UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

CENTRO DE INFORMÁTICA INDUSTRIAL

Módulo cliente-servidor para la comunicación de dispositivos periféricos en Sistemas de Realidad Virtual

Tesis presentada para optar por el Título de Ingeniería en Ciencias Informáticas

Autor: **Suset Espinosa Guedes**

Tutor: **MSc. Liudmila Pupo Peña**

Co-tutor: **MSc. Yoander Cabrera Díaz**

La Habana, Junio de 2011

DECLARACIÓN JURADA DE AUTORÍA

Declaramos ser los únicos autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste, firmamos la presente declaración de autoría en La Habana a los ___ días del mes de ___ del año ___.

Suset Espinosa Guedes

Autor

MSc. Liudmila Pupo Peña

Tutor

DATOS DE CONTACTO

Nombre y Apellidos: Liudmila Pupo Peña

Edad: 26 años

Ciudadanía: cubana

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: MSc en Informática Aplicada

Categoría Docente: Profesor Instructor

E-mail: lpupo@uci.cu

Graduada de la UCI, con cuatro años de experiencia en el tema de la Gráfica Computacional y líder del proyecto Herramienta de Desarrollo para Sistemas de Visión Estereoscópica (HDSVE) del Centro de Informática Industrial (CEDIN).

Agradecimientos.

A lo largo de mis estudios han sido muchas las personas que me han ayudado o apoyado en momentos difíciles, mencionarlas a todas sería engorroso, pero sí quisiera dejar constancia de mi eterno agradecimiento:

A mi madre Marisol y mi padre Heriberto, por darme todo el amor y cariño que una hija puede desear de sus padres, siempre comprendiendo las decisiones que he tomado aunque algunas han provocado mi separación física de ellos.

A mis hermanas Sandra e Ivet, que me han servido de ejemplo a seguir en lo personal y en la carrera.

A mi novio Tony, por todo su amor durante estos casi cinco años de carrera y relación.

A mi amiga del alma Anay, por haber compartido conmigo todos estos años de universidad y conformar el dúo inseparable que somos.

A mi tutora Liudmila por todo su apoyo en la realización de esta tesis, sin su guía no hubiera podido lograrlo.

A todos los compañeros de aula, laboratorio y profesores que tuve durante la carrera.

Y por último, pero muy especialmente, a mi abuela Margot, por ser también una madre para mí. Para ti va mi mayor agradecimiento porque por ti es que he podido terminar mis estudios satisfactoriamente. Muchísimas gracias mi abuelita linda por cuidarme y criarme de la forma que lo hiciste, parte de lo que soy hoy es gracias a ti. Muchísimas gracias por ser tan buena madre y abuela, por haber dejado tu vida a un lado para formar la mía en reiteradas ocasiones y alejarte de tus seres queridos por estar conmigo.

...a mis padres por todo su apoyo

...a mis hermanas por toda su ayuda

...a mi abuela por ser quien es

...y a Tony por todo su amor

Resumen

En una aplicación de Realidad Virtual (RV) frecuentemente se necesita interactuar con varios dispositivos de características disímiles. En la medida que se incrementa el número de éstos, su integración se hace compleja y más aún si se tiene en cuenta que la interfaz para interactuar con cada uno de ellos puede ser diferente. Una capa de abstracción que brinde la información obtenida del dispositivo puede ser necesaria para obtener información final del dispositivo. Incluso puede querer brindarse comportamientos a objetos a partir de los datos obtenidos desde múltiples dispositivos, que pueden estar manejados por diferentes personas. Actualmente, en el desarrollo de productos con la biblioteca *Graphics Library for Stereo Vision* (GLSVe), no es posible la gestión de la información de varios dispositivos a la vez, lo que dificulta su utilización en las aplicaciones que lo necesiten. Es por ello que el objetivo de esta investigación es implementar un módulo que permita la comunicación de diferentes dispositivos conectados a computadoras clientes con la aplicación del servidor para ser utilizados por la biblioteca GLSVe. Para desarrollar la aplicación se utiliza Visual Studio con C♯ como lenguaje de programación y el formato XML para almacenar los datos que arrojan los periféricos, que son luego enviados al servidor. Se logró obtener un módulo que permite manipular la información de varios dispositivos periféricos (*mouse*, teclado, *wiimote* y *webcam*), desde varias computadoras, facilitando la interacción desde la biblioteca GLSVe con estos dispositivos. La arquitectura genérica obtenido puede ser utilizada en aplicaciones de RV que necesiten esta funcionalidad.

Palabras clave: cliente, dispositivo, servidor, RV.

Índice general

Introducción **1**

1. Fundamentación Teórica **5**

1.1. Realidad Virtual y la utilización de los dispositivos de interacción 5

1.1.1. Dispositivos periféricos de interacción 6

1.1.2. Valoración de los dispositivos periféricos 12

1.2. Transmisión de datos por red 13

1.2.1. Protocolos de red 13

1.2.2. Arquitecturas de red 15

1.2.3. Formato de los datos 17

1.3. Aplicaciones relacionadas con la comunicación de dispositivos y el envío de datos por red 18

1.3.1. Sistema de Realidad Virtual basado en múltiples pantallas distribuidas . . 18

1.3.2. Gestión de estados dinámicos compartidos para ambientes virtuales interactivos distribuidos 20

1.3.3. Sistema de dispositivos de entrada virtuales en .Net 21

1.4. Herramientas y tecnologías a usar 23

2. Solución Propuesta **25**

2.1. Dispositivos a manejar 25

2.1.1. *Mouse* 25

2.1.2. Teclado 26

2.1.3. *Wiimote* 26

2.1.4. *Webcam* 28
2.2. Envío de datos por red 29
2.2.1. Protocolo de red 29
2.2.2. Formato de los datos 30
2.3. Diseño lógico de solución 32

3. Ingeniería del sistema **34**
3.1. Modelo de Dominio 34
3.2. Definición de requisitos 35
3.2.1. Requisitos Funcionales 35
3.2.2. Requisitos No Funcionales 36
3.3. Casos de Uso 36
3.3.1. Actor del sistema 36
3.3.2. Diagrama de Casos de Uso 37
3.3.3. Descripción de Casos de Uso 37
3.4. Diagrama de clases del diseño 43
3.5. Patrones de diseño 46
3.6. Diagramas de interacción 48
3.6.1. Diagramas de secuencia por Casos de Uso 48
3.7. Diagramas de despliegue 51
3.8. Diagrama de componentes 51

4. Validación de la solución **53**
4.1. Cliente 53
4.2. Servidor 55

Conclusiones **57**

Recomendaciones **58**

Referencias bibliográficas **59**

Acrónimos **62**

Anexos **64**

Índice de figuras

1.1. Wiimote 7
1.2. Webcam 8
1.3. 5DT Data Glove 9
1.4. Clasificación de los teclados en cuanto a la pulsación 10
1.5. *Mouse* mecánico 11
1.6. *Mouse* óptico 11
1.7. *Mouse* láser 11
1.8. *Touchpad* 12
1.9. Sistema basado en pantallas distribuidas 19
1.10. Ejemplo de sistema KAVE usando NAVERLib 20
1.11. Escritorio remoto 22
1.12. Control de objetos 3D 22
1.13. Pizarra virtual 23

2.1. Aplicación realizada por el autor de la biblioteca *WiimoteLib* para probar su funcionamiento 28
2.2. Formato XML para el *mouse* 30
2.3. Formato XML para el teclado 31
2.4. Formato XML para el *wiimote* 31
2.5. Formato XML para la *webcam* 32
2.6. Diseño lógico de solución 33

3.1. Modelo de Dominio 35
3.2. Diagrama de Casos de Uso 37

3.3. Diagrama de paquetes del diseño 44
3.4. Paquete Cliente 45
3.5. Paquete Servidor 46
3.6. Diagrama de secuencia para el CU1: Permitir Conexión 48
3.7. Diagrama de secuencia para el CU2: Establecer Conexión 49
3.8. Diagrama de secuencia para el CU3: Capturar información dispositivo 50
3.9. Diagrama de secuencia para el CU4: Procesar información 50
3.10. Diagrama de despliegue 51
3.11. Diagrama de componentes para el cliente 52
3.12. Diagrama de componentes para el cliente 52

4.1. Interfaz Cliente: Menú principal 54
4.2. Interfaz Cliente: Conectado al servidor 54
4.3. Interfaz Cliente: Captura del *mouse* 55
4.4. Interfaz Servidor 56
4.5. Interfaz Servidor: Movimiento del teclado, *webcam* y *mouse* 56
6. Patrón utilizado en el seguimiento que se realiza con la *webcam* 64

Índice de tablas

2.1. Propiedades manejadas por el *mouse* 26
2.2. Propiedades manejadas por el teclado. 26
2.3. Propiedades manejadas por el *wiimote*. 27
2.4. Propiedades manejadas a través de la *webcam*. 29

3.1. Actores del sistema 37
3.2. Descripción del CU1: Permitir Conexión 38
3.3. Descripción del CU2: Establecer Conexión 39
3.4. Descripción del CU3: Gestionar información de dispositivos 42
3.5. Descripción del CU4: Procesar información 43

Introducción

Desde mediados del siglo XX se dieron los primeros pasos para recrear una parte de la realidad a través de entornos virtuales. Con el advenimiento de las computadoras personales y la mejora de sus capacidades, en la década del '90 se incrementa el número de herramientas para el desarrollo de estos entornos [Quiroz, 2007].

Estos avances no hubieran podido lograrse sin la creación de distintos dispositivos que permiten mayor interacción del usuario con el sistema. Por ejemplo los guantes permiten la manipulación y selección de objetos en entornos virtuales. Para controlar simuladores y en juegos de carreras se utilizan el volante y los pedales, y existen además los cascos de RV que integran varios sentidos como la visión, audición y orientación.

Existen además varios dispositivos que se emplean para captar posicionamiento y comportamiento del usuario, entre ellos están el *wiimote*, desarrollado por Nintendo como parte de la videoconsola Wii y la *webcam*, que no es más que una pequeña cámara utilizada para la transmisión de imágenes a través de la web principalmente [Educativo, 2009].

En la UCI, surgida en el 2005 como parte de la Batalla de Ideas, se han creado varios proyectos para respaldar económicamente al país. En el CEDIN, perteneciente a la Facultad 5, se encuentra el proyecto HDSVE, que en conjunto con la Universidad de Oviedo de España ha desarrollado la biblioteca GLSVe para construir aplicaciones de RV que utilicen visualización estereoscópica.

En una aplicación de RV frecuentemente se necesita interactuar a la vez con varios dispositivos de características disímiles[1]. En la medida que se incrementa el número de dispositivos a utilizar en un programa, su integración se hace compleja y más aún si se tiene en cuenta que la interfaz para interactuar con cada uno de ellos puede ser diferente. Una capa de abstracción

[1]Ej. webcam en el procesamiento de imágenes para hacer seguimiento de la cabeza del observador, el *wiimote* o los guantes de RV para interactuar con los objetos de la escena, etc.

que brinde la información obtenida del dispositivo puede ser necesaria para obtener información final del dispositivo (sin tener que interactuar con el *driver* o la biblioteca que sirva de interfaz). Puede ser necesario también brindar comportamientos a objetos a partir de los datos obtenidos desde múltiples dispositivos, que incluso pueden estar manejados por diferentes personas.

Teniendo en cuenta lo antes mencionado, la gestión de varios dispositivos periféricos desde una sola computadora puede ser compleja, una de las opciones es permitir la manipulación de estos dispositivos desde varios clientes conectadas a la red que envíen la información a un servidor central para ser utilizada. Actualmente, en el desarrollo de productos con la biblioteca GLSVe, no es posible la gestión de la información de varios dispositivos a la vez, lo que dificulta su utilización en las aplicaciones que lo necesiten.

De ahí que el **problema de investigación** sea: ¿Cómo capturar la información de varios dispositivos periféricos y enviar esta información a una computadora central para ser utilizados por la biblioteca GLSVe?

Partiendo del problema de investigación se puede definir como **objeto de estudio** la captura de información de los dispositivos periféricos y su envío a una computadora central. Teniendo como **campo de acción** la captura de información de los dispositivos periféricos y su envío a una computadora central para ser utilizados por la biblioteca GLSVe.

Como **objetivo general** se propone desarrollar un módulo que permita la comunicación de diferentes dispositivos conectados a una computadora cliente con la aplicación del servidor para ser utilizados por la biblioteca GLSVe.

Para dar cumplimiento al objetivo se han trazado las siguientes **tareas de investigación**:

- Analizar el funcionamiento de los dispositivos periféricos más usados en los sistemas de RV para identificar sus principales características
- Estudiar los protocolos de comunicación en el envío de información por red para seleccionar el más adecuado
- Evidenciar la existencia de aplicaciones que realizan el envío por red de la información de dispositivos periféricos en Sistemas de Realidad Virtual (SRV)
- Elaborar soluciones técnicas que permitan cumplir el objetivo planteado
- Validar el correcto funcionamiento de la solución propuesta

Como **resultado** de este trabajo se espera obtener un módulo que permita utilizar la biblioteca GLSVe para construir aplicaciones que utilicen varios dispositivos periféricos manipulados desde diferentes computadoras. Este módulo debe proveer a GLSVe de un componente importante para lograr mayor inmersión en las aplicaciones de RV que se elaboren.

Se aplicarán los siguientes **métodos científicos de investigación**:

Métodos teóricos:

- Analítico-Sintético: con el objetivo de analizar teorías y documentos relacionados con el envío de datos a través de la red y captura de información arrojada por los dispositivos periféricos de interacción
- Histórico-Lógico para conocer la evolución y existencia de algunos trabajos relacionados con el envío de datos por la red y utilización de dispositivos periféricos en SRV

Métodos empíricos:

- Experimento para poder verificar a través de diferentes pruebas la mejor variante para desarrollar la aplicación final

El documento se ha estructurado por capítulos de la siguiente forma:

Capítulo 1: Fundamentación teórica: se hace una descripción de los dispositivos más usados en la creación de entornos virtuales, específicamente la *webcam*, el *wiimote*, teclado y *mouse*, así como su importancia para lograr una mayor inmersión en el sistema. Se caracterizan algunos de los protocolos y arquitecturas para la transmisión de datos en la red, así como algunos trabajos relacionados con el manejo de dispositivos en entornos virtuales y el envío de datos por la red. Por último, se hace un breve resumen de las tecnologías y herramientas a utilizar.

Capítulo 2: Solución propuesta: después de analizar las diferentes variantes para resolver el problema de investigación se hace una propuesta de solución, describiendo cómo se realiza la captura de información desde los diferentes dispositivos periféricos seleccionados y el formato de fichero para guardar esta información. Se define el protocolo de red a utilizar y el diseño general del sistema.

Capítulo 3: Ingeniería del sistema: teniendo en cuenta la solución propuesta se definen los requisitos funcionales y no funcionales del sistema, y se modelan las características del diseño e implementación del sistema a través de los diagramas correspondientes.

Capítulo 4: Validación de la solución: se documenta la aplicación de prueba realizada que permite verificar el buen funcionamiento del módulo.

Capítulo 1

Fundamentación Teórica

En este capítulo se hace una descripción de los principales dispositivos periféricos utilizados en los entornos de RV para la comprensión de sus características y funcionamiento, así como la importancia de su uso para lograr una mayor inmersión. Se caracterizan algunos de los protocolos más empleados en la transmisión de información a través de la red y algunos de los modelos de arquitectura existentes para diseñarlos. Se presentan investigaciones donde se evidencia la interacción con dispositivos periféricos y el envío de esta información por red. Además se justifica el uso de herramientas y tecnologías utilizadas para el desarrollo de la aplicación, así como la metodología de desarrollo de software empleada.

1.1. Realidad Virtual y la utilización de los dispositivos de interacción

Por lo general, se define la RV como un mundo virtual generado por ordenador (o sistemas informáticos) en el que el usuario tiene la sensación de estar en el interior de este mundo, y dependiendo del nivel de inmersión es posible interactuar con este mundo virtual y sus objetos en mayor o menor grado [RV, 2005].

