UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

CENTRO DE INFORMÁTICA INDUSTRIAL

Módulo de visualización estereoscópica para la biblioteca SceneToolKit

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

Autor: **Yoana Rosa Rios Díaz**

Tutor: **MSc. Liudmila Pupo Peña**

Co-tutor: **Ing. Yasmany Alfonso Monteagudo**

La Habana, Junio de 2011

DECLARACIÓN JURADA DE AUTORÍA

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

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

_____

Firma del autor Yoana Rosa Rios Díaz

_____

Firma del Tutor MSc. Liudmila Pupo Peña

_____

Firma del Co-Tutor Ing. Yasmany Alfonso Monteagudo

Datos de contacto

Nombre y Apellidos: Liudmila Pupo Peña

Edad: 26 años

Ciudadanía: cubana

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: MSc en Informática Aplicada

Categoría Docente: Instructor

E-mail: lpupo@uci.cu

Graduado de Ingenería en Ciencias Informáticas en la UCI en el año 2007, con 6 años de experiencia en los temas de Gráficos por Computadora y Realidad Virtual. Profesora del Dpto. de Visualización y Realidad Virtual.

Quisiera agradecer:

A mis padres por toda su ayuda y apoyo, por haber confiado en mí y ser mi guía, a Josue por ser mi crítico más fuerte, a Ale por su paciencia y por estar siempre cuando lo necesito. A mi otra familia por haberme acogido y hacerme sentir parte de ella, Marlen, Jose, Laury y Aida.

A mis tutores por su constancia y ayuda. En especial a Liudmila gracias a su impulso este trabajo se logró terminar satisfactoriamente, gracias por permitirme realizar este interesante trabajo y por no cansarse de responder a mis dudas. A Yasmany por siempre estar disponible para aclararme dudas de la STK y ayudarme a lograr este resultado, a Alfredo que también puso sus granitos de arena en la realización de este trabajo. A Osvaldo y el equipo de VISMEDIC gracias por confiar en mi trabajo y darme la oportunidad de ver el módulo integrado con la herramienta VISMEDIC, uno de los productos más importantes de la facultad.

A mis amigas por haber aumentado mi familia considerablemente, por estos 5 años llenos de momentos inolvidables, a Libeidy por haberme llevado a Punto de Control, a Liudmila y el viaje inolvidable a Bayamo. A Liusba que aunque nunca me llevaste a Borboyon, pase muy buenos momentos contigo. A Dayanis y Sandra, a los Tutis, Adriel, Yaniel, Yurisel, JC, Andrés, Adruban.

A mis gordos por su cariño incondicional, por haberme apoyado en cada decisión y guiado durante todos estos años, gracias a ellos es el fruto de este trabajo.

A mi hermano por todos sus consejos y opiniones, por su ayuda en todo momento.

A mi novio por ser mi mejor amigo, compañero y mucho más, por haberme aguantado todos estos años, por calmarme cada vez que perdía la paciencia.

Resumen

En la actualidad la gran mayoría de los motores gráficos usados en la creación de sistemas de Realidad Virtual (RV) ya sean juegos, simuladores, etc. permiten mostrar los resultados de forma monocular, es decir se proyecta la escena desde un solo punto de vista, con la ausencia de la principal señal de profundidad: la Visión Estereoscópica (VE). En aras de ganar interacción e inmersión en los sistemas de RV se opta por incorporar la visualización estereoscópica de los contenidos, de este modo se dibujan los elementos que componen la escena desde dos proyecciones cónicas ligeramente diferentes, las cuales son procesadas por el cerebro y se puede apreciar la profundidad real, la escena debe ser observada con algún dispositivo de visualización para percibirla de manera estereoscópica.

El objetivo de este trabajo es incorporar un módulo que brinde funcionalidades estereoscópicas a la herramienta SceneToolKit (STK) creada en la universidad, y la configuración de los elementos geométricos y de las técnicas de visualización.

Se ha tenido como resultado un módulo estereoscópico que permite la visualización de los contenidos en seis de las técnicas estereoscópicas más importantes: anaglifo, visión paralela, visión cruzada, entrelazado, *alterned image*, *quad buffer*. Se permite además la configuración de los elementos geométricos de la escena. Al incorporar la posibilidad de visualización estereoscópica de los contenidos el módulo le brinda mayor realismo e interacción a la STK. Como valor añadido, el módulo se ha acoplado a la herramienta VISMEDIC, elaborada en el Centro de Informática Industrial (CEDIN) para la visualización de imágenes médicas.

Palabras clave: *SceneToolKit*, señal de profundidad, técnicas estereoscópicas, visualización estereoscópica

Índice general

Índice de figuras

Índice de tablas

Introducción

La RV es un sistema interactivo que permite sintetizar un mundo tridimensional ficticio, a través de gráficos 3D. Básicamente consiste en simular la mayor cantidad de percepciones de una persona: la vista, sonido, tacto e incluso sensaciones de aceleración o movimiento. Todas estas sensaciones diferentes deben ser presentadas al usuario de forma que se sienta inmerso en el entorno virtual generado por la computadora. La RV contiene ramas muy variadas de trabajo dentro de las que se encuentran: la visualización científica (física, médica, etc.), entretenimiento, apreciación de espacios arquitectónicos, diseño asistido por computadora *Diseño asistido por computadoras (traducido de las siglas en inglés*: Computer-aided design*)* (CAD), etc. [Mon, 2010].

En las aplicaciones gráficas tradicionales, ambos ojos observan la misma imagen en pantalla producida a través de una única proyección cónica, conocida también como visión monocular, aunque esta representación puede contener la mayoría de las señales de profundidad, está ausente la más importante de ellas: la VE. Por su parte la VE proporciona una mayor amplitud de campo visual, ya que se utilizan dos proyecciones ligeramente diferentes de una imagen y genera un sistema más preciso y con mejor calidad perceptual en la evaluación de las distancias [Marqués, 2004].

El proceso de creación de las aplicaciones gráficas se realiza a través de motores gráficos que brindan funcionalidades que evitan la dificultad de la utilización de bibliotecas gráficas de bajo nivel como *OpenGL* y *Direct3D*. En la actualidad se trata de incorporar la VE a los motores gráficos para aprovechar sus ventajas, básicamente se trata de representar la escena virtual de forma que el cerebro fusione las imágenes de ambos ojos logrando una imagen 3D, tal como lo

hace en el mundo real.

En Cuba la UCI es la institución especializada en la investigación y desarrollo de la rama de la RV, el CEDIN cuenta con varios proyectos que trabajan en esta línea de investigación. Se ha concebido para esto un motor gráfico denominado STK, que permite el rápido desarrollo de aplicaciones finales.

La STK en su versión actual no permite la visualización de los contenidos de forma estereoscópica. Esta limitación dificulta dos elementos importantes: la inmersión de los usuarios en los entornos simulados y la interacción con los objetos a partir de una mejor percepción de profundidad. Es por ello que se evidencia la necesidad de contar con una solución que posibilite la representación estereoscópica de los contenidos.

Se plantea entonces el siguiente **problema de investigación**: ¿Cómo brindar la visualización estereoscópica de los contenidos en la herramienta STK?

El **objeto de estudio** de este trabajo se centra en el proceso de visualización estereoscópica, con el **objetivo general** de desarrollar un módulo que permita brindar la visualización estereoscópica de los contenidos a la STK.

Se establece como **campo de acción** las técnicas usadas para incorporar visualización estereoscópica en los motores gráficos.

El grupo de tareas de investigación definidas para dar cumplimiento al objetivo planteado son las siguientes:

- Análisis de las señales de profundidad, especialmente la de visión estereoscópica, para identificar los elementos a tener en cuenta en su simulación por computadora.
- Investigación de las características y fundamentos de las técnicas de visualización estereoscópica para conocer cómo se produce este efecto.
- Análisis de la arquitectura y funcionalidades del motor gráfico STK para conocer las características a tener en cuenta al incorporarle el módulo.
- Adaptación y desarrollo soluciones técnicas para dar solución al problema científico.

- Diseño e implementación de un entorno virtual de prueba que sirva para mostrar el cumplimiento del objetivo planteado.

Con la realización de estas tareas se pretende brindar a los desarrolladores de productos virtuales del CEDIN de la UCI un sistema que incorporaría visualización estereoscópica de los contenidos a través de diferentes técnicas. Lo cual brindaría más realismo al producto final, ya que se podría percibir con mayor claridad la distancia entre los objetos.

Estructura del documento

A continuación se hace una breve descripción de la estructura del trabajo el cual se ha dividido en cuatro capítulos.

En el capítulo 1 "**Fundamentación Teórica**"se presentan las características de la visión estereoscópica, las técnicas y fundamentos matemáticos utilizados para lograrla, así como las principales características y funcionalidades de la STK para cumplir el objetivo.

En el capítulo 2 "**Soluciones Técnicas**"se describen las características técnicas que fueron seleccionadas y que darán respuesta al problema de investigación planteado.

En el capítulo 3 "**Ingeniería del sistema**"se realiza el levantamiento de requisitos funcionales, se modelan las características del diseño e implementación del sistema a través de los diagramas correspondientes.

En el capítulo 4 "**Valoración de resultados**"se valora el cumplimiento del objetivo, mostrando una aplicación que visualiza las diferentes técnicas implementadas, y además se realizan mediciones para comprobar la precisión del sistema al proyectar las imágenes estereoscópicas.

Finalmente se presentan las conclusiones y recomendaciones obtenidas como resultado de la investigación.

Capítulo 1

Fundamentación Teórica

En este capítulo se presenta la fundamentación teórica vinculada a la visión estereoscópica, como se produce este efecto, además se incluye el estado del arte referente a este tema y las características de la biblioteca para la cual se realiza este trabajo.

1.1. Señales de profundidad

Vivimos en un mundo tridimensional, pero nuestra percepción del mundo se basa en las imágenes planas que se proyectan en nuestras retinas, es por eso que se deben buscar señales que permitan percibir la profundidad. La percepción de profundidad es la capacidad de ver en un espacio tridimensional y de calcular bien distancias. Sin la percepción de profundidad sería difícil conducir un automóvil, andar en bicicleta, atrapar pelotas o simplemente andar por una habitación [Coon, 2005].

Varias señales se combinan para obtener la percepción de tridimensionalidad y se clasifican en fisiológicas y no fisiológicas, y estas a su vez pueden ser monoculares o binoculares. Las señales monoculares se refieren a las señales que se producen cuando se observa la escena desde un solo punto de vista. Las binoculares se perciben cuando se observa la escena desde dos puntos de vista. Las señales fisiológicas son las que provienen del cuerpo y las no fisiológicas se generan en el ambiente. Estas últimas son completamente monoculares, mientras que las señales fisiológicas

pueden ser monoculares o binoculares.

1.1.1. Señales de profundidad no fisiológicas

Las señales de profundidad no fisiológicas y monoculares son la base de la percepción de profundidad en las proyecciones visuales 2D, son usadas para engañar al cerebro y simular aspectos como profundidad y volumen que no pueden ser vistos en fotos y pinturas, entre ellas se encuentran: luces y sombras, solapamiento, tamaño relativo y perspectiva, estas dos últimas muy importantes a la hora de crear escenas estereoscópicas, ya que las imágenes que cuenten con fuertes señales de profundidad monoculares son más reales y fáciles de visualizar cuando se le añaden las señales binoculares [Lipton, 1997a].

A continuación se presenta una breve explicación de algunas de las señales de profundidad mencionadas anteriormente:

Luz y Sombra: La utilización de luz y sombra realza el volumen de los objetos de la escena, los hace lucir más sólidos, como se puede apreciar en la figura 1.1. Además los objetos con brillo o colores brillosos parecen estar más cercanos que los opacos.

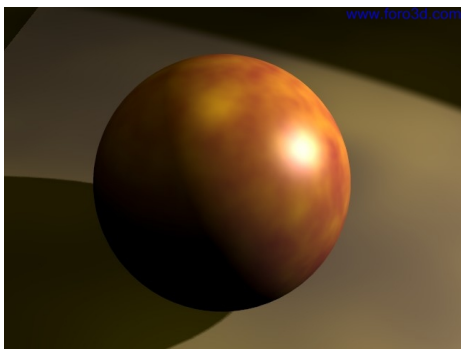

Figura 1.1: Efecto de la luz y la sombra sobre los objetos

Tamaño Relativo: Se tiene en cuenta el tamaño del objeto proyectado dentro de la retina del ojo, los objetos más cercanos parecen más grandes y los alejados lucen más pequeños, la mente ayuda en este sentido, dado que los objetos que se encuentren lejanos en la escena se interpreta como que se encuentran a gran distancia. En la figura 1.2 se puede apreciar el efecto que producen objetos similares con distintos tamaños.

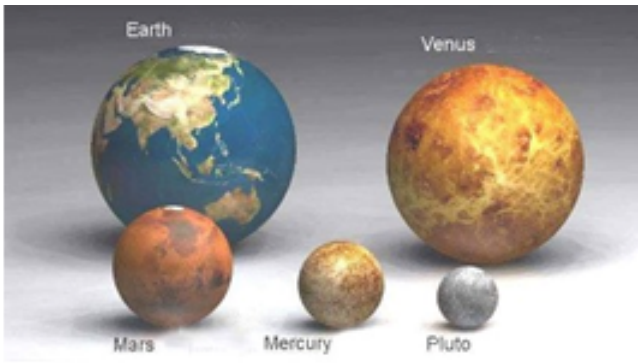

Figura 1.2: Efecto del tamaño relativo de los objetos

Solapamiento: Sucede cuando un objeto que se encuentra en el primer plano o más cercano al observador tapa parcialmente a un objeto más lejano, el segundo objeto se debe encontrar más alejado para que pueda ser cubierto por él primero, ver imagen 1.3 .

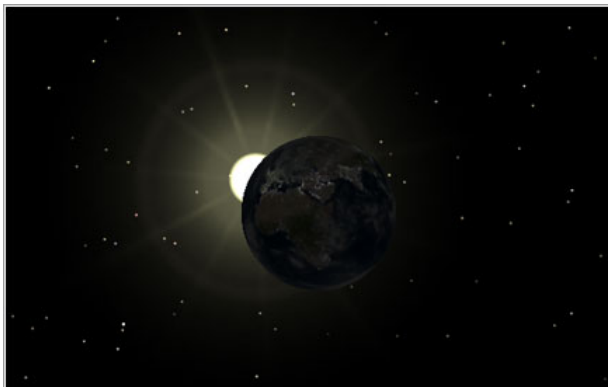

Figura 1.3: Efecto de solapamiento

Textura de gradiente: La textura de un material como el pasto enyerbado hacen que el objeto se vea más real y a la vez produce la sensación de que se encuentra más cercano al observador.

Perspectiva cónica: Es la relación entre los objetos que se encuentran en el primer y segundo plano donde el tamaño de los objetos disminuye al alejarse, puede ampliarse usando líneas paralelas que se alejan hacia el horizonte aumentando la sensación de profundidad, como se aprecia en la figura 1.4.

Figura 1.4: Efecto de la perspectiva

Desplazamiento cromático: Los objetos lejanos, por efecto de la difracción de la luz en la atmósfera aparecen más difuminados. Se observa un desplazamiento cromático hacia los tonos azules en los objetos más alejados. [Quirós et al., 2004]

1.1.2. Señales de profundidad fisiológicas

Las señales fisiológicas de profundidad pueden ser binoculares o monoculares y se basan en la estructura física de los ojos e incluyen: la acomodación o enfoque, la convergencia y la disparidad retinal o paralaje estático. Los seres humanos usan los ojos para apoyar la interpretación tridimensional de las escenas físicas basadas en las señales fisiológicas de profundidad [Pfautz, 2002]. Estas señales, particularmente la convergencia y la disparidad retinal, pueden ser estimuladas por las imágenes presentadas a cada ojo, si estas son de perspectivas ligeramente diferentes y se presentan independientemente y simultáneamente a cada ojo.

