Universidad de las Ciencias Informáticas
Vicerrectoría de Investigación y Postgrado
Dirección de Formación Postgraduada

Mecanismo de acceso a la información de dispositivos de interacción 3D desde un entorno virtual

Tesis presentada en opción al título de

Máster en Informática Aplicada

Autor: Ing. Ernesto de la Cruz Guevara Ramírez

Tutor: MSc. Iván Pérez Mallea

Co-tutor: PhD. Santiago Martín González

La Habana, 2014

En los momentos de mayor crisis o tensión ... solo la Fé en Dios y la imaginación son más importantes que el conocimiento para sacar lo mejor de ti.

Dedicatoria

A Dios, A mi esposa, A mis padres quienes han sido siempre una plataforma en la que apoyarme para todo en la vida, a mi familia

Agradecimientos

Le agradezco a Dios, mi fe en él me permitió conservar la calma y la sabiduría en los momentos de mayor tensión. Agradezco a mis padres por darme su apoyo incondicional en todo momento, Asela, Leo, Yinet y Angel me han sido de mucha ayuda. A mi esposa Mileydi, ella tiene gran parte de la responsabilidad de que este trabajo haya salido en el tiempo indicado, por comprenderme y apoyarme en los innumerables momentos de mi vida, por estar siempre a mi lado en las buenas y en las malas. A mis hermanas Natalie (y su esposo Ledif) y Berenice. A los apóstoles Yunior y Saili. A mis compañeros de trabajo del centro Vertex, en especial a los del Dpto. de Práctica Profesional que me brindaron su apoyo incondicional, Liudmila, Alina, Adrián Peña, Adrián Hernández, Amal, Andy Hernández, Andy Trujillo, Andrés Ochoa, Marvyn, Saylin, Susej, José A. Lores, Álvaro alias el chamaquito y su papá el Dr. Ramón Carrasco y a Enelis. A Luis Guillermo (el Guille) y Rubén Alcolea, este último por sus jocosos comentarios en la revisión del documento. A mis estudiantes, de ellos también he aprendido mucho en este proceso. También agradezco a mis tesistas, Joaquín, Alejandro Ravelo, José Lozano y Yuriaski. Al equipo todoterreno del 305 con su SDK de los Laboratorios Virtuales, Orlay, Alejandro, Jandy, Tabares, José Andrés y Juan Carlos por sus discusiones acaloradas y a veces sin pies ni cabeza, que sirvieron para apaciguar el estrés en muchos momentos y descubrir nuevos puntos de vista. Al equipo de diseño formado por Lorenzo, Diosmel y Alexis que me ayudaron con los modelos 3D y las escenas que se probaron en esta investigación. A la dirección del centro y de la facultad por el apoyo que me dieron durante en proceso de tesis, Omar, Hassan y la decana Mayra. A mis compañeros de los cursos de los diplomados

de la maestría, por los buenos momentos que pasamos juntos, bueno y otros no tanto. A mis tutores, Mallea y Santiago ellos me brindaron mucha ayuda en diferentes etapas del proceso de esta investigación. A Yoan y Yalice por sus consejos y por su ayuda en el desempeño de mi labor investigativa, siempre han estado ahí para apoyarme. A Amado y Yanet, Ivonne y Douglas. A mi abuela Aracelis, a mis tíos Papi, Pura y cheche, Noelvis, Lázaro, Juan (Nené), Marta, Silvia y Décoro que además han sido como unos padres para mi. A mi otra familia Migue y Ada, Ariel, Grisel, Emmanuel y Liz, Yuliet, Rodolfo y su pandilla, Juana y Roger los mirabales de las lomas. A mis primos Alexánder, Alenis y Claudia, José Alberto y familia, Addel, Lidia, Tatiana y Taimí, Cile, Celio, La nena, Yoan, Horacito y Lexis, Diamela y Cucho, Damicela y Liuben, Daciela y Raidel, Yucelis, Yordanis, Yaneilis y Onailis, Pachiro y Daily y José (cubana). Al claustro de profesores de la maestría. A los que no están reflejados explícitamente aquí y que no están olvidados. A los compañeros de viaje de la guagua y Reynaldo su chofer. En general a todos aquellos que de una forma u otra hicieron posible este trabajo les agradezco por ser parte importante de mi vida.

Declaración de autoría

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

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Ing. Ernesto de la Cruz Guevara Ramírez	MSc. Iván Pérez Mallea
Autor	Tutor

Resumen

La interacción con un entorno de realidad virtual requiere el uso de dispositivos construidos específicamente para ese propósito (dispositivos de seguimiento 3D, dispositivos hápticos o *joysticks*). En este contexto la diversidad de tipos de dispositivos de interacción 3D dificulta el acceso a su información y prolonga el tiempo de desarrollo de los entornos de realidad virtual. La interacción en los Laboratorios Virtuales desarrollados en el Centro de Entornos Interactivos 3D - VERTEX se realiza con ratón y teclado. Sin embargo, existen escenarios donde estos dispositivos no garantizan las tareas de interacción necesarias. El objetivo de este trabajo es implementar un mecanismo de acceso a la información de los dispositivos de interacción 3D, que contribuya a disminuir el tiempo de desarrollo de los entornos de realidad virtual, desarrollados por el grupo de Visualización y Realidad Virtual de la UCI e IDEASCAD de la Universidad de Oviedo. Se seleccionó la biblioteca VRPN y se adaptaron sus funcionalidades mediante la incorporación de un nuevo dispositivo de seguimiento 3D basado en visión por computadoras. Se logró una solución multiplataforma probada en sistemas Windows y GNU\Linux. El mecanismo propuesto impacta de forma positiva en el ciclo de vida de los proyectos al disminuir el tiempo de desarrollo de la etapa de codificación de la interacción 3D, constatado con la aplicación de la métrica de estimación de esfuerzo por líneas de código.

Palabras clave: dispositivos de interacción, interacción 3D, interacción hombre-máquina, realidad aumentada, realidad virtual.

Índice general

Índice de figuras

Índice de tablas

Introducción

La metáfora de interacción hombre-máquina provista por las ventanas, el ratón, los íconos y el puntero ha sido ampliamente aceptada desde la llegada de los entornos de escritorios gráficos. Esto trajo consigo la estandarización de un conjunto de técnicas de interacción, así como las tareas básicas de interacción para este tipo de entornos. Como resultado, se desarrollaron Interfaces de Programación de Aplicaciones (GUIs) maduras y estables, se formalizaron reglas para la interacción (Pittarello y Celentano, 2001) y se desarrollaron funcionalidades como parte de Entornos Integrados de Desarrollo (IDEs), para dar soporte al desarrollo de este tipo de interacción entre el usuario y el ordenador.

Por otro lado la interacción en el campo de la Realidad Virtual (RV), está muy lejos de haber llegado a la madurez. No obstante, de forma conceptual en (Herndon, Dam y Gleicher, 1994) se pone de manifiesto que existe un conjunto de tareas básicas de interacción universales, que han sido aceptadas por la comunidad científica y además, se han propuesto otras tantas clasificaciones en (Barrilleaux, 2000; D. A. Bowman et al., 2001; Boyd y Sastry, 1999). Esta situación demuestra sin embargo, que aún no se ha llegado a un consenso general en la estandarización, por lo que no se identifica un conjunto de técnicas de interacción y tareas básicas de interacción como en el caso de las GUIs.

La implementación de una forma de interacción apropiada para aplicaciones de RV o Realidad Aumentada (RA), es posiblemente uno de los mayores retos que tiene que enfrentar cualquier programador. Esto se debe a factores teóricos y prácticos que influyen directamente en la solución final. Desde el punto de vista teórico, el programador tiene que diseñar la interacción casi desde cero, como un conjunto de técnicas de interacción estándares, en un contexto en el que las tareas interactivas no han sido identificadas en detalle. Desde el punto de vista de la práctica, el programador tiene que lidiar con dispositivos de interacción que poseen funcionalidades heterogéneas. Además, está obligado a facilitar la utilización de estos dispositivos de forma simultánea y coordinada para relacionar los Entornos de Realidad Virtual (ERVs)[1] o aumentados con la interacción multimodal. Estos factores reflejan la falta de herramientas apropiadas para implementar técnicas de interacción para RV, que se traduce en la falta de estandarización de las técnicas de interacción 3D mencionada anteriormente.

La importancia de definir un conjunto universal de tareas y técnicas de interacción, radica en que esto puede significar el punto de partida del desarrollo de herramientas para la implementación de interfaces de usuario en 3D (3DUIs). Esto podría solucionar la situación actual, donde las herramientas existentes no

[1] Se utilizará este acrónimo o entorno virtual como un superconjunto, para no tener que especificar continuamente entorno de realidad virtual o aumentado, haciendo la especificación explícita solo cuando el contexto lo necesite.

brindan el soporte necesario para lidiar con la complejidad inherente a la interacción 3D con un ERV. La complejidad está dada por la naturaleza multidisciplinaria de la interacción 3D, en la cual pueden confluir disciplinas como la ergonomía, las 3DUIs, la interacción multimodal e incluso los factores humanos por solo mencionar algunos aspectos.

Esta situación no es ajena a los proyectos de RV que se realizan en las instituciones del país, donde no se conoce, hasta el momento de la elaboración de este proyecto de investigación, sobre una solución que resuelva los inconvenientes anteriormente planteados. Aunque, sí se conoce que se han llevado a cabo acciones de integración de dispositivos de interacción 3D con entornos virtuales en la empresa SIMPRO de las Fuerzas Armadas Revolucionarias. Esta empresa se encarga de producir simuladores de RV donde los usuarios interactúan con un entorno virtual simulado a través de una multitud de dispositivos, muchas veces desarrollados por ellos mismos.

El Centro Entornos Interactivos 3D (VERTEX) de la facultad 5 se han desarrollado trabajos relacionados con la RV y la RA, principalmente proyectos productivos e investigativos. Los trabajos en el área de RV se remontan a los inicios de la creación de la universidad con la producción de un simulador de conducción en colaboración con la empresa SIMPRO antes mencionada. En ese proyecto y otros más, donde se simularon entornos de prácticas de tiro, se utilizaron dispositivos como un casco de inmersión para RV, además de sensores y actuadores no convencionales para llevar a cabo la interacción del usuario con el ERV, así como recibir retroalimentación del mismo.

En este escenario cada dispositivo contaba con su propio *software* controlador, mediante el cual exponía las funcionalidades a través de una interfaz de programación de aplicaciones (API). El API de cada dispositivo, no estaba necesariamente relacionada con las de los otros dispositivos. Esto condujo a que los programadores enfrentaran el diseño de la interacción 3D basándose en sus conocimientos y criterios de trabajos anteriores. La asimilación de nuevas APIs con una curva de aprendizaje elevada, constituyó otro factor que retrasó en más de una ocasión el tiempo de desarrollo de los proyectos.

Esta situación se manifestó también en proyectos actualmente cerrados, como el *Simulador Quirúrgico* y el videojuego *Rápido y Curioso*. En estos proyectos, la interacción se diseñó a través de *joysticks* y de dispositivos como mandos de timón y pedales que no procedían del mismo fabricante y por ende sus interfaces de programación no eran compatibles entre sí. Esto trajo como resultado que los encargados de la interacción 3D de un proyecto, aunque pudieron asimilar la tecnología en un tiempo menor del acostumbrado, no pudieron reutilizar esos conocimientos en otro proyecto.

En intentos por avanzar en la investigación en RV, el Grupo de Visualización y Realidad Virtual (ViViRG), como parte del centro VERTEX, ha incursionado además en la RA. La RA constituye una variante de la RV, la cual incluye la detección del posicionamiento correcto del usuario que mira la escena. Esta funcionalidad permite que la información, que complementa la realidad, esté alineada en la perspectiva correcta del usuario, dando la sensación de que coexiste en el contexto real. El escenario de la RA exige que toda la interacción 3D sea en tiempo real interactivo (al menos 30 cuadros por segundo). Por ese motivo las aplicaciones de RA pueden realizar la funcionalidad de seguimiento (*tracking*) incluyendo varios dispositivos.

Hasta el momento, la interacción para RA en el centro VERTEX se ha realizado solamente mediante técnicas de visión por computadoras, todo en una misma PC. En ocasiones, esto trae consigo el inconveniente de que los cálculos suelen ser intensivos (por ejemplo, procesamiento de video) y de esta forma afectan el rendimiento de la PC y la aplicación en general.

Muchas aplicaciones de RV producidas en VERTEX exigen la interacción directa (por diferentes medios y dispositivos) con el usuario. Se pueden citar ejemplos como en el caso del proyecto Laboratorios Virtuales, aplicaciones de RA, y más reciente un prototipo de *software* de navegación guiada por imágenes médicas. En el último ejemplo, la detección de posicionamiento del instrumental quirúrgico es una funcionalidad de prioridad, lo cual se puede hacer con una combinación de dispositivos de interacción. De forma general en el centro VERTEX existe tal diversidad de dispositivos de interacción aislados, que su combinación en soluciones nuevas incrementa el tiempo en que se termina un producto de RV.

Por otro lado existe un convenio marco de colaboración internacional entre la Universidad de Oviedo (UNIOVI) y la Universidad de las Ciencias Informáticas (UCI), mediante el grupo de Investigación IDEASCAD y el ViViRG del centro VERTEX, sobre temas de RV y visión estereoscópica. El grupo IDEASCAD posee un Cueva de Entornos de Realidad Virtual Automática (CAVE)[2] donde desarrolla sus aplicaciones de RV con una biblioteca de gráficos por computadora y visión estereoscópica de desarrollo propio, llamada GLSVe (Martín et al., 2009). En esta biblioteca se le ha dado soporte a dispositivos de interacción de forma aislada que traen consigo los inconvenientes analizados anteriormente.

En los proyectos de inspección virtual desarrollados para el CAVE se da la necesidad de combinar dispositivos de interacción para realizar tareas de interacción avanzadas. IDEASCAD posee varios dispositivos de interacción que necesitan incorporar al CAVE para utilizar en las aplicaciones que desarrollan, por ejemplo Wiimote, DTrack, ARTrack2, dispositivo inercial Inertia Isense, dispositivo háptico y NDI Polaris Vicra. Las interfaces de programación de los dispositivos mencionados son diferentes entre sí. Además, la combinación de dichos dispositivos ofrece la ventaja de suplir funcionalidades no existentes o crear alternativas a otros dispositivos. Tal es el caso de la rotura del dispositivo de navegación e interacción (FlyStick) que viene acoplado al CAVE con el servidor DTRack, que se sustituyó mediante la incorporación de un Wiimote.

En este caso la concentración de dispositivos en un solo ordenador sobrecarga el procesamiento de la estación de trabajo y en dependencia de la cantidad de dispositivos, se agotan los puertos de conexión de la estación de trabajo. A esto se le suma la necesidad de incorporar técnicas de seguimiento posicional (3D *tracking*) mediante visión por computadoras a la biblioteca GLSVe para aumentar su versatilidad.

En los proyectos del grupo ViViRG, como parte del centro VERTEX y el grupo IDASCAD, se identificaron como regularidades:

- La existencia de diferentes situaciones en las que necesitan incorporar técnicas de interacción, incluso como una combinación de técnicas de interacción 3D y dispositivos de seguimiento posicional.
- La diversidad de dispositivos de interacción, propicia que estos puedan o no, estar localizados en un mismo lugar o estación de trabajo.

[2]Cueva de RV (CAVE por sus siglas en inglés de *Cave Automatic Virtual Environment*)

- La necesidad de combinar los dispositivos en una solución de interacción donde sus datos se adquieran de la forma más uniforme posible.

Los elementos anteriormente expuestos prolongan el tiempo de ejecución de los proyectos y de asimilación de las APIs de los dispositivos de interacción por parte de los desarrolladores. Esto incide negativamente en el avance de los proyectos, lo cual repercute negativamente en el cumplimiento de la planificación inicial de los proyectos productivos y de investigación. Al mismo tiempo, esta situación afecta los productos desarrollados, adicionando una complejidad a la solución que afecta el mantenimiento del *software*.

Después de analizar la situación antes expuesta, se define el **problema científico**: La diversidad de tipos de dispositivos provoca que el acceso a su información, incremente el tiempo de desarrollo de los entornos de RV producidos por los grupos ViViRG e IDEASCAD. Se define el **objeto de estudio**: La adquisición de información de dispositivos de interacción 3D. Para dar solución a la problemática antes mencionada se definió como **objetivo general**: Implementar un mecanismo de acceso a dispositivos de interacción 3D, que contribuya a disminuir el tiempo de desarrollo de los ERV producidos por los grupos ViViRG e IDEASCAD.

El **campo de acción** se enfoca en el acceso a dispositivos de interacción 3D en ERVs. Para guiar la investigación y comprobar los resultados se propone la siguiente **hipótesis**: El acceso a la información de los dispositivos de interacción mediante un mecanismo de acceso, permitirá reducir el tiempo de desarrollo de los ERVs producidos por ViViRG e IDEASCAD.

Objetivos específicos:

1. Establecer los referentes teórico-metodológicos sobre los mecanismos de acceso a la información de los dispositivos de interacción 3D en un ERV.

2. Implementar un mecanismo de acceso a las funcionalidades de los dispositivos para la interacción 3D con un ERV.

3. Validar los resultados obtenidos al introducir un mecanismo de acceso a las funcionalidades de los dispositivos de interacción 3D.

Entre los métodos utilizados en esta investigación se destacan los siguientes:

Histórico-lógico: se utilizó para realizar un análisis de la información consultada en las fuentes bibliográficas. Este método permitió identificar las tendencias y cómo se ha comportado el desarrollo de *software* en torno a las tareas básicas de interacción 3D, las técnicas de interacción 3D y el acceso uniforme a la información de los dispositivos de interacción desde un ERV.

Analítico-sintético: se utilizó para la caracterización de las diferentes clasificaciones de dispositivos de interacción y permitió seleccionar las características específicas a tener en cuenta para la solución.

