices y goma de borrar, un proyector de vídeo que estará proyectando a la pizarra, y una cámara digital conectada a la computadora.

El proyector es un video DMD 1024x768 proyector. La cámara es un equipo controlado por Pan / Tilt / Zoom PAL (768x576 a 25 Fps.) cámara de vídeo. El equipo de la computadora es un estándar de bi-procesador equipado con gráficos (para la salida del proyector) y una

tarjeta de vídeo (Por entrada de cámara). El sistema fue experimentado con orientación de la pizarra en forma vertical y posteriormente en horizontal esta última orientación favorece porque permite más personas que interactúan al mismo tiempo.

La mesa permite que al menos cuatro personas (una en cada lado de la mesa) puedan trabajar al mismo tiempo sin obstruir mutuamente actividad. Así, el nombre del sistema tiene la CAPS de "pizarra mágica", la "mesa de magia". Una foto de la Mesa Mágica (Ver figura 3.1).

Figura 3.1: La mesa mágica en uso, dos usuarios Interactuando con ambas manos.

La configuración de la mesa mágica ha sido recientemente mejorada con la adición de una segunda cámara de video (Ver figura 3.2), a fin de permitir el funcionamiento en paralelo de exploración y servicios de seguimiento. Sólo una cámara tiene que ser dirigible: una dedicado a la exploración.

Figura 3.2: Posición de las cámaras y el proyector para la utilización horizontal.

La cámara de seguimiento tiene un fijo campo de visión establecido para abarcar toda la imagen proyectada sobre la mesa. Se ha estado trabajando en el seguimiento ya sea con dedos o fichas de plástico, aunque hasta hace muy poco el seguimiento de los dedos no era lo suficientemente robusta como para su uso en el sistema.

3.1.1.2 Interacción de la mesa mágica

Los usuarios pueden escribir, dibujar y borrar en la tabla mágica como en una pizarra normal. Cuando se necesita acceder a los servicios digitales más avanzados, se escanean los parches de los usuarios. Un parche es un rectángulo que contiene una copia digital de los códigos impresos.

Una vez creado, el parche y su contenido se pueden mover alrededor de la superficie. Los videos del sistema en uso están disponibles para su descarga desde la web (http://iihm.imag.fr/demos/magicboard/).

Los usuarios interactúan con la mesa mágica mediante la manipulación de las fichas. En el diseño de la interacción, se intenta que coincida con la sencillez y la rapidez de trabajo de forma regular como una pizarra.

Se define un pequeño conjunto de gestos que permiten tres operaciones fundamentales de los parches, (Ver figura 3.3).

a) Creación, b) Eliminación y c) Transformación.

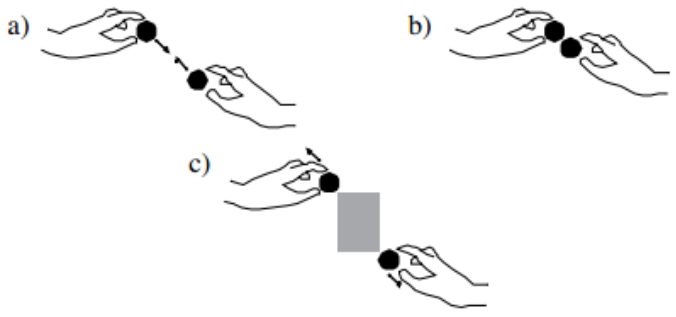

Figura 3.3: Conjunto de gestos definidos para la pizarra mágica.

3.1.2 Pizarrón virtual de bajo costo multi-tacto (virtual board: a low cost multi touch human computer interaction system)

Este pizarrón es un prototipo que utiliza un sistema de control remoto a través de computar la técnica de infrarrojos para detectar múltiples puntos infrarrojos en un lugar, mostradas por los marcadores activos.

El sistema utiliza como marcador de activos, dos guantes con infrarrojos en la punta del dedo índice y un botón en el extremo del dedo medio para que con el pulgar se puedan cambiar de infrarrojos.

Los marcadores han sido diseñados por el movimiento natural de la mano del usuario, dando a los usuarios más fácil manipulación. El sistema puede manipular imágenes para acercar, alejar el zoom, la rotación y la traducción y utiliza dos marcadores de activos, la cámara web detecta estos marcadores después que el software tiene la posición de los marcadores y con cálculos matemáticos muestra la manipulación de imágenes.

Posibles dispositivos utilizados para el control remoto. Kinect y Wiimote, que ofrecen la tecnología por control remoto para la computadora se puede encontrar en el mercado, cada dispositivo utiliza diferentes técnicas para manipular el equipo con sus respectivas características y limitaciones. [2]

3.1.3 Pizarra interactiva eficiente de bajo costo (low-cost efficient interactive whiteboard)

Pizarra interactiva eficiente de bajo costo, utiliza la profundidad de fusión y de vídeo información proporcionada por una cámara de profundidad de bajo costo, es capaz de detectar y rastrear los movimientos del usuario. [3]

El dispositivo pizarrón interactivo (IWB) ha sido ampliamente utilizado en ambientes de enseñanza o en las salas de conferencias. IWB presenta varias ventajas con respecto a una pizarra convencional: permite guardar contenido de la pizarra, la integración de otros contenidos multimedia (videos, e imágenes) y ayuda a captar mayor la atención de la audiencia con la interactiva herramienta de dibujos.

IWB puede estar basada en diferentes tecnologías: superficies activas resistiva o capacitiva, tecnologías ópticas sobre la base de los sistemas de láser o dispositivos activos lápiz; ultrasónicos transmisores, aplica generalmente en las esquinas de pizarra.

A pesar de que estas tecnologías garantizan un seguimiento eficaz de las interacciones del usuario con el sistema de la pizarra, por lo general son caros y no puede ser fácilmente adaptada o movido en diferentes ambientes. Además, estos sistemas sólo pueden detectar la interacción del usuario con la pizarra apoyándose en contacto con la superficie, y no contempla la captura de gestos a mano alzada.

En este trabajo se presenta un eficiente sistema de bajo costo (IWB) basado en un sensor de profundidad comercial de bajo costo. La principal ventaja de este sistema es su alto nivel de portabilidad: puede ser integrados con los equipos comunes, como monitores o proyector y equipos fuera de la plataforma. Además, también permite integrar el reconocimiento de gestos, por lo tanto, amplía las posibilidades de fácil interacción con la pizarra.

El esquema del sistema IWB propuesto (Ver figura 3.4). La cámara profundidad adquiere la información de vídeo y la profundidad de la escena y lo envía a una unidad de procesamiento (PU) que combina la información de profundidad y de vídeo para extraer la silueta del usuario y para detectar y rastrear los movimientos de la mano. Entonces, ya extraídos los movimientos se utilizan para actualizar el contenido de pizarra.

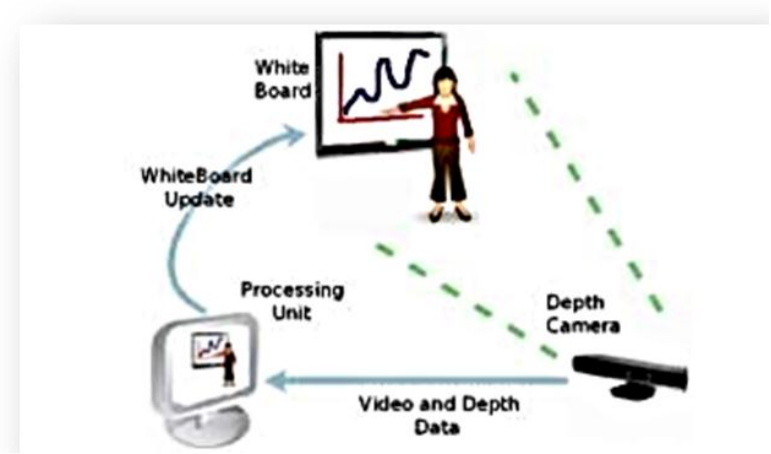

Figura 3.4: Esquema del Sistema Pizarra Interactiva.

Las imágenes se representan por un proyector conectado a la PU (que es una PC fuera de la plataforma, ya que no es hardware especializado necesario). El papel de la PU es fundamental para que funcione correctamente la gestión de vídeo y los datos de profundidad y para garantizar la fiabilidad el seguimiento y el reconocimiento de gestos.

El diagrama de bloques de tareas de procesamiento de la unidad (la UP) se muestra en la figura 3.5.

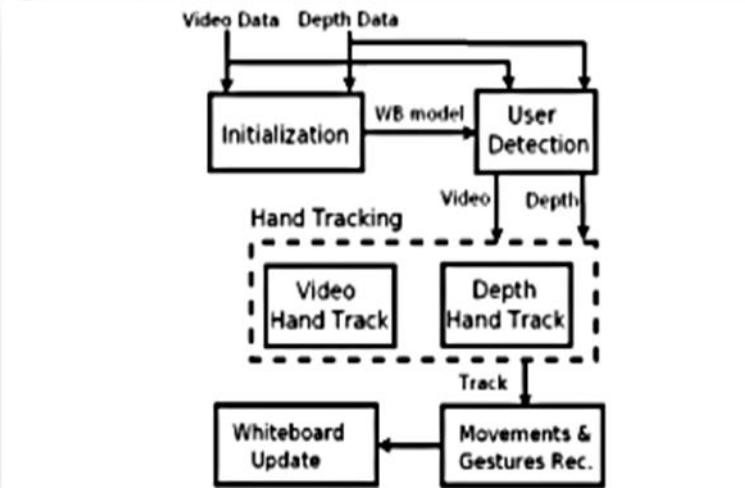

Figura 3.5: Diagrama de bloques para el funcionamiento de la pizarra.

3.1.4 NEKO: un tabletop basada en kinect para manipulación de objetos virtuales 2D y 3D

NEKO (del inglés Natural Environment for Kinect-based Object interaction) un prototipo de tabletop que permite el despliegue y manipulación natural de un entorno virtual acorde a la percepción del usuario. El primordial objetivo en la construcción de NEKO fue crear un espacio interactivo que permitiera manipular objetos virtuales 2D y 3D a través de las manos de una forma semejante a como se haría sobre una mesa convencional.

En particular se buscó evitar las barreras que imponen los dispositivos de entrada y salida convencionales (como el teclado y el ratón). Para cumplir dicha meta se diseñó un prototipo de tabletop, con componentes de hardware comercial de bajo costo, que facilitan al usuario interactuar sin la necesidad de dispositivos de control artificial.

Así mismo se desarrolló una arquitectura de software capaz de identificar patrones relevantes de las manos y rostro de los usuarios con un tiempo de respuesta corto. [4]

3.1.4.1 Prototipo de tabletop

NEKO está compuesto de un prototipo de tabletop y de componentes de software que permiten identificar la interacción del usuario. En general, con el propósito de facilitar la construcción y reproducción del prototipo, la tabletop está compuesta de una mesa metálica e instrumentada con dispositivos comerciales de bajo costo.

El prototipo de la mesa, a diferencia de una mesa convencional, está compuesto de dos tableros, uno posicionado a cierta distancia por encima del otro. Sobre el tablero inferior se encuentra un cristal claro de 90 x 60 centímetros con un grosor de 4 milímetros; el tablero superior posee un cristal obscuro, de 6 milímetros de grosor, con las mismas medidas que el cristal claro (ver figura 3.6).

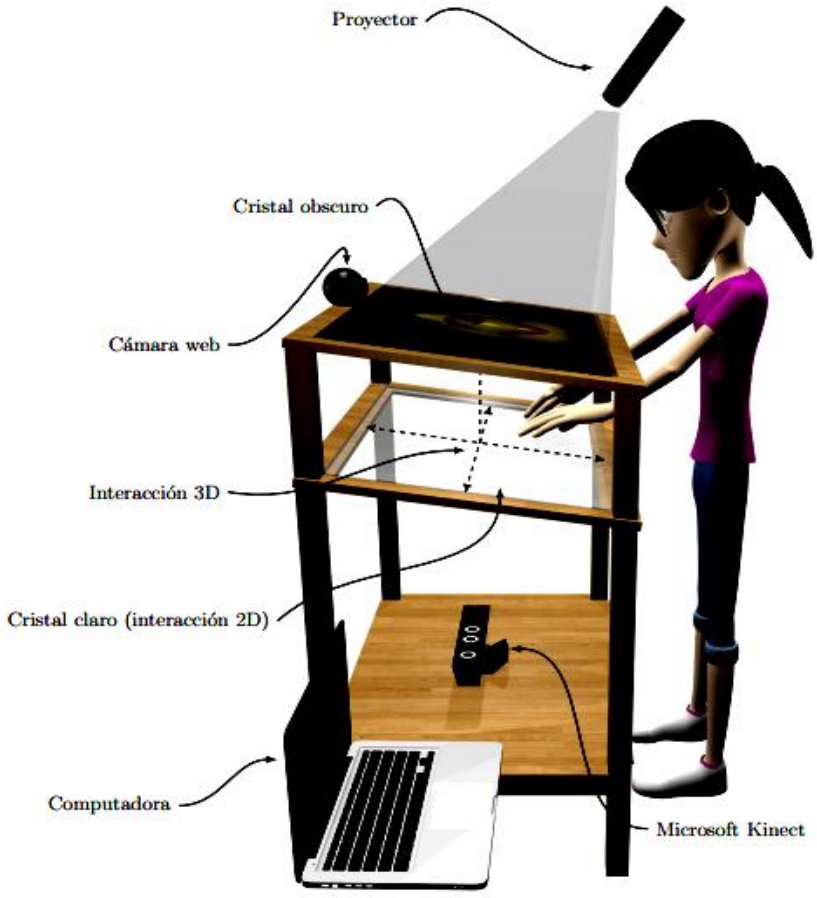

Figura 3.6: Arquitectura de hardware de NEKO.

La figura anterior muestra los elementos que integran la arquitectura de hardware del prototipo de tabletop. La mesa de fierro, compuesta de dos tableros con cristales, está instrumentada de una cámara web, el dispositivo Kinect, un proyector y una computadora.

El dispositivo Kinect, posicionado en la base de la mesa y orientado hacia los tableros, captura imágenes de la interacción realizada por las manos del usuario entre los cristales. Semejante a una interacción sobre un dispositivo multitáctil, la interacción 2D toma lugar sobre la superficie del cristal inferior en donde el usuario interactúa a través del tacto.

La interacción 3D toma lugar en el espacio entre los tableros. Para mantener la correspondencia entrada/salida, los objetos virtuales son proyectados sobre el cristal superior de la mesa, entre las manos y el rostro del usuario.
Así el usuario visualiza los objetos como si estuvieran en sus manos. De igual forma que con la cámara web, se utilizan algoritmos de visión por computadora para procesar la información proporcionada por el dispositivo Kinect.

El proyector es el dispositivo de salida que despliega los objetos virtuales sobre la tabletop. Un proyector es mucho más barato si se compara con pantallas estereoscópicas o monitores del tamaño de los tableros de la mesa. Además, si en lugar de utilizar un proyector de tamaño común se utiliza un pico proyector las posibilidades de movilidad se incrementan, lo que hace posible colocar el proyector en diferentes ubicaciones. Por ejemplo en la base de la mesa, proyectando por debajo de los tableros, o por encima de la mesa para desplegar los objetos sobre el marco superior.

3.1.5 Interfaz de lenguaje natural usando kinect

Es un software realizado para la manipulación de objetos en tercera dimensión, una interfaz natural basada en la interacción con la computadora a través de secuencias de movimientos, que funja como base para el desarrollo de aplicaciones de interacción natural en Qt para sistemas Linux. [5]

La interfaz funciona bajo un paradigma de comandos basados en secuencias de acciones (identificación de movimientos simples, que resultan intuitivos y fáciles de replicar al usuario), que actúan directamente sobre los elementos de la misma aplicación, o en el entorno de escritorio (emulando la entrada de ratón), lo que permite interactuar con otras aplicaciones de la computadora.

La Figura 3.7 muestra una fotografía del sistema dibujando el esqueleto 3D del autor levantando la mano izquierda, también se muestra la malla de acciones en 2D y 3D.

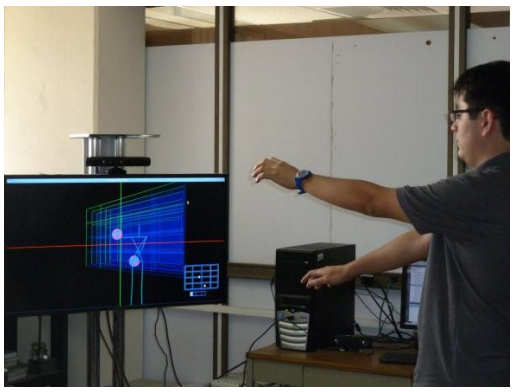

Figura 3.7: Esqueleto 3D y malla de acciones.

La malla de acciones 3D se forma por el conjunto de líneas horizontales y verticales frente al esqueleto. La malla en 2D es el rectángulo subdividido de la derecha. Los cuadrados negros pintados dentro de la malla 2D indican la posición de las manos dentro de la malla.

El rectángulo dividido con líneas verticales que se encuentra en la parte inferior derecha de la imagen (cada línea corresponde a un plano Zn, donde Z0 la primer línea de la derecha) muestra la profundidad de las manos (indicada por las líneas gruesas) dentro de la malla.

3.1.6 Virtual blackboard: colour and human gestures motion tracking

Consiste en el desarrollo de una aplicación informática que hace uso de los métodos habituales de tracking de objetos en imágenes de video y pertenece al campo de la visión por computador. [7]

El programa que se han llevado a cabo ha sido una pizarra virtual, donde el usuario puede pintar simplemente moviendo un círculo rojo delante de una cámara web. Para realizar el seguimiento del objeto de color se ha utilizado la transformada de Hough en la detección y la técnica del Filtro de Kalman para la predicción; además se han realizado algunas pruebas de procesamiento de imagen adicionales, como el cambio de espacios de color, bancarización, etc.

En él se elaboró un filtro por distancia que hace uso del dispositivo Kinect con la intención de que sirva de ayuda en el seguimiento del puntero de la pizarra virtual (ver figura 3.8).

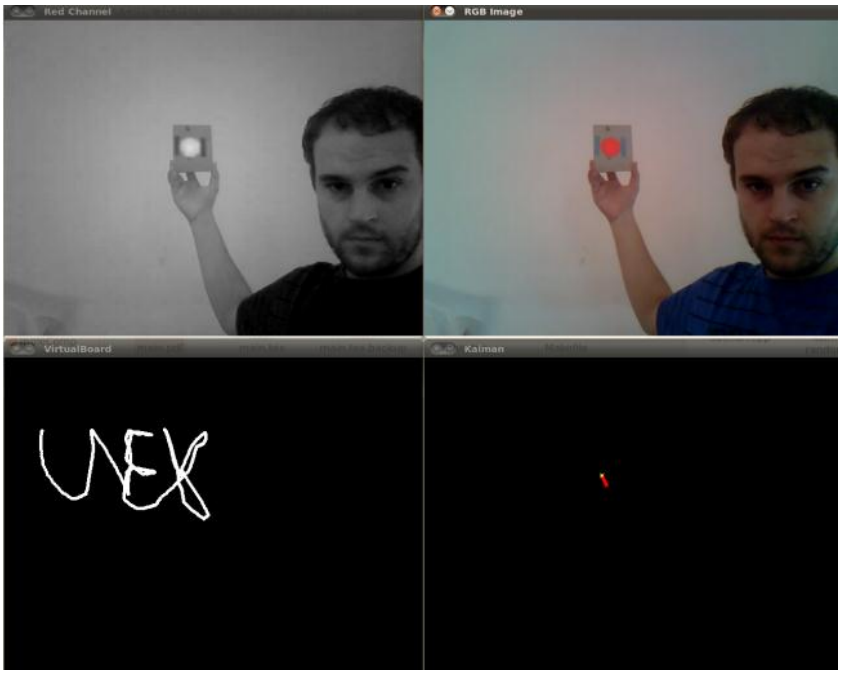

Figura 3.8: Escribiendo UNEX en la pizarra virtual.

CAPÍTULO IV

MARCO TEÓRICO Y CONCEPTUAL

CAPÍTULO IV

4.1 Marco teórico

4.1.1 Tracking

El tracking, se refiere de forma general al seguimiento automático de objetos, ya sea en imágenes bidimensionales o tridimensionales. El seguimiento a su vez se suele dividir en dos tareas importantes, la detección y la predicción.

Ambas tareas están estrechamente relacionadas ya que la detección inicia y corrige a la predicción en la mayoría de los casos, mientras que la predicción permite afinar y acelerar la detección. Debido a que algunos métodos de predicción son altamente utilizados para aplicaciones de tracking, puede resultar confuso cuando son mencionados si se hace referencia a la predicción o a su típico uso en seguimiento de objetos en imagen.

4.1.2 Filtro de Kalman

Uno de los métodos más utilizados para el tracking de objetos en imágenes en movimiento es el filtro de Kalman, que surge de la adaptación del algoritmo de Rudolf E. Kalman. Este método actúa de forma similar a un predictor lineal dónde a partir de una muestra, estimamos la siguiente y posteriormente calculamos el error cometido en dicha predicción, de manera que para el cálculo de la siguiente muestra y teniendo en cuenta dicho error, podamos mejorar la predicción minimizando el error.

Este método es particularmente efectivo cuando se dan unas condiciones ideales como la ausencia de cambios bruscos en la muestra o cierta pasividad del entorno. [8]

4.1.3 Regresión y correlación

La regresión y la correlación son dos técnicas estrechamente relacionadas y comprenden una forma de estimación. En forma más especifica el análisis de correlación y regresión comprende el estudio de los datos muéstrales para saber qué es y cómo se relacionan entre sí dos o más variables en una población. El análisis de correlación produce un número que resume el grado de la correlación entre dos variables; y el análisis de regresión da lugar a una ecuación matemática que explica y predice dicha relación.

El análisis de correlación generalmente resulta útil para un trabajo de exploración cuando un investigador trata de determinar que variables son potenciales importantes, el interés radica básicamente en el grado de la relación y la regresión da lugar a una ecuación que describe, explica y predice dicha relación en términos matemáticos. [9]

4.2 Marco conceptual

4.2.1 Sensor kinect

Kinect es una cámara 3D de Microsoft con tecnología PrimeSense que salió al mercado en noviembre de 2010. Apareció como periférico de su consola Xbox360, pero debido a su relativo bajo precio con respecto a otras cámaras 3D y gracias a la ingeniería inversa con la que se pudo acceder a los datos generados por la cámara, se popularizó en los círculos de la investigación en visión artificial. [8]

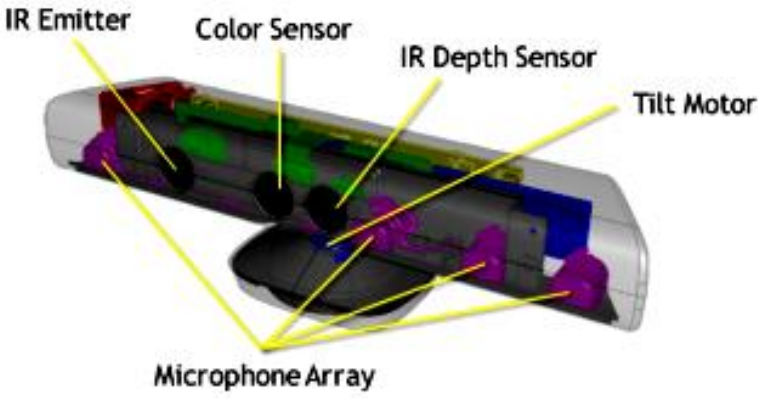

Figura 4.1: Sensor Kinect y partes de las que se compone.

4.2.2 Cámara de profundidad

La cámara de profundidad se compone por la cámara infrarroja y el proyector infrarrojo de luz estructurada, como se muestra en la figura 4.2.

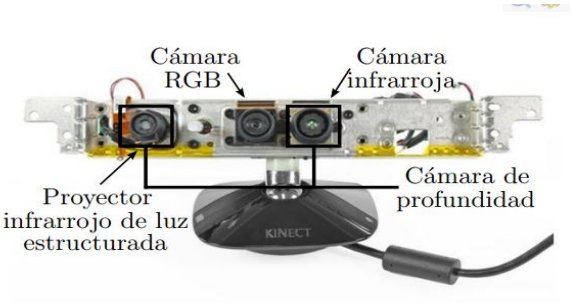

Figura 4.2: Cámaras de Kinect.

La cámara de profundidad utiliza una tecnología de codificación por luz llamada Light Coding desarrollada por PrimeSense que actúa como un escáner 3D para realizar una reconstrucción tridimensional de la escena. Light Coding es similar a los escáneres de luz estructurada, pero en lugar de desplazar una línea de luz, se proyecta un patrón de puntos infrarrojos en la escena.

El patrón de puntos se localiza en un difusor (rejilla) frente al proyector infrarrojo. Al emitir la luz infrarroja, el patrón se dispersa por la escena, proyectándose sobre las personas y objetos presentes en ésta (como se puede observar en la figura 4.3), mientras que la imagen de dicha proyección se obtiene mediante la cámara infrarroja.

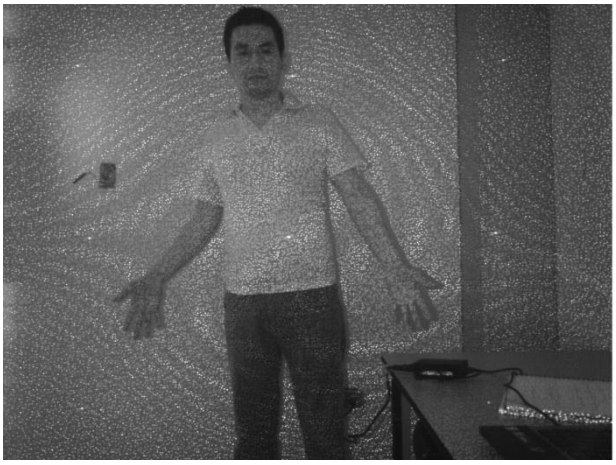

Figura 4.3: Fotografía de la proyección del patrón de puntos infrarrojos.

Una vez que la información es separada del resto, se convierte cada identificación del cuerpo en un esqueleto con articulaciones móviles.

4.2.3 Reconstrucción de objetos

Kinect reconstruye la escena visualizada en 3D para obtener las imágenes de profundidad. La reconstrucción de objetos es un área especial de visión por computadora. Se encarga del desarrollo e investigación de técnicas para reconstruir objetos tridimensionales y el cálculo de la distancia entre el sensor y los objetos de la escena. Es utilizada para identificar y conocer la distancia hacia objetos en la escena y detección de obstáculos (robótica y sistemas de control vehicular), inspección de superficies en sistemas de control de calidad y navegación de vehículos autónomos.

También es utilizada en la estimación de objetos tridimensionales (sistemas de ensamblado automático). La reconstrucción de objetos involucra otras áreas como el procesamiento de imágenes (filtrado, restauración y mejora de imágenes, entre otras) y reconocimiento de patrones (segmentación, detección de bordes y extracción de características, entre otras).

Una de las primeras técnicas usadas para estimar la forma de un objeto o mapa 3D se basa en la triangulación, que utiliza dos cámaras viendo el mismo objeto. El cambio de posición del objeto en ambas imágenes se relaciona con la distancia a dicho objeto, siendo similar al sistema de visión humano. La desventaja que presenta es la baja resolución de la reconstrucción 3D, que depende de la definición y orientación relativa de las cámaras (ángulo y distancia), además el procesamiento de alto nivel que requiere no hace factible su uso en tiempo real.

La tecnología de PrimeSense evita los problemas de ambigüedad de colores e iluminación al convertir la segmentación de la persona en un problema de clasificación de imágenes de profundidad obtenidas en tiempo real mediante un dispositivo de bajo costo.

4.2.4 Rastreo del esqueleto

El rastreo del esqueleto consta de procesar las imágenes de profundidad obtenidas con Kinect para detectar formas humanas e identificar las partes del cuerpo del usuario presente en la imagen. Cada parte del cuerpo es abstraída como una coordenada 3D o articulación.

Un conjunto de articulaciones forman un esqueleto virtual para cada imagen de profundidad de Kinect, es decir, se obtienen 30 esqueletos por segundo. Las articulaciones generadas varían de acuerdo a la biblioteca de Kinect que se utilice. En la biblioteca SDK, cada esqueleto (figura 4.4) está formado por 20 articulaciones ai ={xi,yi,zi} con zi >0 cuyas coordenadas se encuentran expresadas en milímetros con respecto a la posición de Kinect en la escena. [8]

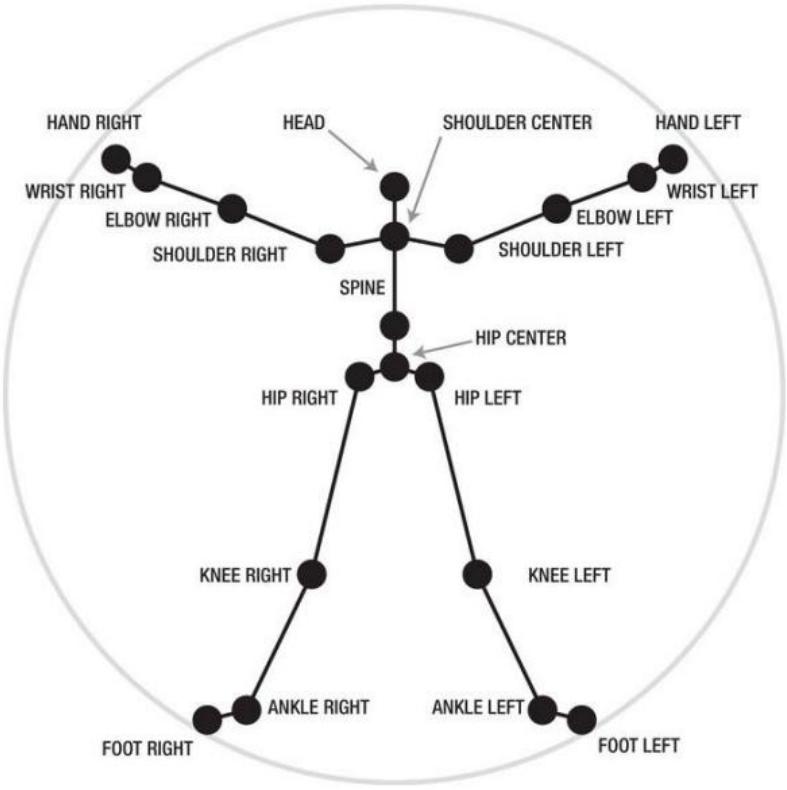

Figura 4.4: Articulaciones obtenidas con el SDK de Kinect.

4.2.5 Biblioteca de desarrollo SDK

El SDK de Windows para Kinect es una herramienta de programación para desarrolladores de aplicaciones. Está abierta para las comunidades académicas y entusiastas de las capacidades ofrecidas por el dispositivo de Microsoft Kinect, conectado a equipos que ejecutan el sistema operativo Windows7.

El SDK incluye controladores, API rica para los flujos de los sensores sin procesar y el seguimiento del movimiento humano, los documentos de la instalación, y recursos materiales de Kinect proporciona capacidades para los desarrolladores que crean aplicaciones con C++, C# o Visual Basic mediante Microsoft Visual Studio 2010. [10]

El SDK de Windows contiene un conjunto de herramientas, ejemplos de código, documentación, compiladores, encabezados y las bibliotecas que los desarrolladores

pueden utilizar para crear aplicaciones que se ejecutan en Microsoft Windows. Se puede utilizar el SDK de Windows para escribir aplicaciones que utilizan los nativos (Win32/COM) o administrador (.NET Framework).

4.2.6 .NET Framework

.NET Framework [11] es una tecnología que admite la compilación y ejecución de la siguiente generación de aplicaciones y servicios Web XML. El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que minimiza los conflictos en el despliegue y versionado de software.
- Ofrecer un entorno de ejecución de código que promueva la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan scripts o intérpretes de comandos.
- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.
- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

4.2.7 Visual C#

C# es un lenguaje de programación que se ha diseñado para compilar diversas aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. Las numerosas innovaciones de C# permiten desarrollar aplicaciones rápidamente y mantener la expresividad y elegancia de los lenguajes de estilo de C.

Visual C# es una implementación del lenguaje de C# de Microsoft. Visual Studio ofrece compatibilidad con Visual C# con un completo editor de código, un compilador, plantillas

de proyecto, diseñadores, asistentes para código, un depurador eficaz y de fácil uso y otras herramientas. La biblioteca de clases de .NET Framework ofrece acceso a numerosos servicios de sistema operativo y a otras clases útiles y adecuadamente diseñadas que aceleran el ciclo de desarrollo de manera significativa. [12]

4.2.8 WPF (Windows Presentation Fundation)

Windows Presentation Foundation (WPF) es un sistema de presentación de próxima generación para crear aplicaciones cliente de Windows que proporcionen una experiencia visual impactante para el usuario. Con WPF, puede crear una amplia gama de aplicaciones independientes y hospedadas en explorador.

El núcleo de WPF es un motor de representación basado en vectores e independiente de la resolución que se crea para sacar partido del hardware de gráficos moderno. WPF extiende el núcleo con un conjunto completo de características de desarrollo de aplicaciones que incluye Lenguaje XAML, controles, enlace de datos, diseño, gráficos 2D y 3D, animación, estilos, plantillas, documentos, multimedia, texto y tipografía.

WPF se incluye en Microsoft .NET Framework, de modo que es posible compilar aplicaciones que incorporen otros elementos de la biblioteca de clases de .NET Framework. [13]

4.2.9 XAML

XAML permite definir la interfaz de usuario; los programadores que trabajan con aplicaciones web lo encontrarán muy fácil de utilizar. La interfaz definida con XAML es, entonces, controlada con el Framework de WPF mediante cualquiera de los lenguajes .NET; en nuestro caso, C#.

XAML está basado en XML y es un lenguaje declarativo. Con él podemos definir y especificar características para clases con una sintaxis basada en XML. Este nos permite hacer cosas como declarar variables o definir propiedades. No es posible definir la lógica del programa con XAML, pero al poder establecer las características de las clases, lo usamos para definir la interfaz de usuario y los elementos gráficos de nuestra aplicación. La lógica se colocará en C#, y esto es lo que se conoce como code behind. [14]

CAPÍTULO V

TÉCNICAS DE FILTRADO PARA LA PREDICCIÓN Y CORRECCIÓN DE PUNTOS DURANTE EL TRACKING

CAPÍTULO V

5.1 Filtración

La disminución del ruido causado por los sensores del Kinect cuando se encuentra en un ambiente de luz no controlada se puede lograr implementando en el programa principal técnicas de Filtración para la corrección de datos obtenidos a partir de un estado que cambia a lo largo del tiempo.

Muchos problemas físicos y científicos requieren estimar el estado de un sistema que cambia a lo largo del tiempo usando una secuencia de medidas ruidosas realizadas sobre el sistema. Una posible solución al problema la representan las aproximaciones del espacio de estados (state-space models). Esta aproximación centra su atención en el vector de estado del sistema, el cual contiene toda la información relevante necesaria para describir el sistema bajo investigación [15].

5.1.1 Filtro de partículas

El filtro de partículas es un método empleado para estimar el estado de un sistema que cambia a lo largo del tiempo. El filtro de partículas se compone de un conjunto de muestras (partículas) y valores o pesos asociados a cada una de esas muestras. Las partículas representan muestras del espacio de estados (estados posibles) del proceso, y los pesos representan muestras de la f.d.p. a posteriori del estado, dadas las observaciones.

Posee cuatro etapas principales:

5.1.1.1 Inicialización

Para realizar el seguimiento (por ejemplo de un objeto sobre una secuencia de imágenes), el filtro de partículas “lanza” al azar un conjunto de puntos (sobre el plano de imágenes en este caso). En esta etapa de inicialización, el conjunto de partículas se puede crear con un estado aleatorio, o se puede emplear algún tipo de información a priori (tamaño del objeto aproximada…)

5.1.1.2 Actualización

En función de la similitud del estado de cada partícula respecto al estado de referencia se le asigna un peso a cada una de ellas.

5.1.1.3 Estimación

A partir de estos valores, se crea un nuevo conjunto de partículas que constituirá la estimación a priori del estado en el siguiente instante de tiempo, para ello suelen emplearse métodos de remuestreo probabilísticos que consideran la probabilidad a posteriori de cada una de las partículas, de forma que aquellas que mejor se ajusten a las medidas disponibles darán lugar a nuevas partículas con mayor probabilidad.

5.1.1.4 Predicción

Una vez que se crea el conjunto de partículas el nuevo instante temporal, se realiza una leve modificación al estado de cada uno de ellos introduciendo algún tipo de ruido auditivo que aporte variabilidad al sistema, con el fin de estimar el estado del objeto en el instante siguiente.

Al terminar la etapa de predicción, se obtiene un nuevo conjunto de partículas al que se le vuelve a aplicar la etapa de actualización, repitiéndose este bucle hasta que termine la secuencia de datos, caso en el cual se volvería a la etapa de inicialización [15].

5.1.2 Filtrado bayesiano recursivo

Para muchos problemas, se requiere una estimación cada vez que se recibe una nueva medida. En este caso, un filtro recursivo es una solución conveniente. Una aproximación mediante filtrado recursivo permite que los datos recibidos sean procesados secuencialmente en vez de en bloque, por lo que no es necesario almacenar todo el conjunto de datos o reprocesar datos existentes cuando una nueva medida está disponible. Un filtro de este tipo consta de dos etapas básicas: predicción y actualización. La etapa de predicción utiliza el modelo de movimiento para predecir el estado de la f.d.p. de un instante al siguiente. [15]

Dado que el estado está normalmente sujeto a perturbaciones desconocidas (modeladas como ruido Gaussiano), la predicción generalmente traslada, deforma y expande la f.d.p.. La etapa de actualización utiliza la última medida para modificar la predicción de la f.d.p..

