# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

**TESIS**

APLICACIÓN GRÁFICOS DE COMPUTADORA 3D Y LA REPRESENTACIÓN ESTEREO 3D (s3D) DE NANOESTRUCTURAS

QUE PARA OBTENER EL TITULO DE:

**INGENIERO EN COMPUTACIÓN**

PRESENTAN:

**JARAMILLO ORTIZ FABIAN**

**MENDOZA ESPINOSA JOSÉ LUIS**


DIRECTOR: DR. JORGE ALFONSO GARCÍA MACEDO

CIUDAD UNIVERSITARIA: 27 / NOVIEMBRE / 2013.

**Agradecimientos**

**Fabian Jaramillo Ortiz**

*A Dios por haberme acompañado y guiado a lo largo de mi carrera, por ser mi fortaleza en los momentos de debilidad y por brindarme una vida llena de aprendizajes, experiencias, por la vida y la familia con la que me bendijo…*

*A mis padres por ser los pilares de mi vida y por brindarme todo su apoyo para alcanzar mis metas. A mi madre Alejandra Concepción Ortiz Nolasco por su amor, confianza y cuidados incondicionales. A mi padre Ramón Jaramillo Luna por darme su cariño y libertad. A mi abuela Guillermina Luna Parra por su apoyo y cuidados. A mi hermana Laura por el apoyo y la fortaleza.*

*A Claudia Gabriela Vela García por brindarme su apoyo incondicional en los momentos difíciles.*

*A la UNAM y a la Facultad de Ingeniería a la que me honra pertenecer y me abrió las puertas a una infinidad de conocimientos.*

*A mis compañeros y amigos que hasta el día de hoy están conmigo y los que ya no están, por sus consejos y entusiasmo, gracias por haber hecho de este camino una experiencia inolvidable.*

*A los proyectos que dieron lugar a esta investigación Conacyt # 179607, PAPIIT IN # 111413 y UCMEXUS # CN-10-457*

*Al Dr. Jorge A. García Macedo por ser el tutor de este trabajo y darme la oportunidad de iniciar mi camino en su laboratorio.*

**José Luis Mendoza Espinosa**

*A Dios por bendecirme e iluminar mi camino en todo momento…*

*A mi familia por brindarme su apoyo incondicional y ayudarme a alcanzar mis metas. A mi madre Olga Espinosa por su fortaleza y cariño que siempre guío e impulsó mis pasos. A mi padre José Luis Mendoza por sus enseñanzas y cariño. A mis hermanos Olga y Adolfo por brindarme su apoyo incondicional. A mi hermana Paola porque siempre estuvo presente para ayudarme en mis problemas y estudios. A todos los quiero mucho.*

*A Karina Bautista Negrete por su cariño y brindarme su apoyo incondicional.*

*A la Universidad Nacional Autónoma de México que me abrió sus puertas y me dio la oportunidad de crecer académica y personalmente dentro de sus aulas*

*A mis compañeros, amigos y todos aquellos que me brindaron su apoyo y me permitieron convivir con ellos.*

*A los proyectos que dieron lugar a esta investigación Conacyt # 179607, PAPIIT IN # 111413 y UCMEXUS # CN-10-457*

*Al Dr. Jorge A. García Macedo por su invaluable apoyo y dirección en este trabajo de tesis y abrirme las puertas de su laboratorio.*

## Índice

## Índice de Imagenes

## Introducción

Este trabajo se realizó con la finalidad de brindar una información concisa sobre conceptos básicos, así como términos técnicos de los sistemas utilizados para el diseño y presentación de contenido multimedia estereoscópico.

En este trabajo se presenta un resumen del software y las técnicas para lograr un efecto estereoscópico confortable. Se ilustraron los resultados obtenidos con imágenes anaglíficas impresas en rojo y azul de forma superpuestas, que permiten apreciar objetos tridimensionales. Por lo cual, para un claro entendimiento de este trabajo, se entrega impreso a color y con lentes anaglíficos 3D.

Nos enfocamos en la aplicación de  Gráficos de Computadora CG 3D y en la novedosa proyección estereoscópica de contenido multimedia 3D (s3D) para la realización de videos de alta definición Full HD (1080p).

Los modelos generados en esta tesis corresponden a nanoestructuras originadas por el tensoactivo SDS (dodecil sulfato de sodio). El patrón de autoensamble de las nanoestructuras se basó en esquemas teóricos y resultados experimentales obtenidos por el Doctor Jorge Alfonso García Macedo. El Dr. Macedo es investigador y responsable del laboratorio de Fotónica de Geles del Instituto de Física de la Universidad Nacional Autónoma de México.

**Objetivo**

Desarrollar una representación tridimensional multimedia en s3D de nanoestructuras obtenidas en el laboratorio de Fotónica de Geles y cuya concepción proviene de esquemas teóricos y de resultados experimentales. El resultado de este trabajo debe ser una representación clara de las nanoestructuras de SDS, así como del mecanismo por el cual se generan.

Con esto mostraremos que gracias a los grandes avances de la computación es posible tener una poderosa herramienta para exhibir los desarrollos nanotecnológicos que se están haciendo, y que son muy difíciles de interpretar o tratar de explicar con palabras o esquemas.

# Capítulo 1. Surgimiento del 3D

En este capítulo nos enfocamos a dar una explicación breve del surgimiento de la tercera dimensión (3D) y su representación estereoscópica (s3D), mencionando las diferencias entre ellas. También se resaltan las diferentes herramientas que existen para visualización. Así mismo, se revisan algunos de los métodos más utilizados para simular el efecto de profundidad, destacando el impacto que esto tiene en el medio social y en la ciencia.

## 1.1 Breve historia del 3D estereoscópico

Históricamente el cine fue la base fundamental de la proyección estereoscópica, ahí se desarrollaron los primeros métodos y tecnologías para generarla a gran escala.

En el año de 1838 Charles Wheatstone creó el estereógrafo dando así un nuevo enfoque a la forma de ver las películas en esa época. A partir de ello surgieron varias estrategias para crear imágenes en movimiento con la ilusión de profundidad.

El primer sistema de cine en 3D que se patentó lo realizó William Freese-Greene en 1890. En 1900, Frederick Eugene Ives patentó una cámara con dos lentes pero no tuvo ninguna repercusión práctica. Lo mismo les sucedió a Edwin S. Porter y William E. Waden cuyo trabajo se quedó en la fase de ensayo. Ellos presentaron al público quince años más tarde una separación de las imágenes basada en el color rojo y verde, donde cada color era filtrado para solo uno de los ojos, gracias a unas gafas con cristales rojo y verde.

No fue hasta el 27 de septiembre de 1922 cuando llegó la primera película en 3D a las salas comerciales de Los Ángeles: The power of Love. Para lograr el efecto tridimensional, el productor Harry K. Fairall y el camarógrafo Robert F. Elder utilizaron la doble proyección a partir de dos tiras de película separando, de nuevo, las imágenes mediante los colores rojo y verde. La película no tuvo éxito pero fue el inicio de un interés real por la cinematografía en 3D.

A principios de los años 50 llegó lo que se llamaría época dorada del 3D con Arch Oboler quién crearía el primer largometraje estereoscópico en color, a partir de dos tiras y filtros Polaroid: la película “Bwana Devil”; llegó a ser un éxito en taquillas y el principio del auge para el 3D. Sin embargo, a pesar de los éxitos de esta llamada época dorada del 3D, en 1953 se detuvo su desarrollo. Hubo muchos motivos por los que decayó el formato; algunos fueron técnicos, como el complejo mecanismo que mantenía las dos tiras de película avanzando de manera sincronizada, así como errores en el proceso que llegaban a provocar dolores de cabeza, náuseas y fatiga ocular en el espectador. Al mismo tiempo, se daban otros pequeños inconvenientes como por ejemplo, el no utilizar el aforo completo de las salas de proyección, debido a que se perdía el efecto tridimensional desde las butacas laterales, o el rechazo del espectador a ponerse unas incómodas gafas de cartón.

### 1.2 Surgimiento de la Computación Gráfica

Los gráficos por computadora surgieron a principios de los cincuenta en el Instituto Tecnológico de Masachussets, Cambridge, EU. En este instituto idearon conectar un osciloscopio a una computadora, de modo que ésta podía controlar directamente la posición del haz catódico. Este sistema podía realizar dibujos simples, moviendo el haz de manera que fuese trazando las líneas que lo componían.

Este sistema de generación de imágenes es denominado VECTORIAL pues sus imágenes están compuestas por vectores unidos entre sí. Pronto se hizo evidente que el sistema no era práctico, así que se empezó a trabajar en una versión mejorada. El resultado fue el sistema RASTER SCAN. En este sistema no es el CPU el que controla el tubo catódico, sino que existe un sistema de circuitos independiente que realiza todo el trabajo. Esta vez, el haz no crea la imagen a partir de segmentos o vectores, sino que lo hace a partir de puntos (pixeles).

Estos dos métodos fueron usados al principio en 2D (que guarda relación con la pintura) y más tarde evolucionó hacia un sistema 3D. El término gráficos 3D por

computadora se refiere a trabajos de arte gráfico que fueron creados con ayuda de computadoras y programas especiales 3D. En general, el término puede referirse también al proceso de crear dichos gráficos, o el campo de estudio de técnicas y tecnología relacionadas con los gráficos 3D.

Durante la década de los 60, William Fetter trabajó para la empresa Boeing, en el diseño de ciertos modelos. Uno de sus proyectos consistió en construir "El hombre Boeing", que consistía en una representación tridimensional del cuerpo humano.

Fue entonces cuando Fetter acuñó el término "computación gráfica". Sin embargo su primer uso comercial no llegaría hasta 1976, cuando Ed Catmull y Fred Parke popularizó el uso de gráficos y animación por computadora en 3D y la usó en una película llamada "Futureworld". Desde entonces la técnica se ha utilizado en muchas películas y se ha convertido en un estándar en el cine, la televisión y los videojuegos.

### 1.3 Explicación del Término 3D

El término 3D en computación se refiere a la realización de imágenes generadas a través de una computadora, basadas en modelos 3D virtuales de objetos como la famosa tetera de agua de Utah. Figura 1.3.1

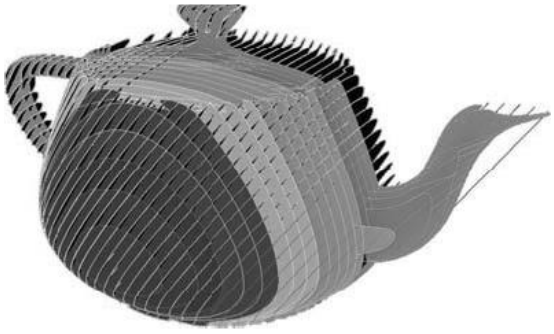


Figura. 1.3.1. Tetera de Utah

Los gráficos 3D se basan en una composición de vectores. En vez de que la computadora almacene la información sobre puntos, líneas y curvas en un plano

bidimensionales, se almacena la posición de puntos, líneas y típicas caras para construir un polígono en un espacio de tres dimensiones.

El proceso de la creación de gráficos tridimensionales comienza con un grupo de fórmulas matemáticas y se convierte en un gráfico en 3D. Las fórmulas matemáticas describen objetos poligonales, tonalidades, texturas, sombras, reflejos, transparencias, translucidez, refracciones, iluminación, profundidad de campo, desenfoques por movimiento, ambiente, punto de vista, etc. Toda esa información constituye un modelo en 3D.

El proceso de transformación de un modelo en 3D hacia una imagen 3D es llamado renderización.

En la computación se utilizan los gráficos en 3D para crear animaciones, gráficos, películas, juegos, realidad virtual, diseño, etcétera. Como ejemplo de videojuegos tenemos “God of War” Figura. (1.3.2.) “Prince of Persia” Figura. (1.3.3), o “Toy Story” Figura (1.3.4). Ejemplo de películas es “La era de Hielo” Figura (1.3.5).

