UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

CENTRO DE COMPUTACIÓN GRÁFICA

Interacción con una Pantalla Horizontal utilizando Wiimote

Trabajo Especial de Grado presentado ante la ilustre
Universidad Central de Venezuela por el bachiller
Marcos Javier Ramírez Rodríguez
para optar al título de Licenciado en Computación

Tutor:
Prof. Rhadamés Carmona

Caracas, Mayo 2014

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
CENTRO DE COMPUTACIÓN GRÁFICA

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designados por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado, presentado por el Bachiller Marcos Ramírez, portador de la cédula de identidad 18.779.304, con el título "**Interacción con una Pantalla Horizontal utilizando Wiimote**", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue dicho trabajo por cada uno de los Miembros del Jurado, se fijó el día 23 de mayo de 2014, a las 5:00 PM, para que su autor lo defienda en forma pública, en el Centro de Computación Gráfica, mediante la exposición oral de su contenido, y luego de la cual respondieron satisfactoriamente a las preguntas que le fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente Acta, en Caracas a los 23 días del mes de mayo del año dos mil catorce, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor Rhadamés Carmona.

_____
Prof. Rhadamés Carmona
(Tutor)

_____
Prof. Esmitt Ramírez
(Jurado Principal)

_____
Prof. Robinson Rivas
(Jurado Principal)

AGRADECIMIENTOS

Les agradezco a mis padres Marcos e Isabel, por su apoyo incondicional e interés constante en mí y por asegurarse de que no me faltara nada durante mi desarrollo académico.

Les agradezco a mis abuelos Froilán e Isabel y tíos Ángel y Luis por todo el interés sobre mi educación. En especial a mi tío Luis por estar pendiente de mi tesis y motivarme. Abuelo tú te habías prometido que tus hijos y tus nietos tendrían la educación que no tuviste y aquí estoy culminando esta etapa de mi vida, lo sigues logrando. A mi tío Ángel gracias por el apoyo que me brindaste mientras estaba en caracas residenciado.

A mi tutor Rhadamés Carmona, por la paciencia que ha tenido en el desarrollo de la tesis.

RESUMEN

TÍTULO

Interacción con una Pantalla Horizontal utilizando Wiimote.

AUTOR

Marcos Ramírez

TUTOR

Prof. Rhadamés Carmona

El Centro de Computación Gráfica de la UCV cuenta con una mesa de proyección horizontal o Workbench, en donde se proyectan imágenes estereoscópicas. Actualmente, la interacción con la mesa de trabajo se logra con dispositivos alámbricos, los cuales son costosos, pueden resultar incómodos, y limitan los movimientos del usuario. En este trabajo se propone utilizar los controles de la consola de video juegos Wii, los cuales se pueden comunicar con el computador vía bluetooth. El rastreo de la posición de la cabeza del usuario se logra colocando unos LEDs infrarrojos en los lentes estereoscópicos para triangular con el control del Wii y determinar su posición. Con la posición del usuario, se realiza el despliegue estereoscópico para lograr el efecto 3D deseado. También se logra la manipulación de los objetos de la escena con otro control del Wii. Estas funcionalidades son encapsuladas en una librería para su posterior uso en aplicaciones que requieran de la interacción con la pantalla de proyección.

PALABRAS CLAVES:

Rastreo de cabeza, interacción humano-computador, realidad virtual

Índice

Capítulo 1. Introducción

Una escena sintética está típicamente compuesta por un mundo virtual en el cual se realizan cambios ya sean propios del sistema o mediante la interacción del usuario a través de la entrada y salida típica basada en teclado y ratón. Este mundo virtual es proyectado y discretizado en imágenes 2D que son desplegadas en secuencia en la pantalla. Dicha pantalla suele ser un monitor o una pantalla de proyección. El realismo alcanzado en estas imágenes sintéticas es relativo si no consideramos el punto de vista real del usuario, puesto que por más efectos que se le agreguen seguiremos viendo una imagen producto de la proyección, que fue generada suponiendo un punto de vista estándar, es decir, el centro de proyección se dispone en forma perpendicular frente a la pantalla.

La estereoscopía, mejora el efecto de realismo en el despliegue, ya que esta técnica combina dos imágenes de un mismo instante de tiempo separadas por una pequeña distancia, haciendo que cada uno de nuestros ojos reciba una perspectiva distinta de la escena. El cerebro humano hace la mezcla de las dos imágenes que percibe cada ojo generando así la sensación de profundidad. Sin embargo, al usuario cambiar de posición respecto a la pantalla, la imagen generada es la misma, lo cual limita el aporte de realismo dado por la estereoscopía.

En el centro de Computación Gráfica de la Universidad Central de Venezuela, contamos con una mesa de trabajo o *workbench* (*CCGworkbench*) en donde las imágenes estereoscópicas son proyectadas sobre la superficie de la mesa. Algunas de sus características son las siguientes:

1) Se tiene una librería [1] que soporta diversos formatos estereoscópicos. Esta librería ha sido utilizada en varias aplicaciones, como por ejemplo RenderALL [2]. Sin embargo, esta librería no había sido pensada para su uso en el *CCGworkbench*, por lo que en la proyección se presenta el problema que cuando el usuario se desplaza frente a la mesa, las imágenes generadas son exactamente las mismas, obteniendo una percepción visual incorrecta del fenómeno a estudiar.
2) Para el sistema de rastreo se cuenta con un sistema alámbrico llamado IsoTrack II fabricado por Polhemus [3], el cual tiene buena precisión. Con este sistema se puede capturar la posición del usuario, y de elementos de interacción como un lápiz óptico. Sin embargo, el hecho de ser alámbrico y costoso limita su usabilidad y la posibilidad de reproducir el sistema.

En un intento para sustituir el dispositivo de rastreo, se implementó previamente una librería que captura la posición del usuario a través de una webcam [4]. Sin embargo, esta librería tiene la limitación de que el fondo tiene que ser controlado, y no puede haber más de un usuario a la vez.

En este trabajo especial de grado se propone realizar una librería (API) para el CCGworbench que lidie con el problema de rastreo de usuario (tanto de la mano para la selección de objetos como de la cabeza) así como de la corrección de la perspectiva en la visualización estereoscópica. Específicamente se propone el siguiente objetivo general y objetivos específicos.

1.1 Objetivo general

Desarrollar un sistema inalámbrico para rastreo o seguimiento de cabeza y de mano para la mesa de trabajo virtual del Centro de Computación Gráfica de la UCV, de bajo costo, que considere la estereoscopía y el punto de vista del observador.

1.2 Objetivos específicos

a. Implementar un sistema de Head Tracking (seguimiento de cabeza) utilizando un control de Wiimote [5]. Este control inalámbrico es el control primario de la consola de videojuegos Wii.
b. Incorporar otro control inalámbrico Wiimote como dispositivo de señalamiento y manipulación de la escena.
c. Generar la imagen estereoscópica desde la perspectiva del usuario.
d. Realizar un API[1] que pueda ser instanciado en distintas aplicaciones, que incorpore el rastreo de cabeza, un dispositivo de señalamiento y la estereoscopía. En la estereoscopía se utilizará anáglifos, y se incorpora el punto de vista del usuario en el proceso de visualización.
e. Desarrollar un sistema simple 3D que permita mostrar las bondades del API.
f. Generar un tutorial paso por paso de cómo utilizar el API para incorporarlo a futuras aplicaciones.

1.3 Justificación de la solución planteada

El Wiimote es un sistema económico de rastreo infrarrojo con alta precisión y resolución, que no necesita de cables. Solo necesita la comunicación bluetooth entre el computador y el control para hacer la conexión con el programa, dando la opción hasta de incluir nuevas funcionalidades mediante los controles que nos ofrece el Wiimote como el nunchuk [6]. El control posee una cámara infrarroja, con una resolución virtual de 1024*768 y un ángulo de 41 grados en el eje X y 31 grados en el eje Y con un precio de alrededor de 40$ por control, lo que lo hace una opción viable económicamente dada la capacidad que posee. Combinando la cámara infrarroja y los acelerómetros, tenemos 6 grados de libertad en el espacio, y gracias a la relatividad podemos tomar el sistema de referencia que nos convenga.

El control del Wiimote toma como referencia la barra de sensores, el cual contiene LEDs infrarrojos que son captados por la cámara del Wiimote y utilizados para determinar la posición

[1] Application Programming Interface o Interfaz de Programación de Aplicaciones en español: Es un grupo de rutinas (conformando una interfaz) que provee un sistema operativo, una aplicación o una biblioteca, que definen cómo invocar desde un programa un servicio que éstos prestan. En otras palabras, una API representa una interfaz de comunicación entre componentes de software.

del control respecto a la barra sensorial (y viceversa) mediante triangulación. Conociendo la posición del usuario, se puede realizar la corrección perspectiva, que consiste en utilizar dicha posición para el despliegue estereoscópico de la escena. Se utilizará la técnica de estereoscopía basada en anaglifos, debido a que es el formato utilizado actualmente en el workbench. Sin embargo, será fácil cambiar a otra técnica de estereoscopía utilizando la librería de formatos estereoscópico presentado por Roman Alonso [1].

Para seleccionar objetos se utilizará otro control de Wii, para así interactuar con la escena con tecnología inalámbrica y a bajo costo. A continuación se presentan los detalles de la arquitectura planteada para la solución, tanto en hardware como en software.

1.3 Arquitectura de software (infraestructura)

El sistema será desarrollado en entorno Windows 7 64 bits utilizando Visual C++ con el IDE de Visual Studio 2010, OpenGL como librería Gráfica, y la librería Wiiuse [7] para capturar la información de los controles de Wii en el computador. El software demostrativo contendrá un escenario 3D en donde el usuario puede percibir las diferentes perspectivas a medida que se mueve frente a la mesa de proyección. Igualmente permitirá seleccionar y mover los objetos de la escena.

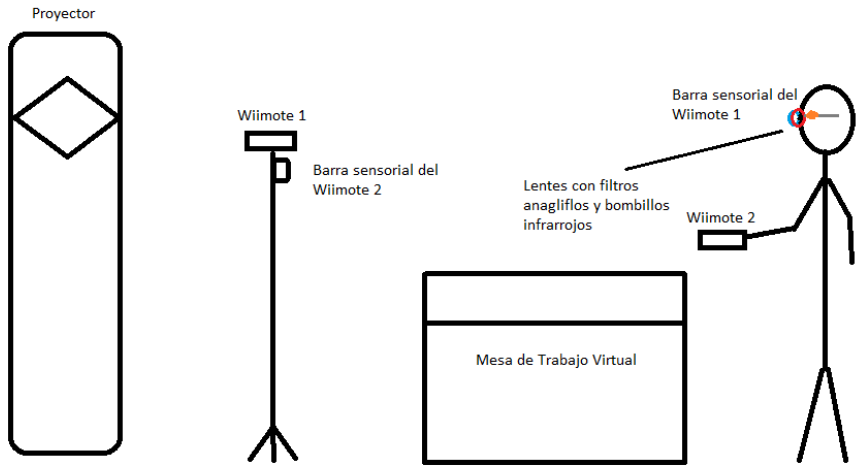


Figura 1.1 – Arquitectura y disposición del hardware.

1.4 Arquitectura de hardware

En la Figura 1.1 se muestra disposición de los elementos de hardware sobre la cual se basará creación del API. Se dispondrá de dos controles de Wii y unos lentes de anáglifos para percibir la imagen estereoscópica. Para el rastreo de la cabeza, se colocarán un par de LEDs infrarrojos extraídos de una barra sensorial (LEDs de la barra sensorial del Wiimote 1) en los lentes de

anáglifos, para que mediante un control de Wii (Wiimote 1 ubicado en una posición fija frente a la mesa) se pueda determinar la posición del usuario por triangulación. Para el dispositivo de señalamiento, se colocará una barra sensorial frente a la mesa (Barra sensorial del Wiimote 2), para rastrear la posición de otro control que el usuario manipula con su mano (Wiimote 2). Este segundo control se utilizará como dispositivo de señalamiento y manipulación de objetos sobre la escena 3D.

Este trabajo se presenta en capítulos. En el capítulo 2 se expone el marco teórico donde hay una reseña histórica, y se explica la realidad virtual y la realidad aumentada, y con ellos las formas de hacer estereoscopía y rastreo del usuario. El capítulo 3 se explica en detalle la corrección perspectiva y el funcionamiento de la mesa virtual que se encuentra en el CCG de la facultad de ciencias de la UCV. El capítulo 4 expone el diseño del hardware utilizado, principalmente cómo se adapta la barra sensorial del control del Wiimote a los lentes de realidad virtual. El capítulo 5 explica el desarrollo de la librería y su uso. El capítulo 6 presenta las pruebas y resultados sobre una escena 3D simple. Y finalmente el capítulo 7 ofrece las conclusiones y trabajos a futuros.

Capítulo 2. Marco teórico

En este capítulo se describen los conceptos básicos de la realidad virtual así como diversas investigaciones y tecnologías relacionadas con la misma, como la cueva o *cave*, mesa de realidad virtual o *workbench*, dispositivos de rastreo de posición, entre otros.

2.1 Reseña histórica

Morton Heilig [8] escribió en la década de 1950 una "Experiencia de Teatro" que podría incluir todos los sentidos de una manera efectiva integrando al espectador con la actividad en la pantalla. Construyó un prototipo de su visión apodado "Sensorama" en 1962, junto con cinco cortometrajes que permitían aumentar la experiencia del espectador a través de sus sentidos (Vista, Olfato, Tacto, y Oído).

En 1968, Ivan Sutherland [9], con la ayuda de su estudiante Bob Sproull, construyeron lo que sería ampliamente considerado el primer visor montado en la cabeza o *Head Mounted Display* (HMD) para Realidad Virtual y Realidad Aumentada. Era muy primitivo en términos de Interfaz de usuario y realismo, y el HMD utilizado por el usuario era tan grande y pesado que debía colgarse del techo, y los gráficos que hacían al ambiente virtual eran simples "modelos de alambres[2]". A finales de los 80's se popularizó el término Realidad Virtual por Jaron Lanier [10], cuya compañía fundada por él creó los primeros guantes y anteojos de Realidad Virtual.

El terminó Realidad Aumentada fue introducido por el investigador Tom Caudell [11] en Boeing, en 1992. Caudell fue contratado para encontrar una alternativa a los tediosos tableros de configuración de cables que utilizan los trabajadores. Salió con la idea de anteojos especiales y tableros virtuales sobre tableros reales genéricos; es así que se le ocurrió que estaba "aumentando" la realidad del usuario.

2.2 Estereoscopía

La estereoscopía, imagen estereográfica, o imagen 3D (tridimensional) es una técnica capaz de recoger información visual tridimensional y/o crear la ilusión de profundidad en una imagen. La ilusión de la profundidad en una fotografía, película, u otra imagen bidimensional se crea presentando una imagen ligeramente diferente para cada ojo, como ocurre en nuestra forma

[2] Los modelos geométricos es la representación de un objeto mediante la unión de varias figuras geométricas (usualmente triángulos) unidos por sus vértices, el resultado de todo esto es lo que se denomina como "modelos de alambres".

habitual de ver. Muchas pantallas 3D usan este método para transmitir imágenes. Fue inventado por Sir Charles Wheatstone[3] en 1840.

La fotografía tridimensional de la industria moderna puede usar escáneres 3D para detectar y guardar la información tridimensional. Actualmente podemos disfrutar de la estereoscopía en cine con el nuevo formato Digital 3D.

Un *formato* estereoscópico es un método usado para asignar píxeles, conjunto de píxeles, líneas o cuadros, a las respectivas imágenes izquierda y derecha, ya sea por cuadros secuenciales, entrelazado estéreo, etc.

El *sistema de selección* estereoscópico es el tratamiento del formato estéreo para poder separar cada imagen y así poder mostrar la imagen correspondiente a cada ojo. Tanto el formato como la selección estéreo deben trabajar juntos. Un formato estereoscópico puede determinar el sistema de selección a utilizar.

A continuación se describen algunos sistemas de formato-selección estereoscópico:

Anaglifos: Se utilizan filtros de colores complementarios, como rojo y azul, o rojo y verde, tanto en la construcción de la imagen estereoscópica (formato), como para separarlas (selección). La imagen presentada por ejemplo en rojo no es vista por el ojo que tiene un filtro del mismo color pero sí ve la otra imagen en azul o verde. Presenta el problema de la alteración de los colores, perdida de luminosidad y cansancio visual después de un uso prolongado.

Polarizadas: se genera la imagen utilizando luz polarizada mediante dos filtros (uno por cada imagen estéreo). Estas imágenes polarizadas se proyectan típicamente en una pantalla no polarizante, y luego son separadas de nuevo mediante filtros polarizados, de manera tal que el usuario reciba la imagen correspondiente por cada ojo. El sistema de polarización no altera colores, aunque hay una cierta pérdida de luminosidad. Se usa tanto en proyección de cine 3D como en monitores de computadores personales mediante pantallas de polarización alternativa.

Cuadros secuenciales + lentes de cristal líquido (LCD – *Liquid-Crystal Display*): En estos sistemas se presentan en secuencia y alternativamente las imágenes izquierda y derecha, en sincronía con unos lentes dotados con obturadores de cristal líquido de forma que cada ojo recibe la imagen correspondiente. A una frecuencia elevada, el parpadeo es imperceptible. Se utiliza en televisores 3D y cines 3D de última generación.

[3] Sir Charles Wheatstone (Gloucester, 6 de febrero de 1802 - París, 19 de octubre de 1875) fue un científico británico, inventor de mucho de los avances de la época victoriana, incluyendo el Estereoscopio (aparato que creaba la ilusión de ver imágenes tridimensionales), la técnica Playfair de codificación, y el caleidófono.

La estereoscopía más utilizada debido a su facilidad y bajo costo consiste en crear una ilusión 3D anaglífica (Figura 2.1) a partir de un par de imágenes 2D. La forma más sencilla de crear en el cerebro la percepción de profundidad es proporcionando a los ojos del espectador dos imágenes diferentes, que representan dos perspectivas del mismo objeto, con una pequeña desviación similar a las perspectivas que de forma natural reciben los ojos en la visión binocular.

Figura 2.1 – Imagen con formato para anaglifos [12].

2.3 Realidad virtual

La realidad virtual [13] se basa en el empleo de computadores y otros dispositivos, cuyo fin es producir una apariencia de realidad que permita al usuario tener la sensación de estar presente en ella. Algunos equipos se completan con trajes y guantes equipados con sensores diseñados para generar estímulos, que intensifican la sensación de realidad. Su aplicación, aunque centrada inicialmente en el terreno de los videojuegos, se ha extendido a otros muchos campos, como la medicina o las simulaciones de vuelo.

La realidad virtual suele dividirse en inmersiva y no inmersiva. A continuación se describe cada tipo.

2.3.1 Realidad virtual inmersiva

Los métodos inmersivos de realidad virtual (Figura 2.2) con frecuencia se ligan a un ambiente tridimensional creado por computador donde el usuario es sumergido en dicho ambiente, el cual se manipula a través de cascos o pantallas de proyección, guantes u otros dispositivos que capturan la posición y rotación de diferentes partes del cuerpo humano. A continuación se describen brevemente diversos dispositivos utilizados para realidad virtual inmersiva.

Figura 2.2 – Ejemplo de realidad virtual inmersiva [14].

Cuevas *CAVE*

La tecnología *Cave Automatic Virtual Environment* o CAVE, es un entorno de realidad virtual inmersiva. Se trata de una sala en forma de cubo en la que hay proyectores orientados hacia las diferentes paredes, suelo y techo (Figura 2.3). Dependiendo del resultado que se quiera obtener se proyecta la imagen a todas o sólo alguna de las paredes de la sala. Fue desarrollado en la Universidad de Illinois en 1992 [15]. Es un entorno de realidad virtual inmersiva, es decir que se recrea un entorno virtual en el que el usuario puede moverse.

La habitación de una *cave* consta de al menos tres paredes y opcionalmente techo y suelo. Las paredes actúan como monitores gigantes que emiten de forma síncrona imágenes para cada ojo, de forma que cuando el usuario lo ve a través de los lentes estereoscópicos activos da sensación 3D. Los lentes están sincronizados con la frecuencia de los monitores o protectores y sólo dejan ver a través del ojo que corresponde cuando se emite cada imagen. El hecho de usar pantallas en vez de HMD (*head-mounted displays*) permite al usuario tener un campo de visión mucho más amplio y más ajustado a la realidad.

En los lentes del usuario se sitúan los dispositivos rastreadores (*trackers*) que informan al computador de como cambiar la proyección del escenario a medida que el usuario se va moviendo. Las imágenes generadas por el computador son proyectadas sobre pantallas de proyección trasera, en ocasiones utilizando un espejo intermedio, de manera tal que la imagen se perciba dentro del CAVE (Figura 2.4).

