UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

RECONOCIMIENTO DE GESTOS DE LA MANO APLICADO AL DESARROLLO DE UNA INTERFAZ KINECT PARA EL MUSEO REGIONAL DE HUAJUAPAN (MureH)

TESIS PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA

León Hernández Edwin Iván

DIRECTOR DE TESIS

Ing. Erik Germán Ramos Pérez

CO-DIRECTOR DE TESIS

Dr. Carlos Alberto Fernández y Fernández

Huajuapan de León, Oaxaca, Febrero 2015

Agradecimientos

Agradezco a **Dios** ser maravilloso que me dio la fuerza y fe para creer lo que me parecía imposible terminar.

A mi **padre Noé León**, a pesar de nuestra distancia física estas siempre junto a mí y aunque nos faltaron muchas cosas por vivir juntos, sé que este momento es tan especial para usted como lo es para mí, sin embargo “sé que en los principios están los fines”.

A mi **madre Rosita**, que me ha enseñado a no desfallecer ni rendirme ante nada y siempre perseverar a través de sus sabios consejos y su amor inmenso, gracias por educarme con valores y ser un gran ejemplo de superación.

A mis hermanos **Yadi**, **Donita** y **Noé** por su paciencia, cariño, por ser mi apoyo y ejemplo para lograr cada una de mis metas, los amo.

A cada uno de los integrantes de las familias **León** y **Hernández**, abuelos, padrinos, tíos, primos, sobrinos son parte de este logro, no me alcanzan las palabras para expresar el orgullo y lo bien que me siento por tener una familia tan asombrosa.

A mi asesor el Ing. Erik Ramos, por su amistad, orientación y ayuda brindada para la realización de esta tesis.

Del mismo modo a mi co-asesor el Dr. Carlos Alberto Fernández y Fernández por su tiempo al revisar la presente.

Agradezco también a mis sinodales Dr. José Aníbal Arias Aguilar, M. C. Verónica Rodríguez López y Lic. Ma. Esperanza Pérez Córdoba Sánchez por su tiempo y contribución al revisar este trabajo.

Aprovecho para agradecer a todos mis maestros que tuve durante mi estancia en la UTM, ya que son parte de mi formación académica.

A mis amigos, por compartir tantas anécdotas, aventuras, experiencias y sobre todo porque sé que siempre podre confiar en ellos.

Dedicatoria

A mi madre:

Con todo mi cariño y amor para la persona que hizo todo en la vida para que yo pudiera lograr mis sueños, por motivarme y darme la mano cuando sentía que el camino se terminaba, a ti mi respeto, amor y agradecimiento infinito.

Contenido

Capítulo 1: Introducción 1
1.1 Objetivos 2
1.1.1Objetivo general 2
1.1.2 Objetivos específicos 2
1.2 Metas 2
1.3 Motivación 3
1.4 Museo Regional de Huajuapan (MureH) 7
1.5 Definición del problema 9
1.6 Hipótesis de la investigación 10
1.7 Justificación del problema 10
1.7.1 Justificación práctica 10
1.8 Delimitación 11
Capítulo 2: Marco Teórico 13
2.1 Interacción Humano-Computadora 13
2.1.1 Usabilidad 16
2.2 Sistemas multimedia 18
2.3 Interfaz Natural de Usuario (NUI) 19
2.4 Kinect 24
2.4.1 Arquitectura del *Kinect* 25
2.4.2 Cámara de Color 29
2.4.3 Cámara de profundidad 31
2.4.4 Rastreo del esqueleto 32
2.5 Reconocimiento de patrones 34
2.6 Definición de gestos de interacción 38
2.7 Metodologías de desarrollo de software 41
2.7.1 Personal Software Process (PSP) 41
Capítulo 3: Diseño del Software 44
3.1 Acercar y Alejar 44
3.2 Deslizar 46
3.3 Rotación 48
3.4 Metodología de desarrollo 51

3.5 Diseño Conceptual 52
3.6 Plantilla de Especificación Operacional (OST) 53
3.6.1. Acercar/Alejar 54
3.6.2 Deslizamiento 55
3.6.3 Rotación 57
3.7 Plantilla Especificación Funcional (FST) 59
3.7.1. Acercar/Alejar 59
3.7.2 Deslizamiento 61
3.7.3 Rotación 62
3.8 Plantilla de Especificación de Estados (SST) 64
3.8.1. Acercar/Alejar 65
3.8.2 Deslizamiento 66
3.8.3 Rotación 68
3.9 Plantilla de Especificación Lógica (LST) 70
3.9.1. Acercar/Alejar 70
3.9.2 Deslizamiento 72
3.9.3 Rotación 75
3.10 Integración del reconocedor de gestos 85
Capítulo 4: Implementación del Software 87
4.1 Requerimientos de Hardware 87
4.2 Requerimientos de Software 88
4.2.1 Direct SDK 88
4.2.2 Microsoft Visual Studio® 2010 Express 89
4.2.3 *Kinect* SDK 90
4.2.4 Microsoft XNA Game Studio 4.0 91
4.3 Construcción del Software 92
4.3.1 Inicializar *Kinect* 92
4.3.2 Obtener el Esqueleto del sensor *Kinect* 94
4.3.3 Cargar modelos 3D 96
4.3.4 Descripción de piezas arqueológicas 99
Capítulo 5: Evaluación y Resultados 100
5.1 Definición de usuarios 101
5.2 Casos de Prueba 102
5.3 Pruebas de rendimiento de MUDI 102

5.3.1 Acercar/Alejar pieza arqueológica 102
5.3.2 Deslizar pieza arqueológica 103
5.3.3 Rotar pieza arqueológica 104
5.4 Pruebas integrales 108
5.4.1 Precondiciones 108
5.4.2 Procedimiento de Prueba: Acercar/Alejar pieza arqueológica 109
5.4.3 Procedimiento de Prueba: Deslizar pieza arqueológica 110
5.3.4 Procedimiento de Prueba: Rotar pieza arqueológica 111
5.4.5 Resultados de las pruebas integrales 111
5.4 Pruebas de usabilidad 115
5.4.1 Contexto de uso 116
5.4.2 Procedimiento de la prueba 116
5.3.5 Ejecución de las pruebas de usabilidad 117
5.3.5 Resultados de las pruebas de usabilidad 121
Capítulo 6: Conclusiones y Trabajo Futuro 125
6.1 Conclusiones 125
6.2 Trabajo Futuro 128
Bibliografía 129
Sitios de Internet 132
Anexo A 136
Manual de Usuario 136

Índice de imágenes

Ilustración 1 Interfaz Multimodal 5
Ilustración 2 Evolución de videojuegos [10] 5
Ilustración 3 Porcentaje de la población oaxaqueña que asiste a los museos al menos una vez al año 8
Ilustración 4 Componentes centrales de HCI 14
Ilustración 5 Usabilidad y Diseño Centrado al Usuario [2] 17
Ilustración 6 Sistemas multimedia 19
Ilustración 7 *Eye tap* de Steve Mann [32] 21
Ilustración 8 Sistema *put-that-there* de Richard A. Bolt [34] 22
Ilustración 9 Interfaz natural de usuario en diferentes consolas de videojuego 23
Ilustración 10 Componentes del *Kinect* 25
Ilustración 11 Diagrama de hardware de *Kinect* [42] 26
Ilustración 12 Componentes de un pixel 29
Ilustración 13 Estructura del vector de bytes de imagen 30
Ilustración 14 Cámara de Color de *Kinect* [43] 30
Ilustración 15 Calculo de la distancia de un objeto con la cámara de profundidad [42] 31
Ilustración 16 Cámara de profundidad del *Kinect* [42] 31
Ilustración 17 Articulaciones del cuerpo detectadas por *Kinect* [42] 33
Ilustración 18 Sistema de reconocimiento de patrones 35
Ilustración 19 Diseño de un sistema de reconocimiento de patrones [47] 36
Ilustración 20 Red neuronal con una única capa oculta [50] 38
Ilustración 21 Etapas de PSP [56] 41
Ilustración 22 Agrandar la pieza arqueológica 45
Ilustración 23 Reducir la pieza arqueológica 45
Ilustración 24 Deslizamiento abajo-arriba 46
Ilustración 25 Deslizamiento arriba-abajo 47
Ilustración 26 Modificación del *Swipe* (deslizamiento) 48
Ilustración 27 Menú de piezas arqueológicas disponibles y seleccionada 48
Ilustración 28 Rotación de pieza arqueológica por un ángulo 48
Ilustración 29 (a) Manos alineadas empezar la rotación. (b) manos desalineadas imposible hacer rotación 49
Ilustración 30 Rotación de pieza arqueológica 49

Ilustración 31 Modificación al gesto de rotación 50
Ilustración 32 Diagrama conceptual del sistema de reconocimiento de gestos 53
Ilustración 33 Tamaño de la ventana de la aplicación 70
Ilustración 34 Comportamiento del *Zoom* dado por el tamaño de la pantalla 71
Ilustración 35 Red de Perceptrones con una única capa oculta para el gesto de rotación 80
Ilustración 36 Integración del reconocedor de gestos 86
Ilustración 37 *Kinect* SDK 1.8 90
Ilustración 38 Inicializar *Kinect* 94
Ilustración 39 Aplicación de reconocimiento de gestos 98
Ilustración 40 Aplicación de reconocimiento de gestos para pruebas de usabilidad .. 98
Ilustración 41 Descripciones de las piezas arqueológicas utilizadas 99
Ilustración 42 Ciclo de construcción de aplicación *Kinect* [12] 100
Ilustración 43 Rendimiento para Acercar/Alejar 103
Ilustración 44 Rendimiento para deslizar 104
Ilustración 45 Gesto válido 105
Ilustración 46 Gesto inválido 105
Ilustración 47 Muestras utilizadas para la fase de entrenamiento y pruebas 106
Ilustración 48 Símbolo para identificar las manos del usuario 109
Ilustración 49 Instrucciones para poder utilizar la aplicación MUDI 109
Ilustración 50 Pruebas integrales para el *zoom* 112
Ilustración 51 Pruebas integrales del deslizamiento 113
Ilustración 52 Pruebas integrales de rotación 114
Ilustración 53 Calificación total al sistema según pruebas con usuarios 115
Ilustración 54 Video de instrucciones MUDI 117
Ilustración 55 Deslizamiento Pruebas de usabilidad a MUDI 118
Ilustración 56 Retroalimentación del *Zoom* aplicación MUDI 119
Ilustración 57 *Zoom* Aplicación MUDI 119
Ilustración 58 Rotación pieza arqueológica MUDI 120
Ilustración 59 Rotar pieza MUDI 120
Ilustración 60 Tareas realizadas en las pruebas de usabilidad 121
Ilustración 61 Métricas de las pruebas de usabilidad 122
Ilustración 62 Diseño final aplicación MUDI *Kinect* no detectado 123
Ilustración 63 Interfaz MUDI final después de pruebas de usabilidad 124

Ilustración 64 Iconos de las manos del sistema MUDI final 124

Índice de tablas

Tabla 1 Características y especificaciones del *Kinect,* parte I [43] 27
Tabla 2 Características y especificaciones del *Kinect*, parte II [43] 28
Tabla 3 Valores de *JointTrackingState* [40] 34
Tabla 4 Ventajas y desventajas de PSP [56] 43
Tabla 5 Diseños de PSP [56] 52
Tabla 6 OST detección de *zoom* 54
Tabla 7 OST *zoom* no detectado 55
Tabla 8 OST *SwipeDowntoUp* o *SwipeUptoDown* detectado 56
Tabla 9 OST Deslizamiento no detectado 57
Tabla 10 OST Rotación detectada 58
Tabla 11 OST Rotación no detectada 59
Tabla 12 FST *Zoom* 60
Tabla 13 FST *SwipeDowntoUp* detectado 61
Tabla 14 FST *SwipeUptoDown* detectado 62
Tabla 15 FST Rotación detectada 64
Tabla 16 FST Modificación a rotación 64
Tabla 17 SST para *zoom* 66
Tabla 18 SST para deslizamiento 68
Tabla 19 SST para Rotación 69
Tabla 20 LST para *zoom* 72
Tabla 21 LST para deslizamiento 75
Tabla 22 LSP para Rotación 78
Tabla 23 LSP Modificado para rotación 79
Tabla 24 Algoritmo de retropropagación 83
Tabla 25 Algoritmo de Propagación 85
Tabla 26 Código C# inicializar *Kinect* 93
Tabla 27 Obtener el esqueleto que proporciona *Kinect* 95
Tabla 28 Dibujado de modelo 3D en pantalla 97
Tabla 29 Muestras del gesto rotación 104
Tabla 30 Ejemplo de 10 corridas para un ejemplo de gesto 106
Tabla 31 Resultados para el clasificador 107
Tabla 32 Calificaciones de las tareas realizadas por los usuarios en pruebas integrales 112

Tabla 33 Usuarios que han visitado el MureH 121

Capítulo 1: Introducción

Las interfaces de usuario fueron creadas para responder a las necesidades de los usuarios otorgando facilidad, amabilidad y confianza durante la interacción con las computadoras. La evolución de las interfaces de usuario ha evolucionado a pasos agigantados, desde utilizar tarjetas perforadas hasta las interfaces gráficas de usuario, cuyo uso ha sido el más predominante en la última década, trasladándose a dispositivos electrónicos que son parte de la vida cotidiana.

Por ende este desarrollo dentro de la computación, permitió la evolución de las interfaces gráficas de usuario a interfaces capaces de adaptarse al usuario. Con esto surge la investigación de la Interacción Natural (por sus siglas en inglés NI).

Los videojuegos se enfocan en la NI para mejorar la experiencia del usuario, claro ejemplo es el *Kinect*, el cuál por medio de la consola Xbox 360 permite la interacción con videojuegos usando solamente movimientos del cuerpo y voz, eliminando así el uso de controles.

El propósito de esta tesis es diseñar, construir y experimentar una interfaz natural establecida en la interpretación de gestos con las manos, utilizando el *Kinect*, así mismo la implementación de técnicas para reconocer dichos gestos y pruebas al sistema con usuarios reales.

1.1 Objetivos

1.1.1 Objetivo general

Desarrollar una interfaz basada en *Kinect* que detecte y reconozca movimientos de las manos, con los podrán interactuar los visitantes con modelos 3D de piezas arqueológicas del MureH (Museo Regional de Huajuapan), tales gestos permitirán rotar, escalar y desplazar la misma.

1.1.2 Objetivos específicos

- Estudiar el estado del arte con respecto al uso de reconocimiento de gestos.
- Estudiar las API´s de desarrollo del *Kinect* que existen actualmente.
- Implementar algoritmos de reconocimiento de gestos, tales como: reconocimiento de patrones, k-vecinos más cercanos y k-medidas.
- Desarrollar la aplicación que reconozca los gestos de las manos.
- Pruebas de factibilidad y experiencia de usuario a dichas interfaces.
- Identificar las características de los movimientos de las manos para su tratamiento.

1.2 Metas

- Verificar el comportamiento de algoritmos para el reconocimiento de gestos con la mano.
- El *Kinect* permitirá incrementar el fomento a la cultura en primera instancia a las escuelas de los alrededores de Huajuapan de León Oaxaca, mediante el registro de visitas.

1.3 Motivación

El uso de la computadora se ha vuelto en un instrumento básico para muchas actividades de la vida cotidiana, entre las que destacan la educación, el trabajo, medicina y en especial el entretenimiento. Gran parte de esto se debe a la facilidad que se presenta gracias a la interfaz que la computadora presenta al usuario para realizar sus tareas y procesar la información [1].

La computadora en la última década se ha vuelto en un instrumento necesario e indispensable en todos los ámbitos sociales y educativos, por tanto, esta se encuentra al alcance de todos. La interfaz es el primer acercamiento entre el usuario y la maquina por consiguiente esta debe ser fácil de usar y sobre todo intuitiva [2].

El crecimiento acelerado computacional desde su origen se ha concentrado primordialmente en el desarrollo de computadoras que sean de mayor potencia, ajustándose a las necesidades de los usuarios, con diseños más reducidos y con una gran diversidad de funcionalidades, la constante es la investigación para la mejora de la interacción, mediante el uso de interfaces de usuario más amigables, por medio del uso del lenguaje natural[1] que se adapten al usuario y no viceversa.

La Interacción Humano Computadora (HCI por sus siglas en inglés) [3] es un área de investigación que se enfoca en el estudio de la interacción entre el humano y la computadora. Del lado humano se concentra en la mejora de los mecanismos de aprendizaje e interacción, mientras que del lado de las computadoras se concentra en la visualización y procesamiento de la información.

[1] El lenguaje natural es el que se utiliza como medio de comunicación habitual entre los humanos, está compuesto por una sintaxis y obedece a los principios de economía y optimalidad.

En la actualidad la interfaz más utilizada es la Interfaz Gráfica de Usuario (GUI por sus siglas en inglés), que sigue el paradigma VIMA (Ventana, Icono, Menú y Apuntador) [4] [5], pues la abstracción de interacción que utiliza con iconos que representen elementos que el humano está acostumbrado a utilizar cotidianamente, y la metáfora del escritorio como contenedor y base de la interacción. Esto ha posicionado a la GUI, junto con el ratón y el teclado, como la interfaz predominante desde los años 90 [6].

La búsqueda por la mejora de la interacción ha llevado a extender las capacidades de la GUI. Actualmente los dispositivos móviles y los videojuegos son los ejemplos más comunes, puesto que la tecnología, recursos y conocimiento avanzan a pasos gigantescos.

Dado que los dispositivos móviles, al contrario de las computadoras convencionales, cuentan con recursos limitados y que el tamaño de la pantalla es reducido (a excepción de las tabletas) evitando con esto visualizar más de una aplicación a la vez.

Por esta razón se optó por extender la interfaz de usuario con el desarrollo de interfaces multimodales [7] que combinan diversos mecanismos de entrada, mediante el uso de acelerómetros, pantallas multitáctiles, reconocimiento de gestos usando las cámaras incorporadas, reconocimiento de comandos de voz, con mecanismos de salida, sonido, vibración, y visualización en pantalla. Tal combinación (ver Ilustración 1) brinda una experiencia multimodal [6], llevando a un nivel de interacción completamente distinto al que se alcanza en una computadora.

Mediante la interfaz multimodal, el usuario puede interactuar con aplicaciones que no se visualizan en la pantalla, pero se están ejecutando. Por ejemplo, es posible

cambiar de canción en un reproductor de audio al agitar el dispositivo, mientras que en la pantalla se lee el correo electrónico, se juega o se envía un mensaje de texto. Todas estas mejoras en las interfaces han abierto una puerta para desarrollar aplicaciones completamente distintas a las que existían hace pocos años.

Ilustración 1 Interfaz Multimodal

Uno de los elementos precisos en el éxito de un videojuego es la experiencia de juego a través de una interfaz intuitiva, fácil de recordar y de aprender para el usuario.

Los controles fueron evolucionando a lo largo del tiempo **(**ver Ilustración 2) para ser más ergonómicos y brindar retroalimentación al usuario de lo que ocurre en el juego (mediante vibración o botones sensibles a la presión).

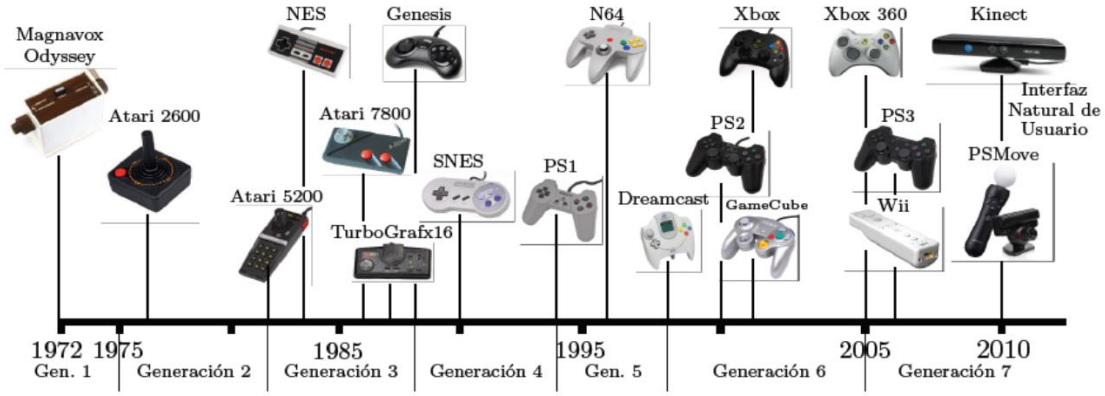


Ilustración 2 Evolución de videojuegos [10]

Recientemente el desarrollo de controles como los de la consola Wii[2] de Nintendo y el sistema sixaxis[3] de la consola Play Station 3 de Sony, brindaron un nivel de interacción más natural. Lo anterior ya que permite poder utilizar los movimientos que se realizan con el control para interactuar con el videojuego, gracias a la incorporación de sistemas de acelerómetros en los controles.

El sensor *Kinect* para la consola Xbox 360 de Microsoft llevó la interacción entre el usuario y el videojuego a un nivel superior en el que no se requieren mandos (controles), al permitir al usuario interactuar con el videojuego mediante movimientos del cuerpo.

La idea de usar el cuerpo como medio de interacción con la computadora y el desarrollo tecnológico dio pauta a la investigación de la Interacción Natural (NI) [8] y la Interfaz Natural de Usuario (por sus siglas en inglés NUI) [9], como sucesora de la GUI. La NUI se basa en el uso de habilidades del cuerpo como movimientos y gesticulaciones, para ser utilizadas como medio de interacción con la computadora, eliminando el uso de dispositivos de entrada. Es decir, el objetivo de las NUI es lograr que el usuario interactúe con la computadora como si ésta fuese otra persona, sin que el usuario esté consciente de utilizar un dispositivo.

El desarrollo del sensor *Kinect* ha creado a la consola Xbox 360 de Microsoft como el primer NUI comercial, que a diferencia de Wii y PSMove, elimina completamente la necesidad de usar el control de la consola al reemplazarlo por el movimiento del cuerpo y comandos de voz. La interacción con el videojuego se logra al visualizar

---

[2] **Wii** es una videoconsola producida por Nintendo y estrenada el 19 de noviembre de 2006 en Norteamérica y el 8 de diciembre del mismo año en Europa. Perteneciente a la séptima generación de consolas, es la sucesora directa de Nintendo GameCube y actualmente sigue compitiendo con la Xbox 360 de Microsoft y la PlayStation 3 de Sony.

[3] Es la primera versión del mando de la PlayStation 3, cuyo nombre significa literalmente «seis ejes» haciendo referencia a los seis (*six*) ejes (*axis*) de detección de movimiento. Fue el controlador oficial para la PlayStation 3 antes de la salida del Dualshock3

los movimientos del usuario como movimientos de un personaje en la pantalla y la realización de acciones cuando el usuario efectúa un movimiento especifico o pronuncia un comando de voz, los cuales son reconocidos por el sistema e interpretados por el videojuego.

Kinect[4] no sólo renovó el concepto de interacción con un videojuego, sino que ha permitido que el desarrollo de las NUI sea fortalecido, gracias al esfuerzo de Open NI [11], y posteriormente Microsoft [12], para extender el uso del *Kinect*, de la consola Xbox 360 hacia las computadoras convencionales e incluso arquitecturas ARM[5].

1.4 Museo Regional de Huajuapan (MureH)

El crecimiento rápido de la tecnología ayuda a que cada vez más personas tengan acceso al uso de dispositivos móviles, computadoras, tabletas. Pero también genera necesidades y mayores expectativas, como puede ser en el ámbito político, económico, social, educativo y cultural.

Tal es el caso de los museos en los cuales la tecnología está tomando gran auge para incentivar y difundir la historia, y cultura resguardada en dichos museos [13].

En México existen 1,227 museos registrados, de los cuales 50 se localizan en el estado de Oaxaca [14]. El estado de Oaxaca, el cual es rico en cultura y con diversas actividades culturales, en los museos solo tiene un porcentaje de 9.2 % de visitas al año [15] tal como se observa en Ilustración 3.

---

[4] El sensor Kinect salió a la venta en Noviembre de 2010 en E. U. A

[5] Arquitecturas ARM: arquitectura RISC (Reduced Instruction Set Computer=Ordenador con Conjunto Reducido de Instrucciones) de 32 bits desarrollada por ARM Holdings. Se llamó Advanced RISC Machine.

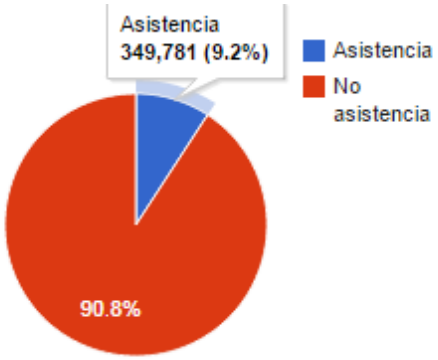


Ilustración 3 Porcentaje de la población oaxaqueña que asiste a los museos al menos una vez al año

El número de visitas al año es muy bajo, este dato muestra que los oaxaqueños tienen poco interés en conocer el patrimonio cultural. Tal vez la falta de organización, financiamiento, promoción e innovación de los museos sea la causa de tener pocos visitantes.

Sin embargo en términos generales existen museos interactivos en gran parte del país, el ejemplo quizá más conocido de este tipo de museos es el “Papalote” en la Cd. de México. El cuál muestra un dato interesante, tan solo en el 2008 se registró un total de 2,404,776 visitantes que representan poco más del 5 % del total de visitas a nivel nacional [16].

Lo anterior muestra que los museos interactivos son más atractivos que los que carecen de uso de tecnología [17]. Por tanto es probable que en los museos convencionales la visita no se tan atractiva, dado que la información se presenta en una hoja de información junto con la pieza expuesta y no existe interacción alguna con la misma dentro del recorrido.

En Huajuapan de León, Oaxaca se localiza el Museo Regional de Huajuapan (MureH). El MureH es el espacio en donde se recupera, conserva y difunde la arqueología, etnografía e historia de la cultura mixteca, y en donde se presentan a

la población diferentes exposiciones y actividades culturales con el objetivo de contribuir a incrementar el acervo cultural de la población [18].

De acuerdo a lo anterior y motivado por el desarrollo rápido de las tendencias tecnológicas en el uso de realidad aumentada, en el control mediante el movimiento de las manos y la visualización e interacción en los museos, este proyecto intenta satisfacer una necesidad de innovación para el acercamiento y vinculación con la cultura a través de una interfaz de promoción de la cultura mixteca en el MureH de una forma agradable, estructurada y fácil de usar.

1.5 Definición del problema

Aproximadamente en el MureH se reciben poco más de 500 visitantes cada mes, los cuales son mayormente alumnos de los diferentes niveles educativos y en menor proporción pero no menos importantes turistas locales y nacionales según datos otorgados por el patronato del museo en el año 2013. En base a la poca afluencia mencionada anteriormente en el MureH, se proyecta aumentar la atención de los visitantes, además de promover la cultura de una forma entretenida y divertida, mediante interfaces interactivas que promuevan y ayuden a cambiar radicalmente la forma de interactuar con las piezas arqueológicas del mismo.

Se pretende incorporar a los visitantes del museo al uso de tecnología virtual, usando los movimientos corporales para manipular objetos con simples gestos o movimientos de las manos. Es decir, se crearán interfaces amigables para el usuario, que pueda visualizar una pieza arqueológica, rotarla, ampliarla y reducirla, usando el reconocedor de gestos de las manos basadas en el esqueleto humano.

La experiencia debe ser agradable para causar impacto en su mente logrando que pueda recordar aspectos culturales.

Kinect es un sensor desarrollado por Microsoft que ofrece un complemento a la computadora como son: los ojos, los oídos y el cerebro. Con *Kinect*, las empresas

y los desarrolladores están creando aplicaciones que permiten a sus clientes interactuar de forma natural con las computadoras simplemente gesticulando y hablando [19].

Es por ello, que se creará el reconocedor de gestos apoyado con e *Kinect,* el cual nos ofrece interacción con lenguaje natural y los objetos virtuales en este caso piezas arqueológicas existentes en el MureH.

1.6 Hipótesis de la investigación

- Hipótesis H0: ¿El uso de librerías de *Kinect*, ayuda en el proceso de reconocimiento de patrones?
- Hipótesis H1: ¿El uso de la aplicación para mejorar la interacción con las piezas arqueológicas, incrementará el número de visitas al museo?

1.7 Justificación del problema

La investigación consiste en el estudio, implementación y evaluación de algoritmos y técnicas de reconocimiento de patrones, así como el manejo de los algoritmos presentados por el *Kinect*, a fin de contribuir a la comunidad sobre la historia de Huajuapan, debido que Huajuapan cuenta con una zona arqueológica llamada “Cerro de las Minas”.

1.7.1 Justificación práctica

El MureH necesita de la implementación de un sistema tecnológico interactivo que sirva como medio de apoyo para atraer a los turistas nacionales y extranjeros, estudiantes, pero principalmente a la comunidad de Huajuapan.

Acercándolos a descubrir un poco de la cultura mixteca e historia que guarda el “Cerro de las Minas” y sus alrededores mediante las piezas arqueológicas

existentes en el museo, de una forma entretenida, y divertida usando la tecnología, principalmente despertando en la población el interés por su cultura.

