Propuesta de tareas y técnicas de interacción con sistema de neuronavegación guiada por imágenes.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Yuriaski Leyva Clemente

Tutor: MSc. Ernesto de la Cruz Guevara Ramírez
Co-tutora: Ing. Mileydi Moreno Mirabal

La Habana, junio de 2015

"Año del 57 Aniversario del Triunfo de la Revolución"

Declaración de Autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yuriaski Leyva Clemente

_____

Firma del Autor

MSc. Ernesto de la Cruz Guevara Ramírez

_____

Firma del Tutor

Ing. Mileydi Moreno Mirabal

_____

Firma de la Co-tutora

Datos del contacto

Tutor: MSc. Ernesto de la Cruz Guevara Ramírez

Máster en Informática Aplicada, se ha desempeñado como profesor, desarrollador e investigador, con 8 años de experiencia en el trabajo con Gráficos por Computadoras (juegos, visualización médica y científica). Alcanzó la categoría de Profesor Asistente. Además posee cinco años de experiencia en el trabajo con Visión por Computadoras y Realidad Aumentada. Actualmente pertenece al Centro Vertex Entornos Interactivos 3D de la Facultad 5 de la Universidad de las Ciencias Informáticas.

E-mail: elguevara@uci.cu

Co-tutora: Ing. Mileydi Moreno Mirabal

Graduada como Ingeniera en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2008. Profesora con categoría Asistente y ocho años de experiencia en el trabajo con Gráficos por Computadoras, Visión por Computadoras y Realidad Aumentada. Actualmente pertenece al Centro Vertex Entornos Interactivos 3D de la Facultad 5 de la Universidad de las Ciencias Informáticas.

E-mail: mmirabal@uci.cu

Dedicatoria

A mis padres que han soportado un duro batallar para que yo llegara a este punto de mi vida tan importante. Ellos son la fuente que me ha proporcionado impulso moral y fuerza de voluntad para vencer todos los obstáculos que se me han presentado. A mis preciosas sobrinas para que tomen mi ejemplo y alcancen grandes metas. A mi primo Alian por ser como un hermano mayor y protegerme siempre. Al resto de mi familia que siempre tengo presente.

Agradecimientos

Le agradezco a mis padres por todos los sacrificios a los que se han sometido por mí. No me alcanzaría la vida para devolverles todo lo que me han dado. Gracias a ellos he tenido una razón por la cual luchar y mantenerme firme. Por ellos he logrado metas que por mí no hubiera alcanzado. A mis lindas sobrinas que sin saberlo me han alegrado solo con ser ellas y estar ahí para mí. A mi hermana que a su forma un poco extraña siempre me ha querido y apoyado. Al Ye, mi cuñado, por ser un buen amigo. A los muchachos del 87202 y del 88203 que los considero parte de mi familia por todo el tiempo que pasamos juntos. A mis compañeros de investigación Lozano y Alejandro Ravelo por brindarme su ayuda incondicional siempre que la necesité. A mis tutores que aunque estaban muy atareados lograron darme una buena atención. A Leiser por sacrificar su tiempo libre y regalármelo. A Nelson por sus ráfagas de consejos y charlas de moral. A Pedro por compartir sus conocimientos y por sus críticas constructivas. A Yosbani por actuar como un primo de sangre. A Osnier por ser un compañero de cuarto silencioso. A todos mis amigos de universidad que son muchos y les agradezco porque en algún momento seguramente hicieron algo por mí o sino tal vez lo hagan en un futuro.

Resumen

En la actualidad se han introducido en la Neurocirugía los neuronavegadores. Herramientas con tecnología de punta que mejoran los resultados de las intervenciones quirúrgicas realizadas en esta especialidad médica. Estos equipos poseen componentes de software y hardware muy costosos en el mercado internacional. Atendiendo a esto en la Facultad 5 de la Universidad de las Ciencias Informáticas se intenta desarrollar un neuronavegador cubano.

El neuronavegador debe contar con una lógica de interacción hombre-máquina que permita al neurocirujano acceder a sus funcionalidades de forma fácil y flexible. La presente investigación responde a cómo garantizar un manejo cómodo del sistema al mismo tiempo que se pueda mantener una retroalimentación de la información que este muestra. El autor hace un estudio sobre interfaz hombre-máquina e interacción con entornos virtuales de manera general, para luego enfocarse en el tipo de sistema estudiado.

Como resultado se propone un conjunto de técnicas y tareas de interacción, para permitir al neurocirujano realizar las acciones más comunes en un neuronavegador. La solución está basada en el Nintendo Wiimote como dispositivo físico de interacción junto con las bibliotecas *Virtual Reality Peripheral Network* y *Visualization Toolkit*. La primera es utilizada para gestionar al dispositivo físico de interacción. La segunda se encarga de brindar el soporte para conformar las transformaciones necesarias que se aplicarán a los objetos en tercera dimensión. Por último la biblioteca *Image-Guided Surgery Toolkit* se usa como mecanismo para manejar los eventos a partir de la interfaz de programación de aplicaciones que brinda para implementar máquinas de estados.

Palabras claves: neuronavegador, interacción, interfaz, tareas y técnicas de interacción.

Índice

Índice de Figuras

Índice de Tablas

Introducción

En las últimas décadas la informática a nivel mundial se ha desarrollado a pasos agigantados y son innumerables las nuevas tecnologías alcanzadas. Cuba es un país que a pesar de todos los obstáculos a los que se ha enfrentado en los últimos tiempos ha logrado mantenerse a la par con el resto del mundo en materia científica. Siendo consecuentes con estos avances y con el objetivo de llevar los beneficios de la informática a todo el pueblo cubano se hicieron necesarios productos de alta calidad en este ámbito. Teniendo esta meta presente se crea la Universidad de las Ciencias Informáticas (UCI), institución que entre otras funciones presta servicio de apoyo informático a otros campos científicos como es el caso de la medicina.

La Neurocirugía es una de las especialidades más complejas de la medicina que ha presentado un crecimiento acelerado en la informatización de sus servicios. Especialidad que se encarga de tratar las enfermedades del sistema nervioso mediante operaciones quirúrgicas [1], las cuales son de altísimo riesgo y demandan condiciones con un gran nivel de seguridad para realizarlas. Estas operaciones suelen ser muy invasivas lo que trae como consecuencia la posibilidad de dejar secuelas al paciente. En resumen son procedimientos médicos muy complejos que se realizan actuando físicamente sobre el cerebro y que como alternativa se pueden apoyar en tecnologías y programas informáticos para lograr mejores resultados que serían reflejados en el bienestar de la sociedad. En otros países ya se han insertado este tipo de tecnologías en la Neurocirugía obteniendo resultados alentadores ([2], véase también [3] y [4]).

Entre las herramientas de última generación que existen actualmente en el campo de la Neurocirugía se destaca el neuronavegador, el cual brinda al cirujano y su equipo profesional, un margen de seguridad alto, que humaniza el tratamiento quirúrgico de lesiones cerebrales. Por lo tanto la neuronavegación se ha convertido en una rutina que ha dejado obsoletos a otros procedimientos en muchos hospitales del mundo. Para practicarla son necesarios equipos que tienen un costo de adquisición en el mercado internacional variable alrededor de los 400 000 dólares [5] sin incluir el costo de mantenimiento y actualización. Por otro lado las posibilidades técnicas y el desarrollo informático alcanzado en Cuba, sirven como base para la fabricación de estos equipos; los cuales una vez desarrollados podrán distribuirse por los hospitales cubanos y de las colaboraciones médicas en el exterior. Además se avizora como una vía para posicionar la industria cubana del software en el mercado internacional, especialmente en el de América Latina, África y Asia, con precios que pudieran acercarse mucho más a su costo real de producción y con elevada calidad tecnológica.

Dentro de la UCI se encuentra el centro VERTEX que se especializa en el estudio de temas referentes a Gráficos por Computadora, entre ellos cómo desarrollar software con técnicas de Realidad Aumentada; y en gran medida presta atención a su aplicación en la medicina. Teniendo en cuenta esto junto con lo planteado en

el párrafo anterior, la Clínica Central Cira García (CCCG), en colaboración con la Sociedad Cubana de Neurología y Neurocirugía (SCNN) y el Centro Internacional de Rehabilitación Neurológica (CIREN) han propuesto a la UCI la fabricación de un neuronavegador.

El hardware en donde se instalan los sistemas de navegación guiada por imágenes, específicamente los de neuronavegación, se ubican normalmente dentro del salón de operaciones y aunque se ha avanzado en la visualización de las regiones anatómicas, los cirujanos aún necesitan de varios dispositivos para gestionar la visualización de las imágenes médicas. En el salón de operaciones es necesario llevar a cabo un conjunto de acciones para interactuar con las imágenes y la representación virtual de las herramientas quirúrgicas de forma que el cirujano pueda tener una retroalimentación visual de las herramientas de navegación y sus posibles trayectorias. En este escenario las computadoras se utilizan como última alternativa debido a que son difíciles de esterilizar y tienen muchos accesorios. No obstante la principal razón es la forma en que el personal médico interactúa con la computadora. La práctica común es que el cirujano delegue parte de la interacción a un asistente supervisado para realizar tareas comunes mediante dispositivos de interacción convencionales como el ratón y el teclado. Las tareas pueden ser apuntar, dar clic y arrastrar. Un ejemplo es que el asistente mueve un ratón guiado verbalmente por el cirujano hasta la región de interés. Luego este último presiona una especie de pedal colocado en el piso para generar eventos de teclas del ratón. Este método suele ser lento y contraproducente en el sentido que aumenta el tiempo de la intervención quirúrgica. La situación se vuelve más aguda cuando el asistente y el cirujano no actúan coordinadamente. Algunas estrategias se han llevado a cabo haciendo uso del reconocimiento de voz, pero se ha comprobado que este sistema no se ajusta a las particularidades de un quirófano donde suele existir ruido por los instrumentos presentes ahí o por el diálogo entre las personas del equipo de cirugía. Otra opción podría ser el uso de pantallas táctiles pero se ha demostrado que entorpecen el área de trabajo debido a los otros equipamientos que normalmente están situados junto al cirujano y el paciente [6].

De forma general se requiere minimizar la cantidad de dispositivos de interacción tales como teclados, ratón y *joysticks* durante la intervención quirúrgica en el salón de operaciones. Como parte del esfuerzo por desarrollar tecnologías que faciliten el uso de las computadoras en el salón de operaciones, se necesita una nueva forma de interacción que permita al cirujano manipular las imágenes y la representación virtual de la herramienta quirúrgica real para aumentar la usabilidad del sistema de navegación guiada por imágenes en los servicios de neurocirugía.

Esta situación permite formular como **problema de la investigación**:

¿Cómo lograr la interacción hombre-máquina con el sistema de neuronavegación guiada por imágenes sin la utilización de ratón ni teclado?

La investigación tiene como **objeto de estudio:** las técnicas y tareas no convencionales de interacción hombre-máquina, por lo que se propone para darle solución a este problema el siguiente **objetivo general**: desarrollar una estrategia de interacción donde el cirujano controle la representación virtual en 3D del objeto real en el neuronavegador sin la utilización de ratón ni teclado; queda constituido así el **campo de acción** como las técnicas y tareas no convencionales de interacción hombre-máquina aplicables a sistemas de navegación guiada por imágenes médicas.

Para resolver el problema científico y darle cumplimiento al objetivo general, se plantearon las siguientes **tareas de investigación**:

Tareas a cumplir por el estudiante:

1. Establecer los referentes teórico-metodológicos sobre las estrategias de interacción no convencionales que se utilicen en la neuronavegación.
2. Analizar las técnicas y tareas de interacción con entornos virtuales, tomando como base la identificación de sus principales fortalezas, debilidades y beneficios.
3. Seleccionar las técnicas y tareas de interacción que mejor se ajusten al cumplimiento del objetivo propuesto.
4. Evaluar posibles bibliotecas de software que se puedan reutilizar para implementar los procedimientos de interacción con entornos virtuales.
5. Implementar un componente de software para la interacción con las imágenes médicas.
6. Probar el componente de software implementado, a partir de un prototipo de software.

Los principales métodos científicos de investigación que se utilizarán para la realización de este trabajo serán los siguientes:

Métodos Teóricos:

- **Histórico-lógico:** este método permitirá conocer los trabajos anteriores que poseen grandes semejanzas con este trabajo en cuestión y las tendencias actuales referidas a las herramientas de neuronavegación guiada por imágenes. Además de conceptos, términos y vocabularios propios del campo de estudio que contribuirán en gran medida al entendimiento del trabajo.
- **Analítico-sintético:** mediante este método se podrán estudiar y analizar una serie de documentos y teorías relacionados con los sistemas de neuronavegación guiada por imágenes y extraer los elementos más importantes de estos, los componentes por los que están conformados, su funcionamiento y la

interacción entre ellos.

Métodos Empíricos:

- **Revisión de documentos:** con la utilización este método se podrá determinar el estado del arte con respecto al objeto de investigación y se logrará profundizar en el campo de acción.
- **Entrevistas:** con las consultas a especialistas en temas relacionados de una forma u otra con este trabajo y principalmente en visualización de imágenes, se logrará obtener información sobre el tema de la investigación.

La estructura del presente trabajo está conformada de la siguiente manera: cuenta en este orden con un resumen, una introducción, tres capítulos donde se encuentra el contenido sustancial del trabajo, conclusiones, recomendaciones, bibliografía y glosario de términos. A continuación se hace una breve descripción del contenido de cada uno de los capítulos:

Capítulo 1 "Fundamentación Teórica": aquí se analizará todo lo referente al objeto de estudio; también se examinarán los conceptos más importantes y los términos relacionados con el problema en cuestión. Además se revisarán soluciones existentes, así como se determinará el estado del arte. Por otro lado se analizarán las principales técnicas y tareas de interacción hombre-máquina que se utilizan en las aplicaciones de realidad virtual, y las bibliotecas utilizadas para el desarrollo de este tipo de aplicaciones enfocadas particularmente a la neuronavegación guiada por imágenes.

Capítulo 2 "Propuesta y descripción de la solución": en este capítulo se obtendrá una visión más general de la solución. En primer lugar se citarán las principales herramientas que se utilizarán para el desarrollo de la aplicación y se explicará de forma detallada la propuesta de solución. También se describirán las tareas y técnicas de interacción seleccionadas. Además, se detallarán los pasos de la metodología XP que describen el desarrollo de la solución.

Capítulo 3 "Construcción de la solución": en este capítulo se tratarán los aspectos fundamentales de implementación de la solución para la elaboración del prototipo funcional y se definirá el estándar de codificación. Además, se demostrará el cumplimiento del objetivo del trabajo a través del análisis del resultado de las pruebas realizadas a la solución.

Capítulo 1. Fundamentación teórica

Este capítulo estará enfocado en definir los aspectos teóricos relacionados con el tema de interacción hombre-máquina que puedan ser aplicables a la medicina específicamente en sistemas de neuronavegación. La investigación abordará la característica de interacción de los entornos virtuales en general, para luego seleccionar los aspectos que puedan ser utilizados en la solución. Por otro lado se analizarán un grupo de técnicas y herramientas especializadas en el área sujeta a la presente investigación. Además se especificarán las bibliotecas y marcos de trabajo existentes que puedan ser utilizados para el desarrollo de la solución. Por último se seleccionará la metodología de desarrollo de software que se ajuste a las características del equipo de desarrollo.

1.1 Conceptos y términos asociados al dominio de la investigación

Para lograr un mejor entendimiento de los temas que se abordarán en la presente investigación, se mostrarán una serie de conceptos identificados durante el proceso investigativo, así como la descripción de cada uno de estos, facilitando así la comprensión de los temas tratados en este documento.

1.1.1 Realidad Virtual

En la informática lo virtual es visto como algo simulado, creado por un ordenador para realizar un determinado fin. Teniendo en cuenta lo expresado en [7] la realidad virtual (RV) es un sistema interactivo que permite sintetizar un mundo tridimensional ficticio, crea así una ilusión de la realidad en la que es sumergido el usuario. Se puede decir que es una técnica fotográfica de 360^{0} que permite al usuario, atrapado en este mundo, moverse en todas direcciones. La RV puede ser considerada en varios aspectos como la interfaz definida entre los seres humanos y el ordenador. Básicamente consiste en tratar de simular todo lo que una persona es capaz de percibir en el mundo real, como son los gráficos, el sonido e incluso sensaciones de aceleración o movimiento. Todas estas sensaciones diferentes son combinadas y presentadas al usuario de forma tal que se sienta inmerso en el universo ficticio generado por el ordenador, hasta el punto de dejar de percibir la realidad y ser engañado al sentir que se encuentra en otro universo.

1.1.2 Entornos virtuales

Los entornos virtuales (EV), según lo dicho en [8], son usualmente descritos como una forma de interacción hombre-máquina que consiste en la simulación sonora y visual por computador del espacio tridimensional en el cual los usuarios pueden tener experiencias interactivas. En este espacio en tercera dimensión (3D) los usuarios se pueden comunicar entre ellos y tener la habilidad de responder a las experiencias en el área e incluso la posibilidad de cambiar la estética de la experiencia en el entorno, así como controlar el movimiento

de avatares o modificar algunas características del mismo. Los EV son a menudo llamados realidad virtual, paisaje virtual, espacio virtual o mundo virtual. El elemento clave es la interactividad que completa al entorno y lo distingue de otras formas de interacción hombre-máquina. Históricamente, los EV fueron concebidos como espacios simulados donde los usuarios podían tener experiencias similares a las de la vida real.