Estereopsis o VE: Cada ojo percibe una posición relativa ligeramente diferente de los objetos en la escena debido a la separación entre ellos, esta diferencia se interpreta como una medida de la profundidad. Esto se conoce como la **disparidad retinal** [Howard and Rogers, 1995] y es la fuente elemental de la percepción de profundidad. Cuando ambas imágenes se fusionan en una global se produce la VE (vista tridimensional). El resultado es una intensa sensación de

profundidad [Coon, 2005], en la imagen 1.5 se percibe como se observaría este efecto.

Figura 1.5: Efecto de la visualización estereoscópica que permite percibir la profundidad

Acomodación o enfoque: Los ojos flexionan el cristalino e incrementan la potencia óptica para enfocar los objetos cercanos. Las sensaciones de los músculos unidos al cristalino retornan al cerebro para que el objeto puede ser visto claramente [GER, 1991]. Esta sensación de cambios sirve para calcular distancias situadas hasta aproximadamente 25 cm de los ojos [Pocock and Richards, 2005]. Dicha información está disponible aun cuando usemos un solo ojo, de manera que la acomodación es una señal monocular.

Convergencia: Es el mecanismo físico que realizan los músculos que hacen rotar los ojos uno hacia el otro o en dirección contraria, es una señal binocular. Para los objetos más cercanos se produce un mayor esfuerzo por parte de los músculos oculares para converger [Morris and Maisto, 2005].

Por último, cuando no se tiene el mecanismo natural para la visión estereoscópica, se dispone de una señal que brinda resultados similares: el **paralaje de movimiento relativo**. Dicha señal es creada por los movimientos laterales de la cabeza del observador y su efecto de tridimensionalidad se conoce como profundidad cinética, demostrando que el movimiento relativo también puede actuar como una referencia de la profundidad [Dijkerman et al., 1999].

En la Tabla 1.1.2 se observa la relación entre las diferentes señales explicadas.

Señales de profundidad	Fisiológicas	No Fisiológicas	Monoculares	Binoculares
Luz y sombra		X	X	
Tamaño relativo		X	X	
Interposición		X	X	
Desviación cromática		X	X	
Perspectiva		X	X	
Estereopsis o VE	X			X
Acomodación	X		X	
Convergencia	X			X

Tabla 1.1: Relación entre las señales de profundidad.

1.2. Visión estereoscópica

Este epígrafe se adentra en la señal de profundidad binocular estereopsis, que es la que mayor efecto cualitativo produce, es más precisa y permite percibir un nuevo nivel de profundidad. Por eso es interés de este trabajo realizar el estudio de cómo se produce la estereopsis.

El ser humano para percibir el mundo utiliza dos partes fundamentales del cuerpo: el cerebro y los ojos, el funcionamiento de ambos en conjunto permite observar el resultado de manera estereoscópica. En términos geométricos, dos proyecciones cónicas monoculares, obtenidas para cada punto de vista, el ojo izquierdo (**Vi**) y el derecho (**Vd**), que tendrán en común el plano de proyección (**Pi**) y la ventana de proyección.

1.2.1. Principios del funcionamiento de la visión humana

Para comprender como se perciben las imágenes estereoscópicas y cuáles son los elementos fundamentales que la producen es necesario entender primeramente cómo funciona la visión humana. A continuación se presentan los conceptos que producen la estereopsis.

El proceso de percepción de una imagen estereoscópica o del mundo se produce a través de tres aspectos fundamentales la paralaje, la disparidad de la retina y la fusión, que es la capacidad de la mente de integrar esas imágenes.

Disparidad de la retina: Es el ángulo de convergencia formado por las líneas rectas que unen un punto determinado del espacio con cada uno de los ojos, este se proyecta dentro de la retina

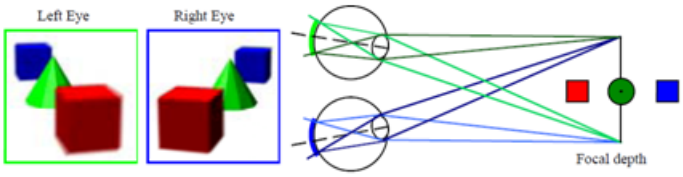

Figura 1.6: Cada ojo percibe una imagen diferente de la escena según la perspectiva desde donde observa

[Marrero, 2010], viene dada por el hecho de que cada ojo tiene un punto de vista diferente al otro, lo cual produce dos imágenes prácticamente idénticas de una misma escena. El cerebro es capaz de obtener una imagen con percepción de profundidad y de volumen a través de las diferencias de dichas imágenes [Lipton, 1997a].

Paralaje: Es la distancia entre las proyecciones de un mismo objeto en cada una de las imágenes correspondiente a cada ojo, se puede apreciar en la pantalla de proyección. Produce la disparidad de la retina que a su vez produce la estereopsis o VE.

La paralaje también se puede trabajar en medidas angulares y se relaciona con la disparidad teniendo en cuenta la distancia a la cual se encuentra el observador de la pantalla. El valor de paralaje se recomienda que no exceda un ángulo de 1.5 grados o 1,2 cm de longitud.

La paralaje se clasifica en horizontal y vertical teniendo en cuenta el eje en que se refleja, también se clasifica según la distancia del observador en:

-**Cero paralaje:** Sucede cuando los puntos homólogos [1] de las imágenes correspondientes a cada uno de los ojos caen uno sobre el otro al proyectar una imagen sobre la otra, cuando observamos un objeto con valores de cero paralaje los ojos convergen sobre la pantalla, esto quiere decir que los ejes de los ojos se cruzan sobre la pantalla, los puntos de las imágenes con cero paralaje se dice que tiene Ajuste de cero paralaje (traducido de las siglas en inglés *zero parallax settings) (ZPS).*

- **Paralaje positivo o descruzado:** Se produce cuando los valores de paralaje se encuentran entre 0 y **d**, donde **d** es la distancia intraocular. Cuando el valor del paralaje sea **d** o muy cercano los ejes de los ojos serán paralelos, esto sucede cuando se observan objetos muy lejanos. Produce

[1] Punto perteneciente a un objeto el cual puede ser proyectado en distintas posiciones en dependencia del punto de vista de cada ojo

la sensación que el objeto se encuentra dentro del monitor o por detrás de este.

- **Paralaje negativo o cruzado:** Es lo contrario al paralaje positivo, en este caso los ejes de los ojos se cruzan o convergen antes de la pantalla. La paralaje negativa se produce con los objetos del primer plano de la imagen, da la impresión que los objetos se encuentran antes de la pantalla o sea en el espacio entre la pantalla y el observador.

- **Paralaje divergente:** Es un tipo de paralaje positivo, es el caso donde los ejes de los ojos nunca convergen, ocurre cuando el valor del paralaje es mayor que **d**. No sucede en la visión de los humanos por lo cual no puede ocurrir en un entorno computarizado [Lipton, 1997a].

En la figura 1.7 se puede apreciar los distintos tipos de paralaje.

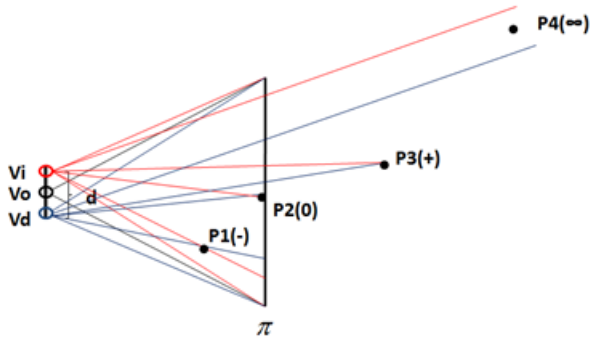

Figura 1.7: Tipos de paralajes existentes

Separación intraocular o inter pupilar

Es la distancia **d** a la cual se encuentran separados los ojos de los humanos, en el adulto promedio suele ser de 2,5 pulgadas o de 6,5 cm [Lipton, 1997a]. Es la distancia que existe entre los ojos o los ejes de los lentes, en el caso de una escena computarizada.

La relación entre la distancia intraocular y el efecto estereoscópico de profundidad es proporcional, es decir que a medida que la distancia intraocular crezca mayor será el efecto de profundidad que produce y viceversa, si las lentes estuvieran tan unidas que los ejes de los lentes se correspondieran, el resultado sería una imagen monocular.

Relación acomodación/convergencia.

La acomodación y la convergencia son procesos neurológicos que se deben realizar por separado, por conjuntos de músculos independientes. Con el paso de los años estos procesos se enlazan entre sí fusionándose como un todo gracias a toda la experiencia visual acumulada.

Esta relación produce dificultades al observar una escena estereoscópica, al observar un objeto en el mundo real los ojos se acomodan en él y a su vez convergen sobre el objeto. En el caso de las escenas estereoscópicas computarizadas, cuando se observa un objeto los ojos se enfocan en la pantalla pero convergen teniendo en cuenta el valor de paralaje de los puntos correspondientes, ver imagen 1.8. Este es el principal aspecto en que difieren la observación del mundo real y de las imágenes estereoscópicas. Al observar una escena estereoscópica se desvía la relación habitual existente entre la acomodación/convergencia, por primera vez durante mucho tiempo estos mecanismos trabajaran por separados provocando incomodidad al observar la imagen [Lipton, 1997a].

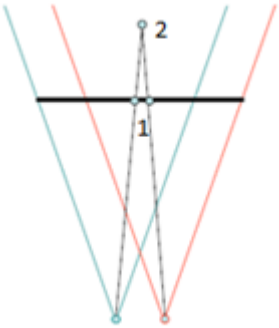

Figura 1.8: Los ojos se acomodan o enfocan en la pantalla de proyección representada por el punto 1 y convergen por detrás de la pantalla en este caso en el punto 2

Distancia del observador

D es la distancia a la que se encuentra el observador de la pantalla de proyección, mientras mayor sea el valor se percibe la escena con mayores valores de paralaje, observándose más tridimensional con un efecto más realzado para los objetos con paralaje negativo.

Existe una fuerte relación entre la distancia **D** del observador y el paralaje y es que dos valores

distintos de paralaje y distancia del observador pueden producir el mismo valor de disparidad de la retina, esto se produce porque el ángulo de la disparidad de la retina varía en dependencia de la distancia del observador. [Lipton, 1997a]

1.2.2. Ajustes de la cámara

Existen dos métodos para ajustar los pares estereográficos de la cámara: el método *Toe-in* y el *Off-Axis* [Zambrano, 2008] ver imagen 1.9.

Para el método *Toe-in*, se considera que las cámaras tienen una apertura fija, tienen una separación simétrica y apunta al mismo punto de enfoque es decir rota el *frustrum* de la cámara sobre el centro de proyección, haciendo que los planos de proyección se corten entre sí, introduciendo un paralaje vertical que causa incomodidad y mareo al observar la escena [Otero et al., 2003], sin embargo se sigue utilizando por su bajo costo de filmación. .

El método Off-Axis se basa en calcular los pares estereográficos de forma que los vectores directores de las cámaras sean paralelos, sin converger a un punto sobre el eje de proyección, este hecho introduce el concepto de *fustrum* asimétrico para cada una de las cámaras.

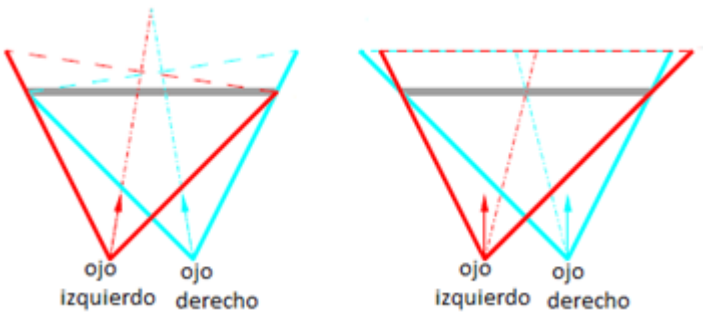

Figura 1.9: Modelo que presenta el ajuste de cámara con el método Toe-in a la izquierda y el Off-Axis a la derecha

En el método *Off-Axis* los volúmenes de visualización de ojo izquierdo y ojo derecho se denomina *frustrum* y está formado por seis planos que definen los límites de la cámara: cercano (*near*), lejano (*far*), izquierdo (*left*), derecho (*right*), arriba (*top*) y abajo (*bottom*). Estos volúmenes deben ser asimétricos respecto a la dirección de la visual. Los parámetros que establecen los

planos *left*, *right*, *top* y *bottom* de la función *frustrum* se calculan teniendo en cuenta el ángulo de apertura vertical y horizontal de la cámara y la distancia (**D**) que existe entre esta y el plano de proyección.

Para calcular el *aspect ratio* se utiliza la ecuación 1.1:

$$a = \frac{w}{h} = \frac{\tan(\beta/2)}{\tan(\alpha/2)} \tag{1.1}$$

De donde se despejan α y β para obtener el ángulo de apertura vertical y horizontal respectivamente, ver ecuación 1.2 .

$$\beta = 2 * \arctan(\frac{w}{2d}), \alpha = 2 * \arctan(\frac{h}{2d}) \tag{1.2}$$

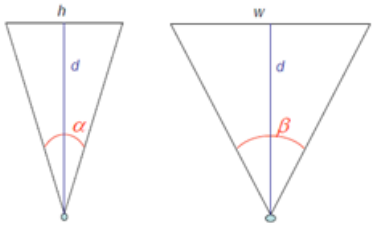

Figura 1.10: Relación entre los ángulos de apertura α y β con respecto a la distancia **d** y el alto y ancho de la pantalla de proyección

La representación para cada una de las cámaras del volumen de visualización asimétrico queda de la siguiente forma.

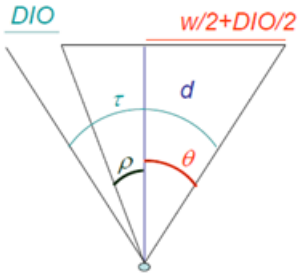

Figura 1.11: Volumen de visualización asimétrico para una de las cámaras

El ángulo τ representa un volumen simétrico lo cual produce un sobrante representado por DIO en la imagen, **d** es la distancia entre la cámara y la pantalla de proyección **Pi**, θ es la mitad del ángulo τ, $\rho + \theta$ forman el ángulo que produce el volumen asimétrico, y la relación entre ellos es la siguiente:

$$\tau = 2 * \theta \tag{1.3}$$

Para obtener los ángulos ρ y θ se utilizan las ecuaciones 1.6 y 1.7, las cuales se despejan de la ecuación 1.4 utilizada para calcular el *aspect ratio*.

$$a^{'} = \frac{\tan(\tau/2)}{\tan(\alpha/2)} = \frac{w + DIO}{h} = a + \frac{DIO}{h} \tag{1.4}$$

$$\theta = \arctan(\frac{w + DIO}{2d}) = \arctan(a * \tan(\alpha/2) + \frac{DIO}{2d}) \tag{1.5}$$

$$\theta = \arctan(a^{'} * \tan(\alpha/2)) \tag{1.6}$$

$$\rho = \arctan(\frac{w - DIO}{2d}) = \arctan(a * \tan(\alpha/2) - \frac{DIO}{2d}) \tag{1.7}$$

En el siguiente epígrafe se detallan los sistemas estereoscópicos existentes, las técnicas, formatos y dispositivos empleados para proyectar las imágenes independientemente y simultáneamente a cada ojo para percibir el efecto estereoscópico.