En el ámbito académico, el presente tema de investigación aporta diversos conocimientos generados sobre el uso del *Kinect* (un dispositivo electrónico capaz de reconocer los movimientos del esqueleto humano), en el cual se hicieron estudios sobre el área de reconocimiento de patrones, y se emplearan diversos algoritmos que permitan generar el reconocedor de gestos.

Por tal motivo es conveniente hacer un reconocedor de gestos basado en los movimientos del esqueleto humano, en base a las necesidades y expectativas tecnológicas que demanda el museo usando tecnologías de información (TI).

1.8 Delimitación

El MureH necesita de una aplicación capaz de reconocer gestos con las manos que involucran las siguientes funciones: visualización del modelo tridimensional, rotación y *zoom* sobre las piezas arqueológicas existentes en el MureH.

Este software y las interfaces realizadas serán solo para el uso de piezas arqueológicas existentes en el MureH, tomando en cuenta que se pretende involucrar a la gente de la comunidad y los turistas que visiten dicho lugar; con el fin de crear vínculos de tecnología, cultura y conocimiento.

Alcances:

- Construir una interfaz amigable y fácil de usar.
- Presentar información importante y verídica respecto a la pieza.
- Libertad por parte del usuario para manipular la pieza arqueológica a su criterio.
- Visualización del modelo tridimensional de la pieza arqueológica.

- Reconocer gestos manuales comúnmente usados para la manipulación de objetos.

Limitaciones:

- El sistema únicamente abarcara las piezas arqueológicas.
- Se requiere de un espacio de 3m x 3m dedicado solo al uso del sistema.
- Solo se usaran 10 piezas arqueológicas.
- Solo podrá interactuar un usuario a la vez con el sistema.
- Las modelos de piezas arqueológicas deben ser archivos con extensión fbx.
- Los archivos de audio deben ser en formato mp3 nativo de Windows.

Capítulo 2: Marco Teórico

2.1 Interacción Humano-Computadora

La interfaz de usuario es el espacio más importante de cualquier aplicación y/o sistema debido a que es ahí en donde se desarrolla la interacción entre el humano y la computadora.

El objetivo primordial de las interfaces es abstraer el funcionamiento de una computadora en una orden o un conjunto de ellas, con el propósito de evitarle al usuario la necesidad de saber el funcionamiento interno de dicho sistema. La interacción puede ser definida como el intercambio de información entre dos o más participantes activos [20].

En términos de programación, un elemento en la interacción es un sistema computacional y el otro el usuario. Como condición importante, tiene que existir en todo momento la retroalimentación en cuanto a la interacción se refiere entre el sistema y el usuario. Si faltara la retroalimentación el sistema podría funcionar de forma errónea según la concepción del usuario y viceversa, pues no podrían autorregularse.

Como se mencionó en [3] y [21], la HCI[6] es el estudio y practica del uso de las computadoras; vista como la capacidad que tiene la computadora para ser usada mediante el entendimiento, la creación del software y otras tecnologías que serían

[6] Interacción Humano Computadora (HCI): En términos generales, podríamos decir que es la disciplina que estudia el intercambio de información mediante software entre las personas y las computadoras. Esta se encarga del diseño, evaluación e implementación de los aparatos tecnológicos interactivos, estudiando el mayor número de casos que les pueda llegar a afectar [21].

útiles para los usuarios, a fin de encontrar una solución efectiva a una situación que involucre el manejo y manipulación de información.

Faulkner, remarca que esta área se enfoca en buscar y entender el razonamiento que siguen los usuarios al realizar tareas específicas, observando el comportamiento y estructura del sistema a fin de estructurar y construir un sistema que le facilite al usuario el cumplimiento de dichas tareas [22].

El usuario en esta área se vuelve en el actor principal, pues se requiere entender su comportamiento durante el proceso de construcción del software para comprender los procesos y capacidades que están asociados a las tareas que desempeñan. Para ello, se requiere desarrollar conocimientos sobre memoria, cognición, oído, tacto y habilidades motrices [23].

El uso del área de HCI, para elaborar sistemas computacionales genera un grado de eficiencia mediante la reducción del número de errores ocasionados por el usuario durante el desarrollo de una tarea. De la misma manera se logra, eficiencia en términos del tiempo que el usuario se lleva en realizar su tarea y con esto se percibe una satisfacción de uso.

HCI se compone de cuatro elementos indispensables [24] (ver Ilustración 4):

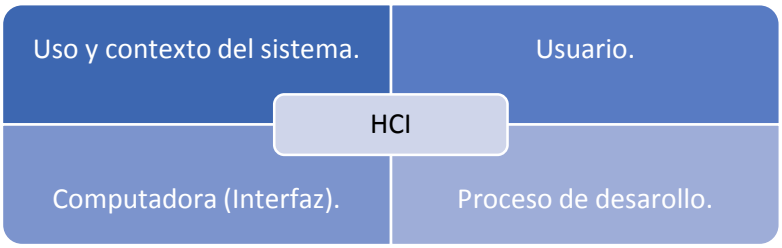


Ilustración 4 Componentes centrales de HCI

El diagrama muestra la importancia de HCI para el diseño de sistemas de calidad, por medio de la aplicación sistemática de conocimientos sobre las características y metas humanas (proceso de desarrollo), sobre las capacidades y limitaciones de la tecnología (computadora). Relacionándose entre sí con aspectos sociales, organizacionales y físicos del entorno de trabajo del usuario (uso y contexto).

HCI es el diseño e implementación de sistemas informáticos interactivos para que los usuarios puedan llevar a cabo el proceso de interacción. Incluye sistemas de escritorio, así como los sistemas integrados en diferentes dispositivos [24].

El éxito de una tecnología simplemente es el resultado de la facilidad con la que el usuario puede interactuar con la misma. Si la interfaz es pobre o difícil de usar, el usuario simplemente ignorará el producto o la tecnología.

Para lograr esto se toman los conceptos más importantes en HCI [25] [2]:

- Funcionalidad: Conjunto de características que hacen que la interfaz sea practica y útil.
- Facilidad de uso: es cuando un usuario utiliza las funciones del sistema fácilmente, correctamente y con claridad.

HCI es una gama de posibilidades en diversas áreas de conocimiento en las cuales el usuario es parte esencial de estudio tales áreas son [21]: psicología social y organizacional, ingeniería, diseño, psicología cognitiva, ciencias de la computación, educación, inteligencia artificial, lingüística, filosofía y sociología, entre otras.

Para un usuario lo importante es realizar sus actividades de manera eficiente, sin tener que pasar por un entrenamiento previo sobre sistemas operativos y comandos básicos, medios y avanzados, para poder realizar algo simple, es por tal motivo que se está usando esta área como apoyo para realizar la interfaz de lenguaje natural.

De ahí la necesidad de elementos como la usabilidad y amigabilidad de un programa de cómputo proporcionando un fácil acceso a los usuarios. La amigabilidad de una interfaz se puede evaluar a consideración de factores como el tiempo de aprendizaje, la velocidad en el desempeño, la tasa de error en el uso, el tiempo del usuario, la manera en que el usuario opera el sistema y la satisfacción subjetiva que se experimenta al utilizar una computadora [26].

En cuanto a la usabilidad del programa dependerá de un adecuado diseño de la Interfaz Gráfica del Usuario (GUI), así como de la complejidad de la aplicación que se diseña.

Por tal motivo se retoman factores en cuanto al diseño de interfaces, pues es conveniente que las interfaces que componen al sistema sean diseñadas tomando en cuenta las necesidades y capacidades de las personas para quienes son destinadas. Sin olvidar que es importante conocer factores personales y sociales que involucren el comportamiento.

Por ello se ha considerado que el sistema a diseñar para el MureH debe ser del todo amigable, entendible y fácil de usar capaz de llamar la atención de los usuarios o visitantes del mismo.

2.1.1 Usabilidad

Hablar de usabilidad es referirse a una medida de que tan “usable” es el uso de un sistema, por tanto algunos autores [2] [27] definen a la usabilidad como:

"La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso, además de ser intuitivo y fácil de usar"

Para el desarrollo de la prueba de usabilidad es necesario definir usuarios, tareas y contexto de uso (ver Ilustración) [28].

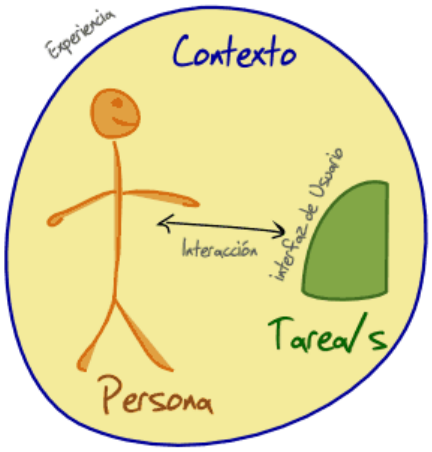


Ilustración 5 Usabilidad y Diseño Centrado al Usuario [2]

Las pruebas de usabilidad son diferentes a las pruebas de caja negra, caja blanca o unitarias realizadas habitualmente por los desarrolladores. Tampoco se trata de hacer una demostración para los usuarios o demostrar que la aplicación y/o sistema funciona, lo que se busca y se desea lograr es aprender más acerca de los usuarios, como piensan, reaccionan a lo que ocurre en el sistema [29].

Las pruebas de usabilidad nunca se llevan a cabo con el cliente ya que se busca encontrar un grado de satisfacción y confianza en los usuarios para garantizar que el sistema realmente será usado como se planifico, y de la mejor manera. Para ello se debe tomar en cuenta a los usuarios, tareas y el contexto que tendrá el sistema.

2.2 Sistemas multimedia

Existe una gran variedad de sistemas computacionales, pues al igual que los dispositivos electrónicos aumentan de forma acelerada, es evidente que la forma de interacción y la experiencia del usuario va revolucionando del mismo modo.

Pues dentro de toda la gama de sistemas existentes es posible encontrar aquellos que mediante la mezcla de texto, imágenes, animaciones, sonido y video brindan información según la entrada de datos y el tipo de aplicación con el que se esté trabajando. Toda esta migración de nuevas tecnológicas y dispositivos electrónicos conlleva a que con mayor frecuencia, diferentes empresas estén apostando por sistemas que puedan ofrecer una experiencia más grata para el usuario final [26].

Cabe destacar que el rendimiento actual de los equipos de cómputo, han permitido a las interfaces tener un comportamiento visual permitiendo a los usuarios poder observar el cambio de los datos con mayor practicidad, acercándolos más a la interacción con los usuarios. Siendo así que se utilicen muchas de ellas para el entrenamiento de personal en diferentes áreas como: la milicia o medicina.

Del mismo modo las interfaces son de gran impacto en el mundo del entretenimiento, un gran ejemplo son los videojuegos. Este constante cambio ha llevado a incorporar diferentes medios de visualización, integrándolos con ambientes de inmersión cada vez más sofisticados.

Hoy en día es muy común encontrar variadas aplicaciones multimedia (ver Ilustración 6) tal como: un reloj digital, celulares, consola de videojuegos e interacciones cotidianas para el ámbito industrial, maquinaria y servicios médicos [21].

Ilustración 6 Sistemas multimedia

El diseño del sistema y el contexto de uso son una parte crucial del desarrollo del sistema. De tal modo que las instrucciones enviadas por el usuario sean comprendidas y ejecutadas por el sistema a su vez que la respuesta que esté ofrezca sea entendible y de fácil apreciación para el usuario. La interfaz a desarrollar ocupa comunicación bidireccional, es decir de usuario a sistema y viceversa.

De acuerdo a las condiciones que establece HCI es necesario que el entorno sea amigable con colores agradables, retroalimentación a manera de interacción y buena calidad de las imágenes, de tal modo que se proporcione un ambiente en el que el usuario se sienta en confianza para interactuar con el sistema.

Actualmente existe un trabajo que propone un sistema multimedia utilizando una máquina de estados finita, capaz de interactuar con un sistema médico a distancia por medio del uso de un *Kinect* como el único dispositivo de entrada. El sistema permite al usuario utilizar las manos y brazos como medio de interacción 2D. [30]

2.3 Interfaz Natural de Usuario (NUI)

Desde el inicio de la computación hasta la actualidad hemos estado inmersos en diversos avances tecnológicos que han hecho que las computadoras estén al

alcance de todos. Las nuevas computadoras evolucionan con pasos titánicos, cambiando la forma de interactuar, por tanto nos hemos tenido que adaptar a las nuevas interfaces que nos ofrecen las computadoras, en lugar que estas sean las que se adapten a nosotros.

La NUI [9] surge motivada por el estudio e investigación de la forma en que se interactúa con el ambiente de trabajo en el contexto computacional y las interfaces naturales de usuario [6], las cuales buscan remplazar las interfaces de usuario actuales, por mecanismos que nos permitan interactuar con la computadora de la misma forma en la que interactuamos diariamente con nuestro entorno, es decir, volviendo transparentes los dispositivos de entrada y facilitando el aprendizaje al usuario para interactuar mediante la interfaz.

Las NUI's son aquellas en las que se interactúa con un sistema y/o aplicación, sin utilizar dispositivos de entrada, como ratón, teclado, lápiz óptico [1]. En lugar de esto se utilizan las manos o las yemas de los dedos.

El término natural se refiere, en términos computacionales, a lo que le es más fácil o común hacer a una persona en su vida diaria, en la cual en todo momento existe retroalimentación inmediata y continua al momento de interactuar con algún objeto o persona. Un ejemplo puede verse claramente en un automóvil, el conductor sabe que tiene que girar el volante para dar vuelta y al mismo tiempo debe acelerar para poder avanzar.

La NUI se enfoca en la construcción de una interfaz adaptada a nuestras capacidades, que permita concentrar nuestra atención en realizar una tarea, en lugar de como realizar una tarea. La interacción se alcanza de forma natural al

mejorar la experiencia de usuario con una interfaz de tal forma que se vuelve invisible al usuario en un proceso de aprendizaje por repetición.

La NUI tiene una historia de desarrollos y experimentos que se remonta a los años 80's en la búsqueda de cambiar la forma de interactuar con las computadoras. Steve Mann fue uno de los primeros en tratar de cambiar la forma de interactuar con las computadoras mediante estrategias de interfaces de usuario con el mundo real y propuso el nombre de NUI para este tipo de interfaces [31], (su trabajo se muestra en Ilustración 7.)

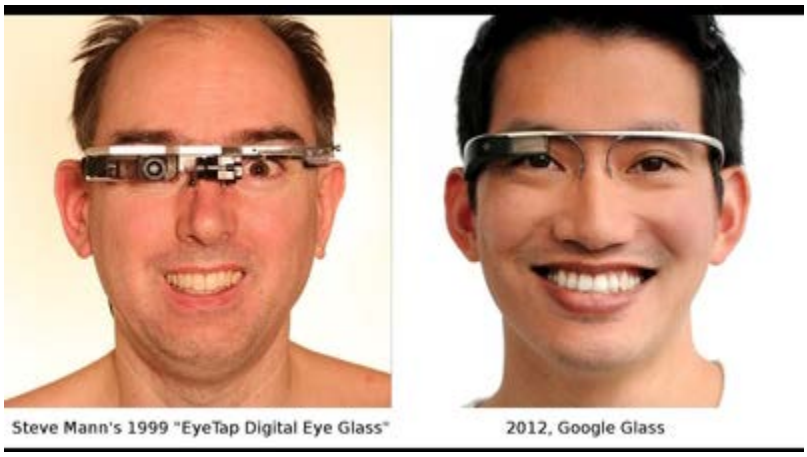


Ilustración 7 *Eye tap* de Steve Mann [32]

Uno de los primeros sistemas que utilizaron el movimiento del cuerpo y la voz para interactuar con una aplicación es *Put-that-there* [33] desarrollado por Richard A. Bolt en 1980 (ver Ilustración 8).

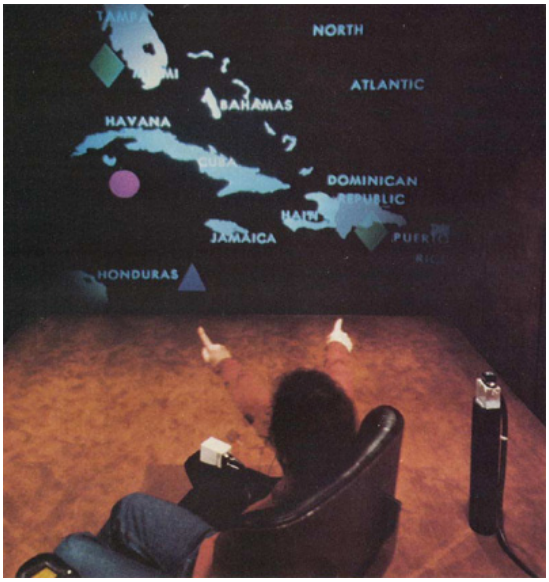


Ilustración 8 Sistema *put-that-there* de Richard A. Bolt [34]

Algunos esfuerzos sobre las investigaciones de la NUI, se ha tratado de volverla portable, entre estas ideas se encuentra la llamada “tecnología de sexto sentido”[7] de Pranav Mistry [34], con el reemplazo del monitor por un picoproyector[8], con el que es posible proyectar la interfaz en cualquier lugar, e interactuar con dicha proyección mediante el reconocimiento de gestos con dedales[9] de color en los dedos a través de una cámara.

La utilización de NUI se ha expandido en los últimos años debido a su integración con el mercado de las consolas de juegos de video, tales como: *Kinect* de Microsoft para la consola XBOX 360, WiiMote en el caso de Nintendo y también en cierta

7 El "Sexto Sentido" trata de aprovechar los avances en la tecnología de la miniaturización y pretende dar un paso adelante respecto a la telefonía móvil, que ya permite disponer de información relevante en movimiento gracias a la conexión a Internet. El objetivo de este proyecto es que el acceso sea más ágil y que los datos dejen de estar confinados en una pantalla, ya sea de ordenador o de terminal móvil.

8 Un picoproyector es una tecnología que implementa el uso de un proyector de imágenes en un dispositivo portátil.

9 Un dedal inteligente es capaz de detectar su posición con precisión en tres dimensiones y de responder a una serie de gestos preprogramados que permiten al usuario interactuar con objetos en un mundo virtual tridimensional.

medida PSMove de Sony (ver Ilustración 9). Incluso se utiliza en la nueva generación de Videoconsolas mostradas en profundidad en la E3 2013, como en la versión actualizada de Microsoft *Kinect* para XBOX One.

A)Kinect para XBOX 360. B) Wii Nintendo. C) PS MOVE SONY

Ilustración 9 Interfaz natural de usuario en diferentes consolas de videojuego

Diseñar interfaces NUI en forma adecuada significa involucrarse con el usuario de manera más directa. Esto implica, analizar y evaluar las diferentes formas en que el usuario puede interactuar con el sistema, y tomar decisiones de diseño que logren un producto que al usarlo genere una experiencia agradable. Esta consideración refleja que es más complejo diseñar interfaces NUI que diseñar las tradicionales interfaces GUI, lo cual obliga a tomar en cuenta una serie de consideraciones y criterios para lograr una aplicación exitosa, enfatizando en una mejor calidad de experiencia del usuario, con respecto a la usabilidad.

Para que una interfaz sea considerada natural, debe cumplir con las siguientes consideraciones [35] [36]

- Crear una experiencia que dé la sensación de ser una extensión del cuerpo.
- Crear una experiencia que le sea natural tanto a usuarios expertos, como a los usuarios nuevos.
- Crear una experiencia que sea auténtica al medio.

- Crear una interfaz de usuario que considere el contexto, tomando en cuenta las metáforas correctas, indicaciones visuales, realimentación y métodos de entrada y salida.
- Evitar caer en la trampa de copiar los paradigmas de interfaz de usuario existentes.

2.4 *Kinect*

Kinect es una tecnología de control para la consola Xbox 360 [37] de Microsoft [38] que permite interactuar con un videojuego sin la necesidad de usar un control; utiliza la tecnología de cámara de profundidad desarrollada por la compañía israelí *PrimeSense*, actualmente absorbida por Apple en noviembre de 2013 [39] la cual permite a *Kinect* "ver" la escena (al usuario y su entorno en tres dimensiones) en tiempo real. Las imágenes de profundidad se obtienen con *Kinect* y posteriormente se procesan por software en la consola para interpretar la escena, detectar personas y rastrear sus movimientos (rastreo del esqueleto). *Kinect* provee captura tridimensional del cuerpo en un esqueleto virtual formado por un conjunto de puntos 3D (articulaciones) relacionados a las partes del cuerpo. Esta abstracción permite detectar gestos del cuerpo al comparar los valores de las articulaciones virtuales contra los valores propuestos de un gesto específico.

La tecnología empleada por este dispositivo, se basa en el entendimiento del movimiento del cuerpo, es decir, la computadora como receptora debe interpretar y entender cada uno de los movimientos del usuario antes que el videojuego pueda responder. Por tanto el procesamiento debe ser en tiempo real para que el usuario no perciba una demora durante la interacción.

2.4.1 Arquitectura del *Kinect*

La estructura del *Kinect* (ver Ilustración 10) se compone de una cámara de profundidad, una cámara RGB y una matriz de 4 micrófonos, que aíslan el ruido ambiental permitiendo utilizar el *Kinect* como un dispositivo, capaz de recibir comandos de voz a la consola de XBOX [40]. Lo integra también una base motorizada, controlada por un acelerómetro de 3 ejes, que le permite rotar horizontalmente para ajustar el campo de visión de las cámaras para visualizar el cuerpo completo del usuario.

Ilustración 10 Componentes del *Kinect*

El *Kinect* internamente contiene un chip PS1080 [41], desarrollado por *PrimeSense*, esté es el encargado de la generación y sincronización de las imágenes de profundidad e imágenes de color (ver Ilustración 11). El chip ejecuta los algoritmos de adquisición de imágenes de profundidad de la escena a partir del sistema de proyección de un patrón de puntos infrarrojos llamado *LightCoding*.

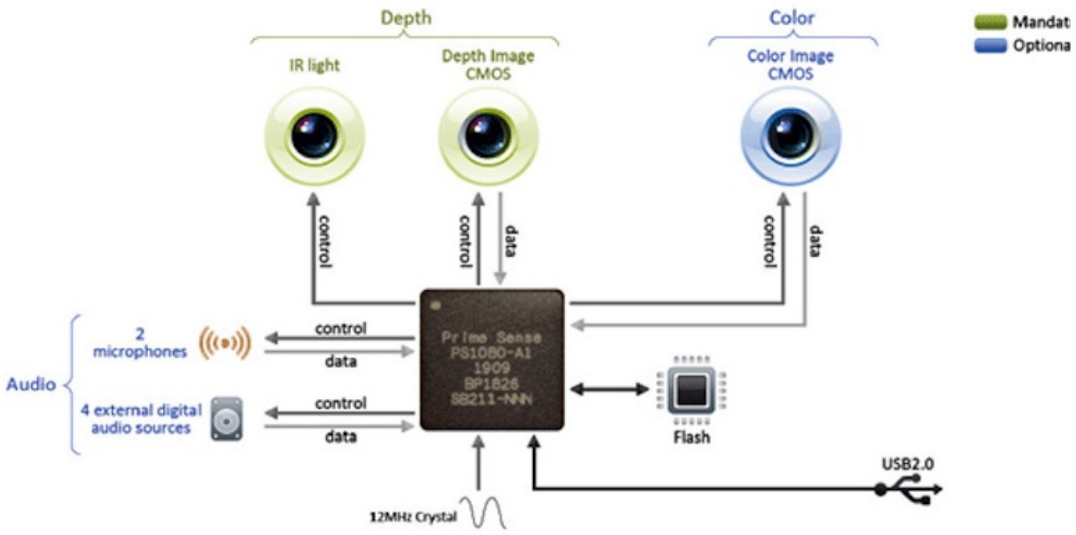


Ilustración 11 Diagrama de hardware de *Kinect* [42]

Kinect funciona mediante un esquema maestro-esclavo, donde el maestro es la computadora y el esclavo es el *Kinect*. El chip genera y mantiene en memoria los cuadros de imagen de profundidad y color a una velocidad de 30 *fms* (cuadros por segundo). El acceso a estos datos se realiza a través de un puerto USB 2.0.

Las Tablas 1 y 2 muestran las características y las especificaciones propias de los sensores tanto para los flujos de color y profundidad como para los sensores de la matriz de micrófonos [43].

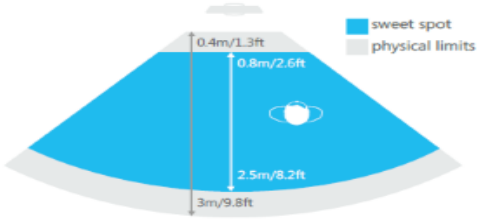
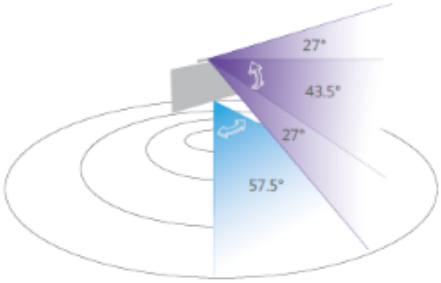
Elemento del sensor	Rango de especificación	Imagen
Flujo de color y profundidad	0.4m a 3m	
Ángulo de visión	Horizontal: 57.5 grados. Vertical: 43.5 grados. Inclinación de -27 a +27 grados de abajo a arriba.	
Esqueleto completo	Reconoce 20 articulaciones.	

Tabla 1 Características y especificaciones del *Kinect,* parte I [43]

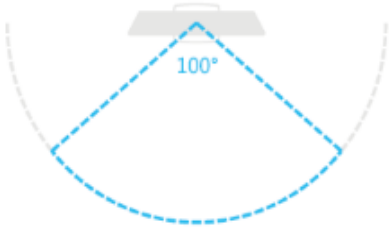
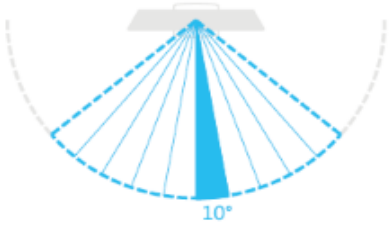
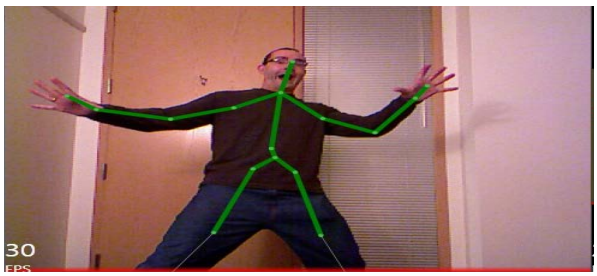
Elemento del sensor	Rango de especificación	Imagen
Audio	Detecta la entrada de audio entre -50 y +50 grados enfrente del sensor *Kinect*.	
Micrófonos	El conjunto de micrófonos puede ser señalado en incremento de 10 grados dentro del rango de 100 grados. Puede ser usado en una sola dirección para eliminar el ruido por completo.	
Tasa de cuadros	30 cuadros por segundo (FPS).	
Resolución de flujo de profundidad	QVGA (320x240) y (640x480).	
Resolución del flujo de color	VGA: (640 x480) y (1280x960)	

Tabla 2 Características y especificaciones del *Kinect*, parte II [43]

2.4.2 Cámara de Color

Kinect utiliza una cámara RGB que obtiene imágenes en color y dos cámaras de infrarrojos para medir la distancia a la que se encuentran los elementos que están en el campo de visión [40]. Con el SDK de *Kinect* para Windows podemos obtener los datos de las cámaras y trabajar con ellos para utilizarlos en nuestras aplicaciones.

Para entender esta codificación hay que saber primero como se estructura una imagen. Una imagen se compone de un conjunto de píxeles. Cada pixel de la imagen tiene 4 componentes, (ver Ilustración 12) que representan los valores de los colores rojo (R), verde (G) y azul (B) más una componente que corresponde con el valor de transparencia (a, alfa) en el caso de imágenes RGBa, o un valor vacío, si es de tipo RGB [44].

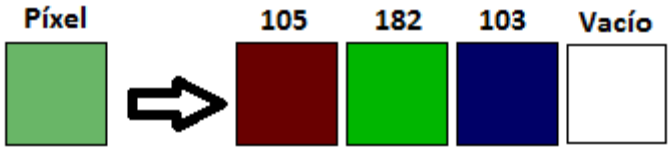


Ilustración 12 Componentes de un pixel

Cada componente del píxel tiene un valor decimal de 0 a 255 lo que corresponde a un byte, (ver Ilustración 13). De esta forma el vector de bytes que obtenemos del sensor, en el caso de la cámara RGB es una representación de esos píxeles organizados de arriba abajo y de izquierda a derecha donde los 4 primeros elementos del vector serán los valores rojo, verde, azul y alfa del píxel de arriba a la izquierda, mientras que los 4 últimos serán del píxel de abajo a la derecha.

Ilustración 13 Estructura del vector de bytes de imagen

Al tener 2 cámaras de infrarrojos cada píxel se corresponde con 2 bytes en el vector, siendo éstos el valor de la distancia de ese píxel a cada cámara. La organización de los píxeles es la misma que con la cámara RGB, los 2 primeros bytes son la distancia del píxel de la posición de arriba a la izquierda al sensor y los 2 últimos son del píxel de abajo a la derecha.