Para definir el problema del seguimiento, considérese un sistema de ecuaciones movimiento y verosimilitud dependientes del tiempo

$$x_t = f_t(x_{t-1}, v_{t-1}) \quad (5.1)$$

$$z_t = h_t(x_t, n_t) \quad (5.2)$$

5.1.3 El método de Monte Carlo

El método Monte Carlo es un método numérico que permite resolver problemas físicos y matemáticos mediante la simulación de variables aleatorias. El método Monte Carlo fue bautizado así por su clara analogía con los juegos de ruleta de los casinos, el más célebre de los cuales es el de Monte Carlo.

La importancia actual del método Monte Carlo se basa en la existencia de problemas que tienen difícil solución por métodos exclusivamente analíticos o numéricos, pero que dependen de factores aleatorios se pueden asociar a un modelo probabilístico artificial (resolución de integrales de muchas variables, minimización de funciones, etc.).

El Método de Monte Carlo da solución a estos problemas posibilitando la realización de experimentos con muestreos estadísticos en una computadora. El método es aplicable a cualquier tipo de problema, ya sea estocástico o determinístico. A diferencia de los métodos numéricos que se basan en evaluaciones en N puntos en un espacio M-dimensional para producir una solución aproximada, el método de Monte Carlo tiene un error absoluto en la estimación que decrece en $1/\sqrt{N}$en virtud del Teorema Central de Límite. [15]

5.1.4 Filtro de Kalman

Otro de los métodos más utilizados para el tracking de objetos e imágenes en movimiento es el filtro de Kalman, que surge de la adaptación del algoritmo de Rudolf E. Kalman. Este método actúa de forma similar a un predictor lineal donde a partir de una muestra, estimamos la siguiente y posteriormente calculamos el error cometido en dicha predicción, de manera que para el cálculo de la siguiente muestra y teniendo en cuenta dicho error, logremos mejorar la predicción minimizando el error.

5.1.4.1 El proceso a ser estimado

El filtro de Kalman tiene como objetivo resolver el problema general de estimar el estado $X \in \Re n$ de un proceso controlado en tiempo discreto, el cual es dominado por una ecuación lineal en diferencia estocástica de la siguiente forma:

$$X_t = AX_{t-1} + w_{t-1} \tag{5.3}$$

con una medida $Z \in \Re^m$, que es

$$Z_t = HX_t + v_t \tag{5.4}$$

Las variables aleatorias w_t y v_t representan el error del proceso y de la medida respectivamente. Se asume que son independientes entre ellas, que son ruido blanco y con distribución de probabilidad normal:

$$p(w) \cong N(0, Q) \tag{5.5}$$
$$p(v) \cong N(0, R) \tag{5.6}$$

En la práctica las matrices de covarianza de la perturbación del proceso, Q y de la perturbación de la medida R, podrían cambiar en el tiempo, por simplicidad en general se asumen que son constantes.

La matriz A se asume de una dimensión de nxn y relaciona el estado en el periodo previo t-1 con el estado en el momento t. La matriz H de dimensión mxn relaciona el estado con la medición Z_t. Estas matrices pueden cambiar en el tiempo, pero en general se asumen como constantes.

5.1.4.2 El algoritmo

El filtro de Kalman estima el proceso anterior utilizando una especie de control de retroalimentación, esto estima el proceso a algún momento en el tiempo y entonces obtiene la retroalimentación por medio de datos observados.

Desde este punto de vista las ecuaciones que se utilizan para derivar el filtro de Kalman se pueden dividir en dos grupos: las que actualizan el tiempo o ecuaciones de predicción. Las del primer grupo son responsables de la proyección del estado al momento t tomando como referencia el estado en el momento. El segundo grupo de ecuaciones son responsables de la retroalimentación, es decir, incorporan nueva información dentro de la estimación anterior con lo cual se llega a una estimación mejorada del estado.

Las ecuaciones que actualizan el tiempo pueden también ser pensadas como ecuaciones de pronóstico, mientras que las ecuaciones que incorporan nueva información pueden considerarse como ecuaciones de corrección.

El algoritmo de estimación final puede definirse como un algoritmo de pronostico-corrección para resolver numerosos problemas. Así el filtro de Kalman funciona por medio de un mecanismo de proyección y corrección al pronosticar el nuevo estado y su incertidumbre y corregirla proyección con una nueva medida, (ver figura 5.1) [16]

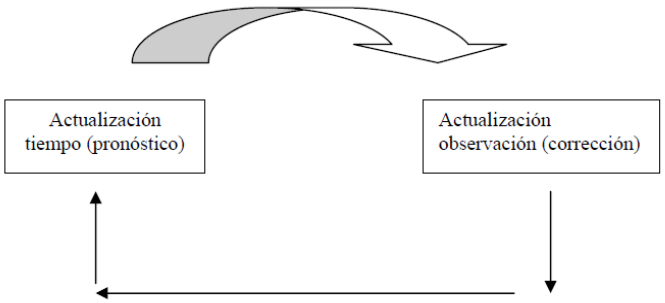

Figura 5.1: El Ciclo del Filtro de Kalman.

El filtro se divide en dos etapas, en la primera se utilizan los datos recopilados para generar una estimación del estado y en la segunda capturando un dato nuevo se procede a una corrección del filtro, así mismo se obtiene una medida mas exacta que la obtenida en ese momento.

Las ecuaciones que definen el sistema son:

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k \tag{5.7}$$

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{5.8}$$

- $\mathbf{x}_k$ y $\mathbf{x}_{k-1}$: Estados del sistema en los instantes ky k-1.

- $\mathbf{F}_k$: Matriz de transicion de estado en el instante k. se define por el modelo del sistema a filtrar.

- $\mathbf{B}_k$: Matriz de control en el instante k. Define las relaciones entre la entrada de control y el estado.

- $\mathbf{u}_k$: Entrada de control en el instante k.

- $\mathbf{w}_k$: Ruido Gaussiano aplicado al estado.

- $\mathbf{z}_k$ y $\mathbf{z}_{k-1}$: Vector con las variables medidas u observaciones del sistema en los instantes y y k-1.

- $\mathbf{H}_k$: Matriz que define el modelo de las observaciones del sistema, que relaciona el estado y la medida en el instante k.

- $\mathbf{v}_k$: Al igual que wk se trata de ruido Gaussiano aplicado a la hora de toman medidas en el instante k.

Ecuacion que define el filro de Kalman para la Predicción.

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \end{aligned} \tag{5.9}$$

Ecuacion que define el filro de Kalman para la Corrección.

$$\begin{aligned} \tilde{\mathbf{y}}_k &= \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \\ \mathbf{S}_k &= \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \end{aligned} \tag{5.10}$$

5.2 Selección de la técnica de filtración

El filtro de Kalman es la asunción de que las variables de estado se distribuyen como una Gaussiana. Por ello el filtro proporciona pobres estimaciones para variables de estado que no sigan esta distribución [15]. A pesar de este problema el filtrado de Kalman exige pocos recursos en Hardware a comparación del filtrado de partículas.

El filtro de Kalman lo podemos encontrar en diversas aplicaciones para el Tracking de objetos o personas y se recomienda su uso debido al bajo consumo de requerimientos. Este método es particularmente efectivo cuando se dan unas condiciones ideales como la ausencia de cambios bruscos en la muestra o cierta pasividad del entorno.

Un ejemplo de la aplicación del filtro de Kalman para el seguimiento de objetos en imágenes de video podemos encontrarlo en Aplicación del filtro de Kalman al seguimiento

de objetos en secuencias de imágenes, se afirma que el filtro de Kalman constituye un buen método para hacer predicciones a lo largo del tiempo de cualquier suceso evolutivo [17].

Los procesos de asociación de datos y aplicación del filtro de Kalman son relativamente estables y con un coste computacional bajo [18].

Comparando las diferentes técnicas se llega a concluir que la mejor selección para el sistema es el filtro de Kalman debido a su bajo consumo de recursos y su amplia biblioteca ofrecida por EmguCV en donde podemos encontrar el algoritmo listo para ser utilizado enviando como parámetros los puntos de las articulaciones a corregir. Podemos observar las comparaciones en la siguiente tabla, donde se detalla a nivel técnico y las necesidades del sistema.

Nombre de Técnica	**Evaluación de puntos**	**Predicción**	**Estimación**	**Recursivo**	**Librería Open Source**	**Librería Plataforma .net**
Filtro de Partículas	SI	SI	SI	SI	NO	NO
Filtrado bayesiano recursivo	SI	SI	NO	SI	NO	NO
Método de Montecarlo	SI	NO	NO	NO	NO	NO
Filtro de Kalman	SI	SI	SI	SI	SI	SI

Tabla 5.2.1: Tabla comparativa de las diferentes técnicas de filtrado.

CAPÍTULO VI

DESARROLLO DEL SOFTWARE “SISTEMA DE SUPERFICIE AUDIOVISUAL CONTROLADO POR MULTIGESTOS KINESTÉSICOS”

CAPÍTULO VI

6.1 Introducción al desarrollo del software “sistema de superficie audiovisual controlado por multigestos kinestésicos”

En este capítulo se describirán los incrementos que se realizaron para obtener el sistema completamente funcional con las características y funcionalidades que se describieron en capítulos anteriores, se hicieron diez incrementos partiendo desde lo básico como es el diseño de la interfaz hasta la programación e implementación de algoritmos de predicción y corrección.

6.2 Incremento 1: interfaz distribución de funcionalidades básicas

A continuación se despliega el incremento inicial de lo que es el sistema de superficie audiovisual.

6.2.1 Análisis

Durante un periodo de una semana se realizaron investigaciones referentes a las diferentes tecnologías donde se puede incorporar el SDK del sensor Kinect, dado las especificaciones y ejemplos publicados por Microsoft, se optó por utilizar el lenguaje de programación C# y tecnología Windows Presentation Foundation (WPF), se decidió utilizar WPF y XAML debido a la forma fácil de programar interfaces gráficas, todos alojados en IDE de Visual Studio 2010.

En todo el proyecto no se utilizó ningún modelo o patrón como MVVM, es una combinación entre estructurada y orientada objetos.

6.2.2 Diseño

Se diseñó un diagrama de secuencias donde se ejemplifica la interacción del usuario con el sistema el cual se muestra a continuación.

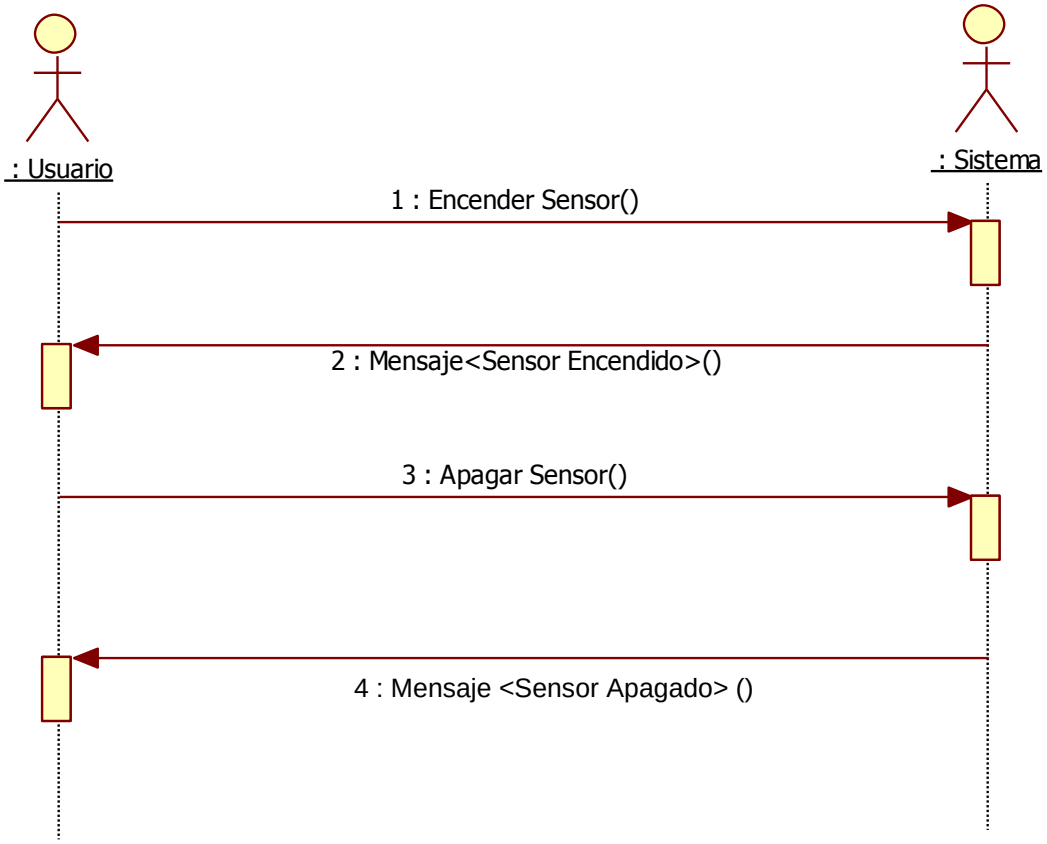

Figura 6.1: Diagrama de secuencias del incremento número 1

El diseño base se realizó de forma rápida incorporando botones, líneas de texto y un lienzo central blanco (ver figura 6.2).

Los botones en este incremento no cuentan con funcionalidad puesto que solo se diseñó para tener una visión de la interacción entre el usuario y programa.

En la parte izquierda superior y derecha se colocaron botones con diseño no estándar como se manejan en programas comerciales.

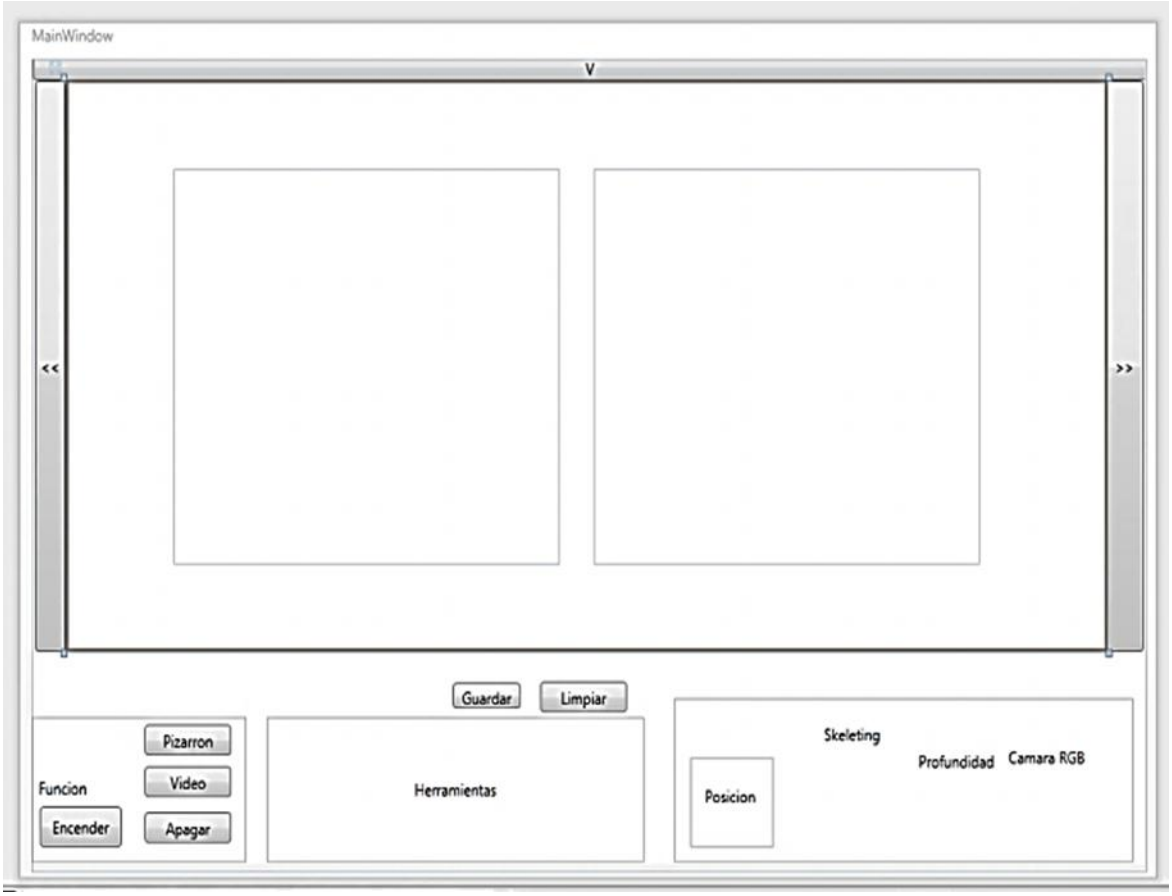

Figura 6.2: Diseño rápido de la interfaz de usuario.

La interfaz UI cuenta con 2 botones para ejecutar eventos como encender y apagar el sensor, visualizar dos ventanas emergentes, guardar y limpiar pantalla.

En la parte inferior derecha se agregó una región que contiene imageViewer que servirá para poder visualizar el Skeleton, profundidad de cámara y la cámara RGB, junto con un cuadro de texto que visualizará la posición de un Joint.

Los botones de forma vertical con símbolo “<<” y “>>” se diseñaron en ese lugar para efectuar eventos de Mouse Hover, este evento es ejecutado al momento de pasar el cursor en el control que fue programado, esta característica de controlar el evento también se le incorporó el botón que muestra el símbolo “v”.

6.2.3 Código

Antes de crear la interfaz, primero se tuvo que crear una solución en Visual Studio 2010, este entorno de desarrollo nos facilita las tareas que se tengan que hacer y proporciona grandes herramientas como el Debug.

Una vez que se creó la solución se procede a crear el proyecto, que lleva el nombre de MainPizarron, en el cuadro de dialogo que nos aparece al momento de crear un proyecto debemos ser específicos en elegir WPF.

6.2.4 Prueba

6.2.4.1 Pruebas por los desarrolladores

Se realizaron pruebas exhaustivas en busca de errores de programación en la ejecución de funciones básicas de la interfaz de usuario, no se encontraron errores por lo cual el incremento pasa sin problemas.

6.2.4.2 Pruebas de Aceptación de usuarios

Tres usuarios probaron la interfaz y utilizaron las funciones básicas del incremento, los usuarios aprobaron la interfaz y la calificaron como amigable.

6.3 Incremento 2: interacción con el usuario y el programa principal con el sensor Kinect

El incremento dos se considera el núcleo de la interacción entre el usuario y el programa. La implementación de las librerías, Dlls, Clases, etc. Tienen gran importancia en esta etapa y se seleccionaron las funciones que retornar posiciones de los Joints de una persona.

Se trabaja sobre la interfaz gráfica UI inicial que se elaboró en el primer incremento, la primera interfaz es útil para visualizar los diferentes valores que devuelve el sensor.

6.3.1 Análisis

Consultando diferentes fuentes para la implementación de SDK´s para el sensor se tomó la decisión de seguir el pequeño ejemplo descrito en el libro Beginning Kinect Programming with the Microsoft Kinect SDK**,** dicho ejemplo está programado para obtener datos de la profundidad de la cámara y obtener información de la cámara RGB. Una vez que se obtuvieron los datos del sensor se procedió a utilizar la Clase Skeleton.

6.3.2 Diseño

Se realizó el siguiente diagrama de secuencias en donde se ejemplifica la interacción en el programa principal, donde el usuario tiene que encender de manera manual, es decir tendrá que dar clic dentro de la ventana donde se encuentra el botón correspondiente. (Ver figura 6.3)

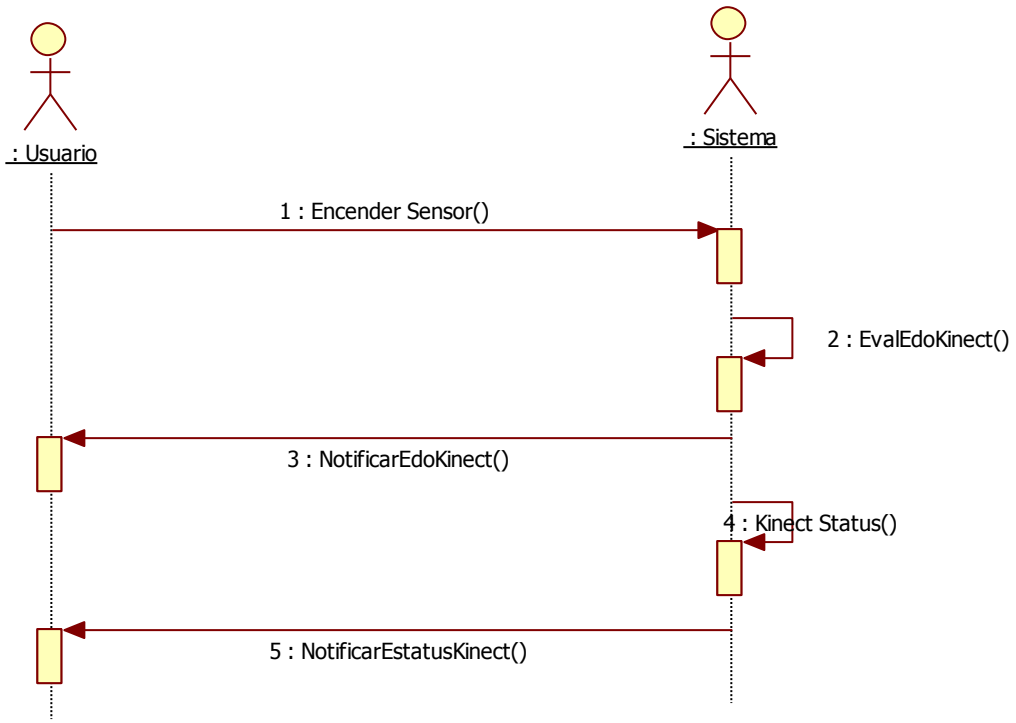

Figura 6.3: Diagrama de secuencias Para inicializar el sensor.

6.3.3 Código

En la región Iniciar_Kinect tenemos la siguiente codificación para utilizar el sensor e inicializar las funciones.

```
/* 1 */
        #region Iniciar_Kinect
        //Iniciamos las funciones para utilizar el Sensor Kinect
        public void Iniciar_Kinect()
        {
            try
            {
                _userInfos = new UserInfo[InteractionFrame.UserInfoArrayLength];
                KinectSensor.KinectSensors.StatusChanged += new EventHandler<StatusChangedEventArgs>
(KinectSensors_StatusChanged); //checa el estado del sensor Kinect
                this.KinectDevice = KinectSensor.KinectSensors.FirstOrDefault(x => x.Status == Kinec
tStatus.Connected);//Tomamos es reporte del estado del sensor

                Estado = 1;
            }
            catch (Exception)
            {
                Estado = 0;
                System.Windows.Forms.MessageBox.Show("Error, verifique el sensor o drivers");
            }
        }
        #endregion Iniciar_Kinect
```

En esta función se inicializan los eventos que dispara la clase KinectSensor, se crea la instancia para poder escucharlo y obtener el estatus que este en primer lugar de la colección KinectSensors. Solo se leerá cuando el sensor esté conectado de lo contrario se omitirá.

Una vez que se tiene los eventos del sensor se procede a crear la función para evaluar el estado actual del sensor, en la siguiente región Evaluar_Estado_Kinect se evalúa el cambio de estado que dispara el evento StatusChangedEventArgs y se utiliza la función Switch para escribir o notifícale al usuario los diferentes estados del sensor.

```
#region Evaluar_Estado_Kinect
        private void KinectSensors_StatusChanged(object sender, StatusChangedEventArgs e)
        {
            switch (e.Status)
            {
                case KinectStatus.Initializing:
                    textBoxMensajes.Text = "Iniciando sensor";
                    break;
                case KinectStatus.Connected:
                    textBoxMensajes.Text = "El sensor esta Conectado";
                    Iniciar_Kinect(); //Iniciamos el sensor
                    break;
                case KinectStatus.NotPowered:
                    textBoxMensajes.Text = "El sensor no esta conectado con el Adaptador de corrient
e";
                    break;
                case KinectStatus.NotReady:
```

```
                textBoxMensajes.Text = "El sensor no se ha iniciado";
                break;
            case KinectStatus.DeviceNotGenuine:
                textBoxMensajes.Text = "El sensor no es Genuino";
                this.KinectDevice = e.Sensor;
                break;
            case KinectStatus.Disconnected:
                textBoxMensajes.Text = "El sensor esta desconectado";
                botonRojo.Visibility = Visibility.Collapsed;
                this.KinectDevice = null;
                break;
            case KinectStatus.DeviceNotSupported:
                textBoxMensajes.Text = "No tiene conectado el sensor Kinect para Windows";
                botonRojo.Visibility = Visibility.Collapsed;
                this.KinectDevice = null;
                break;
            case KinectStatus.Undefined:
                textBoxMensajes.Text = "Estado indefinido";
                botonRojo.Visibility = Visibility.Collapsed;
                this.KinectDevice = null;
                break;
            default:
                textBoxMensajes.Text = "***error estado desconocido***";
                break;
        }
    }
    #endregion Evaluar_Estado_Kinect
```

En total se cuenta con nueve estados diferentes, la notificación es esencial al momento de que el usuario este utilizando el sensor y también es muy importante cuando se estén realizando pruebas por tal motivo se cuidó la parte de notificar desde que el sensor se conecta hasta cuando se presentan fallos del mismo.

En la siguiente región _PropertiesKinect se configura el sensor para que funcione con valores de resolución de profundidad de 640 x 480 pixeles a 30 fps (frames por segundo), también se inicia la clase Skeleton con su respectivo evento para empezar a hacer el Tracking del usuario.

```
//Tomamos el Skeleton
this.KinectDevice.SkeletonFrameReady += KinectDevice_SkeletonFrameReady;

this._KinectDevice.ColorStream.Enable();

this._KinectDevice.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);

this._KinectDevice.SkeletonStream.Enable();

this._KinectDevice.Start(); //Iniciamos el sensor
```

La configuración procede siempre y cuando el sensor este inicializado, por tal motivo evaluamos la variable global _KinectDevice y verificamos que su estatus sea conectado y listo para funcionar.

Una vez configurado el sensor se procede a configurar la clase Skeleton para obtener los Joints que nos conciernen, en el siguiente código se podrán ver diferentes Joints y cada uno es importante para los incrementos que se plantean adelante.

```
                    /* Obtenemos los Joints que se utilizaran en el programa*/
                    Joint HandLeft = GetHandLeft(skeleton);
                    Joint HandRight = GetHandRight(skeleton);
                    Joint Head = GetHead(skeleton);
                    Joint homIzq = GetHombroIzq(skeleton);
                    Joint homDer = GetHombroDer(skeleton);
                    Joint homCen = GetHombroCen(skeleton);
                    Joint CentroCadera = GetCadCen(skeleton);
                    /* Obtenemos los Vectores de profundidad "Z" de los Joints que utilizaran en
el programa*/
                    float zManoDerecha = skeleton.Joints[JointType.HandRight].Position.Z;
                    float zManoIzquierda = skeleton.Joints[JointType.HandLeft].Position.Z;
                    float zCabeza = skeleton.Joints[JointType.Head].Position.Z;

                    if (zCabeza > 0/* && zCabeza < 3*/) //Evalua que el usuaro no esta alejado a
3 metros de distancia y que no se encuentre a menos de un metro de distancia del sensor
                    {
                        botonVerde.Visibility = Visibility.Visible;
                        botonRojo.Visibility = Visibility.Collapsed;
                    }
                    else
                    {
                        botonRojo.Visibility = Visibility.Visible;
                        botonVerde.Visibility = Visibility.Collapsed;
                    }
```

Como se puede apreciar en el código anterior se toman los Joints de mano derecha, mano izquierda, cabeza, hombro izquierdo, hombro derecho, hombro central y cadera central, también se obtiene el ventor Z de la mano derecha, mano izquierda y la cabeza del usuario, estos tres últimas variables servirán para complementar los tres ejes del usuario. También se agregó una pequeña funcionalidad de encender un botón cuando el usuario este posicionado frente al sensor.

Una vez que se tiene toda la parte configuración del sensor se procede a comunicar el **.cs** (Código C#) con el XAML (Código de la interfaz gráfica). La comunicación se basa en eventos y escribir en las propiedades de los controles, para este incremento se escriben en los controles de texto la posición X y Y de la mano izquierda, cabeza y mano derecha, a la vez se mueven tres figuras que representan los Joints en los dos ejes de la pantalla (ver figura 6.4)

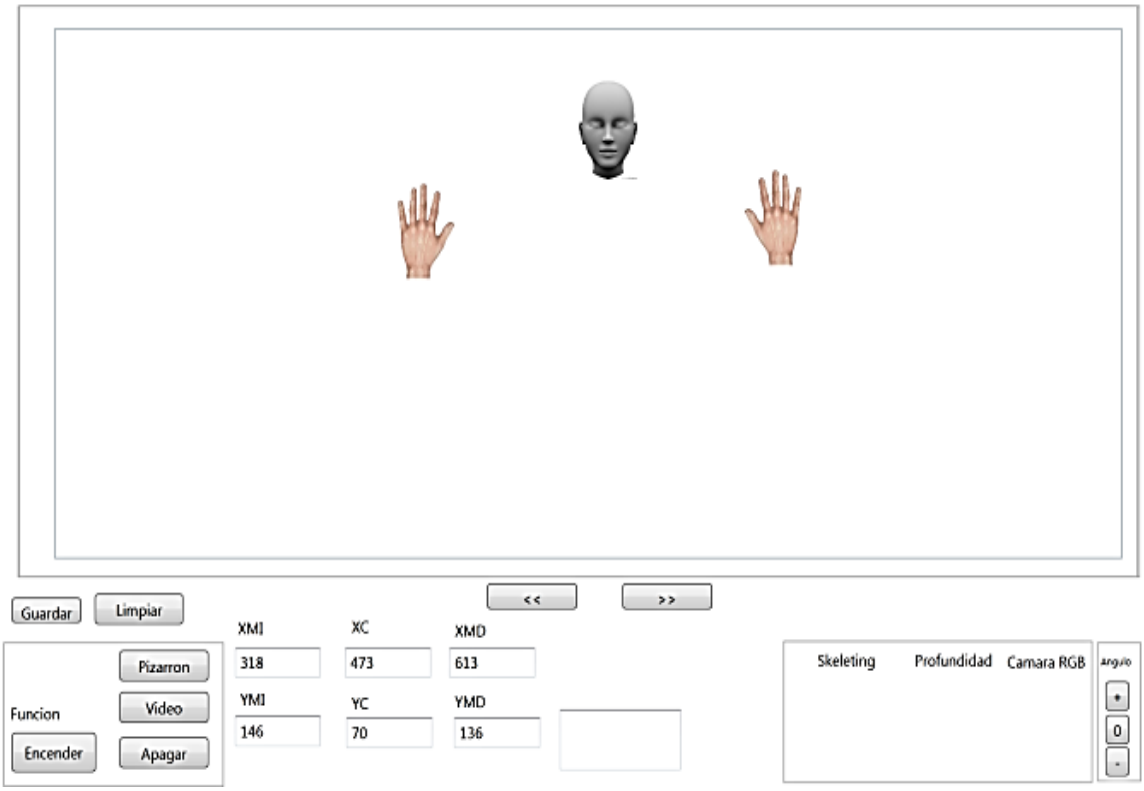

Figura 6.4: Pantalla principal, interfaz Gráfica UI.

6.3.4 Prueba

6.3.4.1 Pruebas por los desarrolladores

Las pruebas realizadas en el incremento se basan en la verificación del movimiento que realiza un usuario común frente al sensor y ver el cambio en los diferentes controles evaluando las cantidades pintadas en tiempo real.

Se realizaron pruebas con tres usuarios de diferentes estaturas y tallas que no están familiarizadas con el sensor Kinect, se les instruyo a los usuarios que se comportaran de manera nutual frente al sensor Kinect, cuando el usuario se aleja o cambia de distancia respecto al sensor se tiene incrementos o decrementos de los valores, observando la variabilidad se incorporará en un incremento una función para cancelar las funciones cuando el usuario se encuentre en una distancia mayor a dos metros.

Los valores obtenidos por el sensor se muestran estables pero se observa una perdida de precisión cuando el usuario emite movimientos bruscos.

Todas las pruebas realizadas se hicieron en un equipo Laptop con las siguientes características:

- Procesador Intel Core 2 duo.

- Ram: 4Gb DDR2

- Video: ATI Radeon 3650 512 Mb

- Disco duro: 500 Gb

6.3.4.2 Pruebas de Aceptación de usuarios

Los usuarios seleccionados para realizar las pruebas fueron los mismos que participaron en las pruebas por los desarrolladores, a este grupo de usuarios se les hizo la pregunta de cómo sintieron la interacción con el kinect respecto a los objetos pintados en la pantalla, los tres opinaron que las imágenes pueden resultar incomodas cuando se requiera hacer uso de una presentación, pero también opinaron su alto valor para poder señalar alguna parte del contenido de la exposición.

Tomando en consideración los objetos pintados en tiempo real para la señalización de los Joints, se ocultaran y se visualizaran cuando se requieran.

6.4 Incremento 3: mejoramiento en la interacción interfaz gráfica-usuario e implementación de funciones para el control de diapositivas a través de gestos kinestésicos

6.4.1 Análisis

La funcionalidad del incremento número tres es visualizar una cantidad de diapositivas en el programa principal y además donde el usuario pueda manipularlas libremente a una distancia remota utilizando únicamente gestos básicos de las extremidades superiores.

Para el desarrollo de esta funcionalidad se necesita modificar la interfaz gráfica y añadir un segmento especial de programación en XAML en el archivo principal, el código C# es el que se encargara de mover las diapositivas a partir de una ruta predeterminada.

Para tener un orden en nuestro proyecto renombraremos la pantalla donde se visualizan las diapositivas y la llamaremos ModoPresentador por lo consiguiente los archivos .XAML (código de diseño de la interfaz) y .CS (Código en C# lógica de programación) se renombraran.

Primeramente se agregó un nuevo control al diseño, el control lleva el nombre de <Image> y su función será la de visualizar las diapositivas en formato de imagen. Este control responderá con el nombre “Image_1” y será controlado en el archivo .cs.

6.4.2 Diseño

Para la creación de este incremento se diseñó el siguiente diagrama de secuencias (ver figura 6.5) donde se demuestran las posibles interacciones del usuario con el sistema.

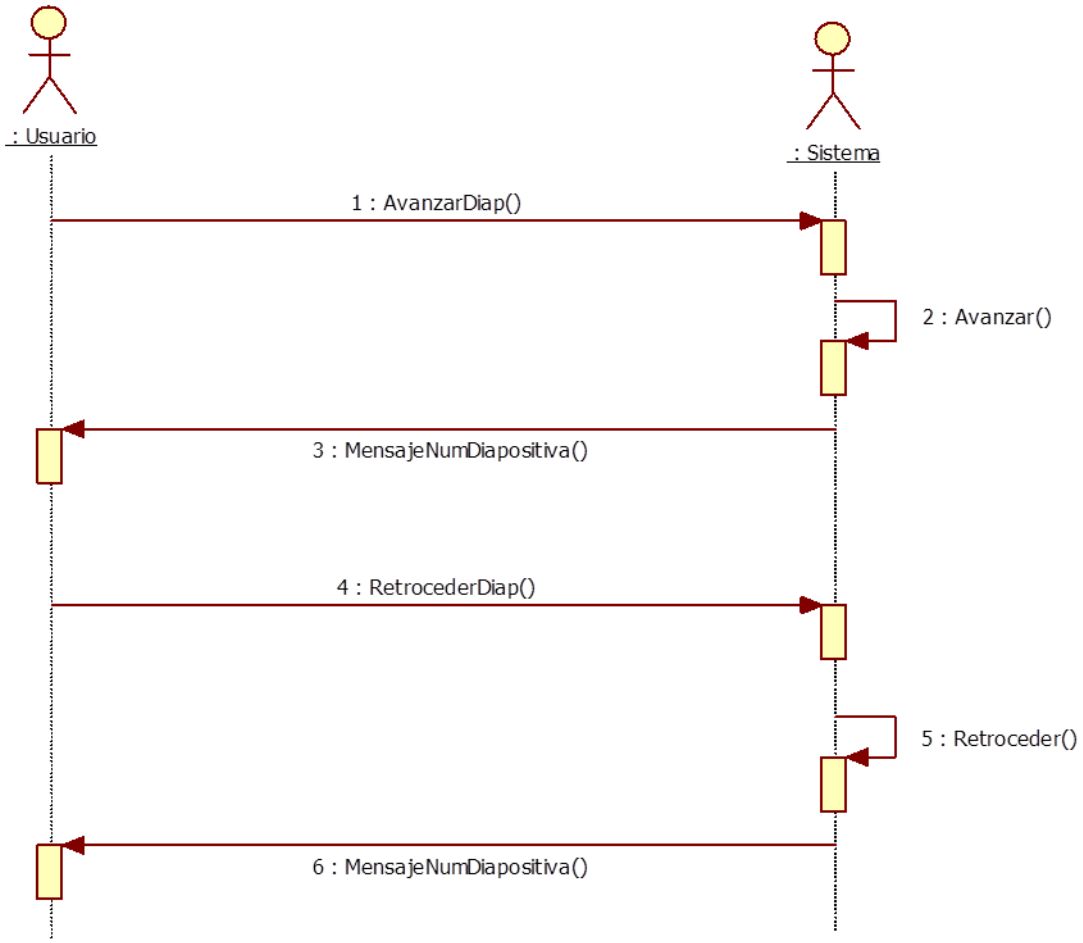

Figura 6.5: Diagrama de secuencias para el avance y retroceso de diapositivas.