La RV juega un papel muy importante en la sociedad hoy día, con esta tecnología se han podido crear aplicaciones que se extienden a casi todas las demás ramas sociales y científicas, por ejemplo en la salud, creando espacios virtuales para tratamientos psicológicos y simulación de operaciones; en educación, para el proceso de aprendizaje; y en el entretenimiento desarrollando

videojuegos, entre otros campos.

Existen dos tipos de RV: la inmersiva y la no inmersiva, en esta última el usuario nunca pierde la noción del entorno que lo rodea, para ello se hace uso de diferentes dispositivos, como los *joysticks*, teclado y *mouse*. A diferencia de la inmersiva (que aleja completamente de la realidad a través de los cascos principalmente) la no inmersiva es más empleada y aceptada [Villanueva, 2002].

La capacidad más importante que debe tener un espacio virtual es la posibilidad que el usuario pueda moverse por este, recorrerlo, explorarlo. El modo más común de hacer esto es mediante el teclado, pero con el paso del tiempo y ante mayores exigencias se han añadido otros dispositivos que incrementan el grado de inmersión en el escenario [Juárez, 2004].

Se ha necesitado crear espacios donde los sentidos humanos se vean estimulados y extendidos al mundo virtual y los dispositivos lo han permitido. Las cámaras por ejemplo permiten captar imágenes en tiempo real y llevarlas a la computadora, logrando que lo que se realiza en el exterior de la pantalla se reproduzca en esta, permitiendo la interacción con lo que ocurre dentro. Lo mismo sucede con aquellos periféricos que captan los movimientos de las manos o el cuerpo de forma general, traduciendo así la posición real de los datos capturados del usuario al entorno virtual.

1.1.1. Dispositivos periféricos de interacción

Existen varios dispositivos de interacción, cada uno creado para un propósito específico, por ejemplo el casco de RV aleja completamente al usuario del entorno real, los mandos tradicionales de video-juegos logran mayor interacción con lo que ocurre en la pantalla que el teclado y el *mouse*. A continuación se describen el *wiimote*, la *webcam*, el *Data Glove* (Guante) por ser algunos de los más usados en estos sistemas, al permitir captar posición y movimiento en el mundo real y llevarlo al sistema virtual. También se hace una breve descripción del teclado y el *mouse*.

Wiimote

El *Wii Remote*, o más conocido como *wiimote* forma parte de las consolas de 7ma generación que han estado invadiendo el mercado de los videojuegos en los últimos años, pertenece a la con-

sola Wii fabricada por Nintendo y es muy aceptada a nivel mundial por las nuevas posibilidades que brinda, incluyendo su fácil manejo y la comunicación con la consola a través de *Bluetooth*.

Su forma es muy parecida a la de un mando de televisor, midiendo 14.6 x 3.5 x 3.1 cm. Contiene los botones A, B (gatillo trasero), 1, 2, Home, mas (+), menos (-), además del botón de encendido, una cruceta digital y una cuerda para evitar caídas accidentales (véase Figura 1.1). Se puede utilizar como puntero y sus capacidades de detección de movimiento dejan atrás otras tecnologías para el control de entornos 3D, como el *joysticks* o la difícil combinación de botones de mandos [Caballero, 2006].

Figura 1.1: Wiimote

Además de las funciones antes mencionadas cuenta con otras que lo hacen aún más versátil y atractivo. La primera es la posibilidad de conectarle otros dispositivos periféricos mediante un puerto de expansión ubicado en la parte inferior. Otra es la vibración (*rumble*), que brinda aún más sensación de realidad ante lo que ocurre en la pantalla y lo que se oye. Por último se puede mencionar un pequeño altavoz dedicado a emitir choques de espadas, colisiones, disparos, etc., que se puede controlar mediante los botones "+" y "-" [Caballero, 2006].

La detección de movimientos se hace a través de un acelerómetro que lleva incorporado, el cual proporciona los datos de giro/inclinación y movimiento, este se conoce como Sistema Micro Electro-Mecánico (MEMS). El MEMS usado en el desarrollo del *wiimote* es el ADXL330, fabricado por Analog Devices y el rango de detección abarca +-3 grados, se usa para detectar y medir vibraciones, aceleración e inclinación [Caballero, 2006].

Otro dispositivo electrónico que posee el Wii es el chip de memoria de 16Kb *Erasable Programmable Read Only Memory* (EEPROM), de la que se disponen 6Kb por parte del usuario para leerla y escribirla libremente, en esta parte es donde se pueden crear los diferentes perfiles. Estos

datos se encuentran localizados entre los registros de memoria 0x0070 al 0x176F. Además de la porción de memoria accesible para el usuario también existen otras dos divisiones: los registros de control y los registros de entrada, haciendo referencia a los diferentes periféricos incluidos en el mando y a los tres ejes del acelerómetro y los botones, respectivamente [Navarro, 2009].

El *wiimote* incluye una cámara de 128x96 píxeles sensible a la luz infrarroja capaz de seguir hasta cuatro objetos móviles. La cámara tiene tres estados: no recogida de datos, recogiendo datos con sensibilidad media y recogiendo datos con sensibilidad alta [Navarro, 2009].

Webcam

La *webcam* no es más que una pequeña cámara usada para capturar y trasmitir imágenes a través de la red (véase Figura 1.2), encontrando su principal uso en mensajerías instantáneas y captura de imágenes en SRV. La cámara toma fotogramas cada cierto tiempo y los envía hacia un destino, la velocidad recomendada para formar un video sin grandes saltos apreciables es de 15 a 30 fotogramas por segundo. Está formada por una lente, un sensor de imágenes y los circuitos asociados [Educativo, 2009].

Figura 1.2: Webcam

La luz de la imagen pasa por la lente y se refleja en un filtro *Red, Green, Blue* (RGB), descomponiendo la luz en los tres colores básicos: rojo, verde y azul. Esta información se concentra en el chip *Charged Coupled Device* (CCD) que asigna valores binarios a cada píxel. La resolución del video es baja (aproximadamente 640x480 píxeles) y dependiendo del modelo, la lente puede girar hasta 360° horizontales [Educativo, 2009].

Guantes

Existen dos tipos principales de guantes de RV: los exoesqueletos y los guantes de datos. Los primeros tienen una estructura mecánica sobrepuesta a la mano y sensores en cada articulación. El material con que están confeccionados los guantes de datos es lycra y tienen cables de fibra óptica por cada dedo, cada una de estas fibras tiene un emisor de luz al inicio y un sensor al final para poder determinar los giros de la muñeca a través de la intensidad de luz recibida [Ledesma, 2010].

Los guantes de datos (*Data Gloves*) están definidos por varios parámetros: el número de sensores, el tipo de aplicación en que se usa, el tipo de conexión periférica, entre otros. Entre los más usados están: *5DT Data Glove* (véase Figura 1.3), con cinco sensores, uno por cada dedo; *CyberGlove*, que posee hasta 22 sensores de alta precisión; *CyberForce*, guante de escritorio con brazo incorporado; *Virtual Hand Studio*, para las aplicaciones 3D [Gua, 2001].

Figura 1.3: 5DT Data Glove

Teclado

El teclado es un dispositivo de entrada inspirado en las máquinas de escribir. Tiene las teclas generalmente divididas en cuatro bloques: bloque de funciones (F1-F12), bloque alfanumérico (letras, números y teclas especiales), bloque especial y bloque numérico. Realiza sus funciones mediante un microcontrolador, que para lograr un sistema flexible le adjudica un valor numérico a cada tecla acorde a su posición física.

En cuanto a la pulsación existen 2 tipos básicos de teclado: los de membrana y los mecánicos. Los de membrana constan de un par de contactos separados por una distancia mínima, aproximadamente el grosor de una hoja de papel, que hacen contacto al oprimir la tecla y esta regresa a su estado original a través de una membrana plástica con una media burbuja que empuja la

tecla hacia arriba. Los mecánicos utilizan un resorte metálico que al ser oprimido por la tecla genera la unión de dos contactos. Ver Figura 1.4 [Tec, 2011].

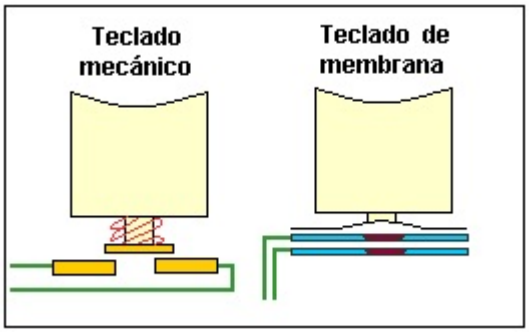

Figura 1.4: Clasificación de los teclados en cuanto a la pulsación

Mouse

El ratón (o *mouse* en inglés) es un periférico de entrada que junto al teclado es imprescindible para muchos de los usuarios de computadoras. Detecta el movimiento en dos dimensiones: horizontal (eje x) y vertical (eje y), y en la pantalla generalmente aparece como un puntero o flecha. Generalmente contiene dos botones (derecho e izquierdo) y una rueda central (*scroll*). Se popularizó en la década de los 80 cuando salió al mercado formando parte de la Macintosh, logrando el objetivo de mejorar la comunicación hombre-máquina existente hasta entonces.

Entre los principales tipos de *mouse* existentes se encuentran [González, 2009b]:

- Mecánico: tienen una esfera plástica o de goma de varias capas en su parte inferior para mover dos ruedas que generan pulsos en respuesta al movimiento de este sobre la superficie (Véase Figura 1.5)
- Óptico: es considerado uno de los más prácticos y usados actualmente. Su funcionamiento se basa en un sensor óptico que fotografía la superficie sobre la que se encuentra y detectando las variaciones entre las sucesivas imágenes se determina si hubo movimiento. Evita la acumulación de suciedad en el eje de transmisión muy frecuente en los mecánicos pero en superficies pulidas causa movimiento nervioso en la pantalla, lo que puede solucionarse con un *mouse pad* (Véase Figura 1.6)

- Láser: usa también una superficie de apoyo pero el haz de luz se sustituye por un láser invisible al ojo humano, lo que significa un aumento en la precisión y sensibilidad, aconsejable principalmente para los diseñadores de gráficos y fanáticos de videojuegos (Véase Figura 1.7)

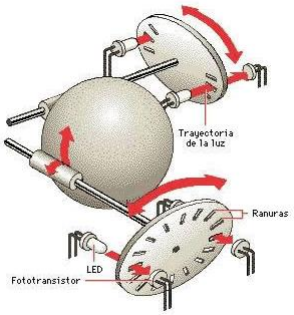

Figura 1.5: *Mouse* mecánico

Figura 1.6: *Mouse* óptico

Figura 1.7: *Mouse* láser

Las computadoras portátiles usan un modelo especial llamado *touchpad* que no es más que un cuadrado sensible al tacto ubicado debajo de la barra espaciadora y cuenta con las siguientes partes (Véase Figura 1.8) [González, 2009a]:

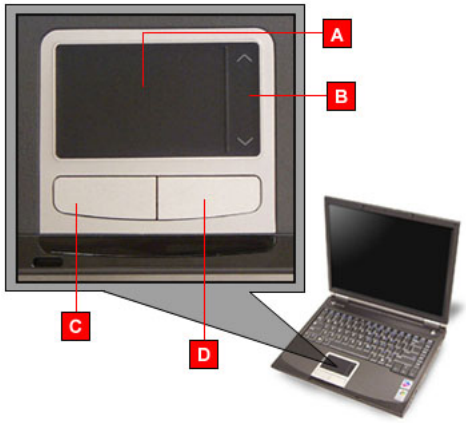

Figura 1.8: *Touchpad*

- A: Superficie sensible al tacto.
- B: Sector de desplazamiento.
- C: Botón izquierdo.
- D: Botón derecho.

1.1.2. Valoración de los dispositivos periféricos

Como se ha podido apreciar, los dispositivos periféricos juegan un papel muy importante en la creación de SRV que logren, al menos, una mínima inmersión del usuario en el mundo virtual que se recrea. Para seleccionar el más adecuado en un sistema hay que tener en cuenta el propósito de su utilización, sus características y el costo que representa para la implantación. Por ejemplo, el teclado y el *mouse* son los más usados en aplicaciones de escritorio debido a que están incluidos en la mayoría de las computadoras, sin embargo, no satisfacen la necesidad de interacción o inmersión que se desea lograr en determinados SRV, como pueden ser los *Computer Aided Virtual Environment* (CAVE). Ocurriendo lo contrario con dispositivos como la *webcam*, los guantes, cascos o mandos de video-juegos que conllevan un gasto adicional pero logran una mayor percepción de realidad por parte del usuario que los utiliza.

Ahora bien, si se desea crear un SRV donde interactúen varios dispositivos a la vez, la manipulación de estos en una sola computadora puede ser ineficiente y conllevar un trabajo engorroso de implementación, si se tiene en cuenta que para la comunicación con cada uno de

ellos se necesitan *drivers* o bibliotecas diferentes. Una opción para solucionar este conflicto es hacer la gestión de los dispositivos desde distintas computadoras y la información que se capture de estos puede ser enviada a otra computadora encargada de manipularla y utilizarla.

1.2. Transmisión de datos por red

La transmisión de información siempre ha sido muy importante para la sociedad, en un principio a través de historias contadas, luego mediante escritos de antepasados y más reciente por medio de la red o internet. Esta última se ha desarrollado llegando al punto de ser imprescindible en muchos sistemas actuales. Para la transmisión de datos a través de la red se emplean distintos protocolos y arquitectura de red, en este epígrafe se hace referencia a algunos de los más usados.

1.2.1. Protocolos de red

Un protocolo es un conjunto de procedimientos y reglas para el envío y recepción de datos en una red, que define el formato y la frecuencia de la información que transita por esta. Se clasifican en dos categorías según el nivel de control de los datos requeridos: orientados a la conexión y no orientados a la conexión. Los primeros controlan la transmisión durante la comunicación establecida y los segundos envían y reciben datos sin notificación alguna [Jeff, 2008].

Existen varios protocolos usados con fines específicos y relacionados entre sí, algunos de ellos son:

- *HyperText Transfer Protocol* (HTTP) se utiliza para acceder a las páginas web
- *File Transfer Protocol* (FTP) para transmisión de archivos
- *Simple Mail Transfer Protocol* (SMTP) para envío de mensajes de correo entre servidores
- *Post Office Protocol* (POP) para correo electrónico
- *User Datagram Protocol* (UDP) es no orientado a la conexión por lo que permite errores en la transmisión a cambio de mayor velocidad

Este conjunto de protocolos se denomina protocolos de internet o conjunto de protocolos TCP/IP, debido a los dos protocolos más importantes que lo conforman: *Internet Protocol* (IP)

y *Transmission Control Protocol* (TCP). A continuación se hace una descripción más detallada de los protocolos TCP, IP y UDP.

Protocolo de Internet

IP es un protocolo no orientado a la conexión que se encarga de repartir los bloques de datos, llamados datagramas o paquetes, enviados a través de la red. Posee un servicio de datagramas no fiable conocido como *best effort*, que hace el envío lo mejor posible pero con poca garantía, sin necesidad de configuración antes de intentar enviarlo aunque no se haya comunicado antes con el receptor [Jeff, 2008].

Para el envío, IP etiqueta los paquetes con cierta información, entre la que destaca las direcciones IP del emisor y receptor, garantizando que los datos irán al destino correcto. Estas direcciones son usadas por los conmutadores de paquetes (*switches*) y enrutadores (*routers*) para decidir el camino a seguir por los datagramas, siempre asegurando que sea el más corto. Si el datagrama a enviar tiene mayor tamaño que el permitido, será dividido y enviado por otro camino si es necesario [Jeff, 2008].

Para determinar el destinatario IP utiliza tres campos: la dirección del equipo, la máscara de subred para establecer la parte de la dirección IP que se relaciona con la red y la pasarela predeterminada para decidir el equipo específico al que se va enviar el mensaje [Jeff, 2010].

La más popular y usada versión del protocolo es IPv4, aunque ya se están agotando las direcciones disponibles, para solucionarlo se propone IPv6 que utiliza direcciones de fuente y destino de 128 *bits*. IP no provee ningún mecanismo para determinar si un paquete llegó a su destino, solo proporciona seguridad de sus cabeceras mediante sumas de comprobación (*checksums*), por lo que si se necesita fiabilidad debe ser proporcionada por los protocolos de la capa de transporte como TCP [Jeff, 2008].