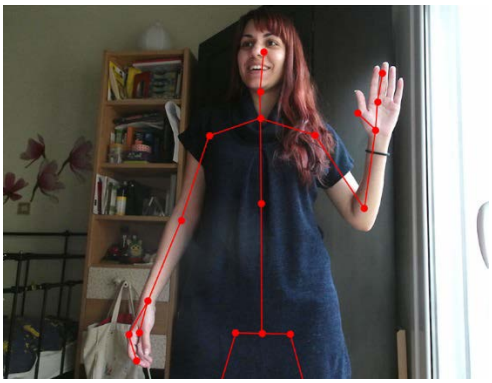


Ilustración 14 Cámara de Color de *Kinect* [43]

En la Ilustración 14 se muestra la cámara de color con una resolución de 640 x 480, con 30fps, en la cual se dibujan las articulaciones obtenidas.

2.4.3 Cámara de profundidad

La cámara de profundidad se compone por la cámara infrarroja y el proyector infrarrojo de luz estructurado.

La cámara de profundidad utiliza una tecnología de codificación de luz llamada *LightCoding* [40] que actúa como un escáner 3D para realizar una construcción tridimensional de la escena.

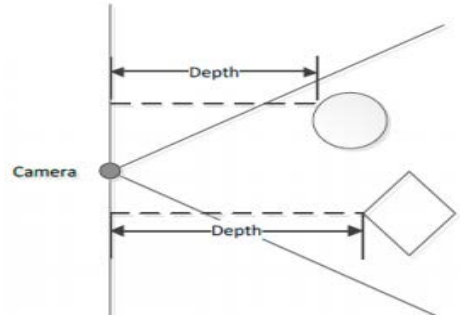


Ilustración 15 Calculo de la distancia de un objeto con la cámara de profundidad [42]

Los datos de la cámara de profundidad (ver Ilustración 15) sirven para calcular la distancia a la que se encuentra un pixel del sensor. Cada pixel representa la distancia en milímetros desde el plano de la cámara al objeto más cercano, como se muestra en la Ilustración 16. Un valor de pixel de 0 indica que el sensor no encontró ningún objeto dentro de su rango en ese lugar.

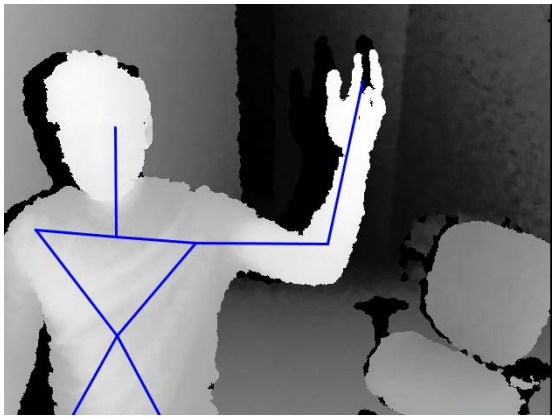


Ilustración 16 Cámara de profundidad del *Kinect* [42]

La Ilustración 16 muestra la cámara de profundidad, con el rastreo del esqueleto dibujando las articulaciones proporcionadas por las librerías de *Kinect*.

2.4.4 Rastreo del esqueleto

Kinect se enfoca en reconocer el esqueleto humano como materia prima, para obtener el esqueleto es necesario procesar las imágenes de profundidad obtenidas de *Kinect* para detectar formas humanas e identificar partes del cuerpo del usuario en el conjunto de imágenes procesadas.

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.

Cada *joint* o articulación tiene una posición, indicada en la Ilustración 17. Con coordenadas <X, Y, Z>. El valor de las coordenadas “X” y “Y” son relativos al espacio del esqueleto, el cual no es igual al obtenido de la cámara de profundidad [42].

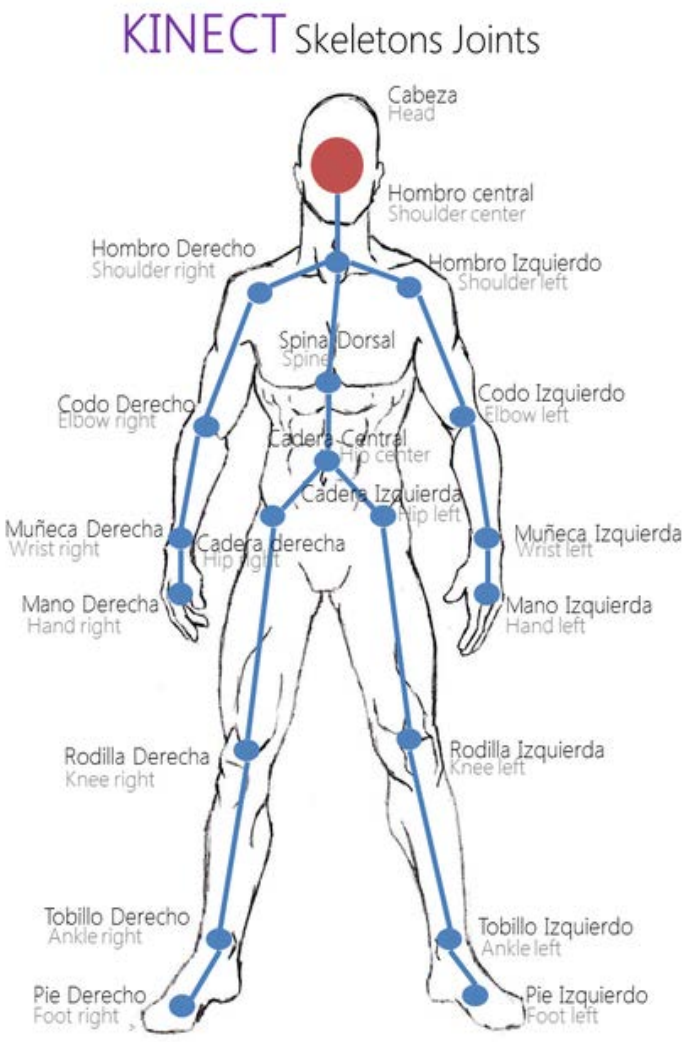


Ilustración 17 Articulaciones del cuerpo detectadas por *Kinect* [42]

Los estados para cada articulación se encuentran en una propiedad *JointTrackingState*, la cual nos indica si se ha detectado una articulación y como se detectó, tal como se muestra en la Tabla 3.

JointTrackingState	**Significado**
Inferred	Es esqueleto no puede ver la articulación en la cámara de profundidad, pero ha calculado una distancia para esa articulación.
NotTracked	La posición del *Joint* es indeterminada, la posición es un vector *zero*.
Tracked	La articulación es detectada y es monitoreada.

Tabla 3 Valores de *JointTrackingState* [40]

2.5 Reconocimiento de patrones

Reconocimiento de patrones es una rama de la Inteligencia Artificial que se ocupa de los procesos sobre Ingeniería, Computación y Matemáticas relacionados con objetos físicos o abstractos, con el propósito de extraer información que permita establecer propiedades de entre conjuntos de dichos objetos [45].

Duda, define el reconocimiento de patrones como [46]:

"El acto de analizar los datos originales y tomar una acción basada en la categoría del patrón"

Bishop [47] lo concibe como:

"El estudio automático de las regularidades de los dato a través del uso de algoritmos para después tomar acciones tal como clasificar los datos en diferentes categorías."

El esquema de un sistema de reconocimiento de patrones consta de varias etapas relacionadas entre sí (los resultados de una etapa pueden modificar los parámetros de etapas anteriores).

La Ilustración 18 muestra un esquema general de un sistema de reconocimiento de patrones en el cual el sensor tiene como propósito proporcionar una representación factible de los elementos del universo a ser clasificados.

Ilustración 18 Sistema de reconocimiento de patrones

Para construir un sistema de reconocimiento de patrones se deben contemplar los siguientes aspectos:

- Sensado: Detección de señales físicas y conversión de estas a datos, en este caso, son obtenidas mediante las cámaras del *Kinect*.
- Segmentación y agrupamiento: Aísla objetos, en este caso con la librería de *Kinect* se obtienen solo las manos.
- Extracción de características: El objetivo es caracterizar un objeto para que pueda ser reconocido por medidas o valores muy similares para objetos de la misma categoría, en este caso obtener las coordenadas de las manos en el tiempo.
- Selección de características: Consiste en seleccionar el mejor número de características a usar. En este caso un conjunto de coordenadas de ambas manos.

Este sistema busca analizar un cierto número de características y poder clasificar señales de entrada, a partir de una secuencia de entrenamiento o un conjunto de vectores de características de cada una de las clases [45].

Kinect por medio de su librería de desarrollo nos da la capacidad de obtener coordenadas de las articulaciones del cuerpo en todo momento, por ello entonces la etapa de extracción de las características se vuelve transparente para el programador, permitiendo enfocarse solamente a la etapa de entrenamiento.

Cuando los gestos son compuestos, se requiere utilizar las habilidades de rastreo de movimientos del cuerpo que ofrece el *Kinect*, así como la capacidad que tiene el sensor de producir un mapa de profundidad en 3D. Integrando esta información con un desarrollo de software orientado al reconocimiento de patrones de movimiento, es posible traducir los gestos a texto de manera muy precisa.

Un aspecto importante en la construcción de un sistema de reconocimiento de gestos, además de escoger una representación en sí misma, es la creación de una base de datos actualizada de los gestos conocidos y el criterio utilizado para reconocerlos (ver Ilustración 19).

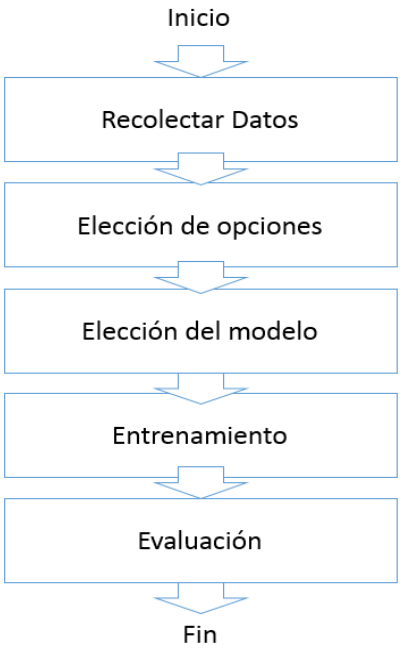


Ilustración 19 Diseño de un sistema de reconocimiento de patrones [47]

Para el reconocimiento de gestos, es posible usar las siguientes técnicas estándar de reconocimiento de patrones [48]:

1. Clasificador Bayesiano
2. Clasificadores no paramétricos
    a. El vecino más próximo
    b. K - vecinos
3. Redes Neuronales

En un estudio realizado a fin de comparar las técnicas mencionadas con anterioridad para el reconocimiento de gestos por visión artificial [48], se obtuvo que usando redes neuronales se consiguió un 95% de clasificación a diferencia del vecino más próximo con un 85.57%, por tal motivo el objetivo sobre reconocimiento de patrones se centró en el uso de redes neuronales.

A su vez las técnicas de redes neuronales multicapa y k-Vecinos tuvieron los mejores porcentajes con diferentes números de características [48], y de acuerdo con [49] los algoritmos de redes neuronales constituyen una excelente opción para el reconocimiento de gestos, puesto que ofrecen: eficiencia, factibilidad, desempeño y facilidad de implementación a diferencia de otros.

Por tal motivo se decidió trabajar con redes neuronales artificiales (RNA). Entre las cuales tenemos como opciones a los perceptrones y las redes multicapa [50] [47].

- Perceptrón: Fue desarrollado por Frank Rosenblat, el cual se utiliza como algoritmo para seleccionar un sub-grupo de un grupo de componentes más grandes.
- El perceptrón multicapa: es una red neuronal formada por múltiples capas, permitiendo resolver problemas que no son linealmente separables a diferencia del perceptrón.

Esta tesis se enfocara más al uso de las redes neuronales multicapa, por su facilidad de implementación [49], específicamente en el algoritmo de Retropropagación el cual fue presentado en 1986 por Hinton y Wiliams. Como a cualquier otro medio de entrenamiento, se le presenta a la red un vector del conjunto de entrenamiento. Al mecanismo de entrenamiento se le llama retropropagación debido a que inicialmente el error de salida de la red es identificado a nivel de su capa de salida [50].

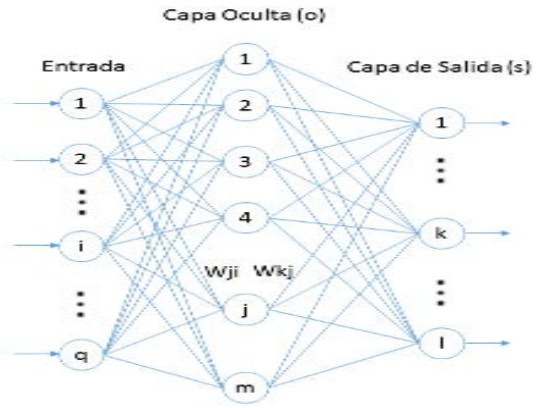


Ilustración 20 Red neuronal con una única capa oculta [50]

En la Ilustración 20 se muestra una red neuronal con una única capa oculta, el cual es el modelo básico de las redes neuronales artificiales multicapa. Una razón más por la cual se eligió este modelo es debido que en la publicación [49], muestra que se obtuvo un máximo de generalización del 99%.

2.6 Definición de gestos de interacción

El sistema a construir se basa en el reconocimiento de gestos con las manos para la manipulación de un objeto virtual, específicamente de un modelo tridimensional de una pieza arqueológica existente en el MUREH.

Por tal motivo, entonces definiremos un gesto el cual puede llegar a ser un término especializado para diferentes disciplinas de estudio. Por ejemplo, en semiótica

(estudio de los signos), los gestos pueden significar, palabras, imágenes, fórmulas matemáticas, mapas. [51].

En las artes, un gesto es usado para describir aspectos expresivos de la danza [40]. Finalmente en diseño de interacción, los gestos se enfocan en la manipulación de la experiencia de usuario en NUI [42].

Eric Hulteen y Gord Kurenbach en 1990 en su publicación llamada: *"Gestures in Human Computer Interaction*", lo definieron como [42]:

"Un gesto es un movimiento del cuerpo que contiene información. Mover la mano para decir adiós es un gesto. Presionar una tecla en un teclado no es un gesto porque el movimiento del dedo para presionar la tecla no es observado y es insignificante. "

Para reiterar bajo el contexto de interfaces de usuario, un gesto:

- Expresa un comando simple,
- Es arbitrario y natural,
- Está basado en convención y
- Puede ser malinterpretado.

Etimológicamente la palabra gesto viene del latín *gestus,* la cual era usada para referirse al lenguaje corporal de las personas. Es decir, la expresión facial, movimiento de las manos y postura del cuerpo que indican su estado de ánimo. *Gestus* viene del verbo *gerere,* que significa traer o llevar a cabo. *Gerere* también tiene el significado de administrar (llevar a cabo las cosas) [52].

Una vez definido un gesto, y entendiendo que estos pueden ser malinterpretados como cuando alguien levanta la mano, que puede entenderse que una persona requiere atención (de llevarse a cabo en un restaurante) u otra cosa (si fuese en otro contexto).

Gestos con las manos, son aquellos que se realizan por una o dos manos, esta a su vez es la categoría con mayor número de gestos debido a la capacidad de la mano humana para adquirir un gran número de configuraciones claramente discernibles, por eso la importancia del lenguaje de los signos.

Según cada escenario de aplicación, los gestos con las manos se pueden dividir en varias categorías, tales como [53]:

- Gestos de conversación.
- Gestos de control.
- Gestos de manipulación.
- Gestos comunicativos.

La categoría en la cual estará basada esta aplicación son los gestos de control, y en la interfaz basada en la visión (VBI) [54] con el objetivo de manipular objetos virtuales mediante el análisis de los gestos que registran. Algunas aplicaciones de control de visualización demuestran el potencial de señalar gestos en HCI.

Se definirá el gesto para referirse a cualquier forma de movimiento que puede ser utilizado como una entrada o interacción para controlar o influir en una aplicación.

2.7 Metodologías de desarrollo de software

Las metodologías de desarrollo de software desde el punto de vista de la ingeniería de software son vistas como un marco de trabajo que permite estructurar, planificar y controlar el proceso de desarrollo de sistemas [55].

2.7.1 Personal Software Process (PSP)

El proceso personal de software (PSP), es un conjunto de prácticas para la gestión del tiempo y mejora de la productividad personal de los programadores e ingenieros de software. Fue propuesto por Watts Humphrey en 1995, en sus inicios dirigido a estudiantes [56].

Las fases del ciclo de PSP son: Planificación, Desarrollo y Postmortem. Las etapas que lo componen se pueden ver en la Ilustración 21.

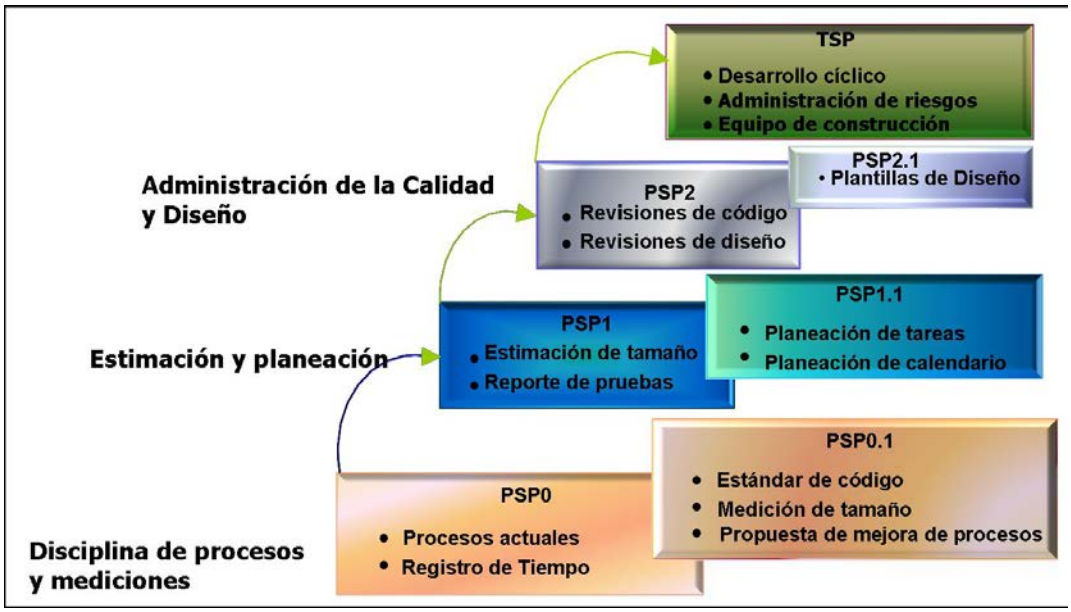


Ilustración 21 Etapas de PSP [56]

Lo interesante de esta metodología es el enfoque que le da a la etapa de diseño, pues el diseño es la idea global del sistema en la cual se plasman los

requerimientos, en primera instancia no se llega a tocar detalles técnicos o de bajo nivel.

El objetivo de tener un diseño, es evitar cometer errores en el desarrollo, logrando tener un mejor conocimiento de que se quiere lograr con el sistema, a su vez el diseño se vuelve en una representación aprovechada por quien va implementar por lo cual debe ser un documento claro, sin ambigüedades y además que descienda a un nivel de detalle de componentes. Por tanto se debe cumplir con 4 plantillas donde se plasma el trabajo [57]:

- Externo-Dinámico: Servicios y mensajes.
- Externo-Estático: Estructura de clases.
- Interno-Estático: Lógica de los programas seudocódigo.
- Interno Dinámico: Diagrama de estados.

La Tabla 4 muestra las ventajas y desventajas de PSP [57].

Desventajas	Ventajas
El uso de LOC (líneas de código) como métrica de estimación tiene sus desventajas, es dependiente del lenguaje.	La parte de calidad ayudará a producir mejores productos de trabajo.
El método de estimación PROBE (proxy basado en estimación) puede no ser efectivo si no existe suficiente correlación entre los datos históricos.	El equipo de trabajo tendrá mayor confianza porque existe una disciplina para el desarrollo de los productos.
Es subjetivo determinar si una parte del software es reutilizable.	Los datos y su análisis permitirán determinar las fortalezas y debilidades.
PSP está especialmente enfocado al desarrollo de software y no toma en cuenta el tiempo empleado en la negociación de los requerimientos con el cliente.	Los datos y su análisis posterior conducirán hacia nuevas ideas para la mejora del proceso.
Seguir PSP al pie de la letra no es viable para muchos ingenieros. Deben ver el método como una estructura para el desarrollo de una práctica de desarrollo de software con calidad.	Se gana un sentido de satisfacción personal.

Tabla 4 Ventajas y desventajas de PSP [56]

Capítulo 3: Diseño del Software

Básicamente cuando se menciona reconocimiento de gestos con las manos, se trata de asignar un cierto número de movimientos de la mano a una determinada acción en este caso rotar, agrandar, reducir y elegir una pieza arqueológica.

Como se mencionó con anterioridad los gestos son convención, tal es el caso de Samsung con su SMART TV [58], en la integración de comandos empleando gestos con las manos.

Dichos comandos empleados en la SMART TV nos ayudarán a definir cada uno de los gestos que emplearemos en dicha aplicación, puesto que son similares a los gestos que maneja el *Kinect* para la interacción con los videojuegos [43].

Para elaborar el diseño de cada uno de los gestos que la aplicación es capaz de reconocer existen diferentes técnicas [59]. La primera técnica es utilizar redes neuronales, las cuales deben entrenarse para tener mejor precisión y calidad de detección. La segunda técnica es definir algorítmicamente cada gesto y la última técnica es comparando con una serie de plantillas prestablecidas.

3.1 Acercar y Alejar

El *zoom* a una pieza arqueológica es básicamente el escalamiento de la misma en los tres ejes (X, Y, Z), según la longitud del gesto.

Este gesto estará definido por una serie de puntos obtenidos de cada una de las posiciones de las manos en el tiempo, cuando las manos estén alineadas se toma

la coordenada de la mano derecha y la mano izquierda respectiva. Es decir para el escalamiento se tiene:

$$Escalamiento = \{(xd, yd)(xi, yi)\}$$

Si se desea ampliar la pieza arqueológica se deben mantener alineadas las manos en línea recta y separarlas hasta lograr el tamaño deseado, como se muestra en la Ilustración 22.

Ilustración 22 Agrandar la pieza arqueológica

Para reducir la pieza arqueológica se deben mantener las manos alineadas horizontalmente (ver Ilustración 23) y juntarlas hasta lograr el tamaño deseado.

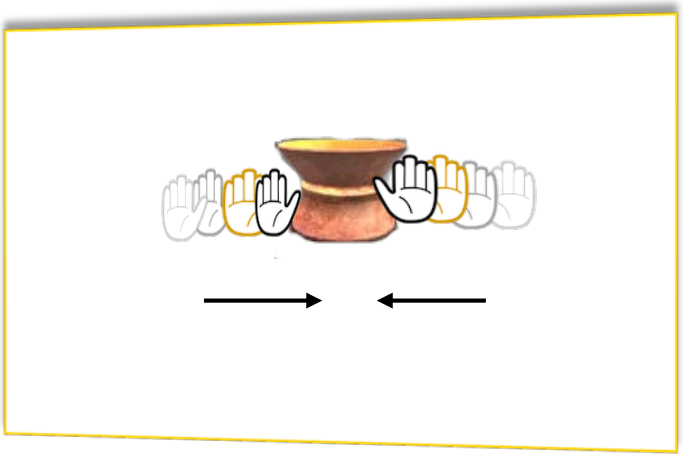


Ilustración 23 Reducir la pieza arqueológica

3.2 Deslizar

La aplicación debe ser capaz de detectar el *swipe* o deslizamiento. Este gesto básicamente, se trata de mover la mano simulando una barra de desplazamiento vertical para que el usuario pueda elegir la pieza arqueológica de su preferencia y poder manipularla.

Para que este gesto pueda ser válido se deben capturar 20 coordenadas de la mano derecha o izquierda, la mano derecha para avanzar a la siguiente pieza o la mano izquierda para seleccionar la pieza anterior.

Este gesto está compuesto por dos acciones para darle mayor libertad de uso al usuario. La primera parte del gesto se realizara únicamente con la mano derecha moviéndola de abajo hacia arriba en línea recta, para poder desplazar y elegir las piezas disponibles, como en la Ilustración 24.

Ilustración 24 Deslizamiento abajo-arriba

La segunda parte del gesto implica solamente el movimiento de la mano izquierda está desplazándose de arriba hacia abajo en línea recta, para poder desplazar y así mismo elegir la pieza de nuestra preferencia, véase Ilustración 25.

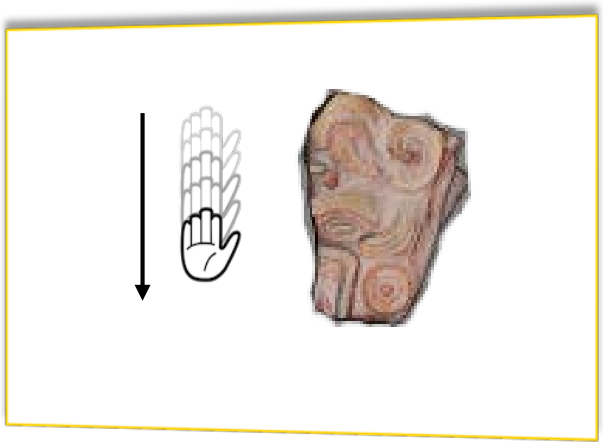


Ilustración 25 Deslizamiento arriba-abajo

Con cada una de los métodos que componen el gesto de deslizamiento, el gesto queda completado y el usuario pueda cambiar o elegir la pieza arqueológica de su preferencia.

En el Capítulo 5 “Evaluación y Resultados”, se mostrará con más detalle que después de concluir las pruebas integrales los resultados arrojaron que este gesto no era comprendido, además que el usuario presentaba ciertas dificultades por lo cual el gesto sufrió ciertas modificaciones.

Para replantear el gesto, este se definió en dos partes (ver Ilustración 26): *SwipeLeftToRight* y *SwipeRighttoLeft*. El mismo ahora se ejecutara solo con el uso de una mano, específicamente la mano derecha.

Este gesto para ser válido tiene que tener una duración entre 500 y 1500 milisegundos con una distancia promedio de 20 cm.

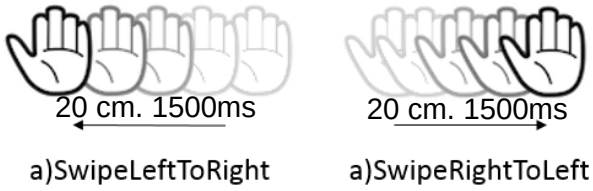


Ilustración 26 Modificación del *Swipe* (deslizamiento)

Además de la retroalimentación, en la parte superior de la pantalla el usuario podrá ver la galería de piezas disponibles para la interacción así como identificar la pieza arqueológica con la cual está trabajando (ver Ilustración 27).

Ilustración 27 Menú de piezas arqueológicas disponibles y seleccionada

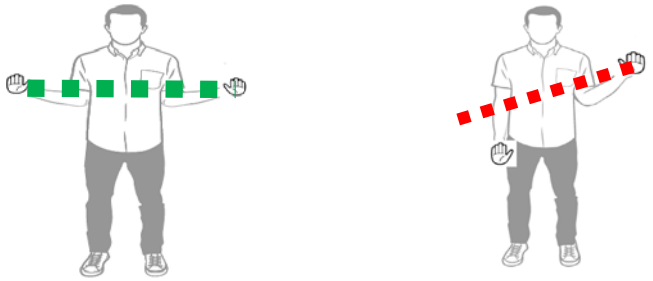
3.3 Rotación

La rotación se utiliza manipulando cada una de las coordenadas de la mano izquierda y derecha respectivamente sobre el plano X, Y Z, a un ángulo obtenido por el movimiento de las manos, esto para poder girar la pieza arqueológica (ver Ilustración 28).

Ilustración 28 Rotación de pieza arqueológica por un ángulo

Por ahora solo se usa el gesto de rotación para manipular la pieza arqueológica en el eje Z, el gesto entonces está definido de la siguiente manera.

Como se observa en la Ilustración 29 las manos deben estar alineadas para poder empezar la rotación, cada vez que las manos se muevan deben formar entre sí una línea recta para garantizar la rotación de la pieza arqueológica.



(a) (b)

Ilustración 29 (a) Manos alineadas empezar la rotación. (b) manos desalineadas imposible hacer rotación

Las manos deben estar alineadas de manera horizontal, una vez alineadas, estas deben de moverse de izquierda a derecha, o derecha a izquierda (ver Ilustración 30), en todo momento las manos deben estar alineadas formando una línea recta.

Simulando el movimiento que se realiza al manejar un coche al utilizar el volante se espera que el usuario pueda realizar la rotación a un ángulo positivo o negativo y poder mover la perspectiva de visión de la imagen.