1.1.3 Interfaz hombre-máquina

Una de las definiciones que en [9] se otorga a la palabra interfaz es: *"conexión física y funcional entre dos aparatos o sistemas independientes"*. Por otro lado puede decirse también que es el espacio donde se realizan interacciones entre humanos y máquinas. Según lo descrito en [10] cuando una persona realiza una acción (operación o movimiento) y su contraparte (equipo o sistema) reacciona de algún modo, conforma la base del concepto de interfaz hombre-máquina (HMI por sus siglas en inglés). En [10] se apunta que el propósito de la HMI en el campo de la informática, cuando un humano introduce datos en una computadora o equipo periférico, es mejorar la facilidad de uso de estos equipos a partir de la optimización y automatización de la relación con ellos. El objetivo de esto es permitir al usuario un control efectivo sobre la máquina y lograr así su propósito mientras que la máquina se retroalimenta de información para contribuir al proceso de toma de decisiones. Generalmente el diseño de interfaces de usuario busca producir una buena experiencia al hacer fácil y eficiente el manejo de la máquina al alcanzar los resultados deseados. Esto significa que el operador necesite introducir la mínima cantidad de información para conseguir lo que desea y que la máquina minimice las salidas indeseadas. En fin, en [11] se dice que para lograr una buena HMI es necesario el estudio y la práctica de la usabilidad. Esto se refiere a crear software y otras tecnologías que las personas quieran usar, sean capaces de usar y encuentren efectivo cuando lo hagan [12].

1.1.4 Interacción

La interacción es una característica esencial de los EV. Se ha publicado mucho sobre técnicas de interacción en RV, pero la búsqueda de las que son verdaderamente intuitivas y naturales aún continúa. La interacción entre los usuarios y los EV es compleja. Los usuarios deben ser capaces de navegar a través del espacio 3D, manipular los objetos virtuales o los parámetros de control de una simulación e interactuar con la interfaz gráfica de usuario (GUI por sus siglas en inglés) en 3D dentro del EV de una manera fácil.

Por interacción debe entenderse no sólo el hecho de que el usuario pueda manipular los objetos en 3D en el entorno, sino también que pueda navegar por este y observar los objetos desde diferentes puntos de vista. De hecho, la navegación no es sino el control del usuario sobre su representación ("*the presence*" o "*the self*", dicho en [13]) en el mundo virtual, lo que en entornos multiusuario se conoce como avatar. No obstante, también podría considerarse control sobre el EV si se considera que es este el que se mueve con respecto al avatar.

Lo descrito hasta aquí no es más que una breve introducción a la interacción. Por su importancia, se le dedica a continuación una serie de apartados en los que se examinan, a su vez, los elementos que componen la propia interacción.

1.1.5 Diálogos interactivos

Para explicar el significado de los elementos básicos, en [14] se recurre a una analogía con el lenguaje natural (véase Tabla 1), haciendo corresponder los diálogos interactivos (DI) con frases. Así se indica que de la misma forma que las frases se construyen a partir de palabras, los DI se componen de secuencias de tareas de interacción. Sin embargo, como se verá a continuación, las tareas de interacción son como el significado de las palabras, y cada una de las diferentes palabras que se pueden usar para expresar un mismo significado es una técnica de interacción. Por ello, sería más correcto decir que al igual que las frases se construyen a partir de palabras, cada una expresando un significado, los DI se componen de secuencias de técnicas de interacción, cada una implementando una tarea de interacción.

Elementos de la interacción	**Elementos del lenguaje natural**
Diálogos interactivos	Frases
Tareas de interacción	Significado de las palabras
Técnicas de interacción	Palabras (el léxico)
Acciones simples con dispositivos de entrada	Letras

Tabla 1: Analogía de James Fuley [14] entre elementos de la interacción y el lenguaje natural.

1.1.6 Tareas de interacción

Según lo descrito en [14], las tareas de interacción (TI) clasifican los tipos fundamentales de información que son introducidas con las técnicas de interacción. Podría decirse también que las TI complejas se descomponen en TI más básicas. En [14] se identifican tres TI compuestas: cajas de diálogo (para especificar múltiples entradas de información), construcción (para crear objetos que requieran dos o más posiciones) y manipulación (para cambiar la forma de un objeto, arrastrarlo, rotarlo o escalarlo).

Las TI básicas se combinan para dar lugar a las TI complejas, cada una de las cuales identifica una unidad de información que el usuario puede introducir y que tiene significado en el contexto de la aplicación. En [14], estas TI básicas son cuatro: situar (una posición), seleccionar (un objeto), texto (una cadena) y cuantificar (un valor numérico).

Relacionado con las TI en el espacio en 3D, en [14] se apunta que las TI de situar y seleccionar se vuelven más

complicadas por lo que se añade una TI adicional que sería la de rotación en 3D, y se señala que con frecuencia se hace necesaria la combinación de varias TI en 3D.

Por otra parte, en [15] se identifican seis TI universales: control del punto de vista, percepción, creación de objetos o definición de modelos, selección de objetos, colocación y edición de objetos, y definición de comportamientos o relaciones entre objetos.

En [16] también se señala que muchas de las TI en EV pueden encuadrarse en una de las categorías que ahí se enumeran, y que se describen como TI universales. En este caso son tres: control del punto de vista o viajar, selección y manipulación. Además, en la taxonomía de selección/manipulación dada en esa publicación aparece otra TI más: liberar objetos. En cuanto a la manipulación en particular, en [17] se citan dos TI básicas: colocación de objetos y rotación de objetos. En ambos trabajos se nombra también la TI de control del sistema, la cual no suele incluirse entre las TI universales pues se entiende que puede ser llevada a cabo mediante tareas de selección y/o manipulación.

Continuando con [16], sus autores dividen a su vez las anteriores TI universales en subtareas. Por ejemplo, la TI selección agrupa las siguientes subtareas: indicación del objeto, indicación de selección y retroalimentación. Los propios autores señalan que es posible que algunas subtareas no estén directamente relacionadas con la TI del usuario, como en este caso la retroalimentación del sistema. En [17] se identifican cuatro TI básicas: navegación, selección, manipulación y entrada de datos. También se distingue entre TI de más alto nivel y TI básicas, indicando que las primeras pueden interpretarse como una combinación de las segundas.

En [18] se enumeran tres componentes que coinciden con algunas de las TI expuestas hasta ahora: navegación, manipulación y acceso. La última TI alude al acceso del usuario a la escena, dejándole que decida qué modelos deben incluirse en el mundo y cuáles extraer de él. Esos tres componentes se basan a su vez en otros tres, que tratan los aspectos más fundamentales del DI entre el usuario y la aplicación: control (interpreta las entradas del usuario como acciones con significado en la aplicación), retroalimentación (permite al diseñador de la aplicación guiar al usuario) y visualización (permite al usuario observar la escena y los resultados de las acciones). Estas componentes básicas podrían asemejarse, por ejemplo, a las subtareas que se describen en [16], por lo que también se les puede llamar TI básicas.

Por último, en [13] se afirma que los DI en RV no son DI como tales, sino la integración de los siguientes cuatro componentes: movimiento y navegación, manipulación de objetos e interacción con ellos, conversación con otros agentes, y características que no son de RV. Este último componente está relacionado con la introducción de datos y el estilo de interacción de la GUI. Relacionado con la manipulación de objetos y la interacción con ellos, este autor también habla de seleccionar, indicando que puede integrarse con consultar, y

cita también acciones como tocar y agarrar. La novedad en esta clasificación de [13] es la inclusión de la conversación con otros agentes como TI.

En las siguientes dos tablas se resumen las principales seis clasificaciones de las TI que se describieron en este apartado según sus autores.

Keeneth Herndon [15]	**James Foley** [14]	**Jon Barrilleaux** [18]
Control del punto de vista. Percepción. Creación de objetos, definición de modelos. Selección de objetos. Colocación y edición de objetos: • Transformaciones afines (por ejemplo trasladar, rotar, escalar, alineación) • Modificar otros parámetros (por ejemplo color, sombreado) Programación • Definir el comportamiento de objetos • Definir relaciones entre objetos	Tareas de Interacción Compuestas: • Cajas de diálogo • Construcción • Manipulación: cambiar forma, arrastrar, rotar o escalar Tareas Básicas de Interacción: • Situar • Seleccionar • Texto • Cuantificar Otras tareas 3D: • Rotación 3D	Tres componentes: • Navegación • Manipulación • Acceso Basados en otros tres: • Control • Realimentación • Visualización

Tabla 2: Tareas de interacción en aplicaciones gráficas en 3D.

Doug Bowman y L. F Hodges [16]	**David Boyd y Lakshmi Sastry** [17]	**Alistair Sutcliffe** [13]
Control del punto de vista o viajar: • Selección de dirección y destino	Navegación Selección Manipulación	Movimiento y navegación. Manipulación de objetos e interacción con ellos:

• Selección de velocidad y aceleración • Entrar condiciones Selección: • Indicación de objeto • Indicación de selección • Realimentación Manipulación: • Sujeción del objeto • Posición del objeto • Rotación del objeto • Realimentación Otras tareas: Liberar: • Indicación de soltar • Posición final del objeto Control del sistema.	Entrada de datos	• Seleccionar, consulta • Tocar, agarrar Conversación con otros agentes. Características que no son de RV: • Entrada de datos • Interacción estilo GUI

Tabla 3: Tareas de interacción en entornos virtuales.

Del análisis a las taxonomías propuestas por estos autores se concluye que difieren en varios aspectos. Aunque todos convergen en que existen TI de alto nivel, o sea más complejas y abarcadoras, que pueden ser divididas en TI más simples. La mayoría de estas bibliografías coinciden en algunas TI complejas, su relación con las TI básicas que las componen se muestra a continuación en la Tabla 4.

Manipulación	**Selección**	**Navegación**
• Traslación • Rotación	• Indicación de objeto • Indicación de selección	• Selección de dirección y destino

• Escalado		• Selección de velocidad y aceleración

Tabla 4: Coincidencias en las taxonomías de tareas de interacción.

1.1.7 Técnicas de interacción

Según [14], las técnicas de interacción (TcI) se definen como formas de usar los dispositivos de entrada para introducir información. Los autores de [16] y [15] coinciden al indicar que para cada una de las TI universales, existen muchas TcI propuestas. En [16] se describe una taxonomía que identifica un conjunto de TI, y divide cada una de ellas en subtareas. La descomposición se completa listando los posibles componentes de TcI para cada una de esas subtareas. Para una TI dada, es posible entonces componer una TcI eligiendo un componente particular de cada una de las subtareas.

Por otra parte, en [17] se explica que una TcI hace referencia al estilo de interacción específico que confiere un proceso de interacción. También coinciden con [14] y con [16] al decir que cada una de las TI básicas puede ser llevada a cabo empleando un número de posibles TcI.

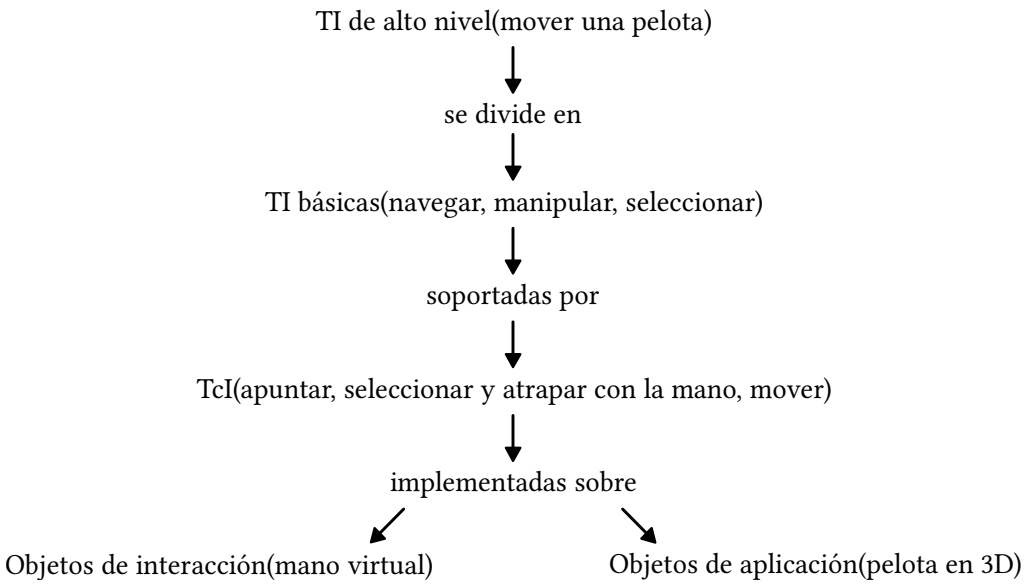

Figura 1: Ejemplo de consecuencia de una tarea de interacción de alto nivel.

1.1.8 Dispositivos físicos de interacción

Herramientas físicas usadas para implementar diferentes TcI. Varias TcI pueden implementarse con un mismo dispositivo físico de interacción (DFI). La cuestión que emerge entonces es determinar si el DFI es el más

apropiado para la TcI dada. Los DFI pueden ser catalogados según el tipo de evento que generen. En [13] se habla de dispositivos discretos y analógicos. A estos últimos, en [16] se les denominan continuos, y se añade otra categoría más, la de los dispositivos de entrada híbrida. Otra clasificación atendería a los grados de libertad (DOF por sus siglas en inglés) del dispositivo, siendo los más populares los orientados a aplicaciones en segunda dimensión (2D) como el ratón [14].

Dispositivo	Grados de libertad	Tareas representativas
Control deslizante, dial	1	Control de volumen
Ratón	2	Selección, dibujo, posición 2D
Ratón 6D, tracker	6	Colocación, control de vista
Guante, máscara	16 o más	Animación de la mano, cara
Traje de cuerpo entero	100 o más	Animación de todo el cuerpo

Tabla 5: Tareas de interacción apropiadas para diferentes dispositivos.

También en [14], se apunta que los dispositivos pueden ser evaluados a tres niveles diferentes: a nivel de dispositivo por sus características físicas, de TI, comparando dispositivos para la TI y de diálogo, evaluando secuencias de TI. A nivel de TI, se pueden establecer relaciones obvias entre DFI y TI según los DOF, como se lista en la Tabla 5, tomada de [15]. Por ejemplo, los dispositivos diseñados para aplicaciones en 2D son apropiados para tareas 2D. Pero como en [15] se explica, la diferencia de DOF entre la propia TI y el DFI puede ser un problema. Por una parte, si el espacio de la TI tiene más DOF que el DFI, la TI toma la forma de un DI más complejo que la simple manipulación directa, como también se explica en [14]. Por ejemplo, un DFI para 2D como el ratón sólo permite establecer en cada momento una correspondencia directa con dos de las dimensiones del espacio en 3D. Al añadir más dimensiones se está forzando al ratón a dar mucho más de lo que se pretendía con él. Esto suele resolverse definiendo diferentes modos entre los que el usuario cambia con los botones del ratón o los de una botonera que se presenta en pantalla llamada *dashboard* (véase [19]). Para [20] estos controles extra se interponen en la tarea principal del usuario. Otro tanto ocurre cuando el espacio de la TI tiene menos DOF que el DFI, pues puede resultar confuso si el dispositivo no está físicamente limitado, e incluso muy fatigoso según lo expuesto en [15]. Esto ocurre, por ejemplo, cuando el DFI fue diseñado para 3D pero el espacio de la TI sólo tiene una o dos dimensiones.

1.1.9 Clasificación de los dispositivos físicos de interacción

Dispositivos convencionales:

Ratón**:** uno de los dispositivos de entrada más difundidos alrededor del mundo junto con los teclados, utilizado

para facilitar el manejo de un entorno gráfico en una computadora. Es capaz de detectar su movimiento en dos dimensiones a partir de su desplazamiento sobre una superficie y transmitirlo a la computadora para traducirlo en una posición del puntero [21].

Teclado: uno de los periféricos de entrada que más se utilizan a nivel mundial, actualmente es extremadamente difícil manejar un computador sin su uso. Su diseño está inspirado en gran medida en los teclados de las antiguas máquinas de escribir y la distribución de sus teclas puede variar en dependencia del idioma para el que se vaya a utilizar [22].

Dispositivos no convencionales:

Joystick 3D**:** con este dispositivo se está en presencia de una variante de la palanca clásica de los videojuegos, en el cual se ha incorporado internamente un dispositivo de seguimiento. De esta forma es posible utilizarlo dentro de un EV asistido por computadora (CAVE por sus siglas en inglés). Los *joysticks* 3D tienen un inconveniente importante, no están estandarizados y sus características pueden diferir mucho según el contexto para el que se fabrican [23].

Figura 2: IBM Spaceball 3D [66].

Space Ball: este es un dispositivo con una superficie similar a un *trackball* como se ilustra en la Figura 2. 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 seis DOF disponibles en el espacio (posición y rotación en el espacio en 3D) [24].

Guantes de datos: los guantes de datos (Figura 3) 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. 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 [25].

Figura 3: Guantes de datos 5DT Ultra Wireless Kit [67].