Protocolo de control de transmisión

TCP es un protocolo orientado a la conexión. Permite multiplexar datos, es decir, la información proveniente de distintas fuentes puede circular simultáneamente en la misma línea. Monitorea el flujo de paquetes para evitar saturación en la red y coloca en el destinatario los datagramas en orden nuevamente cuando vienen del protocolo IP [Kio, 2008].

Para establecer la conexión entre dos aplicaciones debe seguirse el siguiente esquema: en primer lugar los puertos TCP deben estar abiertos, luego la aplicación receptora “escucha” y espera una conexión, esto se conoce como aplicación pasiva y por último la aplicación emisora

realiza un pedido de conexión, siendo esta una aplicación activa. TCP tiene la capacidad de controlar la velocidad de los datos emitiendo los mensajes de tamaño variable [Kio, 2008].

Con el uso de TCP se pueden establecer conexiones de forma segura, antes de comenzar el envío de paquetes se establece la comunicación en línea, es decir, en ambas direcciones, garantizando la recepción mutua de datos. La máquina que solicita la conexión se llama cliente y quien la acepta servidor, por lo que usa la arquitectura conocida como cliente-servidor[Kio, 2008].

La suite TCP/IP tiene grandes ventajas, entre las que destacan su bajo costo de implementación y la necesidad de pocos recursos de red, por lo que es usado no solo en Internet sino también en pequeñas *intranets*. Se utiliza tanto en redes empresariales como en universitarias, generalmente cuando se utilizan muchos enrutadores y conexiones [Marco Antonio Manayay, 2008].

Protocolo de datagrama de usuario

UDP es un protocolo no orientado a la conexión, que provoca que los mensajes se dupliquen o lleguen desordenados al destino, pudiendo hasta no llegar. Utiliza el protocolo IP para transportar los datagramas [Vialfa, 2008].

Su formato contiene el número de puerto UDP de la máquina origen y destino, la longitud del mensaje incluyendo la cabecera, la suma de comprobación de errores y los datos. Se utiliza principalmente para el envío de imágenes en la red que no necesita fiabilidad, por ejemplo en los *chats*.

1.2.2. Arquitecturas de red

Arquitectura cliente-servidor

La arquitectura cliente-servidor es un modelo para el desarrollo de sistemas informáticos donde las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. El cliente es quien inicia el diálogo, solicita los recursos y contiene sólo lo particular de cada usuario, mientras que el servidor es quien responde a la solicitud y contiene la información que debe ser compartida. Generalmente el cliente y el servidor están ubicados en distintas terminales.

Los clientes tienen un papel activo en la comunicación, por lo general puede estar conectado a varios servidores a la vez, normalmente interactúan directamente con los usuarios finales mediante la interfaz gráfica de usuario, captura y valida los datos de entrada. Los servidores

desempeñan un papel pasivo en la comunicación, por lo general aceptan conexiones desde varios clientes aunque en muchas ocasiones el número máximo de conexiones está limitado, no es frecuente que interactúen directamente con los usuarios[Jose Guillermo Valle, 2005].

Esta arquitectura potencia el aprovechamiento de las computadoras para realizar servicios de presentación y ocasionalmente acceso a datos locales, liberando así al servidor de tareas sencillas que pueden realizar otras máquinas rápidamente. Entre las ventajas que presenta se pueden destacar: la centralización del control, que evita que un proceso no autorizado o defectuoso pueda dañar el sistema y facilita la actualización de la información, la escalabilidad, lo significa que se puede aumentar y mejorar la capacidad del cliente y el servidor en cualquier momento, el mantenimiento es fácil por la distribución que tienen las funciones y responsabilidades entre varias terminales independientes. También presenta algunos inconvenientes, como: la congestión del tráfico en la red cuando muchos clientes envían solicitudes a un mismo servidor, el software y hardware del servidor generalmente es específico para sus funciones, lo que puede aumentar el costo, el cliente no dispone de los recursos que puedan existir en el servidor. Además se puede presentar una alta complejidad tecnológica al tener que integrar varios recursos y tecnologías [Jose Guillermo Valle, 2005].

Arquitectura punto a punto

Todo funciona sin clientes ni servidores fijos, sino una serie de nodos que comparten entre sí la información. Permiten el intercambio directo de datos en cualquier formato. Administran y optimizan el uso de ancho de banda, obteniendo más rendimiento en las conexiones y transferencias [de la Luz, 2011].

Existen dos esquemas principales para esta arquitectura: Puro e Híbrido. En el primero los nodos pueden cumplir tres funciones esenciales: la de servidor, la de cliente y como enrutador. En el segundo se implementa una arquitectura donde un nodo puede realizar una consulta a un servidor, con el objetivo de saber dónde están los otros nodos en la red. Una vez hecha la consulta, el nodo podrá establecer la conexión directa con otro nodo para compartir su información. La aplicación P2P debe informar a este servidor de su conexión y desconexión para mantener la integridad del servicio [Antonio Bonilla Egido, 2001].

Entre las principales ventajas que presenta este modelo está que al añadir un nodo a la red no es necesario reestructurarla nuevamente. Se minimiza la congestión debido a que las conexiones se realizan punto a punto y la organización de los nodos es independiente de si un

nodo está conectado o no, permitiendo conectividad variable y escalabilidad. No obstante, un inconveniente es su estructura descentralizada, provocando una gestión más compleja y costosa. Además, la baja calidad en la prevención de ataques la hace vulnerable a la perdida de datos y a los virus [Antonio Bonilla Egido, 2001].

1.2.3. Formato de los datos

A través de la red se puede enviar todo tipo de información, ya sea texto plano o codificado, videos, imágenes, etc. Las técnicas para lograr la integridad de estos datos en la red se han ido perfeccionando con el pasar de los años, creándose para ello diversos formatos de ficheros para su envío.

Para crear un SRV en ocasiones es necesario enviar los datos que se capturan de los distintos dispositivos hacia una computadora que los procese. Dicha información puede ser trasmitida de diferentes formas, que el servidor interpretará y manipulará según lo establecido, siendo más común compactar los ficheros y enviarlos por tramos. A continuación se describen los formatos más empleados en este proceso.

***Rich Text Format* (RTF)**

Pretende que un documento creado en un sistema determinado pueda consultarse y manipularse en cualquier otro contexto diferente. Esto se logra porque internamente un documento RTF se almacena como texto plano, en el que se incluye toda la información de texto en sí como una serie de metadatos que informan del formato del documento y del resto de las propiedades adicionales de este formato. Está formado por 4 elementos básicos: texto sin formato, palabras de control (claves), símbolos de control y grupos. Codifica diferentes tipos de documentos y formatos de texto como PDF, Word, Excel, etc. Se puede utilizar para el envío de datos a través de correo electrónico. Es compatible con varias fuentes, márgenes y otras funciones similares, sin embargo no lo es con función macro, protegiendo el sistema que lo utilice de los virus informáticos [For, 2010].

***eXtensible Markup Language* (XML)**

Es una forma estándar de representar información estructurada fácil de interpretar por los ordenadores y de escribir y entender por los humanos. Utiliza una estructura de árbol y de etiquetas similares a *HyperText Markup Language* (HTML). En este formato los datos se organizan en elementos que a su vez contienen atributos [Act, 2007].

Permite elaborar una estructura propia en dependencia de los datos que se desean transmitir, eliminando complejidades en cuanto a la flexibilidad. Es muy usado en las aplicaciones de Internet, principalmente aquellas que deseen transmitir constantemente datos en la red, pues facilita la constante comunicación cliente-servidor.

***JavaScript Object Notation* (JSON)**

Es un formato simple para el intercambio de datos, soportado por muchos lenguajes de programación como ASP, C♯ y Java. Se puede utilizar como alternativa a la necesidad de XML en el intercambio de información vía AJAX, cuando el volumen de datos que se maneja excede lo razonable, por ejemplo, Yahoo y Google lo utilizan en sus clientes de correo electrónico para aligerar el peso de los documentos de intercambio [Suárez, 2008].

Permite representar estructuras de datos (arreglos) y objetos en forma de texto. La notación de objetos mediante JSON es una de las características principales de JavaScript y es un mecanismo definido en los fundamentos básicos del lenguaje. En los últimos años, se ha convertido en una alternativa al formato XML, ya que es más fácil de leer y escribir, además de ser más conciso. No obstante, XML es superior técnicamente porque es un lenguaje de marcado, mientras que JSON es simplemente un formato para intercambiar datos [Pérez, 2008].

1.3. Aplicaciones relacionadas con la comunicación de dispositivos y el envío de datos por red

En la actualidad se han desarrollado diversas aplicaciones que utilizan los dispositivos periféricos como fuente de obtención de datos respecto a la posición de un individuo en el mundo real. Estos datos se captan desde diferentes terminales y generalmente son manipulados y gestionados en una sola, creando un modelo cliente-servidor. Para la trasmisión al servidor se utilizan protocolos de red, principalmente la suite TCP/IP. A continuación se muestran algunos ejemplos de estos sistemas.

1.3.1. Sistema de Realidad Virtual basado en múltiples pantallas distribuidas

Es un sistema basado en un ambiente CAVE, para lograrlo se usan 5 computadoras. La computadora servidor es la responsable de recibir las señales de entrada provenientes de los dispositivos y distribuirlas hacia las otras cuatro, que son clientes utilizados para la representación

de imágenes que rodean al usuario. Una de estas 4 es también encargada de sincronizar las proyecciones de las demás. Este sistema se puede dividir en tres partes: entradas, cálculo y salidas. La primera se refiere a la captura de datos provenientes de los dispositivos de localización, la segunda al manejo y gestión de estos datos y la última a la inmersión lograda con lo visualizado en las pantallas.

La distribución de hardware es la siguiente: las 5 computadoras descritas anteriormente, proyectores estereoscópicos, dispositivos para sincronizar la señal RGB de las computadoras; periféricos y gafas estereoscópicas; dispositivos de seguimiento de puntos; varillas 3D y las pantallas de proyección (véase Figura 1.9).

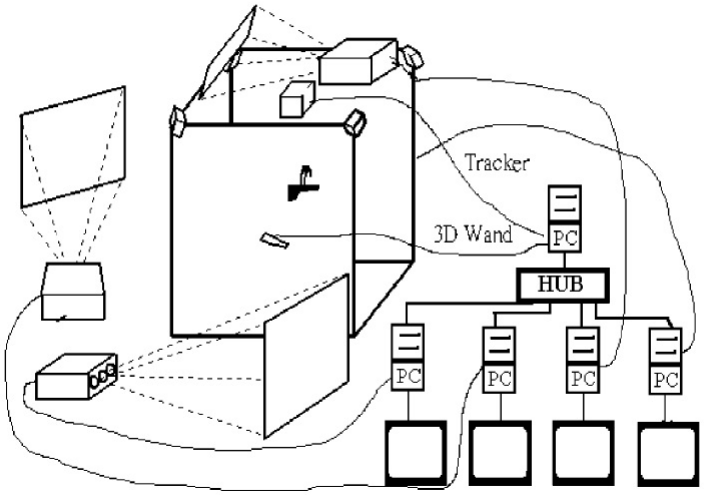

Figura 1.9: Sistema basado en pantallas distribuidas

Las principales funciones del sistema son: *viewing frustum computation*, utilizada para proyectar el mundo virtual en las pantallas, esta vista tiene que ser actualizada constantemente de acuerdo con la posición visual del usuario y necesita dos *viewing frustum* por cada pantalla para lograr visión estereoscópica; proyección estéreo, para lo que se necesita una visión distinta para cada ojo, principalmente en los bordes y esquinas para evitar obstrucción de objetos en el ambiente; sincronización, debido a que los resultados se muestran en diferentes pantallas que son manejadas por distintas computadoras y el sistema se desarrolla en tiempo real.

El sistema de red es lo que permite la sincronización y comunicación entre el servidor y los clientes. Los datos de entrada y localización son manejados en el servidor y enviados a los

clientes a través del protocolo TCP/IP.

Más información en [Li, 2001]

1.3.2. Gestión de estados dinámicos compartidos para ambientes virtuales interactivos distribuidos

El propósito de un ambiente *Distributed Interactive Virtual Environment* (DIVE) es lograr que el usuario perciba que está interactuando con un sistema real, para esto se ha desarrollado un *framework* basado en red llamado NAVERLib.

NAVERLib es un sistema orientado a objetos que proporciona módulos para interacciones, interfaces y componentes virtuales que conforman un SRV. La gestión de estados se clasifica en tres variantes: entre componentes que se encuentran en la misma computadora, entre múltiples computadoras formando una pequeña red y entre varias terminales distribuidos en la red mundial (World Wide Web). Este estudio se centrará en la segunda variante.

Kist cAVE-like (KAVE) es una aplicación que utiliza NAVERLib. En cada nodo algunos componentes funcionan como servidores, actualizando y propagando los estados, esto quiere decir que si el componente que actúa como servidor está activo, cambia el valor del estado y propaga esta información a los demás nodos clientes, como muestra la Figura 1.10 (líneas azules). Si por el contrario, se encuentra en pasivo entonces recibirá los cambios ocurridos en otros sistemas de la red y los propagará al resto de los *hosts* (Figura 1.10, líneas rojas).

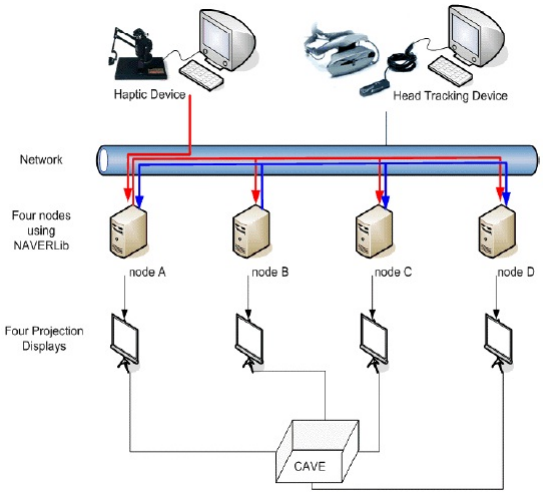

Figura 1.10: Ejemplo de sistema KAVE usando NAVERLib

Cada componente en DIVE debe actualizarse en el tiempo correcto para tratar de simular lo posible la realidad, por ejemplo, el ritmo de actualización de los dispositivos periféricos debe ser 60Hz, mientras que en los componentes de simulación física se recomienda 1KHz.

Más información en [Park, 2004]

1.3.3. Sistema de dispositivos de entrada virtuales en .Net

Consiste en un proyecto donde a partir de un conjunto de información del estado de algunos eventos de dispositivos periféricos, denominados señales de dispositivos, se puedan seleccionar para llevar a cabo tareas en los programas usuarios. Esto se hace con el objetivo de reproducir un comportamiento deseado a partir del manejo de los periféricos.

El sistema se divide en tres capas: adquisición, procesamiento y presentación de datos. La primera es la comunicación directa con los dispositivos y corresponde al módulo de Dispositivo Físico. En la segunda el servidor recibe los datos proporcionados por el módulo de dispositivos, los gestiona y reenvía a los programas clientes, donde se definen los dispositivos virtuales. En la tercera es donde el dispositivo virtual varía sus propiedades, generando las acciones preestablecidas y actuando en consecuencia.

Adoptando la arquitectura cliente-servidor, se tienen dos tipos de clientes: los dispositivos físicos y los virtuales. Los primeros suministran la información real de los periféricos, mientras los últimos cambian su estado en función de estos datos obtenidos a través del servidor.

El envío de mensajes se hace a través de sockets TCP/IP codificándose en formato XML. Estos mensajes contienen información de cuál dispositivo físico representa, la señal que ha cambiado su valor y algún comando a ejecutar en el servidor.

El sistema se puede emplear como escritorio remoto (Figura 1.11), generando los eventos del *mouse* y teclado de forma remota. También para el control de objetos 3D a través del *wiimote*, *InertiaCube* y demás dispositivos (Figura 1.12). Permite controlar el *mouse* mediante la *webcam* y *Nunchuk*, así como crear una pizarra virtual (Figura 1.13).