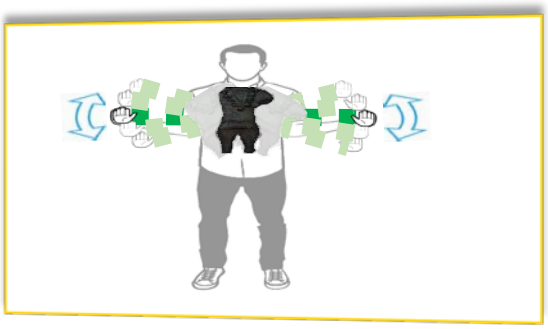


Ilustración 30 Rotación de pieza arqueológica

Los resultados de las pruebas mostrados en el Capítulo 5, indican que el usuario no está cómodo con el gesto de rotación, por tal motivo a este gesto se le hicieron las siguientes modificaciones:

- Quitar la línea verde o roja.
- Capturar 30 coordenadas de ambas manos.
- Indicar al usuario el comienzo y fin del gesto.
- Rotar ±45° si el gesto es aceptado.

Ilustración 31 Modificación al gesto de rotación

En la Ilustración 31 se muestra como se implementó el gesto, para iniciar: el usuario debe mantener sus manos alineadas, en ese momento se mostraran unas manos azules, el usuario debe mover sus manos, simulando mover un volante ya sea de derecha a izquierda o izquierda a derecha según la rotación deseada, el gesto finaliza cuando las manos vuelven a pintarse de blanco.

Para representar este gesto se capturan 30 pares de coordenadas de la mano izquierda y derecha, los cuales se trataron eliminando ciertas perturbaciones que el usuario no identifica al mover las manos.

3.4 Metodología de desarrollo

La ingeniera de software ha tomado gran impuso en las últimas décadas, además de convertirse en una de las industrias más grandes y más influyentes en la sociedad actual. Pues han evolucionado a desarrollar aplicaciones usadas solo para organismos gubernamentales y centros de investigación a aplicaciones más complejas que impregnan diversos aspectos de la vida cotidiana [1].

La intención de PSP es proporcionar orientación a los profesionales de software que estén interesados en el uso de métodos de pruebas eficaces, disciplinadas para mejorar su proceso de desarrollo de software personal [57].

Se eligió PSP como metodología de desarrollo para este trabajo debido a que actualmente se cuenta con la certificación ante el Instituto de Ingeniería de Software (SEI por sus siglas en inglés) [60]. Debido a que la certificación es uno de los mecanismos más utilizados, que una profesión emplea para hacer explícito el conjunto básico de conocimientos y habilidades que se espera que un profesionista pueda dominar, para establecer evaluaciones objetivas de esas competencias básicas, y para proporcionar una base para la continua cualificación de la persona profesional.

El motivo principal por el cual se usó esta metodología es por la serie de procesos establecidos y recomendaciones, además por su gran énfasis en el diseño del software pues este debe ser accesible y comprensible por todos sus usuarios. Los diseños de software deben estar documentados, así como la documentación relacionada con los requerimientos, restricciones y razón de ser.

PSP no especifica las técnicas de diseño, pero proporciona un conjunto de plantillas como un marco para la documentación de diseño. Las plantillas ayudan a asegurar que un sistema y sus módulos están implementados completamente y con precisión. Las plantillas son útiles para orientar a las personas en la producción simple, precisa y completa documentación de diseño.

	Interno	**Externo**
Estático	LST	FST
Dinámico	SST	OST

Tabla 5 Diseños de PSP [56]

Las plantillas de diseño de PSP representan la estructura estática y el comportamiento dinámico de un sistema de software, capturando tanto las características externamente visibles y los detalles internos. Un diseño completo PSP debe contener las siguientes cuatro categorías de elementos de diseño [57]:

- Externo - Dinámico: Operational Specification Template (OST).
- Externo - Estático: Functional Specification Template (FST).
- Interno - Dinámico: State Specification Template (SST).
- Interno - Estático: Logic Specification Template (LST).

3.5 Diseño Conceptual

El diseño conceptual, se desarrolla como primer acercamiento para contemplar las bases de desarrollo en las cuales se agrupara el sistema total. El diseño conceptual es una visión general del sistema, en el cual se muestran todos los módulos y las posibles relaciones entre ellos [56].

En la Ilustración 32 se muestra el diseño conceptual general del sistema en el cual se engloba cada uno de los módulos a construir para lograr el funcionamiento del sistema abarcando cada uno de los requerimientos o gestos que el sistema será capaz de detectar.

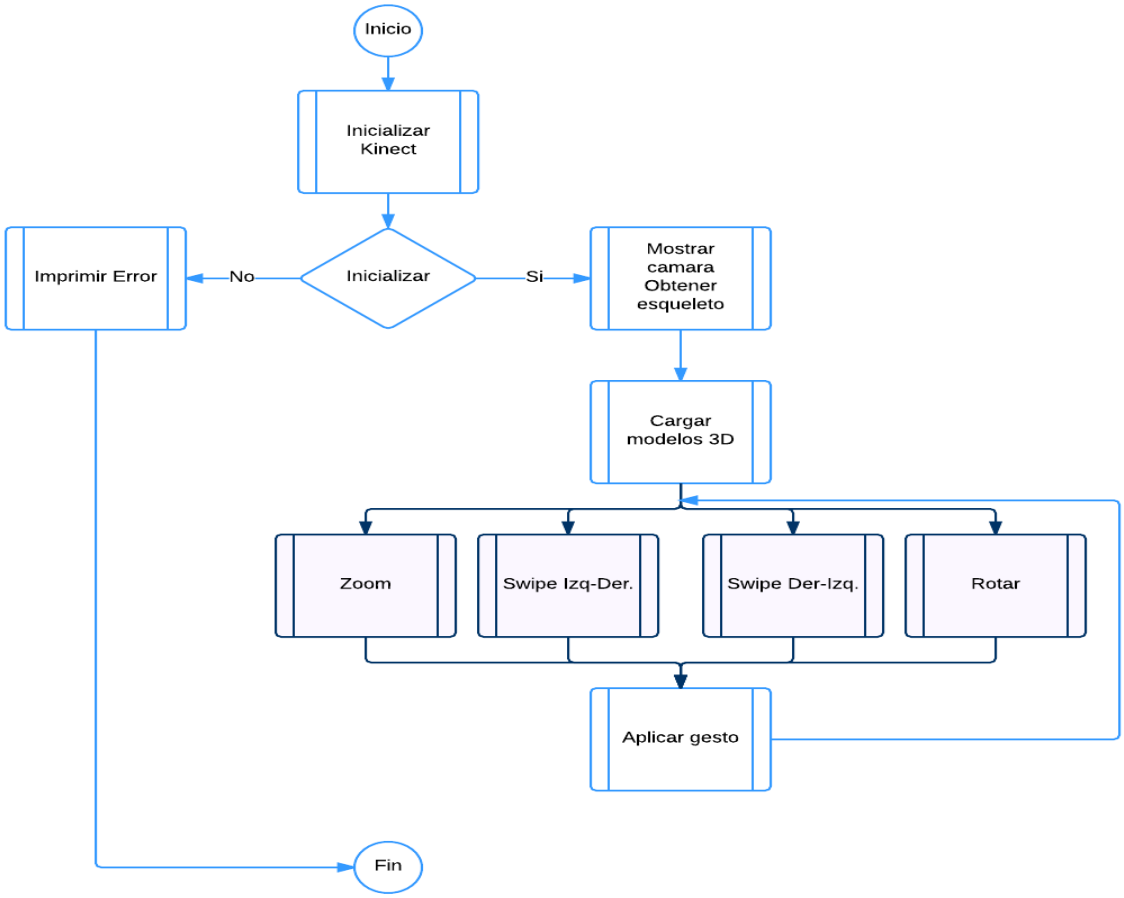


Ilustración 32 Diagrama conceptual del sistema de reconocimiento de gestos

3.6 Plantilla de Especificación Operacional (OST)

El OST enumera las características dinámicas – externas de una parte del sistema software. Describe uno o más escenarios que implican el componente de software

y los actores (usuarios u otros sistemas) que interactúan con el componente [57]. Por ello el OST debe contener los siguientes elementos:

- Componente,
- Acción y
- Comentarios.

Se construirán escenarios operacionales para cubrir el uso normal y anormal de programa, incluyéndose casos de error.

3.6.1. Acercar/Alejar

El primer escenario estará sujeto a la detección del *zoom*, descrito en la Tabla 6**.**

.

Escenario #:1		Objetivo Usuario: Gesto de *zoom* detectado	
Objetivo del Escenario:		Dadas las coordenadas de cada una de las manos el sistema detectara que es un *zoom* in o *zoom* out	
fuente	Paso	Acción	Comentarios
Usuario	1	Ejecuta el sistema	
Sistema	2	Obtiene las coordenadas de cada una de las manos del usuario.	Estas coordenadas se obtienen cada segundo.
Sistema	3	Verifica que las manos estén alineadas horizontalmente.	
Sistema	4	Calcula la distancia que existe entre las manos para determinar si es un *zoom* in o *zoom* out.	El *zoom* máximo y mínimo está determinado por la resolución de la cámara de 640x480.
Sistema		Aplica el *zoom* al modelo tridimensional.	

Tabla 6 OST detección de *zoom*

El segundo escenario, descrito en la Tabla 7 se contempla cuando no se detecte el gesto de *zoom* como válido.

Escenario #:2		Objetivo Usuario: Gesto de *zoom* no detectado	
Objetivo del Escenario:		Dadas las coordenadas de cada una de las manos el sistema detectara que no es *zoom*	
Fuente	Paso	Acción	Comentarios
Usuario	1	Ejecuta el sistema.	
Sistema	2	Obtiene las coordenadas de cada una de las manos del usuario.	Estas coordenadas se obtienen cada segundo.
Sistema	3	Verificar que las manos no estén alineadas.	

Tabla 7 OST *zoom* no detectado

3.6.2 Deslizamiento

El OST para este gesto lo definimos en dos partes por tanto es importante tener para cada una su OST correspondiente.

Primero se plantea el funcionamiento normal de detección del gesto deslizamiento con la mano derecha e izquierda dado que es similar el comportamiento del sistema, véase Tabla 8.

Escenario #:1		Objetivo Usuario: Gesto de SwipeDowntoUp(DeslizarAbajo-Arriba) y SwipeUptoDown(Deslizar Arriba-Abajo)	
Objetivo del Escenario:		Dadas 20 [40] coordenadas de la mano derecha detectar el *SwipeDowntoUp* 0 20 coordenadas de la mano izquierda para detectar el *SwipeUptoDown*.	
Fuente	Paso	Acción	Comentarios
Usuario	1	Ejecuta el sistema.	
Sistema	2	Obtiene las coordenadas de la mano derecha o mano izquierda.	Estas coordenadas se obtienen cada segundo.
Usuario	3	Mueve la mano derecha de abajo hacia arriba de forma recta o la mano izquierda de arriba hacia debajo de forma recta.	
Sistema	4	Almacena 20 coordenadas de la mano derecha o 20 coordenadas de la mano izquierda.	Estas coordenadas serán almacenadas en un vector.
Sistema	5	Analiza y determina si las coordenadas corresponden a un *SwipeDowntoUp* o a un *SwipeUptoDown*.	
Sistema	6	Cambia la pieza arqueológica en la pantalla.	

Tabla 8 OST *SwipeDowntoUp* o *SwipeUptoDown* detectado

Seguidamente se describe el escenario en la Tabla 9 cuando el sistema no sea capaz de detectar el gesto debido a que el movimiento de la mano derecha o izquierda no es el adecuado para poder definir el gesto de deslizamiento.

Escenario #:2		Objetivo Usuario:	Gesto de SwipeDowntoUp(DeslizarAbajo-Arriba) y SwipeUptoDown(Deslizar Arriba-Abajo) no detectado
Objetivo del Escenario:		Dadas 20 coordenadas de la mano derecha 0 20 coordenadas de la mano izquierda verificar que no es un gesto deslizamiento válido.	
Fuente	Paso	Acción	Comentarios
Usuario	1	Ejecuta el sistema.	
Sistema	2	Obtiene las coordenadas de la mano derecha o mano izquierda.	Estas coordenadas se obtienen cada segundo.
Usuario	3	Mueve la mano derecha de abajo hacia arriba de forma recta o la mano izquierda de arriba hacia debajo de forma recta.	
Sistema	4	Almacena 20 coordenadas de la mano derecha o 20 coordenadas de la mano izquierda.	Estas coordenadas serán almacenadas en un vector.
Sistema	5	Analiza y determina si las coordenadas corresponden a un *SwipeDowntoUp* o a un *SwipeUptoDown*.	
Sistema	6	No realiza una acción.	

Tabla 9 OST Deslizamiento no detectado

Los escenarios OST de deslizamiento nos servirán para verificar que este gesto se implementó de forma adecuada.

3.6.3 Rotación

La rotación es uno de los gestos más complicados a reconocer puesto que las manos deben permanecer alineadas durante cada movimiento para lograr la rotación, el OST en la Tabla 10 describe el funcionamiento normal para la detección del gesto.

Escenario #:1		Objetivo Usuario: Detección de rotación izquierda-derecha o derecha izquierda	
Objetivo del Escenario:		Dadas las coordenadas de la mano derecha e izquierda verificar que es una rotación valida	
Fuente	Paso	Acción	Comentarios
Usuario	1	Ejecuta el sistema.	
Sistema	2	Obtiene las coordenadas de la mano derecha y mano izquierda las cuales deben estar alineadas.	Estas coordenadas se obtienen cada segundo.
Usuario	3	Mueve las manos simulando un volante dando vuelta a la izquierda o vuelta a la derecha.	Las manos deben estar alineadas en todo momento
Sistema	4	Registra si es vuelta derecha o izquierda y a su vez las manos estén en línea recta.	
Sistema	5	Aumenta el ángulo de rotación de la pieza arqueológica.	El ángulo aumenta o disminuye según sea el movimiento.
Sistema	6	Modifica la visualización de la pieza arqueológica.	

Tabla 10 OST Rotación detectada

Cuando las manos no estén alineadas el reconocedor de gestos no hará cambio alguno a la pieza arqueológica tal como se describe en el OST de la Tabla 11 cuando el gesto no es válido.

Escenario #:2		Objetivo Usuario: Rotación no detectada	
Objetivo del Escenario:		Dadas las coordenadas de la mano derecha e izquierda verificar que el gesto de rotar no es válido	
Fuente	Paso	Acción	Comentarios
Usuario	1	Ejecuta el sistema.	
Sistema	2	Obtiene las coordenadas de la mano derecha y mano izquierda.	Estas coordenadas se obtienen cada segundo.
Usuario	3	Mueve las manos simulando un volante dando vuelta a la izquierda o vuelta a la derecha.	El sistema muestra que las manos no están alineadas
Sistema	4	Las manos no están alienadas por tanto no es un gesto válido.	
Sistema	5	La pieza arqueológica no sufre cambio alguno.	

Tabla 11 OST Rotación no detectada

3.7 Plantilla Especificación Funcional (FST)

El FST documenta una parte, de un sistema, por ejemplo una clase y las relaciones externas – estáticas y los atributos visibles externamente. En este se describen las acciones (métodos de clase) que el componente tiene disponible para la comunicación externa, incluye la interfaz definida para cada acción, argumentos, restricciones y resultados devueltos [57].

3.7.1. Acercar/Alejar

Para poder detectar si los movimientos de las manos corresponden a un *zoom* y además poder aplicarlo a la pieza arqueológica se requieren de las siguientes variables y métodos descritos en la Tabla 12.

El procedimiento parte de la fórmula matemática de la distancia entre dos puntos [61], tales puntos son $A(x_1, y_1)$ y $B(x_2, y_2)$, en este caso el punto *A* representa las coordenadas de la mano derecha y el punto *B* de la mano izquierda.

La siguiente ecuación se utiliza para obtener la distancia euclidiana entre dos puntos, para posteriormente obtener la escala asociada.

$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$ ……. (Ecuación distancia entre dos puntos)

Clase de Prueba	***ZOOM***
Variables	
Declaración	Descripción
Float manoizqx	Almacena el valor de la posición de la mano izquierda en el eje *X*
Float manoizqy	Almacena el valor de la posición de la mano izquierda en el eje *Y*
Float manoderx	Almacena el valor de la posición de la mano derecha en el eje *X*
Float manodery	Almacena el valor de la posición de la mano derecha en el eje *Y*
Float delta=20	Almacena el margen de error para la alineación
Float pspinex	Almacena el valor del punto central del esqueleto humano en el eje *X*
Float pspiney	Almacena el valor del punto central del esqueleto humano en el eje *Y*
Float escala_inicial=0.5	Almacena el valor de la escala de la pieza arqueológica
Métodos	
Float distancia(float ix, float iy, float dx, float dy)	Calcula la distancia existente entre la posición de la mano derecho y la mano izquierda. $d = \sqrt{(ix - dx)^2 + (iy - dy)^2}$

Tabla 12 FST *Zoom*

3.7.2 Deslizamiento

El deslizamiento maneja dos métodos en la Tabla 13 se muestran las variables para la detección del *SwipeDowntoUp*:

Clase de Prueba	***SwipeDowntoUp***
Variables	
Declaración	Descripción

public struct VectorG
{
public float X;
public float Y;
public float Z;
public DateTime date;
}
	Estructura de datos que almacena las coordenadas de la mano derecha en los ejes X, *Y Z*, además del tiempo en el que se capturan dichas coordenadas.

const int SwipeMininalDuration = 250	Almacena la duración mínima del gesto en milisegundos.
const int SwipeMaximalDuration = 1500	Almacena la duración máxima del gesto en milisegundos.
const float SwipeMinimalLength = 0.4f	Almacena la longitud mínima del gesto.
const float SwipeMaximalHeight = 0.2f	Almacena la elevación máxima del gesto.
List positionList	Lista que almacena los puntos de la mano derecha en la estructura de datos VectorG.
Boolean swipe	Bandera que indica si el gesto fue aceptado.
Métodos	
Void Swipe1(Boolean s)	Aplica el cambio de pieza arqueológica cuando el gesto es aceptado.

Tabla 13 FST *SwipeDowntoUp* detectado

De forma similar la Tabla 14 muestra el FST para el reconocimiento del gesto *SwipeUptoDown*.

Clase de Prueba	***SwipeDowntoUp***
Variables	
Declaración	Descripción

public struct VectorG
{
public float X;
public float Y;
public float Z;
public DateTime date;
}
	Estructura de datos que almacena las coordenadas de la mano izquierda en los ejes *X, Y Z,* además del tiempo en el que se capturan dichas coordenadas.

const int SwipeMininalDuration = 250	Almacena la duración mínima del gesto en milisegundos.
const int SwipeMaximalDuration = 1500	Almacena la duración máxima del gesto en milisegundos.
const float SwipeMinimalLength = 0.4f	Almacena la longitud mínima del gesto.
const float SwipeMaximalHeight = 0.2f	Almacena la elevación máxima del gesto.
List positionList2	Lista que almacena los puntos de la mano izquierda en la estructura de datos VectorG.
Boolean swipe	Bandera que indica si el gesto fue aceptado.
Métodos	
Void Swipe2(Boolean s)	Aplica el cambio de pieza arqueológica cuando el gesto es aceptado.

Tabla 14 FST *SwipeUptoDown* detectado

3.7.3 Rotación

Para poder generar el gesto de rotación son requeridos las siguientes variables y métodos identificados en la Tabla 15.

Para determinar cuándo las manos están alineadas se partió, por calcular los coeficientes de la ecuación general de la recta que pasa por dos puntos, en este

caso el punto A, dado por las coordenadas de la mano derecha y el punto B que representa las coordenadas del punto central del esqueleto, tal cual la ecuación:

$$AX - BY + C = 0 \ldots \text{(Ecuación general de la recta [61])}$$

Obtenidos el coeficiente *A* y *B* nos permitirán determinar si las coordenadas del punto C que son las de la mano izquierda pasan por la recta calculada y con esto determinar la alineación entre las manos.

Clase de Prueba	**Rotar**
Variables	
Declaración	Descripción
Float manoizqx	Almacena el valor de la posición de la mano izquierda en el eje *X*
Float manoizqy	Almacena el valor de la posición de la mano izquierda en el eje *Y*
Float manoderx	Almacena el valor de la posición de la mano derecha en el eje *X*
Float manodery	Almacena el valor de la posición de la mano derecha en el eje *Y*
Float pspinex	Almacena el valor del punto central del esqueleto humano en el eje *X*
Float pspiney	Almacena el valor del punto central del esqueleto humano en el eje *Y*
Float rotarzinicial=0	Almacena el valor del ángulo para la rotación respecto al eje *Z*
Float dpr=0	Almacena la distancia de un punto a una recta
Float a	Almacena la constante a de la ecuación general de la recta
Float b	Almacena la constante b de la ecuación general de la recta
Float c	Almacena la constante c de la ecuación general de la recta
Métodos	
Vector calcula_recta(float mx, float my, float px, float py)	Esta función calcula los coeficientes de la forma general de la recta que pasa por dos puntos: la mano derecha y el punto central del esqueleto. $AX - BY + C = 0$ Donde *A*, *B* y *C* son numero reales dados por: $A = manodery - pspiney$ $B = pspinex - manoderx$ $C = (-A * pspineX) + (-B * pspineY)$

Float calcula_y(float mx, float a, float b, float c)	Esta función dado un punto en x, en este caso la coordenada x de la mano derecho, calcula su correspondiente coordenada y según los coeficientes de la ecuación de la recta calculada. $$Y = \frac{(-A * mx) - C}{B}$$
Float dpunto_recta(float mix, float miy, float a, float b, float c)	Esta función calcula la distancia de un punto a una recta dada por la siguiente ecuación: $$d = \frac{\lvert A * mix + B * mix + C \rvert}{\sqrt{A^2 + B^2}}$$

Tabla 15 FST Rotación detectada

El módulo de rotación fue cambiado después de los resultados obtenidos en las pruebas integrales, por tal motivo en la Tabla 16 se muestra los cambios realizados.

Clase de Prueba	**Rotar**
Métodos	
Declaración	Descripción
Public void ordena();	Ordenar de menor a mayor la coordenada X de los puntos capturados de las manos, internamente algoritmo de la burbuja.
Public void depurar();	Se eliminan coordenadas repetidas

Tabla 16 FST Modificación a rotación

3.8 Plantilla de Especificación de Estados (SST)

El SST documenta el comportamiento interno – dinámico, cuando ese comportamiento se representa como un conjunto de estados, transiciones entre estados, y las acciones asociadas a las transiciones. El SST se puede complementar con un diagrama de estado independiente [57].

Un SST contiene:

- Nombre de los estados y descripciones,
- Funciones y parámetros, y
- Detalles de la transición de estados.

3.8.1. Acercar/Alejar

El *zoom* está compuesto por 5 estados los cuales se describen en la Tabla 17:

Nombre Estado	**Descripción**
Inicio	Define el estado inicial de la máquina de estados, obteniendo las coordenadas de las manos.
Validar	Valida que las coordenadas de las manos estén alineadas de forma horizontal.
Calcular_distancia	Calcula la distancia de las manos, para aplicar el *zoom*.
Aplicar_zoom	Aplica el valor del *zoom* a la pieza arqueológica.
Fin	Termina la detección del gesto.
Función/Parámetro	**Descripción**
Float manoderx	Almacena la coordenada *x* de la mano derecha.
Float manodery	Almacena la coordenada *y* de la mano derecha.
Float manoizqx	Almacena la coordenada *x* de la mano izquierda.
Float manoizqy	Almacena la coordenada *y* de la mano izquierda.
Float escala	Almacena el valor de escalamiento que tendrá la pieza arqueológica.
Boolean gesto	Almacena si el gesto es válido.

Estados/Siguientes Estados	**Condición de transición**	**Acción**
Inicio		Obtener coordenadas de cada una de las manos.
Validar	Tener las coordenadas de las manos.	
Calcular_distancia	No hay transición	
Aplicar_zoom	No hay transición	
Fin	No hay transición	
Validar		Verificar que las coordenadas *Y* de cada mano tengan el mismo valor ±*Error*.
Inicio	No hay transición.	
Calcular_distancia	Si la coordenada *Y* de ambas manos es igual ±Error se calcula distancia	Calcular la distancia entre la coordenada de *manoderx* y *manodery*, para obtener el valor de la escala.
Aplicar_*zoom*	No hay transición	

Fin	Si la coordenada *Y* de ambas manos es diferente *±Error*	
Calcular_distancia		
Inicio	No hay transición	
validar	No hay transición	
Aplicar_zoom	Si la distancia está calculada	Asignar a escala el valor calculado de zoom correspondiente.
fin	No hay transición	
Aplicar_zoom		La pieza arqueológica toma el tamaño de la escala calculada
Inicio	No hay transición	
validar	No hay transición	
Calcular_distancia	No hay transición	
Fin	No hay transición	

Tabla 17 SST para *zoom*

3.8.2 Deslizamiento

El deslizamiento se compone por 6 estados de inicio a fin para completar el gesto, los cuales se muestran en la Tabla 18:

Nombre Estado	**Descripción**
Inicio	Define el estado inicial de la máquina de estados, obteniendo las coordenadas de las manos.
Verificar_Swipe	Verifica que ya se tengan almacenados 20 puntos de la mano derecha o izquierda.
Swipe1	Verifica que los puntos de la mano derecha sean válidos para *SwipeDowntoUp*.
Swipe2	Verifica que los puntos de la mano izquierda sean válidos para *SwipeUpDown*.
Aplicar_Swipe	Muestra la siguiente pieza arqueológica
Fin	Termina la detección del gesto
Función/Parámetro	**Descripción**
List positionList	Lista que contiene los valores de las coordenadas <X, Y, Z> correspondientes a la mano derecha.
List positionList2	Lista que contiene los valores de las coordenadas <*X, Y,Z*> correspondientes a la mano izquierda.
Boolean Swipe	Almacena el valor para saber qué tipo de Swipe se detectó.

Swipe1()	Verifica que sea un *SwipeDowntoUp*.	
Swipe2()	Verifica que sea un *SwipeUptoDown*.	
Estados/Siguientes Estados	**Condición de transición**	**Acción**
Inicio		Capturar 20 puntos de mano derecha o mano izquierda
Inicio	No hay transición	
Verificar_Swipe	Tener 20 puntos almacenados	
Swipe1	No hay transición	
Swipe2	No hay transición	
Aplicar_Swipe	No hay transición	
Fin	No hay transición	
Verificar_Swipe		Tener 20 puntos de la mano derecha o izquierda almacenados.
Inicio	No hay transición	
Verificar_Swipe	No hay transición	
Swipe1	Verificar si los 20 puntos almacenados en positionList de la mano derecha corresponden al SwipeDowntoUp	
Swipe2	Verificar si los 20 puntos almacenados en positionList2 de la mano izquierda corresponden al SwipeUptoDown	
Aplicar_Swipe	No hay transición	
Fin	No hay transición	
Swipe1		Validar punto a punto para saber si es *swipe*=1
Inicio	No hay transición	
Verificar_Swipe	No hay transición	
Swipe1	No hay transición	
Swipe2	No hay transición	
Aplicar_Swipe	Si *swipe*=1	
Fin	Si *swipe*=0	

Swipe2		Validar punto a punto para saber si es *swipe*=2
Inicio	No hay transición	
Verificar_Swipe	No hay transición	
Swipe1	No hay transición	
Swipe2	No hay transición	
Aplicar_Swipe	Si *swipe*=2	
Fin	Si *swipe*=0	
Aplicar_Swipe		
Inicio	No hay transición	
Verificar_Swipe	No hay transición	
Swipe1	No hay transición	
Swipe2	No hay transición	
Aplicar_Swipe	*swipe*=1 o *swipe*=2	Mostrar la pieza anterior o siguiente de las piezas arqueológicas disponibles
Fin	No hay transición	

Tabla 18 SST para deslizamiento

3.8.3 Rotación

El deslizamiento se compone por 4 estados de inicio a fin para completar el gesto, los cuales se muestran en la Tabla 19:

Nombre Estado	**Descripción**
Inicio	Define el estado de inicio, se obtienen los puntos de las manos y el de la mitad del esqueleto humano.
Validar	Valida que sea un gesto de rotación válido.
Aplicar	Aplica la rotación según el ángulo del gesto.
Fin	Termina el gesto.
Función/Parámetro	**Descripción**
List manoder	Almacena las coordenadas *X* y *Y* de la mano derecha.
List manoizq	Almacena las coordenadas *X* y *Y* de la mano izquierda.
List pspine	Almacena las coordenadas *X* y *Y* de la mano spine.
Float dpr=0	Almacena la distancia del punto a la recta.
Vector calcula_recta(float	Esta función calcula los coeficientes de la forma general de la recta que pasa por dos puntos: la mano derecha y el punto central del esqueleto.

mx, float my, float px, float py)	$$AX - BY + C = 0$$ Donde *A, B* y *C* son numero reales dados por: $$A = manodery - pspiney$$ $$B = pspinex - manoderx$$ $$C = (-A * pspineX) + (-B * pspineY)$$
Float calcula_y(float mx, float a, float b, float c)	Esta función dado un punto en x, en este caso la coordenada *x* de la mano derecho, calcula su correspondiente coordenada y según los coeficientes de la ecuación de la recta calculada. $$Y = \frac{(-A * mx) - C}{B}$$
Float dpunto_recta(float mix, float miy, float a, float b, float c)	Esta función calcula la distancia de un punto a una recta dada por la siguiente ecuación: $$d = \frac{\lvert A * mix + B * mix + C \rvert}{\sqrt{A^2 + B^2}}$$

Estados/Siguientes Estados	**Condición de transición**	**Acción**
Inicio		Obtener coordenadas de las manos y mitad del esqueleto humano.
Inicio	No hay transición	
Validar	Puntos almacenados	
Aplicar	No hay transición	
Fin	No hay transición	
Validar		Calcular los coeficientes de la recta: calcula_recta, calcular_y, y finalmente calcular la distancia de un punto a una recta.
Inicio	No hay transición	
Validar	No hay transición	
Aplicar	*Si dpr>0 y dpr<25*	El gesto es aceptado.
Fin	*Si dpr>0 o dpr>25*	El gesto no es aceptado.
Aplicar		
Inicio	No hay transición	
Validar	No hay transición	
Aplicar	No hay transición	
Fin	Fin del gesto	Aplicar la rotación según sea el movimiento de las manos.