Modelación: este método se utilizó para crear abstracciones e interpretar la realidad existente en el centro VERTEX e IDEASCAD. Se utiliza para crear modelos de funcionamiento de los flujos de trabajo enmarcados en el área de la interacción 3D con ERVs utilizados en el desarrollo de la investigación.

Métrica-Líneas de Código (LDC): este método se utilizó para comparar el esfuerzo en LDC necesarias para implementar el acceso a la información de los dispositivos de interacción y cómo este esfuerzo impacta en el tiempo de desarrollo del proyecto, antes y después de la solución propuesta.

La novedad de la investigación está dada por el hecho de que las técnicas de interacción 3D que utilizan las aplicaciones de RV producidas por el centro VERTEX e IDEASCAD no tienen un acceso uniforme a las funcionalidades de los dispositivos de interacción. La utilización de un mecanismo de acceso a la información de los dispositivos de interacción desde un ERV, contribuirá con la disminución del tiempo de desarrollo de los proyectos de RV. Se espera además, disminuir el tiempo de aprendizaje de los programadores, cuando se enfrenten a la incorporación de un nuevo dispositivo de interacción no contemplado inicialmente en la solución general de un ERV.

El documento está estructurado en 3 capítulos. En el Capítulo §1 se exponen los principales conceptos y tendencias, así como los trabajos relacionados con el acceso a las funcionalidades de los dispositivos de interacción 3D desde un entorno de RV, mediante interfaces comunes. En el Capítulo §2 se propone una solución al problema que da inicio a la investigación desde una perspectiva teórica, estableciendo los principales elementos que componen la solución. Finalmente en el Capítulo §3 se valida la solución propuesta mediante la implementación de una aplicación demostrativa. Se hace un análisis del impacto del esfuerzo de codificación en el tiempo de desarrollo.

CAPÍTULO 1

Fundamentación Teórica

1.1. Introducción

La configuración de la comunicación de los dispositivos de interacción con aplicaciones de RV y RA, permanece como uno de los mayores retos para los programadores en la actualidad. La falta de interfaces de comunicación uniformes para la Interacción Multimodal Hombre-Máquina (MHCI) ha elevado la actividad investigativa en este campo. En este capítulo se ofrece una perspectiva general de los principales referentes teóricos relacionados con el tema de investigación.

1.2. Interacción hombre-máquina

La aparición de las computadoras constituyó un momento propicio para la aparición de los primeros programas de ordenador. De ahí que la programación, surgida inicialmente como un arte, haya evolucionado a lo largo del tiempo hasta convertirse en una ciencia, la ingeniería del *software*. Durante las primeras décadas, el esfuerzo de los ingenieros de *software* se dirigió hacia el desarrollo de herramientas y aplicaciones que cumpliesen con todos los requisitos funcionales que se les exigían. La interfaz de usuario no representaba un aspecto importante en el desarrollo, porque las limitaciones técnicas de aquellos momentos imponían austeras interfaces basadas en comandos, formularios o menús, y que en cualquier caso se presuponía que el usuario aprendería a utilizar el sistema con el debido tiempo y esfuerzo.

Sin embargo, esta situación tuvo un cambio a partir de los años 80, en el momento en que la tecnología hizo posible la comercialización de ordenadores para la oficina y el hogar con interfaces gráficas, y las aplicaciones basadas en la metáfora del escritorio y el ratón comenzaron a ser utilizadas por millones de personas. Se reconoció que, por encima de unos requisitos mínimos de funcionalidad, el factor más importante del éxito de una herramienta informática era la facilidad de uso de su interfaz.

La Interfaz de Usuario (en inglés, *User Interface*, UI) pasó así a ocupar un papel protagonista dentro del desarrollo de cualquier aplicación, y como consecuencia directa, creció enormemente el interés por la Interacción Persona-Ordenador (IPO). Conocida también bajo las siglas Interacción Hombre-Computadora

(HCI), se trata de un campo de estudio que conecta conocimientos tan usuales dentro de los estudios de informática, como la programación, la ingeniería del *software*, la inteligencia artificial o la telemática, con otros conocimientos que pueden resultar ajenos a ellos, como la lingüística, la psicología, el diseño, la sociología o la ergonomía (Preece et al., 1994; Vilar, 2010).

La importancia adquirida por este campo de estudio en la actualidad se refleja en el gran número de talleres, conferencias, congresos y otros simposios que se organizan cada año con esta temática. Por ejemplo,la Asociación Interacción Persona-Ordenador (AIPO) organiza todos los años desde el 2000, las jornadas de interacción como un foro en el que docentes, investigadores y profesionales de España y Latinoamérica, exponen y comparten sus ideas (AIPO, 2013). Toda esta actividad se traduce en una extensa bibliografía en la que se recogen los avances que se producen de forma continua en este campo y que le han llevado a su actual estado de madurez. Esto es así, al menos, en cuanto a las interfaces de escritorio.

1.2.1. Interacción con Realidad Virtual

La tecnología de creación de gráficos en 3D y RV se utilizan para simular espacios tridimensionales como los que podemos observar en el mundo real. El grado de correspondencia con la realidad depende de los objetivos del *software* en sí. De esta forma, se pueden encontrar ERVs con los que se pretende reproducir determinados ambientes reales en diferentes etapas de la vida, tanto en el pasado (arqueología), en el presentes (simuladores, ciudades virtuales, tele-presencia), como en el futuro (prototipos virtuales) (Engel y Döllner, 2012; Maimone y Fuchs, 2011; Park et al., 2014; Seth, Vance y Oliver, 2011).

En el caso de los simuladores, se persigue que el usuario pueda desarrollar unas habilidades en el mundo virtual, que luego podrá poner en práctica en el mundo real, y por ello autores como Stuart, 2001, califican este uso como entrenamiento y ensayo "*offline*". Resulta útil cuando el entrenamiento en el mundo real es mucho más costoso o propicia riesgos que se desean evitar, tal es el caso de los simuladores de vuelo o los ensayos de operaciones quirúrgicas. En este caso, una de las prioridades en el diseño de la interfaz es lograr un gran realismo, con el fin de que el conocimiento aprendido o las habilidades entrenadas, puedan trasladarse con éxito al mundo real y no se vean perjudicadas por cualidades o efectos no deseados del SRV (Arhippainen, Pakanen e Hickey, 2013; D. A. Bowman et al., 2004). Otra de las prioridades es lograr que la sensación de presencia del usuario sea elevada, con el fin de que experimente las acciones en el mundo virtual como si estuviera en el ambiente real.

En el caso de la tele-operación (u operación "*online*", como la califica Stuart), también se persigue que se sienta la realidad tal como es, pero la razón ahora es que las acciones que se realizan en el mundo virtual tienen un efecto sobre el ambiente real, que está siendo operado a distancia. Sin embargo, a diferencia de los simuladores, en este caso el operador puede ser asistido por el ordenador en su trabajo, proporcionándole facilidades que en la realidad no tendría.

En cualquier caso, reproducir la realidad exactamente como es no suele ser práctico, ya sea por tiempo de desarrollo, coste en equipos, o por las propias limitaciones de la tecnología. Por ello, es usual que aparezcan ERVs que guardan gran similitud con los reales. En tales entornos los creadores suelen tomarse ciertas

libertades para lograr un objetivo específico, por ejemplo la utilización de un algoritmo automático para generar bosques. El autor puede seguir este camino diluyendo cada vez más la correspondencia existente entre el entorno virtual y el ambiente real, utilizando por ejemplo modelos reales para crear "ambientes hipotéticos". Siguiendo este camino, quedaría atrás cualquier correspondencia con ambientes reales, dando paso a los "mundos imaginarios".

Además de su correspondencia con ambientes que existen en la realidad, se pueden clasificar los mundos virtuales en base a su grado de naturalidad, esto es, la medida en que simulan las leyes físicas que rigen el mundo real y a las que el usuario está habituado. En este sentido Sutcliffe, 2003, distingue entre "entornos virtuales naturales", "entornos naturales híbridos", y "entornos artificiales". Los primeros se asemejan tanto a la realidad que, idealmente, el usuario no se daría cuenta de que son sintéticos. Los entornos naturales híbridos, aunque estén basados en la realidad, permiten romper ciertas leyes básicas de la física, por ejemplo la gravedad. Finalmente, los entornos artificiales guardan poca o ninguna relación con la realidad, en estos se pone de manifiesto la total libertad del autor al crearlos. De forma general imitar la realidad, tratando de salvar al mismo tiempo todas sus dificultades, puede no resultar tan sencillo, lo que demandaría la utilización de tecnologías de adquisición y procesamiento de datos potentes. Además, Eastgate, 2001 advierte que algunas acciones relativamente simples en el mundo real, pueden ser complejas a la hora de interactuar con el entorno virtual . Otro elemento que puede ser aún peor, es que según el mismo autor, muchos problemas de interacción pueden existir ya en el mundo real, y que al ser llevados al entorno virtual pueden ser agravados.

Otro uso dado a los gráficos 3D, es el de aprovechar al máximo el espacio físico que proporciona la pantalla, y que limita la información que puede presentarse de forma simultánea en la misma. Pittarello y Celentano, 2001, describen que las interfaces bidimensionales (2D) proporcionan barras de desplazamiento, lentes de escalado "*zoom*" y otros controles para examinar la información que queda oculta por las limitaciones de espacio y resolución. Sin embargo, es precisamente el enorme número de controles que deben proporcionar esas interfaces para aplicaciones con mucha información, las lleva a ir más allá de los gráficos 2D (Nordahl et al., 2011).

Manipulación de los objetos tridimensionales dentro de un ERV

Además de visualizar la información en tres dimensiones desde diferentes perspectivas, otro uso que suele acompañar al anterior es el de interactuar con la propia información, manipulándola en tres dimensiones. Un ejemplo es la creación y edición de modelos en tres dimensiones, este ejemplo muestra evidencias que indican que no es apropiado para el estilo WIMP (Scali, Wright y Shillito, 2003).

Esta categoría también incluye la entrada de datos basada en gestos corporales. Un ejemplo ilustrativo es la interfaz de la aplicación que podía verse en la película "Minority Report" (2002), en la cual, si bien la información se presentaba en dos dimensiones, el usuario interactúa con ella por medio de gestos manuales expresados en las tres dimensiones del espacio. Este mismo ejemplo se puede ver con el método de interacción propuesto por Microsoft con su dispositivo Kinect Izadi et al., 2011; Microsoft, 2014.

La interacción dentro de la RV, se basa en un conjunto de interfaces gráficas virtuales. Esta constituye

la interfaz de comunicación presente en muchos entornos virtuales, en los cuales el usuario interactúa con objetos virtuales que, en entornos naturales (siguiendo la clasificación de Sutcliffe comentada en la sección 1.2.1), son una imitación de objetos reales. Otros elementos artificiales pueden estar presentes también, típicamente para dar soporte a la interacción, además de las interfaces gráficas 2D, ya sean virtualizadas o no. El máximo grado de inmersión se mantiene en medio-alto, pues aunque el usuario pueda experimentar una fuerte sensación de presencia, se entiende que las limitaciones presentes en las tecnologías de RV impiden aún equipararse con la propia realidad.

1.2.2. Interacción con Realidad Aumentada

La tecnología de creación de gráficos 3D, también se utiliza para crear interfaces de usuario para RA. Según (Azuma et al., 1997), la Realidad Aumentada es una variación de los ERV o SRV como se conocen comúnmente. La RV sumerge al usuario dentro de un ambiente sintético (generado por la computadora). Mientras está inmerso, el usuario no puede ver el mundo real alrededor de él. A diferencia de la RV, la RA le permite al usuario ver el mundo real, con objetos virtuales coexistiendo en el mismo espacio tridimensional con los reales, o solapados con este. De ahí que la RA actúe como complemento de la realidad, en vez de reemplazar completamente la realidad como lo hace la RV.

La diferencia entre RV y RA no es tan clara. La dificultad de proporcionar retroalimentación táctil y de fuerzas en un mundo virtual se puede resolver con la introducción de objetos tangibles en el sistema, lo que aproxima al usuario a la RA (Sutcliffe, 2003). En ese sentido, el paso gradual desde los mundos virtuales hasta el mundo real, fue definido como un continuo *Realidad-Virtualidad*, entre cuyos extremos se encuentran la RA, donde el mundo real predomina sobre el virtual y la Virtualidad Aumentada (VA), donde el mundo virtual predomina sobre el real (Milgram y Kishino, 1994). La VA es un término creado por Milgram para identificar sistemas que son sintéticos, pero que adicionan imágenes y videos del mundo real, como por ejemplo, texturas basadas en fotos y videos capturados de la realidad, que se incorporan a los objetos que existen en el entorno virtual. A todo ese espacio intermedio, Milgram y Kishino lo llamaron Realidad Mixta (MR). La figura 1.1 ilustra este continuo.

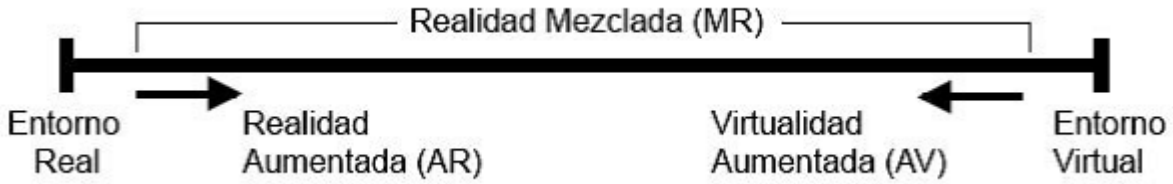


Figura 1.1. Continuo de Realidad-Virtualidad de Milgram y Kishino.

En esta forma de MR, como los elementos virtuales “aumentan” el mundo real, el cual es el medio dominante, el grado de inmersión tiende a incrementar hasta aproximarse a la cota más alta.

1.2.3. Interfaces de interacción 3D

En el diseño de las interfaces de usuario para la interacción con los ERV, se tienen en cuenta varios aspectos. La interfaz de usuario tiene dos funciones primordiales: revelar el estado interno de la aplicación al usuario y cambiar ese estado interno siguiendo las instrucciones del usuario. Con ese doble propósito se suman en la interfaz diferentes elementos. Según (Marsh et al., 1998) en el caso de los sistemas de RV, los componentes principales de todos ellos son tres: la composición de las imágenes, la interacción y el comportamiento. Teniendo en cuenta este desglose por componentes, esta investigación está fuertemente relacionada con el componente de interacción.

Cuando se menciona el término "interacción" en el contexto de los ERV, debe entenderse no sólo el hecho de que el usuario pueda manipular los objetos 3D en el entorno, sino que también pueda navegar por ese entorno y observar los objetos desde diferentes puntos de vista. De hecho, la navegación no es más que el control del usuario sobre su representación en el mundo virtual ("*the presence*" o "*the self*"), lo que en entornos multiusuario se conoce como "avatar" (Sutcliffe, 2003). No obstante, también podría considerarse el control sobre el mundo, si se tiene en cuenta que es este el que se mueve en relación al avatar. Para ser precisos, se debe hablarse de viajar y no de navegar, pues el concepto de navegación incluye un componente motor "viajar" (*travel*) y otro cognitivo "decidir el camino" (*wayfinding*) (D. A. Bowman et al., 2001; Hilliges et al., 2012).

Hasta aquí se hizo una breve introducción a la interacción, como uno de los elementos que conforman las interfaces de usuario tridimensionales. No es objetivo de esta investigación brindar una explicación exhaustiva de la interacción 3D, ya que sobre este tema se han escrito libros enteros. La interacción a su vez está compuesta por otros elementos. Foley et al., 1996, expresa que los elementos básicos de las interfaces de usuario son:

- Diálogos interactivos
- Tareas de interacción
- Técnicas de interacción
- Acciones con dispositivos de entrada

En las siguientes secciones se describen los diferentes aspectos relacionados con los dispositivos de entrada. Estos además de ser imprescindibles para la interacción 3D con un ERV constituyen el centro de esta investigación.

1.3. Dispositivos de interacción

Los dispositivos de entrada pueden ser catalogados según el tipo de evento que generan. Estos se encargan de capturar las acciones del usuario y enviar la información al ordenador. Según (Sutcliffe, 2003), los dispositivos pueden ser discretos y/o analógicos. A estos últimos, se les denomina como dispositivos

continuos, y añaden otra categoría más, la de los dispositivos de entrada híbrida (D. A. Bowman, Chen et al., 2006). Otra clasificación atendería a los Grados de Libertad, (DOF) del dispositivo, siendo los dispositivos más populares los 2D (como el ratón) (Foley et al., 1996). En la tabla 1.1 se pueden apreciar las tareas más representativas soportadas por estos dispositivos.

Tabla 1.1. Clasificación de dispositivos de interacción.

Tipo de dispositivo	**Grados de libertad**	**Tareas representativas**
Control deslizante, dial	1	Control de volumen
Ratón	2	Selección, dibujo, posición 2D
Ratón 6D, *tracker* (en sus diferentes variantes)	6	Orientación, posicionamiento, control de punto de vista
Guante, máscara	16 o más	Animación de la mano, cara
Traje de cuerpo	100 o más	Animación de todo el cuerpo
Híbridos	1 o más	Tareas complejas, (Orientación con selección)

No obstante, Schomaker et al., 1995 en su taxonomía brindan distintas clasificaciones de los dispositivos de entrada. Esta clasificación se refiere a:

- Dispositivos señaladores (ratón, lápiz de Tablet PC, lápiz óptico, pantalla táctil, guante de datos)
- Teclados (ASCII (Qwerty, Dvorac), teclado numérico, teclas de cursor, MIDI)
- Entradas de sonido (Micrófono)
- Entradas de imagen (cámara, sensor de video)
- Otros sensores

En esta taxonomía se observa la posibilidad de que existan otros dispositivos diferentes de los señaladores, que sólo brinden coordenadas de entrada. Por ejemplo, muchos sistemas de ventanas permiten que la tecla **TAB** del teclado mueva el cursor para seleccionar los campos de entrada en un formulario. Esto tiene el mismo efecto que seleccionar el campo correspondiente con el ratón. Los dispositivos de entrada de imagen normalmente no forman parte de la interfaz de usuario de una aplicación, estos generalmente se utilizan en la captura de imágenes. Sin embargo, las cámaras de video, son utilizadas de forma interactiva en aplicaciones de videoconferencia.

