UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

DESARROLLO DE UN VIDEOJUEGO DEL TIPO ESTRATEGIA EN TIEMPO REAL, CON COMUNICACIÓN BASADA EN EL RECONOCIMIENTO DE GESTOS DE LAS MANOS

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA (COMPUTACIÓN)

P R E S E N T A:

ADIDIER MONZERRAT PÉREZ GÓMEZ

DIRECTOR DE TESIS:

DR. JESÚS SAVAGE CARMONA

México, D.F. **2008.**

AGRADECIMIENTOS

A mi familia y amigos por su ayuda incondicional, así mismo agradezco los apoyos académicos y económicos al Posgrado en Ciencia e Ingeniería de la Computación y al CONACyT.

Adidier Monzerrat Pérez Gómez
Agosto del 2008

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

MOTIVACIÓN

En la actualidad, la industria de videojuegos es considerada económicamente fuerte e importante a nivel internacional. Su carácter multidisciplinario, así como su rápida evolución y penetración en el mercado la hace un campo digno de estudio y aplicación de nuevos avances tecnológicos.

El cimiento de esta área de entretenimiento es la búsqueda de diversión a partir de una realidad virtual en función de tecnología y software necesarios para su proyección. Para ello, a lo largo de su relativa corta historia se han desarrollado mejoras en cuanto a niveles de calidad visual y mecánica de los juegos, así como el hardware que permita hacer estas simulaciones. Sin embargo, esta constante evolución en cuanto a novedad y potencia no contemplaba la parte de control o interacción del usuario [STANG2006].

Hasta ahora un videojuego es controlado generalmente con un mando o dispositivo de entrada, a base de pulsar botones y realizar combinaciones que en muchos casos son complejos, algo que supone un gran aprendizaje para poder jugar con naturalidad. Un punto muy particular de lo anterior es el género de juegos de estrategia puesto que resulta complicado su control y manipulación debido a la naturaleza de las acciones requeridas.

Los juegos de estrategia son específicos para las computadoras personales puesto que la interfaz preferida por el usuario para manipularlos está conformada por el *mouse* y teclado ya que otros dispositivos limitan la movilidad o navegación en el ambiente del juego o no permiten una selección adecuada y rápida de las unidades. Razón por la cual, se requiere de un control alternativo diseñado para facilitar la interacción de una manera intuitiva y natural, sin tener que adquirir un dispositivo adicional y aprender complicados términos para su funcionamiento.

Con base a la revisión de tecnologías que buscan una mejor interacción humano-máquina al evolucionar los métodos tradicionales de control, la gesticulación puede ser una forma de ingreso de datos adecuada debido a

características notables como la naturalidad y destreza propias de los seres humanos. Además, el inició de la utilización masiva de cámaras *web* en las computadoras personales permite aplicar y probar sistemas enfocados al uso de avances en Visión por Computadora.

Con tales motivaciones, surge la idea de crear un sistema de comunicación basado en el reconocimiento de gestos de las manos a través del uso de una cámara *web* como alternativa de control para un videojuego de estrategia en tiempo real para computadora, y con ello mostrar la viabilidad de su uso como método de interacción no sólo en aplicaciones de entretenimiento con gráficos tridimensionales, sino también en aplicaciones comunes para manipular una computadora personal.

PROPUESTA DE SOLUCIÓN

La presente tesis propone el desarrollo de un videojuego para computadora personal del género de estrategia en tiempo real con un sistema de control alternativo para su manipulación. Para ello, se plantea crear un *software* que implemente el reconocimiento de gestos de las manos para la generación de eventos de entrada.

Como se trata establecer una comunicación que explote componentes, técnicas y herramientas actuales sin generar un costo adicional, se decidió utilizar herramientas de uso libre. Además, el hardware para la captura de los gestos es una cámara *web* puesto que es un componente de uso común en las computadoras personales que permite aumentar sus funcionalidades al aplicar técnicas de Visión por Computadora.

Con respecto al desarrollo de un videojuego, se busca generar una aplicación que de acuerdo con su lógica de manipulación e interfaz gráfica de usuario permita mostrar el funcionamiento del reconocimiento de los gestos de las manos. Es por ello, que dentro de los géneros existentes de juegos para computadora personal, se eligió el de estrategia en tiempo real ya que su

naturaleza de control representa una interacción viable a probar y además de que existe la necesidad de evolucionar sus dispositivos de entrada tradicionales: *mouse* y teclado.

En suma, la propuesta es desarrollar e integrar un software de reconocimiento de gestos de las manos con una aplicación que requiere nuevas formas de interacción que exploten los avances tecnológicos actuales, con la finalidad de ofrecer una comunicación alternativa al mostrar su implementación y funcionamiento.

OBJETIVOS

El objetivo general de la tesis es desarrollar un videojuego de estrategia en tiempo real con control y comunicación con el usuario a través del reconocimiento de gestos de las manos.

Para alcanzar este objetivo general, es necesario concretarlo en objetivos específicos, los cuales son sintetizados en los siguientes puntos:

- Se plantea aplicar técnicas de las áreas de *Visión por Computadora, Inteligencia Artificial y Graficación* con el fin de desarrollar un sistema que contribuya a la demostración práctica de ofrecer una interacción alternativa humano-máquina en aplicaciones de entretenimiento con gráficas tridimensionales.
- Realizar un estudio de las metodologías y herramientas adecuadas para el desarrollo, con la finalidad de aprovechar las tecnologías existentes.
- Generar un software que en tiempo real implemente el seguimiento de gestos de las manos por medio de marcas (patrones) y una cámara.
- Traducción de los gestos a mensajes del sistema operativo que representen comportamientos propios de dispositivos tales como el teclado y *mouse*.

- Desarrollar un videojuego 3D de estrategia en tiempo real que utilice los comandos que genere el reconocimiento de gestos de las manos.

ALCANCE

Debido a que el presente trabajo se centra en construir un sistema prototipo que permita demostrar la viabilidad del reconocimiento de gestos de las manos como medio de comunicación del usuario con un juego de estrategia en tiempo real, se determinó que para su construcción se haría uso de herramientas que proporcionen las técnicas necesarias. Razón por la cual, el *software* para el reconocimiento de gestos utiliza bibliotecas de distribución libre.

En cuanto al juego de estrategia de tiempo real, es importante mencionar que por cuestiones de tiempo y complejidad, se decidió que su desarrollo se limitara a los elementos básicos de un juego de estrategia en tiempo real como temática, características esenciales del ambiente virtual, inteligencia artificial y visualización de información simple. Aspectos complejos y multidisciplinaros como historia, gráficos detallados, soporte multijugador, entre otros; no entran en el alcance.

Por otro lado, se descarta el desarrollo de un prototipo multiplataforma, por lo que sólo se presenta una versión para el sistema operativo Microsoft Windows. Finalmente, cabe señalar que el sistema propuesto no busca la sustitución de los dispositivos de control convencionales, sino ofrecer una alternativa de interacción que pueda ser integrada a otras aplicaciones existentes.

ESTRUCTURA DE LA TESIS

La presente tesis consta de cuatro capítulos, las conclusiones y dos apéndices.

En el capítulo primero, se presentan los conceptos fundamentales relacionados con el tema: visión por computadora, reconocimiento de gestos,

videojuegos y algunos proyectos relacionados con lo propuesto. Con ello se ofrece la teoría que sustenta el presente trabajo.

El capítulo segundo, expone el marco tecnológico aplicado en el proyecto de tesis. Para ello se introducen las tecnologías utilizadas como parte fundamental de la solución del problema.

El tercer capítulo aborda la descripción del sistema desarrollado. Para ello, se define la arquitectura general y la explicación de cada uno de los componentes.

El cuarto capítulo se centra en la descripción de las pruebas y resultados obtenidos.

En las conclusiones se presenta un resumen de lo obtenido durante el desarrollo del trabajo. Se presentan las contribuciones obtenidas, así como los trabajos a futuro.

El apéndice A contiene la estructura y formato del archivo de configuración que almacena los gestos que implementa la aplicación VInput.

El apéndice B describe la adaptación e integración del navegador Mozilla Firefox con la aplicación VInput.

Capítulo 1

Marco teórico

Este capítulo muestra una panorámica general sobre los fundamentos teóricos y el estado del arte del presente trabajo. Para ello, se ofrece una breve introducción a los temas de Visión por Computadora, reconocimiento de gestos y videojuegos; mostrando sus principales características, aplicaciones, así como la exposición del contexto que propician las contribuciones propuestas.

1.1 VISION POR COMPUTADORA

1.1.1 Concepto

El termino Visión por Computadora (del inglés *Computer Vision*) no cuenta con una definición precisa, razón por la cual existe un gran número de definiciones. Dentro de las más citadas están:

- Ciencia que desarrolla las bases teóricas y algorítmicas para obtener información sobre el mundo real a partir de una o varias imágenes. [HARALICK1992].
- Disciplina que desarrolla sistemas capaces de interpretar el contenido de escenas naturales [CASTLEMAN1996].

En resumen, la Visión por Computadora puede ser considerada como la disciplina que desarrolla un conjunto de todas aquellas técnicas y modelos que permiten la extracción, el procesamiento, análisis y explicación de información obtenida a través de imágenes digitales.

1.1.2 Componentes de un sistema de Visión por Computadora

Un sistema de Visión por Computadora cuenta con ciertos componentes que le permiten realizar el tratamiento de la información visual dependiendo del objetivo que persigue. Dentro de estos componentes están:

Hardware

- ***Cámara***. Es el hardware de captura de imagen.
- ***Fuente de energía***. Es la energía necesaria para realizar la captura de imagen y depende del tipo de análisis que se desea hacer del objeto de estudio. Los tipos de energía utilizados puede ser: luz para fotografía, rayos X y gamma para radiografía o tomografía, ultrasonido para la ecografía, campos magnéticos para magneto-resonancia, calor para la termografía, entre otros.

- ***Sensor***. El sensor debe ser sensible a la energía utilizada.

Software

- ***Interfaz de adquisición***. Es el software que obtiene las imágenes del hardware de captura. Se encarga de identificar cuando se puede adquirir una nueva imagen, acondicionarla y muestrearla para generar la imagen digital y dejarla a disposición de la aplicación que la pida o procese.
- ***Software de procesamiento***. Es el software encargado de procesar la imagen. Dentro de las tareas que puede desempeñar están: el almacenamiento de la imagen digital en determinado formato, la extracción de características deseadas de los elementos, detección, clasificación, mediciones, interpretación, toma de decisiones en base al objeto de estudio, entre otras.

1.1.3 Tareas para el procesamiento

Dentro de las principales etapas que se encuentran en los sistemas de Visión por Computadora están:

- *Adquisición de la imagen*: Una imagen digital es producida por uno o varios sensores y por varios tipos de cámaras.
- *Preprocesamiento*: Antes de que un método de Visión por Computadora se aplique a los datos de la imagen para extraer información específica, es generalmente necesario procesar los datos con técnicas como realce, suavizado, eliminación de ruido, entre otros; para asegurar que satisfacen ciertos criterios.
- *Detección/Segmentación*: En este punto del procesamiento se decide qué puntos o regiones de la imagen son relevantes para la transformación posterior.

- *Extracción de características*: el objetivo es obtener una descripción matemática de los objetos segmentados. Las características de la imagen que se extraen de los datos de la imagen pueden ser: eje principal, área, perímetro, momentos invariantes, patrones de textura, entre otras.

- *Procesamiento de alto nivel*: En este paso la entrada es típicamente un conjunto pequeño de datos (representación matemática), por ejemplo un conjunto de puntos o una región de la imagen que contiene un objeto específico. El procesamiento en esta etapa implica acciones como:

    - Reconocimiento y localización de objetos. Se realiza según las características extraídas del objeto o por correspondencia con modelos.
    - Interpretación de la escena. Esta acción no es necesaria en todas las aplicaciones e involucra elementos de áreas como la Inteligencia Artificial: razonamiento, experiencia, aprendizaje, entre otros.

1.1.4 Áreas relacionadas

Desde sus orígenes, la Visión por Computadora ha utilizado como inspiración el estudio del sistema visual humano, el cual sugiere la existencia de diferentes tipos de tratamiento de la información visual dependiendo de metas u objetivos específicos. Tomando en cuenta lo anterior, la naturaleza de esta disciplina resulta ser compleja y multidisciplinaria puesto que existe una relación estrecha con otras áreas de conocimiento. A continuación, se describen las áreas más cercanas e involucradas con la Visión por Computadora.

Procesamiento de imágenes: área que a través de un conjunto de técnicas tiene como objetivo producir una versión modificada de una imagen con la finalidad de mejorar su calidad o para hacer más evidentes ciertos detalles en ella que se desean hacer notar [GONZALEZ2002]. Algunos de los efectos o transformaciones utilizadas son: compresión de la información contenida,

reducción de ruido, restauración de la imagen, realce o extracción de características (contornos, texturas, color, movimiento, entre otros).

Visión Artificial: esta área tiende a centrarse principalmente en aplicaciones de tipo industrial, por ejemplo, robots y sistemas autónomos basados en visión para inspección o medición. Esto implica que la teoría de control y las tecnologías de sensores de imagen están integradas a menudo para el procesamiento de los datos de la imagen y así controlar a un robot o sistema.

Inteligencia Artificial: una parte significativa de la Inteligencia Artificial se ocupa de la planeación o deliberación autónoma de sistemas que pueden realizar acciones mecánicas, tales como el movimiento de un robot a través de cierto ambiente. Este tipo de proceso típicamente necesita como datos de entrada los proporcionados por un sistema de Visión por Computadora, el cual actúa como el sensor de visión y proporciona la información de alto nivel sobre el ambiente y el robot. Otros campos considerados dentro la Inteligencia Artificial y que se utilizan en la Visión por Computadora son el reconocimiento de patrones y técnicas de aprendizaje.

Neurobiología: se relaciona con la Visión por Computada específicamente con el estudio del sistema biológico de visión. Los resultados obtenidos de esta área han conducido a un subcampo dentro de la Visión por Computadora donde los sistemas artificiales se diseñan con respecto al proceso y comportamiento de los sistemas biológicos en diversos niveles de complejidad.

Física: este campo se relaciona fuertemente con la Visión por Computadora puesto que se utilizan métodos que requieren una comprensión cuidadosa del proceso mediante el cual se reflejan las superficies de objetos y es medida por el sensor para producir los datos de la imagen.

Matemáticas: muchos temas relacionados con la Visión por Computadora se pueden también estudiar desde un punto de vista puramente matemático. Por ejemplo, muchos métodos se basan en estadística, optimización o geometría.

Procesamiento de señales: esta área estudia los métodos para procesar señales, lo cual implica el procesamiento, amplificación e interpretación de señales. Las señales pueden proceder de diversas fuentes y existen varios tipos de procesamiento dependiendo de su naturaleza.

1.1.5 Áreas de aplicación e investigación

La Visión por Computadora tiende a ser un área con una aplicación amplia y multidisciplinaria [PAJARES2001], puesto que se involucran tareas de identificación, discriminación y clasificación de elementos como objetos, acciones o eventos. En la tabla 1.1 se presenta un panorama de las principales áreas de aplicación e investigación de soluciones basadas en Visión por Computadora.

Área	**Aplicaciones**
Industria	De la rama automotriz, financieras, farmacéutica y de cosméticos, de alimentos, textil, básicas de transformación (madera, aluminio, acero, papel), entre otras. • Inspección y control de calidad de productos. • Sistemas de ensamblaje y empaquetamiento. • Identificación, clasificación o medición automática de piezas.
Medicina	• Automatización de pruebas de laboratorio (recuento de células. detección de células anormales). • Diagnóstico basado en imágenes (ultrasonidos, resonancias magnéticas, tomografía, radiología)
Teledetección	• Detección de cambios urbanos • Clasificación del suelo • Control de contaminación • Control del tráfico
Ciencias	• Análisis de turbulencias en fluidos • Cinemática de movimiento (personas)
Otras	• Geología • Ámbito Militar • Robótica móvil • Mediciones meteorológicas • Entretenimiento

Tabla 1.1: Aplicaciones de la Visión por Computadora por áreas.

Para terminar, dentro de estas aplicaciones su ubica al reconocimiento de gestos. En el siguiente apartado se profundiza sobre este tema puesto que es un elemento central en el proyecto de tesis.

1.2 RECONOCIMIENTO DE GESTOS

El reconocimiento de gestos es una aplicación del área de Visión por Computadora donde a través de un conjunto de algoritmos matemáticos, técnicas de procesamiento de imágenes y análisis de series temporales busca la creación de sistemas que identifiquen gestos humanos capturados por una cámara y sean utilizados como dispositivo de control o para transmitir información.

Los gestos son movimientos físicos corporales expresivos o con significado para transmitir información o para descubrir, interactuar y manipular el ambiente. Estos movimientos pueden provenir de los dedos, manos, brazos, cabeza, cara, entre otros. Ahora bien, dado que puede el cuerpo humano expresar una gran variedad de gestos, ¿cuáles son apropiados para detectar? Existen muchos tipos de gestos posibles [KENDON1972] [MCNEIL1992], y dependiendo de los objetivos de los sistemas, es como se realiza esta elección.

Puesto que los gestos se originan de la cara o mano principalmente, muchos planteamientos han sido realizados para comprender el lenguaje de signos. Por otra parte, la mayoría de los sistemas existentes reconocen gestos de la mano y brazo, debido a su manejabilidad y naturalidad de interacción en aplicaciones donde las tareas son principalmente de manipulación y comunicación.

El reconocimiento de gestos de la mano en particular, consiste en realizar un mapeo de los ángulos que forman los segmentos articulados de los dedos, la posición y dirección de la mano, y posiblemente su evolución en el tiempo, hacia un conjunto de símbolos generalmente parametrizados. El gesto comprende este mapeo, por lo que puede ser la última información sobre el estado de la mano y

dedos, es decir su postura, o adicionalmente en su movimiento y configuraciones pasadas.

Debido a que no existe un procedimiento estándar para realizar el reconocimiento de gestos, se tiene una variedad de esquemas de representación y clasificación. Sin embargo, según el tipo de gesto capturado, se puede hacer uso de ciertas técnicas para su procesamiento como se muestra en la tabla 1.2.

Tipo	**Descripción**	**Técnicas**
Gesto estático	Se define por una postura o configuración fija.	Redes neuronales, template matching, clasificación de características geométricas u otra técnica estándar de reconocimiento de patrones para clasificar la postura.
Gesto dinámico	Se define por el movimiento y requiere la consideración temporal de un conjunto de gestos estáticos.	Time-compressing templates, Modelos Ocultos de Markov (Hidden Markov Models), redes Bayesianas, Normalización temporal, flujo óptico, entre otras.

Tabla 1.2: Tipos de gestos y técnicas con respecto a su representación.

Cabe señalar que algunos gestos pueden tener elementos estáticos y dinámicos, donde cierta postura sea importante en una o varias fases. Cuando se producen continuamente, cada gesto se ve afectado por el anterior y posiblemente por el siguiente. En suma, los aspectos a considerar para el reconocimiento de gestos comprende el sentido de la posición, configuración y movimiento.

Para la determinación de los gestos a reconocer se debe tomar en cuenta sugerencias de diseño como [TURK2001]: no deben ser incómodos o requerir de mucha exactitud para no cansar al usuario, deben ser lo más intuitivos y simples como sea posible, no definir gestos muy similares entre sí para no confundir al usuario, entre otras.

Con respecto a los dispositivos que permiten la captura de los gestos, existen varios por lo que se elección depende de situaciones como precisión, resolución, movimiento, comodidad del usuario y costo. Algunos dispositivos son:

rastreadores de campos magnéticos, guantes instrumentados, cámaras y técnicas de Visión por Computadora.

Como ya se ha mencionado, el objetivo del reconocimiento de gestos es el crear sistemas que puedan generar un dispositivo de control y comunicación. Dentro de las tareas de comunicación que un usuario requiere especificar por comandos y/o parámetros a través de gestos están:

- Navegación a través de un espacio.
- Especificación de elementos de interés.
- Manipulación de objetos en el ambiente (cambiar valores o controlar objetos).
- Edición de comandos específicos para tareas.

Existen trabajos que demuestran que la aplicabilidad de esta área puede ofrecer métodos simples y naturales de comunicación con una computadora y programas. Se ha observado mediante experimentos [HUMMELS1998] que resultan ser altamente intuitivas y tienden a ser preferidas como medio de interacción. En el siguiente apartado se mencionan algunos de los trabajos realizados.

1.2.1 Estado del arte de aplicaciones con reconocimiento de gestos

En la actualidad, se han estudiado y propuesto nuevas formas de Interacción Humano-Computadora (HCI, siglas en inglés) más naturales e intuitivas basadas en el reconocimiento de gestos. Desde hace unos años se han desarrollado varias aplicaciones, con este tipo de interacción, que hacen uso de técnicas de áreas como Visión por Computadora e Inteligencia Artificial.

Un aspecto fundamental de tales aplicaciones es que deben ofrecer una interfaz que interprete los comandos explícitos o implícitos relacionados a los movimientos o gestos, de una parte o de todo el cuerpo de las personas. La idea es que ayuden a realizar varias clases de tareas, dentro de las cuales se puede

mencionar: a) Navegación en ambientes virtuales y manipulación de objetos virtuales; b) Dibujo y posicionamiento, reemplazando el *mouse*, teclado u otro dispositivo de entrada; c) Control de diferentes dispositivos electrónicos y robots; d) Comunicación a través de señas.