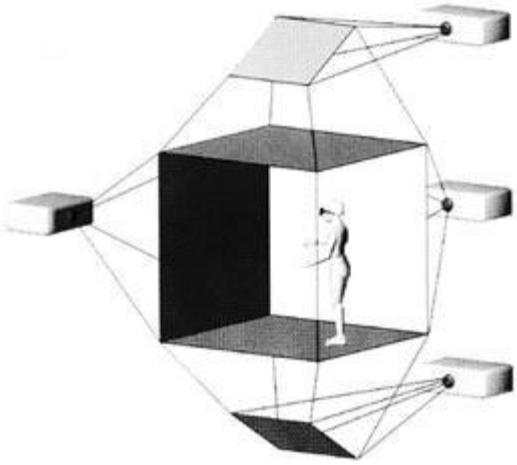


Figura 2.3 – CAVE y sus pantallas de proyección [16].

Es habitual también el uso de algún tipo de guante para manipular los objetos del mundo virtual. Es posible la presencia de más de un usuario en la CAVE, aunque solo uno de ellos afectará con su movimiento al cambio del punto de vista en el mundo virtual, mientras los demás actúan como observadores.

Una de las limitaciones de este sistema, además del precio, es su limitado espacio. Esto se ha solucionado mediante el *redirected-walking* [17]. La idea es engañar al usuario con las imágenes de manera tal que el usuario puede creer que está caminando en una línea recta infinita, cuando en realidad está caminando en círculos.

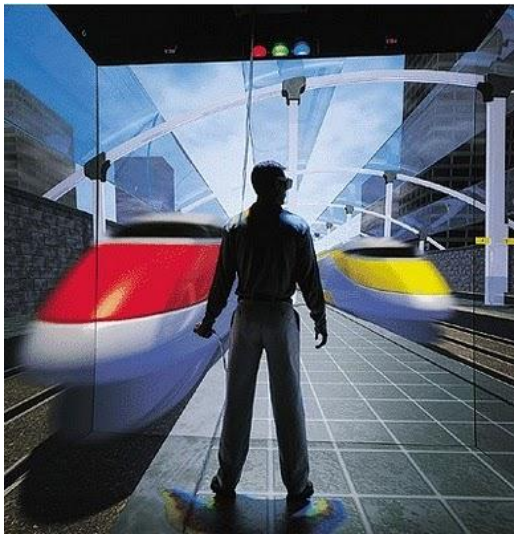


Figura 2.4 – Imagen percibida dentro del CAVE [16].

Guantes

Desde los inicios de la realidad virtual, los guantes han jugado un papel muy importante, ya que permiten al usuario interactuar con el mundo virtual que están observando. Existen distintos tipos de marcas de guantes (DataGlove, PowerGlove, CyberGlove, etc), aunque todos ellos están orientado a este mismo objetivo: permitir al usuario manipular objetos del mundo virtual.

Figura 2.5 – DataGlove [16].

El primer guante comercializado, y uno de los más usados es el DataGlove (Figura 2.5), creado por Zimmerman (VPL Research [18]) para usar conjuntamente con el HMD diseñado por Jaron Lanier [10]. Utiliza la fibra óptica para medir la flexión de los dedos: se emite un haz de luz en un extremo, y un sensor (diodo fotoeléctrico) detectará la intensidad en el otro extremo, que será diferente en función de la posición y flexión de los dedos. Estos guantes deben de ser calibrados para cada usuario antes de su uso para conseguir un funcionamiento correcto.

Figura 2.6 – CyberGlove (Immersion Co.) [19].

El CyberGlove [19] (Figura 2.6) emplea una tecnología diferente para conocer la flexión de los dedos, ya que en lugar de utilizar fibra óptica emplea pequeños sensores magnéticos (entre 18 y 22 por mano) que se localizan en la articulación de cada dedo y permiten medir la distancia entre ellos.

Lentes

Se pueden clasificar en activos y pasivos, según la forma de hacer la selección de las imágenes izquierda y derecha. Los pasivos no necesitan de un componente activo para hacer la selección de la imagen, mientras que los activos sí. A continuación se describen ambos tipos de lentes.

a) Lentes pasivos: Los anaglifos fueron durante décadas los lentes pasivos más populares. Los lentes anaglifos utilizan filtros de color (rojo–azul, rojo–verde, red-cyan o bien ámbar–azul), que permiten visualizar imágenes distintas en cada ojo, dando así un efecto de profundidad relativamente convincente. Hoy en día se utilizan lentes pasivos polarizados (Figura 2.7), principalmente en salas de cine 3D. Estos lentes filtran las ondas de luz provenientes desde diversos ángulos de la pantalla, permitiendo que cada ojo por separado reciba solo la imagen polarizada que le corresponde. Estos lentes fueron inmediatamente más populares que los anaglifos debido a que no utilizan filtros de color que pudiesen distorsionar el color original de la imagen.

Figura 2.7 – Lentes pasivos polarizados, marca Real 3D. Se consiguen actualmente en las películas 3D de cualquier cine.

b) Lentes activos: Utilizan tecnología de cristal líquido LCD (Liquid Crystal Display). Estos poseen sensores infrarrojos (IR) que permiten conectarse de manera inalámbrica con el televisor o monitor 3D (Figura 2.8). En este sistema, las dos imágenes no se muestran al mismo tiempo, sino una a la vez. Los lentes de cristal líquido se van alternando entre un modo "transparente" y un modo "opaco" al mismo tiempo que las imágenes se alternan en la pantalla, es decir, el ojo izquierdo se bloquea cuando la imagen del ojo derecho aparece en la televisión y viceversa. Esto ocurre tan rápido que nuestra mente no puede detectar el parpadeo de los lentes.

Figura 2.8 – Lentes activos 3D Samsung [20].

Dispositivos Hápticos

El término “Interfaz háptica” alude a aquellos dispositivos que permiten al usuario tocar, sentir o manipular objetos simulados en entornos virtuales y sistemas tele-operados (Figura 2.9). En la mayoría de simulaciones realizadas en entornos virtuales, basta con emplear pantallas 3D y dispositivos de sonido 3D estéreo para provocar en el usuario, mediante imágenes y sonidos, la sensación de inmersión dentro del espacio virtual. No obstante, además de provocar en el usuario esta sensación de inmersión, debemos proporcionarle la posibilidad de interactuar con el medio virtual, pudiendo establecer entre el usuario y el entorno virtual una transferencia bidireccional y en tiempo real de información mediante el empleo de interfaces de tipo háptico.

La importancia de las interfaces hápticas es determinante en la realización de tareas típicamente “hápticas”, o en las que se requiera un alto grado de entrenamiento, como pueden ser: administración de anestesia epidural, palpado de bultos cancerígenos, ensamblaje de conjuntos complejos antes de ser fabricados, etc. Ayudan a su vez, a incrementar la sensación de presencia o inmersión del usuario dentro de un entorno simulado, proporcionando restricciones naturales al movimiento de objetos.

Figura 2.9 – Phantom Sensable Technologies [21].

Workbench (Mesa de trabajo)

Se trata de un monitor de visualización relativamente grande, como una especie de mesa, que puede ser contemplado por varias personas simultáneamente, utilizando por lo general lentes estereoscópicos activos (Figura 2.10). El funcionamiento es similar a una de las pantallas de un CAVE.

Es un sistema de realidad virtual semi-inmersiva, ya que los lentes no cubren por completo el campo de visión, y el usuario es consciente de que no está inmerso en el sistema al apartar la vista de la mesa de trabajo. Es decir, no hay un mundo virtual que pueda explorar, sino que el usuario se encuentra en un entorno real con objetos virtuales que puede manipular.

Existen diferentes campos de aplicación para la mesa de trabajo virtual. Uno de los más relevantes podría ser la medicina, donde se podría usar este dispositivo para la práctica quirúrgica.

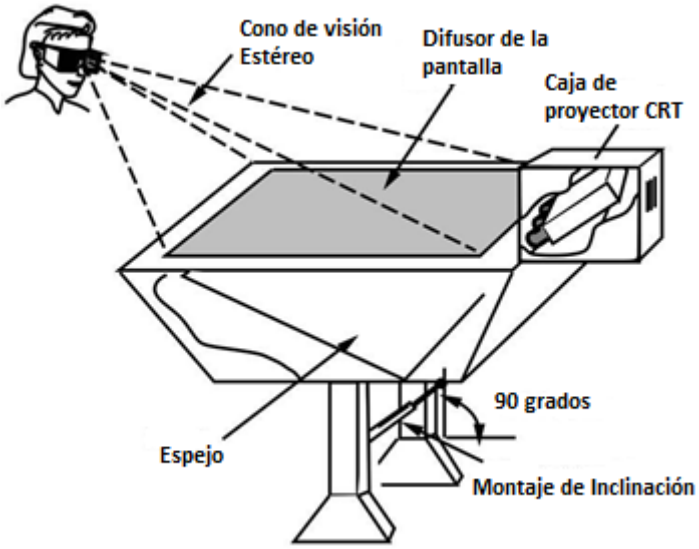


Figura 2.10 – Workbench [16].

2.3.2 Realidad virtual no inmersiva

La realidad virtual no inmersiva es la interacción con medios físicos simples y estándares, como el mouse, teclado y monitor, para interactuar con el sistema virtual. Este enfoque no inmersivo tiene varias ventajas sobre el enfoque inmersivo como son el bajo costo y la fácil y rápida aceptación de los usuarios. Los dispositivos inmersivos son de alto costo y generalmente el usuario prefiere manipular el ambiente virtual por medio de dispositivos familiares como son el teclado y el ratón, y no con cascos pesados o guantes.

La realidad virtual no inmersiva también utiliza el computador y se vale de medios como el que actualmente nos ofrece Internet, en el cual podemos interactuar en tiempo real con diferentes personas en espacios y ambientes que en realidad no existen sin la necesidad de dispositivos adicionales al computador. Nos acercamos en este caso a la navegación, a través de la cual ofrecemos al usuario la posibilidad de experimentar (moverse, desplazarse, sentir) determinados espacios, mundos, lugares, como si se encontrase en ellos.

El alto precio de los dispositivos inmersivos ha generalizado el uso de ambientes virtuales fáciles de manipular por medio de dispositivos más sencillos, como es el ejemplo del importante negocio de las videoconsolas o los juegos en los que numerosos usuarios interactúan a través de Internet. Es a través de Internet como nace VRML, que es un estándar para la creación de estos mundos virtuales no inmersivos, que provee un conjunto de primitivas para el modelaje

tridimensional y permite dar comportamiento a los objetos y asignar diferentes animaciones que pueden ser activadas por los usuarios.

Por último hay que destacar algunas mejoras que facilitan los sistemas de realidad virtual, en lo que se refiere al tratamiento de enfermedades relativas a problemas de movilidad.

A continuación se describen algunos de los dispositivos más populares utilizados para la realidad virtual no inmersiva.

Ratón

Es un dispositivo apuntador utilizado para facilitar el manejo de un entorno gráfico en una computadora. Generalmente está fabricado en plástico y se utiliza con una de las manos. Detecta su movimiento relativo en dos dimensiones por la superficie plana en la que se apoya, reflejándose habitualmente a través de un puntero o flecha en el monitor.

Teclado

Es un periférico de entrada o dispositivo, en parte inspirado en el teclado de las máquinas de escribir, que utiliza una disposición de botones o teclas, para que actúen como palancas mecánicas o interruptores electrónicos que envían información a la computadora.

Palanca de mando (*Joysticks*)

Estos dispositivos de control de dos o tres ejes que son usados desde una computadora o videoconsola hasta un transbordador espacial o los aviones de caza. Se suele diferenciar entre joysticks digitales (que leen cuatro interruptores encendido/apagado en cruceta situada en la base más sus combinaciones y los botones de acción) y joysticks analógicos (que usan potenciómetros para leer continuamente el estado de cada eje, y además de botones de acción pueden incorporar controles deslizantes), siendo estos últimos más precisos.

2.3.3 Ejemplo de realidad inmersiva y no inmersiva

Básicamente el concepto de la realidad inmersiva es aislar al usuario de su entorno real e introducirlo en el mundo virtual que está observando, siguiendo sus movimientos y desplegando el escenario desde su punto de vista, a diferencia de la realidad no inmersiva que usa componentes sencillos de hardware para manipular el entorno virtual que está presenciando.

La situación se pone difusa cuando una parte trata de llegar a la otra, es decir, poniendo de ejemplo la realidad no inmersiva; por ejemplo en video juegos de automóviles usando un joystick

de tipo volante que simula exactamente los controles de un automóvil (incluyendo pedales, giro del volante y demás). Aunque siga siendo realidad virtual no inmersiva, el usuario hace movimientos más naturales (a diferencia del manejo estándar con el teclado y ratón) para desplazarse en el entorno virtual. Sin embargo se sigue viendo una pantalla plana, que hace despliegue de imágenes sintéticas 2D de un mundo 3D.

Entrando en el espacio de la realidad inmersiva, si al sistema mencionado anteriormente se le incorpora una CAVE, el usuario se sumergiría más en el entorno virtual que lo rodea. Si seguimos agregando más elementos, como un dispositivo de rastreo, se puede "posicionar" al usuario en el entorno virtual, consiguiendo una mayor precisión de este en el escenario, logrando que la visión del escenario sea acorde a los movimientos del usuario.

2.4 Realidad aumentada

La realidad aumentada (RA) es el término que se usa para definir una visión directa o indirecta de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real. Consiste en un conjunto de dispositivos que añaden información virtual a la información física ya existente, es decir, añadir una parte sintética virtual a lo real. Esta es la principal diferencia con la realidad virtual, puesto que no sustituye la realidad física, sino que sobreimprime los datos informáticos al mundo real.

Con la ayuda de la tecnología (por ejemplo, añadiendo la visión por computador y reconocimiento de objetos) la información sobre el mundo real alrededor del usuario se convierte en interactiva y digital. La información artificial sobre el medio ambiente y los objetos pueden ser almacenados y recuperados como una capa de información en la parte superior de la visión del mundo real.

La realidad aumentada explora la aplicación de imágenes generadas por computador en tiempo real a secuencias de video como una forma de ampliar el mundo real. La investigación incluye el uso de pantallas colocadas en la cabeza, una pantalla virtual colocada en la retina para mejorar la visualización, y la construcción de ambientes controlados a partir sensores y actuadores.

También existen casos más simples en los que se usan celulares o tabletas para capturar imágenes reales del entorno que lo rodea y agregar a este objetos virtuales, y mediante un sistema centralizado usando internet, las personas pueden compartir estos objetos virtuales en la red, dando información de localización, de modo que cuando otra persona capture ese mismo espacio, podrá observar lo que otras personas ya han hecho.

La definición que ofrece Ronald Azuma [22] en 1997 dice que la realidad aumentada se trata de aumentar el entorno del mundo real con información virtual mejorando las habilidades y sensaciones de las personas. Azuma identifica tres características comunes de las escenas de RA: combina elementos reales y virtuales, es interactiva en tiempo real y está registrada en 3D.

Paul Milgram [23] y Fumio Kishino (*Milgram - Virtuality Continuum* (Virtualidad Continuada) en 1994) definen la realidad mixta (Figura 2.11) como un continuo que abarca desde el entorno real a un entorno virtual puro. En la transición se encuentra la Realidad Aumentada (más cerca del entorno real) que se logra con la visualización del entorno virtual en el real añadiendo objetos virtuales y Virtualidad Aumentada (más cerca del entorno virtual) llevando lo virtual a lo real (como por ejemplo: *Hologramas*[4]).

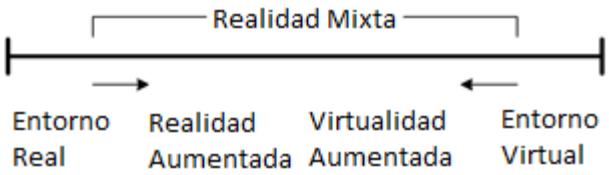


Figura 2.11 – Relación entre el entorno real y el entorno virtual [24].

La RA está se encuentra en desarrollo por universidades y compañías de alta tecnología, y se está implementando con éxito en algunos ámbitos, pero se espera que muy pronto tengamos ya productos de mercado masivo a gran escala.

La idea básica de la realidad aumentada es la de superponer gráficos, audio y otros, a un ambiente real en tiempo real. Podría sonar bastante simple, pero no lo es. Aunque hace décadas que las cadenas de televisión vienen haciendo esto, lo hacen con una imagen estática que no se ajusta al movimiento de las cámaras. La realidad aumentada va mucho más allá de lo que se viene utilizando en televisión(superponer gráficos, audios y videos); ediciones iniciales de Realidad Aumentada se muestran actualmente en eventos deportivos televisados, para mostrar información importante en pantalla, como los nombres de los pilotos de carreras, repeticiones de jugadas polémicas o principalmente, para desplegar publicidad. Estos sistemas despliegan gráficos solo desde un punto de vista.

El punto principal dentro del desarrollo de la RA es un sistema de rastreo del movimiento o *Tracking System* (Sistema de Rastreo). Desde el principio hasta ahora la RA se apoya en "Marcadores" o un arreglo de marcadores dentro del campo de visión de las cámaras para que la computadora tenga un punto de referencia sobre el cual superponer las imágenes. Estos marcadores son predefinidos por el usuario y pueden ser pictogramas exclusivos para cada imagen a ser superpuestas, o formas simples, como marcos de cuadros, o simplemente texturas dentro del campo de visión. Recién en los últimos años el desarrollo de RA "sin rastros" se encuentra

[4] La holografía es una técnica avanzada de fotografía, que consiste en crear imágenes tridimensionales. Para esto se utiliza un rayo láser, que graba microscópicamente una película fotosensible (sensible a la luz). Ésta, al recibir la luz desde la perspectiva adecuada, proyecta una imagen en tres dimensiones.

madurando, añadiendo un nivel más alto de inmersión al no tener que requerir objetos extraños en el ambiente. Los sistemas de computación son mucho más inteligentes, capaces de reconocer formas simples, como el suelo, sillas, mesas, formas geométricas sencillas, como por ejemplo un teléfono celular sobre la mesa, o incluso el cuerpo humano (Figura 2.12). Así, un sistema de seguimiento podría captar, por ejemplo, un puño cerrado y añadir a éste una flor o un sable láser virtual.

(a)  (b) 

Figura 2.12 – (a) La nueva RA es capaz de reconocer la silueta humana. (b) La misma escena con la flor virtual superpuesta [25].

2.5 Dispositivos de rastreo (trackers)

La función de estos dispositivos es permitir que la persona pueda navegar a través del ambiente virtual en el cual está inmerso. Esto se logra mediante sistemas de posicionamiento que utilizan campos electromagnéticos y aquellos que utilizan luz infrarroja.

En 1994, rastrear el movimiento de la cabeza del espectador era un gran problema. El rastreador iba ajustado al espectador. La precisión del rastreador se veía afectado por varios campos de distorsión causados por los objetos metálicos y campos magnéticos.

A diferencia de la tecnología de las pantallas y la tecnología de generación de imágenes, el rastreo no tuvo mercado sustancial a parte de la Realidad Virtual, salvo aplicaciones de entretenimiento. En esa dirección, los diseñadores de dispositivos de captura de movimiento no se enfocaron en mejorar la precisión de esa tecnología.

La Universidad de Carolina del Norte en Chapel Hill, abierta a las investigaciones de rastreo óptico, diseña un dispositivo de captura de movimiento que es sistemáticamente operado en un rango de 18x32 pies, con una precisión de 1mm y 0.1 grados a 1.500 actualizaciones por segundo.

Los Rastreadores comerciales no estándares ofrecen un rango de trabajo de 8 a 10 pies. Los rastreadores de tecnología híbrida parecen más prometedores, combinando tecnologías inerciales, ópticas, ultrasónica y/o magnéticas [13].

Con una amplia gama de rastreadores, otro problema surge: Los cables que los usuarios llevan consigo. La atadura física de los aparatos, no es de mucha molestia cuando el usuario se encuentra de cualquier forma electrónicamente atado, esto se convierte en una molestia ergonómica una vez que el usuario debe caminar alrededor del área fijada de un cuarto. En principio, sustituyendo las conexiones cableadas por las inalámbricas se resuelve el problema expuesto. Los usuarios de los sistemas de proyección envolvente que usan sistemas inalámbricos producidos para la comercialización o producción en masas COTS[5] [26] cumplen este objetivo.

Para los usuarios con HMD el problema se torna más complejo, con dos canales de video de alta definición que deben ser transmitidos (uno por cada ojo), y con los sistemas COTS que aún no poseen las portabilidad requerida. Así mismo, los HMD de visores de rango libre requieren una alimentación de energía que se adhiere al cuerpo del usuario para poder usarlos.