Tabla 19 SST para Rotación

3.9 Plantilla de Especificación Lógica (LST)

El LST documenta las características internas – estáticas de una parte del sistema. En él se describe la lógica interna, utilizando seudocódigo para explicar de forma clara y concisa su funcionamiento [57].

3.9.1. Acercar/Alejar

El *zoom* se obtiene al tener alineadas la mano derecha e izquierda respecto a la coordenada *Y*, para poder calcular el *zoom* correspondiente, se toma el tamaño máximo de la pantalla, puesto que la pantalla será de 640x480 pixeles tal como en la Ilustración 33.

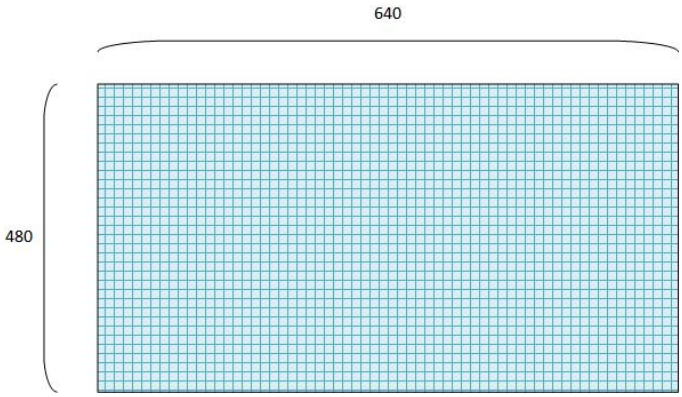
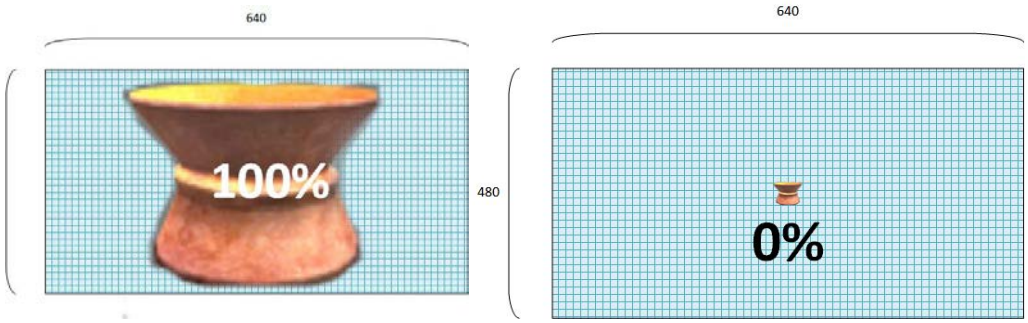


Ilustración 33 Tamaño de la ventana de la aplicación

Por tanto, el escalamiento máximo de la pieza arqueológica tendrá un valor de 2, que equivale al 100% (ver Ilustración 34) de escalamiento que se le va a permitir al usuario, el escalamiento mínimo tendrá un valor de 0.5 equivalente al 0%.



(a) *Zoom* al 100% (b) *Zoom* al 0%

Ilustración 34 Comportamiento del *Zoom* dado por el tamaño de la pantalla

En la Tabla 20 se muestra el LST, el cual muestra el comportamiento del gesto de *zoom* para poder hacer la manipulación de cada una de las piezas arqueológicas:

Parámetros	**Descripción**
Float manoizqx	Almacena el valor de la posición de la mano izquierda en el eje *X*
Float manoizqy	Almacena el valor de la posición de la mano izquierda en el eje *Y*
Float manoderx	Almacena el valor de la posición de la mano derecha en el eje *X*
Float manodery	Almacena el valor de la posición de la mano derecha en el eje *Y*
Float delta=20	Almacena el margen de error para la alineación
Float pspinex	Almacena el valor del punto central del esqueleto humano en el eje *X*
Float pspiney	Almacena el valor del punto central del esqueleto humano en el eje *Y*
Float escala_inicial=0.5	Almacena el valor de la escala de la pieza arqueológica

```
Declaración de librerías: Microsoft.Kinect, System;
Definir constantes:
int delta = 20;
```

```
Obtener las coordenadas de las manos usando Kinect.
```

```
  void Sensor_Skeleton_FrameReady(object sender, SkeletonFrameReadyEventArgs
e);
```

```
            Si  (manoizqy < (manodery + delta) && manoizqy > (manodery -
delta) &&
               (manoizqy < (pspine.Y + delta) && manoizqy > (pspine.Y - delta)
||
                manodery < (pspine.Y + delta) && manodery > (pspine.Y -
delta)))
            Hacer
                    distancia =√((manoizqx − manoderx)² + (manoizqy − manodery),²
                    Si (distancia > 25)
                    {
                        escalainicial = (distancia * 2) / 640;
                    }
```

Tabla 20 LST para *zoom*

3.9.2 Deslizamiento

La implementación del deslizamiento según las especificaciones, la lógica de desarrollo para este gesto se encuentra en el siguiente LST en la Tabla 21, los cambios realizados después de las pruebas se observan en negritas.

Parámetros	**Descripción**

public struct VectorG
{
public float X;
public float Y;
public float Z;
public DateTime date;
}
	Estructura de datos que almacena las coordenadas de la mano derecha en los ejes *X, Y, Z*, además del tiempo en el que se capturan dichas coordenadas.

const int SwipeMininalDuration = 250	Almacena la duración mínima del gesto en milisegundos.
const int SwipeMaximalDuration = 1500	Almacena la duración máxima del gesto en milisegundos.
const float SwipeMinimalLength = 0.4f	Almacena la longitud mínima del gesto.

const float SwipeMaximalHeight = 0.2f	Almacena la elevación máxima del gesto.
List positionList	Lista que almacena los puntos de la mano derecha en la estructura de datos *VectorG*.
List positionList2	Lista que almacena los puntos de la mano izquierda en la estructura de datos *VectorG*.
Boolean swipe	Bandera que indica si el gesto fue aceptado.

```
Declaración de librerías: Microsoft.Kinect, System;

Definir constantes
```

```
Obtener las coordenadas de las manos usando Kinect.

  void Sensor_Skeleton_FrameReady(object sender, SkeletonFrameReadyEventArgs
e);
```

```
void SwipeDowntoUp()

        {

            int start = 0;

            for index = 0 until  19 do

//La comprobación que se realiza es para ver si el movimiento de la mano sigue
siempre //la misma altura y en la dirección correcta.


                if ((Abs(positionList[0].X - positionList[index].X) >
SwipeMaximalHeight) ||   (positionList[index].Y - positionList[index + 1].Y >
-0.01f))

     if ((Math.Abs(positionList[0].Y - positionList[index].Y) >
SwipeMaximalHeight) || (positionList[index].X - positionList[index + 1].X > -
0.01f))


                do

                    start = index;

//La siguiente comprobación es para ver si el movimiento de la mano tiene la
misma //longitud


                if ((Abs(positionList[index].Y - positionList[start].Y) >
SwipeMinimalLength))

if ((Math.Abs(positionList[index].X - positionList[start].X) >
SwipeMinimalLength)
```

```
                do
                    totalMilliseconds = (positionList[index].date -
positionList[start].date).TotalMilliseconds;


//La ultima comprobación es para determinar si cumple con el tiempo minino y
máximo //de la duración del gesto
                    if (totalMilliseconds >= SwipeMininalDuration &&
totalMilliseconds <= SwipeMaximalDuration)
                    do
                           figseleccionada = ((figseleccionada + 1) % 5);
                    end-if
                end-if


            end-if
        end-of-for
}
```

```
void SwipeUptoDown()
        {
            int start = 0;
                      for index = 0 until  19 do
//La comprobación que se realiza es para ver si el movimiento de la mano sigue
siempre //la misma altura y en la dirección correcta.


                if ((Abs(positionList2[0].X - positionList2[index].X) >
SwipeMaximalHeight) || (positionList2[index].Y - positionList2[index + 1].Y <
-0.01f))
    if ((Math.Abs(positionList[0].Y - positionList[index].Y)
>SwipeMaximalHeight) || (positionList[index].X - positionList[index + 1].X < -
0.01f))
                do
                    start = index;


//La siguiente comprobación es para ver si el movimiento de la mano tiene la
misma //longitud


                if ((Math.Abs(positionList2[index].Y - positionList2[start].Y)
> SwipeMinimalLength))
if ((Math.Abs(positionList[index].X - positionList[start].X) >
SwipeMinimalLength))
```

```
                do


                    totalMilliseconds = (positionList2[index].date -
positionList2[start].date).TotalMilliseconds;

//La ultima comprobación es para determinar si cumple con el tiempo minino y
máximo //de la duración del gesto


                    if (totalMilliseconds >= SwipeMininalDuration &&
totalMilliseconds <= SwipeMaximalDuration)

                    do

                            figseleccionada= ((figseleccionada - 1 + 5) % 5);

            end-if

                end-if


            end-if

        end-of-for

        }
```

Tabla 21 LST para deslizamiento

3.9.3 Rotación

La lógica para la aceptación de un gesto por una rotación se describe en la Tabla 22:

Parámetros	**Descripción**
Float manoizqx	Almacena el valor de la posición de la mano izquierda en el eje *X*
Float manoizqy	Almacena el valor de la posición de la mano izquierda en el eje *Y*
Float manoderx	Almacena el valor de la posición de la mano derecha en el eje *X*
Float manodery	Almacena el valor de la posición de la mano derecha en el eje *Y*
Float pspinex	Almacena el valor del punto central del esqueleto humano en el eje *X*
Float pspiney	Almacena el valor del punto central del esqueleto humano en el eje *Y*

Float rotarzinicial=0	Almacena el valor del ángulo para la rotación respecto al eje *Z*
Float dpr=0	Almacena la distancia de un punto a una recta
Float a	Almacena la constante a de la ecuación general de la recta
Float b	Almacena la constante b de la ecuación general de la recta
Float c	Almacena la constante c de la ecuación general de la recta
Métodos	
Vector calcula_recta(float mx, float my, float px, float py)	Esta función calcula los coeficientes de la forma general de la recta que pasa por dos puntos: la mano derecha y el punto central del esqueleto. $$AX - BY + C = 0$$ Donde *A, B* y *C* son numero reales dados por: $$A = manodery - pspiney$$ $$B = pspinex - manoderx$$ $$C = (-A * pspineX) + (-B * pspineY)$$
Float calcula_y(float mx, float a, float b, float c)	Esta función dado un punto en x, en este caso la coordenada *X* de la mano derecho, calcula su correspondiente coordenada y según los coeficientes de la ecuación de la recta calculada. $$Y = \frac{(-A * mx) - C}{B}$$
Float dpunto_recta(float mix, float miy, float a, float b, float c)	Esta función calcula la distancia de un punto a una recta dada por la siguiente ecuación: $$d = \frac{\lvert A * mix + B * mix + C \rvert}{\sqrt{A^2 + B^2}}$$

```
Declaración de librerías: Microsoft.Kinect, System;

Definir constantes
```

```
Obtener las coordenadas de las manos usando Kinect.

  void Sensor_Skeleton_FrameReady(object sender, SkeletonFrameReadyEventArgs
e);
```

```
void rotar() {

            calcula_recta();

            y_calcula=calcula_y();

            dpr=dpunto_recta();
```

```
//verifica que el movimiento de las manos corresponda a la rotación con un
margen //de error
            if (dpr > 0 && dpr <= 20)
            do
                rotary_anterior = rotary_actual;
                rotary_actual = tan(PI * (pspine.Y - manodery) / (pspine.X -
manoderx)));
                rotarzinicial = rotary_actual - rotary_anterior;
                flag = 1;
            end-if
            else
            do
                flag = 0;
            end-else
//La bandera funciona para poder dibujar una línea roja o verde en pantalla a
fin //de ofrecer retroalimentación al usuario.
        }
```

```
void calcula_recta() {
            a = manodery - pspine.Y;
            b = pspine.X - manoderx;
            c1 = -a * pspine.X;
            c2 = -b * pspine.Y;
            c = c1+c2;
        }
```

```
        void calcula_y() {
            y1;
            y1 = -a * manoizqx;
            y1 = y1 - c;
            y_calcula = y1 / b;
        }
```

```
        void dpunto_recta()
        {            ar = a * manoizqx + b * manoizqy + c;
            ar = Abs(ar);
            ab1 = a^2
            ab2 = b^2
```

```
            ab = √(a² + b²)
            dpr = ar / ab;


}
```

Tabla 22 LSP para Rotación

Para este gesto fue necesario hacer unas modificaciones por tanto se presenta el nuevo LSP para rotación (Ver Tabla 23).

```
Declaración de librerías: Microsoft.Kinect, System;
Definir constantes
```

```
Obtener las coordenadas de las manos usando Kinect.
  void Sensor_Skeleton_FrameReady(object sender, SkeletonFrameReadyEventArgs
e);
```

```
public void rotar() {
  if (manoizqy < (manodery + delta) && manoizqy > (manodery - delta) &&
puntos.Count() == 0 && pspine.Y>=manodery)
do
                inicia = true;
end-if
Almacenar_punto();
if (inicia == true && puntos.Count() > 0 && puntos.Count() < 20)
do
Almacenar_puntos();
End-if
  if (puntos.Count() == 20 && inicia == true)
do
 ordena();
 depura();
 d1 = calc_d(puntos[0].X0, puntos[0].Y0, puntos[0].X1, puntos[0].Y1);
end-if
for i = 1; until 19
                do
                    d2 = calc_d(puntos[i].X0, puntos[i].Y0, puntos[i].X1,
puntos[i].Y1);
```

```
end-for
                    if (d2 >= d1 - deltar && d2 <= d1 + deltar)
      do
                        band++;
      end-if
                    else band--;
                if (band >= 10 && puntos[0].Y0 + 10 <= puntos[puntos.Count() -
1].Y0 + deltarp && a1 + 10 >= a2)
                do
                    rotarzinicial = rotarzinicial - 1.0f;
      end-if
}
```

```
Ordenar(){Ordenar Mediante el método de la burbuja}
```

```
Depurar(){Eliminar puntos repetidos.}
```

Tabla 23 LSP Modificado para rotación

Este procedimiento sirvió para poder obtener las coordenadas de las manos para cada gesto de rotación ya sea válido o no, lo cual permitirá obtener el vector de entrada para la implementación de la red neuronal.

Para reconocer como válido el gesto de rotación se utilizó una red neuronal de perceptrón multicapa (ver Ilustración 35), con una capa de entrada, una capa oculta y una de salida.

Según [49] es suficiente con la topología mencionada anteriormente para obtener un buen rendimiento.

Se tomaron 50 muestras para el entrenamiento de la red, esto se debió a que las pruebas de usabilidad requerían únicamente 5 usuarios [62], y cada uno de ellos realizó el ejercicio de rotación 10 veces, logrando con esto cumplir los tiempos

establecidos en las pruebas de usabilidad, tomando en cuenta que también se tendrían que probar los demás gestos.

La capa de entrada recibe 40 neuronas, de las cuales 10 pares de coordenadas corresponden a la mano derecha y las otras 10 corresponden a coordenadas de la mano izquierda. La capa oculta contiene 5 neuronas con funciones de activación sigmoidal debido a que según [49] es suficiente para obtener un buen rendimiento, finalmente la capa de salida contiene solo una neurona la cual indica si el gesto es correcto o incorrecto.

El algoritmo de aprendizaje (ver Tabla 24) utilizado fue de retropropagación implementado en c#, pues es el más utilizado por su facilidad de implementar y los buenos resultados obtenidos en un clasificador de gestos [49].

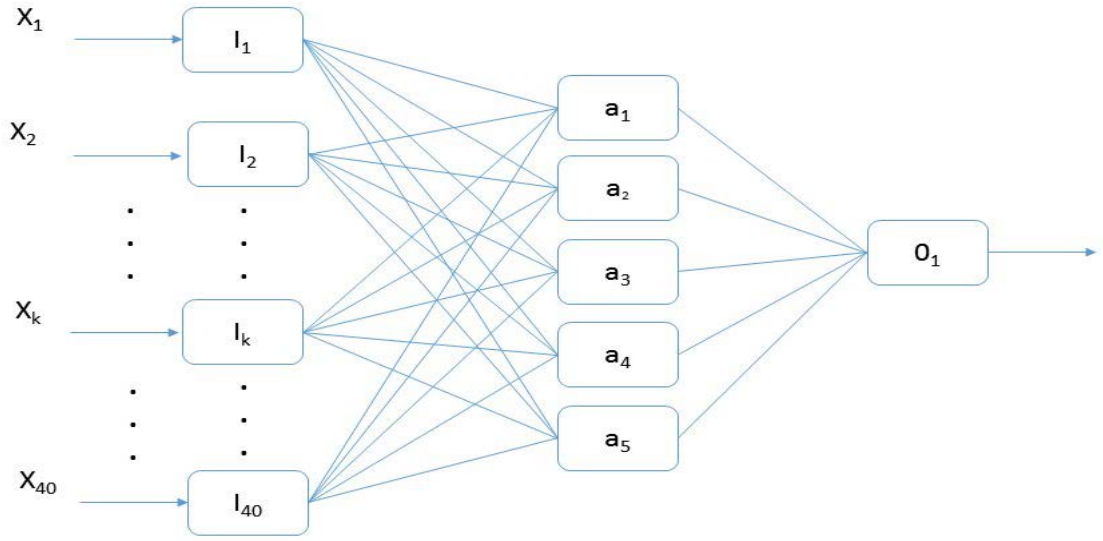


Ilustración 35 Red de Perceptrones con una única capa oculta para el gesto de rotación

La retropropagación para el ajuste de los pesos, comprende de dos fases (ver Tabla 24):

- Fase 1: Actualización de los pesos en las Neuronas en la capa de salida.
- Fase 2: Actualización de los pesos en las Neuronas en la capa oculta.

Este algoritmo requiere que la función de activación sea continua y diferenciable en (-∞, ∞), la cual se usó para aproximar los valores de salida de la red neuronal artificial con una capa oculta, tal como la función sigmoide que es la que se ocupara, descrita como:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Donde su derivada es:

$$g'(z) = \frac{e^{-z}}{(1 + e^{-z})^2}$$

Clase de Prueba	**Rotar**
Variables	
Declaración	Descripción
Int N	Número de elementos en el conjunto de entrenamiento
Int n1	Número de neuronas en la capa de entrada
Int n2	Número de neuronas en la capa oculta
Int n3	Número de neuronas en la capa de salida
Int L	Número de veces que el conjunto entrenamiento será presentado a la red
Int alfa	Coeficiente de aprendizaje
Matrix X[N,n1]	Contiene los N vectores de entrada del conjunto de entrenamiento
Matrix T[N,n3]	Contiene los vectores que se espera la red proporcione

Métodos	
{a,0}=propagation(n1,n2,n3,W_hidden, W_output,x[n]	Calcular los pesos actuales de las neuronas

```
Function  BackPropagacion(int N, int n1, int n2, int n3, int L, double alpha, double[,] X,double[,] T)
    {
        int k, t, n, j, i;
        vector xn [N];
        matrix  W_ocultos[ n2, n1];
        matrix  W_salida = [n3, n2];
        vector delta [n3];
        double SSE = 0.0;
        double SSE_i;
        Random = new Random();

        //Los pesos de las neuronas en la capa oculta se inicializan aleatoriamente en el rango
de (-1,1)
for j=0 until j<n2
        do
                for k=0 until k<n1
                do
                        W_ocultos[j, k] = random(-1, 1);
                end-for
        end-for
        //Los pesos de las neuronas en la capa de salida se inicializan aleatoriamente en el
rango de (-1,1)
for i=0 until n3
        do
        for j=0 until n2
                do
                W_salida[i, j] = random(-1, 1);
                end-for
        end-for
        //Se efectuaran en total L Epocas
for t=0 until L
        do
        for n=0 until N
```

```
        do
                for h=0 until 20
                        Propagacion(n1, n2, n3, W_ocultos, W_salida, xn);
                        dor i=0 until n3
                                se calcula ent_i = Σ_{j=1}^{n2} w_{j,i} a_j
                                se calcula ent_i = Σ_{k=1}^{n1} w_{k,j} i_k
                        end-for
                end-for
        end-for
        ent_j=0
        for k=0 until n1
        do
                ent_j = ent_j + W_ocultos[j, k] * X[n, k];
        end-for
        delta_j=0
        for i=0 until n3
        do
                delta_j = delta_j + W_ocultos[j, i] * delta[i];
                delta_j = delta_j + W_ocultos[i][j] * delta[i];
        end-for
        delta_j = delta_j * g_derivada(ent_j);
        for k=0 until n1
        do
                W_ocultos[j, k] = W_ocultos[j, k] + alpha * X[n, k] * delta_j;
        end-for
        WSF[r,s] = W_salida[r,s];
WHF = W_ocultos;
```

Tabla 24 Algoritmo de retropropagación

Como se nota el algoritmo de retropropagación requiere obtener la salida producida por la red cuando un vector de entrada le es presentado, entonces es que se requiere el algoritmo de propagación [50] (Ver Tabla 25):

Clase de Prueba	Rotar
Variables	
Declaración	Descripción
Int n1	Numero de neuronas en la capa de entrada
Int n2	Numero de neuronas en la capa oculta
Int n3	Numero de neuronas en la capa de salida
Matrix W_hidden[n2,n1]	Contiene los contiene los pesos de las neuronas de la capa oculta
Matrix W_output[n3,n2]	Contiene la información referente a las n3 neuronas que forman la capa de salida.
Vector X[n1]	Vector de entrada q se presenta a la red
Métodos	

```
function Propagacion(int n1, int n2, int n3, double[,] W_ocultos, double[,]
W_salida, double[] X)
        do
            int k, j, i;
            vector  I[n1];
            vector a[n2];
            vector O[n3];
            for k=0 until n1
            do
             I[k] = X[k];
            end-for
                    for j=0 until n2
                    do
                            ent_j = 0.0;
                            for k=0 until n1
                            do
                                  ent_j = ent_j + W_ocultos[j, k] * I[k];
                            end-for
                            a[j] = g(ent_j);
                    end-for
                    for i=0 until n3
                    do
                           ent_i=0.0
```

```
                        for j=0 until n2
                        ent_i = ent_i + W_salida[i, j] * a[j];
                        end-for
                        O[i] = g(ent_i);
                end-for
        end-function
```

Tabla 25 Algoritmo de Propagación

3.10 Integración del reconocedor de gestos

El reconocedor de gestos fue desarrollado de forma modular, lo cual permite que se pueda adicionar gestos en todo momento.

Primeramente el usuario debe estar frente al sensor *Kinect* a una distancia aproximada de 2 a 3 m, esto para lograr que reconozca el esqueleto del usuario de forma correcta.

En todo momento se capturan las coordenadas de las manos, estas se guardan en un vector, para después verificar las condiciones necesarias y poder aplicar el gesto realizado (ver Ilustración 36).

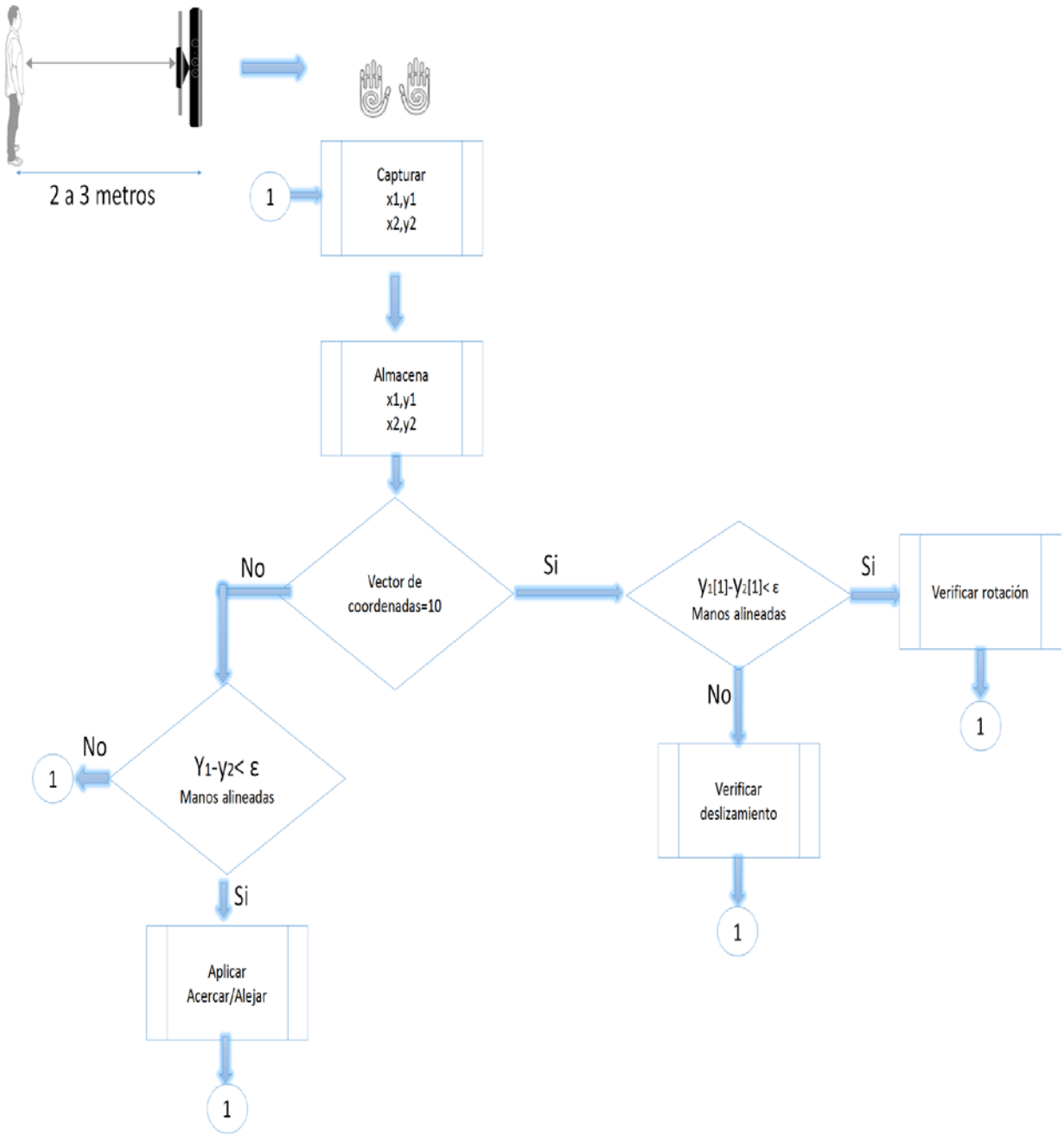


Ilustración 36 Integración del reconocedor de gestos

Capítulo 4: Implementación del Software

PSP puede adaptarse a las necesidades de cada proyecto según las necesidades del usuario. PSP ofrece *Scripts*, *Templates* y métricas que ayudan en la fase de codificación a tener un mayor control del flujo del código, puesto que cada uno de los diseños realizados en esa etapa contienen una notación de diseño apropiada, la notación en los diseños debe ser precisa y completa para representar el diseño total, comprensible y utilizable para otros desarrolladores, debe ser compatible con el lenguaje de implementación que se utilizará [57].

Por tanto PSP, ofrece una ventaja para realizar la codificación puesto que en los diseños se realizan seudocódigos de cada uno de los componentes del sistema lo cual se lleva de forma directa al entorno de programación.

4.1 Requerimientos de Hardware

Las aplicaciones desarrolladas con el SDK de *Kinect* deben ejecutarse en máquinas con sistema operativo Windows 7 nativo (no en máquinas virtuales), debido a que el driver PC *Kinect* y la SDK deben estar instaladas en la computadora en donde se ejecuta la aplicación, funciona tanto como procesadores de 32 y de 64 bits

Las características esenciales del equipo de cómputo se enlistan a continuación [62]:

- Procesador con doble núcleo, 2.66GHz o superior
- Tarjeta gráfica que soporte Microsoft DirectX® 9.0c
- 2GB de RAM, 4 GB recomendada
- 50 Gigabytes disponible en HDDR
- Bus USB 2.0 dedicado

Cabe mencionar las características del sensor *Kinect* a continuación:

- Sensor *Kinect* 1.0 para Windows
- Cable USB/de alimentación

4.2Requerimientos de Software

4.2.1 Direct SDK

Microsoft DirectX[10] es un conjunto de interfaces de programación de aplicaciones (API) para tareas de manipulación relacionadas con multimedia, especialmente programación de video juegos y vídeo en plataformas Microsoft Direct3D [63].