1.3. Sistemas estereoscópicos

Una representación estereoscópica computarizada es un sistema donde la última pieza es la mente humana, se basa en simular el proceso de observación de los humanos. Se proyectan dos imágenes ligeramente diferentes, que son las correspondientes a cada uno de los ojos, cuyas imágenes

pueden ser mostradas usando los sistemas de representación, formatos y técnicas estereoscópicas existentes.

Para lograr percibir las imágenes correspondientes a cada ojo se emplean los sistemas de representación, los cuales pueden ser pasivos y activos [García, 2008], estos incluyen un formato, una técnica y un dispositivo de visualización (gafas), en la figura 1.12 se observan ejemplos de estos dispositivos.

La diferencia radica que en los sistemas pasivos las gafas no contienen mecanismos especiales, en la mayoría de los casos son de cartón. Mientras que en los sistemas de representación activos los dispositivos visuales requieren costosos mecanismos de sincronización electrónicos. Estos dispositivos son las gafas obturadoras (*shutter glasses*), dotadas de dos lentes de cristal líquido (*LCD*) que se oscurecen (obturan) sincronizado con el proyector a la misma frecuencia que la emisión de fotogramas, esta frecuencia de refresco debe ser de al menos 120 Hz para lograr el efecto deseado. Se pueden usar distintos formatos en dependencia de las prestaciones del *hardware* [García, 2008].

1.3.1. Formatos estereoscópicos

Los formatos estereoscópicos definen de qué forma se representan las imágenes izquierda y derecha simultáneamente y se pueden clasificar en:

- Basados en color: Anaglifo, Infinitec.
- Basados en líneas y columnas: Entrelazado vertical y horizontal.
- Basado en frame: *Alternated images*, *side by side* y *quad buffer*.

Figura 1.12: Distintos dispositivos visuales para observar imágenes estereoscópicas, gafas anaglifo, polarizadas y gafas obturadoras

Las técnicas estereoscópicas comprenden el conjunto de pasos a seguir para visualizar una escena estereoscópica.

1.3.2. Sistemas de representación pasivos

Dentro de los sistemas de representación pasivos se encuentran las técnicas anaglifo, polarizado, visión cruzada (*cross eye*) y visión paralela y se caracterizan por mostrar las imágenes simultáneamente [Lipton, 1997b]. A continuación se presentan características de cada una de las técnicas:

Anaglifo: La imagen estereoscópica está conformada por las imágenes correspondientes a cada uno de los ojos, superpuestas una sobre la otra, cada una se muestra usando una pareja de filtros de colores complementarios como rojo-cian o ámbar-azul (R-GB, G-RB, B-RG). Para ser visualizada se usan gafas compuestas por filtros de colores complementarios de igual color que los usados para crear la imagen. El formato utilizado es el anaglifo, cada ojo sólo vera una imagen, ya que la otra será absorbida por el color del lente [Martín et al., 2004].

Presenta el problema de la alteración de los colores, perdida de luminosidad y cansancio visual después de un uso prolongado, sucede el fenómeno del *crosstalk* (uno de los ojos percibe la imagen que no le corresponde) y se pierde congruencia entre los campos de la imagen estéreo.

Ver resultado de aplicar la técnica anaglifo en la figura 1.13

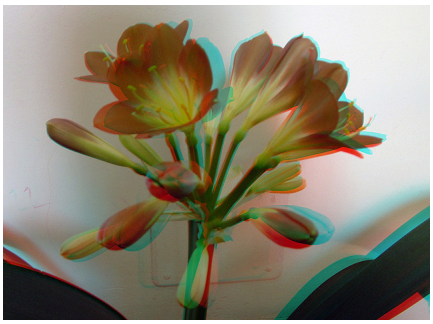

Figura 1.13: Imagen estereoscópica usando la técnica Anaglifo.

Polarizado: Las dos imágenes son proyectadas mediante dispositivos dotados de filtros polarizadores, girados 90 grados uno respecto del otro, donde no se alteran los colores de las imágenes. La pantalla debe conservar la polarización de la luz reflejada. El observador debe utilizar unas gafas con lentes polarizadas de forma congruente con los proyectores. El formato utilizado para presentar esta técnica es el *Side by Side*.

El principal inconveniente es que requiere el uso de proyectores y una pantalla que conserve la polarización de la luz y produce perdida de luminosidad.

Visión Paralela: Se divide el área de renderizado horizontalmente, las imágenes correspondientes a cada ojo se colocan una al lado de la otra, la imagen correspondiente al ojo izquierdo se muestra a la izquierda y a la derecha la del ojo derecho. El formato de presentación es similar al *Side by Side*. Las líneas de visión de ambos ojo se fijan en el infinito para que los ejes de visión queden paralelos. Se puede utilizar un dispositivo de espejos como el visor de Holmes. Presenta la deficiencia que las imágenes deben ser muy pequeñas, como máximo debe existir una distancia de 6,5 cm entre los centros de ambas imágenes.

El resultado de aplicar la técnica de visión paralela se puede ver en la figura1.14

Side by side: Se divide el área de renderizado horizontalmente, las imágenes correspondientes a cada ojo se colocan una al lado de la otra, la imagen que le corresponde al ojo izquierdo se sitúa a la izquierda y la del ojo derecho a la derecha, de ese modo se aprovechan todas las líneas de renderizado.

Visión Cruzada (*Cross Eye*): Es muy similar a la visión paralela, las imágenes se muestran

Figura 1.14: Imagen estereoscópica usando la técnica Visión Paralela.

una al lado de la otra pero invertidas, la imagen derecha se muestra a la izquierda y la izquierda a la derecha, es decir se intercambian las imágenes. Las líneas de visión de ambos ojo se cruzan, fijándose en un punto medio más cercano. De esta forma el ojo derecho observa la imagen izquierda y viceversa. El formato de representación es similar al *Side by Side*. Es posible emplear imágenes con formatos mayores de 6,5 cm, no es necesario el uso de gafas para su visualización.

Para la visión paralela y cruzada se pierde la mitad de la resolución horizontal al mostrarse las imágenes una al lado de la otra.

Ver resultado de aplicar la técnica de visión cruzada en la figura1.15

Figura 1.15: Imagen estereoscópica usando la técnica Visión Cruzada.

1.3.3. Sistemas de representación activos

Entre las técnicas incluidas en los sistemas activos se encuentran *alterned page*, entrelazado vertical y horizontal y el *quad buffer*, a continuación se explican brevemente cada una de ellas.

Alterned Image: Las dos imágenes son mostradas en un monitor de forma alternativa, con una alta frecuencia de refresco. Es necesario tarjetas de video de gama alta compatibles con dispositivos estéreo que cuentan con dos *buffers*, la próxima imagen se carga en el *back buffers* mientras que en el *front buffers* se proyecta la escena.

Entrelazado: Se basa en la utilización de los campos par e impar de un monitor en dirección horizontal o vertical para poner cada una de las dos imágenes. Es decir, que en los campos pares se dibuja la imagen correspondiente al ojo derecho y en los impares la correspondiente al ojo izquierdo, o viceversa.

Ver resultado de aplicar la técnica entrelazado en la figura 1.16

Figura 1.16: Imagen estereoscópica usando la técnica entrelazado.

Quad Buffer: Se puede utilizar en las tarjetas de gama alta *Nvidia Quadro* compatibles con dispositivos estéreos que presentan cuatro *buffers*. Se dispone de *buffers* independientes para las imágenes de la derecha e izquierda, lo cual permite proyectar ambas imágenes simultáneamente.

Es la habilidad de renderizar en el *Front Left* y *Front Right* mientras que en el *Back Left* y *Back Right* se carga el siguiente *frame buffers* izquierdo y derecho. Los *buffers* izquierdo y derecho delanteros proyectan las imágenes mientras los traseros van cargando la próxima imagen y luego estos se intercambian.

Las técnicas activas pueden presentar problemas de parpadeo (*flickering*) y de *crosstalk* o *ghosting*. En el caso del entrelazado se pierde la mitad de la resolución, ya que la escena para uno de los ojos se muestra en los campos pares y la otra en los impares.

En la figura 1.17 se muestran las principales técnicas de visualización estereoscópica, los dispositivos de visualización y de proyección que se utilizan para mostrar la escena.

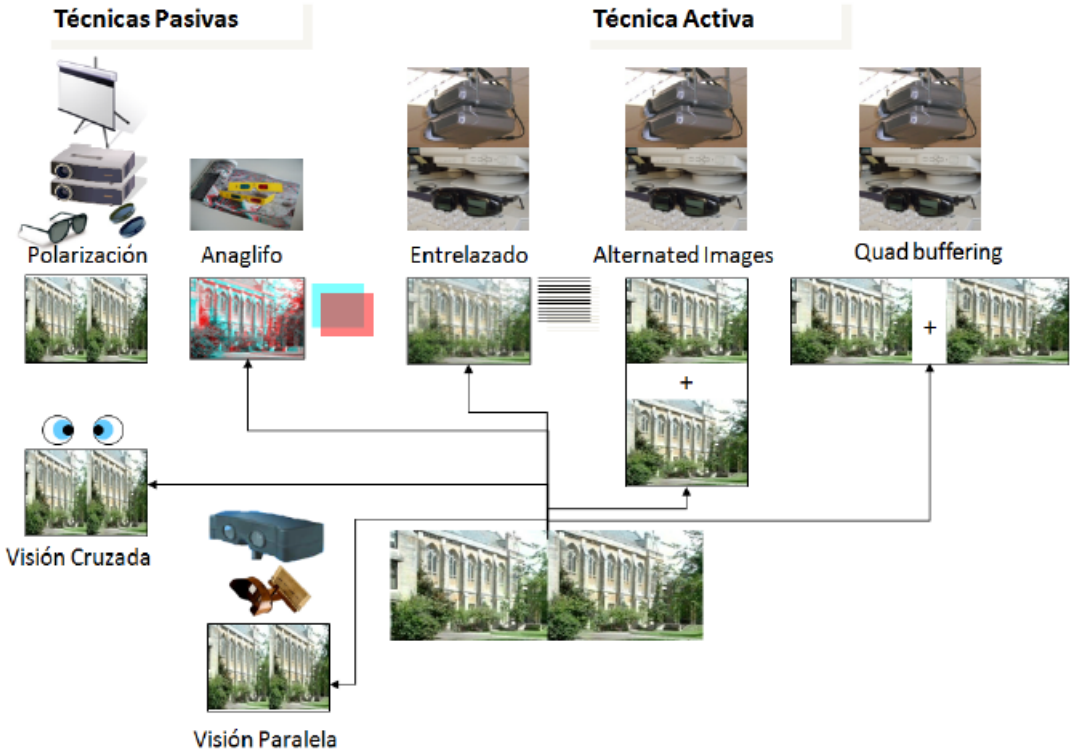

Figura 1.17: Técnicas y dispositivos visuales.

1.4. Motores gráficos con funcionalidades estereoscópicas

En este epígrafe se presentan algunos de los motores gráficos que han incorporado las funciones estereoscópicas para la visualización de la escena.

Para la creación de las aplicaciones de RV se usan motores gráficos que brindan funcionalidades para evitar la dificultad de la utilización de bibliotecas gráficas a bajo nivel.

Después de alcanzar grandes resultados en la creación de aplicaciones gráficas de forma monocular, la incorporación de la visión estereoscópica añade mayor calidad visual y aumenta la sensación de inmersión del usuario en la escena virtual, por lo que en la actualidad los motores gráficos optan por agregar las funcionalidades estereoscópicas.

Los motores gráficos se pueden dividir en motores gráficos propietarios y *open source*. Los motores propietarios se caracterizan por ser usados para la creación de aplicaciones comerciales, especialmente los juegos de consola, por esta razón el uso de la tecnología estereoscópica es más avanzado. En el caso de los motores *Open Source* el desarrollo de las técnicas estereoscópicas se encuentra en sus inicios, existen muy pocos motores que incluyen esta tecnología entre sus funcionalidades.

1.4.1. Motores gráficos propietarios

A continuación se comentan algunos de los motores de las más famosas compañías desarrolladoras de video juegos.

La nueva saga de *Call of Duty* creado por *Treyarch* está disponible en versión 3D estereoscópica, el juego es compatible con equipos *3D-ready HDTV* y computadoras en 3D, utilizando las gafas *shutter glasses*, que ofrecen una profundidad de campo y un nivel de inmersión en pantalla sin precedentes.

Otras compañías que han incluido las funciones estereoscópicas a sus motores gráficos son la: *Ubisoft* con un motor que usa técnicas estereoscópicas similares a las usadas en la película Avatar, *Crytek* ha creado un nuevo motor el *CryENGINE 3* diseñado para incluir efecto 3D estereoscópico, mientras que el motor *Unreal Engine 3* de la *Epic Games* ha comenzado el desarrollo de funciones que permite el trabajo de estas tecnologías.

Drivers estéreo

Algunos fabricantes, tanto de tarjetas gráficas como de dispositivos de visualización, han desarrollado drivers con funciones estéreo (*NVidia3DVision*). Estos *drivers* definen con *OpenGL* dos o más cámaras para una sola escena. El resultado es un estereograma que se puede mostrar en los diversos dispositivos que soportan el *driver* en cuestión (las gafas de obturación, monitores

auto-estéreo, etc.). Obviamente, el ofrecimiento de estos controladores con funciones estéreo representa una ventaja competitiva de primer orden para los fabricantes de hardware. El problema es que estos drivers no son multiplataforma, y el programador no tiene el control total de los parámetros asociados con la naturaleza estéreo de la escena. [NVi, 2010]

1.4.2. Motores gráficos *Open Source*

Algunos de las herramientas *Open source* que añaden estereoscopía son:

OpenSceneGraph (OSG)

OSG es una biblioteca gráfica 3D *Open Source* de alto rendimiento que visualiza contenido de manera monocular y estereoscópica, usada por los desarrolladores de aplicaciones en campos tales como la simulación visual, los juegos, la realidad virtual, la visualización científica, y el modelado.

OSG tiene soporte para la representación estereoscópica en el modo anaglifo (rojo/verde, rojo/cian), *quad buffer* (en estéreo activo usando *shutter glasses*), y estéreo pasivo con la proyección y gafas polarizadas, y modo entrelazado. Permite la configuración de algunos de los parámetros de la escena, como distancia a la que está la pantalla, tamaño de la pantalla y separación interpupilar [OpenSceneGraph, 2010].

Ogre 3D

Object-oriented Graphics Rendering Engine (Ogre) es un motor gráfico de código abierto. Está escrito en C++ y utiliza las *(Application Program Interface)* en español: Interfaz de Programación de Aplicaciones *(API)* gráficas de *Directx* y *OpenGL*. Presenta una colección de *plugins* que lo hacen una herramienta sumamente potente, así como su fácil integración con bibliotecas de sonido, red, física, entre otras. Hace un uso óptimo de la aceleración por hardware. Buena documentación para su empleo y respaldado por una comunidad de desarrollo extensa. Se encuentra publicado bajo la licencia LGPL. [Ogre, 2010]

Ogre cuenta con un *plugin* para estereoscopía, que tiene soporte para el modo anaglifo (red/blue), modo salida dual (para HMDs y dos proyectores con filtros polarizados y gafas polarizadas), y

modo entrelazado horizontal, vertical y *checkboard pattern* (para monitores con filtros lenticulares y *shutter glasses*). Permite la configuración de la distancia focal **D**, y de la separación intraocular **d**.