Algunos dispositivos brindan datos de forma indirecta: por ejemplo, la posición de una palanca de mando (*joystick*) indica el movimiento en una dirección específica. Esto puede traducirse en cambios de coordenadas. Sin embargo, otras tareas como los saltos rápidos de un lugar a otro son difíciles de lograr con un *joystick* y requiere el uso de otros dispositivos. En principio, también es posible traducir dos parámetros de voz (por ejemplo, volumen y tono) en coordenadas de pantalla. Esto podría ser útil en algunas aplicaciones para las personas con discapacidad, las cuales podrían ser capaces de controlar la posición de un objeto de pantalla por medio de comandos de voz.

1.3.1. Dispositivos comunes de interacción

En esta sección se describen los periféricos de entrada más frecuentes en los SRV. Los dispositivos de seguimiento son de los más empleados, estos permiten al sistema conocer la posición y orientación de la cabeza, de las manos, o de todo el cuerpo del usuario en tiempo real interactivo. También es frecuente el uso de guantes de datos (que permiten detectar movimiento de los dedos de la mano) y los micrófonos (que graban la voz del participante). Los periféricos de un SRV se clasifican según el sentido de la información entre el participante y la computadora. Los dispositivos de entrada, también denominados *sensores*, capturan las acciones del usuario (por ejemplo movimientos de la cabeza) y envían esta información a la computadora encargado de llevar a cabo la simulación. Los dispositivos de salida, también denominados *efectores*, generan los estímulos necesarios para los sentidos del usuario, traduciendo las señales de video y audio que reciben de la computadora, en imágenes y sonidos respectivamente.

1.3.2. Clasificación de dispositivos de interacción

Los dispositivos de entrada (*sensores*), además de las clasificaciones vistas al inicio de la sección 1.3, pueden clasificarse de otras formas. En la presente investigación, la clasificación ofrecida en la sección 1.3.1 se enriquece con otros elementos obtenidos a partir del análisis de varias clasificaciones, encontradas en la bibliografía, que la complementan para ofrecer una clasificación más generalizada:

- Dispositivos de seguimiento (Magnéticos, Ópticos, Acústicos, Mecánicos, Inerciales)
- Guantes de datos
- Registro de voz
- Dispositivos de entrada 3D

Esta clasificación ayuda a organizar los dispositivos de acuerdo a la forma en que estos operan, con el objetivo de tener una mejor aproximación al tipo de datos que ofrecen. Esto es especialmente útil para la generalización de sus interfaces. En las próximas secciones se describen los dispositivos de entrada relacionados anteriormente. Los cuales son considerados de RV por dos motivos fundamentales: utilizan el paradigma de la *interacción implícita* (la voluntad del usuario se captura implícitamente en sus movimientos y acciones naturales) y *proporcionan entrada de datos 3D* al sistema, para hacer más sencilla la exploración del mundo virtual.

Dispositivos de seguimiento

Los dispositivos de seguimiento son sensores que tienen como misión capturar la posición y orientación de un objeto real y enviar esta información a la computadora. Existen dispositivos de seguimiento que solamente registran la posición de un objeto, por tanto se dice que miden 3 DOF. Estos 3 DOF son las tres

coordenadas (X, Y, Z) del objeto respecto a un sistema de coordenadas conocido, el cual proporciona el dispositivo de seguimiento. Otros dispositivos de seguimiento sólo registran la orientación de un objeto. En este caso también se tienen 3 DOF, que son los ángulos de rotación alrededor de los ejes (X, Y, Z). Estos ángulos reciben nombres particulares en RV, especialmente cuando lo que se mide es la orientación de la cabeza (ver figura 1.2):

- ***Yaw*** (***azimuth***). Ángulo de rotación respecto al eje vertical. Este ángulo varía por ejemplo cuando se mira de derecha a izquierda.
- ***Pitch*** (***elevation***). Ángulo de rotación respecto un eje horizontal que une las dos orejas. Este ángulo mide la elevación respecto al horizonte, y varía por ejemplo cuando se mira de arriba a abajo.
- ***Roll***. Ángulo de rotación respecto al eje determinado por la dirección de visión. Este ángulo varía por ejemplo cuando se acerca o se aleja la cabeza respecto a un punto que queda al frente.

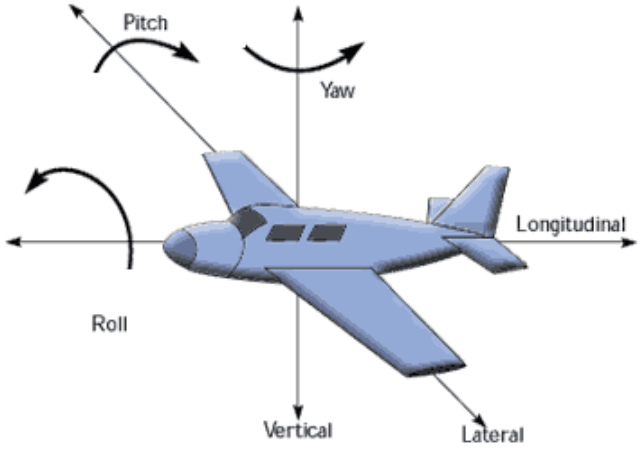


Figura 1.2. Ángulos (Yaw,Pitch, Roll), también conocidos como ángulos de euler.

Finalmente, la tercera clase de dispositivos de seguimiento, proporcionan *posición* y *orientación* del objeto rastreado de forma simultánea, por lo que se denominan dispositivos de seguimiento con 6 DOF. A este tipo de dispositivo, también se le conoce con el término *Tracker*.

Los dispositivos de seguimiento se utilizan con frecuencia en RV para capturar los movimientos de la cabeza del usuario, lo que permite generar las imágenes de acuerdo con su perspectiva o punto de vista. También es frecuente registrar los movimientos de la mano para permitir al usuario interactuar con los objetos de la escena de forma natural. Algunas acciones como agarrar o mover objetos de la escena se pueden realizar de forma mucho más intuitiva mediante un reconocimiento de los movimientos del usuario. Aunque las funcionalidades de los dispositivos de seguimiento no suelen variar, estos si varían en función de la tecnología que utilizan para lograr su objetivo, lo que en ocasiones los hace más idóneos que otros

según el contexto de aplicación. Otro elemento es que no existe una interfaz estándar o protocolo para la comunicación con estos dispositivos, porque los fabricantes generalmente no se ponen de acuerdo.

Los *dispositivos de seguimiento magnéticos* utilizan bobinas para obtener la posición y orientación. Su forma de operación se basa en las variaciones de tensión eléctrica, inducida por una fuente de campo magnético que debe estar siempre próxima a los sensores. Estos dispositivos requieren cables para conectarse a la unidad central, son sensibles a interferencias magnéticas por objetos metálicos ferrosos y tienen un radio de acción de pocos metros. La precisión disminuye conforme se van alejando de la fuente que produce el campo magnético. Esta puede ser una característica inconveniente en un contexto donde se requiera alta precisión. Estos dispositivos suelen ser pequeños, con una alta libertad de movimientos y no precisan de una "línea de visión" entre el emisor y el receptor, lo cual puede ser una ventaja sobre otro tipo de sensores, por ejemplo los ópticos. No obstante, en la actualidad se reportan dispositivos de este tipo con altos niveles de precisión como en el caso de la figura 1.3b.

(a) Polhemus Liberty (Polhemus, 2013)

(b) NDI Plaris Aurora (N. D. Inc., 2013a)

Figura 1.3. dispositivos de seguimiento magnéticos.

Los *dispositivos de seguimiento ópticos* se basan en el procesamiento de imágenes capturadas por una o varias cámaras de óptica y posición conocidas. La imagen se filtra buscando una serie de patrones reconocibles, que dan pistas sobre la posición y orientación del objetivo. Este análisis de imagen puede introducir un retardo, el cual siempre hay que tener presente según el contexto de aplicación. No obstante, con la potencia de cálculo del *hardware* actual este retardo suele ser insignificante. Este tipo de dispositivo tiene el problema de que siempre se debe mantener una "línea de visión" directa entre el emisor y el receptor, y normalmente es sensible a los cambios de iluminación. Para resolver este último contratiempo se han utilizado sensores en el espectro de luz infrarroja con un alto grado de efectividad. Este tipo de dispositivo de seguimiento es uno de los que brinda mejor precisión. Algunos ejemplos los podemos ver en la figura 1.4.

Los *dispositivos de seguimiento acústicos* utilizan sonido ultrasónico y micrófonos para la localización. Suelen soportar distancias mayores que los magnéticos y su latencia también suele ser pequeña (aunque proporcional a la distancia), pero son poco utilizados porque la señal sonora se distorsiona con facilidad. Por otro lado, también presentan el mismo problema de pérdida de "línea de visión" que padecen los dispositivos de seguimiento ópticos.

Los *dispositivos de seguimiento mecánicos* utilizan potenciómetros montados sobre una estructura articulada para medir los ángulos de las articulaciones. Son los más precisos y poseen latencias bajas. Son

(a) NDI Polaris optical family (N. D. Inc., 2013b)

(b) Microsoft Kinect (Microsoft, 2014)

Figura 1.4. dispositivos de seguimiento ópticos.

utilizados con mucho éxito en sistemas de neurocirugía asistida por computadoras, para dar retroalimentación visual al cirujano en tiempo real sobre los instrumentos quirúrgicos. No tienen el problema de pérdida de "línea de visión" ni padecen interferencias externas, pero limitan la libertad de movimientos. Además, acostumbran a ser voluminosos y pesados y suelen ser equipos de alto costo respecto a los analizados hasta el momento.

Los *dispositivos de seguimiento inerciales* se basan en pequeños dispositivos que permiten medir la aceleración con la que se mueven. Tienen un radio de acción prácticamente ilimitado. Sin embargo, el principal inconveniente es que el error en la medición es acumulativo, debido a que cada posición se calcula a partir de la última posición obtenida. Son adecuados para detectar movimientos pero no para obtener una posición absoluta. Este tipo de dispositivo de seguimiento se suele utilizar en combinación con sensores ópticos para resolver estas insuficiencias. El *Wiimote* de *Nintendo* es un ejemplo típico de la afirmación anterior, debido a que es un dispositivo con varios acelerómetros e incorpora un sensor óptico infrarrojo.

Figura 1.5. Nintendo Wiimote.

Guantes de datos

Los *guantes de datos* que se utilizan en RV permiten detectar la posición de los dedos de la mano, normalmente expresada como los ángulos de flexión para cada dedo. Los guantes de datos se utilizan para interactuar con los objetos de la escena mediante gestos naturales, así como para dar órdenes al sistema mediante un lenguaje basado en signos. Estos suelen tener un precio alto en el mercado y se utilizan en contextos de interacción específicos (Ltd., 2014; Seth, Vance y Oliver, 2011).

Figura 1.6. Guantes de datos 5DT *Ultra Wireless Kit*.

Registro de voz

En RV el reconocimiento de voz se utiliza como una forma natural de entrada de órdenes. De forma general los sistemas de reconocimiento de voz están poco desarrollados y normalmente requieren de entrenamiento para tener un funcionamiento aceptable. A pesar de la afirmación anterior, existen algunos sistemas de registro de voz con funcionalidades avanzadas, como es el caso de *SIRI* del sistema operativo para iPhones: *iOS* (A. Inc., 2013).

Dispositivos de acción de entrada 3D

Los dispositivos de entrada 3D permiten interactuar con un modelo 3D de forma más cómoda que los anteriores. Normalmente se clasifican según el número de grados de libertad. A continuación se brinda una descripción de estos dispositivos:

- **Space Ball**: Este es un dispositivo con una superficie similar a un *trackball*. Está diseñado para que el usuario lo empuje con la mano o lo intente girar. El dispositivo emite información sobre la fuerza y el momento angular que se le aplicó. De este modo se pueden obtener datos con 6 DOF disponibles en el espacio (posición y rotación en el espacio 3D).

- ***Joystick 3D***: Con este dispositivo se está en presencia de una variante de la palanca clásica de los videojuegos(*joystick*), en el cual se ha incorporado internamente un dispositivo de seguimiento. De esta forma es posible utilizarlo dentro de un CAVE. Los *joysticks* 3D tienen un inconveniente importante, no están estandarizados y sus características pueden diferir mucho según el contexto para el que se fabrican.

- ***Stylus***: Este dispositivo actúa como un puntero virtual, representado con frecuencia como un rayo con el que se puede apuntar a objetos de la escena virtual, seleccionarlos y manipularlos mediante botones

Figura 1.7. IBM Spaceball 3D.

del propio *stylus*. Con este dispositivo el usuario puede seleccionar y mover objetos, o también navegar a través de menús tridimensionales.

El análisis de las principales características de los dispositivos descritos en esta sección hace notar que existen grupos generales en los que se clasifican estos dispositivos. Sin embargo, estos grupos generales no garantizan que exista un API uniforme para el acceso a su información, aunque se han realizado investigaciones para resolver los inconvenientes que esto trae consigo. De forma general no existe un acuerdo entre los fabricantes en cuanto a la uniformidad de las características de tales dispositivos, las cuales pueden variar incluso dentro del mismo grupo de clasificación. En la sección 1.3.3 se hace una breve descripción del problema que representa la diversidad de los dispositivos de entrada, el cual es la principal problemática que dio origen a la presente investigación.

1.3.3. Problemas asociados a la heterogeneidad de los dispositivos

Una de las razones por la que las interfaces gráficas 2D son más fáciles de diseñar que las 3D, es porque sus conocidas técnicas de interacción están basadas en unos pocos dispositivos físicos, típicamente el ratón, el teclado y la pantalla. En cambio, el espacio de diseño de las interfaces de usuario 3D es significativamente mayor (D. A. Bowman et al., 2001). Los diseñadores tienen que afrontar una gran variedad de dispositivos de entrada y salida con tecnologías y funcionalidades diferentes. La aparición de nuevos dispositivos supone su rápida incorporación en un ambiente de producción de aplicaciones. Como consecuencia, se requiere que los desarrolladores deban asimilar su API cada vez que necesiten desarrollar una nueva aplicación o simplemente incorporar el dispositivo a una aplicación existente. La naturaleza heterogénea de las API de los dispositivos puede provocar cambios significativos en la arquitectura de una aplicación, si su interfaces no son compatibles. Un ejemplo de esa variedad lo encontramos en los guantes de datos, como se ilustra en la figura 1.8.

Cuando se trata de los dispositivos de entrada de datos o periféricos no convencionales, se debe tener en cuenta que las empresas que se dedican a este sector de tecnología no son tantas. No obstante, para describir su diversidad, se puede afirmar que existen al menos tantas como tipos diferentes de dispositivos

Figura 1.8. Diferentes modelos de guantes ofrecen distintas tecnologías y número de sensores.

y tecnologías se encuentran en el mercado. Cada uno de estos dispositivos tiene sus propias ventajas y desventajas, por lo que no se puede decir que exista uno que sea el mejor para todas las aplicaciones y sí en un contexto bien definido. Por otro lado en algunos aspectos estos suelen quedar desfasados en el tiempo frente a las tecnologías emergentes en el mundo de los ordenadores. Sin embargo, como los productos suelen ser costosos, la renovación de los dispositivos antiguos y la adquisición de modelos innovadores no siempre es factible en términos económicos. Foley et al., 1996 reconoce que no todo diseñador de interfaces de usuario tiene el lujo de seleccionar el dispositivo más apropiado, pues en ocasiones la elección está hecha de antemano.

La comunidad científica realiza esfuerzos continuamente para resolver los inconvenientes antes expuestos. La avanzada en este aspecto la lleva la investigación en el campo de la interacción multimodal. En este campo existen resultados tales como metodologías de interacción, arquitecturas y bibliotecas de *software* que intentan resolver este problema desde la perspectiva del diseño de capas de abstracción de dispositivos, agrupados según sus características (Almeida, Silva y Teixeira, 2014; Duque, De la Rosa y Hernandez, 2013; Martínez et al., 2011).

1.4. Bibliotecas para el acceso uniforme a la información de los dispositivos de interacción

El proceso de desarrollo de una aplicación de RV, necesita una etapa donde se debe planificar cómo será la interacción con el usuario. En el momento en que se decide incorporar tareas de interacción con el usuario, los programadores siempre tendrán que crear los nuevos estilos de interacción de forma artesanal (Vilar, 2010). Aunque para crear interfaces 3D, si bien es cierto que el diseñador no tiene tantas ayudas como en el caso de interfaces 2D, no es menos cierto que actualmente existe un buen número de bibliotecas de programación que facilitan esta actividad.

En lo que concierne a la visualización de los contenidos 3D y en cómo estos se presentan al usuario, es usual recurrir a bibliotecas de gráficos 3D con tres clasificaciones diferentes. Estas pueden ser una interfaz estándar para el *hardware* de gráficos, depender del sistema operativo o ser independientes de la plataforma. Ejemplo de estas bibliotecas son OpenGL, Direct3D en los sistemas Windows de Microsoft y Java3D de Oracle respectivamente. Además de representar las primitivas gráficas en la pantalla, también es preciso

gestionar las estructuras de datos que guardan en memoria la escena virtual. Esta tarea se simplifica con el empleo de bibliotecas de más alto nivel, tales como Ogre3D, OpenSceneGraph o Irrlicht que facilitan el uso de grafos de escena (Engine, 2008; Junker, 2006; Osfield, Burns et al., 2004).