Dispositivos de seguimiento óptico: 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 [26]. Ejemplo de este tipo de dispositivos es el Microsoft Kinect (Figura 4) [27].

Figura 4: Dispositivo de seguimiento óptico Microsoft Kinect [68].

Dispositivos de seguimiento inercial: se basan en pequeños dispositivos que permiten medir la aceleración con la que se mueven. Tienen un radio de acción amplio. 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. Ejemplo de este tipo de dispositivo es el Nintendo Wiimote (Figura 5) que posee varios acelerómetros y un sensor óptico infrarrojo [26].

Figura 5: Nintendo Wiimote [69].

Dispositivos de seguimiento magnético: utilizan bobinas para obtener la posición y orientación del objeto que se esté siguiendo. 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. 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 [26]. Ejemplo de este tipo de dispositivo son el NDI Polaris Aurora (véase [28] y el Polhemus Liberty (véase [29]).

Figura 6: Dispositivo de seguimiento magnético (Polhemus Liberty) [67].

Figura 7: Dispositivo de seguimiento magnético (NDI Polaris Aurora) [28].

Dispositivos de interacción por 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 [30]. 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 iOS para iPhones.

1.2 El problema de los dispositivos físicos

Una de las razones por las que las interfaces gráficas 2D son más fáciles de diseñar es porque sus conocidas TcI están basadas en unos pocos DFI, típicamente el ratón, el teclado y la pantalla. En cambio, como se explica en [31], el espacio de diseño de las interfaces de usuario 3D es significativamente mayor, los diseñadores tienen que afrontar una gran variedad de dispositivos de entrada y salida, y la aparición de nuevos productos requiere inventar nuevas TcI y reconsiderar lo aprendido hasta ese momento. Cada una de esas diferentes tecnologías ofrece una ventaja sobre las demás, pero también tiene sus inconvenientes, por lo que no se puede decir que una tecnología sea mejor que las otras en todas las aplicaciones. Además como los productos son costosos, la renovación de los DFI antiguos y la adquisición de modelos innovadores no están al alcance de cualquiera. Ya en [14] se reconocía que no todo diseñador de interfaces de usuario tiene el lujo de seleccionar el DFI más apropiado.

1.2.1 Diseño de técnicas de interacción

De nuevo, el diseñador de interfaces gráficas 2D tiene la ventaja de que las TI y las TcI son pocas, bien conocidas y están soportadas por el sistema operativo, como por ejemplo “apuntar y pulsar” o “arrastrar y soltar” con el ratón, o los atajos con el teclado. En el caso de las interfaces de usuario 3D, si bien es cierto que existen algunas TI y TcI archiconocidas, también lo es que no existe ningún conjunto estándar, y la falta de soporte de software obliga a la implementación desde cero tanto de las TI como de las TcI elegidas por el diseñador.

Aunque como se ha visto anteriormente, existe un cierto conjunto de TI que pueden considerarse universales, no está tan claro qué TcI implementar para llevarlas a cabo. En [31] se habla de dos enfoques a la hora de diseñar las interfaces de usuario en 3D:

- El primero sigue una filosofía artística, orientado a diseños a corto plazo, pero proporciona unos cimientos sobre los que fundar una filosofía más sistemática. Según lo expresado en [31] el éxito en el diseño de la interfaz 3D se basaría en: apoyarse en la investigación sobre el factor humano con respecto a la interacción con ordenadores; reutilizar TcI e ideas aportadas por otros diseñadores; recurrir a la creatividad de uno mismo para inventar nuevas TcI e interfaces; finalmente, usar los modelos y estrategias existentes en el diseño de interfaces 3D. En cuanto a “inventar” nuevas técnicas, los autores distinguen entre ideas que se apoyan en el mundo real como por ejemplo la simulación de la interacción física y las que se basan en la “magia”, tomadas de la literatura o el cine, pero también rompiendo con las ideas preconcebidas del usuario.
- La segunda filosofía es la sistemática, más lenta y metódica, caracterizada por el estudio de las TI del

usuario, TcI existentes, y las características del usuario, el sistema o el entorno que puedan afectar al rendimiento de la interfaz. Está basada en dos componentes: taxonomías, que clasifican y descomponen TI y TcI, y sirven de guía para nuevos diseños; y la evaluación, tanto formativa como incremental, integrada en el ciclo de desarrollo (evaluación secuencial en [32]) o como herramienta de investigación con bancos de pruebas (evaluación por pruebas en [16], véase también [33]).

1.2.2 Diseño de controles

Una vez más, el diseñador de interfaces gráficas 2D tiene a su alcance un conjunto de *widgets* bien conocidos que pueden usarse en una aplicación, como botones y menús. Incluso aunque su apariencia cambie de un sistema de ventanas a otro, su funcionalidad sigue siendo la misma. En cambio, en el diseño de interfaces 3D no existe ningún conjunto de controles 3D estándar al que recurrir. Un camino que suele tomarse para llegar a ese conjunto estándar consiste en preguntarse cuáles son los equivalentes de los *widgets* en el espacio 3D. Pero las posibilidades que ofrecen las interfaces 3D son mayores, pudiendo mezclar por ejemplo *widgets* y modelos de controles reales en una misma interfaz. Además, pueden introducirse nuevos controles no vistos en el mundo real ni en las interfaces gráficas 2D, como por ejemplo el árbol de conos descrito en [34].

1.3 Trabajos relacionados

En [35] se desarrolla algo similar a lo que busca el presente trabajo. Aquí René Wiegers realiza un estudio para mejorar el modo de interacción con los *widgets* en 3D de la *Visualization Toolkit* (VTK). Para ello se enfoca principalmente en la repercusión que tiene el uso de DFI de dos y seis DOF para la realización de TI por parte de los usuarios. Este investigador holandés logra responder cuál es la diferencia en cuanto a consumo de tiempo en utilizar un DFI de dos o seis DOF para completar una TI. Para probar los resultados de este trabajo se realizó un experimento con usuarios escogidos al azar en el cual se le indicaba a estos que realizaran TI con un DIF diseñado para 2D y luego con uno de 3D. El resultado general fue que se reduce el tiempo total de realización de la TI conforme aumentan los DOF del DFI hasta que se igualan a los DOF de la TI. Aunque atendiendo a los resultados individuales se observó que el tiempo de la TI de rotación se mejora al usar un DFI diseñado para aplicaciones 3D, en cuanto a la TI de traslación es más rápido un DFI de 2D y más preciso uno de 3D, por último no se vieron diferencias significativas al realizar la TI de escalado con los diferentes DFI. Basado en estos resultados el autor sugiere la combinación de ambos tipos de DFI para la manipulación de los *widgets* 3D.

El trabajo reflejado en [36] tiene varios puntos en común con la presente investigación. En este sus autores también apuestan por el uso de la VTK. Este trabajo busca resolver el problema de que esta biblioteca solo posee soporte para realizar la interacción con los datos visualizados en 3D a partir de DFI de dos DOF. Para

esto crean una interfaz multimodal para la VTK en un EV. Además se conjugan DFI de seis DOF para realizar la interacción en el espacio en 3D junto con seguimiento de cabeza (*head tracking)* para controlar la cámara en este espacio. La TI de agarrar objetos la realizan con pedales y para el control de la aplicación usan reconocimiento de voz. Aquí la máxima contribución es la integración de la visualización y el control del sistema por diferentes modalidades. La interfaz que proponen los autores permite la manipulación directa del EV. El cursor en 3D que crearon facilita la selección de objetos y vence el problema de selección de puntos en el espacio 3D.

En los trabajos mencionados anteriormente se alcanzan buenos resultados en cuanto a una interacción intuitiva con EV. Aunque al analizarlos a fondo se ve que la característica del uso de varios DFI junto con el reconocimiento de comandos de voz no se ajusta apropiadamente a un quirófano. La razón es que este entorno es muy complejo y permite muy pocas libertades en cuanto al uso de TI y TcI.

1.4 Bibliotecas para la interacción con entornos virtuales

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 (véase [37]). En lo que concierne a la visualización de los contenidos en 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; que brindan a los desarrolladores una interfaz de programación de aplicaciones (API) robusta para aplicar transformaciones a los contenidos en 3D. En el caso específico del desarrollo de aplicaciones para la medicina existen bibliotecas muy útiles como es el caso de la *Image-Guided Surgery Toolkit* (IGSTK) y la VTK.

IGSTK: Biblioteca y/o marco de trabajo sobre el lenguaje de programación C++, es de alto nivel y proporciona funcionalidades comunes orientadas a facilitar el desarrollo de aplicaciones de cirugía guiada por imagen. La IGSTK está enfocada en la robustez utilizando una arquitectura de máquina de estados. Además es multiplataforma, utilizando un entorno de compilación conocido como Cmake para gestionar el proceso de compilación de una manera independiente de la plataforma. Presenta un conjunto de componentes de alto nivel integrado con las bibliotecas de software de código abierto de bajo nivel y las API de los proveedores de hardware. La IGSTK está diseñada para permitir a los investigadores crear rápidamente prototipos y nuevas aplicaciones para la cirugía guiada por imagen. Este conjunto de herramientas proporciona funcionalidades que suelen ser necesarias tanto cuando la implementación de aplicaciones de cirugía guiada por imagen debe

integrarse con dispositivos de seguimiento óptico y electromagnético como cuando es necesaria la manipulación y visualización de datos DICOM. IGSTK se distribuye bajo licencia *Berkeley Software Distribution* (BSD); esta permite su uso sin restricciones, incluso en productos comerciales (véase [38] y [39]).

VTK: Biblioteca de clases de C++ orientada a los campos de gráficos 3D por computadora, procesamiento y visualización de imágenes. Soporta una amplia variedad de algoritmos de visualización incluyendo: escalar, vector, tensor, de textura y métodos volumétricos; junto con técnicas avanzadas de modelado como: el modelado implícito, la reducción de polígonos, el suavizado de mallas, corte, contorno, y la triangulación de Delaunay. El sistema también soporta empaquetado automatizado del núcleo de C++ en Python, Java y Tcl, por lo que las aplicaciones VTK también pueden ser escritas usando estos lenguajes de programación interpretados. Además tiene un amplio marco de visualización de información y un conjunto de *widgets* de interacción 3D. Soporta el procesamiento en paralelo, y se integra con varias bases de datos en herramientas de GUI como Qt y Tk. VTK es multiplataforma por lo que funciona en Linux, Windows, Mac y Unix. También es de código abierto y se distribuye bajo la licencia BSD (véase [40] y [41]).

1.4.1 Otras bibliotecas para la interacción con entornos virtuales

Existen varias bibliotecas que avanzan un paso en la gestión de los DFI. Los desarrolladores comienzan a incorporar dispositivos de RV a medida que van surgiendo, también incluyen el concepto de simulación desacoplada cuya principal ventaja es que la gestión de los DFI y la simulación de la RV pueden ser ejecutadas en ordenadores separados [42]. Este aspecto es especialmente útil cuando existen procesos que incrementan el consumo de recursos computacionales. 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 [43].

En el caso de la herramienta *Virtual Reality Peripheral Network* (VRPN) se soluciona esta limitación [44]. Además de proporcionar soporte para muchos DFI, introduce el concepto de dispositivo de interacción abstracto (DIA). Su diseño arquitectónico define un conjunto de requerimientos o funcionalidades necesarias para cada abstracción de dispositivos. Las aplicaciones, se implementan empleando la definición de un conjunto de DIA, para recoger la entrada del usuario y ofrecer la respuesta adecuada. Esta nueva forma de gestionar los DFI, proporciona la ventaja de que se pueda utilizar cualquier conjunto de dispositivos concretos en una aplicación en tiempo de ejecución [43].

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 HMI. Finalmente VRPN es una biblioteca que cumple con un conjunto de características identificadas como deseables en la bibliografía.

Ejemplos de estas características son: abstracción de dispositivos, transparente a la red, baja latencia, sencillo de integrar y extensible [44].

Por otro lado se investigó lo referente a OpenIGTLink [45] para verificar su posible compatibilidad con VRPN y con el presente trabajo. Esta biblioteca se encarga de definir un protocolo de comunicación, incluyendo un grupo de tipos de mensajes digitales, entre computadoras en una red de área local (LAN por sus siglas en inglés). Un mensaje, la mínima unidad de datos de este protocolo, contiene toda la información necesaria para que sea interpretada por el destinatario. El mensaje comienza con una cabecera de 58 bytes, común a todos los tipos de datos, seguido del cuerpo del mensaje. El formato del cuerpo varía en dependencia del tipo de dato especificado en la cabecera. Desde cualquier destinatario compatible se puede interpretar la cabecera del mensaje, que contiene el tamaño y el tipo de dato del cuerpo; además cada destinatario puede manejar fácilmente cualquier mensaje incluso aquellos con un tipo de dato desconocido. Por lo tanto esta estructura de dos secciones permite a los desarrolladores definir sus propios tipos de datos mientras que se mantenga la compatibilidad con otros software que no puedan interpretarlos. Este mecanismo hace más fácil el desarrollo de interfaces para OpenIGTLink y mejora la compatibilidad.

El uso de la biblioteca OpenIGTLink conjuntamente con VRPN no es necesario ya que esta última además de mantener al desarrollador ajeno a la topología de red, provee una capa de abstracción que hace que todos los dispositivos de la misma clase base parezcan el mismo; por ejemplo, todos los dispositivos de seguimiento se ven como pertenecientes al tipo vrpn_Tracker [43]. Esto significa que generan el mismo tipo de reportes. 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. 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: dispositivo de seguimiento, botón, dispositivo analógico, control deslizante y dispositivo de fuerza. Cada uno engloba un grupo de semánticas para cierto tipo de dispositivo. Existen uno o más servidores para cada tipo de dispositivo, y una clase del lado del cliente para leer los valores provenientes del dispositivo y controlar su funcionamiento [46].

1.5 Metodologías de desarrollo de software

Según [47] la metodología de desarrollo de software es "*un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevos software*".

Si se quiere construir un software con alta calidad, es necesario enfocarse en trabajar de forma organizada, donde se controle y documente todo lo relacionado con el proyecto en cuestión y puedan eliminarse los riesgos que podrían presentarse durante el desarrollo de este, lo cual no puede lograrse sin el empleo de una metodología eficaz que se adapte a las características propias del software que se está desarrollando.

Clasificar las metodologías es una tarea bien difícil por la gran cantidad de propuestas y diferencias en el grado de detalle, información disponible y alcance que poseen. No obstante a grandes rasgos y considerando la filosofía de desarrollo se pueden agrupar en dos grupos: metodologías ágiles o ligeras y metodologías pesadas.

Metodologías ágiles: son aquellas que están orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen por grupos pequeños, se hace hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso [48]. Entre estas metodologías se tiene:

- Extreme Programming (XP).
- SCRUM.
- Crystal Clear.
- Feature Driven Development (FDD).
- Dynamic Systems Development Method (DSDM).
- Adaptive Software Development (ASD).

Metodologías pesadas: están guiadas por una fuerte planificación durante todo el proceso de desarrollo, en el cual se establecen estrictamente las actividades involucradas, los roles definidos, los artefactos que se deben producir, las herramientas y notaciones que serán utilizadas así como el modelado y la documentación detallada [49]. Ejemplos de estas metodologías son:

- Software Capability Maturity Model (SW-CMM)
- Rational Unified Process (RUP)

1.5.1 Selección de la metodología de desarrollo de software

Para este trabajo se escoge una metodología de desarrollo de software ágil dadas algunas de sus características. La mayoría de estas metodologías minimiza riesgos desarrollando software en lapsos cortos. El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. Cada iteración del ciclo de vida incluye: planificación, análisis de requisitos, diseño, codificación, revisión y documentación. Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del

producto al mercado, sino que la meta es tener un prototipo (sin errores) al final de cada iteración. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto. Los métodos ágiles enfatizan la comunicación cara a cara en vez de la documentación [48].

XP es una metodología ágil y está concebida para proyectos con poco tiempo de desarrollo, equipos pequeños y con pocos roles. Propone que el diseño debe ser sencillo, que funcione con todas las pruebas y con el menor número posible de clases y métodos [50]. Además, XP ofrece una solución factible para proyectos con requisitos muy cambiantes, propone la aplicación disciplinada de las diferentes prácticas y el uso de tecnologías con un ciclo rápido de realimentación y que soporten fácilmente el cambio. Por lo antes planteado se selecciona XP como metodología de desarrollo de software.

1.5.2 Metodología de desarrollo XP

Después de analizar las diferentes metodologías de desarrollo de software, se ha llegado a la conclusión de que la más indicada para usar en el desarrollo de este trabajo es XP. Dicha metodología está basada en la simplicidad, la comunicación y el reciclado continuo de código. Además está diseñada para grupos pequeños de programadores con el objetivo de que estén en el mismo puesto de trabajo. XP es una metodología ágil que está orientada más a las personas que a los procesos. Sus principales objetivos son muy simples: la satisfacción del cliente, trata de dar al cliente el software que necesita y cuando lo necesita, y el potenciar al máximo el trabajo en grupo, tanto los jefes de proyectos como clientes y desarrolladores forman parte del equipo y están involucrados en el desarrollo del software. XP es una metodología creada por Kent Beck, se caracteriza por retomar las prácticas de uso tradicional y llevarlas al extremo, de ahí proviene su nombre. Esta metodología detalla varios valores de los métodos ágiles. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios. Además presenta los siguientes valores (véase [51]):

Comunicación: la comunicación permanente es fundamental en XP. Dado que la documentación es escasa, el diálogo frontal cara a cara entre desarrolladores, gerentes y el cliente es el medio básico de comunicación. Una buena comunicación tiene que estar presente durante todo el proyecto.

