UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

CENTRO DE INVESTIGACIÓN EN TECNOLOGÍAS DE INFORMACIÓN Y SISTEMAS

Manipulación de una mano virtual, utilizando el guante P5

T E S I S

PARA OBTENER EL TITULO DE LICENCIADO EN SISTEMAS COMPUTACIONALES

PRESENTAN:
Martínez González Victor Hugo
Pérez Barrera Victor Hugo

ASESOR:
M. En C. Arturo Curiel Anaya
COASESOR:
Dr. Omar Arturo Domínguez Ramírez

Pachuca de Soto, Hgo., Febrero 2006

Índice General

Índice de figuras

Capítulo 2. Estado del arte.

Capítulo 3. Tecnologías de Realidad Virtual.

Capítulo 4. Marco tecnológico.

Capítulo 5. Metodología y desarrollo del sistema.

Índice de tablas

Capítulo 4. Marco tecnológico.

Capítulo 5. Metodología y desarrollo del sistema.

Capítulo 1

Introducción

1.1 Introducción

Hoy en día, los gráficos por computadora son utilizados en varios aspectos de la vida cotidiana. En los últimos años, los equipos de cómputo de alta velocidad, mayor capacidad de memoria, etc., han disminuido su costo de forma gradual, como consecuencia muchas personas tienen acceso a este tipo de tecnologías que les permite adentrarse al mundo de los gráficos por computadora.

De igual manera los gráficos por computadora han evolucionado rápidamente, hasta llegar a lo que en la actualidad se conoce como realidad virtual, mediante la cual se representan ambientes virtuales tridimensionales con un alto grado de realismo.

Esta tecnología tiene muchas aplicaciones en diferentes campos de investigación, entre los cuales destacan:

- Educación: con el desarrollo de sistemas virtuales que permiten a los alumnos interactuar con éstos para facilitar el aprendizaje de los alumnos.

- Arquitectura: con software que permite a los arquitectos diseñar un edificio para corregir detalles previos a la construcción.

- Entretenimiento: creando personajes virtuales que realizan movimientos reales, utilizados en películas, videojuegos, caricaturas, etc.

En el presente trabajo de investigación se pretende utilizar la realidad virtual como un medio de capacitación para realizar una práctica virtual.

El ser humano utiliza la mano como primer contacto con el mundo para conocerlo y desenvolverse en él. Los instrumentos y artefactos que el hombre diseña y que se encuentran en la vida diaria, tienen como base de diseño el concepto de que serán manipulados por una mano humana.

La mano humana es un elemento complejo el cual cuenta con más de 25 grados de libertad que le permiten realizar distintos movimientos. Sin embargo, debido al diseño del guante no es posible utilizar todos estos, por lo tanto el modelo virtual se ajusta al dispositivo utilizado.

Con el fin de proporcionar un alto grado de inmersión al ambiente virtual, se utilizó un dispositivo de entrada. Este dispositivo es el guante de datos P5 (hace referencia a la frase Power of Five, utilizada como eslogan en la publicidad del guante). Después del proceso de búsqueda para la elección del dispositivo, se optó por el P5 principalmente por su bajo costo, ya que en el mercado, actualmente existen dispositivos más complejos que brindan mejor funcionamiento pero su costo se encuentra sobre los 1000 USD.

El modelo virtual de la mano fue diseñado en el lenguaje de modelado de realidad virtual (VRML), para el desarrollo de la interfaz que sirvió como medio de comunicación entre el dispositivo y el modelo virtual se analizaron varias opciones de software hasta encontrar la más adecuada para conseguir los objetivos fijados para el trabajo.

1.2 Objetivo general

Desarrollar un sistema que permita la manipulación de una mano virtual mediante el uso de un dispositivo de entrada, el guante de datos P5.

1.3 Objetivos específicos

- Desarrollar un modelo matemático que represente el movimiento de cada dedo de la mano.
- Diseño y construcción de un modelo virtual, que represente a la mano derecha de un humano, haciendo uso de las técnicas del lenguaje de modelado de realidad virtual.
- Asignación de las funciones obtenidas en el modelo matemático a los nodos correspondientes en el modelo virtual.
- Analizar la mejor tecnología que permita la comunicación del guante de datos con el modelo virtual.
- Examinar los datos del guante correspondientes a los sensores de cada dedo y a la posición del guante en los ejes *X, Y* y *Z*, así como los angulos alrededor de los ejes *X, Y* y *Z*.
- Desarrollo de una interfaz que permita la comunicación entre el mundo virtual y el guante de datos P5.
- Desarrollar una interfaz que muestre la comunicación entre el mundo virtual y Java, utilizando las técnicas de EAI (External Authoring Interface).

1.4 Justificación

En el presente mundo de la computación existe una gran cantidad de ramas o variantes que cada día cobran fuerza a medida que se investiga sobre ellas; tal es el caso de la realidad virtual.

Puesto que la Universidad Autónoma del Estado de Hidalgo tiene gran trascendencia en el desarrollo de proyectos utilizando realidad virtual, surge la inquietud de crear una variante a este tipo de investigación, como es el uso de dispositivos de entrada diferentes al mouse y

teclado; con el fin de brindarle al usuario un medio de interacción más amplio, al proporcionarle mayor grado de inmersión en comparación con los sistemas que se han desarrollado en la máxima casa de estudios de Hidalgo.

Por otra parte la capacidad de retener ideas en el cerebro humano es mayor mediante el uso de gráficos y si a esto se le agrega la utilización de otros sentidos, el aprendizaje de cualquier actividad es retenido por un tiempo prolongado. Es por ello que se decide desarrollar un software con el cual se establezca una comunicación entre el guante P5 y la PC que sea capaz de utilizar la realidad virtual, obteniendo así la base que permita la realización de una futura práctica virtual que pudiera tener impacto en diversas aplicaciones, ya sea en la enseñanza o en la rehabilitación humana.

1.5 Alcances

- El principal alcance que se espera de este proyecto, es la interacción entre el ambiente virtual desarrollado y el guante de datos, para que posteriormente este sea implementado en una práctica virtual.

- Colaborar con la UAEH en el desarrollo de sistemas de realidad virtual bajo el lenguaje de modelado VRML, asignando a estos un grado de inmersión mayor al que tienen actualmente los sistemas desarrollados en esta casa de estudios, utilizando el guante P5.

Capítulo 2

Estado del arte

Se hace mención a varios trabajos que han sido desarrollados utilizando el guante P5, de igual manera se realiza una breve descripción de estos, con el fin de conocer la manera en que fue utilizado este dispositivo para llegar a los objetivos fijados en cada proyecto.

2.1 Teclado Virtual

Este sistema fue presentado como proyecto final para *"Engineering Approaches to Music Perception and Cognition" (ISE-575)*. La idea del proyecto fue crear un teclado virtual para habilitar al usuario de la computadora, que no tenía acceso a un teclado MIDI, tocar y grabar música en su computadora usando un dispositivo tridimensional de entrada. En este proyecto se combinaron la animación de la mano que fue diseñada e implementada para el proyecto *Omni-Grasp* y una librería *Improv* para programar con MIDI en C++. El dispositivo de entrada utilizado en este proyecto fue el guante P5 debido a la posición tridimensional y postura de la mano que proporciona.

El usuario se coloca el P5 y mueve su mano y sus dedos en el espacio como si estuviera tocando el teclado. El P5 proporciona la información con respecto a la ubicación y dirección tridimensional de la mano para el programa. De igual manera envía información con respecto a la posición de los dedos que es interpretada como el ángulo de las articulaciones en los dedos. [4] [73]. La figura 2.1 muestra la interfaz del teclado virtual.

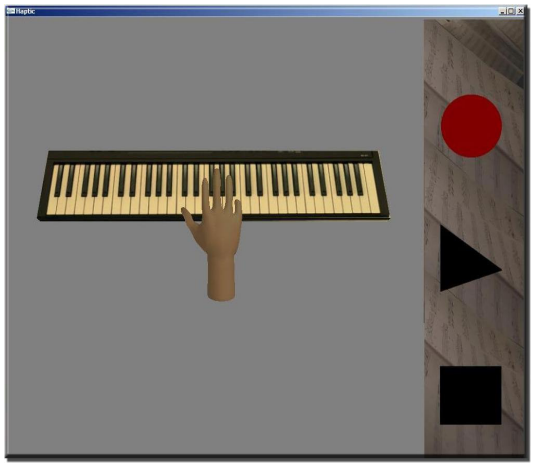


Figura 2.1 Interfaz del teclado virtual

La interfaz del teclado virtual cuenta con tres opciones a la derecha de la pantalla principal, las cuales son: *record* el cual graba cada nota tocada en el teclado; *stop* detiene la grabación y la escribe en un archivo MIDI y *play* reproduce cualquier cosa tocada en el teclado virtual. [4] [72]

2.2 Edición de movimientos con un guante de datos

En este trabajo se propone un nuevo método para editar datos capturados del movimiento de la mano utilizando un guante de datos (figura 2.3) (figura 2.4). El primer animador usaba un guante y mímica del movimiento del cuerpo humano observados en la pantalla gráfica usando la mano. Entonces, una función de mapping convierte los movimientos de la mano que son generados por todo el cuerpo. Finalmente moviendo ligeramente la mano de forma diferente, se genera un nuevo movimiento con forma diferente. Por ejemplo, después de

imitar el movimiento del caminar con movimientos alternativos del dedo índice y del dedo medio, moviendo los dedos rápidamente con pasos largos más grandes, es posible obtener un movimiento similar al que se efectúa cuando corres. Para lograr esta meta, se propone un método para trazar el movimiento de la mano a todo el cuerpo (figura 2.2). Este método no solo puede usarse para revisar el movimiento del cuerpo humano, también para controlar figuras humanas en ambientes en tiempo real tales como juegos y sistemas de realidad virtual. [16].

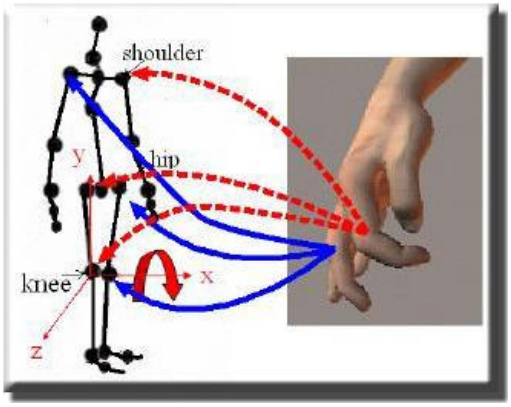


Figura 2.2 Correlación de los dedos y las articulaciones del cuerpo utilizadas para el trazado

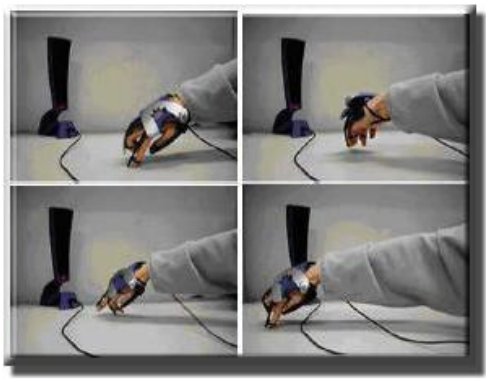


Figura 2.3 Las gesticulaciones de la mano generan los movimientos de zigzag al caminar

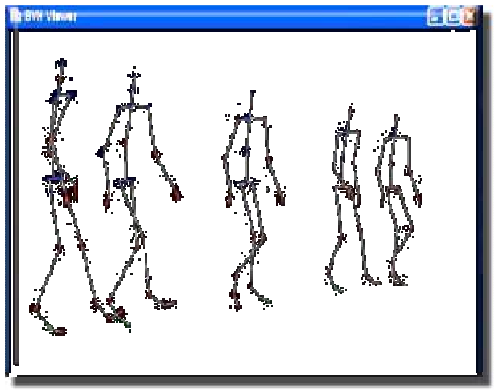


Figura 2.4 La trayectoria original del movimiento al caminar generado por el sistema

2.3 Técnicas de interacción gestual para el cruce dimensional en ambientes inmersivos híbridos

Este trabajo de investigación presenta un conjunto de técnicas de interacción de cruce dimensional para una interfaz de usuario híbrida que integra visualización 2D y 3D así como dispositivos de interacción. En el sistema se presenta un sistema experimental que combina múltiples dispositivos que despliegan ambientes en 3D con una interfaz de usuario en 2D proyectada en una superficie sensible al tacto. Se le conoce como una interfaz de usuario hibrida a una combinación heterogénea de imágenes y tecnologías, en este caso con todas las herramientas utilizadas se forma un ambiente inmersivo hibrido. Esta investigación se enfoca principalmente a las interfaces de usuario en 2D y 3D. Se ha diseñado un cruce dimensional con interacción de gestos que requiere de usar las dos modalidades 2D y 3D. Este contexto de cruce dimensional establece e integra una forma de interacción gestual entre los ambientes 2D y 3D.

Hay tres fragmentos de información necesarios para completar una transición precisa y exitosa de cruce dimensional: el objeto, el tipo de transformación y el destino de la ubicación. Por ejemplo para transformar un objeto de una representación 2D a una representación 3D se necesita especificar el objeto en 2D que será sometido a la transformación, el efecto deseado para una transformación de 2D a 3D y finalmente, donde será ubicado el objeto en 3D una vez que se transforma. El cruce dimensional gestual presentado en este trabajo ha sido desarrollado como parte de *VITA (Visual Interaction Tool for Archeology),* un sistema relativo de visualización en un sitio de excavación arqueológica. Se emplea una serie de dispositivos de inmersión tales como lentes (Sony LDI-D100B), micrófonos, guates (P5) y una mesa sensible al tacto (Diamond touch table).

El cruce dimensional se muestra en la figura 2.5: (a) Selección del objeto en 2D y simula un gesto como si lo estuviera agarrando; (b) El objeto en 2D desaparece, mientras aparece el objeto en 3D de la mesa; (c) Agarra el objeto en 3D; (d) Hace un gesto para presionar la mesa a través del objeto 3D; (e) El objeto 3D desaparece y aparece de nuevo el objeto 2D; (f) El objeto en 2D es proyectado en la mesa. [7] [38]

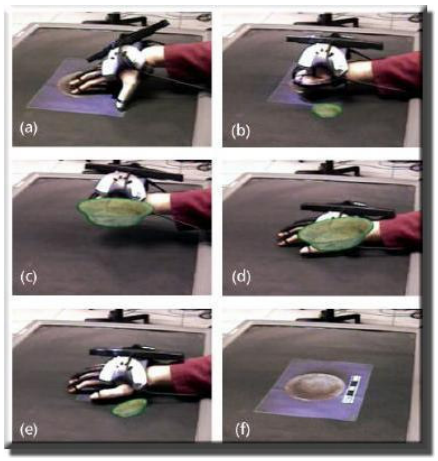


Figura 2.5 Secuencia de cruce dimensional

2.4 Esquema de apoyo de consistencia distribuida para los ambientes interactivos de Realidad Aumentada

Los avances en las redes de computadoras y el render en los sistemas facilita la creación de ambientes colaborativos distribuidos en los cuales la distribución de la información a una ubicación remota permite la comunicación eficiente. El desafío particular son los ambientes de Realidad Aumentada (AR) interactiva distribuida que permiten compartir el conocimiento a través de información en 3D.

En un ambiente AR interactivo distribuido el estado compartido dinámico representa la información cambiante que varias maquinas deben mantener acerca de los componentes virtuales compartidos. Uno de los desafíos en estos ambientes es mantener una vista consistente del estado compartido dinámico en presencia de la inevitable latencia y fluctuación de la red. Una vista consistente de una escena compartida incrementará significativamente la sensación de presencia entre los participantes y la facilidad de su interacción.

Además de las consideraciones de los problemas de latencia y en la perspectiva de las tendencias actuales en aplicaciones distribuidas interactivas, se propone un sistema de arquitectura híbrido para ambientes distribuidos basados en sensores que tienen el potencial de mejorar el comportamiento y la escalabilidad de los sistemas en tiempo real.

La figura 2.6 es un ejemplo de un escenario utilizado en este proyecto. Un cirujano ubicado en su oficina está analizando el modelo en 3D de la mandíbula de uno de sus pacientes, a este le gustaría discutir con uno de sus colegas el procedimiento quirúrgico que seguirá brevemente, pero su oficina se encuentra en otro edificio. Como parte de la interacción tienen que analizar el modelo en 3D de la mandíbula de su paciente. Para esto utilizan una plataforma de visualización distribuida implementada en la red de área local del hospital. Para una visualización estereoscópica cada oficina esta equipada con un HMD (Head mounted display) y un guante. En este escenario la plataforma de visualización distribuida permite a un participante modificar la posición y orientación del modelo en 3D de una Interfaz Gráfica de Usuario (GUI), utilizando el mouse o a través de un guante. [11][64]

Figura 2.6 Interfaz Gráfica de Usuario (GUI) basada en la interacción

2.5 SenseShapes: uso de geometría estadística para la selección de objetos en un sistema de Realidad Aumentada

Se introduce un conjunto de herramientas geométricas estadísticas para identificar los objetos siendo manipulados a través del lenguaje y los gestos en un sistema multimodal de realidad aumentada. Los SenseShapes están en regiones de interés que pueden ser vinculadas a partes del cuerpo del usuario para proveer información acerca de la interacción del usuario con los objetos. Para ayudar en la selección del objeto, se generó un conjunto de datos estadísticos y dinámicamente se seleccionan los datos que son considerados la base de la situación actual.

Los dispositivos de entrada utilizados son, el guante P5 modificado para sensar los ademanes de la mano, un micrófono y dos rastreadores de 6 GDL (grados de libertad) InterSense IS900 para monitorear la posición y orientación de la cabeza y la mano. El IBM Via Voice 10 es usado para el reconocimiento de voz, y el Sony LDI-D100B presenta la realidad aumentada. En la figura 2.7 se muestran los dispositivos utilizados en este sistema de realidad aumentada.

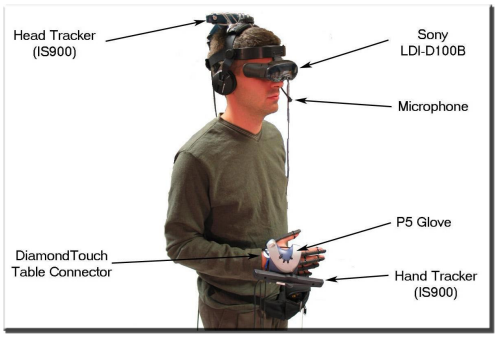


Figura 2.7 Dispositivos de entrada utilizados

El sistema recibe eventos de los ademanes realizados (generalmente “señalar”, “agarrar” y “pulgar hacia arriba”) y comandos de voz, con la ayuda de SenseShapes, se integran acciones validas (figura 2.8). Los SenseShapes también son usados en sistemas de Realidad Aumentada. Los SenseShapes son primitivas (esferas, cubos, cilindros y conos) vinculadas al usuario. [24][37]

Figura 2.8 Un usuario AR interactuando a través de gestos, voz y SenseShapes

2.6 Multiplayer game

Este es un juego multi jugador el cual utiliza el guante P5 para controlar una mano. El motor del juego fue desarrollado con la ayuda de un exporter de Maya, de igual manera en éste se desarrollaron algunos modelados, las texturas, la animaciónes de los huesos y algunos aspectos de colisión y control. Las 2 casas utilizadas en el mundo virtual fueron desarrolladas por Melissa St-Aubin. El juego funciona con el guante o sin él. [55]. La figura 2.9 representa una escena del juego en donde la mano es controlada por el guante de datos P5.

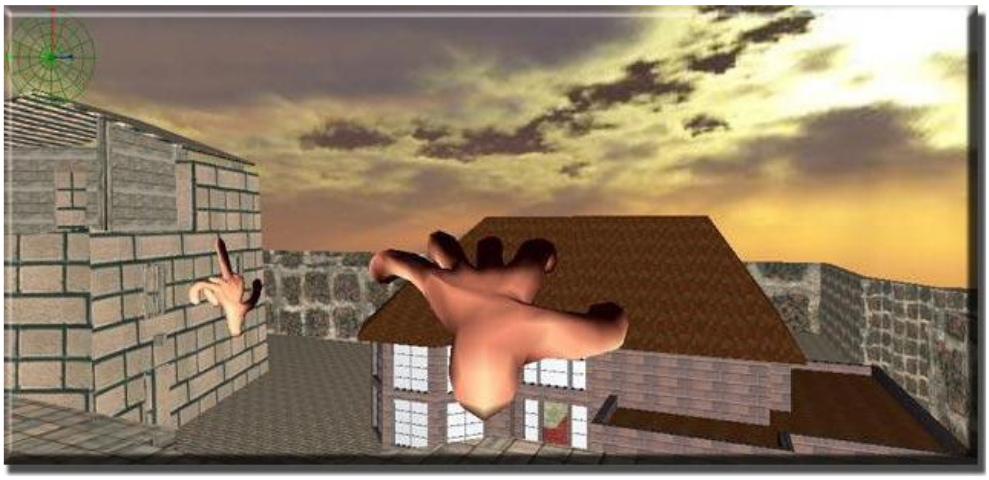


Figura 2.9 Escena del juego, la mano es controlada por el P5 guante

2.7 Robot Glove

El proyecto denominado Robot glove fue desarrollado por Eric Lundquist, dicho trabajo es un ejemplo de aplicación de tele presencia conveniente para propósitos de demostración. El código utilizado para este programa esta escrito en Java. La interfaz P5 Glove es el dll P5 Absolute Mode, proporcionado por Carl Kenners. El brazo robótico Linxmotion es controlado mediante la interfaz RS-232 con la librería comm de Java, está permite teóricamente que el código sea portable a Windows y Linux. El brazo utiliza un servo controlador serial Mini SSC. Este trabajo involucra lecturas de los valores del guante y con éstas es movido el brazo robótico. El eje *yaw* controla la rotación de la base, el *pitch* controla la muñeca, el eje *Z* controla del hombro a la base y el eje *Y* controla el codo. [63]. En la figura 2.10 se representa la interacción entre el guante de datos P5 y el robot Linxmotion.

Figura 2.10 Brazo robótico Linxmotion

2.8 Arqueología virtual de Aphrodisias

El trabajo arqueología virtual de Aphrodisias es una ambiente virtual que puede ser experimentado como una instalación en una variedad de escenarios como un museo o un salón de clases. Este puede ser usado en línea como una herramienta de comunicación por arqueólogos y estudiantes. El ambiente virtual incluye reconstrucciones de Ágora, antigua ciudad de Aphrodisias usando elementos relacionados con la arquitectura, esculturas, colores y texturas originales. En el ambiente se introdujeron nuevos sonidos basados en los dispositivos de entrada que permitirán a los usuarios interactuar con la pantalla con solo aplaudir o chasquear los dedos.

Muchos estudiantes del ITP (Interactive Telecommunications Program) han sido motivados en el diseño del proyecto para incluir la programación de comportamientos reutilizable. El proyecto fue presentado dentro de una instalación portátil o una cabina de bajo costo construida durante la exhibición ITP Summer show.

Aphrodisias está diseñado como una instalación inmersiva que puede ser usada por grupos de alumnos y maestros para estudiar arqueología, historia y urbanismo (figura 2.11). La instalación multi-pantalla es una cabina portátil apropiada para cabinas pequeñas. El sistema de navegación esta basado en distintas cámaras interactivas, los espectadores pueden activar cámaras moviendo la varita mágica enfrente de la pantalla. Por default el sistema de navegación va desde la cámara de cocktail hasta una cámara determinada. Los visitantes también pueden usar un guante P5 u otro dispositivo infrarrojo para navegar a través del escenario virtual. [34]

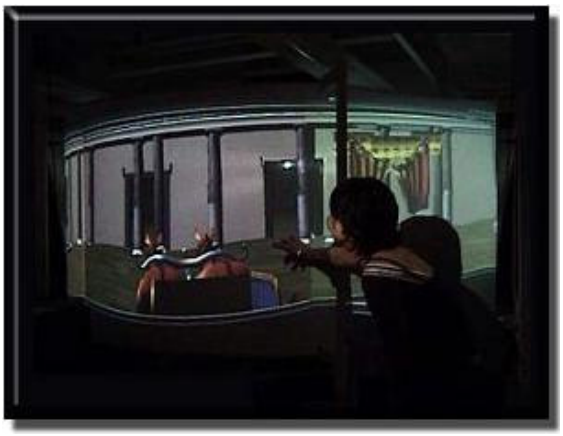


Figura 2.11 Un estudiante dirige a unos caballos tirando de una carreta dentro del Ágora

2.9 P5MIDI

Este proyecto fue desarrollado por Nicolás Fournel; P5MIDI transforma el guante en un controlador MIDI, esto permite controlar el sintetizador y otros programas MIDI con el movimiento de la mano. El P5midi convierte la información que viene de los sensores del guante P5 en mensajes MIDI. El P5 tiene cuatro botones, de los cuales se usan tres para enviar mensaje a un programa de cambio MIDI, el último botón se usa como switch para activar y desactivar el guante.

P5MIDI permite seleccionar el puerto MIDI por el cual serán enviados los mensajes, a este puerto se puede conectar directamente un sintetizador MIDI. De igual manera se puede dirigir el puerto MIDI seleccionado en P5MIDI hacia el puerto MIDI de cualquier otro programa. En este caso se utilizan los dedos para ajustar el filtro, cuando se doblan se modifica la frecuencia de los 5 formatos, una representación gráfica ayuda a visualizar como el programa percibe el doblado de los dedos. Una representación en 3D ayuda a visualizar como el programa percibe la posición de la mano (figura 2.12). [59]

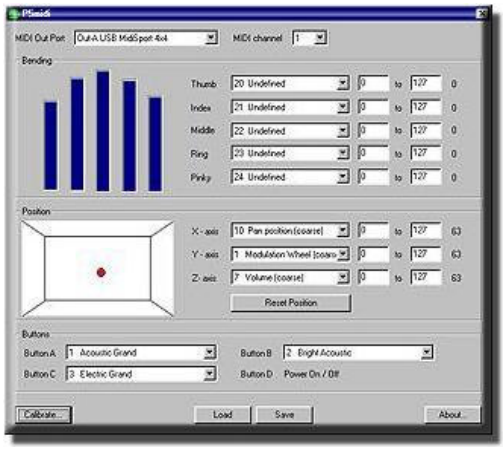


Figura 2.12 Interfaz del programa P5MIDI

2.10 P5 DirectInput Emulator

El programa DirectInput Emulator desarrollado por Carl Kenner brinda la oportunidad de utilizar el guante como un Joystick, debido a que configura los elementos del guante (ejes, botones, dedos) como si fueran los elementos que constituyen un Joystick (palanca, botones), de esta manera se puede utilizar el P5 en cualquier juego donde normalmente se usa un Joystick. Este programa fue desarrollado con el software PIE (Programmable Input Emulator). [57]

Para poder utilizar este programa es necesario configurar una serie de opciones en el programa PPjoy (figura 2.13), debido a que con éste se crea un joystick virtual y se le asignan las características con las que se desea que cuente el Joystick (palanca, botones), de otra manera el programa no cumplirá la función para la que esta diseñado.

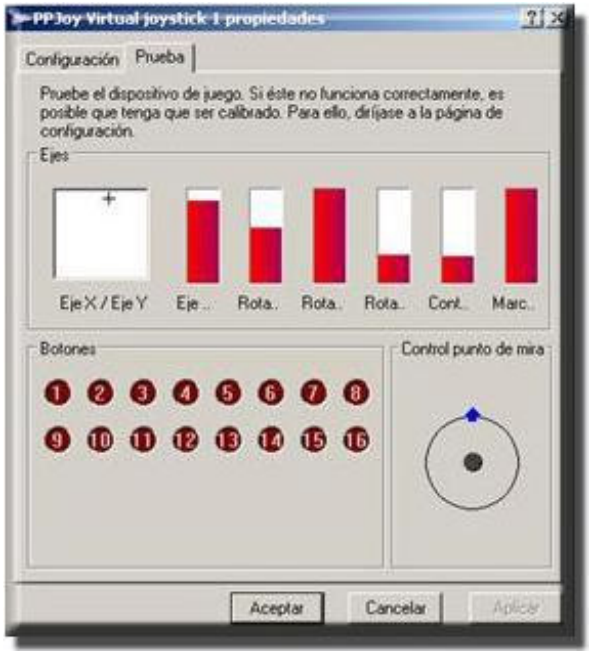


Figura 2.13 Propiedades del Joystick Virtual en PPJoy

2.11 Hand

Este proyecto es un prototipo para un juego, consiste en una mano virtual que puede ser controlada en tiempo real por el guante P5. La mano se encuentra formada por un conjunto de vértices y esqueletos (figura 2.14), cada esqueleto puede controlar múltiples vértices para lograr una simulación más realista. Este trabajo fue desarrollado en OpenGL. Este programa solo funciona si se cuenta con el guante. [55]

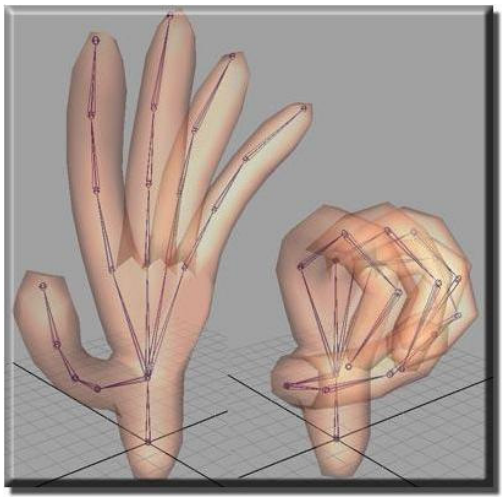


Figura 2.14 El esqueleto y la animación utilizada para formar la mano

2.12 Escultor virtual

El proyecto *Escultor virtual* fue desarrollado por Alexander Ivanov en Visual C++, con el propósito de hacer un software que permitiera realizar el modelado de superficies en 3D con la posibilidad de manipulación táctil con el guante P5 en el espacio virtual. El escultor utiliza un programa llamado Shape Editor el cual fue desarrollado para hacer e importar superficies. La aproximación de superficies se basa en dibujar por medio de las funciones B-spline. Para crear una superficie es necesario seleccionar cualquier superficie estándar, usando el guante P5, se mueve el cursor hacia el lugar al que se desea, ocupando el dedo índice y se jala en la dirección deseada. [75]. En la figura 2.15 se muestra una pantalla del proyecto escultor virtual, en la que se representa un gráfico manipulado con el guante de datos P5.

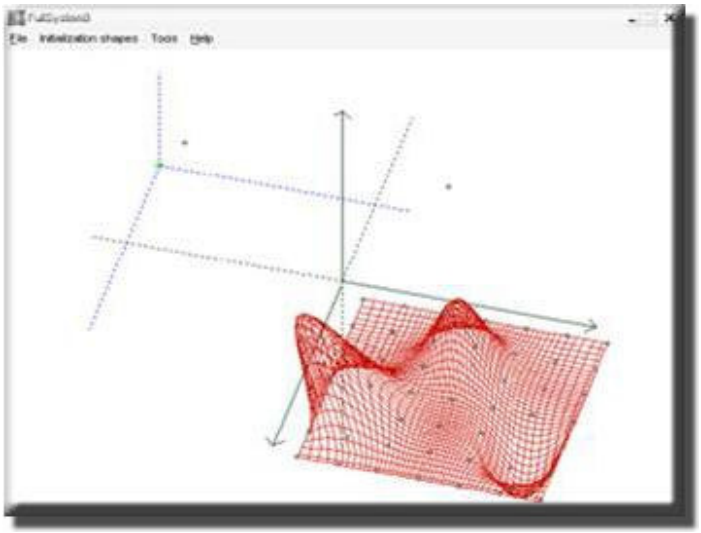


Figura 2.15 Escultor virtual

2.13 SDK P5 glove

Software Development Kit (SDK), es un paquete de desarrollo proporcionado por el fabricante, se apoya únicamente bajo las plataformas Windows y Linux. En el entorno de Windows esta herramienta cuenta con varios ejemplos en donde se hace uso del guante de datos P5 como dispositivo de entrada, los ejemplos están desarrollados bajo lenguaje de programación visual C++ 6.0, haciendo uso de las APIs de DirectX, DirectGraphics y DirectInput.

El proyecto mas interesante es el denominado Hand (figura 2.16), que consta de una mano virtual la cual puede ser manipulada mediante el guante de datos P5, en este existe manipulación entre los dedos del guante y el mundo, así, como movimiento en los ejes "X", "Y" y "Z".

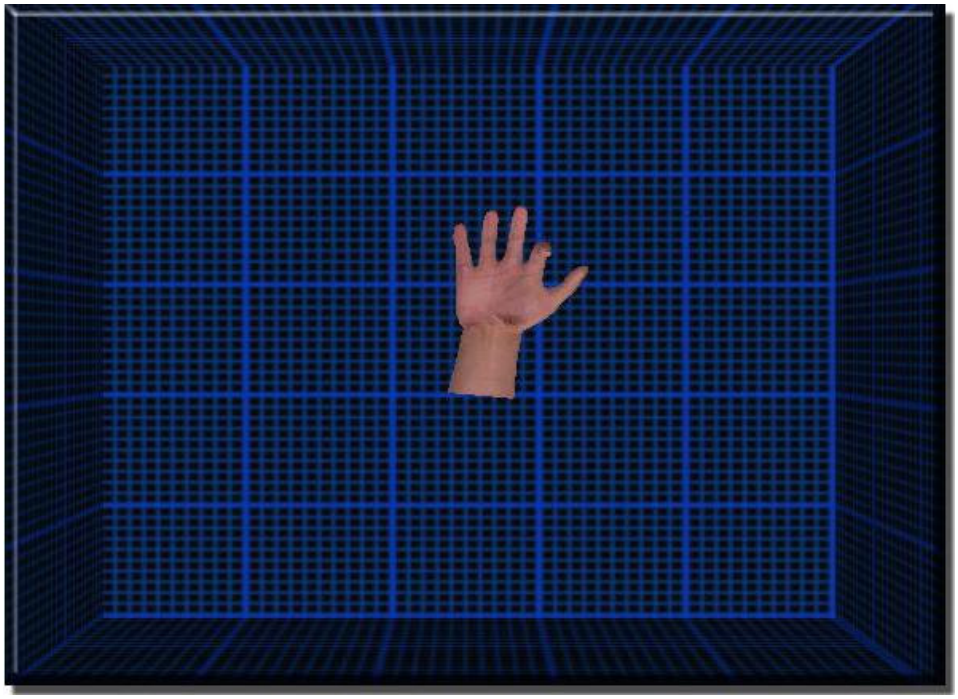


Figura 2.16 Hand (Demo SDK P5)

Capítulo 3

Tecnologías de Realidad Virtual

Se proporciona una descripción general de la realidad virtual en todos sus aspectos, de forma general, es decir se analizarán los puntos más importantes de ésta como lo son:

- Antecedentes.
- Definiciones.
- Características.
- Modelo genérico de un ambiente virtual.
- Tipos de sistemas de realidad virtual.

De igual manera se dan a conocer algunos dispositivos utilizados para obtener un grado de inmersión más alto en los ambientes virtuales, así como las tecnologías que emplean algunos fabricantes para la elaboración de dichos dispositivos.

3.1 Realidad Virtual

3.1.1 Antecedentes

La tecnología de la realidad virtual nació en USA, al principio se enfocó principalmente en el campo militar; se invirtieron millones en el desarrollo de programas de realidad virtual como simuladores de vuelo.

A continuación se mencionan algunas de las investigaciones más sobresalientes en cuanto a la realidad virtual:

- 1960 – 1962, Morton Heiling creo un simulador con sensores, denominado "Sensorama".

- 1965, Ivan Sutherland diseña "The ultimate display", se trataba de un casco con pantalla que permitía que un piloto viera simultáneamente el paisaje real e imágenes gráficas tridimensionales.

- En 1977, fueron creados los primeros guantes utilizables como periféricos de entrada de datos por D. Sandin y R. Sayre en Chicago.

- En 1981, T. Zimerman, investigaba la forma de simular un instrumento de música virtual por medio de movimientos simples de la mano, de esta investigación surge el Dataglove (guante de lycra cuyos cinco dedos están recubiertos con fibras ópticas que hacen la función de sensores de las flexiones), patentado en 1982.

- En 1984, en el centro NASA-Ames, en California, M. McGreevy iniciaba el proyecto VirtualWorkstation para la preparación de misiones en el espacio.

- En 1986, W. Tobinett escribió el primer programa de modelado del Dataglove en un entorno tridimensional.

- En 1988 el laboratorio de realidad virtual de la NASA creo el primer modelo virtual de una edificación, precisamente la del propio laboratorio.

- El año de 1989 señalo los inicios industriales de la realidad virtual con la aparición de los primeros Eyephones de la empresa de California Visual Programming Language Research (VPL). En ese mismo año Rikk Carey y Paul Strauss de Silicon Graphics Inc., iniciaron un nuevo proyecto con el fin de diseñar y construir una infraestructura para aplicaciones interactivas con gráficos tridimensionales.

- En 1994, Mark Pesce y Brian Dehlendorf crearon VRML mailing list o lista de discusión VRML donde se hizo un llamado abierto a todo el público para dar propuestas para una especificación formal de 3D en la red.

3.1.2 Conceptos de realidad virtual

La realidad virtual es la representación de la cosas a través de medios electrónicos (figura 3.1), que da la sensación de estar en una situación real en la que el usuario puede interactuar con lo que le rodea [76].

Figura 3.1 Representación virtual de un templo

Es una combinación de diversas tecnologías e interfaces que permiten a un usuario interactuar de forma intuitiva con un entorno inmersivo y dinámico generado por computadora. La realidad virtual trata de utilizar la mayor cantidad de sentidos para crear la sensación de inmersión [8].

Es simulación por computadora, dinámica y tridimensional, con alto contenido grafico, acústico y táctil, orientada a la visualización de situaciones y variables complejas, durante la cual el usuario ingresa, a través del uso sofisticado de dispositivos de entrada, a "mundos" que aparentan ser reales, resultando inmerso en ambientes altamente participativos, de origen artificial [8].

Es un sistema de computación usado para crear un mundo artificial en donde el usuario tiene la impresión de estar en ese mundo y la habilidad de navegar y manipular objetos en él [77].

Es una tecnología que permite participar en una realidad interactiva y simulada por computadora, en la cual se reemplaza o aumenta la información sensorial que se recibe. La visualización de la escena simulada en tiempo real se adapta a nuevas situaciones, de acuerdo con los movimientos del participante o con otros estímulos introducidos [61].

3.1.3 Características

Las características básicas con las que debe contar un sistema de realidad virtual son las siguientes:

- *Interacción:* Permite al usuario manipular el curso de la acción dentro de una aplicación de realidad virtual, permitiendo que el sistema responda a los estímulos de la persona que la utiliza. Otro aspecto de la interacción es la semántica del ambiente, es decir, las reglas de cómo los componentes del mundo virtual interactúan con el usuario para intercambiar información.

- *Inmersión:* Significa bloquear al usuario de toda distracción y enfocarse sólo en la información u operación sobre la cual se trabaja. [26] [62] La idea de inmersión tiene que ver con el uso de determinadas tecnologías (como por ejemplo la óptica, la estereoscópica, la electrónica, etc.) con el objetivo de configurar modelos que puedan crear la ilusión o sensación de estar dentro de un escenario (construido por computadora). La noción de navegación consiste en crear un determinado modelo por computadora, un espacio, ámbito, mundo o escenario y ofrecer la capacidad a la persona de poder desplazarse, moverse, tocar, deambular como si se encontrara dentro de él, viviéndolo y construyéndolo.

- *Tridimensionalidad:* Tiene que ver directamente con la estimulación de los sentidos del usuario, principalmente la visión, para dar forma al espacio virtual. Los componentes del mundo virtual se muestran al usuario en las tres dimensiones del mundo real.[26][62]

- *Tiempo real*: un sistema de tiempo real es aquel que realiza las operaciones en el momento en el que se le indican, en el caso de los ambientes virtuales, este debe responder de forma inmediata al momento en que se le envía una señal.

3.1.4 Modelo genérico de un ambiente virtual

La figura 3.2 representa el modelo genérico de un ambiente virtual, muestra la entrada y salida que será presentada al usuario final.

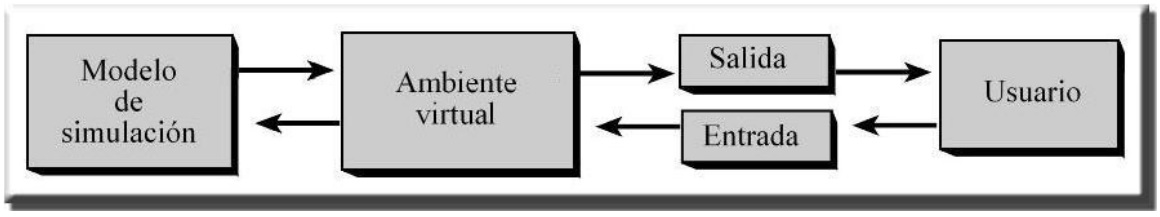


Figura 3.2 Modelo genérico de un ambiente virtual

- *El modelo de simulación:* es aquel que es representado por medio de la computadora.

- *El ambiente virtual:* es el escenario virtual con el cual el usuario tendrá interacción utilizando diferentes dispositivos.

- *Entrada – Salida*: es la forma en que el usuario puede asignar nuevos valores al ambiente (por ejemplo: coordenadas), para que estas sean representadas en el ambiente virtual.

- *El usuario:* es la persona que tendrá interacción con el ambiente virtual. [22]

3.1.5 Tipos de sistemas de realidad virtual

En la actualidad existen varios sistemas de realidad virtual, la clasificación mostrada a continuación es de acuerdo a la forma en que el usuario interactua con los sistemas, entre estos tipos de sistemas de realidad virtual se encuentran:

- *Sistemas de ventana al mundo (Window on World WoW):* A estos sistemas se les conoce de igual manera como "realidad virtual de escritorio", en estos se emplea el monitor de la computadora para mostrar el mundo de una forma visual para el usuario (figura 3.3).

Figura 3.3 Sistema de ventana al mundo (Window on World WoW)

- *Sistemas de Inmersión:* En estos sistemas, se emplean mas sentidos del usuario (vista, oído, tacto) que lo involucren de forma más real con el ambiente virtual, sumergen completamente la visión del usuario en el mundo artificial. Suelen estar equipados con cascos, dispositivos visuales, dispositivos auditivos, o dispositivos táctiles. La figura 3.4 muestra un sistema de inmersion.

Figura 3.4 Sistema inmersivo

- *Mapa de video:* Es una variación del sistema de ventana al mundo, el cual mezcla una conexión de video de la silueta del usuario con un gráfico de computadora en dos dimensiones. En el monitor se observará el cuerpo del individuo interactuando con el mundo.

- *Telepresencia:* Funciona a través de la conexión de sensores entre un operador humano y un robot u otra clase de dispositivo, con el fin de hacer una tarea específica a distancia.

- *Realidad mixta:* La realidad mixta surge como resultado de la combinación de sistemas de telepresencia y realidad virtual. En este caso las entradas generadas por la computadora son mezcladas con telepresencia y/o con la visión del usuario del mundo real. [8]

3.2 Dispositivos de Interacción en ambientes virtuales

La realidad virtual y los ambientes virtuales van más allá de las interfaces típicas en el realismo de la metáfora visual. El apuntar y dar "clic" con un ratón sobre la mesa es muy útil en muchas situaciones, pero no suficiente para ambientes con inmersión. Entonces en vez de un teclado y un ratón, se han desarrollando guantes, ratones tridimensionales, joysticks, etc. También se está estudiando la posibilidad de incorporar aromas en la interacción con el mundo virtual [1] [46].

Actualmente en el mercado están disponibles un conjunto de dispositivos de hardware para determinar la posición de la mano, Head Mounted Displays (HMD), así como sistemas de audio en 3D, sistemas de reconocimiento o síntesis de voz. Al mismo tiempo, muchos laboratorios de investigación están trabajando en el desarrollo de nuevos dispositivos tales como dispositivos force feedback, guantes táctiles, dispositivos de rastreo del ojo o mejorando dispositivos existentes como HMD o sistemas de rastreo. [5][74]

3.2.1 Dispositivos de entrada (INPUT)

El usuario puede trasmitir sus órdenes al sistema de realidad virtual, indicándole que desea desplazarse, cambiar el punto de vista o interactuar con algún objeto. Los dispositivos de salida permiten que el usuario se sienta inmerso en el mundo virtual creado. Dependiendo de la complejidad del sistema de realidad virtual estos dispositivos serán más, y más complejos. Existen además multitud de elementos diseñados para aplicaciones específicas. Los dispositivos de entrada, se clasifican principalmente en dos categorías que son:

- *Elementos de control:* Guantes, trajes de datos, joysticks, mouse, track balls etc.
- *Rastreadores de posición y movimiento*

3.2.1.1 Elementos de control

3.2.1.1.1 Joystick, mouse y track balls

Los elementos de control más sencillos son los ratones (figura 3.5) y Joysticks (figura 3.6), sin embargo estos dispositivos han sido fundamentalmente desarrollados para movimientos bidimensionales y plantean problemas para el desplazamiento tridimensional. Diseños más sofisticados como volantes, Joysticks 3D y trackballs (figura 3.7) permiten el desplazamiento tridimensional de manera más eficiente.

Figura 3.5 Mouse

Figura 3.6 Joystick

Figura 3.7 Trackball

3.2.1.1.2 Guantes

Los dispositivos para medir la posición de la mano deben sensar la flexión de los ángulos de los dedos así como la posición y orientación de la muñeca en tiempo real. Técnicas basadas en imágenes han sido usadas para medir estos dos parámetros directamente observando los movimientos del usuario. Sin embargo desde que se requiere la interpretación de movimientos con las imágenes, esta técnica tiende a ser lenta he imprecisa, aún cuando se utilizan LEDs para identificar los puntos clave en el usuario para simplificar la operación del procesado de las imágenes [5] [74]. Cuando se mueve la

mano, el guante recoge el movimiento y envía una señal eléctrica a la computadora la cual después convierte el movimiento del espacio real en el espacio virtual [33].

Actualmente el dispositivo más utilizado para medir la posición de la mano es el guante. Los guantes son dispositivos de entrada que detectan los ángulos de las articulaciones de los dedos. La medición de la flexión de los dedos es hecha con ayuda de sensores de fibra óptica (DataGlove de VPL), tecnología de tiras de aluminio (Caber Glove de Virtex) o sensores de resistencia (Power Glove de Mattel). El uso de guantes permite a los usuarios una mejor interacción que al utilizar un mouse 3D, debido a que los ademanes de la mano pueden ser reconocidos y convertidos en acciones adecuadas. Además los guantes están equipados con un rastreador el cual se sujeta a la muñeca del usuario con el fin de medir su posición y orientación. [23] [51]

Estos dispositivos utilizan básicamente dos técnicas, los exoesqueletos y los guantes de datos.

- *Los exoesqueletos*, que se aplican también para dedos individuales, piernas o para el manejo de pequeños instrumentos manuales en medicina virtual, consisten en la instalación de una estructura mecánica paralela y sobrepuesta a la mano (adherida a distancia en distintos tramos). Con rotores en cada articulación, sensores y el cableado correspondiente, conformando una compleja instalación normalmente fija. La ventaja de este sistema es su precisión, por lo cual se presta para aplicaciones delicadas. Algunos dispositivos utilizan guantes rígidos, con un mínimo de articulaciones y controles asociados, pero presentan un campo limitado de uso. [20]

- *Guantes de datos*, es un guante equipado con sensores que reporta la posición de la mano y los dedos del usuario a la computadora, para que esta los utilice de forma conveniente. Los guantes de datos son usados comúnmente en ambientes de realidad donde el usuario ve la imagen de un guante y puede manipular los movimientos de dicho ambiente utilizando el guante.

A continuación se mencionan algunos de los guantes más utilizados:

DataGlove (figura 3.8): está hecho de lycra delgada, cuenta con dos herramientas de medición. La primera es la flexión y extensión de cada dedo. Lo hace usando un juego de cables de fibra óptica que funciona a lo largo de cada dedo, con un fotosensor en un extremo y un diodo emisor de luz (LED) en el otro extremo. Cuando una persona que porta el guante dobla un dedo, la luz del LED sale a través de pequeños agujeros en la cubierta del cable. Por lo tanto, menos luz alcanza el fotosensor y genera una señal eléctrica más débil. De este modo, la computadora recibe la entrada sobre qué dedos y por cuanto tiempo se están doblando. La segunda herramienta mide la posición absoluta (ejes *X, Y* y *Z*) y la orientación (*yaw, pitch* y *roll*) de la mano.