No obstante, además del *hardware* de gráficos, las interfaces 3D suelen involucrar otros dispositivos de entrada. Estos dispositivos con frecuencia requieren el uso de APIs específicas para cada uno de ellos. Sin embargo, existen bibliotecas que tratan de unificar diferentes dispositivos bajo una misma interfaz con mayor o menor grado de flexibilidad. Este es el caso de **OIS** (pjcast, 2013), una de las pioneras en este sentido. Se puede decir que su desarrollo se ha visto estancado e incluye solo unos pocos tipos de dispositivos.

De esta forma las bibliotecas de *software* proporcionan capas de abstracción para formar la arquitectura, de modo que sólo hay que preocuparse por escribir el código de la aplicación. Ejemplos de estas bibliotecas para RV son VR Juggler y DIVERSE, y para RA cabe citar ARToolkit y Studierstube, estas dos últimas se emplean en la creación de dispositivos de seguimiento ópticos para la interacción en un ERV (Bierbaum et al., 2001; Kato, 1999; Kelso et al., 2003; Schmalstieg et al., 2002). Estas bibliotecas y otras se analizarán con mayor detalle en la sección 1.5. También se analizarán los conceptos fundamentales que han surgido con la introducción de tales bibliotecas de *software*, como parte de la unificación del acceso a los dispositivos de interacción 3D.

1.5. Análisis de los trabajos relacionados

Desde el punto de vista de la HCI, las plataformas de RV han proporcionado soporte a la interacción con el usuario según su surgimiento. De forma general se han realizado esfuerzos por incorporar la mayor cantidad posible de dispositivos de interacción para experimentar distintas estrategias de interacción con el usuario según el contexto de la aplicación. Esta ha sido la estrategia utilizada en el desarrollo de bibliotecas como DIVE o MASSIVE (Frécon y Stenius, 1998; Greenhalgh, Purbrick y Snowdon, 2000). Estas proveen soporte integrado para los dispositivos de RV más populares en un bajo nivel de abstracción.

No obstante, otros como MR Toolkit, avanzan un paso en la gestión de los dispositivos de interacción (Shaw et al., 1993). Los desarrolladores de esta biblioteca, además de incorporar dispositivos de RV a medida que van surgiendo, también incluyen el concepto de simulación desacoplada. La principal ventaja introducida por este concepto es que la gestión de los dispositivos y la simulación de la RV pueden ser ejecutadas en ordenadores separados. Este aspecto es especialmente útil cuando existen procesos que incrementan el consumo de recursos computacionales, por ejemplo cuando existen procesos de seguimiento posicional mediante visión por computadoras. Sin embargo, esto no soluciona la limitación que subyace en la complejidad inherente de que el programador debe manejar las diferentes interfaces de los dispositivos para acceder a la información brindada por estos.

En el caso de herramientas como VR Juggler y VRPN se soluciona esta limitación (Taylor II et al., 2001). Además de proporcionar soporte para muchos dispositivos y un modelo de simulación desacoplado, introducen el concepto de *dispositivo de interacción abstracto*. Su diseño arquitectónico define un conjun-

to de requerimientos o funcionalidades necesarias para cada abstracción de dispositivos. Las aplicaciones, se implementan empleando la definición de un conjunto de dispositivos abstractos, para recoger la entrada del usuario y ofrecer la respuesta adecuada. Esta nueva forma de gestionar los dispositivos de interacción, proporciona la ventaja de que se pueda utilizar cualquier conjunto de dispositivos concretos en una aplicación en tiempo de ejecución. Esto sucede siempre y cuando estos dispositivos concretos satisfagan los requerimientos de abstracción de los dispositivos de entrada.

Sin embargo, en todos estos enfoques, aún es necesario adaptar la información que llega desde los dispositivos a los requerimientos de la aplicación. Esta adaptación forma parte de la lógica de interacción del usuario con el *software* y puede ser la aplicación de filtros de suavizado o algún algoritmo en específico. El proceso de adaptación de la información puede ser catalogado como una etapa de mayor nivel de abstracción y se puede decir que se encuentra enfocado en cómo esa información va a ser utilizada. Este es un aspecto de suma importancia luego de que se haya logrado la comunicación con los dispositivos y que se debe tener en cuenta a la hora de desarrollar la interacción en los SRV .

La propuesta de otras plataformas, tales como InTML y OpenTracker, ha sido parte de la solución para procesar la información que llega desde los dispositivos (Figueroa, Green y Hoover, 2002; Reitmayr y Schmalstieg, 2001). Estos sistemas utilizan una notación gráfica basada en el flujo de datos para describir tanto los dispositivos de interacción, como los algoritmos que procesan los datos de los dispositivos. El uso de esta notación gráfica de flujo de datos incluye algunos beneficios, como la mejora de la comprensión de la interacción, representada como un diagrama de flujo o la posibilidad de reutilizar los mismos componentes en otros diagramas. Estos dos factores estimulan el prototipado rápido de aplicaciones y la colaboración de los desarrolladores con expertos de otros dominios, por ejemplo expertos en HCI. Otro elemento a destacar es que InTML posee un *software* de edición visual para su notación gráfica, mientras que OpenTracker no posee esta característica.

Sin embargo, estas bibliotecas a pesar de tener los beneficios mencionados, adolecen de funcionalidades que podrían ser deseables en el nivel de abstracción que proponen. Por ejemplo InTML no proporciona una plataforma de ejecución para generar un código ejecutable a partir de la diagramas producidos. Por otra parte OpenTracker, proporciona una plataforma de ejecución tal que lee la configuración generada desde un archivo XML, pero tiene como inconveniente que no proporciona un editor gráfico de los diagramas. Además, esta plataforma a pesar de ser una opción tentadora, presenta periodos de actividad de desarrollo inestables. En el momento en que se efectúa esta investigación se hizo una actualización a la versión 2.0 donde proveen integración con la biblioteca VRPN para extender la funcionalidad de abstracción de dispositivos.

Otras plataformas, tales como NiMMIT y la IFFI, extienden sus conceptos de abstracción y no solo se dedican a gestionar los dispositivos, sino que añaden conceptos y funcionalidades para procesar sus datos (De Boeck et al., 2007; Ray y D. A. Bowman, 2007). Se inspiran en el uso de modelos de flujo de datos para describir tareas de interacción y proporcionan una plataforma de ejecución para crear sus diagramas y transformarlos en código ejecutable. Otras plataformas, como CHASM o StateStream, extienden este concepto, mezclando las gráficas de flujo de datos con los modelos basados en estados, con el fin de brindar un mejor

soporte para los sistemas basados en eventos (Haan y Post, 2009; C. Wingrave y D. Bowman, 2005). Sin embargo, estos enfoques tienden a sobrepasar los límites de las tareas de interacción. Además, en (Martínez et al., 2011) se presenta todo un marco de trabajo de interacción con RV que incluye los conceptos antes mencionados, para brindar tanto a desarrolladores como expertos en HCI una experiencia más completa en el desarrollo rápido de prototipos de aplicaciones de MHCI.

En (Martínez et al., 2011) se propone un marco de trabajo para desarrollar técnicas de interacción para RV, basado en la biblioteca OpenInterface y AFreeCA (Martinez et al., 2010; Serrano et al., 2008). Esta propuesta tiene como objetivo proporcionar un diseño extensible de código abierto y un marco de trabajo para brindar soporte al desarrollo rápido de sistemas de interacción multimodal. En él se ofrece un alto nivel de abstracción, que permite a los investigadores enfocarse solamente en la creación de nuevos métodos de interacción.

Estas características pueden ser útiles si el objetivo que se persigue es independiente de OpenInterface o si se pretende utilizar la plataforma en su totalidad. Sin embargo, puede acarrear desventajas en un contexto en el que exista un marco de trabajo preestablecido para el desarrollo de aplicaciones, con un conjunto de herramientas y su respectiva arquitectura predefinida. En este caso, lejos de ser un beneficio, puede constituir un inconveniente en cuanto a criterios de flexibilidad, por ejemplo cuando se desea integrar con aplicaciones ya establecidas. Esta situación se debe a que OpenInterface provee una plataforma de ejecución muy acoplada con sus herramientas de trabajo. Es preciso mencionar, que si se presenta esta situación, habría que tomar precauciones para hacer modificaciones a la arquitectura y adaptarla a OpenInterface.

Las herramientas analizadas en la presente sección han incorporado características de las anteriores a medida que han surgido. Sin embargo, existen casos en los que esto no sucede exactamente así. Por ejemplo InTML, OpenTracker, NiMMIT, IFFI, CHASM y StateStream incorporan dispositivos de forma masiva, poseen características de alto nivel (procesamiento de datos, plataformas de ejecución), pero no incorporan conceptos como la simulación desacoplada y la abstracción de dispositivos. Estas características no incorporadas, son de vital importancia a la hora de desarrollar un *software* de RV con varios ordenadores distribuidos y separados por varias funciones.

El análisis de las características de los trabajos tratados en esta sección, permitió identificar un conjunto de propiedades a considerar, para cumplir el objetivo de la investigación. Las propiedades, identificadas son:

(A) Soporte masivo de dispositivos

(B) Simulación desacoplada

(C) Abstracción de dispositivos

(D) Notación gráfica (Lenguaje de descripción)

(E) Edición visual

(F) Plataforma de ejecución

(G) Soporte de técnicas y tareas básicas de interacción

En función de las características identificadas se realizó una clasificación ubicándolas en dos grupos principales, características de bajo nivel y de alto nivel. En el conjunto de bajo nivel se ubicaron aquellas características relacionadas solamente con la gestión de los dispositivos y la obtención de sus datos. En el conjunto de características de alto nivel, se ubicaron todas aquellas que se enfocan en la interpretación y representación de los datos obtenidos de los dispositivos. Como las bibliotecas y plataformas analizadas presentan características de uno y otro grupo respectivamente, esta clasificación permitió un mejor análisis a la hora de tomar la decisión de cuál reutilizar para plantear la solución.

En la tabla 1.2, se resumen las características identificadas en los trabajos previos, así como su clasificación según los criterios de bajo y alto nivel de abstracción.

Tabla 1.2. Resumen de propiedades de los trabajos analizados.

Bibliotecas ⧹ Propiedades	**Bajo Nivel**			**Alto Nivel**			
	A	**B**	**C**	**D**	**E**	**F**	**G**
DIVE	Si	No	No	No	No	No	No
MASSIVE	Si	No	No	No	No	No	No
MRToolkit	Si	Si	No	No	No	No	No
VRJuggler	Si	Si	Si	No	No	No	No
VRPN	Si	Si	Si	No	No	No	No
InTML	Si	No	No	Si	Si	No	No
OpenTracker	Si	No	No	Si	No	Si	No
NiMMIT	Si	No	No	Si	Si	Si	No
IFFI	Si	No	No	Si	Si	Si	No
CHASM	Si	No	No	Si	Si	Si	No
StateStream	Si	No	No	Si	Si	Si	No
OpenInterface +AFreeCA	Si	Si	Si	Si	Si	Si	Si

Además de la información obtenida en la tabla 1.2, la selección de la biblioteca estuvo basada según dos criterios adicionales. Los criterios fueron la estabilidad en su desarrollo y la posibilidad de reutilizarla en el marco de trabajo de los grupos ViViRG e IDEASCAD. Fue necesario que la biblioteca tuviese una comunidad de desarrollo activa, la posibilidad de ser reutilizada como un componente separado y que cumpliera con todas las características de bajo nivel (debido al alcance de la investigación). De esta forma la biblioteca que cumplió con estas condiciones fue VRPN, porque VRJuggler tiene menos actividad en su desarrollo, al igual que OpenInterface+AFreeCA. Además OpenInterface+AFreeCA plantea un modelo de desarrollo muy acoplado a su plataforma de ejecución, que lo convierte en una solución poco flexible para incorporarlo al marco de trabajo antes mencionado. En este sentido VRPN tiene un modelo de desarrollo flexible, en el que ofrece componentes de código abierto para ser reutilizados en cualquier proyecto de *software* dentro del contexto de la HCI. Finalmente VRPN es una biblioteca que cumple con un conjunto de características identificadas como deseables en la bibliografía. Las características identificadas a partir del análisis de las

bibliotecas anteriores son: abstracción de dispositivos, transparente a la red, baja latencia, sencillo de integrar y extensible (Taylor II et al., 2001). No obstante esta biblioteca no tenía incorporada la estrategia de seguimiento utilizada en los trabajos del grupo ViViRG, por lo cual debía ser adaptada.

1.6. Consideraciones parciales

El contexto teórico analizado en este capítulo, permitió identificar la gestión de dispositivos como un subcampo dentro de la interacción 3D. La caracterización de las diferentes clasificaciones de dispositivos de interacción, permitió seleccionar las características específicas a tener en cuenta para la solución. Esto constituyó la base para desarrollar interfaces genéricas según las funcionalidades de los dispositivos y garantizar el acceso a sus respectivos datos. El análisis de los trabajos relacionados permitió seleccionar a VRPN como la biblioteca más indicada a reutilizar, para formar parte de la solución al problema de investigación y dar cumplimiento al objetivo de la misma.

CAPÍTULO 2

Propuesta de solución

2.1. Introducción

En el presente capítulo se explica de forma detallada la propuesta de solución al problema de la investigación. Inicialmente se describe la concepción genérica de un SRV con la estructura básica utilizada por los grupos ViViRG e IDEASCAD, donde se pretende incluir la solución propuesta. A continuación se describe cómo se distribuyen los dispositivos según las características de bajo nivel presentadas en la tabla 1.2. Luego se hace una descripción general de las características de la biblioteca VRPN utilizada para la solución. Finalmente se describe la implementación de la solución propuesta.

2.2. Descripción de la solución

Durante el desarrollo de un SRV el foco de atención debe girar entorno a que la simulación e interacción cumplan con los requisitos específicos de la aplicación. Pues de ello depende que el entorno de RV sea mas interactivo y provoque una sensación de inmersión en el usuario. La experiencia demuestra que el proceso de lidiar con las interfaces de renderizado y de los dispositivos suele requerir mucho esfuerzo de los desarrolladores. Esto provoca el aumento del tiempo de desarrollo y los desvía del plan original de la solución de interacción, que pretenden ofrecer al usuario. Por tal motivo los marcos de trabajo para SRV facilitan una vista abstracta del sistema y de los dispositivos de entrada y salida, pues ofrecen componentes con un alto nivel de abstracción, lo cual reduce el esfuerzo necesario para crear los ERV.

Los elementos básicos del renderizado, están resueltos con el surgimiento de las bibliotecas de abstracción del *hardware* gráfico, como OpenGL y DirectX, además de otras de mayor nivel de abstracción, como Ogre3D. Una ventaja es que los investigadores pueden concentrarse en el desarrollo de otros temas, como la extensión de interfaces multisensoriales(Chardonnet y Leon, 2012; Luque et al., 2014) y avanzar más allá de las técnicas básicas de interacción para proveer metáforas de interacción intuitivas y mejorar la interacción de los usuarios con los SRV.

En la figura 2.1 se muestra una vista conceptual de la estructura genérica utilizada por los SRV desarrollados por los grupos ViViRG e IDEASCAD. En esta figura se se describe cómo las diferentes capas interactúan entre sí para producir finalmente la aplicación de RV. El componente principal es el manejador de la escena de RV, en el cual se gestionan los elementos o entidades gráficas de la escena, así como la lógica de la aplicación que se pretende desarrollar. Este componente interactúa con los demás componentes para obtener los parámetros necesarios (inteligencia artificial, física, sonido, video) y así generar las respuestas de la aplicación. Al mismo tiempo el usuario interactúa con la aplicación y recibe la representación de la información mediante la capa de interfaz de usuario, que constituye el punto de entrada de información para el usuario.

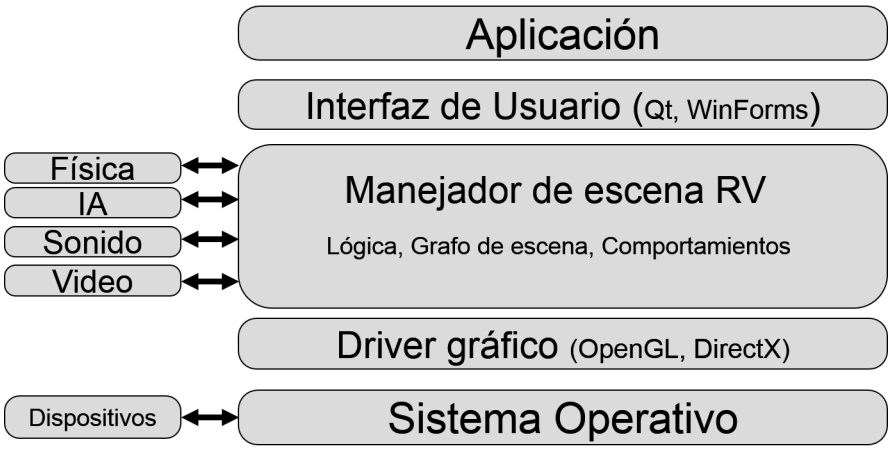


Figura 2.1. Modelo conceptual de un sistema de realidad virtual.

Un elemento importante en este esquema es que los dispositivos se comunican con el SRV de forma local, en la misma estación donde se encuentra desplegado. La cuestión del esquema reside en que los dispositivos están fuertemente acoplados al mismo y puede resultar una desventaja cuando el número de dispositivos aumenta hasta superar la capacidad de conexión de la estación de trabajo. Otra desventaja es que el API está fuertemente acoplada a la solución. Esto provoca que si se quiere intercambiar ese dispositivo por otro que ofrezca los mismos datos, pero con un API diferente habría que redefinir la comunicación con el dispositivo en el sistema, lo cual aumenta el esfuerzo del equipo de trabajo.