El diseño en el incremento fue a nivel control, no se realizó una modificación de estructura, orden o ubicación, dado a que el requerimiento solo es necesaria una implementación en el archivo de diseño del programa, por tal motivo no se visualizó un cambio importante (ver figura 6.6).

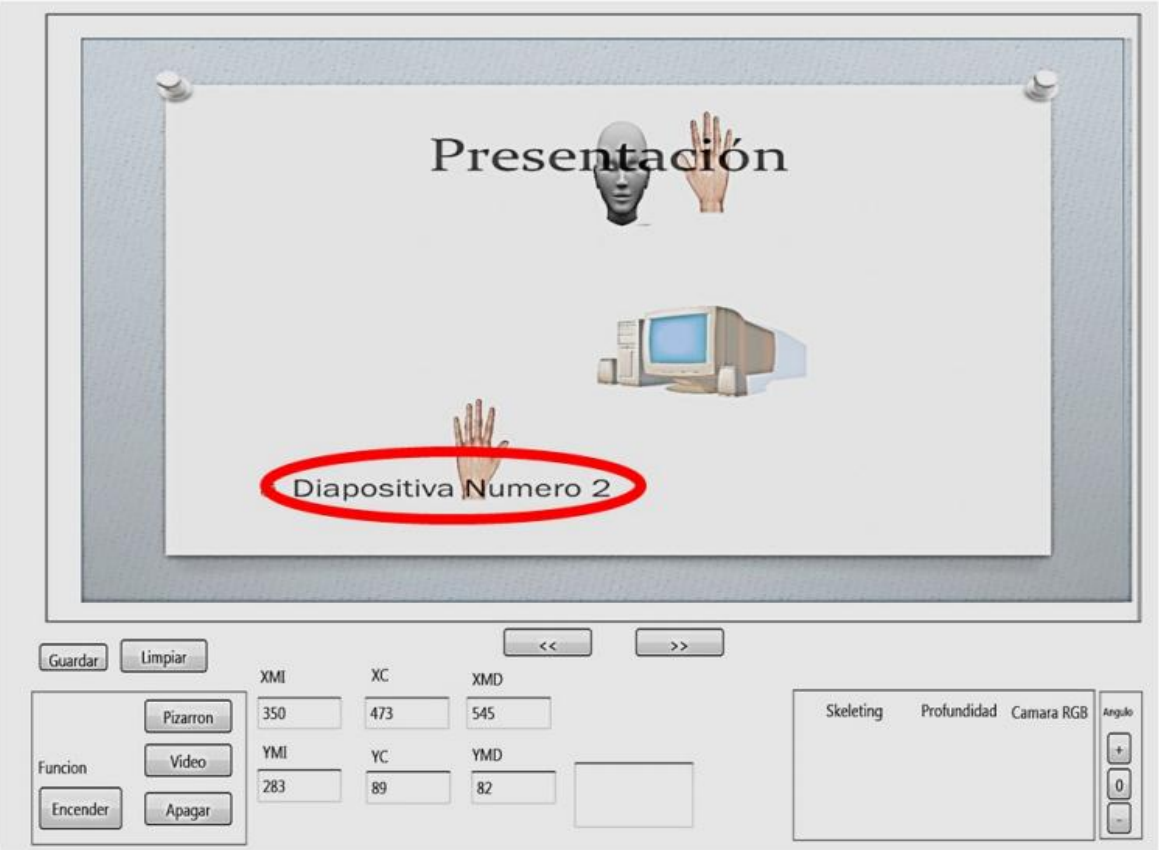

Figura 6.6: Pantalla principal ModoPresentador, interfaz Gráfica UI.

Como se puede apreciar en la figura anterior se visualiza una hoja gris clara con tachuelas que contiene la imagen de una computadora, en la parte inferior y superior podemos encontrar las tres imágenes que simulan las posiciones de la mano derecha, mano izquierda y la cabeza del expositor.

El cambio de diapositiva se da de manera continua por tal motivo se empezó con la diapositiva número 1 y se terminó con la diapositiva número 2.

6.4.3 Código

Dentro del archivo XAML se agregó solo una línea de código, el cual es el encargado de visualizar las imágenes o diapositivas tal como se puede apreciar:

```
<!-- Image main -->
<Image x:Name="Image_1" Grid.RowSpan="26" Grid.ColumnSpan="20" Visibility="Collapsed" />
```

El código anterior es el control implementado con las propiedades de ubicación y visibilidad, también se le llamo con el nombre de “Image_1” para poder comunicar el código .cs

6.4.4 Prueba

6.4.4.1 Pruebas por los desarrolladores

Las pruebas realizadas en este incremento parten de la observación de tres usuarios comunes que no están familiarizados con el sensor Kinect, se les pidió a cada uno de ellos interactuar con el programa (ver figura 6.7), la prueba principal consiste en cambiar de la diapositiva número 1 a la diapositiva número 2.

Durante las pruebas realizadas de interacción se encontraron nuevamente las variaciones en las coordenadas de los Joints de las articulaciones de las manos.

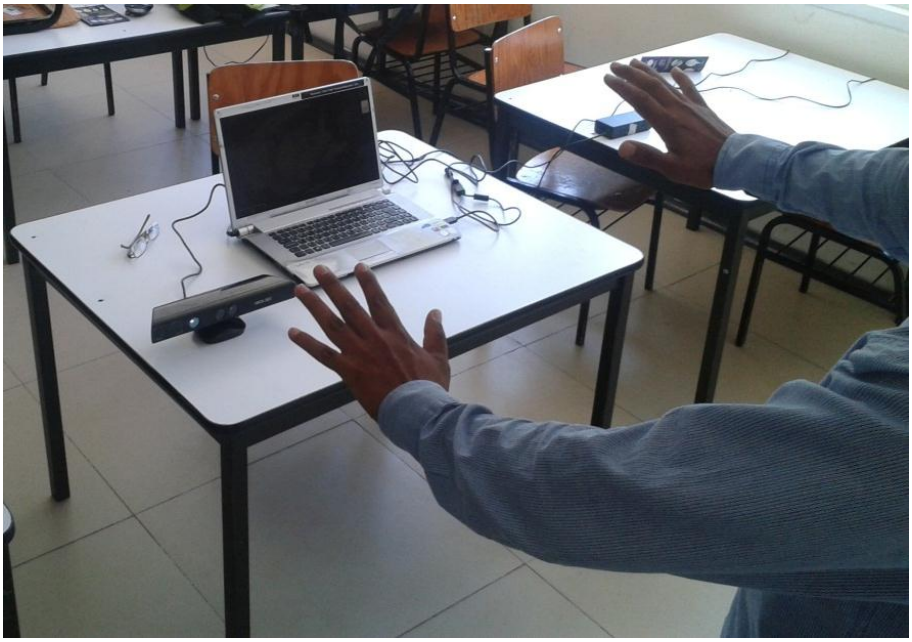

Figura 6.7: Usuario común comunicándose con la interfaz gráfica.

6.4.4.2 Pruebas de Aceptación de usuarios

Tres usuarios probaron el programa durante un periodo de quince minutos, se les pidió con anticipación realizaran movimientos naturales y cambiaran de diapositiva, una vez que lograron completar la acción, dijeron algo importante, la comunicación podría resultar estresante debido al movimiento realizado para cambiar de diapositiva. Por lo consiguiente en los siguientes incrementos se modificara el algoritmo para la detección del gesto avanzar y retroceder.

6.5 Incremento 4: funcionalidad de creación de proyectos personales para guardar diapositivas en formato .JPG y Video en formato .MP4

6.5.1 Análisis

El desarrollo de este incremento tiene como objetivo crear un ambiente de trabajo para la creación de proyectos personales de un expositor, hasta hoy en día todos los programas utilizan un archivo para guardar la información por ejemplo: PowerPoint, para posteriormente ser editados.

Diseñar una interfaz gráfica sencilla fácil de utilizar y con la funcionalidad de guardar cambios realizados por el usuario, toda imagen o video que se agreguen al proyecto serán guardados automáticamente, esto con el fin de poder cargarlos al momento de comenzar la exposición, este incremento está ligado al 5 donde se hace la integración y se saca provecho la funcionalidad de creaciones de proyectos.

6.5.2 Diseño

Los flujos necesarios para la creación de proyectos son completamente fáciles de recordar debido a que se manejó el estándar de Windows es decir, se implementaron controles nativos de Microsoft, como botones, barra de menú entre otros.

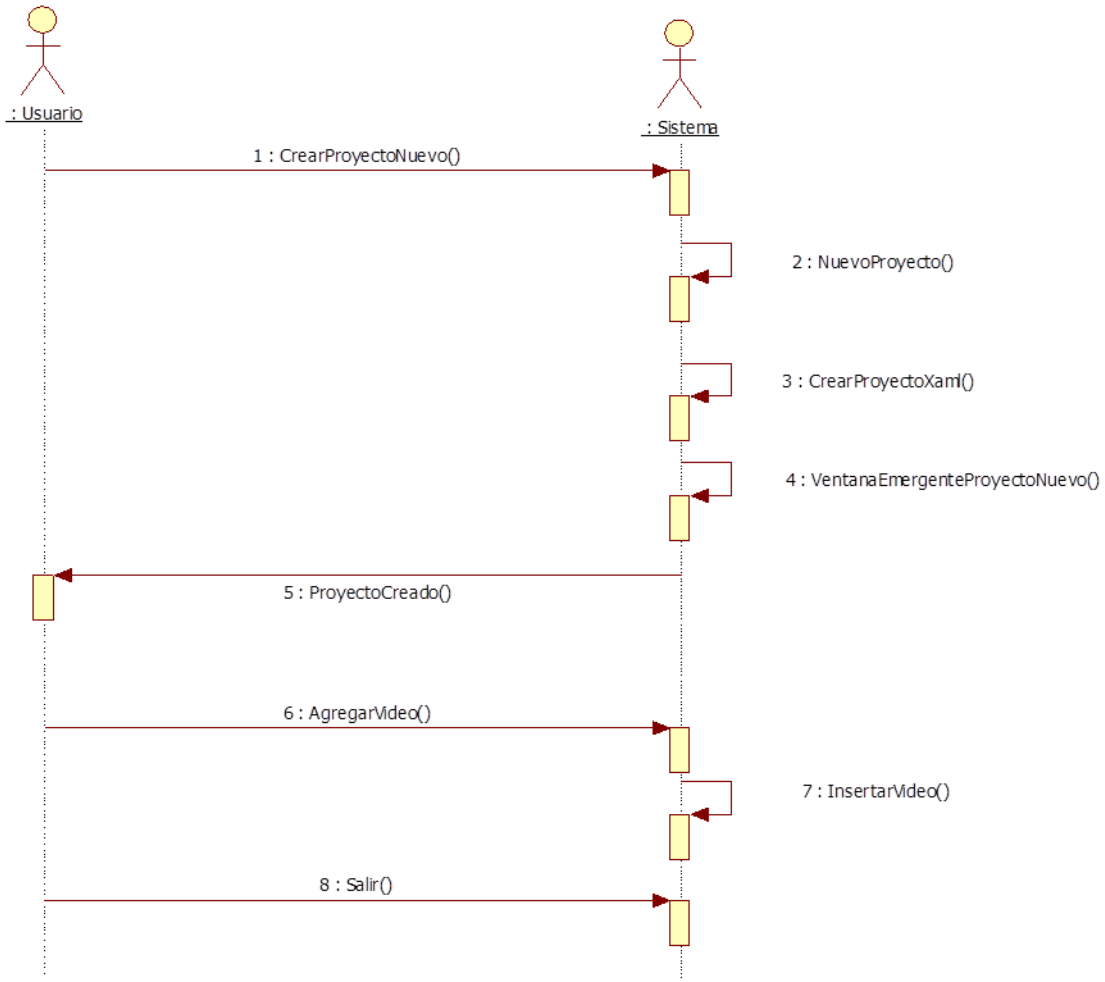

Figura 6.8: Diagrama de secuencias para la creación de proyectos

El sistema completo cuenta al inicio con la pantalla de bienvenida y el Dashboard o panel de control donde se visualizan las diapositivas que se están en el proyecto actual.

Al inicio del programa se visualiza el proyecto predeterminado donde se podrá lanzar a modo presentador, el botón localizado en la parte inferior derecha, esto con el fin de ver o utilizar las funciones del pizarrón virtual.

En la parte superior se diseñó el menú donde se localizan las funciones de creación de proyecto, abrir, cerrar, iniciar modo presentador entre otros (Ver figura 6.9).

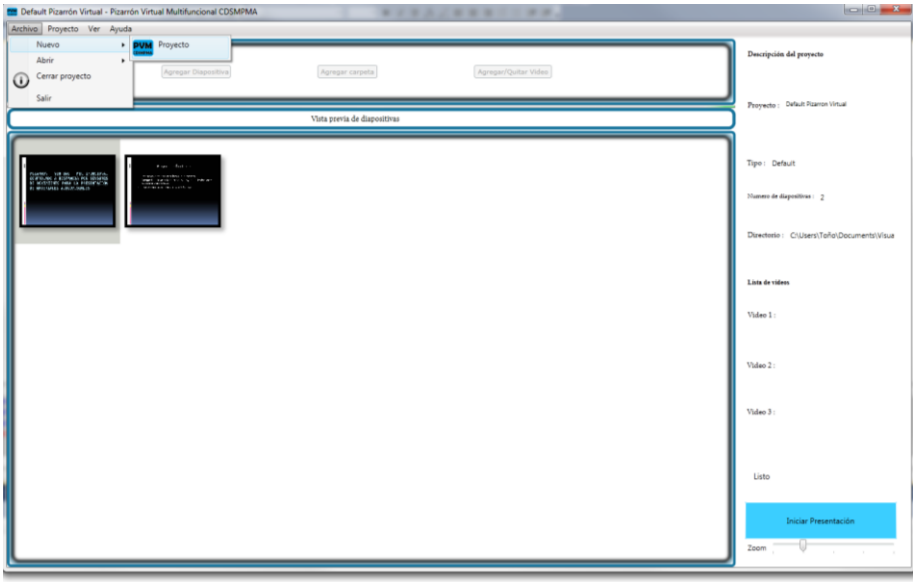

Figura 6.9: Interfaz gráfica principal.

Al momento de hacer clic sobre Proyecto visualizaremos la ventana flotante donde encontraremos controles que nos ayudaran a crear nuestro proyecto, como default y requerimiento inicial de creaciones de proyecto se construye el proyecto en blanco sin ningún tipo de plantilla sin ningún tipo de animaciones o efectos.

El requerimiento no especifica la utilidad de poder cambiar de dirección o ruta del proyecto y por defecto se dejara en la unidad C del sistema: C:\Pizarron Virtual Multifuncional CDSMPMA\Proyectos\Board plus (ver figura 6.10).

La ruta se crea al momento de generar el primer proyecto por tal motivo cuando se necesite revisar la dirección después de la instalación no se encontrara y el proyecto predeterminado se aloja en el directorio raíz del programa para evitar posible error de borrado por parte del usuario.

Una vez que se tiene la dirección a guardar el usuario solo necesita presionar tres botones para poder crear su proyecto personal, la generación del archivo no toma tiempo debido a su simplicidad, la estructura del archivo no cuenta con encriptación alguna y se guarda en

binario con estructura que definen el nombre, cantidad de diapositivas, numero de videos entre otras especificaciones.

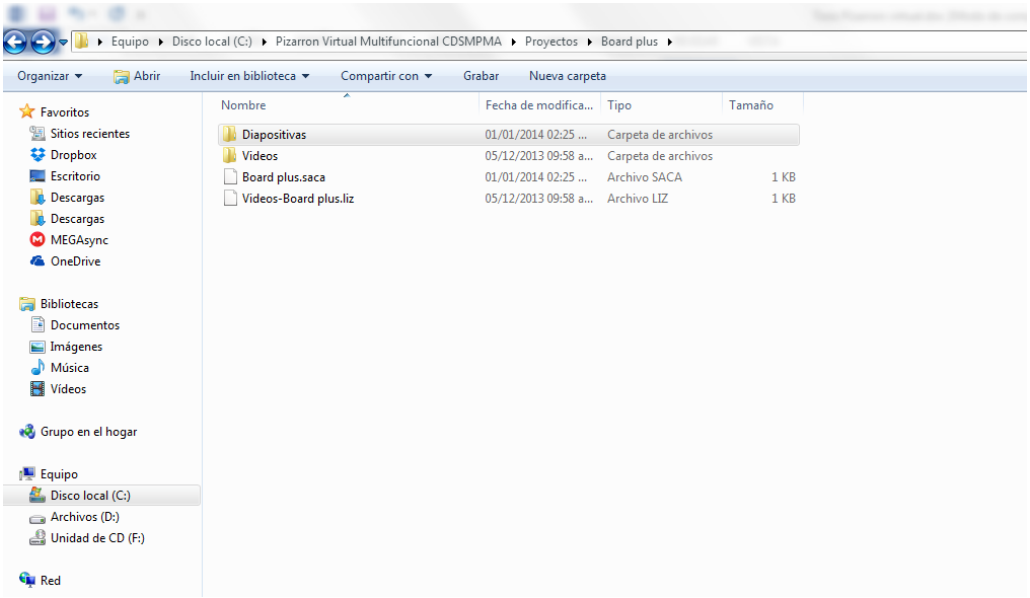

Figura 6.10: Ruta de alojamiento de proyectos.

Los pasos para generar el proyecto consiste en presionar el botón “Seleccionar” el cual define al proyecto de tipo blanco, a continuación se necesita escribir el nombre del proyecto sin límite de caracteres pero limitados de caracteres no válidos definidos en el estándar de Windows (ver figura 6.11).

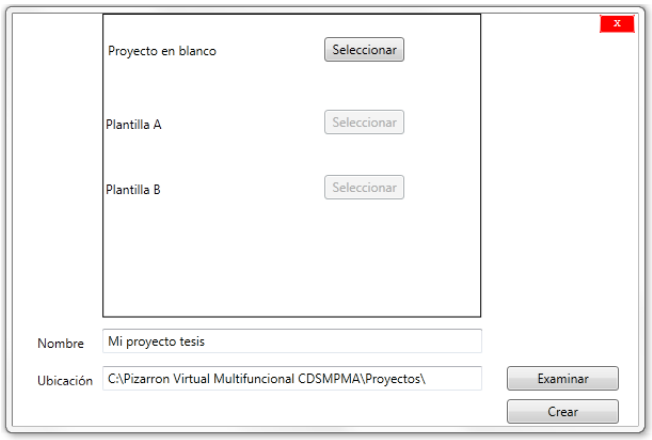

Figura 6.11: Ruta de alojamiento de proyectos.

Una vez que se tiene el proyecto creado, automáticamente se abrirá y se pondrá en estado listo para modificar, la interfaz de edición cuenta con 4 botones que son utilizados para agregar diapositivas de forma individual o por carpeta, también se encuentra el botón de agregar o quitar videos y este último recurso solo ese puede agregar de manera individual sin la opción de poder agregar una carpeta completa (ver figura 6.12).

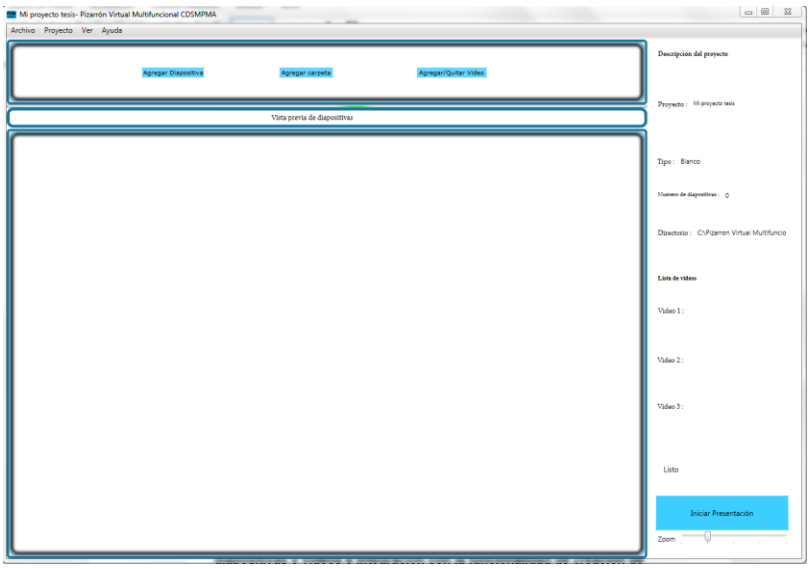

Figura 6.12: Ruta de alojamiento de proyectos.

Para agregar diapositivas al proyecto solo basta seleccionar las que se necesiten, la funcionalidad se utiliza el botón “Agregar Diapositiva” donde se dibujara una ventana emergente donde podremos utilizar para navegar en los directorios del sistema para localizar las diapositivas en formato de imagen y así agregarlas, la interfaz gráfica es fácil de utilizar debido a la utilización de recursos del sistema operativo Windows bajo su estándar (ver figura 6.13).

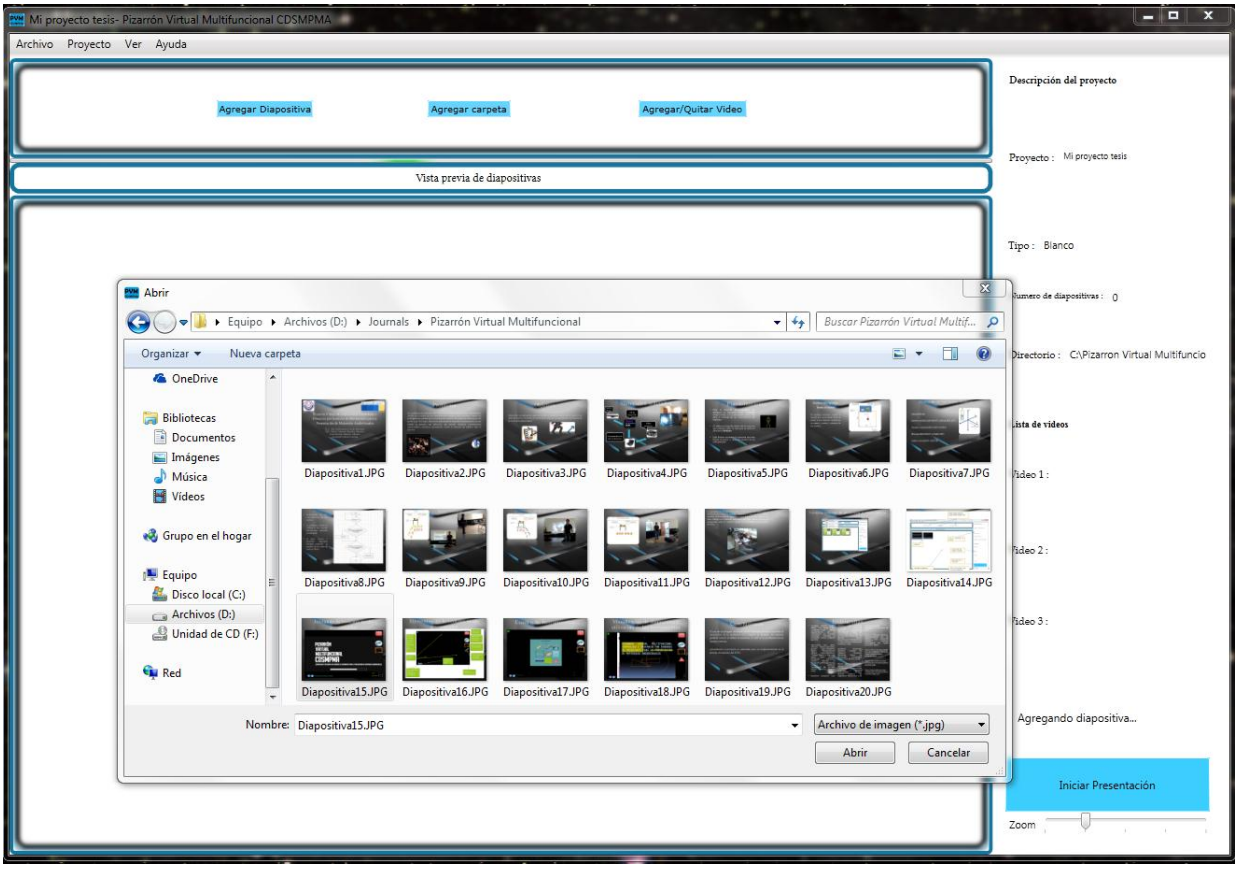

Figura 6.13: Ventana emergente

.

Al momento de agregar las diapositivas que se utilizaran estas se visualizaran en el panel ordenadas en el orden que se agregaron, dentro de la programación se incorpora el algoritmo para escribir en el archivo e ir guardando los identificadores en el orden definido por el usuario, la vista preliminar se muestran en forma de cuadros en miniatura y también se adaptó un control slider para hacer zoom para poder ver más grandes o pequeñas las imágenes preliminares (ver figura 6.14).

Las imágenes agregadas al proyecto se guardan en el directorio del mismo renombrando cada una con un identificador numérico, lo mismo sucede con los archivos de videos, al momento de ser agregados al proyecto.

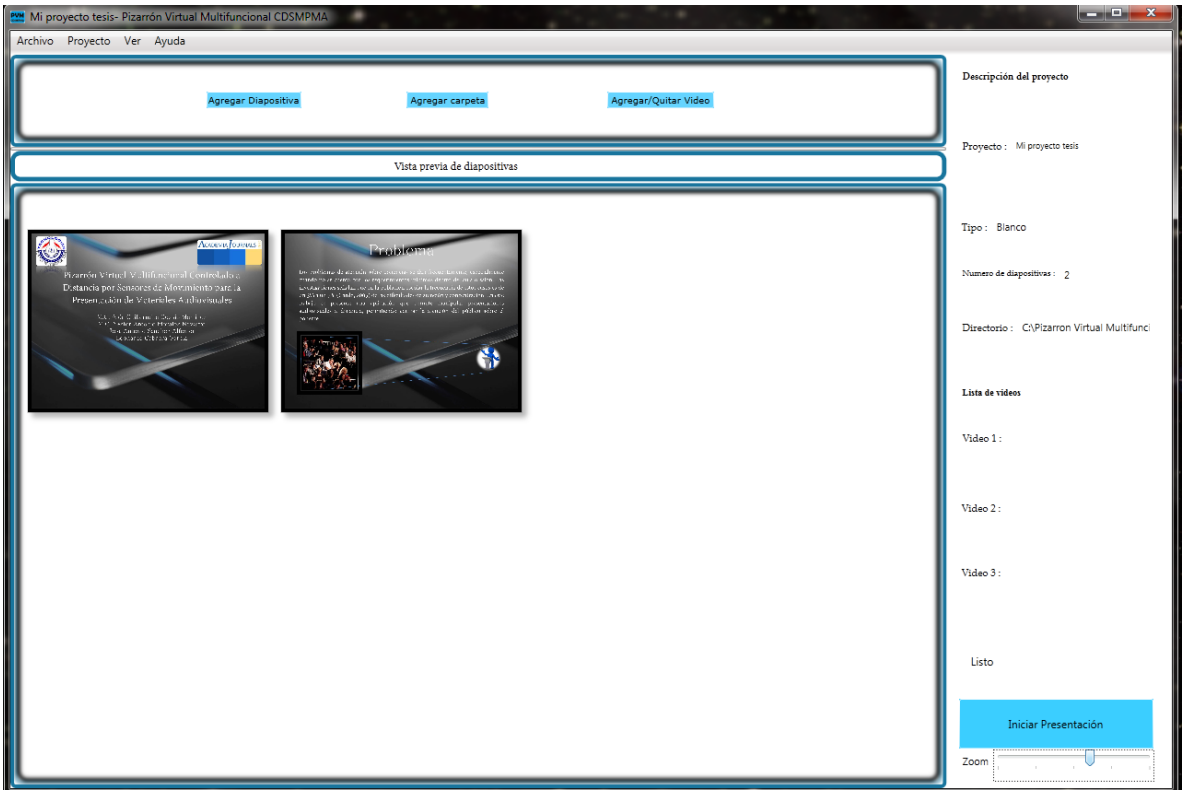

Figura 6.14: Vista preliminar de diapositivas

Se cuenta con el diseño especial para agregar videos, la forma que se diseño es simple pero también básica, es limitado por un número cuyo valor máximo es de 3 por cada proyecto (ver figura 6.15).

El sistema solo puede reconocer el formato .MP4 dado los requerimientos básicos iniciales para el proyecto y por ser un formato estándar.

Como se mencionó anteriormente los videos son guardados en el directorio del mismo proyecto, renombrados con un identificador especial y guardados en el archivo del programa.

No se limitó en cuanto a tamaño de video, es ilimitado, por tal motivo se agregó el pequeño requerimiento para notificar al usuario cuando el video fue agregado correctamente.

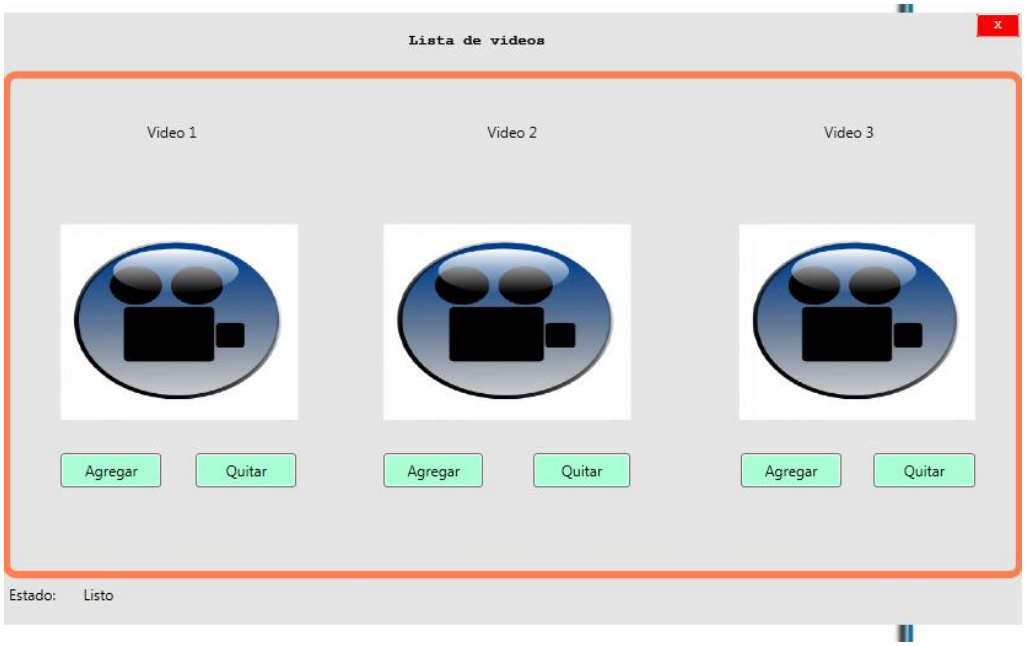

Figura 6.15: Ventana para agregar videos.

Dentro del diseño global se implementa de un apartado nombrado columna descripción del proyecto, donde el usuario pudra visualizar toda la información relevante (Ver figura 6.16).

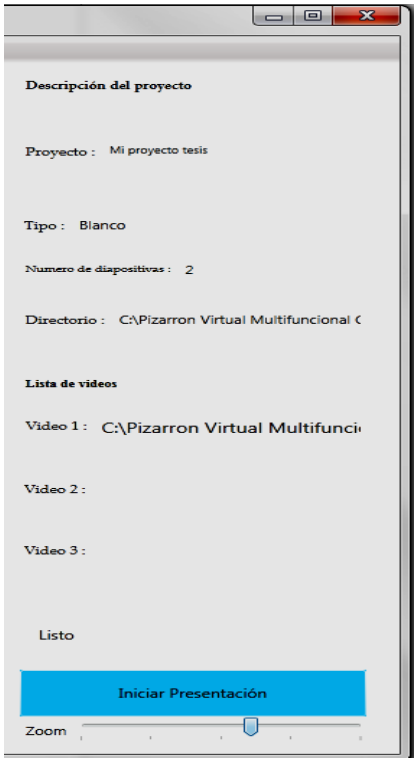

Figura 6.16: Barra de descripción del proyecto.

6.5.3 Código

Siguiendo las estructuras de las interfaces graficas la programación se realiza en código XAML y código .cs, se diseñara la estructura del proyecto de Visual Studio centrando la interfaz gráfica de visualización de proyectos como Main de toda la solución.

Al ser el primer requerimiento inicial en cuanto a visualización y creación de proyectos, se codifica todos los controles y diseños en un solo documento XAML, nombrado MainWindow.xaml.

Los recursos en el main para la interfaz gráfica se definen en las siguientes líneas de código.

```
<!-- Styles and Templates -->
    <Window.Resources>
        <ObjectDataProvider x:Key="Photos" ObjectType="{x:Type er:PhotoCollection}" />
        <!-- Photo Template -->
        <DataTemplate DataType="{x:Type er:Photo}">
            <Grid VerticalAlignment="Center" HorizontalAlignment="Center" Margin="6">
                <!-- Drop Shadow -->
                <Border HorizontalAlignment="Stretch" VerticalAlignment="Stretch" CornerRadius="4" Background="#44000000">
                    <Border.RenderTransform>
                        <TranslateTransform X="5" Y="5" />
                    </Border.RenderTransform>
                    <Border.BitmapEffect>
                        <BlurBitmapEffect Radius="8" />
                    </Border.BitmapEffect>
                </Border>

                <!-- Image Template -->
                <Border Padding="4" Background="Black" BorderBrush="#22000000" BorderThickness="1">
                    <StackPanel Orientation="Vertical">
                        <Image Source="{Binding Image}"/>
                    </StackPanel>
                </Border>
            </Grid>
        </DataTemplate>

        <!-- Main photo catalog view -->
        <Style TargetType="{x:Type ListBox}" x:Key="PhotoListBoxStyle">
            <Setter Property="Foreground" Value="White" />
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type ListBox}" >
                        <WrapPanel Margin="5" IsItemsHost="True" Orientation="Horizontal"
                      ItemHeight="{Binding ElementName=ZoomSlider, Path='Value'}"
                      ItemWidth="{Binding ElementName=ZoomSlider, Path='Value'}"
                      VerticalAlignment="Top" HorizontalAlignment="Stretch" />
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>

        <!-- Style for an individual generic item  - desing for select -->
        <Style TargetType="{x:Type ListBoxItem}">
            <Setter Property="Background" Value="Transparent" />
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type ListBoxItem}" >
                        <Border SnapsToDevicePixels="True" HorizontalAlignment="Stretch" VerticalAlignment="Stretch" Background="{Tem
plateBinding Background}">
                            <ContentPresenter />
                        </Border>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsSelected" Value="True">
                                <Setter Property="Background" Value="#445B6249"   />
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
```

```
    <!-- Removes dotted rectangle focus -->
    <Style TargetType="{x:Type ItemsControl}">
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="{x:Type ItemsControl}" >
                    <WrapPanel IsItemsHost="True" />
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>

    <!-- For metadata properties pane -->
    <Style TargetType="{x:Type GroupBox}">
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="{x:Type GroupBox}" >
                    <Grid>
                        <Border Background="#AAFFFFFF" CornerRadius="4" BorderBrush="#FF18759D" BorderThickness="4">
                            <Border CornerRadius="4" BorderBrush="#88FFFFFF" BorderThickness="1" ClipToBounds="true" >
                                <Border CornerRadius="6" BorderThickness="2" BorderBrush="#FF18759D">
                                    <Border.BitmapEffect>
                                        <BlurBitmapEffect Radius="6" />
                                    </Border.BitmapEffect>
                                    <Border CornerRadius="9" BorderThickness="4" BorderBrush="Black">
                                        <Border.BitmapEffect>
                                            <BlurBitmapEffect Radius="2" />
                                        </Border.BitmapEffect>
                                    </Border>
                                </Border>
                            </Border>
                        </Border>
                        <ContentPresenter Margin="2" />
                    </Grid>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>

    <!-- Default label style -->
    <Style TargetType="{x:Type Label}">
        <Setter Property="FontFamily" Value="Segoe UI" />
        <Setter Property="FontSize" Value="11" />
    </Style>

    <!-- Headers for metadata properties -->
    <Style x:Key="MetadataHeader" TargetType="{x:Type Label}">
        <Setter Property="Background">
            <Setter.Value>
                <LinearGradientBrush StartPoint="0,0.5" EndPoint="1,0.5" >
                    <LinearGradientBrush.GradientStops>
                        <GradientStop Offset="0.5" Color="{x:Static SystemColors.AppWorkspaceColor}" />
                        <GradientStop Offset="2" Color="Transparent" />
                    </LinearGradientBrush.GradientStops>
                </LinearGradientBrush>
            </Setter.Value>
        </Setter>
        <Setter Property="Foreground" Value="White" />
        <Setter Property="FontWeight" Value="Bold" />
    </Style>
</Window.Resources>
```

Como se puede observar las definiciones se centran en el diseño de la interfaz, colores, estilos, etc. Este código puede estar en un documento xaml por aparte y localizado en una carpeta de recursos, se optó por dejarlo sin la estructura separa por ser el requerimiento inicial en la interfaz.

Al terminar de definir los recursos, se inicia la parte importante de codificación respecto a la pre visualización de diapositivas y distribución de controles.

Los controles principales se definen dentro del control DockPanel y dentro de él los controles Menu y Grid nativos de WPF.