A continuación se describen brevemente los tipos de rastreo utilizados en la realidad virtual.

2.5.1 Tipos de rastreo

Existen distintas formas de rastreo, ya sea vía software como rastreo por video, o vía hardware por rastreo de movimiento.

Rastreo por video (video tracking)

Es el proceso de localizar un objeto en movimiento (o múltiples objetos) en el tiempo utilizando una cámara (Figura 2.13). Esto tiene una variedad de usos, algunos de los cuales son: Interacción humano-computador, seguridad y vigilancia, realidad aumentada, entre otros. El rastreo por video puede ser un proceso que consume tiempo debido a la cantidad de datos que contiene el video, y agregando complejidad al caso es posible necesitar el uso de técnicas de reconocimiento de objetos para el rastreo.

[5] El acrónimo COTS (del inglés Commercial Off-The Shelf, o "comercial, de la estantería"), hace referencia a un producto disponible de forma comercial. Es decir, a aquel que no es desarrollado a medida para una demanda única y específica, sino con unos requerimientos más genéricos y con la intención de ser comercializado y vendido de forma más o menos masiva.

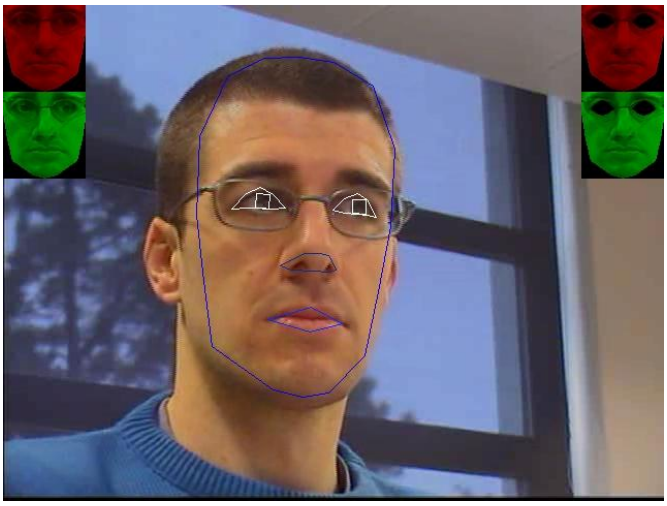


Figura 2.13 – Video tracking (face emotion) [27].

Rastreo de movimiento (motion tracking)

Mediante un conjunto de sensores que se adhieren a un objeto, todos los movimientos de este son digitalizados, para usar la información de movimiento generada, en modelos virtuales similares al objeto siendo rastreado (Figura 2.14). Se usa para mantener el rastro de varios objetos, incluyendo aeroplanos, satélites, vehículos, etc. Tiene la ventaja de proporcionar altas resoluciones de imágenes del objetivo siendo rastreado.

Figura 2.14 – Traje de motion tracking para medir la cinemática de la columna vertebral y de las extremidades [28].

2.5.2 Características de los dispositivos de rastreo

Los dispositivos de rastreo están presentes en todos los sistemas de realidad virtual inmersivos, dando información al computador sobre la posición y orientación del usuario (o alguna parte de su cuerpo: mano, cabeza...). Se pueden destacar varias características comunes a todos ellos:

Grados de libertad (degrees of freedom): Se refiere al número mínimo de parámetros que se necesitan especificar para determinar la velocidad de un mecanismo o el número de reacciones de una estructura u organismo.

- **6 DOF**: Son sistemas ó dispositivos capaces de detectar 6 grados de libertad (Figura 2.15), es decir que detectan la posición obteniendo las coordenadas(*x*, *y*, *z*) y un grado de giro (orientación) en cada uno de los ejes: *yaw*, *pitch*, *roll* (guiñada, cabeceo, alabeo).

- **3 DOF:** Son sistemas ó dispositivos capaces de detectar 3 grados de libertad, es decir que detectan la posición obteniendo las coordenadas (*x*,*y*,*z*).

- **2 DOF:** En esta categoría entran lo que se ha venido usando desde hace ya unas décadas para controlar el computador que es el mouse que tiene dos grados de libertad donde se detecta la posición mediante las coordenadas (*x*,*y*).

La cantidad de grados de libertad es variada entre los distintos tipos de dispositivos ó sistemas. Por ejemplo, el mouse que tiene tan solo dos grados de libertad (*x*, *y*).

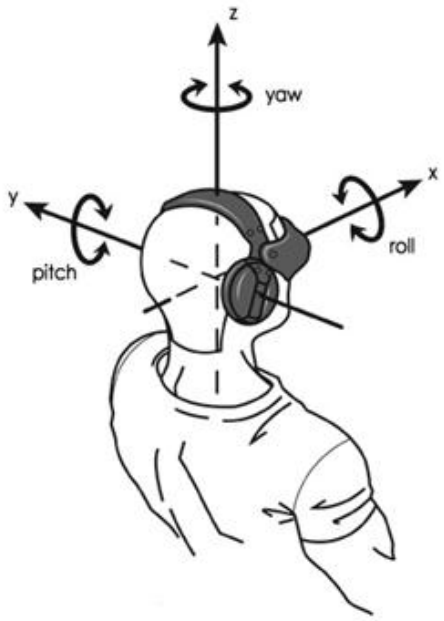


Figura 2.15 – Los 6 Grados de Libertad (6DoF) [16].

Entre otros conceptos que describen la velocidad y exactitud de los dispositivos de rastreo se tienen:

Latencia: tiempo transcurrido entre que se produce un cambio en la posición y/o orientación del objeto y el momento en que se informa al motor de realidad virtual de este cambio. Si este tiempo es lo suficientemente pequeño, no será perceptible para el usuario.

Precisión: diferencia entre la posición real del objeto y la que proporciona el sistema de seguimiento.

Resolución: magnitud del mínimo cambio detectable por el sistema de seguimiento.

Tasa de medida: número de medidas por unidad de tiempo que el sistema de seguimiento proporciona al computador.

Ruido: variaciones en las medidas de la posición del objeto cuando este se encuentra quieto.

En general los *trackers* están compuestos por un dispositivo que genera la señal, un sensor que la recibe y una unidad de control que la procesa y la envía al computador. En algunos sistemas de seguimiento, llamados *inside-out* (Figura 2.16a), los emisores se encuentran repartidos en puntos fijos del entorno virtual, mientras que los sensores se sitúan en el usuario. También existen

sistemas *outside-in* (Figura 2.16b), en los que son los emisores los que lleva el usuario y los receptores se encuentran en el entorno.

El rastreo de cabeza vendría siendo un sistema tipo *outside-in*, ya que este requiere de dispositivos emisor que el usuario necesita llevar y así como otro dispositivo de captura en el entorno, mientras que el señalamiento es de tipo *inside-out*, debido a que el usuario lleva el dispositivo de captura y el dispositivo emisor se encuentra en el entorno.

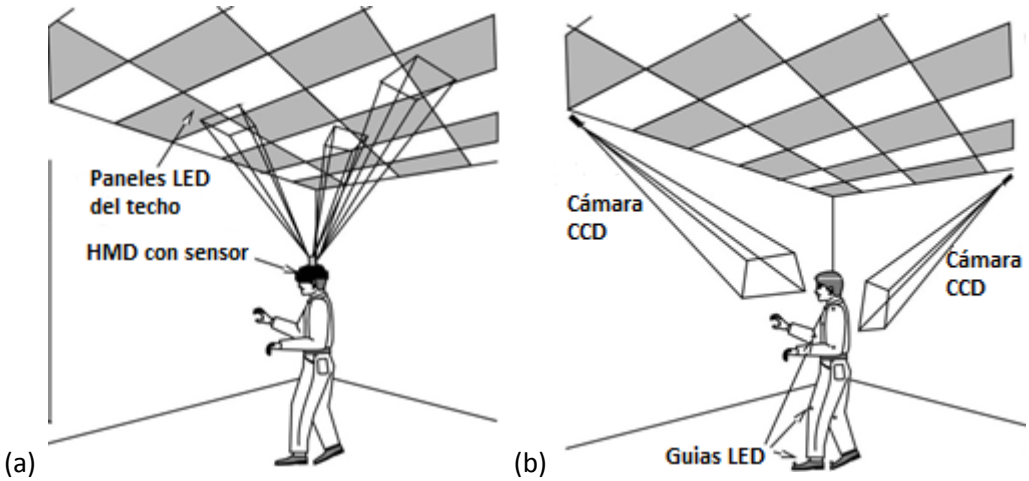


Figura 2.16 – Sistema Inside – Out (a) y sistema Outside – In (b) con cámaras CCD[6] como receptores [16].

2.5.3 Tipos de dispositivo de rastreo

Podemos encontrar distintos tipos de sistemas de rastreo en el mundo de la realidad virtual:

Rastreadores mecánicos [29]: están basados en una conexión física entre el objetivo del seguimiento y un punto fijo. Normalmente el objeto sobre el cual queremos medir su posición y orientación se encuentra en el extremo de un brazo articulado. Estos sistemas proporcionan una latencia muy pequeña, pero tienen el inconveniente de la limitada movilidad y el peso, que los hace menos manejables.

Rastreadores electromagnéticos [29]: estos sistemas miden los campos magnéticos generados por un transmisor fijo para averiguar la posición de un objeto receptor. Para ello hacen uso de la triangulación, empleando 3 emisores y un número variable de receptores. Normalmente

[6] Las cámaras CCD son dispositivos electrónicos muy sensibles, ideados para captar la luz y formar una imagen a partir de ella, pero al mismo tiempo son muy frágiles.

proporcionan tiempos de latencia muy bajos, pero como contrapartida, son sensibles a verse interferidos por cualquier objeto que pueda crear un campo magnético.

Rastreadores ultrasónicos [29]: utilizan ultrasonido producido por un transmisor fijo para determinar la posición y orientación del elemento receptor. Están basados en triangulación. Existen tres emisores de sonido fijos, y en el receptor, que es de forma triangular, se encuentran 3 micrófonos. Análogamente a los electromagnéticos, se pueden ver afectados por la interferencia de otros sistemas que utilicen ultrasonido. Además, suelen proporcionar una tasa de actualización baja.

Rastreadores ópticos [29]: utilizan la luz para conocer la orientación y posición del objeto. Frente a los sistemas anteriores, este proporciona una mayor tasa de actualización y menor latencia. El emisor suele consistir en una serie de LED's infrarrojos y los sensores son cámaras repartidas por el entorno que detectan infrarrojos. Estos sistemas pueden verse afectados por la luz del ambiente u otra radiación infrarroja, y requieren que haya suficiente luz (en caso de cámaras de video normales) y cámaras alrededor del escenario donde se encuentra el objeto a seguir.

Este tipo de seguimiento es el que se emplea por ejemplo en la Wii, cuyo mando está dotado de una cámara para captar la radiación infrarroja enviada por los diez LED's que se encuentran en lo que la Nintendo llama “Barra sensora” y normalmente se encuentra ubicada encima de la T.V. donde se despliega el videojuego. Así, mediante triangulación es posible obtener la posición y rotación del mando [5].

Rastreadores inerciales: usan las propiedades físicas asociadas al movimiento para detectar la aceleración (mediante acelerómetros de 3 ejes que calculan el vector aceleración) y la rotación (mediante giroscopios de 3 ejes) de los objetos y así conocer su posición y orientación. Hay una variedad de rastreadores inerciales que utilizan también magnetómetros para medir el campo magnético terrestre, y reúne la información de los 3 dispositivos (acelerómetro, giroscopio y magnetómetro) para calcular la posición del objeto con mayor exactitud. La principal ventaja de este tipo de sistemas de seguimiento es que son independientes: no necesitan ningún tipo de fuente o referencia externa para funcionar. Además, pueden crearse rastreadores inerciales muy pequeños utilizando técnicas de fabricación de semiconductores, lo cual es otra ventaja añadida. También su baja latencia (normalmente inferior a los 2 ms.) es un factor que los hace muy útiles.

Un ejemplo de ello es el mando de la consola Wii (WiiMote), que funciona gracias a un acelerómetro multieje, o el Wii MotionPlus [30], un accesorio para este mando que contiene un giroscopio multieje para mejorar el seguimiento (la orientación sobre todo) del objeto (el usuario en este caso).

A continuación se mencionan algunas implementaciones exitosas que utilizan el wiimote.

Rastreo de cabeza para pantallas de escritorio de RV "*Desktop VR Displays*" usando el WiiMote

Usando la cámara infrarroja en el WiiMote y una barra de sensores ajustada en la cabeza (dos IR Leds), se puede rastrear con precisión la localización de la cabeza y desplegar la vista dependiendo de las imágenes en la pantalla. Esto efectivamente transforma tu pantalla en un portal para un ambiente virtual. La pantalla reacciona correctamente a los movimientos de la cabeza y el cuerpo como si fuera una ventana real creando una ilusión realística de profundidad y espacio.

El programa solo necesita saber el tamaño de la pantalla y el tamaño de la barra de sensores. El software fue realizado en C# usando DirectX, y se provee una muestra de ejemplo principalmente para desarrolladores sin soporte o documentación adicional. Este sistema fue desarrollado por Johnny Chung Lee [5].

Pizarra interactiva de multi-señalamiento de bajo costo usando el Wiimote

Ya que el Wiimote puede rastrear varias fuentes de luz infrarroja(IR), pueden rastrearse lápices que tienen un led IR en la punta. Apuntando un wiimote en una pantalla proyectada o LCD, se puede crear pizarras o tablas interactivas de muy bajo costo. Dado que el wiimote puede rastrear hasta un máximo de 4 puntos IR, se pueden usar hasta 4 "lápices". También funciona muy bien con pantallas retro-proyectadas. Este sistema fue desarrollado por Johnny Chung Lee [3].

Rastreo de dedos con el Wiimote

Usando un arreglo de LED y alguna cinta reflectiva, se puede usar la cámara infrarroja en el Wiimote para rastrear objetos, como los dedos, en un espacio 2D. Esto permite interactuar con el computador simplemente moviendo las manos en el aire similar a la interacción vista en la película "MinorityReport [31]". El Wiimote puede rastrear hasta 4 puntos IR simultáneamente. El software de red multipunto esta hecho en C# con DirectX. Este sistema fue desarrollado también por Johnny Chung Lee [3].

El precio del Wii Remote en comparación con otras soluciones disponibles en el mercado es el más económico en la relación costo-beneficio (Tabla 1). El Kinect, aunque es una solución aceptable (ya que no necesita dispositivos extras para realizar la detección y el rastreo de la persona), su costo saldría más de dos veces el que ya se ha planteado con el del Wii. Las Webcams, que son relativamente económicas, son incapaces de hacer el rastreo en condiciones de poca iluminación. El TrackIR es una solución de mercado destinada a jugadores de videojuegos, es una solución compacta que la hace la más costosa de todas las mencionadas.

Dispositivo	Precio	Descripción
Wii Remote + Led Bar	~50$	Solución de rastreo de mano del Wii, el cual puede ser re-adaptado para rastrear la cabeza.
Webcam HD 1080p	~60 - 80$	Utilizando procesamiento de imágenes.
Kinect Sensor [32]	120$	Conformado por un sensor de profundidad, una cámara de color VGA y un arreglo de micrófonos.
TrackIR 5 [33]	150$	Solución de mercado lista para su uso.

Tabla 1 – Precios de algunos dispositivos de rastreo.

2.6 Wii

El Wii (Figura 2.17) es una consola de videojuegos diseñada por Nintendo [6], estrenada el 19 de noviembre de 2006. La característica más distintiva de la consola es su control inalámbrico, el Wii Remote, el cual puede usarse como un dispositivo de mano con el que se puede apuntar, además de poder detectar movimientos en un plano tridimensional. Otra de sus peculiaridades es el servicio WiiConnect24, que permite recibir mensajes y actualizaciones a través de Internet en modo de espera. Adicionalmente, la consola puede sincronizarse con la portátil Nintendo DS, lo cual permite que Wii aproveche la pantalla táctil de la Nintendo DS como mando alternativo.

Figura 2.17 – Nintendo Wii.

Composición del Nintendo Wii

El Wii Remote (Figura 2.18) es el control principal de Wii [6]. Utiliza una combinación de acelerómetros y detección infrarroja para determinar su posición en un espacio tridimensional cuando puede captar los LEDs de la barra de LEDs infrarrojo (lo que nintendo llama la "Barra sensorial"). Este diseño le permite a los usuarios controlar el juego mediante gestos físicos, así como presionar los botones clásicos de un controlador estándar. El controlador se conecta a la consola mediante Bluetooth, puede vibrar y tiene un altavoz interno. El Wii Remote puede conectarse a otros dispositivos a través de un puerto propietario ubicado en la base del controlador. El dispositivo que viene incluido con el paquete de la consola Wii es el Nunchuk, el cual cuenta con un acelerómetro y un control tradicional con dos botones y una palanca.

Debido a que la resolución dada por la cámara del control del Wii está predefinida por su hardware, las coordenadas son calculadas en el hardware y entregadas vía bluetooth en una resolución de 1024*768.

Figura 2.18 – Wii Remote.

La "barra sensorial" del Wii (Figura 2.19) es sencillamente un arreglo de LEDs infrarrojos a cada extremo de la barra, teniendo originalmente en los primeros modelos 10 LEDs distribuidos en la barra, de modo que quedaban 5 LEDs infrarrojos en cada extremo (con 3 apuntando hacia al frente y uno apuntando ligeramente hacia la izquierda y otro a la derecha) conectados en series con una resistencia. A su vez estos 5 LEDs estaban conectados a los otros 5 en paralelo. La barra ahora posee solo 6 LEDs infrarrojos en total (con solo un LED apuntando hacia el frente) en su diseño manteniendo todo lo demás idéntico, los cuales son suficiente para ocupar los mismos ángulos que la barra de 10 LEDs. Estos LEDs emiten luz infrarroja la cual el Wii Remote recibe con su cámara receptora para realizar el cálculo de posicionamiento.

Figura 2.19 – “Barra sensora” del Wii.

2.7 Head mounted display (HMD)

Un *Head Mounted Display* [34]o HMD es un dispositivo de visualización similar a un casco, que permite reproducir imágenes creadas por computador sobre un "display" muy cercano a los ojos o directamente sobre la retina de los ojos. En este segundo caso el HMD recibe el nombre de monitor virtual de retina.

Existen dos tipos de HMD:

- Monocular: las imágenes creadas por computador sólo se muestran sobre un ojo (Figura 2.20a).
- Binocular: las imágenes creadas por computador se muestran sobre los dos ojos (Figura 2.20b).

Debido a su proximidad con los ojos el HMD consigue que las imágenes visualizadas resulten mejores que las percibidas por pantallas normales, y permiten incluso englobar todo el campo de visión del usuario. Gracias a que el "display" se encuentra sujeto al casco, este puede seguir los movimientos del usuario, consiguiendo así que éste se sienta integrado en los ambientes creados por computador.

Algunos tipos de HMD (Figura 2.20b) reducen el campo de visión del usuario de modo que no tiene influencias visibles del entorno que lo rodea, permitiendo así la completa inmersión de éste en una realidad virtual, ya que sólo percibirá las imágenes creadas por computador y reproducidas sobre el "display".

Otros tipos de HMD (Figura 2.20a) permiten al usuario ver todo el entorno que lo rodea, e incluir objetos virtuales, introduciendo así lo que se conoce como realidad aumentada o realidad mixta si el usuario puede interactuar con estos objetos virtuales proyectados.

El avance más notable en HMD (Figura 2.20b) ha sido en la resolución de estos dispositivos, también la saturación del color, brillo, y la ergonomía se ha visto mejorada considerablemente. En 1994, se tenía la opción de los costosos y engorrosos HMD-CRT, los cuales tenían excelente resolución y color, o los económicos HMD-LCD, los cuales tenían baja resolución y una saturación pobre. Hoy en día los HMD-LCDs económicos tienen una resolución aceptable de 640x480 pixeles y una buena saturación de colores.

(a)  (b) 

Figura 2.20 – Un HMD Monocular (a) y un HMD Binocular (b).

Capítulo 3. Mesa virtual y corrección perspectiva

En este capítulo se describen detalles de construcción y funcionamiento de la mesa de trabajo, así como trabajos que lidian con el problema de la perspectiva correcta al considerar el punto de vista del usuario.

3.1 Mesa de trabajo (*Workbench*)

Una mesa de trabajo o *workbench* corresponde a una gran pantalla de visualización que permite a múltiples personas observar al mismo tiempo una determinada escena. Las Mesas de Trabajo poseen una pantalla plana de retro-proyección y un proyector posicionado de tal manera que la longitud aproximada sea la de un cuerpo humano. Por lo general, se rastrea la posición de un usuario, y el resto de los usuarios observan la imagen desde la perspectiva del usuario rastreado.