Dentro de los trabajos realizados que se centran en el reconocimiento de gestos de las manos y que buscan ofrecen una alternativa para dispositivos tradicionales de entrada se pueden mencionar:

- Seguimiento del dedo como entrada en tiempo real [FUKUMOTO1994]. Esta investigación está orientada al manejo de un cursor al moverlo en una pantalla proyectada a través de la postura y desplazamiento de la mano dentro de un espacio observado por dos cámaras.
- Seguimiento del dedo como entrada en una aplicación de dibujo de realidad aumentada [CROWLEY1995].
- El proyecto *FingerMouse* [MYSLIWIEC1994] [QUEK1995] puede detectar la mano en una postura. Una vez que es detectada, el dedo índice extendido es encontrado y representa el puntero del ratón.
- Reconocimiento de gestos de la mano obtenidos de imágenes 2D sin dispositivos externos [BYUNG-WOO1999]. Los gestos son marcados por un estado de transición basado en la articulación natural humana. Se reconocen gestos estáticos usando la postura de la mano en ciertos momentos. Para gestos dinámicos, se reconocen analizando sus trayectorias móviles sobre Modelos Ocultos de Markov.
- Sistema de Visión por Computadora para el reconocimiento de personas mediante posturas de la mano en fondos complejos [TRIESCH2001]. El sistema se basa en un EGM (*Elastic Graph Matching*).

En los siguientes apartados, se presentan las aplicaciones que tienen un fin similar al de esta tesis.

1.2.1.1 HandVu

Es un sistema de software de Visión por Computadora que reconoce gestos de la mano en tiempo real y sus salidas pueden funcionar como un dispositivo de control para el usuario (figura 1.1). Consiste en una biblioteca y un módulo base de reconocimiento de gestos que implementa todos los métodos de Visión por Computadora necesarios para detección, seguimiento y reconocimiento de los gestos de la mano [KÖLSCH2004].

Figura 1.1: Interacción con el sistema *HandVu [KÖLSCH2004]*.

El módulo de reconocimiento consta de tres etapas: 1) detecta la presencia de la mano en una postura particular, 2) realiza el seguimiento y 3) reconoce la postura. En cuanto a hardware, utiliza como dispositivo de entrada a una cámara digital colocada en la frente del usuario y como dispositivo de salida de despliegue maneja un visor HMD (*Head Mounted Display*).

HandVu funciona como una biblioteca para el reconocimiento de gestos que se puede incluir en aplicaciones que requieran o desean manejar una interacción con el usuario basada en gestos de la mano. Sin embargo, es necesario utilizar programas adicionales para la adquisición de video y despliegue de la imagen.

Por último, tiene un módulo independiente para la funcionalidad básica de una interfaz de aplicación móvil GIS (Sistema de Información Geográfica). Esta aplicación fue construida para el proyecto Battuta puesto que era adecuado para demostrar la facilidad de adaptar dispositivos de entrada tradicionales con los basados en gestos de la mano.

1.2.1.2 FingARtips

Es un sistema que implementa un conjunto simple de gestos que permiten al usuario de manera directa manipular objetos virtuales en ambientes de Realidad Aumentada (AR).

FingARtips se utiliza en la interfaz gráfica de una aplicación de planeación urbana y utiliza técnicas como [BUCHMANN2004]: interacción basada en las yemas de los dedos, software de procesamiento de imágenes, marcas y dispositivos táctiles (*haptic*) en las yemas de los dedos para dar cierta sensación al usuario de los objetos virtuales.

El sistema hace uso de un guante con ciertas marcas sobrepuestas y reconoce cuatro tipos de gestos: tomar objeto, señalar, navegar y gesto comando (presionar botón virtual). La mano tiene una representación gráfica en el ambiente virtual representada únicamente con el dedo índice y pulgar (figura 1.2). Además, se hace uso de ciertos dispositivos electrónicos en el guante para proporcionar diferentes niveles de presión mediante diferentes intensidades de vibración. Las acciones que el usuario puede realizar son:

- Creación, destrucción, traslación, rotación y escalamiento de edificios.
- Creación y manipulación de calles virtuales.
- Cambiar el modo de visión a Realidad Virtual o Realidad Aumentada.

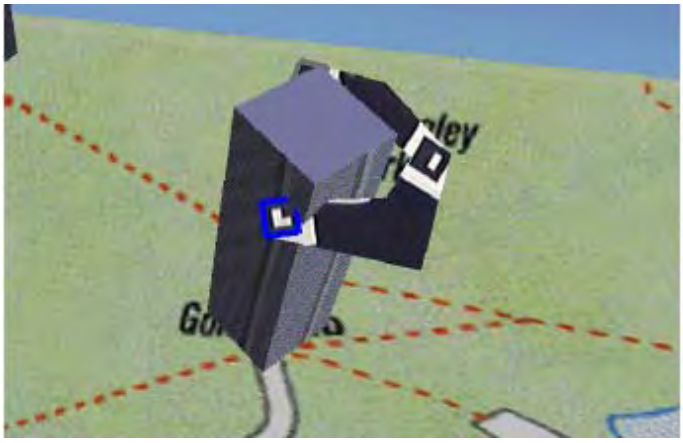


Figura 1.2: Interfaz del sistema *FingARtips [BUCHMANN2004]*.

1.2.1.3 Mouse Gestures

Mouse Gestures es un plug-in o extensión para el navegador Mozilla Firefox que permite controlar funciones básicas del programa con simples gestos del *mouse*, sin necesidad de recurrir a la barra de herramientas.

Esta extensión puede ser instalada para personalizar funciones como abrir enlaces, cerrar ventanas del navegador, ir hacia adelante o atrás en el historial de navegación, entre otras; con simples movimientos del *mouse*. Incluye más de 60 combinaciones personalizables y extensibles.

Permite capturar acciones de los tres botones del *mouse* y detecta ocho direcciones de movimiento del cursor. A través de estas referencias, se tiene un listado de combinaciones de tipo cadena (gestos) para asignar a las diferentes funciones que se pueden realizar en el navegador (figura 1.3). También permite la activación del rastro del cursor y cambiar su color para facilitar al usuario la visualización de los comandos que ejecuta.

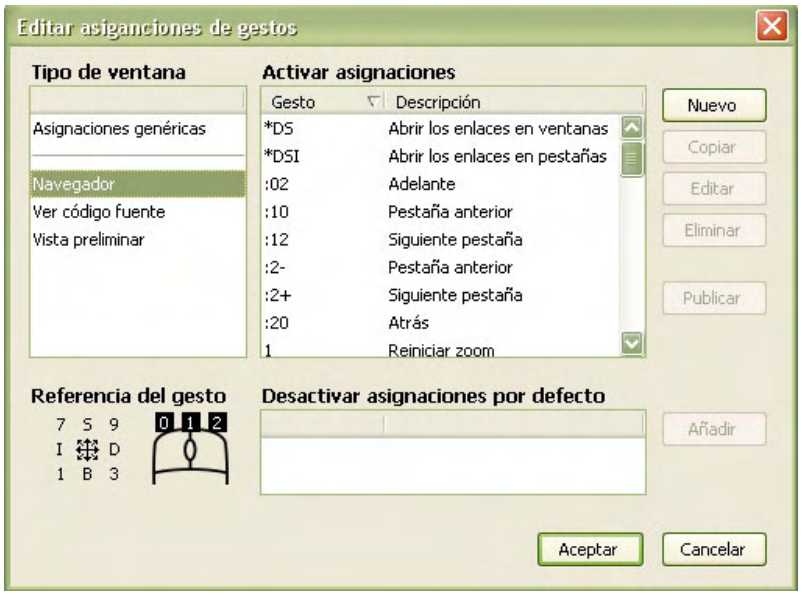


Figura 1.3: Menú de opciones de *Mouse Gestures*.

Actualmente, existe una comunidad que añade nuevos gestos que pueden ser descargados gratuitamente.

1.3 VIDEOJUEGOS

1.3.1 Antecedentes

Resulta difícil especificar cuál fue el primer videojuego, debido principalmente a las múltiples definiciones establecidas a lo largo de su historia; sin embargo, se pueden citar algunos desarrollos que son considerados como los orígenes [MALLIET2005]:

- En 1958 William Higginbotham creó *Tennis for Two* sirviéndose de un programa para el cálculo de trayectorias y un osciloscopio. Este programa era un simulador de tenis de mesa para entretenimiento de los visitantes del *Brookhaven National Laboratory*.
- En 1962 Steve Russell desarrolló un juego para computadora usando gráficos vectoriales: *Spacewar*. En este juego, dos jugadores controlaban la dirección y velocidad de dos naves espaciales que luchaban entre ellas.

En la década de los años 70 se dan los desarrollos que consolidan a esta nueva forma de entretenimiento. En 1972 se presenta la máquina recreativa llamada Pong, la cual es considerada como el primer videojuego y consistía en una rudimentaria partida de tenis o ping-pong [DEMARÍA2002].

Para la década de los 80, se comenzó con un fuerte crecimiento en el sector del videojuego con la llegada al mercado doméstico de nuevas consolas. En la década de los 90 rápidamente los videojuegos en 3D fueron ocupando un importante lugar en el mercado. En cuanto a las PC, se crearon las tarjetas aceleradoras 3D.

A partir del año 2000 compañías como Sony, Microsoft y Nintendo (por mencionar las más importantes) han lanzado consolas como PlayStation 1, 2 y 3, Xbox, Xbox 360, Gamecube, Wii, Game Boy Advance, Nintendo DS y PlayStation Portable (PSP). Algunas de estas consolas cuentan con características como controles inalámbricos que poseen sensores para registrar sus movimientos,

excelente calidad grafica y utilizan DVD o su contenido puede ser bajado de Internet.

Las computadoras personales, en la actualidad son una plataforma de juegos flexible puesto que se le puede añadir componentes con mejora constante, como las tarjetas gráficas y de sonido, accesorios, entre otros.

Como se puede apreciar, la rápida evolución de los videojuegos resulta ser una derivación del auge de la tecnología informática y de su difusión en las últimas décadas.

1.3.2 Concepto de videojuego

Para entender el término de videojuego, es necesario definir que un juego es un sistema de entretenimiento en el cual los participantes se implican en un conflicto artificial definido mediante reglas y que resulta en un estado final cuantificable.

Con respecto a la palabra video, se refiere al dispositivo de despliegue, para lo cual se hace uso de sistemas electrónicos conocidos como plataformas tecnológicas: computadoras personales y portátiles, consolas de videojuegos conectables a una televisión, teléfonos móviles, entre otras. En suma, un videojuego es un programa que contiene las reglas de un juego y aprovecha las características asociadas a una computadora.

En cuanto a la interacción con el usuario, normalmente los videojuegos se manipulan mediante dispositivos llamados controles (botones, *joystick*, teclados, *mouse*, entre otros) y además se hace uso de dispositivos como altavoces y hápticos (dispositivos con vibración o sensación de tacto) con la finalidad de recrear entornos y situaciones virtuales en los cuales el jugador puede controlar a uno o varios personajes (o cualquier otro elemento de dicho entorno), para conseguir uno o varios objetivos por medio de unas reglas determinadas.

Tomando en cuenta a los elementos que definen y componen a un videojuego, sin duda resulta ser un medio de entretenimiento cuyo desarrollo implica un proceso complejo y multidisciplinario. En los siguientes apartados, se describen los principales elementos que se deben considerar para el diseño de un videojuego, así como la clasificación de éstos junto con sus características.

1.3.3 Diseño de videojuegos

El diseño del juego no se puede reducir a un sistema de instrucciones y procesos discretos puesto que contiene elementos artísticos y funcionales. El diseño es el proceso de imaginar un juego, definir la manera de cómo trabaja y describir los elementos que lo componen: conceptual, funcional, artístico, entre otros.

Se pueden distinguir ciertos elementos esenciales para el diseño de un juego de acuerdo al tipo de juego que se desee construir, así como tres áreas específicas para su definición: historia y narrativa, mecánica base e interactividad [ROLLINGS2003].

- ***Historia y narrativa***. Los juegos cuentan con una historia cuya complejidad y profundidad depende del juego. Se puede tener dos situaciones: el juego es la historia o el jugador cuenta una historia al jugar. En cuanto a narrativa significa la parte de la historia que es contada por los diseñadores al jugador, es decir, es la parte representacional no interactiva de la historia.
- ***Mecánica base***. Son las reglas que definen la operación del juego y que son interpretadas por la computadora. Estos mecanismos describen la manera de cómo el juego trabaja y no la manera de cómo el software opera.
- ***Interactividad***. Es la manera que el jugador ve, oye y actúa dentro del ambiente del juego, es decir, la manera como juega. La interactividad implica aspectos como: gráficos, sonidos, interfaz de usuario, entre otros.

1.3.4 Géneros de videojuegos

En el transcurso de la historia de los videojuegos, han surgido diversos "géneros" que presentan características específicas para su clasificación en cierto tipo de juego. La clasificación que se ofrece en la tabla 1.3 se basa en factores, método de juego, objetivos y situaciones [ROLLINGS2003] [SCHOLAND2002].

Tipo	Características
Juegos de acción	• Incluyen normalmente desafíos físicos, sobre todo a nivel personal. • Raramente se incluyen desafíos estratégicos o conceptuales. • Las habilidades dominantes probadas por el jugador son tiempo de reacción y coordinación de vista y manos bajo presión. • Se considera el género más viejo y uno de los más simples. • Se divide en dos subgéneros: con disparo o sin disparo. • Los elementos a considerar para su diseño son: reglas, niveles, puntos de verificación, vidas, energía, límite de tiempo, marcador, aumento de energía y objetos coleccionables. • Ejemplos: Pac-Man, Quake III, Unreal Tournament, entre otros.
Juegos de estrategia	• Incluyen desafíos estratégicos, tácticos y logísticos. • Las habilidades dominantes probadas por el jugador son la administración de recursos escasos (tiempo, dinero, armas, entre otras), prever los comportamientos de los rivales y trazar estrategias de actuación para lograr ciertos objetivos. • La mayoría son para PC y se presentan de dos formas: juegos de estrategia por turnos y juegos de estrategia en tiempo real. • Los elementos a considerar en este género son: el tema, presentación y vista. • Ejemplos: Warcraft, Age of Empires, StarCraft, Black & White, entre otros.
Juegos de deportes	• Este género permite ejercitar diversas habilidades de coordinación y psicomotoras, así como profundizar en el conocimiento de reglas y estrategias de los deportes. • Los elementos a considerar para su diseño son: reglas (de los deportes), modos de competencia (un solo jugador, competitivo, cooperativo o en equipo), condiciones para ganar y perder, entorno (estadio o arena), modelo de interacción, perspectiva, interfaz del usuario y roles del jugador. • Ejemplos: FIFA, NBA, Formula I GrandPrix, Need for Speed, etc.

Tipo	Características
Simuladores y constructores	• Permiten experimentar e investigar el funcionamiento de máquinas, fenómenos, situaciones o procesos. • Los elementos a considerar para su diseño son: reglas, recursos, fuentes, *drains* (actividades que consumen recursos), convertidores, bloqueos, equilibrio estático y dinámico, y desastres. • Ejemplos: Simulador de vuelo Microsoft, Sim City, Tamagotchi, The Incredible Machine, Theme Park, entre otros.
Juegos de rol	• Se caracterizan por: exploración, colección o manipulación de objetos, solución de acertijos y un énfasis reducido en elementos de combate y acción. • Se manejan roles de jugador que mejoran con base a la experiencia y una historia. • Los elementos a considerar para su diseño son: entorno, tono emocional, modelo de interacción, vista sensible al contexto (primera o tercera persona), roles del jugador, estructura, historia (tensión dramática, búsqueda heroica) y desafíos. • Ejemplos: Monkey Island, Final Fantasy, Tomb Raider, entre otros.
Vida artificial	• Mascotas artificiales: son animales reales o de fantasía que simulan: emociones, humor, aprendizaje, reproducción, muerte entrenamiento, entre otras. Ejemplos: Petz, Tamagotchi, etc. • Sims: simula una familia de un hogar suburbano donde se puede manipular las acciones de las personas donde cada una tiene su propia personalidad y gustos. • Los elementos a considerar para su diseño son: genoma, mutación, vida, madurez, selección natural, comportamiento, herencia de características, entre otros.
Rompecabezas	• Desarrollan percepción espacial, lógica, imaginación y creatividad. • Los elementos a considerar para su diseño son: condiciones para ganar, desafíos y la interfaz del usuario. • Ejemplos: Big Brain, Tetris, entre otros.

Tabla 1.3: Clasificación de videojuegos.

En el siguiente apartado se profundiza en los juegos de estrategia en tiempo real, debido a que es el género al que pertenece el videojuego desarrollado.

1.3.4.1 Juegos de estrategia en tiempo real

Los RTS (del inglés *Real Time Strategy*) son juegos de estrategia que se distinguen por la ausencia de turnos, es decir, el tiempo transcurre de forma continua. Razón por la cual, cada jugador puede interactuar con el juego independientemente de las acciones realizadas por los demás participantes.

Consiste principalmente en asumir un rol de un general o dueño de un conjunto de tropas o unidades con ciertos recursos como edificaciones y armas, donde se debe combatir con otros jugadores por el dominio territorial o por algún objetivo. Generalmente se agregan otros factores como recolección de recursos, construcción de nuevas edificaciones y unidades, comercio o desarrollo de tecnología especifica para aumentar el poder de ataque o movilidad.

En sus inicios, se manejaba una representación visual fija de un objeto tridimensional en dos dimensiones. Posteriormente, se incorporó una perspectiva libre por lo que juegos de estrategia en tiempo real ofrecen un ambiente completamente tridimensional (3D), donde el usuario interactúa con sus unidades desde la perspectiva que él desee.

Los elementos a considerar en este tipo de juegos para su diseño y desarrollo son [ROLLINGS2003]:

- La perspectiva del ambiente del juego es 2D o 3D.
- El tema del juego es de conquista, exploración, comercio u otro.
- El entorno del juego es histórico, militar, ficción u otro.
- Si el juego implica unidades: tipo, características y cantidad.
- Mecanismos de producción de unidades: tiempo y costo.
- Que técnicas de Inteligencia Artificial se aplicarán.
- Diseño de la interfaz gráfica e interacción del usuario.

En la actualidad, los juegos de este tipo son construidos principalmente para computadoras personales y portátiles debido a que son diseñados en torno a la

flexibilidad y potencia de dispositivos de entrada muy específicos: teclado y *mouse*. A menudo el jugador requiere seleccionar con precisión varios elementos usando un apuntador, acción que generalmente es más rápida y precisa con el *mouse* que con un control de consola basado en botones. Es así, que la manipulación comprende las acciones generadas por el *mouse*, que son generalmente acompañadas con métodos abreviados de teclado, con los cuales se generan los comandos necesarios.

Lo anterior deriva en el hecho de que para la mayoría de los juegos RTS la interfaz, tanto de entrada (dispositivos) como de salida (despliegue en pantalla), tengan cierto estándar. Con respecto a la interfaz gráfica de usuario está conformada principalmente por elementos como:

- La ventana principal de perspectiva del ambiente o terreno que representa la zona donde el jugador interactúa con los elementos del juego.
- El área de despliegue de un radar o pequeño mapa de ubicación con respecto a la totalidad del terreno.
- La ventana o área de estado mediante la cual el usuario puede ver información que se asocia con el elemento seleccionado e indicadores como la disponibilidad de recursos y nivel de poder.

En el capítulo siguiente se describe el marco tecnológico, es decir, las técnicas y herramientas aplicadas en el desarrollo del sistema propuesto.

Capítulo 2

Marco tecnológico

El presente capítulo describe el soporte tecnológico (entornos y herramientas) utilizado en el desarrollo del sistema que comprende la tesis.

2.1 ARTOOLKIT

Es un conjunto de bibliotecas en lenguaje C y C++ que permiten desarrollar aplicaciones de Realidad Aumentada donde el usuario puede explorar el mundo real con objetos virtuales superpuestos [ARTOOLKIT]. El objetivo de la Realidad Aumentada es mejorar o aumentar la percepción e interacción del usuario en el entorno real a través del uso de contenido complementario de los objetos virtuales 3D que coexisten en el mismo espacio real.

ARToolKit se caracteriza por ofrecer un seguimiento rápido y preciso al utilizar técnicas de Visión por Computadora para calcular la posición real de la cámara y la orientación relativa de ciertas tarjetas marcadas en las cuales el programador puede sobreponer los objetos virtuales necesarios (figura 2.1).

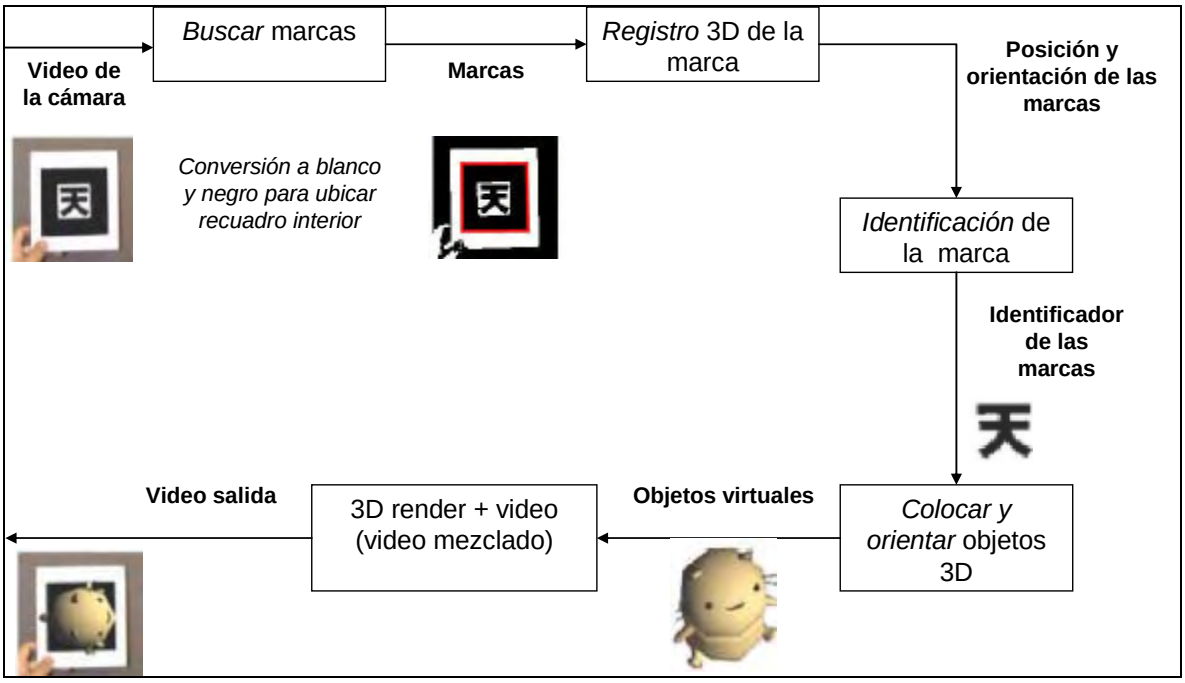


Figura 2.1: Fases de procesamiento de ARToolKit [ARTOOLKIT].