DirectX ampliamente utilizado en el desarrollo de videojuegos para Microsoft Windows , SegaDreamcast , Microsoft Xbox , Microsoft Xbox 360 y Microsoft Xbox One, DirectX SDK es una aplicación que instala varios componentes necesarios para la generación de aplicaciones gráficas en 3D. En concreto, se trata del kit para desarrollo de software que utiliza DirectX.

Para la aplicación a desarrollar se utilizara DirectX para la visualización de cada uno de los modelos tridimensionales de las piezas arqueológicas, puesto que esta librería contiene archivos que mejoran la visualización de objetos 3D y mejora los efectos de sonidos en los videojuegos.

[10] La versión DirectX SDK June 2010 se puede descargar desde: http://www.microsoft.com/en-us/download/details.aspx?id=6812

4.2.2 Microsoft Visual Studio® 2010 Express

Es un entorno de desarrollo integrado (IDE) para sistemas operativos Windows, soporta lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones para muchos otros [64].

C# es un lenguaje de programación multiplataforma orientado a objetos desarrollado y estandarizado por Microsoft como parte de la plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

La sintaxis deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar a la de Java aunque con mejoras derivadas de otros lenguajes.

Aunque C# forma parte de la plataforma .NET, esta es una API, mientras que C# es un lenguaje de programación creado para realizar programas para esta plataforma.

Microsoft recomienda trabajar con visual studio, por la integración con las demás librerías lo cual aligera el trabajo de programación, pues posee herramientas y tutoriales específicos para usar con dicha IDE.

Es el lenguaje de programación en el cual se desarrolló la aplicación de reconocimiento de gestos de la mano.

4.2.3 *Kinect* SDK

Desde el lanzamiento de *Kinect* SDK v.1 para Windows, el equipo de desarrollo ha pasado muchas horas investigando cómo la gente interactúa con *Kinect* (ver Ilustración 37) para Windows y trata de encontrar mejores formas de hacer las cosas.

Ilustración 37 *Kinect* SDK 1.8

El lanzamiento de *Kinect* SDK v.1.8[11] fue el 13 de Septiembre de 2013, la nueva versión tiene compatibilidad con aplicaciones desarrolladas con las versiones anteriores como son 1.0, 1.5, 1.6 y 1.7.

Todo ese trabajo se proyectó en una experiencia de interacción de *Kinect*. La cámara ahora tiene la capacidad de distinguir entre una mano abierta y cerrada y reconocer diferentes movimientos de la mano. Esto abre la puerta a un nuevo campo de posibilidades interactivas, como ser capaces de "tomar" una imagen o un objeto tridimensional y manipularlo, o utilizar un movimiento simple para seleccionar un botón.

Dentro de las nuevas características que ofrece esta librería se encuentran [65]:

[11] Kinect SDK v.1.7 puede descargarse desde: http://www.microsoft.com/en-us/download/details.aspx?id=40278

- Efecto de pantalla verde: Permite borrar el fondo detrás del usuario activo y después intercambiarlo por un fondo artificial.
- Capacidades para modelar objetos 3D: Impresión de objetos en 3D a todo color o agregar fidelidad a las características de los juegos u otras aplicaciones.
- Interfaz de usuario adaptable: muestra cómo construir una aplicación que se adapta de acuerdo con la distancia entre el usuario y la pantalla para detección de gestos y tacto.

Esta librería permitirá obtener imágenes de la cámara de color y la cámara de profundidad, además de obtener el audio de los micrófonos en dado caso que se requiera, básicamente se usa para obtener los puntos del esqueleto.

Actualmente Microsoft lanzo la versión *Kinect* SDK 2.0 el 21 de octubre del 2014, la cual es compatible con Windows 8 y 8.1, pero a su vez requiere del *Kinect* V 2.0 [66].

4.2.4 Microsoft XNA Game Studio 4.0

Microsoft XNA[12] es un conjunto de herramientas con un entorno de ejecución administrado y proporcionado por Microsoft que facilita el desarrollo y gestión de juegos de computadora.

XNA es una herramienta para liberar a los desarrolladores de videojuegos para la manipulación de objetos virtuales eliminando código repetitivo, XNA se anunció

[12] Microsoft XNA Game Studio 4.0 puede descargarse desde: http://www.microsoft.com/en-us/download/details.aspx?id=23714

el 24 de marzo de 2004, en la *Game Developers Conference* en San José, California [67].

El XNA Framework se asegura de que el propio marco se encargue de la diferencia entre plataformas cuando los juegos son soportados desde una plataforma compatible a otra lo que permite que los desarrolladores solo se centren en el desarrollo y la experiencia de juego. XNA Framework se integra con una serie de herramientas, tales como la plataforma múltiple audio (XACT), para ayudar en la creación de contenido. XNA Framework proporciona apoyo para la creación de juegos 2D y 3D y permite el uso de los controladores de Xbox 360 y vibraciones.

Esta librería es de suma importancia pues es la que permitirá cargar y poder manipular el modelo tridimensional, y poder soportar el *Kinect* dentro del juego.

4.3 Construcción del Software

Con el SDK descargado e instalado y el sensor de *Kinect* conectado y preparado para funcionar correctamente ya podemos empezar a programar con Visual Studio 2010 la aplicación utilizando el sensor de *Kinect*.

Primero se creó un proyecto en *visual studio*, con la cual se empezó a construir la aplicación.

4.3.1 Inicializar *Kinect*

EL SDK de *Kinect* permite inicializar el sensor de una manera rápida mediante la integración de la librería de videojuegos XNA.

Primero se debe verificar que el *Kinect* esté conectado y posteriormente obtener la imagen de la cámara de color, esto se logra con la función *DiscoverSensor,* mostrada en la Tabla 26, esto se hace con la versión 1.8 de SDK la cual presenta diferencias a las versiones anteriores.

```
private void DiscoverSensor()
        {
            this.Sensor = KinectSensor.KinectSensors.FirstOrDefault();
            if (this.Sensor != null)
            {
                this.LastStatus = this.Sensor.Status;
                if (this.LastStatus == KinectStatus.Connected)
                {
                    try
                    {   this.Sensor.ColorStream.Enable();
                        this.Sensor.SkeletonStream.Enable();
                        this.Sensor.SkeletonFrameReady += new
EventHandler<SkeletonFrameReadyEventArgs>(Sensor_Skeleton_FrameReady);
                        try
                        {
                            this.Sensor.Start();
                        }
                        catch (IOException)
                        {
                            this.Sensor = null;
                        }
                    }
                    catch (InvalidOperationException)
                    { this.Sensor = null;
                    }
                }
            }
            else
            {
                this.LastStatus = KinectStatus.Disconnected;
            }
        }
```

Tabla 26 Código C# inicializar *Kinect*

Con esta función es posible detectar los diferentes estados de conexión o el motivo por el cual el *Kinect* no se ha inicializado correctamente (ver Ilustración 38):

- El *Kinect* no es original.
- *Kinect* incompatible.
- Requerido.
- Error.
- Inicializando.
- Ancho de banda insuficiente.
- Conectar *Kinect* a corriente eléctrica.
- *Kinect* no preparado.

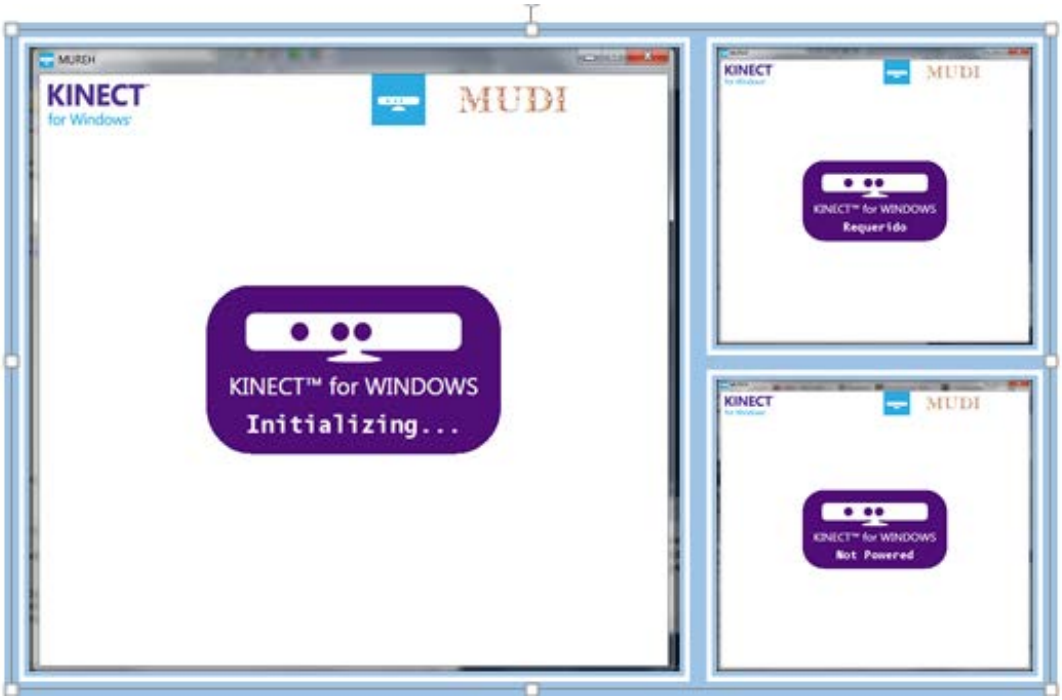


Ilustración 38 Inicializar *Kinect*

4.3.2 Obtener el Esqueleto del sensor *Kinect*

Una vez inicializado el *Kinect*, se prosigue a obtener esqueleto que nos proporciona el sensor de Kinect (ver Tabla 27), con el cual se proporcionan las articulaciones que ayudaron a obtener las coordenadas de las manos las cuales son el área de interés.

```
void Sensor_Skeleton_FrameReady(object sender, SkeletonFrameReadyEventArgs e)
        {
            SkeletonFrame = e.OpenSkeletonFrame();
            if (skeletonFrame != null)
            {
                Skeleton[] skeletonData = new Skeleton[skeletonFrame.SkeletonArrayLength];
                skeletonFrame.CopySkeletonDataTo(skeletonData);
                Skeleton playerSkeleton = (from s in skeletonData where s.TrackingState ==
SkeletonTrackingState.Tracked select s).FirstOrDefault();
                if (playerSkeleton != null)
                {
                    VectorG handRight = new VectorG();
                    VectorG handLeft = new VectorG();
                    SkeletonPoint manoder =
playerSkeleton.Joints[JointType.HandRight].Position;// Joins son puntos de articulación
                    SkeletonPoint manoizq = playerSkeleton.Joints[JointType.HandLeft].Position;//
Joins son puntos de articulación
                    SkeletonPoint spine = playerSkeleton.Joints[JointType.Spine].Position;//
Joins son puntos de articulación
                    pmanoder = this.Sensor.CoordinateMapper.MapSkeletonPointToColorPoint(manoder,
ColorImageFormat.RgbResolution640x480Fps30);
                    pmanoizq = this.Sensor.CoordinateMapper.MapSkeletonPointToColorPoint(manoizq,
ColorImageFormat.RgbResolution640x480Fps30);
                    pspine = this.Sensor.CoordinateMapper.MapSkeletonPointToColorPoint(spine,
ColorImageFormat.RgbResolution640x480Fps30);
...
}
```

Tabla 27 Obtener el esqueleto que proporciona *Kinect*

Para poder detectar gestos de las manos solo compete por ahora detectar las posiciones de las manos en el tiempo, por tanto para obtener dichas coordenadas obtenemos los siguientes valores de cada articulación o *joint* respectivamente.

- *JointType.HandRight*
- *JointType.HandLeft*
- *JoinType.Spine*

Estas coordenadas compusieron cada uno de los vectores de entrada para cada uno de los algoritmos empleados y descritos en el capítulo anterior, los cuales serán capaces de manipular modelos tridimensionales de piezas arqueológicas.

4.3.3 Cargar modelos 3D

Los objetos a manipular son piezas arqueológicas en 3D, las cuales están contenidas en archivos con extensión *fbx,* dichos modelos deben estar contenidos en la carpeta *Load Content* del proyecto, el nombre del archivo fbx al igual que del sonido debe ser igual dado por: “ModeloNo”.

Con los métodos existentes en la librería XNA se creó una variable para alojar los modelos tridimensionales.

```
Model[] modeloactual;
```

En la cual se asignara el nombre del archivo que contiene dicho modelo:

```
modeloactual[i] = Content.Load<Model>("Modelo1.fbx");
```

Además de tener las siguientes variables las cuales sirven para poder rotar, trasladar y escalar el modelo 3D, estas son una matriz de transformaciones:

```
Matrix[] transforms = new Matrix[myModel.Bones.Count];
  myModel.CopyAbsoluteBoneTransformsTo(transforms);
```

Finalmente para poder dibujar en pantalla dicho modelo ejecuta el código que se muestra en la Tabla 28.

```
Matrix view = Matrix.CreateLookAt(new Vector3(240, 320, 1200), new Vector3(0, 0, 0), Vector3.Up);

            Matrix projection = Matrix.CreatePerspectiveFieldOfView(MathHelper.ToRadians(45),
GraphicsDevice.Viewport.AspectRatio, 0.1f, 10000.0f);

            Matrix baseWorld = Matrix.CreateScale(escalas[figseleccionada] * escalainicial) *
Matrix.CreateRotationX(rotars[figseleccionada].X + rotarxinicial) *
Matrix.CreateRotationY(rotars[figseleccionada].Y + rotaryinicial) *
Matrix.CreateRotationZ(rotars[figseleccionada].Z + rotarzinicial) *
Matrix.CreateTranslation(mover);

            foreach (ModelMesh mesh in modeloactual[figseleccionada].Meshes)

            {

                Matrix localWorld = modelTransforms[mesh.ParentBone.Index] * baseWorld;

                foreach (ModelMeshPart part in mesh.MeshParts)

                {

                    BasicEffect e = (BasicEffect)part.Effect;

                    e.World = localWorld;

                    e.View = view;

                    e.Projection = projection;

                    e.EnableDefaultLighting();

                }

                mesh.Draw();

            }
```

Tabla 28 Dibujado de modelo 3D en pantalla

En la Ilustración 39 se muestra la aplicación terminada, con las especificaciones e implementación de los algoritmos descritos en el Capítulo 3, esto antes de aplicar las pruebas integrales.

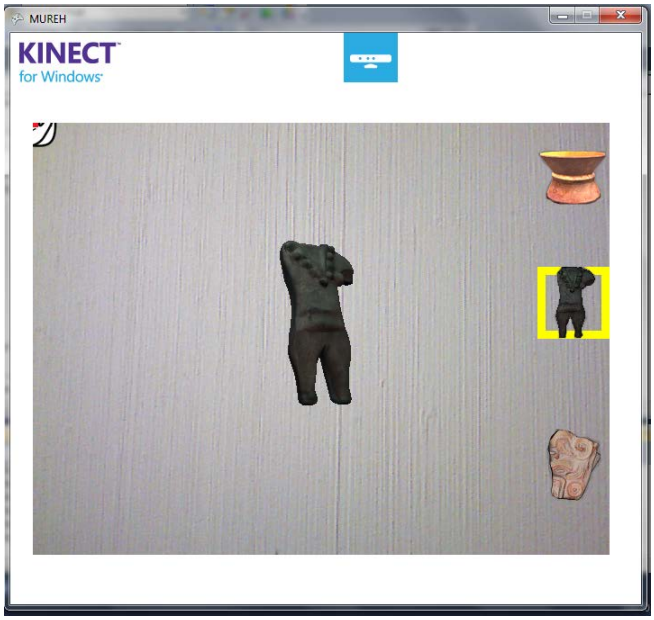


Ilustración 39 Aplicación de reconocimiento de gestos

Efectuados los cambios en la interfaz mostrados en la Ilustración 40, además de añadir el audio que describe a cada una de las piezas arqueológicas a fin de mostrarle al usuario ciertas características del modelo tridimensional.

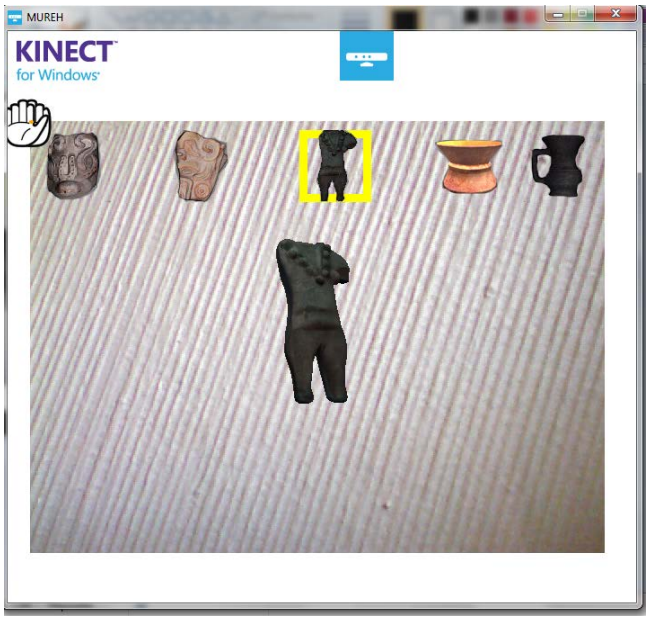


Ilustración 40 Aplicación de reconocimiento de gestos para pruebas de usabilidad

4.3.4 Descripción de piezas arqueológicas

Las descripciones para cada una de las piezas se muestran en la Ilustración 41, esta información fue proporcionada por arqueólogo e investigador del Instituto Nacional de Antropología e Historia (INAH) Iván Rivera, experto en la cultura mixteca.

De acuerdo con los objetivos iniciales de la presente tesis se tomarían 10 piezas, sin embargo el patronato del museo solo nos facilitó el uso de 5 de ellas. Dichas descripciones en el sistema se presentan en audio, mientras el usuario interactúa con la pieza arqueológic

- Representacion de un personaje masculino que porta un collar y un cinturon.
- Procedencia: Cerro de las minas, Mixteca Baja.
- Cronologia: 900 al 152 de nuestra era.

Figurilla

- Esta pieza es usada para poner fragmentos de carbon en su parte superior, y usada en ceremonias importantes a manera de sahumar quiza con incienso.
- Procedencia: Cerro de las minas, Mixteca Baja.
- Cronologia: 300 al 800 de nuestra era.

Bracero

- muestra parcialmente la imagen de un personaje que porta un Yelmo o casco con una figura de un jaguar.
- Procedencia: Cerro de las minas, Mixteca Baja.
- cronologia: 400 al 800 de nuestra era.

Fragmento de urna Ñuiñe 1

- Es una pieza con la imagen de una serpiente.
- Procedencia: Cerro de las minas, Mixteca Baja.
- Cronologia: 400 al 800 de nuestra era.

Fragmento de urna Ñuiñe 2

- Es una jarra cuya caracteristica principal es tener un asa para posiblemente sostener liquidos calientes.
- Procedencia: Cerro de las minas, Mixteca Baja.
- Cronologia: 300 al 800 de nuestra era.

Jarra con asa estribo

Ilustración 41 Descripciones de las piezas arqueológicas utilizadas

Capítulo 5: Evaluación y Resultados

Existen diferentes técnicas para probar un software. Una técnica es construir el software en su totalidad y posteriormente probarlo, tratando de encontrar errores. Aunque a primera vista es muy atractivo para los desarrolladores, dará como resultado un software defectuoso y a su vez será muy difícil o implicara más esfuerzo corregirlo [55].

La propia documentación de *Kinect* SDK, recomienda que el reconocimiento de gestos es un proceso iterativo [12] (Ver Ilustración 42) en el cual debe hacerse un diseño, probar y evaluar dicho gesto a fin de que el gesto se ajuste a mayor número de personas.

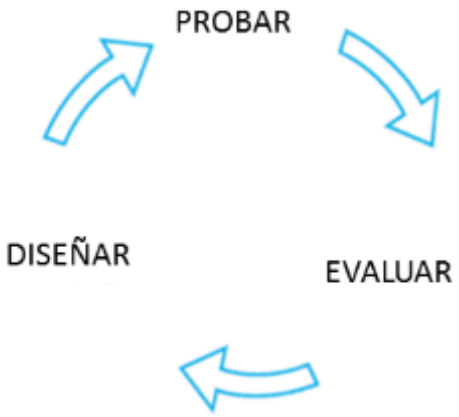


Ilustración 42 Ciclo de construcción de aplicación *Kinect* [12]

El objetivo de este capítulo es mostrar las pruebas obtenidas desde dos perspectivas, primeramente el sistema mediante las pruebas integrales y el usuario mediante pruebas de usabilidad.

5.1 Definición de usuarios

Para poder realizar este tipo de pruebas se utilizaron usuarios reales. Los usuarios para las pruebas integrales fueron definidos como:

- Participaron 3 usuarios, puesto que solo se mide la integración de gestos.
- Estudiantes universitarios.
- Edad de 18 a 25 años.

Los usuarios se eligieron por datos sobre el registro de visitas del año 2012 otorgado por el patronato del MureH. Los cuales indican que al menos durante ese año los visitantes con mayor incidencia son estudiantes universitarios.

Los usuarios para las pruebas de usabilidad, se seleccionaron 5 usuarios de una muestra nueva. Nielsen en su artículo [68] menciona que las mejores pruebas provienen de no más de 5 usuarios. Para esta prueba se requirieron usuarios con las siguientes características:

- Estudiantes
- Edad de 18 a 25 años
- No se requieren conocimientos avanzados en computación.

Participaron 2 hombres y 3 Mujeres estudiantes de diferentes carreras ofertadas en la Universidad Tecnológica de la Mixteca.

5.2 Casos de Prueba

El objetivo de la prueba es: por medio de una interfaz y con ayuda del *Kinect* manipular un modelo tridimensional de una pieza arqueológica existente en el (MureH), usando realidad aumentada para interactuar con la misma.

De ahora en adelante para referirnos a la aplicación de reconocimiento de gestos con las manos será como MUDI (Museo Digital Interactivo).

Los módulos a probar serán los descritos en el Capítulo 3:

- Acercar/Alejar pieza arqueológica.
- Deslizar pieza arqueológica.
- Rotar pieza arqueológica.

El usuario estará enfrente del sensor *Kinect*, tal como se muestra en las especificaciones de *Kinect* para que la interfaz pueda detectar sus manos, con esta se realizara la interacción.

5.3 Pruebas de rendimiento de MUDI

5.3.1 Acercar/Alejar pieza arqueológica

La primer prueba mostro que los usuarios manifestaban que el gesto era correcto y podían realizarlo con facilidad. Este gesto fue detectado de manera geométrica.

5.3.1.1 Precondiciones

Se tomó una muestra de 50 gestos, 10 por cada usuario al realizar el gesto de Acercar/Alejar pieza arqueológica.

5.3.1.2 Resultados

Los resultados obtenidos se muestran en la Ilustración 43, la cual muestra la cantidad de gestos que MUDI detectaba como válidos y no válidos para cada usuario.

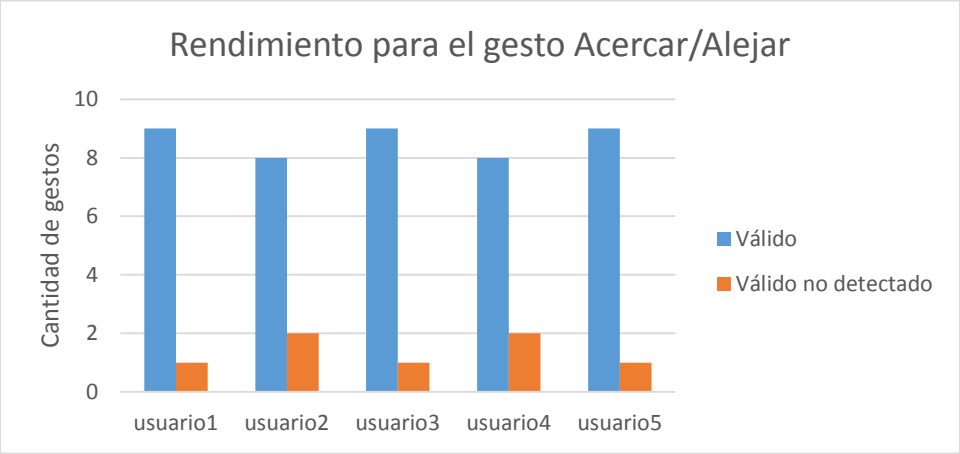


Ilustración 43 Rendimiento para Acercar/Alejar

La cual muestra que el gesto implementado de forma geométrica es bueno, dado que aproximadamente por 9 gestos que detecta como válidos, 1 no logra asociarlo al gesto.

5.3.2 Deslizar pieza arqueológica

5.3.2.1 Precondiciones

Se tomó una muestra de 50 gestos, 10 por cada usuario cuando realizaban el gesto de deslizar pieza arqueológica.

5.3.2.2 Resultados

Los resultados se muestran en la Ilustración 44, en los cuales se observa que por cada 8 gestos, 2 de estos el sistema no pudo identificarlos como un gesto de deslizar.

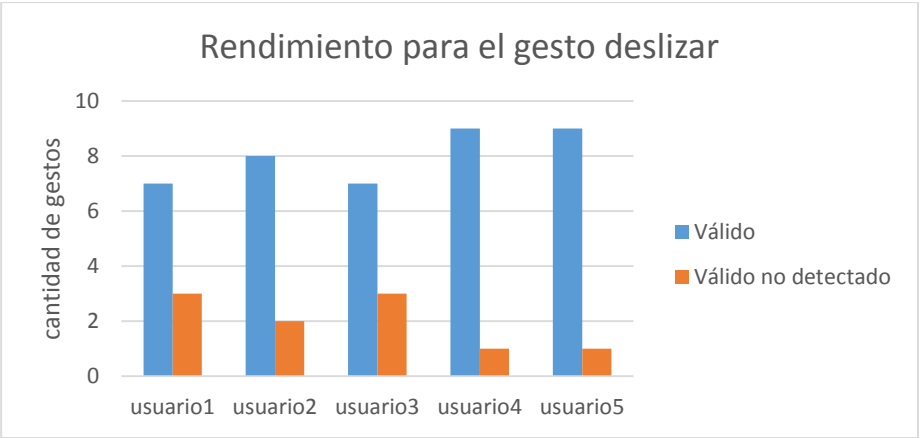


Ilustración 44 Rendimiento para deslizar

5.3.3 Rotar pieza arqueológica

5.3.3.1 Precondiciones

Para efectos de probar el reconocimiento de este gesto se considera una muestra de 50 (ver Tabla 29**),** los gestos fueron obtenidos de los movimientos realizados por los diferentes usuarios**.**

Muestras del gesto rotación	
Clase	Cantidad
P	25
N	25

Tabla 29 Muestras del gesto rotación

En donde P indica que el gesto es válido y N indica que el gesto es inválido. La Ilustración 45 muestra la trayectoria de un ejemplo aparentemente válido de un gesto de rotación, en el cual se hizo un procesamiento de las coordenadas, eliminando repetidas y quitando perturbaciones que el *Kinect* detecta durante el movimiento, sin embargo estas no son perceptibles por el usuario.

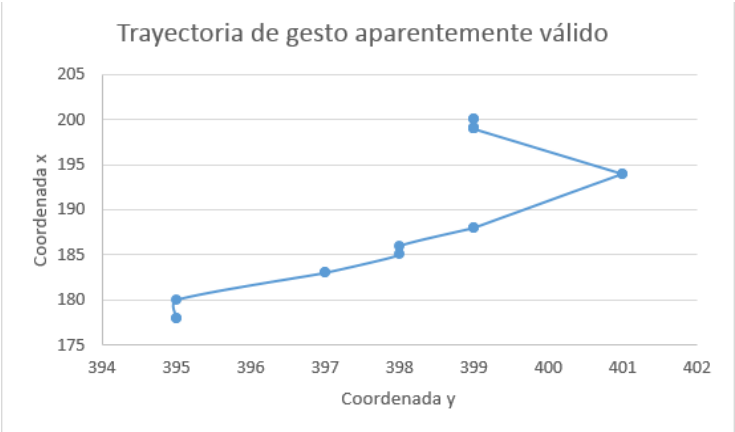


Ilustración 45 Gesto válido

Obsérvese que la gráfica muestra una curva cuando este es válido a diferencia de lo que se muestra en la Ilustración 46 la cual indica el comportamiento para un gesto no válido.

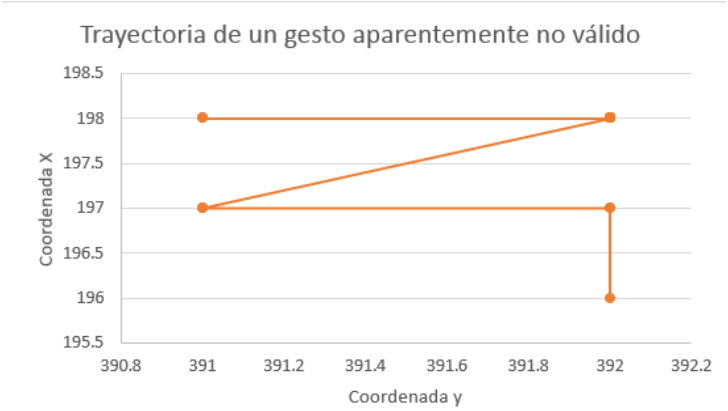


Ilustración 46 Gesto inválido

5.3.3.2 Entradas

- Una matriz de 50*40, la cual contiene 50 ejemplos de gestos, cada uno con 10 coordenadas de posición de cada mano, lo cual se traduce en 40 datos por ejemplo.
- Un vector de 50*1, el cual contiene si un gesto es válido o inválido.