Figura 3.8 DataGlove desarrollado por VPL

PowerGlove (figura 3.9): es una versión más económica que el DataGlove, ambos realizan las mismas funciones usando métodos completamente diferentes. Originalmente, este fue vendido como un control de juego para el Nintendo Home Entertainment System, pero debido a su bajo costo, el PowerGlove ha encontrado rápidamente camino en varias instalaciones de investigación de realidad virtual alrededor del mundo. Para medir la flexión, el PowerGlove tiene una tira cubierta de plástico con un conductor eléctrico. Esta tira es puesta a lo largo de cada dedo y cuando se flexiona, la resistencia eléctrica cambia. Para la posición u orientación absoluta, el PowerGlove usa la técnica de posicionamiento ultrasónico más simple. Los receptores recogen las señales de dos trasmisores ultrasónicos en el guante y los convierten en una posición en el espacio.

Figura 3.9 PowerGlove desarrollado por Mattel

CyberGlove (figura 3.10): guante diseñado principalmente para la manipulación de objetos 3D en la compañía de diseño de ambientes virtuales *CyberCAD*. El guante está construido por una mezcla de 80% de nylon y 20% lycra para lograr la flexibilidad, con una malla en el área de la palma y bajo los dedos que sirve como ventilación (las puntas de los dedos se encuentran abiertas para permitir fácil mecanografía en el teclado o

manipulación de objetos físicos). El CyberGlove utiliza 22 sensores para medir los ángulos de las articulaciones: 3 sensores de flexión y un sensor de abducción por dedo, sensores que miden la posición del pulgar, así como la flexión y abducción de la muñeca. El CyberGlove viene con el software VirtualHand.

Figura 3.10 CyberGlove desarrollado por Virtual Technologies Inc

Pinch Glove (figura 3.11): está basado en un prototipo desarrollado por investigadores del Instituto para la Simulación y Entrenamiento, de la Universidad del Centro de Florida. Diferente a otros guantes, el sistema del *Pinch Glove* no mide los ángulos de las articulaciones de los dedos. En cambio, los guantes son usados en ambas manos y tienen un contacto con dos o más dedos, permitiendo la definición de una variedad de ademanes *"pinch"* los cuales pueden ser correlacionados por el desarrollador de una aplicación. Teóricamente se pueden realizar cerca de 1000 ademanes. Los guantes están hechos de una tela flexible y contienen un sensor eléctrico en cada punta de los dedos. Cada guante en la parte trasera de la mano tiene montado un rastreador espacial. El punto de interacción del usuario en los ambientes virtuales esta representado por un cursor en 3D. La interfaz del sistema llamada Pinch Hand Gesture System, contiene guantes para la mano derecha e izquierda, una interfaz electrónica y el software controlador para la PC.

Figura 3.11 Pinch Glove desarrollado por Fakespace Inc

5th Glove (figura 3.12): este guante utiliza sensores con tecnología de flexores de fibra óptica. Cada dedo del guante se ajusta con un sensor que mide la flexión del dedo. En su más reciente lanzamiento, el *5th Glove* también incluye un sensor de inclinación de 2 ejes que mide la orientación pitch y roll ±60º de la mano del usuario y puede estar montado en dirección horizontal o vertical. Estos nuevos sensores permiten al guante emular un mouse o un joystick. Por su estructura física, el guante usa tela de lycra ligera con los sensores de flexión montados sobre la tela y sujetados con velcro. Al guante se abrocha una pequeña caja electrónica y se coloca en la parte superior de la muñeca. La interfaz del dispositivo es serial RS-232 (3 cables). El número de guantes que se pueden utilizar simultáneamente esta limitado por el número de puertos seriales con los que se cuenta. La interfaz es soportada en el sistema operativo Windows y DOS software que permite la instalación, calibración y representación gráfica de una mano virtual [49].

Figura 3.12 5th Glove desarrollado por Fifth Dimension Technologies

Dexterous Hand Master (DHM) (figura 3.13): Este es un exoesqueleto que usa sensores hall effect para proporcionar la medición de la articulación de flexión de los 4 dedos y el pulgar. Puede ser usado para proporcionar comandos de movimiento en un ambiente virtual, en un ambiente de teleoperación o para grabar la posición de los movimientos de los dedos [49]. A diferencia del *Dataglove* y el *Powerglove*, el *DHM* es capaz de detectar y medir el movimiento de lado a lado de un dedo. Los otros guantes sólo miden la flexión del dedo. El *DHM* es más exacto que cualquiera de los guantes y menos sensitivo al tamaño de la mano del usuario, pero puede ser difícil trabajar con éste. [1][46]

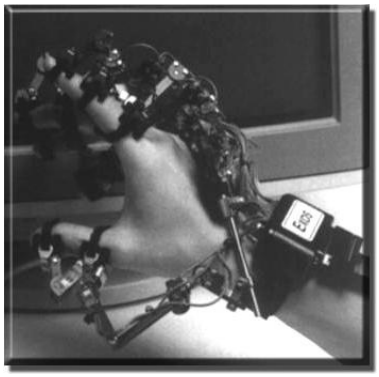


Figura 3.13 Dexterous Hand Master desarrollado por EXOS

La ventaja principal de los diferentes tipos de guantes es que proporcionan un dispositivo de interacción más intuitivo que el ratón o un joystick. Esto es porque los guantes permiten que la computadora lea y represente ademanes de la mano. [1][46]

3.2.1.1.3 Dispositivos hápticos

Los dispositivos hápticos (figura 3.14) son dispositivos que proporcionan salida desde la computadora hacia el usuario simulando retroalimentación de fuerza y tacto. Los usuarios pueden sentir fuerza, tensión, fricción, presión, temperatura y velocidad de los objetos.

Figura 3.14 Dispositivos hápticos

3.2.1.2 Rastreadores (trackers)

La información mínima que requiere un sistema de realidad virtual inmersivo es la posición y orientación de la cabeza del espectador, necesaria para el renderizado apropiado de imágenes. Además pueden ser rastreadas otras partes del cuerpo (por ejemplo la mano que permite la interacción, el pecho o las piernas) para permitir la representación gráfica del usuario. Los objetos en 3D tienen 6 grados de libertad (GDL): coordenadas de posición (desplazamientos *X, Y* y *Z*) y orientación (*yaw, pitch* y *roll*). Cada rastreador debe soportar estos datos o al menos un subconjunto de ellos. [23][51]

Los rastreadores son sensores que se encargan de detectar la posición y orientación de un objeto y deja disponible la información para el resto del sistema de realidad virtual. La función más común de un rastreador es reportar la posición y orientación de la cabeza o de la mano del usuario. [13] [71]

Muchos dispositivos de interacción incorporan alguna clase de rastreo para medir la posición y orientación de la parte del cuerpo a la que lo hayan sujetado. En términos de hardware, por lo general se requieren los siguientes tres componentes: una fuente que genera una señal, un sensor que recibe la señal y una caja de control que procesa la señal y la envía a la computadora. Dependiendo de la tecnología usada, también la fuente o el sensor esta sujetado al cuerpo, con el otro ubicado en un punto fijo en el ambiente, sirviendo como un punto de referencia.

Interferencia o sensibilidad a los factores del ambiente, puede limitar la eficiencia de los dispositivos, dependiendo de la tecnología utilizada, puede ser sensible a grandes objetos metálicos, radiación de la pantalla de los monitores, varios sonidos y objetos que se encuentran entre la fuente y el sensor.

Actualmente existen dos tipos de rastreadores:

- *Activos:* en estos el sensor o la fuente, se encuentra sujetada al punto que va a ser rastreado.

- *Pasivo:* el punto es monitoreado desde una distancia por una o varias cámaras, aunque este enfoque esta siendo favorecido desde la perspectiva del usuario, en este punto el tiempo no es una solución tecnológicamente viable.

La mayor parte de los rastreadores utilizados son activos. [67]

Los dispositivos de rastreo por lo general están basados en tecnología electromagnética, acústica, mecánica u óptica. Más adelante se hace un análisis de cada enfoque así como sus ventajas y desventajas. Antes de mencionar las propiedades de los rastreadores, se define que es un GDL.

Grados de libertad (GDL): Estos son utilizados usualmente para indicar cuantas variables espaciales diferentes puede medir el rastreador. Por ejemplo el rastreador de orientación que se encuentra en un HMD tiene 3 GDL (3 grados de libertad) desde estos se mide *pitch, roll* y *yaw*. Un sistema con 6 GDL también mide la posición en el espacio y proporciona toda la información que se necesita para un rastreo completo. Se debe tomar en cuenta que pueden ocurrir confusiones en los rastreadores mecánicos, desde 6 GDL también pueden ser tomados para indicar el número de articulaciones en un rastreador, pero no necesariamente el número de dimensiones que puede rastrear. [12]. En la figura 3.15 se muestran las seis variables que conforman un rastreo completo.

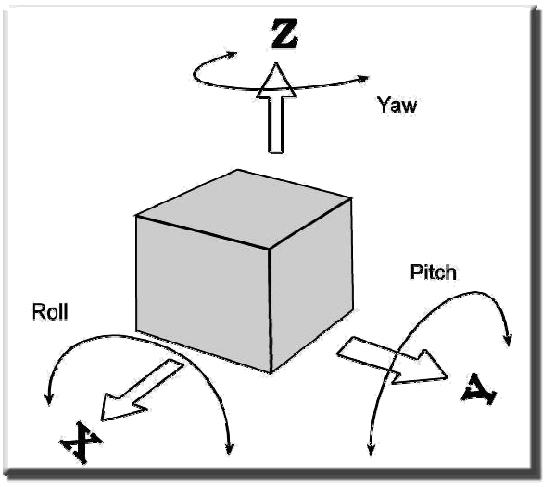


Figura 3.15 Grados de libertad

Las propiedades mas importantes de los rastreadores de 6 GDL, que deben ser consideradas para elegir el dispositivo correcto para una aplicación específica son las siguientes:

- *Índice de actualización:* define cuantas mediciones son realizadas por segundo (medidas en *Hz*) [23] [51].

- *Latencia:* cantidad de tiempo (usualmente medida en *ms*) entre la acción real (física) del usuario y el inicio de la transmisión del reporte que representa esta acción. Valores bajos contribuyen a un mejor desempeño. [23][19]

- *Exactitud:* es la medida del error en la posición y orientación reportadas. Definida generalmente en valores absolutos (*mm* para posición o *grados* para orientación) valores pequeños implican mayor exactitud [23][51]

- *Resolución:* pequeños cambios en la posición y orientación que pueden ser detectados por el rastreador. Valores pequeños implican un mejor desempeño. [23][51].

- *Rango:* capacidad de trabajo, dentro del cual el rastreador puede medir la posición y orientación con su exactitud y resolución específica y la cobertura angular del rastreador. [23] [51] El rango describe la distancia máxima que el sensor puede operar desde la base. [12]

Junto con estas características, se deben de tomar en cuenta algunos otros aspectos como la facilidad de uso, el tamaño, y el peso del dispositivo. Estas características serán utilizadas más a fondo para determinar la calidad y utilidad de los diferentes tipos de rastreadores.

3.2.2 Dispositivos de Salida

- *Generadores de imágenes:* cascos visores, sistemas binoculares, lentes estereoscópicos.

- *Generadores de sonidos:* cascos auditivos para incrementar la sensación espacial.

- *Elementos para la manipulación táctil:* incluyen las sensaciones de fuerzas de retroalimentación.

3.2.2. Generadores de imagen

La visión en profundidad se va a lograr mediante técnicas estereoscópicas, de tal manera que la visión de cada uno de los ojos esta ligeramente desplazada para conseguir la

impresión de profundidad. Los distintos elementos de visualización utilizados en realidad virtual se aprovechan de esta circunstancia.

Fundamentalmente podemos distinguir 3 tipos de elementos generadores de imagen:

Las gafas de obturación (figura 3.16): consisten en gafas de cristal líquido que electrónicamente oscurecen cada ojo, coordinados con el barrido de la imagen del monitor. Al ser la velocidad de obturación a unos 60 cuadros por segundo, el oscurecimiento no se percibe. De este modo se presenta consecutivamente la imagen izquierda y derecha, y las gafas permiten la visión respectiva obteniendo la visión en profundidad.

Figura 3.16 Gafas de Obturación

Head Mounted Display (HMD) (figura 3.17): presentan los ambientes virtuales a través de un dispositivo montado en la cabeza del usuario es parecido a un casco con un visor frente a los ojos. El HMD contiene pantallas separadas para cada uno de los ojos, de tal manera que cada uno observa diferentes vistas de la misma imagen. Estas imágenes se fusionan para que el cerebro produzca una imagen con un efecto en 3D. Los HMD también cuentan con rastreadores de posición para indicar a la computadora la posición actual de la cabeza de esta manera la vista del ambiente virtual puede ser actualizada con los movimientos de la cabeza del usuario.

Figura 3.17 Head Mounted Display

BOOM (Binocular Omni-Orientation Monitor) (figura 3.18): consiste en un brazo mecánico que realiza la función de un rastreador de posición, a la vez que sostiene un visor tipo HMD. Permiten incorporar sistemas de visualización más complejos que en el caso HMD.

Figura 3.18 BOOM (Binocular Omni-Orientation Monitor)

3.2.2.2 Cueva (*cave*)

Es un sistema que permite a uno o más usuarios estar completamente inmersos en un ambiente virtual (figura 3.19). Los usuarios entran a un pequeño cuarto con proyecciones de video en las paredes, las cuales se utilizan para rodear a los usuario con una serie de imágenes generadas por computadora. Los usuarios utilizan gafas para ver el ambiente y un rastreador de posición se coloca en los lentes para determinar la posición de la cabeza. [2][68]

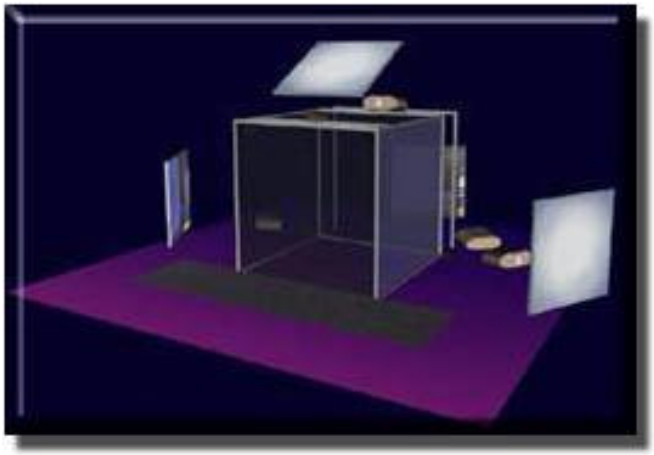


Figura 3.19 Cueva EVL, Universidad de Illinois Chicago

3.2.3 Tecnologías utilizadas para el desarrollo de dispositivos para RV

Debido a la importancia de los dispositivos para la realidad virtual en este trabajo, a continuación se mencionarán algunas de las tecnologías más utilizadas para el desarrollo de algunos dispositivos de interacción para ambientes virtuales.

3.2.3.1 Tecnologías empleadas en guantes

3.2.3.1.1 Electromecánicos

Las estructuras Dexterous Hand Master (DHM) miden la posición de los dedos utilizando un mecanismo de exoesqueleto montado en la mano y conectado a cada dedo. Los eslabones son tales que los ángulos de sus uniones varían como las articulaciones de los dedos con su inclinación. Estos cambios en la posición angular son medidos por un arreglo de sensores Hall-effect en las uniones mecánicas. El exoesqueleto es montado a la mano usando tiras de velcro, debido a que su posición puede variar, el DHM necesita ser recalibrado al iniciar cada sesión. El exoesqueleto es un poco pesado, causando fatiga si se usa por periodos largos de tiempo. En la figura 3.20 se muestra una estructura electromecánica.

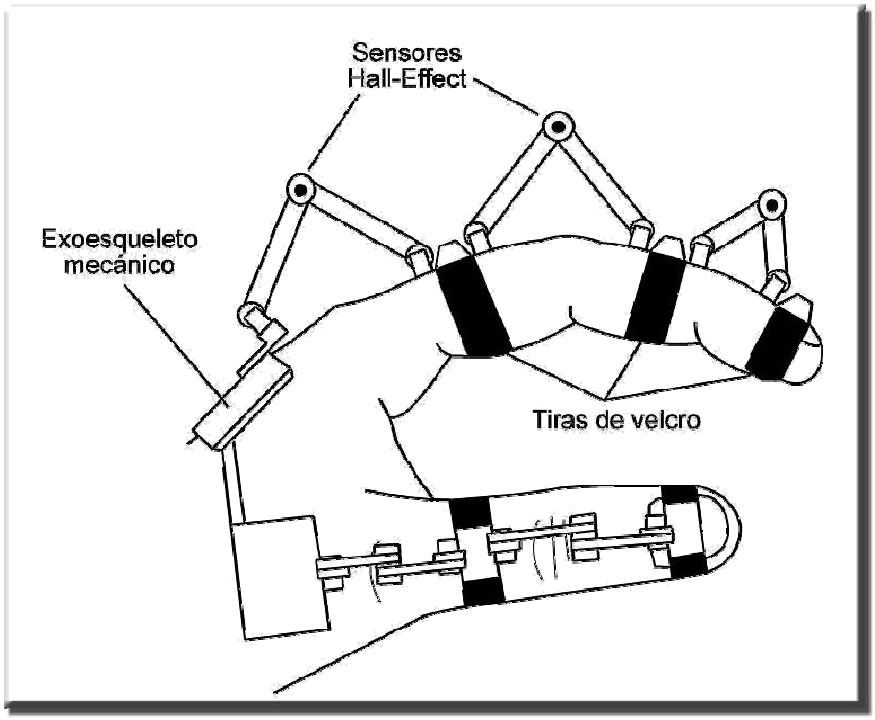


Figura 3.20 Sistema electromecánico

3.2.3.1.2 Localizadores ópticos

El guante de datos original de VPL usaba conexiones de fibra óptica separada pasando por encima de cada unión que se desea medir. El reflejo de la luz de un LED pasa entre la fibra y su intensidad es medida al final por un fototransistor. En la posición de la unión, la pared de la fibra es tratada tal que la luz no es absorbida cuando la fibra esta erguida, pero cuando la fibra esta inclinada, aumenta la cantidad de luz que es refractada. Las fibras ópticas están montadas en un guante de lycra el cual, es de peso ligero, esto significa que las fibras pueden inclinarse mientras el guante es puesto y removido, requiere calibración al inicio de cada sesión. En la figura 3.21 se representa el funcionamiento de un localizador óptico.

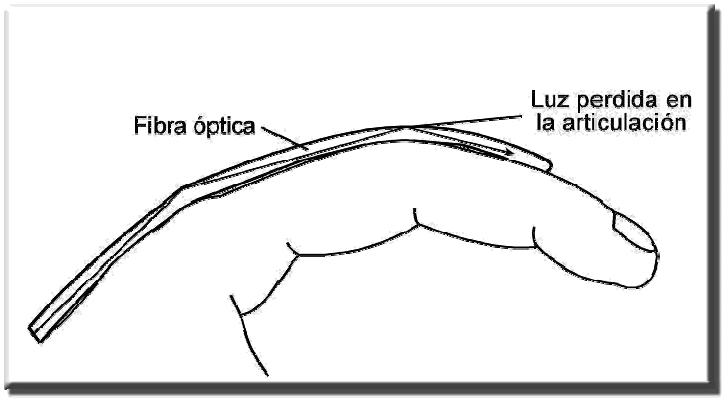


Figura 3.21 Sistema óptico

3.2.3.1.3 Strain Gauges

Los strain gauges (figura 3.22) son dispositivos con filamentos delgados que resisten variaciones ligeras conforme se aplica una tensión. Dos strain gauges están colocadas en cada articulación tal que cuando el dedo es doblado, en una de ellas es medida la compresión y en otra la tensión.

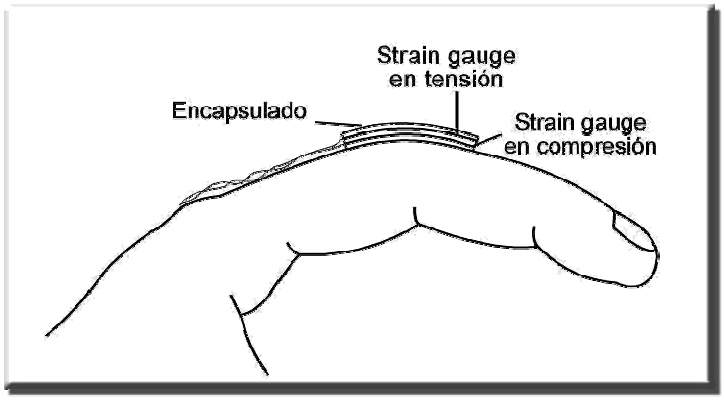


Figura 3.22 Strain Gauge

3.2.3.1.4 Conductive Ink Sensors

Una tecnología de bajo precio utilizada en el PowerGlove, cubre con un substrato una doble capa de un conductive ink que consta de muchas partículas de carbono (figura 3.23). Mientras el dedo dobla el substrato, la longitud de estas superficies incrementa, el espacio entre las partículas de carbono y el incremento en las resistencias de los sensores. El cambio en la resistencia es a menudo mucho mayor que cuando se usan strain gauges.

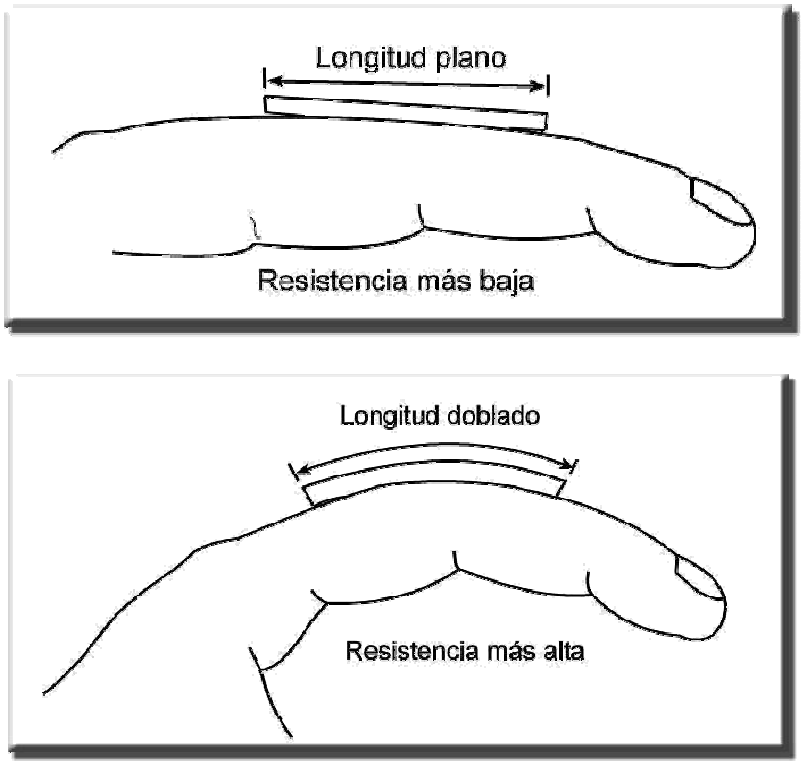


Figura 3.23 Conductive ink sensor

3.2.3.1.5 Guante para retroalimentación

Las tecnologías de realidad virtual han avanzado hasta la fase en que el guante puede ser usado para sentir a través de un mundo virtual. Aunque los ojos y los oídos son importantes para la obtención de información acerca de lo que rodea un mundo en general, no hay duda que las manos también lo son cuando se tiene contacto con objetos del mundo real.

Los dispositivos comerciales que adhieren tacto a los mundos virtuales solo tienen presentes las capacidades más rudimentarias. Estos dispositivos pueden estar ampliamente divididos en tactil feedback, force feedback y termo feedback.

3.2.3.1.6 Tecnologías de tacto

3.2.3.1.6.1 Force feedback

Es un sistema que permite al usuario sentir las fuerzas que son ejercidas sobre un objeto virtual. Si se esta colocando un objeto virtual sobre una mesa virtual, entonces la fuerza ejercida en las manos es la fuerza del propio cuerpo acelerado debido a la gravedad, y los sistemas forced feedback deben ser capaces de soportarlo de la misma forma como lo haría con una mesa real.

Actualmente los sistemas forced feedback específicos para la mano usan una de las dos aproximaciones a estos exoesqueletos.

- La primera usa un exoesqueleto montada por fuera de la mano, similar al usado por el rastreo electromecánico. Las articulaciones constan de un número de poleas que están sujetadas a pequeños motores usando cables. Los motores están montados fuera de la mano para reducir el peso, pero pueden ejercer fuerza en varios puntos de los dedos, tirando de los cables apropiados. La figura 3.24 muestra esta tecnología.

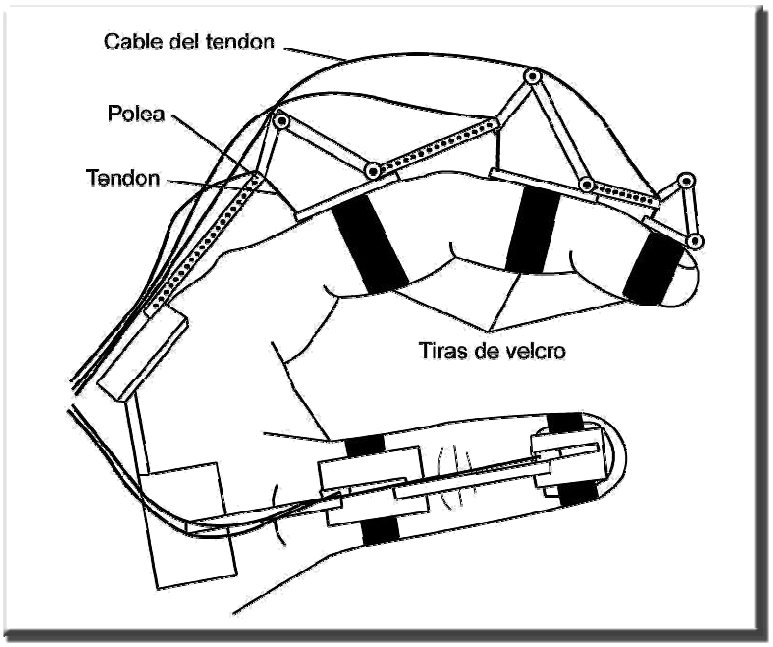


Figura 3.24 Force Feedback

- El segundo sistema se compone de una serie de pequeños pistones neumáticos entre las puntas de los dedos y una placa de apoyo en la palma de la mano (figura 3.25). Las fuerzas solo pueden ser aplicadas en las puntas de los dedos, aplicando presión a los pistones.

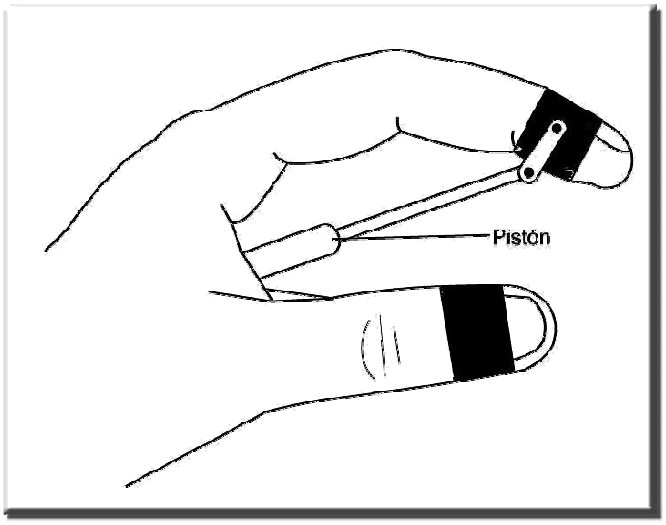


Figura 3.25 Force Feedback basado en pistones

Debido a que estos sistemas reflejan todas las fuerzas hacia alguna parte de la mano o la muñeca, estos permiten agarrar un objeto virtual y sentir su forma, pero no pueden impedir el paso de la mano a través del objeto.

3.2.3.1.6.2 Tactile feedback

El propósito del tactile feedback es transmitir una percepción de la textura o superficie del objeto, de igual manera puede transmitir alguna señal de la superficie geométrica del objeto. Las tecnologías utilizadas para este propósito son las siguientes:

- *Vibro-tactile (figura 3.26):* Se pueden utilizar vibradores eléctricos para hacer vibrar una superficie contra la punta de un dedo a varias frecuencias. Los sistemas de frecuencia únicamente están activados cuando los dedos hacen contacto con la superficie de un objeto virtual, para indicar la colisión.

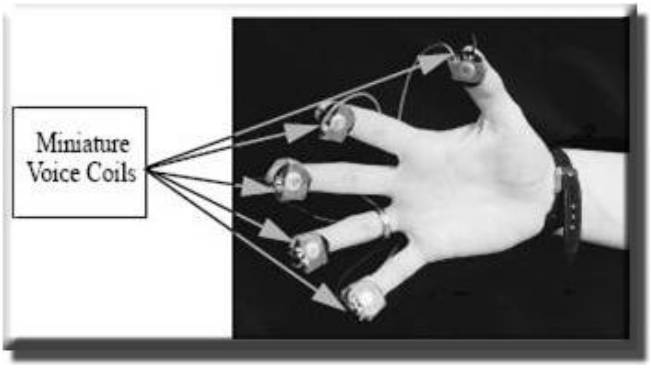


Figura 3.26 Guante con tecnología vibro-tactile

- *Electro-tactile:* Similar a los dispositivos vibro-tactile los dispositivos electro-tactile crean vibraciones de varias frecuencias y amplitudes en la punta de los dedos. Al contrario de los sistemas vibro-tactile, no hay movimientos involucrados, sin embargo, la sensación es causada por impulsos eléctricos aplicados a la piel.

- *Micro pin arrays (figura 3.27):* Al contrario de los métodos anteriores de vibraciones simples, micro pin arrays puede crear la sensación de texturas de superficies complejas. Estos dispositivos constan de una pequeña matriz de pines miniatura que pueden ser extendidos sobre la punta del dedo. Una ventaja del sistema micro pin array sobre los sistemas de vibración simple es la habilidad de crear la marca de los bordes de la superficies del objeto.

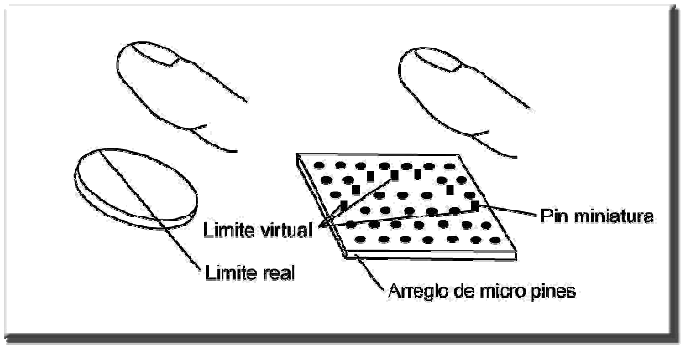


Figura 3.27 Micro pin array

- *Pneumatic (figura 3.28):* los sistemas anteriores están diseñados para crear una sensación de la textura de la superficie de un objeto en la punta de los dedos, pero no proporciona alguna señal que indique la forma del objeto que se esta tocando. Los sistemas pneumatic dan lugar a pequeños air pokets en puntos estratégicos del guante. Estos air pockets pueden expandirse individualmente a una presión específica generando modelos simples que emulan las fuerzas sentidas cuando se toca un objeto real.

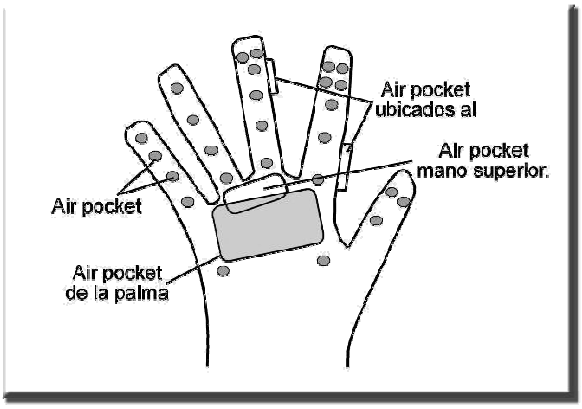


Figura 3.28 Guante con un sistema pneumatic

2.2.3.1.6.3 Termo feedback

Para mundos virtuales en donde es aceptable tocar solo objetos más calientes que la mano, se pueden usar pequeñas resistencias (heating coils) que crean las temperaturas correctas en las puntas de los dedos. Sin embargo, en la mayoría de los casos, la variación de temperaturas, ya sea en aumento o disminución tienen que ser generadas. Si la corriente es controlada por el ambiente virtual computarizado, la temperatura apropiada, calor o frio, puede ser generada en la punta de los dedos en respuesta a que la mano virtual toque un objeto virtual [12].

3.2.3.2 Tecnologías empleadas en rastreadores

3.2.3.2.1 Rastreadores magnéticos

Los rastreadores magnéticos (figura 3.29) son utilizados más a menudo en los dispositivos de rastreo cuando se trabaja en una aplicación inmersiva. Consisten típicamente en: *una parte estática* (el emisor, algunas veces llamado fuente), *un número de partes movibles* (el receptor, algunas veces llamados sensores) y *una estación de control.* Las antenas del emisor proporcionan la corriente, generan los campos magnéticos que son tomados por las antenas del receptor. El receptor envía sus medidas a la unidad de control que calcula la posición y orientación del sensor.

Bajo condiciones óptimas (carencia de algún tipo de interferencia magnética) los rastreadores magnéticos tienen un desempeño relativamente bueno. *Un dispositivo que emplea este tipo de tecnología es el WrightTrac de Vidtronics, Inc*. [32][48]

Ventajas:

- Los sensores son pequeños, ligeros y fáciles de manejar
- No tienen restricción en su línea de visión.
- No son sensibles a interferencia acústica.
- Cuentan con índices de actualización altos y latencia baja.

Desventajas:

- La magnitud de los campos magnéticos decrementa considerablemente con la distancia desde el emisor, la capacidad de trabajo de los rastreadores magnéticos es muy limitada y la resolución empeora cuando la distancia entre el emisor y el receptor aumenta.
- Los campos magnéticos están sujetos a distorsión, causada por los objetos metálicos dentro de este. Por otra parte, cualquier campo magnético externo puede causar distorsión adicional lo cual lleva a medidas erróneas. [23][51]

Figura 3.29 Emisor y receptor del Polhemus Fastrack

3.2.3.2.2 Rastreadores acústicos (ultrasónicos)

Los rastreadores acústicos (figura 3.30) usan ondas ultrasónicas (sobre 20 Khz.) para determinar la posición y la orientación del objeto en el espacio. Debido a que el uso del sonido permite determinar la distancia relativa solo entre dos puntos, se utilizan múltiples emisores (generalmente 3) y receptores (generalmente 3) con geometría conocida para adquirir un sistema de distancias para calcular la posición y orientación. Hay dos clases de rastreadores acústicos:

- *Los que utilizan el tiempo de vuelo (TOF):* miden el tiempo de vuelo de los pulsos ultrasónicos desde la fuente hasta el sensor.

- *Los que utilizan las medidas de la fase coherente (Phase Coherent PC):* comparan la fase de una señal de referencia con la fase de la señal recibida por los sensores. La diferencia entre dos medidas de fase sucesivas permite calcular el cambio de la distancia de la última medición. Debido a que este método entrega datos relativos, (el error tiende a acumularse con el tiempo) el desarrollo de los rastreadores en PC's fue abandonado.

Ventajas:
- Ligeros y pequeños.
- Relativamente baratos. (1000 USD)
- No experimenta interferencia magnética.

Desventajas:
- Campo de visión restringido.
- Experimenta interferencia acústica, el ruido o el eco pueden llevar a medidas erróneas.
- Índice de actualización bajo.[23] [51]

Figura 3.30 Rastreador ultrasónico Logithech de 6 GDL

3.2.3.2.3 Rastreadores ópticos

Los sistemas ópticos (figura 3.31) se basan en medir la luz reflejada o emitida, este sistema tiene 2 componentes importantes: *fuente de la luz y sensores ópticos*. [31]

Usa una variedad de detectores, desde una cámara de video ordinaria hasta un conjunto de LEDs, para detectar luz del ambiente o luz emitida bajo el control de un rastreador de posición. A menudo se utiliza la luz infrarroja para evitar interferencias con otras actividades. [32][48].

Existen varios tipos de configuración para los rastreadores ópticos. Generalmente se pueden dividir en tres categorías:

- **Rastreadores Beacon (faro):** esta tecnología utiliza un grupo de faros (LEDs) y un sistema de cámaras las cuales capturan imágenes del patrón de los faros. Puesto que se conocen las geometrías de los faros y los detectores, se puede determinar la posición y orientación del cuerpo rastreado.

- **Reconocimiento de patrones:** estos sistemas no usan ningún tipo de faros, se encargan de determinar la posición y orientación comparando patrones conocidos que han sido detectados. Hasta ahora no se ha desarrollado ningún sistema que funcione completamente con esta tecnología.

- **Laser ranking:** estos sistemas transmiten sobre el objeto una luz láser que pasa a través de una rejilla de difracción. Un sensor analiza el patrón de difracción en la superficie del cuerpo al que se va a calcular su posición y orientación.

Para todos estos sistemas la exactitud decrementa significativamente cuando incrementa la distancia entre el sensor y el objeto rastreado.

Un dispositivo que emplea este tipo de tecnología es el *SELSPOT de Selcom Ab.* [32][48] Cabe mencionar que el guante P5 emplea este tipo de tecnología para determinar la posición de la mano, donde los LEDs del guante envían señales hacia el receptor (torre).

Ventajas:

- Índice de actualización alto (sobre 240 Hz. En muchos de los casos este es limitado por la velocidad de la computadora utilizada)
- Posibilidad de prolongar al límite más grande la capacidad de trabajo.
- No es sensible a la presencia de objetos metálicos o que produzcan campos magnéticos, no es sensible a interferencia acústica.
- Relativamente buena exactitud: orden de magnitud de 1mm y 0.1º

Desventajas:

- Campo de visión restringido.
- La luz del ambiente y la radiación infrarroja pueden influir en el desempeño.
- Costosos y a menudo muy complicados de construir.
- Dificultad al rastrear más de un objeto con un solo rastreador. [23] [51]

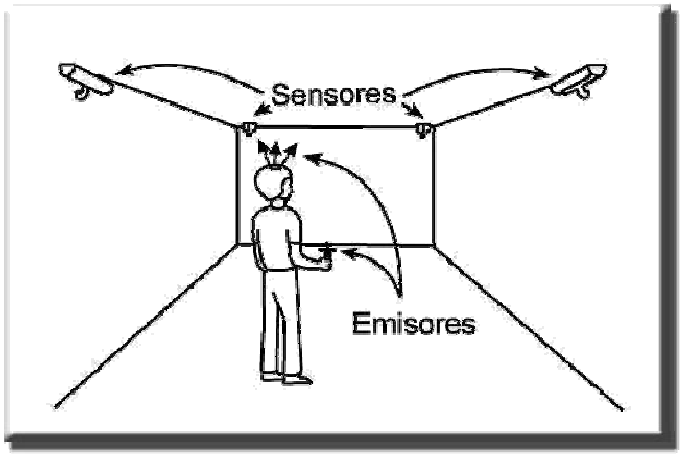


Figura 3.31 Rastreadores ópticos

3.2.3.2.4 Rastreadores mecánicos

Conceptualmente el enfoque más simple, involucra alguna forma de conexión directa entre el objetivo y el ambiente. [31]

Con esta tecnología se miden la posición y orientación usando una conexión mecánica directa entre el punto de referencia y el objetivo. [67] Se usa una liga mecánica de brazos rígidos con articulaciones entre ellos para medir la posición y orientación de un punto libre (unido al final de la estructura) en relación a la base. Los ángulos de las articulaciones se miden con ayuda de engranes o potenciómetros, combinados con el conocimiento de la construcción de articulaciones permite derivar los valores requeridos de posición y orientación.

Ventajas:

- Muy exactos.
- Inmune a cualquier tipo de interferencia (menos a obstáculos mecánicos).
- Índice de actualización alto (arriba de 300 HZ).
- Soporta la tecnología force-feedback.

Desventajas:

- No es completamente libre en sus movimientos debido a sus articulaciones mecánicas.
- Capacidad de trabajo pequeña (cerca de un metro cúbico).
- Solo se puede rastrear un objeto por dispositivo. [23] [51]

Un ejemplo de estos dispositivos es el BOOM desarrollado por *Fake Space Labs* (figura 3.32). [67]

Figura 3.32 BOOM de Fake Space Labs

Capítulo 4

Marco tecnológico

Una herramienta es algo que ayuda a realizar un trabajo, en este capítulo se describen todas las herramientas utilizadas para el desarrollo de este proyecto de investigación. Entre las herramientas que se mencionan se encuentran VRML, Delphi, Java, MATLAB, VrmlPad, Cortona VRML Client 4.2 y JCreator

4.1 Virtual Reality Modeling Language (VRML)

El lenguaje VRML (Virtual Reality Modelling Language) es un lenguaje abierto de descripción de entornos tridimensionales. Se creo en 1994 en Silicon Graphics. [17] VRML se difunde a partir de la conferencia anual sobre World Wide Web; los autores, Tim Bernes Lee y David Roget, presentaron una ponencia titulada "Los Lenguajes de Marcación de Realidad Virtual y el Word Wide Web" y a partir de esta presentación los asistentes a la conferencia se comprometieron a delinear los requerimientos básicos para generar un producto para diseño 3D, que fuese equivalente al estándar HTML. [25]

En 1995 se publicó la versión 1.0, y en 1996 la 2.0. A finales de 1997, con ligeros cambios respecto a la versión 2.0, el lenguaje VRML97 fue estandarizado por ISO (ISO/IEC 14772-1:1997). La filosofía de VRML es muy similar a la de HTML (HyperText Markup Language). Al ser un estándar abierto no está limitado a ninguna aplicación o plataforma, y existen varios visores gratuitos. Por otro lado, VRML no se limita a la descripción de escenas estáticas, sino que puede utilizarse también para describir mundos dinámicos en los que el observador puede interactuar con los objetos que ve.

Los campos de aplicación del lenguaje VRML son muy amplias: documentación técnica, arqueología, demostraciones, enseñanza, etc. [17]

VRML ha sido diseñado para satisfacer los siguientes requerimientos:

- *Autoría:* Permite el desarrollo de programas de computadora, capaces de crear, editar y mantener archivos VRML, así como programas de conversión de otros formatos de archivos 3D, comúnmente utilizados, a archivos VRML.

- *Composición:* Proporciona la posibilidad de utilizar y combinar objetos 3D dinámicos dentro de un mundo VRML y así permitir la reusabilidad.

- *Extensión:* Proporciona la posibilidad de agregar nuevos tipos de objetos no definidos explícitamente en VRML.

- *Capacidad de Implementación:* Permite implementar sobre un amplio rango de sistemas.

- *Desempeño:* Enfatiza funcionamiento interactivo y escalable sobre una amplia variedad de plataformas de cómputo.

- *Escalabilidad:* Permite la descripción de mundos tridimensionales dinámicos arbitrariamente grandes. [27][62]

Una característica importante de los archivos de VRML, es la capacidad de incluir todos los archivos de un ambiente virtual en un solo archivo al mismo tiempo y relacionarlos por medios de ligas. Al incluir los archivos de forma jerárquica se permite la creación de

mundos dinámicos grandes. Por consiguiente, VRML asegura que cada archivo es descrito completamente por los objetos contenidos dentro de él. [29]

Un mundo VRML puede estar distribuido, es decir, residir en múltiples puntos de la red en distintos lugares geográficos, de la misma manera que una página HTML puede estar compuesta de texto en un lugar y de imágenes en otro. Un mundo VRML puede especificar que los escenarios estén en un servidor, mientras que en otro están los objetos para dicha escena.

A diferencia de otros lenguajes como C que son compilados en programas ejecutables, VRML es interpretado y a partir de esto es desplegado el escenario virtual en pantalla. [17]

4.1.1 Browser para VRML

Existen diferentes tipos de browsers para poder visualizar archivos VRML los cuales se conocen como plug-in's.

- Plug-in: es un accesorio que se conecta al navegador con el fin de visualizar o ejecutar aplicaciones adicionales de video, audio, 3D, comunicaciones telefónicas o multimedia, por lo general son de carácter gratuito y de acceso libre a la red. En el mercado existen varios de estos programas entre los que destacan:

  - ✓ Cortona VRML Client.
  - ✓ Blaxuun Contact.
  - ✓ Octagon player.
  - ✓ FreeWRL
  - ✓ BS Contact VRML/X3D
  - ✓ Flux
  - ✓ Vcom3d Venues
  - ✓ Cosmo Player.

4.1.2 Elementos fundamentales en VRML

En un mundo VRML lo que se define es simplemente un grupo de objetos. Estos objetos pueden contener figuras geométricas en 3D, imágenes, colores, etc., en VRML se les llaman *nodos.* Los nodos son el componente principal de un gráfico en VRML, tienen una serie de características esenciales para el desarrollo de ambientes virtuales, estas características definen la conducta de los nodos.

Los elementos que constituyen a los nodos son: campos, eventos, exposed field, Route.

Los *campos* son los parámetros que distinguen un nodo de otro del mismo tipo, funcionan como parámetros de un nodo. [17] Los campos pueden tener varios tipos de datos, a continuación se listan algunos de los tipos mas utilizados en los campos de VRML: [9]

Dato VRML	Descripción
SFBool	Es un valor booleano que puede tomar el valor "True" o "False".
SFFloat	Es un valor de punto flotante de 32 bits.
SFInt32	Es un valor entero de 32 bits.
SFTime	Es el valor absoluto o relativo del tiempo.
SFVec2f	Es un vector con 2 valores de punto flotante.
SFVec3f	Es un vector con 3 valores de punto flotante.
SFColor	Es un vector con 3 valores de punto flotante, los cuales se utilizan para especificar color RGB.
SFRotation	Es un vector de 4 valores de punto flotante, se utiliza para indicar rotación.
SFImage	Arreglo bidimensional, representa una secuencia de números de punto flotante.
SFString	Cadena de caracteres en codificación UTF-8. Compatible con ASCII, debido a que es un subconjunto de UTF 8.
SFNode	Contenedor para un nodo VRML.
MFFloat	Arreglo de valores SFFloat.
MFInt32	Arreglo de valores SFInt32.
MFVec2f	Arreglo de valores SFVec2.
MFVec3f	Arreglo de valores SFVec3.
MFColor	Arreglo de valores SFColor.
MFRotation	Arreglo de valores SFRotation.
MFString	Arreglo de valores SFString.
MFNode	Arreglo de valores SFNode.

Tabla 4.1 Tipos de datos en VRML y su descripción

Un nodo puede ser un parámetro de otro nodo y, por lo tanto, puede ser un campo. Esto se da cuando un nodo es un agrupador de nodos. Un nodo agrupador sirve para que varios objetos tengan unas mismas propiedades. [9]

Otro elemento importante en los nodos de VRML son los *eventos*, que son mensajes enviados de un nodo a otro, definidos por un Route. [29] De acuerdo a las operaciones que realizan los eventos se pueden dividir en tres tipos, eventIn, eventOut y exposed field.

- *EventIn:* son eventos entrantes que aceptan información de fuera del nodo y hacen algo con él. Normalmente, los eventos del eventIn corresponden a un campo en el nodo. Los campos del nodo no son accesibles de fuera del nodo. La única manera en que se pueden cambiar es teniendo un eventIn correspondiente.

- *EventOut:* son eventos emergentes (salientes) que generan información, como cambiar un valor o el tiempo de un clic del ratón.

- *Exposed field:* son campos que pueden recibir eventos para cambiar su valor y genera eventos cuando ha cambiado su valor. Este tipo de campo indica que el nodo tiene dos eventos definidos, set_campo **y** campo_changed**.** Éstos son un eventIn **y** eventOut para el campo que puede usarse para cambiar su valor y notificar al mundo externo cuando ha cambiado.