Figura.1.3.2. God of War

Figura. 1.3.3. Prince of Persia

Figura.1.3.4. Toy Story

Figura.1.3.5. La era de Hielo

Comúnmente cuando se realiza una proyección 3D en realidad solo se logra una proyección bidimensional carente de profundidad, por lo que esta información técnicamente está ausente y no se logran apreciar de manera adecuada las posiciones de los objetos en un espacio determinado.

### 1.4 Diferencias entre 3D y s3D

Mientras que el término 3D es usado para la generación de gráficos por computadora y la realización de animaciones, el término s3D es usado para especificar la estereoscopía ya sea en una imagen o en una animación; haciendo uso de gafas para resaltar el efecto de profundidad.

### 1.5 Tipos de Gafas estereoscópicas

Existen dos clasificaciones para las gafas las cuales son:

- Activas: Utilizan obturadores LCD para impedir la visión de uno de los ojos, mientras se muestra al otro ojo la imagen que le corresponde. El sistema usa un sensor infrarrojo en las propias gafas que se sincroniza con un emisor infrarrojo que está junto a la pantalla de proyección (Figura 1.5.1).

Para funcionar, necesitan energía y además son mucho más caras que las gafas pasivas. Sin embargo, la calidad de visión es de mayor calidad. Otro inconveniente es que se tienen que esterilizar después de su uso en los cines, lo que acarrea un costo adicional al exhibidor.

Figura. 1.5.1. Gafas activas.

- Pasivas: Son mucho más sencillas, y existen 2 tipos:

  - Anáglifos: En éstas gafas se usan los colores rojo-cian, el ojo cubierto por el filtro rojo ve las partes rojas de la imagen como blancas y las partes azules como oscuras, y el cerebro produce la adaptación de los colores.
    Por otro lado, el ojo cubierto por el filtro azul percibe el efecto opuesto; el cerebro fusiona las imágenes recibidas de cada ojo, y las interpreta como una imagen con profundidad. La desventaja es que los filtros de color solo filtran parcialmente cada imagen por lo que dejan pasar parte de la imagen opuesta y esto da lugar a bordes borrosos gruesos de los objetos (Figura 1.5.2).

Figura. 1.5.2. Gafas Rojo-Cian

  - Polarizadas: Usan filtros de polarización, usualmente circular. La imagen derecha se realiza con una polarización circular, por ejemplo derecha, y la izquierda con una polarización opuesta. De esta forma cada lente permite el paso de solamente la imagen con la polarización apropiada (Figura 1.5.3).

Figura. 1.5.3. Lentes Polarizados

Como la polarización nunca es del 100%, siempre se filtra un poco de imagen opuesta en cada ojo y de allí la pérdida de definición final que no es tan buena como la lograda con lentes activos. En la Figura (1.5.4) se muestra cómo se genera la polarización.

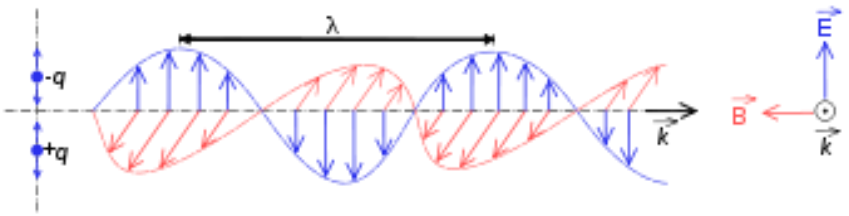



Figura.1.5.4.

La polarización se logra gracias al efecto de polarización electromagnética que es un fenómeno por el cual una onda oscila en un plano, definido por dos vectores, uno paralelo a la dirección de propagación y otro perpendicular. Seleccionado estos planos de vibración (polarización lineal) o provocando un giro de la polarización de la luz en un sentido (polarización circular) se obtiene luz polarizada. Esto se logra con materiales especiales que permiten controlar esta propiedad. Los materiales pueden ser estructuras contenidas en el material del lente, o películas depositadas sobre éste.

Como consecuencia podemos usar este efecto para fabricar filtros y separar la luz en dos componentes, dejando pasar solo una componente de la luz para cada filtro; (Figura 1.5.5).

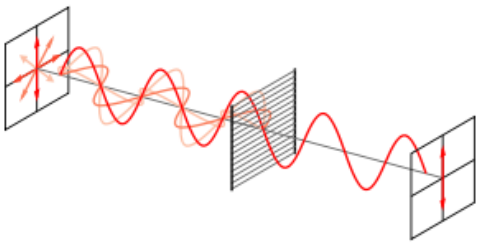


Figura.1.5.5.

Tenemos en la zona izquierda una onda compuesta de dos componentes ortogonales y tras pasar el filtro solo queda una componente.

Se pueden tener tres tipos de polarización pero en las gafas de este tipo se usan solo la lineal y la circular.

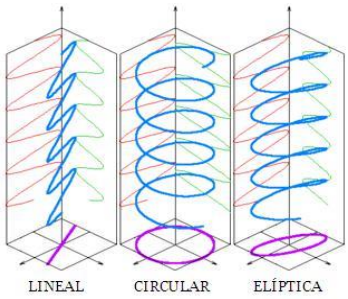



Figura. 1.5.6. Tipos de polarización

**Polarización Lineal:** presentan el problema de que al girar la cabeza las imágenes destinadas a cada ojo se comienzan a mezclar. Cuando giremos nuestra cabeza un mayor porcentaje de la imagen destinada al ojo derecho se verá en el ojo izquierdo y viceversa, hasta que si llegamos a 90º se invierten intercambiando las imágenes de cada ojo completamente. Esta es la tecnología que utilizan en los cines 3D IMAX (Figura 1.5.6).

**Polarización Circular:** éste método se ha comenzado a utilizarse más frecuentemente debido a que reduce las posibilidades de que las imágenes se cambien de ojo al mover nuestra cabeza inclinándola o rotándola (Figura 1.5.6).

**Polarización Elíptica:** éste tipo de polarización corresponde a cualquier otro caso diferente a lo anteriores, es decir, las componentes de polarización electromagnética no están en fase ni en contrafase (ángulos diferentes a 90° y 180°), por lo que no son adecuados para una polarización sencilla como los dos anteriores. (Figura 1.5.6).

## 1.6 Estereoscopia para el Cine y la Ciencia

Actualmente, cada existe una mayor demanda en el uso de la estereoscopía para generar publicidad, películas, videojuegos. Sin embargo, es el cine el que más ha empleado la tecnología 3D para crear principalmente películas infantiles.

A diferencia de las aplicaciones comerciales, en el campo de la ciencia no se ha explotado tanto esta tecnología. Con base en la filosofía de utilizar al máximo los recursos que nos da la computación gráfica nos planteamos un proyecto similar al que se tiene en la UNAM como el observatorio IXTLI.

El observatorio IXTLI es una sala especializada que se ha encargado de mostrar cómo esta tecnología puede ser muy útil en la ciencia ayudándonos a comprender sistemas complejos. Sin embargo, existe el inconveniente de que esta sala solo tiene capacidad para cuarenta personas, mismas que tienen que desplazarse a este lugar para poder estudiar algún material. Esto provoca que solo una pequeña población tenga acceso a este recurso (Figura 1.6.1).

Para poder desarrollar este material educativo y/o científico se debe tener en cuenta varios conceptos que se trataran en el capítulo dos de ésta tesis.

Figura. 1.6.1 Observatorio Ixtli en DGTIC UNAM

# Capítulo 2 Conceptos Básicos del s3D

En este capítulo hablaremos de los conceptos básicos sobre s3D y las herramientas para la generar este tipo de imágenes. Se explican brevemente términos y conceptos comunes que permitirán entender los resultados mostrados en el presente trabajo.

## 2.1 Herramientas que usamos para la generación del contenido multimedia

Nuestra primera meta fue hacer la representación de los esquemas teóricos y experimentales que se manejan en el laboratorio de Fotónica de Geles. Para alcanzar esta primera meta debíamos de representarlos en la computadora, razón por la que decidimos usar un software de modelado, el cual debería de cumplir con las expectativas del 3D. Se decidió usar Autodesk® Maya® 2011 por que éste software nos brinda la posibilidad de animar, jugar con las cámaras, renderizar, etcétera. Es un software muy potente que contiene muchas cualidades para hacer los esquemas teóricos y obtener un correcto modelo.

Sin embargo no sería suficiente con éste software, pues faltaría una etapa de post-producción para armar el video y además agregar pistas de audio, incluir códec de audio y video, y efectos en el mismo. Por eso nos apoyamos en el software Virtual dub para la generación de los videos e incluir un códec de video, y Sony Vegas Pro 11 que nos ayudaría a la renderización de los videos e incluir el audio con su códec y hacer un sonido envolvente de 5.1 canales.

Con el conjunto de este software fue como se alcanzó la primera meta. Luego se nos pidió una segunda meta, que era manipular los modelos para acercarlos, alejarlos, rotarlos, etcétera. Para realizar esta tarea nos apoyamos en Amira que es un poderoso software que usa la generación de geometría en tiempo real.

Hasta este punto solo hemos mencionado el uso del Software, pero para poder llevar a cabo estas tareas con este software se necesita un hardware con especificaciones avanzadas, es decir, un hardware de gama alta que sea capaz de hacer cálculos de geometría y física para la obtención de los modelos, tanto en tiempo real como para los videos. Nos ayudamos de las tarjetas Gráficas de Nvidia propiamente de una tarjeta Quadro de la serie FX (Quadro FX 4600) y una

tarjeta GeForce de la familia GTX (GeForce GTX 480). Con ellas pudimos obtener los resultados y además tener una visualización en s3D con lentes activos para la apreciación de los resultados.

## 2.2 Conceptos de s3D en el espacio de la pantalla 2D

Como mencionamos en el capítulo 1, el 3D real está en el mundo que nos rodea, en tanto que lo que vemos en animaciones o dibujos 3D sigue siendo una imagen 2D (mono). En cambio el s3D es el término que se usa para visualizar imágenes estereoscópicas.

Para ver desplegada la escena en un monitor y dar una sensación de profundidad se debe hacer una cámara virtual que crea un Frustum. El Frustum es un prisma truncado de base rectangular, el cual está conformado por una base que será el plano lejano, el truncamiento lo dará el plano cercano y la cúspide del prisma será la posición de la cámara. Dentro del Frustum podremos visualizar los objetos que nosotros queremos, y fuera de éste los objetos no serán visibles (Figura 2.2.1).

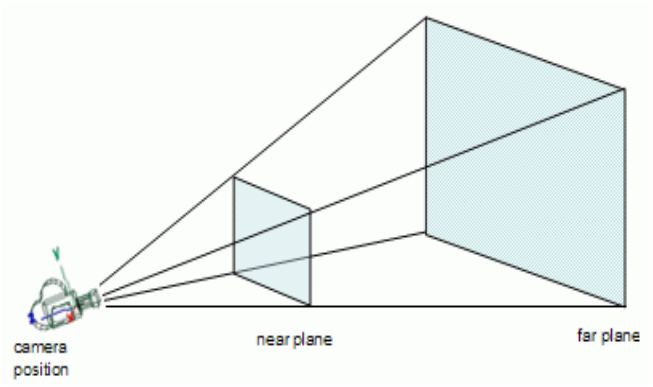



Figura 2.2.1

Dependiendo de la posición de la cámara y de los planos cercano y lejano se crea nuestro Frustum.