En el siguiente epígrafe se detallan las características más relevantes de la biblioteca STK, herramienta a la se le incorporara el módulo estereoscópico.

1.5. Arquitectura de la STK

La biblioteca STK es una herramienta creada en la UCI orientada a aplicaciones gráficas tridimensionales genéricas, esencialmente simuladores y juegos 3D, brinda soporte para las *API OpenGL* y *DirectX*, además de ser compatible con múltiples sistemas operativos (*Windows* y *Linux*) facilitando en cada uno de ellos la interacción con ventanas y periféricos como el (*mouse*, teclado, *joystick*).

Posee un grafo de escena que permite el manejo dinámico de los objetos de la escena, y brinda la posibilidad de aplicar técnicas, atributos *(Attributes)* y evaluadores *(Handlers)* de manera jerárquica a las diferentes formas *(Shapes)* según su distribución dentro del grafo.

Su arquitectura se encuentra dividida por módulos lo que permite la integración y/o acoplamiento de nuevos módulos sin necesidad de una restructuración a gran escala del sistema completo [Quintana and Alfonso, 2010]. Además permite la utilización de las funcionalidades de *OpenGL* directamente.

La STK cuenta con una clase llamada `CApplication` que contiene funcionalidades para el control de la aplicación, dibujado de la escena y trabajo con las cámaras. La clase `CCamera` contiene funciones que permiten la configuración del *frustrum*, la *perspective*, punto de mira y *viewport* de la cámara. Estas funciones son un encapsulamiento a las funciones de *OpenGl*: *glfrustrum* , *glperspective*, *gluLookAt* y *glViewport*, encargadas de la configuración de la cámara de *OpenGL*. También contiene un conjunto de clases matemáticas las cuales contienen la implementación de vectores, matrices, *quaternions*, así como funciones que permiten trabajar con estos elementos.

Capítulo 2

Solución Propuesta

En este capítulo se plantean los elementos a tener en cuenta para dar solución al problema de crear un módulo que brinde funcionalidades estereoscópicas a la STK. Se tienen en cuenta los principales elementos geométricos de una escena estereoscópica, las distintas técnicas y formatos existentes para lograr la representación final de la escena con las características exigidas por el dispositivo de visualización. Además se presentan las herramientas y tecnologías seleccionadas para la confección del sistema.

2.1. OpenGL

En el módulo estereoscópico se usan funcionalidades de *OpenGL* para definir la geometría de la escena estereoscópica y para la configuración de las técnicas de visualización. Se decide utilizar esta *API* por su portabilidad, lo que permite que el sistema pueda ejecutarse en una amplia variedad de arquitecturas y de soportes gráficos. Además la herramienta STK, a la cual se desea integrar el módulo, utiliza la biblioteca *OpenGL* como *API* gráfica y finalmente por las facilidades que brinda para la configuración de una escena estereoscópica cumpliendo con las restricciones necesarias.

Se utilizan las funciones `glStencilFunc` y `glStencilOp` del *stencil buffer* de *OpenGL* para la definición de la máscara y el filtro para la implementación de la técnica entrelazado, la función

`glColorMask` para configurar una máscara que permita dibujar la escena teñida de colores complementarios para la técnica anaglifo.

Las funciones `glFrustum` y `gluLookAt` permiten la configuración del punto de mira y volumen de visualización de la cámara. Aunque estas funciones no se usan directamente ya que el método de dibujado de la STK imposibilita su uso. En la herramienta STK cada cámara tiene asociado su grafo de escena y al realizar el proceso de dibujado se utilizan los valores de configuración asociados a cada cámara, sobrescribiendo los valores de la matriz de *modelview* configurados directamente en *OpenGl*.

Como solución se utiliza las funciones de la clase `CCamera` de la STK que permiten la configuración de estos parámetros mediante las funciones `LookAt` y `setFrustum`. Estas funciones internamente hacen un llamado a las funciones `gluLookAt` y `glFrustum` de *OpenGl*, por lo que en el próximo epígrafe se explica cómo trabajan estas funciones.

2.2. Gestión de elementos geométricos de la escena

En este epígrafe se enuncian los elementos geométricos que componen una escena estereoscópica, así como la relación que existe entre ellos.

Los elementos geométricos que componen una escena estereoscópica son:

- El observador, persona que visualiza la escena.
- Dos cámaras que representan cada uno de los ojos del observador
- La pantalla, donde se proyecta el resultado.

Sobre estos componentes se deben manejar un conjunto de atributos, ver figura 2.1, que garantizan un sistema lo más detallado y realista posible, además que exista una relación de 1:1 entre el entorno gráfico y el mundo real. Para el manejo de estos atributos se usan las funciones matemáticas de la STK relacionadas con las clases `CVector3`, `CVector2` y `CQuaternion`.

Figura 2.1: Relación y atributos de los elementos geométricos de la escena

2.2.1. Configuración del observador

Sobre el observador se manejan tres aspectos fundamentales:

- Posición de la cabeza, definido por un vector llamado *position* de tres componentes (x,y,z).
- Orientación de la cabeza, representada por un *quaternion* definido como *rotation*.
- Distancia inter pupilar (d), es un número real llamado *interaxial*.

Conociendo la posición y orientación de la cabeza se puede saber a qué dirección se encuentran enfocados los ojos en todo momento. El vector *position* representa la posición del punto medio de la cabeza situado sobre la base de la nariz. La posición y la distancia inter pupilar se utilizan para la ubicación de las cámaras, en la figura 2.2 se pueden observar dichos atributos.

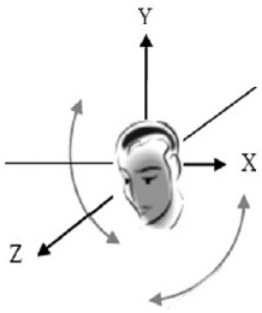

Figura 2.2: Atributos que se desean conocer del observador.

Las cámaras permiten captar la escena, estas están muy ligadas al observador ya que cumplen la función de los ojos en la vida real.

Para obtener la posición de las cámaras es necesario crear un vector que contenga la distancia interpupilar del observador. La posición de la cámara izquierda se obtiene restando la posición del observador con el vector creado, para la cámara de la derecha se realiza la misma operación, pero se suman los vectores, quedando las cámaras una al lado de la otra separadas por la distancia interpupilar.

```
vectorInter = CVector3f::kNewInstance(observer->getInteraxial()/2,0.0f,0.0f);

leftEye = observer->getPosition() - vectorInter;

rightEye = observer->getPosition() + vectorInter;
```

Para el cálculo de los pares estereográficos de las cámaras, de los métodos planteados en el Capítulo 1 se escogió el método *Off Axis*. Se selecciona este ya que *Toe-in* introduce paralaje vertical en la escena, produciendo mayor dificultad en la acomodación del ojo y fatiga ocular. Mientras que el método *Off Axis* permite una perfecta sensación estereoscópica [Otero et al., 2003]. Para su implementación es necesario la configuración del punto de mira y del volumen de visualización de la cámara.

El método *Off-Axis* plantea un volumen de visualización asimétrico por lo que es necesario utilizar la función `glFustrum (left, right, bottom, top, nearVal, farVal)` que permite calcular el valor de los planos *left*, *right*, *top* y *bottom* y establecer un volumen que cumpla con las restricciones del método. Los valores de los planos *near* y *far* se conocen del *display*.

Los planos a calcular reciben inicialmente los valores del tamaño de la pantalla en milímetros (Mm), un valor conocido del *display*, dividido entre dos. Como se muestra a continuación:

```
left = -(sizeMm.fX()/2);

right = sizeMm.fX()/2;

top = sizeMm.fY()/2;

bottom = -(sizeMm.fY()/2);
```

Luego se obtiene la distancia del observador al *display* restando la posición del *display* y la posición de la cámara activa. El vector *center* se calcula hallando la diferencia entre el vector *eyeInDisplay*, que como su nombre lo indica es la proyección de la posición de la cámara activa sobre el *display* y la posición del *display*. Finalmente los planos *left* y *right* que describen un volumen asimétrico se calculan de la siguiente forma:

```
left = (left - center.fX()) * nearDisplay;
```

```
right = (right - center.fX()) * nearDisplay;
```

Donde *nearDisplay* es la proporción que existe entre la distancia del observador a la pantalla de proyección y el plano cercano, lo cual permite que el volumen de proyección tenga una correspondencia 1:1 con el mundo real.

En la figura 2.3 se puede apreciar cómo queda el *frustum* de una cámara después de realizadas las transformaciones anteriores.

```
Camera->SetFrustum(display->getZNear(),display->getZFar(),left,right,top,bottom);
```

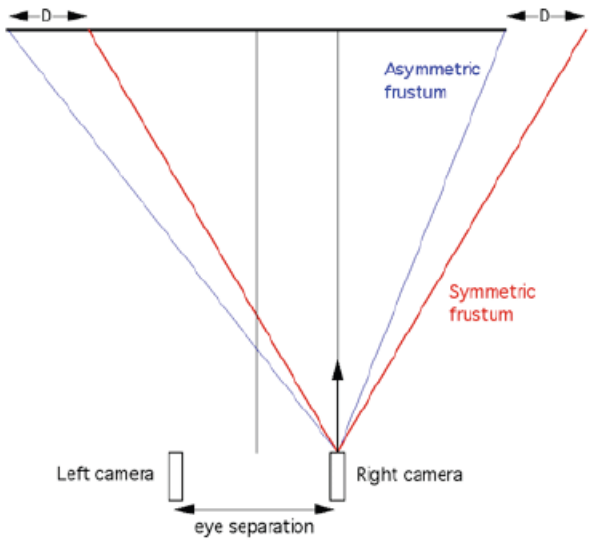

Figura 2.3: *Frustum* asimétrico según el método Off Axis

La función `gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz)` permite definir el posicionamiento de la cámara y otros aspectos que determinan como se observa

la escena. Los tres primeros parámetros representan el punto de vista de la cámara *(eye)*, los tres siguientes son el punto de atención *(center)*, es decir el punto que se está mirando. El vector *eye* y *center* definen la línea de visión de la cámara, los tres últimos *(up)* indican que vector está hacia arriba.

A la función se le pasan tres vectores el primero es la posición de la cámara que se encuentre activa puede ser *leftEye* o *rightEye*. En el caso del vector *center* se calcula igualmente en dependencia de la cámara activa. El vector *displayUp* se calcula hallando el vector ortogonal a los vectores perpendicular y horizontal del *display*, ver figura 2.4.

```
LeftCamera->LookAt(leftEye, center, displayUp);

RightCamera->LookAt(rightEye, center, displayUp);
```

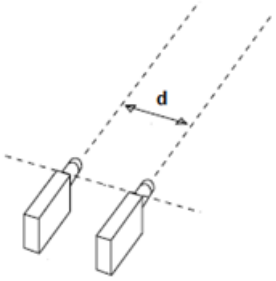

Figura 2.4: Posicionamiento de las cámaras según el método Off-Axis

2.2.2. Configuración del *display*

El *display* es la pantalla donde se proyecta la escena, puede coincidir con el monitor o una pantalla de proyección. De este se conoce:

- La posición de la pantalla que se encuentra definida por un vector llamado *position* con tres componentes (x,y,z), que representan el centro de la pantalla.
- La resolución del *display* en *pixel*, definido por un vector *sizeP* de dos componentes (x,y), donde cada uno representa el ancho y alto.

- El tamaño del *display* en milímetros, definido por un vector *sizeMm* de dos componentes (x,y), donde cada uno representa el ancho y alto.
- La posición del *viewport* de la pantalla, definido por un vector *viewPortPos* de dos componentes (x,y), donde cada valor representa la coordenada (x, y) de inicio del *viewport*, en este caso el tamaño del *vierport* coincide con el tamaño del *display* en mm porque la aplicación se muestra en pantalla completa.
- Los vectores perpendicular, horizontal y *up* de la pantalla representados por tres vectores de tres componentes cada uno.
- Distancia del observador al plano lejano y cercano, definido por dos números reales *znear* y *zfar*.

El tamaño de la pantalla en mm es necesario para obtener el volumen de visualización asimétrico de la escena, mientras que el tamaño en pixel permite la configuración del *viewport*. En la siguiente figura se puede observar los atributos necesarios del *display*.

Figura 2.5: Atributos que se desean conocer del *display*

En el siguiente epígrafe se detalla el proceso de configuración de cada una de las técnicas de visualización estereoscópicas soportadas por el módulo.

2.3. Gestión de técnicas de visualización estereoscópicas

En este módulo se implementan las técnicas pasivas: anaglifo, visión paralela, visión cruzada, y las activas: entrelazado, *alterned page* y *quadbuffer* ya que son las más usadas en los sistemas estereoscópicos existentes en la actualidad.

Las técnicas estereoscópicas activas producen excelentes efectos y generalmente son usadas en ambientes de RV profesionales a pesar de su alto costo, mientras que las técnicas pasivas se destacan por su menor costo de montaje y pueden ser empleadas en aplicaciones más sencillas.

De manera general se necesita la configuración de entidades como filtros, máscaras, trabajo con el *viewport* y los *buffers*, para lograr dibujar la escena estereoscópicamente según las características de cada una de las técnicas.

2.3.1. Selección de filtros y máscaras

Filtros y máscaras son dos conceptos muy relacionados, al establecer una máscara se define el área o patrón de dibujado, los filtros definen como y donde se aplicaran las máscaras, es decir la forma de visualización de la escena según la máscara asociada.

2.3.2. Selección de ojo activo

Otro aspecto importante para dibujar la escena es saber que ojo está activo en cada momento para así conocer con que cámara pintar. El ojo activo es un atributo conocido del observador que tomara los valores *LeftEye* o *RightEye*.

```
enum ActiveEye{LeftEye, RightEye};
```

2.3.3. Técnicas de visualización estereoscópica

En la figura 2.6 se muestra el formato de visualización de cada una de las técnicas. La técnica pasiva anaglifo y la técnica activa entrelazado usan máscaras y filtros para dibujar la escena por lo que sus algoritmos son muy similares.

Mientras que *Alterned Images* y *Quad buffering* trabajan con los *buffers* de las tarjetas de video y sus algoritmos también comparten aspectos en común. Para *SidebySide* y *Crosseye* se realizan transformaciones en el *viewport* que permiten dibujar una escena al lado de la otra.

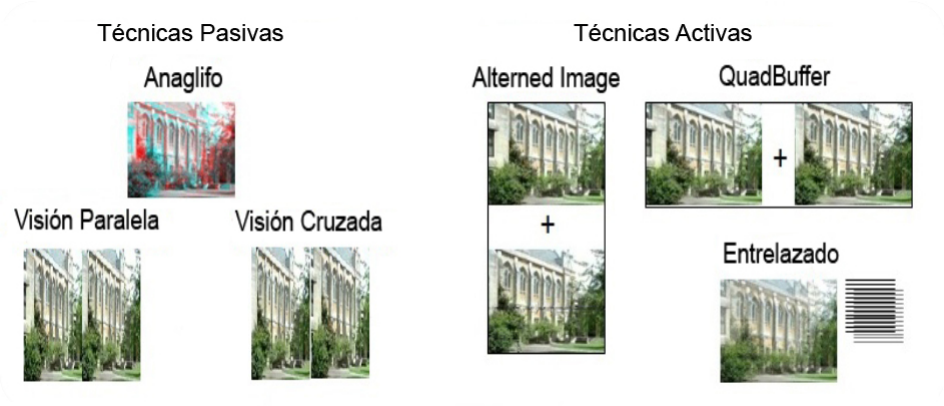

Figura 2.6: Técnicas de visualización estereoscópicas soportadas por el módulo

A continuación se detallan cada uno de los algoritmos propuestos para cada técnica.