Esta biblioteca proporciona además, soporte y guía mediante una descripción completa de elementos y pasos de instalación, así como una serie de recomendaciones y ejemplos simples.

Finalmente, existen versiones extendidas originadas a través de la modificación y adaptación de ARToolKit que mejoran o se especializan en ciertas funciones. En el siguiente apartado se hace mención de la versión ARToolKitPlus, la cual es utilizada en el proyecto de tesis.

2.1.1 ARToolKitPlus

ARToolKitPlus es una biblioteca de software que se puede utilizar para calcular la posición de la cámara y la orientación relativa de ciertas marcas físicas en tiempo real. Surge internamente como parte del proyecto Handheld AR [HANDHELDAR] y consiste en una versión extendida del código de visión de ARToolKit.

En contraste con el ARToolKit, esta biblioteca no se sugiere para principiantes en el desarrollo de software de Realidad Aumentada, requiere que se tenga experiencia como programador en C++ y no cuenta con mucho soporte en cuanto a documentación. Sin embargo, se eligió esta biblioteca para realizar el seguimiento de las marcas necesarias en la interfaz propuesta debido a que ofrece un mejor rendimiento, rapidez y coincide con las características necesarias de ARToolKit eliminando aquellas que no se requieren.

Dentro de las principales características que tiene ARToolKitPlus están [WAGNER2007]:

- Se desarrollo con la finalidad de dar soporte a dispositivos móviles tales como *smartphones*, Asisitentes Personales Digitales (PDAs) y PCs ultra móviles (UMPCs).
- Está disponible bajo la licencia GLP de código abierto.
- Utiliza un marcador binario de patrones que permite una detección de marcas más rápida que la correspondencia de plantillas de marcas en donde se requieren más operaciones utilizada en ARToolKit.
- Incluye una heurística para hacer un ajuste de umbral automático.

- Maneja el efecto de oscurecimiento de los bordes externos del área de la imagen (vignetting) mediante compensaciones.
- Soporta la herramienta de calibración de cámara de ARToolKit.
- Permite una administración de memoria de los dispositivos.

Por último, el procesamiento de ARToolKitPlus se puede considerar como un sistema de flujo de datos donde como entrada están los datos de la imagen y se genera como salida las matrices de posición. Se tiene una secuencia de bloques o etapas que permiten definir el procesamiento a través de la selección de ciertas características o algoritmos (figura 2.2).

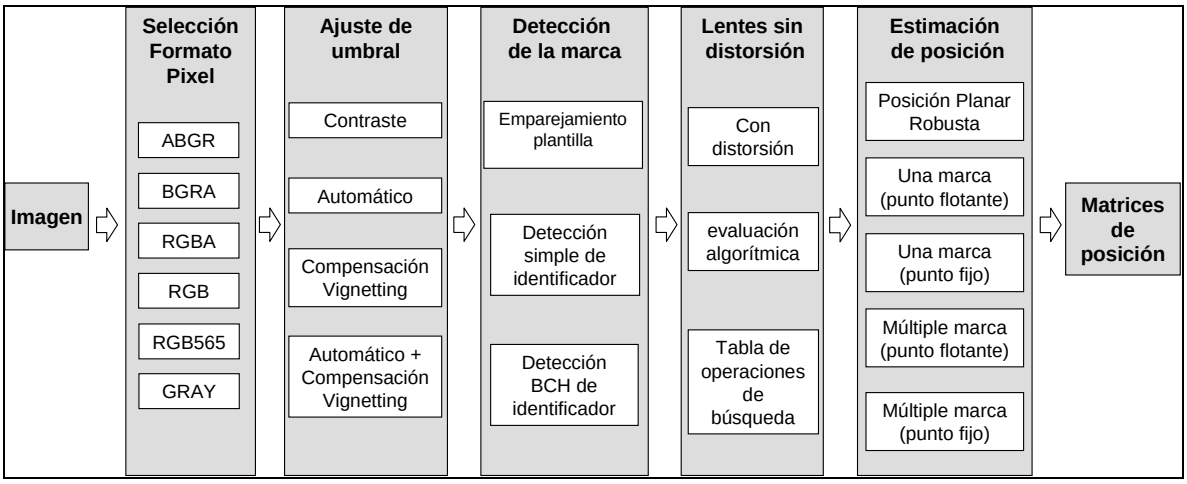


Figura 2.2: Flujo de datos en ARToolKitPlus.

2.2 OPENCV

OpenCV (*Open Source Computer Vision Library*) es una biblioteca de código abierto de visión por computadora desarrollada por Intel que proporciona funciones para el procesamiento de imágenes. En el año 2000 surge como un trabajo de un grupo de reconocidos investigadores en visión por computadora para realizar una nueva biblioteca de estructuras y funciones en lenguaje de programación C.

La biblioteca OpenCV es una API, de aproximadamente 300 funciones escritas en lenguaje C, que se caracteriza por lo siguiente [OPENCV]:

- Funciona en plataformas Mac OS X, Windows y Linux.
- Su uso es libre tanto para su uso comercial como no comercial.
- Entre sus muchas áreas de aplicación destacan: segmentación y reconocimiento de objetos, Interfaz Humano-Computadora (HCI), reconocimiento de rostros, seguimiento y comprensión de movimiento, estructura del movimiento (SFM) y robótica móvil.
- Incluye una biblioteca de aprendizaje automático que contiene: clasificador Bayesiano simple (*Naive Bayes classifier*), algoritmo K vecinos cercanos (K *nearest neighbors*), máquinas de soporte vectorial, árboles de decisión, redes neuronales y algoritmos: *Boosting*, *Random forest*, *Expectation Maximization* (EM).
- Proporciona un marco de trabajo de nivel medio-alto para el desarrollo de aplicaciones de visión por computadora en tiempo real.
- Proporciona un conjunto de funciones agrupadas en bloques (figura 2.3).

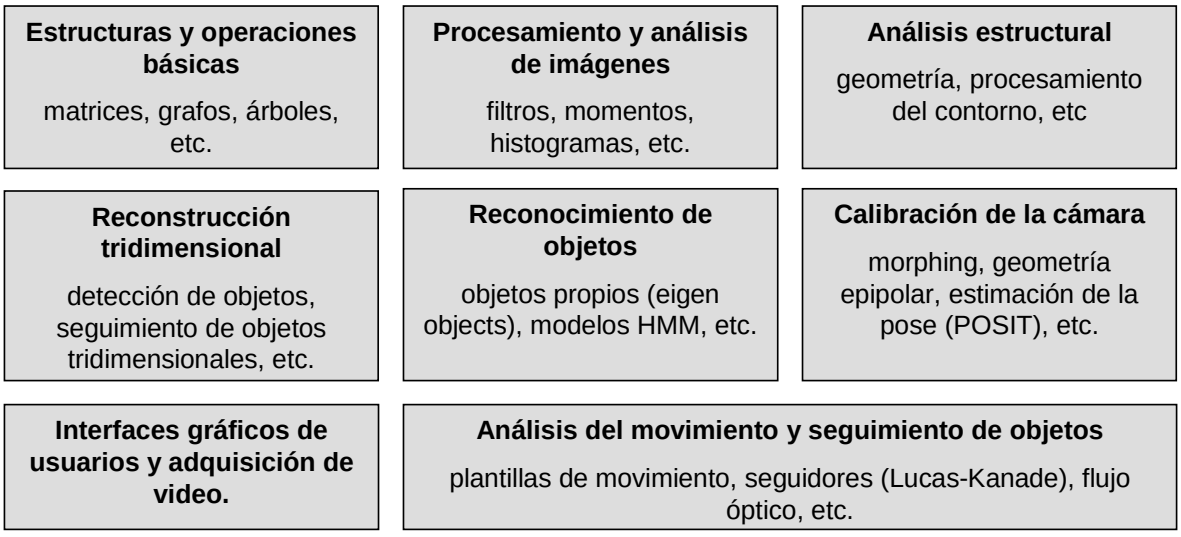


Figura 2.3: Funciones que proporciona la biblioteca OpenCV.

Las funciones utilizadas de OpenCV en el sistema propuesto corresponden al bloque orientado a funciones de interfaces gráficos de usuario y adquisición de

video, en donde a través del conjunto de funciones HighGUI se realizó la captura de video desde la cámara [OPENCVGUIDE].

Dentro de los paquetes de procesamiento de imágenes comerciales y software libre disponibles actualmente, OpenCV fue elegido ya que permite trabajar con la mayoría de las cámaras del mercado, el manejo de video a través de sus funciones es muy sencillo, además que es software libre y multiplataforma.

Cabe mencionar, que se hace uso también de DirectShow para la adquisición de video a través de la biblioteca VideoInput [VIDEOINPUT] puesto que ofrece un acceso más simple a sus funcionalidades y permite trabajar con la estructura de OpenCV como complemento.

2.3 OGRE 3D

El OGRE (Object Oriented Graphics Rendering Engine) es un motor gráfico escrito en el lenguaje de programación C++. Fue diseñado para hacer más fácil e intuitivo el desarrollo de aplicaciones que exploten al máximo el hardware acelerador de gráficos 3D, a través de una interfaz orientada a objetos.

Es una biblioteca estable, confiable, flexible, multiplataforma y completamente equipada para desarrollar aplicaciones gráficas 3D en tiempo real. Una ventaja principal que tiene Ogre3D es que es un proyecto de código abierto bajo licencia LGPL por lo que su uso es gratuito.

Dentro de las principales características de este motor gráfico destacan [OGRE]:

- Proporciona una interfaz de objetos y clases de alto nivel cuyo propósito es proporcionar una solución general para la representación de los gráficos.
- Soporta las plataformas Windows, Linux y Mac OS X.
- Usa un lenguaje de descripción de materiales que permite su administración de forma independiente al código fuente de la aplicación.
- Brinda soporte para texturas PNG, JPEG y BMP.

- En cuanto al modelado, permite la edición externa desde Milkshape, Autodesk 3ds Max, entre otros.
- Existen módulos especialmente diseñados para Ogre3D, que permiten tener soporte para sonido, física, entre otros.
- Maneja muchas de sus características a través de scripts (archivos de texto que pueden ser editados) para hacer más fácil su configuración. Los scripts que utiliza son: materiales, fuentes, partículas, entre otros.
- Cuenta con la documentación y soporte necesario para su instalación y uso.
- Existen aplicaciones que demuestran su utilidad en el desarrollo de videojuegos [ANKH2006].
- La calidad gráfica que ofrece es comparable con motores gráficos 3D comerciales.
- Permite crear ambientes con una mayor sencillez en configuración y comunicación con el hardware en comparación con la utilización de APIs de bajo nivel como DirectX y OpenGL.

2.3.1 Estructura general

Existen tres elementos básicos y fundamentales en OGRE 3D [JUNKER2006]: *entidades, nodos de escena y manejadores de escena*.

- **Entidades:** es la abstracción que representa a un objeto pantalla y puede representar elementos como: auto, robot, árbol, etc.
- **Nodos de escena:** es la abstracción que se encarga de las características de las entidades y no tiene una representación directa en la pantalla. Es una especie de contenedor de entidades y otros elementos como cámaras o luces. Se puede formar una jerarquía de nodos y las posiciones de los nodos de escena son relativas a las de sus padres.
- **Manejadores de escena:** es la abstracción orientada al tratamiento y seguimiento de todos los elementos que se dibujan en la pantalla, es decir, es el encargado de todo lo que tenga que ver con el dibujo del ambiente

virtual: dibujo del terreno, características del ambiente, crear y poner todos los elementos de la escena, entre otros.

En cuanto a la descripción de la arquitectura de este motor gráfico, se tiene que existe un objeto raíz mediante el cual se puede tener acceso a las clases encargadas del manejador de escena, dibujo de ventanas, cargar plugins, entre otras.

Este objeto raíz representa el objeto organizador y el resto de las demás clases de OGRE 3D se pueden clasificar en tres grupos: administración de escena, administración de recursos y dibujo. En la figura 2.4 se muestra el objeto raíz y las clases más representativas.

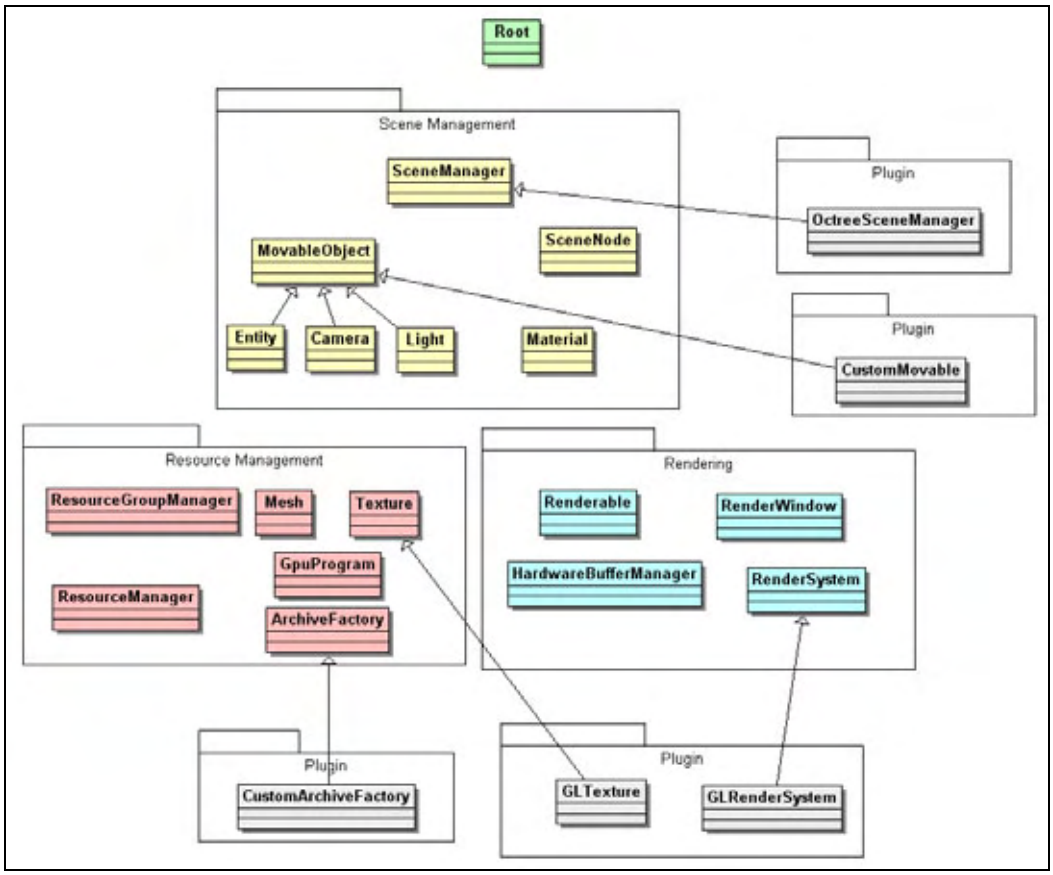


Figura 2.4: Diagrama de clases de OGRE 3D [OGRE].

En el siguiente capítulo se abarca el tema del desarrollo de sistema propuesto. Se describe la arquitectura e implementación del prototipo.

Capítulo 3

Descripción del sistema desarrollado

El presente capítulo aborda la estructura y las características principales del sistema construido. Se describe la arquitectura general y la descripción de cada uno de los componentes utilizados.

3.1 ARQUITECTURA DEL SISTEMA

La arquitectura global del sistema presentada en la figura 3.1 muestra la composición del sistema reconocedor de gestos de la mano (VInput) y la aplicación que hace uso de este (videojuego RTS llamado Bellum), así como la manera en que interactúa con los demás componentes.

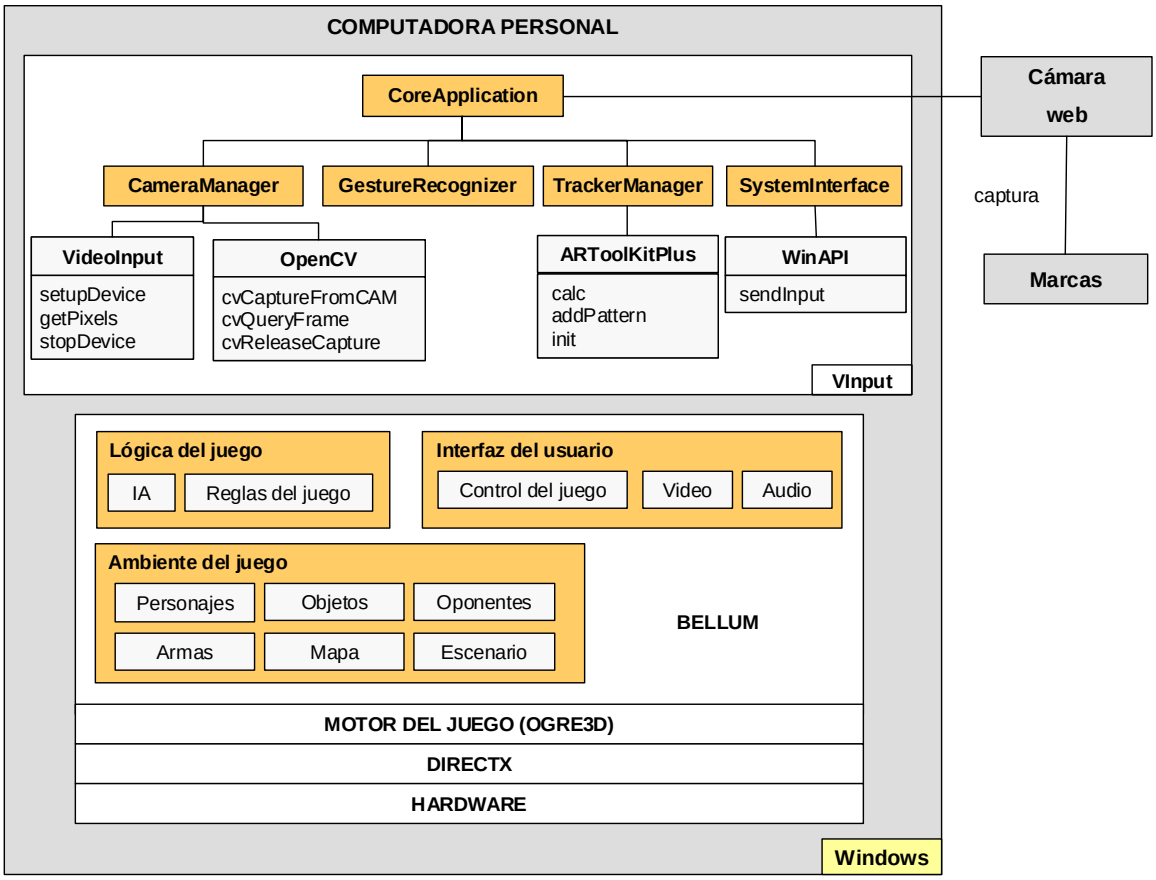


Figura 3.1: Arquitectura general del Sistema.

Como se puede apreciar, se ejecuta el sistema en una computadora personal de manera local bajo el sistema operativo Windows 2000 o superior. En cuanto a los elementos externos necesarios se tienen: la cámara web, las marcas y una iluminación adecuada para el reconocimiento de las marcas utilizadas.

El sistema está formado por dos módulos principales: VInput y Bellum. Para el módulo VInput se manejan las clases `CoreApplication, CameraManager, TrackerManager, SystemInterface y GestureRecognizer`; y éstas a su

vez hacen uso de las bibliotecas OpenCV, VideoInput, WinAPI y ARToolKitPlus para el reconocimiento de gestos de las manos mediante marcas (sección 3.2.2) y transformarlos a comandos del sistema.

Los comandos generados por el módulo de VInput son utilizados para que el usuario pueda interactuar con el videojuego desarrollado llamado Bellum. El comportamiento de Bellum está determinado por los elementos de interfaz del usuario, lógica y ambiente virtual; mientras que el motor OGRE 3D le proporciona la plataforma de ejecución realizando la intermediación con el hardware gráfico a través de la API DirectX.

En los siguientes apartados se describe con mayor detalle los componentes mencionados, así como el proceso mediante el cual el sistema opera.

3.2 PROCESO DE CAPTURA DE GESTOS

El proceso de captura de gestos es realizado por el módulo VInput y está conformado por tres etapas: adquisición de video, seguimiento y reconocimiento de gestos (figura 3.2). A través de este procedimiento se obtienen los eventos del sistema para manipular a Bellum.