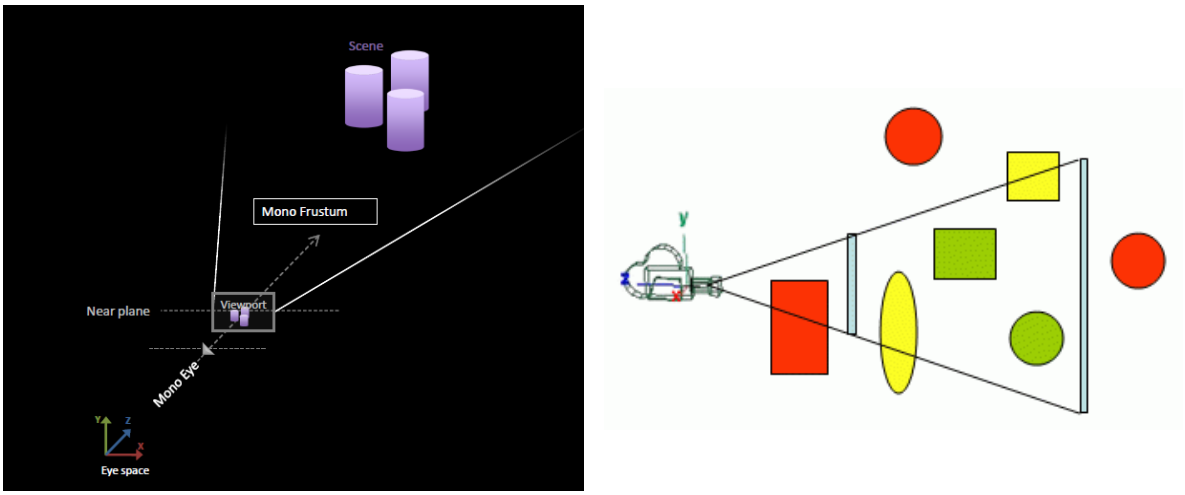



Figura 2.2.2 Vista de una escena mono

Todos los objetos que estén dentro del Frustum serán visibles, los que queden en los límites del Frustum estarán truncados (Figura 2.2.2). Esto sucede en una escena mono, es decir, solo actúa una cámara, en el caso del s3D se tienen dos cámaras las cuales deben de estar paralelas de la cámara central y la separación entre ellas se denomina interaxial o entre ejes (Figura 2.2.3).

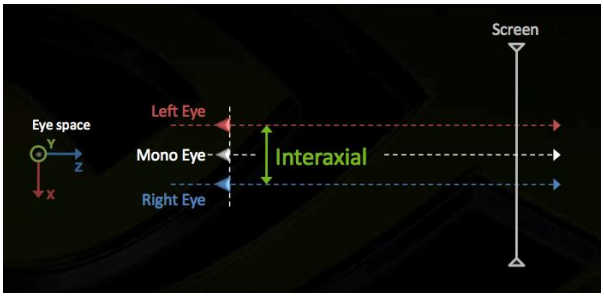



Figura 2.2.3 Separación Interaxial

Cuando se crean estas cámaras cada una de ellas tiene su propio Frustum. Donde los Frustum convergen se crea una pantalla virtual, ésta pantalla virtual es nuestra pantalla de puerto de vista (Figura 2.2.4). Delante de ella los objetos de la escena se muestran como si estuvieran fuera de la pantalla, mientras los que están atrás de ella tendrán un efecto de lejanía (Figura 2.2.5), generando en nuestro plano

cercano dos imágenes para una escena; una imagen para el ojo izquierdo y otra imagen para el ojo derecho (Figura 2.2.5).

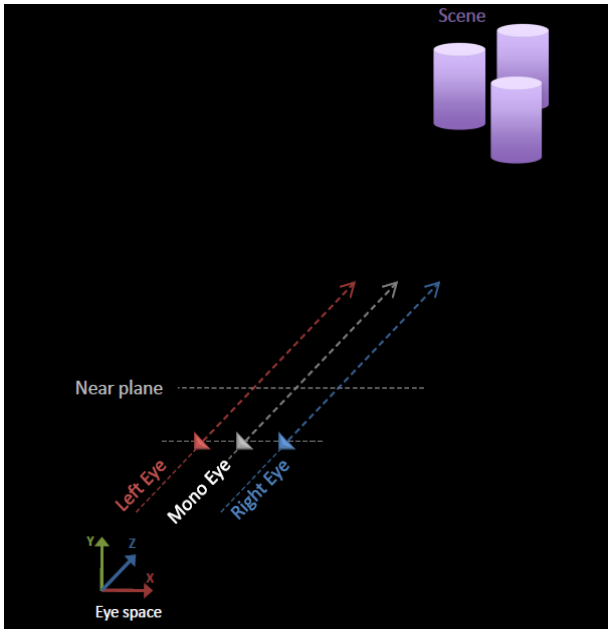



Figura 2.2.4 Plano Cercano

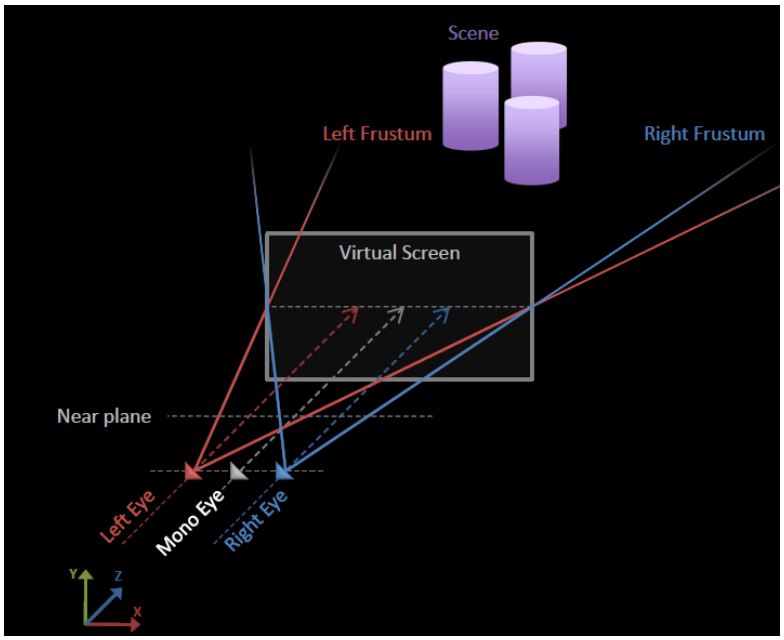



Figura 2.2.5 Convergencia de los Frustum

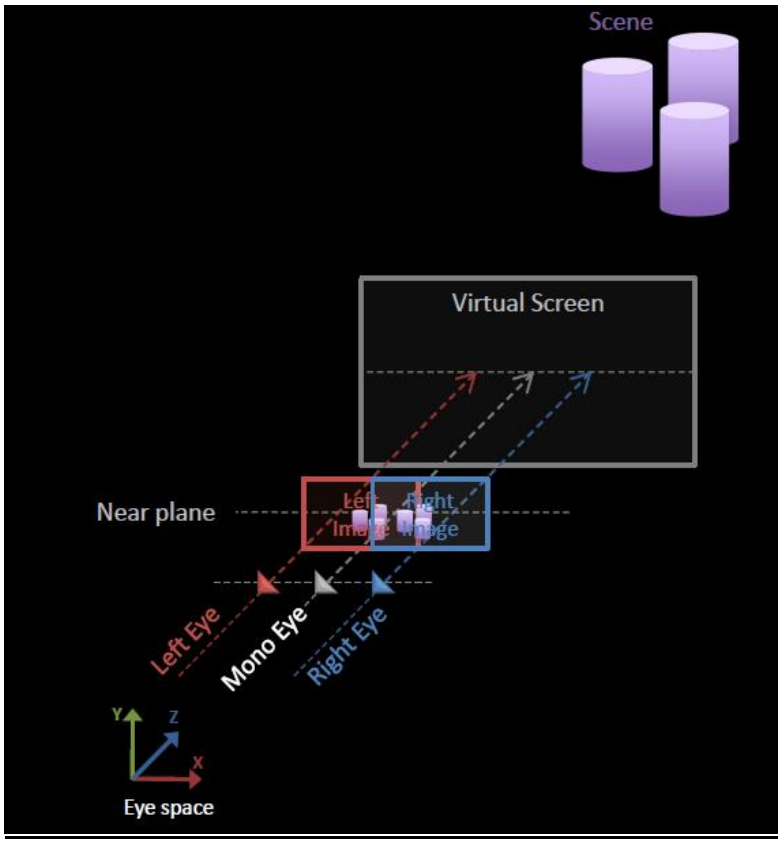
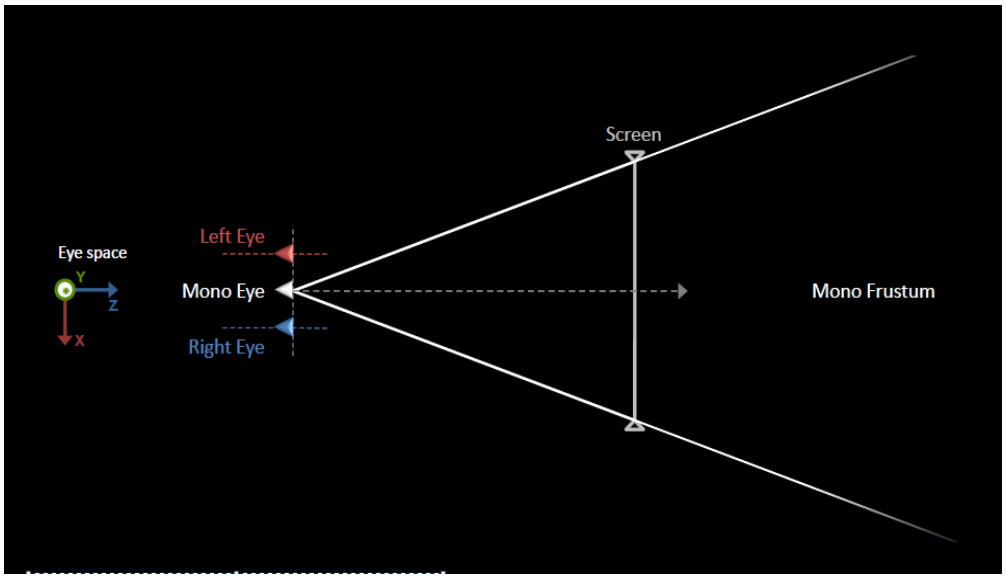
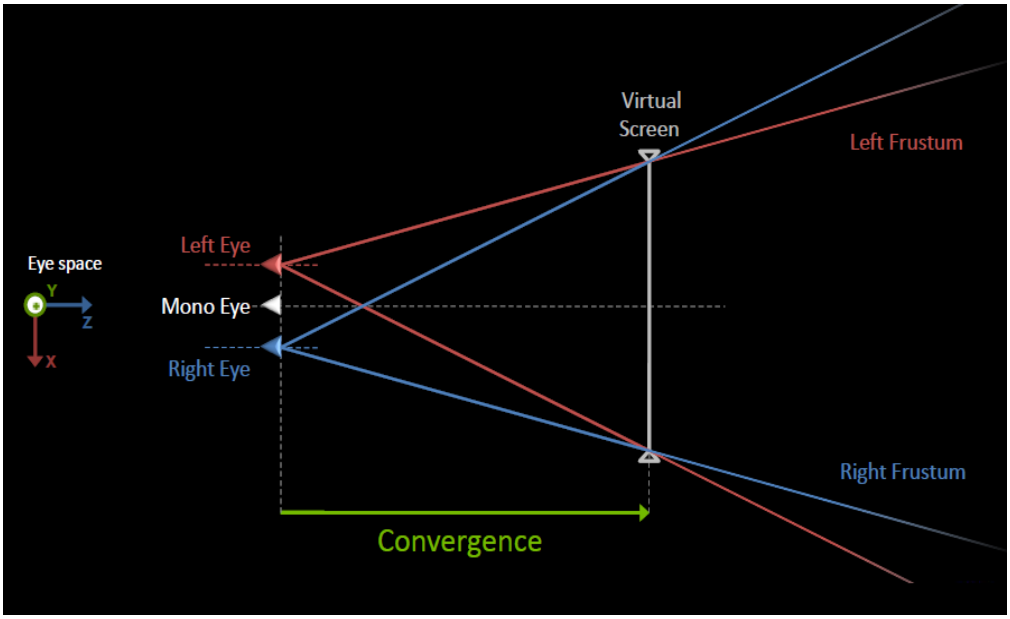



Figura 2.2.6 Imágenes independientes para cada cámara

En resumen con el concepto de Frustum se forma lo que llamamos “matriz de proyección mono”. Así mismo, la matriz de proyección estéreo se conformará por el desplazamiento horizontal tanto a la izquierda como a la derecha del eje “X” de la matriz de proyección mono normal (Figura 2.2.6). La proyección de cada uno de las cámaras deberá de ser paralela a la central y en la convergencia de los nuevos Frustum se obtendrá una pantalla virtual (Figura 2.2.7).




a) La matriz de proyección mono, con su Frustum




b) Convergencia de los Frustum y creación de la pantalla virtual

Figura 2.2.7. Matriz de proyección mono y estéreo

Otro concepto muy importante a tratar es el denominado paralaje o “parallax” en inglés; el paralaje es determinado por la cantidad de separación entre los vértices o las imágenes de una escena. Una forma más fácil de entender el paralaje tal como se utiliza en s3D es empleando la pantalla virtual.