La función principal del control Menú es la de construir un menú principal para hospedar los requerimientos o funciones del pizarrón y el Grid toma el trabajo importante de ordenar los controles que se encuentran definidos en su estructura, tales como Label, Button, GroupBox, ListBox, Slider y ProgresBar, a continuación se mostrará la codificación de todo el visualizador de proyectos.

```
<DockPanel Name="MyPanel" Grid.Row="0" Grid.Column="0" Margin="0,0,0,0">
        <Menu  DockPanel.Dock="Top" Height="26" >
            <MenuItem Header="Archivo">
                <MenuItem Header="Nuevo">
                    <MenuItem Header="Proyecto" Click="MenuItem_Click_NuevoProyecto">
                        <MenuItem.Icon>
                            <Image Source="/PVMCDSMPMA;component/Imagenes/PVMCDSMPMA.ico" />
                        </MenuItem.Icon>
                    </MenuItem>
                </MenuItem>
                <MenuItem Header="Abrir">
                    <MenuItem Header="Proyecto" Click="MenuItem_Click_AbrirProyecto">
                        <MenuItem.Icon>
                            <Image Source="/PVMCDSMPMA;component/Imagenes/PVMCDSMPMA.ico" />
                        </MenuItem.Icon>
                    </MenuItem>
                </MenuItem>
                <MenuItem Header="Cerrar proyecto" Click="MenuItem_Click_Cerrar_proyecto">
                    <MenuItem.Icon>
                        <Image Source="/PVMCDSMPMA;component/Imagenes/Shut%20Down.ico" />
                    </MenuItem.Icon>
                </MenuItem>
                <MenuItem Header="Salir" Click="MenuItem_Click_Salir"/>
            </MenuItem>
            <MenuItem Header="Proyecto">
                <MenuItem Header="Iniciar Presentacion" Click="Inciar_Click">
                    <MenuItem.Icon>
                        <Image Source="/PVMCDSMPMA;component/Imagenes/Easel2.ico" />
                    </MenuItem.Icon>
                </MenuItem>
                <MenuItem Header="Agregar Carpeta" Click="btn_AgregarCarp_Click" IsHitTestVisible="False" Name="menu1">
                    <MenuItem.Icon>
                        <Image Source="/PVMCDSMPMA;component/Imagenes/my%20folder.ico" />
                    </MenuItem.Icon>
                </MenuItem>
                <MenuItem Header="Agregar Diapositiva" Click="btn_agregar_Click" IsHitTestVisible="False" Name="menu2">
                    <MenuItem.Icon>
                        <Image Source="/PVMCDSMPMA;component/Imagenes/my%20pictures.ico" />
                    </MenuItem.Icon>
                </MenuItem>
            </MenuItem>
            <MenuItem Header="Ver">
                <MenuItem Header="Standard"/>
            </MenuItem>
            <MenuItem Header="Ayuda">
                <MenuItem  Header="Acerca de Pizarrón virtual.." Click="Acerca_de">
                    <MenuItem.Icon>
                        <Image Source="/PVMCDSMPMA;component/Imagenes/Help.ico" />
                    </MenuItem.Icon>
                </MenuItem>
            </MenuItem>
        </Menu>

        <!-- Master Container -->
        <Grid DataContext="{Binding Source={StaticResource Photos}}"
              Margin="0,0,0,0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="8*" />
                <ColumnDefinition Width="90*" />
                <ColumnDefinition Width="130*" />
                <ColumnDefinition Width="130*" />
                <ColumnDefinition Width="155*" />
                <ColumnDefinition Width="114*" />
                <ColumnDefinition Width="7*" />
                <ColumnDefinition Width="132*" />
                <ColumnDefinition Width="12*" />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="7*" />
                <RowDefinition Height="39*" />
                <RowDefinition Height="21*" />
                <RowDefinition Height="3*" />
                <RowDefinition Height="42*" />
                <RowDefinition Height="30*" />
                <RowDefinition Height="39*" />
                <RowDefinition Height="38*" />
```

```
            <RowDefinition Height="20*" />
            <RowDefinition Height="69*" />
            <RowDefinition Height="47*" />
            <RowDefinition Height="49*" />
            <RowDefinition Height="23*" />
            <RowDefinition Height="32*" />
            <RowDefinition Height="9*" />
            <RowDefinition Height="38*" />
            <RowDefinition Height="19*" />
            <RowDefinition Height="9*" />
        </Grid.RowDefinitions>

        <!-- Contenedor de imagenes -->
        <GroupBox Grid.Row="4" Grid.ColumnSpan="6" Margin="0,35,0,0" Grid.RowSpan="14">
            <ScrollViewer VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Disabled">
                <ListBox
                        IsSynchronizedWithCurrentItem="True"
                        Name="PhotosListBox"
                        Style="{StaticResource PhotoListBoxStyle}"
                        Margin="5"
                        SelectionMode="Extended"
                        ItemsSource="{Binding}"
                        SelectedIndex="0">
                    <ListBox.ContextMenu>
                        <ContextMenu>
                            <MenuItem Header="Quitar"/>
                        </ContextMenu>
                    </ListBox.ContextMenu>
                </ListBox>
            </ScrollViewer>
        </GroupBox>

        <GroupBox Grid.ColumnSpan="6" Grid.RowSpan="3">
            <StackPanel Grid.ColumnSpan="5" Grid.RowSpan="3">

            </StackPanel>
        </GroupBox>

        <DockPanel DockPanel.Dock="Bottom" Grid.Column="3" Height="39" VerticalAlignment="Bottom" Grid.Row="1">
            <Button DockPanel.Dock="Left" Content="Agregar carpeta"     Name="btn_AgregarCarp" Click="btn_AgregarCarp_Click" IsEn
abled="True"  Background="#FF65D4FF" BorderBrush="#00000000" Focusable="False" FontFamily="Verdana" FontSize="10" Height="20" Width="
91"  />
        </DockPanel>
        <DockPanel DockPanel.Dock="Bottom" Grid.Column="4" Height="39" VerticalAlignment="Bottom" Grid.Row="1">
            <Button DockPanel.Dock="Left" Content="Agregar/Quitar Video"       Name="btn_Video"       Click="btn_Video_Click"
   IsEnabled="True"  Background="#FF65D4FF" BorderBrush="#00000000" Focusable="False" FontFamily="Verdana" FontSize="10" Height="20"
Width="118" />
        </DockPanel>
        <DockPanel DockPanel.Dock="Bottom" Grid.Column="2" Height="39" VerticalAlignment="Bottom" Grid.Row="1">
            <Button DockPanel.Dock="Right" Content="Agregar Diapositiva" Name="btn_agregar"     Click="btn_agregar_Click"     IsE
nabled="True"  Background="#FF65D4FF" BorderBrush="#00000000" Focusable="False" FontFamily="Verdana" FontSize="10" Height="20" Width=
"105"/>
        </DockPanel>

        <DockPanel DockPanel.Dock="Bottom" Margin="1,0,0,0" Grid.Column="7" Grid.RowSpan="4" Grid.Row="1">
            <Label DockPanel.Dock="Top" Content="Descripción del proyecto " FontSize="15" Name="label7" FontWeight="Bold" FontSty
le="Normal" FontFamily="Adobe Arabic" />
            <Label DockPanel.Dock="Left" Height="30" FontSize="11" FontFamily="Palatino Linotype" FontWeight="Normal">Proyecto :<
/Label>
            <Label DockPanel.Dock="Left" Content="" Height="30" Name="NombreProy" FontSize="9" />
        </DockPanel>

        <DockPanel DockPanel.Dock="Bottom" Grid.Row="5" Grid.Column="7">
            <Label DockPanel.Dock="Left" Height="30" FontSize="11" FontFamily="Palatino Linotype" FontWeight="Normal">Tipo :</Lab
el>
            <Label DockPanel.Dock="Left" Content="" Height="30" Name="TipoProy"/>
        </DockPanel>

        <DockPanel DockPanel.Dock="Bottom" Margin="1,0,0,0" Grid.Row="6" Grid.Column="7">
            <Label DockPanel.Dock="Left" Height="30" FontSize="9" FontFamily="Palatino Linotype" FontWeight="Normal">Numero de di
apositivas :</Label>
            <Label DockPanel.Dock="Left" Content="" Height="30" Name="Numero"/>
        </DockPanel>

        <DockPanel DockPanel.Dock="Bottom" Margin="1,0,0,0" Grid.Row="7" Grid.Column="7">
            <Label DockPanel.Dock="Left" Height="30" FontSize="11" FontFamily="Palatino Linotype" FontWeight="Normal">Directorio
:</Label>
            <Label DockPanel.Dock="Left" Content="" Height="30" Name="Directorio"/>
        </DockPanel>

        <DockPanel DockPanel.Dock="Bottom" Margin="1,0,0,0" Grid.Row="9" Grid.Column="7">
            <Label DockPanel.Dock="Top" Content="Lista de videos" Name="label6" FontSize="10" FontWeight="Bold" FontFamily="Times
 New Roman" FontStretch="Expanded" />
            <Label DockPanel.Dock="Left" Height="30" FontSize="11" FontFamily="Palatino Linotype" FontWeight="Normal">Video 1 :</
Label>
            <Label DockPanel.Dock="Left" Content="" Height="30" FontSize="14" Name="label_video1"/>
        </DockPanel>

        <DockPanel DockPanel.Dock="Bottom" Grid.Row="10" Grid.Column="7">
            <Label DockPanel.Dock="Left" Height="30" FontSize="11" FontFamily="Palatino Linotype" FontWeight="Normal">Video 2 :</
Label>
            <Label DockPanel.Dock="Left" Content="" Height="30" FontSize="14" Name="label_video2"/>
```

```
            </DockPanel>

            <DockPanel DockPanel.Dock="Bottom" Margin="0,0,0,2" Grid.Row="11" Grid.Column="7">
                <Label DockPanel.Dock="Left" Height="30" FontSize="11" FontFamily="Palatino Linotype" FontWeight="Normal">Video 3 :</
Label>
                <Label DockPanel.Dock="Left" Content="" Height="30" FontSize="14" Name="label_video3"/>
            </DockPanel>

            <DockPanel DockPanel.Dock="Bottom" Grid.Row="13" Grid.Column="7">
                <Label DockPanel.Dock="Left" Height="30" FontSize="11" FontFamily="Palatino Linotype" FontWeight="Normal"></Label>
                <Label DockPanel.Dock="Left" Content="" Height="30" FontSize="12" Name="Estado" BorderBrush="#FF3E3E3E" />
            </DockPanel>

            <!--Zoom-->
            <DockPanel DockPanel.Dock="Bottom" Margin="1,0,0,0" Height="35" VerticalAlignment="Top" Grid.Row="16" Grid.Column="7" Gri
d.RowSpan="2">
                <Label DockPanel.Dock="Left">Zoom</Label>
                <Slider Name="ZoomSlider"
                        Orientation="Horizontal"
                        Minimum="80"
                        Maximum="400"
                        Value="160"
                        TickFrequency="80"
                        TickPlacement="BottomRight"
                        SmallChange="5"
                        LargeChange="20" />
            </DockPanel>

            <Button Name="Inciar" Content="Iniciar Presentación" Click="Inciar_Click" Focusable="False" Background="#FF3BCEFF" Margin
="1,0,0,0" Grid.Column="7" Grid.Row="15">
                <Button.BorderBrush>
                    <SolidColorBrush />
                </Button.BorderBrush>
            </Button>

            <StatusBarItem Grid.RowSpan="4"></StatusBarItem>
            <ProgressBar Name="progressBar1" Grid.ColumnSpan="6" Grid.Row="3" IsIndeterminate="True">
            </ProgressBar>

            <Border BorderBrush="#FF18759D" BorderThickness="5" Grid.Row="4" Height="33" Name="border1" VerticalAlignment="Top" Grid.
ColumnSpan="6" CornerRadius="8" Background="White"></Border>

            <!--
<Border BorderBrush="#FF18759D" BorderThickness="3" Grid.Column="7" Grid.RowSpan="16" Name="border2" CornerRadius="4" />-->
            <DockPanel DockPanel.Dock="Bottom" Margin="7,0,1,0" Height="33" VerticalAlignment="Top" Grid.Row="4" Grid.ColumnSpan="6">
                <Label DockPanel.Dock="Top" Content="Vista previa de diapositivas" Height="22" Name="label5" Width="199" FontFamily="
Adobe Hebrew" FontSize="12" />
            </DockPanel>
        </Grid>
    </DockPanel>
```

Terminando de realizar la codificación en xaml se definen las estructuras y codificación en código .cs, este código se comunica con la interfaz gráfica por medio de eventos o Binding, dentro del documento nombrado MainWindow.xaml.cs se realiza la programación para la visualización de las diapositivas y creación de proyectos. El documento se dividen en tres regiones: variables, funciones y eventos.

Este requerimiento será utilizado por el requerimiento número cinco tal y como se mencionó con anterioridad, dentro del mismo se implementa el botón para lanzar una ventana que servirá para inicializar la presentación.

A continuación se describe toda la codificación para visualizar las dispositivas del proyecto y lanzar las ventanas secundarias.

Cabe señalar que este documento utiliza una clase nombrada PhotoCollection que se encuentra codificada en el documento nombrado Data.cs.

Dentro del código .cs se utilizan las siguientes referencias:

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

using System.Windows.Forms;
using System.IO;
using System.Threading;
```

El requerimiento crear proyecto se define en su propio código xaml y .cs, como primer punto en codificación se define a continuación el código para la interfaz gráfica xaml:

```
<Window x:Class="PVMCDSMPMA.CrearProyecto.CrearProyecto"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Crear Proyecto" MinWidth="300" Width="600" MaxWidth="600" MinHeight="400" Height="400" MaxHeight="400" WindowStartupLo
cation="CenterScreen" Loaded="Window_Loaded"
        WindowStyle="None">
    <DockPanel>
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="83*" />
                <ColumnDefinition Width="356*" />
                <ColumnDefinition Width="139*" />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="266*" />
                <RowDefinition Height="34*" />
                <RowDefinition Height="32*" />
                <RowDefinition Height="29*" />
            </Grid.RowDefinitions>
            <Label Content="Nombre" Grid.Row="1" Height="23" HorizontalAlignment="Left" Margin="18,11,0,0" Name="Nombre" VerticalAlig
nment="Top" />
            <Label Content="Ubicación" Grid.Row="2" Height="23" HorizontalAlignment="Left" Margin="18,9,0,0" Name="Ubication" Vertica
lAlignment="Top" />
            <Button Content="Examinar" Grid.Column="2" Grid.Row="2" Height="23" HorizontalAlignment="Left" Margin="18,9,0,0" Name="Ex
aminar" Click="btnExaminar" VerticalAlignment="Top" Width="105" />
            <TextBox Grid.Column="1" Grid.Row="1" Height="23" HorizontalAlignment="Left" Name="textBoxNombre" VerticalAlignment="Top"
 Width="356" Margin="0,11,0,0" />
            <TextBox Grid.Column="1" Grid.Row="2" Height="23" HorizontalAlignment="Left" Name="textBoxUbication" VerticalAlignment="T
op" Width="356" Margin="0,9,0,0" />
            <Rectangle HorizontalAlignment="Left" Name="rectangle1" Stroke="Black" Width="355" Grid.Column="1" />
            <Button Content="Crear" Grid.Column="2" Grid.Row="3" Height="23" HorizontalAlignment="Left" Margin="18,6,0,0" Name="butto
n2" VerticalAlignment="Top" Width="105" Click="btnCrear" />
            <Label Content="Proyecto en blanco" Grid.Column="1" Height="37" HorizontalAlignment="Left" Margin="0,21,0,0" Name="label1
" VerticalAlignment="Top" Width="356" />
            <Label Content="Plantilla A" Height="37" HorizontalAlignment="Left" Margin="82,90,0,0" Name="PlantillaA" VerticalAlignmen
t="Top" Width="356" Grid.ColumnSpan="2" />
            <Label Content="Plantilla B" Height="37" HorizontalAlignment="Left" Margin="82,151,0,0" Name="PlantillaB" VerticalAlignme
nt="Top" Width="356" Grid.ColumnSpan="2" />
            <Button Content="Seleccionar" Grid.Column="1" Height="23" HorizontalAlignment="Left" Margin="208,22,0,0" Name="button1" V
erticalAlignment="Top" Width="75" Click="button1_Click" />
            <Button Content="Seleccionar" Height="23" HorizontalAlignment="Left" Margin="208,90,0,0" Name="button3" VerticalAlignment
="Top" Width="75" Grid.Column="1" IsEnabled="False" />
            <Button Content="Seleccionar" Height="23" HorizontalAlignment="Left" Margin="208,151,0,0" Name="button4" VerticalAlignmen
t="Top" Width="75" Grid.Column="1" IsEnabled="False" />
            <Button Background="Red" BorderBrush="#00000000" Content="X" FontFamily="Californian FB" FontSize="9" FontWeight="Normal"
 Foreground="White" Height="19" HorizontalAlignment="Left" Margin="104,0,0,0" Name="button5" VerticalAlignment="Top" Width="35" Grid.
Column="2" Click="button5_Click_1" />
        </Grid>
    </DockPanel>
</Window>
```

Como se puede apreciar en el código anterior no se necesita de mucha codificación para comunicar la interfaz con el código .cs, cabe señalar que no se utiliza Binding para la comunicación, se utilizan eventos de control.

Terminando de definir las funcionalidades que ligan a la creación de proyectos se encuentra la funcionalidad de agregar videos, este necesita de una venta por aparte y se nombra ListaVideo donde se encuentran los documentos xaml y .cs, primeramente se define el código xaml:

```
<Window x:Class="PVMCDSMPMA.ListaVideo"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="ListaVideo" Height="499" Width="831"
        WindowStyle="None"
        ResizeMode="NoResize"
        WindowStartupLocation="CenterScreen" Foreground="Black" Loaded="Window_Loaded">
    <Grid Background="#FFE5E5E5">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="45*" />
            <ColumnDefinition Width="12*" />
            <ColumnDefinition Width="69*" />
            <ColumnDefinition Width="27*" />
            <ColumnDefinition Width="81*" />
            <ColumnDefinition Width="68*" />
            <ColumnDefinition Width="79*" />
            <ColumnDefinition Width="40*" />
            <ColumnDefinition Width="77*" />
            <ColumnDefinition Width="86*" />
            <ColumnDefinition Width="81*" />
            <ColumnDefinition Width="25*" />
            <ColumnDefinition Width="81*" />
            <ColumnDefinition Width="38*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="19*" />
            <RowDefinition Height="25*" />
            <RowDefinition Height="69*" />
            <RowDefinition Height="27*" />
            <RowDefinition Height="19*" />
            <RowDefinition Height="147*" />
            <RowDefinition Height="25*" />
            <RowDefinition Height="26*" />
            <RowDefinition Height="68*" />
            <RowDefinition Height="35*" />
        </Grid.RowDefinitions>
        <Button Content="Agregar" Margin="1,0,0,0" Name="button_agregar1" Background="#FFAAFFD4" Click="button_agregar1_Click" Grid.R
ow="7" Focusable="False" Grid.Column="1" Grid.ColumnSpan="2" />
        <Image Name="image1" Stretch="Fill" Margin="1,0,0,0" Source="/PVMCDSMPMA;component/bin/Debug/Imagenes/video.jpg" Grid.Row="5"
 Grid.Column="1" Grid.ColumnSpan="4" />
        <Button Content="Agregar" Name="button_agregar2" Background="#FFAAFFD4" Click="button_agregar2_Click" Grid.Row="7" Focusable=
"False" Grid.Column="6" />
        <Button Content="Agregar" Name="button_agregar3" Background="#FFAAFFD4" Click="button_agregar3_Click" Grid.Row="7" Focusable=
"False" Grid.Column="10" Margin="1,0,0,0" />
        <Label Content="" Margin="1,0" Name="Video1" Grid.Row="3" Grid.Column="1" Grid.ColumnSpan="4" />
        <Label Content="" Name="Video2" Grid.Row="3" Grid.Column="6" Grid.ColumnSpan="3" />
        <Label Content="" HorizontalAlignment="Left" Name="Video3" Width="187" Grid.Row="3" Grid.ColumnSpan="3" Grid.Column="10" />
        <Label Content="Lista de videos" Height="28" HorizontalAlignment="Left" Margin="15,12,0,0" Name="label4" VerticalAlignment="T
op" Width="147" FontFamily="Simplified Arabic Fixed" FontWeight="Bold" Grid.Column="6" Grid.ColumnSpan="3" Grid.RowSpan="2" />
        <Button Background="Red" BorderBrush="#00000000" Content="X" FontFamily="Californian FB" FontSize="9" FontWeight="Normal" For
eground="White" HorizontalAlignment="Left" Name="button5" Width="37" Click="button5_Click" Grid.Column="13" />
        <Image Name="image2"  Stretch="Fill" Source="/PVMCDSMPMA;component/bin/Debug/Imagenes/video.jpg" Grid.Row="5" Grid.Column="6"
 Grid.ColumnSpan="3" />
        <Image Name="image3"  Stretch="Fill" Source="/PVMCDSMPMA;component/bin/Debug/Imagenes/video.jpg" Grid.Row="5" Grid.ColumnSpan
="3" Grid.Column="10" />
        <Label Content="Video 1" Height="28" HorizontalAlignment="Left" Margin="54,36,0,0" Name="label5" VerticalAlignment="Top" Widt
h="67" Grid.Row="2" Grid.Column="2" Grid.ColumnSpan="3" />
        <Label Content="Video 2" Height="28" HorizontalAlignment="Right" Margin="0,36,55,0" Name="label1" VerticalAlignment="Top" Wid
th="67" Grid.Row="2" Grid.Column="6" Grid.ColumnSpan="3" />
        <Label Content="Video 3" Height="28" HorizontalAlignment="Left" Margin="64,36,0,0" Name="label2" VerticalAlignment="Top" Widt
h="67" Grid.Row="2" Grid.ColumnSpan="3" Grid.Column="10" />
        <Border BorderBrush="Coral" BorderThickness="6" CornerRadius="8" Name="border3" Grid.Row="2" Grid.ColumnSpan="14" Grid.RowSpa
n="7"></Border>
        <Label Content="Estado:" Grid.Row="9" Name="label3" Grid.ColumnSpan="2" />
        <Label Content="" Name="Estado" Margin="2,0,0,0" Grid.Row="9" Grid.Column="2" Grid.ColumnSpan="3" />
        <Button Background="#FFAAFFD4" Content="Quitar" Margin="0,0,1,0" Name="buttonQuitar1" Grid.Row="7" Click="buttonQuitar1_Click
" Focusable="False" Grid.Column="4" />
        <Button Background="#FFAAFFD4" Content="Quitar" Name="buttonQuitar2" Grid.Row="7" Click="buttonQuitar2_Click" Focusable="Fals
e" Grid.Column="8" Margin="0,0,1,0" />
```

```
        <Button Background="#FFAAFFD4" Content="Quitar" Name="buttonQuitar3" Grid.Row="7" Click="buttonQuitar3_Click" Focusable="Fals
e" Grid.Column="12" />
    </Grid>
</Window>
```

Una vez que se tiene definido el código de diseño se realiza la comunicación de controles, en el código .cs se define la lógica para la creación del archivo de control para los videos y se enlaza con el nombre del proyecto que se encuentra abierto definiendo la estructura por medio de cadenas concatenadas separadas por una barra, a continuación se escribe todo el código para agregar videos al proyecto:

```
public partial class ListaVideo : Window
{
    String direccionVideo = null;
    String Nfile = null;
    String NombreProyecto = null;
    String video1 = null;
    String video2 = null;
    String video3 = null;

    public ListaVideo()
    {
        InitializeComponent();
    }

    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        leervideos();
        Estado.Content = "Listo";
    }

    private void leervideos()
    {
        String texto = null;
        System.IO.StreamReader fi = new System.IO.StreamReader(Nfile);
        texto = fi.ReadToEnd();
        fi.Close();
        String cad1 = texto;
        String[] words = cad1.Split('|');
        try
        {
            NombreProyecto = words[0];
            video1 = words[1];
            video2 = words[2];
            video3 = words[3];
            actualizar();
        }
        catch (Exception ex)
        {
            System.Windows.Forms.MessageBox.Show("Archivo dañado");
        }
    }
```

6.5.4 Prueba

6.5.4.1 Pruebas por los desarrolladores

Se realizan pruebas para corroborar el completo funcionamiento de los requerimientos, se prevé que el usuario pueda cometer errores al momento de crear el proyecto y se cuida la función de agregar diapositivas, abrir y cerrar.

Toda la codificación descrita en este requerimiento es completamente funcional y se encuentra lista para ser utilizada por el requerimiento número cinco.

Las pruebas realizadas por los desarrolladores se basan en la ejecución de los controles y realizar todos los flujos posibles que existan, el flujo normal se deja en segundo plano puesto el mismo fue hecho para no fallar, las pruebas van de presionar diferentes teclas de función y agregar archivos de manera forzada.

6.5.4.2 Pruebas de Aceptación de usuarios

Para la ejecución de pruebas de aceptación se invito a tres estudiantes del Instituto Tecnológico de Tuxtla Gutiérrez para utilizar el programa, los estudiantes no tuvieron problema alguno en el manejo de la creación de proyectos.

6.6 Incremento 5: Mejoramiento de la interfaz donde se visualizan las diapositivas y videos e integración con la funcionalidad de creación de proyectos

6.6.1 Análisis

El mejoramiento de esta interfaz es de mucha importancia debido a que será donde se visualizarán las diapositivas y videos, por tal razón el diseño es simple incorporando el mínimo de controles en la parte inferior y superior, se necesita de controles para visualizar el número de diapositiva y video actual, además de un notificador para visualizar mensajes referente al estado del sensor.

En la parte superior se incorpora un notificador en forma de botón para informar al usuario cuando es detectado por el sensor Kinect, esta funcionalidad es completamente nueva, la interfaz subsecuente del visualizador de diapositivas cambia completamente, haciendo que el uso del sistema sea más fácil e intuitivo.

Los iconos o cursores de la mano derecha permanecen en el programa, no se opta por quitarlos debido a que el usuario puede confundirse al momento de manejarlo, por lo tanto se decide dejarlos como apuntadores o señaladores de la posición del usuario, el único que no se encuentra en este requerimiento es el icono de la cabeza.

En la parte superior derecha se incorporan dos botones esenciales, el primero se denomina "Cancelar función" , este botón se utiliza para cancelar funciones que pueda estar realizando el usuario al momento de estar exponiendo, por otra parte el botón de color rojo cumple con la finalidad de cerrar la ventana, este botón puede ser accionado con el cursor del mouse.

Modo presentador es nombrada la ventana donde se visualizaran todos los eventos de la exposición, por tal motivo el documento xaml y .cs son nombrados ModoPresentador, se ubicara en una carpeta dentro la solución nombrada Controlkinect para tener un orden de las funcionalidades y requerimientos pizarrón (ver figura 6.17).

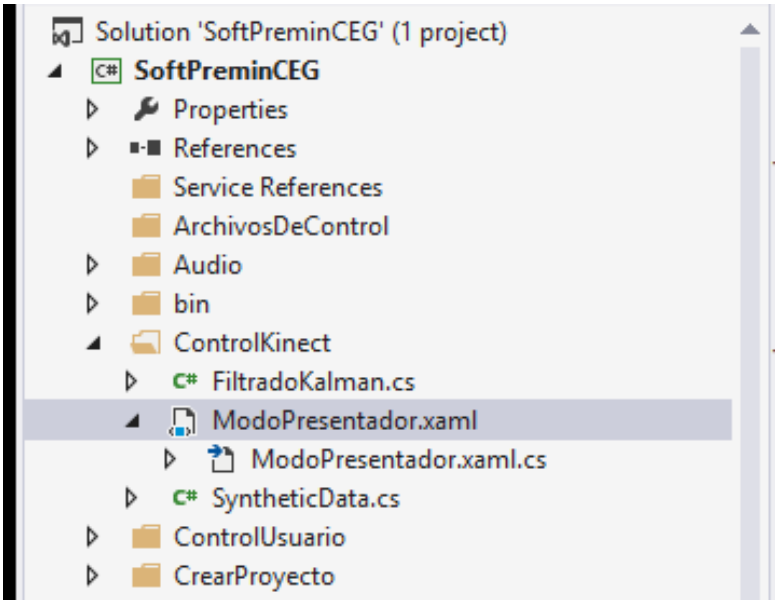

Figura 6.17: Directorio de carpetas solución Sistema de superficie audiovisual.

6.6.2 Diseño

Para llevar acabo el diseño del nuevo incremento se creó un diagrama de secuencias para plasmar aquí las interacciones que el usuario tendrá una vez que lance dentro del programa principal el modo presentador.

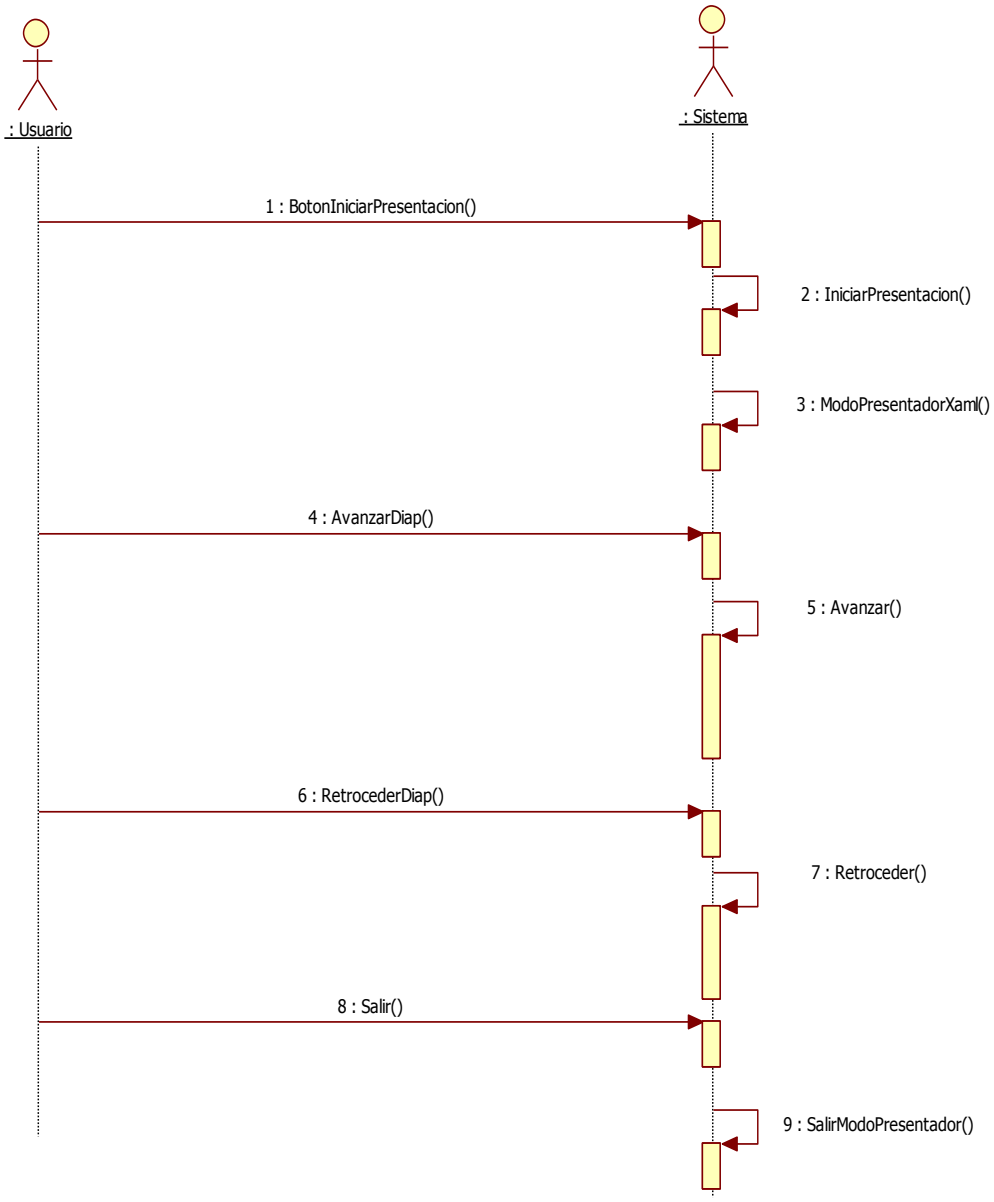

Figura 6.18: Diagrama de secuencias para el modo presentador.

Modo presentador requiere de un diseño sencillo donde se pueda visualizar de forma legible las diapositivas que se agregan en el proyecto personalizado por el usuario, por tal razón el diseño contiene visible seis controles (ver figura 6.19).

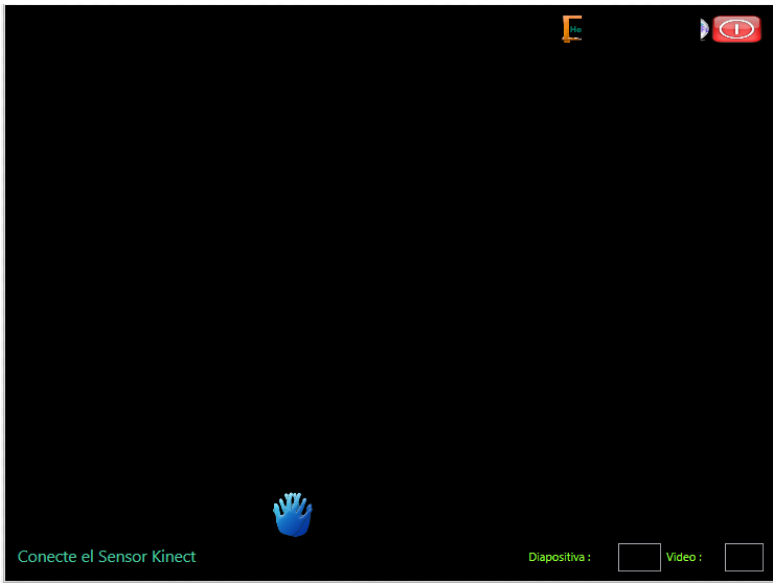

Figura 6.19: Pantalla modo presentador, sin diapositiva.

El modo presentador ocupa automáticamente toda la pantalla y como color de fondo se opta por negro debido a que solo es necesario tener énfasis a las diapositivas del usuario.

Los iconos o cursores de la mano derecha e izquierda cambian completamente a más pequeños con color saturado (Ver figura 6.20).

Figura 6.20: Pantalla modo presentador, con diapositiva

6.6.3 Código

Este requerimiento utiliza diversos controles utilizados para multimedia como lo son: Image, MediaElement e InkCanvas, dentro de la codificación xaml tenemos el siguiente código:

```
<Window x:Class="PVMCDSMPMA.ModoPresentador"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="ModoPresentador" Width="800" Height="600"  WindowStartupLocation="CenterOwner" WindowState="Maximized" Loaded="Window_
Loaded"
        WindowStyle="None"
        ResizeMode="NoResize"
        ShowInTaskbar="False"
        AllowsTransparency="True"
        Background="Transparent" BorderThickness="1">
        <DockPanel Name="PresPanel_1" Margin="0,0,0,0"  Background="Black">
        <Grid  x:Name="LayoutRoot"  Margin="10">
            <Grid.RowDefinitions>
                <RowDefinition Height="12*" />
                <RowDefinition Height="18*" />
                <RowDefinition Height="0*" />
                <RowDefinition Height="0*" />
                <RowDefinition Height="31*" />
                <RowDefinition Height="48*"/>
                <RowDefinition Height="10*"/>
                <RowDefinition Height="40*" />
                <RowDefinition Height="8*" />
                <RowDefinition Height="10*"/>
                <RowDefinition Height="15*"/>
                <RowDefinition Height="13*" />
                <RowDefinition Height="0*" />
                <RowDefinition Height="24*" />
                <RowDefinition Height="11*"/>
                <RowDefinition Height="40*"/>
                <RowDefinition Height="13*" />
                <RowDefinition Height="0*" />
                <RowDefinition Height="12*" />
                <RowDefinition Height="53*"/>
                <RowDefinition Height="17*"/>
                <RowDefinition Height="57*" />
                <RowDefinition Height="56*"/>
                <RowDefinition Height="61*" />
                <RowDefinition Height="0*" />
                <RowDefinition Height="29*" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="17*" />
                <ColumnDefinition Width="3*" />
                <ColumnDefinition Width="6*" />
                <ColumnDefinition Width="7*" />
                <ColumnDefinition Width="18*" />
                <ColumnDefinition Width="66*" />
                <ColumnDefinition Width="213*"/>
                <ColumnDefinition Width="57*" />
                <ColumnDefinition Width="41*" />
                <ColumnDefinition Width="41*" />
                <ColumnDefinition Width="42*" />
                <ColumnDefinition Width="18*" />
                <ColumnDefinition Width="39*" />
                <ColumnDefinition Width="59*" />
                <ColumnDefinition Width="16*"/>
                <ColumnDefinition Width="29*" />
                <ColumnDefinition Width="0*" />
                <ColumnDefinition Width="51*" />
                <ColumnDefinition Width="14*" />
                <ColumnDefinition Width="41*" />
            </Grid.ColumnDefinitions>
            <!-- Image main -->
            <Image x:Name="Image_1" Grid.RowSpan="26" Grid.ColumnSpan="20" Visibility="Collapsed" />
            <!-- Video main -->
            <MediaElement  Name="mediaElement1" LoadedBehavior="Manual" Grid.RowSpan="26" Grid.ColumnSpan="20" Visibility="Collapsed"
/>
            <!-- Pizarron main -->
            <InkCanvas Name="inkCanvas1" Background="Transparent" Grid.RowSpan="26" Grid.ColumnSpan="20" Visibility="Visible" Editing
Mode="GestureOnly" />
            <!-- Cuadro de herramientas -->
            <TextBox Name="textBox1" Grid.Column="14" Grid.Row="24" Background="Transparent" Foreground="White" HorizontalAlignment="
Right" Width="45" FontSize="20" Grid.ColumnSpan="2" Grid.RowSpan="2" />
            <Label Content="Diapositiva :" Grid.Row="24" Grid.Column="12" Grid.ColumnSpan="3" Foreground="#FF97FF3E" Grid.RowSpan="2"
 />
            <Label Content="Video :" Grid.Row="25" Grid.Column="17" Foreground="#FF97FF3E" />
```