5.3.3.3 Procedimiento

Para detectar el gesto de rotación se utilizó un modelo de red neuronal multicapa con 5 neuronas en la capa oculta y en la capa de salida solo una neurona, el algoritmo se ejecutó con 1000 iteraciones.

Se realizaron 10 corridas al clasificador, obteniendo un rendimiento promedio del 79% en la clasificación, es decir el porcentaje de ejemplos buenos y malos que es capaz de detectar el clasificador.

En cada corrida se usó el 80% de los elementos para entrenamiento y el 20% restante para pruebas (ver Ilustración 47). Es importante recalcar que en todas las corridas los elementos para entrenamiento y pruebas fueron elegidos aleatoriamente de la muestra total.

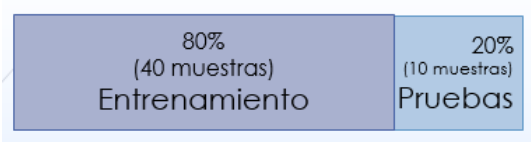


Ilustración 47 Muestras utilizadas para la fase de entrenamiento y pruebas

Un ejemplo de las 10 corridas para el algoritmo con las 50 muestras se muestra en la Tabla 30:

Resultado	Cantidad de muestras
TN	20
TP	18
FN	5
FP	7

Tabla 30 Ejemplo de 10 corridas para un ejemplo de gesto

En donde:

- TN = cantidad de gestos correctamente clasificados en la clase N (correctamente rechazados)
- TP = cantidad de gestos correctamente clasificados en la clase P (correctamente identificados)
- FN = cantidad de gestos incorrectamente clasificados en la clase N (incorrectamente rechazados)
- FP = cantidad de gestos incorrectamente clasificados en la clase P (incorrectamente identificados)

5.3.3.4 Resultados

Los resultados son clasificados de la siguiente manera:

- **Sensibilidad** es la capacidad de detectar los verdaderos positivos, es decir los gestos de la clase P que identificó correctamente.
- **Exactitud** es la proporción de resultados verdaderos entre el número total de casos examinados.

Donde:

- Sensibilidad = TP/(TP+FN)
- Exactitud = (TP+TN)/(TP+FN+FP+TN)

En la siguiente tabla (ver Tabla 31) se muestran los resultados para los ejemplos utilizados en la prueba:

Medida	Valor
Sensibilidad	78%
Exactitud	76%

Tabla 31 Resultados para el clasificador

Aunque son resultados menores al 80% fueron considerados como aceptables debido a que cuando los usuarios están utilizando la aplicación, realizan los gestos

de manera rápida, entonces al no detectar el gesto lo vuelven a repetir inmediatamente y no perciben la repetición.

5.4 Pruebas integrales

Las pruebas de integración son una técnica sistemática para construir la arquitectura del software mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar los componentes probados de manera individual y construir una estructura de programa que se haya establecido por diseño.

La integración incremental es la antítesis del enfoque *big bang*[13] [55]. El programa se construye y prueba en pequeños incrementos, donde los errores son más fáciles de aislar y corregir, así las interfaces tienen más posibilidades de probarse por completo además de que se puede aplicar un enfoque de prueba sistemático [55].

La aplicación de reconocimiento de gestos debe probarse de manera total, y del mismo modo los OST definidos en el Capítulo 3 se deben cumplir en su totalidad.

5.4.1 Precondiciones

La computadora en la cual se ejecuta el sistema debe contar con los requerimientos de Hardware y Software mostrados en el Capítulo 4.

[13] El enfoque de big bang: consiste en combinar todos los componentes por adelantado, es decir, todo el programa se prueba como un todo, usualmente resulta un “caos”, se descubre un conjunto de errores, la corrección se dificulta.

El usuario puede ver la detección y dirección de los movimientos de las manos, la interfaz pondrá las siguientes imágenes (ver Ilustración 48) sobre la mano correspondiente:

Ilustración 48 Símbolo para identificar las manos del usuario

El usuario puede observar el conjunto de instrucciones, las cuales indican los tipos de movimientos a realizar para completar el gesto (ver Ilustración 49).

Ilustración 49 Instrucciones para poder utilizar la aplicación MUDI

5.4.2 Procedimiento de Prueba: Acercar/Alejar pieza arqueológica

Esta prueba consiste en Acercar/Alejar una pieza arqueológica la cual se encuentra en el centro de la pantalla.

5.4.2.1 Entradas

- Coordenadas de cada una de las manos, izquierda y derecha.

5.4.2.2 Resultados

- Cambiar la escala de la pieza arqueológica, ya sea para aumentar la escala o disminuirla.

5.4.2.3 Procedimiento de la prueba

1. Con la aplicación ejecutándose, el usuario se ubica frente al sensor *Kinect*.
2. El sistema dibuja sobre sus manos, una figura de una mano sobre la misma.
3. El usuario mueve sus manos juntándolas o separándolas en línea recta.
4. El usuario finaliza cuando la pieza arqueológica queda en la escala de su agrado.

5.4.2.4 Criterios de aceptación

- El usuario logra cambiar la escala de la pieza arqueológica, sin que el sistema ejecute otro gesto.

5.4.3 Procedimiento de Prueba: Deslizar pieza arqueológica

Esta prueba consiste en cambiar una pieza arqueológica, eligiendo la de su agrado de las mostradas en el menú, puede cambiar hacia abajo o arriba.

5.4.3.1 Entradas

- 20 Coordenadas de la mano derecha, para mover de abajo hacia arriba la pieza arqueológica.
- 20 coordenadas de la mano izquierda, para mover de arriba hacia abajo la pieza arqueológica.

5.4.3.2 Resultados

- Elegir una pieza arqueológica distinta a la que se muestra al ejecutar la aplicación.

5.4.3.3 Procedimiento de la prueba

1. Con la aplicación ejecutándose, el usuario se ubica frente al sensor *Kinect*.
2. El sistema dibuja sobre sus manos, una figura de una mano sobre la misma.

3. El usuario mueve su mano derecha de abajo hacia arriba o su mano izquierda de arriba hacia abajo, en línea recta para poder elegir la pieza de su agrado.
4. El usuario finaliza cuando muestre una pieza arqueológica distinta a la inicial.

5.4.3.4 Criterios de aceptación

- El usuario logra elegir la pieza arqueológica de su gusto.

5.3.4 Procedimiento de Prueba: Rotar pieza arqueológica

Esta prueba consiste en rotar la pieza arqueológica en el eje Z, en dirección según las manecillas de reloj o viceversa.

5.4.4.1 Entradas

- Coordenadas de la mano derecha, la mano izquierda y el punto medio del esqueleto.

5.4.3.2 Resultados

- Girar la pieza arqueológica en sentido a las manecillas del reloj o viceversa, sin que otro gesto se ejecute.

5.4.3.3 Procedimiento de la prueba

1. Con la aplicación ejecutándose, el usuario se ubica frente al sensor *Kinect*.
2. El sistema dibuja sobre sus manos, una figura de una mano sobre la misma.
3. El usuario ve una línea roja punteada que sale de su mano derecha y cruza por el punto medio de su esqueleto.
4. El usuario mueve su mano izquierda, tratando de convertir la línea roja en verde.
5. El usuario sigue moviendo las manos, conservando la línea verde hasta rotar al ángulo de su elección.

5.4.3.4 Criterios de aceptación

- El usuario logra rotar la pieza arqueológica, sin que otro gesto se ejecute.

5.4.5 Resultados de las pruebas integrales

Para la ejecución de las pruebas, fue necesario contar con los siguientes requerimientos.

1. Lugar despejado.
2. Conjunto de métricas.
3. Cuestionario de usuario.

Las métricas para cada caso de prueba se definieron como: velocidad, interfaz, funcionalidad y la integración. Los valores se manejaron en escalas del 0 al 10, donde el 10 representa sencillez y el 0 mucha dificultad al momento de realizar cada una de las tareas solicitadas.

Los resultados se muestran en la siguiente Tabla 32 después de hacer el consenso con los resultados de cada uno de los usuarios

Caso de prueba	Velocidad	Interfaz	funcionalidad	integración
Acercar/Alejar	10	8	9	9
Deslizar	8	8	8	8
Rotar	6	7	5	8

Tabla 32 Calificaciones de las tareas realizadas por los usuarios en pruebas integrales

El caso de prueba de acercar/alejar, los usuarios manifestaban que en un principio tenían dificultad para alinear sus manos, sin embargo la tarea fue completada para todos, la Ilustración 50 muestra cuando los usuarios realizaban el gesto.

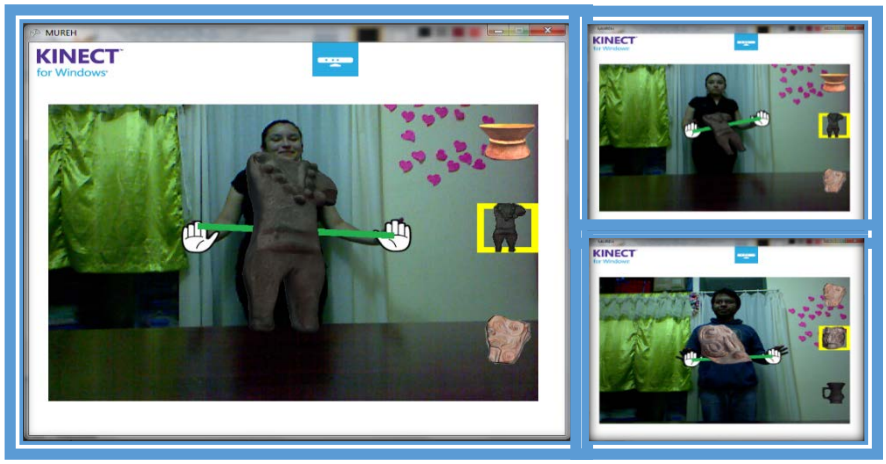


Ilustración 50 Pruebas integrales para el *zoom*

Durante la ejecución del caso de prueba deslizar pieza, mostrado en la Ilustración 51, los usuarios manifestaban que era confuso aplicar ambas manos para poder elegir la pieza, la retroalimentación era mala.

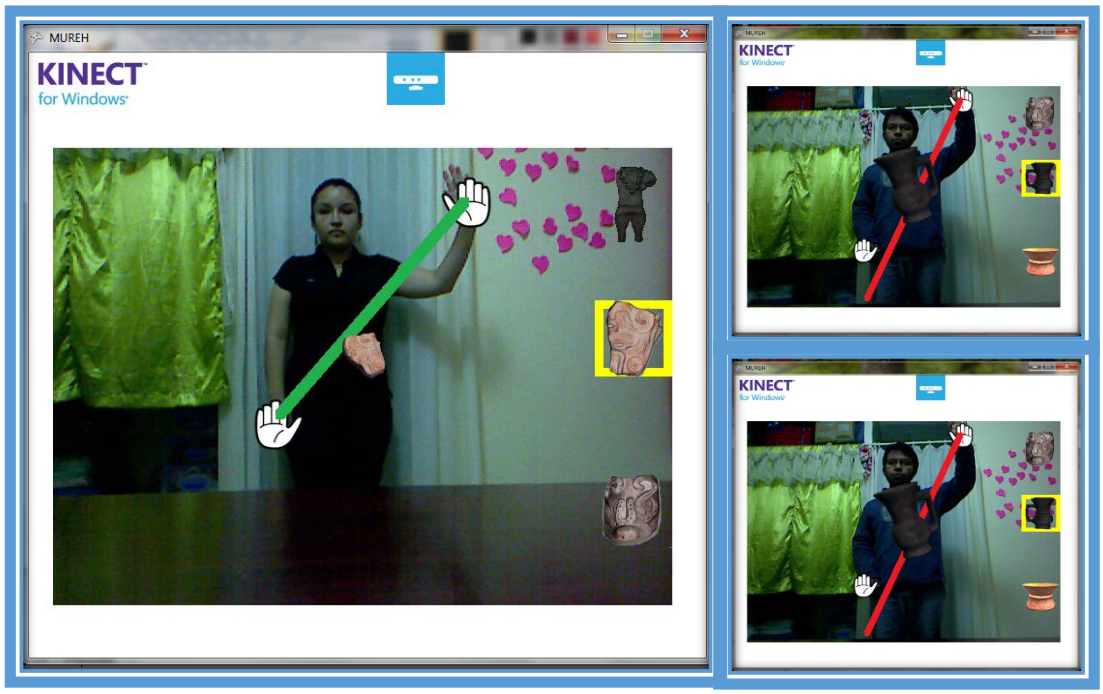


Ilustración 51 Pruebas integrales del deslizamiento

Para el caso de prueba de rotar, los usuarios manifestaban que era mala, no podían encontrar el punto de equilibrio, a pesar que manifestaban que la retroalimentación era adecuada. En la Ilustración 52 se muestra cuando los usuarios emplean el gesto de rotación para poder girar la imagen según su necesidad.

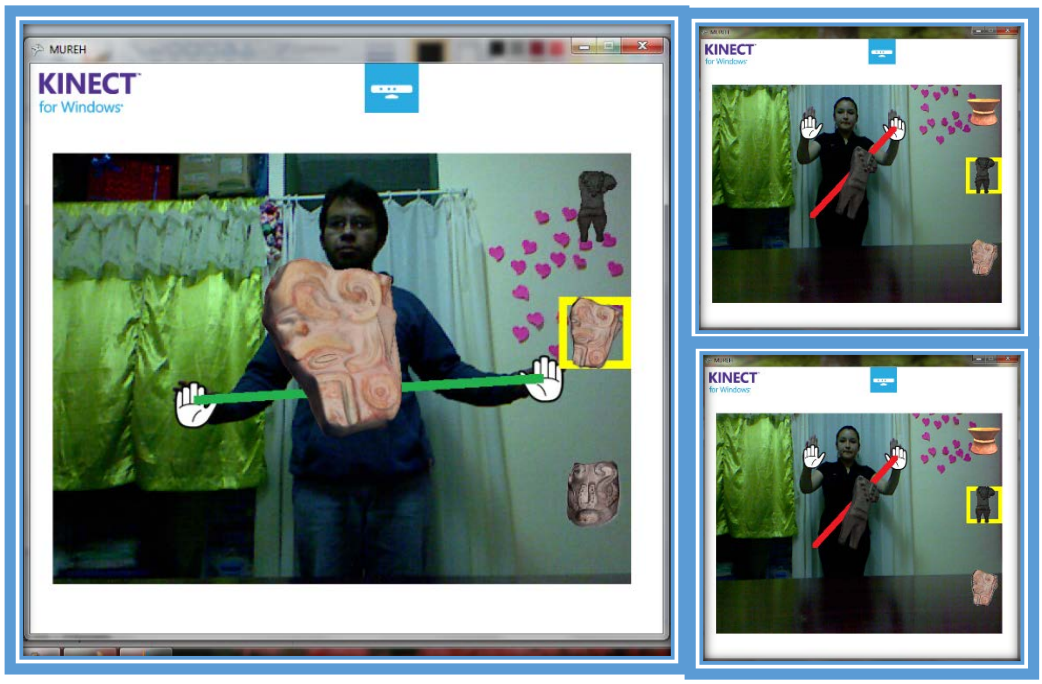


Ilustración 52 Pruebas integrales de rotación

Para todos los usuarios el nivel de usabilidad era de 7, puesto que a su criterio, el sistema podría mejorar en cuanto a:

- Velocidad: Pues a su parecer la velocidad era Regular.
- Mejorar instrucciones: No acostumbran a leer, prefieren ver un ejemplo antes de ejecutar una acción.
- Sonido: El sonido consideran es parte esencial de un juego, además que era más fácil recordar la descripción de la pieza.
- Cambiar el gesto de rotación: Para ellos es complicado, fuera de contexto y estresante.

Por tanto, la calificación total al sistema fue de tres, siendo entonces no aceptable para los usuarios representativos ocupados en esta prueba, para la calificación total se usó una escala del 0 al 5 como se observa en la Ilustración 53.

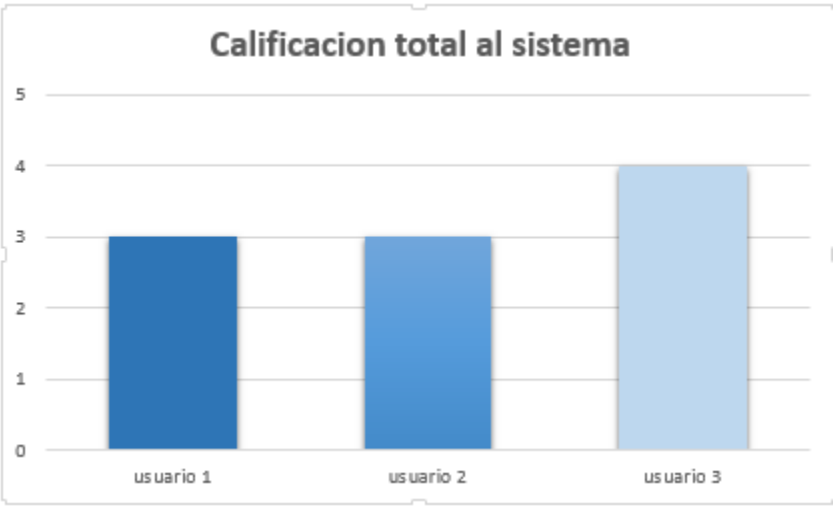


Ilustración 53 Calificación total al sistema según pruebas con usuarios

5.4 Pruebas de usabilidad

El usuario es el elemento más importante durante el desarrollo de un sistema, puesto que se diseña creando funciones que se adapten a sus necesidades, logrando una interacción en la cual el usuario se sienta cómodo.

Esta prueba nos permitió conocer el nivel de aceptación de los usuarios hacia el sistema, en cuanto iconografía, y como realizar las acciones según su intuición, hablando de una NUI.

Se requirió tener definidas las tareas, usuario y contexto de uso, los primeros mostrados con anterioridad.

Con esta prueba, se obtuvieron factores que permitan determinar el nivel de aceptación de cada uno de los usuarios y poder generalizar los resultados y con ello lograr que aumente el número de visitantes en el museo, por ello se deben medir los siguientes parámetros [69]:

- Dificultad de la tarea.
- Tareas completadas.
- Atractivas.
- Satisfacción.

- Aprendizaje para manipular la aplicación.
- Tiempo para completar la tarea
- Velocidad del sistema

5.4.1 Contexto de uso

MUDI, es una aplicación cuyo objetivo es poder interactuar con cada una de las piezas arqueológicas existentes en el museo, puesto que en el libro de visitas los usuarios indican que es aburrido por no interactuar con las piezas.

Se requirió de un ambiente controlado en el cual se le pidió al usuario que pensara que se encontraba bajo el siguiente escenario:

<Un día te encuentras con un amigo de la Universidad y te cuenta de un nuevo sistema que están empleando en el MureH, este sistema es utilizado para poder manipular piezas arqueológicas usando nueva tecnología, él te cuenta que es una nueva experiencia además que es novedoso>

5.4.2 Procedimiento de la prueba

La prueba se efectuó mediante un Script, en el cual el facilitador, le indicaba al usuario el motivo de la prueba, señalando que pensara en voz alta, asimismo que se estaba evaluando la interfaz y no a él.

Atendiendo a los comentarios de las pruebas integrales al usuario antes de poder interactuar con la aplicación, se le presentaron las instrucciones de MUDI mediante un video como se muestra en la Ilustración 54, al inicializarse la aplicación, puesto

que según los resultados de las pruebas integrales se concluyó que regularmente los usuarios no leen instrucciones.

Ilustración 54 Video de instrucciones MUDI

Después de observar las instrucciones siguiendo el Script, se le indico al usuario que realizara cada una de las tareas, las cuales fueron definidas con anterioridad y son las mismas que se realizaron durante las pruebas integrales.

5.3.5 Ejecución de las pruebas de usabilidad

Las pruebas se realizaron en el USALAB (Laboratorio de Usabilidad), ubicado en el Instituto de Computación, perteneciente a la Universidad Tecnológica de la Mixteca.

La primera tarea consistió en elegir una pieza arqueológica como en la Ilustración 55, donde se les pidió a los usuarios empezar a interactuar con el sistema, el cual contenía una variación a la versión probada durante la integración, donde los usuarios manifestaron que el menú de piezas debía situarse en el margen superior de la pantalla del sistema.

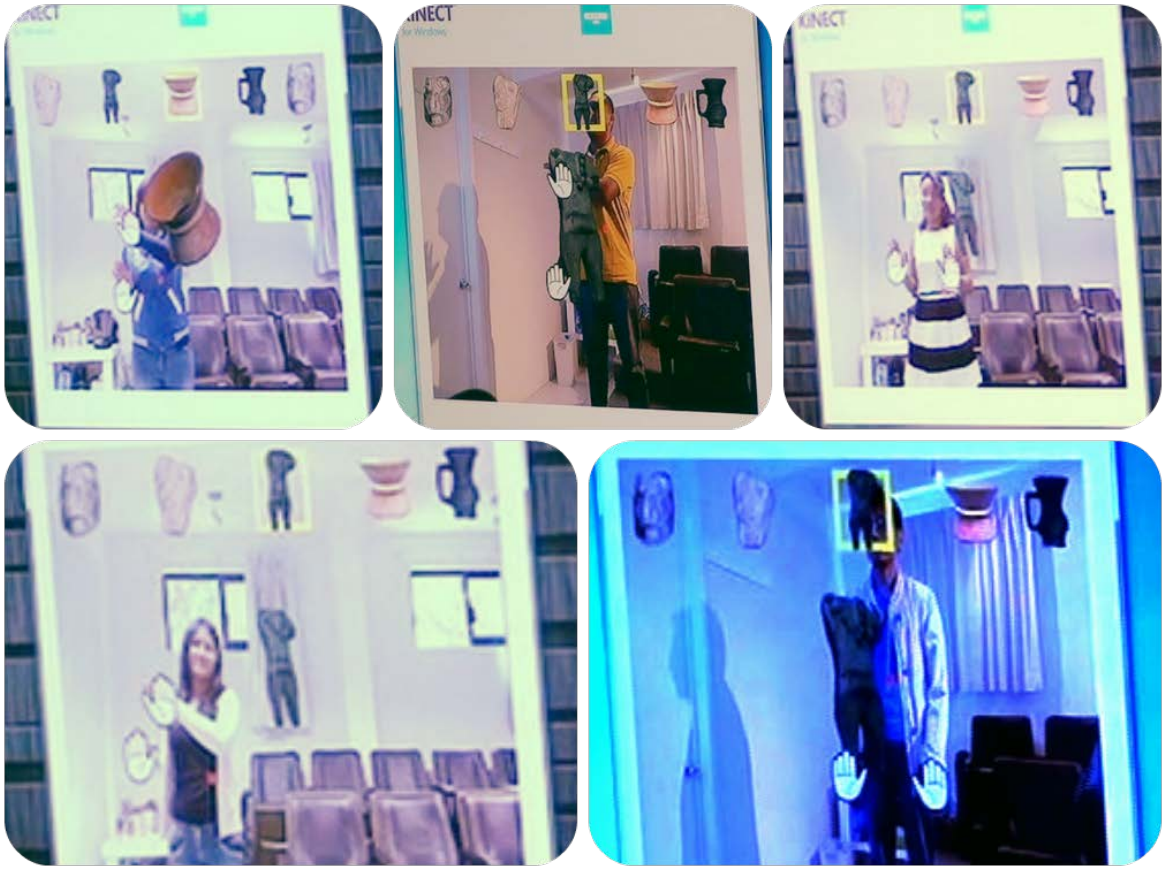


Ilustración 55 Deslizamiento Pruebas de usabilidad a MUDI

El *zoom* el cual era un gesto aceptado por los usuarios desde las pruebas integrales, se mantuvo la retroalimentación, manteniendo la línea punteada en verde tal como se muestra en la Ilustración 56.

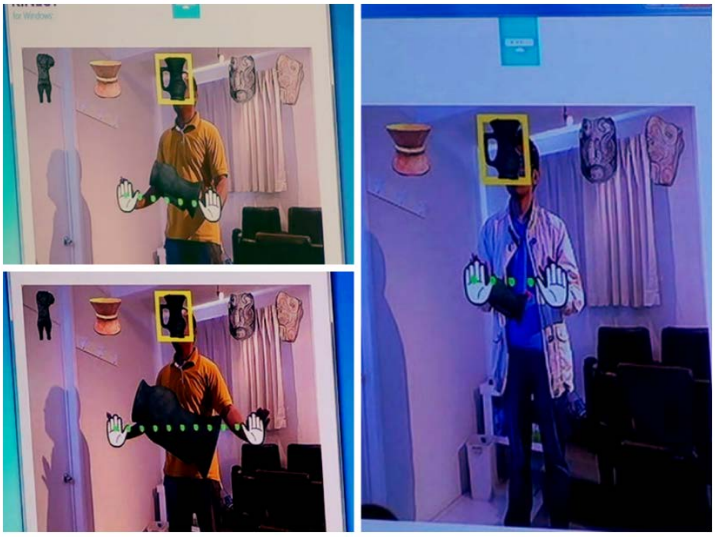


Ilustración 56 Retroalimentación del *Zoom* aplicación MUDI

Los usuarios se manifestaron contentos, puesto que podían manipular la pieza con tan solo alejar o juntar sus manos, véase Ilustración 57 y así poder ver a más detalle la construcción de cada uno de los modelos tridimensionales.

Ilustración 57 *Zoom* Aplicación MUDI

El último gesto evaluado fue la rotación (ver Ilustración 58), para la cual se cambió el flujo del gesto, los usuarios manifestaron su aceptación hacia el gesto.

Ilustración 58 Rotación pieza arqueológica MUDI

La población de usuarios femeninos véase Ilustración 59, de igual forma manifestó buena aceptación ante el gesto de rotación, logrando con ello terminar la prueba.

Ilustración 59 Rotar pieza MUDI

5.3.5 Resultados de las pruebas de usabilidad

El resultado de esta prueba fue obtenido mediante cuestionarios y grabaciones de los usuarios al momento de realizar las pruebas.

En donde se preguntó a cada uno de los usuarios si conocían el MureH, las respuestas se registraron en la Tabla 33:

Usuario	Si	No	Razón(es).
1	×		Visitas escolares.
2	×		Curiosidad.
3	×		Curiosidad.
4		×	No sabía que existía un museo.
5		×	No sabía que existía un museo.

Tabla 33 Usuarios que han visitado el MureH

El promedio de calificaciones obtenidas por cada tarea fue en una escala del 0 al 5, donde 0 representaba cuando una tarea no fue completa y el 5 la tarea completada satisfactoriamente, como se muestra en la Ilustración 60.

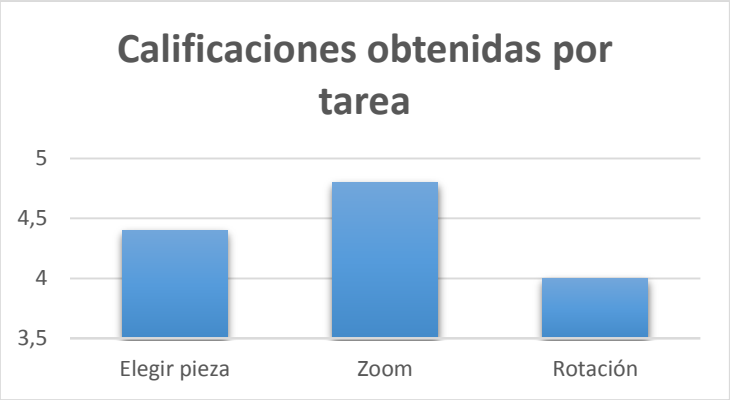


Ilustración 60 Tareas realizadas en las pruebas de usabilidad

Primeramente la elección de la pieza obtuvo un puntaje de 4.4, en el cual se refleja que los usuarios están satisfechos en que solo se use una mano y evitar confusión al incluir la mano izquierda como se había hecho antes.

El *zoom* en las pruebas pasadas obtuvo buena aceptación por los usuarios, pues fue el gesto con menos dificultad, en las pruebas de usabilidad obtuvo un puntaje de 4.8, lo cual indica que es aceptable.

La rotación de la pieza una de los gestos complejos, con las correcciones realizadas, los usuarios manifestaron que era más fácil, divertido y entretenido, solo que presentaba un cierto retraso, debido al análisis que se produce para decidir si es un gesto de rotación válido, el cual obtuvo un puntaje de 4.

Recopilando las frases de los usuarios mientras realizaban las tareas, la frase que todos los usuarios mencionaban es: “me gusta” acompañado con una sonrisa, no obstante existieron más recomendaciones.