La figura 2.2 ilustra con mayor claridad el proceso de comunicación de los dispositivos de interacción con el SRV descrito en la figura 2.1. Los dispositivos se conectan directamente al SRV y los desarrolladores deben ser capaces de manejar las funcionalidades intrínsecas de cada una de sus respectivas APIs, en las distintas etapas de la aplicación. Este proceso se debe realizar por cada aplicación distinta que se desee desarrollar, si se tiene en cuenta que el equipo de desarrollo no siempre va a coincidir y que en cada solución pueden utilizarse dispositivos distintos. A partir de esta situación y en virtud del análisis realizado en la sección 1.5, se decide incorporar una nueva capa al esquema actual descrito en la figura 2.1, para el acceso a los dispositivos. Esta solución se ilustra de forma conceptual en la figura 2.3.

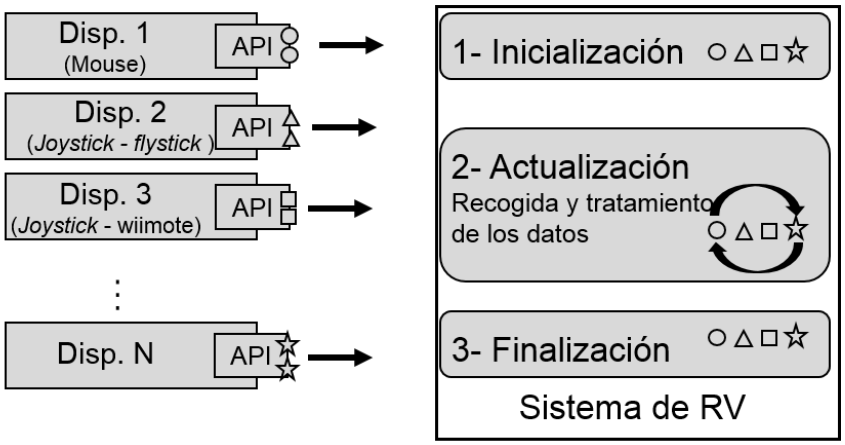


Figura 2.2. Comunicación de los dispositivos con el SRV, mediante sus respectivas APIs.

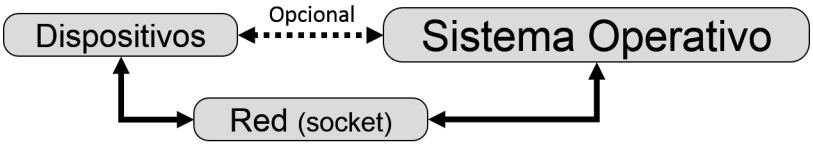


Figura 2.3. Capa de abstracción de los dispositivos añadida en la solución.

La capa adicionada al esquema existente, incorpora de forma inmediata el concepto de simulación desacoplada, al poder recuperar los datos de los dispositivos desde una ubicación remota. Esto le ofrece la ventaja de poder disminuir la carga de trabajo de la estación donde corre el SRV. Además, permite recuperar recursos computacionales para ser utilizados según las exigencias de una buena simulación. En la figura 2.4 se ofrece una visión más detallada de la solución que se brinda en la figura 2.3.

En este nuevo esquema los dispositivos se comunican con el SRV a través de una aplicación de *software* que hace de servidor. El servidor es el encargado de comunicarse con los dispositivos a través de sus respectivas APIs, justo como se describe en la figura 2.2. Cuando se obtienen los datos necesarios de los dispositivos, se envían a los clientes suscritos. Los datos viajan con un formato previamente establecido y estos a su vez actúan como capa de abstracción para el SRV. El formato es una especificación de cómo deben enviarse los datos, de acuerdo al tipo de dispositivo que los genera, según la clasificación descrita en la sección 1.3.2. Finalmente cuando la aplicación termina el proceso de ejecución, la aplicación cliente envía una señal de desconexión. Los recursos ocupados por los dispositivos son liberados cuando termina la ejecución del servidor.

La forma en que se obtienen los datos de los dispositivos con el esquema propuesto, ofrece la ventaja de que el acceso a las APIs se hace una sola vez (en el momento que el servidor lo encuesta). A partir de ese momento y en virtud del formato de los datos, el dispositivo pasa a formar parte de una capa de abstracción que oculta al cliente las singularidades de su APIs. Como los datos se envían a los clientes según la categoría en la que se encuentra clasificado el dispositivo, entonces el desarrollador solo se tiene que preocupar por

dominar el API cliente, el cual va a ser el mismo para cada categoría de dispositivos.

Mediante el uso del nuevo esquema, los desarrolladores y a su vez las aplicaciones desarrolladas por ellos, van a tener una capa intermedia que garantiza la transparencia en el acceso a los dispositivos. La capa intermedia ofrece la ventaja, de que no necesitan preocuparse si los dispositivos están ubicados localmente en la estación del SRV o si están en una estación remota. Por otro lado en el equipo de desarrollo aparecen dos actividades bien definidas, una en cuanto al mantenimiento del servidor, para incorporarle nuevos dispositivos y la otra para los usuarios del cliente que utilizan una interfaz común para acceder a los dispositivos.

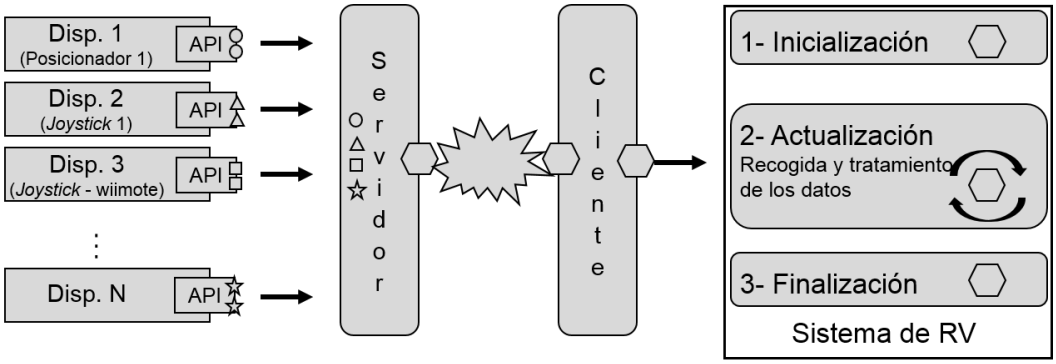


Figura 2.4. Comunicación de los dispositivos con el SRV, mediante la capa de abstracción.

En las próximas secciones se describe de forma detallada cómo se implementó esta solución.

2.3. Arquitectura de VRPN

Se puede decir que VRPN es una biblioteca conformada por un conjunto de clases, diseñadas para implementar una interfaz de comunicación con los dispositivos, a través de una arquitectura cliente-servidor. Su funcionalidad principal radica en la posibilidad de crear servidores para comunicar los datos de los dispositivos físicos con las aplicaciones mediante una interfaz transparente a la red. Propone una clasificación de dispositivos basada en los datos que estos ofrecen (de seguimiento 3D, botones, dispositivos de fuerza, entradas analógicas, sonido y texto). Esta clasificación permite crear conexiones entre las aplicaciones clientes y los dispositivos según el tipo de servicio de datos apropiado para cada dispositivo. La ventaja es que la aplicación permanece ajena al tipo de topología utilizada en la red, incluso bajo este esquema los dispositivos pueden estar conectados a la PC de forma local utilizando programas separados (cliente-servidor) o embebidos en una misma solución.

La clasificación utilizada por VRPN permite que los dispositivos que sean de una misma clase se vean como si fueran un mismo dispositivo, sin tener en cuenta los detalles de su API o el fabricante que los produjo, por ejemplo los dispositivos de *seguimiento* siempre van a tener una clase base vrpn_Tracker. Esto se traduce en que todos los dispositivos de seguimiento producen el mismo tipo de reporte.

La flexibilidad presente en VRPN permite además adaptar las funcionalidades de un tipo de dispositivo específico. Por ejemplo si una aplicación requiere características específicas de un tipo de dispositivo de

seguimiento (aumentar la frecuencia de reporte de los sensores), estas se pueden adaptar a partir de su respectiva clase base. En el caso de que la clase especializada sea utilizada con un dispositivo de seguimiento que no comprenda los comandos especializados, estos serán ignorados.

2.3.1. Caracterización de los dispositivos

La biblioteca VRPN presenta una forma muy particular de incorporar el soporte de dispositivos. Los dispositivos son tratados de forma tal que se evita ofrecer una interfaz en la que no se tengan en cuenta las características específicas de un dispositivo determinado. Al mismo tiempo, también se evita ofrecer una interfaz tan compleja que incorpore todas las características específicas de cada dispositivo que se desee adicionar. Esto solo conduce a la proliferación de casos aislados con el consiguiente aumento de la complejidad de la solución.

El modelo de implementación de VRPN está dado por los siguientes cinco criterios:

- Descomposición de los dispositivos basado en sus funcionalidades
- Asignación de los dispositivos a conexiones dentro del ámbito de VRPN
- Habilitar la exposición de interfaces múltiples
- Ignorar los tipos de mensajes no soportados
- Proveer acceso a todos los mensajes a nivel de aplicación

La descomposición de la interfaz según las funcionalidades, implica que por cada característica especifica de un dispositivo se necesite implementar una funcionalidad distinta. En este caso el controlador de un dispositivo para VRPN expondrá al usuario varias interfaces equivalentes a sus respectivos tipos de dispositivos. Por ejemplo el servidor para el dispositivo Wiimote de Nintendo es un caso típico de la situación descrita en este párrafo. El Wiimote presenta botones, sensores de aceleración y giro, así como un dispositivo de seguimiento infrarrojo. Por otro lado los clientes que decidan comunicarse con este servidor lo hacen como si fueran tres dispositivos distintos, uno por cada función. Esto otorga a VRPN la ventaja de que los clientes no necesitan modificar el código para comunicarse con otro dispositivo distinto mientras se mantengan las mismas funcionalidades. Es preciso aclarar que este nuevo dispositivo puede ser la composición de otros 3 dispositivos distintos, mientras se mantengan las funcionalidades anteriores.

Los dispositivos pueden compartir una misma conexión. Esto brinda la ventaja de poder componer varias funcionalidades lógicamente separadas, con una sola conexión de red. El propósito de esta estrategia es lograr eficiencia en la comunicación de los datos. Por ejemplo en el caso del Wiimote, como este dispositivo posee tres tipos de funcionalidades distintas, los datos se transmiten utilizando un solo objeto conexión vrpn_Connection.

Mientras tanto la exposición de interfaces múltiples habilita a un dispositivo para que se comporte como otros tipos diferentes de dispositivos. Un dispositivo que tenga un botón de giros se puede comportar

como un dial para dar un ángulo de orientación. Este también se puede comportar como un dispositivo del tipo analógico, cuyo valor puede ser utilizado para otras interpretaciones, como la de subir el volumen del sonido. Al utilizar esta funcionalidad, dos clientes se pueden conectar al mismo servidor y utilizar un mismo dispositivo con interpretaciones semánticas diferentes.

Las funcionalidades que son comunes a varios dispositivos del mismo tipo son definidas como parte de un dispositivo base. Esto implica que los controladores de dispositivos derivados puedan implementar esa funcionalidad como una interfaz común de manera opcional. Este criterio permite extender la interfaz común y habilitar el acceso a funcionalidades específicas de algunos dispositivos para mensajes que puedan ser ignorados de forma segura. Como consecuencia directa no es necesario implementar todas las funcionalidades en todos los dispositivos.

El acceso a los mensajes en el nivel de aplicación ofrece una flexibilidad muy útil. Por ejemplo, existen algunos dispositivos que pueden poseer funcionalidades específicas, las cuales no son contempladas por ningún otro dispositivo dentro de su misma clase y que no son aplicables a ninguna de las otras clases de dispositivos. En esta situación particular el dispositivo puede enviar y recibir tipos de mensajes con una codificación propia, para brindar acceso a funcionalidades extendidas sin tener que modificar el código de las clases de la biblioteca. El servidor implementado para este tipo de dispositivo tendría que brindar los manipuladores de mensajes y utilizarlos para comunicarse haciendo uso de la nueva funcionalidad. De esta forma no es necesario modificar la interfaz común de su clase de dispositivo.

2.3.2. Manejo de conexiones

VRPN maneja las comunicaciones entre los clientes y su respectivo servidor a través un objeto "*conexión*". Las conexiones están diseñadas de tal forma que cumplen con las siguientes características:

- Inicio rápido para conectarse a un servidor
- Cuando se establece una conexión, se garantiza el retorno rápido al flujo principal del cliente, aún cuando el servidor no este corriendo
- No depende de puertos TCP preestablecidos en el servidor, ni en el cliente (los puertos se eligen de forma automática)
- Se garantizan los intentos de reconexión minimizando los tiempos de espera en el cliente
- La conexión o reconexión de forma rápida a un servidor que haya dejado de funcionar. Esto garantiza que se pueda reiniciar un servidor sin tener que reiniciar los clientes

Para la creación de una conexión VRPN es necesario crear al menos dos pares: un objeto conexión de escucha, que actúa como servidor y otro conexión remota que actúa como cliente. En el momento que estas conexiones se crean se inicializa la comunicación y el paso de mensajes hasta que uno de los dos pares decide terminar la conexión. En las próximas subsecciones se describe este proceso para establecer la base del funcionamiento de la solución.

Establecer una conexión de escucha (servidor)

Para crear una conexión del tipo servidor lo primero que se debe hacer es crear un objeto conexión, lo cual se realiza mediante la sobrecarga de la siguiente función:

```
vrpn_Connection * connection = vrpn_create_server_connection();
```

La función anterior levanta una conexión que va a escuchar solicitudes en un puerto UDP, el cual puede ser especificado de forma opcional. En caso de que no se le especifique un puerto, se utiliza uno por defecto. A partir de ese momento la conexión hace una comprobación de los paquetes entrantes en este puerto cada vez que se haga un ciclo de actualización. La implementación de VRPN permite múltiples conexiones remotas a un servidor de forma simultánea, enviando el mismo mensaje a cada uno de los clientes. Es válido mencionar que la información que viene de un cliente no es retransmitida a los demás.

Establecer una conexión remota (cliente)

En el lado del cliente, la aplicación no pregunta por una conexión de forma directa, son los dispositivos los que se las arreglan para obtener su propio objeto conexión, lo cual se realiza mediante la función:

```
vrpn_Connection * vrpn_get_connection_by_name (const char * cname, ...);
```

Cuando la aplicación cliente gestiona su conexión utiliza la función anterior y como como primer parámetro utiliza una cadena del tipo "`dispositivo@url`". Por ejemplo para un dispositivo de seguimiento, que está configurado con el identificador "`Tracker0`" en la dirección "`d5l308pc27.uci.cu`", la cadena de conexión sería "`Tracker0@d5l308pc27.uci.cu`". Con esta información la función se encarga de gestionar su propia dirección URL y abrir un puerto para la conexión. En el caso de que la aplicación abra más de un dispositivo con la misma conexión, entonces se devuelve un apuntador al objeto de conexión. Esto es especialmente útil cuando las aplicaciones crean más de un dispositivo conectado al mismo servidor.

Iniciar y detener una conexión en VRPN

Las conexiones siempre son inicializadas por las aplicaciones del lado cliente. El cliente abre un *socket* TCP para escuchar y envía un datagrama UDP a un puerto predeterminado, con la información del nombre de la máquina y el número de puerto del *socket* TCP abierto. En caso de que no haya ningún intento de conexión dentro de un período de tiempo de espera pequeño, se realiza otra solicitud. Si ocurre que varias peticiones fallan, se supone que el servidor está caído o ya existe una conexión activa, por tanto el intento de conexión falla. Este mecanismo permite realizar la conexión con el servidor o la detección de los fallos de forma rápida. Una vez que la conexión de escucha en el servidor recibe la solicitud UDP para la conexión, se establece un enlace TCP con el cliente.

Luego de establecido el enlace TCP, se sincronizan las versiones VRPN, ambas partes acuerdan abrir un nuevo puerto UDP y se envían la descripción a través de la conexión TCP previamente establecida. Cada parte establece la conexión mediante un nuevo enlace UDP. Esto permite una comunicación no orientada a

la conexión, que como consecuencia puede ofrecer menor latencia en el envío de los paquetes. En este punto del proceso, cada objeto de conexión envía a su par la información de su emisor y los tipos de mensajes que deben intercambiar.

La finalización de una conexión entre el servidor y los clientes ocurre cuando una de las partes recibe una excepción en una de las operaciones de lectura, escritura o selección. La operación de selección se realiza cada vez que hay un ciclo de actualización en la lectura de los datos de los dispositivos a través del enlace TCP del objeto conexión. Esto brinda la ventaja de la detección temprana del fallo de conexión. Cuando la conexión se interrumpe, en el servidor comienza nuevamente el proceso de inicialización y en el cliente se dejan de enviar los mensajes.

2.4. Solución propuesta

En este apartado se describe cómo se realizó la propuesta de la solución con un modelo cliente-servidor.

2.4.1. Estructura del servidor

VRPN proporciona clases canónicas para definir varios tipos de interfaces genéricas, según la clasificación descrita en la sección 2.3.1. Cada clasificación de dispositivo se representa con una clase cliente y una servidora que se nombran según una nomenclatura predefinida. En el caso de las clases del lado del servidor la nomenclatura se describe de la forma vrpn_TipoDispositivo. El identificador de las clases del lado del cliente se compone de la forma vrpn_TipoDispositivo_Remote.

Los dispositivos son mapeados en uno o varios tipos de interfaces en correspondencia con la clasificación de VRPN. Los tipos de interfaces genéricas consisten básicamente en: *tracker/dispositivo de seguimiento*, *button/botón*, *analog/dispositivo analógico*, *dial/control deslizante* y *force_device/dispositivo de fuerza*. En el lado del servidor cuando se adiciona un dispositivo concreto, que implementa alguna de las interfaces genéricas, se debería llamar según la convención *vrpn_TipoDispositivo_Nombre*. El nombre sería el que recibe según el dispositivo físico que representa, por ejemplo *vrpn_Tracker_Isense* para los dispositivos de seguimiento *Inertia isense* (I. I. Inc., 2014).