Figura 1.11: Escritorio remoto

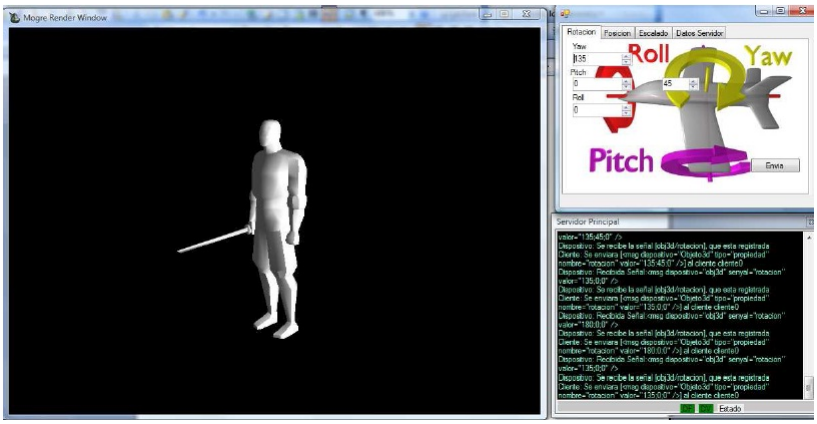

Figura 1.12: Control de objetos 3D

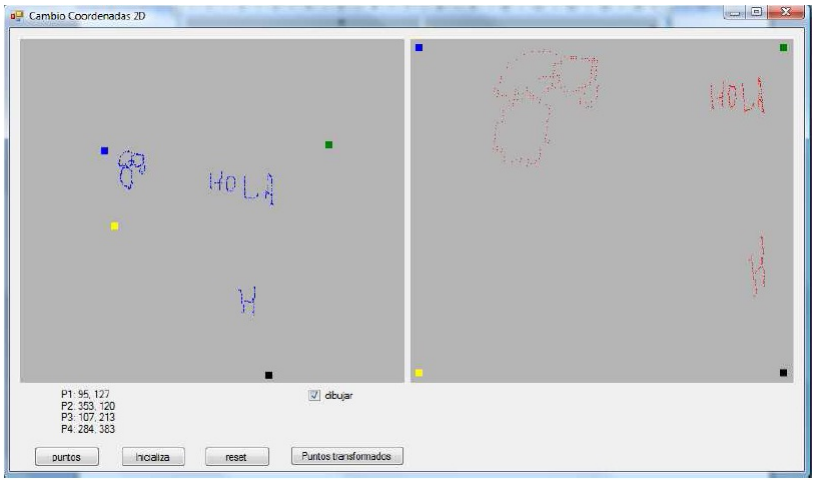

Figura 1.13: Pizarra virtual

Más información en [Blázquez, 2010]

1.4. Herramientas y tecnologías a usar

Son detallados a continuación la metodología de desarrollo de software, así como el conjunto de herramientas y lenguajes utilizados para el desarrollo de la aplicación.

Visual Studio .Net

Se utilizará *Visual Studio 2008* como *Integrated Development Environment* (IDE) de desarrollo por ser una herramienta ampliamente estudiada en el proyecto. Permite construir fácilmente aplicaciones sencillas y además ofrece una amplia gama de lenguajes de programación, editores de código y posibilidad de diseños visuales. Además es compatible con las bibliotecas necesarias para gestionar los dispositivos a utilizar.

C Sharp

Es un lenguaje de programación orientado a objetos desarrollado por Microsoft. Es simple, moderno y uno de los más usados en la actualidad. Se utilizará principalmente por ser un requisito del cliente, pero además posee una amplia documentación para su uso y permite la creación de interfaces sencillas y vistosas.

UML

Unified Modeling Language (UML) es un lenguaje de modelado muy conocido y utilizado

en la actualidad para describir métodos o procesos. Permite construir gráficos para visualizar, especificar y documentar un sistema. Entre los principales diagramas que permite modelar están: Diagrama de Clases, Diagrama de Casos de Uso y Diagrama de Secuencia.

***Rational Unified Process* (RUP)**

Se utiliza la metodología RUP por permitir hacer énfasis en los procesos de análisis y diseño, además de permitir el modelamiento de todo el proceso de desarrollo del software.

Visual Paradigm

Es una herramienta UML utilizada para desarrollar el ciclo de vida completo del software, es decir, análisis, diseño, construcción, prueba y despliegue. Consigue el modelamiento con calidad de aplicaciones, a un menor costo y mejores.

Capítulo 2

Solución Propuesta

En este capítulo se presenta la solución propuesta del sistema. Se esbozan inicialmente los elementos a tener en cuenta de cada uno de los dispositivos manejados en la solución. Se describe cómo será el proceso de transmisión de datos a través de la red y el formato que tendrán los documentos a enviar. Finalmente, se presenta un esbozo del funcionamiento lógico de la aplicación.

2.1. Dispositivos a manejar

Para el desarrollo de la solución final se ha decidido manejar el comportamiento de cuatro dispositivos de los existentes y descritos con anterioridad en el Capítulo 1, los cuales son: *mouse*, teclado, *wiimote* y *webcam*. Se han elegido estos específicamente por ser algunos de los más usados en las aplicaciones gráficas y por ser con los que cuenta el proyecto en estos momentos. Pero se ha diseñado una estructura flexible con el objetivo de poder añadir más dispositivos si se requiere.

A continuación se describe la información manipulada de cada uno de ellos.

2.1.1. *Mouse*

El *mouse* es, posiblemente, el dispositivo de entrada más usado en las aplicaciones cuando funcionan en una computadora convencional, dada su facilidad de manipulación y su bajo costo de adquisición. Para la implementación de este módulo se han tenido en cuenta sus principales propiedades. En la Tabla 2.1.1 se describe la interpretación que se hace de sus propiedades.

Mouse	
Propiedad	Descripción
click_izq	Se capturan a partir de los eventos generados nativamente por la plataforma de desarrollo. Los tipos de eventos pueden ser: Released, Pressed o None
click_der	
dbclick_izq	
scroll	
posición	Es la posición en enteros que tiene el *mouse* cuando está sobre el formulario. Corresponden a los ejes coordenados (x,y).

Tabla 2.1: Propiedades manejadas por el *mouse*

2.1.2. Teclado

El teclado es otro de los dispositivos de entrada más usado, pues generalmente viene en conjunto con la computadora, es muy sencillo y al igual que el *mouse* tiene un muy bajo costo de adquisición. A continuación se representan las propiedades que se tienen en cuenta para la implementación (Ver Tabla 2.1.2)

Teclado	
Propiedad	Descripción
tecla	Se captura a partir de la tecla que genera el evento. Tecla se refiere al nombre de la misma y valor al evento que ocurrió, pudiendo ser de tipo: Pressed o Released.
valor	

Tabla 2.2: Propiedades manejadas por el teclado.

2.1.3. *Wiimote*

El *wiimote*, perteneciente a la videoconsola Wii, en los últimos tiempos se ha convertido en un dispositivo muy empleado en SRV. La versatilidad de funciones que posee y la combinación de estas pueden convertirlo en un poderoso instrumento para determinar la posición y orientación del usuario que lo utiliza. A continuación se describen sus propiedades (Ver Tabla 2.1.3).

Para capturar la información de este dispositivo se utilizó la biblioteca WiimoteLib (Véase Figura 2.1), desarrollada por Brian Peek en C♯, debido a las posibilidades que brinda para la manipulación y control del mando Wii.

Wiimote	
Propiedad	Descripción
A	Hacen referencia a los botones que tiene el mando. Se capturan a partir del evento que generan estos y pueden ser de tipo: Pressed o Released.
B(gatillo trasero)	
Minus	
Home	
Plus	
One	
Two	
Up	Se refiere a las cuatro direcciones que tiene la cruceta del mando. El evento puede ser de tipo: Pressed o Released.
Down	
Left	
Right	
Led1	Recogen información del estado del mando, el evento puede ser de tipo Off u On y puede ser manejado por el usuario al encenderlos o apagarlos.
Led2	
Led3	
Led4	
Rumble	Puede ser activado por el usuario, siendo el evento de tipo On u Off.
A_X	Se refiere al acelerómetro que tiene el mando, sus valores varían entre 0 y 255 para cada uno de los ejes coordenados.
A_Y	
A_Z	

Tabla 2.3: Propiedades manejadas por el *wiimote*.

La biblioteca realiza la detección de cambios de estado del mando a través del método principal `wm_WiimoteChange` y brinda la posibilidad de especificar el tipo de reporte que se quiere obtener, por ejemplo si sólo se quiere saber el estado de los botones se debe optar por seleccionar `InputReport.Buttons`, o si se desea conocer el estado además del acelerómetro y el infrarrojo se debe elegir `InputReport.IRAccel`.

Aunque sólo se hace referencia a las propiedades del mando en sí, es válido destacar que la biblioteca escogida también puede manejar los eventos que son generados por las distintas extensiones que pueden ser conectadas al dispositivo, como es el *Nunchuk*, siendo esta característica muy importante para futuras aplicaciones.

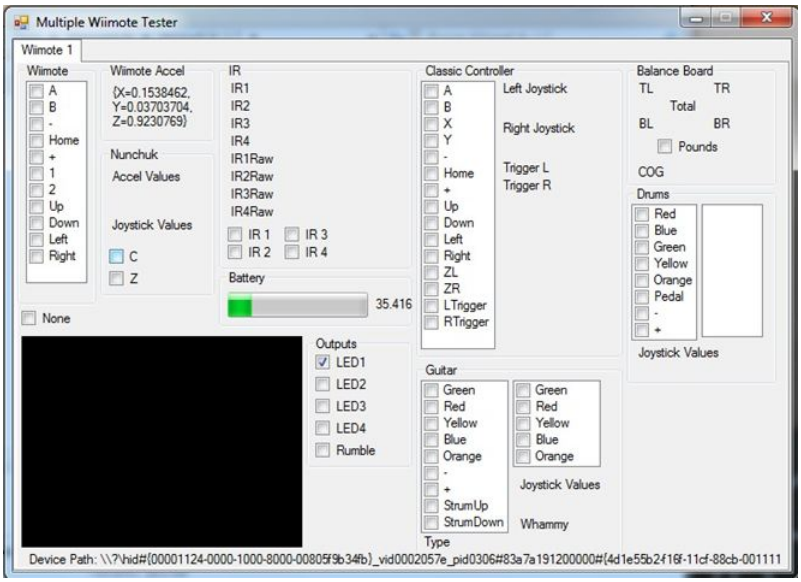

Figura 2.1: Aplicación realizada por el autor de la biblioteca *WiimoteLib* para probar su funcionamiento

2.1.4. *Webcam*

El "reconocimiento" de la información proporcionada por la *webcam* lo realizan las propias personas, no la computadora, por lo que si se pretende utilizar como elemento de entrada, es necesario el empleo de alguna biblioteca que permita una clasificación o procesado de imágenes (y video), con el fin de generar algo detectable por la computadora, y por tanto por el sistema. Se puede seguir el comportamiento de un objeto dentro de las imágenes, a través de puntos que lo identifiquen, utilizando para ello algoritmos de reconocimiento. También se puede seguir algún patrón único o marca que se muestre en las imágenes.

En la Tabla 2.1.4 se expresan las propiedades que son obtenidas a partir del seguimiento de un patrón cuadrado asimétrico, formado por un cuadro negro que contiene otro blanco, siendo este último cuatro veces más pequeño que el primero (Ver Anexo 6). Para este procesamiento de imágenes se utiliza la biblioteca Tracking_Webcam, desarrollada por Liudmila Rodríguez, integrante del proyecto HDSVE.

Webcam	
Propiedad	Descripción
X	Se refiere a las coordenadas en los ejes (x, y, z) del patrón que es utilizado.
Y	
Z	

Tabla 2.4: Propiedades manejadas a través de la *webcam*.

2.2. Envío de datos por red

El envío de información a través de la red en tiempo real se ha convertido en un requisito a tener en cuenta en las aplicaciones de RV. A continuación se describen el protocolo y el formato de datos a usar para la implementación de esta funcionalidad en la propuesta de solución.

2.2.1. Protocolo de red

En el capítulo anterior se explicaron algunas de las arquitecturas de redes más empleadas, de ellas se decidió escoger la arquitectura cliente-servidor para esta aplicación. Esta arquitectura tiene la capacidad de separar los clientes del servidor, pudiendo este último atender peticiones de varios clientes a la vez.

Los datos que se envían en este módulo cliente-servidor corresponden a propiedades que tienen los distintos dispositivos que son manejados. Esta información se genera en el cliente a partir de cambios ocurridos en dichas propiedades y deben llegar al servidor lo más fiel posible, es por ello que se ha decidido emplear el protocolo TCP para lograr este objetivo.

TCP, como se ha descrito en el Capítulo 1, es orientado a la conexión, lo que permite la verificación del envío de datos entre dos aplicaciones. Para establecer la comunicación se hace uso de *sockets* TCP, que son los encargados de abrir y cerrar conexiones y manejar el envío y recepción de la información. Los *sockets* que usa TCP son conocidos como *Stream Sockets*, estos son los responsables de proveer un servicio de conexión confiable.

Para el trabajo con *sockets* se emplean las clases `TcpListener`, `TcpClient` y `Thread` de la plataforma *.Net*, siendo estas contenedora de métodos como `AcceptTcpClient()`, encargada de atender las peticiones de conexión en el servidor y `Connect()` para conectarse desde el cliente.

2.2.2. Formato de los datos

La información que se captura de los distintos dispositivos debe tener un formato estándar para cada uno de ellos, es por esto que se ha elegido XML, por ser muy sencillo y flexible, permitiendo crear el esquema necesario para la aplicación.

Se ha diseñado una estructura lo más común posible para todos los dispositivos. Como se ilustra en las Figuras 2.2, 2.3, 2.4 y 2.5 las etiquetas contenedoras son iguales para todos ellos, es decir, se mantienen sin variaciones las etiquetas <dispositivo> y <propiedades> en cada uno. En la primera de estas el atributo `tipo` se refiere al dispositivo que está siendo analizado en ese momento y `tiempo` almacena el instante en que se generó el cambio de evento.

A continuación se muestra la estructura que tiene para cada dispositivo en particular.

Mouse

En la Figura 2.2 se puede apreciar la estructura del documento creado para guardar la información generada por el *mouse* en cada uno de los cambios que sus propiedades puedan tener. Dentro de <propiedades> se encuentra <mensaje>, cuyos atributos `nombre` y `evento` tienen valores correspondientes a las columnas propiedades y tipo respectivamente de la Tabla 2.1.1. La etiqueta <posicion> almacena los ejes coordenados en que se mueve el *mouse* (x;y).

```
1  <?xml version="1.0" encoding="UTF8"?>
2  <dispositivo tipo="mouse" tiempo="5/23/2011 2:43:47 PM">
3    <propiedades>
4      <mensaje nombre="None" evento="None" />
5      <posicion X="235" Y="174" />
6    </propiedades>
7  </dispositivo>
```

Figura 2.2: Formato XML para el *mouse*

Teclado

El archivo correspondiente al teclado se muestra en la Figura 2.3, en el mismo sólo se almacena en la etiqueta <mensaje> el nombre de la `tecla` que generó el `evento` y de qué tipo fue este, ver también la Tabla 2.1.2.

```
<?xml version="1.0" encoding="UTF8"?>
<dispositivo tipo="teclado" tiempo="5/23/2011 3:17:31 PM">
  <propiedades>
    <mensaje tecla="Down" evento="Released" />
  </propiedades>
</dispositivo>
```

Figura 2.3: Formato XML para el teclado

Wiimote

Para el *wiimote* hay que tener en cuenta un mayor número de propiedades, por ser éste un dispositivo más complejo. Como se puede apreciar en la Figura 2.4, se recoge información de los botones, *leds*, acelerómetro y *rumble*. Los valores que pueden tener los atributos de cada una de las etiquetas mostradas se describen con más detalle en la Tabla 2.1.3.

```
<?xml version="1.0" encoding="UTF8"?>
<dispositivo tipo="wiimote" tiempo="5/4/2011 2:56:31 PM">
  <propiedades>
    <posicion X="131" Y="129" Z="155" />
    <Boton nombre="A" evento="Released" />
    <Boton nombre="B" evento="Released" />
    <Boton nombre="Left" evento="Released" />
    <Boton nombre="Right" evento="Released" />
    <Boton nombre="Up" evento="Released" />
    <Boton nombre="Down" evento="Released" />
    <Boton nombre="One" evento="Released" />
    <Boton nombre="Two" evento="Released" />
    <Boton nombre="Home" evento="Released" />
    <Boton nombre="Minus" evento="Released" />
    <Boton nombre="Plus" evento="Released" />
    <LED nombre="LED1" evento="Off" />
    <LED nombre="LED2" evento="Off" />
    <LED nombre="LED3" evento="Off" />
    <LED nombre="LED4" evento="Off" />
    <Rumble nombre="rumble" evento="Off" />
  </propiedades>
</dispositivo>
```

Figura 2.4: Formato XML para el *wiimote*

Webcam

Como se explicó anteriormente, con la *webcam* se hace un seguimiento del patrón, recogiendo la posición en los ejes coordenados x, y y z que tiene este con respecto a la pantalla. La Figura 2.5 muestra la estructura del documento XML para este dispositivo, (ver más información en la Tabla 2.1.4).

```
1  <?xml version="1.0" encoding="UTF8"?>
2  <dispositivo tipo="webcam" tiempo="5/23/2011 2:17:01 PM">
3    <propiedades>
4      <posicion X="-54" Y="-104" Z="645" />
5    </propiedades>
6  </dispositivo>
```

Figura 2.5: Formato XML para la *webcam*

2.3. Diseño lógico de solución

Para ilustrar el funcionamiento de la solución se ha diseñado un esquema donde se muestra un ejemplo de la distribución de dispositivos en diferentes computadoras. La Figura 2.6 muestra el servidor responsable de almacenar todos los datos provenientes de los clientes, estos son los encargados de capturar la información arrojada por los distintos dispositivos y guardar ésta en los documentos correspondiente a cada uno para hacer más fácil el envío.