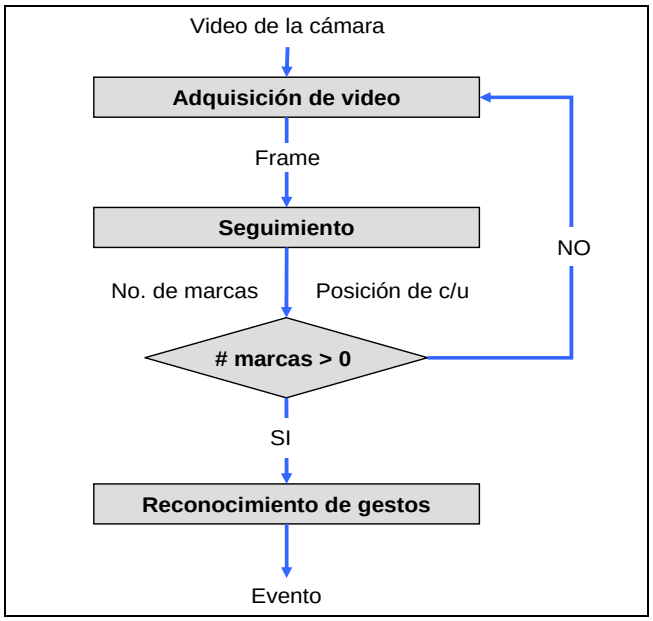


Figura 3.2: Proceso de captura de gestos.

3.2.1 Adquisición de video

El sistema reconocedor de gestos, está diseñado para obtener video de un amplio rango de cámaras puesto que hace uso de las bibliotecas OpenCV y VideoInput, por lo tanto, todas aquellas que soporten “Video for Windows” (VfW) o “DirectShow” pueden ser utilizadas. Algunas cámaras pueden no ofrecer la calidad requerida, es por ello, que se presentan las siguientes características recomendadas:

- Conexión usb 2.0 de alta velocidad o Firewire (IEEE 1394).
- Video sin compresión.
- Tasa de 30 cuadros por segundo.
- Manejar video de resolución 320 x 240 o 640 x 480 píxeles.

Otros aspectos a considerar para la adquisición de video son:

- El entorno debe tener iluminación suficiente y constante. Este es un requerimiento heredado de ARToolKitPlus para el tratamiento de la imagen.
- Se sugiere un fondo de superficie homogénea tanto en color como textura para evitar ruido en la imagen.
- El usuario debe tener colocadas las marcas correspondientes y realizar los movimientos para que puedan ser captados por la cámara.

En esta etapa, la clase CameraManager inicializa y configura la cámara a utilizar para la adquisición del video. Primero, se verifica la existencia de una cámara, en caso de no existir se notifica que no es posible inicializarla. Cabe señalar que VInput detecta la primera cámara que tiene registrada el sistema operativo, por lo que si existen varias conectadas no se puede elegir entre ellas.

Una vez inicializada la cámara, se empiezan a tomar los frames (cuadro de video a tratar) que son las entradas para la siguiente fase: seguimiento.

3.2.2 Seguimiento

Como ya se ha mencionado, se optó por utilizar la biblioteca ARToolKitPlus para el seguimiento. Para esta etapa, se hace uso de la clase `TrackerManager` que inicializa el tracker de ARToolKitPlus y los parámetros requeridos como número, tipo y tamaño de marcas, así como la carga de aquellas que serán reconocidas.

Estas marcas que utiliza ARToolKitPlus deben ser planas y estar formadas por un marco de color negro que contiene en su interior un patrón (figura 3.3). El reconocimiento de la marca ocurre en dos pasos: primero se reconoce el contorno cuadrado y después se correlaciona el patrón interior con los patrones previamente cargados en memoria.

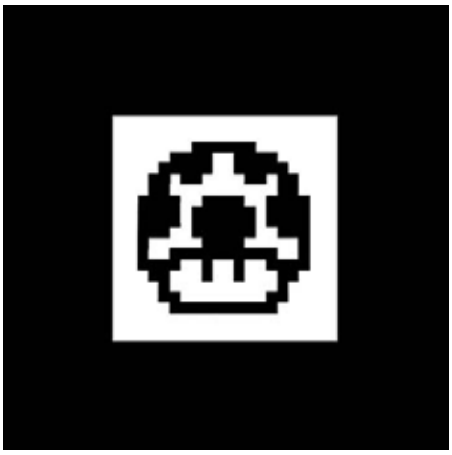


Figura 3.3: Tipo de marca de ARToolkitPlus.

Cabe puntualizar, que las marcas que maneja el sistema son impresas en hojas de papel y adheridas a la mano del usuario con cualquier tipo de pegamento.

Una vez terminada la inicialización del tracker, se recibe un frame (cuadro de video) de la cámara y ARToolKitPlus realiza el siguiente procedimiento:

1. La imagen se transforma en binaria a partir del umbral determinado por defecto. Todos los píxeles cuya intensidad supere el umbral pasan a ser

negros, el resto pasan a ser blancos, dando una imagen en blanco y negro donde las zonas oscuras pasan a ser blancas y viceversa.

2. En la imagen binaria, se buscan todos los marcos blancos que pudiesen ser candidatos de contener marcas conocidas.
3. Una vez obtenidos los candidatos se comprueba que contienen un patrón conocido de entre los almacenados en memoria, de modo que se eliminan falsos positivos en la detección.
4. Para aquellas marcas donde se encuentren patrones conocidos se obtienen los datos de posición e inclinación relativa a la cámara, que se guardarán en una matriz de transformación geométrica homogénea.

Finalmente, cuando el tracker da como salida el número de marcas encontradas y la posición de cada una, se pasa a la etapa de reconocimiento de gestos.

3.2.3 Reconocimiento de gestos

Los gestos son precargados a partir de un archivo de configuración. En este archivo es almacenado cada gesto como una cadena formada por la combinación de cuatro tipos de elementos: U, D, L, R (*Up, Down, Left, Right* respectivamente). Cabe señalar, que a través del archivo de configuración se pueden modificar las combinaciones establecidas en las cadenas, puede ser consultado en el apéndice A. En la sección 3.2.3.1 se describe cada uno de los gestos implementados.

Ahora bien, se manejan dos tipos de gestos con base al número de marcas utilizadas. Para el reconocimiento de los gestos que implican una sola marca se realiza el siguiente proceso:

1. La clase `GestureRecognizer` recibe de `TrackerManager` la coordenada de la marca (centro geométrico) con respecto a la imagen tomada que representa la posición actual del puntero ($Mouse_T$) y guarda la posición anteriormente recibida ($Mouse_{T-1}$) (figura 3.4).

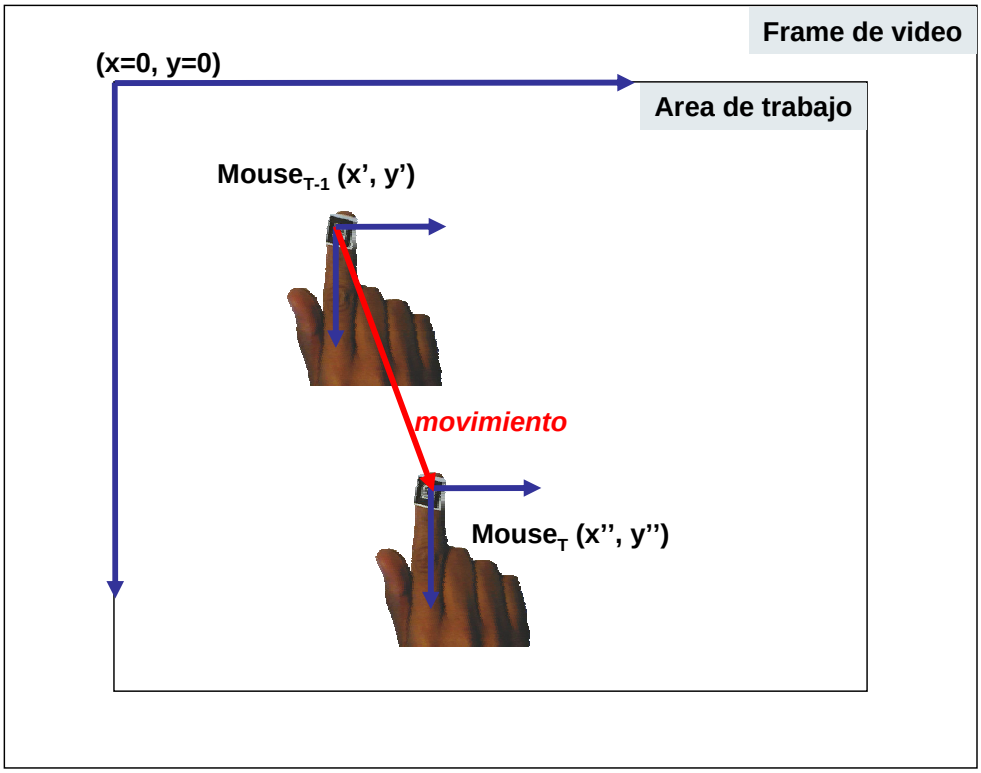


Figura 3.4: Coordenada de la marca con respecto a la imagen.

2. Se calcula la diferencia entre la posición anterior del puntero (x', y') con la actual (x'', y'').

   Δx = x'' – x'
   Δy = y'' – y'

3. Se determina a través de las siguientes condiciones la dirección hacia donde se ha movido la marca:

   - ***Caso 1*:** Si **|Δx|** es menor que **|Δy|** y además **Δy** es menor que cero, entonces el movimiento es ***Down***.
   - ***Caso 2:*** Si **|Δx|** es menor que **|Δy|** y además **Δy** es mayor o igual a cero, entonces el movimiento es ***Up***.
   - ***Caso 3:*** Si **|Δy|** es mayor o igual que **|Δx|** y además **Δx** es menor que cero, entonces el movimiento es ***Left***.
   - ***Caso 4:*** Si **|Δy|** es mayor o igual que **|Δx|** y además **Δx** es mayor o igual a cero, entonces el movimiento es ***Right***.

Estas cuatro direcciones generadas corresponden a la ubicación espacial de la figura 3.5.

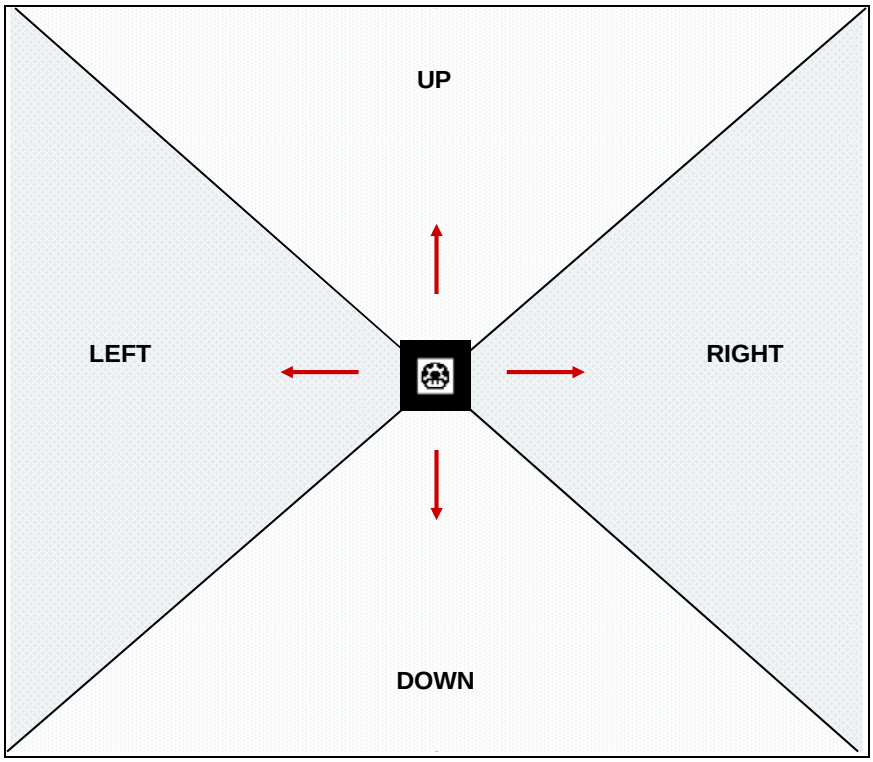


Figura 3.5: Coordenadas de ubicación.

Cabe señalar, que cuando una coordenada quede en la frontera de dos ubicaciones, se coloca en left o right según sea el caso.

4. El movimiento de la marca da como resultado una secuencia de direcciones, la clase `GestureRecognizer` determina el gesto realizado por el usuario mediante la comparación de la secuencia generada con la combinación de direcciones que forman a los gestos almacenados. Cuando se identifica un gesto, se envía el evento a la clase `SystemInterface`, la cual manda el mensaje correspondiente al sistema operativo.

5. Cuando existe movimiento de la marca, la clase `SystemInterface` recibe de `TrackerManager` la posición y mueve el puntero del mouse. Para representar este movimiento en la pantalla se hace una

correspondencia entre la resolución de ésta con la imagen capturada. Se maneja un borde de seguridad que define el área de trabajo de representación en la pantalla puesto que no se podría acceder a las orillas ya que la marca requiere ser capturada en su totalidad para su detección (figura 3.6).

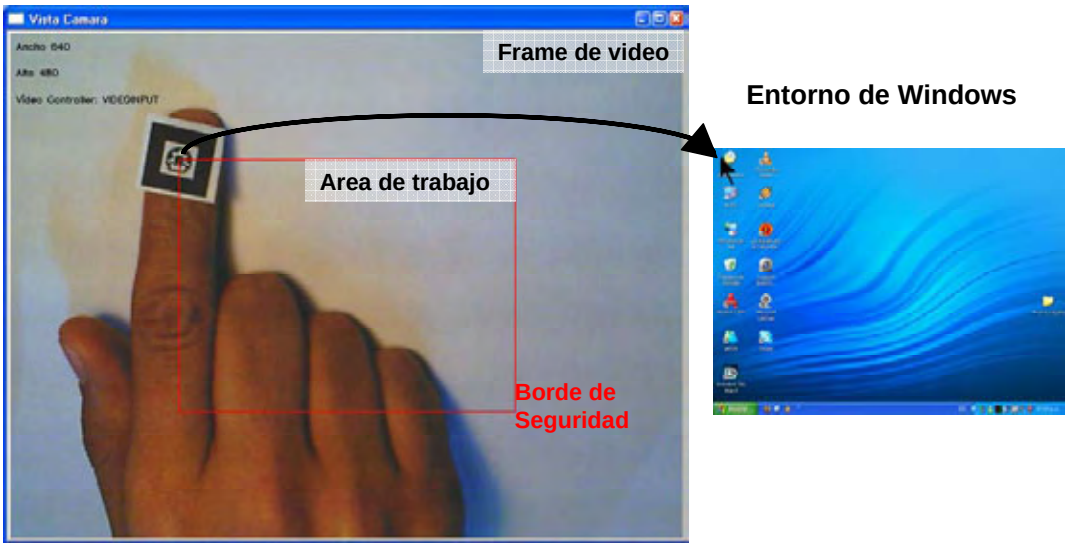


Figura 3.6: Correspondencia de la resolución de pantalla con la imagen capturada.

Con respecto a los gestos que implican dos marcas, el *Zoom In* y *Zoom Out*, no se realiza la etapa de comparación de secuencias de direcciones. En su lugar, `GestureRecognizer` mide la distancia entre las posiciones de las marcas recibidas: si la distancia aumenta se traduce como un *Zoom Out* o si la distancia disminuye se interpreta como un *Zoom In*.

Cuando se identifica que la acción es un *Zoom Out* o *Zoom In*, `GestureRecognizer` notifica a `SystemInterface` para que se genere el evento de sistema correspondiente.

3.2.3.1 Gestos implementados

En este apartado se describe cada uno de los gestos que son reconocidos por el sistema desarrollado. En la tabla 3.1 se muestra gráficamente la posición y

movimiento que el usuario realiza con las marcas para su generación, así como su descripción.

Para cada gesto es necesario que las marcas sean colocadas bajo la cámara y se tenga un área de trabajo cómoda, como se muestra en la figura 3.7.

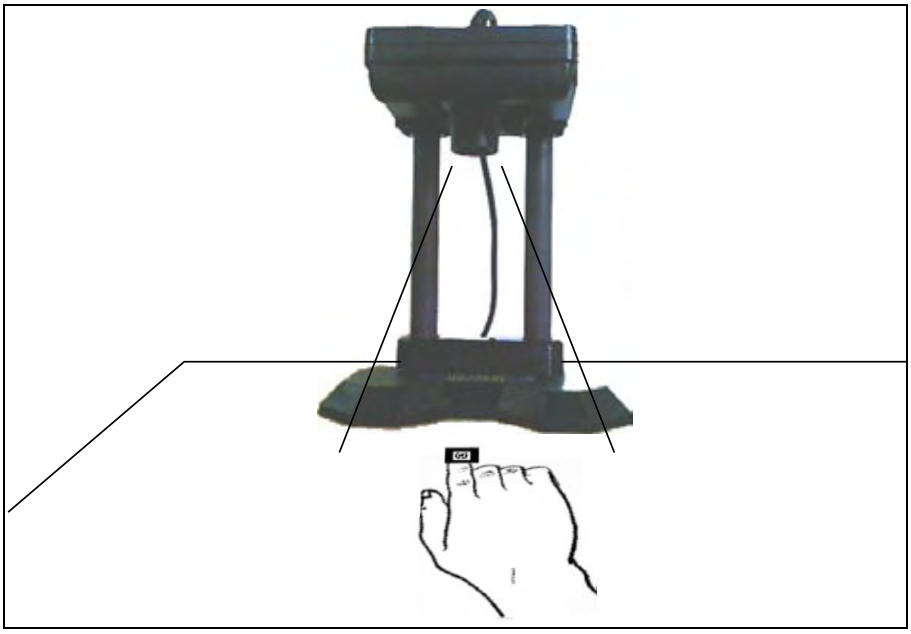


Figura 3.7: Colocación de cámara y marcas.

Cabe puntualizar que se sugiere utilizar los dedos de las manos, sin embargo puede colocarse la marca en donde se desee, siempre y cuando le sea posible a la cámara captar tanto la marca(s) como el movimiento requerido.

La mayor parte de las pruebas del prototipo desarrollado se realizaron bajo las siguientes condiciones:

- Marcas de tamaño de 1.5 x 1.5 centímetros.
- La cámara ubicada paralelamente a la superficie de trabajo de color blanca y lisa a una distancia de 20 centímetros.

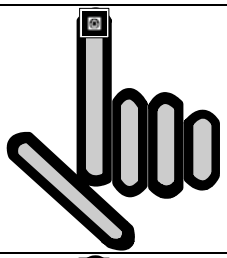
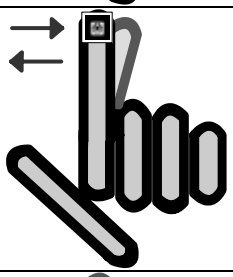
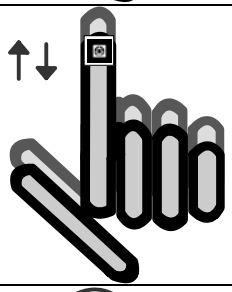
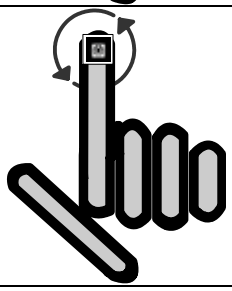
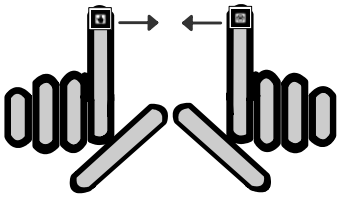
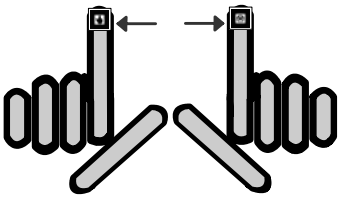
Posición	Gesto	Descripción
	Movimiento del puntero	Generar el movimiento de desplazamiento continuo sobre una superficie plana hacia donde se desea llevar el puntero en la pantalla.
	Clic izquierdo (LeftClick)	Generar el movimiento de desplazamiento continuo sobre una superficie plana de izquierda (L) a derecha (R) 3 veces (LRLRLR) sobre el objeto que se desea hacer la acción de clic izquierdo en la pantalla.
	Clic derecho (RightClick)	Generar el movimiento de desplazamiento continuo sobre una superficie plana de arriba (U) y abajo (D) 3 veces (UDUDUD) sobre el objeto que se desea hacer la acción de clic derecho en la pantalla.
	Botón medio del mouse (MiddleClick)	Generar el movimiento de desplazamiento continuo sobre una superficie plana de forma circular en sentido contrario a las manecillas del reloj, equivalente a generar la secuencia arriba, izquierda, abajo y derecha 3 veces (ULDRULDRULDR) sobre el área que se desea hacer la acción.
	Zoom In	Generar el movimiento de desplazamiento continuo sobre una superficie plana de forma que las dos marcas se alejen entre si horizontalmente sobre el área que se desea hacer la acción de *Zoom In* en la pantalla.
	Zoom Out	Generar el movimiento de desplazamiento continuo sobre una superficie plana de forma que las dos marcas se acerque entre si horizontalmente sobre el área que se desea hacer la acción de *Zoom Out* en la pantalla.

Tabla 3.1: Gestos implementados.

3.3 VIDEOJUEGO BELLUM

Como ya se ha mencionado, Bellum es un juego para computadora personal del género de estrategia en tiempo real que utilizará el software de reconocimiento de gestos de las manos para su control y manipulación.