Anaglifo: El algoritmo utilizado para la implementación de la técnica anaglifo es el siguiente:

1- Inicializar *buffer*

2- Calcular la proyección cónica (ojo izquierdo)

3- Activar filtro de color para escena izquierda

4- Dibujar la escena

5- Desactivar filtro de color para escena izquierda

6- Calcular la proyección cónica (ojo derecho)

7- Activar filtro de color para escena derecha

8- Dibujar la escena

9- Desactivar filtro de color para escena derecha

10- Volcar el *buffer*

El paso inicial para los algoritmos correspondientes a todas las técnicas es inicializar los *buffers*, en dependencia de la cantidad que utilice la técnica activa.

Para calcular la posición cónica (izquierda) se establece el ojo izquierdo como ojo activo para dibujar primero la escena desde el punto de vista de la izquierda, además del posicionamiento y cálculo del volumen de visualización de la cámara.

Para activar el filtro en el caso de anaglifo se utiliza la función de *OpenGL* `glColorMask` que recibe cuatro parámetros *red*, *green*, *blue*, *alpha*. Estos valores especifican si los componentes de color individuales en el *frame buffer* deben o no dibujarse. Luego se dibuja la escena y se desactiva el filtro pasándole a la función `setEyeFilter` el parámetro *NoFilterGlasses* para limpiar los filtros.

```
filter_Mask->setEyeFilter(NoFilterGlasses, LeftEye);
```

Por último se repiten los pasos anteriores a partir del cálculo de la posición cónica pero para la cámara de la derecha.

Visión Paralela: Es una técnica pasiva que se basa en dividir horizontalmente la pantalla a la mitad y pintar la escena en dos *viewports* ubicados uno al lado del otro. El algoritmo utilizado para la implementación de esta técnica se expone a continuación:

1- Inicializar *buffer*

2- Calcular la proyección cónica (ojo izquierdo)

3- Activar filtro de color para escena izquierda

4- Dibujar la escena

5- Desactivar filtro de color para escena izquierda

6- Calcular la proyección cónica (ojo derecho)

7- Activar filtro de color para escena derecha

8- Dibujar la escena

9- Desactivar filtro de color para escena derecha

10- Volcar el *buffer*

Para esta técnica es necesaria la creación de dos *viewports* de igual dimensión. La función `SetViewPort(x, y, width, height)` de la STK ejecuta internamente la función `glViewport(x, y, width, height)` de *OpenGl* como se explico anteriormente, los dos primeros parámetros representan la coordenada de inicio del *viewport*, y los dos últimos son el ancho y alto del *viewport* respectivamente y se obtiene por el tamaño en *pixel* del *display*.

Para las técnicas visión paralela y cruzada el tamaño del *viewport* es igual a la resolución de la pantalla, pero el objeto no se encuentra en el centro de la pantalla, para lograr que los objetos se proyecten en el centro de cada uno de los *viewports* hay que trasladar el valor x de posición inicial para cada uno de ellos. Para el primer *viewport* el valor inicial seria el resultado de restar cero menos un cuarto de la resolución de la pantalla, para el segundo se resta un cuarto de la resolución menos el punto que representa la mitad de la pantalla horizontalmente.

En la figura 2.7 se puede apreciar cómo quedan posicionados los *viewports* para lograr que los objetos queden centrados.

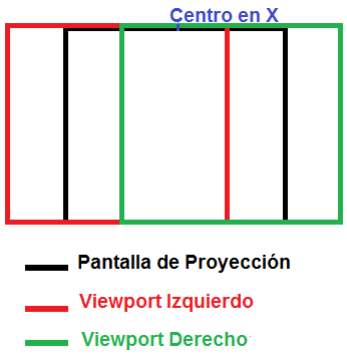

Figura 2.7: Posición de los *viewports* para las técnicas de visión paralela y visión cruzada

Visión Cruzada:

Es muy similar a la visión paralela, la diferencia es que las imágenes se invierten, a la izquierda se dibuja la escena de la cámara derecha y a la derecha la escena vista desde la cámara izquierda.

El algoritmo utilizado para la implementación de esta técnica se expone a continuación:

1- Inicializar *buffer*

2- Calcular la proyección cónica (ojo izquierdo)

3- Activar filtro de color para escena izquierda

4- Dibujar la escena

5- Desactivar filtro de color para escena izquierda

6- Calcular la proyección cónica (ojo derecho)

7- Activar filtro de color para escena derecha

8- Dibujar la escena

9- Desactivar filtro de color para escena derecha

10- Volcar el *buffer*

Se realiza la misma configuración de los *viewports* de la técnica de visión paralela, la diferencia entre estos algoritmos es al pintar, lo que se hace se invertir el orden de dibujado, es decir se dibuja primero con la cámara de la derecha y luego con la cámara izquierda.

Entrelazado Horizontal: Es una técnica activa que pinta una escena en las filas pares y la otra en las impares, es necesario sincronizarla con las gafas obturadoras.

El algoritmo utilizado para la implementación de esta técnica se expone a continuación:

1- Inicializar *buffer*

2- Calcular la proyección cónica (ojo izquierdo)

3- Activar filtro entrelazado (filas pares horizontales) para escena izquierda

4- Dibujar la escena

5- Desactivar filtro entrelazado para escena izquierda

6- Calcular la proyección cónica (ojo derecho)

7- Activar filtro entrelazado (filas impares horizontales) para escena derecha

8- Dibujar la escena

9- Desactivar filtro entrelazado para escena derecha

10- Volcar el *buffer*

Esta técnica emplea el *Stencil buffer* de *OpenGL* y las funciones `glStencilFunc` y `glStencilOp` ligadas a este *buffer*, esto permite definir *pixel* a *pixel* donde se desea pintar. En la función *InterlacedMask* se crea una máscara que define como dibujar la escena solo en las filas pares.

A la función `setEyeFilter` se le pasa como parámetro *InterlacedGlasses* para activar el filtro correspondiente a la técnica de entrelazado, este filtro define que la escena captada por la cámara izquierda se dibuja cuando el valor del *Stencil buffer* sea igual a la máscara definida en la función anterior, mientras que la escena vista desde de la cámara derecha se dibuja cuando el valor sea distinto del almacenado en la máscara.

Alterned Image: Es una técnica activa donde se muestran una escena detrás de la otra, es necesario que el *display* tenga una velocidad de refresco de 120 Hz.

El algoritmo utilizado para la implementación de esta técnica se expone a continuación:

1- Inicializar el *back buffer*

2- *if* vista es izquierda *then*

3- Calcular la proyección cónica (ojo izquierdo)

4- Dibujar la escena en el *back buffer*

5- *else*

6- Calcular la proyección cónica (ojo derecho)

7- Dibujar la escena en el *back buffer*

8- *end if*

9- Intercambiar *buffers*

Esta técnica inicializa el *buffer* trasero inicialmente y en dependencia del ojo activo calculan la proyección cónica, dibujan la próxima escena en el *buffer* trasero, y luego se intercambian los *buffers*, para mostrar la escena en el *buffer* delantero. Después de terminado el proceso para una de las cámaras se llama a la función `SwapEye()` que cambia el ojo activo para que se realice el proceso con la otra cámara.

Quad buffer: La peculiaridad de esta técnica activa es que utiliza cuatro *buffers* por lo que precisa de tarjetas Nvidia Quadro compatibles con la tecnología estéreo.

1- El algoritmo utilizado para la implementación de esta técnica se expone a continuación:

2- Inicializar los *back buffer*

3- Calcular la proyección cónica (ojo izquierdo)

4- Dibujar la escena en el *back buffer* izquierdo

5- Calcular la proyección cónica (ojo derecho)

6- Dibujar la escena *back buffer* derecho

7- Volcar los *buffers*

Se inicializan los *buffers* traseros, se calcula la proyección cónica para las dos cámaras similar a las técnicas anteriores y se dibuja la escena en los *buffers* traseros correspondientes a cada cámara, de ese modo mientras se muestran las dos escenas en los *buffers* delanteros se va dibujando en los traseros, finalmente se intercambian los *buffers* traseros con los delanteros. No es necesario llevar el control del ojo activo, ya que se muestran ambas imágenes a la vez.

2.4. Integración de la visualización estereoscópica en la STK

La solución propuesta en los epígrafes anteriores se encarga solamente de la configuración de los elementos geométricos que componen la escena estereoscópica, así como de la configuración de las técnicas pero no del dibujado de la escena, porque el objetivo final es brindar un módulo que permita incorporar las técnicas estereoscópicas a la STK.

Para probar las funcionalidades de la propuesta de solución se confecciona un demo que utiliza las funcionalidades de dibujado de la STK y la configuración de los elementos geométricos de la escena, así como las técnicas de visualización estereoscópicas del módulo.

Este demo hereda de la clase `STKMain` de la STK y permite sobrescribir la función `OnDraw()` que es donde se realiza toda la lógica de dibujado de la biblioteca STK, en el siguiente algoritmo se describe el orden de las operaciones necesarias para dibujar la escena estereoscópica para las distintas técnicas estereoscópicas exceptuando el caso de *Alterned Image*.

1- Configurar la técnica de visualización activa (ojo izquierdo)

2- Dibujar escena

3- Limpiar los buffers

4- Configurar la técnica activa (ojo derecho)

5- Dibujar escena

Para configurar la técnica de visualización se realizan las operaciones correspondientes, mencionadas anteriormente. Al terminar la configuración de la escena estereoscópica para la cámara izquierda se realiza el dibujado de los elementos de la escena utilizando las funciones de la STK. Luego se limpian los buffers y se repiten los pasos anteriores pero para la cámara de la derecha.

En el caso de *Alterned Image* solo se dibuja la escena desde un punto de vista en el ciclo de renderizado por lo que el algoritmo utilizado cambia ligeramente:

1- Configurar la técnica *Alterned Image* (ojo activo)

2- Si el ojo activo es el derecho

3- Dibujar la escena de la cámara izquierda

4- Sino

5- Dibujar la escena de la cámara derecha

Para la configuración de *Alterned Image* se sigue la misma lógica que para el resto de las técnicas. Se realiza el dibujado de la escena en dependencia del ojo que se encuentre activo, si es el ojo

derecho se dibuja la escena de la cámara izquierda y viceversa, ya que en la configuración de la técnica se cambia el ojo activo antes del dibujado de la escena, de ese modo en la próxima entrada a la función de dibujado `OnDraw()` queda listo para configurar la otra cámara.

Para el dibujado de la escena la STK utiliza las siguientes funciones:

```
pkRenderEngine()->Push(leftC);

leftC->pkSceneToDraw()->Draw(leftC);

pkRenderEngine()->Pop(leftC);
```

Es necesario añadir una función `InitializeCamera` a la clase Interfaz donde se crean las cámaras utilizando la entidad `CCamera` de STK, con estas cámaras y el modo activo se crea el objeto *stereo* instancia de la clase principal del módulo estereoscópico, utilizando el patrón *Singleton*.

Capítulo 3

Ingeniería del sistema

En este capítulo se presenta la ingeniería del sistema. Se definen los diagramas y artefactos que se generan en las etapas: Levantamiento de requisitos, Diseño e Implementación del sistema dentro del proceso de desarrollo.

3.1. Características del sistema

En este epígrafe se enuncia las reglas del negocio, se define el modelo de dominio que contiene los principales conceptos presentes en el sistema. Se listan los requisitos funcionales y no funcionales del sistema, además se presenta el Diagrama de Casos de Uso del Sistema donde se aprecia cómo quedan agrupados los requisitos funcionales.

3.1.1. Reglas del negocio

Con el propósito de definir las condiciones que deben satisfacerse para el correcto funcionamiento del módulo de visión estereoscópica se establecen como reglas del negocio las siguientes:

- Lograr una relación 1:1 entre el espacio usuario y el espacio modelo para brindar una representación exacta en el monitor de las dimensiones reales de los objetos que se necesitan representar.

- Cumplir las normas que rigen la estereoscopía, donde se tengan en cuenta, para una correcta representación de la escena en dimensiones reales, los atributos asociados al observador y a la pantalla.
- La escena se debe visualizar utilizando las técnicas de visualización estereoscópicas: anaglifo, visión paralela, visión cruzada, entrelazado vertical, *alterned image*, *quad bufferer*.

3.1.2. Modelo de Dominio

El modelo de dominio mostrado en la figura 3.1 representa un acercamiento a la solución propuesta donde se modelan los principales conceptos involucrados en el sistema así como sus relaciones.

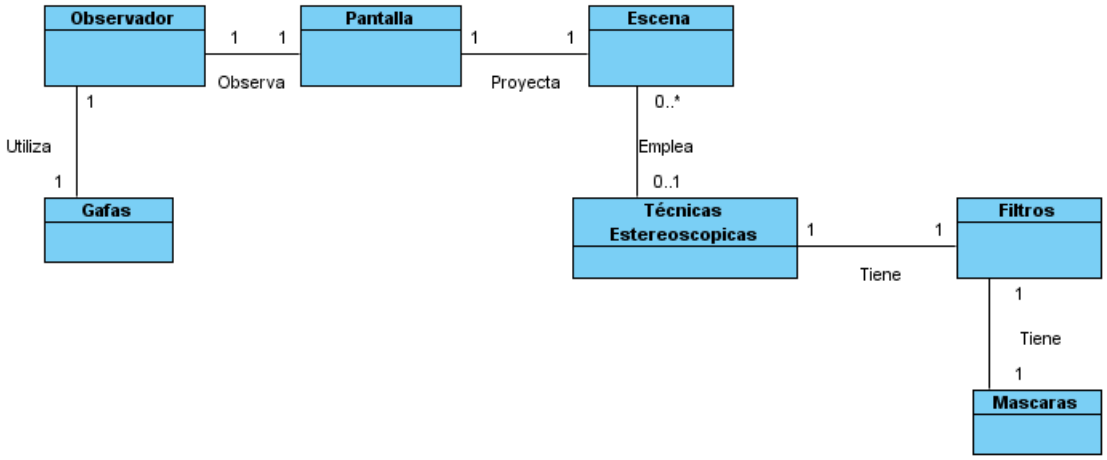

Figura 3.1: Modelo de dominio

3.1.3. Glosario de términos

Algunos de estos conceptos se mencionan en los capítulos anteriores, por lo que a continuación se brinda una breve explicación de lo que representa cada una de estas clases conceptuales.

Observador: Representa a la persona que observa la escena.

Gafas: Dispositivo visual que permite al observador percibir la imagen correspondiente a cada uno de los ojos.

Pantalla: Superficie donde se proyecta la escena, puede ser un monitor convencional o una pantalla de proyección.

Escena: Compuesto por los elementos que se visualizan en la pantalla, la cual puede ser de proyectada de forma monocular o estereoscópica.

Técnicas estereoscópicas: Técnicas para visualizar la escena de manera estereoscópica entre las que se encuentran: anaglifo, visión paralela y cruzada, entrelazado, *alterned page* y *quad buffer*.

Filtros: Definen como y donde se dibujan el contenido de la escena en la pantalla.

Máscara: Restringe de qué forma se va a dibujar la escena.