Simplicidad: la sencillez es esencial para que todos puedan entender el código, y se trata de mejorar mediante recodificaciones continuas.

Retroalimentación: básicamente el continuo contacto con el usuario, al entregarle las sucesivas versiones en funcionamiento del producto, permite que este dé su valoración y comunique, cada vez mejor, lo que realmente quiere en el producto.

Respeto: el valor del respeto en XP establece que todos en el equipo dan y reciben el respeto que merecen

como integrantes del equipo y los aportes de cada integrante son valorados por todos.

Coraje: básicamente es trabajar muy duro durante las horas dedicadas a ello.

XP define ciertas prácticas como son:

Programación por parejas: los programadores programan de a pares, dos por ordenador, de modo que mientras uno codifica directamente el otro corrige lo que este escribe, le señala vías alternativas y piensa lo próximo que hay que hacer (esto se conoce comúnmente como navegar).

Desarrollo guiado por pruebas: esto consiste en que se escribe una prueba automática y a continuación se escribe el código suficiente para pasar dicha prueba y después se rehace el código, principalmente para mejorar la legibilidad y eliminar duplicaciones.

Estandarización de código: todos los programadores del equipo deben seguir un estándar para codificar, de modo que el código sea inteligible para cada uno de los desarrolladores.

Conclusiones parciales

Después de analizar la bibliografía pudo observarse que los autores difieren en la terminología de TI; aunque están de acuerdo en que existen numerosas TI complejas que a su vez se conforman por TI más simples. La mayoría de los autores presentan coincidencias en cuanto a las TI de manipulación, selección y navegación. En cuanto al diseño de TcI para aplicaciones médicas se debe seleccionar un enfoque sistemático, dado que este le aporta mayor seriedad a la etapa de diseño de las TcI. También se estudiaron varias alternativas de bibliotecas para su posible utilización en este trabajo; aunque se determinó que la IGSTK proporciona una base fuerte para el desarrollo de aplicaciones de visualización de imágenes médicas. Después de analizar varias metodologías para el desarrollo de software se escogió XP dado que es la más adecuada en correspondencia con este trabajo.

Capítulo 2. Propuesta y descripción de la solución

En el presente capítulo se describe la propuesta de solución. Se seleccionan las herramientas que formarán parte del ambiente de desarrollo, se define la estructura principal del prototipo funcional y se explica de forma detallada la solución. Además se describirán las fases de exploración y planificación de la metodología XP seleccionada para el desarrollo de la solución.

2.1 Soluciones técnicas

A continuación se citan las principales herramientas y tecnologías utilizadas para la realización de este trabajo. Vale destacar que para el desarrollo del software se escogió C++ como lenguaje de programación.

Lenguaje de programación: fue escogido el lenguaje C++ debido a que constituye uno de los lenguajes más robustos y usados a nivel mundial, además es de alto nivel y multiparadigma pues abarca tres paradigmas de la programación: la programación estructurada, la programación orientada a objetos y la programación genérica. Este es un lenguaje diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C. Por otro lado es un lenguaje de programación estandarizado y ampliamente difundido [52]. Además de que las principales bibliotecas que se ajustan a este trabajo están enfocadas a este lenguaje principalmente.

Marco de trabajo e IDE: Qt como marco de trabajo con Qt Creator como entorno de desarrollo integrado (IDE por sus siglas en inglés) fueron las opciones seleccionadas. La utilización del lenguaje C++ en este trabajo constituye la razón por la que se escogió Qt, ya que este último está desarrollado en C++ y utiliza este lenguaje como estándar con extensiones incluyendo señales y *slots* que simplifican la manipulación de eventos. Por otro lado Qt Creator fue escogido dado que además de poseer un buen completamiento, ser ligero, multiplataforma y un buen IDE de C++, también está diseñado especialmente para ser usado conjuntamente con Qt por los propios desarrolladores del marco de trabajo [53].

Generador de sistemas de compilación: la herramienta seleccionada fue Cmake; un sistema extensible y abierto que controla el proceso de construcción de manera independiente en diferentes sistemas operativos y compiladores. Está diseñado para ser usado en conjunto con el sistema de construcción nativo de un entorno. Se utilizan ficheros simples de configuración en cada directorio fuente llamados “CMakeLists.txt” para generar ficheros de estándar de construcción. Puede compilar código fuente, crear bibliotecas, generar *wrappers* y construir ejecutables en combinaciones arbitrarias. Además soporta construcciones *in-place* y *out-of-place*, y por lo tanto se pueden realizar múltiples construcciones a partir de un único árbol fuente. También soporta la construcción de bibliotecas estáticas y dinámicas. Otra de sus buenas características es que genera un fichero caché que es diseñado para ser usado con un editor gráfico. Por ejemplo, cuando se ejecuta CMake, este

localiza los ficheros de inclusión, bibliotecas, ejecutables y puede encontrar otras directivas de construcción opcionales. Dicha información es reunida en la memoria de acceso rápido, que puede ser cambiada por el usuario antes de que se generen los ficheros de construcción nativos. Cmake está diseñado para soportar complejas jerarquías de directorios y aplicaciones que dependen de varias bibliotecas. Por ejemplo, soporta proyectos consistentes en múltiples bibliotecas donde cada una puede contener varios directorios, y la aplicación depende de las bibliotecas además de un código adicional. También puede manejar situaciones donde se deben construir ejecutables para poder generar código que es después compilado y enlazado en una aplicación final (véase [54] y [55]).

Sistema de control de versiones: Git es un sistema de control de versiones gratuito y de código abierto; distribuido y diseñado para manejar desde pequeños a grandes proyectos de forma rápida y eficiente. Por otro lado es fácil de manipular y tiene un tamaño compacto con un rendimiento increíblemente rápido. Supera a las herramientas de SCM como Subversion, CVS, Perforce y ClearCase con la capacidad de hacer ramificaciones locales sin mucho costo y la posibilidad de manejar varios flujos de trabajo. Git es un proyecto fundado por Linus Torvalds para manejar el árbol fuente del núcleo de Linux [56]. Al principio se enfocó bastante en las necesidades del desarrollo de este núcleo, pero se ha expandido más allá de su enfoque inicial y ahora es usado por otros proyectos. Cada directorio de trabajo de Git es un repositorio completo con plenas capacidades de gestión de revisiones, sin depender del acceso a la red o de un servidor central [57].

Metodología de desarrollo de software: se selecciona XP, aprovechando las características expuestas en el capítulo anterior en el epígrafe 1.5.2.

Lenguaje de modelado: se escogió el lenguaje de modelado unificado (UML por sus siglas en inglés) para realizar el modelado del análisis y diseño de la aplicación debido a que ofrece un modo estándar de visualizar, especificar, documentar y construir los artefactos del sistema. Aunque con la utilización de la metodología XP se reduce la generación de diagramas, en ocasiones puede ser muy útil para representar ideas y diseños.

Herramienta CASE: Visual Paradigm es una herramienta para realizar la ingeniería de software asistida por computadora (CASE por sus siglas en inglés) profesional que soporta el ciclo de vida completo del desarrollo de un software: análisis y diseño, construcción, pruebas y despliegue. Este software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad. Permite dibujar todo tipo de diagramas de clases, código inverso, generar código desde diagramas y viceversa además de generar documentación. Por otro lado es una herramienta libre y multiplataforma usada en los proyectos de la UCI para gestionar la calidad de software [58].

2.2 Descripción de la solución

El desarrollo del presente trabajo se centra fundamentalmente en la identificación de las TI y TcI. Todo esto con el fin de lograr DI funcionales a la hora del trabajo con el neuronavegador y la interacción del especialista con este. No se trata de obtener una aplicación completamente lista y liberada para utilizar en una consulta médica; sino demostrar a través de un prototipo de aplicación que los procedimientos para realizar la interacción con el neuronavegador son posibles a partir del uso combinado de tecnologías ya existentes.

2.2.1 Requerimientos de interacción 3D

Las TcI y TI que se proponen aquí no son adecuadas para cualquier aplicación de RV genérica. Han sido específicamente diseñadas para la aplicación, la tarea, el dominio y el usuario típico del escenario que aquí se presenta. Con el fin de mejorar la usabilidad de las técnicas desarrolladas, se consideraron principalmente las necesidades de los clínicos con respecto al campo de análisis de imágenes médicas.

Del estudio realizado hasta aquí se han definido algunos requisitos que deben tenerse debidamente en cuenta:

- Los componentes de hardware no obstructivos serán preferibles a los obstructivos. Pues los médicos prefieren usar una interfaz natural. Siguiendo este punto de vista un EV semi-inmersivo se debe preferir a uno totalmente inmersivo [59], y los dispositivos de entrada inalámbricos serán preferibles por encima de los cableados.
- Las TcI intuitivas serán preferibles a las complejas de modo que aseguren un adecuado grado de usabilidad. Los médicos no están dispuestos a utilizar nuevas herramientas si estas requieren consumo de tiempo capacitándose y configurando sistemas.
- Una interactividad en tiempo real interactivo (mayor o igual a treinta fotogramas por segundo) debe proporcionarse a fin de permitir una interacción natural.
- El DFI debe ser ergonómico. Si los clínicos tienen que usar el sistema de RV por un tiempo extendido, un dispositivo de entrada demasiado pesado o difícil de manejar puede ser rechazado.

2.2.2 Dispositivo físico de interacción seleccionado

Teniendo en cuenta todo lo anteriormente planteado se seleccionó al Nintendo Wiimote (NW en adelante) como DFI pues sus características cumplen con estos requisitos, algunas de ellas ilustradas en la Figura 5. Atendiendo además a que el NW puede ser fácilmente aislado del ambiente al introducirlo en una bolsa plástica para de esta forma evitar contaminaciones dentro del salón de operaciones. A continuación se muestran los botones que se utilizarán para realizar las TcI que luego implementarán las TI.

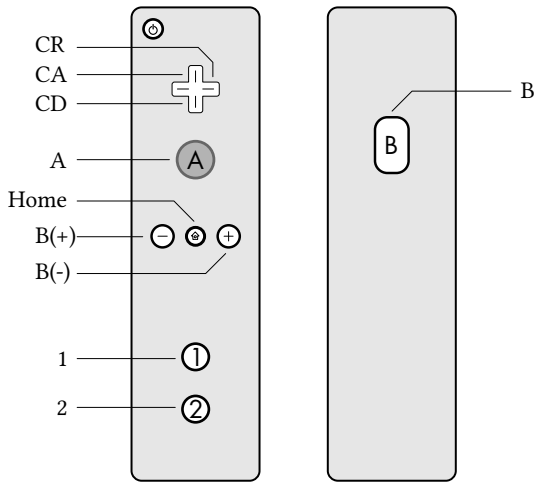

Figura 8: Notación de los botones que se utilizan en la interacción.

2.2.3 Tareas de interacción seleccionadas

Siguiendo las directrices más recientes destinadas a mejorar la facilidad de uso de las TcI 3D en aplicaciones del mundo real, el problema de la especificidad ha tenido un papel primordial en este diseño. Por ejemplo, se seleccionó la taxonomía de [17], pero no todas las TI 3D universales (navegación, selección, manipulación, entrada de datos) son necesarias en la interacción con los datos médicos volumétricos. Sólo hay un objeto en la escena, el cual fue reconstruido a partir de la superposición de los cortes axiales en 2D tomados de las tomografías computarizadas. En este contexto no hay necesidad de agarrar o seleccionar un objeto en el sistema, sino sólo apuntar a una parte de él. La TI de navegación es innecesaria en la aplicación, mientras que la de manipulación es esencial para inspeccionar mejor los datos y regiones de interés. Dado esto, cuatro tipos diferentes de especificidad han sido considerados:

Especificidad de Aplicación: la presente aplicación es una herramienta en la que se aplican tecnologías de RV para el análisis médico y biológico de imágenes.

Especificidad de tareas: gracias a la retroalimentación continua de conocimientos tomados de los médicos, se logra identificar las tareas necesarias para el área de aplicación de interés. Por ejemplo, la tarea de navegación se ha considerado innecesaria, mientras que la manipulación ha sido considerada como la más importante.

Especificidad de usuarios: los médicos han jugado un papel primordial en el diseño de las TI. Con el fin de diseñar una interacción amigable con el usuario, es necesario que los médicos puedan ser capaces de utilizar sus habilidades de la vida real para la interacción con el sistema. Desde este punto de vista, una interacción natural debería ser escogida por encima de una inteligente pero poco natural.

Durante el proceso de diseño, se ha preguntado a los médicos, paso a paso, para identificar sus necesidades de interacción.

Las tareas básicas de interacción que se identificaron fueron las siguientes:

- Rotación/traslación: para visualizar el modelo de todos los posibles puntos de vista.
- Acercar/alejar: para enfocar a mejores regiones de los datos.
- Fijar: para señalar un punto preciso de los datos visualizados.
- Seleccionar: para escoger el objeto que se desea transformar.

Para un mayor entendimiento y representación de lo antes planteado se ha modelado el funcionamiento de la interacción entre el usuario y el sistema a partir de una máquina de estado (Figura 9).

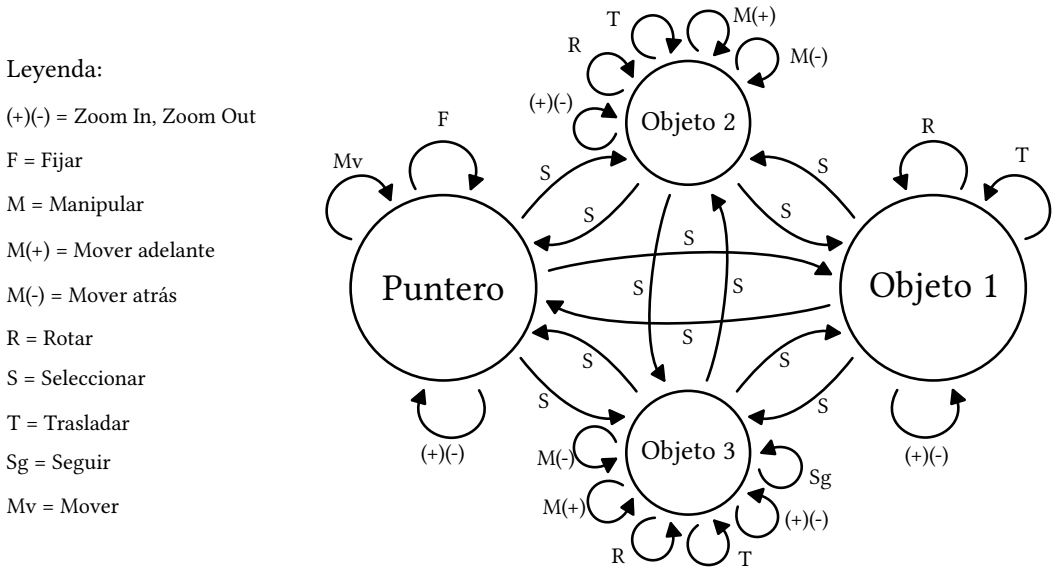

Figura 9: Máquina de estados de la interacción.

El sistema se puede encontrar en cuatro estados, los cuales comparten la TI de acercar o alejar la escena a partir del bucle *Zoom In-Out*. Desde el estado de puntero se puede manipular el puntero, al moverlo o fijarlo en un punto. En los estados de manipulación se puede aplicar transformaciones al objeto 1, al objeto 2 y al objeto 3. El primero representa al objeto en 3D reconstruido a partir de las imágenes tomográficas. El segundo permite manipular al objeto en 3D que representa a la herramienta quirúrgica. El tercero es una segunda representación de la herramienta quirúrgica. Esto se añade para aumentar la retroalimentación proporcionada al neurocirujano. Además, se permite manipular esta herramienta para comparar posibles posiciones con

respecto a la herramienta original. Las transiciones entre cada uno de las estados se ilustran en la Figura 9. Vale resaltar que la máquina de estados es a su vez un grafo fuertemente conexo. Esto posibilita que la TI de selección de estado se pueda dominar de forma sencilla.

2.2.4 Implementación de la máquina de estados

Para llevar a cabo esta tarea es necesario utilizar las clases "*igstkStateMachine, igstkStateMachineInput e igstkStateMachineState*" del marco de trabajo IGSTK, lo cual abstrae al desarrollador de este proceso. Es necesario definir las variables que se comportarán como estados y como entradas junto con los métodos a realizar en cada transición. El cómo realizar la implementación usando esta manera se explica detalladamente a continuación.