Este videojuego es una aplicación interactiva, donde se le muestra al usuario por medio de gráficos tridimensionales un escenario de batalla. El usuario (jugador) es una unidad de robots que se enfrenta a otra unidad de robots que tiene que destruir. Para ganar es necesario encontrar y eliminar a todos los enemigos a través de disparos con el arma que se le asigna a cada uno de los robots.

Puesto que el objetivo de Bellum es mostrar la interacción del usuario al hacer uso de las principales acciones de este tipo de aplicación con el reconocimiento de gestos de las manos a través de marcas, se construyeron sólo los elementos de un juego de estrategia en tiempo real que permitieran cumplir con este objetivo.

Para la codificación de los elementos que conforman a Bellum (véase figura 3.1) se utilizó el entorno integrado de desarrollo Microsoft Visual Studio 2005 con el lenguaje de programación C++. Además, se anexaron las bibliotecas del motor gráfico OGRE 3D que proveen las clases necesarias para generar una ventana en el sistema operativo y el dibujo del ambiente virtual.

Mediante el lenguaje de programación C++ se construyeron los objetos y escena del ambiente virtual, el flujo del juego y los elementos para la interfaz gráfica del usuario. En los siguientes apartados se describen los elementos relacionados con la estructura y organización: gráficos, flujo del juego, inteligencia artificial e interfaz de usuario.

3.3.1 Gráficos

Los gráficos de Bellum están determinados por los objetos que se visualizan en la escena. Estos objetos fueron creados con herramientas de modelado 3D y edición de imágenes, para posteriormente ser exportados al formato que utiliza OGRE 3D. El ambiente virtual de Bellum consta de los modelos: terreno, robot y vegetación.

Para la creación del terreno se tiene un mapa de altura (figura 3.8) que es representado por medio de una imagen bidimensional que tiene una profundidad de 8 bits, en escala de grises. Cada píxel corresponde a un punto de altura del terreno a representar, 0xFF (color negro) es el valor más bajo y 0x00 (color blanco) es el valor máximo de altura. Por último, son colocadas en este mapa las texturas correspondientes.

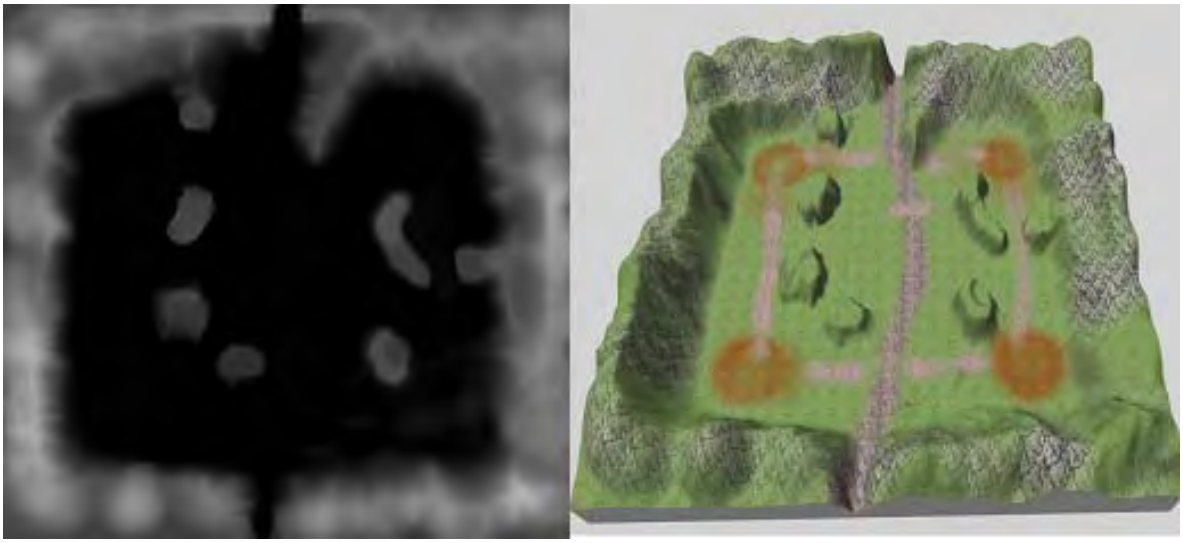


Figura 3.8: Mapa de altura del terreno del videojuego Bellum.

Con respecto a los modelos de robot y vegetación del terreno, fueron creados y editados en el modelador MilkShape y sus texturas con Adobe Photoshop. En la tabla 3.2 se muestran las posibles animaciones tanto para el robot como los elementos de vegetación en el ambiente virtual.

Modelo	Animación	Descripción
Robot	Desocupado (Idle)	Cuando permanece en estado de espera.
	Caminando (Walk)	Cuando se desplaza de un punto a otro en el terreno.
	Muerto (Die)	Cuando es derrotado.
	Disparando (Shoot)	Cuando dispara a un enemigo.
	Extra	Se utiliza para propósitos múltiples.
Tile	Desocupado (Idle)	Se utiliza para propósitos múltiples en objetos o elementos del terreno, por ejemplo movimiento de las ramas de los árboles con el viento.

Tabla 3.2: Animaciones de los modelos del videojuego Bellum.

Finalmente, se hizo uso de los scripts de partículas de OGRE 3D para crear los efectos de explosión, humo y fuego cuando el personaje ataque a enemigos (disparo de balas) o sea derrotado.

3.3.2 Flujo del juego

El flujo del juego es controlado por un manejador de estados, el cual controla el estado activo y conforme a la situación actual, determina el siguiente. Los cuatro estados posibles para el flujo son:

- IntroState. Estado inicial que permite manejar los eventos: comenzar el juego, acceder al tutorial de gestos o finalizar la aplicación.
- PlayState. Estado que maneja el desarrollo del juego: despliega los gráficos y la información del juego de acuerdo al progreso, acciones del usuario y cálculos realizados por la inteligencia artificial del juego (ver sección 3.3.3).
- PauseState. Estado que detiene la ejecución del juego: progreso y acciones de cada elemento. Permite manejar los eventos de continuar con el juego o terminar la aplicación.

- EndState Estado final que termina con la ejecución de la aplicación.

3.3.3 Inteligencia Artificial

La Inteligencia Artificial, en la construcción de videojuegos, permite desarrollar aplicaciones con comportamientos complejos. Esta área de conocimiento proporciona un conjunto de técnicas y algoritmos de programación diseñados para simular que los personajes o elementos, que interactúan con el usuario u otros componentes en el ambiente virtual, tengan un comportamiento inteligente e independiente.

Este aspecto fue considerado para la construcción del videojuego Bellum. Para ello, se hizo uso de las técnicas de Maquinas de Estados Finitos (FSM), algoritmo de Dijkstra y campos potenciales para proporcionar información y acciones a los personajes para que actúen conforme a la situación que enfrentan en el juego.

3.3.3.1 Máquina de Estados Finitos

Las Máquinas de Estados Finitos (FSM por sus siglas en inglés) son máquinas abstractas que poseen un número limitado de estados predefinidos. Una FSM también define el conjunto de condiciones que determinan cuando cada estado debe cambiar, por lo que su comportamiento está descrito por el estado activo.

Las FSM se utilizan para modelar el comportamiento de un sistema, y pueden ser representadas de forma genérica como lo muestra la figura 3.9 donde cada estado es ilustrado por un círculo. Se tienen definidos cuatro estados {Xi, X1, X2, X3} y cinco funciones de transición {Y1, Y2, Y3, Y4, Y5}. El estado Xi es el inicial y la función de transición que provee un estimulo para cambiar al estado X1 es Y1. Cabe destacar, que en los estados X3 y X2 existe más de una transición

que puede generar saltos a estados diferentes, lo cual brinda ramificaciones que pueden proporcionar una mayor complejidad al sistema.

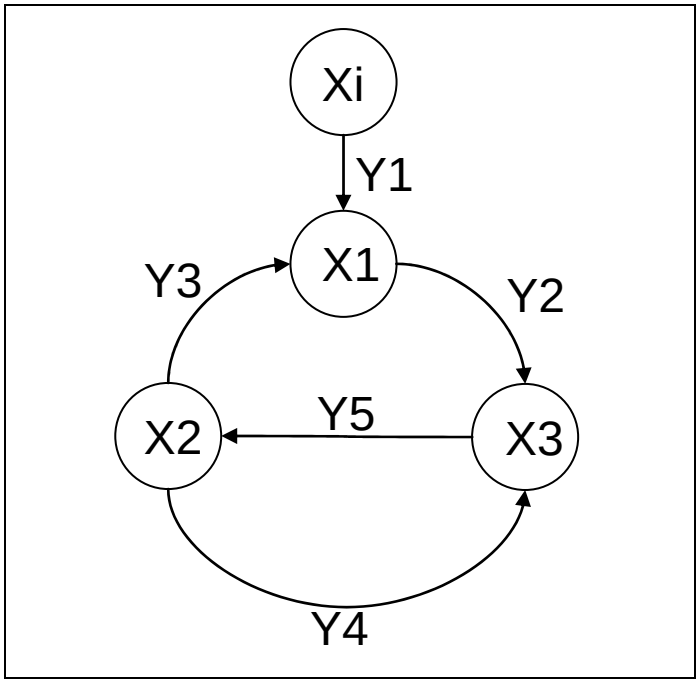


Figura 3.9: Diagrama genérico de una Máquina de Estados Finitos.

En el videojuego Bellum se requiere modelar los comportamientos de los personajes los cuales son determinados por su movimiento dentro del ambiente virtual, la detección, persecución y ataque a enemigos. Para determinar la acción a realizar de cada personaje, se utilizo una Máquina de Estados Finitos donde el comportamiento del personaje está representado en los estados, y las acciones están restringidas por ciertas condiciones [BUCKLAND2004].

En la tabla 3.3 se muestran los estados (figura 3.10) que definen el comportamiento para los personajes que son controlados por la computadora y los que son controlados por el usuario, las transiciones que son movimientos de un estado a otro y las condiciones que deben cumplirse para permitir un cambio de estado.

Estado actual	Condición	Estado de transición	Unidad
Esperar	Si está vivo y manejador ordena movimiento	*Patrullar*	controlada por la computadora
Esperar	Si está vivo y existe enemigo cercano	*Atacar*	
Esperar	Si energía actual está agotada	*Morir*	
Patrullar	Si está vivo y existe enemigo cercano	*Atacar*	
Patrullar	Si está vivo y llegó a la zona objetivo	*Patrullar*	
Patrullar	Si energía actual está agotada	*Morir*	
Atacar	Si está vivo y energía actual es menor que el umbral de dolor (energía actual/2)	*Huir*	
Atacar	Si está vivo y no hay enemigos cercanos	*Patrullar*	
Atacar	Si energía actual está agotada	*Morir*	
Huir	Si está vivo y llegó a la zona objetivo	*Patrullar*	
Huir	Si energía actual está agotada	*Morir*	
Esperar	Si está vivo y usuario ordena movimiento	*Patrullar*	controlada por el usuario
Esperar	Si energía actual está agotada	*Morir*	
Patrullar	Si está vivo y existe enemigo cercano	*Atacar*	
Patrullar	Si está vivo y llegó a la zona objetivo	*Esperar*	
Patrullar	Si energía actual está agotada	*Morir*	
Atacar	Si está vivo y no hay enemigos cercanos	*Esperar*	
Atacar	Si energía actual está agotada	*Morir*	

Tabla 3.3: Estados de transición de los personajes del videojuego Bellum.

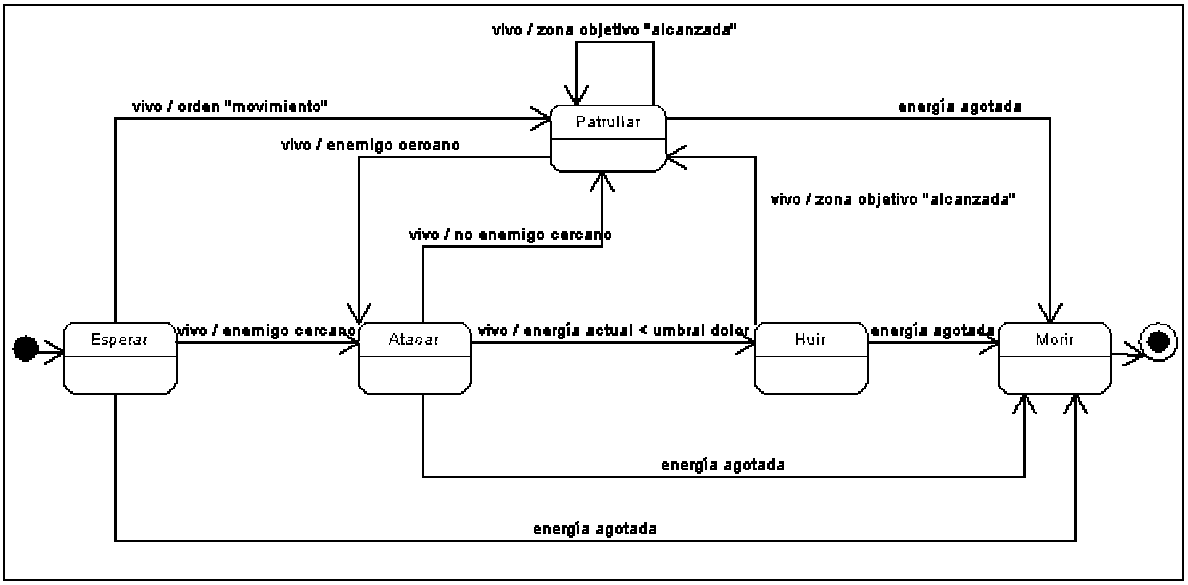


Figura 3.10: Diagrama de estados de los personajes del videojuego Bellum.

3.3.3.2 Algoritmo de Dijkstra

El Algoritmo de Dijkstra [CORMEN2001], también llamado algoritmo de caminos mínimos, determina el camino más corto dado un nodo origen al resto de los nodos de un grafo dirigido ponderado, donde cada arista tiene un peso positivo que representa la distancia entre los nodos que enlaza.

El procedimiento consiste en tomar en cada etapa el nodo cuya distancia sea menor con respecto al origen y con base al análisis del costo para ir a los nodos adyacentes se va trazando la ruta más corta hacia el nodo objetivo. El algoritmo se define de la siguiente manera:

```
DIJKSTRA (G, w, S)
INICIALIZAR-ORIGEN (G, S)
        Por cada nodo v perteneciente a V
                d[v] ← ∞
                ∏[v] ← null
        d[s] ← 0
S ← 0
Q ← V [G]
Mientras Q ≠ 0
        u ← EXTRAER-MINIMO (Q)
        S ← S ∪ {u}
        Por cada nodo v que sea adyacente a u
                REFINAR (u, v, w)
                        Si d[v] > d[u] + w (u, v) entonces
                                d[v] ← d[u] + w (u, v)
                                ∏[v] ← u
```

Implementación del algoritmo de Dijkstra [CORMEN2001].

Donde:

V es un conjunto de nodos y *E* es un conjunto de aristas

G = (V, E) es el grafo dirigido ponderado positivamente

S es el conjunto de nodos que representa el camino más corto

Q es la cola de prioridad de nodos

d[v] se define como la distancia de *s* a *v*

∏[v] se refiere al nodo padre de *v*

w (u, v) es el peso positivo del nodo *u* a *v*

Para tener una ejecución más rápida de la búsqueda de caminos, se optó que por cada nodo se almacenara una matriz con las rutas más cortas hacia todos los nodos del grafo al inicio de la ejecución del juego.

De lo expuesto en el apartado de Máquina de Estados Finitos, para los estados de patrullar y huir se implementó el algoritmo de Dijkstra para la búsqueda de caminos entre las áreas del terreno del juego. Para ello, se tiene un grafo en el ambiente virtual donde cada nodo representa un área y las aristas los caminos mediante los cuales los personajes se mueven entre las áreas de la escena (figura 3.11).

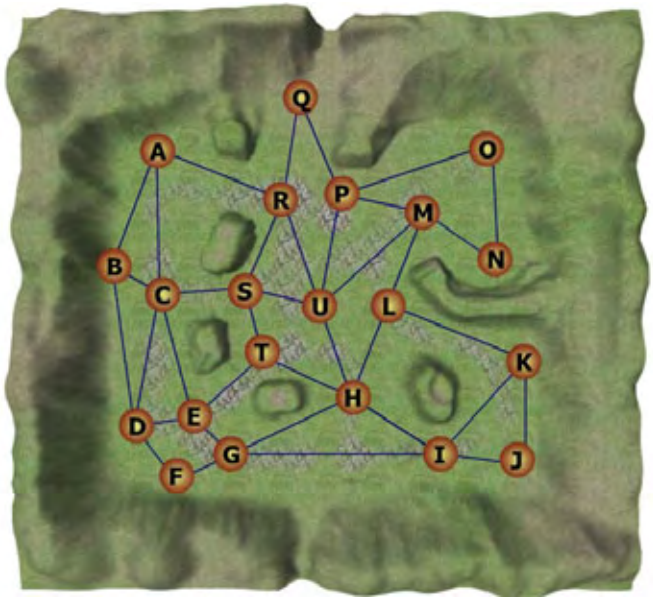


Figura 3.11: Grafo de áreas para el terreno del videojuego Bellum.

3.3.3.3 Campos potenciales

El método de campos potenciales permite construir una ruta dada una configuración inicial-final. Para ello, se considera al elemento a mover como una partícula bajo la influencia de un campo potencial artificial, cuyas variaciones locales modelan la estructura del espacio libre [GREENGARD1988] [ARKI1998].

El comportamiento del elemento móvil en una posición *p* del espacio, determinado por el campo potencial artificial *U* consiste en la suma de potenciales de atracción y repulsión, es decir:

$$U(p) = U_a(p) + U_r(p)$$

El *potencial de atracción* $U_a(p)$ es el objetivo o posición destino con fuerza de atracción, de forma que a medida que el elemento se acerca, éste disminuya su influencia. Por otra parte, el *potencial repulsivo* $U_r(p)$ permite al elemento alejarse de los obstáculos cuando éste se encuentre próximo a ellos.

Se tiene una fuerza artificial $F(p)$ inducida por el potencial $U(p)$ que afecta al elemento en la posición actual:

$$F(p) = -\nabla U(p)$$

cuyo resultado es la suma de una fuerza de atracción proveniente de la posición destino, y de una fuerza de repulsión producida por los obstáculos del entorno.

$$F(p) = F_a(p) + F_r(p)$$

Ahora bien, la implementación de este método en el videojuego Bellum se debe a que las áreas determinadas por el grafo que utiliza el algoritmo de Dijkstra, no tienen una correspondencia directa con cada ubicación posible, por lo que se utilizan campos potenciales para llegar desde el área más cercana a cierta ubicación. El algoritmo se implementó de la siguiente manera:

1. Se ubica la posición inicial y destino del robot a mover.

2. Calcular la fuerza de atracción $F_a(p)$ ejercida por la posición destino p_{dest} sobre el robot con base a la siguiente expresión:

$$F_a(p) = \varepsilon\left(p - p_{dest}\right)$$

donde ε es un factor de escala de la fuerza de atracción.

3. Desde la posición inicial se trazan un número finito de mediciones sobre el espacio para la localización de los obstáculos.

4. Se obtiene la fuerza de repulsión ejercida por cada obstáculo localizado en el paso anterior. Dada la posición actual p, la fuerza de repulsión ejercida desde la posición del obstáculo p_{obs} resulta:

$$F_r(p) = \eta\left(\frac{1}{|p - p_{obs}|} - \frac{1}{d_0}\right)\left(\frac{1}{|p - p_{obs}|^2}\right)\left(\frac{(p - p_{obs})}{|p - p_{obs}|}\right)$$

donde η es un factor de escala de la fuerza de repulsión y d_0 es la distancia de influencia establecida para el campo potencial de cada obstáculo

5. Calcular la sumatoria de fuerzas repulsivas.

6. Se obtiene la fuerza resultante inducida por el potencial que afecta al robot al sumar el total de fuerzas repulsivas con la fuerza de atracción.

7. Obtener el vector unitario de movimiento según la fuerza resultante obtenida en el paso anterior.

$$f(p) = \frac{F(p)}{|F(p)|}$$

8. Calcular la nueva posición para el robot al sumar la posición inicial con el vector unitario de movimiento calculado. Se repite el algoritmo hasta que el robot llega a la posición destino.

Según [LATOMBE1991] existe como desventaja de este método, el hecho de caer en un mínimo local lo cual provoca el no alcanzar la posición destino. Para evitar lo anterior, se maneja un número máximo de iteraciones del algoritmo en el cual se alcanzará la posición destino.

3.3.4 Interfaz gráfica de usuario

La interfaz de usuario proporciona la capacidad de moverse y manipular el juego. Para el desarrollo de esta interfaz se consideró la visualización de información y la entrada de datos del usuario con base al género de estrategia en tiempo real al que pertenece Bellum. En cuanto a las herramientas utilizadas para su creación están: Adobe Photoshop, Autodesk 3ds Max y las bibliotecas de OGRE 3D. Para la visualización de información se manejaron cuatro pantallas de acuerdo al estado del juego:

- En *IntroState* se utilizó una pantalla de inicio de Bellum (figura 3.12) mediante una imagen descriptiva con las opciones para comenzar el juego, entrar al tutorial de gestos o salir de la aplicación.