Finalmente, las clases del cliente, así como las del servidor definen una interfaz con los métodos que deben ser llamados para que exista la comunicación. Es importante destacar que mediante la composición se puede definir un dispositivo con complejidad superior, que contemple varios tipos de datos o varios dispositivos para que funcionen como si de un solo dispositivo compuesto se tratase. Los dispositivos pueden reportar varios tipos de información, como el *Nintendo Wiimote*, que ofrece datos de *tracking*, acelerómetros y botones. La implementación de la composición de los dispositivos combina varios tipos de clases canónicas para representar su información.

Adaptación del servidor

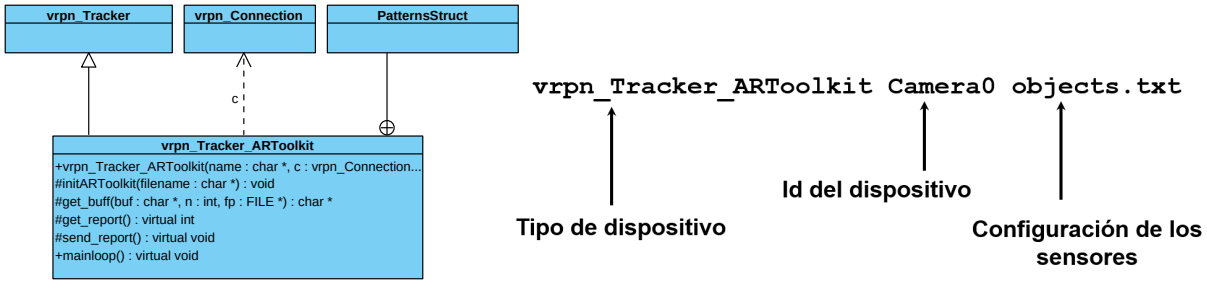
El servidor implementado brinda soporte a dispositivos tradicionales y se le incluye el soporte a un nuevo dispositivo de seguimiento 3D basado en visión por computadoras. Los algoritmos de visión por computadoras utilizados siguen la estrategia de seguimiento propuesta por ATRToolKit, la cual ha sido utilizada por los desarrolladores de los LV para probar otros métodos de interacción. Esta biblioteca tiene la ventaja de ofrecer técnicas de seguimiento 3D de bajo costo, a través de la detección de marcadores cuadrados mediante el uso de *webcams*.

La adaptación se realizó a partir de un conjunto de reglas predefinidas. Se definió una clase derivada de las clases genéricas del servidor, para acceder a los datos del nuevo dispositivo. La clase base proporciona los métodos para manejar la conexión, el mensaje donde viajan los datos y el registro de los tipos de mensajes. Además, proporciona los métodos para empaquetar y desempaquetar los mensajes, de esta forma se asegura que la clase derivada implemente las mismas interfaces.

La implementación de una instancia de servidor genérico permite la creación de un servidor para casi cualquier dispositivo conectado a la computadora. Mediante esta instancia se crean otras instancias que corresponden a los dispositivos especificados en un archivo de configuración. El archivo describe el tipo de dispositivo, su identificador y algunos parámetros de configuración que el dispositivo necesite para la ejecución. Cuando se inicia el servidor, se inicializan los dispositivos especificados en el archivo de configuración y comienza la etapa de actualización. Dentro de la etapa de actualización, el servidor llama a la función *mainloop()* de la instancia del servidor genérico. Este último ejecuta el bucle principal de cada dispositivo para recuperar sus datos, el cual es una función con el mismo nombre *mainloop()*.

El diagrama de clases resultante de la adaptación del vrpn_Tracker con la biblioteca ARToolKit se muestra en la figura 2.5a, a la izquierda. La creación de la nueva clase del lado del servidor sigue la convención especificada con anterioridad, vrpn_Tracker_ARToolKit. Se adiciona el método initARToolkit() que recibe como parámetro la dirección de un archivo, donde se encuentran especificado la distribución de los marcadores que se van a utilizar. Esta función prepara todo lo necesario para la sincronización de los marcadores (simples y compuestos) con los sensores que propone VRPN. Mediante el método mainloop() se recupera el estado actual de los marcadores (posición y orientación) y se envía a los clientes utilizando las funcionalidades del servidor.

A partir de la implementación de la clase vrpn_Tracker_ARToolKit, se realizan los cambios necesarios en el servidor para adaptar su funcionalidad. La adaptación implementada se incorporó como un nuevo dispositivo de seguimiento 3D. En la figura 2.5b a la derecha, se muestra la línea de configuración con los parámetros que se adiciona en el archivo de configuración del servidor. La línea de configuración se describe como sigue, vrpn_Tracker_ARToolkit es el nombre del dispositivo adicionado al servidor, *Camera0* en este caso es el identificador con el que se debe incorporar el dispositivo en el cliente y *objects.txt* es un archivo donde aparece la configuración de los marcadores que se van a utilizar como sensores para el seguimiento 3D. Luego resta configurar la aplicación del lado del cliente para que utilice el nuevo dispositivo.



(a) Diagrama de clase del dispositivo de seguimiento basado en ARToolKit.

(b) Línea de configuración de los sensores.

Figura 2.5. Diseño y configuración del dispositivo de seguimiento 3D.

Archivos de configuración

La implementación del servidor necesita una configuración en la que se describe cuáles son los dispositivos que se van a utilizar en la ejecución y los parámetros que va a utilizar cada uno respectivamente. Esta información se suministra a través del archivo *DalvirgSettings.cfg*, que se muestra en la figura 2.6. En este archivo se muestra la configuración de cuatro dispositivos utilizados en la investigación: Mouse, Wiimote, dispositivo de seguimiento NDI Polaris y dispositivo de seguimiento basado en ARToolKit, este último como parte de la adaptación de VRPN. En cada caso se siguió la convención *vrpn_TipoDispositivo_Nombre* especificada en la sección 2.4.1, seguida de los parámetros para cada dispositivo.

```
#Configuración del ratón
vrpn_Mouse  Mouse0

#Configuración para el Nintendo Wiimote
vrpn_WiiMote        Cursor3DBtn 1 1 1 1

#--Configuración de Polaris Vicra para software de IDEASCAD--
vrpn_Tracker_NDI_Polaris Tracker0 COM8 1
uniovi2011.rom
700339.rom
8700338.rom
#-------------------------------------------------------------

#Línea de configuración para Artoolkit Tracker
vrpn_Tracker_ARToolkit Cursor3DTkr objects.txt
```

Figura 2.6. Archivo de configuración de los dispositivos para el servidor.

El archivo *objects.txt* existe como consecuencia de la adaptación del servidor (sección 2.5a). Este es el que describe cuales van a ser los marcadores, de los cuales se debe enviar la información de posición y orientación y la información de la cámara. Cada uno de los marcadores se identifican dentro del contexto de VRPN como sensores de seguimiento y su identificador se asigna por orden de aparición comenzando desde cero hasta la cantidad de marcadores menos uno.

```
#Configuración de video
WDM_camera_flipV.xml

#Parámetros de la cámara
camera_para.dat

#Número de patrones a reconocer
1

#Sensor 0
cube
mouse3D.patt
40.0
0.0 0.0
```

Figura 2.7. Archivo de los marcadores en la adaptación del servidor.

2.4.2. Estructura del cliente

En lado del cliente es donde se aprovecha el potencial de la solución que se propone en este trabajo. Si aparece un nuevo dispositivo en el servidor, no es necesario realizar ningún cambio con respecto a la interfaz del dispositivo que está en el cliente. Esto es posible porque el dispositivo se ubica dentro de una de las clasificaciones de la sección 2.3.1 y así se garantiza que no se viole la interfaz de la clase genérica correspondiente al tipo de datos que envía el dispositivo. En el cliente existen interfaces que garantizan la comunicación con las clases que representan a los dispositivos en el servidor.

Para ilustrar la afirmación del párrafo anterior y poner en práctica la adaptación descrita en la sección 2.4.1, bastaría con crear una instancia genérica de un dispositivo de interacción de tipo seguimiento 3D. La información que varía es la dirección del servidor y el identificador del dispositivo, parámetros que se pueden controlar además mediante un archivo de configuración en el cliente. A partir de ese momento el cliente está listo para comenzar a recibir los datos. Un ejemplo de un programa cliente completo para el dispositivo implementado aparece en el listado de código de la figura 2.8.

```
1  #include "vrpn_Tracker.h"
2
3  void handle_transform(void *cdata, const vrpn_TRACKERCB t) {
4      printf("Sensor %d, Pos(%5.3f, %5.3f, %5.3f),Rot(%5.3f, %5.3f, %5.3f)\n",
5      t.sensor, t.pos[0], t.pos[1], t.pos[2], t.quat[0], t.quat[1], t.quat[2]);
6  }
7
8  int main() {
9      vrpn_Tracker_Remote *artTrkr = new vrpn_Tracker_Remote("Camera0@DeviceHost");
10     artTrkr->register_change_handler(NULL, handle_transform);
11     while (1) {
12       artTrkr->mainloop();
13       sleep(1);
14     }
15     return EXITSUCCESS;
16 }
```

Figura 2.8. Implementación básica del cliente para el dispositivo de seguimiento 3D de la sección 2.4.1.

Por otro lado las funcionalidades de los dispositivos se pueden agrupar mediante un objeto compuesto genérico, que permite crear asociaciones de distintos dispositivos como si fueran uno solo como se aprecia en la figura 2.9. De esta forma se pueden crear dispositivos compuestos con diferentes funcionalidades y utilizarlos como si fueran uno solo.

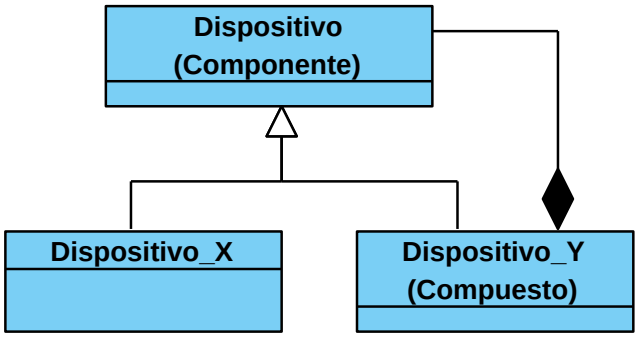


Figura 2.9. Implementación del cliente, como una composición de objetos.

2.5. Consideraciones parciales

La reutilización de funcionalidades de la biblioteca VRPN permitió implementar un servidor para el envío de los datos de dispositivos de interacción mediante una arquitectura cliente-servidor. Se adaptaron las funcionalidades de VRPN mediante la incorporación de un nuevo dispositivo de seguimiento 3D basado en visión por computadoras. La solución propuesta actúa como una capa de abstracción transparente a la red, que se le adiciona a la arquitectura de los sistemas de RV desarrollados por IDEASCAD y ViViRG.

CAPÍTULO 3

Evaluación de los Resultados

3.1. Introducción

En este capítulo se presenta la evaluación de la solución propuesta. Se diseña un componente de interacción 3D para un ejercicio del Laboratorio Virtual (LV) de Ensamblaje de un Ordenador. Luego se procede a comparar el esfuerzo de codificación que un desarrollador necesitaría, para implementar el componente. Se describen los resultados de implementación mediante el uso de la variante tradicional y mediante el uso de la solución propuesta. La comparación se establece a través de la métrica de estimación por líneas de código definida en Pressman, 1997a. Seguidamente se analiza cómo impacta el esfuerzo dedicado en el tiempo de desarrollo del proyecto durante la etapa de codificación, a través de distintos niveles de productividad. Finalmente se ofrece un análisis de cómo afecta la solución propuesta al rendimiento de la aplicación.

3.2. Componente de interacción: Cursor 3D

La validez de la solución propuesta en este trabajo se comprueba con la implementación de un prototipo de aplicación, donde se utilicen las funcionalidades del componente de interacción 3D presentado en esta sección. Se describe un caso de estudio sencillo en el que se combina la utilización de dos tipos de dispositivos, para interactuar con el LV de Ensamblaje de un Ordenador desarrollado en el centro VERTEX. A continuación se describen los dispositivos que se utilizaron para crear la solución de interacción, el entorno virtual y el entorno de ejecución.

3.2.1. Dispositivo de interacción

En la interacción hombre-máquina, existen casos en los que se ha utilizado un ratón 3D como dispositivo de interacción con 6 DOF en un SRV(Francese, Passero y Tortora, 2012; Gallo, De Pietro y Marra, 2008; Lawrence et al., 2013). En este trabajo se consideró diseñar un ratón 3D mediante la combinación de dos dispositivos, para demostrar la validez de las funcionalidades adicionadas al marco de trabajo de los LVs.

Para elaborar la prueba se decidió utilizar dispositivos de fácil adquisición. La selección del dispositivo de interacción se hizo en base a la facilidad de obtención en el mercado (disponibilidad), el costo de adquisición y la facilidad de instalación en una PC convencional, donde se ejecutan los LVs.

El primer dispositivo que se tuvo en cuenta fue el que se describió en la sección 2.4.1. Luego se seleccionó el segundo dispositivo a través del análisis de dos propuestas que se presentan a continuación. La primer propuesta fue el Microsoft Kinect, debido a su amplia disponibilidad y a existencia de versiones compatibles con la PC, junto con sus respectivas APIs. La segunda propuesta fue el mando del Nintendo Wii, que aunque no fue desarrollado de forma específica para su uso en un PC, se puede configurar para este fin. Esto es posible gracias a la existencia de controladores de código abierto, con los que se puede integrar a la computadora con independencia del sistema operativo. Ambos dispositivos se encuentran disponibles en la red comercial de accesorios para videojuegos. Sin embargo, el mando del Nintendo Wii Remote (Wiimote) ofrece menor complejidad para su puesta en funcionamiento y tiene un menor costo de adquisición.

Aunque el esfuerzo requerido para la puesta en funcionamiento de ambos dispositivos con la PC no tiene diferencias significativas, se decidió utilizar el Wiimote para la prueba, con el objetivo de ilustrar la validez de la solución con el menor esfuerzo. Esta selección se realiza para demostrar las capacidades de la solución que se propone y no las capacidades de interacción que ofrecen los dispositivos en sí. Para demostrar cuál se ajusta mejor a un escenario típico de un LV, se debe realizar un estudio de estrategia de HCI con mayor profundidad.

3.2.2. Diseño de un dispositivo compuesto

En esta sección se describe la configuración de un ratón 3D como un dispositivo de interacción compuesto nuevo, para interactuar en un espacio definido por el usuario. El ratón 3D está compuesto por un Wiimote que aporta los eventos de botones y una webcam para realizar el seguimiento con 6 DOF en el espacio, mediante algoritmos de visión por computadoras. La explicación de esta configuración se describe a continuación.

Wiimote

El Wiimote se puede conectar a un PC de escritorio a través de una interfaz bluetooth. Utiliza un acelerómetro interno para detectar cualquier cambio en la orientación en 2 ejes coordenados del espacio. Además presenta un sensor óptico capaz de seguir cuatro emisores de luz infrarroja diferentes (LED). Según la concepción original estos LEDs infrarrojos se colocan en la barra de sensores de Nintendo Wii. La CPU de la consola Wii es capaz de calcular la distancia entre el Wiimote y la barra de sensores a través de algoritmos de triangulación.

Sin embargo, un Wiimote por sí solo conectado a una PC, no puede hacer uso de esta característica debido a la falta de disponibilidad de la consola Wii. Una variante de este problema es recurrir a la implementación de algoritmos de triangulación propios. No obstante, esta alternativa no se tuvo en cuenta debido a que se

quería evaluar la adaptabilidad y flexibilidad de la solución que se propone en este trabajo por encima de la estrategia de interacción en sí.

Sistemas de referencia

La estrategia de trabajo del algoritmo de visión por computadoras del dispositivo de la sección 2.4.1, detecta los marcadores presentes en cada fotograma del video. Luego, utiliza técnicas de análisis de imágenes para calcular la transformación geométrica de cada marcador detectado respecto a la cámara (Kato, 1999). Esto significa que se debe calcular la traslación, rotación y escala de cada marcador con respecto a la cámara en cada fotograma. El componente implementado en el servidor actúa como un dispositivo de seguimiento 3D (ver sección 2.4.1). Este almacena el identificador y los datos de transformación de cada marcador. Finalmente los datos se envían a los clientes conectados.

La aplicación cliente recibe los datos de transformación codificados como un vector de traslación y un cuaternión que indica la orientación del marcador. Los datos recibidos se sincronizan con los datos del mundo virtual para que la interacción fluya sin dificultad. Por tanto, se establece un mapeo entre el sistema de coordenadas del mundo real calculado por ARToolKit y el del mundo virtual como se muestra en la figura 3.1.

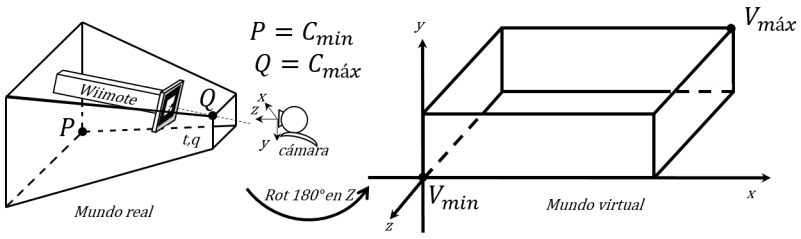


Figura 3.1. Representación de la relación del espacio de interacción en el mundo real con el espacio virtual.

Las transformaciones están dadas en un sistema de coordenadas que tiene una rotación de **180**° alrededor del eje **Z**, con respecto al sistema de coordenadas del mundo virtual. El eje **Z** en ambos sistemas de coordenadas se encuentra dirigido hacia el punto de vista del usuario. De esta forma se puede establecer una relación espacial directa entre ambos sistemas de referencia. Al existir una transformación de rotación con respecto al eje **X**, existe una correspondencia entre los planos **XY** de ambos sistemas. Luego, para relacionarlos se decidió establecer un mapeo de coordenadas, entre el sistema de coordenadas del mundo real y el del mundo virtual.