A comienzo de los 90, el Oficial de la *Naval Research's Computer Science Liaison Scientist*, Larry Rosenbaum [35], lideró un equipo de ingenieros de Realidad Virtual Naval para la creación de un gran monitor (Figura 3.1). Los usuarios veían la pantalla verticalmente o inclinada horizontalmente como una mesa o un banco. Los usuarios usaban lentes especiales mientras veían la mesa de trabajo, tal y como si estuvieran en un sistema CAVE (Cueva). Cada usuario veía la misma imagen proyectada desde la pantalla de la Mesa de Trabajo. Debido a la proyección estereoscópica y los obturadores de los lentes, el objeto mostrado en la mesa de trabajo parecía ser tridimensional.

La razón por la cual los investigadores de realidad virtual sienten que la mesa de trabajo no es una verdadera representación de la creación de un ambiente virtual se debe a la inmersión, porque el usuario está viendo una pantalla que no ocupa su campo de visión. El usuario permanece consciente de que esta en el mundo real, aunque ese mundo ahora incluye objetos virtuales que él puede manipular. No hay un ambiente virtual que explorar; si el usuario se aleja de la pantalla, solo observará el resto del sitio donde se encuentra. Sin embargo, esto no cambia el hecho de que las mesas de trabajo pueden ser muy útiles.

Un uso para la mesa de trabajo de visualización es el entrenamiento médico. Un cirujano puede practicar un procedimiento en un paciente virtual tri-dimensional mientras se encuentra rodeado de un grupo médico real. Si el cirujano quisiera realizar el mismo procedimiento mientras lleva un HMD, las personas alrededor de él pueden ser personajes bajo control computarizado o avatares de computadoras representando otros humanos. Con la mesa de trabajo de visualización, la interacción con otras personas es natural y completamente real.

Figura 3.1 – Uso de Workbench con VR en el laboratorio de la fuerza naval estadounidense [35].

Las mesas de trabajo de visualización son también útiles para estrategia militar. Los programadores pueden crear campos de batallas tri-dimensionales realísticos, dando al personal militar una situación precisa de la vista de la batalla. Un buen modelo puede también revelar potenciales cuellos de botella o campamentos enemigos ocultos.

Otras aplicaciones usando las mesas de trabajo de visualización pueden ser la visualización de investigaciones científicas o investigación y desarrollo de productos. En particular, el Centro de Computación Gráfica de la Facultad de Ciencias de la UCV posee una mesa de trabajo virtual (Modelo QVW-001, Largo: 1.68m, Altura: 0.91m, Ancho: 1.30m, Peso: 206Kg, Serial: 02-100) (Figura 3.2), que es utilizada para realizar investigación principalmente en el área médica. Actualmente cuenta con un proyector con el cual se puede implementar estereoscopía, utilizando la técnica de anaglifos.

Figura 3.2 – Mesa de trabajo del centro de computación gráfica de la Facultad de Ciencias [36].

3.2 Corrección de perspectiva

Para determinar la posición de la barra de LEDs infrarrojos (en este caso los lentes) relativa a la cámara sensora del wiimote, se leen y analizan la ubicación de los puntos infrarrojos (que están posicionados uno a cada lado de los lentes) dados por el wiimote. Utilizando triangulación (Figura 3.3), la distancia que se obtiene entre los dos puntos (LEDs IR) permite deducir la distancia (línea roja) que existe entre los lentes y el wiimote en un momento dado. La posición de ambos puntos determina el ángulo tanto del plano X como Y de los lentes, los cuales, combinados con la distancia mencionada anteriormente, permite el cálculo de la posición *positionX*, *positionY* y *positionZ* con respecto al wiimote. Tomando en cuenta el desplazamiento y la orientación del wiimote que tienen los lentes con respecto a la pantalla de la mesa, se pueden calcular las coordenadas *X*, *Y* y *Z* de la posición del usuario.

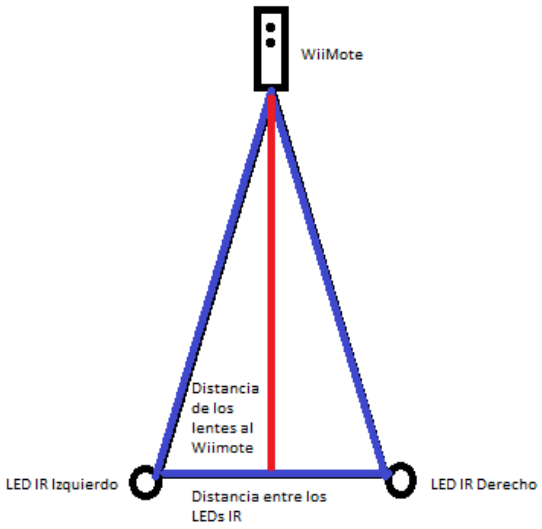


Figura 3.3 – Triangulación.

Los lentes tienen que estar siempre paralelos a la pantalla plana de la cámara del Wiimote, porque de otra forma la rotación causaría que la distancia percibida de las dos fuentes de luz infrarrojo sea errada. Una solución a esto sería implementar una barra con 4 puntos infrarrojos no planares permitiendo así rastrear la rotación.

Para lograr que la interfaz genere una ilusión de profundidad, los parámetros de visualización tienen que ajustarse dada la información sobre la posición de la cabeza del usuario. En la mayoría de interfaces de representación 3D, existe la noción de un plano virtual de la pantalla y una cámara virtual. La cámara y el plano definen un Volumen Delimitado (*Bounding Volume*), el cual es frecuentemente referido como *Viewing Frustum* (Figura 3.4).

Cuando se despliega la imagen, los datos tridimensionales tienen que ser mostrados sobre una pantalla bidimensional. A fin de lograr esto, la proyección sobre el plano imágen tiene que ser aplicada. Esta proyección hace que los objetos más distantes se reduzcan más de tamaño que los más cercanos. Dos parámetros que controlan esta proyección son la posición de la cámara en relación a la pantalla, y el tamaño de la pantalla. Ajustando cualquiera de estos parámetros, el ángulo en el cual los objetos son visibles, llamado Campo de Visión (FOV: Field of View), puede ser cambiado (Figura 3.4). Esto es lo mismo que se logra cuando una persona se acerca a una ventana, o se aleja de ella. Cuando se acerca a la ventana, más espacio de la parte exterior será visible, lo que significa que el Campo de Visión (FOV) de la persona se ha incrementado. Para Incrementar el Campo de Visión, similar al caso de la ventana, tanto la cámara puede ser movida hacia el plano de la pantalla, o el plano de la pantalla se puede ampliar.

Ajustando la posición de la cámara en la localización de la cabeza del usuario en el mundo virtual, y el plano de la pantalla en la localización de la mesa de trabajo (*Workbench*), una imagen de perspectiva corregida se genera con la vista actual del usuario. Esto se conoce como "*FishTank VR*" [37]. Esto responde inmediatamente la pregunta de cómo realizar la corrección de perspectiva. Simplemente ajustando la posición de la cámara virtual y el plano virtual de la pantalla, dando una añadida ilusión de perspectiva.

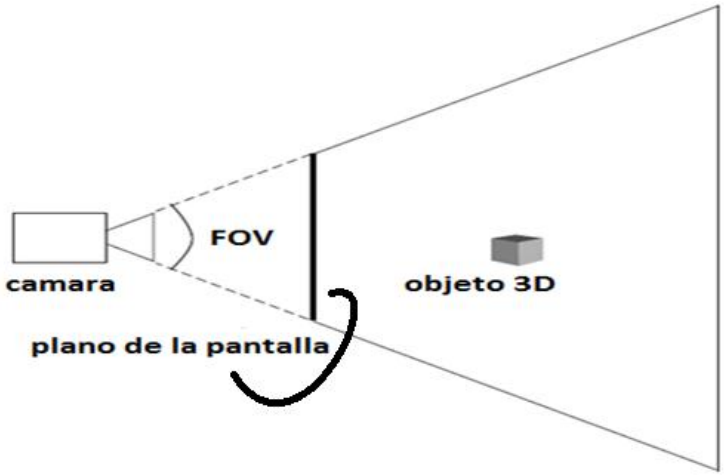


Figura 3.4 – Vista del Frustum [37].

3.3 Estereoscopía

Como se explicó en 2.2, existen varios métodos para el despliegue de pares estéreo [1], muchos de estos son estrictamente incorrectos ya que introducen paralaje vertical.

3.3.1 Paralaje

Paralaje (parallax en inglés) es el cambio aparente de la posición de un objeto observado producido por una variación de la posición del observador. La distancia entre puntos homólogos entre las proyecciones para el ojo izquierdo y derecho es llamada paralaje horizontal. Los objetos que se encuentran en frente del plano de proyección (foco de visión) tienen un paralaje negativo (están delante del foco de visión), mientras que los objetos que se encuentran detrás del plano de proyección tienen un paralaje positivo (están detrás del foco de visión).

Existen tres tipos de paralaje:

- Paralaje negativo: Si un objeto está ubicado en frente del plano de proyección, entonces la proyección para el ojo izquierdo se encuentra a la derecha, y la proyección para el ojo derecho se encuentra a la izquierda. Esto se conoce como paralaje negativo.

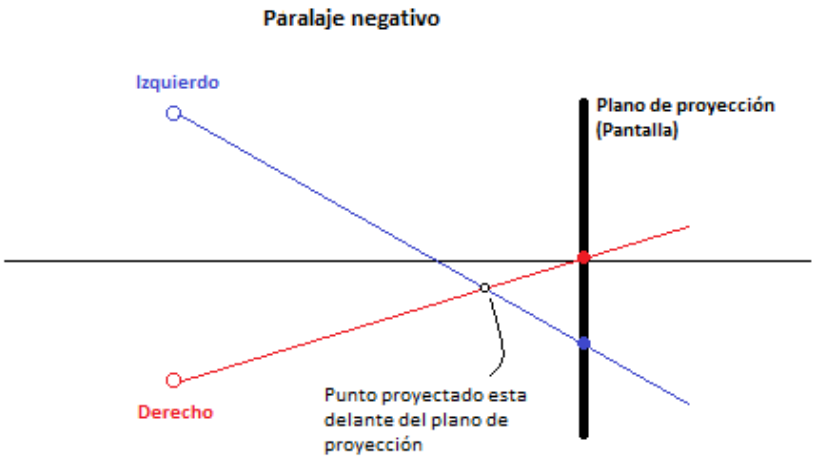


Figura 3.5 – Paralaje negativo

- Paralaje positivo: Si un objeto está detrás del plano de proyección, la proyección para el ojo izquierdo se encuentra a la izquierda y la proyección para el ojo derecho se encuentra a la derecha. Dado que las proyecciones están del mismo lado que sus respectivos ojos se le dice paralaje positivo.

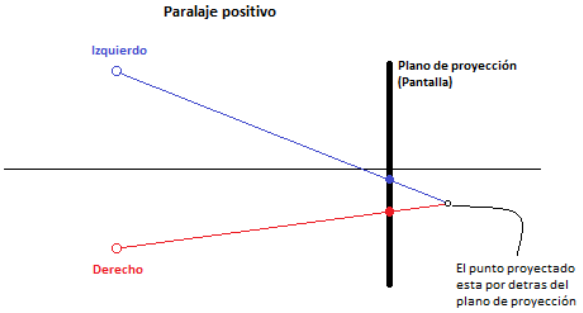


Figura 3.6 – Paralaje positivo

- Paralaje cero: Si un objeto se encuentra en el plano de proyección, entonces su proyección en el plano de foco coincide para ambos ojos. Por lo tanto no hay paralaje horizontal.

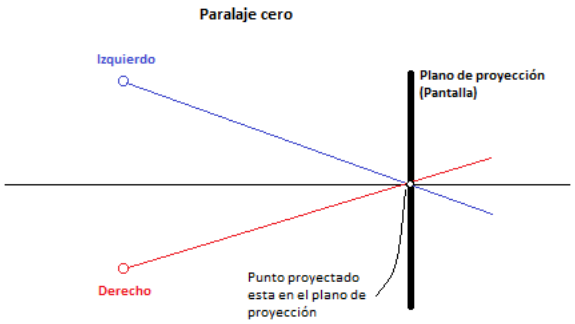


Figura 3.7 – Paralaje cero

Hay dos formas de crear pares estéreos: el método Off-axis y el método Toe-in. A continuación se describe cada una de ellas.

3.3.2 "Off-Axis" – desplazado de Eje

Esta proyección no introduce paralaje vertical y por lo tanto crea los pares de imágenes estéreos menos estresantes o incómodas. En este método se requiere que la dirección de la vista de ambas cámaras sea totalmente paralela basada en la distancia intraocular además de que los frustums izquierdo y derecho sean asimétricos (ver Figura 3.8). Note que en el plano de proyección coinciden ambos frustums.

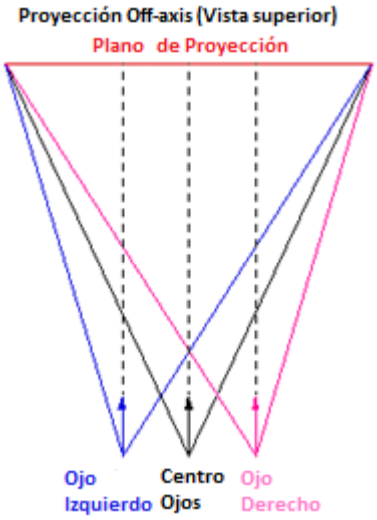


Figura 3.8 – Proyección Off Axis

3.3.3 "Toe-In"

En esta proyección las cámaras tienen una apertura fija y simétrica, cada cámara apunta a un único punto focal. Las imágenes creadas usando este método siguen siendo estereoscópicas pero debido al paralaje vertical que introduce causa incomodidad. El paralaje vertical aumenta desde el centro del plano de proyección y es más notable a medida que la apertura de la cámara se incrementa (ver Figura 3.9).

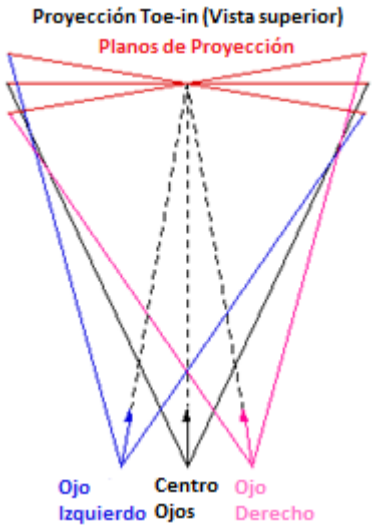


Figura 3.9 – Proyección Toe-in

3.3.4 Características de estereoscopía

- Distancia intraocular (DIO): Es la distancia de separación de los ojos, mientras más distancia intraocular tenga el individuo mayor será el paralaje.
- Eje óptico: Es la línea imaginaria que se forma desde el centro del ojo o cámara hasta el punto de enfoque.
- Punto de convergencia: Es el punto donde se intersectan los ejes ópticos, es decir, el plano donde se fija la mirada que puede estar ubicado antes o después de la posición del objeto percibido.

Paralaje vertical (Vertical Parallax en inglés) es la disparidad causada por la rotación de las imágenes proyectadas de cada ojo. En la Figura 3.10.a se puede apreciar el paralaje vertical que se forma al utilizar Toe-in (principalmente en la esquina izquierda inferior y

derecha). A diferencia de la Figura 3.10.b donde no hay paralaje vertical utilizando Off-axis. Causando menos estrés en la fusión de la imágenes, y por lo tanto una representación estereoscópica correcta.

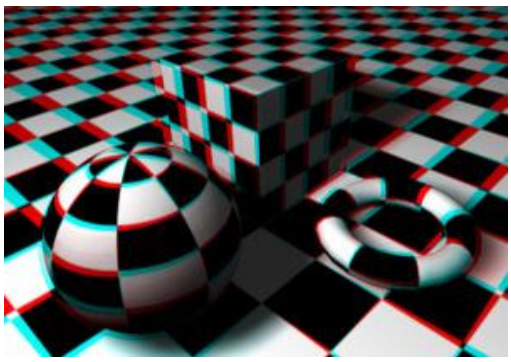
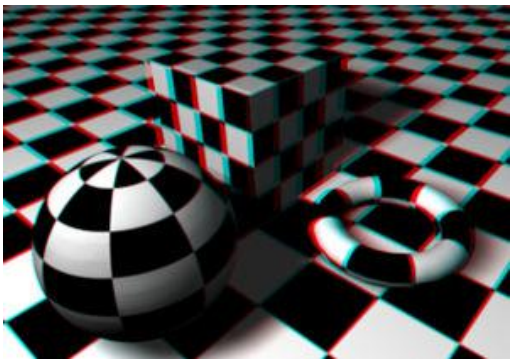
a)  b) 

Figura 3.10 – a) Imagen con paralaje vertical b) Imagen sin paralaje vertical [38]

Otra forma de ver el paralaje vertical es si tomamos dos planos idénticos, y los rotamos respecto a una línea vertical (ver Figura 3.11), se puede notar el desplazamiento vertical de puntos homólogos.

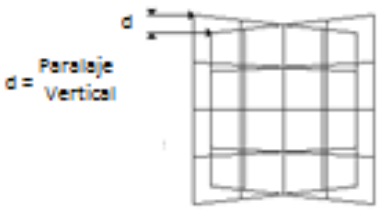


Figura 3.11 – Paralaje vertical.

3.4 Mesas de trabajo de realidad virtual en forma de L (*L-Shaped Workbench VR*)

Cuenta con dos superficies de proyección ortogonal 3D, haciendo una visualización muy cercana a los soñados hologramas futurísticos. Estas mesas de trabajo de RV son más inmersivas que las mesas convencionales de Realidad Virtual. Están hechos de madera, para no interferir con los sistemas de rastreo electromagnéticos. Tienen espejos de alta reflexión para la superficie de proyección horizontal, 2 áreas de proyección de aproximadamente 1,80 metros por 1,10 metros

cada uno. Cada área cuenta con un proyector y un espejo que refleja la proyección. Si un objeto de la imagen se mueve hacia arriba, este se puede seguir viendo en la pantalla superior (Figura 3.12).

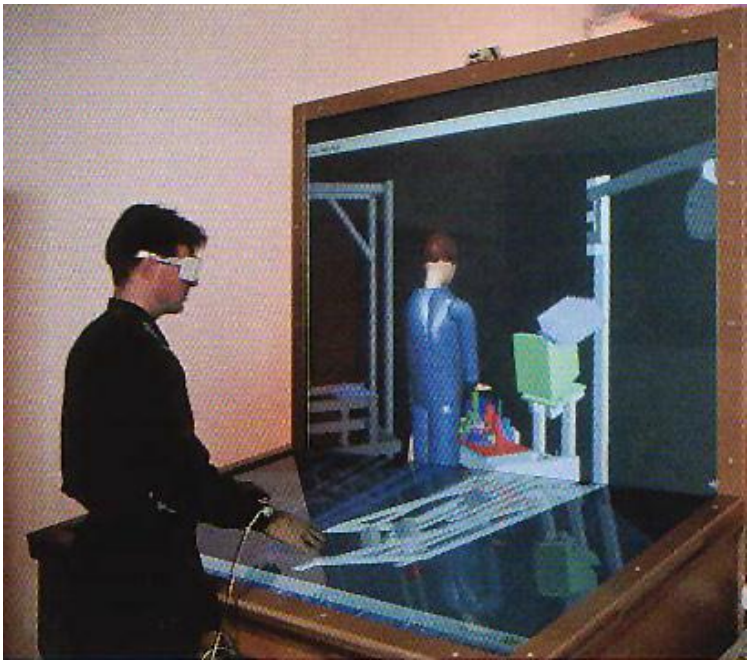


Figura 3.12 – El TAN HOLOBENCH™ L-Shaped Workbench VR [39].

3.5 Arquitectura del workbench del CCG

El funcionamiento del Workbench se lleva cabo gracias a 3 dispositivos claves: un proyector, un espejo de alta reflexión y un computador con una tarjeta gráfica 3D que posee 2 dos salidas de video.

Las 2 señales de video van conectadas a dos proyectores. Cada señal corresponde a la imagen generada para uno de los ojos del usuario. Para desplegar las imágenes sobre la mesa, estas se proyectan en el espejo de alta reflectividad, y este a su vez proyecta sobre la pantalla que está sobre la mesa. A cada proyector se le coloca un filtro polarizado distinto, que sólo deja pasar la luz que viaja en una dirección específica. Sin embargo, estas dos imágenes se funden en la pantalla de proyección que está sobre la mesa. Para lograr la estereoscopía, la pantalla no es polarizante, por lo que las imágenes que corresponden a cada ojo pueden volver a separarse aplicando el filtro polarizado correspondiente en los lentes que utiliza el usuario (Figura 3.13). Actualmente uno de estos proyectores está dañado, por lo que la estereoscopía se logra con anaglifos.