  Los exposed field definen cómo se comporta el eventIn y el eventOut correspondiente. Para todas las clases del exposed field, cuando ocurre un evento, se cambia el valor del campo, con un cambio correspondiente en la apariencia de la escena, y se envía un eventOut con el nuevo valor del campo. Esto permite el encadenamiento de eventos a través de muchos nodos. [29]

Para hacer cosas útiles con los eventos, se tienen que unir de alguna forma, esta unión es conocida como *route*. Route es la conexión entre un nodo que genera un evento y un nodo que recibe un evento, es decir, son las sentencias que se encargan de relacionar los eventos de entrada con los eventos de salida. [9]

4.1.3 Sistemas de coordenadas en VRML

El sistema de coordenadas en VRML es de gran importancia para el desarrollo de los ambientes virtuales, ya que por medio de éstas se puede definir la posición y orientación de un objeto dentro del mundo virtual. Las coordenadas en VRML se definen de la siguiente forma: el eje X es horizontal, Y es vertical y Z atraviesa la pantalla hacia fuera o dentro según sea el caso (figura 4.1).

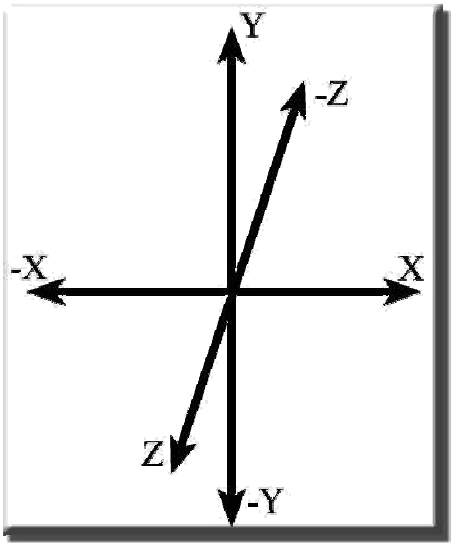


Figura 4.1 Sistema de coordenadas

El sistema de rotación en VRML trabaja con los valores en radianes y no en grados directamente, esta se representa en los ambientes virtuales como se muestra en la figura 4.2:

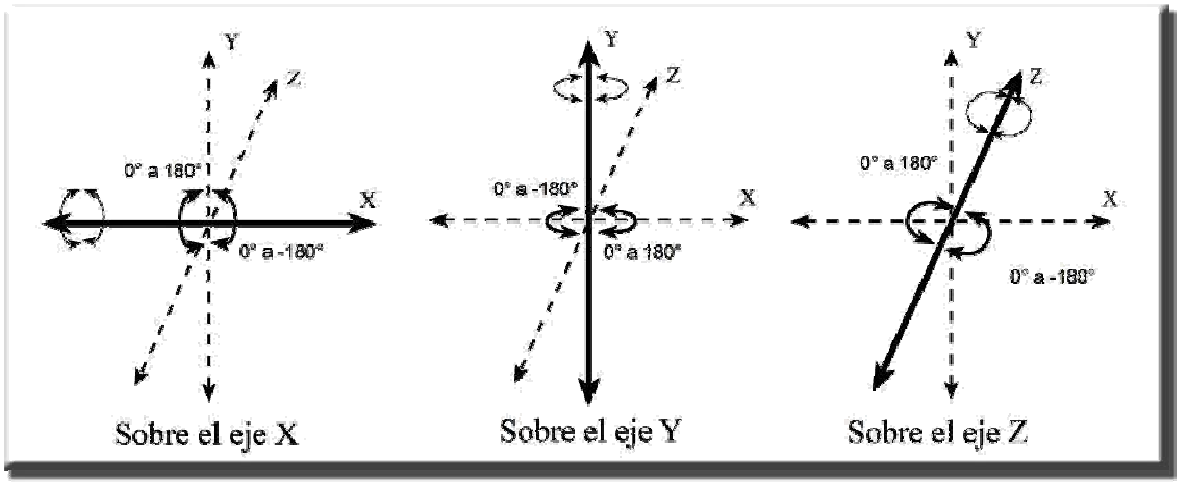


Figura 4.2 Rotación de los objetos en VRML

4.1.4 Comportamientos en ambientes virtuales

Comportamiento se refiere a la modificación del estado de los objetos dentro de un escenario virtual, en respuesta a alguna acción del usuario o al cambio de algún otro objeto en la escena. Este cambio de estado ocurre cuando cambia alguno de los atributos del objeto, por ejemplo, la posición (traslación), el tamaño (escala), la orientación (rotación) o su apariencia.

En VRML, el comportamiento se controla mediante eventos. Algunos nodos VRML generan eventos en respuesta a cambios en el ambiente o interacción del usuario. Una vez generados, los eventos son enviados a sus destinos para ser procesados por el nodo receptor. Este proceso puede cambiar el estado de un nodo o generar eventos adicionales.

El comportamiento puede ser de dos tipos, *simple y complejo*, según sea el grado de dificultad de la operación a la que será sometido:

- *Comportamiento simple:* es aquel en el cual los cambios en el estado de los objetos dependen exclusivamente de eventos internos generados por nodos descritos dentro del programa VRML, sin la intervención de ningún lenguaje ajeno a VRML. La programación de estos eventos se realiza enrutando lo eventos de un nodo a otro.

- *Comportamiento complejo:* es aquel comportamiento en el cual los cambios en el estado de los objetos dependen de un programa escrito en un lenguaje ajeno a VRML. Los cambios dinámicos en la escena pueden ser estimulados por las acciones programadas en un script, paso de mensajes, comandos del usuario o

protocolos de comportamiento. Esto le permite a VRML interactuar con otros lenguajes como Java o Javascript.[27]

4.1.4.1 Asignación de comportamientos

La asignación de comportamientos se clasifica de acuerdo a las siguientes dimensiones: ubicación y comunicación.

4.1.4.1.1 Ubicación

Se refiere a la localización física del código que define el comportamiento, puede ser de dos tipos:

- Local: el código de comportamiento se ejecuta en la máquina donde ésta la escena VRML.

- Remota: el código de comportamiento se puede estar ejecutando en cualquier máquina en la red.

4.1.4.1.2 Comunicación

Se refiere a la forma como se realizan el envío o la recepción de datos de la escena VRML al programa de comportamiento:

- *Dependiente*: consiste en enviar un mensaje, desde el escenario VRML al programa de comportamiento para actualizar los valores de los objetos dentro del escenario.

- *Independiente*: consiste en un programa de comportamiento que realiza una tarea determinada y envía datos al escenario para que actualice el estado de los objetos dentro de él.

- *Dúplex*: Es una mezcla de los dos modelos anteriores. El esquema de comunicación dúplex es más completo y complejo.

4.1.4.2 Lenguajes de programación

VRML brinda soporte para Java y Javascript como lenguajes de asignación de comportamientos: esto se realiza utilizando el API correspondiente a cada lenguaje, en el cual se definen todas las clases y métodos para tener acceso a los nodos de VRML, a los mecanismos que se encargan de comunicar a Javascript con VRML se le llama *Java Scripting Authoring Interface (JSAI)* y a la comunicación entre Java y VRML se le conoce como *External Authoring Interface (EAI)* (figura 4.3).

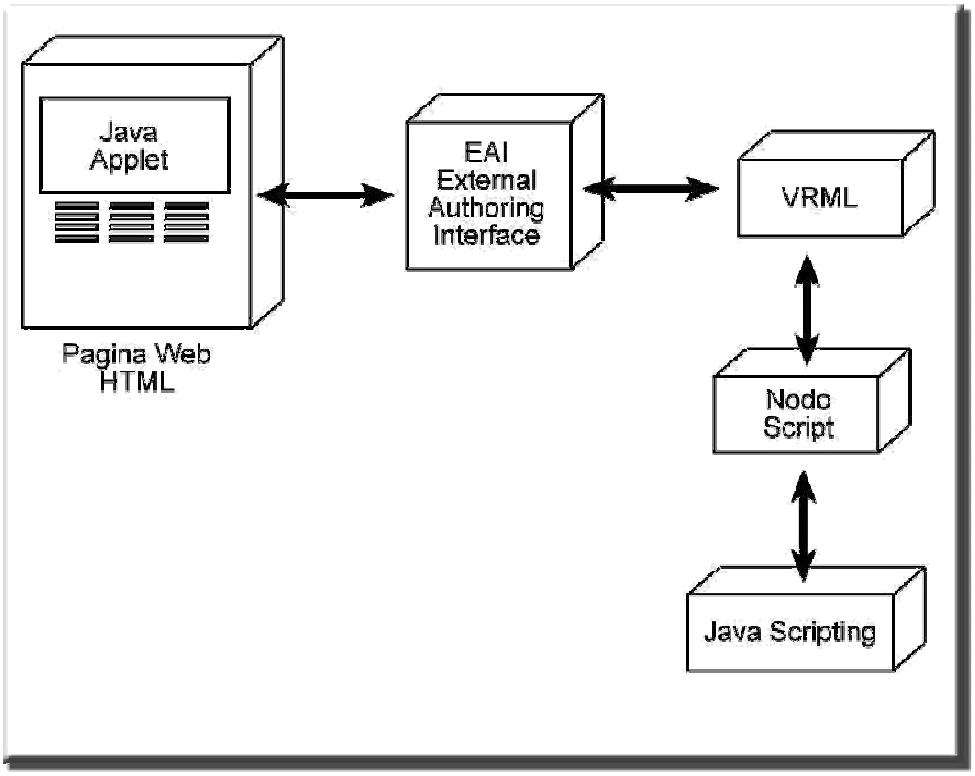


Figura 4.3 JSAI y EAI permiten asignar comportamientos a VRML

4.1.4.2.1 Java Scripting Authoring Interface. (JSAI)

El Java Scripting Authoring Interface es una forma de comunicar la escena VRML con Javascript desde adentro de la propia escena. Javascript es un lenguaje de programación el cual es interpretado por el navegador cuando se ejecuta. En la actualidad Javascript es de gran ayuda cuando se trata de asignar una animación a un ambiente virtual y esto lo hace mediante el uso del nodo script. El API para este lenguaje de programación se encuentra en las especificaciones VRML 2.0. [47]

Javascript es más fácil de programar que Java y los scripts realizados en éste no requieren de compilación para ser ejecutados. Los scripts simples escritos en Javascript son generalmente más pequeños que un script equivalente en Java, y poner el código del script directamente en un archivo VRML significa que el navegador tiene que realizar menos peticiones a la red.

Javascript es un lenguaje de programación secuencial. En esto es muy diferente VRML, que es más descriptivo. VRML declara la definición de diversos objetos, cada uno de los cuales cuenta con distintos atributos, los cuales, en su conjunto, describen un mundo completo. Un programa de Javascript es una sucesión de órdenes que se ejecutan en algún momento para dar lugar a un conjunto de resultados que se desean obtener. [44]

Un programa de Javascript es una sucesión de sentencias, cada una de los cuales ejecutan ciertas órdenes. Por ejemplo, la sentencia

```
x = a + b;
```

El programa toma los valores de a y b, y los suma, almacenando el resultado en x

4.1.4.2.2 External Authoring Interface (EAI)

El External Authoring Interface fue creado por la necesidad que tenían los desarrolladores de ambientes virtuales de comunicarse con una escena desde fuera de ésta. De esta necesidad surge un API para applets de Java que permite tener acceso a los nodos de una escena VRML, es decir sirve como puente entre HTML y VRML.

El EAI permite comunicar una escena VRML y un applet de Java embebido en una página HTML. El applet puede estar en la misma página o en un frame, de hecho permite realizar todas las operaciones que un frame puede permitir. [52]

El EAI permite manipular la escena en VRML de la siguiente forma:

- El navegador de VRML se encarga de la geometría y el render del ambiente virtual.
- Java se encarga del comportamiento y la lógica.

Para el manejo de VRML desde Java, se utiliza un paquete llamado vrml.external. A continuación se muestran las clases incluidas en el paquete vrml.external.* de Java, que son clases derivadas de Java.lang.RuntimeException

Clases	Descripción
vrml.external.Browser	Clase del navegador, es la representación del mundo VRML en el Applet de Java.
vrml.external.Node.	Clases de nodos, representación de un nodo VRML
vrml.external.field.*	Clase de tipos, representa todos los campos de un nodo correspondientes al lenguaje VRML.
vrml.external.exception.*	Clase de excepciones, representa el manejo de errores.

Tabla 4.2 Clases del paquete vrml.external

4.1.4.2.2.1 Formas de acceso a la escena VRML

El EAI permite cuatro tipos de acceso a las escenas de VRML:

- Acceso a la funcionalidad del Browser.
- Enviar eventos hacia los eventIns del nodo dentro de la escena.
- Leer el último valor enviado del eventOut del nodo dentro de la escena.
- Notifica cuando los eventos son enviados por los eventOuts del nodo dentro de la escena.

EAI fue desarrollado posteriormente a JSAI, de ahí que los tres primeros tipos de acceso también son utilizados por JSAI. Existen dos diferencias conceptuales entre EAI y JSAI. El primero tiene que obtener un puntero hacia el nodo por medio del cual se puede tener acceso a los eventIns y eventOuts del nodo. Cuando se crea un archivo VRML el nodo Script puede tener un puntero hacia un nodo que tenga el constructor USE. Puesto que el Applet no tiene acceso implícito a este mecanismo de instancia se proporciona un método explicito para obtener el puntero al nodo mediante el nombre que utiliza en el constructor DEF. El segundo es que no se puede crear un Route entre la escena VRML y el Applet. Sin embargo, el applet debe crear un método para que sea llamado cuando ocurra un eventOut. Este método se registra con el eventOut de un nodo. Cuando el eventOut genera un evento, se llama al método registrado. [19]

Los objetos eventIn son usados para asignar información acerca del estado del nodo VRML o, en otras palabras, para enviar un evento de entrada al nodo. Un eventOut es un evento enviado por un nodo VRML (figura 4.4). Para poder hacerlo existen dos operaciones, una para recuperar información acerca del estado del nodo y otra para asignar una función que será llamada cada vez que se envíe un evento de salida al nodo.

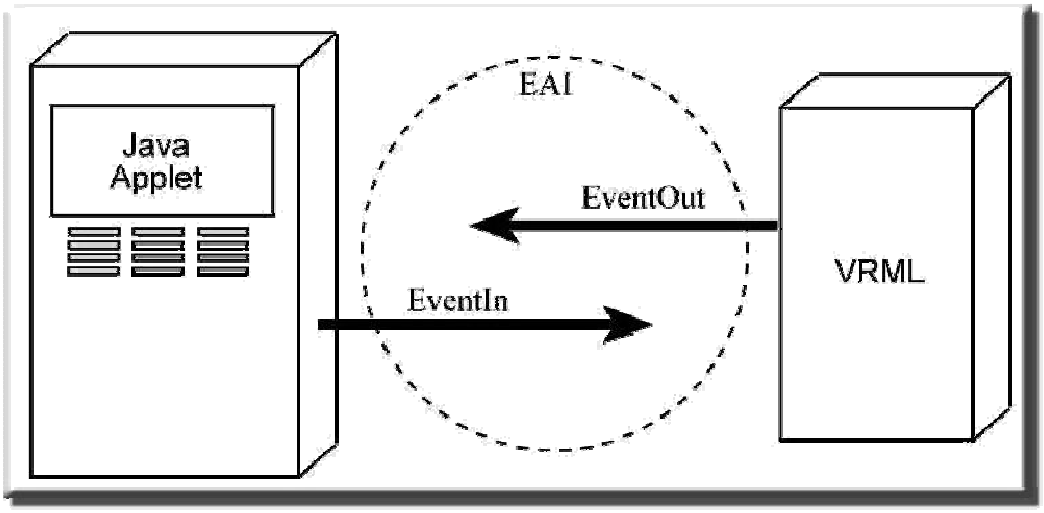


Figura 4.4 Funcionamiento de los EventIn y EventOut

A los nodos de VRML se les puede asignar un nombre utilizando el constructor DEF. Por medio del Applet de puede tener acceso a cualquier nodo que utiliza el constructor DEF. [19]

4.2 Scripts

Las habilidades de animación proporcionadas por los sensores e interpoladores de VRML, son utilizados para la animación relativamente directa, sin embargo, para animaciones complejas, los sensores e interpoladores son en algunas ocasiones muy limitados. En estas situaciones se puede utilizar el nodo Script de VRML.

Un nodo Script es el esqueleto de un nodo, el cual cuenta con campos, eventIns y eventOuts, pero no tienen acciones propias. En su lugar, se pueden asignar sus propias acciones al nodo script utilizando un programa Script. Los programas Script son pequeñas aplicaciones escritas en los lenguajes de programación Java o Javascript.

Utilizando un programa Script y un nodo Script, se pueden crear nodos para realizar acciones complejas. Las acciones típicas de un programa script incluyen la salida de las trayectorias calculadas, como la trayectoria de un objeto cayendo debido a la gravedad. Otra acción típica de un programa script es la manipulación de la inteligencia artificial de un monstruo en un juego o monitorear el nivel de vida de un jugador. Se puede utilizar programas script para comunicarse con computadoras remotas en Internet o rastrear las actividades de los jugadores en un ambiente multi jugador.

El nodo Script, junto con los programas Script escritos en lenguajes Java y Javascript, proporcionan una extensión de gran alcance a VRML, con el cual se pueden programar sensores e interpoladores personalizados.

4.2.1 Nodo Script

El nodo Script puede describir un sensor o un interpolador personalizado como cualquier otro nodo o interpolador, el nodo requiere una lista de campos, eventIns y eventOuts. El sensor o interpolador personalizado también requiere una descripción acerca de que se va a hacer con estos campos, eventIns y eventOuts.

Los campos, eventIns y eventOuts, describen la interfaz de las acciones del nodo. Por ejemplo, un nodo Script que calcula la trayectoria de una figura cayendo, puede tener campos los cuales especifican los valores de la fuerza de gravedad, la altura inicial, la velocidad, entre otros. El mismo script de la gravedad puede incluir una fracción de tiempo tal como un Time Sensor y un eventOut con una posición en 3D la cual será asignada al nodo Transform de la figura cayendo.

Además de la interfaz, el sensor o interpolador personalizado también necesita una descripción de que es lo que tiene que hacer. Esta descripción es el programa Script, el cual

generalmente es escrito en Java o Javascript. Por ejemplo, el programa script para el nodo Script de la gravedad, realiza los cálculos necesarios para generar la trayectoria para la animación de la figura cayendo.

4.2.2 Script en Java y Javascript

Java es un lenguaje de programación creado por Sun Microsystems, Javascript es un lenguaje más sencillo desarrollado por Netscape Communications. Ambos lenguajes permiten añadir animación a páginas Web o controlar actividades dentro de un mundo VRML. A pesar de la similitud en sus nombres, los lenguajes son considerablemente diferentes. Javascript es más fácil de aprender, mientras que Java proporciona rasgos extensos necesarios para proyectos de programación más grandes. Puede utilizarse cualquiera de estos lenguajes para escribir el programa para un nodo script.

Las caracteristicas del browser VRML, para un programa script se encuentran clasificadas como parte del API del lenguaje. El API del programa script incluye características en las siguientes categorías:

- Características para acceder a los campos y a los eventOuts de la interfaz del script del nodo Script.
- Características para convertir valores de los tipos de datos de Javascript a VRML.
- Funciones para manejar la inicialización, la detección, y la entrega de eventos.
- Métodos para tener acceso a las características del browser, incluyendo cambios en el mundo actual, y cargar nuevos mundos.

4.2.2.1 Conversión entre tipo de datos

Tanto Java como Javascript proporcionan sus propios tipos de datos cada uno para manejar datos numéricos, cadenas y otros tipos de datos.

- En Java, el API proporciona un extenso conjunto de clases para cada tipo de datos de VRML. Los métodos en estas clases proporcionan la conversión entre los tipos de dato de VRML y Java respectivamente.
- En Javascript, el API automáticamente convierte los datos de VRML a Javascript sin necesidad de llamar a ninguna función. En la siguiente tabla se muestra los tipos de datos en VRML así como sus correspondientes tipos en Javascript.

Tipos de datos	
VRML	**Javascript**
SFBool	Bolean
SFColor/MFColor	Arreglo de números
SFFloat/MFFloat	Número o arreglo de números.
SFImage	Arreglo de números
SFInt32/MFInt32	Número o arreglo de números.
SFNode/MFNode	Objeto o arreglo de objetos.
SFRotation/MFRotation	Arreglo de números.
SFString/MFString	Cadena o arreglo de cadenas.
SFTime	Número
SFVec2f/MFVec2f	Arreglo de números.
SFVec3f/MFVec3f	Arreglo de números.

Tabla 4.3 Equivalencia de datos entre VRML y Javascript

Las APIs de Java y Javascript proporcionan un tipo de datos especiales diseñados para tener influencia sobre el valor de un nodo. Utilizando las características del API, el script puede leer y escribir los exposed fields, eventIns y eventOuts de cualquier nodo al que tenga acceso.

4.2.2.2 Manejo de eventos

Un programa script puede responder a tres tipos de actividades:

- Inicialización.
- Cierre (shutdown).
- Recepción de eventos.

Inicialización: ocurre cuando el script es el primero en leerse y antes de que se le haya enviado cualquier evento. Es cuando el script tiene la oportunidad de poner valores iniciales y preparase para la recepción de eventos futuros.

Cierre (shutdown): ocurre cuando el script es eliminado, quizás como el resultado de dejar el navegador o leer un nuevo mundo. Es cuando un script limpia los valores para los cuales fue creado.

Recepción de eventos: ocurre cuando es recibido un nuevo evento de entrada por un eventIn del script. Es cuando el script debe recibir un valor nuevo de un evento, realiza una operación y genera uno o más eventos utilizando los eventOuts.

4.3 Borland Delphi

Delphi es desarrollado por Borland, la primera versión se lanzo al mercado en el año 1995 y corría bajo el sistema Windows 3.1, Delphi realmente no es un lenguaje, sino una IDE (Integrated Development Environment), es decir el espacio de trabajo para Object Pascal, el cual es un lenguaje de programación de alto nivel.

Object Pascal, es un lenguaje que surge a partir del desarrollo del Borland Pascal 7.0, un lenguaje que ocupa un lugar muy importante en la programación. Object Pascal como su nombre indica es Pascal orientado a objetos. Delphi incorpora un modelo completo de programación orientada a objetos, incluyendo encapsulación, herencia simple y polimorfismo.

En Delphi, los objetos se comunican mediante mensajes, de esta manera cuando se realiza una acción se envía un mensaje que genera un evento. Además Delphi utiliza la RAD (Rapid Application Development), que permite desarrollar programas de forma rápida y visual. Delphi utiliza como componentes los controles ActiveX. Desde el punto de vista de su implementación, los componentes ActiveX son objetos COM (Component Object Model) implementados en bibliotecas de enlace dinámico (DLL). Un objeto COM es un objeto con formato binario estándar definido por Microsoft, que permite que diferentes lenguajes de programación hagan uso de las propiedades, métodos y eventos del mismo [21].

A continuación se definen algunos conceptos importantes para la programación en Delphi, como lo son los objetos, componentes, propiedades, eventos y métodos.

Objetos: es parecido a un registro, pero se caracteriza porque además de contener miembros de distintos tipos (como números, cadenas, punteros, etc.), también es capaz de contener definiciones de procedimientos y funciones. [21]

Componentes: es cualquier elemento que se puede insertar en una ficha, aún si su función es visual o no, cabe mencionar que un componente es un objeto, la ventaja de estos es que el usuario puede manipular sus propiedades aunque no conozca el funcionamiento de este en un 100%.

Propiedades: por medio de éstas se pueden personalizar los componentes y todos los objetos con los que cuenta Delphi, con el fin de que cumplan las funciones deseadas. Se pueden manejar como si fueran las variables de los objetos ya que para tener acceso a estas generalmente se tiene que indicar su nombre y el objeto al que pertenecen.

Eventos: son señales que el entorno recibe desde distintos elementos, como puede ser el ratón, el teclado o un temporizador, estos son redirigidos a las aplicaciones, que en caso de aceptarlos deberán responder a la acción indicada.

Métodos: son procedimientos ó funciones que permiten realizar una determinada acción en el componente.

4.3.1 Entorno de programación

Delphi cuenta con un entorno de programación sencillo y amigable, esta formado por: VCL (Visual Components Library), formularios, inspector de objetos, editor y explorador de código.

VCL (Visual Components Library) (figura 4.5): es un conjunto de componentes básicos que brindan una serie de objetos y da la posibilidad de utilizarlos sin necesidad de llamar a las distintas API's de Windows para que dibujen en pantalla el componente, de igual manera se puede manipular las propiedades de dichos componentes.

Figura 4.5 Librería Visual de Componentes

Formularios (figura 4.6): es el contenedor de los componentes el cual conforma la interfaz de la aplicación.

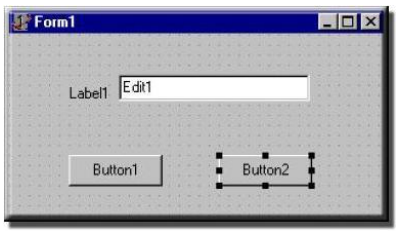


Figura 4.6 Formularios

Inspector de objetos (figura 4.7): permite modificar de forma sencilla, rápida y visual las propiedades de los objetos con los que cuenta el formulario, de igual forma permite controlar los eventos.

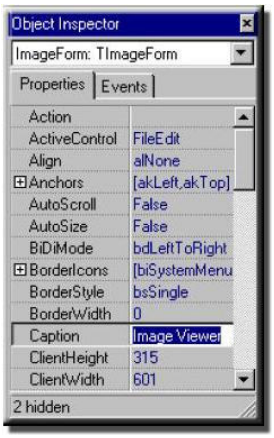


Figura 4.7 Inspector de objetos

Editor de código: es un editor de textos en el cual se escribe el código del programa.

Explorador de código (figura 4.8): se encuentra en la parte izquierda del editor, muestra en forma de árbol todos los elementos asociados al módulo (clases, propiedades, variables, etc.)

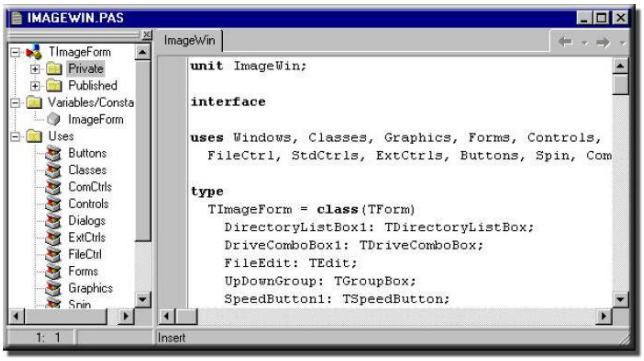


Figura 4.8 Editor y explorador de código

4.3.2 Tipos de archivo en Delphi

A continuación se mencionan algunos de los tipos de archivo que son generados por Delphi cuando se crea un proyecto. Los principales son .PAS, .DFM, .DPR, .EXE, .DCU, .RES, entre otros.

- *.PAS (Delphi Source File):* en Delphi, los archivos .Pas contienen el código fuente para una unidad o un formulario. Los archivos fuente de la unidad contienen la mayor parte del código de una aplicación. La unidad contiene el código fuente para cualquier manejador de eventos relacionados a los eventos del formulario o a los componentes que contiene éste. Un archivo .Pas se puede visualizar mediante un editor de texto.

- ***.DCU (Delphi Compiled Unit):*** es el archivo de una unidad compilada (.pas). Por default la versión compilada de cada unidad es almacenada por separado en un archivo en formato binario con el mismo nombre que el archivo de la unidad, pero con extensión .dcu.

- *.DFM (Delphi Form):* estos archivos tienen relación con los archivos .pas. El archivo .DFM contiene los detalles (propiedades) de los objetos que están en un formulario. Delphi copia la información del archivo .DFM al código del archivo .EXE.

- *.DPR (Delphi Project):* el archivo .DPR es el archivo central para un proyecto en Delphi, de hecho un archivo fuente Pascal. Sirve como el primer punto de entrada para el ejecutable. El .DPR contiene referencia hacia otros archivos en el proyecto y ligas a formularios con sus unidades asociadas.

- ***.RES (Windows Resource File):*** un archivo fuente de Windows, generado automáticamente por Delphi y necesario para el proceso de compilación. Este archivo en formato binario contiene la versión y el icono principal de la aplicación. El archivo también puede contener otros recursos utilizados por la aplicación.

- *.EXE (Application Executable):* es un archivo en formato binario con el archivo ejecutable del proyecto, este se crea utilizando todos los archivos .DCU del proyecto.

- *. ~ (Delphi Backup Files):* los archivos que su nombre termina con .~ son respaldos de los archivos que han sido guardados.

- *.DPK (Delphi Package):* este archivo contiene el código fuente de un paquete, que puede ser utilizado en Delphi. El archivo fuente de un paquete es similar a un archivo de proyecto, pero los .DPK son utilizados para construir librerías de enlace dinámico (DLL) para llamar un paquete.

- *.DOF:* este archivo de texto contiene las situaciones actuales para las opciones del proyecto, tales como el compilador y ligas a las situaciones, directorios, directivas condicionales y parámetros de la línea de comandos. [41]

4.4 Lenguaje Java

Java es una plataforma de software desarrollada por Sun Microsystems a finales de los años ochenta, originalmente llamado OAK por los ingenieros de Sun, Java fue diseñado para correr en computadoras incrustadas principalmente en electrodomésticos. Sin embargo, en 1995, dada la atención que estaba produciendo la web, fue distribuido para sistemas operativos tales como Microsoft Windows. Java llegó a ser extremadamente popular cuando Sun Microsystems introdujo la especificación J2EE (Java 2 Enterprise Edition). Este modelo permite, entre otros, una separación entre la presentación de los datos al usuario (JSP o Applets), el modelo de datos (EJB), y el control (Servlets). [50]

4.4.1 Características

- *Simple:* es un lenguaje sencillo de aprender, debido a que es más simple que otros lenguajes de programación (como C o C++) ya que elimina encabezados de archivos, manejo de apuntadores, herencia múltiple, sobrecarga de operadores, etc.[30]

- *Orientado a objetos:* es totalmente orientado a objetos, para ser un lenguaje orientado a objetos, debe cumplir con las siguientes características.

  - ✓ *Herencia:* es el concepto que define la adopción de todas las características de una clase por parte de otra clase que es definida como descendiente de la

primera. Java permite el empleo de la herencia, característica que permite definir una clase tomando como base a otra clase ya existente. Esto es una de las bases de la reutilización de código. En Java, la herencia se especifica agregando la cláusula extends después del nombre de la clase. Al heredar de una clase base, se heredan tanto los atributos como los métodos, mientras que los constructores son utilizados, pero no heredados.

- ✓ *Polimorfismo:* la palabra polimorfismo significa múltiples formas y es la propiedad que tiene un lenguaje de programación de que una clase, pueda tener diferentes comportamientos. [56]

- ✓ *Encapsulación:* se define como el proceso de empaquetar los métodos y los datos en un objeto. El objeto se encarga de ocultar sus datos al resto de los objetos. La encapsulación permite mayor seguridad en el acceso a los datos ya que este acceso depende directamente de cada objeto. [52]

- *Distribuido:* proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets, establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas. [60]

- *Interpretado y compilado:* es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador. Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se haya portado el intérprete y el sistema de ejecución en tiempo real (run-time). Siendo la máquina virtual la que interpreta los bytecodes.

- *Robusto:* libera al programador del compromiso de tener que controlar especialmente la asignación de memoria a las necesidades específicas del programa. Este lenguaje posee una gestión avanzada de memoria llamada gestión de basura, y un manejo de excepciones orientado a objetos integrados. Sus características de memoria evitan el manejo de apuntadores, ya que se ha prescindido por completo de estos, y la recolección de basura elimina la necesidad de liberación explícita de memoria. [60]

- *Seguro:* debido a la naturaleza distribuida de Java, la seguridad es de gran importancia para el lenguaje, entre las medidas de seguridad se incluye: restricciones en los applets, implantación redefinible de sockets y objetos de administración de seguridad definidos por el usuario.

- *Arquitectura neutral:* está diseñado para que un programa escrito en este lenguaje sea ejecutado correctamente independientemente de la plataforma en la que se esté actuando (Macintosh, PC, UNIX, etc.). Para conseguir esto utiliza una compilación en una representación intermedia que recibe el nombre de bytecodes, que pueden interpretarse en cualquier sistema operativo con un intérprete de Java. [35]

- *Portable:* los programas se compilan en el código de bytes de arquitectura neutral y se ejecutarán en cualquier plataforma con un intérprete de Java.

- *Multithreading:* proporciona múltiples flujos de control que se ejecutan de manera concurrente dentro de uno de sus programas. Los hilos permiten que un programa emprenda varias tareas de cómputo al mismo tiempo, característica que da soporte para trabajar en red o en animación. [30]

- *Dinámico:* el lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la red. [35]

4.4.2 JVM (Java Virtual Machine)

La máquina virtual de Java es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (bytecodes de Java), el cual es generado por el compilador del lenguaje Java. La gran ventaja de la JVM es aportar portabilidad al lenguaje de manera que Sun ha creado máquinas virtuales para diferentes arquitecturas y así un programa .class puede ser interpretado por cualquier plataforma. En la figura 4.9 se muestra como es interpretado el codigo fuente de Java por la JVM.

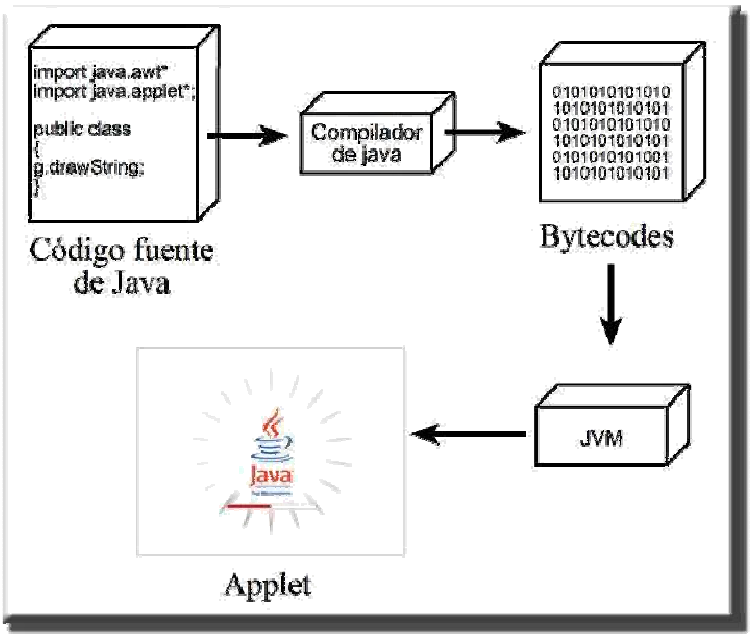


Figura 4.9 La JVM interpreta los bytecodes generados al compilar el programa

4.4.3 JNI (Java Native Interface)

El JNI es un mecanismo desarrollado por Sun Microsystems que permite ejecutar código nativo desde Java y viceversa.

El código nativo, son funciones escritas en un lenguaje de programación como C o C++ para el sistema operativo donde se ejecuta la maquina virtual.

JNI tiene una interfaz bidireccional que permite a las aplicaciones Java llamar a código nativo y viceversa. Es decir JNI soporta dos tipos de interfaces:

- Métodos nativos: que permite llamar funciones implementadas en librerías nativas.
- Invocación de interfaz: que permite incrustar una maquina virtual de Java en una aplicación nativa.

4.4.4 Applets

El código de Java puede proporcionar contenido ejecutable en documentos Web, esto se logra incluyendo applets en dichos documentos. Los applets de Java agregan contenido ejecutable en páginas Web. Un applet es un programa de Java que puede recuperarse con un explorador Web y ejecutarse en la computadora local al mismo tiempo. [30]

Los applets pueden mejorar la calidad de una página, con imágenes en movimiento, efectos de sonido e interacción con usuarios que ofrecen respuestas

4.5 MATLAB

MATLAB es el nombre abreviado de “MATrix LABoratory”. Es un programa que realiza cálculos numéricos con vectores y matrices. Es un lenguaje de alto desempeño para el cálculo técnico. Este integra cálculo, visualización y programación en un ambiente fácil de usar donde los programas y soluciones son expresados en una notación matemática amigable. Los temas incluidos en esta plataforma son matemáticas y cálculo, desarrollo de algoritmos, adquisición de datos, modelado, simulación de prototipos, análisis de datos, exploración y visualización, además de ofrecer gráficas científicas y técnicas así como el desarrollo de aplicaciones, incluyendo la construcción de la interfaz gráfica de usuario. Una de sus capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones.

El entorno de MATLAB está constituido por varias ventanas, como lo son:

- *Comand Window:* es la ventana sobre la que se trabaja y en la que se introducirán todos los comandos.

- *Launch Pad:* vinculos a apartados del programa.

- *WorkSpace:* información sobre los elementos que se han creado.

- *Comand History:* contiene todos los comandos introducidos.

- *Current Directory:* árbol que contiene todos los ficheros con la extensión .m que se encuentran en la carpeta. [3]

Con las características anteriormente mencionadas se argumenta que MATLAB es un sistema interactivo en el cual el elemento básico de datos es un arreglo que no requiere dimensionarse. Lo cual permite resolver muchos problemas de cálculo técnico, especialmente aquellos que contengan formulaciones con matrices y vectores. Tomando en cuenta la gran utilidad del ambiente matemático en este software, se decide utilizar para la elaboración de diversos programas de simulación digital con el propósito general de validar experimentalmente el MCDP de los dedos de la mano.

4.6 VrmlPad

Como se mencionó anteriormente, VRML es un lenguaje de modelado mediante el cual se puede describir la apariencia de una escena en tres dimensiones, de igual manera permite crear mundos virtuales generados en 3D. Para diseñar un mundo virtual, únicamente se necesita un editor de textos, en el cual se creará el archivo wrl correspondiente al mundo. Sin embargo, existen editores profesionales donde se puede crear los mundos virtuales deseados, en este trabajo se utilizó el denominado VrmlPad. [69]

VrmlPad proporciona las siguientes características:

- *Auto completar inteligente:* cuenta con un comando para completar el código, despliega una lista dentro del editor de texto, la cual contiene los identificadores VRML de acuerdo al texto que se esta escribiendo. Los identificadores VRML incluyen palabras claves como PROTO, nombres de campos y nodos, tipos de campos, propiedades, métodos, etc.

- *Detección de errores:* permite detectar errores en la escritura del código, debido a que cuenta con un rango de errores semánticos y advertencias, identificadores indefinidos, definición de nodos duplicados, entre otras.

- *Resalta la sintaxis:* para facilitar el desarrollo del código VRML, resalta la sintaxis de este en diferentes colores, con el fin de diferenciar entre palabras reservadas, campos, nodos, etc.

- *Árbol de la escena:* cuenta con un árbol de escena, en la parte izquierda del editor, en el que se presentan todos los nodos utilizados, así como sus características.

- *Operaciones sobre recursos:* ésta opción permite realizar operaciones (renombrar, ver, navegar) sobre los recursos que utilizan en el ambiente virtual, dichos recursos pueden ser imágenes o archivos wrl.

- *Mapa de rutas:* muestra las rutas que se han asignado en la escena, permite ver la conexión que existe entre eventos y exposed field. De igual manera brinda la oportunidad de agregar nuevas rutas a la escena.

- *Depuración de script:* con esta herramienta se pueden encontrar y corregir errores en el código del script de la escena y en páginas HTML con controles de Cortona embebidos.

- *Automatización:* permite a los diseñadores manipular el programa desde aplicaciones externas usando C, C++, Visul Basic o Delphi.

- *Documentos múltiples:* en el editor VrmlPad se pueden abrir varios documentos en la misma interfaz del editor, cambiar fácilmente entre ellos y modificar cualquiera de los documentos que se encuentran abiertos.

- *Vista previa de la escena:* cuenta con una opción mediante la cual se puede ver la escena antes de terminar el código por completo, con el fin de poder corregir problemas con el diseño.

- *Publicar:* organiza y optimiza la escena con dependencias para publicarla en la red. Ubica los archivos en un servidor web o lo envía por correo electrónico. [70]

4.7 Cortona VRML Client 4.2

Cortona VRML Client es un visualizador web 3D, rápido y altamente interactivo, el cual es ideal para ver modelos en 3D en la web. Cortona es compatible con diversas tecnologías para el desarrollo 3D y sobre todo con VRML. Trabaja como un plug-in para los navegadores de Internet (Internet Explorer, Netscpe Navigator, Mozilla, etc.) y aplicaciones de Office (Power Point, Word, etc.) [39]

Los movimientos en un espacio tridimensional utilizando este visualizador son similares al movimiento de una cámara. Cuenta con una serie de botones por medio de los cuales se puede interactuar con el ambiente virtual. La manera en que se manipula la cámara, simula a una persona que está viendo y actuando recíprocamente con el escenario. De igual manera el diseñador del escenario puede asignar las cámaras que sean necesarias para darle mayor realismo a su ambiente, pero cabe mencionar que solo se puede activar un punto de vista a la vez. Las formas de navegación permitidas son walk, fly y study. [40]

La ventana de navegación de Cortona esta formada por (figura 4.10):

- *Barra de herramientas vertical:* la cual cuenta con los botones que especifican el modo de navegación en el mundo.

- *Barra de herramientas horizontal:* cuenta con un conjunto de botones para cambiar la posición en el mundo.

- *La ventana 3D:* muestra el ambiente virtual.

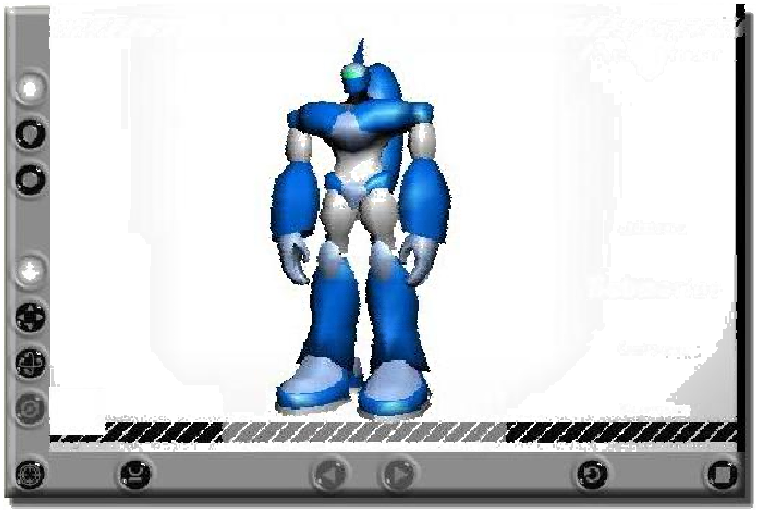


Figura 4.10 Ventana de navegación de Cortona

4.7.1 Características principales

- Soporte completo para VRML, incluyendo Java y Javascript.
- Soporte de modernos aceleradores 3D mediante Direct X y OpenGL.
- Render avanzado.
- Render en modo full screen con soporte para gafas 3D.
- Optimización para procesadores Intel Pentium III y Pentium 4.
- Instalación automática para Internet Explorer y Netscape Navigator.
- Nodos y capacidades adicionales que extienden las especificaciones VRML.
- Soporte para Macromedia Flash.
- Automatización de la interfaz VRML basado en la tecnología de automatización ActiveX
- Soporte para External Authoring Interface (EAI) e Internet Explorer.

4.8 JCreator

JCreator es un IDE (Integrated Development Environment) para la tecnología de Java, el cual permite desarrollar programas sobre este lenguaje de programación de una forma sencilla. JCreator es una herramienta escrita completamente en lenguaje nativo de Windows, de igual manera brinda las siguientes características:

- La interfaz permite el manejo de los proyectos de forma simple.
- Permite utilizar diferentes perfiles del JDK.
- Con la plantilla de proyectos, la escritura del código se hace de forma más rápida.
- Por medio del navegador de clases se puede ir a cualquier clase sin tener que buscarla línea por línea.
- Depurar errores con una interfaz fácil e intuitiva.
- No es necesario trabajar desde el prompt de MS-DOS.
- Los asistentes facilitan el desarrollo de algunos proyectos.
- Los requerimientos del sistema de JCreator son más bajos en comparación a otros IDE de Java.

JCreator cuenta con dos tipos de herramientas los cuales pueden ser configurados por el usuario:

- El primero es el Java Development Kit (JDK) el cual se puede utilizar para compilar, depurar errores y correr un proyecto.
- La segunda herramienta es más general y permite extender las capacidades de JCreator de acuerdo a las necesidades del programador, permitiendo llamar funciones y utilidades externas.

Capítulo 5

Metodología y desarrollo del sistema

Se describe la metodología utilizada en este trabajo de investigación, así como, las tareas realizadas para obtener el MCDP, el modelado de la mano virtual, el desarrollo de los programas generado para la obtención de los datos del guante de datos P5, y la interfaz que sirve de comunicación entre el guante y el modelo virtual.

5.1 Metodología

Para hacer un trabajo de investigación, es preciso seguir determinados procedimientos que permitan alcanzar el fin propuesto, no es posible obtener un conocimiento racional, sistemático y organizado si se actua de cualquier modo, es necesario seguir algún camino concreto que lleve a la meta establecida. A éstos caminos y procedimientos a seguir se pueden definir como metodología de investigación.

La metodología debe reflejar la estructura lógica del proceso de investigación, desde la elección de un enfoque metodológico específico, hasta la forma como se van a analizar, interpretar y presentar los resultados.

Deben detallarse, los procedimientos, técnicas, actividades y demás estrategias metodológicas requeridas para la investigación. Debe indicarse el proceso a seguir en la recolección de la información, así como en la organización, sistematización y análisis de los datos.

A continuación se define que es una metodolgía:

Metodología se compone de los términos *método* y *logos*, que significa explicación, juicio, tratado, estudio de los métodos, es decir representa la manera de organizar el proceso de la investigación, de controlar sus resultados y de presentar posibles soluciones a un problema que conlleva la toma de decisiones.

Método, de raíces griegas *meta* que significa con, y *odos*, camino; esto es, manera de proceder para descubrir algo o alcanzar un fin. El método representa la manera de conducir el pensamiento o las acciones para alcanzar un fin.

En base a la ingeniería de software, una metodología se define de la siguiente forma:

Una metodología es el conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a desarrollar nuevas aplicaciones informáticas. [54]

Normalmente la metodología constituirá un conjunto de fases divididas en subfases (módulos, etapas, pasos, etc.) Esta descomposición del proceso de desarrollo guía a los desarrolladores en la elección de las técnicas que se deben elegir para cada estado del proyecto, además de facilitar la planificación, gestión, control y evaluación de los proyectos.