Para establecer el mapeo de coordenadas entre ambos sistemas de referencia, se necesitan algunos datos que a continuación se describen. Los puntos C_{min} y $C_{máx}$ que representan la posición mínima y máxima respectivamente del volumen de visión que la cámara puede captar en el mundo real. Los puntos V_{min} y $V_{máx}$, como la posición mínima y máxima respectivamente del volumen de interacción del mundo virtual. El vector de traslación y el cuaternión (t, q), que definen la posición y orientación del marcador detectado por ARToolKit respectivamente. Con estos datos el mapeo que representa el espacio de interacción, es la razón

que existe entre el vector formado por los puntos C_{min} y $C_{máx}$ y el vector formado por los puntos V_{min} y $V_{máx}$. Luego las expresiones que describen el mapeo de las coordenadas son:

$$C = C_{máx} - C_{min} \tag{3.2.1}$$

$$V = V_{máx} - V_{min} \tag{3.2.2}$$

$$S = (\frac{V_x}{C_x}; \frac{V_y}{C_y}; \frac{V_z}{C_z}) \tag{3.2.3}$$

$$t'_x = (t_x - C_{min_x}) * S_x + V_{min_x} \tag{3.2.4}$$

$$t'_y = (t_y - C_{min_y}) * S_y + V_{min_y} \tag{3.2.5}$$

$$t'_z = (t_z - C_{min_z}) * S_z + V_{min_z} \tag{3.2.6}$$

$$q'_{wxyz} = (q_w; q_x; q_y, q_z) \tag{3.2.7}$$

La expresión que define el cuaternión resultante, está determinada por una conversión de matriz a cuaternión. Esta conversión se realiza a partir de la matriz de transformación, que detecta el algoritmo de visión por computadoras cuando calcula los valores de traslación y rotación del marcador físico. La estrategia de seguimiento con 6 DOF ofrecida por el método que propone ARToolKit, resulta una alternativa factible para la implementación del seguimiento del cursor 3D. De esta forma, la limitación (2 DOF) que brinda el Wiimote queda resuelta y la composición de los dispositivos de interacción se hace efectiva.

3.2.3. Entorno virtual

La prueba del funcionamiento del cursor 3D requiere un espacio de interacción en un mundo virtual para demostrar sus capacidades. La lógica presentada en el código fuente de la figura 2.8 se implementó en una clase para cumplir con los requerimientos de la arquitectura del *software* orientada a objetos de los LVs. El entorno virtual en el que se hizo la prueba, corresponde al escenario de un ejercicio del Laboratorio Virtual de Ensamblaje de una Computadora.

En la figura 3.2, se observa un escenario virtual, en el que se muestra el cursor virtual en forma de saeta. Este se mueve por el espacio de interacción a medida que el usuario mueve el ratón 3D (implementado con el Wii y el marcador de ARToolKit). Los objetos que constituyen el espacio de trabajo son los componentes internos de un ordenador de mesa (tarjeta madre, tarjeta de video, microprocesador, batería de tarjeta madre y sistema de disipación de calor). En el caso de la la figura 3.2, el cursor tiene seleccionada una pieza interna de la computadora.

Se definió un ejercicio de interacción clásico en el que el usuario solo debía seleccionar, arrastrar y soltar elementos virtuales dentro del espacio de interacción. No es objetivo de este trabajo desarrollar una lógica de interacción que vaya más allá de estas tres acciones, pues son las necesarias para demostrar la comunicación de los dispositivos de interacción como un todo.

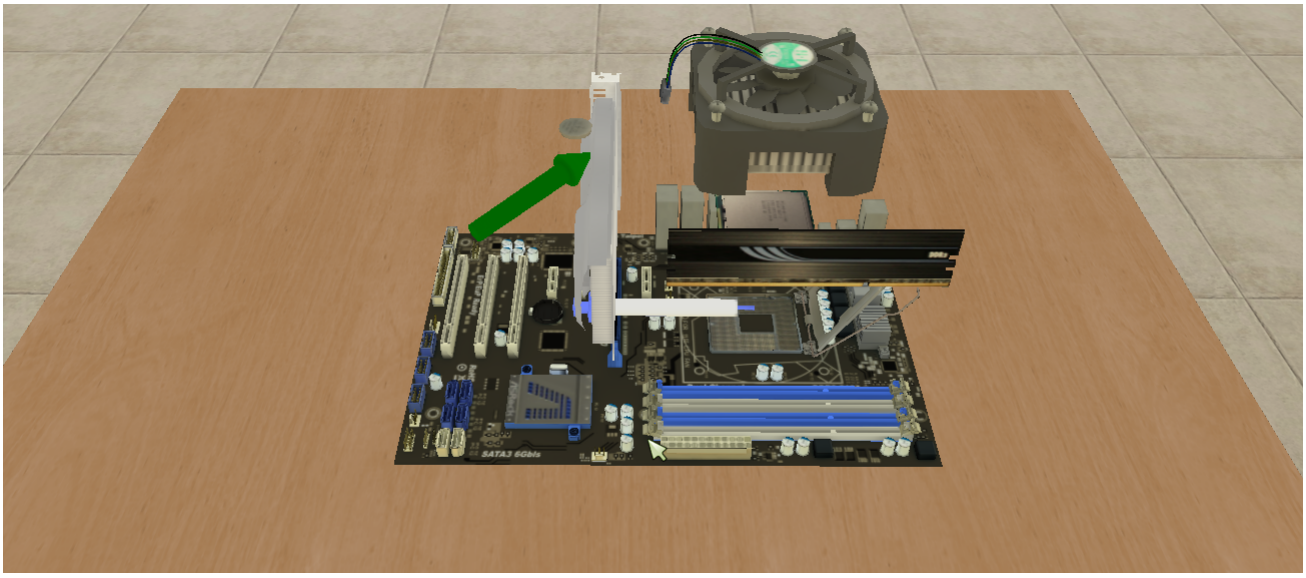


Figura 3.2. Cursor 3D en el entorno virtual de ensamblaje del PC.

La comunicación requirió la utilización de un cliente genérico del tipo *vrpn_Tracker_Remote* para la escucha de los datos de seguimiento ofrecidos por el algoritmo de visión por computadoras y otro del tipo *vrpn_Button_Remote* para los datos de los botones presionados en el Wiimote. En la figura 3.3 se muestra el diagrama de la clase diseñada para implementar el cursor 3D. Con estos datos se definió la lógica de interacción del prototipo de aplicación.

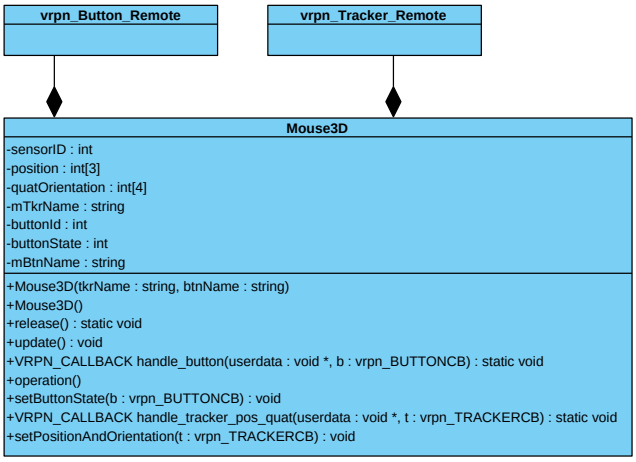


Figura 3.3. Diagrama de clase para el Cursor 3D.

La lógica para la interacción con los objetos virtuales del ERV mediante el cursor 3D, está definida según los pasos del algoritmo siguiente:

1. Mientras el usuario mueve el cursor por el espacio de trabajo donde se encuentran los objetos del entorno virtual

   a Si el ratón 3D no tiene un botón presionado

      i Si la posición del cursor virtual corresponde con un objeto seleccionable, se colorea para mostrar que está seleccionado

      ii Si no, si existe algún objeto anteriormente seleccionado, se deselecciona y se queda en la posición que el cursor virtual tenía en el momento en que se liberó el botón

   b Si no (el ratón 3D tiene un botón presionado)

      i Si existe un objeto seleccionado, el objeto virtual se mueve junto al cursor 3D

2. Fin

3.3. Análisis del esfuerzo para el código del cursor 3D

El esfuerzo requerido para desarrollar un *software* siempre es un elemento necesario a la hora estimar el costo total del desarrollo de un proyecto. Según (Pressman, 1997b), un aumento del esfuerzo siempre va a impactar negativamente en el tiempo de desarrollo del producto y por ende en la productividad del equipo de desarrollo. En esta sección se hace un análisis de la productividad y el esfuerzo dedicado a desarrollar el componente de la sección 3.2 para determinar el impacto que tiene la solución propuesta en el tiempo de desarrollo de un producto. El análisis se realiza mediante la comparación de los resultados obtenidos utilizando la alternativa tradicional, con la alternativa de la solución propuesta.

Una medida directa del producto dentro del proceso de ingeniería del *software* es el esfuerzo aplicado. El esfuerzo se mide en las LDCs que un ingeniero de *software* es capaz de escribir en un tiempo determinado dentro del ciclo de vida del proyecto. Un enfoque tradicional de la ingeniería de *software* ha sido considerar la productividad de *software* como el número de LDC producidas por Persona-Mes (PM) de esfuerzo. Según (Pressman, 1997a), existen variaciones en cuanto a la definición de qué es una LDC y difiere en cuanto al nivel del lenguaje de programación. Por lo que no se puede utilizar cuando coexisten distintos lenguajes de programación en un proyecto o entre equipos de desarrollo distintos, donde varía la cantidad de personas y sus habilidades.

Esta comparación debe existir cuando las condiciones son similares y situadas en un contexto bien definido. En condiciones de similitud, es pertinente utilizar la medida LDC para extraer métricas de productividad (Pressman, 1997a). A esto se suma que debe existir de antemano una información histórica de desarrollos similares.

El autor de esta investigación considera que existen las condiciones necesarias para analizar el esfuerzo para el caso que se expone en la sección 3.2. Esta consideración está sustentada en que el desarrollo de los productos de VERTEX y en especial de los LVs, se realiza con un marco de trabajo que utiliza el lenguaje de programación C++ y los datos históricos confirman un solo desarrollador como encargado de la interacción.

Otra consideración para establecer el contexto de la comparación, es que el análisis está enfocado solamente a la etapa de codificación. No se tiene en cuenta el esfuerzo administrativo, de pruebas o soporte, con el objetivo de considerar el impacto neto que tiene la solución en la etapa de codificación del producto. Es necesario aclarar que el esfuerzo medido será en cantidad de LDC adicionadas al producto en sí.

Los resultados de las LDCs necesarias para crear el cursor 3D con la variante anterior a la solución que se propone en este trabajo se muestran en la tabla 3.1.

Tabla 3.1. Estimación del esfuerzo mediante la métrica LDC antes de la solución.

Funcionalidades	**LDC**
Del dispositivo basado en ARToolKit	337
Del dispositivo basado en Wiimote	652
Total	989

Los resultados de las LDCs necesarias para crear el cursor 3D una vez que se aplicó la solución propuesta se muestran en la tabla 3.2. En la figura 3.4 se observa que con la utilización de la solución propuesta se registró un incremento de aproximadamente un 45 % de LDC con respecto a la versión anterior. Esto se debe en particular a que, además de la implementación de los dispositivos como componentes de la biblioteca de dispositivos, se debe escribir el código del servidor que gestiona los dispositivos, así como el cliente.

Tabla 3.2. Estimación del esfuerzo mediante la métrica LDC con la solución propuesta.

Funcionalidades	**LDC**
Del dispositivo basado en ARToolKit	385
Del dispositivo basado en Wiimote	874
De gestión en el servidor	787
De gestión en el cliente	127
Total	2173

Con la solución propuesta ocurre algo contradictorio, puesto que se observa un incremento de las LDCs. Sin embargo, el verdadero beneficio se observa en las LDC del lado del cliente (ver tabla 3.2), en el que solo necesita 127 LDC para definir el dispositivo compuesto. Esta situación ocurre la primera vez que se define un nuevo dispositivo en el servidor. Luego a medida que se utilice este dispositivo en desarrollos posteriores, se escribirán una cantidad código para conectarse al dispositivo en el lado del cliente, en proporción a las observadas para el cursor 3D. Esto deviene como resultado directo de la reutilización de componentes de *software*, debido a que se accede a los dispositivos mediante clases abstractas (ver sección 2.8), que tipifican las clasificaciones de las funciones de los dispositivos.

Los valores descritos en las tablas 3.1 y 3.2 son resultados de la observación, debido a que fueron obtenidos luego de haber escrito el *software* que compone la solución y por lo tanto no constituyen estimaciones. Estos son datos que se pueden utilizar como histórico para estimaciones de esfuerzo en proyectos futuros,

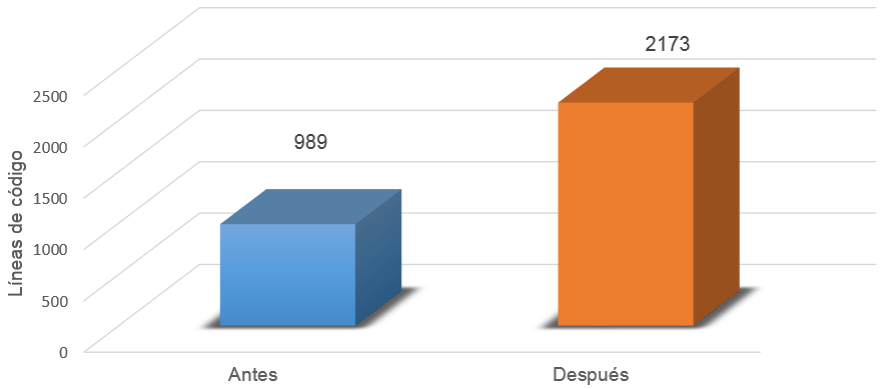


Figura 3.4. Líneas de código escritas en la variante anterior a la solución propuesta y con la solución propuesta.

relacionados con la solución que se plantea en este trabajo.

La productividad medida en LDCPM puede constituir una medida subjetiva en cuanto a estimar el tiempo de desarrollo de un proyecto debido a que las habilidades de codificación de los desarrolladores pueden variar. Para ver como impacta la productividad en el tiempo de desarrollo de los proyectos que utilicen esta propuesta de solución, se realizó un análisis de productividad. Se identificaron 3 niveles de productividad, basados en los datos históricos obtenidos de desarrolladores de proyectos anteriores de los LVs.

En el contexto de los LVs, los datos de niveles de productividad obtenidos se corresponden de la siguiente forma:

- **Nivel Bajo**: Estudiantes de nuevo ingreso en el proyecto con menos de un semestre
- **Nivel Medio**: Desarrolladores del proyecto con experiencia previa del marco de trabajo de los LVs
- **Nivel Alto**: Desarrolladores del marco de trabajo de los LVs

Por tanto el contexto de productividad se estableció para los desarrollos de LVs, para otro tipo de desarrollo habría que estimar los valores de productividad. Los resultados obtenidos se observan en la figura 3.5.

Con los resultados de la figura 3.5 se observa que para el desarrollo de la solución propuesta, una persona con un nivel bajo de productividad bajo tarda aproximadamente 4 meses y medio, en contraste con la variante previa que tardaría unos 2 meses. Así mismo un desarrollador con productividad media puede implementar la solución propuesta en 2 meses y la variante anterior en aproximadamente 1 mes. Finalmente se observa que una persona con niveles de productividad alta lo puede hacer en 1 mes la solución completa y 2 semanas la variante anterior.

Los resultados de reducción del tiempo de desarrollo se observan cuando se utiliza la solución en el lado del cliente, como se describe en la figura 3.6 para la implementación del cursor 3D. En este caso la comunicación con el dispositivo para un desarrollador con bajos niveles de productividad podría tardar aproximadamente una semana, uno con nivel medio podría hacerlo en 3 días y uno con nivel alto podría

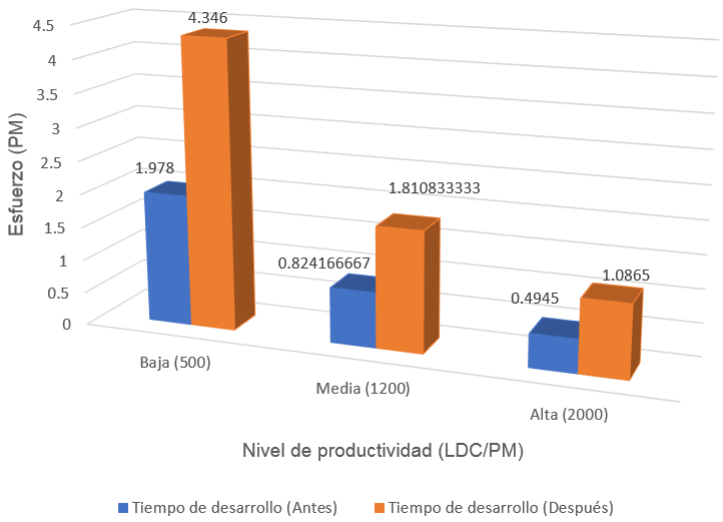


Figura 3.5. Tiempo de desarrollo de una persona que utiliza la propuesta de solución, de acuerdo a tres niveles de productividad.

hacerlo en un día. Por lo tanto se observa una reducción de tiempo considerable con un promedio aproximado de un 85 % del tiempo de desarrollo.