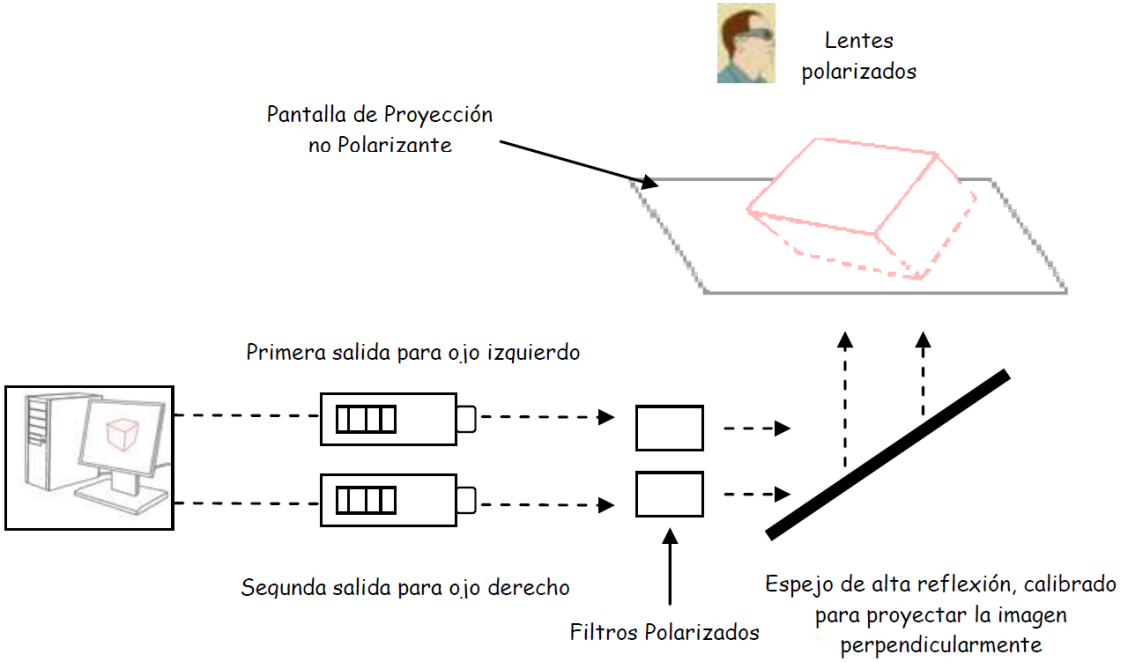


Figura 3.13 – Arquitectura del CCGWorkbench [4].

3.6 Trabajo previo

En el CCG se implementó un sistema de rastreo de video utilizando una cámara web. La tesis "Localizador de Usuario para la Mesa de Trabajo Virtual utilizando Captura de Video vía Webcam" **[4]** utiliza una cámara para saber la posición aproximada del usuario con respecto a la mesa de trabajo virtual, mediante la captura y procesamiento de imágenes. Todo esto es implementado como una librería para integración con otros proyectos de realidad virtual que requieran el uso del workbench con rastreo de posición de usuarios.

Debido a que la intensidad de la luz influye en la detección del movimiento con la cámara, la aplicación se puede ajustar mediante unos parámetros para aplicar unos filtros u otros dependiendo de la intensidad de la luz presente en el ambiente de implementación. Uno de los mejores resultados se obtuvo donde se tenía un fondo constante unicolor y las luces encendidas, independientemente del color de la ropa que el usuario utilice. Esta implementación no toma en cuenta la corrección perspectiva la cual es importante para poder desplegar la imagen correcta en la mesa virtual desde el punto de vista del usuario.

Capítulo 4. Solución propuesta

En este capítulo se describe como se construyó el hardware para realizar el rastreo del usuario utilizando el control del wii, la barra sensora y lentes de anaglifos. Se describe adicionalmente la implementación de la librería WiiW (Wii para Workbench) que sirve de puente ente el hardware mencionado y una aplicación de visualización estereoscópica para una pantalla horizontal, como es el caso de la mesa de trabajo o workbench. Así, se implementan los algoritmos para hacer el rastreo de la cabeza y la corrección perspectiva.

4.1 Diseño del hardware

Se dispone de dos controles de Wii, los cuales serán utilizados para rastrear la posición del usuario y para control de la aplicación como dispositivo de señalamiento, y se dispone de dos barras de LEDs IR las cuáles son modificadas, para ajustarse a las necesidades del sistema.

Siguiendo el siguiente esquema referencial se puede construir una barra de LEDs "casera". Con un par de LEDs infrarrojos, una batería de 1.5 Voltios o más y una resistencia opcional (necesaria si se quiere usar una batería con mayor voltaje) que depende de la cantidad de voltaje y LEDs a utilizar, el esquema para este circuito es el indicado en la Figura 4.1.

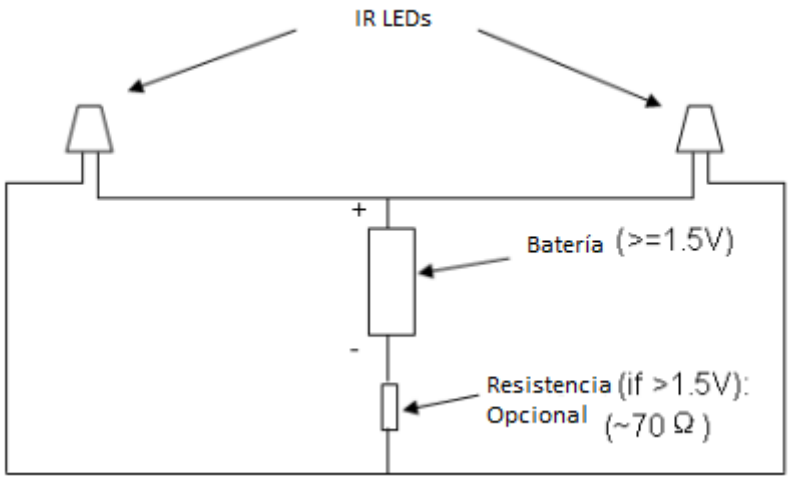


Figura 4.1 – Esquema referencial para la realización de la barra de LEDs infrarrojo [40].

4.1.1 Barra de LEDs IR

La barra “sensora” del Wii viene originalmente con un arreglo de 6 LEDs IR en sus últimas versiones (las primeras versiones venían con 10 LEDs) distribuidos 3 LEDs al extremo izquierdo y 3 LEDs al extremo derecho de la barra. En cada lado hay una resistencia de 20 Ohm. Los LEDs están conectados en paralelo al cable principal que se conecta a una salida de 12 Voltios proveniente de la consola del Wii. Para nuestro caso particular, es necesario que esta barra tenga su propia fuente de energía para que sea movible y se pueda usar con la mesa de trabajo. Esta barra será utilizada para el wiimote que servirá como apuntador.

Para convertir esta barra en “inalámbrica” se le adhiere un encapsulador para 4 baterías AA al cable de alimentación principal, y se le cambian las resistencias a 10Ohm para que de esta forma tenga la mayor cantidad de corriente posible sin perjudicar la duración de los LEDs (ver Figura 4.2).

Figura 4.2 – Barra de LEDs IR

Figura 4.3 – Controles Wii

4.1.2 Lentes estereoscópicos con LEDs IR

Los lentes estereoscópicos fueron adquiridos a un precio de 39Bs, con montura de plástico y con la combinación rojo y azul para anáglifos dado que no se encontraba rojo y cyan.

Para colocarle los LEDs a los lentes se sacaron de una barra "sensora" Wii, y se le hizo la siguiente modificación: Se le extrajeron 2 LEDs de los 3 LEDs del lado izquierdo y 2 LEDs de los 3 LEDs del derecho, dejando uno de cada extremo de la barra. Se cambiaron las resistencias de 20 Ohm a 10 Ohm, para una mejor estabilidad de corriente, ya que nivela la potencia con la que los LEDs iluminan y la durabilidad a este nivel de iluminación para evitar sobrecarga. El cable principal se le conecta un encapsulado de dos baterías AA.

A la montura de los lentes se le adhirieron dos LEDs IR conectados por un cable. Estos LEDs están conectados a una resistencia de 10 Ohm cada uno, para evitar el exceso de energía y para que tenga una duración más prolongada con las dos pilas recargables AA de 1.2 Voltios a 2700 mAh.

Figura 4.4 – Elaboración de los lentes.

4.2 Diseño de WiiW

WiiW son las siglas de Wii Workbench. Es una librería, para ser usada en aplicaciones de despliegue desarrolladas en OpenGL, con el objetivo de incrementar el nivel de inmersión e interacción con la mesa de realidad virtual.

A continuación se describe el procedimiento utilizado para realizar el rastreo de cabeza, así como para la manipulación de objetos de la escena.

4.2.1 Rastreo de la cabeza

Los LEDs IR están ubicados en la cabeza del usuario (al nivel de cada ojo). Estos LEDs son detectados por la cámara infrarroja del wiimote, y nos va a permitir determinar la ubicación espacial del usuario respecto al control de wiimote. Esa posición debe ser transformada de coordenadas de wiimote a coordenadas físicas de la mesa (coordenadas de workbench). Conociendo la ubicación del usuario respecto a la mesa, se puede determinar así la ubicación del usuario en la escena virtual para saber dónde ubicar la cámara para hacer el despliegue.

La cámara infrarroja del wiimote captura imágenes de 1024x768. Así, la posición de los LEDs infrarrojos capturados por esta cámara estarán en el rango de 0..1023 para *x* y 0 ..767 para *y*. El primer procedimiento a realizar es llevar esos valores *x*, *y* de coordenadas de wiimote a coordenadas de workbench. Para ello, primero se realiza un proceso de calibración. El usuario debe colocar los lentes sobre la esquina superior izquierda de la mesa que corresponde al punto

origen de las coordenadas de workbench, luego sobre la esquina superior derecha y por último el usuario debe colocarse los lentes en su cabeza, y ubicarse en la esquina derecha de la mesa viendo hacia el wiimote. En cada una de estas tres posiciones, el usuario debe presionar el botón "2" del segundo wiimote para que el sistema capture esas posiciones del usuario en coordenadas de wiimote. Estas posiciones están a su vez relacionadas con coordenadas físicas del workbench, lo que va a permitir luego llevar cualquier posición (*x*,*y*) de coordenadas de wiimote a coordenadas de workbench vía interpolación.

Sean (x_1,y_1) y (x_2,y_2) las posiciones de los LEDs capturados en coordenadas de wiimote, separados por una distancia *r* (Ver Figura 4.5). Sean además los puntos de calibración *EsqIzq*, *EsqDer* y *EsqDerAltura* en coordenadas de wiimote. Y finalmente, sean los rangos de la mesa *XFisicoMinimo*..*XFisicoMaximo* para *x*, y *YFisicoMinimo*..*YFisicoMaximo* para *y* en coordenadas de workbench. Podemos llevar el punto medio de los LEDs (*eyePosition*) de coordenadas de wiimote a coordenadas de workbench por simple interpolación. Para el eje x utilizamos la siguiente fórmula:

$$eyePositionWorkbench.x = (1-t) * XFisicoMinimo + t * XFisicoMaximo$$

donde:

$$t = \frac{eyePosition.x - EsqIzq.x}{EsqDer.y - EsqIzq.y}$$

Para el workbench asumiremos que *XFisicoMinimo* =0 y *XFisicoMaximo*=*WorkbenchWidth*, donde *WorkbenchWidth* es el ancho real de la mesa.

Análogamente se calcula posición en *y* con la pequeña diferencia que promediamos (para una mejor precisión) las dos alturas inferiores que se tienen de la calibración:

$$eyePositionWorkbench.y = (1-t) * YFisicoMinimo + t * YFisicoMaximo$$

donde:

$$YMenor = \frac{EsqIzq.y + EsqDer.y}{2} \quad \text{y} \quad t = \frac{eyePosition.y - YMenor}{EsqDerAltura.y - YMenor}$$

Para no utilizar del usuario su altura real, asumiremos que *YFisicoMinimo*=0 y Y*FisicoMaximo*=0.8, donde 0.8 es la diferencia entre la altura promedio de una persona (1.70m aprox.) y la altura del workbench (aprox. 0.9).

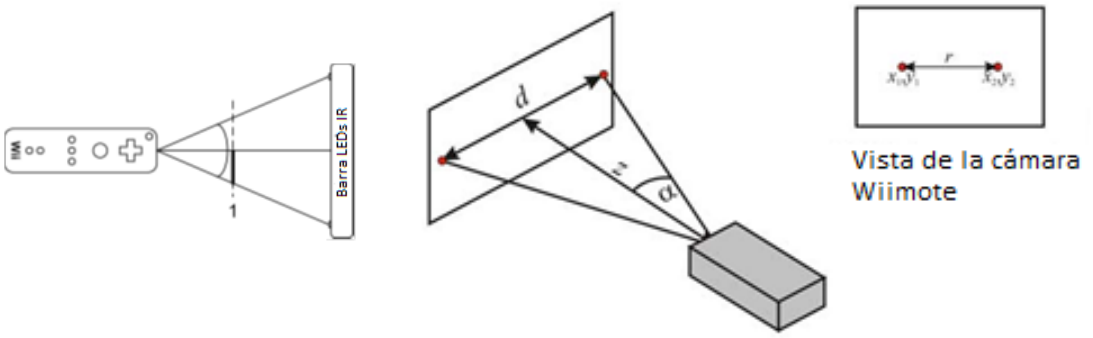


Figura 4.5 – Interpretación de la posición de los LEDs.

Para calcular la distancia del wiimote al usuario en coordenadas de wiimote, primero aproximamos el ángulo alfa (ver Figura 4.5) sabiendo que el campo de visión vertical del wiimote es distinto al campo horizontal:

$$\alpha = \frac{\left(\frac{HFOV}{1024} + \frac{VFOV}{768}\right)\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}{4}$$

donde *HFOV* y *VFOV* es el campo de visión horizontal y vertical respectivamente de la cámara del wiimote en radianes. Estos campos de visión tienen un valor angular de 41 y 31 grados respectivamente. La distancia del usuario al wiimote denotada por $eyePositionWorkbench.z$ se calcula de la siguiente forma:

$$eyePositionWorkbench.z = \frac{dotDistanceMilimeters}{2 * \tan(\alpha) * 1000}$$

donde dotDistanceMilimeters es la distancia física entre los LEDs colocados en los lentes, y medida en milímetros. Esta fórmula se obtiene de la fórmula de la tangente de alfa, y de utilizar triángulos semejantes. Se divide entre 1000 para de una vez realizar la conversión a metros.

Debido a la posición y orientación del wiimote con respecto a la mesa, su sistema de coordenadas se encuentra de tal forma que el eje *z* del wiimote viene siendo el eje *y* de la mesa, y el eje *y* del wiimote es el eje *z* de la mesa. Como se puede observar en la Figura 4.7 el wiimote se encuentra apuntando en la dirección donde se encuentra la cabeza del usuario, de manera ortogonal.

Una vez calculado la posición del ojo en coordenadas de workbench, se procede a realizar la conversión a coordenadas de escena. Las coordenadas de escena tienen el punto de origen en el centro de la pantalla (Ver Figura 4.6), lo que vendría siendo el centro de la mesa, entonces hay que desplazar las coordenadas en el eje *x*.

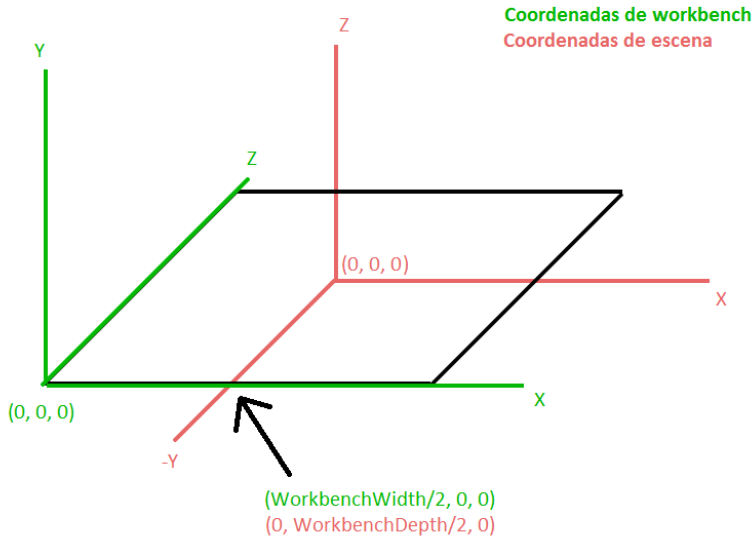


Figura 4.6 – Coordenadas de workbench y coordenadas de escena

$$positionX = eyePositionWorkbench.x - \frac{WorkbenchWidth}{2}$$

Para el eje *y* y el eje *z* se intercambian los valores ya que el sistemas de coordenadas de la escena se encuentra rotada 90 grados de tal forma que el eje *y* de la escena coincide con el eje *z* de las coordenadas de workbench, y el eje *z* de la escena coincide con el eje *y* de las coordenadas de workbench.

La posición de *z* inicial (*InicialZ*) obtenida de la calibración, se resta por la posición *z* de la coordenada de workbench y por *WorkbenchDepth*/2, siendo este resultado la posición en *y* de las coordenadas de escena.

$$positionY = InicialZ - eyePositionWorkbench.z - \frac{WorkbenchDepth}{2}$$

Para el eje *z* simplemente se asigna de forma directa la posición en *y* de la coordenada de workbench, ya que esta posee la altura que hay entre superficie de la mesa y comienzo de la escena hasta la posición de la cabeza del usuario:

$$positionZ = eyePositionWorkbench.y$$

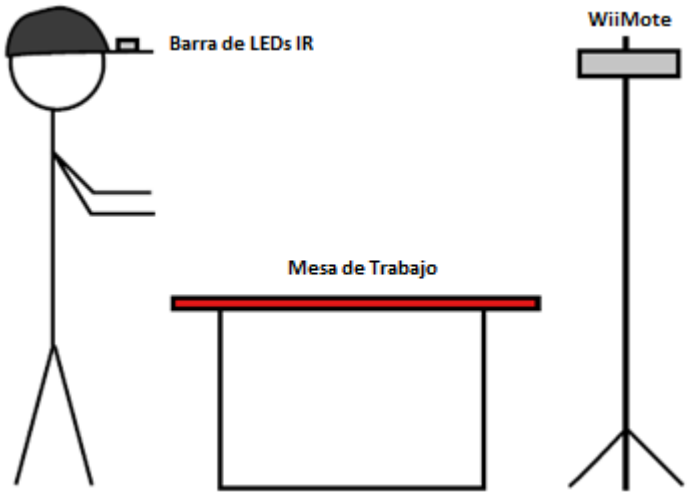


Figura 4.7 – Configuración de la mesa para el uso del headtracking [37].

4.2.1.1 Configurando la cámara

La vista estéreo off-axis requiere de volúmenes de vista asimétricos. Para estéreo en una sola pantalla, sin rastreo de cabeza, el ojo izquierdo y derecho son desplegados desplazando el vértice del frustum de visión a la izquierda o derecha por la distancia intraocular. Esto arroja dos vistas sobrepuestas de la misma escena 3D. Visto desde arriba el frustum de visión luce como en la Figura 4.8:

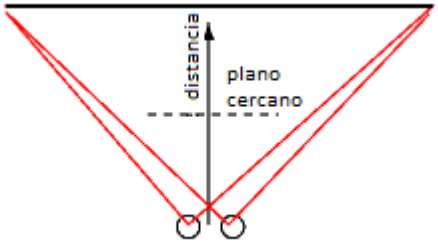


Figura 4.8 – Frustum de visión estereoscópica

El desplazamiento del ojo es negativo para el ojo izquierdo, positivo para el derecho. Si el desplazamiento es cero, da lugar a un mono-frustum de visión estándar. Adicionalmente, debido a

que los bordes del frustum están especificados en el plano de corte cercano, se necesita escalar estos valores por la proporción entre el plano cercano y la distancia de la cabeza a la pantalla, antes de pasarlos a OpenGL®.

Esto quiere decir que no se puede usar rutinas como gluPerspective para configurar la matriz de proyección de OpenGL®, sino que hay que usar glFrustum y calcular las aristas del volumen de vista explícitamente.