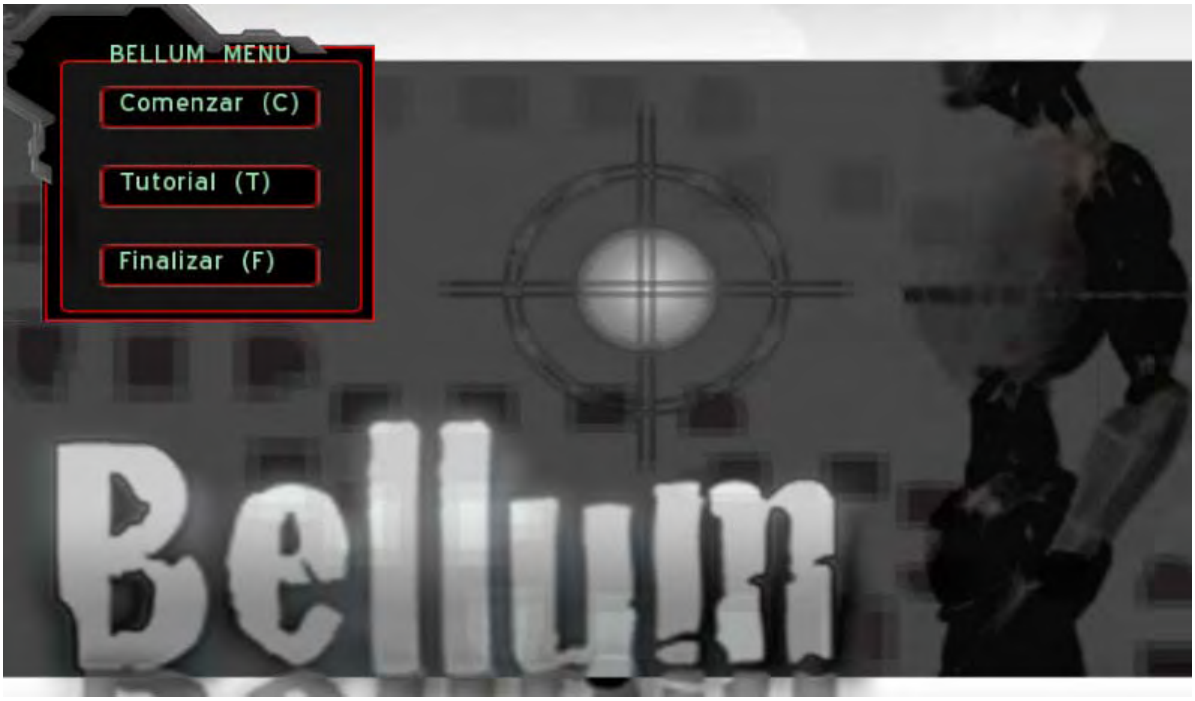


Figura 3.12: Pantalla de inicio del videojuego Bellum.

- En *PlayState* se utilizó una pantalla principal (figura 3.13) donde se desarrolla el juego, en la cual se visualizan los siguientes elementos:
  - El título de la pantalla con el nombre del juego.
  - El terreno y personajes 3D.
  - Mapa de ubicación de los robots con respecto a la totalidad del terreno.
  - Botón para la opción detener el juego.
  - Imágenes decorativas para mostrar datos del robot seleccionado.
  - Indicadores para informar al jugador de su estado: puntuación de robots vivos que pertenecen al usuario, nivel de energía de cada robot y la cantidad monetaria disponible para crear robots.

Figura 3.13: Pantalla principal del videojuego Bellum.

- En *PauseState* se utilizó una pantalla de opción donde se le pregunta al usuario si desea continuar o terminar la aplicación (figura 3.14). Si la opción

elegida es continuar, se retorna al juego que fue detenido. Si la acción es terminar la aplicación, se pasa a la pantalla del estado EndState.

Figura 3.14: Pantalla de la opción Pausa del videojuego Bellum.

- En *EndState* se utilizó una pantalla final de Bellum (figura 3.15) mediante una imagen descriptiva con los créditos y se mantiene abierta por cierto período de tiempo.

Figura 3.15: Pantalla final del videojuego Bellum.

Con respecto a las acciones o datos de entrada del usuario, se utilizó el programa VInput desarrollado como dispositivo de interacción y el sistema de administración de entradas que OGRE 3D implementa a través de la biblioteca OIS. En la tabla 3.4 están las acciones de entrada para Bellum y los eventos necesarios para su generación.

Acción	Evento
Posicionar el puntero en cierta ubicación de la pantalla.	Mover puntero del mouse
Seleccionar un robot	Gesto clic izquierdo
Desplazar un robot a una área del terreno	Seleccionar robot y marcar la ubicación destino con el gesto clic izquierdo sobre el área.
Atacar robot enemigo	Desplazar un robot a la ubicación del enemigo
Crear robots del tipo “Mech Worker”	Gesto clic izquierdo sobre el botón “Mech Worker”
Crear robots del tipo “Mech Elite”	Gesto clic izquierdo sobre el botón “Mech Elite”
Detener juego	Gesto clic izquierdo sobre el botón “Pausa”
Trasladar la cámara hacia el último robot seleccionado.	Gesto clic medio
Quitar la selección a un robot	Gesto clic derecho
Alejar la cámara del terreno.	Gesto *Zoom Out*
Acercar la cámara al terreno.	Gesto *Zoom In*
Mover la cámara sobre el terreno horizontalmente	Mover puntero del mouse hacia el extremo derecho u izquierdo de la pantalla.
Mover la cámara sobre el terreno verticalmente.	Mover puntero del mouse hacia el extremo superior o inferior de la pantalla.

Tabla 3.4: Acciones de entrada del usuario del videojuego Bellum.

3.3.5 Tutorial de gestos

El tutorial de gestos tiene como objetivo ayudar al usuario en el aprendizaje de los gestos de las manos implementados. Para ello, se manejan cinco prácticas donde cada una está asociada a un gesto y el usuario efectúa una secuencia de tareas.

Cabe señalar, que este tutorial se utilizó también para probar el rendimiento de los gestos (ver capítulo 4).

Antes de comenzar cada práctica, se le presenta al usuario la descripción del objetivo y la forma de cómo realizar el gesto (figura 3.16). Las prácticas correspondientes a los gestos de clic izquierdo, derecho y medio comprenden el realizar el gesto sobre la secuencia de botones que aparecen en la pantalla. Finalmente, las prácticas de los gestos de *Zoom In* y *Zoom Out* comprenden el realizar el gesto sobre el robot que se muestra en la pantalla.

Figura 3.16: Pantalla explicativa de una práctica del tutorial de gestos.

En el capítulo siguiente se describen las pruebas y resultados obtenidos del software basado en gestos de las manos con la aplicación Bellum y el navegador Mozilla Firefox.

Capítulo 4

Pruebas y resultados

En este capítulo se describen las pruebas realizadas al software desarrollado y se exponen los resultados obtenidos. Se incluyen gráficas y tablas de resultados así como sus correspondientes interpretaciones.

El objetivo principal de probar con usuarios reales el prototipo del sistema desarrollado, es observar el desempeño y funcionalidad del software basado en gestos de las manos con la aplicación Bellum y el navegador Mozilla Firefox. En los siguientes apartados se describen las pruebas realizadas y los resultados obtenidos.

4.1 DESCRIPCIÓN DE LAS PRUEBAS

Tomando en cuenta el objetivo principal de probar el software de reconocimiento de gestos de las manos a través de marcas, se realizaron pruebas en dos fases:

- Pruebas de rendimiento y usabilidad del reconocimiento de gestos.
- Pruebas de usabilidad de los gestos en el videojuego.
- Pruebas de usabilidad de los gestos en el navegador Mozilla Firefox.

Para realizar estas pruebas se utilizó un grupo de 10 personas cuyo perfil fue determinado por su edad, nivel académico, experiencia usando computadoras, frecuencia de uso de videojuegos (figura 4.1); con la finalidad de obtener datos de utilidad para explicar los resultados. A los participantes que manifestaron no hacer uso de videojuegos no se les aplicó la prueba de jugar con Bellum.

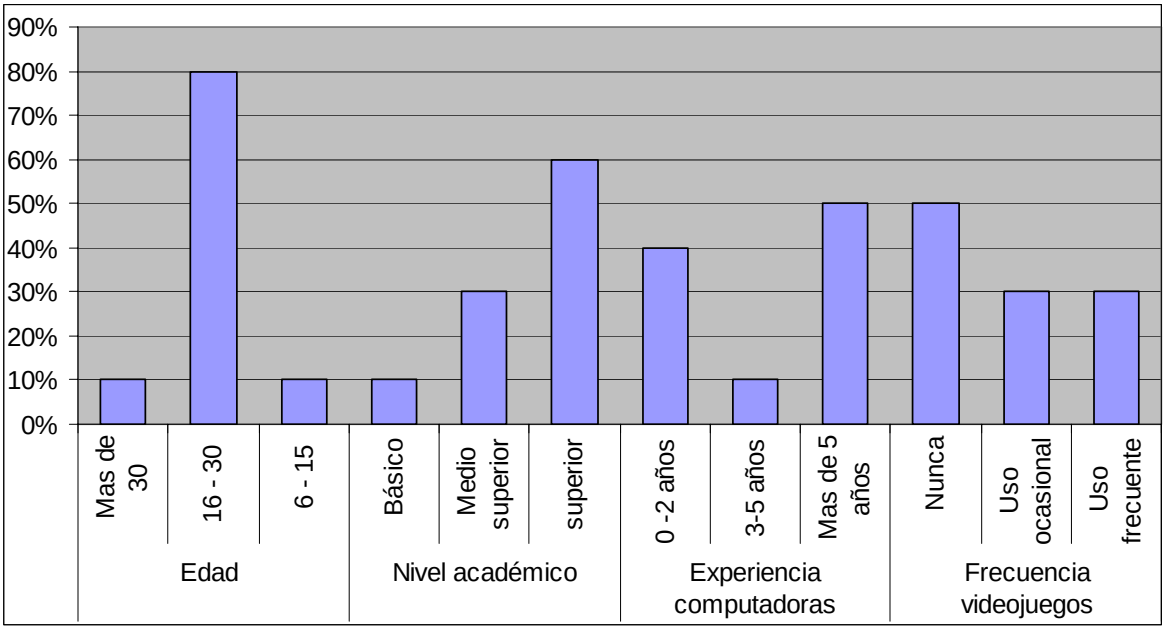


Figura 4.1: Perfil de los participantes de las pruebas.

A este grupo de participantes se les explicó el objetivo y se les consultaron datos para conocer su perfil a través de un cuestionario pre-evaluación. Después se les indicó una lista de tareas a realizar con el software a probar. Al final, contestaron a un cuestionario post-evaluación para registrar los resultados obtenidos. Posteriormente esos datos se procesaron junto con los recabados en archivos de texto por el software de prueba, se obtuvieron las tablas y gráficas de los resultados y con ello la evaluación perseguida.

En cuanto a los materiales utilizados, se manejaron los que se muestran en la tabla 4.1. Cabe señalar que los cuestionarios pueden ser consultados en el CD anexo a este documento.

Tipo	Descripción
Equipo	Una computadora personal con un procesador AMD Athlon 64 X2 4000+ AM2 Dual Core, 1GB en RAM y una tarjeta aceleradora gráfica Nvidia 6100 con 256 en RAM de video. Monitor y teclado.
	Cámara web Microsoft LifeCam VX-1000.
	Lámpara con un foco de 25 Watts.
	Marcas impresas en papel de 1.5 x 1.5 cm.
Software	VInput: software que implementa el reconocimiento de gestos de las manos a través de marcas para generar eventos de entrada.
	Bellum: videojuego del género de estrategia en tiempo real.
	Tutorial de Bellum: software que genera casos de prueba y registra tanto el tiempo y acciones del usuario.
	Mozilla Firefox 2.0.0.12
Escrito	Nota de confidencialidad de los datos y presentación.
	Cuestionario pre-evaluación sobre perfil del participante.
	Escenarios de uso y tareas a realizar
	• Cuestionario post-evaluación sobre el reconocimiento de gestos. • Cuestionario post-evaluación sobre la usabilidad de los gestos en el videojuego. • Cuestionario post-evaluación sobre la usabilidad de los gestos en el navegador Mozilla Firefox.

Tabla 4.1: Materiales utilizados en la pruebas del sistema.

A continuación se describe en qué consistieron las pruebas de cada una de las fases, así como los resultados obtenidos.

4.1.1 Pruebas de rendimiento y usabilidad del reconocimiento de gestos

El objetivo de estas pruebas es medir dos aspectos del software basado en gestos de las manos: (1) establecer las condiciones límite en la interfaz gráfica de usuario para obtener un reconocimiento de gestos adecuado y (2) conocer la facilidad con la que los usuarios desarrollan una interacción con este software. Para cada aspecto se determinaron los elementos a medir y el método mediante el cual se obtendrían estos datos.

4.1.1.1 Rendimiento del reconocimiento de gestos

Para conocer las condiciones límite, se creó un software que registra valores con base en las acciones realizadas por el usuario. Este software, tutorial de gestos (figura 4.2), establece prácticas en donde cada una está relacionada con un gesto implementado y están conformadas por una secuencia de tareas sobre elementos gráficos con variaciones. En la tabla 4.2 se muestran las tareas de cada práctica junto con el elemento gráfico y sus diferentes valores a probar.

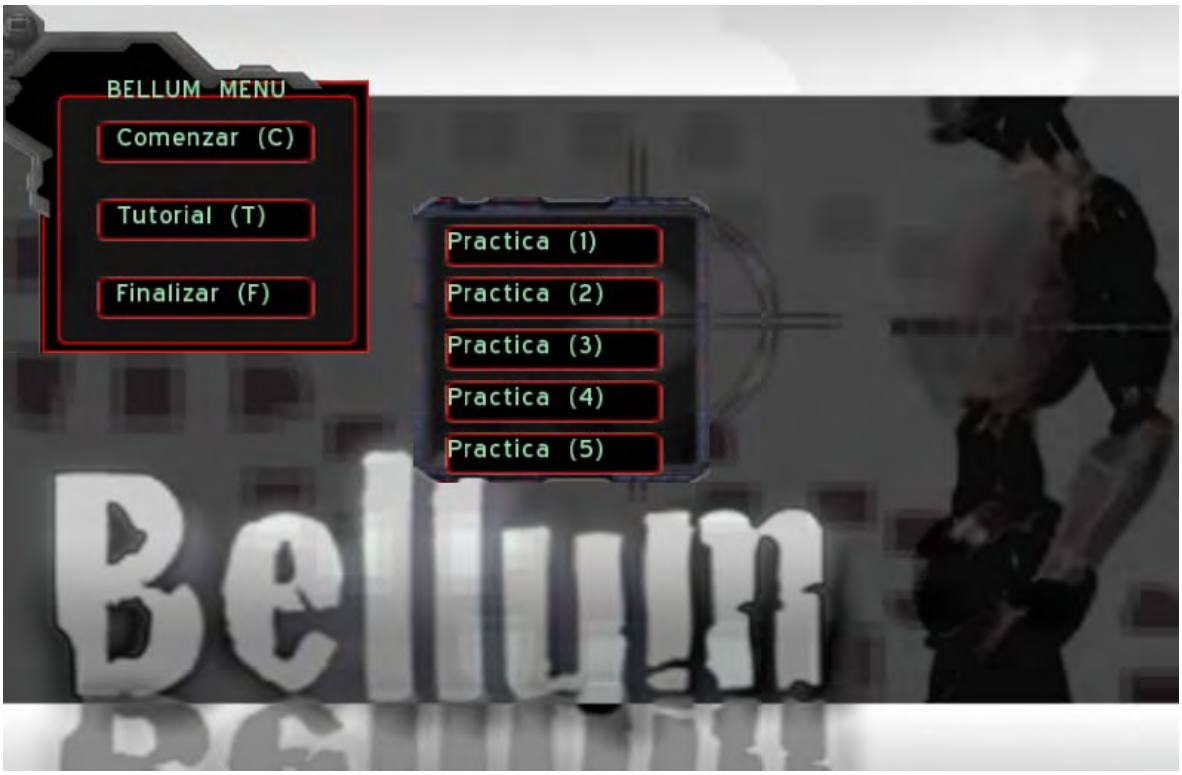


Figura 4.2: Interfaz de usuario del software de prueba “Tutorial de gestos”.

Práctica	Variable	Tarea	Valor
Realizar gesto clic izquierdo	Área del botón	1	150 x 40 píxeles
		2	135 x 36 píxeles
		3	120 x 32 píxeles
		4	105 x 28 píxeles
		5	90 x 24 píxeles
		6	75 x 20 píxeles
		7	60 x 16 píxeles
		8	45 x 12 píxeles
Realizar gesto clic derecho	Área del botón	1	100 x 100 píxeles
		2	90 x 90 píxeles
		3	80 x 80 píxeles
		4	70 x 70 píxeles
		5	60 x 60 píxeles
		6	50 x 50 píxeles
		7	40 x 40 píxeles
		8	30 x 30 píxeles
Realizar gesto clic medio	Área del botón	1	100 x 100 píxeles
		2	90 x 90 píxeles
		3	80 x 80 píxeles
		4	70 x 70 píxeles
		5	60 x 60 píxeles
		6	50 x 50 píxeles
		7	40 x 40 píxeles
		8	30 x 30 píxeles
Realizar gesto Zoom Out	Distancia de la cámara virtual en el eje coordenado Z[1] con respecto al origen del ambiente tridimensional.	1	100 cm
		2	120 cm
		3	140 cm
		4	160 cm
		5	180 cm
		6	200 cm
		7	220 cm
		8	240 cm
Realizar gesto Zoom In	Distancia de la cámara virtual en el eje coordenado Z con respecto al origen del ambiente tridimensional.	1	240 cm
		2	220 cm
		3	200 cm
		4	180 cm
		5	160 cm
		6	140 cm
		7	120 cm
		8	100 cm

Tabla 4.2: Prácticas y tareas del software de prueba “tutorial de gestos”.

[1] La equivalencia de medida utilizada es: 1 unidad de cámara en OGRE 3D es igual a 1 cm.

El tutorial de gestos procesa y almacena información en archivos de texto para su análisis posterior. Los datos almacenados son:

- Tiempo en completar cada práctica (en segundos).
- Número de la última tarea realizada por cada práctica.
- Número de errores cometidos por tarea de cada práctica.

Los resultados obtenidos en cuanto al tiempo utilizado para terminar cada práctica por cada participante y a nivel grupo se muestran en la tabla 4.3 y el número de la última tarea realizada por cada práctica en la tabla 4.4.

Práctica	**Participantes**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Total**	**Prom**
Realizar gesto clic izquierdo	1:29	0:20	1:54	0:20	0:40	1:19	0:14	0:36	1:47	0:42	9:21	0:56
Realizar gesto clic derecho	2:00	0:21	0:59	0:18	0:59	1:32	0:17	1:10	0:54	1:25	9:55	0:59
Realizar gesto clic medio	1:32	0:22	0:25	0:30	0:33	0:37	0:15	0:26	0:43	1:09	6:32	0:39
Realizar gesto Zoom Out	0:04	0:06	0:05	0:20	0:09	0:10	0:05	0:03	0:02	0:01	1:05	0:06
Realizar gesto Zoom In	0:01	0:02	0:01	0:33	0:17	0:13	0:10	0:01	0:04	0:02	1:24	0:08
Total	5:06	1:11	3:24	2:01	2:38	3:51	1:01	2:16	3:30	3:19		
Promedio	1:01	0:14	0:40	0:24	0:31	0:46	0:12	0:27	0:42	0:39		

Tabla 4.3: Tiempo de los participantes en realizar cada practica.

Práctica	**Participantes**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Realizar gesto clic izquierdo	8	8	8	8	8	8	8	8	8	8
Realizar gesto clic derecho	4	8	8	8	8	8	8	8	8	8
Realizar gesto clic medio	8	8	8	8	8	8	8	8	8	8
Realizar gesto Zoom Out	8	8	8	8	8	8	8	8	8	8
Realizar gesto Zoom In	8	8	8	8	8	8	8	8	8	8

Tabla 4.4: Número de la última tarea realizada por cada práctica.

Tomando en cuenta que una práctica se considera exitosa cuando el tiempo para su realización no sobrepasa 2 minutos y de acuerdo al número de la última tarea realizada en aquellas prácticas no exitosas de la tabla 4.4, se puede

observar que sólo un participante no completó una práctica por lo cual el porcentaje de éxito de la prueba fue alto.

Los resultados obtenidos en cuanto al número de errores cometidos por todos los participantes por tarea de cada práctica se muestran en la tabla 4.5 y en la figura 4.3 la tendencia de estos datos. Se considera error cuando la tarea a realizar no alcanzó el objetivo ya sea por no generar el gesto adecuadamente o no acertar en el área destinada.

Práctica	**Tareas**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Realizar gesto clic izquierdo	5	8	1	3	0	9	7	17
Realizar gesto clic derecho	2	1	5	1	2	1	8	12
Realizar gesto clic medio	1	5	6	5	2	7	4	11
Realizar gesto Zoom Out	3	4	1	8	9	2	7	7
Realizar gesto Zoom In	2	5	5	7	2	0	3	11

Tabla 4.5: Número de errores por tarea de las prácticas.