La pantalla virtual indica la localización en el espacio donde aparecen las imágenes o vértices que se están observando. Se dice que los objetos que están en la pantalla virtual no tienen paralaje o que son de cero paralaje y por lo general se observan como si fueran de dos dimensiones.

Los objetos que se encuentran detrás de la pantalla virtual tienen paralaje positivo, mientras que los que aparecen enfrente de la pantalla tienen paralaje negativo.

### 2.3 El concepto de Confort

Con base en la información hasta el momento explicada ya se cuenta con herramientas para decidir los aspectos importantes que requiere una visualización confortable. Debido a que el propósito de una imagen s3D es hacer una simulación de nuestro entorno, se deben evitar visiones borrosas, dobles, o que nuestro cerebro no pueda reconstruir correctamente.

Como se mencionó anteriormente el paralaje es fundamental en la visualización s3D, pues con ésta herramienta se trabaja para que nuestros modelos sean claros. En caso de emplear paralaje positivo o negativo las imágenes mostradas serán confusas y producirán cansancio al visualizarlas. Las recomendaciones esenciales son:

- Las **cámaras izquierda** y **derecha** deben mostrar una **perspectiva paralela** a la **cámara central**, así como el valor **Interaxial** debe ser aproximadamente **6.5 [cm] (**Figura 2.3.1).

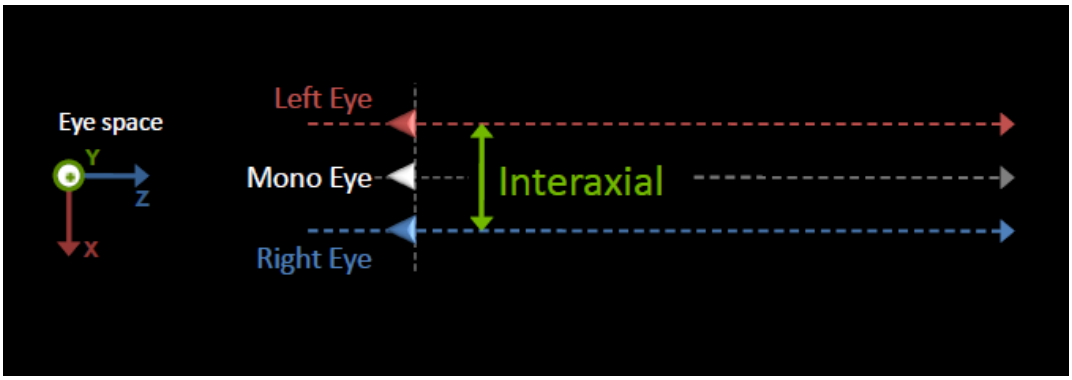



Figura 2.3.1. EL valor promedio de Interaxial

- La convergencia de los Frustum llamada **pantalla virtual** pasara a ser nuestra **pantalla real (**Figura 2.3.2).

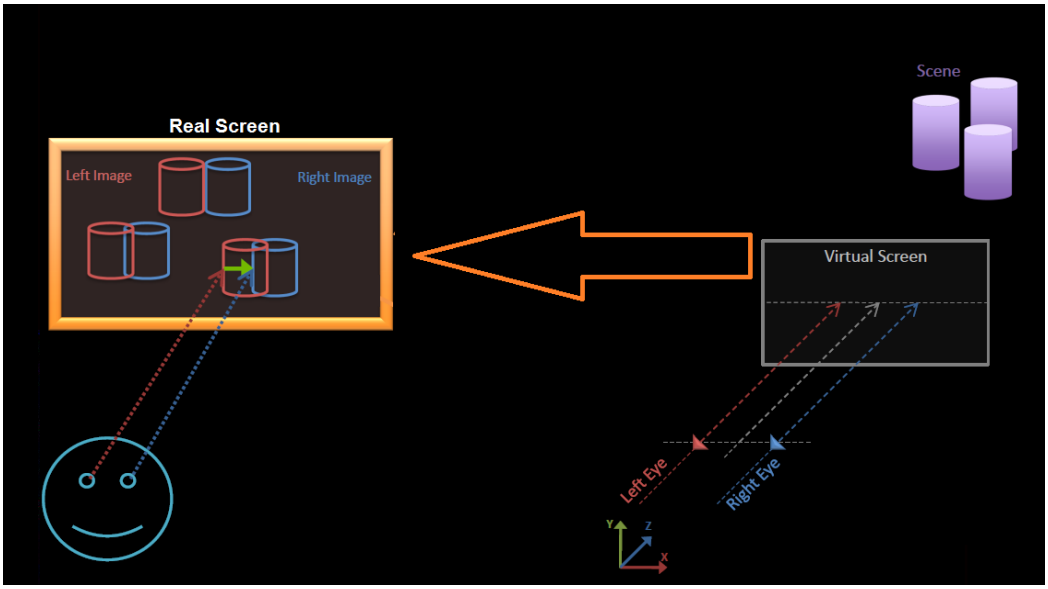



Figura 2.3.2 Pantalla Real

- Según sea el tamaño del **ancho** de la **pantalla real** se normalizará la **separación ocular**.
- **Separación ocular = separación interocular / ancho de la pantalla real**
- El paralaje máximo al **infinito** será la **separación ocular**. Esta separación es la más confortable para el uso del estéreo s3D.

- La separación ocular es un promedio y debería usarse como el valor máximo de separación. Se toma esta separación como un promedio, pues no todas las personas tienen la misma separación interocular que es de aproximadamente 6.5 cm
- Evitar siempre que un espectador observe una divergencia (Figura 2.3.3).

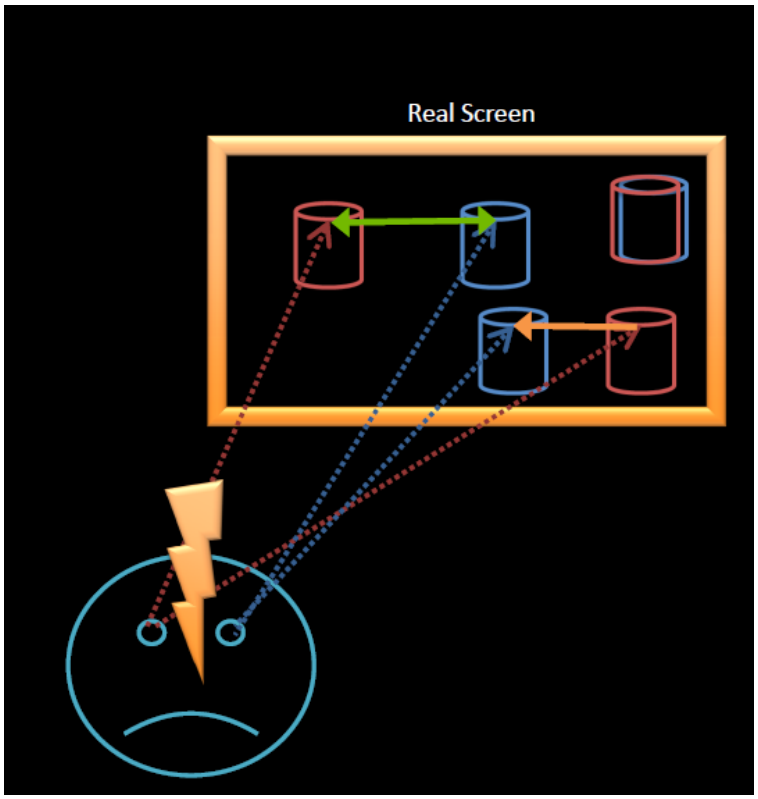



Figura 2.3.3. Mal uso del paralaje
generando divergencia en las imágenes.

### 2.4 Ortoestereoscopia

Como se ha explicado anteriormente, un efecto estereoscópico confortable se obtiene a través de procedimientos que respetan parámetros geométricos, los cuales son aproximados a las proporciones de la visión humana. La raíz latina *Ortho*, que significa *igual a*, se utiliza en ortoestereoscopia para señalar que la

distancia interaxial de las cámaras o puntos de captura de la imagen es la misma distancia interocular humana, 65 milímetros o 2.5 pulgadas aproximadamente.

Usando cámaras binoculares o lentes especializados se logran capturar imágenes estereoscópicas con una percepción de la profundidad muy aproximada a la visión humana.

Figura 2.4.1. Cámara binocular para capturar imágenes ortoestereoscópicas.

Cabe destacar que los parámetros antes mencionados se utilizan en la estereoscopía para objetos del mundo real. Sin embargo son herramientas que permiten establecer los principios para la presentación de escenas modeladas por computadora, donde las distancias y dimensiones son relativas.

## 2.5 Arreglo de Cámaras

Los arreglos de las cámaras son muy importantes, de estos depende una correcta visualización y se evitan problemas de divergencia o cansancio visual.

Como sabemos la estereoscopía no es nueva y existen varias formas de tener estos arreglos de cámaras que se desarrollaron y usaron en los años 90’s, y aún en nuestros días algunos sistemas están basados en ellos, estas técnicas son:

- El uso de dos cámaras.
- Uso una cámara y dos películas.
- Uso de una cámara y una película.

**El uso de dos cámaras**. Este sistema comenzó a usarse en los principios de la fotografía, cuando se colocaron dos cámaras una al lado de la otra para representar a los ojos humanos. En un principio esto produjo un s3D incorrecto ya que la distancia entre las dos lentes no siempre fue la de los ojos y no se tomó en cuenta la distancia de la escena ni los ligeros desfasamientos verticales entre las dos lentes; es decir no se conocía el concepto que nosotros llamamos separación ocular. Como consecuencia de esto se desarrollaron tres nuevas técnicas con el uso de dos cámaras.

a) Dos cámaras de lado a lado sincronizadas. Este formato se utilizó por Universal y Columbia Pictures para sus propias producciones (Figura 2.5.1).

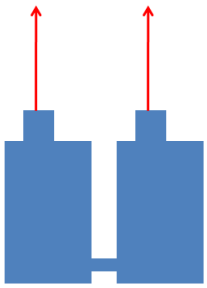


Figura. 2.5.1. Dos cámaras sincronizadas

b) Dos cámaras, una cámara frente a la otra con la escena a un lado de ellas y reflejada en cada uno por medio de un espejo. Este se utilizó hasta finales de 1950 bajo el nombre de “Natural Vision” entre otros (Figura 2.5.2).

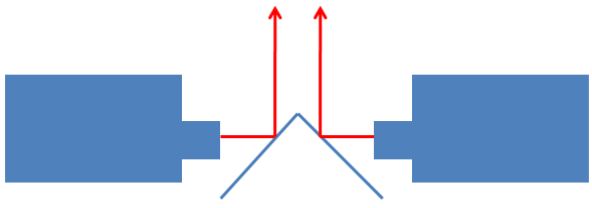


Figura 2.5.2. Cámaras frente a frente

c) Dos cámaras colocadas a 90 grados entre ellas con una mitad de divisor de haz y un espejo entre las cámaras. La colocación crítica del espejo y la pérdida de luz eran los inconvenientes. Fue utilizado principalmente por Warner Bros. y Technicolor ( Figura 2.5.3).