Al usar el API de la IGSTK se debe ser muy cuidadoso, pues sus desarrolladores brindan una serie de macros con el objetivo de facilitar el manejo de las funcionalidades del marco de trabajo. Cuando se hace necesario utilizar este para implementar una máquina de estados, el procedimiento que se debe seguir es el siguiente. Primero, tener claro que la máquina va a manejar el comportamiento de una clase definida por el desarrollador. Las clases de IGSTK a incluir y la declaración de la clase manejada por la máquina de estados en el encabezado (*header*) deben quedar como se ilustra en la Figura 10 y la Figura 11 respectivamente.

```
19
20  #include <IGSTK/igstkStateMachine.h>
21  #include <IGSTK/igstkCoordinateSystemInterfaceMacros.h>
22  #include <IGSTK/igstkStateMachineState.h>
23  #include <IGSTK/igstkStateMachineInput.h>
24  #include <IGSTK/igstkMacros.h>
25
```

Figura 10: Clases de la IGSTK a incluir en la cabecera

```
42
43  class EventController : public QObject ,igstk::Object
44  {
45   Q_OBJECT
46
47   igstkStandardAbstractClassTraitsMacro( EventController, igstk::Object )
48
49  private:
50          typedef igstk::StateMachine<EventController> StateMachine;
51
```

Figura 11: Declaración de la clase contenedora de la máquina de estados

También en el archivo de encabezado se declaran los estados y las entradas de la máquina de estados como se ilustra en la Figura 12.

```
51
52  //States
53          igstkDeclareStateMacro(Pointing);
54          igstkDeclareStateMacro(Manipulating);
55
56  //Inputs
57          igstkDeclareInputMacro(ChangeState);
58          igstkDeclareInputMacro(PointerFix);
59
```

Figura 12: Declaración de los estados y las entradas en la cabecera de la clase que contiene la máquina de estados.

El constructor se declara como se muestra en la Figura 13.

```
12
13  EventController::EventController(QVTKWidget *widget, vtkActor *actor)
14      :m_StateMachine(StateMachine(this))
15  {
16
```

Figura 13: Declaración del constructor de la clase que contiene la máquina de estados.

Entonces en el archivo fuente se añaden los estados, las entradas y se declaran las transiciones entre los estados. Para esto es necesario pasar por parámetro a la macro *"igstkAddTransitionMacro"* un estado inicial, una entrada, un estado destino y una acción a realizar cuando ocurra esta transición. Todo esto en el constructor de la clase en cuestión. Estando esto listo se puede llamar al método *"setReadyToRun"* para indicar que la máquina está lista para ejecutarse. Se ejemplifica lo antes planteado en la Figura 14 y la Figura 15 respectivamente.

```
91
92      //Addind states
93      igstkAddStateMacro( Pointing );
94      igstkAddStateMacro( Manipulating );
95
96      //Adding inputs
97      igstkAddInputMacro( ChangeState );
98      igstkAddInputMacro( ZoomIn );
99
100     //Adding transition
101     igstkAddTransitionMacro( Pointing,
102                              ChangeState,
103                              Manipulating,
104                              PointingToManipulating);
105
```

Figura 14: Inicialización de los estados y entradas en el constructor de la clase que contiene la máquina de estados.

```
154
155     this->m_StateMachine.SetReadyToRun();
156
```

Figura 15: Llamada para declarar que la máquina de estados está lista para ejecutarse.

2.2.5 Técnicas de interacción seleccionadas

Las TcI que se llevarán a cabo con el NW son en su mayoría realizar la acción de presionar botones en dependencia a lo que se desee hacer o visualizar. Luego a partir de los sensores de giro del NW se tomarán los valores para realizar las transformaciones a los objetos. A continuación en la Tabla 6 se detallan cada una de las TcI y se ilustra la TI que implementa. Se utilizará la notación descrita en la Figura 8 para nombrar los botones que se usarán del NW:

TI (presionar y liberar)	**Puntero**	**Objeto 1**	**Objeto 2**	**Objeto 3**
CA	No ocurren cambios	No ocurren cambios	Avanza en el vector de la orientación en que se encuentre	Avanza en el vector de la orientación en que se encuentre
CD	No ocurren cambios	No ocurren cambios	Retrocede en el vector de la orientación en que se encuentre	Retrocede en el vector de la orientación en que se encuentre
CR	No ocurren cambios	No ocurren cambios	No ocurren cambios	Toma la posición y la orientación del Objeto 2.
B (+)	Zoom (+) a toda la escena	Zoom (+) a toda la escena	Zoom (+) a toda la escena	Zoom (+) a toda la escena
B (-)	Zoom (-) a toda la escena	Zoom (-) a toda la escena	Zoom (-) a toda la escena	Zoom (-) a toda la escena
Home	Pasa al estado de manipulación del objeto 1	Pasa al estado de puntero	Pasa al estado de puntero	Pasa al estado de puntero

A	Se mantiene fijo en un punto	Se comienza a aplicar la rotación	Se comienza a aplicar la rotación	Se comienza a aplicar la rotación
B	Mover	Trasladar	Trasladar	Trasladar
1	Pasa a manipulación del Objeto 2	Pasa a manipulación del Objeto 2	Pasa a manipulación del Objeto 1	Pasa a manipulación del Objeto 2
2	Pasa a manipulación del Objeto 3	Pasa a manipulación del Objeto 3	Pasa a manipulación del Objeto 3	Pasa a manipulación del Objeto 1

Tabla 6: Acciones a realizar con cada botón del Nintendo Wimote.

2.2.6 Funcionamiento de la aplicación

VRPN es una biblioteca conformada por un conjunto de clases, diseñadas para implementar una interfaz de comunicación con los DFI, a través de una arquitectura cliente-servidor. Su funcionalidad principal radica en la posibilidad de crear servidores para comunicar los datos de los DFI 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, o sea desplegados ambos en una sola computadora sin el uso de redes. Por otro lado la descomposición de la interfaz según las funcionalidades, implica que por cada característica específica de un dispositivo se necesite implementar una funcionalidad distinta. En este caso el controlador de un dispositivo para VRPN expone al usuario varias interfaces equivalentes a sus respectivos tipos de dispositivos. Por ejemplo el servidor para el dispositivo NW es un caso típico de la situación descrita en este párrafo. El NW 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 tres dispositivos distintos, mientras se mantengan las funcionalidades anteriores.

Relacionado con el desarrollo de aplicaciones de RV basadas en VRPN y una serie de aspectos con mayor orden de complejidad estuvo vinculada la tesis de maestría del tutor del presente trabajo Msc. Ernesto de la Cruz Guevara Ramírez [43]; el cual desarrolló entre otros componentes un servidor basado en VRPN. El servidor

implementado brinda soporte a dispositivos tradicionales incluyendo al NW. Al servidor se le incluye el soporte a un nuevo dispositivo de seguimiento 3D, basado en visión por computadoras. El nuevo DFI cumple con todas las especificidades abordadas en el epígrafe 2.2.1. A partir de este servidor las aplicaciones del lado del cliente pueden tener acceso a los datos generados por los DFI.

A partir de los planteamientos anteriores y el trabajo realizado en [43], se propone hacer uso de la arquitectura cliente-servidor para implementar un cliente usando VRPN que utilice los datos del servidor construido. Esta arquitectura permite generar y manejar los eventos de forma desacoplada, por lo que garantiza fluidez en la interacción con el nuronavegador. De esta forma el NW se conecta del lado del servidor para que este se encargue de obtener los eventos que genera. Luego a partir de la conexión establecida desde el lado del cliente se toman los valores que envía el servidor para traducirlos a eventos entendibles para el neuronavegador.

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(constchar * cname , . . .);

Cuando la aplicación cliente gestiona su conexión utiliza la función anterior y 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 “d5l306pc5.uci.cu”, la cadena de conexión sería “Tracker0@d5l306pc5.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 utilice 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* mediante el protocolo para el control de la transmisión (TCP por sus siglas en inglés) para escuchar y envía un datagrama mediante el protocolo de datagrama de usuario (UDP por sus siglas en inglés) 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. Una conexión entre el servidor y los clientes finalizará 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 [43].

2.2.7 Descripción de la aplicación cliente

Con los aspectos definidos en el epígrafe anterior, solo resta consumir los datos producidos por el DFI que provienen del servidor. Luego a partir de estos datos se manejan los eventos de botones y sensores de giro y movimiento. Haciendo uso de la máquina de estados de la Figura 9 se manejan las transformaciones a los objetos representados en la escena en 3D. Para esto se utilizaron el marco de trabajo IGSTK y las bibliotecas VRPN y VTK, combinándolos para dar solución al problema de la investigación. El flujo de datos se ilustra en la Figura 16.

Figura 16: Flujo de datos desde el dispositivo de interacción hasta el neuronavegador.

Diagrama de clases:

La metodología XP plantea que se pueden crear diagramas para un mejor entendimiento de las tareas, flujos, métodos de desarrollo y de las funcionalidades, siempre que su creación no implique mayor esfuerzo que la implementación de estos. Siguiendo este principio se elaboró el diagrama de clases del diseño (Figura 17), el cual ayuda a comprender los conceptos más importantes del contexto y define las colaboraciones entre ellos. El diagrama que se muestra solo busca ilustrar las clases diseñadas y su relación con las clases utilizadas de las bibliotecas VRPN, VTK e IGSTK.

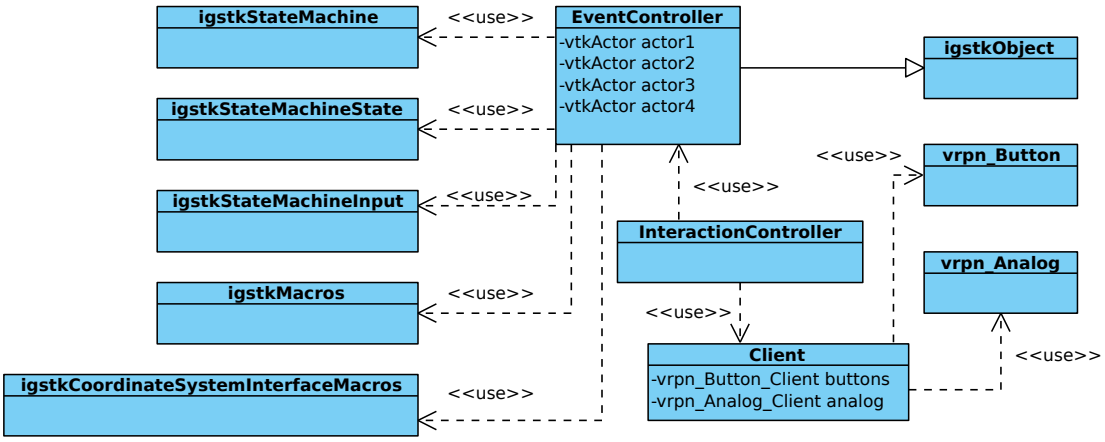

Figura 17: Diagrama de clases.

2.3 Exploración

La metodología XP, comienza en su ciclo de vida con la fase de exploración, proponiendo definir durante esta etapa el alcance general del proyecto. Además, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo se realiza una familiarización con las herramientas, tecnologías y prácticas que se emplearán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. Las estimaciones realizadas en esta fase son primarias, ya que están basadas en datos de alto nivel y podrían variar cuando se analicen con mayor detalle en cada iteración. Para esta fase se sugiere una extensión de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con las tecnologías [60].

2.3.1 Actores del sistema

Se define como actores del sistema, a aquellas personas que interactúan de una forma u otra con los mecanismos propuestos, demandando una o más funcionalidades de estos, que pueden estar vinculadas al proceso de desarrollo o no. Esto incluye tanto a los operadores humanos como a sistemas externos. Particularmente en este trabajo existe un solo actor, o sea el neurocirujano, que interactuará con el sistema de neuronavegación guiada por imágenes.

Actor del Sistema	Descripción
Neurocirujano	Es la persona encargada de realizar la interacción con el sistema a partir de la ejecución de las funcionalidades: rotar, acercar/alejar y trasladar el objeto 3D, así como fijar el puntero en un sitio de interés.

Tabla 7: Actor del sistema.

2.3.2 Historias de usuario

Las historias de usuario (HU) son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento las HU pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada HU es lo suficientemente comprensible y delimitada para que los desarrolladores puedan implementarla en unas semanas [61]. Los desarrolladores dialogarán de forma directa con los clientes cuando llegue la hora de la implementación, para así obtener los detalles necesarios.

Los contenidos de las fichas de las HU quedan estructurados de la siguiente manera:

Número: a cada HU se le asigna un número para facilitar su identificación por parte del equipo de desarrollo.

Nombre de HU: nombre descriptivo de la HU.

Fecha: fecha en la cual fue redactada la HU.

Usuario: actor protagonista de la HU.

Prioridad en el Negocio: grado de prioridad que se le asigna el cliente a la HU en dependencia de sus necesidades. Los valores que pueden tomar son: alta, media o baja.

Riesgo en Desarrollo: grado de complejidad que le asigna el equipo de desarrollo a la HU luego de analizarla. Los valores que pueden tomar son: alto, medio o bajo.

Tipo de Actividad: define el tipo de actividad en la HU. Los valores que puede tomar son: nueva, corrección y mejora.

Iteración Asignada: número de la iteración en la cual será implantada la HU.

Puntos Estimados: unidades de tiempo estimadas por el equipo de desarrollo para darle cumplimiento a la HU. Una unidad de tiempo equivale a una semana de trabajo de 40 horas.

Puntos Reales: unidades de tiempo reales que el equipo de desarrollo necesitó para darle cumplimiento a la HU. Una unidad de tiempo equivale a una semana de trabajo de 40 horas.

Descripción: descripción simple sobre lo que debe hacer la funcionalidad a la que se hace referencia.

Observaciones: aquellos detalles relevantes que serán resueltos tras la conversación del equipo desarrollador con el cliente.

A continuación se muestran las HU definidas por el equipo de desarrollo.

Historia de usuario	
Número: 1	Nombre de HU: Rotar
Fecha: 10-02-2015	Usuario: Neurocirujano
Prioridad en el negocio: alta	Riesgo de desarrollo: medio
Tipo de actividad: nueva	Iteración Asignada: 1
Puntos estimados: 1	Puntos reales: 2
Descripción: el sistema debe ser capaz de traducir a comandos de interacción los datos recibidos desde el servidor y generados por el DFI para aplicar al objeto en 3D la transformación de rotación.	
Observaciones: el usuario debe ser capaz de ver la representación del objeto 3D desde varios puntos de vista para analizar puntos de interés que no se muestran inicialmente.	

Tabla 8: Historia de usuario 1.

Historia de usuario	
Número: 2	Nombre de HU: Acercar/Alejar
Fecha: 10-02-2015	Usuario: Neurocirujano
Prioridad en el negocio: alta	Riesgo de desarrollo: medio
Tipo de actividad: nueva	Iteración asignada: 1
Puntos estimados: 1	Puntos reales: 2
Descripción: el sistema debe ser capaz de traducir a comandos de interacción los datos recibidos desde el servidor y generados por el DFI para aplicar al objeto en 3D la transformación de *zoom*.	
Observaciones: es necesario que el neurocirujano pueda acercarse a puntos de la representación del objeto 3D lo suficiente como para visualizarlos mejor.	

Tabla 9: Historia de usuario 2.

Historia de usuario	
Número: 3	Nombre de HU: Trasladar
Fecha: 10-02-2015	Usuario: Neurocirujano

Prioridad en el negocio: media	Riesgo de desarrollo: medio
Tipo de actividad: nueva	Iteración asignada: 2
Puntos estimados: 1	Puntos reales: 2
Descripción: el sistema debe ser capaz de traducir a comandos de interacción los datos recibidos desde el servidor y generados por el DFI para aplicar al objeto en 3D la transformación de traslación.	
Observaciones: aunque trasladar al objeto en 3D es una tarea de interacción de menor importancia dado que existe solo un objeto 3D en la escena, es bueno que el usuario pueda trasladar este objeto.	

Tabla 10: Historia de usuario 3.

Historia de usuario	
Número: 4	Nombre de HU: Fijar puntero
Fecha: 10-02-2015	Usuario: Neurocirujano
Prioridad en el negocio: media	Riesgo de desarrollo: medio
Tipo de actividad: nueva	Iteración asignada: 2
Puntos estimados: 1	Puntos reales: 2
Descripción: el sistema el sistema debe ser capaz de traducir a comandos de interacción los datos recibidos desde el servidor y generados por el DFI para lograr mantener al puntero en un punto invariable.	
Observaciones: es necesario que en el caso de que el neurocirujano requiera señalar un punto de interés, pueda hacerlo sin preocuparse por que el puntero se mueva de lugar por variaciones de posición indeseadas del dispositivo físico de interacción.	

Tabla 11: Historia de usuario 4.

Historia de usuario	
Número: 5	Nombre de HU: Mover puntero
Fecha: 10-02-2015	Usuario: Neurocirujano
Prioridad en el negocio: alta	Riesgo de Desarrollo: medio
Tipo de actividad: nueva	Iteración asignada: 1
Puntos estimados: 1	Puntos reales: 2

Descripción: el sistema debe ser capaz de traducir a comandos de interacción los datos recibidos desde el servidor y generados por el DFI para lograr mover al puntero.
Observaciones: es imprescindible lograr el movimiento del puntero por la ventana de la aplicación dado que se necesita alcanzar determinados puntos en los que se encuentran datos u otros objetos para señalar regiones de interés.

Tabla 12: Historia de usuario 5.

Historia de usuario	
Número: 6	Nombre de HU: Seleccionar
Fecha: 10-02-2015	Usuario: Neurocirujano
Prioridad en el negocio: alta	Riesgo de desarrollo: medio
Tipo de actividad: nueva	Iteración asignada: 1
Puntos estimados: 1	Puntos reales: 2
Descripción: el sistema debe ser capaz de traducir a comandos de interacción los datos recibidos desde el servidor y generados por el DFI para lograr seleccionar un determinado objeto de la escena en 3D con el objetivo de manipularlo.	
Observaciones: en la escena se encuentran representados varios objetos en 3D. Para interaccionar con ellos es necesario seleccionarlo antes.	