3.1.4. Captura de requisitos

El módulo estereoscópico tiene que cumplir una serie de condiciones y capacidades para dar solución al problema de brindar las funcionalidades estereoscópicas a la herramienta STK. Estas condiciones y funcionalidades son los requisitos funcionales y no funcionales del sistema y se definen a continuación.

Requisitos funcionales

Los requisitos funcionales que debe cumplir el sistema son:

1- Definir atributos del observador.

1.1- Definir posición.

1.2- Definir orientación.

1.3- Definir distancia interaxial.

2.1- Modificar atributos del observador.

2.1- Modificar posición del observador.

2.2- Modificar orientación del observador.

2.3- Modificar distancia interaxial.

3- Definir atributos del *display*

3.1- Definir posición.

3.2- Definir resolución de la pantalla.

3.3- Definir tamaño de la pantalla en mm.

3.4- Definir posición del *viewport*.

3.5- Definir vector perpendicular.

3.6- Definir vector horizontal.

4- Aplicar técnica de visualización estereoscópica.

4.1 - Aplicar técnica anaglifo.

4.2 - Aplicar técnica visión paralela.

4.3 - Aplicar técnica visión cruzada.

4.4 - Aplicar técnica entrelazado.

4.5 - Aplicar técnica *alterned image*.

4.6 - Aplicar técnica *quad buffer*.

Requisitos no funcionales

Los requisitos no funcionales que debe cumplir el sistema detectados a partir de la investigación realizada son:

Rendimiento:

Como aplicación de tiempo real, debe tener alto grado de velocidad de procesamiento o cálculo. Esta velocidad debe ser al menos 24 *frames* por segundo y preferiblemente de 30 a 60 *frames* por segundo.

Soporte:

El sistema debe ser multiplataforma, de tal forma que se pueda utilizar tanto en Windows como en Linux.

Usabilidad:

Los usuarios deben tener conocimientos básicos de programación gráfica. El módulo estereoscópico está concebido para ser reutilizable por aplicaciones finales que usen la STK como motor gráfico.

Legales:

Todas las bibliotecas que se utilizan son libres bajo licencias compatibles con la GPL.

Software:

Sistemas Operativos: Windows y GNU/Linux.

IDE: Sobre Windows Visual Studio 2008 y sobre Linux Code::Blocks 8.02

Bibliotecas y *API* gráficas: *OpenGl* y debe estar instalado la herramienta STK.

Hardware:

Requerimientos mínimos: Procesador Intel Pentium 4, 2.4 GHz, 1 GB de memoria RAM.

Para las técnicas *alterned image* y *quad buffer* es necesario un monitor o pantalla con frecuencia de refresco de 120 Hz, además específicamente para *quad buffer* se necesita tarjetas gráficas de la familia *NVIDIA Quadro*.

Diseño e implementación:

Debe ser implementada en el lenguaje C++ estándar. Se utiliza la biblioteca STK como motor gráfico para el dibujado. Se rige por la filosofía de Programación Orientada a Objetos.

3.1.5. Modelo de casos de uso del sistema

Los casos de uso del sistema se generan a partir del agrupamiento de las funcionalidades que deben ser brindadas por el módulo estereoscópico a la biblioteca STK.

Definición de actores del sistema

Los actores de un sistema son agentes externos, roles que las personas o dispositivos juegan cuando interactúan con el software. En este caso es la STK quien hará uso de la visualización estereoscópica de los contenidos de la escena, el actor del sistema será llamado **STK**.

Actor	Justificación
STK	Es quien se beneficia con la visualización estereoscópica de la escena mediante las seis técnicas que soporta el módulo, y con la configuración de los elementos geométricos.

Tabla 3.1: Actor del sistema.

Diagrama de caso de uso del sistema

En la figura 3.2 se puede apreciar cómo queda conformado el Diagrama de Caso de Uso donde se encuentran agrupados los requerimientos funcionales que debe brindar el sistema.

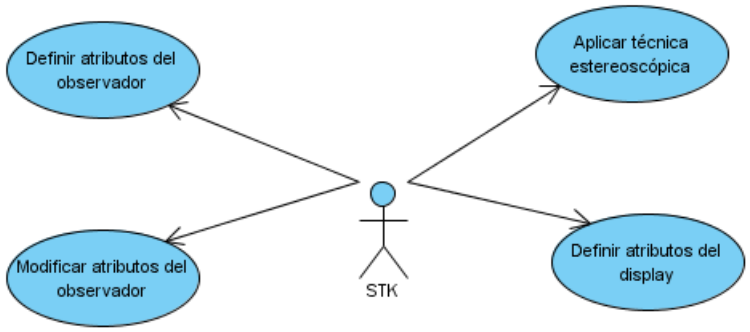

Figura 3.2: Diagrama de casos de usos del sistema

Descripción de casos de uso del sistema

Cada caso de uso tiene asociado una descripción de las funciones que ejecuta el sistema como respuesta a las acciones del usuario. Las tablas presentadas a continuación argumentan los flujos operacionales para cada caso de uso.

CU Seleccionar técnica de visualización estereoscópica.

Nombre del caso de uso	Seleccionar técnica de visualización estereoscópica.
Actores	STK
Propósito	Configurar la técnica de visualización estereoscópica activa.
Resumen: El caso de uso inicia cuando se activa una técnica de visualización estereoscópica.	
Referencias	RF- 16, RF- 17, RF- 18, RF- 19, RF- 20, RF- 21, RF- 22
Precondiciones	Se debe haber ejecutado anteriormente los CU1 “Definir atributos del *display*” y CU2 “Definir atributos del observador”
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- Se activa una de las técnicas de visualización estereoscópicas.	1.1- Se configura la técnica que se encuentre activa entre anaglifo, visión paralela, visión cruzada, entrelazado, *alterned image* y *quad buffer*. 1.2- Se configura el punto de mira, volumen de visualización y *viewport* de la cámara izquierda. 1.3- Se activa el filtro. 1.4- Se cambia el ojo activo. 1.5- Se desactiva el filtro. 1.6- Se repite el proceso del dos al seis para la cámara derecha.
Postcondiciones:	Se configura la técnica estereoscópica activa.
Prioridad:	Crítica.

Tabla 3.2: Caso de uso Seleccionar técnica de visualización estereoscópica.

CU Definir valores del observador

Nombre del caso de uso	Definir valores del observador
Actores	STK
Propósito	Establecer los valores de los atributos del observador
Resumen: El actor debe proporcionar la posición y orientación de la cabeza y distancia interaxial.	
Referencias	RF-1, RF-2, RF-3, RF-4
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- Proporciona los valores correspondientes a la posición, rotación de la cabeza y distancia interaxial.	1.1- Se almacena la posición, rotación de la cabeza y distancia interaxial. 1.2- Se crea el observador con los parámetros proporcionados.
Postcondiciones:	Quedan definidos los atributos del observador.
Prioridad:	Crítica.

Tabla 3.3: Caso de uso Definir valores del observador.

CU Modificar atributos del observador.

Nombre del caso de uso	Modificar atributos del observador
Actores	STK
Propósito	Modificar el valor de la posición, orientación o distancia interaxial del observador.
Resumen: El caso de uso inicia cuando se desea modificar la posición , orientación o distancia interaxial del observador	
Referencias	RF-5, RF-6, RF-7, RF-8
Precondiciones	Se debe haber ejecutado el CU2 “Definir atributos del observador”.
Curso normal de los eventos	

Acción del actor	Respuesta del sistema
1- Si desea modificar la posición, ir a la sección “Mover”. a) Si desea modificar la orientación, ir a la sección “Rotar”. b) Si desea modificar la distancia interaxial, ir a la sección “Interaxial”.	
Sección: “Mover”	
Acción del actor	Respuesta del sistema
1- Proporciona el eje y la distancia que se mueve.	1.1- Se modifica el valor de la posición teniendo en cuenta el valor proporcionado por el usuario.
Sección: “Rotar”	
Acción del actor	Respuesta del sistema
1- Proporciona el eje y el ángulo de rotación.	1.1- Se modifica el valor de la orientación teniendo en cuenta el valor proporcionado por el usuario.
Sección: “Interaxial”	
Acción del actor	Respuesta del sistema
1- Proporciona el valor de distancia interaxial.	1.1- Se modifica el valor existente sumando al valor proporcionado por el usuario.
Postcondiciones:	Quedan actualizados los valores modificados del observador.
Prioridad:	Normal

Tabla 3.4: Caso de uso Modificar atributos del observador.

CU Definir valores del *display*

Nombre del caso de uso	Definir valores del *display*
Actores	STK
Propósito	Establecer los valores de los atributos del *display*
Resumen: El actor debe proporcionar la posición, resolución y tamaño de la pantalla, además de la posición del *viewport* y los vectores perpendicular y horizontal.	
Referencias	RF-9, RF-10, RF-11, RF-12, RF-13, RF-14, RF-15
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1.1- Proporciona los valores correspondientes a los vectores posición, perpendicular y horizontal, resolución, tamaño en mm de la pantalla y posición del *viewport*.	1.2- Se almacena la posición, resolución y tamaño de la pantalla, así como los vectores perpendicular y horizontal y la posición del *viewport*. 1.3- Se crea el *display* con los parámetros proporcionados.
Postcondiciones:	Quedan definidos los atributos del *display*.
Prioridad:	Crítica.

Tabla 3.5: Caso de uso Definir valores del *display*.

3.2. Diseño del sistema

A continuación se define el Diagrama de Clases del Sistema y los Diagramas de Secuencia para cada Caso de Uso.

3.2.1. Diagrama de Clases del Diseño del Sistema

El Diagrama de Clases del Diseño muestra cómo queda compuesta la estructura de clases a emplear en la implementación del sistema. En la figura 3.3 se aprecia la estructura de clases del módulo estereoscópico.

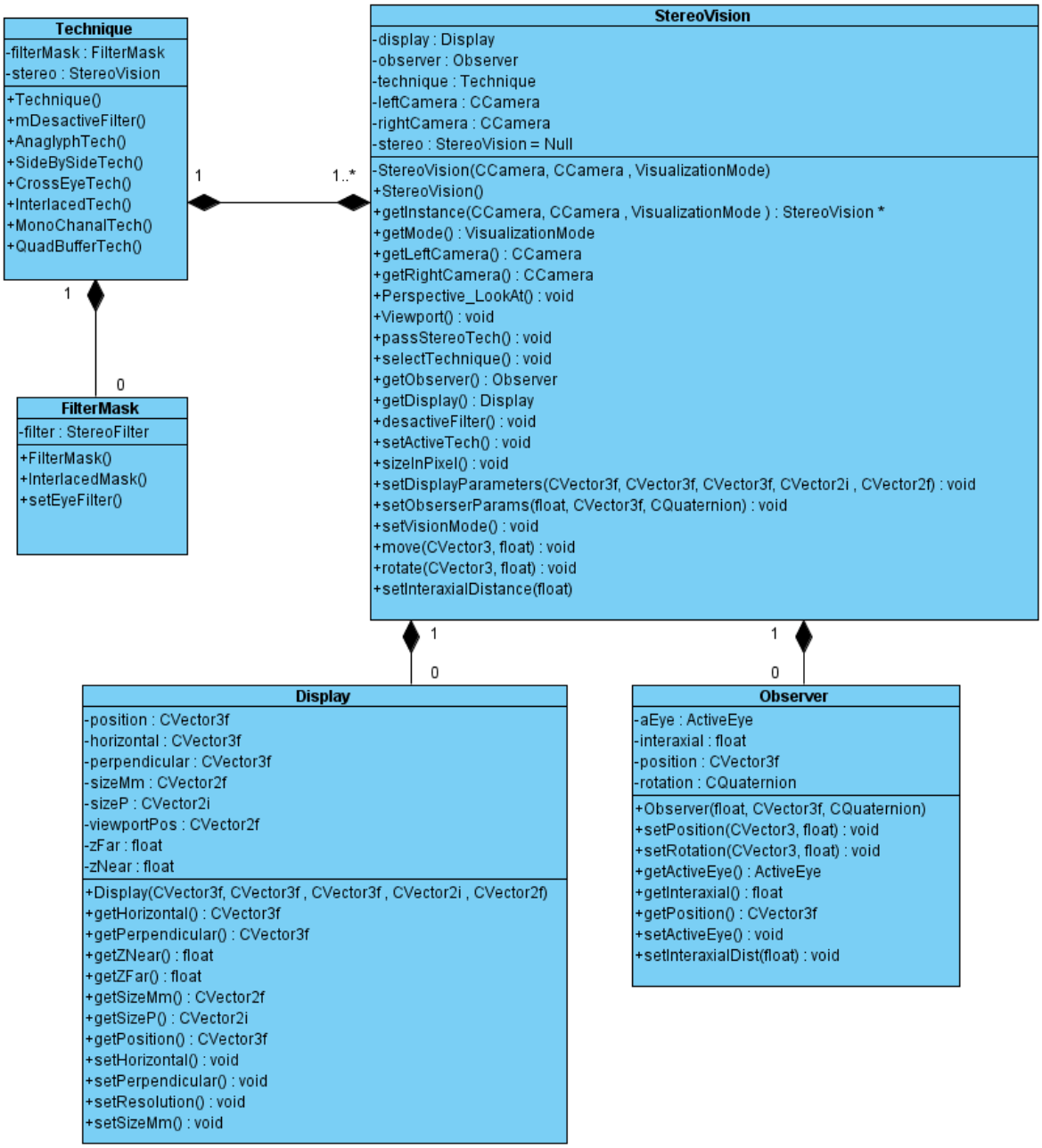

Figura 3.3: Diagrama de Clases del Diseño del Sistema

3.2.2. Diagramas de Secuencia del Diseño

Los Diagramas de Secuencia del Diseño permiten modelar a través de mensajes la interacción entre objetos de un sistema. A continuación se presentan los diagramas de secuencia para cada

caso de uso.

DS del Diseño del CU Definir atributos del observador, ver figura 3.4.

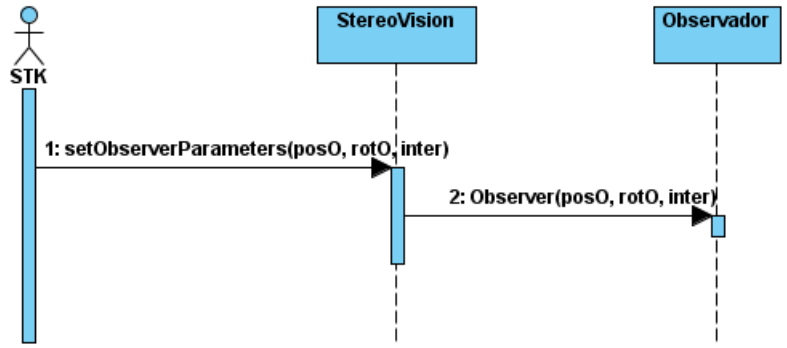

Figura 3.4: Diagrama de Secuencia del CU Definir atributos del observador

DS del Diseño del CU Modificar atributos del observador, ver figura 3.5.