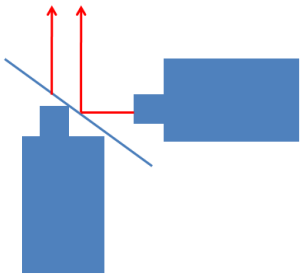


Figura 2.5.3. Cámaras a 90 grados

2) **Una cámara / dos películas.** El concepto de una sola cámara con dos lentes y un mecanismo de operación para dos películas se remontan a la década de 1920 (Figura 2.5.4).

Figura 2.5.4. Una cámara y dos películas

3) **Una cámara / una película**. Esta idea se remonta a los años de 1930 y 1940

a) Se toman dos puntos de vista a dos diferentes alturas verticales en lugar de la base horizontal prescrita (Figura 2.5.5).

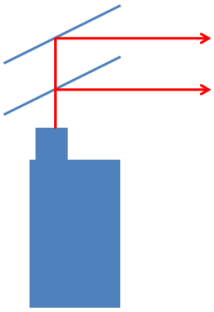


Figura 2.5.5.

b) Dos vistas se obtienen por el uso de prismas en ángulo para reflejar las imágenes (Figura 2.5.6).

Figura 2.5.6. Usando prisma en la cámara

c) Dos vistas se obtienen usando una sola lente con filtros de color detrás de ella que tiene el efecto de dividir la imagen verticalmente (Figura 2.5.7).

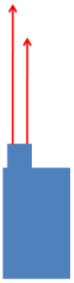


Figura 2.5.7. Cámara con filtros

Debido a que el presente trabajo se realizó usando el software Autodesk Maya 2011, se utilizó el método de dos cámaras con las debidas recomendaciones antes mencionadas. Se editaron las propiedades estereo de las dos cámaras utilizadas en este trabajo con la finalidad de que el Frustum fuera desplegado correctamente. Para poder realizar estas ediciones es necesario programar en python dichas propiedades de posición; a continuación mostraremos el código.

stereoCameraCenterCamShape.nearClipPlane=
stereoCameraCenterCamShape.zeroParallax / 2;

La Figura (2.5.8). Muestra cómo queda el Frustum.

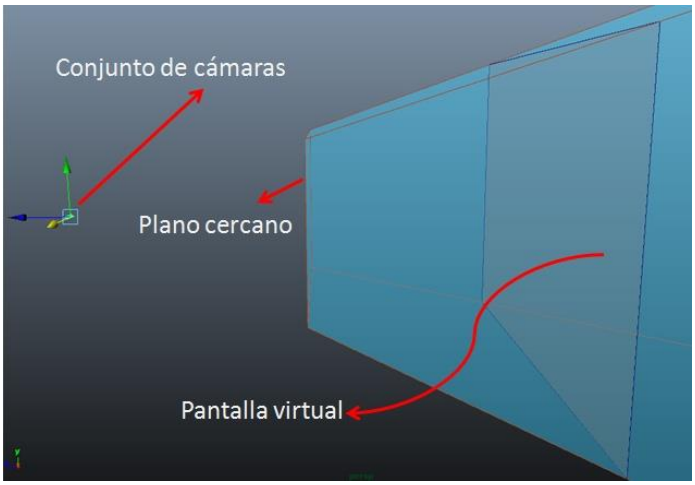



Figura. 2.5.8. Muestra de la pantalla en Autodesk Maya 2011.

Para que cada cámara respete las reglas de separación interocular debemos inicializar la variable de ajuste en 6.5, lo cual se hace a través de obtener la distancia entre la cámara y el objeto más cercano de la escena multiplicándolo por un factor de ajuste de 1/30. Esto nos asegura que no se sobrepase tal valor y cause un efecto de divergencia. A continuación se muestra el código:

float $ajuste = 6.5;

float $distancia_objeto;

$distancia_objeto = distanceDimensionShape1.distance/30;

```
stereoCameraCenterCamShape.interaxialSeparation = $distancia_objeto;

if  (stereoCameraCenterCamShape.interaxialSeparation > 6.5)

        stereoCameraCenterCamShape.interaxialSeparation = $ajuste;
```

Como siguiente paso, el uso de las cámaras debe de estar paralelo a la cámara central, de otra manera si hacemos que las cámaras izquierda y derecha giren podríamos ocasionar que la imagen tenga una divergencia mayor a la separación ocular mencionada, además de que la pantalla virtual sería demasiado angosta; obteniendo resultados muy confusos para su visualización.

### 2.6 Generación de s3D en tiempo real

Una vez explicado el arreglo de las cámaras y la forma en que nosotros implementamos la técnica de dos cámaras con separación ocular, el siguiente tema a tratar es sobre el despliegue en tiempo real de una escena en s3D.

Para realizar la visualización en este ámbito hicimos uso de Interfaces de Programación de Aplicaciones (API's) que nos ayudan a desplegar este tipo de ambientes que se requieren renderizar en tiempo real.

Una API permite una abstracción en la programación entre distintas capas o interfaces de programación, comúnmente es entre lenguajes de bajo nivel y lenguajes más abstractos, de forma tal que se puede interactuar casi directamente con el hardware a través de funciones nativas del sistema operativo o mediante una aplicación específicamente diseñada con ese propósito.

DirectX es un conjunto de APIs desarrolladas para facilitar la comunicación en la plataforma Microsoft Windows y consta de las siguientes API's:

- Direct3D: utilizado para el procesamiento y la programación de gráficos en tres dimensiones, una de las características más usadas de DirectX.
- Direct Graphics: para dibujar imágenes en dos dimensiones (planas) y para representar imágenes en tres dimensiones.

- DirectInput: para procesar datos del teclado, mouse, palanca de mando (joystick) y otros controles para juegos.
- DirectPlay: para comunicaciones en red.
- DirectSound: para la reproducción y grabación de sonidos de ondas.
- DirectMusic: para la reproducción de pistas musicales compuestas con DirectMusic Producer.
- DirectShow: para reproducir audio y vídeo con transparencia de red.
- DirectSetup: para la instalación de componentes DirectX.
- DirectCompute: lenguaje e instrucciones especiales para el manejo de cientos o miles de hilos de procesamiento, especial para procesadores de núcleos masivos.

Por otro lado tenemos OpenGL, una API multilenguaje y multiplataforma la cual puede producir gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geometrías simples, tales como puntos, líneas y triángulos.
Se usa ampliamente en CAD, realidad virtual, representación científica, visualización de información y simulación de vuelo.

En este trabajo también hicimos uso del software Amira, el cual está basado en el API de OpenGL y al complementarlo con la arquitectura de una tarjeta Nvidia Quadro FX 4600 fuimos capaces de desplegar escenas s3D en tiempo real.

### 2.7 Tipos de render para la obtención de imágenes

Cuando hablamos de la obtención de las imágenes, la obtención de un video, así como un audio procesado a través de la computadora y que fue creado o solamente modificado, a todo esto se le conocerá como renderización. Para ello existen diferentes tipos de software llamados render. Estos renders calculan las reflexiones y refracciones, iluminaciones y soluciones borrosas, además de crear materiales con iluminación propia, elementos traslúcidos (ropa o papel) o utilizan texturas de dispersión.

En forma particular hablaremos de los renders que ocupamos para los modelos hechos en Autodesk Maya 2011. Existen varios tipos de renders, en este caso nos enfocaremos en el que tiene incorporado Autodesk Maya, que es Mental Ray. En función de la capacidad que tenía nuestra tarjeta gráfica y CPU esta fue la mejor opción.

Cuando nos referimos a que hay varios tipos de renders queremos decir que están los que funcionan con el procesador y la tarjeta gráfica (Híbridos) y los que solo trabajan con la tarjeta gráfica. Existen otros que son de tipo híbrido, los cuales son más lentos ya que usualmente para una escena se pueden tardar días en terminar el proceso de renderización.

Un render que probamos y que resultó muy eficiente es Octane Render; un poderoso render que únicamente utiliza el procesador de la tarjeta gráfica y que es capaz de mostrar los cambios casi en tiempo real dependiendo de la cantidad de núcleos y memoria de la tarjeta gráfica. Pero no se utilizó en este proyecto debido a que la tarjeta gráfica utilizada no contaba con las características mínimas para ese programa. Esto habla de que se necesita de una tarjeta gráfica de altas capacidades tanto en procesador gráfico como en memoria para poder llevar a cabo el desarrollo completo de una producción.

Otro tipo de Render es Renderman, conocido por ser usado en los proyectos Pixar Animation Studios. Este render es híbrido, pero la tarjeta gráfica que utilizamos no cumple con los requerimientos mínimos del programa, por lo cual quedó descartado también. V-ray es también un render poderoso que se puede usar en Autodesk Maya a través de la instalación de un plug-in, pero tampoco era compatible con la tarjeta gráfica que se utilizó.

# Capítulo 3.El Modelado con Autodesk® Maya® 2011

En este capítulo abordaremos los elementos básicos que conforman una geometría en los programas para crear modelos por computadora.

**Desarrollo de los modelos**

El Software Autodesk® Maya® 2011 ofrece herramientas para el desarrollo de animaciones, modelados, simulaciones, renderizado, rastreo de movimiento y composición de imágenes. También es utilizado para crear efectos visuales, desarrollo de videojuegos, post producción y otros tipos de desarrollos visuales.

Para crear nuestros modelos nos apoyamos en este software debido a que el flujo de trabajo es más fácil y es compatible con el hardware utilizado.

## 3.1 Introducción a las Técnicas De Modelado Con Polígonos Y Nurbs En Autodesk® Maya® 2011

Para construir los modelos teníamos tres opciones de geometrías: Nurbs, Polígonos y Subdivisiones. Cada selección cuenta con diferentes herramientas para modelar, el segundo problema fue decidir a partir de qué tipo de geometría se realizaría el modelado porque se puede usar cualquier tipo de geometría para poder construir objetos simples o complejos. Podemos usar un tipo de geometría como un punto de partida para pasar a otro tipo.

En general, si uno construye formas orgánicas lo más recomendable es usar Nurbs o subdivisiones.

**Nurbs**: Son figuras creadas a base de curvas y superficies cuyos componentes son básicamente los vértices de control, las isoparamétricas (isoparms) y los loops enteros de isoparamétricas (hulls) (Figura 3.1.1).

La geometría de Nurbs proporciona una superficie suave con menos puntos de control que facilita la edición de las superficies.

Sin embargo, los Nurbs son superficies limitadas a cuatro lados, por lo que generan complicaciones al momento de definir los tipos de formas orgánicas que se pueden construir a partir de una sola superficie. Por esta razón es adecuado usar subdivisiones debido a que son capaces de representar muchos más tipos de formas con una sola superficie.

Este tipo de geometría deriva de curvas y superficies creadas por los vértices de control (CVs); los cuales nos permitieron comenzar con curvas que que posteriormente se usaron para generar superficies. Este proceso de trabajo ofrece resultados muy precisos que pueden ser fácilmente controlados. Todas las superficies Nurbs son superficies de parches de cuatro lados, aunque esta superficie puede ser alterada usando diversas herramientas de edición.

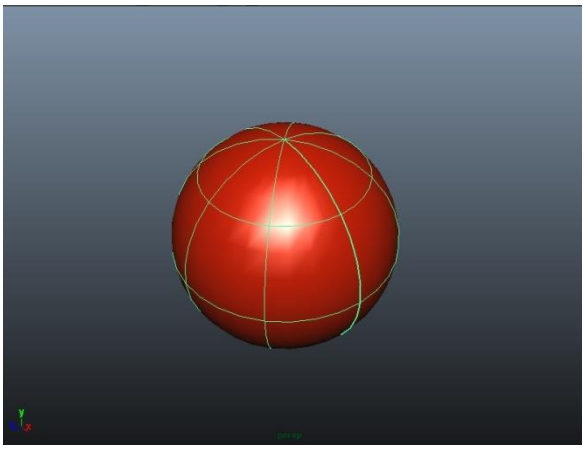


Figura.3.1.1. Esfera hecha con Nurbs

**Polígonos**: Son los objetos más fáciles de modelar por su falta de complejidad y su mayor número de herramientas. Sus componentes básicos son las caras, aristas y vértices (Figura 3.1.2).