Los parámetros que se necesitan son el ancho de la pantalla, la distancia de la posición de la cabeza del usuario a la pantalla, el desplazamiento del ojo izquierdo y derecho, y el plano de corte cercano. Para este sencillo caso se asume que la posición de la cabeza del usuario está en el origen, o centrado, relativo a la pantalla. Los bordes horizontales del frustum de visión son la mitad del ancho de la pantalla más y menos la distancia de desplazamiento del ojo derecho e izquierdo respectivamente.

$$aristaIzquierda = -\left(\frac{ancho}{2}\right) \pm desplazamientoOjo$$

$$aristaDerecha = \left(\frac{ancho}{2}\right) \pm desplazamientoOjo$$

Donde *aristaIzquierda* y *aristaDerecha* son el límite izquierdo y derecho del frustum y *desplazamientoOjo* es la mitad de la distancia entre los ojos. Con rastreo de cabeza, el frustum de visión se convierte más asimétrico. Se necesita la posición de la cabeza como un parámetro extra. En la Figura 4.9 hay dos pares de frustum de visión diferentes para dos ubicaciones de la cabeza distinta:

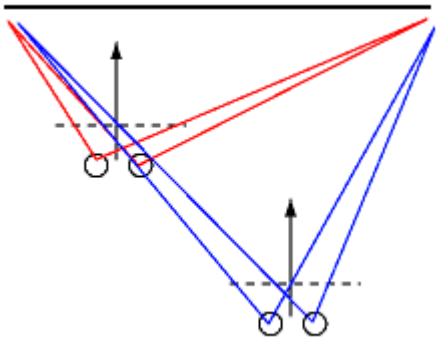


Figura 4.9 – Estereoscopia con rastreo de cabeza

Dado que la posición por defecto de la cabeza se encuentra centrada en (0,0,0), estos nuevos valores del frustum pueden ser calculados añadiendo la posición de la cabeza:

$$aristaIzquierda = \left(-\left(\frac{ancho}{2}\right) - cabeza.x\right) \pm desplazamientoOjo$$

$$aristaDerecha = \left(\left(\frac{ancho}{2}\right) - cabeza.x\right) \pm desplazamientoOjo$$

Si la cabeza no se mueve del origen, esto da el mismo resultado que el mencionado anteriormente para estereoscopia sin rastreo de cabeza. Por lo tanto se puede usar estos simples cálculos para todas las pantallas no estéreo y estéreo.

El plano de corte cercano es relativo a la distancia de la cabeza a la pantalla, así que los valores de escala para las aristas del frustum de visión variaran a medida que la cabeza se mueva adelante o atrás. Por último, sin importar lo que el usuario realmente haga, se asume que la dirección de la mirada es perpendicular a la pantalla. Esto se debe a que todos los sistemas de 3D actuales despliegan la escena para una vista perpendicular y no pueden manejar proyecciones oblicuas. Esto no es físicamente preciso.

El movimiento de la cabeza puede también cambiar la forma vertical del frustum de visión, pero los cálculos son similares que los de la arista horizontal.

Ahora entendiendo el objetivo, se construye la matriz de la siguiente forma:

Primero se calculan los límites del frustum (*top*, *bottom*, *left* y *right*), para esto se necesitan la relación de aspecto *aspectRatio*, la relación entre el plano lejano y la distancia del usuario a la pantalla *ndlf* y la distancia intraocular *mEyeSeparation*. Donde *nearPlane* es la distancia al plano cercano y *farPlane* la distancia al plano lejano.

$$aspectRatio = \frac{WorkbenchWidth}{WorkbenchDepth}$$

$$ndlf = \frac{nearPlane}{positionZ}$$

$$mEyeSeparation = 2 * positionZ * tan\left(1.5 * \frac{\frac{PI}{180}}{2}\right)$$

para calcular el *top* y *bottom* se realiza de la siguiente forma:

$$top = ndlf * (\frac{WorkbenchDepth}{2} - positionY)$$

$$bottom = ndlf * (-\frac{WorkbenchDepth}{2} - positionY)$$

y para el *left* y *right*:

$$left = \left(\left(\left(-\frac{WorkbenchDepth}{2}\right) * apectRatio - positionX\right) \pm 0.5 * mEyeSeparation\right) * ndlf$$

$$right = \left(\left(\left(\frac{WorkbenchDepth}{2}\right) * apectRatio - positionX\right) \pm 0.5 * mEyeSeparation\right) * ndlf$$

Para configurar la cámara, se deben conocer tres vectores. Estos son, la posición de la cámara, el objetivo de la cámara y el vector hacia arriba. La posición de la cámara es el vector (*positionX*, *positionY*, *positionZ*), el objetivo de la cámara es (*positionX*, *positionY*, 0) y el vector hacia arriba es (0, 1, 0). Con este vector hacia arriba, la pantalla y la escena permanecerán igual incluso si el usuario inclina su cabeza.

Además de estos tres vectores, la cámara también necesita saber los límites del mundo virtual a ser mostrado. Normalmente cuando se despliega un mundo 3D los límites no cambian, pero es necesario aquí porque se quiere fijar la pantalla a una cierta posición en el mundo virtual. Estos

límites pueden ser determinados con la distancia *positionZ* y las coordenadas *positionX*, *positionY*. La *positionX* y *positionY* determinan que tanto se debe desplazar el frustum en estas dos coordenadas, mientras que la distancia *positionZ* es el factor de escalamiento de los límites del frustum. Cuando todas estas variables se han determinado, la matriz de proyección queda de la siguiente forma Figura 4.10.

$$\begin{bmatrix} 2 * \frac{nearPlane}{right - left} & 0 & \frac{right + left}{right - left} & 0 \\ 0 & 2 * \frac{nearPlane}{top - bottom} & \frac{top + bottom}{top - bottom} & 0 \\ 0 & 0 & \frac{-farPlane + nearPlane}{farPlane - nearPlane} & \frac{-2 * farPlane * nearPlane}{farPlane - nearPlane} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Figura 4.10 – Matriz de proyección

4.2.1.2 Robustez

Un elemento importante en la aproximación para el rastreo de cabeza es que el Wiimote sea capaz de rastrear los LEDs en todo momento. Sin embargo esto no es siempre el caso, uno de los problemas es que la barra de LEDs puede ser movida fuera del campo de visión de la cámara. Esto pasa si el usuario se mueve lejos o fuera del ángulo de la cámara. La otra situación que se puede presentar es si la línea de visión es obstruida, ya sea por el usuario, otra persona o algún objeto.

En la solución de Lee [5] se considera la posición del usuario cuando ambos LEDs son visibles. Si no, la perspectiva simplemente no es cambiada y así, el contenido de la pantalla se "congela". Cuando ambos LEDs son visibles otra vez, se calcula un nuevo conjunto de matrices, basándose solamente sobre las nuevas posiciones de los LEDs. Mientras esto de hecho funciona, puede causar que la perspectiva cambie y por lo tanto el contenido de la pantalla cambia dramáticamente si el usuario se movió demasiado desde la última vez que los LEDs fueron visibles. Si hay demasiada obstrucción o los LEDs no son lo suficientemente brillantes, esto causaría un efecto de parpadeo debido al cambio repentino en la perspectiva.

Cuando el Wiimote es incapaz de detectar un LED, envía un valor por defecto. Por lo tanto es fácil detectar cuando uno o los dos LEDs están fuera de la vista. Cuando al menos un LED no es visible por el Wiimote, hay que decidir qué respuesta debe dar la aplicación al problema. Una solución posible es simplemente usar el valor como si los datos fueran correctos. Sin embargo, esto ofrece un resultado muy pobre porque hará que la pantalla se mueva una distancia muy larga instantáneamente. Otra solución es pausar la aplicación hasta que los LEDs sean detectados nuevamente. Esta solución notificará al usuario inmediatamente y hará que el usuario mismo

resuelva el problema (por ejemplo: volver al rango de visión o remover el objeto obstruyendo), pero arruinaría el flujo de la aplicación. Se eligió implementar otra opción, la cual consiste en mantener el último valor correcto detectado. Esta evita que la pantalla haga movimientos repentinos y no arruine el flujo de la aplicación. Un problema de esto es que el usuario quizá no note que los LEDs están fuera del rango de visión. Para evitarlo, se puede dar un indicador visual, por ejemplo un pequeño icono parpadeando en la esquina de la pantalla.

El segundo problema, es qué hacer cuando los LEDs vuelven a ser detectados. No importa cuál de las soluciones arribas mencionadas se elija, eso todavía seguirá siendo un problema. Si alguien se interpone por pocos segundos entre la cámara del Wiimote y el usuario, las coordenadas X y Y del usuario serán bastante similar antes y después. Sin embargo, si el usuario hace un movimiento significativo, las coordenadas detectadas cambiarán en consecuencia, produciendo un movimiento grande y repentino en la pantalla.

Esto sugiere que en vez de considerar el último valor válido para los LEDs, el anterior al mismo también sea considerado. En esta implementación se ponderan ambos valores: el último valor se pesa con 0.7 y el anterior con 0.3. Efectivamente esto resulta en una interpolación entre las dos posiciones, lo cual suaviza la transición entre la posición nueva y vieja. Incluso aunque la pantalla se comporte diferente a la vista del usuario por un momento, la ventaja de que no aparezcan saltos compensa este problema. Sin embargo, se tiene que decidir qué tan larga será esta transición.

Mientras más larga la transición, más suave el comportamiento, pero si es muy larga, la respuesta del sistema a los movimientos del usuario parecerá lenta. Alrededor de un segundo sería conveniente. En vez de usar solo algunos de los viejos valores cuando los LEDs no han sido detectados. La ventaja es que esto es una forma simple de reducir los ruidos de alta frecuencia, lo cual es discutido en la siguiente sección. La desventaja es que la capacidad de respuesta total se reduce.

4.2.1.3 Ruidos de alta frecuencia

En la solución de J. Lee el movimiento detectado por el Wiimote es usado directamente en el cálculo de la perspectiva. Sin embargo, cuando una persona se mueve de un punto a otro, no es solo el movimiento general el que se detecta, sino también un temblor (jitter) ligero debido a los pequeños movimientos hacia atrás y hacia adelante. Por ejemplo esto puede pasar si el usuario no es capaz de mantener la cabeza quieta, gira constantemente y asiente con la cabeza. Cuando los movimientos que contienen estos ruidos de alta frecuencia son mapeados directamente dentro del mundo virtual puede resultar en cambios de perspectivas incluso si se está quieto.

Se puede argumentar que las pequeñas variaciones constantes de la posición de la cabeza son normales y por lo tanto se supondría realístico el mapeo directo de estos movimientos a la

pantalla. Sin embargo, cuando se muestran en la pantalla, no parece realístico. En cambio pareciera como si la pantalla estuviese sacudiéndose lo cual hace de eso un efecto indeseable.

Para evitar este problema se necesita realizar un análisis de datos. La solución más simple es descartar todo movimiento por debajo de un umbral. Esto soluciona el problema de que la pantalla vibre cuando el usuario está quieto, pero cuando el usuario se mueve este problema aún se produciría. Otra opción es aplicar un filtro a los datos. Considerando los datos como una señal donde los datos relevantes están hechos de baja frecuencia y el ruido es hecho de las frecuencias altas, se puede realizar un filtro de paso bajo de la señal para reducir el ruido.

El filtro de paso bajo funciona dejando las frecuencias bajas pasar y reducir la amplitud de las frecuencias por encima de alguna frecuencia de corte suavizando la señal. Esto es hecho realizando una convolución[7] de la señal con algún filtro. El filtro de paso bajo más simple es el filtro de caja el cual simplemente corta las frecuencias por encima de un umbral especificado. Sin embargo esto puedo conducir a un efecto indeseado debido a la discontinuidad del filtro. Una mejor opción es usar un filtro de suavizado. Uno de los más usados es el filtro Gaussiano, el cual se define con la siguiente función

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

donde σ es la desviación estándar de la distribución. Debido a la forma de campana de la función, se logra un filtro de suavizado adecuado que con un filtro de caja.

Cuando elegimos la frecuencia de corte se busca un valor en el que todas las pequeñas variaciones sean removidas mientras que los verdaderos movimientos quedan intactos. Esto exige un equilibrio. Si la frecuencia de corte es baja, la mayoría o todo el ruido se remueve y la perspectiva permanecería muy calmada. Esto resulta en una capacidad de respuesta menor. Por otro lado, si es usada una frecuencia de corte alta, podría permitir pasar mucho ruido, haciendo el filtro inútil. Por lo tanto se debe ser cauteloso cuando se elige la frecuencia de corte.

Otro inconveniente con el método es que se necesita retrasar la señal un poco. Si el núcleo Gaussiano usado tiene un tamaño de 7, la señal reportada estaría tres muestras atrás. Esto se debe a que las tres muestras previas y las tres siguiente, deben ser consideradas cuando se filtra cualquier muestra. Si el soporte del filtro crece, puede resultar en un retraso notable desde que la barra LED se haya movido por última vez.

[7] Es un operador matemático que transforma dos funciones *f* y *g* en una tercera función que en cierto sentido representa la magnitud en la que se superponen *f* y una versión trasladada e invertida de *g*.

En este trabajo, se utiliza un filtro de caja[8], en el que se promedian las últimas N posiciones del usuario. Se hizo de esta forma por su sencillez, y se logra una fluidez aceptable en el movimiento.

4.2.2 Dispositivo de señalamiento

Para controlar el cursor utilizando el dispositivo de señalamiento se leen las posiciones dadas por el wiimote, utilizando la función *GetCursorPositionAbsolute* de la librería wiiusecpp. Estas posiciones ya vienen traducidas de tal forma que se puede hacer un mapeo a una pantalla de 1024x768; sin embargo las pantallas actuales tienen una resolución de hasta 1920x1080, por lo que hay que realizar un mapeo a la resolución actual del monitor.

Para lograr esto se calcula la relación entre la resolución actual y la resolución en la que viene la posición del cursor y este valor se usa para escalar la posición del cursor que viene del wiimote. Teniendo *xCursor* posición del cursor leído del wiimote, y la resolución *screenResX* se cálcula el resultado final mediante la fórmula:

$$CursorX = xCursor * \frac{screenResX}{1024}$$

Existe un problema en el mapeo directo que genera saltos en el movimiento. Para solucionar esto se suaviza el cambio de una posición a la otra mediante el promedio de las últimas N posiciones, logrando una aceptable fluidez en el movimiento.

Para el cálculo de la posición del cursor en el eje Y se realiza de manera similar, a diferencia de que el rango en este eje es de 0 a 767.

4.3 Implementación

Ilustración del código de la librería WiiW.

4.3.1 Función *Refresh*

Esta función es el núcleo de los cálculos para el rastreo de cabeza y la estereoscopía. Una vez inicializada y ejecutándose la aplicación, la función *Refresh* debe ser invocada cada vez que se vaya a hacer el despliegue de la escena, es decir, una vez por cada frame. Esta función es la que

[8] Un filtro de caja es el promedio de todos los elementos dentro de una región.

interpreta los datos para el cálculo de la posición del usuario relativa a la pantalla, y calcula los frustums de ambas cámaras de la estereoscopía.

En primer lugar se debe actualizar los datos del wiimote (Ver Código 1), mediante el uso de la función *RefreshWiimote*, que se encarga de actualizar los datos que el wiimote obtiene de la lectura de la cámara infrarroja, así como de los demás sensores del wiimote, el cuál es un método de la librería wiiuse.cpp.

```
wii.RefreshWiimotes();              //Refrescando la lectura de los controles
```

Código 1 – Lectura de los wiimotes.

Se comprueba que se leyeron los puntos en el condicional y son extraídos y almacenados para su posterior uso. Ver Código 2 :

```
// Comprobación de lectura de puntos IR
if (success && wiimotes.isUsingIR() && wiimotes.GetNumDots() >= 2){

      vector dots = wiimotes.GetDots();
      iterator i = dots.begin();
      int index;
      int x, y;

      // Obteniendo las coordenadas del primer punto infrarrojo
      i->GetCoordinate(x,y);
      punto1.x = x;
      punto1.y = y;
      i++;

      // Obteniendo las coordenadas del segundo punto infrarrojo
      i->GetCoordinate(x,y);
      punto2.x = x;
      punto2.y = y;

       .
       .
       .
```

Código 2 – Lectura y comprobación de los puntos.

Se verifican que los puntos leídos sean visibles comprobando que estén dentro del rango (Ver Código 3), luego el método calcularPosiciónOjo promedia estos puntos en *eye_position.x* y *eye_position.y* y calcula la distancia entre ellos almacenándola en *eye_position.z* (Ver Código 4).

```
// Solo se calcula la posición si son visibles los puntos, es decir, si se
// encuentran en el rango de resolución del wii.
if(punto1.y < 769.0f && punto2.y <769.0f){

      // Se ordenan los puntos, alamacenando el menor en punto1 y el mayor en
      // punto2
      if(punto1.x > punto2.x )
      {
            float aux;
            aux = punto2.x;
            punto2.x = punto1.x;
            punto1.x = aux;

            aux = punto2.y;
            punto2.y = punto1.y;
            punto1.y = aux;
      }

      calcularPosicionOjo();
       .
       .
       .
```

Código 3 – Validación de rango de los puntos.

```
void WiiHead::calcularPosicionOjo(){

            // Promediando los puntos leidos en X y Y
            eye_position.x = (punto1.x + punto2.x)*0.5f;
            eye_position.y = (punto1.y + punto2.y)*0.5f;

            // Cálculo de la distancia entre los puntos y almancenado en Z para su
            // posterior uso
            eye_position.z = sqrt(pow(punto1.x - punto2.x, 2)+
                                  pow(punto1.y - punto2.y, 2));
}
```

Código 4 – Pre-cálculo de la posición del ojo.

En el Código 5 se utiliza un case para en el caso que se entre en modo de calibración guardar los valores correspondientes a la esquina izquierda, derecha de la mesa y la altura en la esquina derecha de la mesa. Se delimitan los puntos obtenidos durante el refresh para que no se salgan del rango de calibración. Inicialmente el rango de calibración abarca todo el rango de la cámara infrarroja.

```
bool ultimo=false;
// Si se presionó el botón de alineación entonces se procede a capturar los puntos
// de calibración
if(AlignPress)
{
	switch(posIniContador)
	{
	// Posición de calibración del lado izquierdo de la mesa
	case 0:
		EsqIzq.x = eye_position.x;
		EsqIzq.y = eye_position.y;
		posIniContador++;
		break;
	// Posición de calibración del lado derecho de la mesa
	case 2:
		EsqDer.x = eye_position.x;
		EsqDer.y = eye_position.y;
		posIniContador++;
		break;
	// Posición de calibración con los lentes puesto estando en el lado derecho
	// de la mesa
	case 4:
		EsqDerAltura.x = eye_position.x;
		EsqDerAltura.y = eye_position.y;
		posIniContador=-1;
		AlignPress = false;
		ultimo=true;
		break;
	default:
		break;

	}
}
// En estos condicionales se verifica que los puntos leídos se encuentren dentro
// del rango de calibración si no es así se coloca el limite más cercano
if(eye_position.x < EsqIzq.x)
	eye_position.x = EsqIzq.x;
if(eye_position.x > EsqDer.x)
	eye_position.x = EsqDer.x;
if(eye_position.y < (EsqIzq.y+EsqDer.y)/2)
	eye_position.y = (EsqIzq.y+EsqDer.y)/2;
if(eye_position.y > EsqDerAltura.y)
	eye_position.y = EsqDerAltura.y;
```

Código 5 – Calibración

En el Código 6, se realiza la interpolación lineal utilizando los valores de calibración: *EsqIzq*, *EsqDer* y *EsqDerAltura* para calcular el valor en *X* y *Y* en coordenadas de workbench de la posición del usuario. HFOV y VFOV es el campo de visión horizontal y vertical respectivamente de la cámara sensora del wiimote, *alfa* es el ángulo promedio formado por la distancia entre los dos

LEDs y la cámara sensora del wiimote, *eye_position.z* en este momento es la distancia que hay entre los dos puntos leídos. Luego se procede a calcular la posición del usuario final y es almacenado en *eye_position*.

```
// Cálculo de t para la interpolación del eje X
float t = (eye_position.x - EsqIzq.x) / (EsqDer.x - EsqIzq.x);
eye_position.x = (1-t)*XFisicoMinimo + t*XFisicoMaximo;      //X

// Cálculo de t para la interpolación del eje Y
t = (eye_position.y - (EsqIzq.y+EsqDer.y)/2) /
       (EsqDerAltura.y - (EsqIzq.y+EsqDer.y)/2);
eye_position.y = (1-t)*YFisicoMinimo + t*YFisicoMaximo; //Y

// Ángulo de visión de la cámara sensora del wiimote en X:HFOV y Y:VFOV
float HFOV = 41.0f*PI/180;
float VFOV = 31.0f*PI/180;

// Promedio de los ángulos
float alfa = ((HFOV/1024.0f+VFOV/768.0f) * eye_position.z) / 4.0f;

// Cálculo la posición del ojo en el eje Z
eye_position.z = dotDistanceInMm / (2.0f*tan(alfa)) / 1000; //Z
```

Código 6 – Calculo de la posición final del ojo.