5.1.1 Metodología empleada

Para la realización del presente trabajo de investigación no se asigno una metodología en específico, sin embargo, se siguieron una serie de pasos que permitieran lograr los objetivos establecidos para este trabajo, dichos pasos se muestran en la figura 5.1:

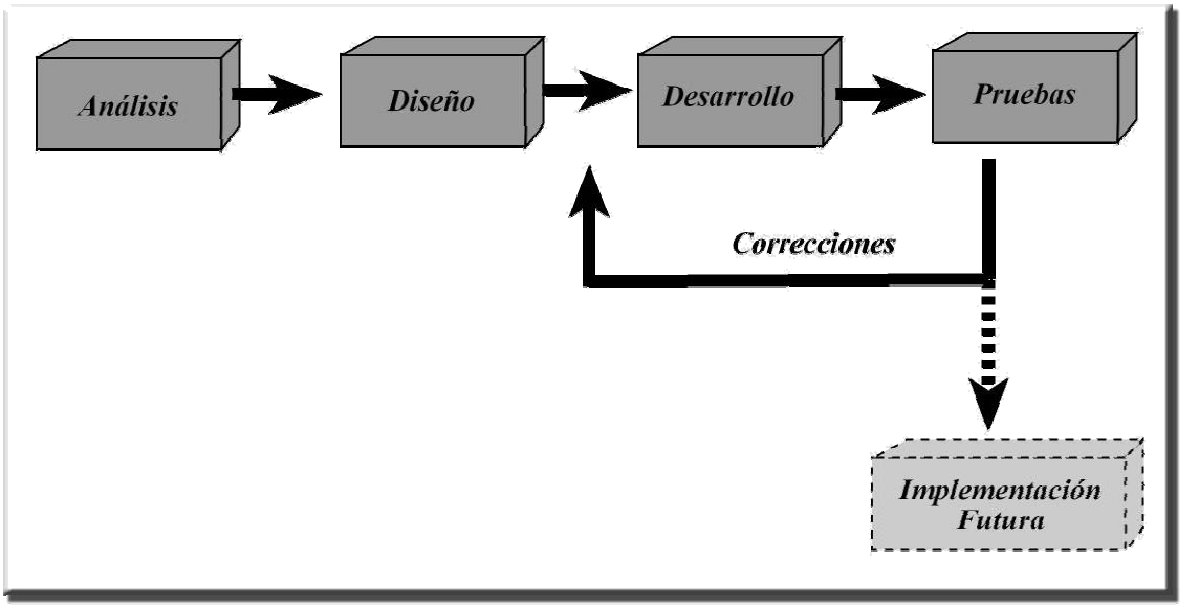


Figura 5.1 Diagrama representativo de la metodología empleada

- *Análisis:* en esta etapa es indispensable conocer los requerimientos del sistema con el fin de tener una idea clara acerca de lo que se va a desarrollar, de igual manera organizar toda la información recopilada, para buscar la solución correcta de acuerdo a los requerimientos del proyecto. Entre las tareas que se realizaron en esta etapa se encuentran:

  - ✓ Características del trabajo a realizar.
  - ✓ Conocer los trabajos existentes, que hacen uso del dispositivo y que herramientas fueron utilizadas para realizarlos.
  - ✓ Herramientas a utilizar para el desarrollo de la interfaz.
  - ✓ Anatomía de la mano humana.
  - ✓ Funcionamiento del guante P5.

  La figura 5.2 presenta cada una de las tareas que se llevaron a cabo en la etapa de análisis de esta investigación.

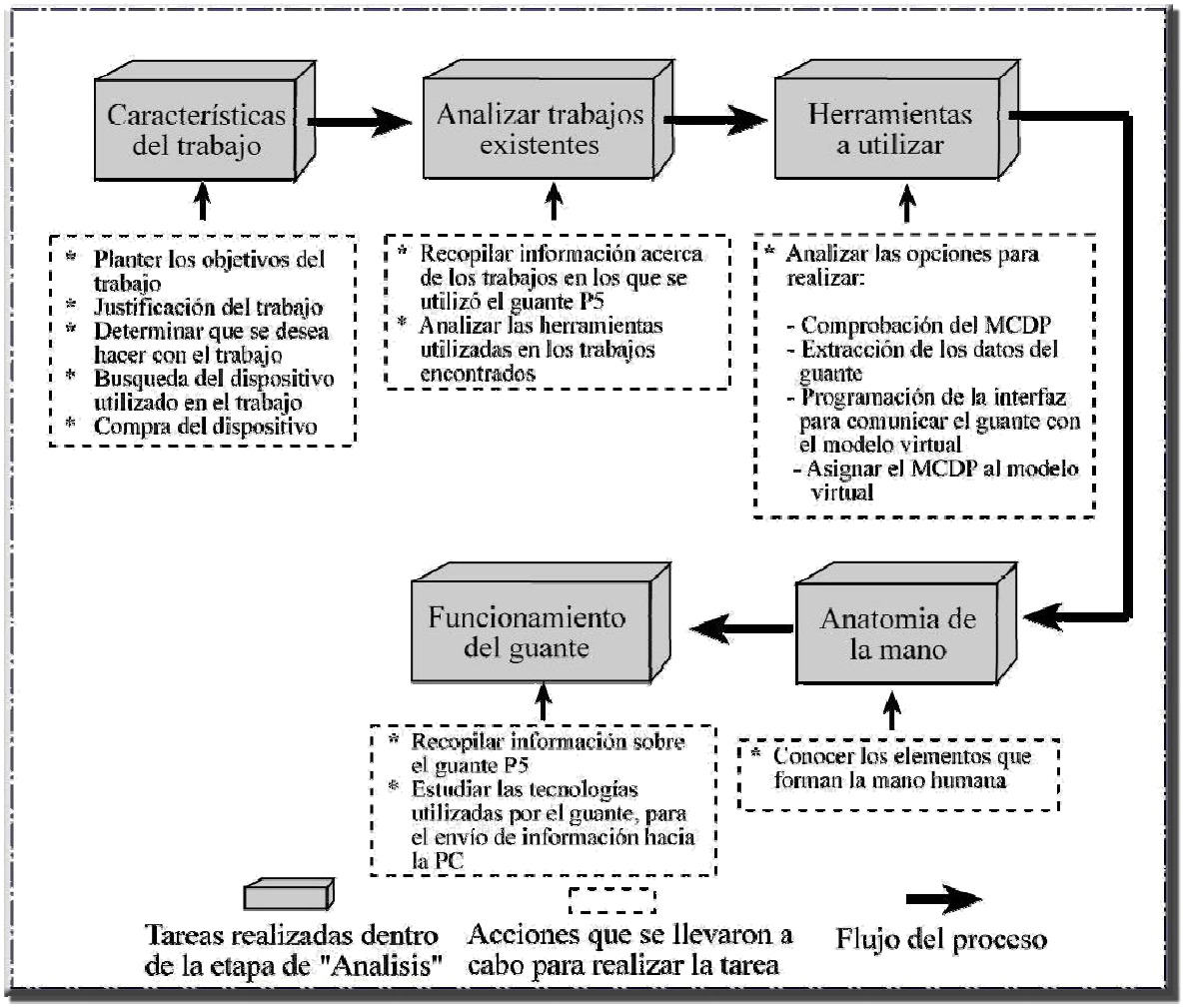


Figura 5.2 Tareas realizadas en la etapa de Análisis

El guante de datos P5, es un dispositivo que gracias a su diseño, tecnologías que emplea y al bajo costo que tiene, han sido factores para que sea utilizado en varios proyectos de investigación, en lo cuales se emplearon librerías como OpenGL y DirectX para la generación de graficos 3D que interactuen con el guante. Sin embargo, el objetivo de éste trabajo, es utilizar VRML como herramienta para realizar el modelo virtual.

Para lograr la comunicación del guante con el modelo virtual, se analizaron dos líneas de investigación:

- ✓ El desarrollo de un Applet utilizando EAI para enviar la información al modelo virtual, por otro lado, se uso la tecnología JNI de Java, para obtener los datos del guante. Sin embargo, no se logró comunicar el guante con el modelo virtual, debido a problemas presentados por las maquinas virtuales, ya que para ejecutar el applet, es necesaria la maquina virtual de Java (JVM) y para cargar el modelo virtual

embebido en una página HTML se necesita la maquina virtual de Microsoft, ya que si se desea ejecutar con la JVM no muestra la escena virtual.

- ✓ Desarrollando la interfaz de comunicación en Delphi para enviar los datos del guante al modelo virtual.

La figura 5.3 muestra las tecnologías encontradas en algunos de los trabajos realizados actualmente, mientras que en la parte sombreada se encuentran las tecnologías que se estudiaron en este proyecto.

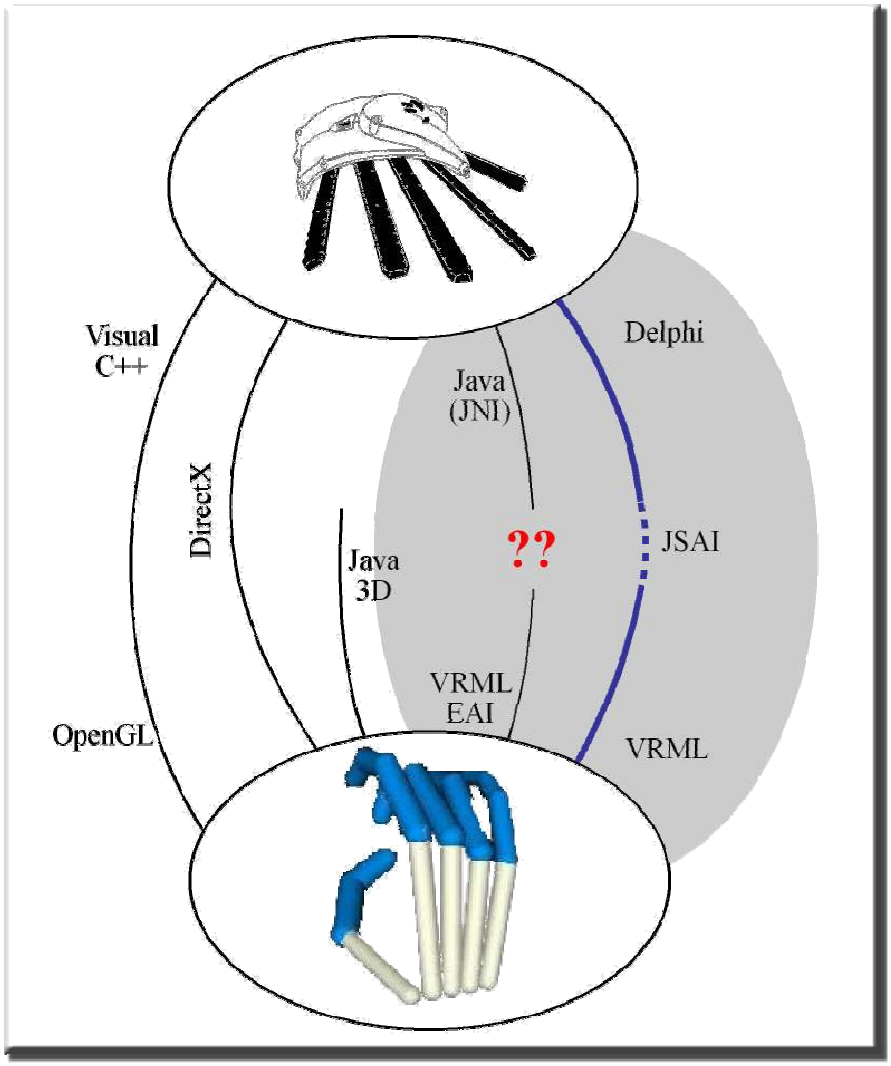


Figura 5.3 Tecnologías analizadas durante el desarrollo de éste trabajo

- *Diseño:* en esta etapa se llevo a cabo el diseño del modelo correspondiente a la mano virtual. Algunas de las tareas realizadas en esta etapa son las siguientes:

- ✓ Medición de los eslabones que serán utilizados en el modelo tridimensional y en el modelo matemático.
- ✓ Obtención del Modelo Cinemático Directo de Posición (MCDP).
- ✓ Evaluación del MCDP en MATLAB.
- ✓ Modelado del esqueleto de la mano humana.

La figura 5.4 muestra de manera general las tareas realizadas en la etapa de diseño.

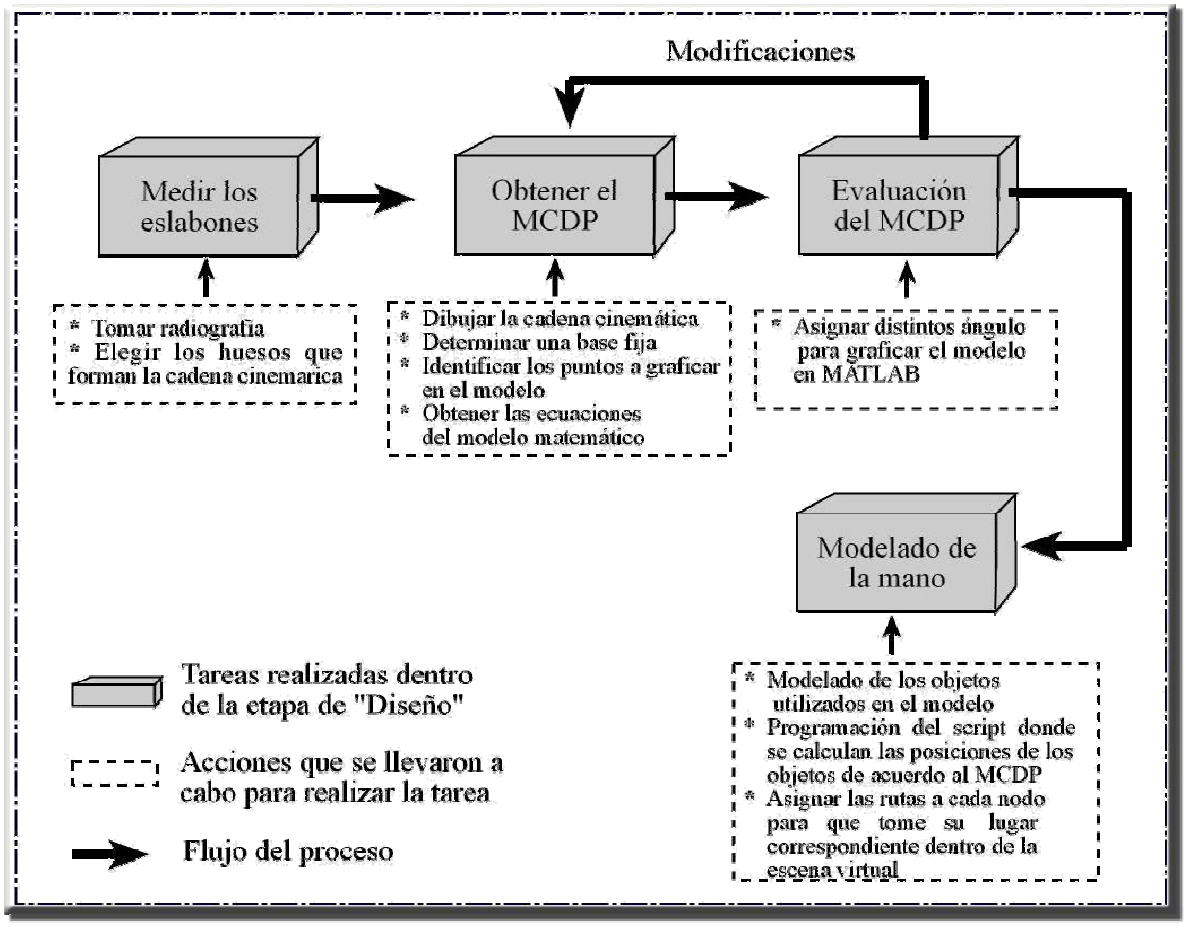


Figura 5.4 Tareas realizadas en la etapa de Diseño

- *Desarrollo:* una vez diseñado el modelo virtual utilizado en este trabajo, se procedio a desarrollar la interfaz para la comunicación del dispositivo y el modelo virtual. Entre las tareas que se realizaron en esta etapa se encuentran:
  - ✓ Elaboración de la interfaz EAI para manipular la mano virtual mediante un applet en una página web.

- ✓ Desarrollo de programas en Delphi para analizar las señales que serán enviadas al mundo tridimensional.

- ✓ Desarrollo de la interfaz, que se encargará de la comunicación entre el guante y el modelo virtual.

La figura 5.5 hace referencia a los pasos que se llevaron a cabo en la etapa de desarrollo.

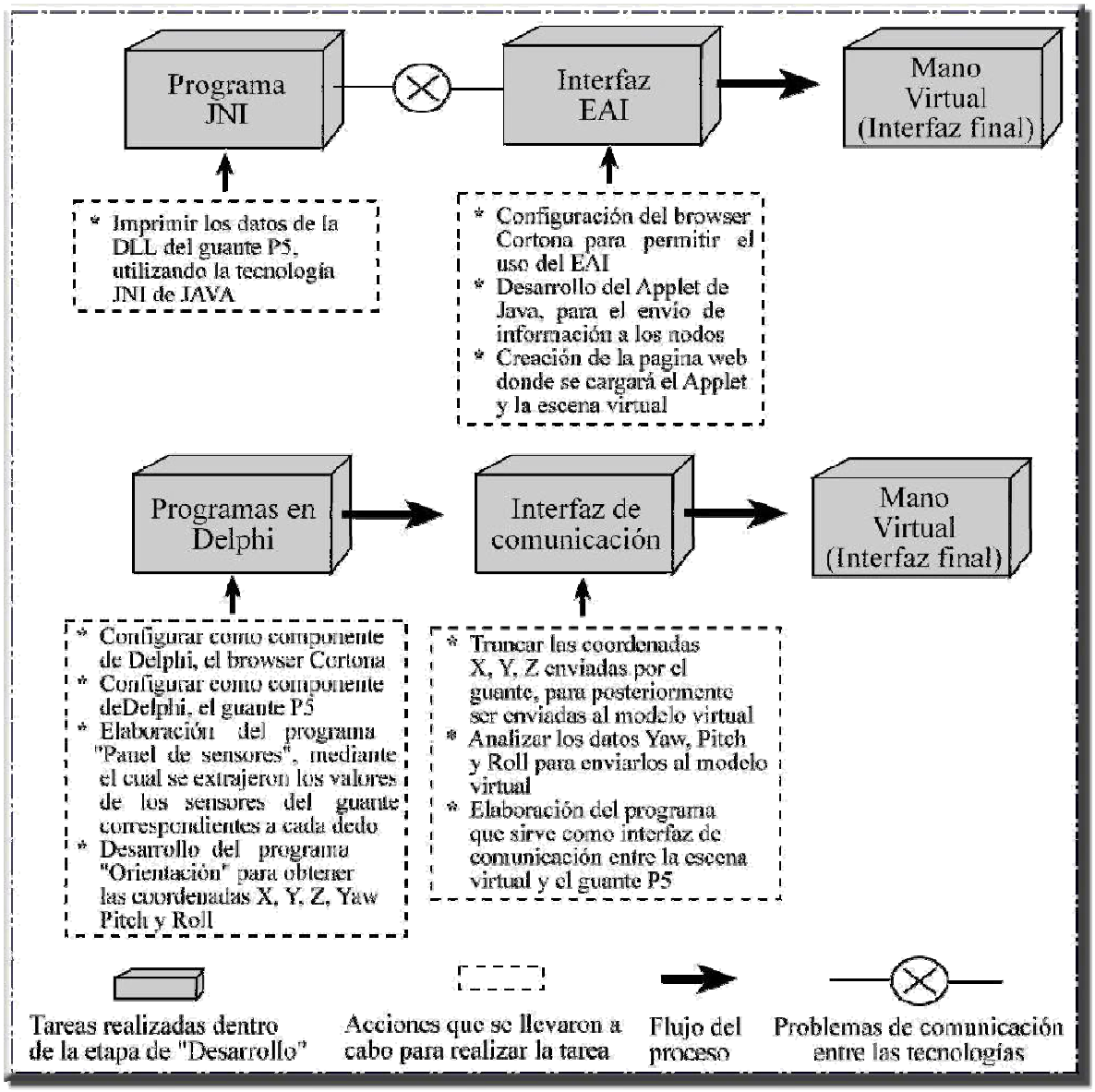


Figura 5.5 Tareas realizadas en la etapa de Desarrollo

- *Pruebas:* en esta etapa se realizar una serie de pruebas con el fin de corregir errores que se presentaron durante el desarrollo del sistema. La tabla 5.1 presenta las pruebas realizadas en este trabajo.

Pruebas realizadas	Problemas a resolver
En el programa de MATLAB, se asignaron distintos valores a los ángulos de cada eslabón.	Correcta graficación del MCDP. Verificar el funcionamiento del modelo.
En el modelo virtual, modificar los datos correspondientes a cada ángulo.	El objeto virtual tome un lugar que no le corresponde. Mala asignación de las rutas en VRML, es decir que los datos no sean enviados al nodo correcto (figura 5.6).
Imprimir todos los datos enviados por el guante.	Encontrar la información necesaria para cada uno de los datos que se manejan en el modelo virtual.
Asignación de las coordenadas X, Y, Z enviadas por el guante al modelo virtual.	Evitar movimientos repentinos del modelo virtual, debido a que el guante envía valores muy grandes.
Asignación de las coordenadas de rotación Yaw, Pitch y Roll.	Lograr movimientos lo menos brusco posibles, ya que el guante es muy inestable con estos valores.
Interacción del guante con el modelo virtual.	Asignar correctamente los datos de cada sensor al objeto representativo en el modelo virtual.

Tabla 5.1 Pruebas realizadas durante el desarrollo del trabajo

Figura 5.6 Problema generado por una mala asignación de rutas

- *Implementaciónes futuras:* En esta etapa se pretende obtener la interfaz final que permite la correcta comunicación entre el guante y el software, para manipular una práctica virtual con la ayuda del dispositivo y darle un grado de inmersión más grande a dicha práctica.

5.2 Analisis

5.2.1 Anatomía de la mano

La mano es la parte del cuerpo humano que se extiende desde la muñeca hasta el extremo de los dedos, es un elemento importante del cuerpo, por que mediante éste se realiza la mayor parte de las tareas que se llevan a cabo en la vida diaria de una persona.

5.2.1.1 Huesos que forman la mano

Los huesos de la muñeca, que en conjunto se llaman huesos del carpo; los metacarpianos en la porción de la palma y las falanges en los dedos (27 huesos en total), contribuyen con sus articulaciones a dar la precisión y movilidad característica de la mano del hombre, la mejor maquina jamás creada (figura 5.7) (tabla 5.2).

5.2.1.1.1 Carpo

Los huesos del carpo son pequeños, irregulares y están firmemente unidos entre si por ligamentos en un arco cóncavo hacia adelante; las caras posteriores son, en general, mayores que las anteriores; los huesos del carpo están dispuestos en dos hileras: una proximal que comprende de dentro hacia fuera y lo conforman el escafoides, semilunar, piramidal y pisiforme, y una hilera distal, constituida por trapecio, trapezoide, hueso grande y hueso ganchoso; todos ellos poseen varias articulaciones, excepto el pisiforme que se articula sólo con el piramidal. La concavidad de la cara palmar del carpo es aumentada por la proyección hacia adelante del pisiforme y el hueso ganchoso en la porción interna, y del escafoides y el trapecio hacia fuera. La situación anterior del escafoides y el trapecio es importante, pues permite que el primer metacarpiano disponga delante de los otros metacarpianos. El pisiforme se palpa fácilmente a la altura del pliegue distal de la muñeca, y también se mueve con facilidad cuando esta relajado el tendón del cubital anterior, que se inserta en él. [18]

5.2.1.1.2 Metacarpo

Se encuentra formado por cinco huesos largos o huesos metacarpianos que se designan como primero, segundo, etc. del dedo pulgar al dedo meñique respectivamente, y forman el

esqueleto de la palma y dorso de la mano. Se articula por arriba con la segunda hilera de huesos del carpo y por abajo con las primeras falanges de los dedos.

5.2.1.1.3 Falanges

Forman el esqueleto de los dedos, siendo en número de dos para el pulgar (falange y falangeta) y tres para los otros cuatro dedos (falange, falangina y falangeta). [10]

Figura 5.7 Huesos de la mano

Número	Hueso
1	Trapecio
2	Trapezoide
3	Escafoides
4	Ganchoso
5	Grande
6	Piramidal
7	Pisiforme
8	Semilunar

Tabla 5.2 Huesos que forman el carpo de la mano

5.2.2 Guante de datos P5

La compañía Essential Reality tardo cerca de tres años en desarrollar el guante P5 (figura 5.8), esta basado en la tecnología del antiguo Power Glove de Nintendo/Mattel, el cual se lanzó al mercado en el año de 1989 y vendió cerca de 1.6 millones de unidades. Originalmente el diseño comenzó con el guante completo, pero al final se optó por quitar la tela, permitiendo que manos de diferentes tamaños puedan trabajar con el mismo guante. [45] El guante de datos P5 salio al mercado el 28 de Octubre de 2002. [53]

Essential Reality compró la tecnología del Power Glove de Nintendo/Mattel, pero a ésta le agregaron rotación. [65]

Figura 5.8 Guante de datos P5 de Essential Reality

El P5 esta diseñado ergonómicamente, es un dispositivo que brinda al usuario interacción intuitiva con ambientes virtuales y en 3D (sitios web, juegos, software educativo, etc.). Las tecnologías que utiliza el guante son el *rastreo óptico (optical tracking)* y *bend sensor*, las cuales le proporcionan al sistema movimientos con 6 GDL (X, Y, Z, yaw, pitch y roll), brinda la capacidad de extender la mano, tocar y manipular objetos y personajes en espacios 3D, como se hace en la vida real. [58]

El P5 tiene tiras angostas que se colocan a lo largo de la parte superior de cada dedo, estas tiras se sujetan al del dedo por anillos de plastico, las tiras contienen la tecnología denominada *"bend sensor"* que envía los impulsos a la computadora, leyendo los movimientos de los dedos. El guante es sujetado a la mano por una correa elástica negra, con el fin de que se amolde al tamaño de cualquier mano. En un principio el guante estaba dirigido a niños y jóvenes que interactúan con video juegos, pero Essential Reality comenzó a tomar en cuenta todas las aplicaciones posibles, en animaciones, campamentos

militares, científicos y médicos. El guante se conecta a la computadora a través del puerto USB y funciona en sistemas operativos como Windows, Linux y Mac OS. [45]

El P5 esta cubierto por LEDs infrarrojos y cuenta con un receptor (torre) que puede leer las señales y los movimientos de la mano en tres dimensiones. [65]

5.2.2.1 Especificaciones del P5

5.2.2.1.1 Especificaciones de los sensores de los dedos

- 5 mediciones independientes para cada dedo.
- Grado de resolución de 0.5
- Índice de actualización de 60 *Hz*

5.2.2.1.2 Especificaciones del sistema de rastreo

- Sistema de rastreo óptico.
- Rango de 91.44*cm* desde el receptor.
- Índice de actualización de 45 *Hz.*
- 6 grados de libertad (*X, Y, Z, yaw, pitch* y *roll*).

5.2.2.1.3 Especificaciones X, Y, Z

- 0.3175*cm* de resolución a un rango de 91.44*cm* desde el receptor.
- 1.27*cm* de exactitud a un rango de 91.44*cm* desde el receptor.

5.2.2.1.4 Especificaciones yaw, pitch y roll

- 3 grados de resolución.
- 3 grados de exactitud.

5.2.2.1.5 Especificaciones del USB

- Compatible con USB 1.1
- Compatible con especificaciones HID.
- 2 interfaces USB proporcionadas, en modo nativo (P5) y en modo estándar. (Mouse). [14]

5.2.2.2 Tecnologías utilizadas por el Guante P5

El guante P5 utiliza dos tipos de tecnologías para enviar las señales al receptor (figura 5.9), el cual se encargará de enviarlas a la PC, estas tecnologías son: *rastreo óptico*, para determinar la posición de la mano y *bend sensor*, la cual se encarga de determinar el grado de flexión en cada dedo.

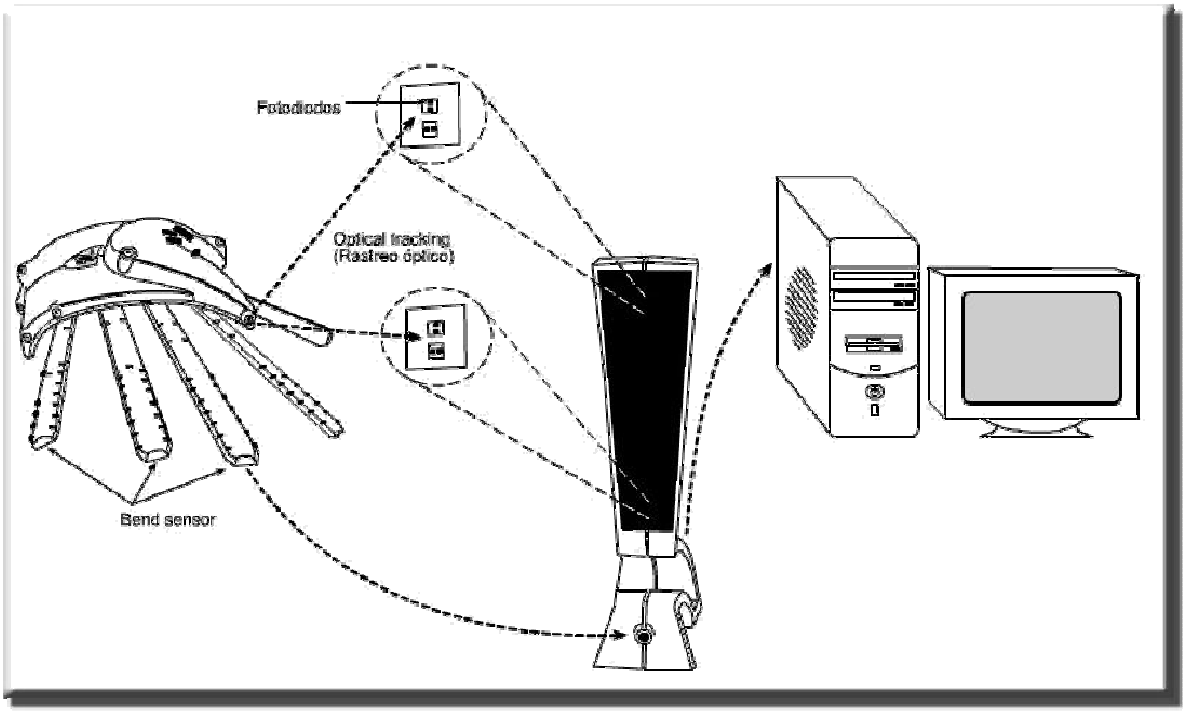


Figura 5.9 Tecnologías utilizadas por el guante para determinar la posición.

5.2.2.2. Rastreo óptico (Optical tracking)

Como ya se mencionó en el capítulo 3, la tecnología denominada rastreo óptico funciona con el envío de señales de luz que salen de una fuente hacia un receptor, el guante P5 cuenta con esta tecnología, por lo que a continuación se describirá más a detalle como funciona.

El guante cuenta con un conjunto de 8 LEDs en la parte superior de éste (figura 5.10), los cuales se utilizan como la fuente de las señales de luz que serán enviadas para determinar la posición en la cual se encuentra el guante, sin embargo, no todos los LEDs son visibles para el receptor, debido a que el grado de intensidad disminuye de acuerdo a la posición en la que se encuentra el guante, de esta manera solo se puede recibir la señal de 4 de ellos que en este caso son los que tengan mayor intensidad en su brillo.

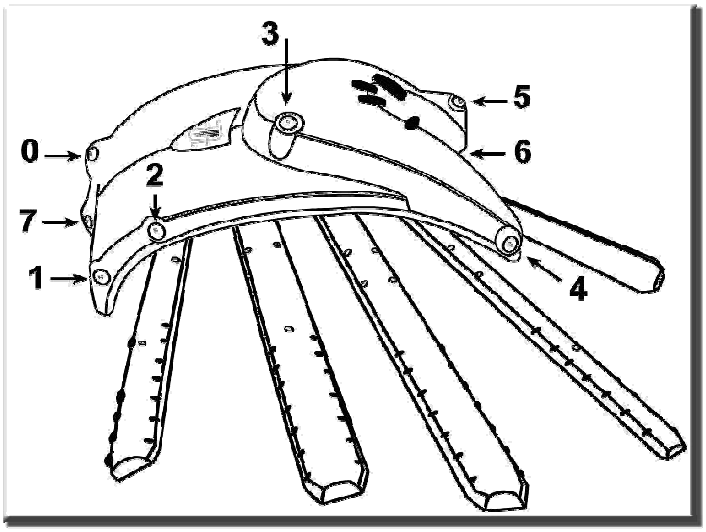


Figura 5.10 LEDs del guante P5

Para que el receptor pueda recibir la señal proveniente del guante, el receptor cuenta con dos paneles de sensores (figura 5.11), uno en la parte superior de la torre y otro en la parte inferior (figura 5.12), cada panel ésta compuesto por un conjunto de fotodiodos los cuales se encargan de detectar la posición *X* y *Y* (figura 5.13).

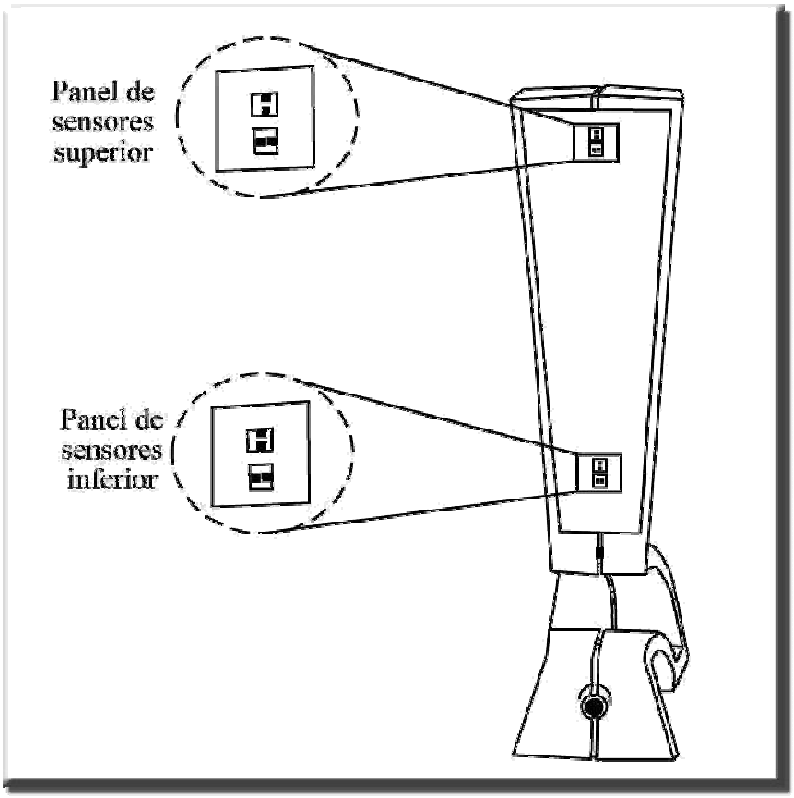


Figura 5.11 Paneles de sensores del receptor

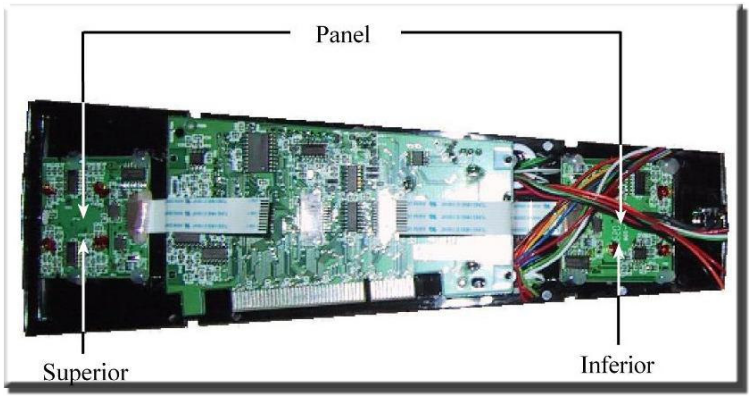


Figura 5.12 Paneles de sensores

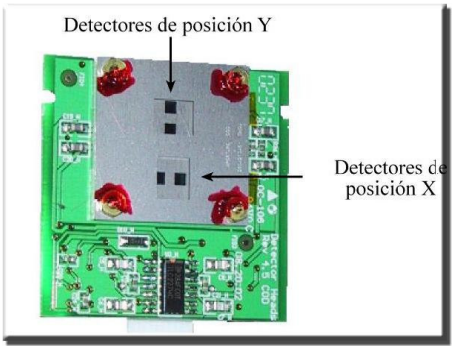


Figura 5.13 Fotodiodos detectores de posición X y Y

Cada panel de sensores se encuentra formado por 4 fotodiodos, estos miden la cantidad de luz que disminuye en cada uno. Los dos fotodiodos que se encuentran en la parte superior forman el detector Y mientras que los de la parte inferior forman el detector X.

A continuación se menciona un pequeño ejemplo para explicar como funcionan los fotodiodos en el dispositivo:

Cuando el guante se encuentra a la izquierda de la torre, el fotodiodo de la izquierda del detector X comienza a ser obscurecido parcialmente por la pared dejando menos luz en el sensor, mientras tanto el fotodiodo de la derecha se encenderá totalmente. Cuando el guante se encuentra a la derecha, el fotodiodo de la derecha se obscurecerá parcialmente y el fotodiodo de la izquierda se encenderá totalmente. Esto permite medir la posición horizontal de los LEDs. El detector Y trabaja de la misma forma para medir la posición vertical.

La cantidad de sombra que disminuye en los fotodiodos es proporcional a la posición horizontal y vertical de los LED. [15]

Los valores obtenidos, no se pueden utilizar directamente, debido a que el panel de sensores no está totalmente vertical por que la torre tiene una pequeña curvatura, con los valores que obtiene el receptor se encargará de obtener las posiciones *X, Y* y *Z.*

Los paneles de sensores se encuentran a las orillas del receptor, se encuentran un poco inclinados, por lo tanto a los valores verticales del panel superior se encuentra por debajo de 10° y los del panel inferior se encuentran sobre 17° (figura 5.14).

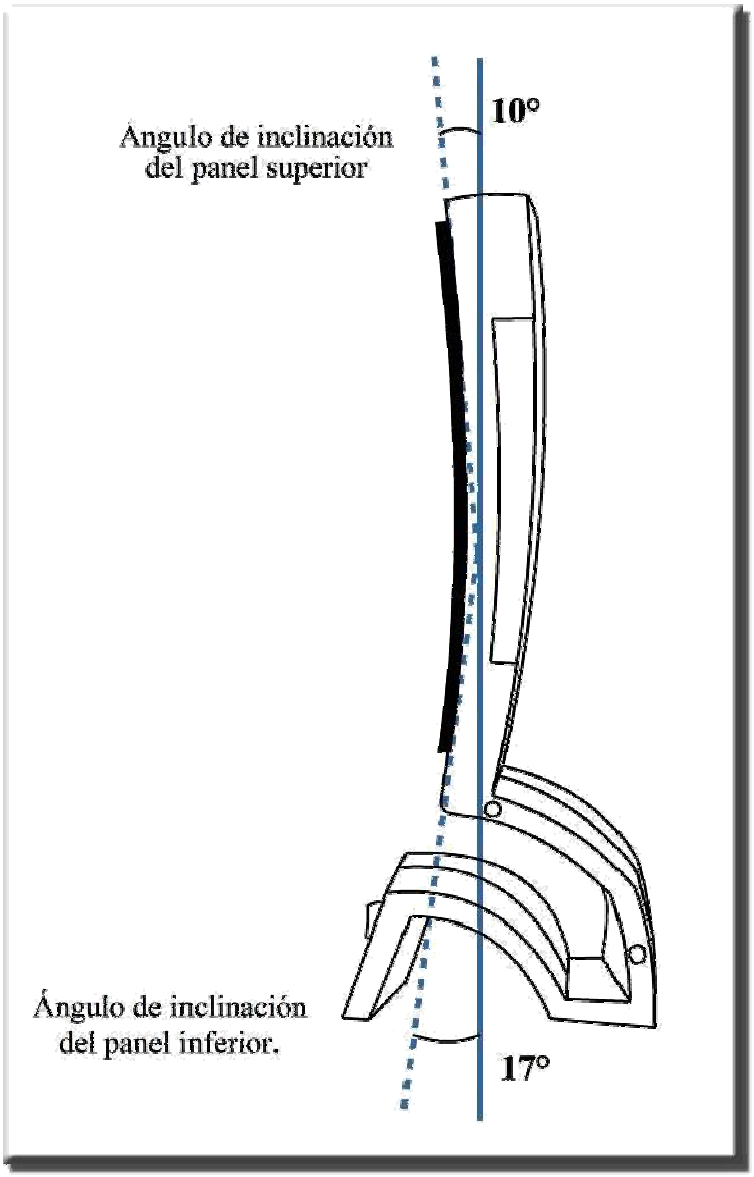


Figura 5.14 Inclinación del receptor

5.2.2.2.2 Bend Sensor

Los Bend Sensor son tiras largas de plástico (figura 5.15), los cuales varían su resistencia cuando se encuentran doblados. Originalmente fueron desarrollados por Nintendo Corp, para un dispositivo que sacaron a la venta denominado Power Glove.

El bend sensor es un producto que esta cubierto por un substrato de plástico que cambia la conductividad eléctrica cuando se encuentra doblado. Al sensor se pueden conectar sistemas electrónicos para medir a detalle la cantidad de doblez o movimientos que en éste ocurren. Un movimiento de una sola pulgada puede producir cerca de 200,000 datos. [6]

Figura 5.15 Bend sensor

El sensor mide 0.63cm (0.25pulgadas) de ancho, 11.43cm (4.5pulgadas) de largo y 0.48cm (0.19pulgadas) de grosor (figura 5.16).

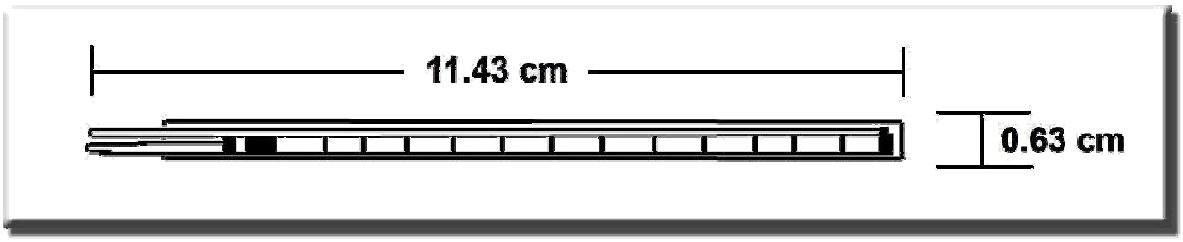


Figura 5.16 Medidas del bend sensor

La forma en que el dispositivo trabaja esta basada en su construcción. El sensor tiene una estructura de polímero de plástico delgado el cual se puede doblar fácilmente, sobre el polímero se encuentra una capa de carbono que moldea una resistencia, el carbono tiene unos cuadros pequeños de metal que se encuentra pegados al centro de la tira Cuando el dispositivo se encuentra doblado los espacios entre las placas de metal se incrementan y por lo tanto la resistencia del sensor se eleva. [43]

El bend sensor es un componente que cambia su resistencia cuando se dobla. Cuando el sensor no se encuentra doblado tiene una resistencia de 10, 000 Ω (10 KΩ). Cuando el bend sensor se encuentra doblado la resistencia incrementa gradualmente. Por ejemplo cuando se encuentra doblado a 90º la resistencia varía en un rango de 30 KΩ a 40 KΩ.

Algunas de las aplicaciones de los bend sensor son:

- Detección de colisiones en robots móviles.
- Guantes o trajes virtuales.
- Aplicaciones de física y experimentos.

- Control automovilístico.
- Dispositivos médicos.
- Controles industriales.
- Periféricos de computadora.
- Instrumentos musicales.
- Dispositivos de medición.
- Juegos de realidad virtual. [42] [66]

El guante P5 cuenta con 5 bend sensor (figura 5.17), los cuales se encuentran ubicados dentro de las tiras correspondientes a cada dedo. Como se mencionó anteriormente, los sensores envían datos de acuerdo al grado de flexión de éste, por lo tanto, en el guante se utilizan para conocer el grado de flexión que tienen los dedos. [15]

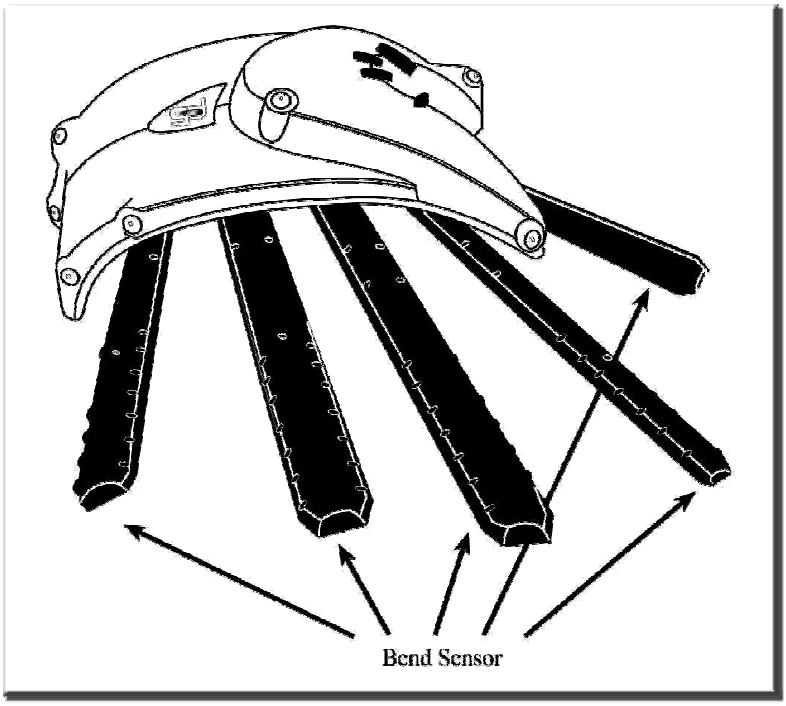


Figura 5.17 Bend sensor del guante P5

Los valores de los bend sensor en el guante toman una serie de valores, el valor 63 representa el valor máximo de flexión que tiene el dedo, mientras que el 0 representa el valor mínimo de flexión que tiene el dedo. Para el envío de los datos correspondientes a cada sensor del guante, lo realiza mediante el cable que se encuentra conectado del guante hacia el receptor, posteriormente éste se encarga de formar los paquetes de datos que serán enviados a la PC. [15]

5.3 Diseño

5.3.1 Medidas de los eslabones

Con el fin de diseñar un modelo más apegado a la realidad, se opto por tomar como base una mano que ajuste al tamaño del guante, y de ésta obtener las medidas correspondientes a cada hueso que la forman. Para tener una aproximación más exacta a estas medidas se tomó una radiografía de la mano que sirvió como base (figura 5.18).

Figura 5.18 Radiografía de la mano

Con la ayuda de la radiografía se obtuvieron las siguientes medidas, que posteriormente serán utilizadas en el modelo matemático (figura 5.19).

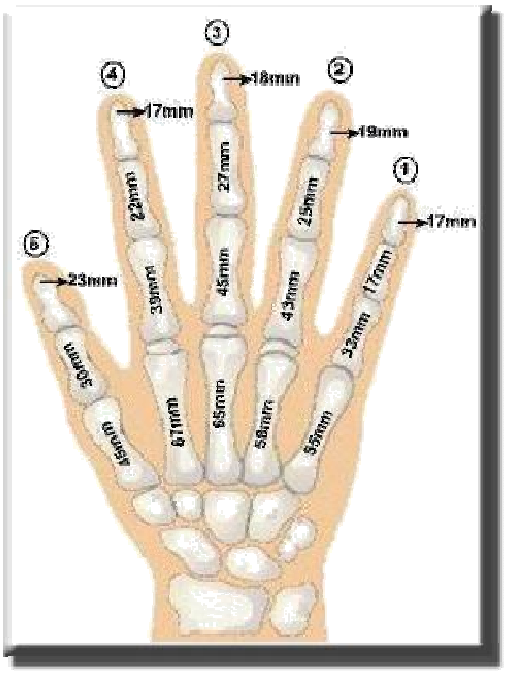


Figura 5.19 Longitud en milímetros de los huesos de la mano

5.3.2 Modelo Cinemático Directo de Posición (MCDP)

La cinemática es la ciencia del movimiento que trata a éste sin importarle las fuerzas que lo causan. Dentro de la cinemática se estudia la posición, la velocidad, aceleración y todas las derivadas de las variables de posición de mayor orden con respecto al tiempo o cualquier otra variable.

En la cinemática directa, los valores de las articulaciones son conocidos, el problema consiste en determinar la posición en el plano cartesiano del extremo de la cadena cinemática. Usando la cinemática directa, se puede rotar una articulación determinada cierto número de grados, la rotación resultante es automáticamente transmitida a la siguiente articulación de la cadena, actualizando su estructura.

Dado que un robot puede considerarse como una cadena cinemática formada por objetos rígidos o eslabones unidos entre sí mediante articulaciones, se puede establecer un sistema de referencia fijo situado en la base del robot y describir la localización de cada uno de los eslabones con respecto a dicho sistema de referencia. [36]

5.3.2.1 MCDP de los dedos

Como se mencionó anteriormente, con la ayuda del MCDP se puede obtener la posición y orientación del punto final de una cadena cinemática, para el modelo matemático usado en el modelo virtual, se utilizó como cadena cinemática la estructura ósea del dedo, localizando el punto final de la cadena en el extremo de la falangeta del dedo, tomando como base fija el metacarpiano, asignándole a este las coordenadas *(X=0, Y=0)*, cabe resaltar que la cadena se graficará en las coordenadas *X* y *Y*. La coordenada del efector final se encuentra en el punto *(X, Y)*. La figura 5.20 muestra la cadena cinemática de un dedo.