```
            <TextBox Name="textBox2" Grid.Column="19" Grid.Row="25" Background="Transparent" Foreground="White" HorizontalAlignment="
Right" Width="40" FontSize="20" />
            <Button Content="Color" Name="buttonColorAzul" Foreground="#FF0C00FF"  FontWeight="Bold" Focusable="False" DataContext="{
Binding}" FontSize="9" Grid.Row="5" Background="#FF0022FF" MouseEnter="buttonPaletaColorAzul_MouseEnter" Grid.Column="19" Visibility=
"Collapsed"/>
            <Button Content="Color" Name="buttonColorRojo" Foreground="#FF3B00FF"  FontWeight="Bold" Focusable="False" DataContext="{
Binding}" FontSize="9" Grid.Row="7" Background="#FFFF1111" Grid.RowSpan="2" MouseEnter="buttonPaletaColorRojo_MouseEnter" Grid.Column
="19" Margin="1,0,0,0" Visibility="Collapsed"/>
            <Button Content="Color" Name="buttonColorRosa" Foreground="#FF3B00FF"  FontWeight="Bold" Focusable="False" DataContext="{
Binding}" FontSize="9" Grid.Row="10" Background="#FFF411FF" Grid.RowSpan="4" MouseEnter="buttonPaletaColorRosa_MouseEnter" Grid.Colum
n="19" Margin="1,0,0,0" Visibility="Collapsed"/>
            <Button Content="Color" Name="buttonColorVerde" Foreground="#FF3B00FF"  FontWeight="Bold" Focusable="False" DataContext="
{Binding}" FontSize="9" Grid.Row="15" Background="#FF11FF3C" Grid.RowSpan="2" MouseEnter="buttonPaletaColorVerde_MouseEnter" Grid.Col
umn="19" Margin="1,0,0,0" Visibility="Collapsed"/>
            <Button Content="Color" Name="buttonColorBlanco" Foreground="#FF3B00FF"  FontWeight="Bold" Focusable="False" DataContext=
"{Binding}" FontSize="9" Grid.Row="19" Background="White" MouseEnter="buttonPaletaColorBlanco_MouseEnter" Grid.Column="19" Margin="1,
0,0,0" Visibility="Collapsed"/>

            <Image Name="image2" Stretch="Fill" Grid.Column="17" Source="/PVMCDSMPMA;component/Imagenes/Continuar.png" Grid.RowSpan="
2" />
            <Button Content="Cancelar Funcion" Name="buttonActivClick" Grid.Column="17" Foreground="#FF3B00FF" MouseEnter="buttonActi
vClick_MouseEnter" FontWeight="Bold" Focusable="False" DataContext="{Binding}" FontSize="9" BorderBrush="#CD000000" OpacityMask="#940
00000" Grid.RowSpan="2">
                <Button.Background>
                    <SolidColorBrush />
                </Button.Background>
            </Button>
            <Image Name="image1" Stretch="Fill" Grid.Column="13" Source="/PVMCDSMPMA;component/Imagenes/pizarron.png" Grid.RowSpan="2
" Grid.ColumnSpan="2" />
            <Button Content="Herramientas" Name="buttonHerramientas" Grid.Column="13" Background="Transparent" Foreground="#FF00FFD5"
 MouseEnter="buttonHerramientas_MouseEnter" FontWeight="Bold" Focusable="False" FontSize="9" BorderThickness="0" OpacityMask="#7B0000
00" Grid.RowSpan="2" Grid.ColumnSpan="2" />
            <Image Name="image5" Stretch="Fill" Grid.Column="18" Source="/PVMCDSMPMA;component/Imagenes/cerrar.png" Grid.RowSpan="2"
Grid.ColumnSpan="2" />
            <Button Click="Button_Click_Salir"
                    Background="Transparent"
                    Grid.Column="18"
                    MouseEnter="Btn_cerrar_MouseEnter"
                    MouseLeave="Btn_cerrar_MouseLeave"
                    Name="Btn_cerrar"
                    BorderBrush="Transparent"
                    BorderThickness="0"
                    Padding="-4" OpacityMask="#7B000000" Grid.RowSpan="2" Grid.ColumnSpan="2">
                <StackPanel Visibility="Collapsed" Name="stack_cerrar" Height="4" Width="6">
                </StackPanel>
            </Button>
            <Image Name="PintarImage" Stretch="Fill" Grid.Column="12" Source="/PVMCDSMPMA;component/bin/Debug/Imagenes/lapiz.png" Vis
ibility="Collapsed" Grid.RowSpan="2" />
            <Image Name="MarcatextoImage" Stretch="Fill" Grid.Column="12" Source="/PVMCDSMPMA;component/Imagenes/plumon.png" Visibili
ty="Collapsed" Grid.RowSpan="2" />
            <Image Name="LineaImage" Stretch="Fill" Grid.Column="12" Visibility="Collapsed" Source="/PVMCDSMPMA;component/Imagenes/li
nea.png" Grid.RowSpan="2" />
            <Image Name="CuadroImage" Stretch="Fill" Grid.Column="12" Visibility="Collapsed" Source="/PVMCDSMPMA;component/Imagenes/c
uadro.png" Grid.RowSpan="2" />
            <Image Name="VideoImage" Stretch="Fill" Grid.Column="12" Visibility="Collapsed" Source="/PVMCDSMPMA;component/Imagenes/vi
deo.png" Grid.RowSpan="2" />
            <Image Name="EncerrarImage" Stretch="Fill" Grid.Column="12" Visibility="Collapsed" Source="/PVMCDSMPMA;component/Imagenes
/encerrar.png" Grid.RowSpan="2" />
            <Image Name="TrianguloImage" Stretch="Fill" Grid.Column="12" Visibility="Collapsed" Source="/PVMCDSMPMA;component/Imagene
s/triangulo.png" Grid.RowSpan="2" />
            <Image Name="CirculoImage" Stretch="Fill" Grid.Column="12" Visibility="Collapsed" Source="/PVMCDSMPMA;component/Imagenes/
circulo_azul.png" Grid.RowSpan="2" />
            <Image Name="botonRojo" Stretch="Fill"  Visibility="Collapsed" Grid.ColumnSpan="3" Source="/PVMCDSMPMA;component/Imagenes
/boton-rojo.png" Grid.RowSpan="2" />
            <Image Name="botonVerde" Stretch="Fill" Source="/PVMCDSMPMA;component/Imagenes/Boton_verde.png" Visibility="Collapsed" Gr
id.ColumnSpan="3" Grid.RowSpan="2" />

            <Image Name="reprod" Stretch="Fill" Grid.Column="8" Visibility="Collapsed" Source="/PVMCDSMPMA;component/Imagenes/reprodu
cir.png" OpacityMask="#64000000" Grid.RowSpan="2" />
            <Button Name="reproducir" Grid.Column="8" Background="#00000000" Foreground="#FFB1F279" BorderBrush="#00000000" Visibilit
y="Collapsed" FontSize="10"  Focusable="False" Click="Reproducir" OpacityMask="#7B000000" MouseEnter="reproducir_MouseEnter" Grid.Row
Span="2" />
            <Image Name="paus" Stretch="Fill" Grid.Column="9" Visibility="Collapsed" Source="/PVMCDSMPMA;component/Imagenes/pausa.png
" OpacityMask="#64000000" Margin="0,6,2,0" Grid.RowSpan="2" />
            <Button Content="" Name="pausa" Grid.Column="9"  Background="#00000000" Foreground="#FFB1F279" BorderBrush="#00000000" Vi
sibility="Collapsed" FontSize="15" Focusable="False" Click="Pausa" OpacityMask="#7B000000" MouseEnter="pausa_MouseEnter" Margin="0,0,
2,0" Grid.RowSpan="2" />
            <Image Name="sto" Stretch="Fill" Grid.Column="10" Visibility="Collapsed" Source="/PVMCDSMPMA;component/Imagenes/stop.png"
 OpacityMask="#64000000" Grid.RowSpan="2" />
            <Button Content="" Name="stop" Grid.Column="10" Background="#00000000" Foreground="#FFB1F279" BorderBrush="#00000000" Vis
ibility="Collapsed" FontSize="15" Focusable="False" Click="Stop" OpacityMask="#7B000000" MouseEnter="stop_MouseEnter" Grid.RowSpan="2
" />

            <TextBox Name="textBoxMensajes" Background="Transparent" Foreground="#FF4FCEA5" FontSize="17" Text="Conecte el Sensor Kin
ect" Grid.Row="24" IsReadOnly="True" Focusable="False" BorderBrush="#00000000" Grid.ColumnSpan="7" Grid.RowSpan="2" />

            <!-- Control kinect -->
            <Canvas x:Name="GameBoardElement" Grid.RowSpan="26" Grid.ColumnSpan="20">
                <Image x:Name="HandIzqCursorElement"
                       Width="57"
                       Height="69"
                       RenderTransformOrigin="0.5,0.5"
```

```
                    Canvas.Left="260"
                    Canvas.Top="485"
                    Source="/PVMCDSMPMA;component/Imagenes/ManoIzq.png">
                <Image.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform x:Name="HandIzqCursorScale" ScaleX="1"/>
                    </TransformGroup>
                </Image.RenderTransform>
            </Image>
            <Image x:Name="HandDerCursorElement"
                    Width="55" Height="65" RenderTransformOrigin="0.5,0.5" Canvas.Left="262" Canvas.Top="486" Source="/PVMCDSMPM
A;component/Imagenes/ManoDer.png">
                <Image.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform x:Name="HandDerCursorScale" ScaleX="1"/>
                    </TransformGroup>
                </Image.RenderTransform>
            </Image>

            <Image x:Name="HeadCursorElement"
                    Width="25" Height="25" RenderTransformOrigin="0.5,0.5" Canvas.Left="314" Canvas.Top="697" >
                <Image.RenderTransform>
                    <TransformGroup>
                        <ScaleTransform x:Name="HeadCursorScale" ScaleX="1"/>
                    </TransformGroup>
                </Image.RenderTransform>
            </Image>
            <Image Canvas.Left="0" Canvas.Top="50" Name="p1" Stretch="Fill"   Visibility="Collapsed" Height="50" Width="50" Sourc
e="/PVMCDSMPMA;component/Imagenes/mira.png" />
            <Image Canvas.Left="0" Canvas.Top="50" Name="p2" Stretch="Fill"   Visibility="Collapsed" Height="50" Width="50" Sourc
e="/PVMCDSMPMA;component/Imagenes/mira.png" />
            <Image Canvas.Left="0" Canvas.Top="50" Name="p3" Stretch="Fill"   Visibility="Collapsed" Height="50" Width="50" Sourc
e="/PVMCDSMPMA;component/Imagenes/mira.png" />
        </Canvas>
    </Grid>
</DockPanel>
</Window>
```

A continuación se visualiza el código en .cs, debido al número elevado de líneas de código y para evitar ser repetitivo solo se visualizaran las nuevas funciones, el código completo se escribirá en el último requerimiento para tener un panorama más claro.

Funciones para avanzar o retroceder diapositivas:

```
private void Avanzar()
        {
            try
            {
                this.inkCanvas1.Strokes.Clear();
                inkCanvas1.Children.Clear();
                if (numero1.Equals(Convert.ToInt16(NumArchs))) //si llego al limite de archivo , reiniciar conteo
                {
                    textBox1.Text = Convert.ToString(numero1);
                    FDiapo();
                }
                else
                {
                    numero1++;
                    textBox1.Text = Convert.ToString(numero1);
                    String URL = Convert.ToString(DirArchs + "\\Diap_PV_ " + numero1 + ".jpg");
                    BitmapImage imagen = new BitmapImage(new Uri(URL));
                    Image_1.Source = imagen;
                    Storyboard AnimacionOpaca = (Storyboard)FindResource("AnimacionOpaca");
                    AnimacionOpaca.Begin();
                }
            }
            catch
            {
                textBox1.Text = Convert.ToString(" ");
                System.Windows.Forms.MessageBox.Show("No se han cargado diapositivas en el proyecto");
            }
        }

        private void Retroceder()
        {
            try
            {
                this.inkCanvas1.Strokes.Clear();
                inkCanvas1.Children.Clear();
                if (numero1 > 1)
```

```
        {
            numero1--;
            textBox1.Text = Convert.ToString(numero1);
            String URL = Convert.ToString(DirArchs + "\\Diap_PV_ " + numero1 + ".jpg");
            BitmapImage imagen = new BitmapImage(new Uri(URL));
            Image_1.Source = imagen;
            Storyboard AnimacionOpaca = (Storyboard)FindResource("AnimacionOpaca");
            AnimacionOpaca.Begin();
            imactual = imagen;
        }
    }
    catch
    {
        textBox1.Text = Convert.ToString(" ");
        System.Windows.Forms.MessageBox.Show("No se han cargado diapositivas en el proyecto");
    }
}
```

La integración del incremento de creación de proyectos queda de la siguiente manera, donde la figura 6.21 se observa la funcionalidad de pasar el directorio actual del proyecto al ModoPresentador.

MainWindow.xaml.cs

```
85          Usuario = usuario;
86      }
87
88      public void actualizarVisualizador()
89      {
90          Photos.Path = DirMultimedia;
91      }
92
93      private void IniPresn()
94      {
95          Thread.Sleep(1000);
96          string[] filePaths = Directory.GetFiles(DirMultimedia, "*.jpg");
97          ModoPresentador MP = new ModoPresentador();
98          MP.setDir(DirMultimedia, NumeroArchivos.ToString(), Video1, Video2, Video3);
99          MP.ShowDialog();
100     }
101
102     public void leerDireccionInicial(String dirOfApp)
103     {
104         DirMultimedia = dirOfApp;
105     }
106
107     public void actualizarValoresCampos()
```

Figura 6.21 Funcionalidad de pasar el directorio del proyecto creado al ModoPresentador.

Dentro del modo presentador tenemos las siguientes líneas de código para completar el puente de comunicación con la funcionalidad de crear proyectos. (Ver figura 6.22).

```
public void setDir(String mainDirArchs, String NumerArchivs, String v1, String v2, String v3)
{
    DirArchs = mainDirArchs;
    NumArchs = NumerArchivs;
    Video1 = v1;
    Video2 = v2;
    Video3 = v3;
}
```

Figura 6.22: Funcionalidad de Leer los parámetros que envía la funcionalidad crear proyectos.

De este modo el modo presentador utiliza el directorio que se pasa como parámetro entre otros para tener el control de la presentación.

6.6.4 Prueba

6.6.4.1 Pruebas por los desarrolladores

Las pruebas realizadas se centran en verificar la correcta lectura del directorio para poder visualizar las diapositivas o videos. El sistema no tiene problemas en leer el directorio donde se encuentra el proyecto creado por el usuario, durante las pruebas se cargaron alrededor de cien diapositivas, se observo un incremento de tamaño en el directorio debido a la copia que se realiza de las imágenes, partiendo de este detalle como trabajo futuro se podría realizar una compresión de archivos al momento de guardar el proyecto, seria una mejora importante para ahorrar espacio en la unidad de almacenamiento

La codificación cumple con los requerimientos impuestos para la construcción del sistema.

6.6.4.2 Pruebas de Aceptación de usuarios

Se invito a tres usuarios al azar para probar el programa, con anticipación se les informo que tenían que crear un proyecto, agregar diapositivas y videos, todos los usuarios no tuvieron problemas al manejar el programa, pero si hubieron complicaciones o dudas al momento de decidir donde guardar el proyecto, el detalle surge en el requerimiento del mismo, todos los proyectos se crearan y guardaran en el disco del usuario, especialmente en la unidad C: , todos los usuarios intentaban abrir el proyecto en donde se creo, pero la implementación de crear un ejecutable para arrancar el proyecto no se previo, por lo consiguiente esa seria una mejor para una versión a futuro.

6.7 Incremento 6: mejoramiento de las funciones para interpretar gestos del usuario

6.7.1 Análisis

En este incremento se plasmará la implementación de algoritmos básicos para el control de interpretación de gestos, en incrementos anteriores solo utiliza una condición simple para poder cambiar de una diapositiva a otra. El algoritmo evalúa siete coordenadas y tres ejes (X, Y y Z), el desarrollo no requiere de una interfaz gráfica por esa razón se omite el diseño y se incorpora el diagrama de flujo.

6.7.2 Diseño

La lógica para evaluación de gesto avance y retroceso se basan en la detección de choques o intersecciones de coordenadas específicas entre los Joints de las manos y cabeza.

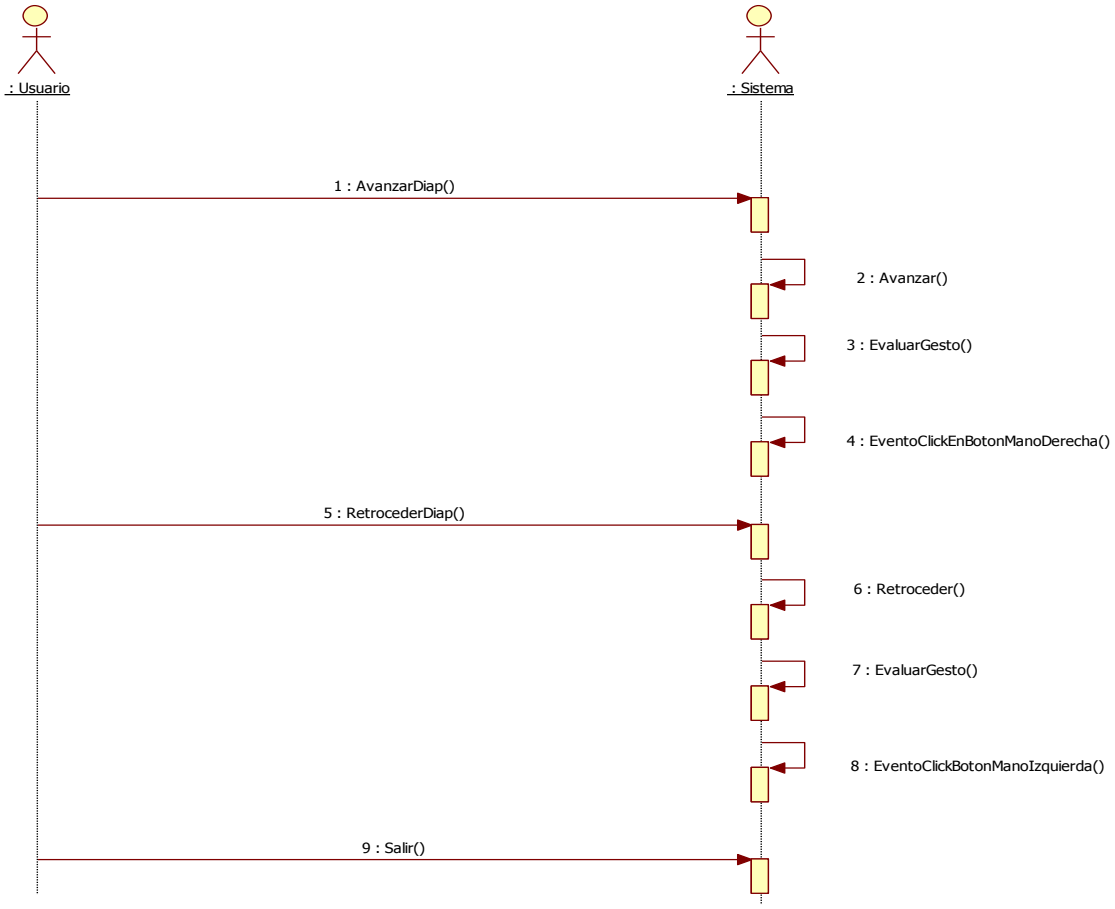

Figura 6.23 Diagrama de secuencias para mejorar gestos del usuario.

Las variables para la evaluación son:

ymDer = eje Y de la mano derecha.

Xh = eje X de cabeza, zManDerecha

yh = eje Y de la cabeza,.

xmDer = eje X de mano derecha.

De la misma forma se tienen las variables para la mano izquierda en los ejes X Y y Z. en base a estas se tienen las evaluaciones correspondientes a cada gesto para de la mano derecha e izquierda. (Ver figura 6.24).

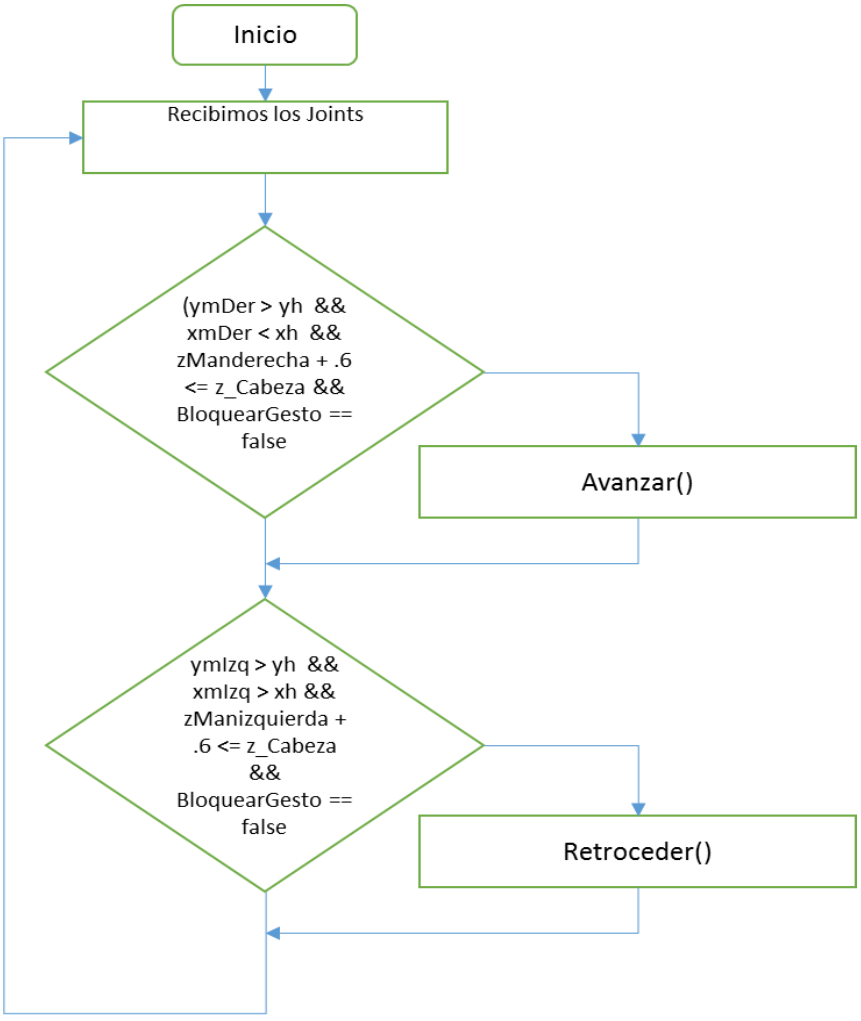

Figura 6.24 Diagrama de flujo evaluación de gestos, avance y retroceso de diapositiva.

6.7.3 Código

Como se pudo observar en el algoritmo anterior plasmado en diagrama de flujo a continuación se definen las funciones principales para la interpretación de gestos:

```
private void EvaluarGesto(int xmIzq, int ymIzq, int xmDer, int ymDer, int xh, int yh, int xhomIzq, int yhomIzq, int xhomDer, int yhom
Der, int xhomCen, int yhomCen, int xcadcen, int ycadCen, float zManderecha, float zManizquierda, float z_Cabeza)
       {
          sensibilidadTouchMano = .3;

           if (Estado == 1) //Aplicacion no finlizada
           {
               MoverMotor(yh); //Movemos el motor en el vector y, se toma como referencia la altura de la cabeza respecto al margen s
uperior del programa

               float centro, punto;
               centro = (yhomCen - ycadCen) / 2;
               punto = ycadCen + centro;

               //System.Windows.Forms.MessageBox.Show("x");

               EventoClickEnBotonManoDerecha(xmDer, ymDer, zManderecha, zManizquierda, z_Cabeza); //Enviamos lecturas de puntos al cu
rsor

               if (zManderecha + sensibilidadTouchMano <= z_Cabeza)
               {
                   HandDerCursorElement.Visibility = System.Windows.Visibility.Visible;
               }
               else
               {
                   HandDerCursorElement.Visibility = System.Windows.Visibility.Collapsed;
               }

               if (zManizquierda + sensibilidadTouchMano <= z_Cabeza)
               {
                   HandIzqCursorElement.Visibility = System.Windows.Visibility.Visible;
               }
               else
               {
                   HandIzqCursorElement.Visibility = System.Windows.Visibility.Collapsed;
               }

               if (zManderecha + sensibilidadTouchMano <= z_Cabeza && ClicRightGripped == true) // Gesto pintar con mano derecha, si
el BloquearClick de pintar esta desactivado
               {
                   pintarConManoDerecha(xmDer, ymDer);
               }
               else
               {
                   band = 0;
               }

               if (zManizquierda + sensibilidadTouchMano <= z_Cabeza && ClicLeftGripped == true) // Gesto pintar con mano izquierda,
si el BloquearClick de pintar esta desactivado
               {
                   pintarConManoIzquierda(xmIzq, ymIzq);
               }
               else
               {
                   band2 = 0;
               }

               /*Modo presentador*/

               if (ymDer > yh  && xmDer < xh  && zManderecha + .6 <= z_Cabeza && BloquearGesto == false) //Gesto avanzar diapositiva
               {
                   Thread.Sleep(500);
                   clickAudio();
                   ScaleTransform scale = new ScaleTransform(1, 1);
                   Image_1.RenderTransform = scale;
                   try
                   {
                       Avanzar();
                   }
                   catch
                   {
                       System.Windows.MessageBox.Show("No se han cargado diapositivas en el proyecto");
                   }
               }

               if (ymIzq > yh  && xmIzq > xh && zManizquierda + .6 <= z_Cabeza && BloquearGesto == false)//Gesto retroceder diapositi
va
               {
                   Thread.Sleep(500);
                   ScaleTransform scale = new ScaleTransform(1, 1);
```

```
                Image_1.RenderTransform = scale;
                try
                {
                    Retroceder();
                }
                catch
                {
                    System.Windows.MessageBox.Show("No se han cargado diapositivas en el proyecto");
                }
            }

            /*Modo presentador video*/
            if (ymDer > yh && xmDer < xh  && zManderecha + .5 <= z_Cabeza && BloquearGestovideo == false)
            {
                Thread.Sleep(500);
                if (contadorv >= 3)
                {
                    this.inkCanvas1.Strokes.Clear();
                    inkCanvas1.Children.Clear();

                    contadorv = 0;
                    reproducirVideoContenedor(++contadorv);
                    textBox2.Text = contadorv.ToString();
                }
                else
                {
                    this.inkCanvas1.Strokes.Clear();
                    inkCanvas1.Children.Clear();

                    reproducirVideoContenedor(++contadorv);
                    textBox2.Text = contadorv.ToString();
                }
            }

            if (ymIzq > yh && xmIzq > xh  && zManizquierda + .5 <= z_Cabeza && BloquearGestovideo == false)
            {
                Thread.Sleep(500);
                if (contadorv <= 1)
                {
                    this.inkCanvas1.Strokes.Clear();
                    inkCanvas1.Children.Clear();

                    contadorv = 1;
                    reproducirVideoContenedor(contadorv);
                    textBox2.Text = contadorv.ToString();
                }
                else
                {
                    this.inkCanvas1.Strokes.Clear();
                    inkCanvas1.Children.Clear();

                    reproducirVideoContenedor(--contadorv);
                    textBox2.Text = contadorv.ToString();
                }
            }

            if (ymIzq - 100 < punto && ymDer -
100 < punto && xmIzq < xh && xmDer > xh && zManderecha + sensibilidadTouchMano < z_Cabeza && zManizquierda + sensibilidadTouchMano <
z_Cabeza) //Gesto Zoom
            {
                float distance = Math.Abs(xmDer - xmIzq);
                image_Zoom(distance, xh, yh);
            }

            if (ymDer == ymIzq && xmDer == xmIzq)//Gesto Salir de Modo presentador
            {
                try
                {
                    Estado = 0; // Aplicación finalizada
                    reiniciarANgulo();
                    this._KinectDevice.Stop();
                    this.Close();
                }
                catch (Exception)
                {
                    this.Close();
                }
            }

        }
    }
```

6.7.4 Prueba

6.6.4.1 Pruebas por los desarrolladores

Las pruebas efectuadas en el requerimiento se basan en la interacción del usuario (desarrollador) con el sistema, se avaluaron las funcionalidades de avanzar y retroceder en un ambiente controlado tal y como se especifica en este documento.

La interacción del usuario con la interfaz se hace más intuitiva y se reduce el número de errores durante el tracking y evaluación de gestos es. (Ver figura 6.25).

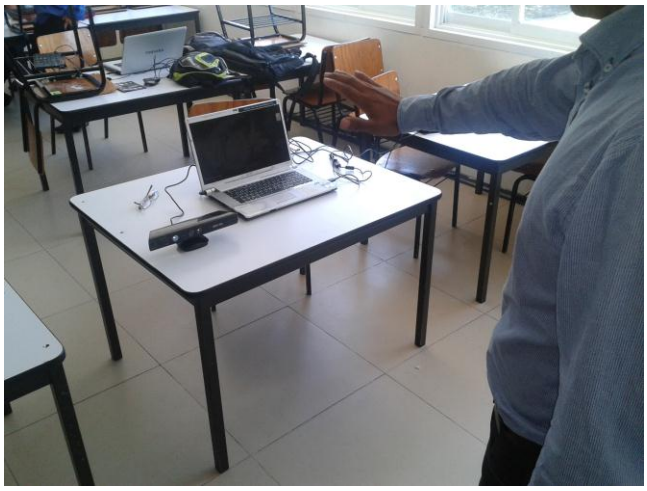

Figura 6.25: Usuario interactuando con el sistema efectuando la función de avanzar diapositiva.

6.6.4.2 Pruebas de Aceptación de usuarios

Durante esta etapa de prueba se invitó a tres usuarios para que realizaran los gestos de adelantar y retroceder, cada uno de ellos le tomo un promedio de diez minutos aprender dichas acciones, una vez más se hizo notoria la variabilidad de puntos durante el tracking con movimientos bruscos y con un ambiente no controlado de manera correcta, se prevé que en el incremento donde se realice la mejora del tracking ayude al algoritmo a ser más preciso para la evaluación de las coordenadas, pasando la observación, los usuarios aceptaron este incremento.

6.8 Incremento 7: integración de animaciones durante el cambio de diapositivas

6.8.1 Análisis

En el planteamiento del problema se menciona el poco interés de los espectadores por la falta de herramientas de interacción llamativas y una visualización con un grado de énfasis para hacer la presentación más elaborada, por tal razón en este incremento se incorporan animaciones al momento de hacer un avance o retroceso de diapositivas.

Aunque en la mayoría de software destinado para el propósito de realizar presentaciones se dispone de estas herramientas, el hecho de hacer un barrido o transición de una imagen al compás de los movimientos de las manos es de gran atracción para las personas que estén presenciando la cátedra.

6.8.2 Diseño

El código debe de escribirse en el documento modo presentador y se registra como recurso para que pueda ser utilizado por el control, a continuación se escribe el código xaml para tal requerimiento:

```
<Window x:Class="PVMCDSMPMA.ModoPresentador"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="ModoPresentador" Width="800" Height="600"  WindowStartupLocation="CenterOwner" WindowState="Maximized" Loaded="Window_
Loaded" WindowStyle="None" ResizeMode="NoResize" ShowInTaskbar="False" AllowsTransparency="True" Background="Transparent" BorderThick
ness="1">

    <!-- Recursos -->
    <Window.Resources>

        <Storyboard x:Name="AnimacionTransparente" x:Key="AnimacionTransparente">
            <DoubleAnimation
                Duration="00:00:01.50"
                From="1" To="0"
                Storyboard.TargetProperty="Opacity"
                Storyboard.TargetName="Image_1"/>
        </Storyboard>
        <Storyboard x:Name="AnimacionOpaca" x:Key="AnimacionOpaca">
            <DoubleAnimation
                Duration="00:00:01.50"
                From="0"
                To="1"
                Storyboard.TargetProperty="Opacity"
                Storyboard.TargetName="Image_1" />
        </Storyboard>
    </Window.Resources>
```

Como se puede observar en el código se hace el uso de Storyboard y DoubleAnimation, y el diagrama de secuencias queda estipulado de la siguiente manera.

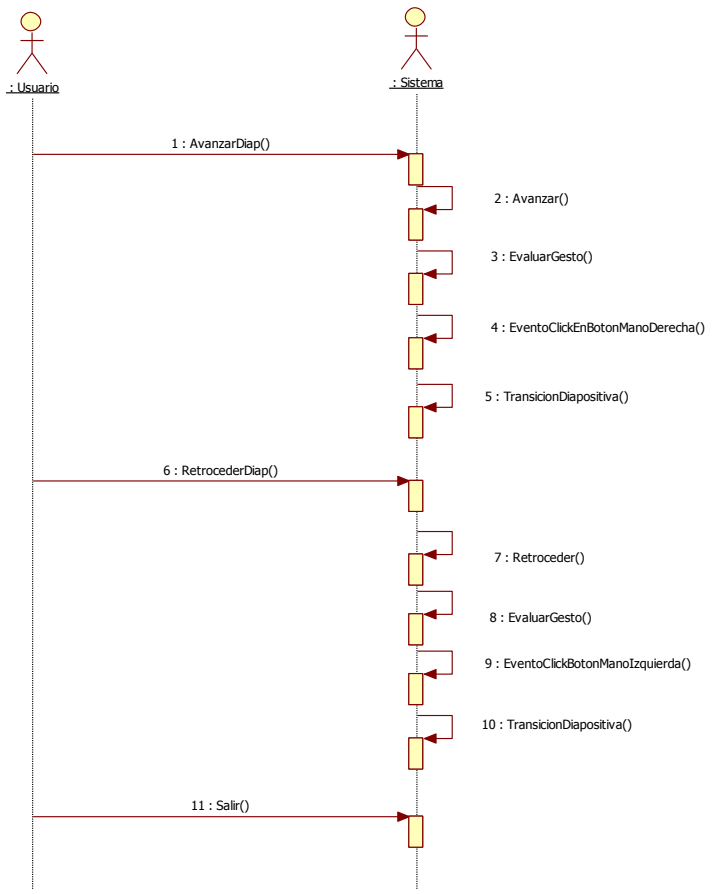

Figura 6.26: Diagrama de secuencias para efectos de transiciones entre diapositivas.

6.8.3 Código

Dentro del código .cs no se tiene que realizar algún cambio debido a que se hace el binding con el control.

6.8.4 Prueba

6.8.4.1 Pruebas por los desarrolladores

Las pruebas efectuadas no son complejas, solo basta observar la transición que realiza la diapositiva al momento de efectuar el gesto avanzar o retroceder.

6.8.4.2 Pruebas de Aceptación de usuarios

En estas pruebas de transición de diapositivas como mejora visual, se invito a tres personas para hacer uso del programa, se les pidió realizar el gesto de avanzar y retroceder,

la opinión de los usuarios fue de mucha importancia, hicieron mención si las transiciones se podían configurar como lo hacen los programas especiales para elaboración de presentaciones, la respuesta fue que no se podía hacer esa función, por tal motivo el requerimiento encontrado por parte de los usuarios queda para un trabajo a futuro o la versión dos del sistema.

6.9 Incremento 8: creación de herramientas interactivas para edición en tiempo de presentación

6.9.1 Análisis

El sistema necesita de un requerimiento esencial para poder tener una presentación profesional, por tal motivo se implementa un cuadro de herramientas donde se alojan funciones para la modificación de la diapositiva en tiempo de presentación, cabe señalar que las modificaciones que se realizan no modifican la diapositiva y solo pueden ser vistas al momento de estar exponiendo, una vez terminada la presentación o ya sea utilizando la función de borrado, todos los elementos pintados son eliminados.

6.9.2 Diseño

El diseño requiere de una sección y ventana por aparte, por esa razón se crea un directorio especial para alojar los archivos xaml y .cs, (ver figura 6.27).

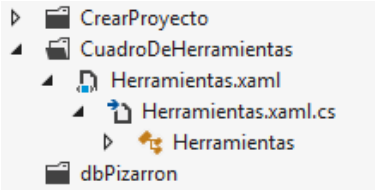

Figura 6.27: Carpeta especial para la ventana de herramientas interactivas.

Dentro de la carpeta tendremos el código .xaml en donde construiremos el diseño, para efectuar las funcionalidades se requiere de poder controlar botones de tamaño grande.

La interacción entre el usuario el programa es secuencial y dinámica, cada herramienta se comporta de manera diferente, por lo mismo el usuario necesita aprender y entrenar con las

herramientas por un tiempo, una vez que se domina, la manipulación y uso de las herramientas se hacer de manera fácil y natural.

Las herramientas son utilizadas haciendo uso del clic con la mano derecha o izquierda según sea el caso, a continuación se define un diagrama de secuencia para ver el comportamiento del sistema (ver figura 6.28).

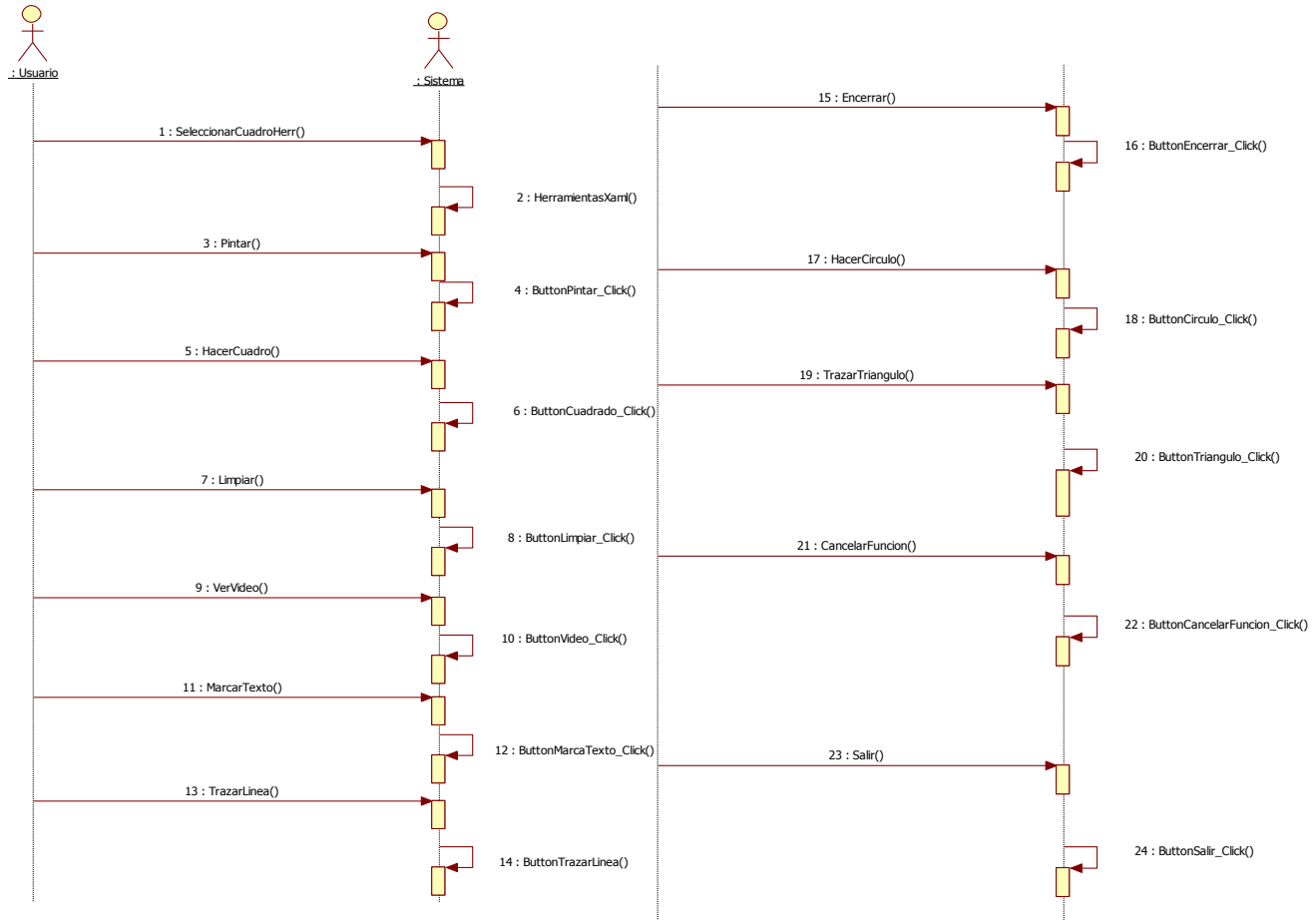

Figura 6.28: Diagrama de secuencias para la interacción con el cuadro de herramientas.

En total se requieren de diez controles de tipo Button y un control Grid para definir la ubicación de cada control, los botones se les fue asignado una imagen que corresponde a la acción a realizar (ver figura 6.29).

Figura 6.29: Cuadro de Herramientas distribución de contenido y diseño.

Como se puede apreciar en la imagen anterior, el diseño es básico y no requiere de mucho tiempo de elaboración, cabe recalcar que la mayor parte de tiempo en desarrollo lo toma el algoritmo para enlazar el evento del control y el pintado en la diapositiva.

6.9.3 Código

El cuadro de herramientas consta del siguiente código xaml el cual define nombre, alineaciones y eventos:

```
<Window x:Class="PVMCDSMPMA.Herramientas"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        WindowStartupLocation="CenterScreen"
        Title="Herramientas" Height="600" Width="777" OpacityMask="Black"
        WindowStyle="None"
        ResizeMode="NoResize"
        ShowInTaskbar="False"
        AllowsTransparency="True"
        Background="CadetBlue" BorderBrush="#00000000">
    <Grid OpacityMask="Black">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="128*" />
            <ColumnDefinition Width="208*" />
            <ColumnDefinition Width="6*" />
            <ColumnDefinition Width="194*" />
            <ColumnDefinition Width="12*" />
            <ColumnDefinition Width="141*" />
```

```
            <ColumnDefinition Width="67*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="55*" />
            <RowDefinition Height="56*" />
            <RowDefinition Height="11*" />
            <RowDefinition Height="99*" />
            <RowDefinition Height="23*" />
            <RowDefinition Height="136*" />
            <RowDefinition Height="143*" />
            <RowDefinition Height="38*" />
        </Grid.RowDefinitions>
        <Button Content="Cerrar" Height="63" HorizontalAlignment="Left" Name="buttonCerrar" VerticalAlignment="Bottom" Width="530" Cl
ick="buttonCerrar_Click" FontSize="32" Background="#00F5F5DC" Focusable="False" Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="6" Bord
erBrush="#00000000" OpacityMask="#7B000000" />
        <Image Grid.Column="1" Grid.Row="3" Height="80" HorizontalAlignment="Left" Margin="40,0,0,0" Name="image1" Stretch="Fill" Ver
ticalAlignment="Top" Width="125" Source="/PVMCDSMPMA;component/Imagenes/lapiz.png" />
        <Image Grid.Column="1" Grid.Row="6" Height="143" HorizontalAlignment="Left" Name="image2" Stretch="Fill" VerticalAlignment="T
op" Width="207" Source="/PVMCDSMPMA;component/Imagenes/linea.png" />
        <Image Grid.Column="1" Grid.Row="5" Height="136" HorizontalAlignment="Left" Name="image3" Stretch="Fill" VerticalAlignment="T
op" Width="208" Source="/PVMCDSMPMA;component/Imagenes/cuadro.png" />
        <Image Grid.Column="3" Grid.ColumnSpan="2" Grid.Row="5" Height="136" HorizontalAlignment="Left" Name="image4" Stretch="Fill"
VerticalAlignment="Top" Width="194" Source="/PVMCDSMPMA;component/Imagenes/borrador.png" />
        <Image Grid.Column="5" Grid.ColumnSpan="2" Grid.Row="3" Height="99" HorizontalAlignment="Left" Name="image5" Stretch="Fill" V
erticalAlignment="Top" Width="142" Source="/PVMCDSMPMA;component/Imagenes/video.png" />
        <Image Grid.Column="3" Grid.ColumnSpan="2" Grid.Row="3" Height="99" HorizontalAlignment="Left" Name="image6" Stretch="Fill" V
erticalAlignment="Top" Width="194" Source="/PVMCDSMPMA;component/Imagenes/plumon.png" />
        <Image Grid.Column="3" Grid.ColumnSpan="2" Grid.Row="6" Height="143" HorizontalAlignment="Left" Name="image7" Stretch="Fill"
VerticalAlignment="Top" Width="194" Source="/PVMCDSMPMA;component/Imagenes/encerrar.png" />
        <Image Grid.Column="5" Grid.Row="5" Height="136" HorizontalAlignment="Left" Name="image8" Stretch="Fill" VerticalAlignment="T
op" Width="140" Source="/PVMCDSMPMA;component/Imagenes/circulo_azul.png" Margin="1,0,0,0" />
        <Image Grid.Column="5" Grid.Row="6" Height="143" HorizontalAlignment="Left" Name="image9" Stretch="Fill" VerticalAlignment="T
op" Width="140" Source="/PVMCDSMPMA;component/Imagenes/triangulo.png" Margin="1,0,0,0" />
        <Button Content="Pintar" Height="99" HorizontalAlignment="Right" Name="button1" VerticalAlignment="Bottom" Width="208" Click=
"buttonPintar_Click" FontSize="24" Background="#00F5F5DC" Focusable="False" Grid.Row="3" Grid.Column="1" BorderBrush="#00000000" Opac
ityMask="#7B000000" Foreground="#FF070706" />
        <Button Content="Limpiar" Height="136" HorizontalAlignment="Left" Name="buttonLimpiar" VerticalAlignment="Top" Width="194" Cl
ick="buttonLimpiar_Click" FontSize="24" Background="#00F5F5DC" Focusable="False" Grid.Row="5" Grid.Column="3" Grid.ColumnSpan="2" Bor
derBrush="#00000000" OpacityMask="#7B000000" Foreground="Black" />
        <Button Content="Marcatexto" Height="99" HorizontalAlignment="Left" Name="buttonLinea" VerticalAlignment="Bottom" Width="194"
 Click="buttonMarcatexto_Click" FontSize="24" Background="#00F5F5DC" Focusable="False" Grid.Row="3" Grid.Column="3" Grid.ColumnSpan="
2" BorderBrush="#00000000" OpacityMask="#7B000000" Foreground="White" />
        <Button Content="Trazar Cuadro" Height="136" HorizontalAlignment="Left" Name="buttonCuadro" VerticalAlignment="Top" Width="20
7" Click="buttonCuadro_Click" FontSize="24" Background="#00F5F5DC" Focusable="False" Grid.Row="5" Grid.Column="1" BorderBrush="#00000
000" Foreground="White" OpacityMask="#7B000000" />
        <Button Content="Video" Height="99" HorizontalAlignment="Left" Name="buttonVideo" VerticalAlignment="Top" Width="142" FontSiz
e="24" Background="#00F5F5DC" Focusable="False" Click="buttonVideo_Click" Grid.Row="3" Grid.Column="5" BorderBrush="#00000000" Opacit
yMask="#7B000000" Foreground="White" Grid.ColumnSpan="2" />
        <Button Content="Trazar Linea" Height="143" HorizontalAlignment="Left" Name="buttonTrazarLinea" VerticalAlignment="Top" Width
="207" FontSize="24" Background="#00F5F5DC" Focusable="False" Click="buttonTrazarLinea_Click" Grid.Row="6" Grid.Column="1" BorderBrus
h="#00000000" OpacityMask="#B4E23737" Foreground="#FFE8E8E8" />
        <Button Content="Encerrar" Height="143" HorizontalAlignment="Left" Name="buttonEncerrar" VerticalAlignment="Top" Width="194"
FontSize="24" Background="#00F5F5DC" Focusable="False" Click="buttonEncerrar_Click" Grid.Row="6" Grid.Column="2" BorderBrush="#000000
00" OpacityMask="#7B000000" Grid.ColumnSpan="2" Margin="5,0,0,0" />
        <Button Content="Circulo" Height="136" HorizontalAlignment="Left" Name="buttonCirculo" VerticalAlignment="Top" Width="141" Fo
ntSize="24" Background="#00F5F5DC" Focusable="False" Click="buttonCirculo_Click" Grid.Row="5" Grid.Column="5" BorderBrush="#00000000"
 OpacityMask="#7B000000" />
        <Button Content="Triangulo" Height="143" HorizontalAlignment="Left" Name="buttonTriangulo" VerticalAlignment="Top" Width="141
" FontSize="24" Background="#00F5F5DC" Focusable="False" Click="buttonTriangulo_Click" Grid.Row="6" Grid.Column="5" BorderBrush="#000
00000" OpacityMask="#7B000000" />
        <Label Content="Cuadro de Herramientas" Height="49" HorizontalAlignment="Left" Margin="139,25,0,0" Name="label1" VerticalAlig
nment="Top" Width="231" FontSize="20" FontWeight="Normal" FontFamily="Arial" Grid.Column="1" Grid.ColumnSpan="3" />

    </Grid>
</Window>
```

Dentro del archivo .cs tenemos la clase principal de la ventana del cuadro de herramientas, a continuación se muestra el código responsable de la escritura de un pequeño archivo para poder guardar la función de la herramienta seleccionada:

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

```
using System.Xml;

namespace PVMCDSMPMA
{
    /// <summary>
    /// Lógica de interacción para Herramientas.xaml
    /// </summary>
    public partial class Herramientas : Window
    {

        public Herramientas()
        {
            InitializeComponent();
        }

        private void buttonCerrar_Click(object sender, RoutedEventArgs e)
        {
          //  Funcion(" ");
            this.Close();
        }

        private void buttonPintar_Click(object sender, RoutedEventArgs e)
        {
            Funcion("Pintar");
            this.Close();
        }

        private void buttonCuadro_Click(object sender, RoutedEventArgs e)
        {
            Funcion("Cuadro");
            this.Close();
        }

        private void buttonLimpiar_Click(object sender, RoutedEventArgs e)
        {
            Funcion("Limpiar");
            this.Close();
        }

        private void buttonVideo_Click(object sender, RoutedEventArgs e)
        {
            Funcion("Video");
            this.Close();
        }

        private void buttonMarcatexto_Click(object sender, RoutedEventArgs e)
        {
            Funcion("Marcatexto");
            this.Close();
        }

        private void buttonTrazarLinea_Click(object sender, RoutedEventArgs e)
        {
            Funcion("Linea");
            this.Close();
        }

        private void buttonEncerrar_Click(object sender, RoutedEventArgs e)
        {
            Funcion("Encerrar");
            this.Close();
        }

        private void buttonCirculo_Click(object sender, RoutedEventArgs e)
        {
            Funcion("Circulo");
            this.Close();
        }

        private void buttonTriangulo_Click(object sender, RoutedEventArgs e)
        {
            Funcion("Triangulo");
            this.Close();
        }

        public void Funcion(String valor)
        {

            string dir = Environment.CurrentDirectory + @"\Funcion.xml";
            XmlTextWriter Funcion = new XmlTextWriter(dir, Encoding.UTF8);
            Funcion.Formatting = Formatting.Indented;
            Funcion.WriteStartDocument(false);
            Funcion.WriteComment(" :) ");

            Funcion.WriteStartElement("ValorDeFuncion");
            Funcion.WriteElementString("Valor", valor);
            Funcion.WriteEndElement();

            Funcion.Flush();
            Funcion.Close();
        }
        } }
```

Para completar la funcionalidad se requiere la lectura del archivo y ejecución de funciones que se encuentran en el archivo ModoPresentador.cs. La ventana de herramientas es ejecutada cuando se lleva el cursor de la mano derecha al icono en forma de pizarrón, una vez que es elegido la función retorna el evento y ejecuta la función de lectura:

```
private void buttonHerramientas_MouseEnter(object sender, MouseEventArgs e)
        {
            /*Habilitamos el Clic*/
            BloquearClick = false;

            /*Bloqueamos la interpretación de gestos*/
            BloquearGesto = true;

            /*Bloqueamos las funciones*/
            BloquearFuncionPintar        = true;
            BloquearFuncionMarcatexto    = true;
            BloquearGestovideo           = true;
            BloquearFuncionPintarCuadro  = true;
            BloquearFuncionPintarCirculo = true;
            BloquearFuncionPintarLinea   = true;
            BloquearFuncionEncerrar      = true;
            BloquearFunciontriangulo     = true;

            FuncionZoom = true;

            /*Visualizar ventana de herramientas*/
            Herramientas herr = new Herramientas();
            herr.ShowDialog();

            /*Habilitar la interpretacion de gestos*/
            BloquearGesto = false;

            /*Leer archivo generado por la ventana herramientas*/
            LeerFuncion();
        }
```

El contenido de la función es el siguiente:

```
private void LeerFuncion()
        {
            /*Leer archivo xml generado por la ventana herramientas*/
            String valor = String.Empty;
            string dir = Environment.CurrentDirectory + @"\Funcion.xml";
            XmlTextReader reader = new XmlTextReader(dir);

            while (reader.Read())
            {
                reader.MoveToContent();
                if (reader.NodeType == System.Xml.XmlNodeType.Text)
                    valor = reader.Value;
            }
```

La tarea principal de la función es colapsar controles de tipo Button definidos y programados para cada herramienta, como ejemplo podemos encontrar la herramienta de pintado, esta herramienta utiliza una paleta de colores que se pintan a un lado de la diapositiva y tiene colores definidos: Azul, Rojo, Rosa, Blanco y Verde, la paleta es minimalista y fácil de utilizar.

Cada función desencadena, deshabilita y habilita los controles que se necesiten. Las herramientas cuentan con funciones programadas normalmente inicializadas por el disparo de los eventos del Mouse.

Debido a la rápida programación de las funcionalidades el sistema utiliza dos recursos de hardware: Kinect y Mouse, combinando los eventos de cada dispositivo se logran utilizar controles de WPF.

El sensor de movimiento se encarga de disparar eventos del sekeleton recolectando información de las coordenadas de cada Joint explicadas con anterioridad en este documento y el cursor del mouse es controlado por un Joint, así mismo el evento del Kinect sirve para mover y ejecutar los eventos del Mouse.

Definimos al inicio de la clase principal la variable a utilizar para controlar el ratón:

```
public partial class ModoPresentador : Window
    {
        #region Variables

        [DllImport("user32")]
        public static extern int SetCursorPos(int x, int y);
        private const int MOUSEEVENT_MOVE = 0x0001;
        private const int MOUSEEVENT_LEFTDOWN = 0x0002;
        private const int MOUSEEVENT_LEFTUP = 0x0004;
        [DllImport("user32.dll", CharSet = CharSet.Auto, CallingConvention = CallingConvention.StdCall)]
        public static extern void mouse_event(int dwFlags, int dx, int dy, int cButtons, int dwExtraInfo);
```

Anteriormente se explicó cómo obtener los puntos de las articulaciones del expositor y se definió las funcionalidades para poder cambiar de diapositiva, ahora se define como controlar el evento del ratón para desencadenar las funciones de las herramientas.

A continuación definiremos la función encargada de recolectar los puntos de la mano derecha, la función efectuara el evento de clic solo cuando el vector Z de la mano derecha se encuentre más alejada del vector Z de la cabeza del expositor, cabe señalar que esta función no es la mejor para disparar los eventos del ratón dado que siempre habrán variaciones de distancia entre la cabeza del expositor y su mano, por el mismo motivo se debe de implementar otro tipo de funcionalidad para solventar los problemas de distancia.

```
public void EventoClickEnBotonManoDerecha(float x, float y, float zmd, float zmi, float zmh) //Función hacer clic sobre cualquier
botón
        {
            if (transform == true)
            {
                trans();
            }

            SetCursorPos((int)x, (int)y); //Mueve la posición del cursor


            if (BloquearClick == false && BloquearClick != true)
            {
                if (zmd + sensibilidadTouchMano >= zmh)
                {

                    this.Cursor = Cursors.Hand;  //cursor tipo hand
                    mouse_event(MOUSEEVENT_LEFTDOWN, (int)x, (int)y, 0, 0);
                    Thread.Sleep(1000);
                    mouse_event(MOUSEEVENT_LEFTUP, (int)x, (int)y, 0, 0);

                    if (BloquearFuncionMarcatexto == false)
```

```
                {
                    tomarPuntosMouse((double)x, (double)y);
                }

                if (BloquearFuncionPintarCuadro == false)
                {
                    tomarPuntosMouse((double)x, (double)y);
                }

                if (BloquearFunciontriangulo == false)
                {
                    tomarPuntosMouse((double)x, (double)y);
                }

                if (BloquearFuncionPintarCirculo == false)
                {
                    tomarPuntosMouse((double)x, (double)y);
                }

                if (BloquearFuncionPintarLinea == false)
                {
                    tomarPuntosMouse((double)x, (double)y);
                }

                if (BloquearFuncionEncerrar == false)
                {
                    tomarPuntosMouse((double)x, (double)y);
                }
            }
        }
        if (bloquearClickArrastrar == false)
        {
            if (zmd + sensibilidadTouchMano <= zmh)
            {
                mouse_event(MOUSEEVENT_LEFTDOWN, (int)x, (int)y, 0, 0);
                //Thread.Sleep(1000);
                //mouse_event(MOUSEEVENT_LEFTUP, (int)x, (int)y, 0, 0);
            }
            if (zmd >= zmh - .3)
            {
                mouse_event(MOUSEEVENT_LEFTUP, (int)x, (int)y, 0, 0);
            }
        }
    }
```

La siguiente función es la encargada de pintar objetos en la diapositiva, utiliza el control Canvas para pintar elipses y líneas, con estos recursos se dibujan los círculos, cuadrados, pintado a mano alzada, marcatexto y todas las herramientas del sistema.

Cada vez que se pase por la función se incrementa una variable global, útil para evaluar las funciones de la Herramienta seleccionada.

```
private void tomarPuntosMouse(double px, double py)
    {
        cantidadClick = cantidadClick + 1;

        if (BloquearFuncionPintarCirculo == false)
        {
            if (cantidadClick == 1)
            {
                px1 = px;
                py1 = py;
                p1.Visibility = Visibility.Visible;
                Canvas.SetLeft(p1, px1 - 20);
                Canvas.SetTop(p1, py1 - 20);
            }
            if (cantidadClick == 2) //Envía a pintar la linea
            {
                px2 = px;
                py2 = py;
                p2.Visibility = Visibility.Visible;
                Canvas.SetLeft(p2, px1 - 20);
                Canvas.SetTop(p2, py1 - 20);
                pintarCirculo(px1, py1, px2, py1);
            }
        }

        if (BloquearFuncionMarcatexto == false)
        {
            if (cantidadClick == 1)
            {
```

```
            px1 = px;
            py1 = py;
            p1.Visibility = Visibility.Visible;
            Canvas.SetLeft(p1, px1 - 20);
            Canvas.SetTop(p1, py1 - 20);
        }
        if (cantidadClick == 2) //Envía a pintar la línea
        {
            px2 = px;
            py2 = py;
            p2.Visibility = Visibility.Visible;
            Canvas.SetLeft(p2, px1 - 20);
            Canvas.SetTop(p2, py1 - 20);
            Marcatexto(px1, py1, px2, py1);
        }
    }

    if (BloquearFuncionEncerrar == false)
    {
        if (cantidadClick == 1) //Toma el punto A
        {
            px1 = px;
            py1 = py;
            p1.Visibility = Visibility.Visible;
            Canvas.SetLeft(p1, px1 - 20);
            Canvas.SetTop(p1, py1 - 20);
        }
        if (cantidadClick == 2) //Toma el punto B y envía a pintar la línea
        {
            px2 = px;
            py2 = py;
            p2.Visibility = Visibility.Visible;
            Canvas.SetLeft(p2, px1 - 20);
            Canvas.SetTop(p2, py1 - 20);
            Encerrar(px1, py1, px2, py2);
        }
    }

    if (BloquearFuncionPintarCuadro == false)
    {
        if (cantidadClick == 1) //Toma el punto A
        {
            px1 = px;
            py1 = py;
            p1.Visibility = Visibility.Visible;
            Canvas.SetLeft(p1, px1 - 20);
            Canvas.SetTop(p1, py1 - 20);
        }
        if (cantidadClick == 2) //Toma el punto B y envía a pintar la línea
        {
            px2 = px;
            py2 = py;
            p2.Visibility = Visibility.Visible;
            Canvas.SetLeft(p2, px1 - 20);
            Canvas.SetTop(p2, py1 - 20);
            PintarCuadro(px1, py1, px2, py2);
        }
    }

    if (BloquearFuncionPintarLinea == false)
    {
        if (cantidadClick == 1) //Toma el punto A
        {
            px1 = px;
            py1 = py;
            p1.Visibility = Visibility.Visible;
            Canvas.SetLeft(p1, px1 - 20);
            Canvas.SetTop(p1, py1 - 20);
        }
        if (cantidadClick == 2) //Toma el punto B y envía a pintar la línea
        {
            px2 = px;
            py2 = py;
            p2.Visibility = Visibility.Visible;
            Canvas.SetLeft(p2, px1 - 20);
            Canvas.SetTop(p2, py1 - 20);
            PintarLinea(px1, py1, px2, py2);
        }
    }

    if (BloquearFunciontriangulo == false)
    {
        if (cantidadClick == 1) //Toma el punto A
        {
            px1 = px;
            py1 = py;
            p1.Visibility = Visibility.Visible;
            Canvas.SetLeft(p1, px1 - 20);
            Canvas.SetTop(p1, py1 - 20);
        }
        if (cantidadClick == 2) //Toma el punto B y envía a pintar la línea
        {
```

```
                px2 = px;
                py2 = py;
                p2.Visibility = Visibility.Visible;
                Canvas.SetLeft(p2, px2 - 20);
                Canvas.SetTop(p2, py2 - 20);
            }
            if (cantidadClick == 3) //Toma el punto B y envía a pintar la línea
            {
                px3 = px;
                py3 = py;
                px4 = px1;
                py4 = py1;
                p3.Visibility = Visibility.Visible;
                Canvas.SetLeft(p3, px3 - 20);
                Canvas.SetTop(p3, py3 - 20);
                PintarTriangulo(px1, py1, px2, py2, px3, py3, px4, py4);
            }
        }
    }
```

La evaluación continúa y entran en subfunciones definidas para cada herramienta, cada una de ellas evalúa los puntos recolectados por el sensor Kinect y en base a ellos define la posición de cada elemento para que sea pintado en el Canvas. Básicamente se construyó un graficador en el cual se pintan controles para representar la funcionalidad de cada Herramienta, a continuación se definen las funciones para cada Herramienta:

```
public void PintarTriangulo(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4)
    {
        Line lineA1 = new Line();
        lineA1.X1 = x1;
        lineA1.Y1 = y1;
        lineA1.X2 = x2;
        lineA1.Y2 = y2;
        SolidColorBrush colorBrush = new SolidColorBrush();
        switch (PaletaColor)
        {
            case "Coral": colorBrush.Color = Colors.Coral;
                break;

            case "Rojo": colorBrush.Color = Colors.Red;
                break;

            case "Azul": colorBrush.Color = Colors.Blue;
                break;

            case "Rosa": colorBrush.Color = Colors.Pink;
                break;

            case "Verde": colorBrush.Color = Colors.Green;
                break;

            case "Blanco": colorBrush.Color = Colors.White;
                break;

        }
        lineA1.StrokeThickness = 3;
        lineA1.Stroke = colorBrush;
        inkCanvas1.Children.Add(lineA1);
        //Image_1.LayoutTransform.
        Line lineA2 = new Line();
        lineA2.X1 = x2;
        lineA2.Y1 = y2;
        lineA2.X2 = x3;
        lineA2.Y2 = y3;
        lineA2.StrokeThickness = 3;
        lineA2.Stroke = colorBrush;
        inkCanvas1.Children.Add(lineA2);

        Line lineB2 = new Line();
        lineB2.X1 = x3;
        lineB2.Y1 = y3;
        lineB2.X2 = x4;
        lineB2.Y2 = y4;
        lineB2.StrokeThickness = 3;
        lineB2.Stroke = colorBrush;
        inkCanvas1.Children.Add(lineB2);
        cantidadClick = 0;
        p1.Visibility = Visibility.Collapsed;
        p2.Visibility = Visibility.Collapsed;
        p3.Visibility = Visibility.Collapsed;
    }
```

```
public void PintarCuadro(double x1, double y1, double x2, double y2)
{
    Line lineA1 = new Line();
    lineA1.X1 = x1;
    lineA1.Y1 = y1;
    lineA1.X2 = x1;
    lineA1.Y2 = y2;
    SolidColorBrush colorBrush = new SolidColorBrush();
    switch (PaletaColor)
    {
        case "Coral": colorBrush.Color = Colors.Coral;
            break;

        case "Rojo": colorBrush.Color = Colors.Red;
            break;

        case "Azul": colorBrush.Color = Colors.Blue;
            break;

        case "Rosa": colorBrush.Color = Colors.Pink;
            break;

        case "Verde": colorBrush.Color = Colors.Green;
            break;

        case "Blanco": colorBrush.Color = Colors.White;
            break;

    }
    lineA1.StrokeThickness = 3;
    lineA1.Stroke = colorBrush;
    inkCanvas1.Children.Add(lineA1);
    //Image_1.LayoutTransform.
    Line lineA2 = new Line();
    lineA2.X1 = x1;
    lineA2.Y1 = y2;
    lineA2.X2 = x2;
    lineA2.Y2 = y2;
    lineA2.StrokeThickness = 3;
    lineA2.Stroke = colorBrush;
    inkCanvas1.Children.Add(lineA2);

    Line lineB1 = new Line();
    lineB1.X1 = x2;
    lineB1.Y1 = y2;
    lineB1.X2 = x2;
    lineB1.Y2 = y1;
    lineB1.StrokeThickness = 3;
    lineB1.Stroke = colorBrush;
    inkCanvas1.Children.Add(lineB1);

    Line lineB2 = new Line();
    lineB2.X1 = x2;
    lineB2.Y1 = y1;
    lineB2.X2 = x1;
    lineB2.Y2 = y1;
    lineB2.StrokeThickness = 3;
    lineB2.Stroke = colorBrush;
    inkCanvas1.Children.Add(lineB2);

    cantidadClick = 0;//Reiniciamos conteo de clics
    //Ocultamos puntos de referencia
    p1.Visibility = Visibility.Collapsed;
    p2.Visibility = Visibility.Collapsed;
    p3.Visibility = Visibility.Collapsed;
}

public void PintarLinea(double x1, double y1, double x2, double y2)
{
    Line lineA1 = new Line();
    lineA1.X1 = x1;
    lineA1.Y1 = y1;
    lineA1.X2 = x2;
    lineA1.Y2 = y2;
    SolidColorBrush colorBrush = new SolidColorBrush();
    switch (PaletaColor)
    {
        case "Coral": colorBrush.Color = Colors.Coral;
            break;

        case "Rojo": colorBrush.Color = Colors.Red;
            break;

        case "Azul": colorBrush.Color = Colors.Blue;
            break;

        case "Rosa": colorBrush.Color = Colors.Pink;
            break;

        case "Verde": colorBrush.Color = Colors.Green;
            break;
```

```
        case "Blanco": colorBrush.Color = Colors.White;
            break;

    }
    lineA1.StrokeThickness = 3;
    lineA1.Stroke = colorBrush;
    inkCanvas1.Children.Add(lineA1);

    cantidadClick = 0;//Reiniciamos conteo de clics
    //Ocultamos puntos de referencia
    p1.Visibility = Visibility.Collapsed;
    p2.Visibility = Visibility.Collapsed;
    p3.Visibility = Visibility.Collapsed;
}

public void Encerrar(double x1, double y1, double x2, double y2)
{
    Line lineA1 = new Line();
    double valorY1 = 0;
    double ancho = Math.Abs(y2 - y1);
    valorY1 = ((y2 - y1) / 2);
    valorY1 = y1 + valorY1;

    lineA1.X1 = x1;
    lineA1.Y1 = valorY1;
    lineA1.X2 = x2;
    lineA1.Y2 = valorY1;

    SolidColorBrush colorBrush2 = new SolidColorBrush();
    colorBrush2.Color = Color.FromArgb(150, 255, 238, 0);
    lineA1.StrokeThickness = ancho+10;
    lineA1.Stroke = colorBrush2;
    inkCanvas1.Children.Add(lineA1);

    cantidadClick = 0;//Reiniciamos conteo de clics
    //Ocultamos puntos de referencia
    p1.Visibility = Visibility.Collapsed;
    p2.Visibility = Visibility.Collapsed;
    p3.Visibility = Visibility.Collapsed;
}

public void Marcatexto(double x1, double y1, double x2, double y2)
{
    Line line = new Line();
    line.X1 = x1;
    line.Y1 = y1;
    line.X2 = x2;
    line.Y2 = y2;
    SolidColorBrush colorBrush2 = new SolidColorBrush();

    colorBrush2.Color = Color.FromArgb(150, 255, 238, 0);

    line.StrokeThickness = 20;
    line.Stroke = colorBrush2;
    inkCanvas1.Children.Add(line);

    cantidadClick = 0;//Reiniciamos conteo de clics
    //Ocultamos puntos de referencia
    p1.Visibility = Visibility.Collapsed;
    p2.Visibility = Visibility.Collapsed;
    p3.Visibility = Visibility.Collapsed;
}

private void pintarCirculo(double px1, double py1, double px2, double py2)
{
    double xc = px1;
    double yc = py1;

    double xf = px2;
    double yf = py2;

    double rad = Math.Sqrt(Math.Pow((xc - xf), 2) + Math.Pow((yc - yf), 2));

    int n = 1000;
    for (int i = 0; i <= n; i++)
    {
        double f1 = 2 * Math.PI * i / n;
        double x = xc + Math.Abs(rad) * Math.Cos(f1);
        double y = yc + Math.Abs(rad) * Math.Sin(f1);
        trazarPuntos(x, y);
    }

    cantidadClick = 0;
    band = 0;
    p1.Visibility = Visibility.Collapsed;
    p2.Visibility = Visibility.Collapsed;
    p3.Visibility = Visibility.Collapsed;
}

private void trazarPuntos(double x, double y)
{
```

```
        if (band == 0)
        {
            GX2 = x;
            GY2 = y;
            band = 1;
        }
        else
        {
            Line lineP = new Line();
            lineP.X2 = x;
            lineP.Y2 = y;

            lineP.X1 = GX2;
            lineP.Y1 = GY2;

            SolidColorBrush colorBrush = new SolidColorBrush();
            switch (PaletaColor)
            {
                case "Coral": colorBrush.Color = Colors.Coral;
                    break;

                case "Rojo": colorBrush.Color = Colors.Red;
                    break;

                case "Azul": colorBrush.Color = Colors.Blue;
                    break;

                case "Rosa": colorBrush.Color = Colors.Pink;
                    break;

                case "Verde": colorBrush.Color = Colors.Green;
                    break;

                case "Blanco": colorBrush.Color = Colors.White;
                    break;

            }
            lineP.StrokeThickness = 3;
            lineP.Stroke = colorBrush;
            inkCanvas1.Children.Add(lineP);
            GX2 = x;
            GY2 = y;
        }
    }
```

Se pasaran las veces necesarias en cada función para poder completar la figura de manera correcta siempre y cuando se dispare el evento clic con la mano derecha.

6.9.4 Prueba

6.9.4.1 Pruebas por los desarrolladores

Se realizaron pruebas en un aula Donde fueron participes tres usuarios del instituto Tecnológico de Tuxtla Gutiérrez, los usuarios tenían diferente talla y estatura, se tomaron las muestras de cada control pintado con la mano derecha e izquierda.

Las funciones hacen su trabajo de pintar correctamente cada figura geométrica y así mismo la funcionalidad es cumplida.

Anteriormente se recalcó el problema de reconocer el evento para disparar el clic, el gesto es llevar la mano derecha hacia al frente del sensor sin caminar, al tomar de regencia las profundidades o vector z de la cabeza con la mano y así mismo evaluarlas, se encuentra el problema de poca precisión.

Para mejorar el o corregir el error que se genera durante el tracking se pretende utilizar una técnica de filtrado de corrección de puntos, mas adelante se detallara este nuevo requerimiento

Figura 6.30: Usuario pintando con la mano derecha e izquierda.

6.9.4.2 Pruebas de Aceptación de usuarios

Tres usuarios probaron el sistema realizando diferentes usos de los controles, gestos y herramientas, cabe mencionar que estos ya habían interactuado con anterioridad con el sistema, no se observo problema alguno cuando utilizaron las funciones o herramientas y la opinión recibida por cada uno de ellos fue positiva.

6.10 Incremento 9: Integración de función para detectar apertura y cierre de las manos

6.10.1 Análisis

Retomando el problema de la variabilidad entre la profundidad de la mano derecha y la cabeza del expositor para disparar el evento de tipo clic, se tiene la necesidad de

reprogramar la función, por tal motivo este incremento define la función de detección de apertura y cierre de la mano derecha o izquierda.

Microsoft Kinect cuenta con la biblioteca Interaction la cual cuenta con distintas funcionalidades en ella incorpora la detección de la apertura y cierre de las dos manos.

6.10.2 Diseño

No se requiere de un diseño en especial debido a que solo se necesita definir la clase InteractionClient. No se requiere de la implementación de controles ni de ventanas o carpetas especiales para contener el archivo .cs. Sin embargo se diseña el diagrama de secuencias que el sistema tiene que llevar acabo para la detección del gesto antes mencionado.

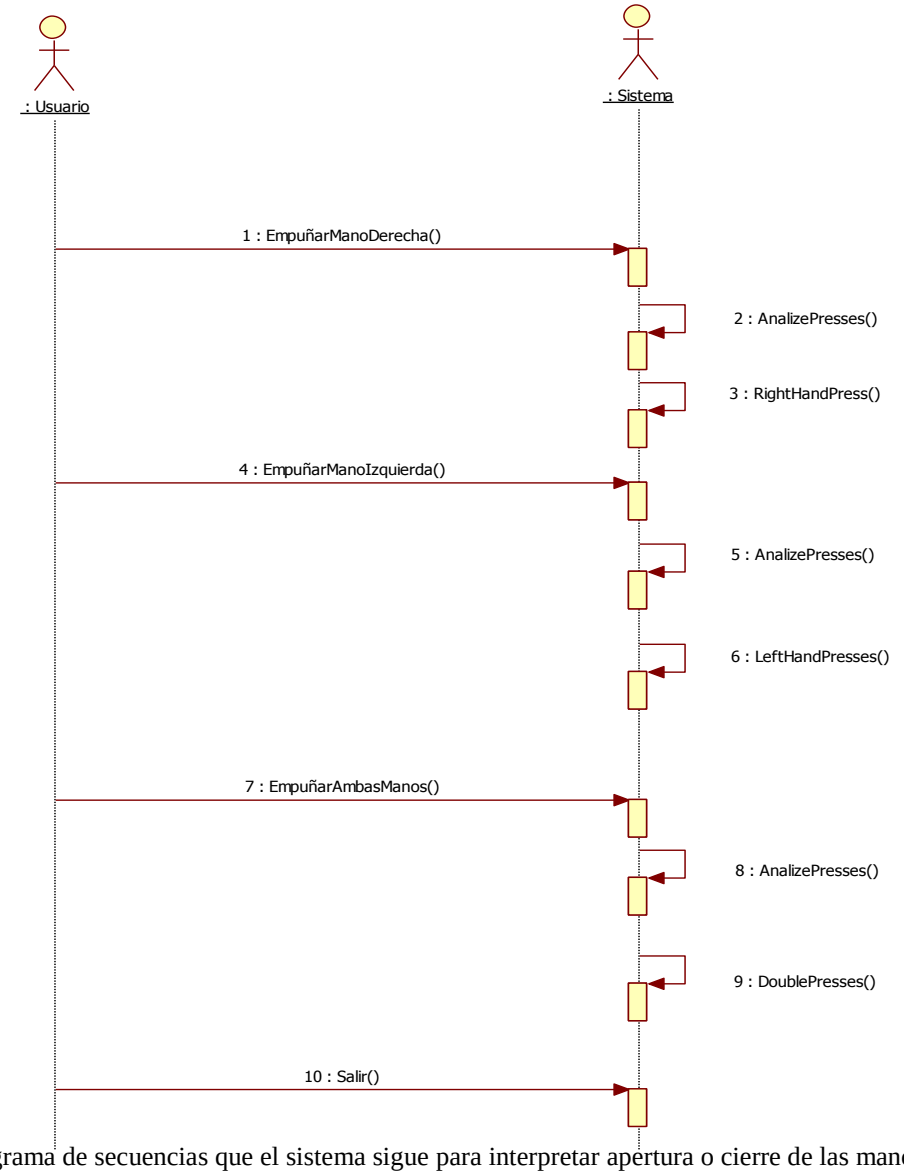

Figura 6.31: Diagrama de secuencias que el sistema sigue para interpretar apertura o cierre de las manos.

6.10.3 Código

Antes de definir las clases, funciones y variables se tiene que crear el archivo InteractionClient donde definiremos el código para la detección de la apertura y cierre de las manos.

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

using Microsoft.Kinect.Toolkit.Interaction;

namespace PVMCDSMPMA
{
    class InteractionClient : IInteractionClient
    {
        public InteractionInfo GetInteractionInfoAtLocation(
        int skeletonTrackingId,
        InteractionHandType handType,
        double x,
        double y)
        {
            var result = new InteractionInfo();
            result.IsGripTarget = true;
            result.IsPressTarget = true;
            //result.PressAttractionPointX = 0.5;
            //result.PressAttractionPointY = 0.5;
            //result.PressTargetControlId = 1;
            return result;
        }
    }
}
```