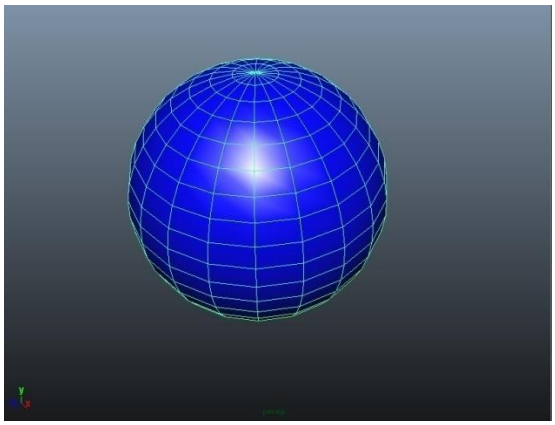


Figura. 3.1.2. Esfera hecha con Polígono

**Subdivisiones**: Son un híbrido entre las Nurbs y los Polígonos. Sin embargo no se pueden modelar usando ambos estilos a la vez, para ello se escoge en qué modo se desea modelar (Modo estándar o Modo Polígono). Poseen los mismos componentes que las Nurbs y los Polígonos además de un modo de refinamiento por niveles para obtener mayor subdivisión geométrica y conseguir así mayor detalle de modelado. Figura. (3.1.3).

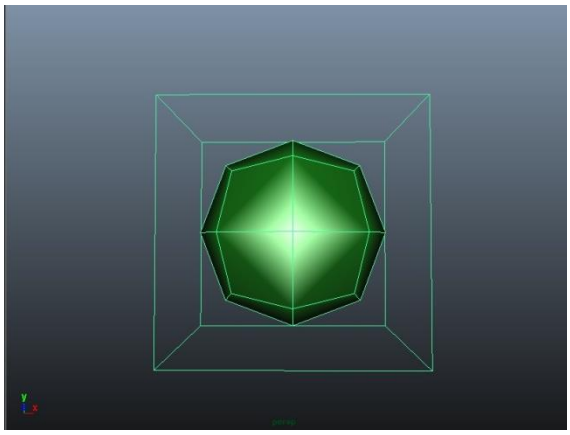


Figura. 3.1.3. Esfera hecha con Subdivisiones

### 3.1.1. Componentes de las Nurbs

Para crear superficies Nurbs es importante comprender cómo trabajan las curvas. Los mismos principios detrás de las curvas Nurbs son aplicados a las superficies Nurbs, ya que estas dos están estrechamente relacionadas.

La gran diferencia entre curvas y superficies es que la curva tiene solamente una dirección, mientras que la superficie tiene dos direcciones. Estas dos direcciones tienen un origen y juntas definen la normal de esta superficie que determina el frente y la parte trasera de una superficie.

Los componentes de una superficie Nurbs son muy similares a los de una curva, pero los puntos finales llamados EndPoint (EP) no pueden ser editados. Las superficies Nurbs tienen CVs, Hulls y Spans que definen la forma de un parche de cuatro lados. En una superficie existen los llamados Isoparms que son las líneas formadas por la intersección de los CV´s.

**Curve Direction**: El inicio de la curva es definido por un cuadrado y una pequeña “u” que define la dirección de la curva.

**Control Vértices (CV)**: Estos puntos definen la forma de la curva. Dependiendo del grado de la curva, el CV indistintamente controlará la forma de la curva.

**Span**: Es una sección de la curva. Cada span es como una pequeña curva que tiene una relación continua con el siguiente span. Es decir, uno o un grupo de span forman una curva.

**Curve Point**: Uno puede seleccionar este punto que representa la medida de “U” a todo lo largo de la curva. El valor de “U” es dependiente del parámetro de la curva.

**Hulls**: Nos muestra una línea recta que conecta los CVs y la curva. Cuando se selecciona un hull estamos seleccionando los CVs asociados.

**Edit Point (EP)**: Son puntos que existen sólo dentro de la curva que definen el inicio y el término de los spans.

En la siguiente se muestran los elementos mencionados (Figura 3.1.4).

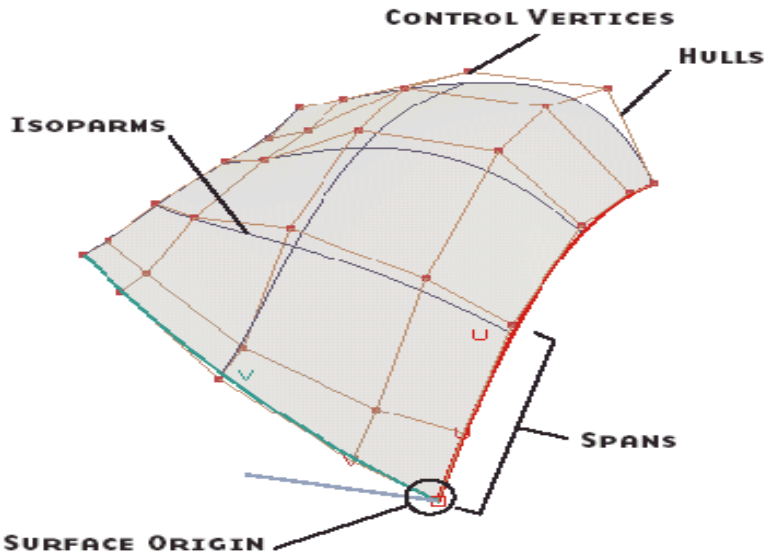



Figura. 3.1.4. Elementos de un Nurbs

### 3.1.2. Componentes de los Polígonos

Los polígonos pueden ser definidos como un número de puntos conectados que crean una forma o cara. Los puntos están conectados por aristas o edges que rodean a la cara resultante. Una cara puede ser triangular, cuadrada, o de N números de aristas o vértices, que en su conjunto forman una malla poligonal.

Cada malla poligonal está formada por componentes que son modificables para ayudar a crear y editar nuevas mallas. Hay varias herramientas que nos permiten crear y deformar la malla geométrica para crear las formas que se requieran.

Maya nos entrega un conjunto de poligonales primitivas que nos podrán dar un punto de partida para crear modelos. Estas primitivas poligonales, con excepción del plano, son superficies cerradas. Todas ellas tienen historia de construcción, por lo que sus divisiones en UV podrán modificarse una vez creadas. Las divisiones en UV son dos coordenadas que se asignan a cada uno de los vértices de un modelo; es decir, así como un vértice tiene tres coordenadas X, Y, Z, referentes a su posición en el espacio, también tienen dos coordenadas más que determinan su posición en un plano bidimensional que representa la textura.

Para modificar un polígono debemos conocer sus principales estructuras:

**Edge:** Se le llama así a cada borde que forma una cara y está delimitado por los vértices de la misma.

**Faces:** Se le nombra así a las caras que forman la figura y están rodeadas por los edges y los vértices (Vertex).

**Vertex:** Son los vértices que forman la figura y además delimitan a cada cara (Face) y borde (Edge) de la figura.

Mientras menos subdivisiones tenga un objeto, será más redondo, y al contrario mientras más subdivisiones se tengan, será menos redondo (Figura 3.1.5).

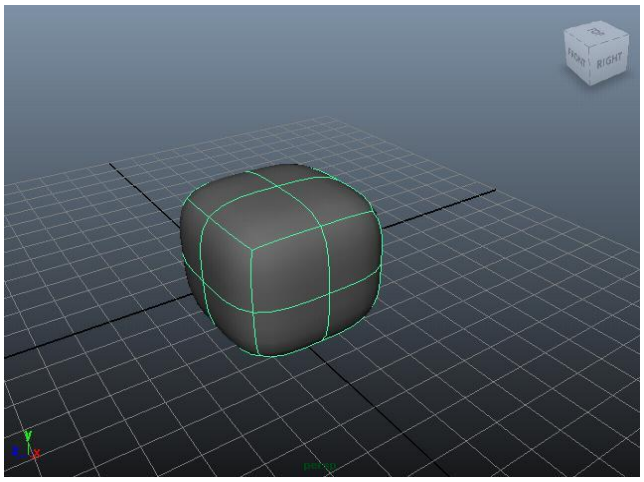


Figura. 3.1.5. Ejemplo de la creación de un cubo

Por ejemplo en la Figura. (3.1.6) se pueden observar los elementos antes mencionados. Los cuales podemos editar por separado escogiendo uno de estos tres.

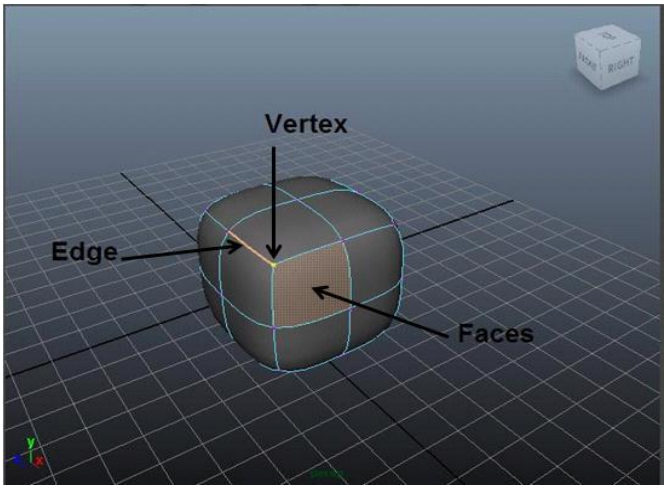


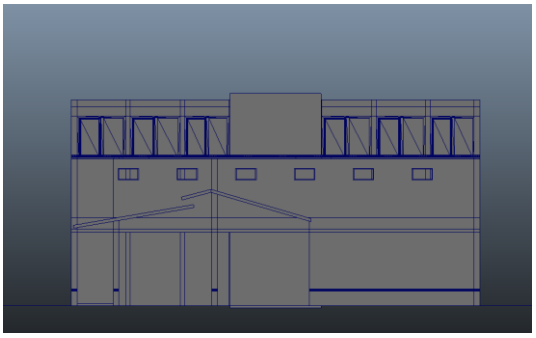

Figura. 3.1.6. Elemento de los Polígonos

### 3.2 Modelado De Objetos Del Mundo Real

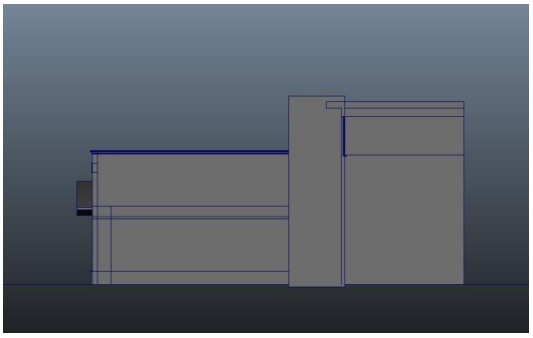
Una vez conocidos estos conceptos que se han explicado de una manera muy breve podremos escoger la geometría que más nos convenga para modelar nuestros objetos, por ejemplo podremos modelar a un humano, un animal, una casa o incluso objetos que no podemos ver a simple vista como las estructuras moleculares.

Abajo se muestran modelos realizados con el Software Autodesk® Maya® 2011.