Para los dedos, meñique, anular, medio e índice se utilizó el mismo modelo, solo cambio la longitud de cada eslabón de la cadena, debido a que estas cuatro están formadas por la misma cantidad de eslabones (figura 5.21).

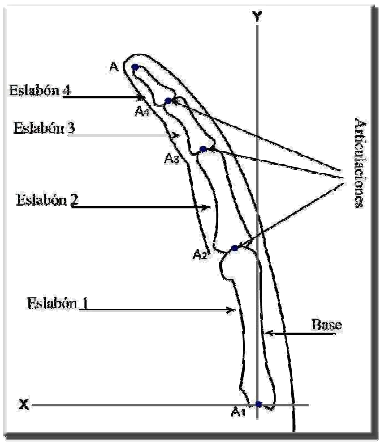


Figura 5.20 Cadena cinemática

La cadena cinemática que se utilizó en el modelo está compuesta por los siguientes puntos:

$$A_1 = (0,0)$$
$$A_2 = (X_2, Y_2)$$
$$A_3 = (X_3, Y_3)$$
$$A_4 = (X_4, Y_4)$$
$$A=(X, Y)$$

Donde:

- A_1 *= nodo inicial del metacarpiano de cada dedo.*
- A_2 *= nodo final del metacarpiano y nodo inicial de la falange.*
- A_3 *= nodo final de la falange y nodo inicial de la falangina.*
- A_4 *= nodo final de la falangina y nodo inicial de la falangeta.*
- *A = nodo final de la falangeta, este es el punto final de la cadena.*

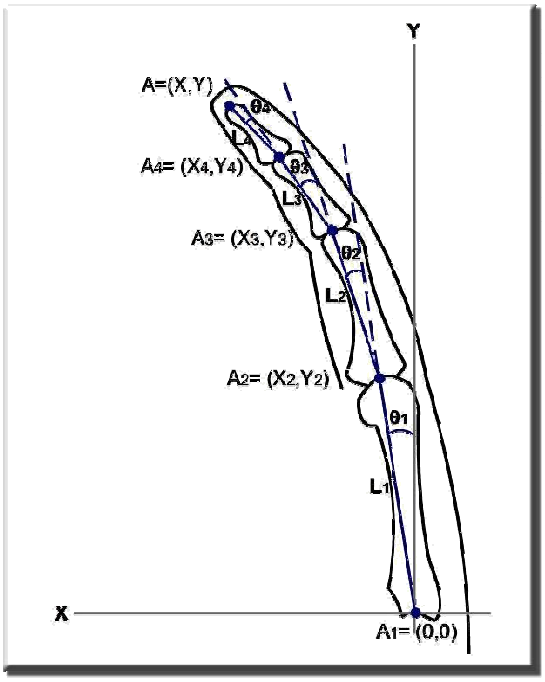


Figura 5.21 MCDP del dedo

Las ecuaciones correspondientes al nodo $A_2 = (X_2, Y_2)$ son las siguientes:

$$X_2=L_1C_1$$
$$Y_2=L_1S_1$$

Las ecuaciones correspondientes al nodo $A_3 = (X_3, Y_3)$ son las siguientes:

$$X_3=L_1C_1+ L_2C_{12}$$
$$Y_3= L_1S_1+ L_2S_{12}$$

Las ecuaciones correspondientes al nodo $A_4 = (X_4, Y_4)$ son las siguientes:

$$X_4 = L_1C_1 + L_2C_{12} + L_3C_{123}$$
$$Y_4 = L_1S_1 + L_2S_{12} + L_3S_{123}$$

Las ecuaciones correspondientes al nodo A = (X, Y) son las siguientes:

$$X = L_1C_1 + L_2C_{12} + L_3C_{123} + L_4C_{1234}$$
$$Y = L_1S_1 + L_2S_{12} + L_3S_{123} + L_4S_{1234}$$

Donde:

- L = *longitud de cada eslabón de la cadena*
- θ = *ángulo de inclinación del eslabón.*
- $C_1 = Cos\ \theta_1$
- $S_1 = Sen\ \theta_1$
- $C_{12} = Cos\ \theta_1 + \theta_2$
- $S_{12} = Sen\ \theta_1 + \theta_2$
- $C_{123} = Cos\ \theta_1 + \theta_2 + \theta_3$
- $S_{123} = Sen\ \theta_1 + \theta_2 + \theta_3$
- $C_{123} = Cos\ \theta_1 + \theta_2 + \theta_3 + \theta_4$
- $S_{1234} = Sen\ \theta_1 + \theta_2 + \theta_3 + \theta_4$

5.3.2.2 MCDP del dedo pulgar

El MCDP correspondiente al dedo pulgar es diferente al anterior, debido a que este solo cuenta con 3 eslabones (figura 5.22).

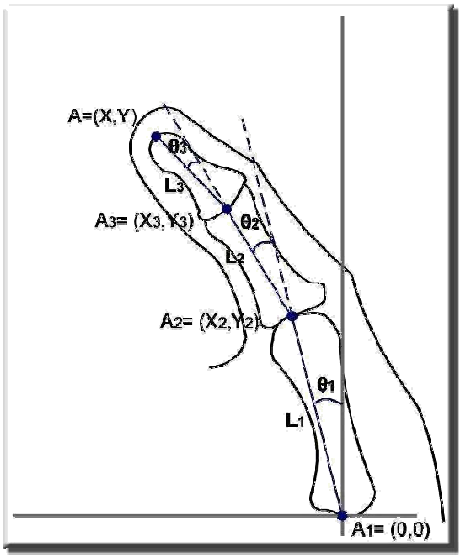


Figura 5.22 MCDP del dedo pulgar

La cadena cinemática que se utilizó en el modelo está compuesta por los siguientes puntos:

$$A_1 = (0,0)$$
$$A_2 = (X_2, Y_2)$$
$$A_3 = (X_3, Y_3)$$
$$_A = (X, Y)$$

Las ecuaciones correspondientes al nodo $A_2 = (X_2, Y_2)$ son las siguientes:

$$X_2 = L_1 C_1$$
$$Y_2 = L_1 S_1$$

Las ecuaciones correspondientes al nodo $A_3 = (X_3, Y_3)$ son las siguientes:

$$X_3 = L_1 C_1 + L_2 C_{12}$$
$$Y_3 = L_1 S_1 + L_2 S_{12}$$

Las ecuaciones correspondientes al nodo A = (X, Y) son las siguientes:

$$X = L_1 C_1 + L_2 C_{12} + L_3 C_{123}$$
$$Y = L_1 S_1 + L_2 S_{12} + L_3 S_{123}$$

Donde:

- L = *longitud de cada eslabón de la cadena*
- θ = *ángulo de inclinación del eslabón.*
- $C_1 = Cos\ \theta_1$
- $S_1 = Sen\ \theta_1$
- $C_{12} = Cos\ \theta_1 + \theta_2$
- $S_{12} = Sen\ \theta_1 + \theta_2$
- $C_{123} = Cos\ \theta_1 + \theta_2 + \theta_3$
- $S_{123} = Sen\ \theta_1 + \theta_2 + \theta_3$

La tabla 5.3 muestra la longitud de cada eslabón utilizado en la cadena cinemática de cada dedo:

Dedo	**L_1**	**L_2**	**L_3**	**L_4**
Pulgar	54mm	30mm	23mm	
Ìndice	37mm	39mm	22mm	17mm
Medio	65mm	45mm	27mm	18mm
Anular	58mm	43mm	25mm	19mm
Meñique	55mm	33mm	17mm	17mm

Tabla 5.3 Longitudes utilizadas en el MCDP

5.3.3 Comprobación del MCDP

Una vez obtenido el modelo matemático correspondiente a la cadena cinemática de cada dedo, el siguiente punto fue comprobar que éste funcione de manera correcta para las necesidades del presente trabajo. Para esto se utilizó el software MATLAB mediante el cual se resolvieron las ecuaciones obtenidas en el modelo, para que posteriormente fueran graficadas y asignarles valores a cada uno de los ángulos de los eslabones con el fin de analizar el comportamiento del modelo matemático.

Para la representación gráfica de la cadena cinemática, solo se utilizó la cadena de los dedos meñique y pulgar, debido a que las cadenas correspondientes a los demás dedos de la mano, son similares a la del meñique, por lo tanto, se utilizó la misma cadena para el resto de los elementos.

El programa realizado en MATLAB, esta compuesto por dos figuras de dos gráficas cada una, mediante las cuales se representó la cadena cinemática de cada uno de los dedos mencionados anteriormente.

El código utilizado para dibujar las graficas del dedo meñique es el siguiente:

```
% Graficas del MCDP del dedo meñique
figure (1)
subplot(2,1,1); plot(xa,ya, '-bo',...
            'LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor',[.49 1 .63],...
            'MarkerSize',12)
title ('Representación grafica del MCDP del dedo meñique')
grid
subplot(2,1,2); plot(x,y, '-bo',...
            'LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor',[.49 1 .63],...
            'MarkerSize',12)
grid
```

La figura 5.23 muestra las graficas del MCDP del dedo meñique; la primer grafica representa la cadena cinemática representativa de las ecuaciones del MCDP, y la segunda representa la forma en que se gráfico en el modelo virtual, parece tener dimensiones diferentes, sin embargo, esto no es así, esta ilusión es creada por la vista y la escala asignada por MATLAB a la gráfica.

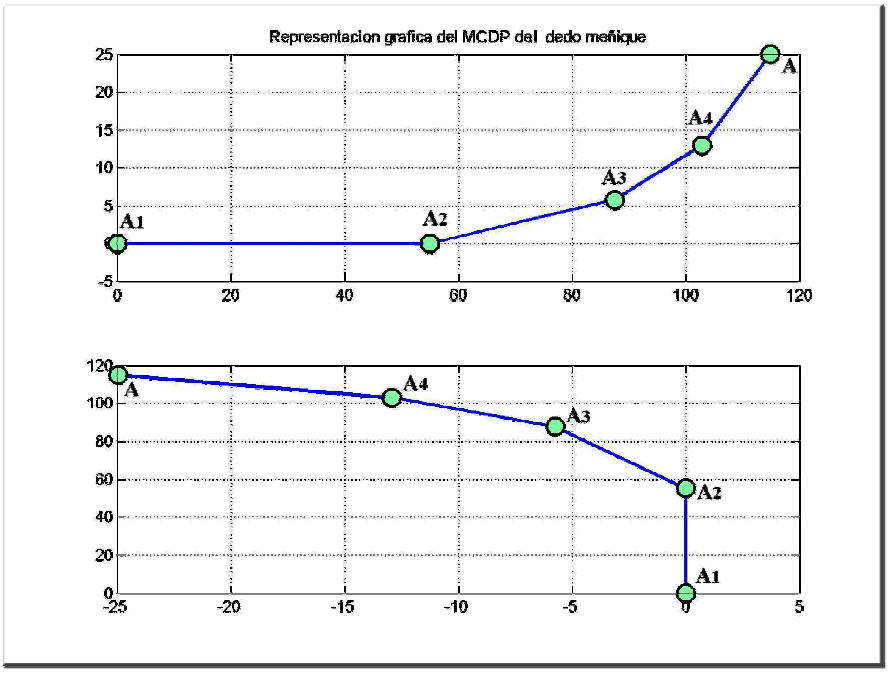


Figura 5.23 Gráficas obtenidas en MATLAB para el dedo meñique

La figura 5.24 muestra la grafica del MCDP del dedo pulgar, la segunda grafica muestra la posición de la cadena cinemática utilizada en el modelo virtual.

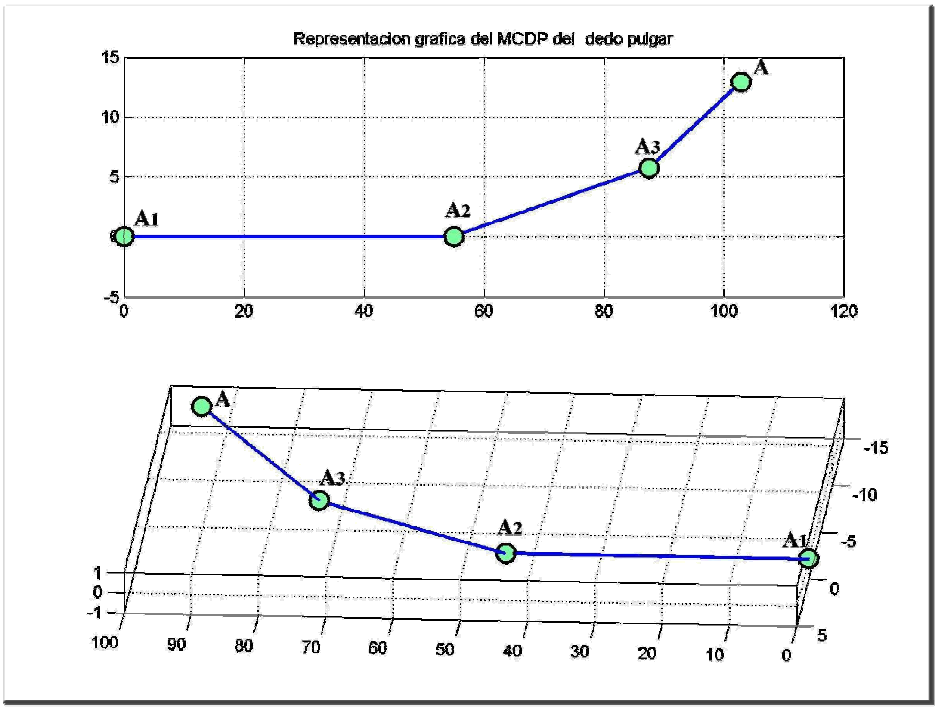


Figura 5.24 Gráficas obtenidas en MATLAB para el dedo pulgar

La figura 5.25 muestra la gráfica obtenida en MATLAB, junto al modelo virtual correspondiente al dedo meñique, donde se muestra los puntos representativos en el modelo. El modelo matemático obtenido cumple con todos los requerimientos necesarios para representar los movimientos permitidos por el guante, por lo tanto, se puede utilizar en el modelo virtual para asignar comportamientos complejos a éste.

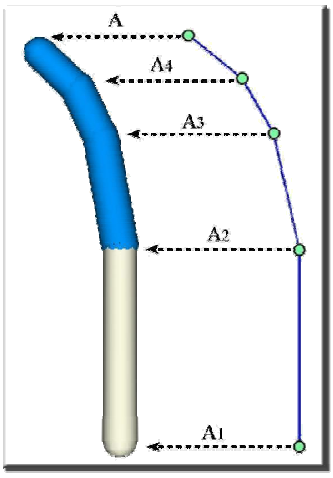


Figura 5.25 Comparación de la cadena con el modelo virtual

5.3.4 Modelado de la mano

Para el modelado en 3D de la mano se utilizó el Lenguaje de Modelado de Realidad Virtual (VRML), el browser Cortona VRML Client 4.2 para visualizar el modelo y el editor del lenguaje VRML llamado VRMLPad.

El modelo de la mano esta formado básicamente por un conjunto de cilindros que constituyen los eslabones de la cadena y pequeñas esferas que forman los nodos, cabe resaltar que para el diseño de la mano virtual también se utilizaron las medidas de los huesos, tomando la escala 10mm:1unidad en VRML. Por ejemplo, para el dedo meñique que tiene una longitud L1=55mm para su primer eslabón, en el modelo de VRML se creó un cilindro con una longitud de 5.5 unidades (figura 5.26).

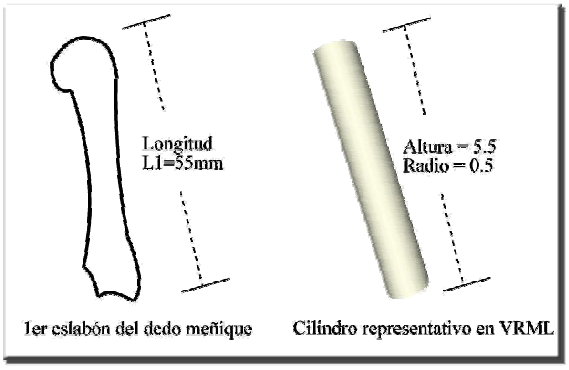


Figura 5.26 Escala entre las dimensiones de la cadena cinemática y el modelo en VRML

Para la creación del cilindro se utilizó el nodo Shape, el cual cuenta con dos campos: *appearance* y *geometry*. El campo apperance hace referencia al nodo del mismo nombre, mediante el cual se controlan los atributos visuales que se aplican sobre los objetos. El campo geometry, determina un nodo de geometría, que en este caso será un cilindro. Para asignar, una posición y orientación al cilindro se utilizó el nodo Transform. El código para el diseño del cilindro quedó de la siguiente forma:

```
DEF Cilindro1 Transform
 {
        center 0 -2.75 0
        children [
                Shape {
                        appearance Appearance {material Material {diffuseColor 1 1 0.9} }
                        geometry Cylinder
                          {
                                radius 0.5
                                height 5.5
                          }
                        }
                ]
}
```

Cuando se crea un objeto geométrico en VRML posee un centro geométrico (figura 5.27), el cual se encuentra en el origen de las coordenadas de su sistema específico. En base a este centro geométrico es que se crean los objetos (el objeto se crea a partir del centro geométrico, la mitad de unidades hacia un extremo de cada eje), de igual manera, sirve como pivote para realizar operaciones sobre el objeto como: escalación y rotación.

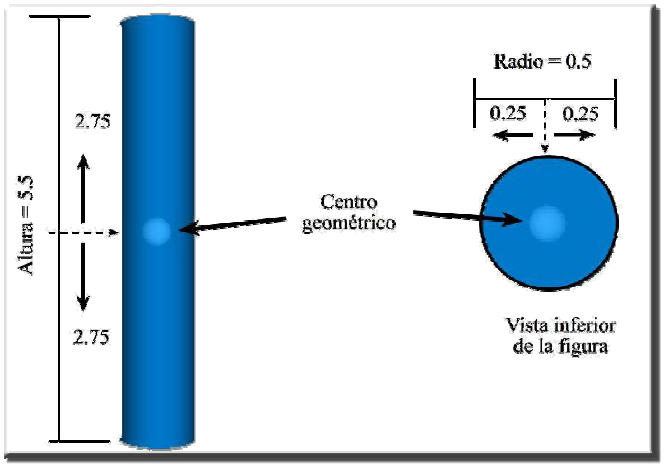


Figura 5.27 Centro geométrico en VRML

De ahí la importancia de éste para el diseño del modelo virtual, debido que los cilindros tienen que rotar sobre la base, para esto solo se tiene que mover el centro geométrico hasta la base de la figura. Esto se hace utilizando el campo del nodo Transform., Por ejemplo, el cilindro anterior tiene una altura de 5.5, por lo tanto la figura se dibuja 2.75 unidades hacia arriba del centro geométrico y 2.75 unidades hacia abajo, sobre el eje Y. Por lo tanto, el

nuevo centro geométrico de la figura estará a -2.75 sobre el eje Y, utilizando el campo center quedaría de la siguiente forma (figura 5.28):

Center 0 -2.75 0

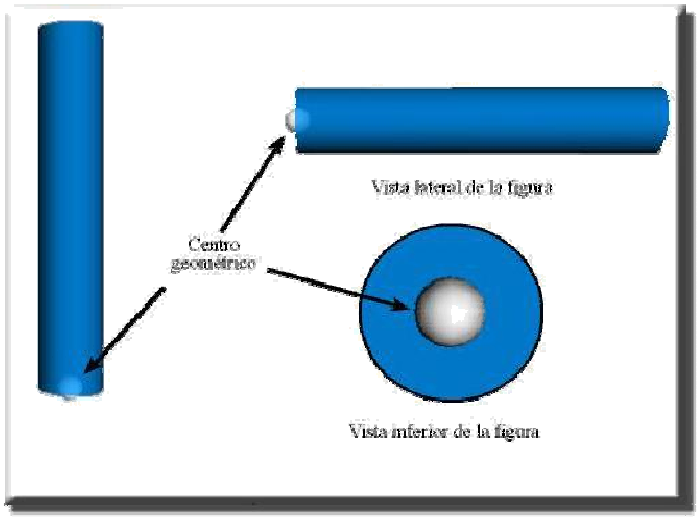


Figura 5.28 Ubicación del centro geométrico después de la operación Center

El centro geométrico se ubica en la parte inferior del cilindro para obtener la rotación de forma correcta y el cilindro no rote a partir del centro de la figura.

Para el modelo de los puntos A, A_1, A_2, A_3 y A_4, se utilizaron esferas, con un radio del mismo tamaño que el radio de cada cilindro, nuevamente para el diseño de las esferas se hizo mediante el nodo shape. El código para el diseño de la esfera quedó de la siguiente forma:

```
DEF Esfera1 Transform
 {
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 1 1 0.9 } }
                        geometry Sphere
                                {
                                radius 0.5
                                }
                        }
                ]

}
```

A cada una de las figuras le corresponde una posición y un ángulo de rotación, para determinar la posición tanto de los cilindros como de las esferas se utilizó el MCDP, mediante el cual se realizaron los cálculos correspondientes hasta encontrar la posición sobre los ejes *X, Y* y *Z* respectivamente. El ángulo asignado a la rotación de cada cilindro es

el que se sustituye en el modelo cinemático. Para realizar las operaciones necesarias para el cálculo de dichas posiciones se utilizó un script, el cual será descrito más adelante.

Para el modelado de un dedo completo, se utilizaron 4 cilindros y 5 esferas (figura 5.29), asignando a cada cilindro un radio con una pequeña diferencia entre ellos, con el fin de dar mayor realismo a éste.

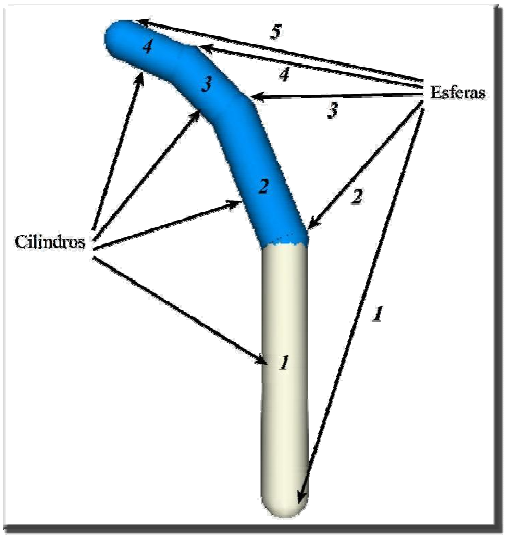


Figura 5.29 Componentes del dedo en VRML

El código completo de la mano se encuentra en el ***Anexo A.*** En la figura 5.30 se muestra el modelo completo de la mano.

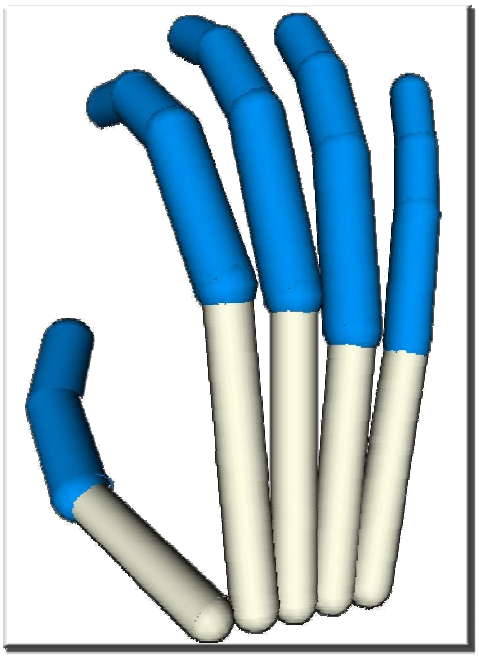


Figura 5.30 Modelo final de la mano

5.3.4.1 Programación del Script

VRML cuenta con características importantes para el desarrollo de ambientes virtuales con un alto grado de realismo, sin embargo, para realizar las animaciones que se desean se utiliza un lenguaje externo a éste.

Para la asignación de movimientos complejos al ambiente virtual desarrollado previamente, se utilizó el lenguaje Javascript, con el fin de realizar los cálculos matemáticos correspondientes al MCDP de cada dedo, aunque el trabajo es repetitivo, se tienen que establecer los valores correctos a cada eslabón, de igual manera el centro geométrico de cada figura tiene que ser reubicado correctamente en la base de cada cilindro, en caso contrario, la figura tendrá una rotación incorrecta.

En el caso del presente trabajo se decidió colocar el script dentro del mismo código VRML, al final del código que crea la mano.

Para generar eventos de forma periódica se utilizó un TimeSensor. Al TimeSensor se le asignó el nombre Reloj, como se ve a continuación:

```
DEF Reloj TimeSensor {
   cycleInterval 10.0
   loop TRUE
   startTime 1.0
   stopTime 0.0
}
```

Para cada dedo se utilizan 13 EventIns, en los cuales se almacenarán los nuevos valores de rotación y traslación para los objetos del modelo.

```
eventOut        SFVec3f        TrasladaA_changed1
eventOut        SFVec3f        TrasladaA_changed2
eventOut        SFVec3f        TrasladaA_changed3
eventOut        SFVec3f        TrasladaA_changed4
eventOut        SFVec3f        TrasladaA_changed5
eventOut        SFVec3f        TrasladaC1A_changed
eventOut        SFVec3f        TrasladaC2A_changed
eventOut        SFVec3f        TrasladaC3A_changed
eventOut        SFVec3f        TrasladaC4A_changed
eventOut        SFRotation     RotaC1A_changed
eventOut        SFRotation     RotaC2A_changed
eventOut        SFRotation     RotaC3A_changed
eventOut        SFRotation     RotaCil4A_changed
```

Mediante la función initialize se inicializaron los valores de los sensores, así como la obtención de la equivalencia en grados de éstos, también en esta función se reciben los datos de cada sensor del guante. A continuación se muestra fragmento del código utilizado:

```
function initialize()
```

```
{
        /* Valores de los sensores */
        menique=5
        anular=10
        medio=15
        indice=20
        pulgar=15

        /* Convertir los valores de cada sensor de 0-63 en angulos
           para los objetos en el ambiente virtual */

                /**** Anular ****/
                tA1=(anular*90)/63;
                tA2=(anular*90)/63;
                tA3=(anular*90)/63;
}
```

Se utilizó un evento de entrada el cual tiene el nombre de CalculaModelo, a cada evento de entrada corresponde una función del mismo nombre, por lo tanto dentro del script existe una función llamada CalculaModelo donde se llevaran a cabo los cálculos matemáticos correspondientes al MCDP, para determinar la posición de cada figura utilizada en el modelo virtual. Por ejemplo, dentro de esta función el código para calcular la posición del segundo eslabón es el siguiente:

```
function CalculaModelo()
{

/*Convertir los ángulos en radianes para evaluarlos en el modelo */

angulo1=(Math.PI*(0+90))/180;
angulo2=(Math.PI*t1)/180;

/*Calcular el modelo cinemático para encontrar la posición de cada objeto del dedo */

X1=0;
Y1=0;
X2=5.5*(Math.cos(angulo1));
Y2=5.5*(Math.sin(angulo1));

/*Asignar a cada objeto su nueva posición de acuerdo a los cálculos obtenidos en el modelo */

/*************** Eslabón 2 ***************************/

Traslada_changed2[0]=X2;
Traslada_changed2[1]=Y2-4;
Traslada_changed2[2]=0;

TrasladaC2_changed[0]=X2;
TrasladaC2_changed[1]=Y2-2.4;
```

```
TrasladaC2_changed[2]=0;

RotaC2_changed[0]=0;
RotaC2_changed[1]=0;
RotaC2_changed[2]=1;
RotaC2_changed[3]=(angulo1 +angulo2)-dif_ang;
}
```

Mediante el uso de rutas se asignan los nuevos valores de traslación y rotación a los campos correspondientes de un nodo en específico (figura 5.31). En el siguiente código se asigna una nueva posición al objeto Esfera_A2, la cual se almacenó en el eventIn *TrasladaA_changed,* Modelo es el nombre del nodo Script. La ruta correspondiente al Reloj se asigna a la función CalculaModelo, para generar eventos de forma constante.

```
ROUTE     Reloj.fraction_changed     TO     Modelo.CalculaModelo

#******************** Asignar las rutas al dedo anular ***************************

ROUTE     Modelo.TrasladaA_changed2     TO     Esfera_A2.translation
ROUTE     Modelo.TrasladaC2A_changed    TO     Cilindro_A2.translation
ROUTE      Modelo.RotaC2A_changed       TO     Cilindro_A2.rotation
```

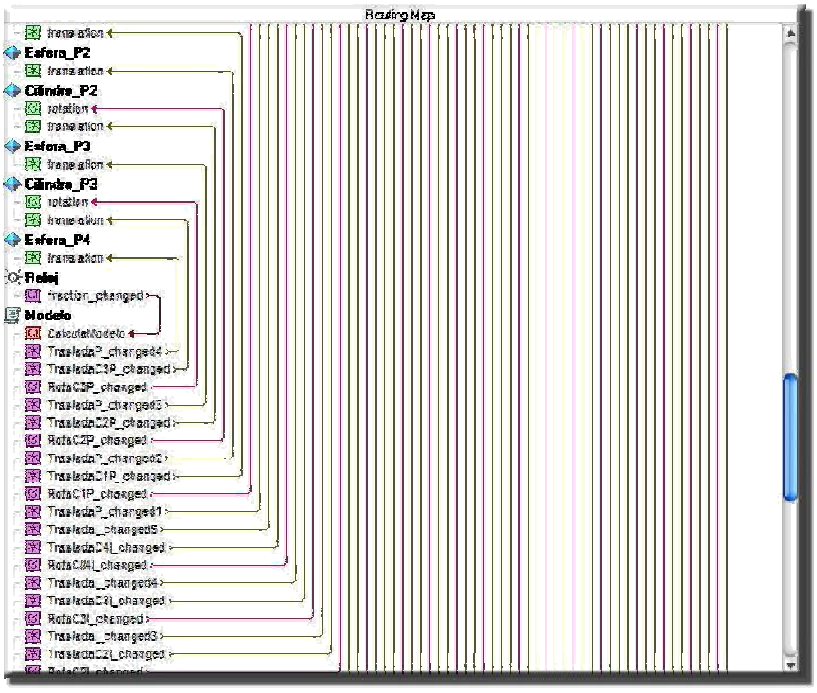


Figura 5.31 Mapa de rutas en VrmlPad

El código completo del programa Script utilizado para asignar los movimientos de la mano se encuentra en el ***Anexo A*.**

5.4 Desarrollo

5.4.1 Programas en Delphi

Antes de comenzar a programar se tiene que agregar los componentes que se utilizarón para el desarrollo de los programas.

5.4.1.1 Componente Cortona VRML Client 4.2

Para visualizar un ambiente virtual utilizando un programa desarrollado en Delphi, es necesario agregar el browser Cortona VRML Client 4.2. Para agregarlo se siguieron los pasos que se indican a continuación:

1. El browser Cortona se agrega como un control ActiveX, en el menú "Component" en la opción "Import ActiveX Control".

2. En esta opción se busca el browser dentro de una lista de componentes ActiveX, el nombre del control ActiveX de cortona es ParallelGraphics Cortona VRML Client 2.1 Type Library (Versión 1.0) (figura 5.32), posteriormente se hace clic en la opción *Install*.

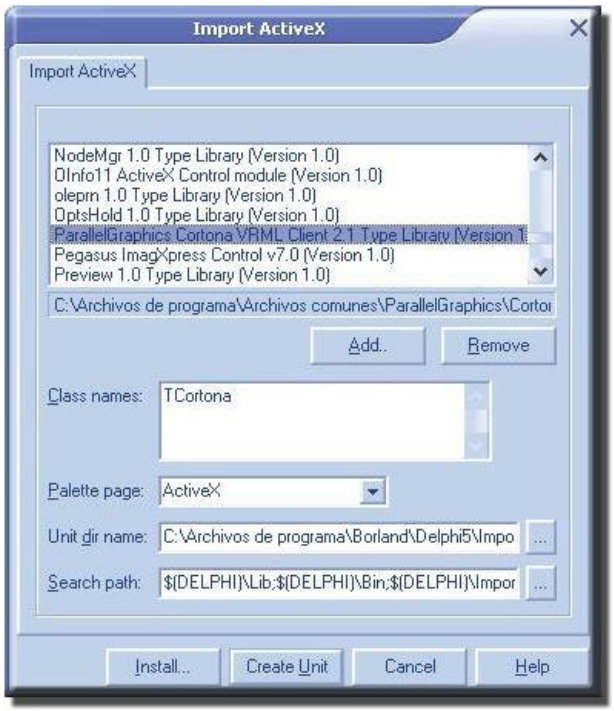


Figura 5.32 Importar control ActiveX

3. En la opción Install, en el botón Browse se selecciona la carpeta y el nombre del archivo con extensión dpk, mediante el cual se instalará el componente, una vez seleccionada la ruta y el nombre del archivo se hace clic en la opción OK (figura 5.33).

Figura 5.33 Ruta de instalación del componente

4. Ahora se ha creado el archivo con extensión dpk para poder instalar el browser Cortona como componente, se da clic en la opción Install y en el cuadro de dialogo emergente seleccionar la opción Yes (figura 5.34).

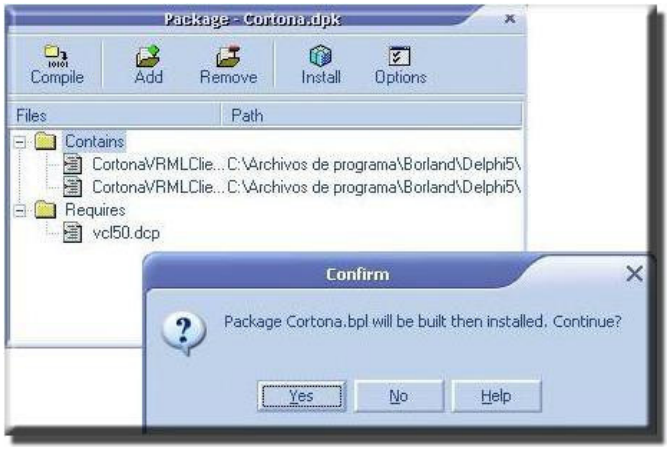


Figura 5.34 Instalar componente

5. Una vez que este componente ha sido agregado, enviará el siguiente mensaje de confirmación (figura 5.35).

Figura 5.35 Confirmación de la instalación del componente

6. Ya que ha sido instalado, se cierra el archivo y se guardan los cambios correspondientes.

7. La ultima operación para agregar el componente, es asignar la ubicación del archivo con extensión dpk, con el fin de que encuentre todos los recursos del componente Cortona. Esto se hace en el menú Environment Options.

8. Seleccionar la pestaña Library. Hacer clic en los puntos suspensivos de la opción Library Path (figura 5.36).

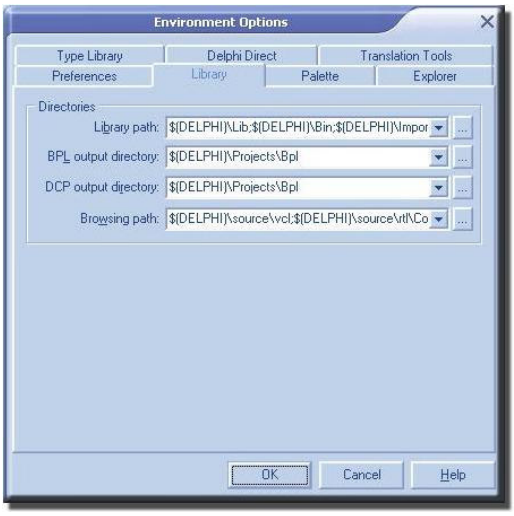


Figura 5.36 Opciones del ambiente

9. En la ventana con el nombre Directories, seleccionar nuevamente los puntos suspensivos y posteriormente, ubicar la ruta en donde se crearon los archivos correspondientes al componente Cortona (figura 5.37).

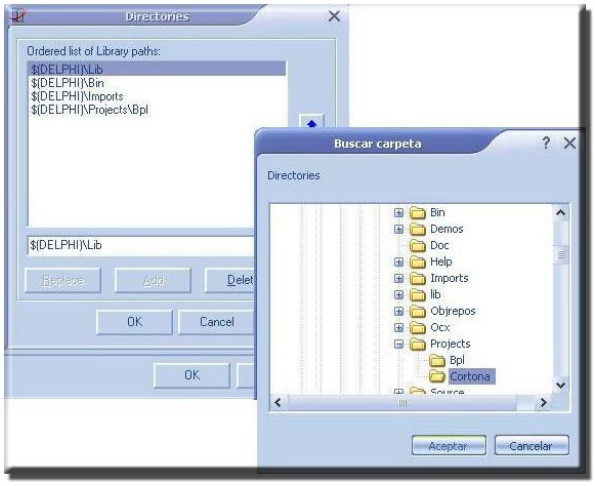


Figura 5.37 Ruta de los archivos del componente

10. Finalmente para agregar la ruta hacer clic en el botón Add y posteriormente en el botón OK.

11. Una vez concluidos los pasos anteriores, se puede utilizar el browser Cortona como componente de Delphi y manipular algunas de sus características (figura 5.38).

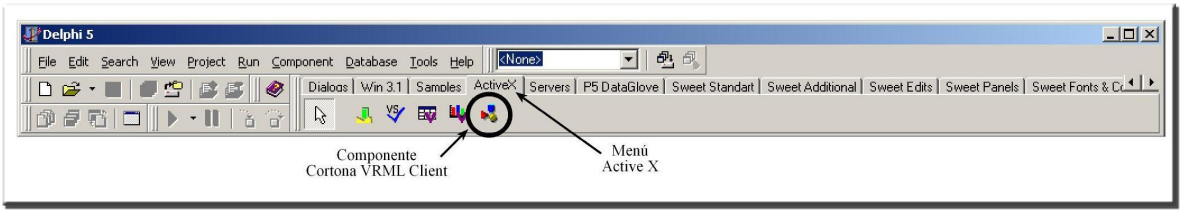


Figura 5.38 Componente Cortona en el menú ActiveX

5.4.1.2 Componente para Delphi P5 DataGlove

Un componente de gran importancia en este trabajo de investigación es el correspondiente al guante P5, este componente se agrega de forma diferente, debido a que el guante P5 no cuenta con un control ActiveX.

El archivo fue descargado de la página: http://web.archive.org/web/20030623032323/

Para agregar este componente se siguieron los pasos que se muestran a continuación:

1. Una vez abierto Delphi, se procede a abrir el archivo con nombre DataGloveD5.dpk, en la opción Open del menú File.

2. En la ventana de instalación dar clic sobre el botón Install (figura 5.39).

Figura 5.39 Instalar el componente P5 DataGlove

3. Cerrar la ventana de instalación guardando los cambios correspondientes.

4. Agregar la ruta de la carpeta donde se encuentran los archivos correspondientes a este componente (figura 5.40).

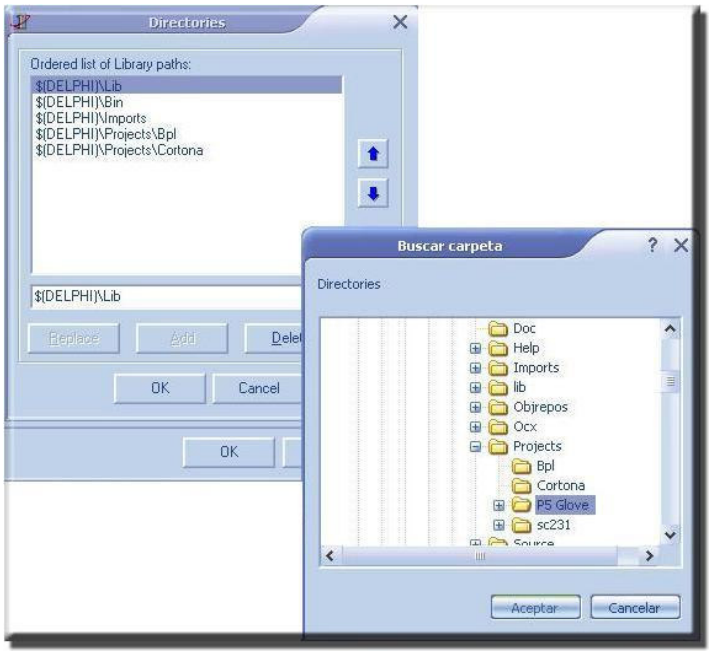


Figura 5.40 Ruta de los archivos del componente

5. Finalmente, el componente se puede utilizar, este componente crea una pestaña adicional con el nombre de P5DataGlove (figura 5.41).

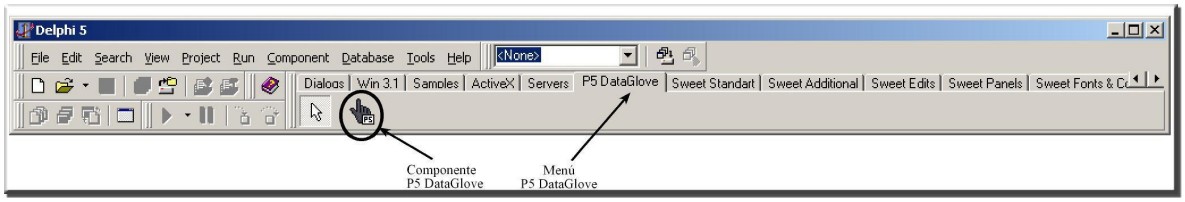


Figura 5.41 Menú P5 DataGlove

5.4.1.3 Programas

En Delphi se desarrollaron programas para el análisis de las señales de los sensores, la posición y orientación del guante y el programa principal para la comunicación con el modelo virtual de la mano, estos programas se describen a continuación

5.4.1.3.1 Panel de sensores

Para el análisis de las señales de cada sensor del guante, se desarrolló un programa que lleva por nombre *"Panel de sensores",* la finalidad de este programa es analisar los datos que son enviados a la PC por cada bend sensor, estos datos varían entre 0 y 63 según sea el grado de flexión que tengan los sensores en el guante. Los valores que se muestran en este programa fueron utilizados para determinar la posición de cada eslabón de la cadena en el ambiente virtual.

Los datos provenientes del guante se almacenaron en una variable, para posteriormente ser enviados a los objetos correspondientes en los programas.

Para obtener de forma gráfica los datos, se utilizaron progress bar (barras progresivas) que muestran los valores de cada sensor en tiempo real (figura 5.42), para poder tener los valores exactos del guante, este debe ser calibrado de forma correcta con anterioridad.

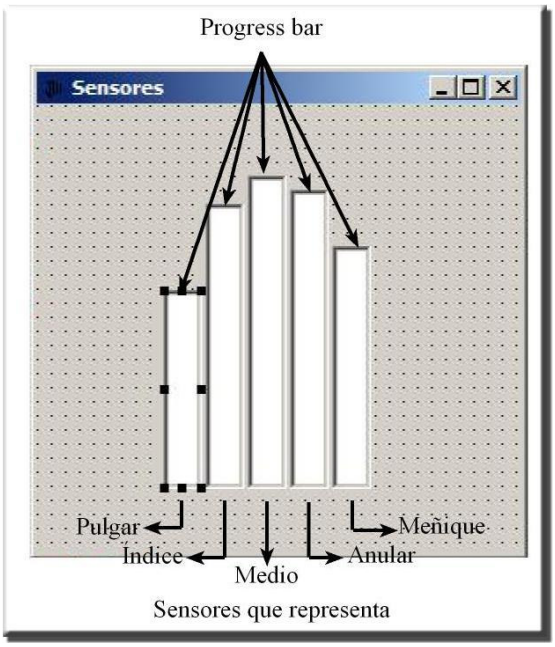


Figura 5.42 Progress bar del programa

El código para realizar la barra progresiva correspondiente al dedo pulgar es el siguiente:

```
object Pulgar: TSCProgress
   Left = 72
   Top = 105
   Width = 22
   Height = 113
   BorderProps.Color = clSilver
   BorderProps.Width = 1
   Max = 63
   Orientation = scoVertical
```

```
   PositionColor = clBackground
   Style = scpsXP
   TabOrder = 1
   OnChange = PulgarChange
End
```

Para representar el valor numérico de los datos se utilizó una etiqueta, se envió el valor de la variable en la cual se encuentran almacenados dichos datos a la propiedad caption del objeto label (figura 5.43).

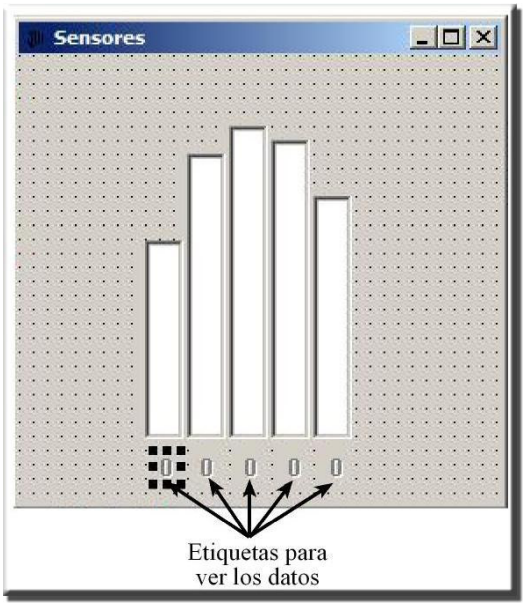


Figura 5.43 Etiquetas

El código para realizar una etiqueta es el siguiente:

```
object SCLabel3: TSCLabel
   Left = 76
   Top = 224
   Width = 17
   Height = 20
   AutoSize = True
   Caption = '0'
   Color = cl3DLight
   Enabled = False
   Font.Charset = DEFAULT_CHARSET
   Font.Color = clWindowText
   Font.Height = -13
   Font.Name = 'MS Sans Serif'
   Font.Style = [fsBold]
   Hottrack = False
   ParentColor = False
```

```
  ParentFont = False
  TabOrder = 6
end
```

Para que el objeto P5DataGlove pueda realizar las operaciones correctas y enviar los datos al lugar correcto (figura 5.44), es necesario asignarle el nombre del procedimiento donde se realiza la captura de los datos del guante, todas las capturas se deben realizar dentro de este proceso, ya que solo este recibe señales del guante, el nombre del procedimiento donde se realizaron todas las operaciones se llama *PulgarChange*

Figura 5.44 Objeto P5DataGlove

El código para realizar el objeto P5DataGlove1 es el siguiente:

```
object P5DataGlove1: TP5DataGlove
  Interval = 1
  OnProcess = PulgarChange
  GloveType = gtLeftHand
  Left = 16
  Top = 208
end
```

El siguiente código muestra las instrucciones utilizadas para la captura de los datos del guante, el envío de éstos al progress bar correspondiente y a las etiquetas para mostrar los datos en el formulario:

```
procedure TForm1.PulgarChange(Sender: TObject);

var Sensor_pulgar:byte;
var Sensor_indice:byte;
var Sensor_medio:byte;
var Sensor_anular:byte;
```

```
var Sensor_menique:byte;

begin

Sensor_pulgar:=P5DataGlove1.FingerThumb;
Sensor_indice:=P5DataGlove1.FingerIndex;
Sensor_medio:=P5DataGlove1.FingerMiddle;
Sensor_anular:=P5DataGlove1.FingerRing;
Sensor_menique:=P5DataGlove1.FingerPinky;

Pulgar.Position:=Sensor_pulgar;
Indice.Position:=Sensor_indice;
Medio.Position:=Sensor_medio;
Anular.Position:=Sensor_anular;
Menique.Position:=Sensor_menique;

SCLabel3.Caption := InttoStr(Sensor_pulgar);
SCLabel4.Caption := InttoStr(Sensor_indice);
SCLabel5.Caption := InttoStr(Sensor_medio);
SCLabel6.Caption := InttoStr(Sensor_anular);
SCLabel7.Caption := InttoStr(Sensor_menique);

end;
```

La figura 5.45 muestra el programa final:

Figura 5.45 Programa "Panel de sensores"

5.4.1.3.2 Orientación

El desplazamiento de la mano dentro del ambiente virtual es un punto importante, para el análisis de las coordenadas enviadas por el guante a la PC, se desarrolló un programa

denominado *"Orientación"*, por medio de este programa se capturaron las señales de posición y orientación del guante completo, para que más adelante fueran enviadas al modelo virtual.