Como se puede observar cada computadora cliente tiene asociado uno o varios dispositivos, cada una de ellas podrá procesar los datos arrojados por estos y registrar los cambios que ocurren en el fichero .xml. Luego se envía esta información al servidor que se encarga de interpretarla y utilizarla.

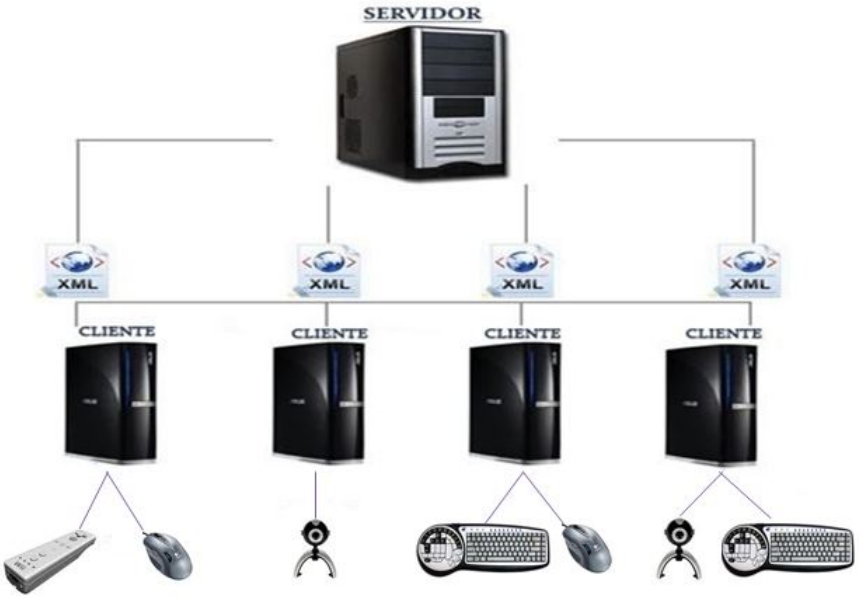

Figura 2.6: Diseño lógico de solución

Capítulo 3

Ingeniería del sistema

En este capítulo se documentan las etapas de análisis, diseño e implementación de la solución propuesta. Se define el actor que interactúa con el sistema y los principales requerimientos. Se describen los principales diagramas de estas etapas: Modelo de Dominio, Diagrama de Casos de Uso, Diagrama de Clases del Sistema, Diagramas de Secuencia para cada Caso de Uso y Diagrama de Componente.

3.1. Modelo de Dominio

Para lograr una mejor comprensión inicial de la concepción del sistema se hace un esbozo de los conceptos que lo conforman, describiendo su funcionamiento a través de clases conceptuales y las relaciones que se agrupan en el **Modelo de Dominio** (Véase Figura 3.1).

Entre los principales conceptos manejados se encuentran:

- **Dispositivos**: son los diferentes dispositivos que se manipulan en el sistema, entre los que se encuentran el **Mouse**, **Teclado**, **Webcam** y **Wiimote**, aunque es posible añadir algún otro si se requiere.
- **Cliente**: es el encargado de manipular toda la información correspondiente a los dispositivos y establecer la comunicación con el servidor.
- **Servidor**: es quien recibe los datos del cliente y procesa la información de los dispositivos.
- **Conexión**: a través de la conexión se envían y reciben datos entre el cliente y el servidor.

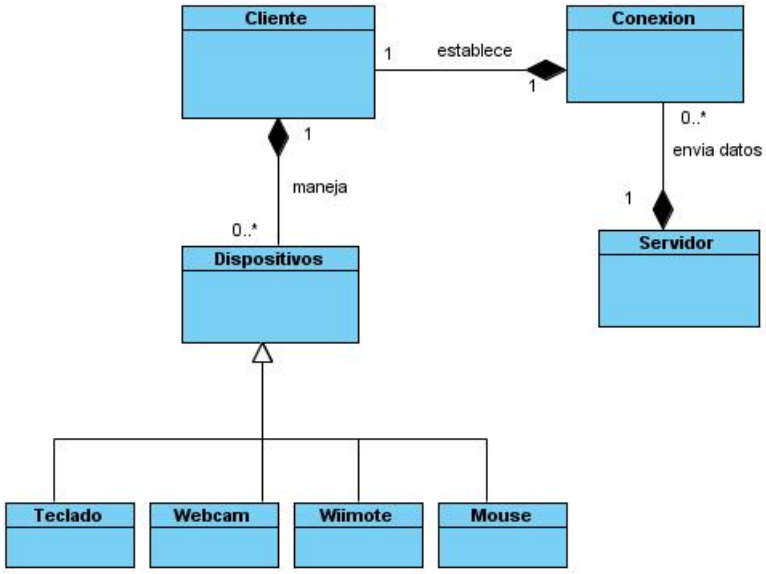

Figura 3.1: Modelo de Dominio

3.2. Definición de requisitos

Antes de comenzar el desarrollo de cualquier sistema es necesario tener claro lo que se quiere hacer, para ello se definen sus capacidades y propiedades. Al conjunto de condiciones o capacidades que debe tener un producto se denomina Requisitos Funcionales y el conjunto de cualidades o propiedades se conoce como Requisitos No Funcionales. A continuación se definen los requerimientos a tener en cuenta.

3.2.1. Requisitos Funcionales

- **RF1**: Establecer conexión
- **RF2**: Capturar información del teclado
- **RF3**: Capturar información del *mouse*
- **RF4**: Capturar información del *wiimote*
- **RF5**: Capturar información de la *webcam*
- **RF6**: Enviar información de dispositivos

- **RF7**: Permitir conexiones
- **RF8**: Procesar información del teclado
- **RF9**: Procesar información del *mouse*
- **RF10**: Procesar información del *wiimote*
- **RF11**: Procesar información de la *webcam*

3.2.2. Requisitos No Funcionales

- Hardware: la computadora donde se ejecutará debe ser Pentium IV o superior, con al menos 1Gb de RAM
- Software: el Sistema Operativo (SO) en el que debe ejecutarse es Windows XP o superior
- Implementación: debe utilizarse el IDE de desarrollo Visual Studio y C♯ como lenguaje de programación
- Modificabilidad: el sistema debe ser capaz de permitir modificaciones futuras, como agregar funcionalidades o manejo de algún otro dispositivo

3.3. Casos de Uso

Se han concebido los siguientes Caso de Uso (CU) que agrupan los requerimientos funcionales definidos:

- CU1: Permitir Conexión
- CU2: Establecer Conexión
- CU3: Gestionar información de dispositivos
- CU4: Procesar información.

3.3.1. Actor del sistema

Los actores son aquellos agentes que interactúan con el sistema. Para este caso se ha definido el descrito en la Tabla 3.3.1.

Nombre del actor	Descripción
GLSVe	Es la biblioteca beneficiada con las acciones a realizar (gestionar información de diferentes dispositivos y enviarla a una computadora central). Se encarga de utilizar la información obtenida.

Tabla 3.1: Actores del sistema

3.3.2. Diagrama de Casos de Uso

La interacción que existe entre los actores y los CU ya definidos se muestra a continuación en la Figura 3.2

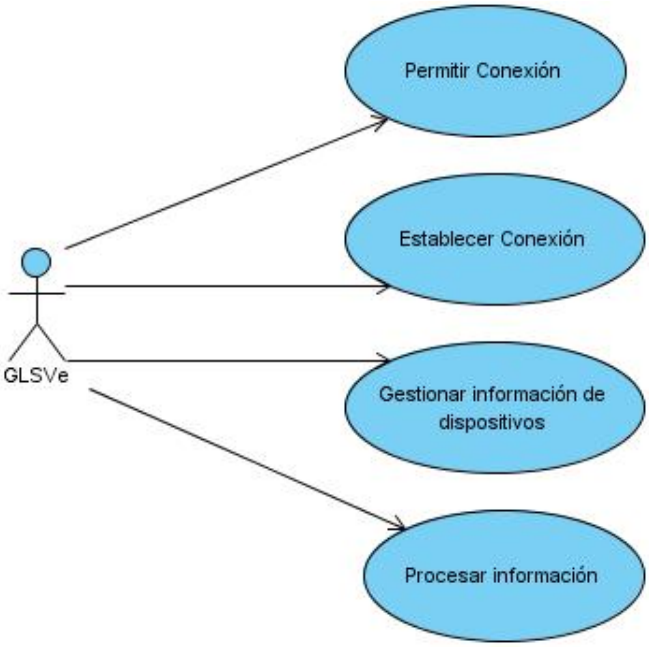

Figura 3.2: Diagrama de Casos de Uso

3.3.3. Descripción de Casos de Uso

A continuación se hace una descripción textual de los Casos de Uso definidos anteriormente.

Nombre del Caso de Uso	Permitir Conexión
Actores	GLSVe
Resumen	El caso de uso se inicia cuando se desea permitir conexiones. Se abre un puerto en el servidor por el cual se comunicarán los clientes.
Referencias	RF7
Poscondiciones	El sistema está listo para recibir datos de los clientes.
Flujo normal de los eventos	
Acción del actor	Respuesta del Sistema
1. Indica el puerto a abrir.	1.1 Si el puerto es correcto se abre el puerto indicado y se queda esperando a recibir conexiones y datos.
Flujo alterno	
Acción del actor	Respuesta del Sistema
	1.1 Si el puerto no es correcto o no está disponible se lanza el mensaje: "No se pudo abrir el puerto".

Tabla 3.2: Descripción del CU1: Permitir Conexión

Nombre del Caso de Uso	Establecer Conexión
Actores	GLSVe
Resumen	El caso de uso se inicia cuando es necesario establecer una conexión con el servidor para enviar información. Se indica la dirección IP y el puerto por el que se debe conectar al servidor.
Referencias	RF1
Precondiciones	Que el servidor haya abierto un puerto.
Poscondiciones	El cliente estará listo para enviar los datos.
Flujo normal de los eventos	
Acción del actor	Respuesta del Sistema
1. Se brindan los datos: ➢ Dirección IP ➢ Puerto	1.1 Se verifica que el IP y puerto sean correctos. 1.2 Se establece la conexión al servidor por el puerto y la dirección IP especificada.
Flujo alterno	
Acción del actor	Respuesta del Sistema
	1.1 Si el puerto o el IP son incorrectos se notifica al usuario a través del mensaje: "Asegúrese de haber escrito correctamente el puerto y la dirección IP del servidor". 1.2 Si el servidor no permite conexiones muestra el mensaje: "No se pudo conectar con el servidor. Asegúrese que se haya abierto el puerto en el servidor".

Tabla 3.3: Descripción del CU2: Establecer Conexión

<table>
<tr><td>Nombre del Caso de Uso</td><td colspan="2">Gestionar información de dispositivos</td></tr>
<tr><td>Actores</td><td colspan="2">GLSVe</td></tr>
<tr><td>Resumen</td><td colspan="2">El caso de uso se inicia cuando se desea capturar la información de algún dispositivo y enviarla al servidor. Se recogen los datos obtenidos del dispositivo y se actualiza la información en el fichero .xml correspondiente que es enviado al servidor.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF2, RF3, RF4, RF5 y RF6.</td></tr>
<tr><td>Precondiciones</td><td colspan="2">Que esté conectado el dispositivo

Que se haya establecido la conexión con el servidor.</td></tr>
<tr><td>Poscondiciones</td><td colspan="2">Se obtiene y envía la información del dispositivo.</td></tr>
<tr><td colspan="3">Flujo normal de los eventos</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. Se realiza la captura de información de los dispositivos:
➢ Teclado.
➢ Mouse.
➢ Wiimote.
➢ Webcam.</td><td colspan="2">1.1 El sistema, en dependencia de la acción solicitada, hace lo siguiente:
➢ Si decide obtener información del teclado, ir a la sección Capturar información del teclado.
➢ Si decide obtener información del mouse, ir a la sección Capturar información del mouse.
➢ Si decide obtener información del wiimote, ir a la sección Capturar información del wiimote.
➢ Si decide obtener información de la webcam, ir a la sección Capturar información de la webcam.
2. Actualiza la información del dispositivo seleccionado.
3. Escribe los datos capturados en el fichero correspondiente:
➢ Para el teclado: “teclado.xml”.
➢ Para el mouse: “mouse.xml”.
➢ Para el wiimote: “wiimote.xml”.
➢ Para la webcam: “webcam.xml”.
4. Envía la información del dispositivo.</td></tr>
</table>

Flujo alterno	
Acción del actor	Respuesta del sistema
	1.3 Si no se puede escribir el fichero, se muestra el mensaje: ➢ Para el teclado: “No se pudo escribir teclado.xml” ➢ Para el mouse: “No se pudo escribir mouse.xml” ➢ Para el wiimote: “No se pudo escribir wiimote.xml” ➢ Para la webcam: “No se pudo escribir webcam.xml” 1.4 Si no se ha establecido la conexión con el servidor se cancela el envío y se muestra el mensaje: "No se pudo enviar los datos. Asegúrese de estar conectado al servidor".