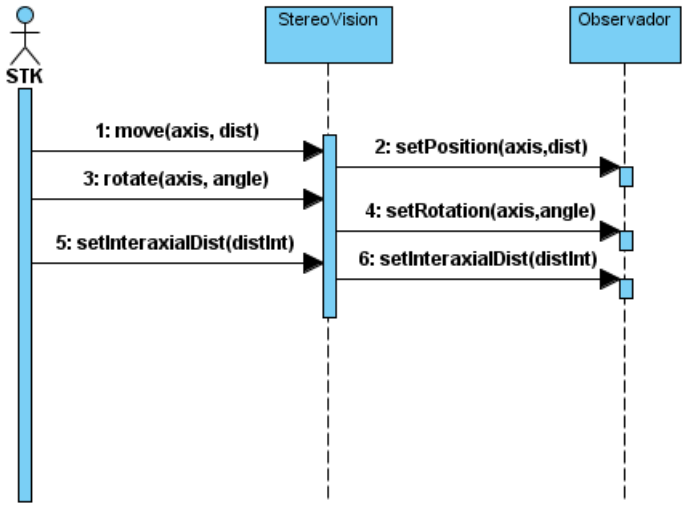

Figura 3.5: Diagrama de Secuencia del CU Modificar atributos del observador

DS del Diseño del CU Definir atributos del *display*, ver figura 3.6.

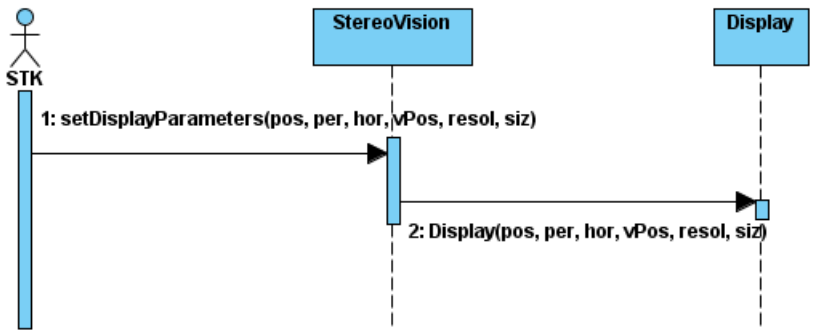

Figura 3.6: Diagrama de Secuencia del CU Definir atributos del *display*

DS del Diseño del CU Seleccionar técnica de visualización estereoscópica, ver figura 3.7.

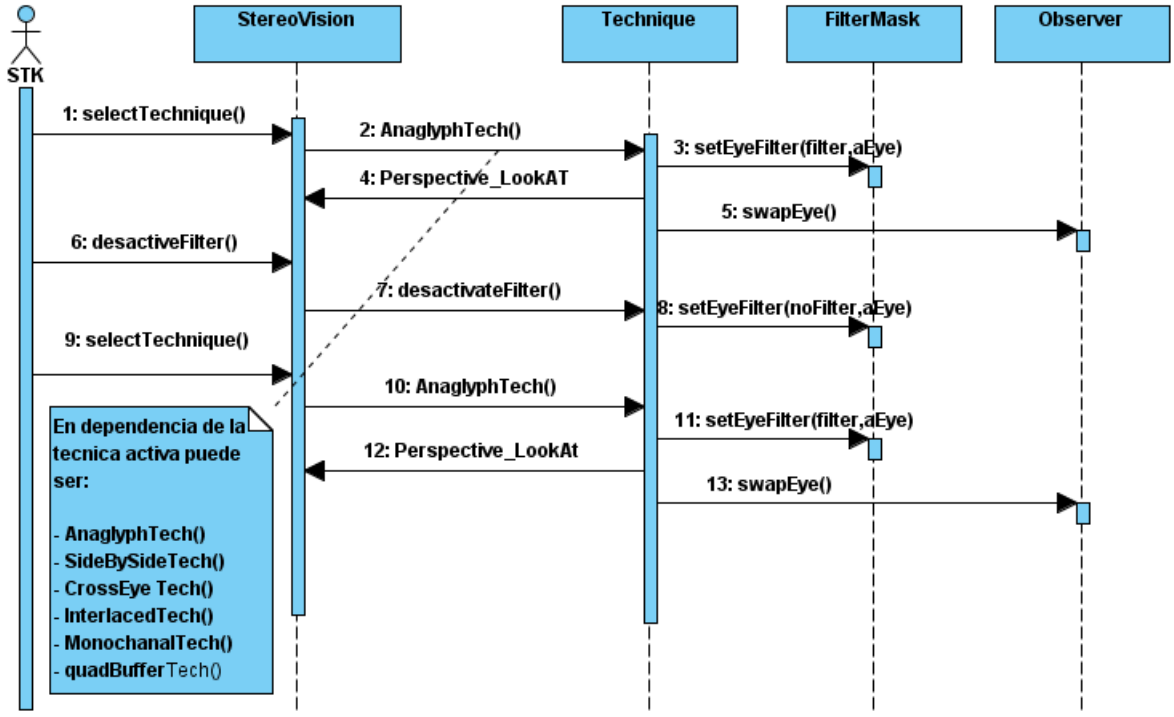

Figura 3.7: Diagrama de Secuencia del CU Seleccionar técnica de visualización estereoscópica

3.3. Implementación del sistema

En este epígrafe se presenta el Diagrama de Componentes del sistema y los patrones de diseños utilizados en la solución propuesta.

3.3.1. Diagrama de Componentes

Las clases resultantes del análisis y el diseño se hacen físicas mediante componentes, estos son los archivos que forman la aplicación. En la figura 3.8 se puede apreciar los archivos que componen el modulo estereoscópico.

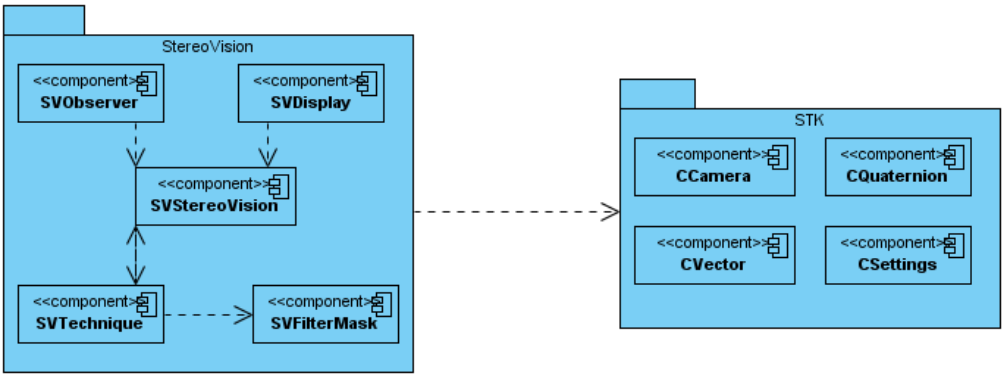

Figura 3.8: Diagrama de Componentes

3.3.2. Patrones de diseño

Un patrón de diseño nombra, abstrae e identifica los aspectos claves de un diseño estructurado común, que lo hace útil para la creación de diseños orientados a objetos reutilizables [Larman, 2004].

En la solución propuesta se utilizan los patrones *singleton*, creador, bajo acoplamiento y alta cohesión.

Singleton: Garantiza que exista una instancia única para la clase *StereoVision* y proporciona un punto de acceso global a ella. A partir de esta instancia se podrán controlar más fácilmente el acceso al resto de las clases que tienen responsabilidades bien definidas.

Creador: El patrón creador es un patrón GRASP que guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se conecte con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. En este caso la clase *StereoVision* es la responsable de la creación del observador y del *display*.

Bajo Acoplamiento: Este patrón estimula asignar una responsabilidad, de modo que su colocación no incremente el acoplamiento produciendo resultados negativos propios de un alto acoplamiento. El bajo acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios.

Alta Cohesión: Se da una alta cohesión cuando los elementos de un componente, por ejemplo una clase, colaboran para producir algún comportamiento bien definido. Una clase de alta cohesión posee un número relativamente pequeño de responsabilidades, con una importante funcionalidad relacionada y poco trabajo por hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande. Este patrón mejora la calidad y facilidad con que se puede entender el diseño, se genera un bajo acoplamiento y permite fomentar la reutilización.

En el diseño de clases propuesto se hace uso de los patrones bajo acoplamiento y alta cohesión ya que las clases son independientes y con responsabilidades bien definidas y los objetos colaboran para realizar las tareas más complejas.

Capítulo 4

Valoración de resultados

En este capítulo se ofrece una valoración del cumplimiento de los objetivos y de los resultados obtenidos con la culminación del módulo estereoscópico, además se realizan pruebas que permiten validar la precisión del módulo al proyectar la paralaje sobre la pantalla de proyección.

4.1. Resultados obtenidos

A continuación se muestran a través de imágenes los resultados alcanzados con la realización de este módulo.

Se logra un módulo capaz de incorporar la visualización estereoscópica de los contenidos de la escena a la biblioteca STK, contiene la implementación de seis de las técnicas estereoscópicas existentes y permite la definición de los elementos geométricos que componen la escena estereoscópica, como el observador, el *display* y las cámaras que representan los ojos del observador. Además se puede modificar los parámetros del observador en tiempo de ejecución, en caso de que la persona desee moverse, girar la cabeza o modificar la distancia interaxial se mantenga la relación 1:1 con la aplicación gráfica, al actualizar estos parámetros se calcula nuevamente la relación geométrica entre todos los elementos y se dibuja la escena consecuentemente con los cambios realizados.

La integración con la biblioteca STK fue sencilla y cómoda, posibilito el uso de las funciones de

OpenGl para la configuración de las técnicas estereoscópicas y solo fue necesario utilizar de la biblioteca las funciones para la configuración del *frustum*, *lookat* y *viewport* de las cámaras.

En la imagen 4.1 se muestran los valores de configuración utilizados en el demo. Se puede apreciar la posición que ocupan los elementos geométricos de la escena: el observador, cada una de las cámaras, el *display*, además el tamaño y resolución de pantalla y la distancia del observador a la pantalla. Además se encuentran los valores finales de los planos *left*, *right*, *top* y *bottom* que definen el volumen asimétrico de visualización, así como los valores utilizados para obtener este resultado.

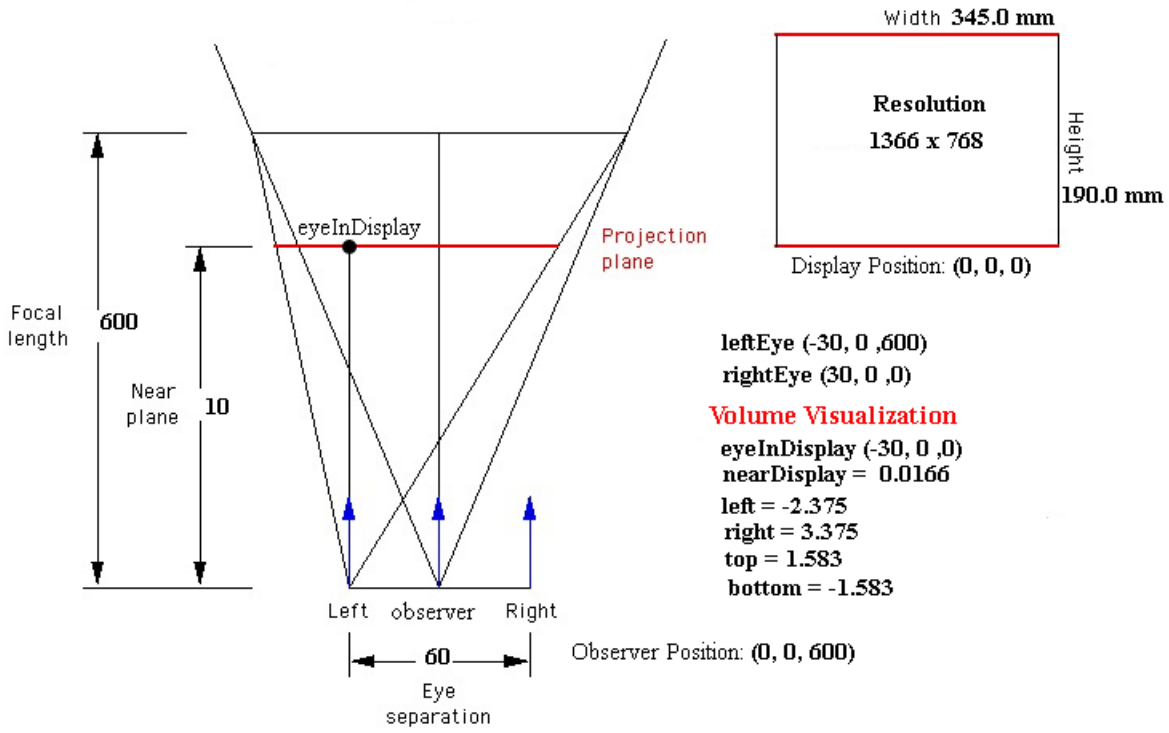

Figura 4.1: Configuración de los elementos geométricos empleada en el demo.

A continuación se presentan imágenes donde se puede percibir los resultados alcanzados teniendo en cuenta la configuración explicada anteriormente. En la imagen 4.2 se puede apreciar la escena dibujada usando la técnica anaglifo.

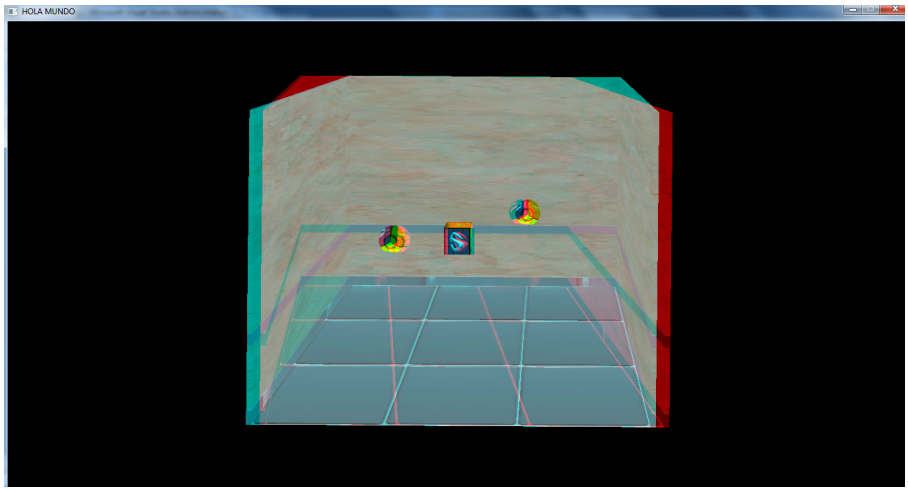

Figura 4.2: Escena dibujada usando la técnica anaglifo

En la imagen 4.3 se puede apreciar la escena dibujada usando la técnica visión paralela.

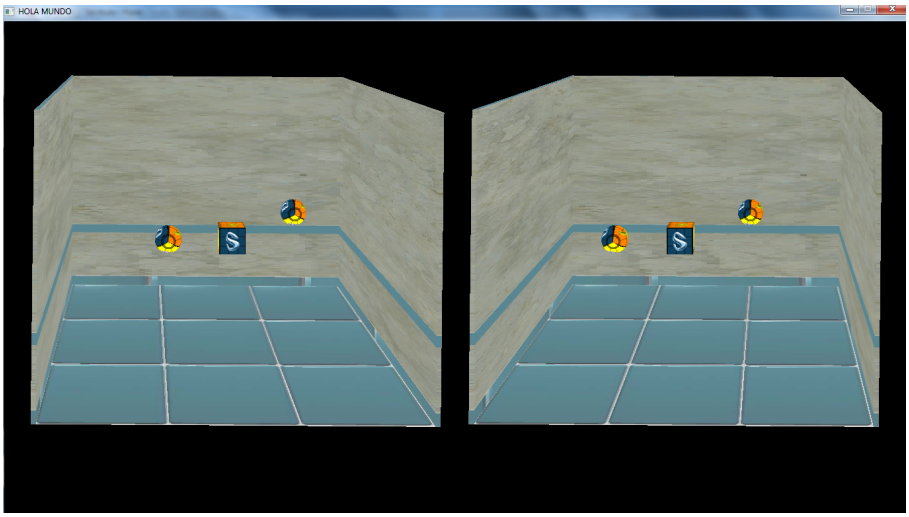

Figura 4.3: Escena dibujada usando la técnica visión paralela

En la imagen 4.4 se puede apreciar la escena dibujada usando la técnica visión cruzada.