Un aspecto importante encontrado en estos datos, es que el guante envía datos con magnitudes muy grandes para el ambiente virtual, por lo que fue necesario truncar estos datos, para evitar desplazamientos bruscos del modelo virtual. De igual manera se analizó la forma en que trabajan los datos del guante, este envía señales de 0° a 180° en una dirección y de 0° a -180° para que de esta manera se cumplan los 360° para lograr un giro completo. Otro aspecto de gran importancia a destacar son los datos yaw, pitch y roll, que envía el guante, éstos son muy inestables, y varían de forma muy significativa con un pequeño movimiento del guante, de ahí que la rotación realizada por el modelo virtual sea muy rápida.

Como el objetivo de este programa fue el análisis de los datos, solo se utilizaron etiquetas dentro del formulario. En el programa se muestran tanto los datos originales, como los datos truncados que fueron utilizados en el modelo virtual (figura 5.46).

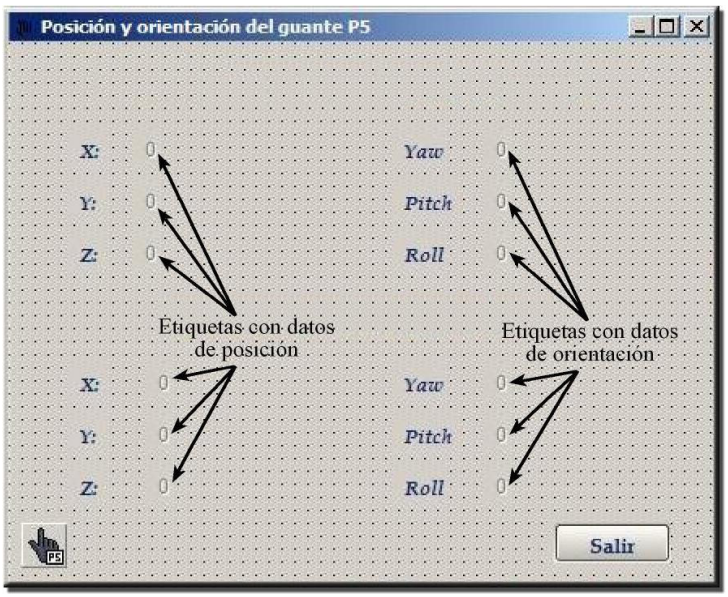


Figura 5.46 Etiquetas del programa orientación

Con el fin de que solo se puedan ver los datos y no realizar modificaciones, se deshabilitaron las etiquetas para que no pudieran ser modificadas, el código para realizar una de estas etiquetas es el siguiente:

```
bject SCLabel1: TSCLabel
   Left = 296
   Top = 200
   Width = 15
   Height = 19
   AutoSize = True
```

```
  Caption = '0'
  Enabled = False
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clHighlight
  Font.Height = -12
  Font.Name = 'Arial'
  Font.Style = []
  ParentFont = False
  TabOrder = 0
  OnClick = SCLabel1Click
 End
```

El procedimiento utilizado para la captura de las señales del guante se llama SCLabel1Click y se tiene que hacer referencia a este procedimiento en el objeto P5DataGlove1.

Por medio del siguiente código se realiza el objeto P5DataGlove1:

```
object P5DataGlove1: TP5DataGlove
  Interval = 1
  OnProcess = SCLabel1Click
  GloveType = gtLeftHand
  Left = 8
  Top = 296
 end
```

Con el siguiente código se realiza el procedimiento *SCLabel1Click* en el cual se capturan todas las señales necesarias para analizar las coordenadas de posición y de orientación respectivamente.

```
procedure TForm1.SCLabel1Click(Sender: TObject);

var X,Y,Z,Yaw,Pitch,Roll:Integer;

begin

//Datos de posición del Guante P5
//Datos truncados
X := Trunc(P5DataGlove1.Position.PosX) div 100;
Y := Trunc(P5DataGlove1.Position.PosY) div 100;
Z := Trunc(P5DataGlove1.Position.PosZ) div 100;

SCLabel7.Caption:=IntToStr(X);
SCLabel8.Caption:=IntToStr(Y);
SCLabel9.Caption:=IntToStr(Z);

//Datos originales X, Y, Z
SCLabel19.Caption:=FloatToStr(P5DataGlove1.Position.PosX);
SCLabel20.Caption:=FloatToStr(P5DataGlove1.Position.PosY);
SCLabel21.Caption:=FloatToStr(P5DataGlove1.Position.PosZ);
```

```
//Datos de orientación del Guante P5

//Datos truncados
Yaw := Trunc(P5DataGlove1.Position.YAW) div 10;
Pitch := Trunc(P5DataGlove1.Position.PITCH) div 10;
Roll := Trunc(P5DataGlove1.Position.ROLL) div 10;

SCLabel1.Caption:=IntToStr(Yaw);
SCLabel2.Caption:=IntToStr(Pitch);
SCLabel3.Caption:=IntToStr(Roll);

//Datos originales Yaw, Pitch y Roll
SCLabel22.Caption:=FloatToStr(P5DataGlove1.Position.Yaw);
SCLabel23.Caption:=FloatToStr(P5DataGlove1.Position.Pitch);
SCLabel24.Caption:=FloatToStr(P5DataGlove1.Position.Roll);

end;
```

La figura 5.47 muestra el programa final:

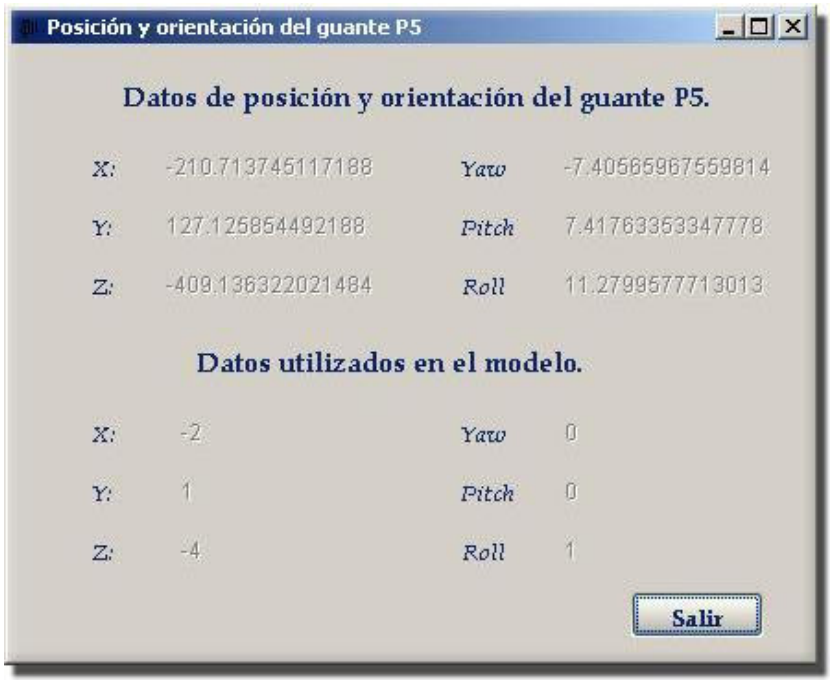


Figura 5.47 Interfaz del programa "orientación"

5.4.1.3.3 Mano

Finalmente ya que se analizaron los datos se procedió a realizar el programa mediante el cual se pasarían los datos del guante al modelo virtual. El programa Mano esta formado por los siguientes objetos (figura 5.48):

- Memo: dentro de este objeto se cargó el contenido del archivo VRML correspondiente al modelo virtual.

- Cortona: este objeto es el browser que permite visualizar la escena dentro del programa.

- P5DataGlove: con este objeto se obtienen los datos del guante, los cuales son enviados cada 1ms.

- Timer: con la ayuda de este objeto se guardaron los nuevo valores dentro del archivo VRML, de igual manera se utilizó para actualizar la escena dentro del browser Cortona, estas operaciones las realiza cada 400ms.

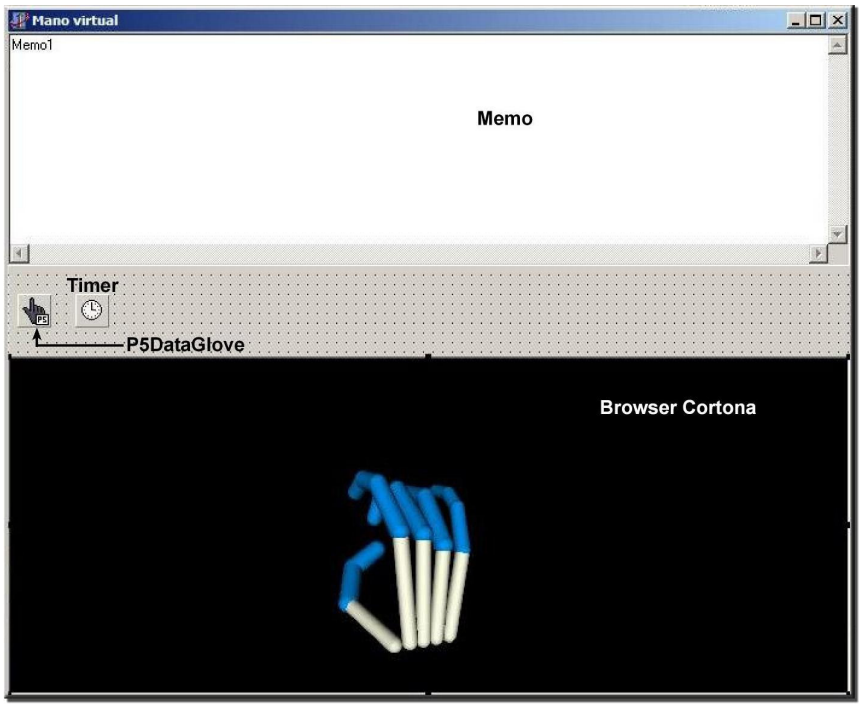


Figura 5.48 Elementos del programa

Por medio del siguiente código se realiza el objeto Memo1:

```
object Memo1: TMemo
   Left = 0
   Top = 544
   Width = 698
   Height = 4
   Align = alBottom
   Lines.Strings = (
     'Memo1')
   ScrollBars = ssBoth
```

```
  TabOrder = 1
  Visible = False
end
```

En el programa final el objeto Memo se encuentra oculto (figura 5.49), con el fin de que solo se vea la escena virtual, pero este objeto se encuentra trabajando en la parte trasera del browser Cortona.

Figura 5.49 Propiedades del objeto Memo1 en el inspector de objetos

El siguiente código crea el objeto TTimer

```
object Timer1: TTimer
   Interval = 400
   OnTimer = Timer1Timer
   Left = 184
   Top = 104
 End
```

El Timer utilizado para guardar los datos en el archivo VRML trabaja bajo un intervalo de 400ms (figura 5.50) y el objeto TP5DataGlove, funciona en un intervalo de 1ms (figura 5.51).

Figura 5.50 Propiedades del objeto TTimer

Figura 5.51 Propiedades del objeto TP5DataGlove

Con respecto al objeto Cortona se deshabilitó el logo que muestra cada vez que se carga una escena (figura 5.52), con el fin de tener mejor render para el modelo virtual. Para que el browser se ajuste al tamaño del formulario, se seleccionó alClient en la propiedad Align (figura 5.53).

Figura 5.52 Deshabilitar el logo de Cortona

Figura 5.53 Propiedad Align

En el procedimiento llamado P5DataGlove1Process se realizó la captura de los datos, así como la modificación del archivo VRML que se encuentra dentro del objeto Memo1. El código del este procedimiento es el siguiente:

```
procedure TForm1.P5DataGlove1Process(Sender: TObject);
var D1,D2,D3,D4,D5:Extended;

begin
D1:=63-P5DataGlove1.FingerPinky;
D2:=63-P5DataGlove1.FingerRing;
D3:=63-P5DataGlove1.FingerMiddle;
D4:=63-P5DataGlove1.FingerIndex;
D5:=63-P5DataGlove1.FingerThumb;


Memo1.Lines[795]:=copy(Memo1.Lines[795],1,10)+FloatToStr(D1);
Memo1.Lines[796]:=copy(Memo1.Lines[796],1,9)+FloatToStr(D2);
Memo1.Lines[797]:=copy(Memo1.Lines[797],1,8)+FloatToStr(D3);
Memo1.Lines[798]:=copy(Memo1.Lines[798],1,9)+FloatToStr(D4);
Memo1.Lines[799]:=copy(Memo1.Lines[799],1,9)+FloatToStr(D5);

end;
```

En el programa final, solo se muestra el browser con la escena correspondiente, pero todos los objetos realizan la función que les fue asignada en el programa, aunque éstos no sean visibles. La figura 5.54 muestra el funcionamiento del programa.

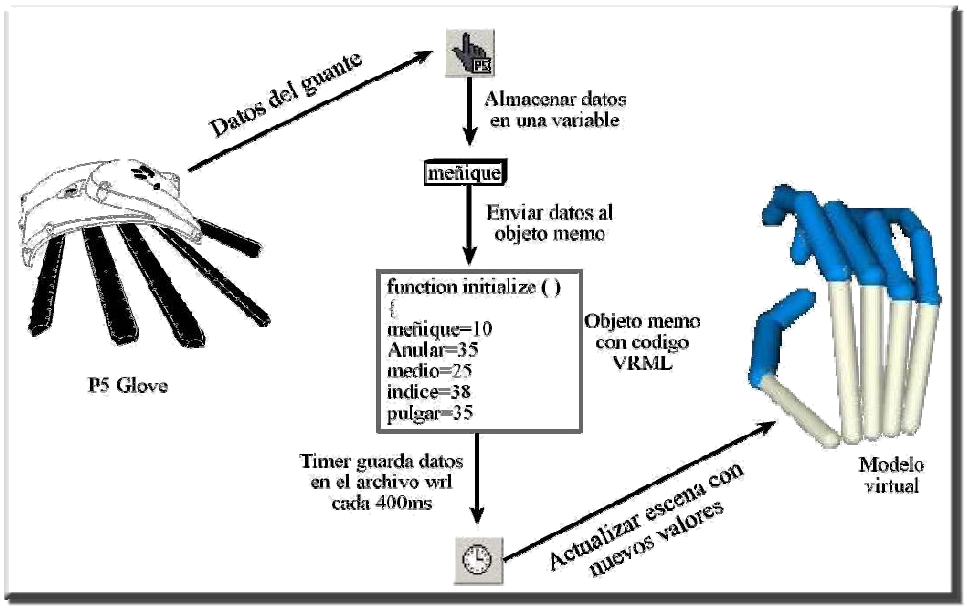


Figura 5.54 Funcionamiento del programa "Mano"

En la figura 5.55 se muestra el programa Mano, con el objeto Memo visible al lado izquierdo del formulario y el browser al lado derecho.

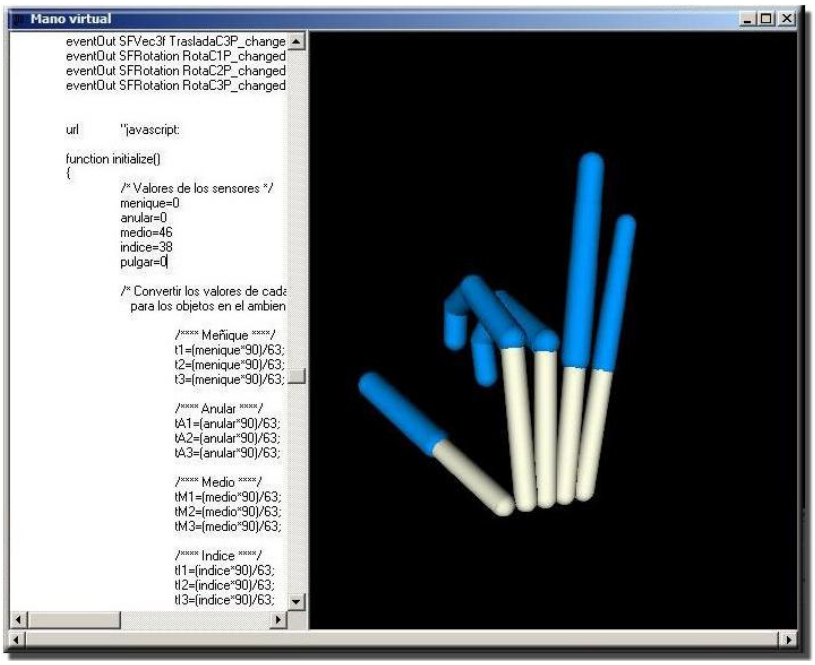


Figura 5.55 Programa con el objeto Memo visible

La figura 5.56 muestra el programa final:

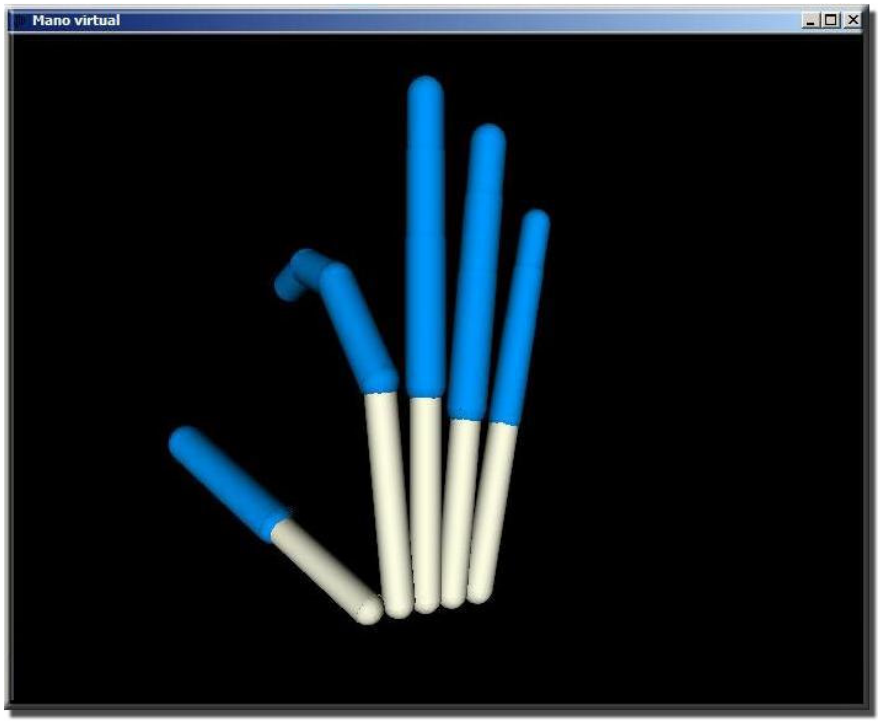
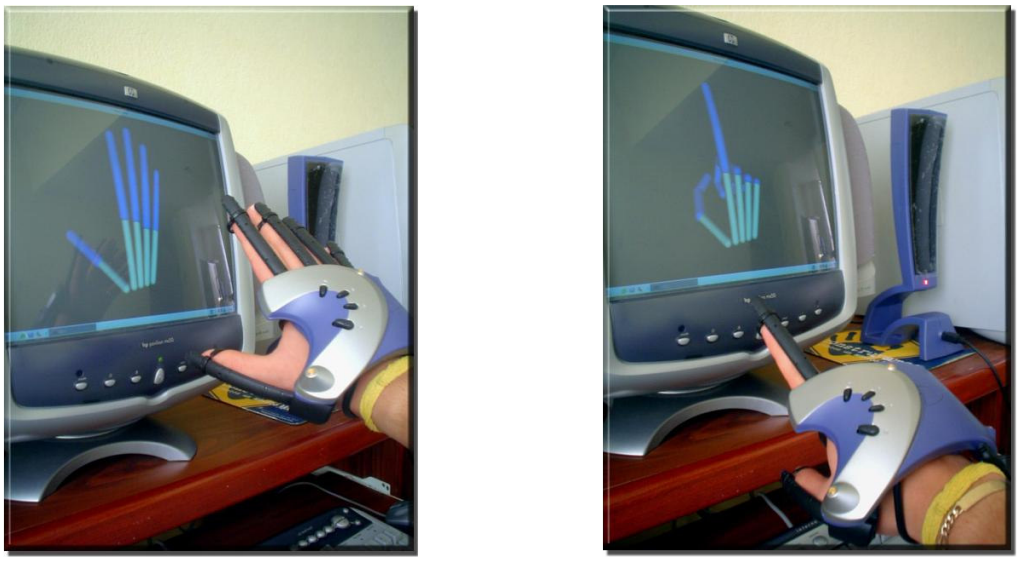


Figura 5.56 Interfaz del programa "Mano"

En las imágenes 5.57 (a) y (b) y 5.58 se muestra al programa correspondiente al modelo virtual de la mano interactuando con el guante de datos P5.



(a) **(b)**

Figura 5.57 Interacción de la interfaz con el guante

Figura 5.58 Rotación de la mano virtual

5.4.2 External Authoring Interface (EAI)

Otra forma de manipular las características de los objetos en un ambiente virtual es utilizando un applet. Para lo cual se utilizó una página web en la cual se hace referencia al modelo virtual de la mano y mediante Java se realiza la conexión entre el applet y el ambiente virtual.

Se desarrolló un applet con 3 campos de texto para cada dedo (figura 5.59), mediante los cuales se asignan nuevos valores para los ángulos de los 3 eslabones móviles de la cadena cinemática. Y un botón para enviar estos nuevos valores a los objetos correspondientes. Además de poder manipular la posición de cada dedo se cuenta con 3 campos más para cambiar a una nueva posición la mano completa.

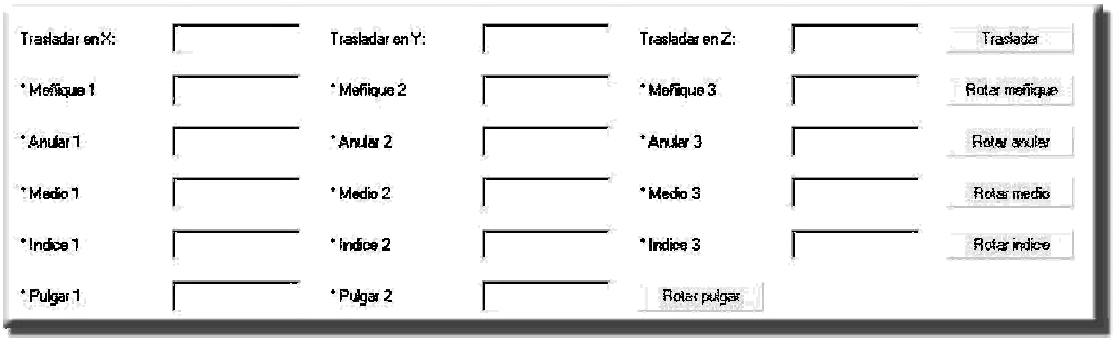


Figura 5.59 Applet de Java

Para establecer los nuevos valores de rotación a cada objeto del ambiente se utilizó el siguiente código:

Para manipular el objeto se tiene que hacer referencia a éste:

```
gbTrans_cil_anu2 = myBrowser.getNode("Cil_anular2");
try
{
set_rotation_anu2=(EventInSFRotation) gbTrans_cil_anu2.getEventIn("rotation");
} catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }
```

Donde:

gbTrans_cil_anu2: es un objeto de tipo nodo
set_rotation_anu2: es un EventIn de tipo SFRotation
Cil_anular2: es el nombre del objeto al cual se le asignan estos nuevos valores.
rotation: es la característica que se va a manipular en el objeto.

Para obtener la información que se encuentra en los campos de texto se utilizó el siguiente código:

```
class Escucha_cil_anu3 implements TextListener
    {
        public void textValueChanged(TextEvent e)
        {

            String s=e.paramString();
            if (s.equals("TEXT_VALUE_CHANGED"))
            {

                String texto3=Tex_cil_anu3.getText();
                grados[0]=(Double.valueOf(texto3).doubleValue());
                radianes[0]=( (grados[0] * 3.1416) / 180);
                String valor=(String.valueOf (radianes[0]));
                fin[0]=(Float.valueOf(valor).floatValue());
                rotar_Anu2[3]=fin[0];

            }
        }
    }
```

Con este código se toma el texto que se encuentra en el campo de texto, se convierte de una cadena a un dato de tipo double, con el fin realizar las operaciones correspondientes a la conversión de grados a radianes y posteriormente se almacena en la casilla correspondiente a la rotación en el arreglo rotar_Anu2[3] para finalmente enviarlo al objeto.

Una vez que se cuenta con la información correcta almacenada en los arreglos, se tiene que enviar a los objetos, esto se realiza con la ayuda de un botón. El código correspondiente al botón anular es el siguiente:

```
class EscuchadorBoton_anular implements ActionListener
 {
```

```
public void  actionPerformed(ActionEvent e)
                {
                        String s=e.getActionCommand();
                        if (s.equals("Rotar anular"))
                        {
                                if (rotar_Anu2[3] >= 0 && rotar_Anu2[3] <= 1.5708)
                                        {
                                        set_rotation_anu3.setValue(rotar_Anu2);
                                        }
                        }
                }
}
```

Mediante este código cuando se da clic sobre el botón *Rotar anular* los valores contenidos en el arreglo rotar_Anu2, se asignan al cilindro con el nombre Cil_anular en su campo correspondiente a la rotación utilizando el eventIn set_rotation_anu2.

Finalmente para hacer referencia al modelo virtual y al applet en la página web se utiliza el siguiente código:

```
<EMBED SRC="Mano.wrl" Width=600 Height=450><BR><BR>

<APPLET CODE="Mano.class" Width=800 Height=220>

</APPLET>
```

La figura 5.60 muestra el applet y el modelo virtual:

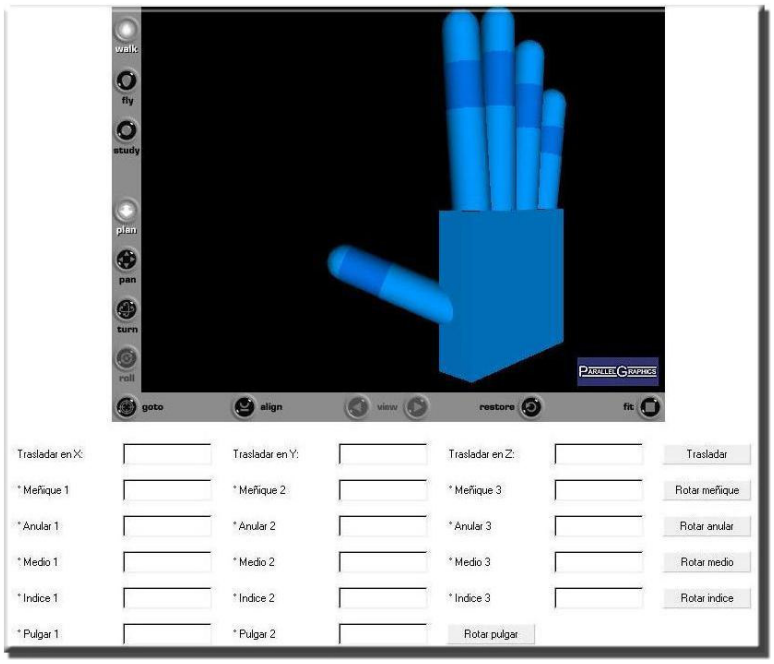


Figura 5.60 Applet con modelo virtual

Como se muestra en la etapa de análisis correspondiente a la metodología, el applet fue de una las tecnologías analizadas para comunicar el guante con el modelo virtual, utilizando JNI para obtener los datos de la DLL del guante; sin embargo, cuando se trato de realizar la comunicación entre las dos partes, se encontraron problemas en la forma en que la PC interpreta ambos códigos, debido a que el código correspondiente al applet es interpretado por el compilador JIT de Microsoft, mientras que el código para obtener los datos del guante es interpretado por el compilador JRE de Sun Microsystems. Por lo tanto solo se pudo interpretar un código a la vez y no ambos como era la idea a desarrollar.

Conclusiones

- Este trabajo presenta una forma distinta de realidad virtual a la que se ha desarrollado anteriormente en la universidad, puesto que cuenta con una interfaz donde se aplica un grado de inmersión mayor al ambiente virtual en VRML, haciendo uso de la tecnología de un dispositivo de entrada de datos como lo es el guante de datos P5.

- Fue posible la obtención de conocimientos sobre distintas tecnologías que engloban el uso de dispositivos externos, tales como los guates de datos, exoesqueletos y otros dispositivos que proporcionan mayor grado de inmersión a la realidad virtual.

- La implementación de un dispositivo externo en un ambiente virtual proporciona mayor sensación de inmersión hacia el usuario, cuando éste interactúa con el ambiente, haciendo que éste utilice más de un sentido proporcionandole una mayor capacidad de aprendizaje.

- La interfaz presenta pequeños retardos al momento de renderizar la escena virtual, presentando una serie de parpadeos visibles, esto debido a la actualización de la información enviada por el guante de datos P5 al archivo wrl que describe el mundo virtual, para mostrarlo nuevamente.

- Además de la interfaz realizada se analizó otra forma de realizar la comunicación entre el dispositivo y la PC, puesto que en un principio se intento desarrollar un módulo de comunicación que enlazara applets desarrollados con Java EAI y las clases de Java JNI del guante de datos P5. En esta investigación se logró desarrollar el applet, el cual responde correctamente a los datos enviados, por otra parte se logró comprender completamente el API que comunica a Java con el guante P5 mediante JNI, faltando la comunicación entre estas dos partes.

- El guante no interactúa con los demás actores del escenario virtual.

    - ✓ No existe un envío de mensaje duplex (ida y vuelta) solo del guante a su representación virtual. Esto debido al modelo del guante, que no cuenta con retroalimentación, para que el usuario pueda sentir los objetos que se encuentran en el ambiente virtual.

Trabajos futuros

- Diseñar un conjunto de objetos virtuales en VRML para poder manipularlos utilizando el guante virtual.

- La continuación de este trabajo corresponde a la implementación de la mano virtual desarrollada, en una práctica virtual, con el fin de que la práctica sea más entendible para el usuario y que éste adquiera los conocimientos para los que fue diseñada de forma más rápida y sencilla, para ésto es necesario ajustar el modelo virtual de la mano con el escenario virtual de la práctica para que ambos interactúen de forma correcta. Este trabajo está siendo desarrollado por un alumno de la maestría en ciencias computacionales del CITIS.

- El uso del dispositivo en este trabajo de investigación, despertó el interés de algunos investigadores en el área de electrónica del CITIS, por lo que propusieron un tema de tesis a nivel maestría, y actualmente se encuentra en desarrollo un proyecto mediante el cual se desea controlar un robot, utilizando el guante de datos P5. De igual manera se están diseñando distintos objetos virtuales en Java3D para su manipulación con este dispositivo.