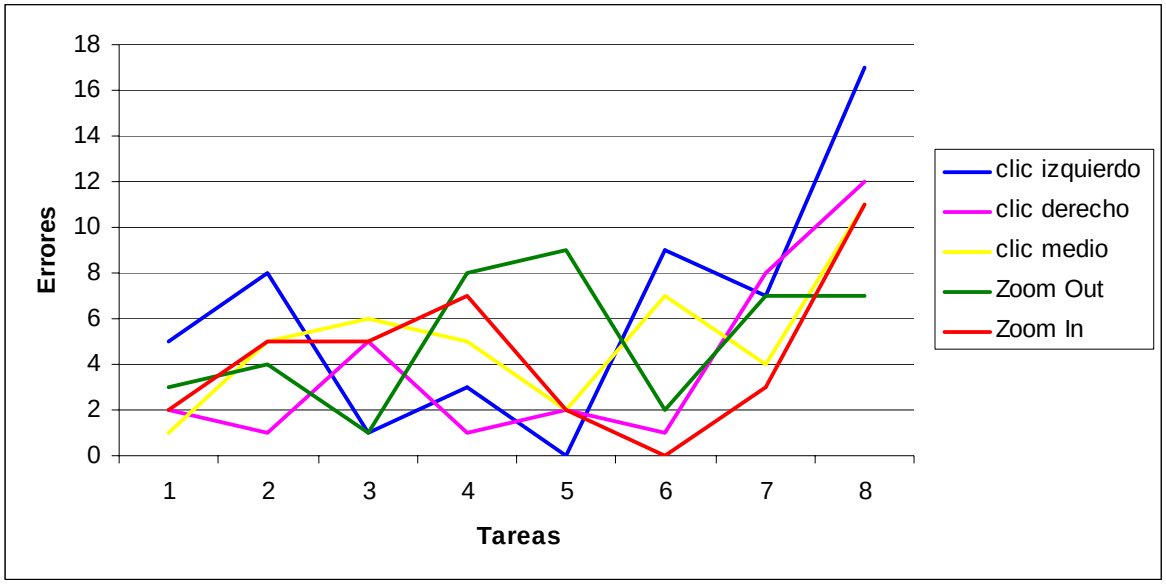


Figura 4.3: Comportamiento de los errores por tarea de las prácticas.

De estos datos se observa que se pueden reconocer los gestos en elementos gráficos de tamaño pequeño. Sin embargo, la cantidad de tiempo y

errores aumenta, además que los usuarios se llegan a incomodar por no realizar el gesto en el primer intento. Es por ello, que se sugieren las siguientes dimensiones para el área de los elementos gráficos en futuros trabajos:

- Gesto clic izquierdo: 105 x 28 píxeles, que fue el tamaño desplegado por la tarea 5 de la practica 1.
- Gesto clic derecho: 70 x 70 píxeles, que fue el tamaño desplegado por la tarea 4 de la practica 2.
- Gesto clic medio: 100 x 100 píxeles, que fue el tamaño desplegado por la tarea 1 de la practica 3.

Con respecto a los gestos de *Zoom Out* y *Zoom In*, no son necesarias ciertas dimensiones sino la velocidad con la que se hace el gesto para el reconocimiento adecuado de las marcas.

4.1.1.2 Usabilidad del reconocimiento de gestos

Para el aspecto de conocer la facilidad con la que los usuarios interactúan con el software basado en gestos de las manos, se utilizó un cuestionario post-evaluación que puede ser consultado en el CD anexo a este documento. En este cuestionario el participante da su punto de vista y conocimiento adquirido a través de un conjunto de preguntas cerradas.

Los resultados obtenidos en la tabla 4.6 muestran que la mayoría de los participantes valoraron en el cuestionario como fácil el realizar los gestos, el 70% opinó que la forma de generar los gestos es entendible y la interacción fácil, y un 50% manifestó que fue buena la navegación en el tutorial. Además, todos los participantes respondieron que les pareció cómodo el realizar las prácticas.

Otro dato cualitativo interesante es el hecho de que el 30% de los participantes sugirieron mejorar la forma de generar los gestos en cuanto a movimientos y posición.

Finalmente, en cuanto al aprendizaje de los gestos, la mayoría contestó correctamente la secuencia de movimientos; sin embargo, hubo un porcentaje de error ya que se observó confusión en los gestos cuyas secuencias son parecidas.

Pregunta	**Respuesta**	**Porcentaje**
1. ¿Cómo te has sentido al realizar las prácticas del tutorial?	Cómodo	100%
	Incómodo	0%
2. Al desplazar el puntero con el dedo sentiste que la navegación fue	Buena	50%
	Regular	50%
	Mala	0%
3. La forma de generar los gestos es	Entendible	70%
	Confusa	30%
	Mala	0%
4. ¿Cómo valoras la dificultad para realizar el gesto clic izquierdo?	Fácil	100%
	Regular	0%
	Difícil	0%
5. Elije la secuencia de movimientos correcta para generar el clic izquierdo	Correcto	80%
	Error	20%
6. ¿Cómo valoras la dificultad para realizar el gesto clic derecho?	Fácil	60%
	Regular	30%
	Difícil	10%
7. Elije la secuencia de movimientos correcta para generar el clic derecho	Correcto	70%
	Error	30%
8. ¿Cómo valoras la dificultad para realizar el gesto clic medio?	Fácil	60%
	Regular	40%
	Difícil	0%
9. Elije la secuencia de movimientos correcta para generar el clic medio	Correcto	100%
	Error	0%
10. ¿Cómo valoras la dificultad para realizar el gesto Zoom Out (alejarse)?	Fácil	50%
	Regular	30%
	Difícil	20%
11. Elije la secuencia de movimientos correcta para generar el Zoom Out	Correcto	70%
	Error	30%
12. ¿Cómo valoras la dificultad para realizar el gesto Zoom In (acercarse)?	Fácil	40%
	Regular	40%
	Difícil	20%
13. Elije la secuencia de movimientos correcta para generar el Zoom In	Correcto	90%
	Error	10%
14. ¿Qué cambiarías de la forma de generar los gestos?	Movimientos y posición	30%
	Parte del cuerpo utilizada	10%
	Nada	60%
15. En general consideras que el utilizar este tipo interacción es	Fácil	70%
	Regular	30%
	Difícil	0%

Tabla 4.6: Resultados del cuestionario de usabilidad del tutorial de gestos.

4.1.2 Prueba de usabilidad de los gestos en el videojuego

Para conocer la facilidad con la que los usuarios interactúan en el videojuego Bellum con el software basado en gestos de las manos, se utilizó un cuestionario post-evaluación que puede ser consultado en el CD anexo a este documento. En este cuestionario el participante da su opinión general, a través de un conjunto de preguntas cerradas, sobre el funcionamiento del software de reconocimiento de gestos de las manos en la interacción con un juego de estrategia en tiempo real.

Como ya se había mencionado, sólo los participantes que en el cuestionario pre-evaluación mostraron experiencia en juegos para computadora hicieron esta prueba. A este grupo de participantes se les explicó el objetivo y se les indicó que antes de jugar leyeran las instrucciones, en donde se explica la lógica y acciones para su uso. Al término de la lectura, se les pidió que jugaran. Al final, ya sea con la victoria o pérdida del juego, contestaron al cuestionario para registrar los resultados obtenidos.

Los resultados obtenidos (figura 4.4) muestran que todos los participantes les pareció cómodo el jugar con los gestos, la manipulación les fue entendible y consideraron factible esta interacción alternativa en videojuegos de este género.

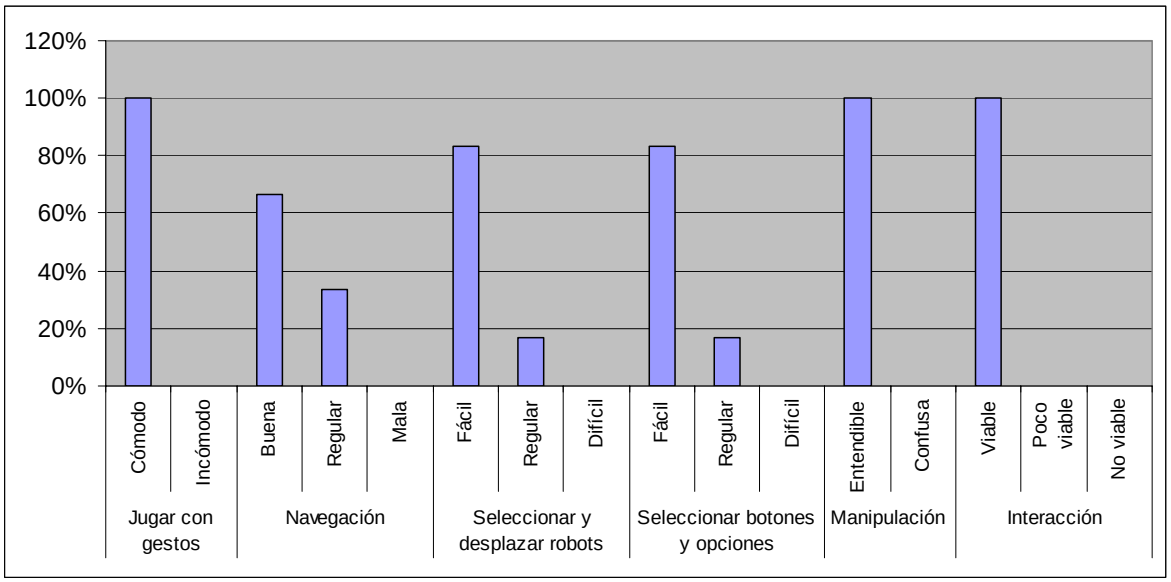


Figura 4.4: Resultados de la prueba de usabilidad de los gestos en Bellum.

4.1.3 Prueba de usabilidad de los gestos con Mozilla Firefox

Aunque en un principio se determinó que Bellum sería la aplicación que explotara las capacidades del reconocimiento de gestos de las manos implementado, con base en la experiencia de la construcción de VInput y de experimentar sus posibilidades, se efectuaron pruebas también con el navegador Mozilla Firefox.

Se consideró conveniente observar el comportamiento con un software que en la actualidad esté siendo usado y que cumpla con las características de ser gratuito y flexible en lo referente a realizar modificaciones en su interfaz gráfica para realizar las pruebas.

Para hacer uso del software reconocedor de gestos de las manos mediante marcas en el navegador Mozilla Firefox, fue necesario modificar la interfaz gráfica de usuario en cuanto al tamaño de algunos de sus elementos. Cabe mencionar, que sin esta modificación funcionaba VInput, sin embargo los iconos y botones para manipular al navegador resultaban ser pequeños para el reconocimiento del gesto al efectuar el movimiento requerido.

Razón por la cual, se modificó el tema de Mozilla Firefox, el cual es un archivo que contiene la definición de los elementos necesarios para editar el aspecto de la interfaz gráfica del programa. El procedimiento y la descripción de los elementos modificados pueden ser consultados en el apéndice B.

Para realizar las pruebas sobre el funcionamiento del software de reconocimiento de gestos de las manos en la interacción con este navegador, se pidió a los participantes que navegaran durante cinco minutos haciendo uso de los gestos y después se les pidió contestar a un cuestionario post-evaluación.

Los resultados obtenidos (figura 4.5) sirvieron para determinar que los participantes pudieron navegar al tener a su disposición las acciones básicas para hacerlo a través de los gestos. El 60% de los participantes les resultó un poco complicado el mover el puntero ya que manifestaron que aún no tenían la práctica

suficiente para hacerlo. Sin embargo, el 90% piensa que es viable interactuar mediante los gestos con el navegador Mozilla Firefox.

Con base a las reacciones y comentarios que los participantes externaron en el momento de hacer la prueba, se hizo evidente la carencia de algunos eventos que comúnmente usan los usuarios de este navegador como el seleccionar texto y arrastrar ventanas emergentes.

Al final de la prueba, los participantes mostraron haberse acostumbrado a manipular Mozilla Firefox con el software basado en gestos de las manos mediante marcas.

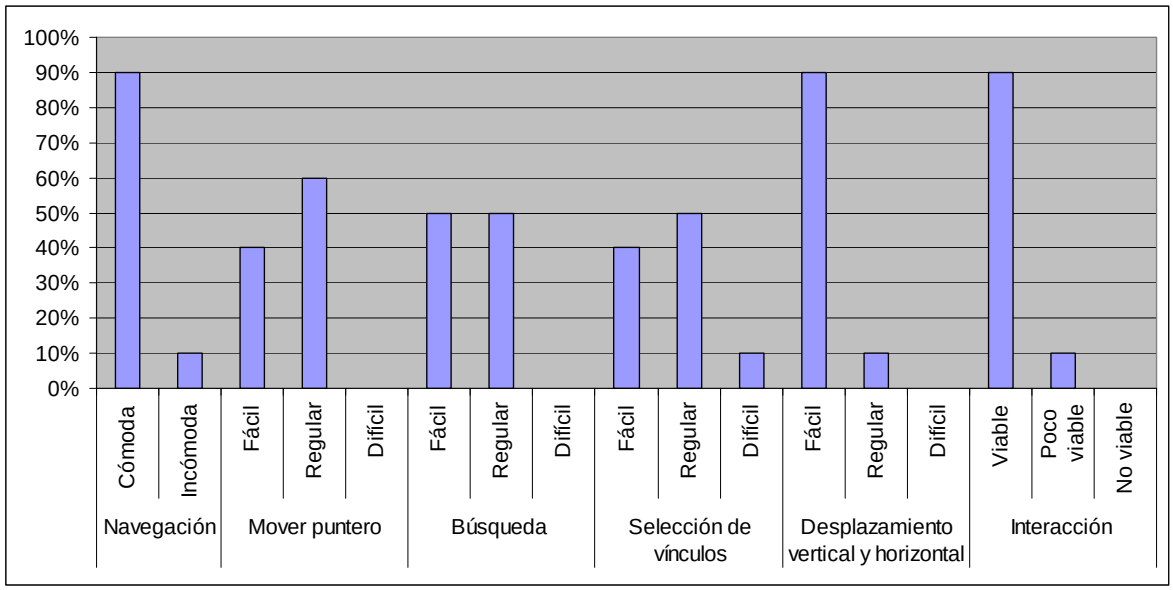


Figura 4.5: Resultados de las pruebas de usabilidad de los gestos en Mozilla Firefox.

En el siguiente apartado se presentan las conclusiones de la tesis presentada, las principales aportaciones realizadas y algunas de las futuras líneas de investigación que se derivan del presente trabajo.

Conclusiones

En este apartado se presentan las conclusiones generales con respecto a los objetivos planteados. Adicionalmente, se exponen las contribuciones y los trabajos a futuro identificados.

En esta tesis se ha tratado satisfacer la necesidad de ofrecer una alternativa de interacción basada en el reconocimiento de gestos.

Con base en el software desarrollado, se concluye que el objetivo de construir una aplicación que reconoce gestos, los cuales son generados por las manos del usuario a través del uso de marcas, para la manipulación de un videojuego ha sido cumplido. Lo anterior se logró mediante la aplicación de técnicas de las áreas de Visión por Computadora, Inteligencia Artificial y Graficación, así como el uso de las tecnologías como ARToolKitPlus, OpenCV y OGRE 3D.

El software basado en gestos de las manos y el videojuego presentados conforman un prototipo que puede considerarse como una herramienta para analizar la viabilidad del concepto de interacción a partir de gestos con marcas en las manos en aplicaciones con gráficos 3D.

El uso de marcas proporcionó una alternativa fácil de implementar el reconocimiento de los gestos, en comparación con técnicas como filtros de partículas, Kalman, entre otras. Además de que se hace uso de un dispositivo común en las computadoras personales, la cámara web, por lo que no se requiere de la compra de nuevo hardware.

Con respecto al videojuego, se obtuvo un prototipo adecuado que permite presentar la interacción y las posibles capacidades de tener un dispositivo basado en el reconocimiento de gestos de la mano. Por lo que se considera que la industria de videojuegos resulta adecuada y fértil para la aplicación de este tipo de propuestas puesto que requiere de nuevos dispositivos de entrada que sean innovadores, de uso fácil y que exploten los avances tecnológicos.

Los resultados obtenidos de las pruebas realizadas para evaluar el rendimiento del software con variaciones en la interfaz gráfica de usuario,

muestran que su operación es correcta aún cuando los elementos gráficos tengan un tamaño pequeño, puesto que se obtiene un reconocimiento de gestos y una generación de eventos adecuados. Sin embargo, para minimizar el esfuerzo del usuario al tratar de introducir un comando se han recomendado ciertas dimensiones a considerar en trabajos futuros.

En cuanto a los resultados obtenidos de los cuestionarios de usabilidad, permitieron ver que los usuarios mostraron interés y consideraron viable este tipo interacción. Un alto porcentaje de los participantes manifestaron que la manipulación y navegación les resultó fácil y entendible. Sin embargo, un punto necesario para darle mayor continuidad a este trabajo, es el hecho de implementar las mejoras observadas en estas pruebas para perfeccionar los movimientos e implementar más eventos de entrada.

En conclusión, tomando en cuenta lo anterior y todo el trabajo de la tesis, se considera que se ha desarrollado una alternativa de interacción humano-máquina gracias a la cohesión y aprovechamiento de diferentes áreas de computación, en beneficio del desarrollo de dispositivos de entrada que cumplan con las exigencias del presente y futuro.

CONTRIBUCIONES

Con base en los resultados obtenidos, se considera que las principales contribuciones del trabajo de tesis son las siguientes:

1. Se desarrolló una alternativa de interacción con interfaces gráficas. La idea es que el reconocimiento de gestos pueda ayudar o reemplazar en un futuro a los dispositivos tradicionales como teclado y mouse para realizar tareas con las computadoras.
2. Se ofrece un software que explote el uso de cámaras web en las computadoras personales para generar un nuevo dispositivo de entrada de bajo costo.

3. Demostración de integración de conocimiento y tecnologías de las áreas de Visión por Computadora, Inteligencia Artificial y Graficación.

4. Viabilidad de utilización en aplicaciones donde es necesario contar con interacción con el usuario de forma natural, novedosa, atractiva y de uso fácil; además de que vaya de acuerdo con la evolución en avances tecnológicos.

TRABAJOS A FUTURO

En resumen, de acuerdo con los resultados obtenidos y el trabajo realizado, se concluye que se cumplieron los objetivos; sin embargo existen cuestiones que pueden ser materia de trabajos futuros. Dentro de estos trabajos a futuro identificados están:

1. Mejorar las combinaciones de los gestos con la finalidad de que sean más adecuados para el usuario.

2. Aumentar la detección de direcciones de movimiento a ocho para tener un rango de combinaciones más amplio en la generación de gestos.

3. Incorporar nuevos gestos y mensajes al sistema operativo para extender la funcionalidad y mejorar el desempeño de la aplicación VInput.

4. Desarrollar aplicaciones con interfaz gráfica de usuario adecuada para el uso de VInput.

5. Es conveniente implementar un *tracker* propio para evitar dependencia con las bibliotecas utilizadas.

6. Hacer portable a VInput y Bellum en otras plataformas.

7. En cuanto al videojuego desarrollado, se puede enriquecer la aplicación con elementos como: incluir mejores técnicas de Inteligencia Artificial, crear

nuevos escenarios, incluir una historia o narrativa, agregar modo multijugador, mejorar los gráficos, crear interacción con el medio, añadir la funcionalidad de trasladarse mediante el radar, entre otros.

8. Vincular cada gesto con un sonido para dotar de una retroalimentación auditiva al usuario.

BIBLIOGRAFÍA

[ANKH2006] Sitio web del juego Ankh. Disponible en: http://www.ankh-game.com/

[ARKI1998] Arki, Ronald. *Behavior-Based Robotics*. MIT Press, Cambridge, MA Junio, 1998.

[ARTOOLKIT] *ARToolKit* Coordinate Systems. Disponible en: http://www.hitl.washington.edu/artoolkit/documentation/cs.htm.

[ARTOOLKITPLUS] Biblioteca de *ARToolKitPlus*. Disponible en: http://studierstube.icg.tu-graz.ac.at/handheld_ar/artoolkitplus.php

[BUCHMANN2004] Buchmann, V; Violich, S; Billinghurst, M; Cockburn, A. *FingARtips - Gesture Based Direct Manipulation in Augmented Reality*. Singapore: Proceedings of Graphite '04: International Conference on Computer Graphics and Interactive Techniques, Págs. 212-221. Junio 2004.

[BUCKLAND2004] Buckland, Mat. *Programming Game AI by Example*. Plano, Texas: Wordware Publishing, Inc. 2004. Págs. 44-48.

[BYUNG-WOO1999] Byung-Woo, Min; Ho-Sub, Yoon; Soh, Jung; Ohashi, Takeshi; Ejima, Toshiaki. *Visual Recognition of Static Dynamic Gesture: Gesture-Driven Editing System*. Visual Languages and Computing, vol. 10, Págs. 291-309, 1999.

[CASTLEMAN1996] Castleman, K. R. *Digital Image Processing*. Prentice-Hall, Englewood Cli's, New Jersey, 1996.

[CORMEN2001] Cormen, Thomas H; Leiserson, Charles E; Rivest, Ronald L. Stein, Clifford. *Introduction to Algorithms*. 2ª Edición. MIT Press, Cambridge, MA. 2001.

[CROWLEY1995] Crowley, J. L; Berard, F; Coutaz, J. *Finger Tracking as an Input Device for Augmented Reality*. In Intl. Workshop on Automatic Face and Gesture Recognition, 1995.

[DEMARÍA2002] DeMaría, Rusel; Wilson, Johnny L. *High Score! La historia ilustrada de los videojuegos*. (Traducción) McGraw Hill/Interamericana de España S.A. Madrid, 2002.

[FUKUMOTO1994] Fukumoto, M; Suenaga, Y; Mase, K. *Finger-Pointer: Pointing Interface by Image Processing.* Computers & Graphics, 18(5):633-642, 1994.

[GONZALEZ2002] Gonzalez, Rafael C; Woods, Richard. E. *Digital Image Processing*. Prentice Hall, Upper Saddle River, NJ. 2002. Pág. 793.

[GREENGARD1988] Greengard, Leslie. *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, Cambridge, MA, 1988.

[HANDHELDAR] Sitio web de *Handheld Augmented Reality*. Disponible en: http://studierstube.icg.tu-graz.ac.at/handheld_ar/

[HARALICK1992] Haralick, R. M; Shapiro, L. G. *Computer and robot vision*. Addison-Wesley Publishing Co., New York, 1992.