Se verifica que se haya presionado el botón de calibración, de ser así se ajusta el nuevo parámetro inicial de la distancia *eye_position.z* almacenándola en los tres puntos de calibración, y luego se procede con la conversión de coordenadas de workbench a coordenadas de escena como se explica en 4.2.1, mientras se almacena en el arreglo circular para realizar el promedio de los últimos *Neyepos* posiciones del usuario, y al final es almacenada la posición actual promediada en *eye_position*. Ver Código 7.

```
// Si se acaba de leer el último punto de calibración
if(ultimo){
	// Se guarda la posición en Z cálculada anteriormente
	EsqDerAltura.z = eye_position.z;
	MessageBox(NULL, L"Calibración realizada con éxito.", L"Calibración", MB_OK |
		MB_ICONINFORMATION);
	ultimo = false;
}

//Conversión de coordenadas de workbench a coordenadas de escena
eye_poslist[pc].x = eye_position.x - WorkbenchWidth/2;
eye_poslist[pc].y = EsqDerAltura.z - eye_position.z - WorkbenchDepth/2;
eye_poslist[pc].z = eye_position.y;

// Cálculo de la siguiente posición en el arreglo circular donde se almacenan las
// posiciones del usuario
pc = (pc+1)%Neyepos;

// Promediando las ultimas N posiciones
point promedio;
for(int i=0;i<Neyepos;i++){
	promedio += eye_poslist[i];
}
promedio /= Neyepos;

// Asignando la posición final del usuario a eye_position
eye_position = promedio;
```

Código 7 – Alineación.

Como se puede apreciar en el Código 8 se realiza los cálculos para el frustum izquierdo y derecho. El nombre de las variables empieza con mL y mR respectivamente para cada lado. *vFOV* es el campo de visión que debería tener el usuario dado la el tamaño de la pantalla y la distancia a ella. *ndlf* es la relación que existe entre el plano cercano y la distancia del usuario con respecto a la pantalla, el cual es llamado punto focal. El *aspectRatio* es la relación de tamaño entre el ancho y la altura de la superficie de la mesa.

Luego se calculan los valores del frustum izquierdo y derecho como se menciona en 4.3.

```
// Distancia de separación de los ojos
mEyeSeparation = 2*(eye_position.z)*tan(1.5*PI/180/2);

// Relación entre el plano cercano y la posición en Z
float ndlf = mNearClippingDistance / eye_position.z;

// Aspect ratio
mAspectRatio = WorkbenchWidth/WorkbenchDepth;

// Cálculo de los limites del frustum izquierdo
mLtop = ndlf*(WorkbenchDepth/2 - eye_position.y );
mLbottom = ndlf*(-WorkbenchDepth/2 - eye_position.y );
mLleft = (( -(WorkbenchDepth/2) * mAspectRatio - eye_position.x) + 0.5 *
          mEyeSeparation) * ndlf;
mLright = (( (WorkbenchDepth/2) * mAspectRatio - eye_position.x)) + 0.5 *
          mEyeSeparation) * ndlf;

// Cálculo de los limites del frustum derecho
mRtop = ndlf*(WorkbenchDepth/2 - eye_position.y );
mRbottom = ndlf*(-WorkbenchDepth/2 - eye_position.y );
mRleft = (( -(WorkbenchDepth/2) * mAspectRatio - eye_position.x) - 0.5 *
          mEyeSeparation) * ndlf;
mRright = (( (WorkbenchDepth/2) * mAspectRatio - eye_position.x) - 0.5 *
          mEyeSeparation) * ndlf;
```

Código 8 – Cálculo de los límites del frustum izquierdo y derecho.

4.3.2 Función *ApplyStereoscopy*

La función *ApplyStereoscopy* es la que se encarga de realizar el despliegue de la escena. Recibe como parámetro tres funciones que son *drawFunction*, *actualizarProyección* y *actualizarVista*. La función *drawFunction* se utiliza para el despliegue de la escena mientras que las otras dos son para actualizar la matriz vista y la matriz de proyección. Ver Código 9.

```
void WiiHead::ApplyStereoscopy(void drawFunction,
                               void actualizarProyeccion,
                               void actualizarVista){

            // Render del frustum izquierdo
            ApplyLeftFrustum(actualizarProyeccion, actualizarVista);
            glColorMask(true, false, false, false);

                  drawFunction();

            glClear(GL_DEPTH_BUFFER_BIT);

            // Render del frustum derecho
            ApplyRightFrustum(actualizarProyeccion, actualizarVista);
            glColorMask(false, true, true, false);

                  drawFunction();

            glColorMask(true, true, true, true);

}
```

Código 9 – ApplyStereoscopy.

La matriz de proyección se construye de la siguiente forma para ambas funciones de configuración de frustum. Ver Código 10.

```
      // Valores de la matriz de proyección
      float a = -(mFarClippingDistance+mNearClippingDistance) /
                (mFarClippingDistance-mNearClippingDistance);
      float b = -(2*mFarClippingDistance*mNearClippingDistance) /
                (mFarClippingDistance-mNearClippingDistance);

      float A = 0, B = 0;

      // Construcción de la matriz de proyección
      glm::mat4 projection
      (2*mNearClippingDistance/(mRright-mRleft), 0, 0, A, \
      0, 2*mNearClippingDistance/(mRtop-mRbottom),0 , B, \
      (mRright+mRleft)/(mRright-mRleft),(mRtop+mRbottom)/(mRtop-mRbottom),a,-1, \
      0,0,b,0);
```

Código 10 – Construcción de la matriz de proyección.

ApplyLeftFrustum se encarga de construir y actualizar las matrices de proyección y de vista para el frustum izquierdo. Recibe como parámetro las dos funciones usadas para actualizar dichas matrices. Ver Código 11.

```
void WiiHead::ApplyLeftFrustum(void actualizarProyeccion),
                               void actualizarVista)
{
            .
            .
            .

     //Función para actualizar la projección del usuario
     actualizarProyeccion)(projection);

     // Valores de la matriz vista
     glm::vec3 eyepos(eye_position.x - mEyeSeparation/2, eye_position.y,
                      eye_position.z);
     glm::vec3 eyedir(eye_position.x - mEyeSeparation/2, eye_position.y, 0);
     glm::vec3 up(0.0, 1.0, 0.0);

     // Construcción de la matriz vista
     glm::mat4 vista = glm::lookAt(eyepos,
                                   eyedir,
                                   up);
     // Función para actualizar la vista
     actualizarVista(vista);

};
```

Código 11 – ApplyLeftFrustum.

ApplyRightFrustum se encarga de construir y actualizar las matrices de proyección y de vista para el frustum derecho. Recibe como parámetro las dos funciones usadas para actualizar dichas matrices. Ver Código 12.

```
void WiiHead::ApplyRightFrustum(void actualizarProyeccion,
                                void actualizarVista)
{
            .
            .
            .

     //Función para actualizar la proyección del usuario
     actualizarProyeccion(projection);

     glm::vec3 eyepos(eye_position.x + mEyeSeparation/2, eye_position.y,
                      eye_position.z);
     glm::vec3 eyedir(eye_position.x + mEyeSeparation/2, eye_position.y, 0);
     glm::vec3 up(0.0, 1.0, 0.0);

     glm::mat4 vista = glm::lookAt(eyepos,
                                   eyedir,
                                   up);
     // Función para actualizar la vista
     actualizarVista(vista);

};
```

Código 12 – ApplyRightFrustum.

4.3.3 Dispositivo señalador

Para el manejo del segundo control se usa el método *GetCursorPositionAbsolute* que retorna un valor para el puntero, el cual luego se mapea a la resolución de la pantalla. Para evitar los ruidos de alta frecuencia se calcula el promedio de los últimos *Ncurpos* (actualmente con un valor de 5) posiciones del cursor. Ver Código 13.

```
// Obteniendo posición del cursor según el wiimote
wm.GetCursorPositionAbsolute( xCursor, yCursor );

// Obteniendo resolución de la pantalla
resoluciónPantalla(screenResX, screenResY);

// Realizando el escalamiento de las coordenadas del cursor del wii a las
// coordenadas de la pantalla actual
cursor_poslist[cpc].x = xCursor*(screenResX/1024.0f);
cursor_poslist[cpc].y = yCursor*(screenResY/768.0f);
cpc = (cpc+1)%Ncurpos;

// Cálculando promedio de las últimas N posiciones del cursor
point promedio;
for(int i=0;i<Ncurpos;i++){
      promedio += cursor_poslist[i];
}
promedio /= Ncurpos;

// Asignando posición final del cursor
int CursorX = (int) promedio.x;
int CursorY = (int) promedio.y;
```

Código 13 – Cursor.

4.4 Uso de la librería WiiW

Antes que nada se deben conectar los controles de Wii mediante buetooth al computador. Se crea una instancia de WiiW con los parámetros necesarios, después de haber creado la ventana para el uso de OpenGL. En este momento es cuando se toma control de los Wii Remote que están conectados vía bluetooth, con algún software de manejo de bluetooth en modo HID[9].

Se debe crear una instancia del WiiW e inicializarla con los parámetros que necesita como el tamaño de la pantalla, las dimensiones del workbench ancho, alto y profundidad, el plano cercano y plano lejano. Se invoca a la función *Start* para que la librería configure los controles. Una vez hecho esto, en la función de despliegue después de inicializar la matriz de modelo, se blanquea el buffer de color, se ejecuta la función *Refresh* y se llama a *ApplyStereoscopy* con sus respectivos parámetros, que son el método de despliegue de la escena, de actualización de las matrices de vista y proyección.

[9] Dispositivo de Interfaz Humana por sus siglas en español DIH: hace referencia a un tipo de interfaz de usuario para computadores que interactúan directamente, tomando entradas proveniente del usuario, y pueden entregar una salida a los usuarios.

1) Se crea un nuevo objeto de WiiW::Wiihead.
2) Se ejecuta el método *Start* para que se inicialicen los controles y el conjunto completo de variables del Wiihead, *AlturaPantalla* en milímetros y las medidas de la mesa en metros. En esta etapa los controles deben estar conectados por bluetooth. Ver Código 14.

```
//Se instancia el objeto de WiiHead
wiihead = new WiiHead(AlturaPantalla, AnchoMesa, AlturaMesa, ProfundidadMesa,
                      PlanoCercano, PlanoLejano);

//Inicio de conección y lectura de controles Wii
wiihead->Start();
```

Código 14 – Creación en inicialización del objeto WiiHead.

3) Una vez la aplicación esté lista para hacer el despliegue, esta debe tener las dos funciones de actualización para las matrices de proyección y vista, así como la función de despliegue de los objetos.
4) Se ejecuta el método *Refresh* del Wiihead para que se actualicen los datos de la posición del usuario, y los límites del frustum.
5) Una vez hecho el refresco se procede a hacer el despliegue de la escena usando el método *ApplyStereoscopy*, la cual recibe la función de dibujado de objetos, la de actualización de la matriz de proyección y de actualización de la matriz de vista. Ver Código 15.

```
//Actualiza la posición del usuario
wiihead->Refresh();

//Aplica estereoscopía
wiihead->ApplyStereoscopy(DrawSceneElements,actualizarProyeccion,actualizarVista);
```

Código 15 – Actualización y despliegue de la escena usando WiiHead.

4.5 Pruebas y resultados

Las pruebas se realizaron con los siguientes componentes de hardware:

- Un proyector, marca Sharp, modelo Notevision PG-M20X [41], que se encuentra fijo a una estructura que sirve como sistema de graduación para el proyector (ver Figura 4.11).

Figura 4.11 – Proyector Notevision de la mesa de trabajo.

- La mesa de trabajo en la cual se proyecta la imagen tiene las siguientes características: Ancho 1,68 metros, largo 1.30 metros, alto 0.91 metros, Peso 206Kg, modelo QVW-001, serial 02-100. Fue diseñada en la facultad de ingeniería de la Universidad Central de Venezuela (ver Figura 4.12).

Figura 4.12 – Mesa de trabajo del CCG.

- La aplicación se ejecutó en una PC con procesador Intel Core 2 Duo E6550 de 2.33Ghz, tarjeta de video Nvidia GeForce 8800gts, 4 Gb de memoria RAM, sistema operativo XP.

Los modelos 3D utilizados para la prueba final, fueron descargados de TurboSquid [42]:

- Un tablero de ajedrez
- Varias piezas de Ajedrez
- Un dado

También otros modelos utilizados durante la realización de la librería, incluía un modelo 3D de una cara, una columna, entre otros. Pero fueron reemplazados por los anteriormente mencionados para la aplicación demostrativa.

Para realizar la prueba se creó una aplicación demostrativa usando VAO[10] y VBO[11] [43] que es la forma clásica de hacer despliegue con OpenGL 3.0 y versiones más recientes. Se creó una escena sintética la cual tiene un tablero de ajedrez con algunas piezas; este tablero está ubicado en el paralaje cero que virtualmente termina quedando al nivel de la superficie de la mesa. Para usar la librería WiiW, se siguieron los pasos indicados en 4.4 para la configuración.

El Wiimote uno (que rastrea el usuario) debe estar en la parte trasera de la mesa a la altura de la cabeza del usuario, mientras el usuario usa los lentes para que el Wiimote rastree su posición. El Wiimote dos será usado como dispositivo señalador; la barra de LEDs de este control se puede ubicar en un sitio visible y no lejos, como el borde trasero de la mesa.

Se hicieron pruebas de exactitud, de tal forma que si el usuario se desplaza 10cm en el mundo real, la posición del usuario debe desplazarse 10cm en la escena. Sin embargo existe un error él cual es dependiente de la exactitud en la calibración. Debido a que la interpolación arroja un valor parametrizado (un valor a intervalo por saltos) el cual tiene incrementos constantes entre un punto y el siguiente. Entonces mientras mayor sea la distancia física de un eje con respecto a la resolución de lectura virtual de este mismo eje, mayor serán los incrementos. Por ejemplo, si se tiene que el rango de resolución virtual del eje X va de 200 a 800 pixeles (dada la calibración), este rango debe ser mapeado al ancho de la mesa que son 1680 milímetros, entonces el incremento en el movimiento generado entre un pixel y otro dadas estas medidas, sería 2,8 mm por cada pixel leído por el wiimote. Este error se ve disminuido por el suavizado generado por el filtro de caja.

[10] Vertex Array Oject (VAO) es un objeto de OpenGL que almacena todo el estado necesario para suministrar los datos de los vértices.
[11] Vertex Buffer Object (VBO) es un objeto buffer que es utilizado como la fuente de origen de los datos del arreglo de vértices.

La librería fue probada por dos usuarios, los cuales reportaron una sensación de que los objetos virtuales pertenecen al mundo real, pues mientras los usuarios se mueven alrededor de la mesa, la perspectiva cambia adecuadamente, tal y como ocurriría en la vida real. Sin embargo en ciertas ocasiones deja de percibirse el rastreo debido a la luz incandescente que entorpece la lectura de los LEDs IR por la cámara sensora del Wiimote, o cuando el usuario voltea y por este motivo se pierde la fluidez de la inmersión. El dispositivo de señalamiento tiene buen manejo de la escena.

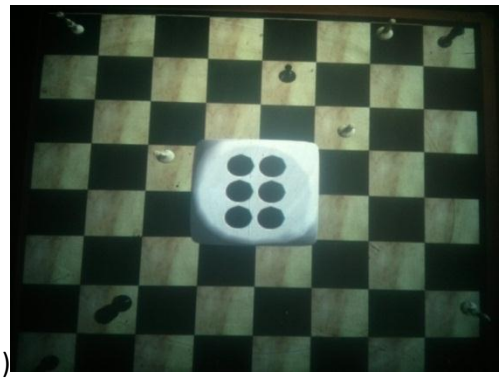
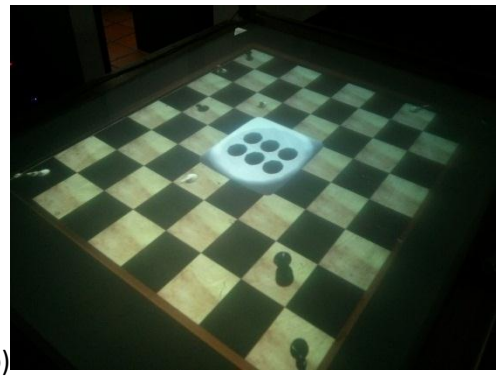
La aplicación que incluye la librería de rastreo de cabeza y estereoscopía se ejecutó a una velocidad de 150f.p.s. (*frames per second* o cuadros por segundo), en la que se pudo manipular las piezas que están sobre el tablero de ajedrez levantando las piezas y moviéndolas de un lado a otro del mismo.

Para las pruebas de sobrecarga que genera la librería sobre la aplicación, se ejecutó la aplicación sin utilizar la librería en la cual la aplicación por si sola posee una velocidad de 150 FPS. Cuando se activó la librería, la aplicación se ejecutó a una velocidad de 150FPS lo que indica que la sobrecarga que genera la aplicación es imperceptible.

Se tomaron fotos desde diferentes ángulos de visión (Figura 4.13), una desde la esquina izquierda de la mesa, otra en el centro y la última en la esquina derecha de la mesa, donde se puede notar el cambio de perspectiva. En la mesa, las piezas de ajedrez se proyectan de tal forma que la deformación que se genera en el despliegue dado el punto de vista logra el ajuste necesario para que los ojos lo interpreten como si se ubicaran por encima de la mesa.

Figura 4.13 – Resultados.

Los resultados de la perspectiva del usuario fueron satisfactorios ya que se percibe el nivel de profundidad en la escena proyectada en el sistema de realidad virtual utilizando la corrección perspectiva en base a la posición del usuario diseñado en el API. Nótese como en la Figura 4.14a la imagen muestra la proyección desde la parte de arriba del workbench mientras que la imagen 5.4b muestra la misma proyección de la imagen anterior pero fotografiada desde la posición real del usuario. Se observa que la perspectiva no es correcta. La imagen 5.4c es la proyección correcta de la escena considerando la posición del usuario frente a la mesa de proyección.

(a)  (b) 

(c) 

Figura 4.14 – Imágenes generadas en el CCGWorkbench. (a) Imagen generada a ser proyectada hacia el workbench, sin considerar la posición real del usuario. (b) Imagen anterior proyectada en el workbench y fotografiada desde el punto de vista del usuario. (c) Imagen generada y proyectada sobre la mesa, considerando la ubicación del usuario.

El uso del API para las aplicaciones es simple. Se requiere asignarle los parámetros de inicialización a la librería, y para generar cada cuadro de imagen, se debe invocar al método *Refresh* y *ApplyStereoscopy,* para actualizar los datos del rastreo y realizar el despliegue de la escena.

Capítulo 5. Conclusiones y trabajos a futuro

Acorde con los objetivos planteados, en este trabajo se implementó un sistema para rastreo de usuario a bajo costo para pantallas horizontales. El sistema utiliza el control de Wii para el rastreo del usuario, y otro control de Wii para que el usuario interactúe con la escena como un mouse virtual. Se generó una librería que integra la comunicación con los controles de Wii vía bluetooth, y el despliegue estereoscópico, considerando la posición real del usuario frente a la pantalla horizontal. Se demostró mediante una simple aplicación cómo se integra esta librería a un sistema de despliegue 3D que utilice la librería gráfica OpenGL®.

Hubieron ciertas dificultades, entre ellas conseguir una buena potencia y estabilidad en la energía de los LEDs IR para que tuviesen un alcance óptimo. A lo largo del desarrollo de la librería se encontraron varios percances que se fueron solucionando. En el desarrollo de la aplicación demostrativa se utilizaron varias librerías externas para la lectura de los objetos. Estas librerías requieren de OpenGL 3.0 o superior, por lo que hubo que adaptar el API para soportar esta versión de OpenGL®.

Como se pudo apreciar en las pruebas y resultados, que considerando la posición del usuario para el rendering se obtiene una perspectiva correcta, lo cual genera una mejor sensación de inmersión para el usuario rastreado por el sistema. El sistema de rastreo tiene la versatilidad de que es inalámbrico, preciso y económico, lo cual facilita reproducir el CCGWorkbench en otros centros de investigación.

Como trabajo a futuro consideramos la posibilidad de añadir soporte para más controles, para que de esta forma 2 o más usuarios más puedan ser rastreados por el sistema, a pesar de que únicamente se considere la ubicación de uno de los usuarios al momento de realizar el despliegue. Es deseable también probar la usabilidad del API, adaptando una alguna aplicación ya existente como por ejemplo RenderAll [2].

Referencias

[1] Román Alonso Klimova, "Visualización de Imágenes estereoscópicos en Diferentes Formatos," Universidad Central de Venezuela, Caracas, Tesis de Pregrado 2001.