Hacemos referencia a la biblioteca Microsoft.Kinect.Interaction en el archivo ModoPresentador.xaml.cs, después definimos las variables necesarias y métodos.

```
/*Variables del grip*/
        private InteractionStream _interactionStream;
        private UserInfo[] _userInfos;
        bool ClicRightGripped = false;
        bool ClicLeftGripped = false;
/*----------------*/


            #region Grip
        void interactionStream_InteractionFrameReady(object sender, InteractionFrameReadyEventArgs e)
        {
            //textBoxForeachGrip.Text = x++.ToString();
            using (var frame = e.OpenInteractionFrame())
            {
                if (frame == null)
                    return;

                frame.CopyInteractionDataTo(_userInfos);
                if (_userInfos != null && _userInfos.Length > 0)
                {
                    bool rightHandPress = false;
                    bool leftHandPress = false;
                    bool rightHandGrip = false;
                    bool leftHandGrip = false;
                    bool rightHandRelease = false;
                    bool leftHandRelease = false;

                    foreach (var info in _userInfos)
                    {
                        foreach (var hand in info.HandPointers)
                        {
                            if (hand.HandType == InteractionHandType.Right)
                            {
                                if (!rightHandPress && hand.IsPressed)
                                {
                                    rightHandPress = true;
                               //             txtFuncion.Text = "rightHandPress = true;";

                                }
                                if (!rightHandGrip && hand.HandEventType == InteractionHandEventType.Grip)
                                {
```

```
                        rightHandGrip = true;
                //        txtFuncion.Text = "rightHandGrip = true;";
                        ClicRightGripped = true;
                    }
                    if (!rightHandRelease && hand.HandEventType == InteractionHandEventType.GripRelease)
                    {
                        rightHandRelease = true;
                //        txtFuncion.Text = "rightHandRelease = true;";
                        ClicRightGripped = false;
                    }
                }

                if (hand.HandType == InteractionHandType.Left)
                {
                    if (!leftHandPress && hand.IsPressed)
                    {
                        leftHandPress = true;
                //        txtFuncion.Text = "leftHandPress = true;";

                    }
                    if (!leftHandGrip && hand.HandEventType == InteractionHandEventType.Grip)
                    {
                        leftHandGrip = true;
                //   txtFuncion.Text = "leftHandGrip = true;";
                        ClicLeftGripped = true;
                    }
                    if (!leftHandRelease && hand.HandEventType == InteractionHandEventType.GripRelease)
                    {
                        leftHandRelease = true;
                //   txtFuncion.Text = "leftHandRelease = true;";
                        ClicLeftGripped = false;
                    }
                }
            }
        }
        AnalyzePresses(rightHandPress, leftHandPress);
        AnalyzeGrips(rightHandGrip, rightHandRelease, leftHandGrip, leftHandRelease);
    }


    if (IsLeftGripped && IsRightGripped)
    {
        if (!IsDoubleGripped)
        {
            IsDoubleGripped = true;
           // txtFuncion.Text = "IsDoubleGripped = true;";
        }
    }
    else
    {
        IsDoubleGripped = false;
    }

    if (IsLeftPressed && IsRightPressed)
    {
        if (!IsDoublePressed)
        {
            IsDoublePressed = true;
           // txtFuncion.Text = "IsDoublePressed = true;";
            ClicA();
        }
    }
    else
    {
        IsDoublePressed = false;
    }
}
}

private void AnalyzePresses(bool rightHandPress, bool leftHandPress)
{
    if (!rightHandPress)
    {
        IsRightPressed = false;
    }
    else
    {
        IsRightPressed = true;
    }

    if (!leftHandPress)
    {
        IsLeftPressed = false;
    }
    else
    {
        IsLeftPressed = true;
    }
}

private static void AnalyzeGrips(bool rightHandGrip, bool rightHandRelease, bool leftHandGrip, bool leftHandRelease)
{
```

```
    if (!rightHandGrip)
    {

        if (rightHandRelease)
        {
            IsRightGripped = false;
        }
    }
    else
    {
        IsRightGripped = true;
    }

    if (!leftHandGrip)
    {
        if (leftHandRelease)
        {
            IsLeftGripped = false;
        }
    }
    else
    {
        IsLeftGripped = true;
    }
}

public static bool IsRightGripped
{
    get;
    set;
}

public static bool IsLeftGripped
{
    get;
    set;
}

public static bool IsDoubleGripped
{
    get;
    set;
}

private bool _isRightPressed;
public bool IsRightPressed
{
    get { return _isRightPressed; }
    set
    {
        if (_isRightPressed != value)
        {
            _isRightPressed = value;
            if (value == true)
            {
                //SendKeyStroke(PUSH_RT);
            }

        }
    }
}

private bool _isLeftPressed;
public bool IsLeftPressed
{
    get { return _isLeftPressed; }
    set
    {
        if (_isLeftPressed != value)
        {
            _isLeftPressed = value;
            if (value == true)
            {
                //SendKeyStroke(PUSH_LFT);
            }
        }
    }
}

public static bool IsDoublePressed
{
    get;
    set;
}
#endregion Grip
```

Para que se dispare el evento Interaction es necesario modificar la inicialización del sensor Kinect y definir funciones para la inicialización del mismo.

A continuación se crean y modifican las funciones para inicializar la interacción:

```
#region _PropertiesKinect
        private KinectSensor KinectDevice
        {
            get { return this._KinectDevice; }
            set
            {
                if (this._KinectDevice != value)
                {
                    //Uninitialize
                    if (this._KinectDevice != null) //Detenemos el sensor
                    {
                        this._KinectDevice.Stop();
                        this._KinectDevice.SkeletonFrameReady -= KinectDevice_SkeletonFrameReady;
                        this._KinectDevice.SkeletonStream.Disable();
                        this._FrameSkeletons = null;
                    }

                    this._KinectDevice = value;

                    //Initialize
                    if (this._KinectDevice != null) //Iniciamos el sensor
                    {
                        if (this._KinectDevice.Status == KinectStatus.Connected)
                        {
                            this._KinectDevice.SkeletonStream.Enable(); //Habilitamos el Skeleton
                            this._FrameSkeletons = new Skeleton[this._KinectDevice.SkeletonStream.FrameSkeletonArrayLength];

                            this._interactionStream = new InteractionStream(KinectDevice, new InteractionClient());
                            this._interactionStream.InteractionFrameReady += new
EventHandler<InteractionFrameReadyEventArgs>(interactionStream_InteractionFrameReady);
                            this._KinectDevice.DepthFrameReady += new
EventHandler<DepthImageFrameReadyEventArgs>(sensor_DepthFrameReady);

                            this.KinectDevice.SkeletonFrameReady += KinectDevice_SkeletonFrameReady; //Tomamos el Skeleton

                            this._KinectDevice.ColorStream.Enable();
                            this._KinectDevice.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
                            this._KinectDevice.SkeletonStream.Enable();


                            this._KinectDevice.Start(); //Iniciamos el sensor
                            KinectDevice.ElevationAngle = 0;
                            textBoxMensajes.Text = "El sensor esta Conectado";
                        }
                    }
                }
            }
        }

        void sensor_DepthFrameReady(object sender, DepthImageFrameReadyEventArgs e)
        {
            using (DepthImageFrame frame = e.OpenDepthImageFrame())
            {
                if (frame == null)
                    return;
                _interactionStream.ProcessDepth(frame.GetRawPixelData(), frame.Timestamp);
            }
        }

        #endregion _PropertiesKinect
        /* 4 */
        #region _Kinect_Skeleton_FrameReady

        private void KinectDevice_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)//Cuando se detecta una persona se
obtiene el Skeleton
        {
            using (SkeletonFrame frame = e.OpenSkeletonFrame())
            {

                if (frame != null)
                {
                    try
                    {
                        frame.CopySkeletonDataTo(this._FrameSkeletons);

                        //Grip
                        var accelerometerReading = KinectDevice.AccelerometerGetCurrentReading();
                        this._interactionStream.ProcessSkeleton(this._FrameSkeletons, accelerometerReading, frame.Timestamp);
                        //Grip
                    }
```

```
                    catch (InvalidCastException)
                    {

                    }

                    Skeleton skeleton = GetPrimarySkeleton(this._FrameSkeletons);// Tomamos el skeleton del usuario

                    if (skeleton == null)// No se encontró ningún usuario
                    {
                        skeletonUsuario = 0;
                        ocultarCirculos();
                        botonRojo.Visibility = Visibility.Visible;
                        botonVerde.Visibility = Visibility.Collapsed;
                    }
                    else if (skeleton != null && skeletonUsuario == 1) //Tomamos el eskeleton de un solo usuario
                    {
                        /* Obtenemos los Joints que se utilizaran en el programa*/
                        Joint HandLeft = GetHandLeft(skeleton);
                        Joint HandRight = GetHandRight(skeleton);
                        Joint Head = GetHead(skeleton);
                        Joint homIzq = GetHombroIzq(skeleton);
                        Joint homDer = GetHombroDer(skeleton);
                        Joint homCen = GetHombroCen(skeleton);
                        Joint CentroCadera = GetCadCen(skeleton);
                        /* Obtenemos los Vectores de profundidad "Z" de los Joints que utilizaran en el programa*/
                        float zManoDerecha = skeleton.Joints[JointType.HandRight].Position.Z;
                        float zManoIzquierda = skeleton.Joints[JointType.HandLeft].Position.Z;
                        float zCabeza = skeleton.Joints[JointType.Head].Position.Z;

                        if (zCabeza > 0/* && zCabeza < 3*/) //Evalúa que el usuario no está alejado a 3 metros de distancia y que no
se encuentre a menos de un metro de distancia del sensor
                        {
                            botonVerde.Visibility = Visibility.Visible;
                            botonRojo.Visibility = Visibility.Collapsed;
                            IniTracking(Head, HandLeft, HandRight, homIzq, homDer, homCen, CentroCadera, zManoDerecha,
zManoIzquierda, zCabeza);
                        }
                        else
                        {
                            botonRojo.Visibility = Visibility.Visible;
                            botonVerde.Visibility = Visibility.Collapsed;
                            reiniciarANgulo();
                            ocultarCirculos();
                        }
                    }
                }
            }
        }
```

Una vez que se define el código necesario para la detección de la apertura y cierre se procede a reemplazar el código donde se evalúa la profundidad de la mano derecha y la cabeza del expositor, se reemplaza (`if (zmd + sensibilidadTouchMano >= zmh)`) por (`(ClicRightGripped == true)`) y la función queda definida de la siguiente manera:

```
public void EventoClickEnBotonManoDerecha(float x, float y, float zmd, float zmi, float zmh) //Función hacer clic sobre cualquier
boton
        {
            if (transform == true)
            {
                trans();
            }

            SetCursorPos((int)x, (int)y); //Mueve la posición del cursor


            if (BloquearClick == false && BloquearClick != true)
            {
                if (ClicRightGripped == true)
                {

                    this.Cursor = Cursors.Hand;  //cursor tipo hand
                    mouse_event(MOUSEEVENT_LEFTDOWN, (int)x, (int)y, 0, 0);
                    Thread.Sleep(1000);
                    mouse_event(MOUSEEVENT_LEFTUP, (int)x, (int)y, 0, 0);
                    ClicRightGripped = false;
```

6.10.4 Prueba

6.10.4.1 Pruebas por los desarrolladores

La funcionalidad trabaja correctamente y se hace más sencilla la interacción entre el expositor con la presentación, el usuario no se ve forzado en hacer un gesto complejo, solo es necesario levantar un poco la mano en el lugar que se desea pintar y cerrar para poder realizar la acción de cierre (clic) (ver figura 6.32).

Figura 6.32: Usuario realizando gesto de apertura/cierre de la mano derecha

6.10.4.2 Pruebas de Aceptación de usuarios

Se tomaron cinco usuarios comunes al asar todos sin experiencia en el uso del sensor Kinect, se les pidió a cada uno de ellos realizara la apertura y cierre de la mano derecha o izquierda para poder pintar o seleccionar una de las herramientas del sistema, al ser tan fácil de realizar la acción, todos los usuarios encontraron esta funcionalidad muy básica y sencilla, fácil de aprender, no les tomo mucho tiempo familiarizarse con el mismo. Por lo tanto los usuarios aceptaron este requerimiento sin ningún detalle.

.

6.11 Incremento 10: mejoramiento en la obtención de coordenadas aplicando técnicas de filtrado de Kalman

6.11.1 Análisis

Es de primordial importancia resolver el problema de la poca precisión que se encuentra en la obtención de puntos de las articulaciones por parte del expositor, el problema se centra especialmente cuando no se cuenta con buena iluminación o condiciones esenciales que necesita el sensor para poder hacer el tracking de las articulaciones de la persona, sumado los 30 frames obtenidos cada segundo hace que un movimiento rápido hecho por el usuario no pueda ser plasmado adecuadamente en el graficador virtual programado en el sistema y por lo tanto todas las funciones de cálculo para detectar gestos o herramientas se ven afectadas por el ruido en el ambiente.

En este incremento se programarán las clases y funciones necesarias para implementar la técnica del Filtro de Kalman para predicción y corrección de puntos.

Para la creación del requerimiento de tratamiento de puntos se necesita de la librería OpenCV para utilizar clases para el tratamiento digital de imágenes, debido a la plataforma de desarrollo .Net es necesaria la utilización de EmguCv la cual provee las mismas funciones de OpenCV pero orientado a lenguaje de programación de alto nivel.

6.11.2 Diseño

El diseño para este incremento es completamente a nivel estructura de programación, dentro del archivo ModoPresentador dentro de la solución del proyecto se tiene que crear el método para inicializar e instanciar la clase FiltradoKalman misma que también se creara con su propio archivo.cs. (Ver figura 6.33).

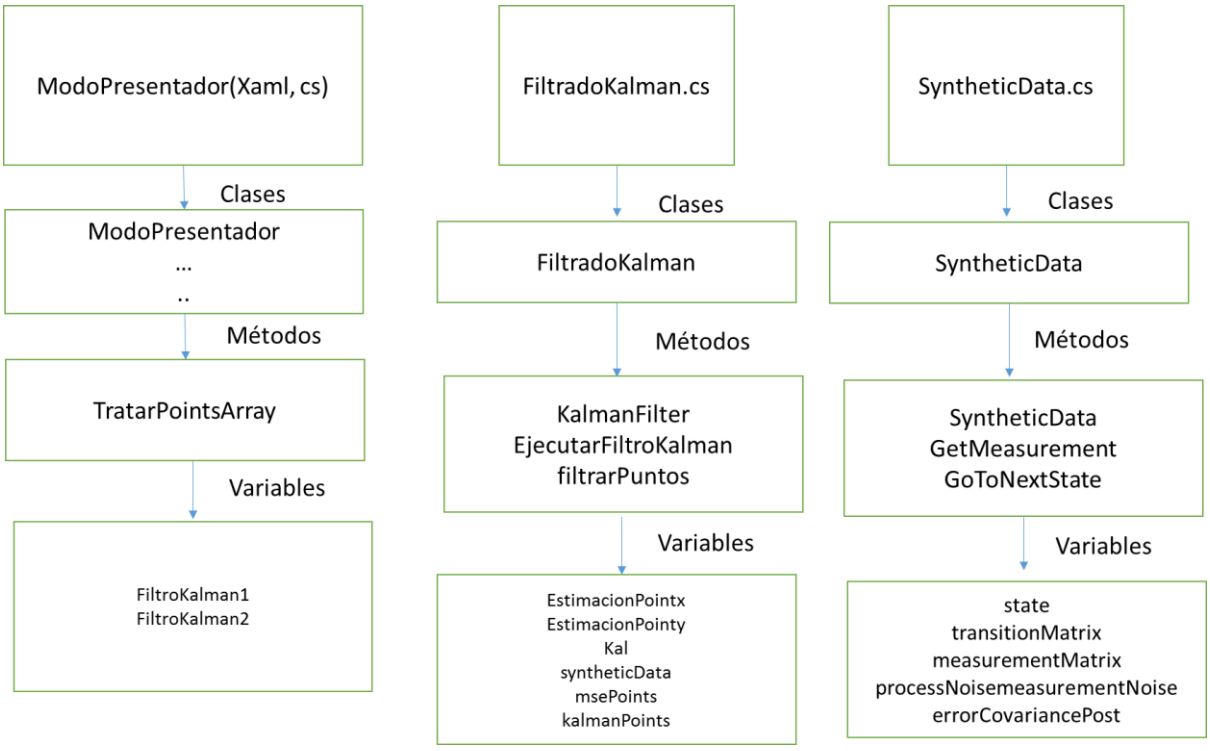

Figura 6.33: Diagrama de estructura del archivo ModoPresentador, FiltradoKalman y SymtheticData.

La clase FiltradoKalman utiliza clases y funciones de EmguCV y de la clase SyntheticData, a partir de las librerías y clases se puede modificar los valores de las matrices de transición y medición que necesita el filtro.

A continuación se plasma el diagrama de secuencia correspondiente a este incremento.

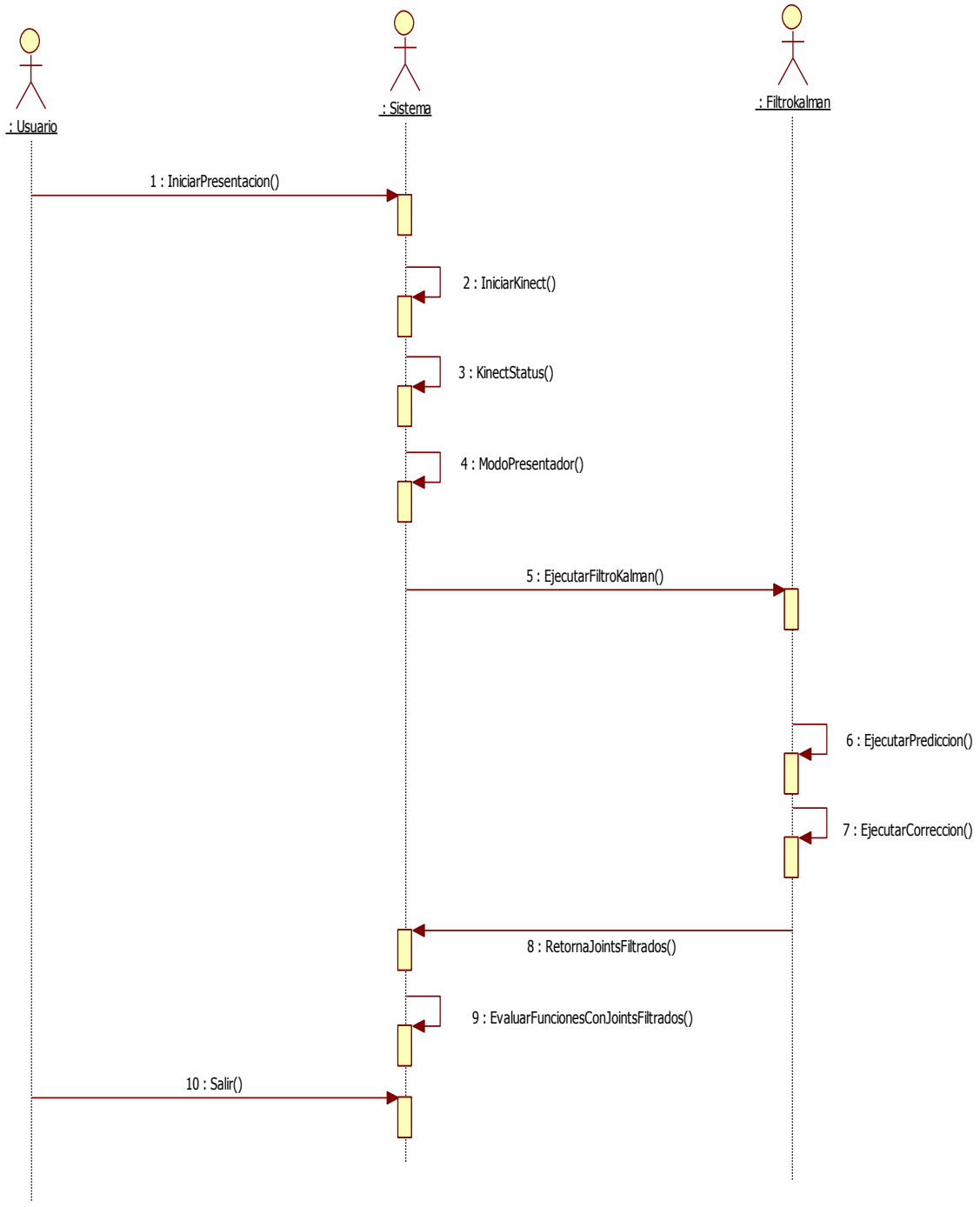

Figura 6.34: Diagrama secuencia para la aplicación del filtro de Kalman.

6.11.3 Código

Codificando los requerimientos para el filtrado primeramente se empieza con la clase SyntheticData en la cual se encuentra la preparación de las matrices esenciales para el filtrado.

Esta clase requiere las referencias de Emgu.CV, Emgu.CV.Structure y Emgu.CV.UI.

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

using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.UI;

namespace PVMCDSMPMA
{
    public class SyntheticData
    {
        public Matrix<float> state;
        public Matrix<float> transitionMatrix;
        public Matrix<float> measurementMatrix;
        public Matrix<float> processNoise;
        public Matrix<float> measurementNoise;
        public Matrix<float> errorCovariancePost;

        public SyntheticData()
        {
            state = new Matrix<float>(4, 1);
            state[0, 0] = 0f; // x-pos
            state[1, 0] = 0f; // y-pos
            state[2, 0] = 0f; // x-velocity
            state[3, 0] = 0f; // y-velocity

            transitionMatrix = new Matrix<float>(new float[,]
            {
            {1, 0, 1, 0},  // x-pos, y-pos, x-velocity, y-velocity
            {0, 1, 0, 1},
            {0, 0, 1, 0},
            {0, 0, 0, 1}
            });

            measurementMatrix = new Matrix<float>(new float[,]
            {
            { 1, 0, 0, 0 },
            { 0, 1, 0, 0 }
            });

            measurementMatrix.SetIdentity();
            processNoise = new Matrix<float>(4, 4); //Linked to the size of the transition matrix
            processNoise.SetIdentity(new MCvScalar(1.0e-4)); //The smaller the value the more resistance to noise
            measurementNoise = new Matrix<float>(2, 2); //Fixed accordiong to input data
            measurementNoise.SetIdentity(new MCvScalar(1.0e-1));
            errorCovariancePost = new Matrix<float>(4, 4); //Linked to the size of the transition matrix
            errorCovariancePost.SetIdentity();
        }
        public Matrix<float> GetMeasurement()
        {
            Matrix<float> measurementNoise = new Matrix<float>(2, 1);
            measurementNoise.SetRandNormal(new MCvScalar(), new MCvScalar(Math.Sqrt(measurementNoise[0, 0])));
            return measurementMatrix * state + measurementNoise;
        }

        public void GoToNextState()
        {
            Matrix<float> processNoise = new Matrix<float>(4, 1);
            processNoise.SetRandNormal(new MCvScalar(), new MCvScalar(processNoise[0, 0]));
            state = transitionMatrix * state + processNoise;
        }
    }
}
```

Posteriormente se creará la Clase FiltradoKalman la cual hace referencia de Emgu.CV, Emgu.CV.Structure, Emgu.CV.UI y Emgu,CV.Util. Dentro de esta clase se crearan las variables necesarias para el tratamiento de los puntos que son enviados del ModoPresentador los cuales vienen tal y como el sensor Kinect lo interpreto.

En el código se definen los métodos que recibirán los puntos X y Y posteriormente se hará el tratamiento de predicción y corrección para retornarlos a la clase principal.

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


using System.Drawing;
using System.Windows.Forms;

using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.UI;
using Emgu.CV.Util;

namespace PVMCDSMPMA
{
    public class FiltradoKalman
    {
        #region Variables
        public float EstimacionPointx, EstimacionPointy;
        #endregion

        #region Kalman Filter and Poins Lists
        PointF[] oup = new PointF[2];
        PointF[] oup2 = new PointF[2];
        public Kalman kal;
        public SyntheticData syntheticData;
        public List<PointF> msePoints;
        public List<PointF> kalmanPoints;
        #endregion

        public void KalmanFilter()
        {
            msePoints = new List<PointF>();
            kalmanPoints = new List<PointF>();
            kal = new Kalman(4, 2, 0);
            syntheticData = new SyntheticData();
            Matrix<float> state = new Matrix<float>
                (
                    new float[]
                    {
                        0.0f,
                        0.0f,
                        0.0f,
                        0.0f
                    }
                 );

            kal.CorrectedState = state;
            kal.TransitionMatrix = syntheticData.transitionMatrix;
            kal.MeasurementNoiseCovariance = syntheticData.measurementNoise;
            kal.ProcessNoiseCovariance = syntheticData.processNoise;
            kal.ErrorCovariancePost = syntheticData.errorCovariancePost;
            kal.MeasurementMatrix = syntheticData.measurementMatrix;
        }

        public void EjecutarFiltroKalman(int px, int py)//Recibe posición de la mano derecha
        {
            PointF inp = new PointF(px, py);
            oup = new PointF[2];
            oup = filtrarPuntos(inp);
        }

        public PointF[] filtrarPuntos(PointF pts)//se aplica el filtrado de kalman
        {
```

```
            syntheticData.state[0, 0] = pts.X;
            syntheticData.state[1, 0] = pts.Y;

            //********* Predicción **************
            Matrix<float> Prediccion = kal.Predict();
            PointF predictPoint = new PointF
                (
                    Prediccion[0, 0], Prediccion[1, 0]
                );

            //********* Corrección **************
            Matrix<float> estimated = kal.Correct
                (
                    syntheticData.GetMeasurement()
                );

            //********* Estimación **************
            PointF estimatedPoint = new PointF
                (
                    estimated[0, 0], estimated[1, 0]
                );
            syntheticData.GoToNextState();

            //********* Resultados **************
            PointF[] results = new PointF[2];
            results[0] = predictPoint;
            results[1] = estimatedPoint;

            EstimacionPointx = estimatedPoint.X;
            EstimacionPointy = estimatedPoint.Y;
            return results;
        }
    }
}
```

Finalmente en la clase principal del modo presentador se instancia las variables de clase del filtrado de kalman.

Se necesitan de variables para cada mano, izquierda y derecha, por lo cual se hace el uso de variables múltiples para tener los vectores X y Y corregidos

```
/*-------------Filtro---------------*/
public FiltradoKalman FiltroKalman1;
public FiltradoKalman FiltroKalman2;
public bool FiltroKalmanActivado = true;
public int puntoFiltradoManoDerecha_x,
           puntoFiltradoManoDerecha_y,
           puntoFiltradoManoIzquierda_x,
           puntoFiltradoManoIzquierda_y = 0;
/*--------------------------------*/
```

La inicialización del filtro se hace en el constructor del Modo Presentador

```
public ModoPresentador()
        {
            this.Loaded += new RoutedEventHandler(Window_Loaded);
            InitializeComponent();
            this.Trasladar = new TranslateTransform();

            this.FiltroKalman1 = new FiltradoKalman(); // Filtro para mano derecha
            this.FiltroKalman1.KalmanFilter();

            this.FiltroKalman2 = new FiltradoKalman(); // Filtro para mano izquierda
            this.FiltroKalman2.KalmanFilter();

            Iniciar_Kinect();
        }
```

Posteriormente se necesita la creación de un nuevo método el cual debe de recibir los parámetros, puntos de la mano derecha e izquierda en los ejes X y Y para poder corregir mediante el filtro.

Una vez que se envían los puntos para su corrección se toma y se envía a los métodos para evaluación de gesto y al graficador de objetos en la pantalla.

```
private void TratarPointsArray(int[] PointsOfKinect, float zManoDerecha, float zManoIzquierda, float zCabeza)
        {
            if (FiltroKalmanActivado)
            {
                FiltroKalman1.EjecutarFiltroKalman(PointsOfKinect[2], PointsOfKinect[3]); //puntos (xy) de mano
Izquierda
                puntoFiltradoManoIzquierda_x = Convert.ToInt32(FiltroKalman1.EstimacionPointx);
                puntoFiltradoManoIzquierda_y = Convert.ToInt32(FiltroKalman1.EstimacionPointy);

                FiltroKalman2.EjecutarFiltroKalman(PointsOfKinect[4], PointsOfKinect[5]); //puntos (xy) de mano
Derecha
                puntoFiltradoManoDerecha_x = Convert.ToInt32(FiltroKalman2.EstimacionPointx);
                puntoFiltradoManoDerecha_y = Convert.ToInt32(FiltroKalman2.EstimacionPointy);
                MoverImagenesCursorManos
                    (
                        PointsOfKinect[0],
                        PointsOfKinect[1],
                        puntoFiltradoManoIzquierda_x,
                        puntoFiltradoManoIzquierda_y,
                        puntoFiltradoManoDerecha_x,
                        puntoFiltradoManoDerecha_y
                    );
                EvaluarGesto
                    (
                        puntoFiltradoManoIzquierda_x,
                        puntoFiltradoManoIzquierda_y,
                        puntoFiltradoManoDerecha_x,
                        puntoFiltradoManoDerecha_y,
                        PointsOfKinect[0],
                        PointsOfKinect[1],
                        PointsOfKinect[6],
                        PointsOfKinect[7],
                        PointsOfKinect[8],
                        PointsOfKinect[9],
                        PointsOfKinect[10],
                        PointsOfKinect[11],
                        PointsOfKinect[12],
                        PointsOfKinect[13],
                        zManoDerecha,
                        zManoIzquierda,
                        zCabeza
                    );
            }
            else
            {
                MoverImagenesCursorManos
                    (
                        PointsOfKinect[0],
                        PointsOfKinect[1],
                        PointsOfKinect[2],
                        PointsOfKinect[3],
                        PointsOfKinect[4],
                        PointsOfKinect[5]
                    );
                //Se aumentaron 20  puntos en x de mano derecha y 10 en y para centrar con e cursor
                //EvaluarGesto(xmIzq + 4, ymIzq + 4, xmDer + 20, ymDer + 10, xh, yh, xhomIzq, yhomIzq, xhomDer,
yhomDer, xhomCen, yhomCen, xcadCen, ycadCen, zManoDerecha, zManoIzquierda, zCabeza);
                //Se aumentaron 20  puntos en x de mano derecha y 10 en y para centrar con e cursor
                EvaluarGesto
                    (
                        PointsOfKinect[2] + 4,
                        PointsOfKinect[3] + 4,
                        PointsOfKinect[4] + 20,
                        PointsOfKinect[5] + 10,
```

```
                    PointsOfKinect[0],
                    PointsOfKinect[1],
                    PointsOfKinect[6],
                    PointsOfKinect[7],
                    PointsOfKinect[8],
                    PointsOfKinect[9],
                    PointsOfKinect[10],
                    PointsOfKinect[11],
                    PointsOfKinect[12],
                    PointsOfKinect[13],
                    zManoDerecha,
                    zManoIzquierda,
                    zCabeza
                );
        }
    }
```

6.11.4 Prueba

6.11.4.1 Pruebas por los desarrolladores

Las pruebas realizadas consisten en comparar la forma en que se pinta la pantalla utilizando la herramienta que requiere más precisión el cual es el lápiz. Para obtener el pintado a mano alzada con la mano derecha sin filtro se agregó en el código una variable global, modificando la variable global en falso se pinta la pantalla con los valores puros obtenidos del Joint mano derecha (Ver figura 6.35).

Figura 6.35: Pintado a mano alzada sin filtro utilizando herramienta lápiz.

Cuando no se utiliza elfiltro el pintado es inestable con picos en la trayectoria de pintado, los picos son los errores que se generan durante el tracking, por otra parte tenemos el pintado estable y suave cuando se utiliza el filtro (Ver figura 6.36).

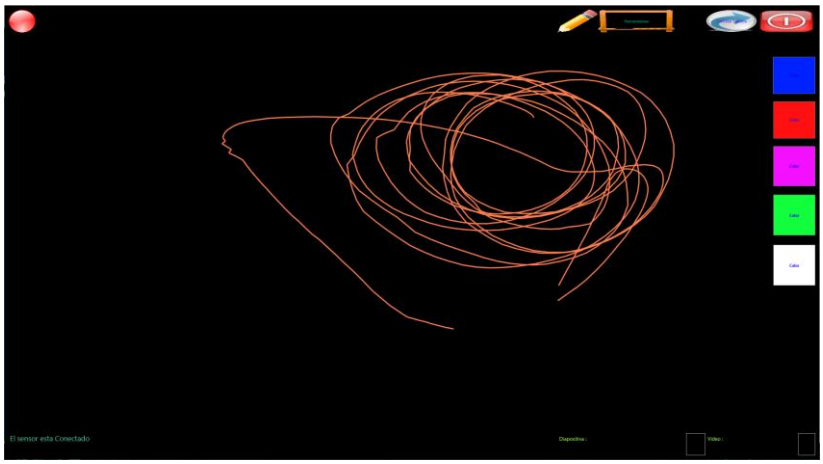

Figura 6.36: Pintado a mano alzada con filtro utilizando herramienta lápiz.

Se realizo una prueba modificando el programa principal para dejar con filtro la mano izquierda y sin filtro la mano derehca, los resultados al momento de comparar son evidentes, el filtrado suavisa y disminuye el ruido durante el tracking, (Ver figura 6.37)

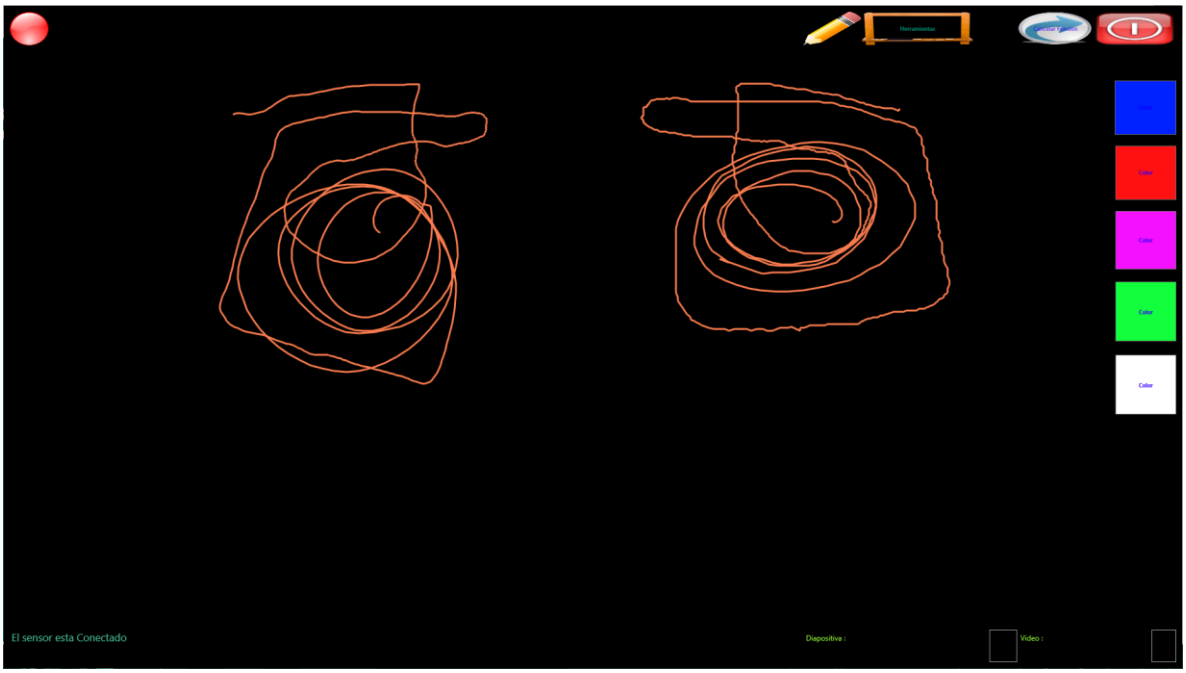

Figura 6.37: Pintado a mano alzada con filtro en mano izquierda y sin filtro en mano derecha utilizando herramienta lápiz.