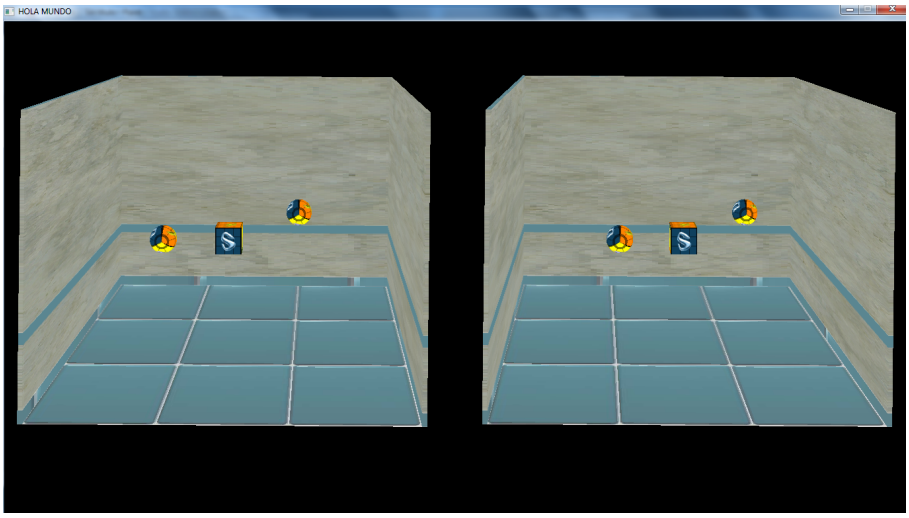

Figura 4.4: Escena dibujada usando la técnica visión cruzada

En la imagen 4.5 se puede apreciar la escena dibujada usando la técnica entrelazado.

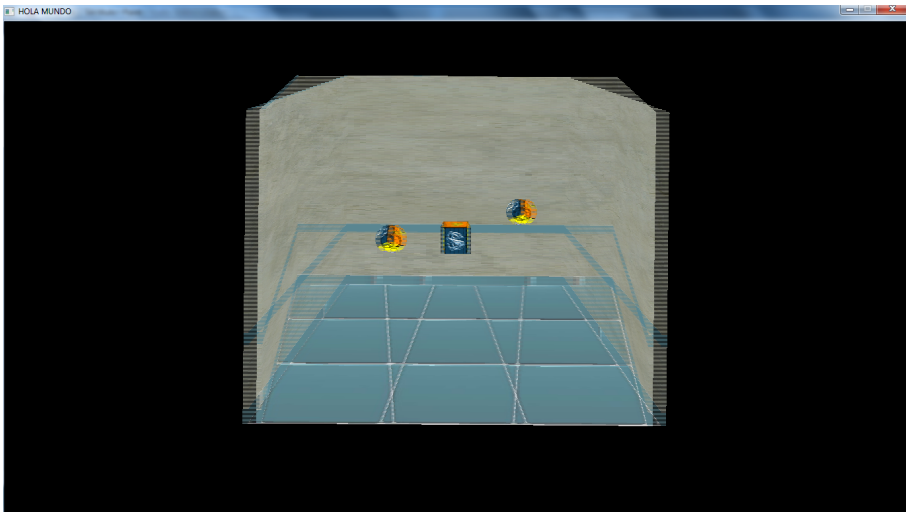

Figura 4.5: Escena dibujada usando la técnica entrelazado

En el caso de las técnicas *alterned image* y *quad buffer* el resultado no puede ser representado ya que las imágenes correspondientes a cada ojo no pueden ser captadas simultáneamente.

Por las características con que se desarrolla el módulo estereoscópico se realiza una versión genérica que permite la integración del módulo a cualquier sistema, el cual se debe encargar

del dibujado de la escena como se explica en el capítulo 2. Como valor agregado el módulo se integra con el software *VISMEDIC* creado en el CEDIN encargado de la visualización de imágenes médicas. A continuación se presenta una imagen donde se puede percibir los resultados alcanzados con la técnica de visión paralela.

En la imagen 4.6 se puede apreciar un volumen cargado por el software *VISMEDIC* dibujado usando la técnica visión paralela.

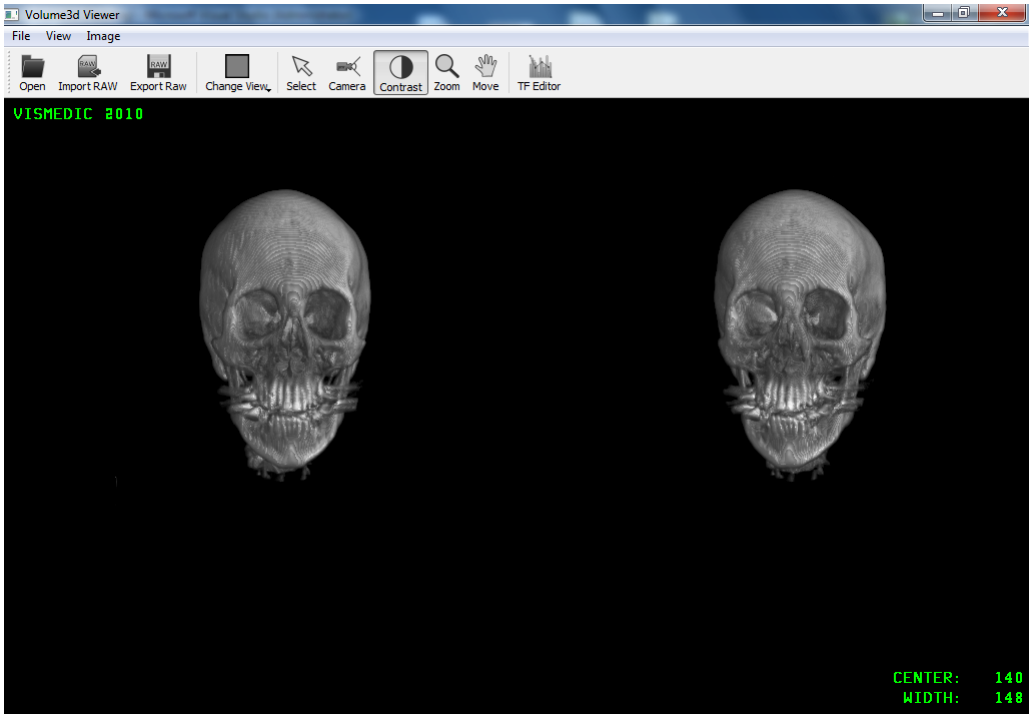

Figura 4.6: Volumen dibujada usando la técnica visión paralela con la herramienta *VISMEDIC*

4.2. Validación de resultados

Se realizaron pruebas al sistema para validar la precisión al proyectar las imágenes estereoscópicas para cada uno de los ojos. Conociendo la distancia interaxial, la distancia del observador a la pantalla y la distancia a la que se encuentra el observador del objeto es posible calcular la

paralaje que se debe proyectar en pantalla utilizando la ecuación 4.1 .

$$paralaje = distInteraxial * \frac{distPant - distObj}{distObj} \tag{4.1}$$

En la siguiente tabla 4.2 se presentan los valores de paralaje obtenidos para objetos situados en distintas posiciones en la escena como se muestra en la imagen 4.7.

Objetos	Tamaño en mm	Resolución	Distancia Interpupilar	Distancia a la pantalla	Distancia al objeto	Paralaje esperado(mm)	Paralaje obtenido(mm)
Esfera 1	345.0 x 190.0	1366 x 768	60	600	680	-7.06	-7.03
Esfera 2	345.0 x 190.0	1366 x 768	60	600	550	5.45	5.50
Caja	345.0 x 190.0	1366 x 768	60	600	600	0	0.01
Esfera 3	345.0 x 190.0	1366 x 768	60	600	580	2.08	2.04
Esfera 4	345.0 x 190.0	1366 x 768	60	600	630	-2.86	-2.88

Tabla 4.1: Paralajes obtenidos para los objetos de una escena.

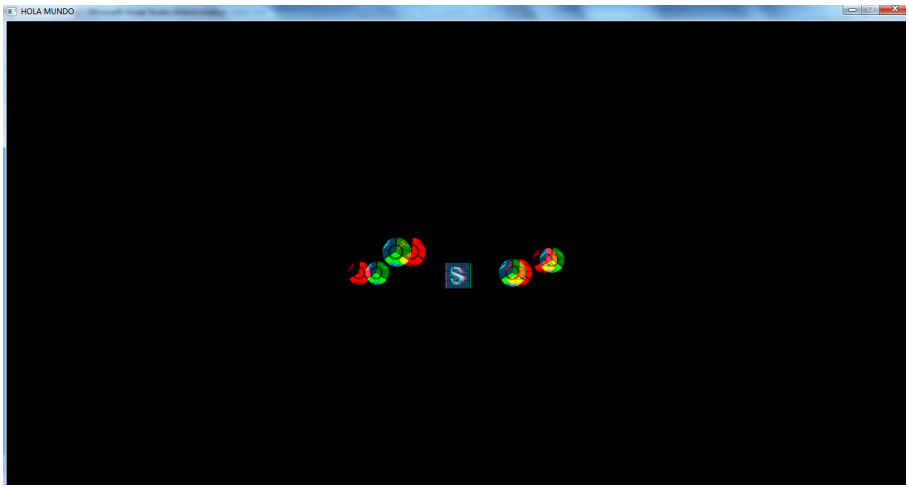

Figura 4.7: Objetos posicionados en la escena para validar los valores de paralaje.

Los valores obtenidos permiten calcular la media de error absoluto y relativo que introduce la aplicación al proyectar la paralaje en pantalla. El error absoluto es igual a Δ E(x) = 0.03 y el error relativo Δ e(x) = 0.01 lo cual da un por ciento de error de un 1 %.

Conclusiones

Con la realizacion de este trabajo se concluye:

- Se logró un módulo que permite la visualización estereoscópica de la escena para la biblioteca STK mediante las técnicas: anaglifo, visión paralela, visión cruzada, entrelazado, *alterned image* y *quad buffer*, y la configuración de los elementos geométricos que componen una escena estereoscópica.
- Se realizó una versión genérica que permite la visualización estereoscópica de la escena, para lo que es necesario la utilización de un motor de dibujado.
- Se integró el módulo a la biblioteca STK y al software VISMEDIC satisfactoriamente, brindando en ambos casos un mayor nivel de profundidad en la escena.

Recomendaciones

- Añadir al módulo estereoscópico un puntero 3D para permitir a los usuarios moverse en profundidad por la escena.
- Incorporar el módulo de estimación de pose para obtener la posición y rotación del usuario de manera dinámica.

Referencias bibliográficas

[NVi, 2010] (2010). Nvidia 3d vision. http://www.nvidia.com/object/3D-Vision-Main.html.

[Mon, 2010] (2010). Realidad virtual. http://www.monografias.com/trabajos11/realitual/realitual.shtml.

[Coon, 2005] Coon, D. (2005). *Fundamentos de Psicología*. International Thonson Editores, 10ma edition.

[Dijkerman et al., 1999] Dijkerman, C., Milner, A. D., and Carey, D. P. (1999). *Motion parallax enables depth processing for action in a visual form agnostic when binocular vision is unavailable. Neuropsychologia*. Number 37(13):1505–1510.

[García, 2008] García, I. N. R. (2008). Sistema de visión estereoscópica basado en anaglifo para aplicaciones de realidad virtual. Master's thesis, Instituto Politécnico Nacional, México D.F.

[GER, 1991] GER (1991). Biofísica ii. http://www.canalsocial.net/ger/ficha_GER.asp?id=5980&cat=fisica.

[Howard and Rogers, 1995] Howard, I. P. and Rogers, B. J. (1995). *Binocular Vision and Stereopsis*. Oxford University Press (OUP)., 1st edition.

[Larman, 2004] Larman, C. (2004). *UML y Patrones. Introducción al análisis y diseño orientado a objeto.* S.l.: Preason.

[Lipton, 1997a] Lipton, L. (1997a). *Stereo-Graphics Developers' Handbook*. StereoGraphics Corporation.

[Lipton, 1997b] Lipton, L. (1997b). Stereo-vision formats for video and computer graphics. In *Stereoscopic Displays and Virtual Reality Systems IV*, pages 239–244.

[Marqués, 2004] Marqués, L. A. (2004). La técnica estereoscópica, visión monocular. http://usuarios.arsystel.com/luismarques/documentacion/txt/02100_monocular.htm.

[Marrero, 2010] Marrero, E. (2010). Sensación y percepción. http://academic.uprm.edu/~eddiem/psic3001/id61.htm.

[Martín et al., 2004] Martín, S., Suárez, J., Rubio, R., and Gallego, R. (2004). Aplicación de los sistemas de visión estereoscópica en las enseñanzas técnicas. Technical report.

[Morris and Maisto, 2005] Morris, C. G. and Maisto, A. A. (2005). *Introducción a la psicología*. Pearson Educación, Mexico.

[Ogre, 2010] Ogre (2010). Ogre web site. http://www.ogre3d.org.

[OpenSceneGraph, 2010] OpenSceneGraph (2010). Openscenegraph web site. http://www.openscenegraph.org/projects/osg.

[Otero et al., 2003] Otero, A., Flores, J., Saco, P., and Arias, J. E. (2003). Diseño e implementacion de una experiencia audiovisual colectiva inmersiva con fines educacionales. *Instituto de Investigaciones Tecnológicas, Universidad de Santiago de Compostela.*

[Pfautz, 2002] Pfautz, J. D. (2002). *Depth perception in computer graphics*. University of Cambridge.

[Pocock and Richards, 2005] Pocock, G. and Richards, C. D. (2005). *Fisiología Humana: La base de la Medicina:*. MASSON SA, 2da edition.

[Quintana and Alfonso, 2010] Quintana, A. and Alfonso, Y. (2010). Motor de render genérico. Ingeniero en ciencias informáticas, Universidad de Ciencias Informaticas, CEDIN. C. Habana.

[Quirós et al., 2004] Quirós, J. S., González, S. M., García, R. R., and Santos, R. G. (2004). *Desarrollo de un sistema de Visión Estereoscópica aplicado a los estudios geológicos.*

[Zambrano, 2008] Zambrano, M. M. (2008). Interacción con objetos deformables. Maestra en ciencias en la especialidad de computación, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional. Departamento de Computación. México D.F.

Acrónimos

API (Application Program Interface) *en español: Interfaz de Programación de Aplicaciones*

RV Realidad Virtual

VE Visión Estereoscópica

CAD *Diseño asistido por computadoras (traducido de las siglas en inglés*: Computer-aided design*)*

UCI Universidad de las Ciencias Informáticas

CEDIN Centro de Informática Industrial

STK SceneToolKit

OSG OpenSceneGraph

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

ZPS *Ajuste de cero paralaje (traducido de las siglas en inglés* zero parallax settings)

Ogre Object-oriented Graphics Rendering Engine

RUP *El Proceso Unificado Racional* (Rational Unified Process *en inglés)*

UML *Lenguaje Unificado de Modelado* (Unified Modeling Language *en inglés)*