Anexo A

Código VRML

```
#VRML V2.0 utf8
DEF     Mano Transform
{
        translation         0 -3 -15
        rotation 0 1 0 -1
        children
        [

# ************************** Dedo meñique ***************************************

DEF     Menique Transform
{
        translation         0 0 -1.5
        center 0 -4.25 0
        rotation 1 0 0 -0.2
        children [

DEF Esfera1 Transform
{
        children [
                Shape {
                        appearance Appearance {material          Material {diffuseColor 1 1 0.9 } }
                        geometry Sphere
                                {
                                        radius 0.5
                                }
                        }
                ]
}

DEF Cilindro1 Transform
{
        center 0 -2.75 0
        children [
                Shape {
                        appearance Appearance {material          Material {diffuseColor 1 1 0.9} }
                        geometry Cylinder
```

```
                                        {
                                        radius 0.5
                                        height 5.5
                                }
                        }
                ]
}

DEF Esfera2 Transform
{
        children [
                Shape {
                        appearance Appearance {material         Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.55
                                }
                        }
                ]
}

DEF Cilindro2 Transform
{
        center 0 -1.65 0
        children [
                Shape {
                        appearance Appearance {material         Material {diffuseColor 0 .6 1 } }
                        geometry Cylinder
                                {
                                        radius 0.55
                                        height 3.3
                                }
                        }
                ]
}

DEF Esfera3 Transform
{
        children [
                Shape {
                        appearance Appearance {material         Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.55
                                }
                        }
                ]
}

DEF Cilindro3 Transform
{
        center 0 -0.85 0
        children [
```

```
                Shape {
                        appearance Appearance {material         Material {diffuseColor 0 .6 1 } }
                        geometry Cylinder
                                {
                                        radius 0.53
                                        height 1.7
                                }
                        }
                ]
}

DEF Esfera4 Transform
{
        children [
                Shape {
                        appearance Appearance {material         Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.53
                                }
                        }
                ]
}

DEF Cilindro4 Transform
{
        center 0 -0.85 0
        children [
                Shape {
                        appearance Appearance {material         Material {diffuseColor 0 .6 1 } }
                        geometry Cylinder
                                {
                                        radius 0.5
                                        height 1.7
                                }
                        }
                ]
}

DEF Esfera5 Transform
{
        children [
                Shape {
                        appearance Appearance {material         Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.5
                                }
                        }
                ]
}

# ********** Fin del transforma Menique **********
```

```
 ]
}

# ********************************* Dedo Anular ************************************

DEF     Anular Transform
{
        translation         0 0 -0.55
        center 0 -4 0
        rotation 1 0 0 -0.1
        children
        [

DEF Esfera_A1 Transform
{
        children [
                Shape {
                        appearance Appearance {material          Material {diffuseColor 1 1 0.9 } }
                        geometry Sphere
                                {
                                        radius 0.5
                                }
                        }
                ]
}

DEF Cilindro_A1 Transform
{
        center 0 -2.9 0
        children [
                Shape {
                        appearance Appearance {material          Material {diffuseColor 1 1 0.9} }
                        geometry Cylinder
                                {
                                        radius 0.5
                                        height 5.8
                                }
                        }
                ]
}


DEF Esfera_A2 Transform
{
        children [
                Shape {
                        appearance Appearance {material          Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.65
                                }
                        }
                ]
```

```
}

DEF Cilindro_A2 Transform
{
        center 0 -2.15 0
        children [
                Shape {
                        appearance Appearance {material        Material {diffuseColor 0 .6 1 } }
                        geometry Cylinder
                                {
                                        radius 0.65
                                        height 4.3
                                }
                        }
                ]
}

DEF Esfera_A3 Transform
{
        children [
                Shape {
                        appearance Appearance {material        Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.65
                                }
                        }
                ]
}

DEF Cilindro_A3 Transform
{
        center 0 -1.25 0
        children [
                Shape {
                        appearance Appearance {material        Material {diffuseColor 0 .6 1} }
                        geometry Cylinder
                                {
                                        radius 0.63
                                        height 2.5
                                }
                        }
                ]
}

DEF Esfera_A4 Transform
{
        children [
                Shape {
                        appearance Appearance {material        Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.63
```

```
                                }
                        }
                ]
}

DEF Cilindro_A4 Transform
{
        center 0 -0.95 0
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 0 .6 1} }
                        geometry Cylinder
                                {
                                        radius 0.6
                                        height 1.9
                                }
                        }
                ]
}

DEF Esfera_A5 Transform
{
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.6
                                }
                        }
                ]
}

# ***************************** Fin del transform Anular ********************************
 ]
}

# ********************************* Dedo Medio ************************************

DEF     Medio Transform
{
        translation     0 0 0.4
        center 0 -4 0
        children
        [

DEF Esfera_M1 Transform
{
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 1 1 0.9 } }
                        geometry Sphere
                                {
```

```
                                        radius 0.5
                                }
                        }
                ]
}

DEF Cilindro_M1 Transform
{
        center 0 -3.25 0
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 1 1 0.9 } }
                        geometry Cylinder
                                {
                                        radius 0.5
                                        height 6.5
                                }
                        }
                ]
}

DEF Esfera_M2 Transform
{
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.64
                                }
                        }
                ]
}

DEF Cilindro_M2 Transform
{
        center 0 -2.25 0
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 0 .6 1} }
                        geometry Cylinder
                                {
                                        radius 0.64
                                        height 4.5
                                }
                        }
                ]
}

DEF Esfera_M3 Transform
{
        children [
                Shape {
```

```
                    appearance Appearance {material      Material {diffuseColor 0 .6 1 } }
                    geometry Sphere
                        {
                            radius 0.64
                        }
                    }
                ]
}

DEF Cilindro_M3 Transform
{
        center 0 -1.35 0
        children [
                Shape {
                    appearance Appearance {material      Material {diffuseColor 0 .6 1} }
                    geometry Cylinder
                        {
                            radius 0.61
                            height 2.7
                        }
                    }
                ]
}

DEF Esfera_M4 Transform
{
        children [
                Shape {
                    appearance Appearance {material      Material {diffuseColor 0 .6 1 } }
                    geometry Sphere
                        {
                            radius 0.61
                        }
                    }
                ]
}

DEF Cilindro_M4 Transform
{
        center 0 -0.9 0
        children [
                Shape {
                    appearance Appearance {material      Material {diffuseColor 0 .6 1} }
                    geometry Cylinder
                        {
                            radius 0.59
                            height 1.8
                        }
                    }
                ]
}

DEF Esfera_M5 Transform
```

```
{
        children [
                Shape {
                        appearance Appearance {material         Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.59
                                }
                        }
                ]
}

# ***************************** Fin del transform Medio ********************************
]
}


# ********************************* Dedo Indice ************************************

DEF Indice Transform
{
        translation     0 0 1.35
        center 0 -4 0
        rotation 1 0 0 0.1
        children
        [

DEF Esfera_I1 Transform
{
        children [
                Shape {
                        appearance Appearance {material         Material {diffuseColor 1 1 0.9 } }
                        geometry Sphere
                                {
                                        radius 0.5
                                }
                        }
                ]
}

DEF Cilindro_I1 Transform
{
        center 0 -3.35 0
        children [
                Shape {
                        appearance Appearance {material         Material {diffuseColor 1 1 0.9 } }
                        geometry Cylinder
                                {
                                        radius 0.5
                                        height 6.7
                                }
                        }
                ]
```

```
}

DEF Esfera_I2 Transform
{
        children [
                Shape {
                        appearance Appearance {material        Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.63
                                }
                        }
                ]
}

DEF Cilindro_I2 Transform
{
        center 0 -1.95 0
        children [
                Shape {
                        appearance Appearance {material        Material {diffuseColor 0 .6 1} }
                        geometry Cylinder
                                {
                                        radius 0.63
                                        height 3.9
                                }
                        }
                ]
}

DEF Esfera_I3 Transform
{
        children [
                Shape {
                        appearance Appearance {material        Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.63
                                }
                        }
                ]
}

DEF Cilindro_I3 Transform
{
        center 0 -1.1 0
        children [
                Shape {
                        appearance Appearance {material        Material {diffuseColor 0 .6 1} }
                        geometry Cylinder
                                {
                                        radius 0.61
                                        height 2.2
```

```
                                        }
                                }
                        ]
}

DEF Esfera_I4 Transform
{
        children [
                Shape {
                        appearance Appearance {material        Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.61
                                }
                        }
                ]
}

DEF Cilindro_I4 Transform
{
        center 0 -0.85 0
        children [
                Shape {
                        appearance Appearance {material        Material {diffuseColor 0 .6 1} }
                        geometry Cylinder
                                {
                                        radius 0.58
                                        height 1.7
                                }
                        }
                ]
}

DEF Esfera_I5 Transform
{
        children [
                Shape {
                        appearance Appearance {material        Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.58
                                }
                        }
                ]
}

# ********************************* Fin del transform Indice ********************************
]
}

#********************************* Dedo Pulgar ***************************************

DEF     Pulgar2 Transform
```

```
{
        translation 0 0 2.4
        rotation 1 0 0 0.6
        center 0 -4 0
        children [

DEF     Pulgar1 Transform
{
        rotation 0 0 1 1
        center 0 -4 0
        children [

DEF Pulgar Transform
{
        center 0 -4 0
        rotation 0 1 0 -1.5
        children [

DEF Esfera_P1 Transform
{
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 1 1 0.9 } }
                        geometry Sphere
                                {
                                        radius 0.5
                                }
                        }
                ]
}

DEF Cilindro_P1 Transform
{
        center 0 -2.25 0
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 1 1 0.9 } }
                        geometry Cylinder
                                {
                                        radius 0.5
                                        height 4.5
                                }
                        }
                ]
}

DEF Esfera_P2 Transform
{
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
```

```
                                        radius 0.67
                                }
                        }
                ]
}

DEF Cilindro_P2 Transform
{
        center 0 -1.5 0
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 0 .6 1 } }
                        geometry Cylinder
                                {
                                        radius 0.67
                                        height 3.0
                                }
                        }
                ]
}

DEF Esfera_P3 Transform
{
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.67
                                }
                        }
                ]
}

DEF Cilindro_P3 Transform
{
        center 0 -1.15 0
        children [
                Shape {
                        appearance Appearance {material      Material {diffuseColor 0 .6 1 } }
                        geometry Cylinder
                                {
                                        radius 0.67
                                        height 2.3
                                }
                        }
                ]
}

DEF Esfera_P4 Transform
{
        children [
                Shape {
```

```
                        appearance Appearance {material          Material {diffuseColor 0 .6 1 } }
                        geometry Sphere
                                {
                                        radius 0.67
                                }
                        }
                ]
}

]
}
]
}

# ******************************** Fin del transform Pulgar ******************************
]
}

#********************************* Fin del transform Mano ***************************
]
}

#************************************ Script ***************************************

DEF Reloj TimeSensor
{
   cycleInterval 10.0
   loop TRUE
   startTime 1.0
   stopTime 0.0
}


DEF Modelo Script
{
        eventIn SFFloat CalculaModelo

# ************************ EventOut para el dedo meñique ******************************

        eventOut SFVec3f Traslada_changed1
        eventOut SFVec3f Traslada_changed2
        eventOut SFVec3f Traslada_changed3
        eventOut SFVec3f Traslada_changed4
        eventOut SFVec3f Traslada_changed5
        eventOut SFVec3f TrasladaC1_changed
        eventOut SFVec3f TrasladaC2_changed
        eventOut SFVec3f TrasladaC3_changed
        eventOut SFVec3f TrasladaC4_changed
        eventOut SFRotation RotaC1_changed
        eventOut SFRotation RotaC2_changed
        eventOut SFRotation RotaC3_changed
        eventOut SFRotation RotaCil4_changed
```

```
# ********** EventOut para el dedo anular * **********

        eventOut SFVec3f TrasladaA_changed1
        eventOut SFVec3f TrasladaA_changed2
        eventOut SFVec3f TrasladaA_changed3
        eventOut SFVec3f TrasladaA_changed4
        eventOut SFVec3f TrasladaA_changed5
        eventOut SFVec3f TrasladaC1A_changed
        eventOut SFVec3f TrasladaC2A_changed
        eventOut SFVec3f TrasladaC3A_changed
        eventOut SFVec3f TrasladaC4A_changed
        eventOut SFRotation RotaC1A_changed
        eventOut SFRotation RotaC2A_changed
        eventOut SFRotation RotaC3A_changed
        eventOut SFRotation RotaCil4A_changed

# ********** EventOut para el dedo medio * **********

        eventOut SFVec3f TrasladaM_changed1
        eventOut SFVec3f TrasladaM_changed2
        eventOut SFVec3f TrasladaM_changed3
        eventOut SFVec3f TrasladaM_changed4
        eventOut SFVec3f TrasladaM_changed5
        eventOut SFVec3f TrasladaC1M_changed
        eventOut SFVec3f TrasladaC2M_changed
        eventOut SFVec3f TrasladaC3M_changed
        eventOut SFVec3f TrasladaC4M_changed
        eventOut SFRotation RotaC1M_changed
        eventOut SFRotation RotaC2M_changed
        eventOut SFRotation RotaC3M_changed
        eventOut SFRotation RotaCil4M_changed

# ********** EventOut para el dedo índice * **********

        eventOut SFVec3f TrasladaI_changed1
        eventOut SFVec3f TrasladaI_changed2
        eventOut SFVec3f TrasladaI_changed3
        eventOut SFVec3f TrasladaI_changed4
        eventOut SFVec3f TrasladaI_changed5
        eventOut SFVec3f TrasladaC1I_changed
        eventOut SFVec3f TrasladaC2I_changed
        eventOut SFVec3f TrasladaC3I_changed
        eventOut SFVec3f TrasladaC4I_changed
        eventOut SFRotation RotaC1I_changed
        eventOut SFRotation RotaC2I_changed
        eventOut SFRotation RotaC3I_changed
        eventOut SFRotation RotaCil4I_changed

# ********** EventOut para el dedo pulgar * **********

        eventOut SFVec3f TrasladaP_changed1
        eventOut SFVec3f TrasladaP_changed2
        eventOut SFVec3f TrasladaP_changed3
```

```
eventOut SFVec3f TrasladaP_changed4
eventOut SFVec3f TrasladaC1P_changed
eventOut SFVec3f TrasladaC2P_changed
eventOut SFVec3f TrasladaC3P_changed
eventOut SFRotation RotaC1P_changed
eventOut SFRotation RotaC2P_changed
eventOut SFRotation RotaC3P_changed


url     "Javascript:

function initialize()
{
        /**************  Valores de los sensores ******************/
        menique=46
        anular=33
        medio=34
        indice=4
        pulgar=33

        /* **********Convertir los valores de cada sensor de 0-63 en ángulos
                        para los objetos en el ambiente virtual ***************/

                /**** Meñique ****/
                t1=(menique*90)/63;
                t2=(menique*90)/63;
                t3=(menique*90)/63;

                /**** Anular ****/
                tA1=(anular*90)/63;
                tA2=(anular*90)/63;
                tA3=(anular*90)/63;

                /**** Medio ****/
                tM1=(medio*90)/63;
                tM2=(medio*90)/63;
                tM3=(medio*90)/63;

                /**** Indice ****/
                tI1=(indice*90)/63;
                tI2=(indice*90)/63;
                tI3=(indice*90)/63;

                /**** Pulgar ****/
                tP1=(pulgar*90)/63;
                tP2=(pulgar*90)/63;
}

function CalculaModelo()
{

/***************************** Dedo Meñique ****************************************/
/**************Convertir los ángulos en radianes para evaluarlos en el modelo ****************/
```

```
        angulo1=(Math.PI*(0+90))/180;
        angulo2=(Math.PI*t1)/180;
        angulo3=(Math.PI*t2)/180;
        angulo4=(Math.PI*t3)/180;


/****Calcular el modelo cinematico para encontrar la posición de cada esfera del dedo meñique ****/

        X1=0;
        Y1=0;
        X2=5.5*(Math.cos(angulo1));
        Y2=5.5*(Math.sin(angulo1));
        X3=X2+(3.3*(Math.cos((angulo1+angulo2))));
        Y3=Y2+(3.3*(Math.sin((angulo1+angulo2))));
        X4=X3+(1.7*(Math.cos((angulo1+angulo2+angulo3))));
        Y4=Y3+(1.7*(Math.sin((angulo1+angulo2+angulo3))));
        X=X4+(1.7*(Math.cos((angulo1+angulo2+angulo3+angulo4))));
        Y=Y4+(1.7*(Math.sin((angulo1+angulo2+angulo3+angulo4))));
        dif_ang=(Math.PI*90)/180;


/**** Asignar a cada eje su nueva osición de acuerdo a los calculos obtenidos en el modelo ****/

/*************** Eslabon 1 ***************************/
        Traslada_changed1[0]=X1;
        Traslada_changed1[1]=Y1-4;
        Traslada_changed1[2]=0;

        TrasladaC1_changed[0]=X1;
        TrasladaC1_changed[1]=Y1-1.3;
        TrasladaC1_changed[2]=0;

        RotaC1_changed[0]=1;
        RotaC1_changed[1]=0;
        RotaC1_changed[2]=0;
        RotaC1_changed[3]=angulo1-dif_ang;

/*************** Eslabon 2 ***************************/

        Traslada_changed2[0]=X2;
        Traslada_changed2[1]=Y2-4;
        Traslada_changed2[2]=0;

        TrasladaC2_changed[0]=X2;
        TrasladaC2_changed[1]=Y2-2.4;
        TrasladaC2_changed[2]=0;

        RotaC2_changed[0]=0;
        RotaC2_changed[1]=0;
        RotaC2_changed[2]=1;
        RotaC2_changed[3]=(angulo1 +angulo2)-dif_ang;
```

```
/*************** Eslabon 3 **************************/

        Traslada_changed3[0]=X3;
        Traslada_changed3[1]=Y3-4.01;
        Traslada_changed3[2]=0;

        TrasladaC3_changed[0]=X3;
        TrasladaC3_changed[1]=Y3-3.2;
        TrasladaC3_changed[2]=0;
        RotaC3_changed[0]=0;
        RotaC3_changed[1]=0;
        RotaC3_changed[2]=1;
        RotaC3_changed[3]=(angulo1+angulo2+angulo3)-dif_ang;

/*************** Eslabon 4 **************************/

        Traslada_changed4[0]=X4;
        Traslada_changed4[1]=Y4-4.01;
        Traslada_changed4[2]=0;
        TrasladaC4_changed[0]=X4;
        TrasladaC4_changed[1]=Y4-3.2;
        TrasladaC4_changed[2]=0;

        RotaCil4_changed[0]=0;
        RotaCil4_changed[1]=0;
        RotaCil4_changed[2]=1;
        RotaCil4_changed[3]=(angulo1+angulo2+angulo3+angulo4)-dif_ang;

/*************** Esfera 5 **************************/

        Traslada_changed5[0]=X;
        Traslada_changed5[1]=Y-4.05;
        Traslada_changed5[2]=0;

/***************************** Dedo amular ****************************************/

/**** Calcular el modelo cinematico para encontrar la posición de cada esfera  ****/

        angulo_A1=(Math.PI*(0+90))/180;
        angulo_A2=(Math.PI*tA1)/180;
        angulo_A3=(Math.PI*tA2)/180;
        angulo_A4=(Math.PI*tA3)/180;

        XA1=0;
        YA1=0;
        XA2=5.8*(Math.cos(angulo_A1));
        YA2=5.8*(Math.sin(angulo_A1));
        XA3=XA2+(4.3*(Math.cos((angulo_A1+angulo_A2))));
        YA3=YA2+(4.3*(Math.sin((angulo_A1+angulo_A2))));
        XA4=XA3+(2.5*(Math.cos((angulo_A1+angulo_A2+angulo_A3))));
        YA4=YA3+(2.5*(Math.sin((angulo_A1+angulo_A2+angulo_A3))));
        XA=XA4+(1.9*(Math.cos((angulo_A1+angulo_A2+angulo_A3+angulo_A4))));
        YA=YA4+(1.9*(Math.sin((angulo_A1+angulo_A2+angulo_A3+angulo_A4))));
```

```
/***Asignar a cada eje su nueva posición de acuerdo a los cálculos obtenidos en el modelo ****/

/*************** Eslabon 1 ***************************/

        TrasladaA_changed1[0]=XA1;
        TrasladaA_changed1[1]=YA1-4;
        TrasladaA_changed1[2]=0;

        TrasladaC1A_changed[0]=XA1;
        TrasladaC1A_changed[1]=YA1-1.15;
        TrasladaC1A_changed[2]=0;

        RotaC1A_changed[0]=1;
        RotaC1A_changed[1]=0;
        RotaC1A_changed[2]=0;
        RotaC1A_changed[3]=angulo_A1-dif_ang;

/*************** Eslabon 2 ***************************/

        TrasladaA_changed2[0]=XA2;
        TrasladaA_changed2[1]=YA2-4;
        TrasladaA_changed2[2]=0;

        TrasladaC2A_changed[0]=XA2;
        TrasladaC2A_changed[1]=YA2-1.91;
        TrasladaC2A_changed[2]=0;

        RotaC2A_changed[0]=0;
        RotaC2A_changed[1]=0;
        RotaC2A_changed[2]=1;
        RotaC2A_changed[3]=(angulo_A1 +angulo_A2)-dif_ang;

/*************** Eslabon 3 ***************************/

        TrasladaA_changed3[0]=XA3;
        TrasladaA_changed3[1]=YA3-4.02;
        TrasladaA_changed3[2]=0;

        TrasladaC3A_changed[0]=XA3;
        TrasladaC3A_changed[1]=YA3-2.82;
        TrasladaC3A_changed[2]=0;

        RotaC3A_changed[0]=0;
        RotaC3A_changed[1]=0;
        RotaC3A_changed[2]=1;
        RotaC3A_changed[3]=(angulo_A1+angulo_A2+angulo_A3)-dif_ang;

/*************** Eslabon 4 ***************************/

        TrasladaA_changed4[0]=XA4;
        TrasladaA_changed4[1]=YA4-4.05;
        TrasladaA_changed4[2]=0;
```

```
        TrasladaC4A_changed[0]=XA4;
        TrasladaC4A_changed[1]=YA4-3.15;
        TrasladaC4A_changed[2]=0;

        RotaCil4A_changed[0]=0;
        RotaCil4A_changed[1]=0;
        RotaCil4A_changed[2]=1;
                                RotaCil4A_changed[3]=(angulo_A1+angulo_A2+angulo_A3+angulo_A4)-
dif_ang;

/********** Esfera 5 **********/

        TrasladaA_changed5[0]=XA;
        TrasladaA_changed5[1]=YA-4.1;
        TrasladaA_changed5[2]=0;

/********** Dedo Medio **********/

/**** Calcular el modelo cinematico para encontrar la posición de cada esfera  ****/

        angulo_M1=(Math.PI*(0+90))/180;
        angulo_M2=(Math.PI*tM1)/180;
        angulo_M3=(Math.PI*tM2)/180;
        angulo_M4=(Math.PI*tM3)/180;

        XM1=0;
        YM1=0;
        XM2=6.5*(Math.cos(angulo_M1));
        YM2=6.5*(Math.sin(angulo_M1));
        XM3=XM2+(4.5*(Math.cos((angulo_M1+angulo_M2))));
        YM3=YM2+(4.5*(Math.sin((angulo_M1+angulo_M2))));
        XM4=XM3+(2.7*(Math.cos((angulo_M1+angulo_M2+angulo_M3))));
        YM4=YM3+(2.7*(Math.sin((angulo_M1+angulo_M2+angulo_M3))));
        XM=XM4+(1.8*(Math.cos((angulo_M1+angulo_M2+angulo_M3+angulo_M4))));
        YM=YM4+(1.8*(Math.sin((angulo_M1+angulo_M2+angulo_M3+angulo_M4))));


/**** Asignar a cada eje su nueva posición de acuerdo a los cálculos obtenidos en el modelo ****/

/********** Eslabon 1 **********/

        TrasladaM_changed1[0]=XM1;
        TrasladaM_changed1[1]=YM1-4;
        TrasladaM_changed1[2]=0;

        TrasladaC1M_changed[0]=XM1;
        TrasladaC1M_changed[1]=YM1-0.8;
        TrasladaC1M_changed[2]=0;

        RotaC1M_changed[0]=1;
        RotaC1M_changed[1]=0;
        RotaC1M_changed[2]=0;
```

```
        RotaC1M_changed[3]=angulo_M1-dif_ang;

/*************** Eslabon 2 ***************************/

        TrasladaM_changed2[0]=XM2;
        TrasladaM_changed2[1]=YM2-4;
        TrasladaM_changed2[2]=0;

        TrasladaC2M_changed[0]=XM2;
        TrasladaC2M_changed[1]=YM2-1.8;
        TrasladaC2M_changed[2]=0;

        RotaC2M_changed[0]=0;
        RotaC2M_changed[1]=0;
        RotaC2M_changed[2]=1;
        RotaC2M_changed[3]=(angulo_M1 +angulo_M2)-dif_ang;

/*************** Eslabon 3 ***************************/

        TrasladaM_changed3[0]=XM3;
        TrasladaM_changed3[1]=YM3-4.01;
        TrasladaM_changed3[2]=0;

        TrasladaC3M_changed[0]=XM3;
        TrasladaC3M_changed[1]=YM3-2.7;
        TrasladaC3M_changed[2]=0;

        RotaC3M_changed[0]=0;
        RotaC3M_changed[1]=0;
        RotaC3M_changed[2]=1;
        RotaC3M_changed[3]=(angulo_M1+angulo_M2+angulo_M3)-dif_ang;

/*************** Eslabon 4 ***************************/

        TrasladaM_changed4[0]=XM4;
        TrasladaM_changed4[1]=YM4-4.05;
        TrasladaM_changed4[2]=0;

        TrasladaC4M_changed[0]=XM4;
        TrasladaC4M_changed[1]=YM4-3.2;
        TrasladaC4M_changed[2]=0;

        RotaCil4M_changed[0]=0;
        RotaCil4M_changed[1]=0;
        RotaCil4M_changed[2]=1;
        RotaCil4M_changed[3]=(angulo_M1+angulo_M2+angulo_M3+angulo_M4)-dif_ang;

/*************** Esfera 5 ***************************/

        TrasladaM_changed5[0]=XM;
        TrasladaM_changed5[1]=YM-4.1;
        TrasladaM_changed5[2]=0;
```

```
/****************************** Dedo Indice ****************************************/

/**** Calcular el modelo cinematico para encontrar la posición de cada esfera  ****/

        angulo_I1=(Math.PI*(0+90))/180;
        angulo_I2=(Math.PI*tI1)/180;
        angulo_I3=(Math.PI*tI2)/180;
        angulo_I4=(Math.PI*tI3)/180;

        XI1=0;
        YI1=0;
        XI2=6.7*(Math.cos(angulo_I1));
        YI2=6.7*(Math.sin(angulo_I1));
        XI3=XI2+(3.9*(Math.cos((angulo_I1+angulo_I2))));
        YI3=YI2+(3.9*(Math.sin((angulo_I1+angulo_I2))));
        XI4=XI3+(2.2*(Math.cos((angulo_I1+angulo_I2+angulo_I3))));
        YI4=YI3+(2.2*(Math.sin((angulo_I1+angulo_I2+angulo_I3))));
        XI=XI4+(1.7*(Math.cos((angulo_I1+angulo_I2+angulo_I3+angulo_I4))));
        YI=YI4+(1.7*(Math.sin((angulo_I1+angulo_I2+angulo_I3+angulo_I4))));

/****Asignar a cada eje su nueva osición de acuerdo a los calculos obtenidos en el modelo ****/

/*************** Eslabon 1 ***************************/

        TrasladaI_changed1[0]=XI1;
        TrasladaI_changed1[1]=YI1-4;
        TrasladaI_changed1[2]=0;

        TrasladaC1I_changed[0]=XI1;
        TrasladaC1I_changed[1]=YI1-0.7;
        TrasladaC1I_changed[2]=0;

        RotaC1I_changed[0]=1;
        RotaC1I_changed[1]=0;
        RotaC1I_changed[2]=0;
        RotaC1I_changed[3]=angulo_I1-dif_ang;

/*************** Eslabon 2 ***************************/

        TrasladaI_changed2[0]=XI2;
        TrasladaI_changed2[1]=YI2-4;
        TrasladaI_changed2[2]=0;

        TrasladaC2I_changed[0]=XI2;
        TrasladaC2I_changed[1]=YI2-2.1;
        TrasladaC2I_changed[2]=0;

        RotaC2I_changed[0]=0;
        RotaC2I_changed[1]=0;
        RotaC2I_changed[2]=1;
        RotaC2I_changed[3]=(angulo_I1 +angulo_I2)-dif_ang;
```

```
/*************** Eslabon 3 **************************/

        TrasladaI_changed3[0]=XI3;
        TrasladaI_changed3[1]=YI3-4.01;
        TrasladaI_changed3[2]=0;

        TrasladaC3I_changed[0]=XI3;
        TrasladaC3I_changed[1]=YI3-2.95;
        TrasladaC3I_changed[2]=0;

        RotaC3I_changed[0]=0;
        RotaC3I_changed[1]=0;
        RotaC3I_changed[2]=1;
        RotaC3I_changed[3]=(angulo_I1+angulo_I2+angulo_I3)-dif_ang;

/*************** Eslabon 4 **************************/

        TrasladaI_changed4[0]=XI4;
        TrasladaI_changed4[1]=YI4-4.01;
        TrasladaI_changed4[2]=0;

        TrasladaC4I_changed[0]=XI4;
        TrasladaC4I_changed[1]=YI4-3.2;
        TrasladaC4I_changed[2]=0;

        RotaCil4I_changed[0]=0;
        RotaCil4I_changed[1]=0;
        RotaCil4I_changed[2]=1;
        RotaCil4I_changed[3]=(angulo_I1+angulo_I2+angulo_I3+angulo_I4)-dif_ang;

/*************** Esfera 5 **************************/

        TrasladaI_changed5[0]=XI;
        TrasladaI_changed5[1]=YI-4.05;
        TrasladaI_changed5[2]=0;


/***************************** Dedo Pulgar ***************************************/

/**** Calcular el modelo cinematico para encontrar la posición de cada esfera  ****/

        angulo_P1=(Math.PI*(0+90))/180;
        angulo_P2=(Math.PI*tP1)/180;
        angulo_P3=(Math.PI*tP2)/180;

        XP1=0;
        YP1=0;
        XP2=4.5*(Math.cos(angulo_P1));
        YP2=4.5*(Math.sin(angulo_P1));
        XP3=XP2+(3.0*(Math.cos((angulo_P1+angulo_P2))));
        YP3=YP2+(3.0*(Math.sin((angulo_P1+angulo_P2))));
        XP=XP3+(2.3*(Math.cos((angulo_P1+angulo_P2+angulo_P3))));
        YP=YP3+(2.3*(Math.sin((angulo_P1+angulo_P2+angulo_P3))));
```

```
/****Asignar a cada eje su nueva posición de acuerdo a los cálculos obtenidos en el modelo ****/

/*************** Eslabon 1 ***************************/

        TrasladaP_changed1[0]=XP1;
        TrasladaP_changed1[1]=YP1-4;
        TrasladaP_changed1[2]=0;

        TrasladaC1P_changed[0]=XP1;
        TrasladaC1P_changed[1]=YP1-1.8;
        TrasladaC1P_changed[2]=0;

        RotaC1P_changed[0]=1;
        RotaC1P_changed[1]=0;
        RotaC1P_changed[2]=0;
        RotaC1P_changed[3]=angulo_P1-dif_ang;

/*************** Eslabon 2 ***************************/

        TrasladaP_changed2[0]=XP2;
        TrasladaP_changed2[1]=YP2-4;
        TrasladaP_changed2[2]=0;

        TrasladaC2P_changed[0]=XP2;
        TrasladaC2P_changed[1]=YP2-2.55;
        TrasladaC2P_changed[2]=0;

        RotaC2P_changed[0]=0;
        RotaC2P_changed[1]=0;
        RotaC2P_changed[2]=1;
        RotaC2P_changed[3]=(angulo_P1 +angulo_P2)-dif_ang;

/*************** Eslabon 3 ***************************/

        TrasladaP_changed3[0]=XP3;
        TrasladaP_changed3[1]=YP3-4.01;
        TrasladaP_changed3[2]=0;

        TrasladaC3P_changed[0]=XP3;
        TrasladaC3P_changed[1]=YP3-2.9;
        TrasladaC3P_changed[2]=0;

        RotaC3P_changed[0]=0;
        RotaC3P_changed[1]=0;
        RotaC3P_changed[2]=1;
        RotaC3P_changed[3]=(angulo_P1+angulo_P2+angulo_P3)-dif_ang;

/*************** Eslabon 4 ***************************/

        TrasladaP_changed4[0]=XP;
        TrasladaP_changed4[1]=YP-4.05;
        TrasladaP_changed4[2]=0;
```

```
        }
  "
}

#******************************** ASIGNACIÓN DE RUTAS ************************

ROUTE Reloj.fraction_changed  TO       Modelo.CalculaModelo

#******************** Asignar las rutas al dedo meñique ***********************

ROUTE Modelo.Traslada_changed1       TO      Esfera1.translation
ROUTE Modelo.Traslada_changed2       TO      Esfera2.translation
ROUTE Modelo.Traslada_changed3       TO      Esfera3.translation
ROUTE Modelo.Traslada_changed4       TO      Esfera4.translation
ROUTE Modelo.Traslada_changed5       TO      Esfera5.translation
ROUTE Modelo.TrasladaC1_changed      TO      Cilindro1.translation
ROUTE Modelo.TrasladaC2_changed      TO      Cilindro2.translation
ROUTE Modelo.TrasladaC3_changed      TO      Cilindro3.translation
ROUTE Modelo.TrasladaC4_changed      TO      Cilindro4.translation
ROUTE Modelo.RotaC1_changed          TO      Cilindro1.rotation
ROUTE Modelo.RotaC2_changed          TO      Cilindro2.rotation
ROUTE Modelo.RotaC3_changed          TO      Cilindro3.rotation
ROUTE Modelo.RotaCil4_changed        TO      Cilindro4.rotation

#******************** Asignar las rutas al dedo anular ************************

ROUTE Modelo.TrasladaA_changed1      TO      Esfera_A1.translation
ROUTE Modelo.TrasladaA_changed2      TO      Esfera_A2.translation
ROUTE Modelo.TrasladaA_changed3      TO      Esfera_A3.translation
ROUTE Modelo.TrasladaA_changed4      TO      Esfera_A4.translation
ROUTE Modelo.TrasladaA_changed5      TO      Esfera_A5.translation
ROUTE Modelo.TrasladaC1A_changed  TO      Cilindro_A1.translation
ROUTE Modelo.TrasladaC2A_changed  TO      Cilindro_A2.translation
ROUTE Modelo.TrasladaC3A_changed  TO      Cilindro_A3.translation
ROUTE Modelo.TrasladaC4A_changed  TO      Cilindro_A4.translation
ROUTE Modelo.RotaC1A_changed         TO      Cilindro_A1.rotation
ROUTE Modelo.RotaC2A_changed         TO      Cilindro_A2.rotation
ROUTE Modelo.RotaC3A_changed         TO      Cilindro_A3.rotation
ROUTE Modelo.RotaCil4A_changed       TO      Cilindro_A4.rotation

#******************** Asignar las rutas al dedo medio *************************

ROUTE Modelo.TrasladaM_changed1      TO      Esfera_M1.translation
ROUTE Modelo.TrasladaM_changed2      TO      Esfera_M2.translation
ROUTE Modelo.TrasladaM_changed3      TO      Esfera_M3.translation
ROUTE Modelo.TrasladaM_changed4      TO      Esfera_M4.translation
ROUTE Modelo.TrasladaM_changed5      TO      Esfera_M5.translation
ROUTE Modelo.TrasladaC1M_changed  TO      Cilindro_M1.translation
ROUTE Modelo.TrasladaC2M_changed  TO      Cilindro_M2.translation
ROUTE Modelo.TrasladaC3M_changed  TO      Cilindro_M3.translation
ROUTE Modelo.TrasladaC4M_changed  TO      Cilindro_M4.translation
```

```
ROUTE Modelo.RotaC1M_changed        TO    Cilindro_M1.rotation
ROUTE Modelo.RotaC2M_changed        TO    Cilindro_M2.rotation
ROUTE Modelo.RotaC3M_changed        TO    Cilindro_M3.rotation
ROUTE Modelo.RotaCil4M_changed      TO    Cilindro_M4.rotation

#******************** Asignar las rutas al dedo índice ***************************

ROUTE Modelo.TrasladaI_changed1     TO    Esfera_I1.translation
ROUTE Modelo.TrasladaI_changed2     TO    Esfera_I2.translation
ROUTE Modelo.TrasladaI_changed3     TO    Esfera_I3.translation
ROUTE Modelo.TrasladaI_changed4     TO    Esfera_I4.translation
ROUTE Modelo.TrasladaI_changed5     TO    Esfera_I5.translation
ROUTE Modelo.TrasladaC1I_changed    TO    Cilindro_I1.translation
ROUTE Modelo.TrasladaC2I_changed    TO    Cilindro_I2.translation
ROUTE Modelo.TrasladaC3I_changed    TO    Cilindro_I3.translation
ROUTE Modelo.TrasladaC4I_changed    TO    Cilindro_I4.translation
ROUTE Modelo.RotaC1I_changed        TO    Cilindro_I1.rotation
ROUTE Modelo.RotaC2I_changed        TO    Cilindro_I2.rotation
ROUTE Modelo.RotaC3I_changed        TO    Cilindro_I3.rotation
ROUTE Modelo.RotaCil4I_changed      TO    Cilindro_I4.rotation

#******************** Asignar las rutas al dedo pulgar ***************************

ROUTE Modelo.TrasladaP_changed1     TO    Esfera_P1.translation
ROUTE Modelo.TrasladaP_changed2     TO    Esfera_P2.translation
ROUTE Modelo.TrasladaP_changed3     TO    Esfera_P3.translation
ROUTE Modelo.TrasladaP_changed4     TO    Esfera_P4.translation
ROUTE Modelo.TrasladaC1P_changed    TO    Cilindro_P1.translation
ROUTE Modelo.TrasladaC2P_changed    TO    Cilindro_P2.translation
ROUTE Modelo.TrasladaC3P_changed    TO    Cilindro_P3.translation
ROUTE Modelo.RotaC1P_changed        TO    Cilindro_P1.rotation
ROUTE Modelo.RotaC2P_changed        TO    Cilindro_P2.rotation
ROUTE Modelo.RotaC3P_changed        TO    Cilindro_P3.rotation
```

Anexo B

Código Delphi

Código del formulario del programa Panel de sensores.

```
object Form1: TForm1
  Left = 387
  Top = 204
  BorderStyle = bsToolWindow
  Caption = 'Sensores'
  ClientHeight = 254
  ClientWidth = 275
  Color = clInactiveBorder
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clActiveBorder
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
    OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object SCButton1: TSCButton
    Left = 200
    Top = 224
    Width = 72
    Height = 25
    Caption = 'Salir'
    Font.Charset = ANSI_CHARSET
    Font.Color = clNavy
    Font.Height = -12
    Font.Name = 'Georgia'
    Font.Style = []
    HighlightColor = clActiveBorder
    HottrackColor = clAppWorkSpace
    ParentFont = False
    RoundColor = clInactiveBorder
    Style = scbsXP
    TabOrder = 0
    OnClick = SCButton1Click
  end
  object Pulgar: TSCProgress
```

```
    Left = 72
    Top = 105
    Width = 22
    Height = 113
    BorderProps.Color = clSilver
    BorderProps.Width = 1
    Max = 63
    Orientation = scoVertical
    PositionColor = clBackground
    Style = scpsXP
    TabOrder = 1
    OnChange = PulgarChange
  end
  object Indice: TSCProgress
    Left = 96
    Top = 56
    Width = 22
    Height = 162
    BorderProps.Color = clSilver
    BorderProps.Width = 1
    Max = 63
    Orientation = scoVertical
    PositionColor = clBackground
    Style = scpsXP
    TabOrder = 2
  end
  object Medio: TSCProgress
    Left = 120
    Top = 40
    Width = 22
    Height = 178
    BorderProps.Color = clSilver
    BorderProps.Width = 1
    Max = 63
    Orientation = scoVertical
    PositionColor = clBackground
    Style = scpsXP
    TabOrder = 3
  end
  object Anular: TSCProgress
    Left = 144
    Top = 48
    Width = 22
    Height = 170
    BorderProps.Color = clSilver
    BorderProps.Width = 1
    Max = 63
    Orientation = scoVertical
    PositionColor = clBackground
    Style = scpsXP
    TabOrder = 4
  end
  object Menique: TSCProgress
```

```
    Left = 168
    Top = 80
    Width = 22
    Height = 138
    BorderProps.Color = clSilver
    BorderProps.Width = 1
    Max = 63
    Orientation = scoVertical
    PositionColor = clBackground
    Style = scpsXP
    TabOrder = 5
  end
  object SCLabel3: TSCLabel
    Left = 76
    Top = 224
    Width = 17
    Height = 20
    AutoSize = True
    Caption = '0'
    Color = cl3DLight
    Enabled = False
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = [fsBold]
    Hottrack = False
    ParentColor = False
    ParentFont = False
    TabOrder = 6
  end
  object SCLabel4: TSCLabel
    Left = 99
    Top = 224
    Width = 17
    Height = 20
    AutoSize = True
    Caption = '0'
    Enabled = False
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = [fsBold]
    Hottrack = False
    ParentFont = False
    TabOrder = 7
  end
  object SCLabel5: TSCLabel
    Left = 124
    Top = 224
    Width = 17
    Height = 20
```

```
    AutoSize = True
    Caption = '0'
    Enabled = False
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = [fsBold]
    Hottrack = False
    ParentFont = False
    TabOrder = 8
  end
  object SCLabel6: TSCLabel
    Left = 149
    Top = 224
    Width = 17
    Height = 20
    AutoSize = True
    Caption = '0'
    Enabled = False
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = [fsBold]
    Hottrack = False
    ParentFont = False
    TabOrder = 9
  end
  object SCLabel7: TSCLabel
    Left = 173
    Top = 224
    Width = 17
    Height = 20
    AutoSize = True
    Caption = '0'
    Enabled = False
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = [fsBold]
    Hottrack = False
    ParentFont = False
    TabOrder = 10
  end
  object SCLabel1: TSCLabel
    Left = 8
    Top = 8
    Width = 258
    Height = 20
    AutoSize = True
    BiDiMode = bdRightToLeft
```

```
    Caption = 'Datos de los sensores del guante P5'
    Font.Charset = ANSI_CHARSET
    Font.Color = clNavy
    Font.Height = -13
    Font.Name = 'Georgia'
    Font.Style = [fsBold]
    Hottrack = False
    ParentBiDiMode = False
    ParentFont = False
    TabOrder = 11
  end
  object P5DataGlove1: TP5DataGlove
    Interval = 1
    OnProcess = PulgarChange
    GloveType = gtLeftHand
    Left = 16
    Top = 208
  end
end
```

Código fuente del archivo Panel.pas

```
unit Panel;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  SCControl, SCStdControls, DataGlove, jpeg, SCImageBox, ExtCtrls,
  OleCtrls, SHDocVw, SCEdits, SCMaskEdit, SCAdvEdits, SCDateTimeControls;

type
  TForm1 = class(TForm)
    SCButton1: TSCButton;
    Pulgar: TSCProgress;
    Indice: TSCProgress;
    Medio: TSCProgress;
    Anular: TSCProgress;
    Menique: TSCProgress;
    SCLabel3: TSCLabel;
    SCLabel4: TSCLabel;
    SCLabel5: TSCLabel;
    SCLabel6: TSCLabel;
    SCLabel7: TSCLabel;
    P5DataGlove1: TP5DataGlove;
    SCLabel1: TSCLabel;
    procedure SCButton1Click(Sender: TObject);
    procedure PulgarChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.SCButton1Click(Sender: TObject);
begin
close
end;

procedure TForm1.PulgarChange(Sender: TObject);
var Sensor_pulgar:byte;
var Sensor_indice:byte;
var Sensor_medio:byte;
var Sensor_anular:byte;
var Sensor_menique:byte;
begin

Sensor_pulgar:=P5DataGlove1.FingerThumb;
Sensor_indice:=P5DataGlove1.FingerIndex;
Sensor_medio:=P5DataGlove1.FingerMiddle;
Sensor_anular:=P5DataGlove1.FingerRing;
Sensor_menique:=P5DataGlove1.FingerPinky;
Pulgar.Position:=Sensor_pulgar;
Indice.Position:=Sensor_indice;
Medio.Position:=Sensor_medio;
Anular.Position:=Sensor_anular;
Menique.Position:=Sensor_menique;

SCLabel3.Caption := InttoStr(Sensor_pulgar);
SCLabel4.Caption := InttoStr(Sensor_indice);
SCLabel5.Caption := InttoStr(Sensor_medio);
SCLabel6.Caption := InttoStr(Sensor_anular);
SCLabel7.Caption := InttoStr(Sensor_menique);
end;

end.
```

Código fuente del formulario del programa orientación.

```
object Form1: TForm1
  Left = 192
  Top = 107
  Width = 442
  Height = 359
  Caption = 'Posición y orientación del guante P5'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
```

```
 Font.Color = clWindowText
 Font.Height = -11
 Font.Name = 'MS Sans Serif'
 Font.Style = [fsItalic]
   KeyPreview = True
 OldCreateOrder = False
 PixelsPerInch = 96
 TextHeight = 13
 object SCLabel1: TSCLabel
   Left = 296
   Top = 200
   Width = 15
   Height = 19
   AutoSize = True
   Caption = '0'
   Enabled = False
   Font.Charset = DEFAULT_CHARSET
   Font.Color = clHighlight
   Font.Height = -12
   Font.Name = 'Arial'
   Font.Style = []
   ParentFont = False
   TabOrder = 0
   OnClick = SCLabel1Click
 end
 object SCLabel2: TSCLabel
   Left = 296
   Top = 232
   Width = 15
   Height = 19
   AutoSize = True
   Caption = '0'
   Enabled = False
   Font.Charset = DEFAULT_CHARSET
   Font.Color = clHighlight
   Font.Height = -12
   Font.Name = 'Arial'
   Font.Style = []
   ParentFont = False
   TabOrder = 1
 end
 object SCLabel3: TSCLabel
   Left = 296
   Top = 264
   Width = 15
   Height = 19
   AutoSize = True
   Caption = '0'
   Enabled = False
   Font.Charset = DEFAULT_CHARSET
   Font.Color = clHighlight
   Font.Height = -12
   Font.Name = 'Arial'
```

```
  Font.Style = []
  ParentFont = False
  TabOrder = 2
 end
 object SCButton1: TSCButton
  Left = 336
  Top = 296
  Width = 72
  Height = 25
  Caption = 'Salir'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clHighlight
  Font.Height = -13
  Font.Name = 'Book Antiqua'
  Font.Style = [fsBold]
  HottrackColor = clGray
  ParentFont = False
  Style = scbsXP
  TabOrder = 3
  OnClick = SCButton1Click
 end
 object SCLabel4: TSCLabel
  Left = 240
  Top = 200
  Width = 32
  Height = 21
  AutoSize = True
  Caption = 'Yaw'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clHighlight
  Font.Height = -12
  Font.Name = 'Book Antiqua'
  Font.Style = [fsBold, fsItalic]
  ParentFont = False
  TabOrder = 4
 end
 object SCLabel5: TSCLabel
  Left = 240
  Top = 232
  Width = 37
  Height = 21
  AutoSize = True
  Caption = 'Pitch'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clHighlight
  Font.Height = -12
  Font.Name = 'Book Antiqua'
  Font.Style = [fsBold, fsItalic]
  ParentFont = False
  TabOrder = 5
 end
 object SCLabel6: TSCLabel
  Left = 240
```

```
    Top = 264
    Width = 32
    Height = 21
    AutoSize = True
    Caption = 'Roll'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clHighlight
    Font.Height = -12
    Font.Name = 'Book Antiqua'
    Font.Style = [fsBold, fsItalic]
    ParentFont = False
    TabOrder = 6
  end
  object SCLabel7: TSCLabel
    Left = 88
    Top = 200
    Width = 15
    Height = 19
    AutoSize = True
    Caption = '0'
    Enabled = False
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clHighlight
    Font.Height = -12
    Font.Name = 'Arial'
    Font.Style = []
    ParentFont = False
    TabOrder = 7
  end
  object SCLabel8: TSCLabel
    Left = 88
    Top = 232
    Width = 15
    Height = 19
    AutoSize = True
    Caption = '0'
    Enabled = False
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clHighlight
    Font.Height = -12
    Font.Name = 'Arial'
    Font.Style = []
    ParentFont = False
    TabOrder = 8
  end
  object SCLabel9: TSCLabel
    Left = 88
    Top = 264
    Width = 15
    Height = 19
    AutoSize = True
    Caption = '0'
    Enabled = False
```

```
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clHighlight
    Font.Height = -12
    Font.Name = 'Arial'
    Font.Style = []
    ParentFont = False
    TabOrder = 9
  end
  object SCLabel10: TSCLabel
    Left = 40
    Top = 200
    Width = 20
    Height = 21
    AutoSize = True
    Caption = 'X:'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clHighlight
    Font.Height = -12
    Font.Name = 'Book Antiqua'
    Font.Style = [fsBold, fsItalic]
    ParentFont = False
    TabOrder = 10
  end
  object SCLabel11: TSCLabel
    Left = 40
    Top = 232
    Width = 18
    Height = 21
    AutoSize = True
    Caption = 'Y:'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clHighlight
    Font.Height = -12
    Font.Name = 'Book Antiqua'
    Font.Style = [fsBold, fsItalic]
    ParentFont = False
    TabOrder = 11
  end
  object SCLabel12: TSCLabel
    Left = 40
    Top = 264
    Width = 19
    Height = 21
    AutoSize = True
    Caption = 'Z:'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clHighlight
    Font.Height = -12
    Font.Name = 'Book Antiqua'
    Font.Style = [fsBold, fsItalic]
    ParentFont = False
    TabOrder = 12
  end
```

```
 object SCLabel13: TSCLabel
   Left = 40
   Top = 56
   Width = 20
   Height = 21
   AutoSize = True
   Caption = 'X:'
   Font.Charset = DEFAULT_CHARSET
   Font.Color = clHighlight
   Font.Height = -12
   Font.Name = 'Book Antiqua'
   Font.Style = [fsBold, fsItalic]
   ParentFont = False
   TabOrder = 13
 end
 object SCLabel14: TSCLabel
   Left = 40
   Top = 88
   Width = 18
   Height = 21
   AutoSize = True
   Caption = 'Y:'
   Font.Charset = DEFAULT_CHARSET
   Font.Color = clHighlight
   Font.Height = -12
   Font.Name = 'Book Antiqua'
   Font.Style = [fsBold, fsItalic]
   ParentFont = False
   TabOrder = 14
 end
 object SCLabel15: TSCLabel
   Left = 40
   Top = 120
   Width = 19
   Height = 21
   AutoSize = True
   Caption = 'Z:'
   Font.Charset = DEFAULT_CHARSET
   Font.Color = clHighlight
   Font.Height = -12
   Font.Name = 'Book Antiqua'
   Font.Style = [fsBold, fsItalic]
   ParentFont = False
   TabOrder = 15
 end
 object SCLabel16: TSCLabel
   Left = 240
   Top = 56
   Width = 32
   Height = 21
   AutoSize = True
   Caption = 'Yaw'
   Font.Charset = DEFAULT_CHARSET
```

```
  Font.Color = clHighlight
  Font.Height = -12
  Font.Name = 'Book Antiqua'
  Font.Style = [fsBold, fsItalic]
  ParentFont = False
  TabOrder = 16
 end
 object SCLabel17: TSCLabel
  Left = 240
  Top = 88
  Width = 37
  Height = 21
  AutoSize = True
  Caption = 'Pitch'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clHighlight
  Font.Height = -12
  Font.Name = 'Book Antiqua'
  Font.Style = [fsBold, fsItalic]
  ParentFont = False
  TabOrder = 17
 end
 object SCLabel18: TSCLabel
  Left = 240
  Top = 120
  Width = 32
  Height = 21
  AutoSize = True
  Caption = 'Roll'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clHighlight
  Font.Height = -12
  Font.Name = 'Book Antiqua'
  Font.Style = [fsBold, fsItalic]
  ParentFont = False
  TabOrder = 18
 end
 object SCLabel19: TSCLabel
  Left = 80
  Top = 56
  Width = 15
  Height = 19
  AutoSize = True
  Caption = '0'
  Color = clBtnFace
  Enabled = False
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clHighlight
  Font.Height = -12
  Font.Name = 'Arial'
  Font.Style = []
  ParentColor = False
  ParentFont = False
```

```
  TabOrder = 19
 end
 object SCLabel20: TSCLabel
  Left = 80
  Top = 88
  Width = 15
  Height = 19
  AutoSize = True
  Caption = '0'
  Enabled = False
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clHighlight
  Font.Height = -12
  Font.Name = 'Arial'
  Font.Style = []
  ParentFont = False
  TabOrder = 20
 end
 object SCLabel21: TSCLabel
  Left = 80
  Top = 120
  Width = 15
  Height = 19
  AutoSize = True
  Caption = '0'
  Enabled = False
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clHighlight
  Font.Height = -12
  Font.Name = 'Arial'
  Font.Style = []
  ParentFont = False
  TabOrder = 21
 end
 object SCLabel22: TSCLabel
  Left = 296
  Top = 56
  Width = 15
  Height = 19
  AutoSize = True
  Caption = '0'
  Enabled = False
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clHighlight
  Font.Height = -12
  Font.Name = 'Arial'
  Font.Style = []
  ParentFont = False
  TabOrder = 22
 end
 object SCLabel23: TSCLabel
  Left = 296
  Top = 88
```

```
    Width = 15
    Height = 19
    AutoSize = True
    Caption = '0'
    Enabled = False
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clHighlight
    Font.Height = -12
    Font.Name = 'Arial'
    Font.Style = []
    ParentFont = False
    TabOrder = 23
  end
  object SCLabel24: TSCLabel
    Left = 296
    Top = 120
    Width = 15
    Height = 19
    AutoSize = True
    Caption = '0'
    Enabled = False
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clHighlight
    Font.Height = -12
    Font.Name = 'Arial'
    Font.Style = []
    ParentFont = False
    TabOrder = 24
  end
  object SCLabel25: TSCLabel
    Left = 56
    Top = 16
    Width = 328
    Height = 23
    AutoSize = True
    Caption = 'Datos de posición y orientación del guante P5.'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clHighlight
    Font.Height = -15
    Font.Name = 'Book Antiqua'
    Font.Style = [fsBold]
    ParentFont = False
    TabOrder = 25
  end
  object SCLabel26: TSCLabel
    Left = 96
    Top = 160
    Width = 219
    Height = 23
    AutoSize = True
    Caption = 'Datos utilizados en el modelo.'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clActiveCaption
```

```
    Font.Height = -15
    Font.Name = 'Book Antiqua'
    Font.Style = [fsBold]
    ParentFont = False
    TabOrder = 26
  end
  object P5DataGlove1: TP5DataGlove
    Interval = 1
    OnProcess = SCLabel1Click
    GloveType = gtLeftHand
    Left = 8
    Top = 296
  end
end
```

Código fuente del archivo Orientación_cod.pas del programa orientación

```
unit Orientacion_cod;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  SCControl, SCStdControls, DataGlove, ExtCtrls;

type
  TForm1 = class(TForm)
    SCLabel1: TSCLabel;
    SCLabel2: TSCLabel;
    SCLabel3: TSCLabel;
    SCButton1: TSCButton;
    SCLabel4: TSCLabel;
    SCLabel5: TSCLabel;
    SCLabel6: TSCLabel;
    P5DataGlove1: TP5DataGlove;
    SCLabel7: TSCLabel;
    SCLabel8: TSCLabel;
    SCLabel9: TSCLabel;
    SCLabel10: TSCLabel;
    SCLabel11: TSCLabel;
    SCLabel12: TSCLabel;
    SCLabel13: TSCLabel;
    SCLabel14: TSCLabel;
    SCLabel15: TSCLabel;
    SCLabel16: TSCLabel;
    SCLabel17: TSCLabel;
    SCLabel18: TSCLabel;
    SCLabel19: TSCLabel;
    SCLabel20: TSCLabel;
    SCLabel21: TSCLabel;
    SCLabel22: TSCLabel;
    SCLabel23: TSCLabel;
```

```
    SCLabel24: TSCLabel;
    SCLabel25: TSCLabel;
    SCLabel26: TSCLabel;
    procedure SCButton1Click(Sender: TObject);
    procedure SCLabel1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.SCButton1Click(Sender: TObject);
begin
close
end;

procedure TForm1.SCLabel1Click(Sender: TObject);
var X,Y,Z,Yaw,Pitch,Roll:Integer;
begin

//Datos de posición del Guante P5

//Datos truncados
X := Trunc(P5DataGlove1.Position.PosX) div 100;
Y := Trunc(P5DataGlove1.Position.PosY) div 100;
Z := Trunc(P5DataGlove1.Position.PosZ) div 100;
SCLabel7.Caption:=IntToStr(X);
SCLabel8.Caption:=IntToStr(Y);
SCLabel9.Caption:=IntToStr(Z);

//Datos originales X, Y, Z
SCLabel19.Caption:=FloatToStr(P5DataGlove1.Position.PosX);
SCLabel20.Caption:=FloatToStr(P5DataGlove1.Position.PosY);
SCLabel21.Caption:=FloatToStr(P5DataGlove1.Position.PosZ);

//Datos de orientación del Guante P5

//Datos truncados
Pitch:=P5DataGlove1.Position.Pitch;
Pitch:=(((Pitch*3.1416)/180)-1.57)*(-1);
Roll:=P5DataGlove1.Position.ROLL;
Roll:=((Roll*3.1416)/180)-1.57;
Yaw:=P5DataGlove1.Position.YAW;
Yaw:=((Yaw*3.1416)/180)*(-1)
SCLabel1.Caption:=IntToStr(Yaw);
SCLabel2.Caption:=IntToStr(Pitch);
```

```
SCLabel3.Caption:=IntToStr(Roll);

//Datos originales Yaw, Pitch y Roll
SCLabel22.Caption:=FloatToStr(P5DataGlove1.Position.Yaw);
SCLabel23.Caption:=FloatToStr(P5DataGlove1.Position.Pitch);
SCLabel24.Caption:=FloatToStr(P5DataGlove1.Position.Roll);
end;
end.
```

Código del formulario del pograma Mano

```
object Mano virtual: TForm1
  Left = 263
  Top = 152
  Width = 706
  Height = 575
  Caption = 'Mano virtual'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  WindowState = wsMaximized
  OnCreate = FormCreate
  PixelsPerInch = 96
  TextHeight = 13
  object Cortona1: TCortona
    Left = 0
    Top = 1
    Width = 698
    Height = 547
    ParentColor = False
    Align = alClient
    TabOrder = 0
   end
  object Memo1: TMemo
    Left = 0
    Top = 0
    Width = 698
    Height = 1
    Align = alTop
    Lines.Strings = (
      'Memo1')
    ScrollBars = ssBoth
    TabOrder = 1
    Visible = False
  end
  object P5DataGlove1: TP5DataGlove
    Interval = 1
    OnProcess = P5DataGlove1Process
```

```
    GloveType = gtLeftHand
    Left = 64
    Top = 96
  end
  object Timer1: TTimer
    Interval = 600
    OnTimer = Timer1Timer
    Left = 184
    Top = 104
  end
end
```

Código correspondiente al archive Mano_cod.pas del programa Mano

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  OleCtrls, CortonaVRMLClientLibrary_TLB, StdCtrls, DataGlove, Math,
  ExtCtrls;

type
  TForm1 = class(TForm)
    Cortona1: TCortona;
    Memo1: TMemo;
    P5DataGlove1: TP5DataGlove;
    Timer1: TTimer;
    procedure FormCreate(Sender: TObject);
    procedure P5DataGlove1Process(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
Memo1.Lines.LoadFromFile('Mano.wrl');
Cortona1.Scene:='Mano.wrl';
end;

procedure TForm1.P5DataGlove1Process(Sender: TObject);
```

```
var D1,D2,D3,D4,D5,Roll,Pitch,Yaw:Extended;



begin
D1:=63-P5DataGlove1.FingerPinky;
D2:=63-P5DataGlove1.FingerRing;
D3:=63-P5DataGlove1.FingerMiddle;
D4:=63-P5DataGlove1.FingerIndex;
D5:=63-P5DataGlove1.FingerThumb;

Pitch:=P5DataGlove1.Position.Pitch;
Pitch:=(((Pitch*3.1416)/180)-1.57)*(-1);
Roll:=P5DataGlove1.Position.ROLL;
Roll:=((Roll*3.1416)/180)-1.57;
Yaw:=P5DataGlove1.Position.YAW;
Yaw:=((Yaw*3.1416)/180)*(-1);

Memo1.Lines[5]:=copy(Memo1.Lines[5],1,16)+FloatToStr(Roll);
Memo1.Lines[10]:=copy(Memo1.Lines[10],1,16)+FloatToStr(Pitch);
Memo1.Lines[15]:=copy(Memo1.Lines[15],1,16)+FloatToStr(Yaw);
Memo1.Lines[812]:=copy(Memo1.Lines[812],1,10)+FloatToStr(D1);
Memo1.Lines[813]:=copy(Memo1.Lines[813],1,9)+FloatToStr(D2);
Memo1.Lines[814]:=copy(Memo1.Lines[814],1,8)+FloatToStr(D3);
Memo1.Lines[815]:=copy(Memo1.Lines[815],1,9)+FloatToStr(D4);
Memo1.Lines[816]:=copy(Memo1.Lines[816],1,9)+FloatToStr(D5);

end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
Memo1.Lines.SaveToFile('Mano.wrl');
Cortona1.Scene:='Mano.wrl';
end;

end.
```

Anexo C

Código MATLAB

```
% Programa para comprobar el MCDP obtenido de los dedos meñique y pulgar

close all      % Cierra las ventanas
clear all      % Limpia las variables
clc            % Limpia el Comand Window


% Conversión de ángulos a radianes

angulo1=((90+0)*pi)/180;
angulo2=(10*pi)/180;
angulo3=(15*pi)/180;
angulo4=(20*pi)/180;

angulop1=((90+0)*pi)/180;
angulop2=(10*pi)/180;
angulop3=(15*pi)/180;

% Ángulos theta para la cadena cinemática del dedo meñique

theta1=[angulo1];
theta2=[angulo2];
theta3=[angulo3];
theta4=[angulo4];

% Ángulos theta para la cadena cinemática del dedo pulgar

pulgart1=[angulop1];
pulgart2=[angulop2];
pulgart3=[angulop3];


% Longitudes de los eslabones para la cadena cinemática del dedo meñique

La1=55;
La2=33;
La3=17;
La4=17;
```

```
% Longitudes de los eslabones para la cadena cinemática del dedo pulgar

Lp1=45;
Lp2=30;
Lp3=23;


% Calcular MCDP para el dedo meñique para grafica 1

% Calcular punto A1
x1=0;
y1=0;

% Calcular punto A2
x2=La1.*cos(theta1(:,1));
y2=La1.*sin(theta1(:,1));

% Calcular punto A3
x3=x2+La2.*cos(theta1(:,1)+theta2(:,1));
y3=y2+La2.*sin(theta1(:,1)+theta2(:,1));

% Calcular punto A4
x4=x3+La3.*cos(theta1(:,1)+theta2(:,1)+theta3(:,1));
y4=y3+La3.*sin(theta1(:,1)+theta2(:,1)+theta3(:,1));

% Calcular punto A
x5=x4+La4.*cos(theta1(:,1)+theta2(:,1)+theta3(:,1)+theta4(:,1));
y5=y4+La4.*sin(theta1(:,1)+theta2(:,1)+theta3(:,1)+theta4(:,1));

% Almacenar los puntos en dos arreglos x y y respectivamente, para la
% graficación de estos

x=[x1 x2 x3 x4 x5];
y=[y1 y2 y3 y4 y5];


% Calcular MCDP para el dedo meñique para grafica 2

% Calcular punto A1
xa1=0;
ya1=0;

% Calcular punto A2
xa2=La1.*cos((theta1(:,1)-1.5708));
ya2=La1.*sin((theta1(:,1)-1.5708));

% Calcular punto A3
xa3=xa2+La2.*cos((theta1(:,1)-1.5708)+theta2(:,1));
ya3=ya2+La2.*sin((theta1(:,1)-1.5708)+theta2(:,1));

% Calcular punto A4
xa4=xa3+La3.*cos((theta1(:,1)-1.5708)+theta2(:,1)+theta3(:,1));
ya4=ya3+La3.*sin((theta1(:,1)-1.5708)+theta2(:,1)+theta3(:,1));
```

```
% Calcular punto A
xa5=xa4+La4.*cos((theta1(:,1)-1.5708)+theta2(:,1)+theta3(:,1)+theta4(:,1));
ya5=ya4+La4.*sin((theta1(:,1)-1.5708)+theta2(:,1)+theta3(:,1)+theta4(:,1));

% Almacenar los puntos en dos arreglos x y y respectivamente, para la
% graficación de estos

xa=[xa1 xa2 xa3 xa4 xa5];
ya=[ya1 ya2 ya3 ya4 ya5];


% Calcular MCDP para el dedo pulgar

% Calcular punto A1
xp1=0;
yp1=0;

% Calcular punto A2
xp2=Lp1.*cos(pulgart1(:,1));
yp2=Lp1.*sin(pulgart1(:,1));

% Calcular punto A3
xp3=xp2+Lp2.*cos(pulgart1(:,1)+pulgart2(:,1));
yp3=yp2+Lp2.*sin(pulgart1(:,1)+pulgart2(:,1));