[2] Rhadames Carmona, A. Madero, and O. Garcia, "RenderAll: Visualizador Estereoscópico de Superficies y Volúmenes," in *Proceedings del VII Congreso Internacional de Métodos Numéricos en Ingeniería y Ciencias Aplicadas*, 2004, pp. TC27-TC34.

[3] Inition. (2014, Febrero) Polhemus Isotrak II from Inition. [Online]. http://inition.co.uk/3D-Technologies/polhemus-isotrak-ii

[4] Jorge Enrique Valbuena Garzón, "Localizador de Usuario para la Mesa de Trabajo Virtual utilizando Captura de Video vía Webcam," Universidad Central de Venezuela, Caracas, Tésis de Pregrado 2007.

[5] Johnny Chung Lee. (2011, Diciembre) Johnny Chung Lee - Projects. [Online]. http://johnnylee.net/projects/wii

[6] Nintendo. (2012, Oct.) Wii Official Site at Nintendo. [Online]. http://www.nintendo.com/wii

[7] Michael Laforest. (2011, Diciembre) GitHub. [Online]. https://github.com/paulburton/wiiuse

[8] Morton Heilig. (2012, Mayo) THE FATHER OF VIRTUAL REALITY. [Online]. http://www.mortonheilig.com/

[9] Ivan E. Sutherland, "A head-mounted three dimensional display," in *American Federation of Information Processing Societies*, Salt Lake, 1968, pp. 757-764.

[10] Jaron Lanier. (2012, Marzo) Jaron Lanier's Homepage. [Online]. http://www.jaronlanier.com/

[11] Thomas P. Caudell and D.W. Mizell, "Augmented reality: an application of heads-up display technology to manual manufacturing processes," in *Proceedings of the TwentyFifth Hawaii International Conference on System Sciences*, vol. 2, Seattle, Marzo 1992, pp. 659-669.

[12] Wikipedia.org. (2012, Mayo) Stereograph as an educator - Wikipedia, the free encyclopedia. [Online]. http://en.wikipedia.org/wiki/File:Stereograph_as_an_educator_-_anaglyph.jpg

[13] Frederick P., Jr. Brooks, "What's Real About Virtual Reality?," in *IEEE Computer Graphics and Applications*, Chapel Hill, 1999, pp. 16-27.

[14] Dirk Donath and Holger Regenbrecht, "Using Immersive Virtual Reality Systems for Spatial Design in architecture.," in *Proceedings of the 2nd AVOCAAD Conference*, Brussels, 1999.

[15] C. Cruz-Neira, D. Sandin, T. DeFanti, R. Kenyon, and J. Hart, "The CAVE®: Audio Visual Experience Automatic Virtual Environment," in *Communications of the ACM*, vol. 35, Illinois, 1992, pp. 65-72.

[16] SABIA. (2012, Marzo) S.A.B.I.A. Sistemas Adaptativos y Bioinspirados en Inteligencia Artificial. [Online]. http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/Realidad%20Virtual/web/dispositivos.html

[17] Sharif Razzaque, David Swapp, Melt Slater, Mary C. Whiton, and Anthony Steed, "Redirected Walking in Place," in *EGVE '02 Proceedings of the workshop on Virtual environments 2002*, London, UK, 2002, pp. 123-130.

[18] Gareth Branwyn. (2012, Mayo) VPL Research, Inc. [Online]. http://www.streettech.com/bcp/BCPgraf/StreetTech/VPL.html

[19] CyberGlove Systems LLC. (2012, Mayo) Cyber Glove Systems. [Online]. http://www.cyberglovesystems.com

[20] Samsung. (2012, Marzo) SSG-2100AB - Información general. [Online]. http://www.samsung.com/latin/consumer/tv-audio-video/television/tv-accessories/SSG-2100AB/ZD?subsubtype=3d-glasses

[21] Cristina Martín Doñate, "INTERFACES HÁPTICOS. APLICACION EN ENTORNOS," Universidad de Jaén, España, Reporte Técnico 2004.

[22] Ronald Azuma, "A Survey of Augmented Reality," in *Teleoperators and Virtual Environments 6*, Malibu, 1997, pp. 355-385.

[23] Paul Milgram, "A taxonomy of mixed realiy visual displays," in *IEICE Transactions on Information Systems*, vol. E77-D, Ontario, 1994.

[24] Mohammad Mokarom Hossain, "Tracking apper notes on a distributed Physical-Virtual bulletin board," Umeå University, Umeå, Masters Thesis 2004.

[25] Juan de Urraza, "La Realidad Aumentada," Universidad Católica “Nuestra Señora de la Asunción”, Asunción - Paraguay, Reporte Informativo 2009.

[26] Emilio García Roselló, "Reutilización de COTS," Universidad de Vigo, Vigo, Tesis Doctoral 2009.

[27] ISE Lab. (2012, Marzo) ISE Lab, Centre de Visió per Computador. [Online]. http://www.cvc.uab.es/~poal/hse/hse.htm

[28] Department of Orthopaedics. (2012, Marzo) Department of Orthopaedics : Brown Alpert Medical School : Research. [Online]. http://biomed.brown.edu/Medicine_Departments/ORTHOPAEDICS/research/BioEng/research_bio-research-invivo.php

[29] Gregory Baratoff and Scott Blanksteen. (2014, Febrero) Human Interface Technology Laboratory. [Online]. http://www.hitl.washington.edu/scivw/EVE/I.D.1.b.TrackingDevices.html

[30] CNET. (2014, Febrero) Wii Motion Plus Review - cnet. [Online]. http://reviews.cnet.com/game-accessories/wii-motionplus/4505-10110_7-33690777.html

[31] Amazon. (2012, Junio) The Internet Movie Database (IMDb). [Online]. http://www.imdb.com/title/tt0181689/

[32] Microsoft. (2014, Febrero) Kinect for Xbox 360 -Xbox.com. [Online]. http://www.xbox.com/en-US/xbox360/accessories/kinect/KinectForXbox360

[33] NaturalPoint. (2012, Nov.) TrackIR. [Online]. http://www.naturalpoint.com/trackir/products/trackir5/

[34] Jannick Rolland Jannick Rolland, "Head-Mounted Display Systems," University of Arizona, Arizona, Tesis.

[35] How stuff works. (2012, Mayo) How Virtual Reality Gear Works. [Online]. http://electronics.howstuffworks.com/gadgets/other-gadgets/VR-gear3.htm

[36] Centro de Computación Gráfica. (2012, Mayo) Centro de Computación Gráfica, Universidad Central de Venezuela. [Online]. http://ccg.ciens.ucv.ve/

[37] Harold Bruintjes, "Perspective corrected view on (touch-)table surfaces," University of Twente, Reporte Técnico 2009-2011.

[38] Sebastian Schneider. (2013, November) [noeol]. [Online]. http://www.noeol.de/s3d/vertical_parallax.html

[39] Barco. (2012, Marzo) Barco | Simulation & virtual reality | L-shaped 3D projection table with 2 orthogonal projection surfaces | TAN Holobench™. [Online]. http://www.barco.com/en/simulation_virtualreality/product/961

[40] Build Wii For Fun. (2012, Mayo) Build Wii. [Online]. http://www.buildwii.com/theguide

[41] Projector Central. (2014, Abril) Projector Central. [Online]. http://www.projectorcentral.com/Sharp-Notevision_PG-M20X.htm

[42] TurboSquid. (2014, Mayo) 3D Models for Professionals : TurboSquid. [Online]. http://www.turbosquid.com/

[43] OpenGL®. (2014, Mayo) Vertex Specification - OpenGL.org. [Online]. https://www.opengl.org/wiki/Vertex_Specification#Vertex_Array_Object

[44] Paul Borke. (2013, November) Paul Borke. [Online]. http://paulbourke.net/exhibition/vpac/theory.html

Anexos

Librerias de Windows para el Wiimote:

GlovePIE

Librería que se puede conectar con controles de varias consolas, para uso general, sin una documentación que lo soporte.

GlovePIE son las siglas de GloveProgrammable Input Emulator. Aunque no tiene nada que ver con Guantes de Realidad Virtual, fue originalmente concebido como un sistema para emular entradas de Joysticks y Teclados usando los Guantes de Realidad Esencial P5. Ahora soporta emular cualquier clase de entrada, usando cualquier clase de dispositivo, incluyendo Polhemus, Intersense, Ascension, WorldViz, 5DT, y productos eMagin . Tambiencontrolasalidas MIDI y OSC.

WiiYourself

Está diseñada para su uso en C++; soporta varios Wiimotes; también soporta nunchuks, periféricos Guitar Hero y Balance Board, Motion+ y Controles Clásicos, estimación de la orientación, manipulación de los LEDs y vibración. Aunque los altavoces están en fase experimiental, tiene soporte para sondeo (pollings) y callbacks. Detecta desconexiones y pérdidas de paquetes. Usa hilos para realizar multitarea y salida de depuración extensa. La única limitación es que solo fue desarrollado para correr en Windows.

WiiuseCpp

WiiuseCpp es una API de C++ construida sobre Wiiuse [7]. WiiuseCpp transforma wiiuse a paradigma orientado a objeto y provee funciones para interactuar con las funcionalidades de las estructuras del Wiiuse.

Wiiuse es una librería escrita en C que se puede conectar con varios controles de Nintendo Wii. Soporta detección de movimiento, rastreo Infrarrojo (IR), nunchuk, control clásico, y los controles de Guitar Hero 3 (un juego de género musical). Usa un solo hilo o subproceso, no usa bloqueos lo que hace que sea una API limpia y ligera.

Librería Wiiuse C

A continuación se demuestra una forma para utilizar la librería:

Se inicializa un arreglo de objetos wiimote, donde MAX_WIIMOTES es el número máximo de wiimotes que se quiere crear y conectar como se muestra en el Código 1.

```
wiimote** wiimotes;

wiimotes =  wiiuse_init(MAX_WIIMOTES);
```

Código 16 – Creación del arreglo de wiimotes.

Ahora necesitamos encontrar los wiimotes disponibles, le damos a la función *wiiuse_find* el arreglo que creamos anteriormente. Adicionalmente, indicamos la cantidad máxima de wiimotes en los que estamos interesados en encontrar (MAX_WIIMOTES) y un tiempo de espera de 5 segundos. Esto retorna la cantidad de wimotes que están en modo visible[12] (Código 17).

```
int found = wiiuse_find(wiimotes, MAX_WIIMOTES, 5);
if (!found) {
    printf ("No wiimotes found.");
    return 0;
}
```

Código 17 – Consulta de wiimotes disponibles.

Una vez encontrados los wiimotes, se conecta a ellos con la función *wiiuse_connect* pasándole el arreglo de wiimotes y MAX_WIIMOTE, obteniendo como resultado un entero que indica la cantidad de conexiones establecidas con los wiimotes como se muestra en el Código 18.

```
int connected = wiiuse_connect(wiimotes, MAX_WIIMOTES);
if (connected)
    printf("Connected to %i wiimotes (of %i found).\n", connected, found);
else {
    printf("Failed to connect to any wiimote.\n");
    return 0;
}
```

Código 18 – Conexión a los wiimotes disponibles.

Una vez conectado se puede empezar a realizar un *feedback* con el control, indicándole al usuario que está listo para su uso. Se usa *wiiuse_set_leds* con el primer parámetro que indica el

[12] Cuando un dispositivo Bluetooth está en "modo visible", otros dispositivos Bluetooth pueden detectarlo y enlazarse o conectarse a él.

control al cual va dirigido la manipulación de los leds, seguido del led que se quiere encender o apagar. Con *wiiuse_rumble* se prende y apaga la vibración por un corto período de tiempo como se puede apreciar en el Código 19.

```
int k = 0;
for (; k<MAX_WIIMOTES; k++){
wiiuse_set_leds(wiimotes[0], WIIMOTE_LED_1);
wiiuse_set_leds(wiimotes[0], WIIMOTE_LED_2);
wiiuse_set_leds(wiimotes[0], WIIMOTE_LED_3);
wiiuse_set_leds(wiimotes[0], WIIMOTE_LED_4);
}
wiiuse_set_leds(wiimotes[0], WIIMOTE_LED_1);
wiiuse_set_leds(wiimotes[1], WIIMOTE_LED_2);
wiiuse_set_leds(wiimotes[2], WIIMOTE_LED_3);
wiiuse_set_leds(wiimotes[3], WIIMOTE_LED_4);
wiiuse_rumble(wiimotes[0], 1);
wiiuse_rumble(wiimotes[1], 1);

#ifndef WIN32
    usleep(200000);
#else
    Sleep(200);
#endif

wiiuse_rumble(wiimotes[0], 0);
wiiuse_rumble(wiimotes[1], 0);
```

Código 19 – Feedback con el/los controles.

Para saber si existe un evento de cualquier tipo se usa *wiiuse_poll* que chequea las banderas “flags” para saber si se presentó cualquier tipo de evento. Retorna 0 si no hay eventos; en caso contrario se revisa para cada control su estado de eventos, manejándolos por tipos como se indica en el Código 20.

```
while (1) {
    if (wiiuse_poll(wiimotes, MAX_WIIMOTES)) {
        /*
         *  This happens if something happened on any wiimote.
         *  So go through each one and check if anything happened.
         */
        int i;
        for (i = 0; i < MAX_WIIMOTES; i++) {
            switch (wiimotes[i]->event) {
                case WIIUSE_EVENT:
                    /* a generic event occured */
                    handle_event(wiimotes[i]);
                    break;

                case WIIUSE_STATUS:
                    /* a status event occured */
                    handle_ctrl_status(wiimotes[i]);
                    break;

                case WIIUSE_DISCONNECT:
                case WIIUSE_UNEXPECTED_DISCONNECT:
                    /* the wiimote disconnected */
                    handle_disconnect(wiimotes[i]);
                    break;
```

Código 20 – Ciclo para el *poll* y manejo de eventos.

En el método *handle_event* se encapsula el manejo con lo referente a los eventos de los puntos infrarrojos obteniendo los datos del cursor, la distancia Z a la que se encuentra el wiimote de la barra de LEDs infrarrojo y la posición de cada punto infrarrojo por individual según podemos apreciar en el Código 21.

```
/*
 *  If IR tracking is enabled then print the coordinates
 *  on the virtual screen that the wiimote is pointing to.
 *
 *  Also make sure that we see at least 1 dot.
 */
if (WIIUSE_USING_IR(wm)) {
    int i = 0;

    /* go through each of the 4 possible IR sources */
    for (; i < 4; ++i) {
        /* check if the source is visible */
        if (wm->ir.dot[i].visible)
            printf("IR source %i: (%u, %u)\n", i, wm->ir.dot[i].x, wm->ir.dot[i].y);
    }

    printf("IR cursor: (%u, %u)\n", wm->ir.x, wm->ir.y);
    printf("IR z distance: %f\n", wm->ir.z);
}
```

Código 21 – Forma de obtener los datos infrarrojos.

Con esto se demuestra de forma básica el manejo de los puntos de interés para este seminario, como lo es la obtención de los datos infrarrojos. Nótese que se puede controlar dentro del ciclo donde se realiza la revisión de los eventos. Es posible agregar un intervalo de tiempo para controlar la revisión o chequeo de los eventos, evitando la saturación de solicitudes y permitiendo recursos para otras tareas.

Documentación

Los archivos que conforman la librería son:

src/classic.c	Classiccontrollerexpansiondevice
src/classic.h	Classiccontrollerexpansiondevice
src/definitions.h	General definitions
src/dynamics.c	Handles the dynamics of the wiimote
src/dynamics.h	Handles the dynamics of the wiimote
src/events.c	**Handleswiimoteevents**
src/events.h	**Handleswiimoteevents**
src/guitar_hero_3.c	Guitar Hero 3 expansiondevice
src/guitar_hero_3.h	Guitar Hero 3 expansiondevice
src/io.c	Handlesdevice I/O (non-OS specific)
src/io.h	Handlesdevice I/O
src/io_nix.c	Handles device I/O for *nix
src/io_win.c	Handles device I/O for Windows
src/ir.c	**Handles IR data**
src/ir.h	**Handles IR data**
src/nunchuk.c	Nunchukexpansiondevice
src/nunchuk.h	Nunchukexpansiondevice
src/os.h	Operatingsystemrelateddefinitions
src/wiiuse.c	General wiimoteoperations
src/wiiuse.h	**API header file**
src/wiiuse_internal.h	General internalwiiusestuff

Tabla 2 - Lista de todos los archivos documentados de la librería Wiiuse

Diagrama de colaboración:

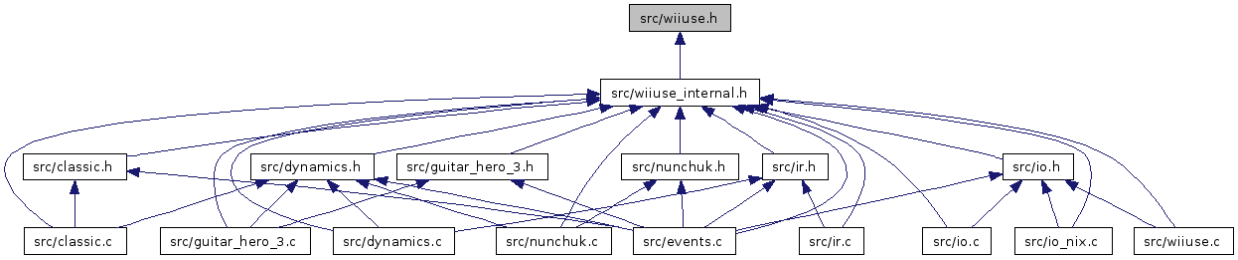


Figura 0.1 – Diagrama de colaboración de la librería

Librería externa que usa:

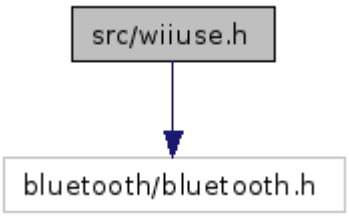


Figura 0.2 – Librería externa que usa wiiuse.

Estructuras de datos que la librería posee:

accel_t	Estructura del Acelerometro
classic_ctrl_t	Dispositivo de expansión del control clasico
expansion_t	Dispositivo de Expansiongenérico conectado al Wiimote
gforce_t	Estructura para la fuerza de Gravedad
guitar_hero_3_t	Dispositivo de Expansión Guitar Hero 3
ir_dot_t	**Estructura para una sola fuente IR**
ir_t	**Estructura IR. Contiene toda los datos relacionados con el rastreo IR**
joystick_t	Estructura de calibración del Joystick
nunchuk_t	Dispositivo de Expansión “Nunchuk”
orient_t	Estructura de orientación
read_req_t	Estructura de solicitud de lectura de datos
vec2b_t	Vector de bytes x,y sin signo
vec3b_t	Vector de bytes x,y,z sin signo
vec3f_t	Vector de Reales x,y,z con signo
wiimote_state_t	Datos importantes de los eventos previos
wiimote_t	Estructura del Wiimote

Tabla 3 - Lista de Estructuras de la librería Wiiuse

La estructura **ir_t** es la encargada de almacenar los datos relacionados con el rastreo IR por lo que se muestra con un poco más de detalle sus dependencias y datos que maneja:

1. Diagrama de colaboración:

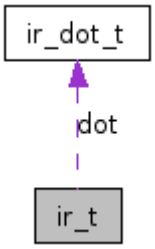


Figura 0.3 – Diagrama de colaboración de la estructura ir_t

2. Campos de dato de la estructura ir_t:

structir_dot_t	dot [4]	Puntos IR
byte	num_dots	Numeros de puntos en este momento
enumaspect_t	aspect	Aspect Ratio(Relación de Aspecto) de la pantalla
enumir_position_t	pos	Posición de la barra de sensores IR
unsignedint	vres [2]	Resolución virtual de la pantalla IR
int	offset [2]	Desplazamiento de corrección XY del IR
int	state	Mantiene el estado del Rastreo IR
int	ax	Coordenada X absoluta
int	ay	Coordenada Y absoluta
int	x	Coordenada X calculada
int	y	Coordenadas Y calculada
float	distance	Distancia entre pixeles entre los primeros dos puntos IR

float	z	Distancia calculada

Tabla 4 – Campos de datos de la estructura ir_dot

La estructura **ir_dot_t**, es la estructura base de almacenamiento para una fuente de datos IR, en la cual se muestran en la Tabla 5 los datos que engloba:

byte	visible	Visibilidad del punto IR
unsignedint	x	Coordenada interpolada X
unsignedint	y	Coordenada interpolada Y
short	rx	Coordenada X en “crudo” (0-1023)
short	ry	Coordenada Y en “crudo” (0-767)
byte	order	Orden creciente del valor del Eje X
byte	size	Tamaño del punto IR (0-15)

Tabla 5 – Campos de datos de la Estructura ir_dot_t