[HUMMELS1998] Hummels, C; Stappers, P. J. *Meaningful Gestures for Human Computer Interaction: Beyond Hand Postures*. In Proc. IEEE Intl. Conference on Automatic Face and Gesture, abril 1998.

[JUNKER2006] Junker, Gregory. *Pro OGRE 3D Programming*. APRESS. USA, 2006.

[KENDON1972] Kendon, A. *Some relationships between body motion and speech*. *An analysis of an example*. Ed. A Siegman and B. Pope, Studies in Dyadic Communication, New York, Pergamon Press, 1972.

[KÖLSCH2004] Kölsch, Mathias. *Vision Based Hand Gesture Interfaces for Wearable Computing and Virtual Environments*. 2004.

[LATOMBE1991] Latombe, Jean-Claude. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[LEWIS1993] Lewis, Clayton; Rieman, John. *Task-centered user interface design: A pratical introduction*. Shareware. New York, 1993.

[MALLIET2005] Malliet, S; de Meyer, G. *Handbook of Computer Game Studies*. Joost Raessens and Jeffrey Goldstein, Eds. The MIT Press, Cambridge, MA, 2005. Págs. 23-45.

[MCNEIL1992] McNeill, David. *Hand and Mind: What Gestures Reveal about Thought*. Chicago: University of Chicago Press, 1992.

[MYERS1996] Myers, Brad. *A brief history of human-computer interaction technology*. Reporte técnico. Rep. CMU-CS-96-163. Carnegie Mellon University, 1996.

[MYSLIWIEC1994] Mysliwiec, T. A. *FingerMouse: A Freehand Computer Pointing Interface*. Reporte técnico VISLab-94-001, Vision Interfaces and Systems Lab, The University of Illinois. Chicago, 1994.

[OGRE] Sitio web de *OGRE 3D*. Disponible en: http://www.ogre3d.org./

[OPENCV] Sitio web de *Open Source Computer Vision Library*. Disponible en: http://www.intel.com/technology/computing/opencv/index.htm

[OPENCVGUIDE] Sitio web de *OpenCV Coding Style Guide*. Disponible en: http://www.intel.com/technology/computing/opencv/coding_style/

[PAJARES2001] Pajares, Gonzalo; De la Cruz, Jesús. M. *Visión por Computador. Imágenes Digitales y Aplicaciones*. 1ª edición, RA-MA, Madrid, 2001.

[QUEK1995] Quek, F. K. H; Mysliwiec, T; Zhao, M. *FingerMouse: A Freehand Pointing Interface*. In Proc. Int'l Workshop on Automatic Face and Gesture Recognition, Págs. 372-377, junio 1995.

[ROLLINGS2003] Rollings, Andrew; Adams, Ernest. *On Game Design*. New Riders Publishing. Junio 2003.

[SCHOLAND2002] Scholand, Michael. (Traducción por Lidia Cámara). *Localización de videojuegos.* 2002.

[STANG2006] Stang, Bendik. The Book of Games Volume 1: The Ultimate Guide to PC and Video Games. 1ª edición. Gamexplore N.A. Inc. Noviembre, 2006.

[TRIESCH2001] Triesch, J; von der Malsburg, Christoph. *A System for Person-Independent Hand Posture Recognition against Complex Backgrounds.* IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 23, No. 12. Págs. 1449-1453. Diciembre, 2001.

[TURK2001] Turk, M. *Gesture recognition*. In K. Stanney, editor, Handbook of Virtual Environments: Design, Implementation and Applications. Lawrence Erlbaum Associates Inc. Diciembre 2001.

[VIDEOINPUT] Sitio web para la biblioteca *VideoInput*. Disponible en: http://muonics.net/school/spring05/videoInput/

[WAGNER2007] Wagner, Daniel; Schmalstieg, Dieter. *ARToolKitPlus for Pose Tracking on Mobile Devices*. Institute for Computer Graphics and Vision, Graz University of Technology. 2007.

Apéndice A

Archivo de configuración de VInput

Esta sección describe el formato del archivo de configuración que almacena los gestos que implementa la aplicación VInput.

La estructura del archivo de configuración XML define la gramática de los gestos utilizados por la aplicación VInput.

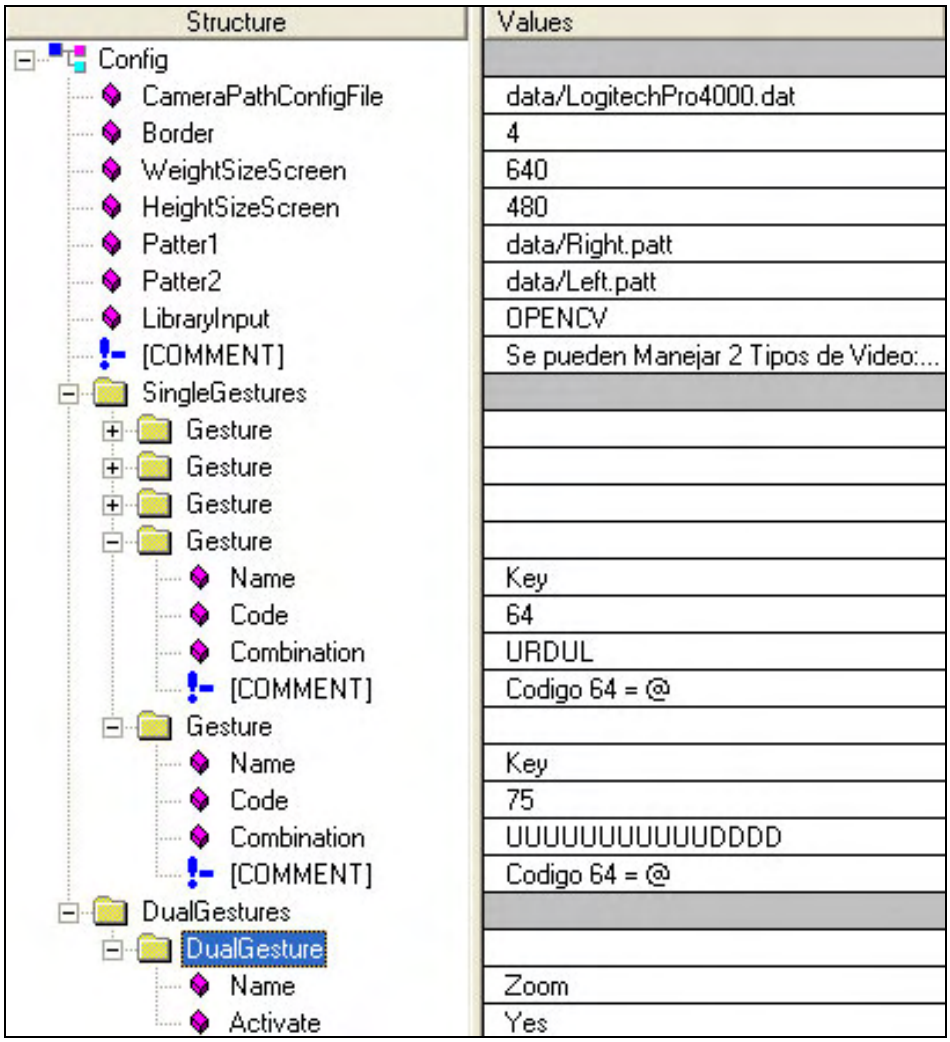


Figura A.1: Estructura del archivo de configuración XML de VInput.

En la figura A.1 se tiene la estructura jerárquica conformada por los siguientes elementos:

- ***Config***: Es el elemento raíz del archivo que se compone por los elementos *SingleGestures* y *DualGestures*. Define los aspectos generales de la aplicación VInput. Todos los atributos definidos son requeridos.
- ***SingleGestures***: Elemento que puede ser compuesto por uno o varios elementos *Gesture*. Define una lista de gestos que utilicen una marca.

- ***Gesture***: Elemento que representa a un gesto que utiliza una marca y que será reconocido por VInput. Sólo el atributo code puede ser opcional.
- ***DualGestures***: Elemento que puede ser compuesto por un sólo elemento *DualGesture*.
- ***DualGesture***: Elemento que representa a un gesto que utiliza dos marcas y que será reconocido por VInput. Todos los atributos definidos son requeridos.

Cada elemento debe tener sus atributos definidos con valores válidos. En la tabla A.1 se describen las definiciones básicas de formato para los atributos de cada elemento.

Elemento	**Atributo**	**Descripción**
Config	CameraPathConfigFile	Representa la dirección relativa o absoluta del archivo que contiene las características de la cámara a utilizar por el tracker de ARToolKitPlus.
	Border	Determina el tamaño del borde que delimita el área de trabajo. Es de tipo numérico entero mayor que 0 y se recomienda que los valores estén dentro del rango de 3 a 9.
	WeightSizeScreen	Determina el ancho de la resolución de la cámara para su inicialización.
	HeightSizeScreen	Determina el alto de la resolución de la cámara para su inicialización.
	Patter1	Representa la dirección relativa o absoluta del archivo que utiliza ARToolKitPlus para identificar la marca 1.
	Patter2	Representa la dirección relativa o absoluta del archivo que utiliza ARToolKitPlus para identificar la marca 2.
	LibraryInput	Nombre de la biblioteca para la adquisición de video, con la cual se va inicializar al programa VInput. Los posibles valores son: OPENCV o VIDEOINPUT

Elemento	Atributo	Descripción
Gesture	Name	Es el nombre del gesto. Los posibles valores son: Left Click, Middle Click, Right Click y Key.
	Combination	Cadena formada por la combinación de las direcciones: L, R, U o D. Representa al gesto.
	Code	Para el gesto “Key” representa el código del carácter ASCII a representar.
DualGesture	Name	Es el nombre del gesto. El posible valor es Zoom.
	Activate	Habilita o deshabilita el gesto Zoom In y Zoom Out. Los posibles valores son: Yes o No.

Tabla A.1: Atributos de los elementos del archivo de configuración XML de VInput.

Apéndice B

Adaptación de la interfaz gráfica de Mozilla Firefox

Esta sección describe la adaptación realizada a la interfaz gráfica del navegador Mozilla Firefox para hacer uso del software reconocedor de gestos de las manos mediante marcas.

Para la modificación del aspecto de la interfaz gráfica del programa se editó el tema por defecto llamado *Winstripe* ubicado en el archivo "classic.jar" dentro del directorio de instalación de Mozilla Firefox. Al extraer el contenido de este archivo manteniendo la estructura de directorios, se tiene una carpeta llamada *skin*, así como dos archivos *preview.png* e *icon.png*. Dentro de *skin* está el directorio *classic* a partir del cual se hicieron las modificaciones. La tabla B.1 contiene los directorios comprendidos en *classic* y la descripción de contenido de cada uno.

Directorio	**Descripción**
browser	Contiene todos los iconos de la barra de herramientas, así como los iconos del administrador de marcadores y la ventana de preferencias.
global	Contiene casi todos los archivos CSS (*Cascading Style Sheets*), también llamadas hojas de estilo, que definen la apariencia del navegador.
mozapps	Contiene todos los estilos e iconos para los accesorios del navegador, como el administrador de extensiones y el asistente de actualización.
help	Contiene todos los archivos para la ventana de ayuda.
communicator	Contiene una hoja de estilo de uso general.

Tabla B.1: Directorios del tema por defecto de Mozilla Firefox.

El proceso para la edición e instalación del tema modificado comprendió los siguientes pasos:

1. Copiar los archivos necesarios. Se creó el directorio “VInput_Theme” donde se copiaron los directorios *browser*, *global*, *communicator*, *help*, y *mozapps*; así como los archivos *icon.png* y *preview.png*.

2. Obtener y modificar los archivos *contents.rdf* e *install.rdf*. Se obtuvieron en Internet y ambos son bases de datos en formato XML que se utilizan para describir el skin. En estos archivos se especificó que el nombre del tema es *vInput_Theme* y fueron colocados dentro del directorio creado.

3. Modificar parámetros de los archivos CSS del directorio global. Las hojas de estilo indican al navegador como visualizar los botones y otros controles, donde poner las imágenes, que borde y que relleno debería colocar alrededor de ellos, entre otros. Para mejorar el uso de VInput, se modificó

en los archivos *xulscrollbars.css* y *browser.css* los parámetros de: *scrollbarbutton, toolbar, menubutton, arrowscrollbox* y *scrollbutton*. El cambio consintió en aumentar el tamaño de estos componentes.

4. Empaquetar el directorio creado en archivo JAR.

5. Ejecutar la instalación del tema modificado. Finalmente, para ver los cambios en la interfaz de Firefox solamente se abre la ventana de temas en Mozilla y arrastrar el archivo .jar en ella, como se muestra en la figura B.1.

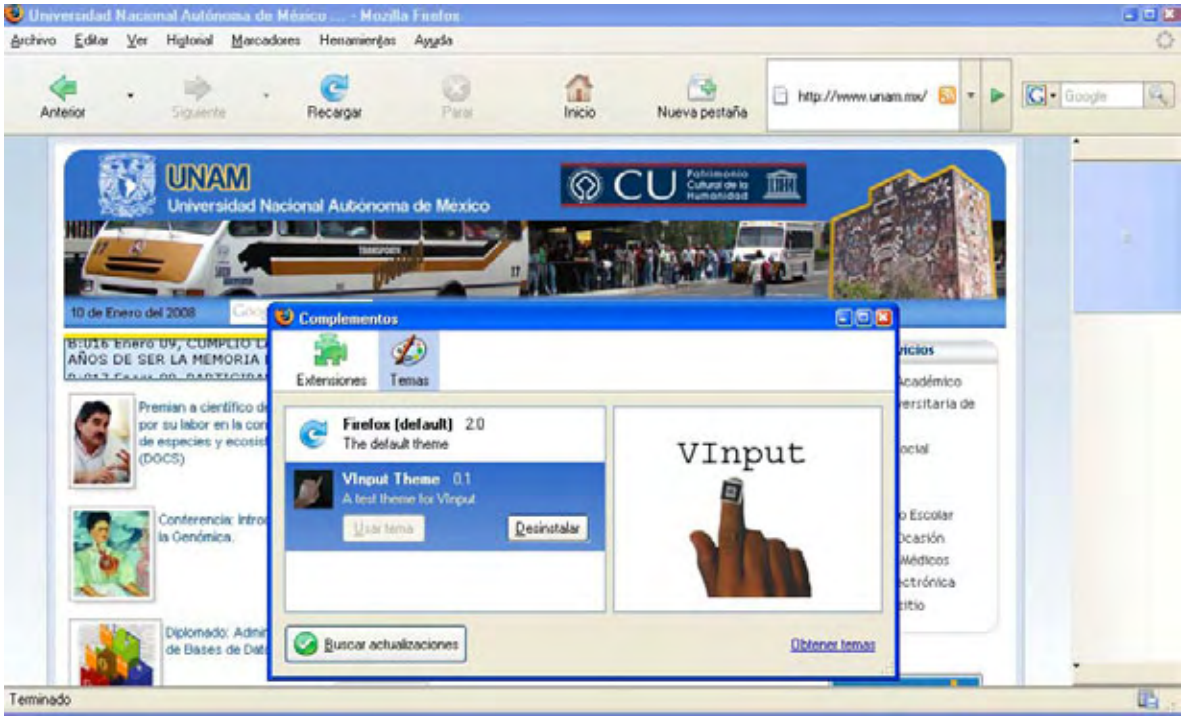


Figura B.1: Interfaz gráfica modificada de Mozilla Firefox integrada con VInput.

GLOSARIO

API	*Application Programming Interface.* Interfaz de Programación de Aplicaciones.
Autodesk 3ds Max	Herramienta para la plataforma Windows que permite la creación de modelos y animaciones 3D.
Bit	Acrónimo de *Binary digit* (dígito binario). Es un dígito del sistema de numeración binario para representar o codificar información en un dispositivo digital.
BMP	Abreviatura de BitMaP (mapa de bits) es un tipo de formato usado para almacenar imágenes digitales.
CD	Abreviación de *Compact Disc*, un CD sirve para almacenar datos digitales.
DirectShow	Es un marco de trabajo y API desarrollado por Microsoft para realizar operaciones con archivos multimedia, especialmente para reproducir o capturar audio y video.
DirectX	Es un paquete distribuido gratuitamente de APIs creadas para el desarrollo de tareas relacionadas con el soporte multimedia en la plataforma Microsoft Windows.
Dispositivo Háptico	Dispositivo que reproduce sensaciones táctiles y que proporciona una retroalimentación de fuerzas al usuario.
DVD	Abreviación de *Digital Versatile Disc* (Disco Digital Versátil), un DVD sirve para almacenar datos digitales.
FireWire	El IEEE 1394 o FireWire es un estándar multiplataforma para entrada/salida de datos en serie. Suele utilizarse para la interconexión de dispositivos digitales como cámaras digitales y videocámaras a computadoras.
GIS	Abreviación de *Geographic Information System*. Sistema informático que permite almacenar y relacionar atributos alfanuméricos o información geográficamente referenciada con entidades gráficas.
GLP	Licencia Pública General que está orientada principalmente a proteger la libre distribución, modificación y uso de software.
Grafo	Es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.

HLSL	Abreviación de *High Level Shading Language* (Lenguaje de Sombreado de Alto Nivel) desarrollado por Microsoft.
HMD	Acrónimo de *Head-Mounted Display*. Es un dispositivo de despliegue para la cabeza o como parte de un casco, que tiene una pequeña pantalla óptica en cada ojo.
JAR	Acrónimo de *Java Archive*. Es un formato o tipo de archivos comprimidos.
Joystick	Dispositivo de entrada, típico de los videojuegos, consistente en una palanca que gira sobre un pivote y que permite controlar el movimiento de un objeto en la pantalla.
JPEG	Acrónimo de *Join Photograph Expert Group*. Formato de archivo gráfico con compresión que se utiliza para mostrar imágenes en color de alta resolución.
Lenguaje C	Es un lenguaje de programación estructurado de nivel medio independiente del sistema operativo y de propósito general.
Lenguaje C++	Es un lenguaje de programación orientado a objetos de nivel alto que extiende al lenguaje de programación C.
LGPL	Acrónimo de *Lesser General Public License* (Licencia Pública General Menor). Esta licencia se aplica a cualquier programa o trabajo que contenga una nota puesta por el propietario de los derechos del trabajo estableciendo que su trabajo puede ser distribuido bajo los términos de esta licencia.
Maya	Software dedicado al desarrollo de gráficos en 3D, efectos especiales y animación.
Milkshape	Herramienta de creación de modelos 3D para Windows. Es un modelador poligonal, capaz de importar y exportar varios formatos.
Motor gráfico	Conjunto de rutinas de programación que permiten la creación y la representación de objetos tridimensionales.
Mouse	Periférico de entrada de la computadora de uso manual utilizado como entrada o control de datos. Se utiliza con una de las dos manos del usuario y detecta su movimiento relativo en dos dimensiones por la superficie horizontal en la que se apoya, reflejándose habitualmente a través de un puntero o flecha en el monitor.
OpenGL	Acrónimo de *Open Graphics Library*. Es una especificación estándar desarrollada por Silicon Graphics Inc. (SGI) que define una API multilenguaje y multiplataforma para aplicaciones que produzcan gráficos 2D y 3D.

OIS	Acronimo de *Object oriented Input System*. Es una biblioteca de código abierto que permite manejar entradas de teclados, mouse, y joysticks.
PC	Acrónimo de *Personal Computer* (Computadora personal).
PDA	Acrónimo de *Personal Digital Assistant* (Ayudante personal digital). Computadora portátil de mano.
Píxel	Acrónimo de *Picture Element*. Unidad básica que compone una imagen digital, ya sea una fotografía, un fotograma de video o un gráfico.
Plug-in	Es un programa que añade nuevas funciones a un determinado sistema ya existente.
PNG	Acrónimo de *Portable Network Graphics* (Gráficos Portátiles de Red). Es un formato de imágenes gráficas comprimidas sin perdida (reducción del tamaño del archivo).
Realidad Aumentada	Es el concepto de combinar el mundo real con objetos virtuales. Para ello, se hace una superposición de datos generados por computadora sobre el campo de visión primaria.
Realidad Virtual	Proceso de simulación de la realidad mediante la generación de entornos sintéticos en tiempo real o representación de cosas a través de medios electrónicos.
Render	Es un proceso de cálculo desarrollado por la computadora destinado a generar una imagen 2D a partir de una escena 3D, con lo cual se pone en pantalla los ambientes y objetos.
ROM	Acrónimo de *Read Only Memory* (memoria de solo lectura). En esta memoria se almacena ciertos programas e información de manera permanente.
RTS	Acrónimo de *Real Time Strategy* (Estrategia en Tiempo Real). Es un género o tipo de juego de estrategia en los que no hay turnos sino que el tiempo transcurre de forma continua.
Tracker	Programa utilizado para el rastreo o seguimiento de las marcas que pudieran estar en el video captado por la cámara.
UMPC	Acrónimo de *Ultra-Mobile PC* (PC Ultra Móvil). Computadora personal portátil con entrada táctil, manuscrita o por teclado.
USB	Acrónimo de *Universal Serial Bus* (Bus Universal en Serie). Es una interfaz de tipo plug and play entre una computadora y ciertos dispositivos permitiendo que sean conectados o desconectados al sistema sin necesidad de reiniciar. Cuando se conecta un nuevo dispositivo, se agrega el software necesario para que pueda funcionar.

VfW	Abreviación de *Video for Windows*. Es un marco de trabajo multimedia desarrollado por Microsoft para reproducir video digital.
XML	Acrónimo de *Extended Markup Language* (lenguaje de marcas extensible). Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C) que permite definir la gramática de lenguajes específicos.