Tabla 13: Historia de usuario 6.

2.3.3 Diseño de casos de prueba

La metodología XP propone que después de que se realicen las HU definidas por el cliente se deben diseñar los casos de pruebas que cada HU debe vencer en la fase de pruebas. Las pruebas de aceptación son consideradas pruebas de caja negra. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos [62]. Las tablas con los diseños de casos de pruebas se encuentran en el epígrafe 3.5 del Capítulo 3, donde además se le incluyen los resultados a los que se arriba a partir de su ejecución.

2.4 Fase de planificación

La planificación es una fase corta donde el cliente establece la prioridad de cada HU, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación y el resultado es un plan de entregas. Esta fase

dura unos pocos días. Las estimaciones de esfuerzo asociadas a la implementación de las HU son establecidas por los programadores utilizando como medida el punto, lo que equivale a una semana ideal[1] de programación. Las HU generalmente valen de uno a tres puntos. La planificación se puede realizar basándose en el tiempo o el alcance. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuantos puntos se pueden completar. Al planificar según el alcance, se divide la suma de puntos de las HU seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación [60].

2.4.1 Estimación de esfuerzos por historias de usuario

Teniendo en cuenta la prioridad que tiene una determinada HU en el desarrollo del componente se decide en qué iteración será implementada. Las HU que cuentan con mayor importancia por ser funcionalidades indispensables para la aplicación deben ser implementadas en las primeras iteraciones del ciclo de desarrollo. A continuación se muestra mediante la Tabla 14 la planificación de las diferentes HU para cada iteración teniendo en cuenta su prioridad.

No	Historia de usuario	Prioridad	Esfuerzo estimado en puntos
1	Rotar	alta	1,0
2	Acercar/Alejar	alta	1,8
3	Trasladar	media	1,6
4	Fijar puntero	media	0,8
5	Mover puntero	alta	0,8
6	Seleccionar	media	1,0

Tabla 14: Estimación de esfuerzo por historias de usuario.

2.4.2 Plan de iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El plan de entrega está compuesto por iteraciones de no más de tres semanas de duración. En la primera iteración se intenta establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra estableciendo las HU que fuercen la creación de la arquitectura antes mencionada. Sin embargo esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para la entrega [61].

1 Trabajando 40 horas a la semana.

Una vez definidas las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, se decidió por el equipo de desarrollo construir el sistema en dos iteraciones, las cuales se describen de manera más detallada a continuación:

Iteración 1: esta iteración tiene como objetivo darle cumplimiento a las HU que se consideraron con mayor importancia para el desarrollo de la herramienta. Al concluir dicha iteración el sistema contará con todas las funcionalidades descritas en las HU 1, HU 2, y la HU 5 las cuales aluden a la rotación, acercamiento y alejamiento con respecto a la representación 3D de los objetos reales y al movimiento del puntero.

Iteración 2: esta iteración tiene como objetivo darle cumplimiento a la HU 3, la cual responde a la traslación de la representación 3D de los objetos reales, a la HU 4 la cual está relacionada con mantener el puntero en un punto fijo y de interés junto con la HU 6 que apunta a seleccionar determinado objeto en 3D de la escena.

2.4.3 Plan de duración de iteraciones

Como parte del ciclo de vida de un proyecto guiado por la metodología de desarrollo XP, se crea el plan de duración de las iteraciones que se llevarán a cabo durante el desarrollo del proyecto. Este plan tiene como objetivo fundamental mostrar la duración de cada una de las iteraciones en las que está dividida la fase de desarrollo del proyecto, además se muestra el orden en que serán implementadas las HU en cada iteración según la prioridad asignada por el cliente.

Iteración	Historias de usuario	Duración total de la iteración
Iteración 1	Rotar Acercar/Alejar Mover puntero	3,6 semanas 144 horas
Iteración 2	Trasladar Fijar puntero Seleccionar	3,4 semanas 136 horas

Tabla 15: Plan de duración de las iteraciones.

2.4.4 Plan de entrega

Como resultado de la fase de planificación se genera el plan de entrega que plantea una fecha para la entrega de cada iteración.

No.	Historias de usuario	**Final Iteración 1** (27/3/2015)	**Final Iteración 2** (21/4/2015)

1	Rotar	V1.0	
2	Acercar/Alejar	V1.0	
3	Trasladar		V1.1
4	Fijar puntero		V1.1
5	Mover puntero	V1.0	
6	Seleccionar		V1.1

Tabla 16: Plan de entrega de las iteraciones.

2.5 Arquitectura del sistema

A partir del proceso de diseño de la arquitectura de un sistema, se define una solución para sus requisitos técnicos y operacionales. Este proceso define qué componentes conforman el sistema, cómo se relacionan entre ellos, y cómo mediante su interacción llevan a cabo las funcionalidades especificadas.

Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma. En este modelo, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio) [63].

Se seleccionó la arquitectura cliente-servidor para su uso en la aplicación. El uso de VRPN para la gestión del DFI es la razón que fundamenta esta selección (véase epígrafe 1.4.1).

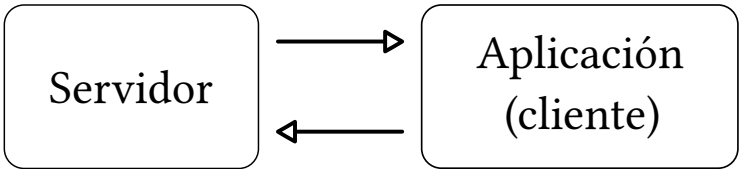

Figura 18: Arquitectura cliente servidor.

Conclusiones parciales

Con la selección de las TI de manipulación y selección y su respectiva implementación a partir de las TcI rotar y presionar botones realizadas con el NW como DFI se logró una estrategia de interacción para el sistema de neuronavegación guiada por imágenes. Se propone como solución realizar los DI a partir de un grafo fuertemente conexo por la compatibilidad que existe entre este tipo de representación y su implementación mediante una máquina de estados. Por otro lado la utilización de VRPN reduce el consumo de recursos

computacionales gracias a su orientación hacia la arquitectura cliente-servidor. Además VRPN permite que se gestionen los DFI del lado del cliente de forma genérica. Gracias a esto se elimina la necesidad de manejar los DFI dependiendo de la interfaz que brindan sus fabricantes. También se resuelve el problema de variedad de los DFI no estandarizados. Con el uso de VRPN no importa qué DFI se conecte del lado del servidor siempre que mantenga las funcionalidades del DFI que se espera del lado del cliente.

Capítulo 3. Construcción de la solución

En el presente capítulo se describe la fase de implementación a través de las tarjetas de Contenido, Responsabilidad y Colaboración (CRC), el estándar de codificación y los patrones de diseño utilizados. Además se realizan las pruebas de aceptación a partir de las HU como parte de la fase de pruebas de la metodología XP.

3.1 Diseño del sistema

La metodología XP sugiere que hay que conseguir diseños simples y sencillos. Es necesario procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible e implementable que a la larga costará menos tiempo y esfuerzo desarrollar. Dicha metodología, no requiere la descripción del sistema mediante diagramas de clase utilizando la notación UML, sino que en su lugar se guía por técnicas como las tarjetas CRC. Esto no implica que no se empleen dichos diagramas para obtener una mejor visión y comunicación entre el equipo de trabajo, siempre y cuando no posea una alta complejidad y defina información importante.

3.1.1 Tarjetas CRC

El uso de las tarjetas C.R.C permite al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica. Una tarjeta CRC no es más que una ficha de papel o cartón que representa a una entidad del sistema a las cuales asigna responsabilidades y colaboraciones. Las tarjetas CRC representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad. La mayor ventaja de las tarjetas CRC es permitir reducir el modo de pensar procedural y apreciar la tecnología de objetos. Las tarjetas CRC permiten que todo el equipo del proyecto contribuya al diseño. A continuación se definen las tarjetas CRC del sistema.

EventController	
Súper Clase: igstk::Object, QObject	
Responsabilidad	**Colaboración**
Manejar los eventos con la máquina de estado Manejar la visualización en el QVTKWidget.	Client QVTKWidget

Tabla 17: Tarjeta CRC de la clase EventController.

Client	
Súper Clase: QObject	
Responsabilidad	**Colaboración**
Establecer la conexión con el servidor VRPN. Actualizar los eventos tomados del servidor. Proporcionarle los datos a EventController para su funcionamiento.	vrpn_Button_Client vrpn_Analog_Client

Tabla 18: Tarjeta CRC de la clase Client.

3.1.2 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a este tipo de problemas que están sujetos a contextos similares. Se deben tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

En el desarrollo de la aplicación se emplearon los siguientes **patrones de software para la asignación general de responsabilidades (GRASP** por sus siglas en inglés**)** que definen buenas prácticas de programación [64]:

- **Experto:** Asigna una responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Esto nos indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento).
- **Creador:** guía la asignación de responsabilidades relacionadas con la creación de objetos. Se aplica en todos los casos donde una clase tiene la responsabilidad de crear una nueva instancia de la otra. El patrón creador ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, que usa directamente las instancias creadas del objeto y almacena o maneja varias instancias de la clase.
- **Bajo Acoplamiento:** el bajo acoplamiento fomenta el aumento de la reutilización y la eliminación de

las redundancias, creando clases más independientes y con mayor resistencia al impacto de los cambios, que aumentan la productividad y la posibilidad de reutilización. Este patrón se basa en la idea de tener las clases lo menos ligadas entre sí. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

- **Alta Cohesión:** la cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase, además de que una alta cohesión garantiza que clases con responsabilidades estrechamente relacionadas no realicen un trabajo enorme, y que cada elemento del diseño debe realizar una labor única dentro del sistema.

Estos patrones se utilizaron de forma general en la implementación de las clases de la solución como una guía del diseño de software para garantizar la calidad del código.

Los patrones de la Banda de los Cuatro (GOF, por sus siglas en inglés**) empleados en el diseño de la aplicación son** [65]**:**

- **State (estado):** el patrón estado permite a un objeto cambiar su comportamiento cuando su estado interno cambia. Se usa en la clase EventController para manejar los estados por los que transita la visualización.
- **Singleton (Instancia Única):** garantiza la existencia de una única instancia de una clase y la creación de un mecanismo de acceso global a las propiedades y funcionalidades de dicha instancia. El acceso al servidor en el sistema se diseñó mediante la creación de una única instancia para impedir que se creen múltiples conexiones innecesarias. Se encuentra en la implementación de la clase Client.
- **Observer (Observador):** este patrón define una relación de dependencia del tipo uno a muchos entre objetos de manera que cuando uno de ellos cambia su estado, notifica este cambio a todos los dependientes. Este patrón por lo general es flexible y sencillo a la vez. Se utiliza cuando el desarrollador necesita notificar a otros objetos de un evento. En principio, lo que sucede es que un objeto (llámese observador) se inscribe a otro objeto (llámese sujeto) y este le avisa cuando un evento es disparado (o cuando el estado del sujeto ha cambiado). Se usa principalmente para establecer la comunicación entre las clases EventController (observador) y Client (sujeto).

3.2 Estándar de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento

del código. La metodología XP enfatiza en la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación. Seguidamente se enumeran una serie de convenciones de denominación del código similares a las usadas en el desarrollo de la IGSTK.

En general los nombres serán formados usando el cambio de minúscula por mayúscula para indicar palabras separadas y no se usarán guiones bajos. Los nombres de las variables serán escogidos cuidadosamente con el objetivo de representar lo mejor posible su significado detrás del código. Además se explicarán los nombres, y las abreviaturas serán usadas en el único caso de que sea muy común su uso. Mientras no resulte en nombres muy largos, este puede servir como la documentación de lo que representa. Este estándar de codificación en parte se basa en el utilizado por los desarrolladores de IGSTK.

Clases: se nombrarán comenzando con letra mayúscula y en consecuencia a su rol. En caso de que el nombre sea compuesto entonces todo el nombre se escribirá junto y al comenzar cada uno lo hará con mayúscula, como por ejemplo *"EventController"*.

Archivos: estos deben llevar el mismo nombre que la clase pero sin mayúsculas. Los archivos de encabezado llevarán la extensión *".h"*, mientras que los de implementación tomarán la extensión *".cpp"*.

Métodos y funciones: las funciones globales y los métodos de las clases, ya sean estáticos o miembros de clases, se nombrarán comenzando con minúsculas. En caso de que el nombre sea compuesto entonces sus restantes partes comenzarán con mayúsculas para diferenciar uno de otro. Cuando se hace referencia a los métodos de la clase en el código, se debe utilizar un puntero explícito; su uso ayuda a aclarar el método exacto, de donde es originario y desde donde está siendo invocado. Del mismo modo el espacio de nombres global (::) se debe utilizar cuando se hace referencia a una función global.

Variables miembros: las variables miembros se nombrarán comenzando con una letra minúscula. En caso de que el nombre sea compuesto entonces sus restantes partes comenzarán con mayúsculas. Todas las variables se declararán como privadas con métodos *"get"* y *"set"* para acceder a ellas. Cuando se hace referencia a una variable, se debe utilizar un puntero. Las clases derivadas deben utilizar la combinación de esto y métodos *"get"* y *"set"*.

Variables locales: las variables locales comenzarán en minúsculas. Existe más flexibilidad en la asignación de nombres de variables locales.

Parámetros de plantilla: seguirán las reglas normales con el nombramiento, salvo que deberían empezar con una T, o una V mayúscula. Los de tipo empiezan con la letra T, mientras que los parámetros de plantilla comenzarán con la letra V.

Typedefs: se recomienda el uso de "t*ypedef*" pues se mejora significativamente la legibilidad del código y facilita la declaración de combinaciones sintácticas complejas. Desafortunadamente, la creación de *"typedef"* es equivalente a crear otro lenguaje de programación. Por lo tanto deben utilizarse moderadamente. La regla principal para su creación es que deben terminar con la palabra *"Type"*. Ejemplo: typedef Tpixel PixelType.

Uso de guiones bajos: no deben usarse guiones bajos. La única excepción es cuando se definen las variables miembros, variables de preprocesador y macros. En estos casos, se permiten guiones bajos para separar las palabras.

3.3 Desarrollo de iteraciones

Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una de las iteraciones una versión funcional que implementa las HU asignadas a la iteración. Para lograr un análisis y desarrollo más detallado de las HU, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. El cliente, por lo tanto, también debe participar activamente durante la fase de desarrollo de iteraciones del ciclo. Las iteraciones son también utilizadas para medir el progreso del proyecto.

3.4 Implementación

En esta fase se realiza la implementación de las HU que fueron seleccionadas por cada iteración. Primero se lleva a cabo un chequeo del plan de iteraciones por si es necesario realizar modificaciones. Como parte de este plan se crean tareas de programación para ayudar a organizar la implementación exitosa de las HU.

3.4.1 Tareas de ingeniería por cada historia de usuario

Con el objetivo de profundizar sobre los artefactos generados en las fases de Exploración y Planificación, se creó un grupo de actividades por cada HU de la iteración 1(Tabla 19) y la iteración 2 (Tabla 20).

Iteración 1.

No.	Historias de usuario	No.	Tareas por historias de usuario
1	Rotar	1.1	Conectarse al servidor
		1.2	Consumir los datos del servidor
		1.3	Generar evento de interacción para rotar
		1.4	Visualizar el objeto en 3D rotado
2	Acercar/Alejar	2.1	Generar evento de interacción para realizar el *zoom* a la escena en 3D.

		2.2	Visualizar la escena en 3D acercada o alejada
5	Mover puntero	5.1	Generar evento de interacción para mover el puntero
		5.2	Visualizar el movimiento del puntero

Tabla 19: Tareas de ingeniería por historias de usuario de la primera iteración.

Iteración 2.

No.	Historias de usuario	No.	Tareas de ingeniería por historias de usuario
3	Trasladar	3.1	Generar evento de interacción para trasladar el objeto 3D
		3.2	Visualizar el objeto en 3D trasladado
4	Fijar puntero	4.1	Generar evento de interacción para fijar el puntero
		4.2	Inmovilizar el puntero en el punto que se encuentre.
6	Seleccionar	6.1	Generar evento de interacción para seleccionar un objeto.
		6.2	Aplicar las secuencias de tareas de interacción al objeto seleccionado.

Tabla 20: Tareas de ingeniería por historias de usuario de la segunda iteración.

Descripción de las tareas de ingeniería: una vez que se conozcan las tareas creadas por cada HU es necesario realizar una descripción general de cada una de ellas, identificando el número de la tarea, programador encargado y el número de la HU a la que pertenece. Las tareas de ingeniería de las HU correspondientes a la iteración 1 pueden ser localizadas en las siguientes tablas.

Tareas de ingeniería de la primera iteración:

Tarea de ingeniería	
No. de tarea: 1.1	**No. de HU:** 1
Nombre de tarea: Conectarse al servidor.	
Tipo de tarea: implementación	Puntos estimados: 0,3
Fecha inicio: 02-03-2015	Fecha fin: 04-03-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: tiene como fin, establecer la conexión con el servidor VRPN para consumir los eventos generados por el DFI. Es la primera tarea a realizar pues el resto tiene dependencias de ella.	