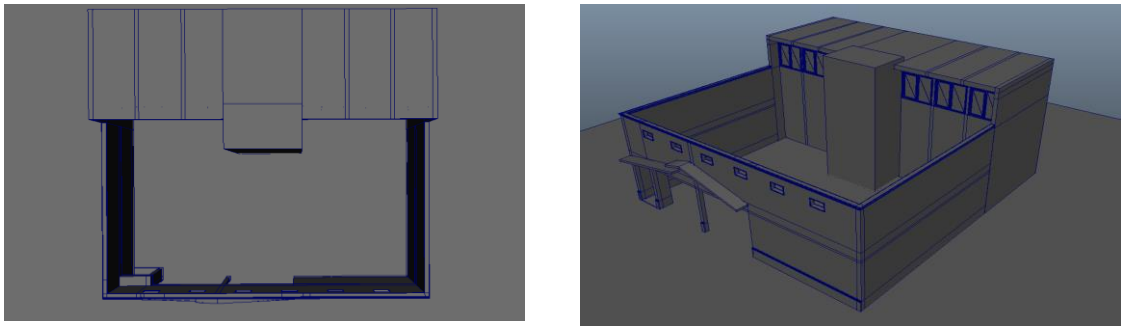
El primer ejemplo corresponde al modelo de una casa con un amplio jardín (Figura 3.2.1).



Vista Frontal



Vista Lateral



Vista Superior Perspectiva

Figura.3.2.1. Modelación de una casa en sus diferentes vistas

La variedad y complejidad de los objetos que pueden ser modelados es ilimitada y el uso de las herramientas de Maya® para hacerlo es independiente de usuario a usuario, determinando la complejidad del modelo que se le quiera dar (Figura 3.2.2).

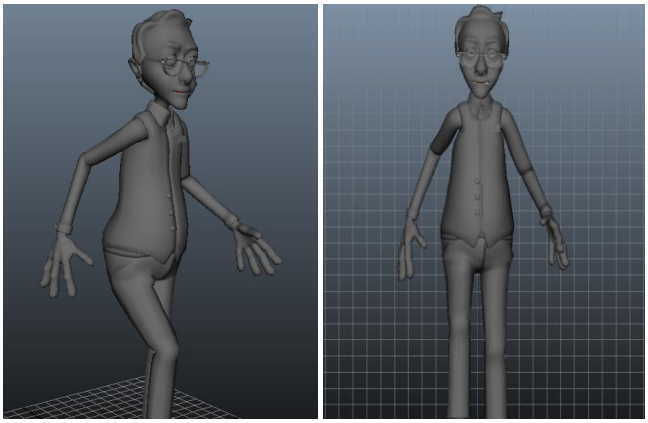


Figura. 3.2.2. Modelación de un humano

Considerando la amplia capacidad de modelación que va desde estructuras macroscópicas hasta microscópicas, en este trabajo mostramos el modelado 3D estéreo de algunas estructuras moleculares. La primer modelación que se nos pidió hacer fue la representación del tensoactivo dodecil sulfato de sodio (SDS),

un tensoactivo o surfactante que representa una molécula que controla la tensión superficial entre dos fases, y el alineamiento de varias de estas moléculas controlan la separación de las fases. Los tensoactivos tienen una parte hidrofóbica (evita la interacción con moléculas del agua) y una parte hidrofílica (interacciona con moléculas del agua), esta característica permite una orientación de las moléculas dependiendo del ambiente hidrofóbico o hidrofílico en el que se encuentren. Además las moléculas del tensoactivo pueden ser iónicas o no iónicas, lo cual influye también en su alineación.

Para poder realizar este modelo lo que hicimos fue crear doce esferas que nos representaran los Carbonos (C), continuamos formando cuatro esferas que nos representaran los Oxígenos (O), una esfera que nos representara el Azufre (S) y una esfera que nos representara el Sodio (Na) (Figura 3.2.3).

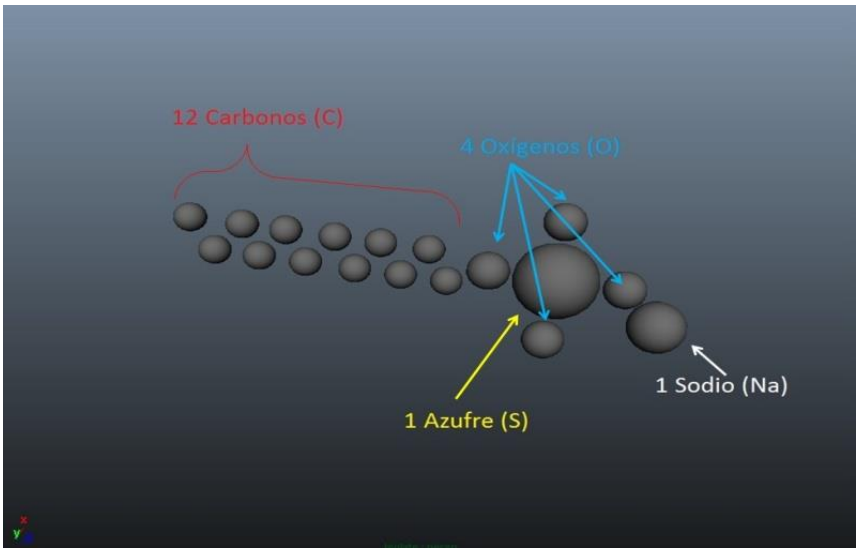



Figura 3.2.3. Tensoactivo dodecil sulfato de sodio

### 3.3 Texturizado De Los Objetos

Hasta este momento solo se ha modelado el objeto, el siguiente paso es aplicarle “color” para lo cual también se emplea el texturizado. El texturizado es usado en videojuegos, filmes, caricaturas, espectaculares, etc. Es un paso muy importante para que los objetos modelados creen la ilusión de autenticidad.

Las texturas se consideran el alma del objeto modelado, pues con este paso se da vida al entorno del objeto y al objeto, generando un camino de nuevas ideas. Una vez que se ha obtenido una textura bien aplicada podemos crear la ilusión de realismo que tanto se busca.

Dentro del Software Autodesk® Maya® 2011 existen muchos tipos de materiales que nos ayudan a dar textura, pero para dar una impresión de realismo al objeto muchas veces no basta con poner solo un tipo de textura. Este software cuenta con una vasta gama de métodos y tipos de textura que nos ayudan a crear ese ambiente real que tanto se desea.

Desde Autodesk® Maya® 2011 podemos asignar estos materiales a los diferentes objetos, los materiales son llamados de diferentes maneras y cada uno de ellos poseen diferentes cualidades que se pueden editar para lograr un determinado efecto. O se puede crear un mapa de textura que es un archivo de color RGB de 32 bits que puede ser generado en Adobe Photoshop o en algún otro programa de edición de imágenes. Dentro de los diferentes tipos de texturas están las llamadas texturas “tileable” (azulejo) que son usadas para repetirse en un área grande, su cualidad es el de tener bordes, o no tenerlos para llevar a cabo la repetición. Mientras las texturas personalizadas son únicas para un modelo específico o área. Una textura personalizada tiene un espacio UV único y una resolución fija. A estos mapas se les asigna~~ra~~ un tipo de material, y se modifica para lograr el efecto que nosotros deseamos.

Con respecto a los materiales, uno de los más comunes es el Lambert, el cual es un material monótono que produce un efecto suave sin brillo. Es un material ideal para las superficies sin brillo como la cerámica, la tiza, la pintura mate, etcétera; y se asigna por defecto.

En nuestro caso asignaremos un material tipo Phong y dependiendo de las esferas asignaremos el color, quedando las doce esferas que nos representan los Carbonos (C) en color rojo, a las 4 esferas que nos representan los Oxígenos en color azul, al azufre (S) en color Amarillo y al Sodio (Na) en color blanco (Figura 3.3.1).

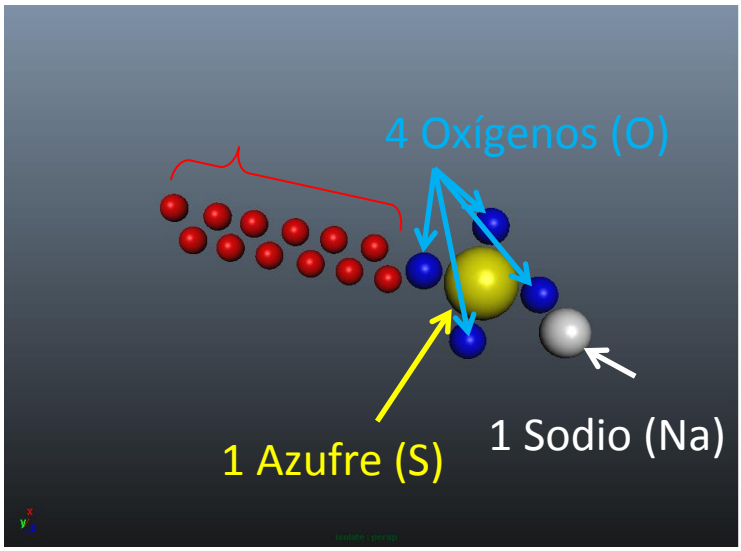



Figura 3.3.1. Aplicación de la textura.

La segunda modelación realizada corresponde a la representación de una micela de SDS a partir del resultado anterior. Es decir, un grupo ordenado de moléculas del tensoactivo.

**Creación de una Micela**

Una micela es una formación geométrica de moléculas que tienen una "cabeza" polar y una "cola" de naturaleza no polar. Pueden estar adheridas a una partícula o glóbulo de sustancias que se encuentran en un medio en el cual no son solubles. Un caso típico de micela es la que forma el jabón en contacto con un

glóbulo de grasa adherido a un tejido. El jabón es en esencia una sal sódica de un ácido graso que se obtiene mediante el proceso de saponificación. Pues bien, la parte sódica de la molécula de jabón tiene características polares, es decir se disuelve en agua (que también es polar), mientras que la "cola" corresponde al resto de la cadena carbonada que es apolar y se disuelve en las grasas y aceites. Un ejemplo lo podemos ver en la Figura 3.3.2.

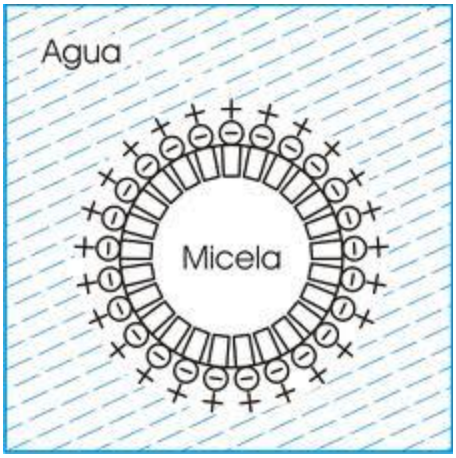



Figura 3.3.2. Micela de un surfactante iónico

Una vez terminado el proceso de texturizado del SDS vamos a agrupar esta figura y la llamaremos SDS1, para después duplicarla 16 veces y así formar una circunferencia que nos representara un plano de la micela, posteriormente seleccionamos todos los SDS, los agrupamos y los llamamos SDS_CIRCULAR (Figura 3.3.3).