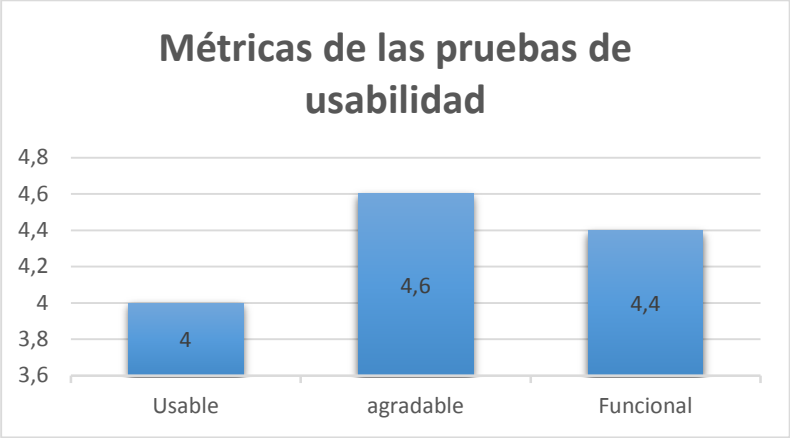


Ilustración 61 Métricas de las pruebas de usabilidad

Los índices medidos en esta prueba son usabilidad, funcionalidad y que tan agradable es el sistema, los cuales en usabilidad se obtuvo un puntaje de 4, lo que indica que el sistema tiene buena usabilidad; la funcionalidad fue de 4.5 puesto que el tiempo de detección del gesto de rotación tarda poco más de 1 segundo; y finalmente el sistema es agradable con un puntaje de 4.6, véase Ilustración 61, lo cual indica que con las mejoras recomendadas a la interfaz estos puntajes pueden mejorar.

Finalmente la interfaz obtenida tomando en cuenta las recomendaciones de las personas se mejoró, en la cual se le añadió un nuevo diseño, que contextualizara la aplicación mejorando la experiencia de uso.

La Ilustración 62 muestra cuando no se reconoce el *Kinect* por alguna cuestión mencionada en el Capítulo 4.

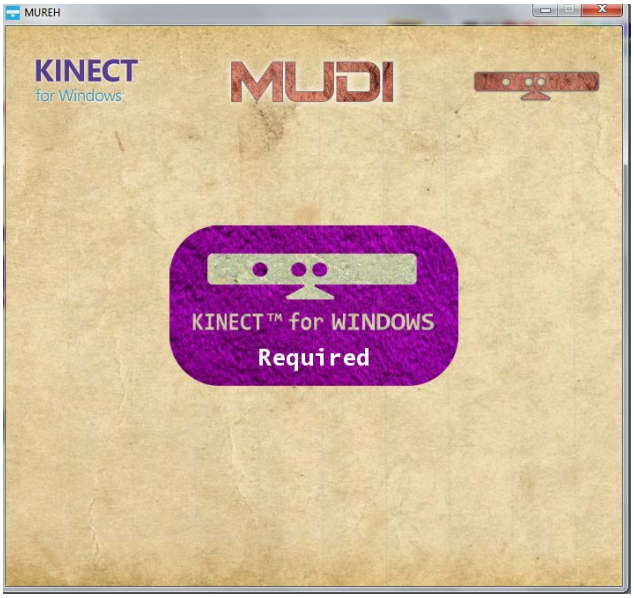


Ilustración 62 Diseño final aplicación MUDI *Kinect* no detectado

En el momento en el que la interfaz detecta el sensor *Kinect*, se optó por un diseño en pergamino que tuviera una apariencia natural y que fuera amena para el usuario, obteniendo la interfaz MUDI final mostrada en la Ilustración 63.

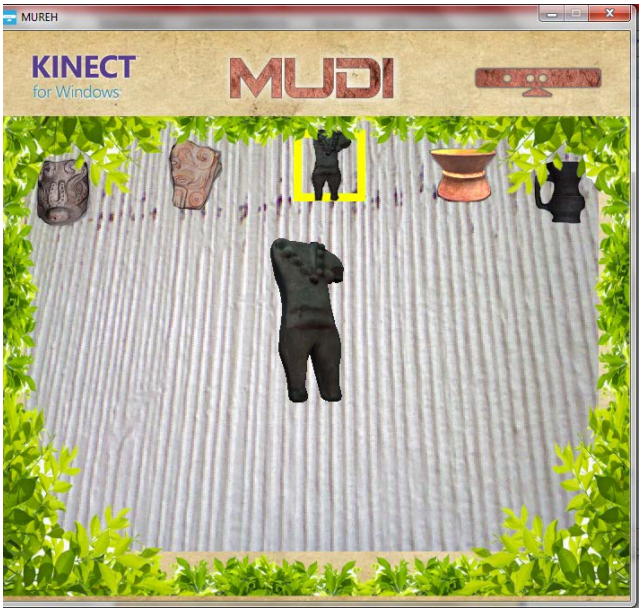


Ilustración 63 Interfaz MUDI final después de pruebas de usabilidad

Sumando mejorar la iconografía de MUDI se optó por cambiar el diseño de la silueta de las manos a una representación prehispánica de las manos. En la Ilustración 64 se muestran las manos blancos que indican que la aplicación está lista para empezar a recibir la interacción y las manos azules que le indican al usuario que puede empezar la rotación del modelo 3D de la pieza arqueológica.

a) Manos Blancas b) Manos Azules

Ilustración 64 Iconos de las manos del sistema MUDI final

Capítulo 6: Conclusiones y Trabajo Futuro

6.1 Conclusiones

Una interfaz natural de usuario se puede encontrar en estos momentos en todos los ámbitos, sean estos políticos, sociales, educativos, por tanto se construyó una interfaz capaz de traducir movimientos del cuerpo a comandos de interacción utilizando el *Kinect* como medio entre el usuario y dicha interfaz.

Puesto que el *Kinect* es un sensor Microsoft la aplicación fue desarrollada para el mismo, usando C # como lenguaje base, SDK *Kinect* 1.8: debido que la versión posterior requiere de un *Kinect* V2, el cual en este momento tiene un precio alto en el mercado, aun así las versiones anteriores existe cierta incompatibilidad.

Como base inicial de esta tesis, se hizo una investigación sobre las interfaces naturales de usuario y la interacción natural trasladada a las diferentes consolas de videojuegos, desarrolladas por diversas marcas comerciales, enfocándome en el sensor *Kinect*, y su API de desarrollo.

Se estudió el funcionamiento del *Kinect*, la arquitectura, y la forma de abstraer las imágenes de cada una de las cámaras, así como las capacidades del rastreo del esqueleto usando el SDK desde la versión 1.0 hasta la 1.8, con lo cual pude identificar las mejoras tanto en rendimiento, velocidad y facilidad al momento de trasladar al entorno de desarrollo.

Motivado por las capacidades de los televisores SMART TV de Samsung, me di a la tarea de empatar los gestos utilizados por SMART TV y la capacidad del sensor *Kinect*. Por tanto, la secuencia de movimientos de las manos se definen por una

serie de reglas o acciones, puesto que la combinación de acciones para definir comandos brinda gran flexibilidad para detectar movimientos de una o ambas manos, permitiendo utilizar el sistema para controlar una variedad de aplicaciones bajo un paradigma de interacción natural.

El sistema MUDI se adapta al tamaño y a la posición del usuario respecto a la cámara, ofreciendo retroalimentación en todo momento.

Se colocó una imagen sobre la silueta de las manos del usuario, la cuales siguen la articulación de la misma, además, se incorporó sonido para mejorar la interacción logrando gran aceptación entre los usuarios.

Normalmente los usuarios no acostumbran a leer instrucciones y si lo hacen es difícil recordar por un lapso de tiempo grande, por lo tanto, se optó por introducir la explicación de las piezas arqueológicas con audio, esto logro que los usuarios manifestaran su aceptación por escuchar a alguien que narrara la descripción.

El sistema de detección de gestos se realizó tomando en cuenta la metodología PSP, la cual me permitió mantener organizada la información respecto a variables y métodos.

Con ello se pudieron detectar las etapas del reconocedor de gestos la cual, consiste en: obtener las coordenadas de las manos, identificar las reglas válidas para cada gesto y finalmente aplicar el gesto a dicho modelo tridimensional, logrando un porcentaje aceptable de exactitud, debido que la detección del gesto es en tiempo real.

La implementación de la red neuronal artificial fue fácil, aunque cabe señalar que según sea el ejemplo o gesto que se quiera detectar es necesario crear un modelo que se ajuste a las características específicas del problema.

Se logró desarrollar la aplicación denominada MUDI, para que los usuarios puedan recordarla, es un software, construido para el SO Windows 7, capaz de adoptar la integración de nuevos gestos, pues tiene una base sólida para construir nuevos mecanismos de interacción natural.

La interfaz está probada por usuarios, los cuales aportan mejoras, con lo cual se garantiza que el usuario tenga un grado de satisfacción al momento de utilizar MUDI, y este pueda recomendarla a las demás personas.

Se detectó que los archivos de audio deben ser creados en Windows nativo, puesto que el IDE de desarrollo muestra incompatibilidad con los mismos.

Después de haber realizado este trabajo y los resultados obtenidos por las pruebas de usabilidad, es posible que el MureH tenga mayor afluencia si se hace la debida difusión y promoción, puesto que las personas que hicieron las pruebas de usabilidad afirmaban que asistirían y recomendarían la visita al mismo.

6.2 Trabajo Futuro

Una forma de que el sistema mejore, es mediante las sugerencias obtenidas en el desarrollo de las pruebas, y a su vez con la gama de posibilidades que ofrece la librería de desarrollo SDK de *Kinect*.

A continuación se enlistan las mejores y oportunidades para el sistema MUDI:

- Implementación de gestos para manipular la rotación de la pieza arqueológica en el eje *X* y *Y*, puesto que ahora solo se implementó la rotación en el eje *Z*.
- Obtener las características de las manos para detectar movimientos con los dedos, mediante la clasificación sobre las imágenes de profundidad y con esto realizar aplicaciones en las que involucren movimientos finos como un teclado o simular el uso de un mouse.
- Implementar el “Green screen”, para contextualizar con paisajes alusivos a el lugar en el cual fue encontrada cada pieza arqueológica, para que el usuario experimente escenarios inversivos de entornos virtuales.
- Al usar dos Kinect`s es posible entonces poder detectar varios esqueletos y por lo tanto poder asignar actividades grupales colaborativas.

Bibliografía

[1] I. SOMMERVILLE y T. J. A. D. TORRES, Ingenieria del Software, Mexico: Pearson Education, 2002.

[2] J. Nielsen, Usability enginnering (Interactive Technologies), Morgan Kaufmann/Academic Press, 2014.

[3] A. N. a. T. P. M. Stuart K. Card, The psychology of Human Computer Interaction, USA: L, Erlbaum Associates Inc., 1983.

[4] L. d. H. e. I. d. S. Grupo LoUISE, «Nuevos Entornos y Paradigmas de Interacción,» Universidad de Castilla – La Mancha .

[6] A. L. a. D. W. Jhilmil Jain, The future of natural user interfaces, Canada: ACM, 2011.

[7] J. C. G. C. a. A. S. Jean Vanderdonckt, Multimodality for plastic User Interfaces: Models Methods and Principles, Berlin, 2007.

[8] A. Valli, The design of natural interacion, USA, 2008.

[13] E. R. J. H. E. P. C. M. G. Moisés Ramírez, «"Design of interactive Museographic exhibits using Augmented Reality",» de *22st International Conference on Electronics Communications and Computers. (CONIELECOMP 2013)*, Cholula, Puebla, 11-13 de Marzo de 2013.

[17] M. R. E. N. D. F. J. H. M. G. E. P. Erik Ramos, «"Based Kinect application to promote Mixtec culture",» de *The 2013 Iberoamerican Conference on Electronics Engineering and Computer Science - CIIECC 2013*, San Luis Potosí, México, 24 al 26 de Abril 2013.

[20] L. R. Baena, «Fundamentos de la Interaccion Persona-Computadora,» Facultad de Informatica, Universidad Pontificia de Salamanca (Campus Madrid), 2005.

[21] P. A. Booth, An Introduction to Human-Computer Interaction, New York: Psychology Press, 1989.

[22] C. Faulkner, «The Essence of Human-computer Interaction,» ISBN: 0-13-751975-3, Prentice Hall, 1998.

[23] J. Preece, «Human Computer Interaction,» Addison Wesley, 1994.

[24] T. J. R. Flor E. Narciso, «La interaccion Humano-Computadora (MODIHC),» de *XXVII Conferencia Latinoamericana de Informática CLEI2001* , 2001.

[25] A. Mathew, «Human-Computer Interaction (HCI): An overview,» de *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference* , Shanghai, 10-12 June 2011.

[26] B. SHNEIDERMAN y C. PLAISANT, Diseño de Interfaces de usuario (4a edicion), MAdrid: Pearson educacion, 2005.

[27] I. 9126-1:2001, «International Standard (2001). 1. Software engineering-Product quality-Part 1: Quality model,» 2005.

[29] H. Belson, «Usability,» 2012.

[30] A. P. P. y. M. C. L. Gallo, «Controller-free exploration of medical image data: experiencing the Kinect,» de *24th IEEE International Symposium on computer-Based Medical Syste, (CBMS)*, 2011.

[31] S. Mann, Inteligent image processing, USA, Noviembre 2001.

[33] R. A. Bolt, «Put-that-there: Voice and gesture at the graphics interface,» de *In proceedings of the 7th annual conference on Computer Graphics and interactive techniques, SIGGRAPH*, Washington, USA, 1980.

[34] P. M. a. P. Maes, «Sixthsense: a wearable gestural interface,» de *In Proceedings of the ACM SIGGRAPH* , Japon, 2009.

[36] P. V. R. C. V. F. Peredo V. Iván, «Manipulación de un robot Lego Mindstorms NXT a través de una NUI avanzada con reconocimiento de voz y gestos.,» Revista Electronica de Divulgacion de la Investigacion, Revista Universitaria del SABES, Universidad Politécnica de Querétaro, 2014.

[40] J. A. Jarrett Webb, Beginning Kinect Programming with the Microsft Kinect SDk, United States of America: ISBN 9781430241041, 2012.

[41] t. teardown, «The kinect for Xbox 360,» de *The engineering and Technology Journal*, abril 2011.

[42] D. Catute, Programming with the Kinect for Windows Software Development Kit, United States of America: ISBN 978-0-7356-6681-8, 2012.

[43] M. Corporation, «Human Interface Guidelines,» v 1.8.

[46] P. E. H. D. G. S. Richard o. Duda, Pattern Classification Second Edition, USA: ISBN: 0471056693, 200.

[47] C. M. Bishop, Pattern Recognition and Machine Learning, NY, USA: Springer, 2006.

[48] H. L. C. P. E. C. B. P. Sandra E. Nope R. Esp., «Estudio comparativo de técnicas para el reconocimiento,» de *E.I.E.E*, Universidad del Valle, CaliColombia, 10 de Diciembre de 2008.

[49] L. B. L. M. L. Vargas, «Sistema de Identificacion de Lenguajes de señas usando Redes Neuronales Artificiales.,» de *Revista Colombiana de Fisica Vol. 42, No. 2 de 2010*, Colombia, 25 de Mayo 2010.

[50] R. P. Aguila, Una introduccion al Computo Neuronal Artificial, El Cid Editor ISBN 978-1-4135-2424-6, Septiembre 2012.

[54] A. M. J. G. A. D. G. M. '. ,. C. M. F. P. R. M. S. S. M. C. María José Abásolo, «Visión por Computador e Informática Gráfica. Realidad Virtual, Realidad Aumentada e Interfaces avanzadas,» de *XV workshop de investigadores en ciencias de la computacion* , Parana, 2013.

[55] R. S. Pressman, Ingenieria del Software Un enfoque practico Septima Edicion, Mexico: Mc Graw Hill ISBN: 978-607-15-0314-5, 2010.

[56] W. S. Humphrey, The Personal Software Process (PSP), Carnegie Mellon, 2000.

[57] R. C. T. A. C. J. M. W. N. Marsha Pomeroy-Huff, The personal Software Process (PSP) Body of Knoledge, Version 2.0, USA: Carnegie Mellon, Agust 2009.

[61] S. SMITH, ALGEBRA, TRIGONOMETRIA Y GEOMETRIA ANALITICA, Mexico: Prentice Hall, 1998.

[62] J. NIELSEN, «Why You Only Need to Test with 5 Users,» 19 March 2000.

[69] A. B. M. P. M. d. C. S. T. H. S. C. y. J. M. C. L. María E. Alva Obeso, «Definiendo una estructura de evaluacion para medir la usabilidad,» de *Conferencia IADIS Ibero-Americana*, España, 2007.

[71] K. S. a. J. Sutherland, The Scrum Guide, 2013.

Sitios de Internet

[5] P. M. M. C. P. L. J. P. G. I. T. F. Federico Borja Lagoa Caridad, «INTERFACES GRÁFICAS DE USUARIO,» [En línea]. Disponible en: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Interfaces/enlightment/definiciones_3.html. [Último acceso: 28 Noviembre 2014].

[9] A. Valli, «Notes on natural interaction,» 2005. [Online]. Available: http://www.idemployee.id.tue.nl/g.w.m.rauterberg/Movies/NotesOnNaturalInteraction.pdf. [Date accessed: 12 August 2014].

[10] G. Flores, «videojuegos (:: evolucion de las consolas y controles,» 1 Noviembre 2012. [En línea]. Disponible en:: http://gabyswag223414.blogspot.mx/2012/11/evolucion-de-las-consolas-y-controles.html. [Último acceso: 12 Agosto 2014].

[11] I. ccipital, «OpenNI 2 Downloads and Documentation | The Structure Sensor,» 29 October 2014. [Online]. Available: http://structure.io/openni. [Date accessed: 23 March 2014].

[12] M. D. Network, «Kinect for Windows Programming Guide,» 2014. [Online]. Available: http://msdn.microsoft.com/en-us/library/hh855348.aspx. [Date accessed: 13 August 2014].

[14] Concaculta, «Cultura en México-museos-Sistema de Información Cultural--CONACULTA,» [En línea]. Disponible en: http://www.sic.gob.mx/index.php?table=museo&disciplina=&estado_id=20&municipio_id=0. [Último acceso: 21 Febrero 2014].

[15] Conaculta, «Infraestructura y Patrimonio, Hábitos y Prácticas Culturales.,» [En línea]. Disponible en: http://mapa.sic.gob.mx/index.php?estado_id=20&tema=museo. [Último acceso: 25 Febrero 2014].

[16] Conaculta, «Museos interactivos, museos para las vacaciones,» [En línea]. Disponible en: http://www.conaculta.gob.mx/. [Último acceso: 22 Febrero 2014].

[18] P. M. e. a. MureH, «, Museo Regional de Huajuapan,» [En línea]. Disponible en: http://www.mureh.org.mx/. [Último acceso: 30 Enero 2014].

[19] M. Corporation, «Kinect for Windows| Voice, Movement & Gesture Recognition Technology,» [Online]. Available: http://www.microsoft.com/en-us/kinectforwindows/.

[Date accessed: 30 January 2014].

[28] G. i. Eduardo Mercovich, «Taller: Sitios Web de Bibliotecas Universitarias,» Junio 2004. [En línea]. Disponible en: http://www.inspiro.com.ar/articulos/bibliotecas.html. [Último acceso: 21 Septiembre 2014].

[32] S. Mann, «Steve Mann's Blog,» 2012. [Online]. Available: http://eyetap.blogspot.ca/2012/07/physical-assault-by-mcdonalds-for.html. [Date accessed: 21 October 2014].

[35] P. Santana, «Interfaces Naturales de Usuario - La experiencia de la Universidad de Colima | SG,» Abril 2014. [En línea]. Disponible en: http://sg.com.mx/revista/43/interfaces-naturales-usuario-la-experiencia-la-universidad-colima#.VGJl7vmG-b9. [Último acceso: 23 Junio 2014].

[37] Microsoft, «Todo Xbox 360,» 2014. [En línea]. Disponible en: http://www.xbox.com/es-ES/xbox-360/why-xbox-360. [Último acceso: 21 Junio 2014].

[38] Microsoft, «Microsoft Mexico | Dispositivos y servicios,» 2014. [En línea]. Disponible en: http://www.microsoft.com/es-mx/default.aspx. [Último acceso: 20 Junio 2015].

[39] E. Press, «Yahoo! Noticias España,» 25 Noviembre 2013. [En línea]. Disponible en: https://es.noticias.yahoo.com/apple-adquiere-primesense-desarrolladora-sensor-kinect-xbox-092257697.html. [Último acceso: 30 Octubre 2014].

[44] M. Corporation, «Kinect usar las camaras del sensor - MSDN España - Site Home – MSDN Blogs,» 2011. [En línea]. Disponible en: http://blogs.msdn.com/b/esmsdn/archive/2011/07/20/reto-kinect-usar-las-c-225-maras-del-sensor.aspx. [Último acceso: 25 Junio 2014].

[45] U.-. F. d. i. B. Informatica, «Reconocimiento de Patrones,» [En línea]. Disponible en: http://redyseguridad.fi-p.unam.mx/proyectos/biometria/basesteoricas/reconocimiento.html. [Último acceso: 13 Agosto 2014].

[51] M. Murcia, «Semiotica del Gesto,» Prezi, 6 Noviembre 2012. [En línea]. Disponible en: https://prezi.com/axj7wqth2dea/semiotica-del-gesto/. [Último acceso: 21 Octubre 2014].

[52] E. Latin, «Etimologia de Gesto,» 2014. [En línea]. Disponible en: http://etimologias.dechile.net/?gesto. [Último acceso: 12 Mayo 2014].

[53] M. Graham, «Clasificacion de los gestos o tipos de movimientos,» Doris Humphrey, [En línea]. Disponible en: http://agrega.juntadeandalucia.es/repositorio/05042013/1a/es-an_2013040513_9141406/clasificacin_de_gestos_o_tipos_de_movimientos.html. [Último acceso: 28 Marzo 2014].

[58] Samsung, «Samsung SMART TV,» 2013. [En línea]. Disponible en: http://www.samsung.com/co/smarttv/book/guide_book_3p_si/main.html. [Último acceso: 12 Febrero 2014].

[59] M. Corporation, «Reto SDK Kinect: Reconocer gestos con Skeletal tracking – MSDN España - Site Home - MSDN Blogs,» 22 Agosto 2011. [En línea]. Disponible en: http://blogs.msdn.com/b/esmsdn/archive/2011/08/22/reto-sdk-kinect-reconocer-gestos-con-skeletal-tracking.aspx. [Último acceso: 9 Enero 2014].

[60] (. ClearModel, «PSP developers - Certifications - Clearmodel,» 2013. [Online]. Available: http://seicertification.clearmodel.com/certified-professionals/psp-developers/. [Date accessed: 17 May 2014].

[63] Microsoft, «Kinect para windows | Soporte tecnico de Kinect | Configuracion,» [En línea]. Disponible en: http://support.xbox.com/es-MX/xbox-on-other-devices/kinect-for-windows/kinect-for-windows-setup. [Último acceso: 2 Abril 2014].

[64] Microsoft, «DirectX Software Development Kit,» [Online]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=6812. [Date accessed: 1 March 2014].

[65] Microsoft, «Visual Studio Professional con MSDN,» [Online]. Available: http://www.visualstudio.com/products/visual-studio-professional-with-msdn-vs. [Date accessed: 9 January 2014].

[66] K. f. W. Team, «Kinect for windows BLOG,» 17 Septiembre 2013. [En línea]. Disponible en: http://blogs.msdn.com/b/kinectforwindows/archive/2013/09/16/updated-sdk-with-html5-kinect-fusion-improvements-and-more.aspx. [Último acceso: 24 Junio 2014].

[67] M. Corporation, «Kinect for Windows SDK 2.0,» 21 October 2014. [Online].

Available: http://www.microsoft.com/en-us/download/details.aspx?id=44561.

[Date accessed: 2 November 2014].

[68] Microsoft, «Windows 8 Visual Basic XNA sample in VB.NET for Visual Studio 2008,»

Microsoft, 13 Julio 2011. [En línea]. Disponible en:
https://code.msdn.microsoft.com/windowsapps/visual-basic-xna-29cd4963.

[Último acceso: 12 Noviembre 2014].

[70] Xbox666, «XBOX 666,» Octubre 2011. [En línea]. Disponible en:

http://xbox666.com/2011/10/. [Último acceso: 13 Agosto 2014].

Anexo A
Manual de Usuario

Implementación del Software

a) Requerimientos de hardware:
   - Computadora
     - 4gb de RAM
     - 50 GB HDD
     - Puerto USB 2.0 Dedicado
   - Sensor *Kinect* 1.0
   - Cable USB de alimentación

b) Requerimientos de Software:
   - Direct SDK 9.0
   - *Kinect* SDK 1.8
   - Microsoft XNA Game Studio 4.0

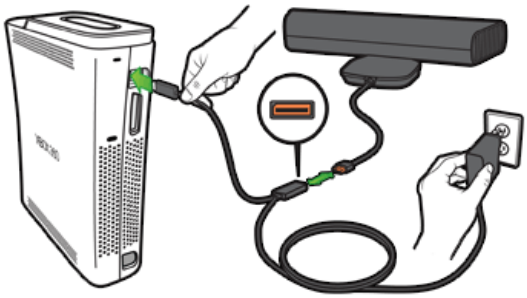


Figura 1 Conexión del Kinect a PC

Acceso al Sistema MUDI

a) Ejecute la aplicación MUDI[14].
b) Conecte el sensor *Kinect* a la PC
c) Colóquese a una distancia de tal forma que pueda verse completamente en la pantalla (2 a 3 m aprox. del sensor *Kinect*).

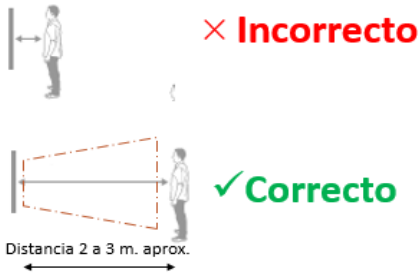


Figura 2 Iniciar la aplicación MUDI

d) Enseguida usted verá una pantalla similar con la cual podrá interactuar con las piezas arqueológicas (Si no se visualiza esta pantalla, revise la sección de error de *Kinect de este manual*).

[14] El cual debe estar previamente instalado, incluido en el CD de software de esta tesis.

Figura 3 Visualización de MUDI

Acercar/Alejar Pieza arqueológica

a) Usted en todo momento verá un icono de una mano respectivamente.

Figura 4 Acercar/Alejar MUDI

b) Debe mantener sus manos alineadas a la altura de la cintura, enseguida aparece una línea punteada verde.
c) Para poder Acercar separe sus manos, hasta lograr el tamaño adecuado.

Figura 5 Acercar pieza arqueológica

d) Para poder alejar junte sus manos, hasta conseguir el tamaño deseado.

i

Figura 6 Alejar pieza arqueológica

Deslizar pieza arqueológica

a) En la parte superior usted puede ver un menú de piezas disponibles, de las cuales usted puede elegir la de su preferencia.

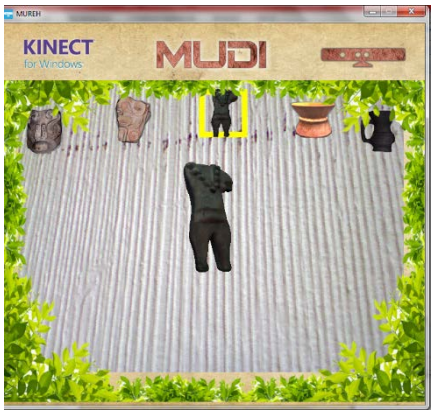


Figura 7 Menú de piezas arqueológicas

b) Usted debe mover su mano derecha en línea recta:
   - Hacia la izquierda para elegir la pieza a la izquierda.
   - Hacia la derecha para elegir pieza a la derecha.

a) Deslizar derecha-izquierda b) Deslizar izquierda-derecha

Figura 8 Deslizar pieza arqueológica

Rotar pieza arqueológica

a) El gesto de rotación inicia cuando usted tiene alineadas las manos, y estas cambian de color a azul.

Figura 9 Manos para empezar la rotación

b) Enseguida usted debe mover las manos, simulando mover un volante, ya sea de izquierda a derecha o viceversa según la rotación deseada.

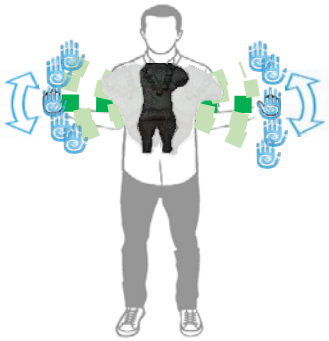


Figura 10 Gesto de rotación.

c) Las manos se mantienen azules hasta que el gesto haya terminado.

Descripción de la pieza arqueológica

a) Cada vez que usted elige una pieza arqueológica distinta usted escuchara un audio en el cual se menciona la descripción de la pieza en cuestión.

Error al inicializar *Kinect*

a) Si usted al inicializar la aplicación ve una pantalla como la siguiente:

Figura 11 Error al conectar *Kinect*

b) Entonces deberá leer el error y verificar su conexión del *Kinect* con la PC o la corriente eléctrica.

[i] NOTA: En las Figuras 3 a 6 las imágenes que se muestran de los iconos de las manos se encuentran invertidos.