Tabla 21: Tarea de ingeniería 1.1.

Tarea de ingeniería	
No. de tarea: 1.2	**No. de HU:**1
Nombre de tarea: Consumir los datos del servidor.	
Tipo de tarea: implementación	Puntos estimados: 0,2
Fecha inicio: 04-03-2015	Fecha fin: 06-03-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: luego de lograr la conexión con el servidor, es necesario tomar los datos que este envía para implementar la interacción con los objetos en 3D a partir de ellos.	

Tabla 22: Tarea de ingeniería 1.2.

Tarea de ingeniería	
No. de tarea: 1.3	**No. de HU:**1
Nombre de tarea: Generar evento de interacción para rotar.	
Tipo de tarea: implementación	Puntos estimados: 0,4
Fecha inicio: 06-03-2015	Fecha fin: 10-03-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: se hace necesario a partir de los datos que envía el servidor permitir rotar los objetos en 3D.	

Tabla 23: Tarea de ingeniería 1.3

Tarea de ingeniería	
No. de tarea: 1.4	**No. de HU:**1
Nombre de tarea: Visualizar el objeto en 3D rotado.	
Tipo de tarea: implementación	Puntos estimados: 0,1
Fecha inicio: 10-03-2015	Fecha fin: 10-03-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: para probar principalmente que se puede rotar un objeto en 3D se necesita visualizar esta rotación a partir de un prototipo de software.	

Tabla 24: Tarea de ingeniería 1.4.

Tarea de ingeniería	
No. de tarea: 2.1	**No. de HU:** 2
Nombre de tarea: Generar evento de interacción para realizar el zoom a la escena en 3D.	
Tipo de tarea: implementación	Puntos estimados: 1,0
Fecha inicio: 11-03-2015	Fecha fin: 17-03-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: se hace necesario a partir de los datos que envía el servidor permitir acercar o alejar los objetos en 3D para visualizar mejor las regiones de interés.	

Tabla 25: Tarea de ingeniería 2.1.

Tarea de ingeniería	
No. de tarea: 2.2	**No. de HU:** 2
Nombre de tarea: Visualizar la escena en 3D acercada o alejada.	
Tipo de tarea: implementación	Puntos estimados: 0,8
Fecha inicio: 18-03-2015	Fecha fin: 23-03-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: para probar principalmente que se puede acercar o alejar los objetos en 3D se necesita visualizar esta transformación a partir de un prototipo de software.	

Tabla 26: Tarea de ingeniería 2.2.

Tarea de ingeniería	
No. de tarea: 5.1	**No. de HU:** 5
Nombre de tarea: Generar evento de interacción para mover el puntero.	
Tipo de tarea: implementación	Puntos estimados: 0,5
Fecha inicio: 24-03-2015	Fecha fin: 26-03-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: se hace necesario permitir el movimiento del puntero por la escena, a partir de los datos que	

envía el servidor para señalar puntos de interés sobre los objetos en 3D.

Tabla 27: Tarea de ingeniería 5.1.

Tarea de ingeniería	
No. de tarea: 5.2	**No. de HU:** 5
Nombre de tarea: Visualizar el movimiento del puntero.	
Tipo de tarea: implementación	Puntos estimados: 0,3
Fecha inicio: 26-03-2015	Fecha fin: 27-03-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: para probar principalmente que se puede mover el puntero se necesita visualizar este movimiento a partir de un prototipo de software.	

Tabla 28: Tarea de ingeniería 5.2.

Las tareas de ingeniería relacionadas con las HU que presentan prioridad media para el cliente y pertenecientes a la iteración 2 pueden ser localizadas en las siguientes tablas.

Tareas de ingeniería de la segunda iteración:

Tarea de ingeniería	
No. de tarea: 3.1	**No. de HU:** 3
Nombre de tarea: Generar evento de interacción para trasladar el objeto 3D.	
Tipo de tarea: implementación	Puntos estimados: 1,0
Fecha inicio: 27-03-2015	Fecha fin: 03-04-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: se hace necesario permitir el movimiento del objeto seleccionado por la escena para localizar y visualizar regiones de interés a partir de los datos que envía el servidor.	

Tabla 29: Tarea de ingeniería 3.1.

Tarea de ingeniería	
No. de tarea: 3.2	**No. de HU:** 3
Nombre de tarea: Visualizar el objeto en 3D trasladado.	

Tipo de tarea: implementación	Puntos estimados: 0,6
Fecha inicio: 03-04-2015	Fecha fin: 08-04-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: para probar principalmente que se puede trasladar un objeto en 3D se necesita visualizar esta transformación a partir de un prototipo de software.	

Tabla 30: Tarea de ingeniería 3.2.

Tarea de ingeniería	
No. de tarea: 4.1	**No. de HU:** 4
Nombre de tarea: Generar evento de interacción para fijar el puntero.	
Tipo de tarea: implementación	Puntos estimados: 0,5
Fecha inicio: 08-04-2015	Fecha fin: 10-04-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: se necesita que llegado el puntero a un sitio de interés, se pueda fijar para no perder exactitud en el punto señalado.	

Tabla 31: Tarea de ingeniería 4.1.

Tarea de ingeniería	
No. de tarea: 4.2	**No. de HU:** 4
Nombre de tarea: Inmovilizar el puntero en el punto que se encuentre.	
Tipo de tarea: implementación	Puntos estimados: 0,3
Fecha inicio: 13-04-2015	Fecha fin: 14-04-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: se necesita visualizar a partir de un prototipo de software que se puede mantener al puntero fijo en un punto.	

Tabla 32: Tarea de ingeniería 4.2.

Tarea de ingeniería	
No. de tarea: 6.1	**No. de HU:** 6

Nombre de tarea: Generar evento de interacción para seleccionar un objeto.	
Tipo de tarea: implementación	Puntos estimados: 0,7
Fecha inicio: 14-04-2015	Fecha fin: 17-04-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: con el objetivo de elegir un objeto en 3D de entre los que se encuentran representados en la escena. Se necesita la opción de seleccionar para poder manipular dicho objeto según la elección del usuario.	

Tabla 33: Tarea de ingeniería 6.1.

Tarea de ingeniería	
No. de tarea: 6.2	**No. de HU:** 6
Nombre de tarea: Aplicar las secuencias de tareas de interacción al objeto seleccionado.	
Tipo de tarea: implementación	Puntos estimados: 0,3
Fecha inicio: 20-04-2015	Fecha fin: 21-04-2015
Programador responsable: Yuriaski Leyva Clemente	
Descripción: para probar que se puede seleccionar un objeto en 3D y transformarlo en consecuencia a esto, se necesita visualizar estas transformaciones a partir de un prototipo de software.	

Tabla 34: Tarea de ingeniería 6.2.

3.5 Pruebas

Una vez concluida la fase de implementación comienza la realización de las pruebas al sistema, las cuales permiten a los desarrolladores y al cliente concluir si lo desarrollado tiene la calidad requerida o si cuenta con errores en su funcionamiento. Existen varios tipos de pruebas de software, a continuación se tratan las pruebas realizadas a la solución.

3.5.1 Pruebas de aceptación

Estas pruebas son desarrolladas por el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, buscan una cobertura de la especificación de requisitos y del manual de usuario, por lo tanto permiten al cliente evaluar al sistema desarrollado.

Iteración 1: según la planificación de las iteraciones realizadas en el Capítulo 2 epígrafe 2.4.2, en la primera iteración se deben probar las HU que tienen una prioridad alta para el negocio. En esta iteración se diseñan y

aplican los casos de prueba para las siguientes HU:

HU 1: Rotar

HU 2: Acercar/Alejar

HU 5: Mover puntero

<table>
<tr><th colspan="4">Caso de prueba de aceptación</th></tr>
<tr><td colspan="2">Número: 1</td><td colspan="2">HU: Rotar.</td></tr>
<tr><td colspan="4">Probador encargado: Yuriaski Leyva Clemente</td></tr>
<tr><td colspan="4">Descripción: se comprueba la funcionalidad de rotar los objetos en la escena a partir del prototipo confeccionado para este fin.</td></tr>
<tr><td colspan="4">Condición de ejecución:
1. Debe existir una conexión con el servidor de VRPN.</td></tr>
<tr><td colspan="4">Entrada/Pasos de ejecución:
1. Seleccionar el objeto que se desea rotar.
2. Presionar el botón A del NW
3. Girar el NW hacia donde se desee rotar el objeto.</td></tr>
<tr><td colspan="2">Casos válidos</td><td>Resultado esperado</td><td>Resultado de prueba</td></tr>
<tr><td colspan="2">Seleccionar uno de los objetos representados en la escena para rotarlo, es necesario haber presionado con anterioridad el botón A del NW.</td><td>El objeto rota.</td><td>Satisfactorio</td></tr>
<tr><td>No.</td><td>Casos no válidos</td><td>Resultado esperado</td><td>Resultado de prueba</td></tr>
<tr><td>1</td><td>Presionar el botón B.</td><td>El objeto se va a trasladar.</td><td rowspan="4">Satisfactorio</td></tr>
<tr><td>2</td><td>Presionar el botón más (+).</td><td>La escena se va a acercar.</td></tr>
<tr><td>3</td><td>Presionar el botón menos (-).</td><td>La escena se va a alejar.</td></tr>
<tr><td>4</td><td>Presionar el botón A en estado de puntero.</td><td>El puntero se va a fijar en un punto.</td></tr>
</table>

Evaluación de la prueba: Satisfactorio

Tabla 35: Prueba a la historia de usuario 1.

Caso de prueba de aceptación			
Número: 2	HU: Acercar/Alejar.		
Probador encargado: Yuriaski Leyva Clemente			
Descripción: se comprueba la funcionalidad de acercar o alejar la escena a partir del prototipo confeccionado para este fin.			
Condición de ejecución: 2. Debe existir una conexión con el servidor de VRPN.			
Entrada/Pasos de ejecución: 1. Presionar el botón más (+) del NW para acercar. 2. Presionar el botón menos (-) del NW para alejar.			
No.	**Casos válidos**	**Resultado esperado**	**Resultado de prueba**
1	Presionar el botón más (+)	Se acerca toda la escena.	Satisfactorio
2	Presionar el botón menos (-)	Se aleja toda la escena.	
Casos no válidos		**Resultado esperado**	**Resultado de prueba**
Presionar cualquier otro botón del NW.		Las transformaciones que se visualizan no son de *zoom*.	Satisfactorio
Evaluación de la prueba: Satisfactorio			

Tabla 36: Prueba a la historia de usuario 2.

Caso de prueba de aceptación	
Número: 3	HU: Mover puntero.
Probador encargado: Yuriaski Leyva Clemente	
Descripción: se comprueba la funcionalidad de mover el puntero a partir del prototipo confeccionado para este fin.	
Condición de ejecución:	

3. Debe existir una conexión con el servidor de VRPN.			
Entrada/Pasos de ejecución: 1. Presionar el botón B del NW desde el estado de puntero para mover el puntero por la escena.			
No.	**Casos válidos**	**Resultado esperado**	**Resultado de prueba**
1	Presionar el botón B	Se mueve el puntero.	Satisfactorio
No.	**Casos no válidos**	**Resultado esperado**	**Resultado de prueba**
1	Presionar el botón A	Se fija el puntero en la posición que se encuentre	Satisfactorio
2	Presionar cualquier otro botón del NW.	Las transformaciones que se visualizan no son de movimiento del puntero.	
Evaluación de la prueba: Satisfactorio			

Tabla 37: Prueba a la historia de usuario 5.

Iteración 2: en esta iteración se diseñan y aplican los casos de prueba para las siguientes historias de usuario:

HU 3: Trasladar

HU 4: Fijar puntero

HU 6: Seleccionar

Caso de prueba de aceptación	
Número: 4	HU: Trasladar.
Probador encargado: Yuriaski Leyva Clemente	
Descripción: se comprueba la funcionalidad trasladar a partir del prototipo confeccionado para este fin.	
Condición de ejecución: 4. Debe existir una conexión con el servidor de VRPN.	
Entrada/Pasos de ejecución: 1. Seleccionar el estado de manipulación de cualquier objeto de la escena excluyendo al puntero. 2. Presionar el botón B del NW.	

<table>
<tr><td colspan="4">3. Rotar al NW en dependencia a donde se quiera trasladar el objeto.</td></tr>
<tr><td>Casos válidos</td><td colspan="2">Resultado esperado</td><td>Resultado de prueba</td></tr>
<tr><td>Desde el estado de manipulación de cualquier objeto representado en la escena, girar al NW. Es necesario que anteriormente se haya presionado el botón B de este para indicar que se van a realizar la transformación de traslación.</td><td colspan="2">El objeto se traslada por la escena en 3D.</td><td>Satisfactorio</td></tr>
<tr><td>Casos no válidos</td><td>No.</td><td>Resultado esperado</td><td>Resultado de prueba</td></tr>
<tr><td rowspan="3">Desde el estado de manipulación si se presiona otro botón que no sea el B, entonces no se visualizarán las trasformaciones que se desean o no se visualizará cambio alguno.</td><td>1</td><td>Si presiona el botón A, el objeto rota.</td><td rowspan="3">Satisfactorio</td></tr>
<tr><td>2</td><td>Si presiona el botón (+) o (-) se le hace un acercamiento o alejamiento a la escena respectivamente</td></tr>
<tr><td>3</td><td>En cualquier otro caso no se visualizan cambios.</td></tr>
<tr><td colspan="4">Evaluación de la prueba: Satisfactorio</td></tr>
</table>

Tabla 38: Prueba a la historia de usuario 3.

Caso de prueba de aceptación	
Número: 5	HU: Fijar puntero.
Probador encargado: Yuriaski Leyva Clemente	
Descripción: se comprueba la funcionalidad de fijar el puntero a partir del prototipo confeccionado para este fin.	
Condición de ejecución: 5. Debe existir una conexión con el servidor de VRPN.	
Entrada/Pasos de ejecución:	

<table>
<tr><td colspan="4">1. Seleccionar el estado de puntero.
2. Presionar el botón A del NW.
3. Para desbloquear al puntero se presiona el botón A de nuevo.</td></tr>
<tr><td>Casos válidos</td><td colspan="2">Resultado esperado</td><td>Resultado de prueba</td></tr>
<tr><td>Desde el estado de puntero se presiona el botón B del NW.</td><td colspan="2">El puntero se fija en el punto en que se encuentra.</td><td>Satisfactorio</td></tr>
<tr><td>Casos no válidos</td><td>No.</td><td>Resultado esperado</td><td>Resultado de prueba</td></tr>
<tr><td rowspan="3">Desde el estado de puntero si se presiona otro botón que no sea el A, entonces no se visualizarán las trasformaciones que se desean o no se visualizará cambio alguno.</td><td>1</td><td>Si presiona el botón B, el puntero se mueve por la escena.</td><td rowspan="3">Satisfactorio</td></tr>
<tr><td>2</td><td>Si presiona el botón (+) o (-) se le hace un acercamiento o alejamiento a la escena respectivamente</td></tr>
<tr><td>3</td><td>En cualquier otro caso no se visualizan cambios.</td></tr>
<tr><td colspan="4">Evaluación de la prueba: Satisfactorio</td></tr>
</table>

Tabla 39: Prueba a la historia de usuario 4.

<table>
<tr><th colspan="2">Caso de prueba de aceptación</th></tr>
<tr><td>Número: 6</td><td>HU: Seleccionar.</td></tr>
<tr><td colspan="2">Probador encargado: Yuriaski Leyva Clemente</td></tr>
<tr><td colspan="2">Descripción: se comprueba la funcionalidad de seleccionar el un objeto a partir del prototipo confeccionado para este fin.</td></tr>
<tr><td colspan="2">Condición de ejecución:
6. Debe existir una conexión con el servidor de VRPN.</td></tr>
<tr><td colspan="2">Entrada/Pasos de ejecución:</td></tr>
</table>

<table>
<tr><td colspan="4">1. Desde el estado de puntero se selecciona el objeto 1 al presionar el botón Home del NW.
2. Desde el estado de manipulación del objeto 1 se selecciona el objeto 2 al presionar el botón 1 del NW.
3. Desde el estado de manipulación del objeto 1 se selecciona el objeto 3 al presionar el botón 2 del NW.
4. Desde el estado de manipulación de cualquier objeto se puede llegar al estado de puntero al presionar el botón Home del NW.</td></tr>
<tr><td>Casos válidos</td><td colspan="2">Resultado esperado</td><td>Resultado de prueba</td></tr>
<tr><td>Se visualiza en dependencia del objeto seleccionado las TI que busca el usuario al realizar determinada TcI.</td><td colspan="2">El puntero se fija en el punto en que se encuentra.</td><td>Satisfactorio</td></tr>
<tr><td>Casos no válidos</td><td>No.</td><td>Resultado esperado</td><td>Resultado de prueba</td></tr>
<tr><td rowspan="3">Desde el estado de puntero si se presiona otro botón que no sea el A, entonces no se visualizarán las trasformaciones que se desean o no se visualizará cambio alguno.</td><td>1</td><td>Si presiona el botón B, el puntero se mueve por la escena.</td><td rowspan="3">Satisfactorio</td></tr>
<tr><td>2</td><td>Si presiona el botón (+) o (-) se le hace un acercamiento o alejamiento a la escena respectivamente</td></tr>
<tr><td>3</td><td>En cualquier otro caso no se visualizan cambios.</td></tr>
<tr><td colspan="4">Evaluación de la prueba: Satisfactorio</td></tr>
</table>

Tabla 40: Prueba a la historia de usuario 6.