En las imágenes anteriores e puede observar la gran diferencia entre utilizar y no utilizar el filtro de Kalman. Cuando no se utiliza la técnica de corrección el pintado se torna con cambios fáciles de notar (Ver figura 6.38).

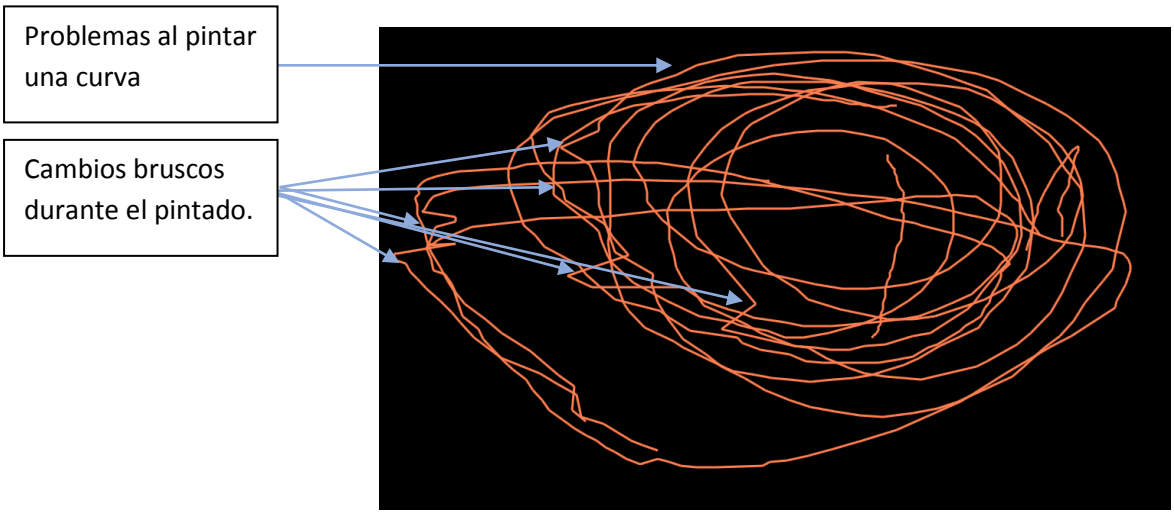

Figura 6.38: Pintado a mano alzada sin filtro en mano derecha, presencia de errores durante el pintado.

Cuando el filtro es utilizado se nota la ausencia de ruido y por lo tanto carencia de errores en el pintado, las curvas se pueden notar nítidas y sin cambios bruscos (Ver figura 6.39)

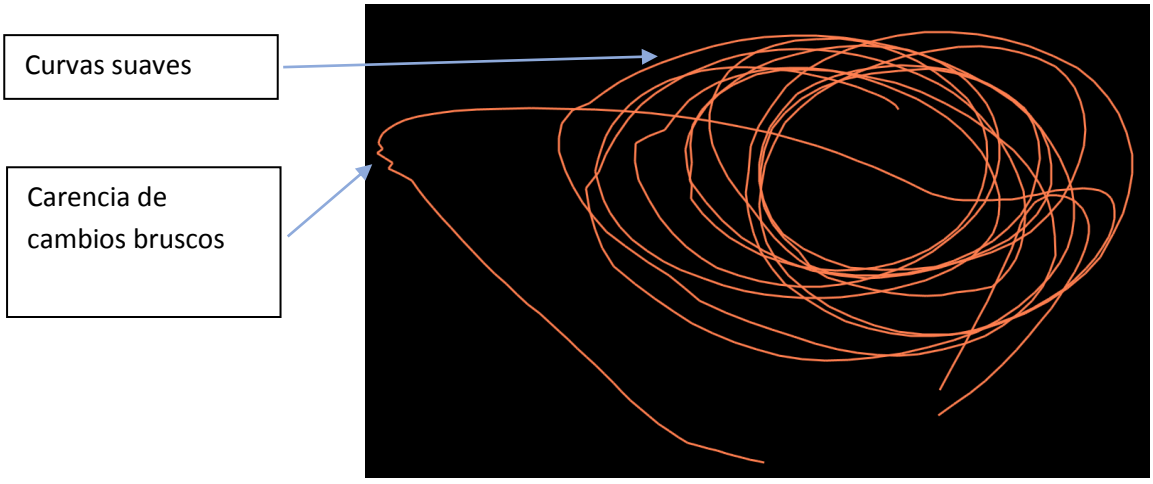

Figura 6.39: Pintado a mano alzada con filtro en mano derecha, ausencia de errores durante el pintado.

6.11.4.2 Pruebas de Aceptación de usuarios

Para poder realizar pruebas de aceptación por parte del usuario, se hizo la selección de cinco estudiantes del Instituto Tecnológico de Tuxtla Gutiérrez, donde se les instruyo para poder utilizar el sistema, cada usuario hizo la prueba con las diferentes funciones de las herramientas, pero la prueba en utilizar el pintado a mano fue la ideal para evaluar el comportamiento de los cursores del usuario.

Todos los usuarios realizaron el pintado con filtro y sin filtro, al final se les pidió una opinión respecto al funcionamiento del mismo, los comentarios fueron los esperados, ellos notaron que con el filtrado podían pintar con una buena precisión y sin el filtro tenían dificultades al realizar trazos, pero para algunas personas no esperaron una gran mejoría, por lo consiguiente la aceptación por los usuarios fue del 90%.

CAPÍTULO VII

IMPLEMENTACIÓN Y PRUEBAS DE LA PRIMERA VERSIÓN DEL SOFTWARE

7.1 Resultados en la utilización del sistema de superficie audiovisual controlado por multigestos kinestésicos

7.1.1 Interacción entre el usuario final y el sistema

Las pruebas se realizaron en el salón de clases de la M.C. Aida Guillermina Cossio Martínez asesora de la tesis en la clase de formulación y evaluación de proyectos de software, de esta clase se seleccionan tres alumnos al azar para hacer una presentación, previamente se les pidió que su presentación la convirtieran en imágenes JPG mismas que se ubicaran en una carpeta.

Al cargar su carpeta de imágenes al sistema no tuvieron ningún problema puesto que las ventanas emergentes son bajo el paradigma Windows, sin embargo durante la presentación ahora con la ayuda del Kinect se observa que para la correcta utilización del sistema los usuarios primero tienen que entrenarse en el ambiente que el software ofrece, es decir que es cuestión de tiempo para que los ponentes puedan manipular de manera correcta su material de exposición.

Sin embargo los resultados fueron satisfactorios, la mayoría de los usuarios se sintieron conectados con el sistema, reconocieron las ventajas, se observó una gran conexión entre los usuarios con el material de exposición, los gestos de avance, retroceso, zoom, grip (apertura y cierre de mano) fueron fáciles de aprender.

7.1.2 Interacción con la implementación del filtrado

El filtrado de Kalman cumple con la parte de filtrar los puntos y corregir el ruido que se genera en el tracking de la persona (Mano izquierda y derecha) respecto al sensor Kinect, cuando el filtro está activo se aprecia la enorme diferencia en el pintado, las líneas y círculos son más suaves y por lo tanto los usuarios se notan satisfechos por la facilidad en la utilización.

7.2. Pruebas de graficación sin filtro y graficación utilizando el filtro de Kalman

Durante las pruebas realizadas con el sensor y el sistema se modifica al programa para escribir en dos archivos un aproximado de 500 puntos del movimiento de la mano derecha, en uno de ellos se escriben los puntos adquiridos sin aplicarles algún filtro, es decir se escribieron tal y cual lo entrega el sensor Kinect, en el segundo archivo se escriben los puntos obtenidos aplicándoles el filtro de Kalman, a continuación se enlistan para realizar la comparativa de dichos resultados.

Puntos del rastreo de las manos sin filtro

No.	X	y
1	807	626
2	807	634
3	807	634
…	…	…
36	790	564
37	790	564
38	792	557
…	…	…
41	822	537
42	834	562
43	840	566
44	840	566
45	840	566
…	…	…
112	662	427
113	665	418
114	662	425
…	…	…
120	715	434
…	…	…
194	751	414
195	751	412
196	733	414
…	…	…
500	709	299

Aplicándoles el filtro de Kalman

No.	x	Y
1	769	596
2	849	666
3	848	667
…	…	…
36	799	593
37	796	584
38	794	575
…	…	…
41	801	551
42	808	549
43	816	549
44	823	549
45	829	550
…	…	…
112	719	426
113	698	422
114	679	421
…	…	…
120	659	429
…	…	…
194	739	421
195	741	419
196	739	417
…	…	…
500	673	284

En las siguientes graficas se observa las diferencias de dispersión de los puntos adquiridos.

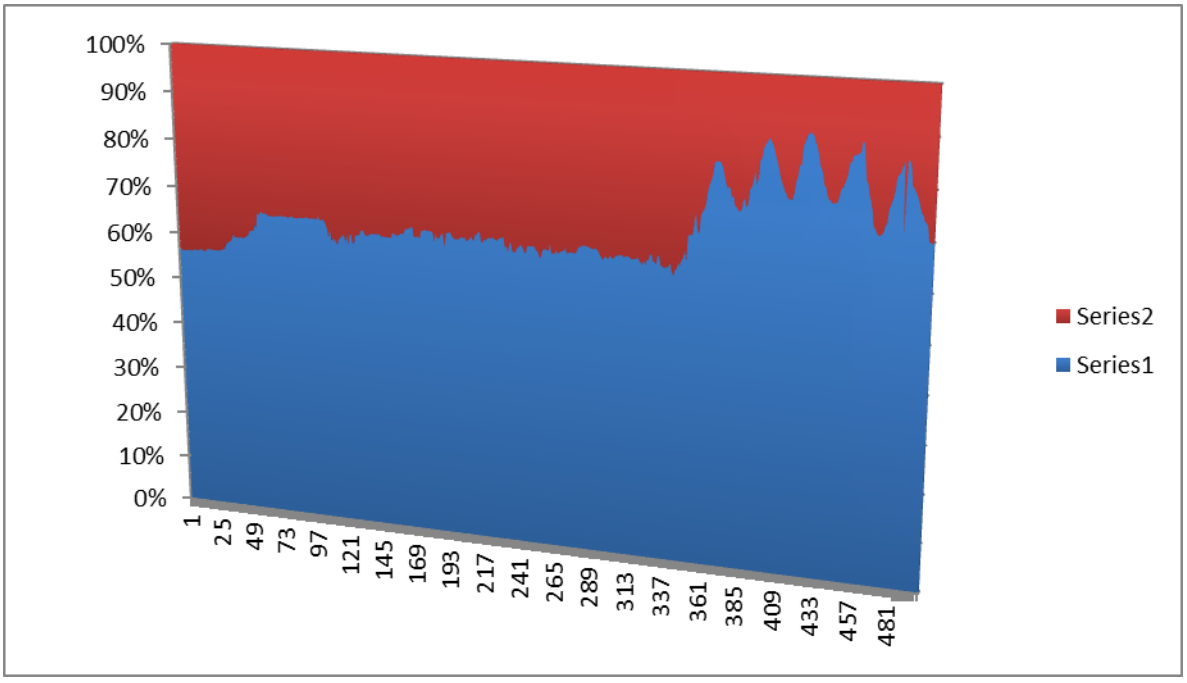

Figura 7.1 grafica 1: Resultados del rastreo de la mano derecha sin filtro.

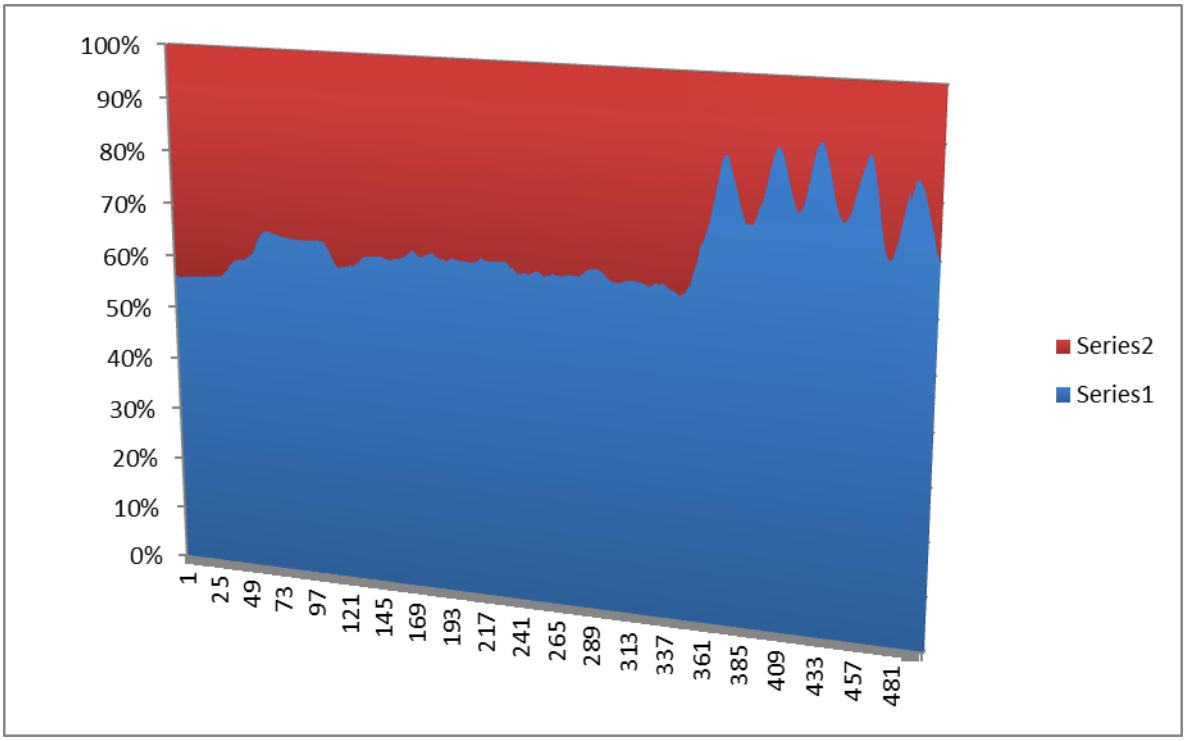

Figura 7.2 grafica 2: Resultados del rastreo de la mano derecha aplicando filtro de Kalman

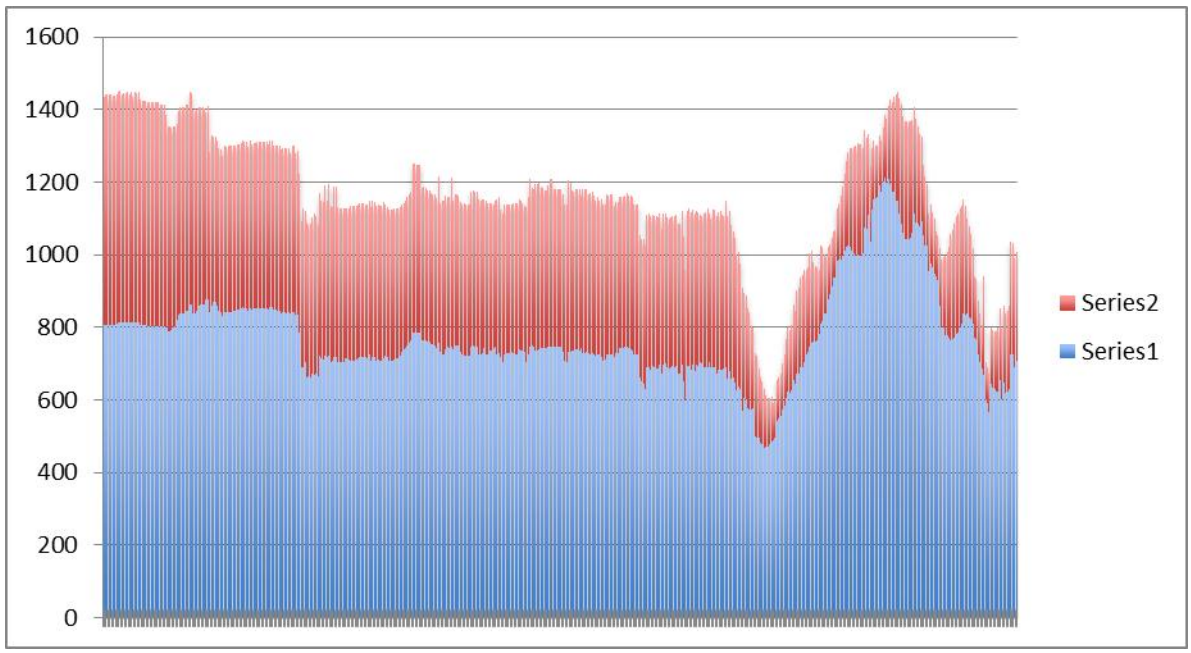

Figura 7.3 grafica 3: Resultado del rastreo de la mano derecha sin filtrar.

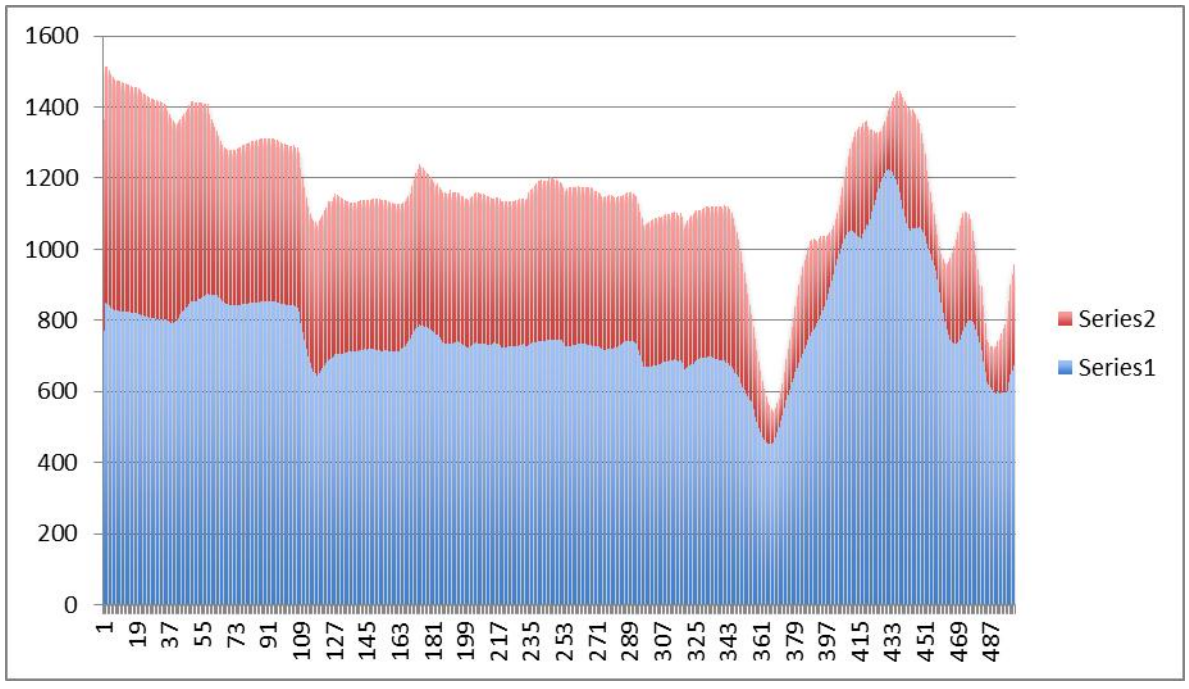

Figura 7.4 grafica 4: Resultados del rastreo de la mano derecha aplicando filtro de Kalman

CAPÍTULO VIII

CONCLUSIÓN

8.1 Conclusión

El presente trabajo de tesis presentó la primera versión de software para la manipulación remota de diapositivas en formato JPG con la ayuda de un sensor Kinect, desarrollado con hardware comercial de bajo costo y software libre, que permite el despliegue y manipulación de un entorno virtual acorde a la percepción del usuario.

El principal objetivo en la construcción del sistema fue implementar técnicas de filtrado que utilizan la predicción y correlación de puntos espaciales durante el tracking para recolección de coordenadas, con ello mejorar el programa "Pizarrón virtual multifuncional controlado a distancia por sensores de movimientos para la presentación de materiales audiovisuales", con función de apertura y cierre de las manos para el control de selección utilizando como sensor de movimientos un Kinect y como visualización de imágenes un video proyector.

Además permite al usuario poder hacer resaltos o pintar en su material de exposición sin la necesidad de instrumentos o dispositivos de control artificial que ocupen las manos del expositor.

Para cumplir con esta meta se construyó un prototipo inicial, luego se desarrolló bajo los incrementos antes mencionados hasta llegar a la versión del software que actualmente cuenta con funciones más interactivas que podrían ser de gran ayuda para tener la atención del público espectador, con el dispositivo Kinect, un proyector y una computadora.

Se concluye de las variables que se tomaron en cuenta para llevar acabo toda la investigación de este trabajo fueron retomadas de tal manera que se mejora el proceso de tracking del usuario con la utilización de la predicción y la correlación, por lo tanto se mejora la precisión en la utilización del sistema en un 90% durante una presentación.

Por lo tanto la hipótesis:

El sistema de superficie audiovisual controlado por multigestos Kinestésicos, se eficientizará en un 99% la precisión del tracking (seguimiento) del usuario durante el proceso de recolección de coordenadas de los Joints o Nodos de la clase Skeleton, filtrando las variaciones y ruidos mediante el uso de predicción y correlación de puntos espaciales.

Se rechaza, si bien la elección del tipo de filtro fue la adecuada y se halla mejorado en gran porcentaje el proceso de seguimiento del usuario, así como la precisión en el uso del sistema, no se alcanzó el porcentaje deseado de eficiencia debido a:

1.- Baja resolución en las cámaras del sensor utilizado.

2.- Se requiere de un equipo de mayor rendimiento al equipo utilizado para realizar todas las pruebas, para mejores resultados de velocidad de procesamiento.

3.- Para alcanzar un nivel óptimo del uso del sistema es necesario que el usuario final se familiarice con el sistema, esto después de varias interacciones con el mismo para entrenarse en la utilización.

CAPÍTULO IX

TRABAJOS FUTUROS

9.1 Trabajos futuros

9.1.1 Mejoramiento en la implementación de técnicas de filtrado

Se puede llegar a mejorar la velocidad de procesamiento en el cálculo de predicción y corrección de puntos, en este trabajo se abusó en hardware al CPU, lo ideal sería utilizar las tarjetas gráficas incorporadas en las nuevas laptops o en los procesadores de última generación, con el fin de reducir la alta demanda en el procesador.

9.1.2 Kinect 2

Cuando se elaboró el sistema de superficie, aún no estaba a la venta el sensor Kinect 2 de Microsoft, pero en el año 2014 fue lanzado a la venta y también el SDK para desarrolladores, en el nuevo sensor se presentan mejoras y nuevas funciones con las que el Kinect 1 carece, el nuevo sensor tiene la capacidad de reconocer los dedos de las manos, teniendo esa nueva función e implementándolo al sistema de superficie audiovisual se podría mejorar en gran medida la interacción del usuario.

Referencias

[1] François, Bérard. (2003). “The Magic Table: Computer-Vision Based Augmentation of a Whiteboard for Creative Meetings”. IEEE International Conference in Computer Vision, I, 1 - 4.

[2] Lee, W. C., Dora Adriana, M. S., Ignacio L. M., Rubén P.G. & Ulises J. M. (2012). “Virtual Board: A low cost Multi Touch Human Computer Interaction System”. **Journal Article** Procedia Technology.

[3] Massimo C., Luis S., & Romolo C. (2012). “Low-cost efficient Interactive Whiteboard”. Consumer Electronics (ICCE), IEEE. 686 – 687.

[4] Paul, C. A. (2012). “NEKO: un tabletop basada en Kinect para manipulación de objetos virtuales 2D y 3D”. Tesis de Maestría, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Unidad Zacatenco Departamento de Computación, México DF.

[5] Sergio Herman, P. B. (2012). “Interfaz de Usuario Natural Usando Kinect”. Tesis de Maestría, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Unidad Zacatenco Departamento de Computación, México DF.

[7] Leandro, P. S. (2011). “Virtual Blackboard: Colour and Human Gestures Motion Tracking”. Proyecto de fin de grado, Escuela Politécnica de Cáceres, Ingeniería Técnica de Telecomunicaciones, Cáceres.

[8] Roberto, M. P. (2012). “detección y extracción de características de la mano humana mediante tecnología primesense”. Proyecto de fin de grado, Universidad Carlos III de Madrid, Departamento de ingeniería de sistemas y automática, Leganés.

[9] Badii, M. H., A. Guillen; E. Cerna; J. Valenzuela & J. Landeros. (2012). “Análisis de Regresión Lineal Simple para Predicción”. International Journal of Good Conscience. 67-81. UANL, San Nicolás, N.L. & UAAAN, Buenvista, Coah., México.

[10] http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/about.aspx

[11] http://msdn.microsoft.com/en-us/vstudio/aa496123.aspx

[12] http://msdn.microsoft.com/es-es/library/kx37x362.aspx

[13] http://msdn.microsoft.com/es-es/library/aa970268(v=vs.110).aspx

[14] http://img.redusers.com/imagenes/ebook/lpcu217/notagratis.pdf

[15] Álvaro R. M. (2009). “Estudio del filtro de partículas aplicado al seguimiento de objetos en secuencias de imágenes”. Proyecto fin de carrera, Universidad Carlos III de Madrid.

[16] Álvaro S. R. (2003). “El filtro de Kalman”. Nota Técnica, Banco central de costa rica, División Económica, Departamento de Investigaciones Económicas.

[17] Patricia R. M. (2003). ”Aplicación del filtro de Kalman al seguimiento de objetos en secuencias de imágenes”. Proyecto de fin de carrera., Universidad Rey Juan Carlos

[18] Alejandro L. Á. (2012) “Detección y Extracción de Características de la mano Humana mediante Tecnología Primesense”. Proyecto fin de grado. Universidad Carlos II de Madrid.

ANEXOS

Diagrama de clases 1/3

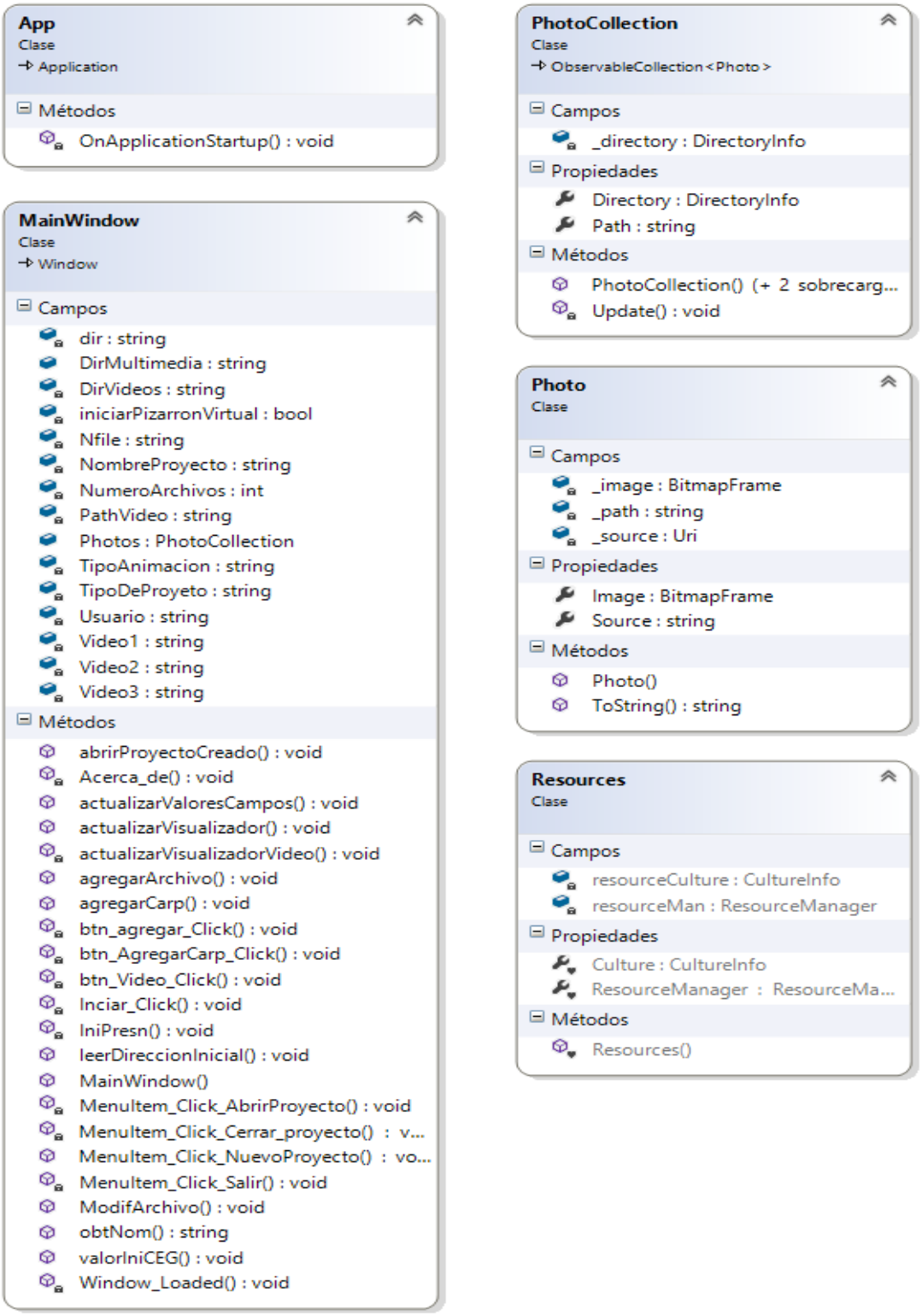

Diagrama de clases 2/3

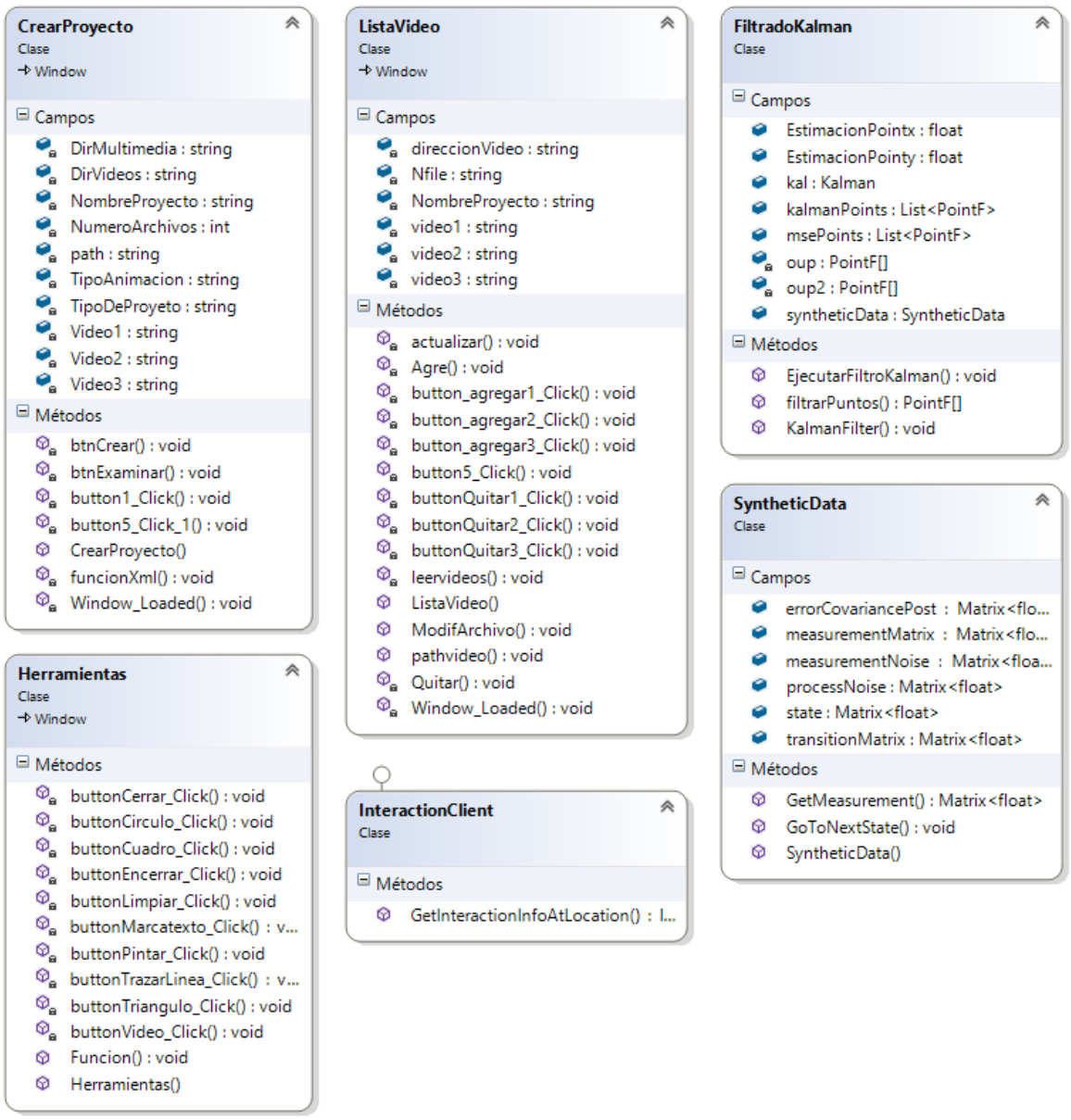

Diagrama de clases 3/3

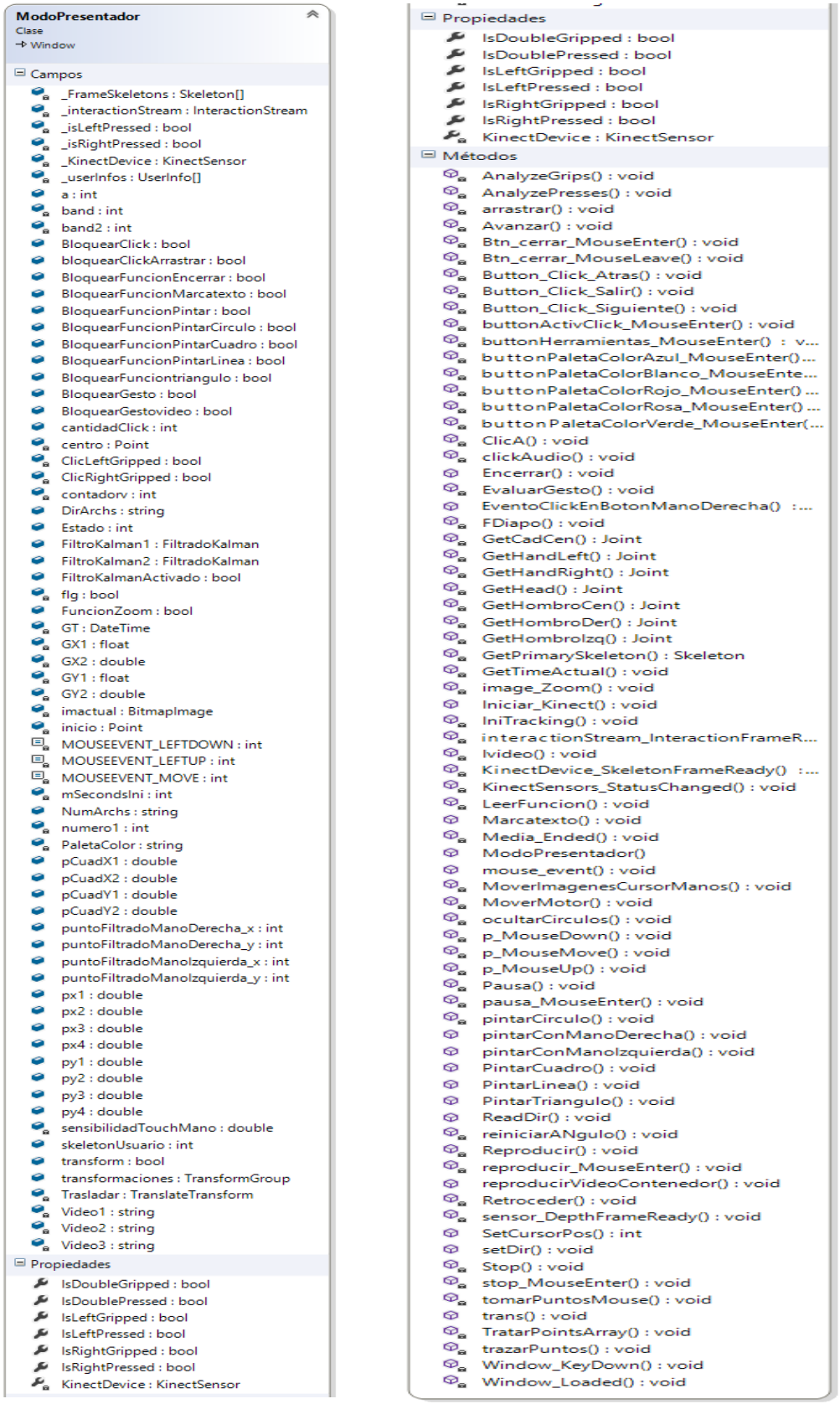

Diagrama de clases completo