% Calcular punto A
xp4=xp3+Lp3.*cos(pulgart1(:,1)+pulgart2(:,1)+pulgart3(:,1));
yp4=yp3+Lp3.*sin(pulgart1(:,1)+pulgart2(:,1)+pulgart3(:,1));

% Almacenar los puntos en dos arreglos x y y respectivamente, para la
% graficación de estos

xp=[xp1 xp2 xp3 xp4];
yp=[yp1 yp2 yp3 yp4];

% Calcular MCDP del dedo pulgar para grafica 2

% Calcular punto A1
xpa1=0;
ypa1=0;

% Calcular punto A2
xpa2=La1.*cos((pulgart1(:,1)-1.5708));
ypa2=La1.*sin((pulgart1(:,1)-1.5708));

% Calcular punto A3
xpa3=xpa2+La2.*cos((pulgart1(:,1)-1.5708)+pulgart2(:,1));
ypa3=ypa2+La2.*sin((pulgart1(:,1)-1.5708)+pulgart2(:,1));

% Calcular punto A
xpa4=xpa3+La3.*cos((pulgart1(:,1)-1.5708)+pulgart2(:,1)+pulgart3(:,1));
ypa4=ypa3+La3.*sin((pulgart1(:,1)-1.5708)+pulgart2(:,1)+pulgart3(:,1));
```

```
% Almacenar los puntos en dos arreglos x y y respectivamente, para la
% graficación de estos

xpa=[xpa1 xpa2 xpa3 xpa4];
ypa=[ypa1 ypa2 ypa3 ypa4];

% Graficas del MCDP del dedo meñique
figure (1)
subplot(2,1,1); plot(xa,ya, '-bo',...
            'LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor',[.49 1 .63],...
            'MarkerSize',12)
title ('Representacion grafica del MCDP del dedo meñique')
grid
subplot(2,1,2); plot(x,y, '-bo',...
            'LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor',[.49 1 .63],...
            'MarkerSize',12)
grid

% Graficas del MCDP del dedo pulgar
figure (2)
subplot(2,1,1); plot(xpa,ypa, '-bo',...
            'LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor',[.49 1 .63],...
            'MarkerSize',12)
title ('Representacion grafica del MCDP del dedo pulgar')

grid
subplot(2,1,2); plot(xp,yp, '-bo',...
            'LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor',[.49 1 .63],...
            'MarkerSize',12)
grid
```

Anexo D

Código Java

```
import Java.applet.Applet;
import Java.awt.*;
import Java.util.*;
import Java.awt.event.*;
import vrml.external.*;
import vrml.external.field.*;
import vrml.external.exception.*;


public class Mano extends Applet

{


Browser          myBrowser;

Node    gbTrans_cil_men;
Node    gbTrans_cil_men2;
Node    gbTrans_cil_men3;
Node    gbTrans_cil_men0;
Node    gbTrans_cil_anu;
Node    gbTrans_cil_anu2;
Node    gbTrans_cil_anu3;
Node    gbTrans_cil_anu0;
Node    gbTrans_cil_medio;
Node    gbTrans_cil_medio2;
Node    gbTrans_cil_medio3;
Node    gbTrans_cil_medio0;
Node    gbTrans_cil_indi;
Node    gbTrans_cil_indi2;
Node    gbTrans_cil_indi3;
Node    gbTrans_cil_indi0;
Node    gbTrans_cil_pul;
Node    gbTrans_cil_pul2;
Node    gbTrans_cil_pul3;
Node    gbTrans_cil_pul0;
```

```
EventInSFColor      gbColor             = null;
EventInSFVec3f      set_translation     = null;
EventInSFRotation   set_rotation        = null;
EventInSFRotation   set_rotation2       = null;
EventInSFRotation   set_rotation3       = null;
EventInSFRotation   set_rotation_anu    = null;
EventInSFRotation   set_rotation_anu2   = null;
EventInSFRotation   set_rotation_anu3   = null;
EventInSFRotation   set_rotation_medio  = null;
EventInSFRotation   set_rotation_medio2 = null;
EventInSFRotation   set_rotation_medio3 = null;
EventInSFRotation   set_rotation_indi   = null;
EventInSFRotation   set_rotation_indi2  = null;
EventInSFRotation   set_rotation_indi3  = null;
EventInSFRotation   set_rotation_pul    = null;
EventInSFRotation   set_rotation_pul2   = null;
EventInSFRotation   set_rotation_pul3   = null;

double[ ] grados = new double[3];
double[ ] radianes = new double[3];
float [ ] fin = new float [3];
float[ ]  trasladarMano=new float[3];
float[ ]  rotar_Meni=new float[4];
float[ ]  rotar_Meni1=new float[4];
float[ ]  rotar_Meni2=new float[4];
float[ ]  rotar_Anu=new float[4];
float[ ]  rotar_Anu1=new float[4];
float[ ]  rotar_Anu2=new float[4];
float[ ]  rotar_medio=new float[4];
float[ ]  rotar_medio1=new float[4];
float[ ]  rotar_medio2=new float[4];
float[ ]  rotar_indi=new float[4];
float[ ]  rotar_indi1=new float[4];
float[ ]  rotar_indi2=new float[4];
float[ ]  rotar_pul=new float[4];
float[ ]  rotar_pul1=new float[4];

// Variables trasladar
Label Eti_tras_x;
TextField Tex_tras_x;
Label Eti_tras_y;
TextField Tex_tras_y;
Label Eti_tras_z;
TextField Tex_tras_z;

/****************************** Meñique ****************************/

//Variables rotar primer cilindro
Label Eti_cil_men1;
TextField Tex_cil_men1;

//Variables rotar cegundo cilindro
Label Eti_cil_men2;
```

```
TextField Tex_cil_men2;

//Varuables rotar tercet cilindro
Label Eti_cil_men3;
TextField Tex_cil_men3;

/****************************** Anular ***************************/

//Variables rotar primer cilindro
Label Eti_cil_anu1;
TextField Tex_cil_anu1;

//Variables rotar cegundo cilindro
Label Eti_cil_anu2;
TextField Tex_cil_anu2;

//Varuables rotar tercet cilindro
Label Eti_cil_anu3;
TextField Tex_cil_anu3;

/****************************** Medio ****************************/

//Variables rotar primer cilindro
Label Eti_cil_medio1;
TextField Tex_cil_medio1;

//Variables rotar cegundo cilindro
Label Eti_cil_medio2;
TextField Tex_cil_medio2;

//Varuables rotar tercet cilindro
Label Eti_cil_medio3;
TextField Tex_cil_medio3;

/****************************** Indice ***************************/

//Variables rotar primer cilindro
Label Eti_cil_indi1;
TextField Tex_cil_indi1;

//Variables rotar cegundo cilindro
Label Eti_cil_indi2;
TextField Tex_cil_indi2;

//Varuables rotar tercet cilindro
Label Eti_cil_indi3;
TextField Tex_cil_indi3;

/****************************** Pulgar ***************************/

//Variables rotar primer cilindro
Label Eti_cil_pul1;
TextField Tex_cil_pul1;
```

```
//Variables rotar cegundo cilindro
Label Eti_cil_pul2;
TextField Tex_cil_pul2;

// Botones
Button Trasladar=null;
TextArea output = null;
Button Rotar = null;
Button Rotar_anular = null;
Button Rota_medio = null;
Button Rota_indi= null;
Button Rota_pul= null;

public void init()
{
	this.setLayout(new GridLayout(6, 7, 20, 15));
	this.setBackground(Color.white);
	this.setForeground(Color.black);
	// Campo de texto salida
		output = new TextArea(5, 40);
	//add(output);

/************************** Trasladar mano **************************/

//Valores de la opción trasladar
Eti_tras_x=new Label ("Trasladar en X: ");
Tex_tras_x=new TextField();
Eti_tras_y=new Label ("Trasladar en Y: ");
Tex_tras_y=new TextField();
Eti_tras_z=new Label ("Trasladar en Z: ");
Tex_tras_z=new TextField();

/************************** Meñique **************************/

// Valores del primer cilindro
Eti_cil_men1 = new Label ("° Meñique 1");
Tex_cil_men1 = new TextField();

// Valores del segundo cilindro
Eti_cil_men2 = new Label ("° Meñique 2");
Tex_cil_men2 = new TextField();

// Valores del tercer cilindro
Eti_cil_men3 = new Label ("° Meñique 3");
Tex_cil_men3 = new TextField();

/************************** Anular **************************/

// Valores del primer cilindro
Eti_cil_anu1 = new Label ("° Anular 1");
Tex_cil_anu1 = new TextField();
```

```
// Valores del segundo cilindro
Eti_cil_anu2 = new Label ("° Anular 2");
Tex_cil_anu2 = new TextField();

// Valores del tercer cilindro
Eti_cil_anu3 = new Label ("° Anular 3");
Tex_cil_anu3 = new TextField();

/****************************** Medio ***************************/

// Valores del primer cilindro
Eti_cil_medio1 = new Label ("° Medio 1");
Tex_cil_medio1 = new TextField();

// Valores del segundo cilindro
Eti_cil_medio2 = new Label ("° Medio 2");
Tex_cil_medio2 = new TextField();

// Valores del tercer cilindro
Eti_cil_medio3 = new Label ("° Medio 3");
Tex_cil_medio3 = new TextField();

/****************************** Indice ***************************/

// Valores del primer cilindro
Eti_cil_indi1 = new Label ("° Indice 1");
Tex_cil_indi1 = new TextField();

// Valores del segundo cilindro
Eti_cil_indi2 = new Label ("° Indice 2");
Tex_cil_indi2 = new TextField();

// Valores del tercer cilindro
Eti_cil_indi3 = new Label ("° Indice 3");
Tex_cil_indi3 = new TextField();

/************************** Pulgar *********************************/

// Valores del primer cilindro
Eti_cil_pul1 = new Label ("° Pulgar 1");
Tex_cil_pul1 = new TextField();

// Valores del segundo cilindro
Eti_cil_pul2 = new Label ("° Pulgar 2");
Tex_cil_pul2 = new TextField();

/************************** Trasladar mano ************************/
add(Eti_tras_x);
add(Tex_tras_x);
Tex_tras_x.addTextListener(new Escucha_trasladar());
add(Eti_tras_y);
add(Tex_tras_y);
Tex_tras_y.addTextListener(new Escucha_trasladar1());
```

```
add(Eti_tras_z);
add(Tex_tras_z);
Tex_tras_z.addTextListener(new Escucha_trasladar2());
add(Trasladar=new Button("Trasladar"));
Trasladar.addActionListener(new EscuchadorBoton());

/****************************** Meñique ******************************/

//Agregar valores del primer cilindro al Applet
add(Eti_cil_men1);
add(Tex_cil_men1);
Tex_cil_men1.addTextListener(new Escucha_cil_men());

//Agregar valores del segundo cilindro al Applet
add(Eti_cil_men2);
add(Tex_cil_men2);
Tex_cil_men2.addTextListener(new Escucha_cil_men2());

//Agregar valores del tercer cilindro al Applet
add(Eti_cil_men3);
add(Tex_cil_men3);
Tex_cil_men3.addTextListener(new Escucha_cil_men3());

// Boton rotar
add(Rotar=new Button("Rotar meñique"));
Rotar.addActionListener(new EscuchadorBoton_meni());

/****************************** Anular ******************************/

//Agregar valores del primer cilindro al Applet
add(Eti_cil_anu1);
add(Tex_cil_anu1);
Tex_cil_anu1.addTextListener(new Escucha_cil_anu());

//Agregar valores del segundo cilindro al Applet
add(Eti_cil_anu2);
add(Tex_cil_anu2);
Tex_cil_anu2.addTextListener(new Escucha_cil_anu2());

//Agregar valores del tercer cilindro al Applet
add(Eti_cil_anu3);
add(Tex_cil_anu3);
Tex_cil_anu3.addTextListener(new Escucha_cil_anu3());

// Boton anular
add(Rotar_anular=new Button("Rotar anular"));
Rotar_anular.addActionListener(new EscuchadorBoton_anular());

/****************************** Medio ******************************/

//Agregar valores del primer cilindro al Applet
add(Eti_cil_medio1);
add(Tex_cil_medio1);
```

```
Tex_cil_medio1.addTextListener(new Escucha_cil_medio());

//Agregar valores del segundo cilindro al Applet
add(Eti_cil_medio2);
add(Tex_cil_medio2);
Tex_cil_medio2.addTextListener(new Escucha_cil_medio2());

//Agregar valores del tercer cilindro al Applet
add(Eti_cil_medio3);
add(Tex_cil_medio3);
Tex_cil_medio3.addTextListener(new Escucha_cil_medio3());

// Boton rotar
add(Rota_medio=new Button("Rotar medio"));
Rota_medio.addActionListener(new EscuchadorBoton_medio());

/********************************** Indice ****************************/

//Agregar valores del primer cilindro al Applet
add(Eti_cil_indi1);
add(Tex_cil_indi1);
Tex_cil_indi1.addTextListener(new Escucha_cil_indi());

//Agregar valores del segundo cilindro al Applet
add(Eti_cil_indi2);
add(Tex_cil_indi2);
Tex_cil_indi2.addTextListener(new Escucha_cil_indi2());

//Agregar valores del tercer cilindro al Applet
add(Eti_cil_indi3);
add(Tex_cil_indi3);
Tex_cil_indi3.addTextListener(new Escucha_cil_indi3());

// Boton rotar
add(Rota_indi=new Button("Rotar indice"));
Rota_indi.addActionListener(new EscuchadorBoton_indi());

/********************************** Pulgar ****************************/

//Agregar valores del primer cilindro al Applet
add(Eti_cil_pul1);
add(Tex_cil_pul1);
Tex_cil_pul1.addTextListener(new Escucha_cil_pul());

//Agregar valores del segundo cilindro al Applet
add(Eti_cil_pul2);
add(Tex_cil_pul2);
Tex_cil_pul2.addTextListener(new Escucha_cil_pul2());

// Boton rotar
add(Rota_pul=new Button("Rotar pulgar"));
Rota_pul.addActionListener(new EscuchadorBoton_pul());
```

```
}

public void start()
 {
        myBrowser = Browser.getBrowser(this);

/************************** Trasladar mano **************************/

        gbTrans_cil_men0 = myBrowser.getNode("Mano");
        try {
                set_translation = (EventInSFVec3f) gbTrans_cil_men0.getEventIn("translation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

/***************************** Meñique *****************************/

        rotar_Meni[0]=0;
        rotar_Meni[1]=0;
        rotar_Meni[2]=1;
        gbTrans_cil_men = myBrowser.getNode("Cilindro1");
        try {
                set_rotation = (EventInSFRotation) gbTrans_cil_men.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

        rotar_Meni1[0]=0;
        rotar_Meni1[1]=0;
        rotar_Meni1[2]=1;
        gbTrans_cil_men2 = myBrowser.getNode("Cilindro2");
        try {
                set_rotation2 = (EventInSFRotation) gbTrans_cil_men2.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

        rotar_Meni2[0]=0;
        rotar_Meni2[1]=0;
        rotar_Meni2[2]=1;
        gbTrans_cil_men3 = myBrowser.getNode("Cilindro3");
        try {
                set_rotation3 = (EventInSFRotation) gbTrans_cil_men3.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

/***************************** Anular *****************************/

        rotar_Anu[0]=0;
        rotar_Anu[1]=0;
        rotar_Anu[2]=1;
        gbTrans_cil_anu = myBrowser.getNode("Cil_anular1");
        try {
                set_rotation_anu = (EventInSFRotation) gbTrans_cil_anu.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

        rotar_Anu1[0]=0;
        rotar_Anu1[1]=0;
        rotar_Anu1[2]=1;
        gbTrans_cil_anu2 = myBrowser.getNode("Cil_anular2");
```

```
        try {
                set_rotation_anu2 = (EventInSFRotation) gbTrans_cil_anu2.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

        rotar_Anu2[0]=0;
        rotar_Anu2[1]=0;
        rotar_Anu2[2]=1;
        gbTrans_cil_anu3 = myBrowser.getNode("Cil_anular3");
        try {
                set_rotation_anu3 = (EventInSFRotation) gbTrans_cil_anu3.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

/************************************ Medio ************************/

        rotar_medio[0]=0;
        rotar_medio[1]=0;
        rotar_medio[2]=1;
        gbTrans_cil_medio = myBrowser.getNode("Cil_medio1");
        try {
                set_rotation_medio = (EventInSFRotation) gbTrans_cil_medio.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

        rotar_medio1[0]=0;
        rotar_medio1[1]=0;
        rotar_medio1[2]=1;
        gbTrans_cil_medio2 = myBrowser.getNode("Cil_medio2");
        try {
                set_rotation_medio2 = (EventInSFRotation) gbTrans_cil_medio2.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

        rotar_medio2[0]=0;
        rotar_medio2[1]=0;
        rotar_medio2[2]=1;
        gbTrans_cil_medio3 = myBrowser.getNode("Cil_Medio3");
        try {
                set_rotation_medio3 = (EventInSFRotation) gbTrans_cil_medio3.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

/************************************ Indice ************************/

        rotar_indi[0]=0;
        rotar_indi[1]=0;
        rotar_indi[2]=1;
        gbTrans_cil_indi = myBrowser.getNode("Cil_indice1");
        try {
                set_rotation_indi = (EventInSFRotation) gbTrans_cil_indi.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

        rotar_indi1[0]=0;
        rotar_indi1[1]=0;
        rotar_indi1[2]=1;
        gbTrans_cil_indi2 = myBrowser.getNode("Cil_indice2");
        try {
```

```
                set_rotation_indi2 = (EventInSFRotation) gbTrans_cil_indi2.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

        rotar_indi2[0]=0;
        rotar_indi2[1]=0;
        rotar_indi2[2]=1;
        gbTrans_cil_indi3 = myBrowser.getNode("Cil_indice3");
        try {
                set_rotation_indi3 = (EventInSFRotation) gbTrans_cil_indi3.getEventIn("rotation");
             } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

/********************************** Pulgar ***************************/
        rotar_pul[0]=1;
        rotar_pul[1]=0;
        rotar_pul[2]=1;
        gbTrans_cil_pul = myBrowser.getNode("Cil_pulgar1");
        try {
                set_rotation_pul = (EventInSFRotation) gbTrans_cil_pul.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }

        rotar_pul1[0]=0;
        rotar_pul1[1]=1;
        rotar_pul1[2]=1;
        gbTrans_cil_pul2 = myBrowser.getNode("Cil_pulgar2");
        try {
                set_rotation_pul2 = (EventInSFRotation) gbTrans_cil_pul2.getEventIn("rotation");
            } catch (InvalidEventInException iee) { output.append("PROBLEMS!: " + iee + "\n"); }
}

public void stop()
 {
        myBrowser = null;
        gbColor = null;
        set_translation = null;
 }

/****************************** Escuchadores trasladar ************************/

class Escucha_trasladar implements TextListener
{
        public void textValueChanged(TextEvent e)
         {
          String s=e.paramString();
          if (s.equals("TEXT_VALUE_CHANGED"))
          {
                String texto=Tex_tras_x.getText();
                trasladarMano[0] = Float.valueOf(texto).floatValue();
          }
         }
}

class Escucha_trasladar1 implements TextListener
 {
```

```
        public void textValueChanged(TextEvent e)
         {
          String s=e.paramString();
          if (s.equals("TEXT_VALUE_CHANGED"))
            {
                String texto1=Tex_tras_y.getText();
                trasladarMano[1] =Float.valueOf(texto1).floatValue();
            }
         }
}

class Escucha_trasladar2 implements TextListener
 {
        public void textValueChanged(TextEvent e)
         {
          String s=e.paramString();
          if (s.equals("TEXT_VALUE_CHANGED"))
            {
                String texto2=Tex_tras_z.getText();
                trasladarMano[2] =Float.valueOf(texto2).floatValue();
            }
         }
}

/***************************** Escuchadores rotar meñique ************************/

class Escucha_cil_men implements TextListener
{
 public void textValueChanged(TextEvent e)
  {
        String s=e.paramString();
        if (s.equals("TEXT_VALUE_CHANGED"))
            {
                String texto3=Tex_cil_men1.getText();
                grados[1]=(Double.valueOf(texto3).doubleValue());
                radianes[1]=( (grados[1] * 3.1416) / 180);
                String valor=(String.valueOf (radianes[1]));
                fin[1]=(Float.valueOf(valor).floatValue());
                rotar_Meni[3]=fin[1];
            }
  }
}

class Escucha_cil_men2 implements TextListener
{
 public void textValueChanged(TextEvent e)
  {
        String s=e.paramString();
        if (s.equals("TEXT_VALUE_CHANGED"))
            {
                String texto3=Tex_cil_men2.getText();
                grados[2]=(Double.valueOf(texto3).doubleValue());
                radianes[2]=( (grados[2] * 3.1416) / 180);
```

```
                String valor=(String.valueOf (radianes[2]));
                fin[2]=(Float.valueOf(valor).floatValue());
                rotar_Meni1[3]=fin[2];
            }
   }
}

class Escucha_cil_men3 implements TextListener
{
 public void textValueChanged(TextEvent e)
   {
        String s=e.paramString();
        if (s.equals("TEXT_VALUE_CHANGED"))
            {
                String texto3=Tex_cil_men3.getText();
                grados[0]=(Double.valueOf(texto3).doubleValue());
                radianes[0]=( (grados[0] * 3.1416) / 180);
                String valor=(String.valueOf (radianes[0]));
                fin[0]=(Float.valueOf(valor).floatValue());
                rotar_Meni2[3]=fin[0];
            }
   }
}

/***************************** Escuchadores rotar anular **********************/

class Escucha_cil_anu implements TextListener
{
 public void textValueChanged(TextEvent e)
   {
        String s=e.paramString();
        if (s.equals("TEXT_VALUE_CHANGED"))
            {
                String texto3=Tex_cil_anu1.getText();
                grados[1]=(Double.valueOf(texto3).doubleValue());
                radianes[1]=( (grados[1] * 3.1416) / 180);
                String valor=(String.valueOf (radianes[1]));
                fin[1]=(Float.valueOf(valor).floatValue());
                rotar_Anu[3]=fin[1];
            }
   }
}

class Escucha_cil_anu2 implements TextListener
{
 public void textValueChanged(TextEvent e)
   {
        String s=e.paramString();
        if (s.equals("TEXT_VALUE_CHANGED"))
            {
                String texto3=Tex_cil_anu2.getText();
                grados[2]=(Double.valueOf(texto3).doubleValue());
                radianes[2]=( (grados[2] * 3.1416) / 180);
```

```
                    String valor=(String.valueOf (radianes[2]));
                    fin[2]=(Float.valueOf(valor).floatValue());
                    rotar_Anu1[3]=fin[2];
                }
    }
}

class Escucha_cil_anu3 implements TextListener
{
 public void textValueChanged(TextEvent e)
    {
          String s=e.paramString();
          if (s.equals("TEXT_VALUE_CHANGED"))
               {
                    String texto3=Tex_cil_anu3.getText();
                    grados[0]=(Double.valueOf(texto3).doubleValue());
                    radianes[0]=( (grados[0] * 3.1416) / 180);
                    String valor=(String.valueOf (radianes[0]));
                    fin[0]=(Float.valueOf(valor).floatValue());
                    rotar_Anu2[3]=fin[0];
               }
    }
}

/***************************** Escuchadores rotar medio ***********************/

 class Escucha_cil_medio implements TextListener
 {
  public void textValueChanged(TextEvent e)
     {
          String s=e.paramString();
          if (s.equals("TEXT_VALUE_CHANGED"))
               {
                    String texto3=Tex_cil_medio1.getText();
                    grados[1]=(Double.valueOf(texto3).doubleValue());
                    radianes[1]=( (grados[1] * 3.1416) / 180);
                    String valor=(String.valueOf (radianes[1]));
                    fin[1]=(Float.valueOf(valor).floatValue());
                    rotar_medio[3]=fin[1];
                }
    }
}

class Escucha_cil_medio2 implements TextListener
{
 public void textValueChanged(TextEvent e)
    {
          String s=e.paramString();
          if (s.equals("TEXT_VALUE_CHANGED"))
               {
                    String texto3=Tex_cil_medio2.getText();
                    grados[2]=(Double.valueOf(texto3).doubleValue());
                    radianes[2]=( (grados[2] * 3.1416) / 180);
```

```
                    String valor=(String.valueOf (radianes[2]));
                    fin[2]=(Float.valueOf(valor).floatValue());
                    rotar_medio1[3]=fin[2];
                }
    }
}

class Escucha_cil_medio3 implements TextListener
{
 public void textValueChanged(TextEvent e)
    {
          String s=e.paramString();
          if (s.equals("TEXT_VALUE_CHANGED"))
               {
                    String texto3=Tex_cil_medio3.getText();
                    grados[0]=(Double.valueOf(texto3).doubleValue());
                    radianes[0]=( (grados[0] * 3.1416) / 180);
                    String valor=(String.valueOf (radianes[0]));
                    fin[0]=(Float.valueOf(valor).floatValue());
                    rotar_medio2[3]=fin[0];
               }
    }
}

/***************************** Escuchadores rotar índice ************************/

class Escucha_cil_indi implements TextListener
{
 public void textValueChanged(TextEvent e)
    {
          String s=e.paramString();
          if (s.equals("TEXT_VALUE_CHANGED"))
               {
                    String texto3=Tex_cil_indi1.getText();
                    grados[1]=(Double.valueOf(texto3).doubleValue());
                    radianes[1]=( (grados[1] * 3.1416) / 180);
                    String valor=(String.valueOf (radianes[1]));
                    fin[1]=(Float.valueOf(valor).floatValue());
                    rotar_indi[3]=fin[1];
               }
    }
}

class Escucha_cil_indi2 implements TextListener
{
 public void textValueChanged(TextEvent e)
    {
          String s=e.paramString();
          if (s.equals("TEXT_VALUE_CHANGED"))
               {
                    String texto3=Tex_cil_indi2.getText();
                    grados[2]=(Double.valueOf(texto3).doubleValue());
                    radianes[2]=( (grados[2] * 3.1416) / 180);
```

```
                    String valor=(String.valueOf (radianes[2]));
                    fin[2]=(Float.valueOf(valor).floatValue());
                    rotar_indi1[3]=fin[2];
                }
    }
}

class Escucha_cil_indi3 implements TextListener
{
 public void textValueChanged(TextEvent e)
    {
            String s=e.paramString();
            if (s.equals("TEXT_VALUE_CHANGED"))
                {
                    String texto3=Tex_cil_indi3.getText();
                    grados[0]=(Double.valueOf(texto3).doubleValue());
                    radianes[0]=( (grados[0] * 3.1416) / 180);
                    String valor=(String.valueOf (radianes[0]));
                    fin[0]=(Float.valueOf(valor).floatValue());
                    rotar_indi2[3]=fin[0];
                }
    }
}

/***************************** Escuchadores rotar pulgar **********************/

class Escucha_cil_pul implements TextListener
 {
  public void textValueChanged(TextEvent e)
    {
             String s=e.paramString();
             if (s.equals("TEXT_VALUE_CHANGED"))
                {
                     String texto3=Tex_cil_pul1.getText();
                     grados[1]=(Double.valueOf(texto3).doubleValue());
                     radianes[1]=( (grados[1] * 3.1416) / 180);
                     String valor=(String.valueOf (radianes[1]));
                     fin[1]=(Float.valueOf(valor).floatValue());
                     rotar_pul[3]=fin[1];
                }
    }
 }

class Escucha_cil_pul2 implements TextListener
{
 public void textValueChanged(TextEvent e)
    {
            String s=e.paramString();
            if (s.equals("TEXT_VALUE_CHANGED"))
                {
                    String texto3=Tex_cil_pul2.getText();
                    grados[2]=(Double.valueOf(texto3).doubleValue());
                    radianes[2]=( (grados[2] * 3.1416) / 180);
```

```
                String valor=(String.valueOf (radianes[2]));
                fin[2]=(Float.valueOf(valor).floatValue());
                rotar_pul1[3]=fin[2];
            }
    }
}

/****************************** Escuchador botón trasladar ***********************/

class EscuchadorBoton implements ActionListener
{
public void  actionPerformed(ActionEvent e)
    {
        String s=e.getActionCommand();
I       f (s.equals("Trasladar"))
        {
            if (trasladarMano[0] >= 0.0 || trasladarMano[0] < 0.0 && trasladarMano[1] >= 0.0 ||
            trasladarMano[1] < 0.0 && trasladarMano[2] >= 0.0 || trasladarMano[2] < 0.0 )
            {
                set_translation.setValue(trasladarMano);
            }
        }
    }
}

/****************************** Escuchador botón rotar meñique ***********************/

class EscuchadorBoton_meni implements ActionListener
{
public void  actionPerformed(ActionEvent e)
    {
        String s=e.getActionCommand();
        if (s.equals("Rotar meñique"))
        {
        if (rotar_Meni[3] >= -0.35 && rotar_Meni[3] <= 1.5708)
            {
                set_rotation.setValue(rotar_Meni);
            }
        if (rotar_Meni1[3] >= 0 && rotar_Meni1[3] <= 1.5708)
            {
                set_rotation2.setValue(rotar_Meni1);
            }
        if (rotar_Meni2[3] >= 0 && rotar_Meni2[3] <= 1.5708)
            {
                set_rotation3.setValue(rotar_Meni2);
            }
        }
    }
}

/****************************** Escuchador botón rotar anular ***********************/

class EscuchadorBoton_anular implements ActionListener
```

```
{
public void  actionPerformed(ActionEvent e)
    {
            String s=e.getActionCommand();
            if (s.equals("Rotar anular"))
            {
            if (rotar_Anu[3] >= -0.35 && rotar_Anu[3] <= 1.5708)
               {
                     set_rotation_anu.setValue(rotar_Anu);
               }
            if (rotar_Anu1[3] >= 0 && rotar_Anu1[3] <= 1.5708)
               {
                     set_rotation_anu2.setValue(rotar_Anu1);
               }
            if (rotar_Anu2[3] >= 0 && rotar_Anu2[3] <= 1.5708)
               {
                     set_rotation_anu3.setValue(rotar_Anu2);
               }
            }
   }
}

/****************************** Escuchador botón rotar medio ***********************/

class EscuchadorBoton_medio implements ActionListener
{
 public void  actionPerformed(ActionEvent e)
    {
            String s=e.getActionCommand();
            if (s.equals("Rotar medio"))
            {
            if (rotar_medio[3] >= -0.35 && rotar_medio[3] <= 1.5708)
               {
                     set_rotation_medio.setValue(rotar_medio);
               }
            if (rotar_medio1[3] >= 0 && rotar_medio1[3] <= 1.5708)
               {
                     set_rotation_medio2.setValue(rotar_medio1);
               }
            if (rotar_medio2[3] >= 0 && rotar_medio2[3] <= 1.5708)
               {
                     set_rotation_medio3.setValue(rotar_medio2);
               }
            }
   }
}

/****************************** Escuchador botón rotar índice ***********************/

class EscuchadorBoton_indi implements ActionListener
{
 public void  actionPerformed(ActionEvent e)
    {
```

```
            String s=e.getActionCommand();
            if (s.equals("Rotar indice"))
            {
            if (rotar_indi[3] >= -0.35 && rotar_indi[3] <= 1.5708)
               {
                     set_rotation_indi.setValue(rotar_indi);
               }
            if (rotar_indi1[3] >= 0 && rotar_indi1[3] <= 1.5708)
               {
                     set_rotation_indi2.setValue(rotar_indi1);
               }
            if (rotar_indi2[3] >= 0 && rotar_indi2[3] <= 1.5708)
               {
                     set_rotation_indi3.setValue(rotar_indi2);
               }
            }
   }
}
```

*/****************************** Escuchador botón rotar pulgar **********************/*

```
class EscuchadorBoton_pul implements ActionListener
{
 public void  actionPerformed(ActionEvent e)
    {
            String s=e.getActionCommand();
            if (s.equals("Rotar pulgar"))
            {
            output.setText("Operacion: "+rotar_pul[3]);
            set_rotation_pul.setValue(rotar_pul);
            set_rotation_pul2.setValue(rotar_pul);
            }
   }
}
```

Glosario

ActiveX: es un conjunto de tecnologías desarrolladas por Microsoft que permiten a los componentes de software interactuar entre sí en un ambiente conectado de red, como Internet, independientemente del lenguaje de desarrollo en que fueron creados.

Applet: es un programa de Java que puede recuperarse con un explorador web y ejecutarse en la computadora local al mismo tiempo.

Browser: programa que sirve para buscar y visualizar información de la WWW. Los Browsers de VRML son programas que interpretan el contenido de un programa en este lenguaje t muestra en pantalla el escenario virtual descrito por ese programa.

Cinemática: es la rama de la mecánica que estudia el movimiento de los cuerpos sin tener en cuenta sus causas.

Clase: son declaraciones o abstracciones de objetos, lo que significa, que una clase es la definición de un objeto. Cuando se programa un objeto y se definen sus características y funcionalidades, realmente se programa una clase.

Componente: parte discreta de un sistema capaz de operar independientemente, pero diseñada, construida y operada como parte integral del sistema.

Dispositivo háptico: son dispositivos que proporcionan salida desde la computadora hacia el usuario simulando retroalimentación de fuerza y tacto.

Dynamic Linking Library: archivos con código ejecutable que se cargan bajo demanda del programa por parte del sistema operativo en tiempo de ejecución.

Evento: es la generación de un mensaje y su envío de un nodo a otro.

Fibra óptica: filamento de pequeño diámetro de silicio u otro material apropiado capaz de canalizar en su interior una señal luminosa que se envía a uno de sus extremos.

Frame: cuadro o marco único e identificable dentro de una aplicación.

Host: es un término difuso, ya que escapa a una definición única: se utiliza con muchos significados, en función del contexto. Así, en general, se denomina host a un ordenador que permite a los usuarios comunicarse con otros ordenadores de una red.

Hub: tiene varios significados, según el contexto en el cual es empleado: en informática un hub o concentrador es un ordenador que hace las funciones de servidor de los demás ordenadores que se encuentran conectados en forma radial al ordenador central.

Interfaz: sistema de comunicación de un programa con su usuario.

Lenguaje de programación: es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un programa informático.

Metodología: se refiere a los métodos de investigación en una ciencia.

Método: modo ordenado de proceder para llegar a un resultado o fin determinado, para descubrir la verdad y sistematizar los conocimientos. En Java los métodos son subrutinas que definen la interfaz de una clase, sus capacidades y comportamiento.

Modelado: proceso de creación de los objetos en un software.

Modelo: es una conceptualización de un evento, un proyecto, una hipótesis, el estado de una cuestión, que se representa como un esquema con símbolos descriptivos de características y relaciones más importantes con un fin: ser sometido a modelización como un diseño flexible, que emerge y se desarrolla durante el inicio de la investigación como una evaluación de su relevancia.

Nodo: son el componente principal de un gráfico en VRML, tienen una serie de características esenciales para el desarrollo de ambientes virtuales, estas características definen la conducta de los nodos.

Pitch: es el ángulo de rotación alrededor del eje Y.

Plug-in: es un programa que se conecta a un programa anfitrión para extender la funcionalidad de este último. El programa huésped tiene ciertas capacidades que el anfitrión no posee.

Protocolo: descripción técnica de un estándar, incluyendo las reglas de diseño y funcionamiento. Un conjunto de reglas.

Realidad Aumentada ó Realidad Mixta: es un campo de investigación de la computación que trata de la combinación de mundo real y datos generados por computadora. Se refiere al hecho de añadir un objeto virtual al mundo real utilizando algún dispositivo.

Realidad Virtual: es la representación de la cosas a través de medios electrónicos, que da la sensación de estar en una situación real en la que el usuario puede interactuar con lo que le rodea.

Render: cálculo que realiza la computadora para presentar una imagen en pantalla.

Roll: es el ángulo de rotación alrededor del eje X.

Script consiste básicamente en un conjunto de funciones que se ejecutan en un momento determinado.

Técnica: es el procedimiento o el conjunto de procedimientos que tienen como objetivo obtener un resultado determinado, ya sea en el campo de la ciencia, de la tecnología, de las artesanías o en otra actividad.

Tecnología: término general que se aplica al proceso a través del cual los seres humanos diseñan herramientas y máquinas para incrementar su control y su comprensión del entorno material.

Telepresencia: presencia remota.

Yaw: es el ángulo de rotación alrededor del eje Z.

Acrónimos

2D Dos dimensiones

3D Tres dimensiones

AC Alternating Current

API Application Programming Interface

AR Augmented Reality

AWT Abstract Windowing Toolkit

BOOM Binocular Omni-Orientation Monitor

CAVE Cave Automatic Virtual Environment

COM Component Object Model

CPU Central Processing Unit

DC Direct Current

DHM Dexterous Hand Master

DLL Dynamic Linking Library

EAI External Authoring Interface

EJB Enterprise JavaBeans

EVL Electronic Visualization Laboratory

GDL Grados de Libertad

GUI Grafic User Interface

HID Light Emitting Diode

HMD Head Mounted Displays

HTML HyperText Markup Language

Hz Hertz

IDE Integrated Development Environment

ITP Interactive Telecommunications Program

J2EE Java 2 Enterprise Edition

JAR Archivo de Java

JDBC Java Batabase Connectivity

JDK Java Development Kit

JNI Java Native Interface

JSAI Java Scripting Authoring Interface

JSP Java Server Page

LED Human Interface Device

MCDP Modelo Cinemático Directo de Posición

MIDI Musical Instrument Digital Interface

MR Mixed Reality

NASA National Aeronautics and Space Administration

PC Phase Coherent

PC Personal Computer

PIE Programmable Input Emulator

POO Programación Orientada a Objetos

PPJoy Parallel Port Joystick

RAD Rapid Application Development

RMI Remote Method Invocation

RV Realidad Virtual

SDK Software Development Kit

TOF Time of Fly

URL Uniform Resource Locator

USB Universal Serial Bus

USD United Stated Dollar

UTF-8 Unicode Transformation Format

VCL Visual Components Library

VITA Visual Interaction Tool for Archeology

VPL Visual Programming Language

VRML Virtual Reality Modeling Language

Bibliografía

[1] Abascal, J., Garay, N., *"Dispositivos"*, Universidad del paìs Vasco

[2] Aguinis, H., Henle, C., Beaty, J., *"Virtual Reality Technology: a new tool for personnel selection."* , International Journal of selection and assessment March / June 2001

[3] Atencia, J., Nestar, R., *"Aprenda Matlab 6.0 como si estuviera en primero",* Universidad de Navarra, Escuela Superior de Ingenieros, 2001

[4] Baharak, Z., *"Virtual Keyboard"* Enrineering Approaches to Music Perception and Cognition, University of Southern California.

[5] Balaguer, F., Mangili, A., *"Virtual environments"*, Computer Graphics Laboratory, Swiss Federal Institute of Technology, Lausanne Switzerland.

[6] *"Bend sensor technology mechanical application design guide"*, Flexpoint Sensor system, 1997

[7] Benko, W., Ishak, E., Feiner, S., "*Cross-Dimensional Gestural Interaction Techniques for Hybrid Immersive Environments"*, Columbia University, New York, NY

[8] Fonseca, Y., Medina, F., *"Arquitecturas virtuales de ciudad universitaria: ICBI – A, ICBI –B, ICBI –C, ICEA, ICEA –A, Edificio de postrado, economía y la unidad central de laboratorios"* Universidad Autónoma del Estado de Hidalgo, 2005.

[9] García, J., Díaz, J:, Portillo, P., Contreras, M., Guzmán, R., *"Aplicación de VRML a la creación de un catálogo industrial interactivo en Internet"*, Universidad de Málaga, Departamento de expresión gráfica, diseño y proyectos,
2002.

[10] Gutiérrez Cirlos, G., "Anatomía, Fisiología e higiene", Kapelusz Mexicana, 3ª edición, 1984.

[11] Hamza-Lup, F., Hughes, C., Rolland, J., *"Distributed Consistency Maintenance Scheme forInteractive Mixed Reality Environments"*, College of Optics and Photonics: CREOL/FPCE, University of Central Florida.

[12] Hollands, R., Hand, C., Clark, S., "The Virtual Reality Homebrewer's Handbook", Ed. John Wiley & Sons, New York, U.S.A., 1996

[13] Holloway, R., Lastra, A., *"Virtual environments: a rurvey of the technology"*, University of North Carolina Chapel Hill.

[14] *"Informatión bookLET"*, Essential Reality, 2002

[15] Kenner, C. "Unwarping the LED positions", Julio 2004

[16] Lam, W., Zou, F., Komura, T., *"Motion editing with data glove"*, City University of Hong Kong.

[17] Lobo, P., Gómez, J. *"Aplicaciones del lenguaje VRML (Virtual Reality Modeling Language) a la ciencia de materiales"*, Madrid España.

[18] Lockhart, R., Hamilton, G., Fyfe, F., "*Anatomía humana",* Interamericana, 1ª edición, 1965.

[19] Marrin, C., *"Proposal for a VRML 2.0 Informative Annex External Authoring Interface Referente"*, Silicon Graphics Inc.

[20] Márquez, J., García, R., Santelices, I., "introducción práctica a la realidad virtual" Ediciones U. Bío-Bío, Concepción, 2001

[21] Marteens, I., *"La cara oculta de Delphi"*, Pamplona, España, 1997

[22] Martínez, S., Vega, I., *"Simulación de riesgos de sismos en una casa-habitación utilizando realidad virtual"*, Universidad de Colima, Facultad de Telemática, 2004

[23] Mazuryk, T., Gervautz, M., *"Virtual reality history, applications technology and future"*, Institute of Computer Graphics, Vienna Universit of Technology, Austria.

[24] Olwal, A., Benko, H., Feiner, S., *"SenseShapes: Using Statistical Geometry for Object Selection in a Multimodal Augmented Reality System."*, IEEE 2003, Departament of Computer Cience, Columbia University, New Yor, NY.

[25] Parra, J., García, R., Santelices, I., *"Introducción práctica a la realidad virtual"*, 2001

[26] Quintero, R., Alvarado, J., Núñez, G., *"Asignación de comportamientos complejos a mundos virtuales VRML utilizando C++",* Revista Digital Universitaria, UNAM, 2001.

[27] Quintero, R., Alvarado, J., Nuñez, G., *"Asignación de comportamientos complejos a mundos virtuales VRML utilizando C++",* Revista Digital

Universitaria, UNAM, 2001.

[28] Tirado, F., *"La virtualización de la sociedad"*, Universidad Autónoma de Barcelona.

[29] *"Virtual Reality Modeling Language, Version 2.0, ISO/IEC WD 14772"*

[30] Wang, P., *"Java con programación orientada a objetos y aplicaciones en la World Wide Web "*, International Thomson Editores, 2000

[31] Welch, G., Foxlin, E., "Motion tracking: no Oliver bullet, but a respectable arsenal", University of North Carolina, IEEE 2002.

[32] Youngblut, C., Jhonson, R., Nash, S., Wienclaw, R., Will, C., *"Review of virtual environment interface technology", 1996*

Referencias electrónicas

[33] *"Active interaction devices"*, http://www.hitl.washington.edu/scivw/EVE/ I.D.1.a.ActiveInteraction.html

[34] *"Aphrodisias 2003"*, http://www.tinkering.net/aphro.html

[35] *"Características de Java"*, http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_3.htm

[36] *"Cinemática directa"*, http://proton.ucting.udg.mx/materias /robotica /r166 /r79 /r79.htm

[37] *"Computer Science at Columbia University"*, http://www1.cs.columbia.edu/ ~benko /projects/SenseShapes/olwala_senseshapes. pdf

[38] *"Computer Science at Columbia University"*, http://www1.cs.columbia.edu/ ~benko/ publications/2005/Benko_Xdim_VR05. pdf

[39] *"Cortona"*, http://www.parallelgraphics.com/products/cortona/

[40] *"Cortona user's guide"*, http://www.parallelgraphics.com/developer/products/ cortona/help/

[41] " *Filename Extensions in Delphi* ", http://delphi.about.com /od /beginners/a /aa032800a.htm

[42] *"Flexible bend sensor"*, ttp://www.imagesco.com/catalog/flex/FlexSensors.html

[43] *"Flex sensor"*, http://www.harris-educational.com/Probeware/experiments/TSA /flex-sensors.htm

[44] *"Floppy's VRML97 Tutorial",* http://web3d.vapourtech.com/tutorials/ vrml97/ es/tut41.html

[45] *"Future in your hand"*, *http://news.bbc.co.uk/1/hi/sci/tech/1768818.stm*

[46] *"Griho"*, http://griho.udl.es/ipo/pdf/06Dispos.pdf

[47] *"How to animate your favorite diet"*, *http://www.vruniverse.com/vrdj/article1 /tutorial1.html*

[48] *"Human interface technology lab"*, http://www.hitl.washington.edu/scivw/scivw-ftp/publications/IDA-pdf/TRACK.PDF

[49] *"Human interface technology lab"*, http://www.hitl.washington.edu/scivw/scivw-ftp/publications/IDA-pdf/PRIMARY.PDF

[50] *"Introducción a Java"*, http://tikal.cifn.unam.mx/~jsegura/LCGII/Java2.htm

[51] *"Institute of computer graphics and algorithms"*, http://www.cg.tuwien.ac.at /TR/96/TR-186-2-96-06 Paper.pdf

[52] *"Java"*, http://www.unav.es/cti/manuales/Java/indice.html

[52] "*Java & VRML"*, *http://jdj.sys-con.com/read/35890.htm*

[53] *"MC Squared Incorporated"*, *http://www.mcsqd.com/html/projects.html*

[54] *"Metodologías",* http://web.madritel.es/personales3/edcollado/ingsw/tema2/2-4.htm

[55] *"OpenGL demos"*, http://rich99.com/opengl.htm

[56] *"Polimorfismo"*, http://www.qualitrain.com.mx/objeIndirecto/polimorfismo.htm

[57] *"P5 community page"*, http://www.zzz.com.ru/index.php?area=pages&action= view_page&page_id=11

[58] *"P5 Developers information"*, *http://www.videogamealliance.com/VGA/ video_game /P5 /P5_Developers.php*

[59] *"P5 Midi"*, http://www.nicolasfournel.com/P5midi.htm

[60] *"¿Qué es Java?"*, http://www.iec.csic.es/criptonomicon/Java/quesJava.html

[61] *"Realidad virtual, simulación y vicualización",* http://www.decom.es/ area.php?cod=1

[62] *"Revista digital universitaria"*, http://www.revista.unam.mx/vol.2/num2/art2/

[62] *"Revista digital universitaria"¸* http://www.revista.unam.mx/vol.1/num2/art2/

[63] *"Robot glove project"*, http://www.robotgroup.net/index.cgi/RobotGlove

[64] *"School of Electrical Engineering and Computer Science"*, http://www.cs.ucf.edu /~ceh/ Publications/Papers/CrossCutting /CITSA04Distr Consistency.pdf

[65] *"SCI-Tech"*, *http://archives.cnn.com/2002/TECH/ptech/04/26/keyless.keyboards. idg/*

[66] *"Spectra symbol"*, http://www.spectrasymbol.com/sensors.html

[67] *"Tracking Devices"*, http://www.hitl.washington.edu/scivw/EVE/I.D.1.b. TrackingDevices.html

[68] *"University of Colorado at Denver and Healt Sciences Center"*, http://carbon. cudenver.edu /~haguinis/IJSA2001.pdf

[69] *"VrmlPad"*, http://www.parallelgraphics.com/products/vrmlpad/

[70] *"VrmlPad feautures"*, http://www.parallelgraphics.com/products/vrmlpad/ features

[71] "*Visualization and Graphics Research Groups"*, http://graphics.cs.ucdavis.edu/ ~staadt/ECS289H-WQ02/holloway93virtual.pdf

[72] *"Virtual Keyboard"*, http://www-scf.usc.edu/~ise575/projects/zali/bzISE575/ VirtualKeyboard1.htm

[73] *"Virtual Keyboard project"*, http://www-scf.usc.edu/~ise575/projects/zali/ bzISE575

[74] *"Virtual Reality Lab"*, ttp://ligwww.epfl.ch/~thalmann/papers.dir/06.virtual.pdf

[75] *"Virtual sculptor"*, http://www.axel.nm.ru/p5glove/index.html

[76] *"VRML"*, http://:www.activamente.com.mx/vrml/

[77] *"VRML",* http://usuarios.lycos.es/artofmusic /the_matrix_vr/definicion_vr.html