Conclusiones parciales

Se pudo comprobar que es recomendable usar los estándares de codificación utilizados en la IGSTK o la VTK cuando se desarrolla sobre ellas. Esto brinda el beneficio de prevenir que en un futuro se requieran cambios en el código si se necesita integrar la solución encontrada con alguna de estas bibliotecas. En la fase de implementación se pudo constatar que el uso de máquinas de estados provee un mecanismo que permite la reutilización de los eventos que producen los DFI. La ejecución de las pruebas a la solución desarrollada demostraron la factibilidad del diseño arquitectónico seleccionado dado que todos los resultados de los casos

de prueba fueron satisfactorios.

Conclusiones

Con el desarrollo de esta investigación:

- Se desarrolló una estrategia de interacción mediante la cual se controla la representación virtual en 3D de los objetos reales mediante el NW como DFI.
- La solución propuesta para la interacción en el neuronavegador basada en TI y TcI permitió establecer un grafo de navegación entre las diferentes TI y su posterior implementación a partir de una máquina de estados. Esto benefició a la aplicación con un número alto de acciones a partir de la reutilización de los eventos del DFI (NW) en función del estado de interacción activo en la aplicación.
- La selección de la arquitectura Cliente-Servidor a partir de VRPN permitió una solución desacoplada que permite dividir la carga de procesamiento entre dos estaciones de trabajo. Al mismo tiempo se elimina la necesidad de manejar los DFI dependiendo de la interfaz que brindan sus fabricantes, con un aumento en la flexibilidad de la solución, al contar con la propiedad de poder utilizar diferentes dispositivos siempre y cuando generen el mismo tipo de eventos.
- La ejecución de las pruebas de aceptación demostró la factibilidad del diseño arquitectónico seleccionado dado que todos los resultados de los casos de prueba fueron satisfactorios.

Recomendaciones

Se recomienda que en próximos trabajos se investigue y se valore la posibilidad de añadir los aspectos que a continuación se listan:

- Una nueva TI que permita al neurocirujano realizar cortes a la representación en 3D de la anatomía intracraneal del paciente y proporcionarle una mayor retroalimentación de posibles regiones dañadas en el cerebro.
- Realizar la interacción a partir del reconocimiento de gestos corporales con dispositivos de seguimiento óptico.

Bibliografía

[1] "Diferencias entre un neurocirujano y un neurólogo," *eHow en Español*. [Online]. Available: http://www.ehowenespanol.com/diferencias-neurocirujano-neurologo-hechos_443648/. [Accessed: 03-Jun-2015].

[2] "Recuperación pronta con uso de neuronavegadores :: La Razón :: 9 de junio de 2015." [Online]. Available: http://razon.com.mx/spip.php?article250381. [Accessed: 10-Jun-2015].

[3] B. Valadez, "Hospitales del Edomex usan neuronavegadores," *Milenio*. [Online]. Available: http://www.milenio.com/cultura/Hospitales-Edomex-usan-neuronavegadores_0_473952604.html. [Accessed: 10-Jun-2015].

[4] "Nuestra Señora del Rosario | Hospitales Católicos de Madrid." [Online]. Available: http://www.hospitalescatolicos.es/hospitales-nuestra-senora-del-rosario/. [Accessed: 10-Jun-2015].

[5] R. R. M. Y. A. Más, "Revista Receta Médica… y Algo Más: NEURONAVEGADOR," *Revista Receta Médica… y Algo Más*, domingo, abril de 2012. [Online]. Available: http://revistarecetamedica.blogspot.com/2012/04/neuronavegador.html. [Accessed: 10-Jun-2015].

[6] C. Graetzel, T. Fong, S. Grange, and C. Baur, "A non-contact mouse for surgeon-computer interaction," *Technol. Health Care*, vol. 12, no. LSRO2-ARTICLE-2004–001, 2004.

[7] "http://pilicita.jimdo.com. [Online] [Cited: 2 19, 2012.] http://pilicita.jimdo.com/realidad-virtual."

[8] "Virtual Environment of Real Sport Hall and Analyzing Rendering Quality." [Online]. Available: https://www.academia.edu/11344304/Virtual_Environment_of_Real_Sport_Hall_and_Analyzing_Rendering_Quality. [Accessed: 24-Mar-2015].

[9] "Diccionario de la lengua española." [Online]. Available: http://lema.rae.es/drae/?val=interfaz. [Accessed: 02-May-2015].

[10] M. Kurosu, *Human-Computer Interaction: Human-Centred Design Approaches, Methods, Tools and Environments: 15th International Conference, HCI International 2013, Las Vegas, NV, USA, July 21-26, 2013, Proceedings*. Springer, 2013.

[11] Kumar, *Human Computer Interaction*. Firewall Media, 2005.

[12] E. Manchón, "¿Qué es la usabilidad? Definición de Usabilidad," *Alzado.org*. [Online]. Available: http://www.alzado.org/articulo.php?id_art=39. [Accessed: 02-Jun-2015].

[13] A. Sutcliffe, *Multimedia and Virtual Reality: Designing Multisensory User Interfaces*. Psychology Press, 2003.

[14] J. D. Foley, *Computer Graphics: Principles and Practice*. Addison-Wesley Professional, 1996.

[15] K. P. Herndon, A. van Dam, and M. Gleicher, "K.P. Herndon, A. van Dam y M. Gleicher. The Challenges of 3D Interaction: a CHI '94 workshop. ACM SIGCHI Bulletin, Vol. 26, No. 4, octubre 1994, pp. 36-46.," 1994.

[16] D. A. Bowman and L. . Hodges, "D.A. Bowman y L.F. Hodges. Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments. Journal of Visual Languages and Computing, Vol. 10, No. 1, 1999, pp. 37-53.," 1999.

[17] D. Boyd and L. Sastry, "D. Boyd y L. Sastry. Development of the INQUISITIVEInteraction Toolkit – Concept and Realisation. UserCentered Design and Implementation of VirtualEnvironments (UCDIVE) Workshop, York, Reino Unido, septiembre de 1999," 1999.

[18] J. Barrilleaux, "J. Barrilleaux. 3D User Interfaces with Java3D. ManningPublications, 2001.," 2001.

[19] "What is dashboard? - Definition from WhatIs.com," *SearchCIO*. [Online]. Available: http://searchcio.techtarget.com/definition/dashboard. [Accessed: 23-Jun-2015].

[20] J. Nielsen, "J. Nielsen. 3D is Better than 2D. Jakob Nielsen's Alertboxfor November 15, 1998. URL: http://www.useit.com/alertbox/981115.html," 1998.

[21] *A 45 años de la primera aparición pública del mouse*. .

[22] "Definición de Teclado," *Definición ABC*. [Online]. Available: http://www.definicionabc.com/tecnologia/teclado.php. [Accessed: 12-Jun-2015].

[23] "Joystick Definition." [Online]. Available: http://www.informaticamoderna.com/Joystick.htm. [Accessed: 06-Dec-2015].

[24] "Space Ball Definition." [Online]. Available: http://www.nekochan.net/wiki/Space_Ball. [Accessed: 06-Dec-2015].

[25] "Data Glove Definition." [Online]. Available: http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/ncp/. [Accessed: 05-Dec-2015].

[26] "Dispositivos de seguimiento." [Online]. Available: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/Realidad%20Virtual/web/dispositivos/trackers.html. [Accessed: 06-Dec-2015].

[27] "Microsoft Kinect." [Online]. Available: https://www.microsoft.com/en-us/kinectforwindows/. [Accessed: 05-Feb-2015].

[28] "Aurora," *Medical*. [Online]. Available: http://www.ndigital.com/medical/products/aurora/. [Accessed: 12-Jun-2015].

[29] "Polhemus Liberty." [Online]. Available: http://polhemus.com/motion-tracking/all-trackers/liberty. [Accessed: 05-Dec-2015].

[30] "Reconocimiento del Habla." [Online]. Available: http://cvc.cervantes.es/obref/congresos/sevilla/tecnologias/mesaredon_casacuberta.htm. [Accessed: 05-Dec-2015].

[31] D. A. Bowman, E. Krujiff, J. . J. LaViola, and I. Poupyrev, "D.A. Bowman, E. Kruijff, J.J. LaViola Jr. e I. Poupyrev. AnIntroduction to 3-D User Interface Design. Presence, Vol.10, No. 1, febrero 2001, pp. 96-108.," 2001.

[32] J. L. Gabbard, D. Hix, and J. E. Swan II, "J.L. Gabbard, D. Hix y J.E. Swan II. User-Centered Designand Evaluation of Virtual Environments. IEEE ComputerGraphics and Applications, Vol. 19, No. 6, noviembre 1999,pp. 51-59.," 1999.

[33] D. A. Bowman, J. L. Gabbard, and D. Hix, "D.A. Bowman, J.L. Gabbard y D. Hix. A Survey ofUsability Evaluation in Virtual Environments: Classificationand Comparison of Methods. Presence: Teleoperators and Virtual Environments, Vol. 11, No. 4, 2002, pp. 435-455.," 2002.

[34] G. G. Robertson, J. D. Mackinlay, and S. K. Card, "Cone trees: animated 3D visualizations of hierarchical information," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1991, pp. 189–194.

[35] R. Wiegers, "Interaction with 3D VTK widgets A quantitative study on the influence of 2DOF and 6DOF input devices on user performance," 2006.

[36] A. J. Kok and R. Van Liere, "A multimodal virtual reality interface for 3D interaction with VTK," *Knowl. Inf. Syst.*, vol. 13, no. 2, pp. 197–219, 2007.

[37] P. Vilar, "Designing the User Interface: Strategies for Effective Human-Computer Interaction," *J. Am. Soc. Inf. Sci. Technol.*, vol. 61, no. 5, pp. 1073–1074, 2010.

[38] I. L. Cleary K and Cleary K, Ibanezb L, Ranjana S, Blake B, "IGSTK: a software toolkit for image-guided surgery applications," in *[http://www.cars-int.org CARS] 2004 – Computer Assisted Radiology and Surgery. Proceedings of the 18th International Congress and Exhibition*, vol. 1268, 2004, pp. 473–9.

[39] A. Enquobahrie and Z. Yaniv, *IGSTK: The Book*. 2009.

[40] L. S. Avila, *The VTK User's Guide*, 11th ed. Kitware, 2010.

[41] B. Lorensen, K. Martin, and W. Schroeder, *The Visualization Toolkit*, 4th ed. Kitware, 2006.

[42] C. Shaw, M. Green, J. Liang, and Y. Sun, "Decoupled simulation in virtual reality with the MR toolkit," *ACM Trans. Inf. Syst. TOIS*, vol. 11, no. 3, pp. 287–317, 1993.

[43] E. de la Cruz Guevara Ramírez, "Mecanismo de acceso a la información de dispositivos de interacción 3D desde un entorno virtual," Universidad de Ciencias Informáticas(UCI), 2014.

[44] R. M. Taylor II, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser, "VRPN: a device-independent, network-transparent VR peripheral system," in *Proceedings of the ACM symposium on Virtual reality software and technology*, 2001, pp. 55–61.

[45] J. Tokuda, G. S. Fischer, X. Papademetris, Z. Yaniv, L. Ibanez, P. Cheng, H. Liu, J. Blevins, J. Arata, and A. J. Golby, "OpenIGTLink: an open network protocol for image-guided therapy environment," *Int. J. Med. Robot.*, vol. 5, no. 4, pp. 423–434, 2009.

[46] "VRPN." [Online]. Available: http://www.cs.unc.edu/Research/vrpn/.

[47] M. Piatinni, "PIATTINI, M. Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión. 1996.," 1996.

[48] J. Canós, P. Letelier, and M. C. Penadés, "Metodologías Ágiles en el desarrollo de Software," *Univ. Politécnica de Valencia, Valencia*, 2003.

[49] R. H. Uñoja, "Ingenieria de Software: METODOLOGIAS DE DESARROLLO DE SOFTWARE TRADICIONALES VS AGILES," *Ingenieria de Software*, abril de 2012. .

[50] D. Wells, "Extreme Programming: A gentle introduction," *Extreme Program. November2004 Dispon. Sur Httpwww Extrem. Org*, 2001.

[51] P. por pmoinformatica.com, "Los 5 valores de la programación extrema (XP) - La Oficina

de Proyectos de Informática." .
[52] B. Stroustrup, "The C Programming Language," in *1*, Third., 1997.
[53] "Qt | Cross-platform application & UI development framework," *Qt*. [Online]. Available: http://www.qt.io/. [Accessed: 03-May-2015].
[54] "CMake." [Online]. Available: http://www.cmake.org/. [Accessed: 12-Jun-2015].
[55] "CMake Tutorial | La plaga Tux." [Online]. Available: http://plagatux.es/2009/12/tutorial-cmake/. [Accessed: 29-Apr-2015].
[56] "Tech Talk: Linus Torvalds on git (at 00:01:30)." [Online]. Available: https://www.youtube.com/watch?v=4XpnKHJAok8&t=1m30s.
[57] "Git web page." [Online]. Available: http://git-scm.com/.
[58] "Visual Paradigm Web Page." [Online]. Available: www.visual-paradigm.com.
[59] "Tipos De Realidad Virtual," 14:26:46 UTC.
[60] P. Letelier and M. C. Penadés, "Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)," 2012.
[61] D. Fontanarossa, J. Garderes, L. Batalla, and A. Gatto, "Gestión de Software.Extreme Programming (XP).," 2006.
[62] J. Joskowicz, "Reglas y prácticas en eXtreme Programming," *Univ. Vigo*, p. 22, 2008.
[63] "DISEÑO ARQUITECTONICO.pdf," *Google Docs*. [Online]. Available: https://docs.google.com/document/d/1C7cPJSR2mjAAuTYRfSR4N1Xb5BzS4czb0llPirSAYrU/edit?usp=embed_facebook. [Accessed: 11-Jun-2015].
[64] Universidad Cooperativa de Colombia, "Patrones de diseño y frameworks." [Online]. Available: http://ingenieriasw2.blogspot.com/p/patrones-de-diseno-y-frameworks.html.
[65] E. Freeman, E. Robson, B. Bates, and K. Sierra, *Head first design patterns*. O'Reilly Media, Inc., 2004.
[66] "Harlander.com - Gebrauchte Notebooks & gebrauchte Computer günstig kaufen!," *Harlander.com*. [Online]. Available: http://www.harlander.com/Artikel/8564/179/Maeuse-3Dconnexion_SpaceBall_5000_FLX_USB_20_3D_Maus.htm. [Accessed: 23-Jun-2015].
[67] S. Grade, "Equipment," *UCL - Equipment*, 18-Jul-2012. [Online]. Available: https://www.uclouvain.be/en-415776.html. [Accessed: 23-Jun-2015].
[68] "Microsoft's Kinect: The All Seeing Eye On Top Of The Television," *Thomas Dishaw.com*. [Online]. Available: http://thomasdishaw.com/microsofts-kinect-the-all-seeing-eye-on-top-of-the-television/. [Accessed: 23-Jun-2015].
[69] A. sickr, "Wii Remote Rapid Charging Cradle And Rechargeable Battery Available On NA Nintendo Store," *My Nintendo News*. .

Glosario de términos

Socket: designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.

El término *socket* es también usado como el nombre de un API para la familia de protocolos TCP/IP, provista usualmente por el sistema operativo.

Los *sockets* de Internet constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados. Un *socket* queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

Macro: es una abreviatura de macroinstrucción. Además se define como una serie de instrucciones que se almacenan para que se puedan ejecutar de manera secuencial mediante una sola llamada u orden de ejecución. Dicho de otra manera, una macroinstrucción es una instrucción compleja, formada por otras instrucciones más sencillas. Esto permite la automatización de tareas repetitivas.

Zoom: es el efecto de acercamiento o alejamiento de la imagen obtenido mediante este objetivo.

Widgets: son componentes de la interfaz de usuario reutilizables tales como un botón, una barra de desplazamiento, un área de control, o área de edición de texto, que puede recibir entradas desde el teclado o el ratón y se puede comunicar con una aplicación o con otro *widget*.

3D: se refiere a tridimensional; en física, geometría y análisis matemático, un objeto o ente es tridimensional si tiene tres dimensiones. Es decir cada uno de sus puntos puede ser localizado especificando tres números dentro de un cierto rango, por ejemplo, anchura, longitud y profundidad. El espacio alrededor de las personas es tridimensional a simple vista. Hoy en día es posible la simulación por computadoras del espacio tridimensional mediante cálculos basados en la proyección de entornos tridimensionales sobre pantallas bidimensionales, como monitores o televisores.

Usabilidad: característica de los software usables. La Organización Internacional para la Estandarización (ISO por su sigla en inglés) dispone de dos definiciones de usabilidad. La primera apunta que la usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso. La segunda expone que la usabilidad es la efectividad, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico. Esta es una definición centrada en el concepto de calidad en el uso, es decir, se refiere a cómo el usuario realiza tareas específicas en escenarios específicos con efectividad.