Sección “Capturar información del teclado”	
Acción del actor	Respuesta del Sistema
1. Modifica el estado del teclado.	1.1 Se captura, asociada al estado del teclado modificado, la siguiente información: ➢ Instante de tiempo en que ocurrió ➢ Evento (Pressed o Released) ➢ Tecla que cambió de estado

Sección “Capturar información del *mouse*”	
Acción del actor	Respuesta del Sistema
1. Modifica el estado del *mouse.*	1.1 Se captura, asociada al estado del *mouse* modificado, la siguiente información: ➢ Instante de tiempo en que ocurrió ➢ Evento correspondiente a los botones del *mouse* (Pressed, Released o None) ➢ Nombre del botón que incurrió en el evento (cl_Izq, cl_Der, dbcl_Izq, cl_Medio, Scroll o None) ➢ Posición en el eje x ➢ Posición en el eje y

Sección “Capturar información del *wiimote*”.	
Acción del actor	Respuesta del Sistema

1. Modifica el estado del *wiimote*.	1.1 Si el *wiimote* está conectado a la computadora, a través de la biblioteca "WiimoteLib", se obtiene la siguiente información: ➢ Instante de tiempo en que ocurrió ➢ Posición en el eje x ➢ Posición en el eje y ➢ Posición en el eje z ➢ Estado de los botones (*A, B, Left, Right, Up, Down, One, Two, Home, Minus, Plus*) ➢ Estado de los *leds* (LED1, LED2, LED3, LED4) ➢ Estado del *rumble* (vibrador)
Flujo alterno	
Acción del actor	Respuesta del sistema
	1.1 Si el wiimote no está conectado a la computadora se cancela la captura y se muestra el mensaje: "No se ha podido conectar el wiimote. Asegúrese de haberlo adicionado a los dispositivos bluetooth"
Sección "Capturar información de la *webcam*".	
Acción del actor	Respuesta del Sistema
1. Modifica información asociada a la *webcam*.	.1 Si hay alguna *webcam* conectada a la computadora, a través de la biblioteca "Tracking_Webcam" se obtiene la siguiente información: ➢ Instante de tiempo en que ocurrió ➢ Posición en el eje x ➢ Posición en el eje y ➢ Posición en el eje z
Flujo alterno	
Acción del actor	Respuesta del sistema
	2.1 Si no hay ninguna webcam conectada se cancela la captura y se muestra el mensaje: "Asegúrese de tener la cámara conectada"

Tabla 3.4: Descripción del CU3: Gestionar información de dispositivos

Nombre del Caso de Uso	Procesar información
Actores	GLSVe
Resumen	El caso de uso se inicia cuando al servidor llegan datos enviados desde los clientes. Estos datos son recibidos y procesados.
Referencias	RF8, RF9, RF10 y RF11.
Precondiciones	Que se hayan enviado datos desde algún cliente
Poscondiciones	Se obtienen los datos recibidos desde los clientes.
Flujo normal de los eventos	
Acción del actor	Respuesta del Sistema
1. Obtiene la información que envió el cliente.	1.1 Lee la información y en dependencia del dispositivo al que corresponda, hace lo siguiente: ➢ Si es el teclado, escribe la información en el fichero “teclado.xml”. ➢ Si es el *mouse*, escribe la información en el fichero “mouse.xml”. ➢ Si es el *wiimote*, escribe la información en el fichero “wiimote.xml”. ➢ Si es la *webcam*, escribe la información en el fichero “webcam.xml”. 1.2 Actualiza la información del dispositivo al cual corresponde el fichero que fue recibido.
Flujo alterno	
Acción del actor	Respuesta del sistema
	1.1 Si se leen un mensaje vacío significa que el cliente se desconectó y el servidor se quedará esperando por nuevas conexiones o procesando las ya existentes.

Tabla 3.5: Descripción del CU4: Procesar información

3.4. Diagrama de clases del diseño

Los diagramas de diseño ayudan a tener una mejor comprensión de la implementación del sistema a través de la representación de las clases que lo conforman, que pueden estar agrupadas en paquetes o subsistemas. A continuación se muestra el Diagrama de Clases del Diseño

compuesto por dos paquetes: Cliente y Servidor (Véase Figura 3.3), donde están agrupadas las distintas clases que los conforman.

Figura 3.3: Diagrama de paquetes del diseño

Dentro del paquete Cliente se encuentra la clase ControlCliente que se encarga de actualizar la información capturada de los dispositivos y escribirla en el fichero .xml correspondiente. ConexionCliente envía al servidor el fichero que se actualizó además de establecer y terminar la conexión con este, esta clase es la interfaz entre el cliente y el servidor. Dispositivo es la clase "padre" de Mouse, Teclado, Webcam y WiimoteC, cada una perteneciente a los dispositivos que se gestionan (Ver Figura 3.4). Estas últimas clases también se encuentran en el paquete Servidor, con igual estructura (Ver Figura 3.5 para más detalles).

El paquete Servidor presenta una estructura similar al Cliente, como se mencionó antes contiene la clase Dispositivo y sus "hijas". También cuenta con la clase ConexionServidor que es la encargada de permitir conexiones de forma asíncrona a la aplicación y leer los datos que envían los clientes. Además está la clase ControlServidor cuya función principal es actualizar el estado de los dispositivos a través de los ficheros .xml (Ver Figura 3.5).

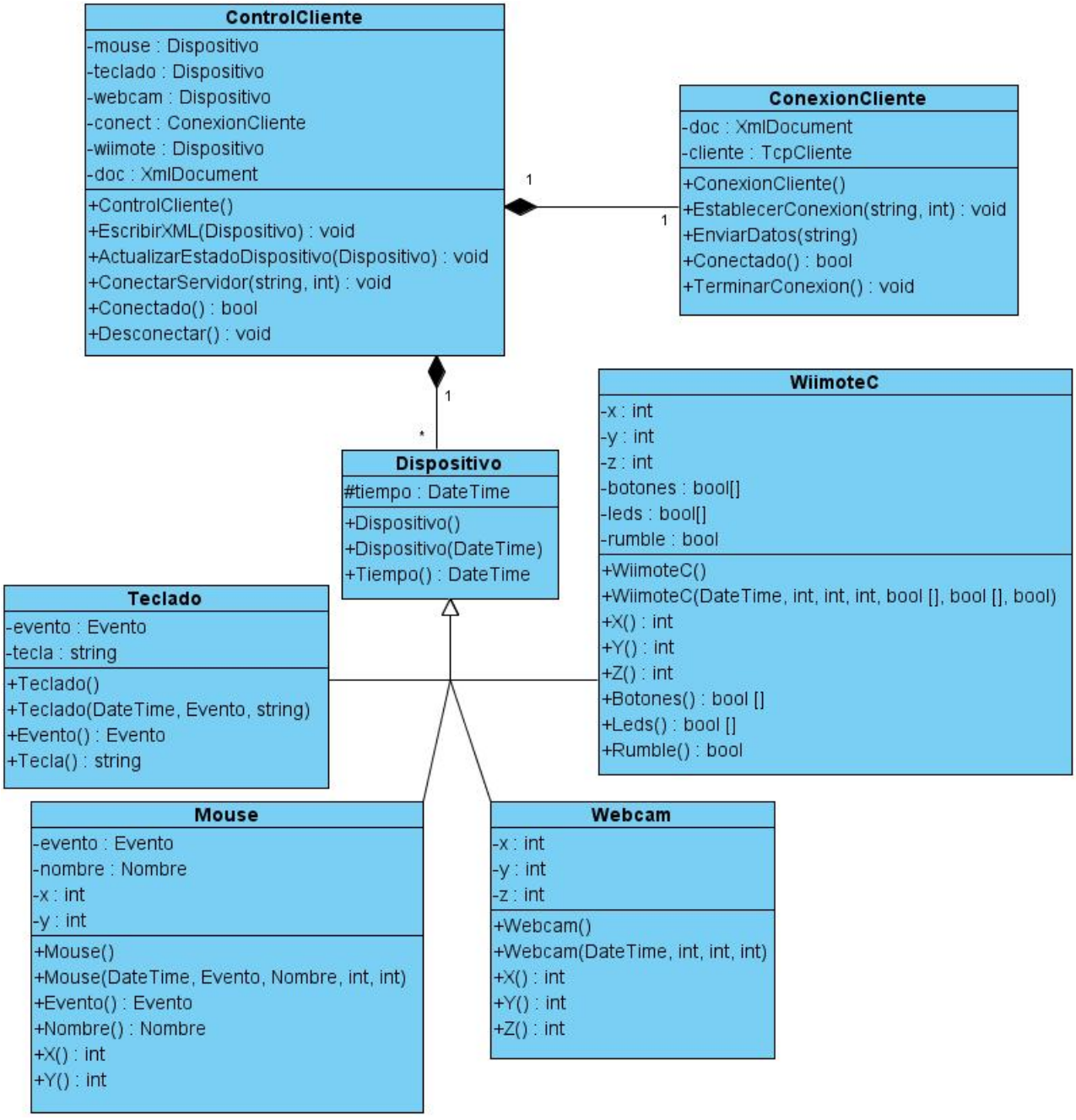

Figura 3.4: Paquete Cliente

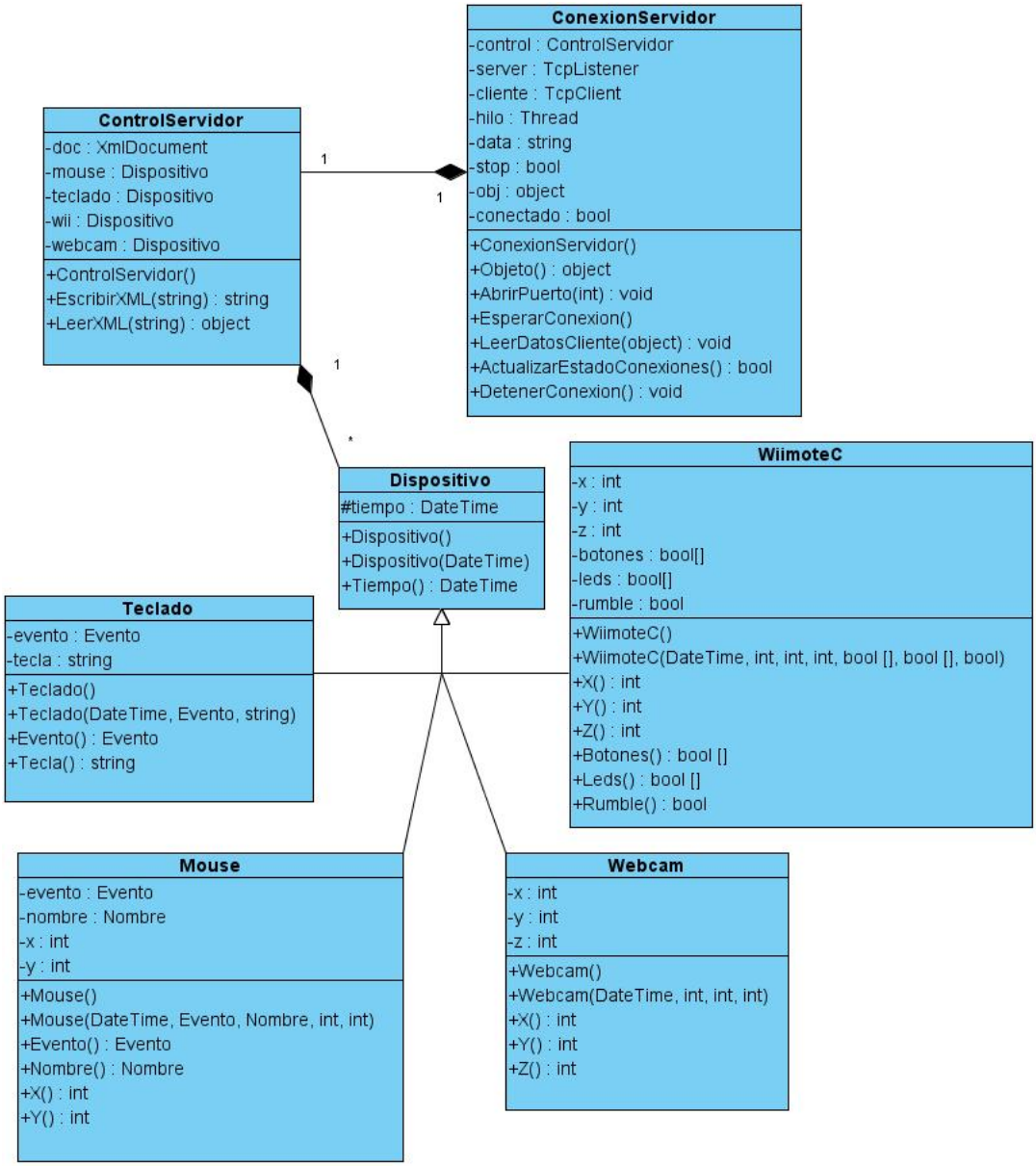

Figura 3.5: Paquete Servidor

3.5. Patrones de diseño

Para crear un lenguaje común y entendimiento entre los diseñadores y desarrolladores surgieron los patrones de diseño. Un patrón es una información que captura la estructura esencial y la perspicacia de una familia de soluciones probadas con éxito para un problema repetitivo que surge en un cierto contexto y sistema. Existen dos grandes grupos de patrones de diseño:

los *General Responsibility Assignment Software Patterns* (GRASP) o Patrones Generales de Software para Asignar Responsabilidades y los *Gang of Four* (GoF), conocido como la Banda de los Cuatro [Valls, 2009]. Cada uno de ellos define distintos patrones que son muy comunes en la realización de aplicaciones software hoy día. A continuación se resumen los usados en esta solución.

Patrones GRASP[Saavedra, 2007]

- Creador: En las clases ControlCliente y ControlServidor se establecen instancias de la clase Dispositivo, creando objetos de tipo Mouse, Teclado, Webcam y WiimoteC para la manipulación de estos

- Alta cohesión y bajo acoplamiento: Se le asigna a cada clase la responsabilidad que le corresponde de acuerdo a sus funciones, por ejemplo, las clases ConexionCliente y ConexionServidor son las únicas que establecen comunicación entre el cliente y el servidor y a través de éstas es que se realiza el envío y recepción de los datos, evitando dependencias en otras clases que hubieran podido realizar alguna de estas funciones

- Controlador: Como sus nombres lo indican las clases ControlCliente y ControlServidor son las las encargadas de controlar los eventos que ocurran, representando al sistema global

Patrones GoF

- Estado: La gestión de un dispositivo requiere permitir al objeto que lo identifique modificar su comportamiento en la medida que su estado va cambiando, principalmente cuando el comportamiento de este objeto depende de su estado en tiempo real. En este caso los objetos que representan a los distintos dispositivos manipulados modifican constantemente, en tiempo de ejecución, su estado y en dependencia de este asumen un comportamiento

- Solitario: Para la relación entre las clases se han creado objetos únicos que son los que permiten un acceso global a las funciones de la clase que representan, por ejemplo, la clase ControlCliente sólo contiene una instancia de cada una de las clases que representan los dispositivos manipulados. Así mismo sucede con su homóloga en el servidor

3.6. Diagramas de interacción

Los diagramas de interacción permiten ilustrar paso a paso cómo es que se llevan a cabo los procesos internos del sistema. Se realizan para comprender la lógica establecida para la implementación y pueden ser de dos tipos: secuencia y colaboración. Generalmente se presentan independientes para cada CU.

3.6.1. Diagramas de secuencia por Casos de Uso

Los diagramas de secuencia muestran explícitamente la interacción que se establece entre las clases, representando los métodos que poseen y los mensajes que intercambian durante los procesos. A continuación se hace la descripción y representación del diagrama de secuencia para cada uno de los CU descritos anteriormente.

Como se puede observar en la Figura 3.6, correspondiente al diagrama de secuencia para el CU1: Permitir Conexión, éste comienza cuando se solicita a la clase ConexionServidor que abra el puerto definido a través del método `AbrirPuerto()`. Si este puerto es inaccesible se muestra un mensaje informándolo, y si el proceso no tuvo problemas se invoca el método `EsperarConexion()` y el servidor ya está listo para recibir datos del cliente, terminando así el proceso.

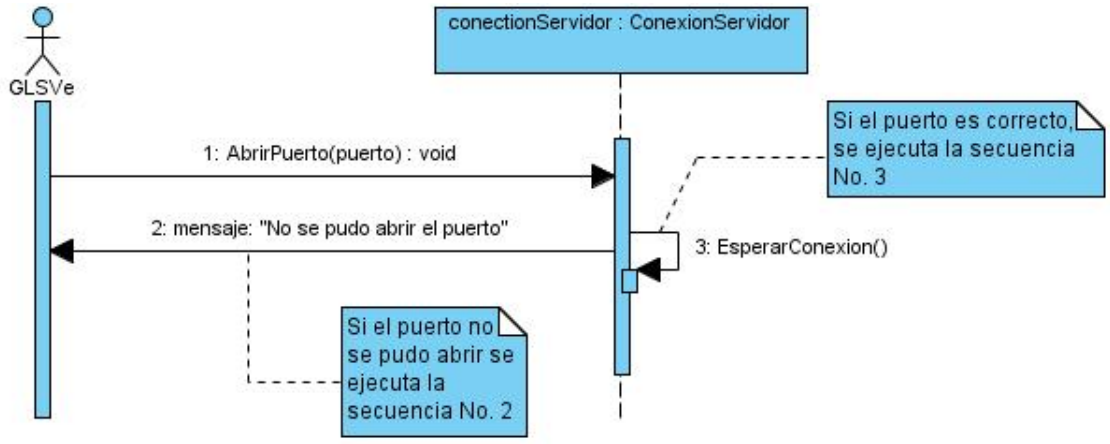

Figura 3.6: Diagrama de secuencia para el CU1: Permitir Conexión

La Figura 3.7 muestra el diagrama de secuencia para el CU2: Establecer Conexión. Como se puede observar el CU inicia cuando se especifica el puerto y la dirección IP del servidor,

luego la clase ControlCliente a través del método `ConectarServidor()`, invoca la funcionalidad `EstablecerConexion()`, que es la encargada de comunicarse con la aplicación del servidor. Si los datos del puerto o la dirección IP son incorrectos, se muestra un mensaje de error indicándolo, en caso contrario se conecta con el servidor y puede enviar información, de esta forma termina el proceso.