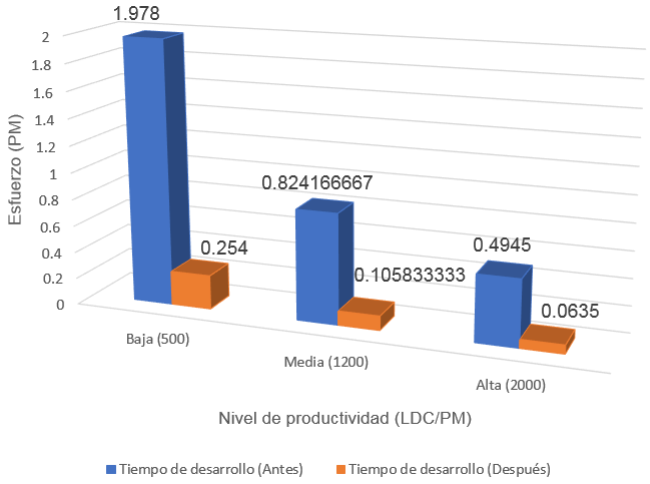


Figura 3.6. Tiempo de desarrollo en meses por una persona del cursor 3D, de acuerdo a tres niveles de productividad.

Estos resultados se corresponden con los alcanzados en la práctica con estudiantes de la práctica profesional que se han insertado en los proyectos y con tesistas que han desarrollado tesis de RA y de RV con interacción, en la facultad 5. Mediante la observación directa, se aprecia que los estudiantes tardan aproximadamente dos meses en desarrollar una funcionalidad con características similares al cursor 3D, con la alternativa anterior a la solución propuesta. Para comprobar en la práctica cuánto tarda una persona con nivel bajo de productividad, se realizó un prueba con 3 estudiantes de quinto año de la facultad 5, que realizan

su práctica profesional en el centro VERTEX. Luego de una sesión de clases capacitación con la solución implementada, se les indicó que realizaran una funcionalidad como la del cursor 3D.

La tarea solo estuvo enfocada en la comunicación con los dispositivos que conforman el cursor 3D y mostrar los datos en un registro de consola. El objetivo era medir el tiempo que dedicaron a la comunicación y obtención de los datos de los dispositivos, sin que hubiese una lógica para esos datos. Luego de la capacitación, los tres estudiantes implementaron la tarea indicada en otras tres sesiones de trabajo de la asignatura Práctica Profesional, lo cual se corresponde con la estimación hecha en la figura 3.6.

3.4. Análisis de tiempo de respuesta (Latencia)

La solución presentada en este trabajo brinda una capa de abstracción sobre comunicación por la red. Por tanto se introduce una latencia que puede influir en el tiempo de respuesta del entorno virtual. Para calcular la latencia introducida se hicieron varias pruebas entre ordenadores corriendo sistemas operativos Windows 7 y GNU\Linux Ubuntu 13.04 en una red de área local cableada. Las pruebas realizadas mediante comandos **ping** mostraron un promedio de tiempo de latencia de aproximadamente 0.8 milisegundos desde el servidor hasta el cliente. El tiempo medido para el paso de mensajes a nivel de aplicación demoró en promedio aproximadamente 2.8 milisegundos, desde el momento en que se genera el mensaje en el servidor hasta que el mensaje es procesado por la función de manipulación de mensajes.

Los LVs son aplicaciones de entrenamiento virtual y no requieren un tiempo de respuesta crítico, por lo que se puede afirmar que los valores obtenidos en la medición no introducen una latencia significativa, si se comparan con el nivel de abstracción alcanzado para acceder a la información de los dispositivos de interacción.

El entorno virtual se desarrolló con el marco de trabajo de los LVs basada en el motor de gráficos OGRE 3D en su versión 1.9. El hardware utilizado para la prueba fue un ordenador portátil HP EliteBook, con un procesador Intel Core i7 4600U 2.7 GHz, 8 GB de DDR3 y con chipset de video Intel(R) HD Graphics 4400 con 2112 MB de memoria RAM.

3.5. Consideraciones parciales

En este capítulo se comprobó la validez de la solución propuesta mediante el desarrollo de un prototipo de aplicación. La aplicación de la métrica LDC al prototipo de aplicación, para comparar el esfuerzo necesario al implementar el acceso a la información de los dispositivos de interacción, demostró que la solución propuesta reduce el tiempo de desarrollo en un 85 % con respecto a la variante anterior. Con los valores obtenidos en el análisis de tiempo de respuesta de la aplicación se puede afirmar que no se introduce una latencia significativa, si se tiene en cuenta la reducción del esfuerzo necesario para acceder a la información de los dispositivos y el nivel de abstracción alcanzado por la solución propuesta.

Conclusiones

El análisis de los referentes teóricos permitió seleccionar a VRPN como la biblioteca a reutilizar, para formar parte de la solución al problema de investigación. Además, se provee como solución un mecanismo de acceso a la información de los dispositivos de interacción para los ERV desarrollados por los grupos de investigación ViViRG e IDEASCAD.

En la solución implementada se adaptaron las funcionalidades de la biblioteca VRPN mediante la incorporación de un nuevo dispositivo de seguimiento 3D basado en visión por computadoras. Se logró una solución multiplataforma probada en sistemas Windows y GNU\Linux.

La información obtenida mediante la aplicación de la métrica LDC al prototipo de aplicación que incorpora el mecanismo de acceso a la información de los dispositivos de interaccion 3D en un ERV, posibilitó validar la solución propuesta para disminuir el tiempo de desarrollo de los ERV en los grupos de investigación ViViRG e IDEASCAD.

Acrónimos

AIPO Asociación Interacción Persona-Ordenador

API Interfaz de Programación de Aplicaciones (por sus siglas en inglés de *Application Programming Interface*)

CAVE Cueva de Entornos de Realidad Virtual Automática (por sus siglas en inglés de *Cave Automatic Virtual Environment*)

DOF Grados de Libertad, (por sus siglas en inglés de *Degree Of Freedom*

GUI Interfaz de Programación de Aplicaciones, (por sus siglas en inglés de *Graphics User Interface*)

HCI Interacción Hombre-Computadora (por sus siglas en inglés de *Human-Computer Interaction*)

ERV Entorno de Realidad Virtual

GLSV Graphics Library Stereo Vision for OpenGL

IDE Entorno Integrado de Desarrollo (por sus siglas en inglés de *Integrated Development Environment*)

IPO Interacción Persona-Ordenador

LV Laboratorio Virtual

LDC Líneas De Código

PM Persona-Mes

RV Realidad Virtual

RA Realidad Aumentada

SRV Sistema de Realidad Virtual

UCI Universidad de las Ciencias Informáticas

VERTEX Centro Entornos Interactivos 3D

VRPN Virtual-Reality Peripheral Network

UNIOVI Universidad de Oviedo

ViViRG Grupo de Visualización y Realidad Virtual

VA Virtualidad Aumentada

MHCI Interacción Multimodal Hombre-Máquina (por sus siglas en inglés de *Multimodeal Human-Computer Interface*)

MR Realidad Mixta (por sus siglas en inglés de *Mixed Reality*)

WIMP Ventanas, Iconos, Menús y Puntero, (por sus siglas en inglés de *Window,Icon, Menu,Pointer*)

TCP Transmission Control Protocol

UDP User Datagram Protocol

Referencias bibliográficas

AIPO (2013). *Asociación Interacción Persona-Ordenador*. URL: `http://aipo.es` (visitado 30-09-2014).

Almeida, Nuno, Samuel Silva y António Teixeira (2014). «Design and Development of Speech Interaction: A Methodology». En: *Human-Computer Interaction. Advanced Interaction Modalities and Techniques*. Springer, págs. 370-381.

Arhippainen, Leena, Minna Pakanen y Seamus Hickey (2013). «Studying Depth in a 3D User Interface by a Paper Prototype as a Part of the Mixed Methods Evaluation Procedure. Early Phase User Experience Study». En: *ACHI 2013, The Sixth International Conference on Advances in Computer-Human Interactions*, págs. 35-40.

Azuma, Ronald T et al., (1997). «A survey of augmented reality». En: *Presence* 6.4, págs. 355-385.

Barrilleaux, Jon (2000). *3D user interfaces with Java 3D*. Manning Publications Co.

Bierbaum, Allen et al., (2001). «VR Juggler: A virtual platform for virtual reality application development». En: *Virtual Reality, 2001. Proceedings. IEEE*. IEEE, págs. 89-96.

Bowman, Doug A, Jian Chen et al., (2006). «New Directions in 3D User Interfaces.» En: *IJVR* 5.2, págs. 3-14.

Bowman, Doug A et al., (2001). «An introduction to 3-D user interface design». En: *Presence: Teleoperators and virtual environments* 10.1, págs. 96-108.

— (2004). *3D user interfaces: theory and practice*. Addison-Wesley.

Boyd, David y Lakshmi Sastry (1999). «Development of the INQUISITIVE Interaction Toolkit-Concept and Realisation». En: *Proc. of Workshop on User Centered Design and Implementation of Virtual Environments UCDIVE?99*. Citeseer.

Chardonnet, J-R y J-C Leon (2012). «Designing interaction in virtual worlds through a passive haptic peripheral». En: *RO-MAN, 2012 IEEE*. IEEE, págs. 284-289.

De Boeck, Joan et al., (2007). «High-level modeling of multimodal interaction techniques using nimmit». En:

Duque, Carlos, Fernando De la Rosa y Jose Tiberio Hernandez (2013). «Multimodal interaction architecture applied to navigation in maps». En: *Computing Colombian Conference (8CCC), 2013 8th*. IEEE, págs. 1-6.

Eastgate, Richard Mark (2001). «The structured development of virtual environments: enhancing functionality and interactivity». Tesis doct. University of Nottingham.

Engel, Juri y Jürgen Döllner (2012). «Immersive Visualization of Virtual 3D City Models and its Applications in E-Planning». En: *International Journal of E-Planning Research (IJEPR)* 1.4, págs. 17-34.

Engine, Irrlicht (2008). «A free open source 3d engine». En: *Avaiable: http://irrlicht. sourceforge. net*.

Figueroa, Pablo, Mark Green y H James Hoover (2002). «InTml: a description language for VR applications». En: *Proceedings of the seventh international conference on 3D Web technology*. ACM, págs. 53-58.

Foley, James D et al., (1996). «Computer graphics: principles and practice.» En: *ISBN: 0-201-12110-7*.

Francese, Rita, Ignazio Passero y Genoveffa Tortora (2012). «Wiimote and Kinect: gestural user interfaces add a natural third dimension to HCI». En: *Proceedings of the International Working Conference on Advanced Visual Interfaces*. ACM, págs. 116-123.

Frécon, Emmanuel y Mårten Stenius (1998). «DIVE: A scaleable network architecture for distributed virtual environments». En: *Distributed Systems Engineering* 5.3, pág. 91.

Gallo, Luigi, Giuseppe De Pietro e Ivana Marra (2008). «3D interaction with volumetric medical data: experiencing the Wiimote». En: *Proceedings of the 1st international conference on Ambient media and systems*. ICST (Institute for Computer Sciences, Social-Informatics y Telecommunications Engineering), pág. 14.

Greenhalgh, Chris, Jim Purbrick y Dave Snowdon (2000). «Inside MASSIVE-3: flexible support for data consistency and world structuring». En: *Proceedings of the third international conference on Collaborative virtual environments*. ACM, págs. 119-127.

Haan, Gerwin de y Frits H Post (2009). «StateStream: a developer-centric approach towards unifying interaction models and architecture». En: *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*. ACM, págs. 13-22.

Herndon, Kenneth P, Andries van Dam y Michael Gleicher (1994). «The challenges of 3D interaction: a CHI'94 workshop». En: *ACM SIGCHI Bulletin* 26.4, págs. 36-43.

Hilliges, Otmar et al., (2012). «HoloDesk: direct 3d interactions with a situated see-through display». En: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, págs. 2421-2430.

Inc., Apple (2013). *Apple iOS - SIRI*. URL: `https://www.apple.com/ios/siri/` (visitado 30-09-2013).

Inc., Inertia Intersense (2014). *InterSense | Precision Motion Tracking Solutions | IS-900 System*. URL: `http://www.intersense.com/pages/20/14` (visitado 30-09-2013).

Inc., Northern Digital (2013a). *Medical Aurora - Medical*. URL: `www.ndigital.com/medical/aurora` (visitado 30-09-2013).

— (2013b). *Polaris Optical Tracking Systems*. URL: http://www.ndigital.com/medical/products/polaris-family/ (visitado 30-09-2013).

Izadi, Shahram et al., (2011). «KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera». En: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, págs. 559-568.

Junker, Gregory (2006). *Pro OGRE 3D Programming (Pro)*. Berkely, CA, USA: Apress. ISBN: 1590597109.

Kato, Hirokazu (1999). «ARToolKit». En: *http://www. hitl. washington. edu/artoolkit/*.

Kelso, John et al., (2003). «DIVERSE: a framework for building extensible and reconfigurable device-independent virtual environments and distributed asynchronous simulations». En: *Presence: Teleoperators and Virtual Environments* 12.1, págs. 19-36.

Lawrence, Kira et al., (2013). «Investigation of interaction modalities designed for immersive visualizations using commodity devices in the classroom». En: *Design, User Experience, and Usability. Health, Learning, Playing, Cultural, and Cross-Cultural User Experience*. Springer, págs. 209-218.

Ltd., Virtual Realities (2014). *Data Gloves - Virtual Realities*. URL: http://www.vrealities.com/products/data-gloves/5dt-data-glove-5-ultra-2 (visitado 19-09-2014).

Luque, Francisco Pedro et al., (2014). «Integration of multisensorial stimuli and multimodal interaction in a hybrid 3DTV system». En: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 11.1s, pág. 16.

Maimone, Andrew y Henry Fuchs (2011). «Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras». En: *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*. IEEE, págs. 137-146.

Marsh, Timothy et al., (1998). «A shared framework of virtual reality». En: *Proceedings of UK-VRSIG* 98.

Martinez, Diego et al., (2010). «AFreeCA: Extending the Spatial Model of Interaction.» En: *CW*, págs. 17-24.

Martín, Santiago et al., (2009). «GLSV: Graphics library stereo vision for OpenGL». En: *Virtual reality* 13.1, págs. 51-57.

Martínez, Diego et al., (2011). «A framework to develop VR interaction techniques based on openinterface and AFreeCA». En: *Human-Computer Interaction–INTERACT 2011*. Springer, págs. 1-18.

Microsoft (2014). *Kinect for Windows*. URL: http://www.microsoft.com/en-us/kinectforwindows/ (visitado 30-09-2014).

Milgram, Paul y Fumio Kishino (1994). «A taxonomy of mixed reality visual displays». En: *IEICE TRANSACTIONS on Information and Systems* 77.12, págs. 1321-1329.

Nordahl, Rolf et al., (2011). «A multimodal architecture for simulating natural interactive walking in virtual environments». En: *PsychNology* 9.3, págs. 245-268.

Osfield, Robert, Don Burns et al., (2004). *Open scene graph*.

Park, Jiyong et al., (2014). «Digital Holographic Printing Methods for 3D Visualization of Cultural Heritage Artifacts». En: *Digital Presentation and Preservation of Cultural and Scientific Heritage* IV, págs. 69-78.

Pittarello, Fabio y Augusto Celentano (2001). «Interaction locus: a multimodal approach for the structuring of virtual spaces». En: *Proceedings of HCItaly 2001 Symposium*.

pjcast (2013). *Object Oriented Input System*. URL: `http://sourceforge.net/projects/wgois/` (visitado 30-09-2013).

Polhemus (2013). *Asociación Interacción Persona-Ordenador*. URL: `www.polhemus.com` (visitado 30-09-2013).

Preece, Jenny et al., (1994). *Human-computer interaction*. Addison-Wesley Longman Ltd.

Pressman, Roger S (1997a). «Ingeniería del Software: Un enfoque práctico». En: *Ingeniería del Software: Un enfoque práctico*. Mikel Angoar. Cap. Proceso del software y métricas del proyecto.

— (1997b). *Ingeniería del Software: Un enfoque práctico*. Mikel Angoar.

Ray, Andrew y Doug A Bowman (2007). «Towards a system for reusable 3D interaction techniques». En: *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*. ACM, págs. 187-190.

Reitmayr, Gerhard y Dieter Schmalstieg (2001). «OpenTracker-An Open Software Architecture for Reconfigurable Tracking Based on XML.» En: *vr* 1, pág. 285.

Scali, Silvia, Mark Wright y Ann Marie Shillito (2003). «3D Modelling is not for WIMPs». En: *Proceedings of HCI international*.

Schmalstieg, Dieter et al., (2002). «The studierstube augmented reality project». En: *Presence: Teleoperators and Virtual Environments* 11.1, págs. 33-54.

Schomaker, Lambert et al., (1995). «A taxonomy of multimodal interaction in the human information processing system». En:

Serrano, Marcos et al., (2008). «The openinterface framework: a tool for multimodal interaction.» En: *CHI'08 Extended abstracts on human factors in computing systems*. ACM, págs. 3501-3506.

Seth, Abhishek, Judy M Vance y James H Oliver (2011). «Virtual reality for assembly methods prototyping: a review». En: *Virtual reality* 15.1, págs. 5-20.

Shaw, Chris et al., (1993). «Decoupled simulation in virtual reality with the MR toolkit». En: *ACM Transactions on Information Systems (TOIS)* 11.3, págs. 287-317.

Stuart, Rory (2001). *The Design of virtual environments*. Barricade Books, Incorporated.

Sutcliffe, Alistair (2003). *Multimedia and virtual reality: designing multisensory user interfaces*. Psychology Press.

Taylor II, Russell M et al., (2001). «VRPN: a device-independent, network-transparent VR peripheral system». En: *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, págs. 55-61.

Vilar, Polona (2010). «Designing the User Interface: Strategies for Effective Human-Computer Interaction». En: *Journal of the American Society for Information Science and Technology* 61.5, págs. 1073-1074.

Wingrave, Chadwick y Doug Bowman (2005). «Chasm: Bridging description and implementation of 3d interfaces». En: *Proceedings of IEEE VR 2005 Workshop on New Directions in 3D User Interfaces*. Citeseer, págs. 85-88.

Generado con LaTeX