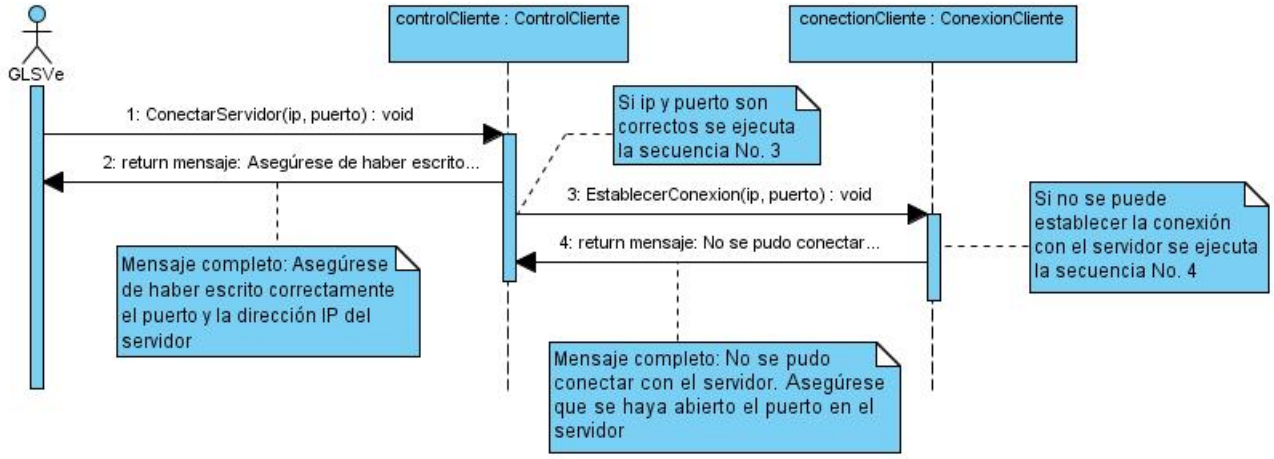

Figura 3.7: Diagrama de secuencia para el CU2: Establecer Conexión

El diagrama de secuencia para el CU3: Gestionar información de dispositivos (Ver Figura 3.8) muestra el proceso inicia cuando se ha elegido cuál dispositivo se va a capturar, obteniéndose la información de su estado. Luego se invocará la funcionalidad `ActualizarEstadoDispositivo()` e inmediatamente se pasa a escribir estos nuevos datos, a través del método `EscribirXML()` en el fichero .xml, que es enviado al servidor.

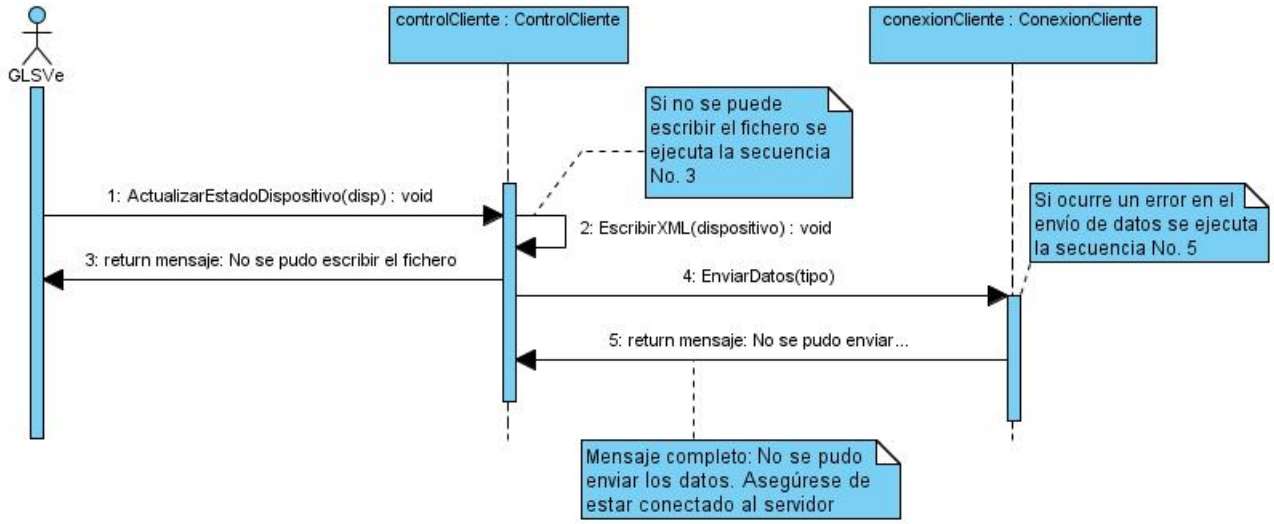

Figura 3.8: Diagrama de secuencia para el CU3: Capturar información dispositivo

Por último, como se puede observar en la Figura 3.9, el diagrama de secuencia para el CU4: Procesar Información, muestra cómo el servidor, una vez que el cliente ha enviado datos, los lee a través del método `LeerDatosCliente()` que se encuentra en la clase ConexionServidor, en este se invoca la funcionalidad `EscribirXML()` de la clase ControlServidor, que renueva la información del dispositivo en su fichero .xml para después obtener un objeto con los datos actualizados por medio de `LeerXML()`, terminando así el CU.

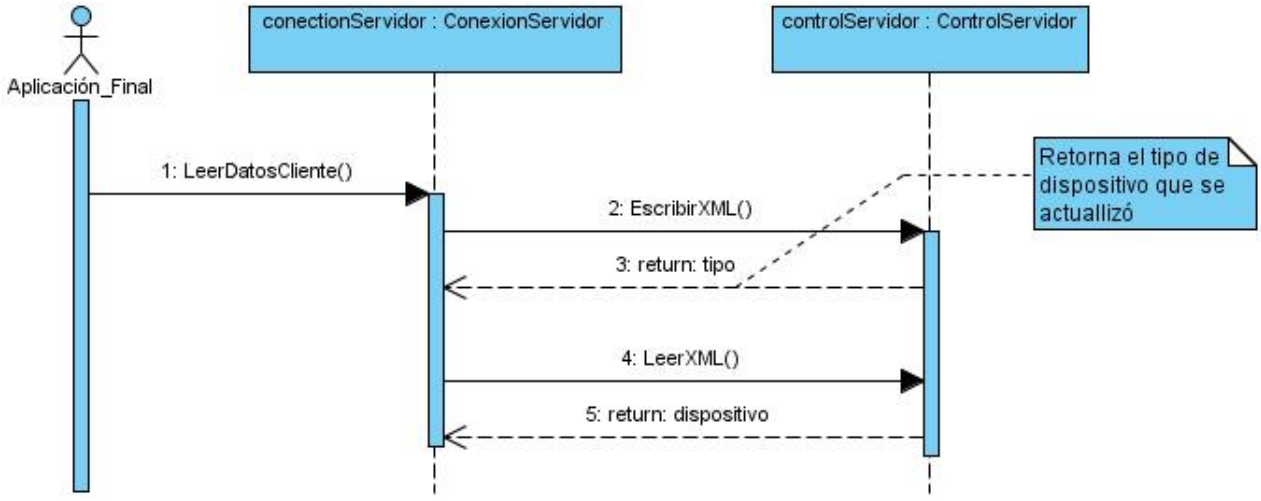

Figura 3.9: Diagrama de secuencia para el CU4: Procesar información

3.7. Diagramas de despliegue

El sistema realiza funcionalidades de red para su correcto funcionamiento, por tanto está compuesto por nodos clientes y un nodo servidor que se comunican a través de conexiones TCP/IP. En el diagrama de despliegue mostrado en la Figura 3.10 están representados los clientes que pueden estar conectados al servidor mediante el nodo PC Cliente y el servidor mediante PC Servidor. También se muestran los dispositivos periféricos con que se puede interactuar desde los clientes.

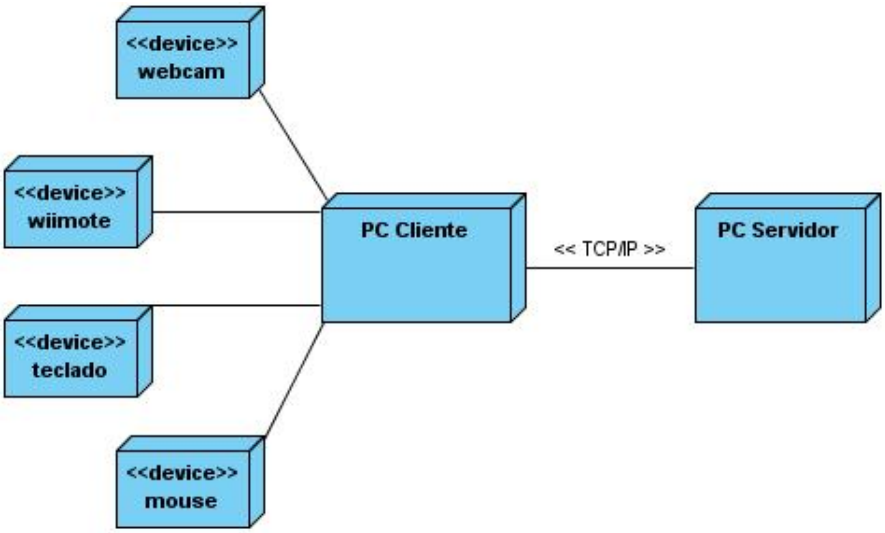

Figura 3.10: Diagrama de despliegue

3.8. Diagrama de componentes

Los diagramas de componentes muestran los elementos software que constituyen una parte reusable, las interfaces y sus relaciones, considerándose en ocasiones como un diagrama de clases a gran escala. Para su elaboración hay que tener en cuenta los requisitos relacionados con la facilidad de desarrollo, la gestión de software, reutilización, los lenguajes de programación y herramientas utilizadas [DDC, 2011].

Para la solución final de esta aplicación se han elaborado dos diagramas de componentes, debido a la división física que tiene la misma y la utilización indistinta de componentes en cada una de las aplicaciones, llámense Cliente y Servidor.

Para el Cliente (Ver Figura 4.4), los componentes mouse, teclado, wiimote y webcam corresponden a los ficheros .xml que se utilizan para almacenar la información de los periféricos que su nombre indica. La biblioteca Cliente.dll corresponde al paquete de clases descrito en el capítulo anterior en la Figura 3.4, y es quién se encarga de la gestión de los dispositivos y lo relacionado con la comunicación por red con el servidor.

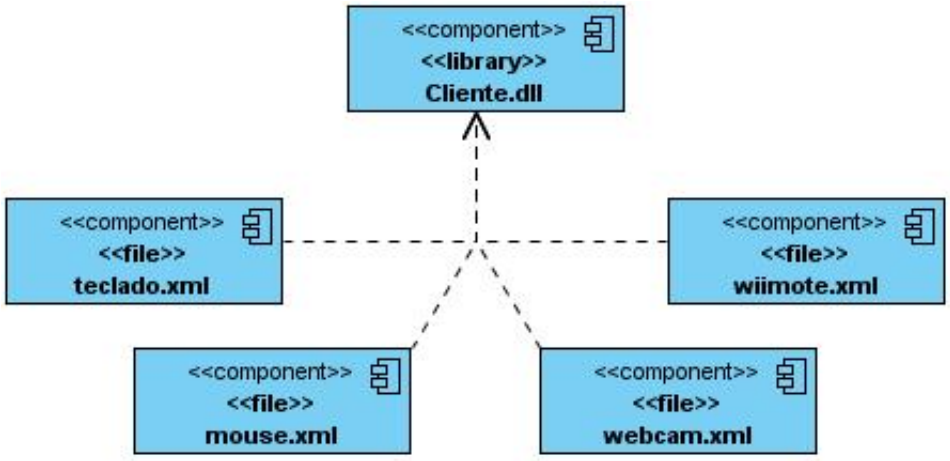

Figura 3.11: Diagrama de componentes para el cliente

En Servidor.dll se agrupan las clases descritas en el diagrama de diseño del paquete Servidor, este componente tiene asociados los ficheros .xml con igual estructura y propósito que en el cliente.(Ver Figura 4.5).

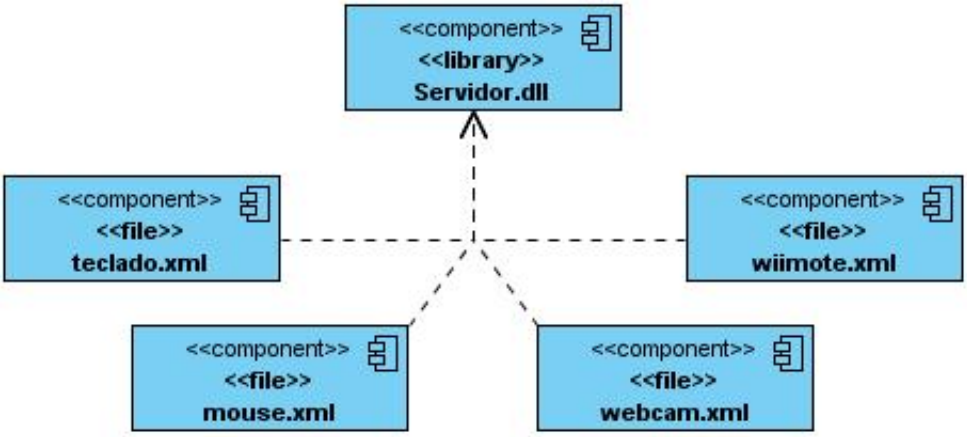

Figura 3.12: Diagrama de componentes para el cliente

Capítulo 4

Validación de la solución

En este capítulo se realiza la validación de las clases creadas a partir de su integración a la biblioteca GLSVe. Como se ha mencionado, la solución final consta de dos sistemas: Cliente y Servidor, y cada uno de ellos contiene un paquete de clases agrupados en una biblioteca que le permiten realizar las funcionalidades que dan sentido a su nombre. Para la validación en la parte del cliente se creó una aplicación con Visual Studio 2008 que ofrece una interfaz fácil de usar y comprender, la misma brinda las posibilidades de capturar los datos arrojados por los dispositivos y lo relacionado con la comunicación por red a través de la biblioteca creada. Para el servidor se emplea la biblioteca GLSVe con el objetivo de mostrar los cambios recibidos en cada uno de los dispositivos manipulados.

A continuación se muestra mediante imágenes y sus descripciones el uso de las bibliotecas Cliente.dll y Servidor.dll en las aplicaciones de validación.

4.1. Cliente

En la Figura 4.1 se muestra la interfaz creada para el cliente, en la misma se encuentra desplegado el menú principal y a través de este se puede establecer la conexión con el servidor, como aparece señalado en la imagen, y capturar la información de los dispositivos.

Una vez seleccionada la opción de conectarse al servidor, se deben especificar los datos requeridos. Al presionar el botón que se muestra en la Figura 4.2 la aplicación se conectará si los datos son correctos.

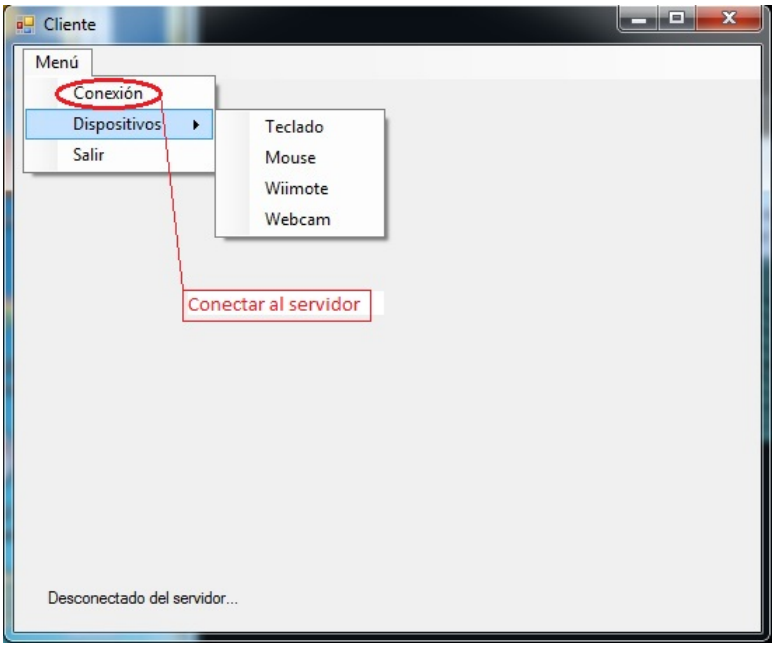

Figura 4.1: Interfaz Cliente: Menú principal

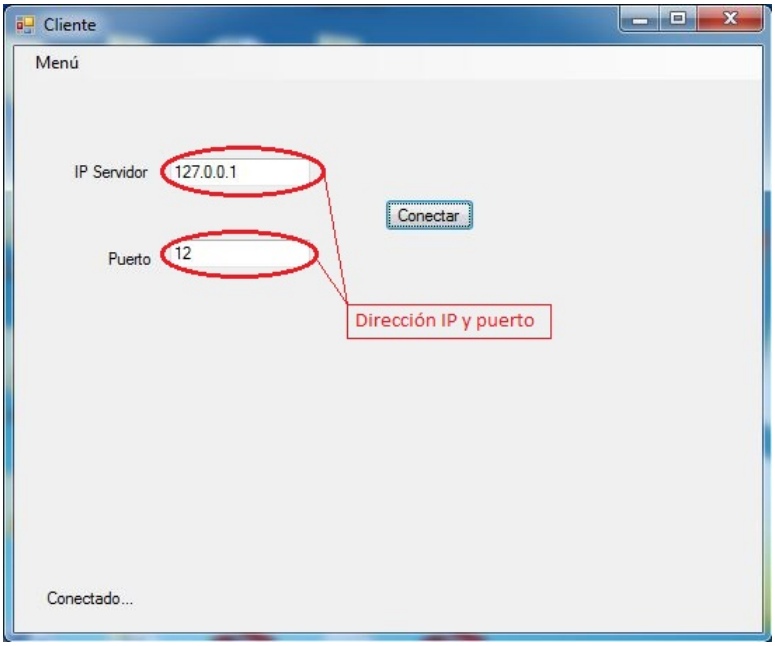

Figura 4.2: Interfaz Cliente: Conectado al servidor

Luego de haber establecido comunicación con el servidor, la aplicación cliente está lista para enviar la información que se capture de los dispositivos. Para ello, como se muestra en la Figura 4.1 se escoge cuál de los cuatro manejados es el deseado, e inmediatamente comienza el proceso de captura y envío de datos descrito con anterioridad. La Figura 4.3 muestran la obtención de datos del *mouse* donde se representa la posición de los ejes coordenados y opresión de uno de sus botones.

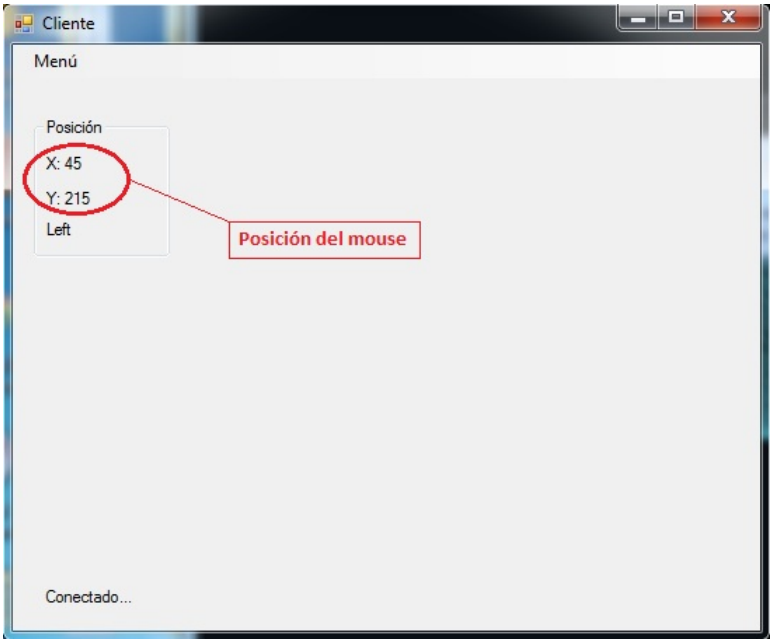

Figura 4.3: Interfaz Cliente: Captura del *mouse*

4.2. Servidor

Para el servidor se utiliza la biblioteca GLSVe en el dibujado de objetos en la interfaz. Se crean cuatro esferas de diferentes colores, la blanca recibirá los cambios correspondientes al dispositivo teclado, la verde es para el *mouse*, la rosada para el *wiimote* y la azul para la *webcam*. También se muestra un apartado donde se debe especificar el puerto a abrir y se muestra el estado actual de la conexión (Ver Figura 4.4).

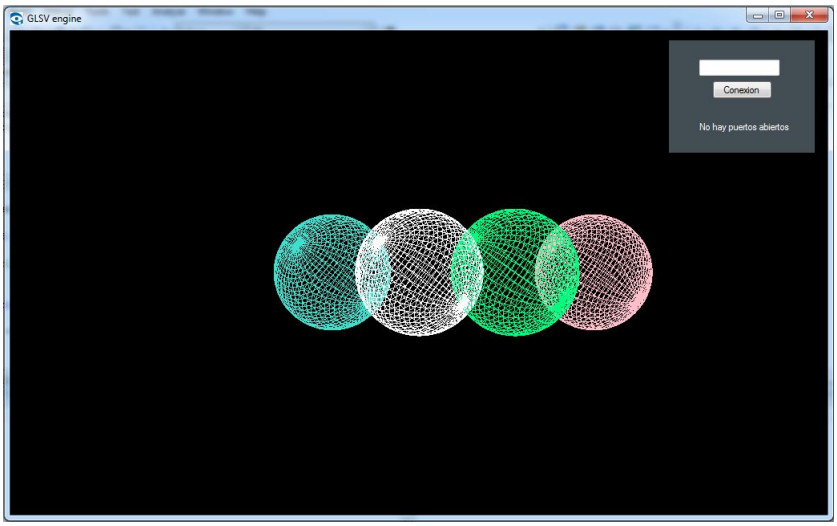

Figura 4.4: Interfaz Servidor

Una vez que son leídos datos de algún dispositivo la esfera correspondiente a este actualiza su posición, quedando como muestra la Figura 4.5, donde ya se han recibido datos de la *webcam*, el teclado y el *mouse*.

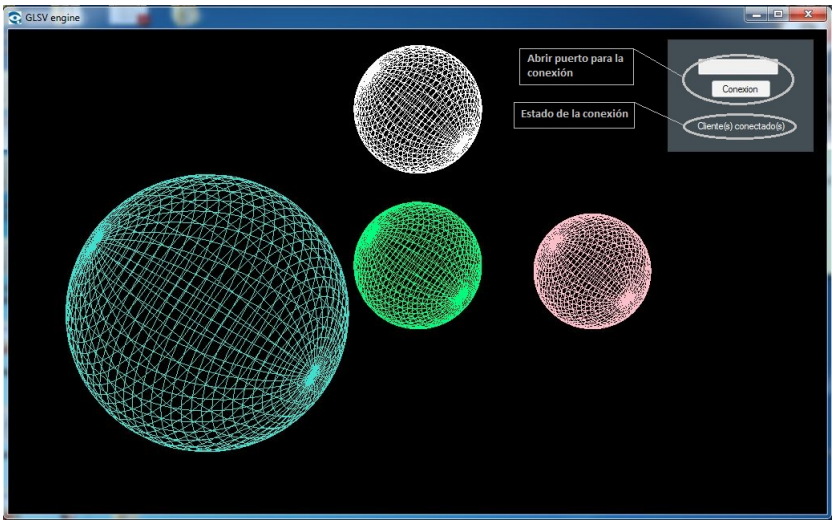

Figura 4.5: Interfaz Servidor: Movimiento del teclado, *webcam* y *mouse*

Conclusiones

Con la realización de este trabajo se concluye que:

- Se logró la realización de un módulo cliente-servidor para la comunicación de dispositivos periféricos que es utilizado por la biblioteca GLSVe en la realización de SRV
- La arquitectura de clases genérica obtenida puede ser utilizada por aplicaciones de RV que necesiten envío de datos correspondientes al estado de distintos dispositivos

Recomendaciones

Para este trabajo se recomienda:

- Incluir la gestión de la información de otros dispositivos periféricos que puedan ser empleados en SRV, como los guantes y las extensiones del *wiimote*, entre otros
- Utilizar el módulo creado en la realización de aplicaciones de RV que incluyan el manejo de dispositivos

Referencias bibliográficas

[Gua, 2001] (2001). Guante virtual. http://es.encydia.com/pt/Guante_virtual.

[RV, 2005] (2005). Realidad virtual. http://www.realidadvirtual.com/que-es-la-realidad-virtual.htm.

[Act, 2007] (2007). *Programación con ActionScript 3.0*. Adobe Systems Incorporated.

[Kio, 2008] (2008). Tcp. http://es.kioskea.net/contents/internet/tcp.php3.

[For, 2010] (2010). Introducción al formato rtf. http://www.sgoliver.net/blog/?page_id=1020.

[Tec, 2011] (2011). El teclado de la computadora. http://www.informaticamoderna.com/Teclado.htm.

[Antonio Bonilla Egido, 2001] Antonio Bonilla Egido, J. M. P. (2001). Aplicaciones distribuidas: P2p. http://studies.ac.upc.edu/FIB/CASO/seminaris/2q0304/M9.pdf.

[Blázquez, 2010] Blázquez, P. C. (2010). Sistema de dispositivos de entrada virtuales en .net.

[Caballero, 2006] Caballero, D. (2006). Reportaje: Cómo funciona el wiimote. http://www.revogamers.net/article.php?articleId=35&page=1.

[DDC, 2011] DDC (2011). Fase de construcción. http://eva.uci.cu/mod/resource/view.php?id=14094.

[de la Luz, 2011] de la Luz, S. (2011). Las topologías de redes p2p: Explicación y ejemplos. http://www.redeszone.net/2011/01/15/las-topologias-de-las-redes-p2p-peer-to-peer-explicacion-y-ejemplos/.

[Educativo, 2009] Educativo, B. T. (2009). Guía de recomendaciones. http://www.enlaces.cl/bazar_tec/uploads/media/Guia_Compra_Webcam.pdf.

[González, 2009a] González, R. (2009a). Qué es un touchpad? http://pcexpertos.com/2009/09/que-es-un-touchpad.html.

[González, 2009b] González, R. (2009b). Tipos de mouse y su definición. http://pcexpertos.com/2009/10/tipos-de-mouse-y-su-defincion.html.

[Jeff, 2008] Jeff (2008). Protocolos. http://es.kioskea.net/contents/internet/protocol.php3.

[Jeff, 2010] Jeff (2010). Ip. http://es.kioskea.net/contents/internet/protip.php3.

[Jose Guillermo Valle, 2005] Jose Guillermo Valle, J. G. (2005). Definición arquitectura cliente servidor. http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml.

[Juárez, 2004] Juárez, R. R. (2004). Interactividad en un entorno virtual inteligente: Manejo de dispositivos de realidad virtual y gestión de interacciones. Master's thesis, Universidad Politécnica De Madrid.

[Ledesma, 2010] Ledesma, M. J. (2010). Introducción a la realidad virtual.

[Li, 2001] Li, W.-J. (2001). A pc-based distributed multiple display virtual reality system.

[Marco Antonio Manayay, 2008] Marco Antonio Manayay, Darwin Víctor Herrera, D. M. (2008). Protocolo tcp/ip. http://www.scribd.com/doc/6314962/Protocolo-TCP-IP.

[Navarro, 2009] Navarro, M. A. G. (2009). Estudio de las posibilidades de un wiimote como interfaz de control. Master's thesis, Escuela Politécnica Superior de Castelldefels.

[Park, 2004] Park, Y. (2004). Dynamic shared state management for distributed interactive virtual environment.

[Pérez, 2008] Pérez, J. E. (2008). Introducción a ajax. http://www.librosweb.es/ajax/pdf/introduccion_ajax.pdf.

[Quiroz, 2007] Quiroz, J. D. C. (2007). Fundamentos de realidad virtual. http://www.google.com.cu/url?sa=t&source=web&cd=3&ved=0CCkQFjAC&url=http%3A%2F%2Faugusta.uao.edu.co%2Fgiisoft%2Frv%2Fdescargas%2FDispositivos%2520de%2520despliegue.ppt&rct=j&q=fundamentos%20de%20realidad%20virtual&ei=BKbfTa6RHKT50gGTncyeCg&usg=AFQjCNHm-3JWYNfC129RhvAYT2qZ11RE5A&cad=rja. Unidad 1: Realidad Virtual y entornos visruales.

[Saavedra, 2007] Saavedra, J. A. (2007). Patrones grasp (patrones de software para la asignación general de responsabilidades). parte ii. http://jorgesaavedra.wordpress.com/2007/05/08/patrones-grasp-patrones-de-software-para-la-asignacion-general-de-responsabilidadparte-ii

[Suárez, 2008] Suárez, J. M. S. (2008). Introducción a json. http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=prototypejsAjaxJSON.

[Valls, 2009] Valls, I. P. (2009). Paso 3: Introducción a los patrones de diseño. http://www.javadabbadoo.org/cursos/infosintesis.net/javase/paqawt/selectorcolores/paso03patrones.html. Definición de los patrones de diseño.

[Vialfa, 2008] Vialfa, C. (2008). Diferencias entre los protocolos tcp y udp. http://es.kioskea.net/faq/1559-diferencias-entre-los-protocolos-tcp-y-udp.

[Villanueva, 2002] Villanueva, M. A. S. (2002). La realidad virtual como herramienta en la enseñanza de la anatomía humana para el 4to y 5to grado de nivel primario. Master's thesis, Universidad de las Américas Puebla. Capítulo 1. Panorama General de la Realidad Virtual.

Acrónimos

API Interfaz de Programación de Aplicaciones

UCI Universidad de las Ciencias Informáticas

CEDIN Centro de Informática Industrial

HDSVE Herramienta de Desarrollo para Sistemas de Visión Estereoscópica

GLSVe *Graphics Library for Stereo Vision*

RV Realidad Virtual

SRV Sistemas de Realidad Virtual

MEMS Sistema Micro Electro-Mecánico

RGB *Red, Green, Blue*

CCD *Charged Coupled Device*

HTTP *HyperText Transfer Protocol*

FTP *File Transfer Protocol*

SMTP *Simple Mail Transfer Protocol*

POP *Post Office Protocol*

UDP *User Datagram Protocol*

TCP *Transmission Control Protocol*

IP *Internet Protocol*

DIVE *Distributed Interactive Virtual Environment*

KAVE *Kist cAVE-like*

XML *eXtensible Markup Language*

UML *Unified Modeling Language*

SO Sistema Operativo

IDE *Integrated Development Environment*

RTF *Rich Text Format*

IS Ingeniería de Software

CU Caso de Uso

GRASP *General Responsibility Assignment Software Patterns*

GoF *Gang of Four*

HTML *HyperText Markup Language*

EEPROM *Erasable Programmable Read Only Memory*

CAVE *Computer Aided Virtual Environment*

RUP *Rational Unified Process*

JSON *JavaScript Object Notation*

Anexos

Figura 6: Patrón utilizado en el seguimiento que se realiza con la *webcam*