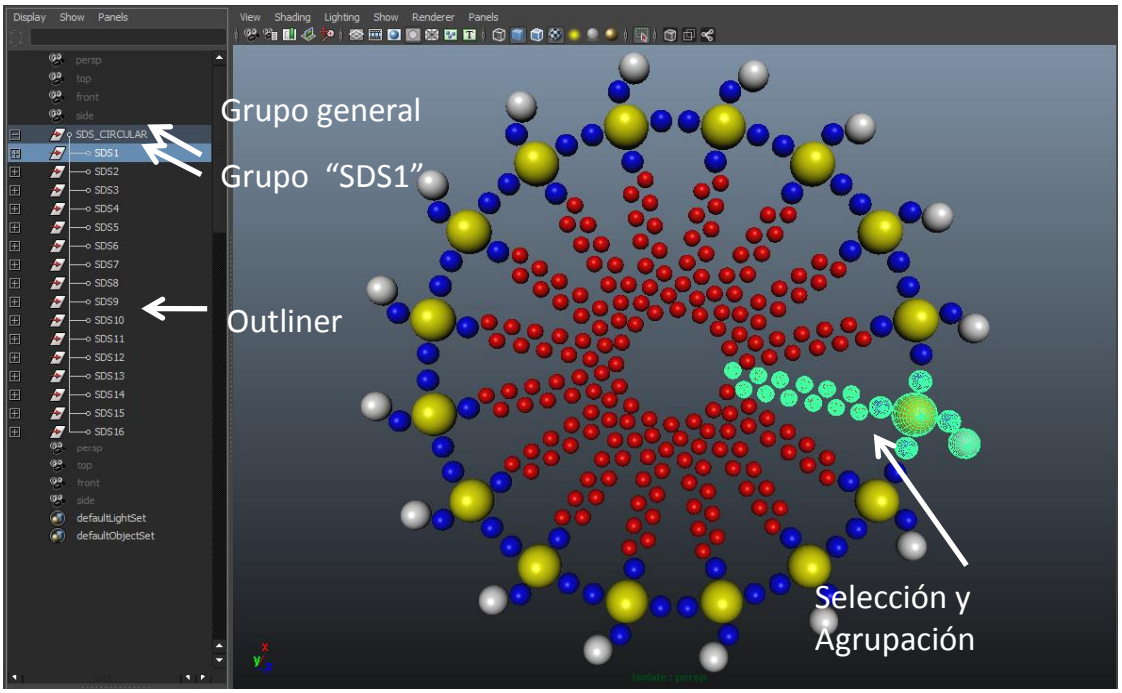



Figura 3.3.3. SDS_CIRCULAR.

### 3.4 Animación De Los Modelos En Autodesk® Maya® 2011

La animación es el proceso utilizado para dar la sensación de movimiento a un objeto inanimado, en Autodesk® Maya® 2011 el proceso se realiza a través de fotogramas o imágenes que en secuencia darán la sensación de movimiento a nuestros modelos.

La animación consiste en el realismo que se pueda dar al objeto, como ya lo hemos mencionado antes, se trata de simular el mundo real y para conseguirlo hay que detallarlo y hacerlo de la manera más natural posible.

Como hemos mencionado en la modelación, la animación también se aplica a personas, objetos que se caen, o se rompen y a una infinidad de cosas que se pueden animar.

En nuestro caso, en un principio cada molécula de SDS va estar ordenada de forma diferente, tanto en la traslación como en la rotación; y pasado determinado tiempo va a llegar a tomar de nuevo la circunferencia que se tenía (Figura 3.4.1).

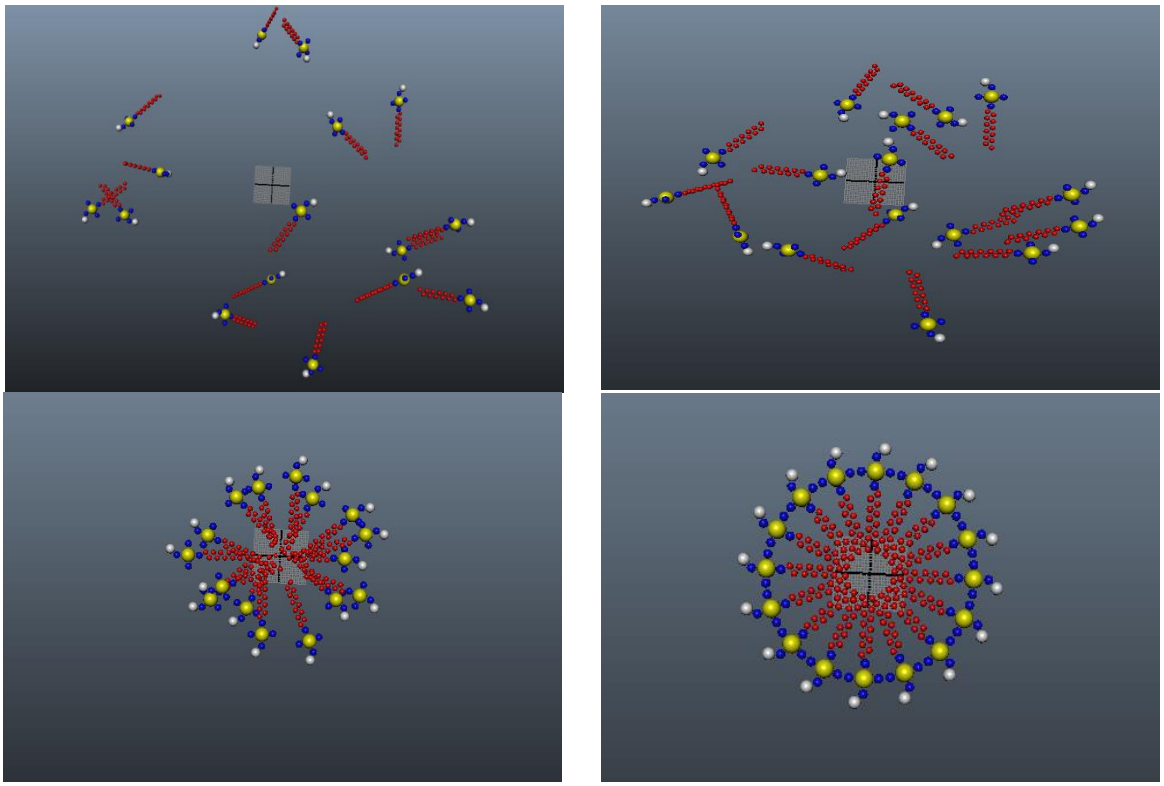


Figura 3.4.1. Proceso por fotograma animado

### 3.5 Creación Del Arreglo De Cámaras En Maya 2011

En nuestro entorno todo lo que percibimos lo hacemos por medio de nuestros cinco sentidos, pero en especial hacemos uso de la vista y con ella percibimos el color y profundidad de los objetos. Por ejemplo, podemos saber si un objeto está detrás de otro, si se encuentra encima o debajo de otro, podemos inclusive ser capaces de saber si es una textura lisa, rugosa, etcétera; gracias a este sentido nuestro cerebro es capaz de dar el efecto de s3D. Para el espectador siempre existirá una ventana estereoscópica debido al paralaje del borde de las dos imágenes (Figura 3.5.1).

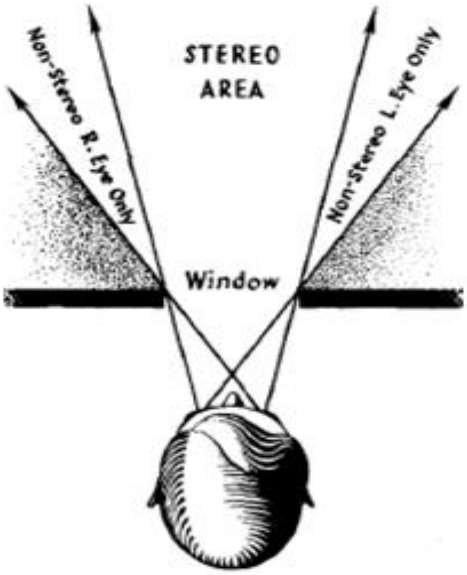
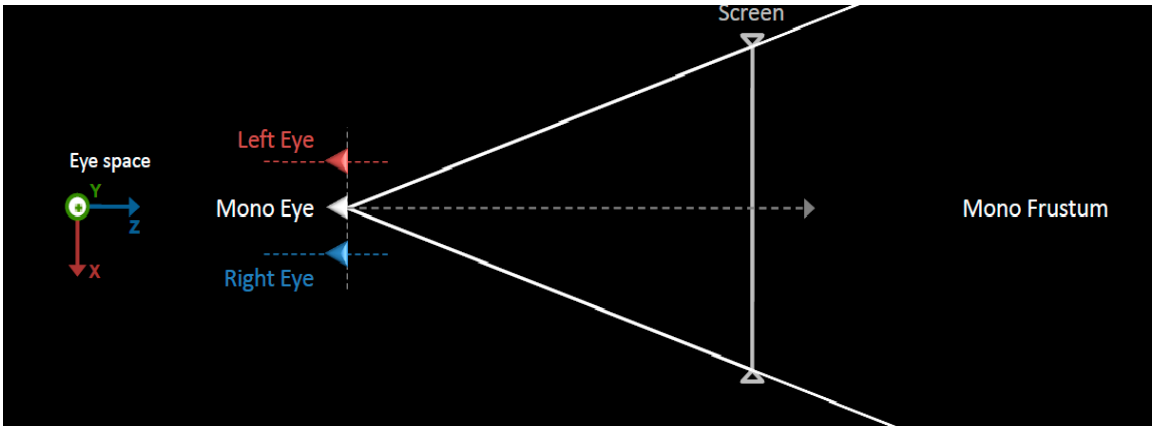


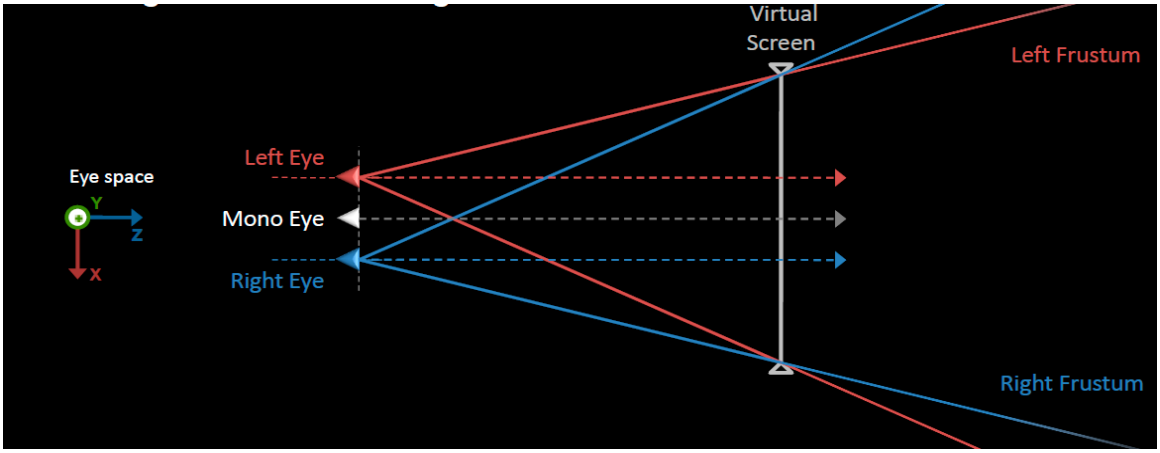

Figura 3.5.1.

Para obtener una vista s3D, es decir, tener un conjunto de cámaras asignando una al ojo izquierdo y una al ojo derecho; Autodesk® Maya® 2011 nos da esta opción de creación de cámara estéreo. Pero no solo basta con poner la cámara estéreo para poder obtener un efecto s3D, además se deben de ajustar los parámetros de dichas cámaras. Para lograr este efecto debemos saber que cada cámara dará una imagen en 2D y dependiendo de los parámetros de la imagen obtenida, como el efecto de sombras, luces, juego de profundidad, etcétera, es con lo que daremos en conjunto una sensación de s3D.

Como anteriormente se mencionó, en nuestro mundo real podemos percibir los objetos en tres dimensiones, para poder simular este comportamiento a través de Autodesk® Maya® 2011 debemos tener en cuenta los conceptos mencionados en el capítulo 2 de este trabajo. Por ejemplo la matriz de proyección, con la cual vamos a configurar las cámaras hechas en Autodesk® Maya®. Debemos tener en cuenta que esta matriz de proyección es un desplazamiento horizontal de una proyección; creando así un Frustum para cada cámara u ojo y en donde los dos Frustum convergen es donde veremos nuestra imagen estéreo (Figura 3.5.2).




Frustum mono




Frustum Estéreo

Figura 3.5.2. Muestra de la región Frustum

### 3.6 Sugerencias Para La Composición y Visualización De La Escena s3D

Como ya se mencionó en el capítulo 2, algo muy importante que se debe tener en cuenta es que para poder dar una sensación correcta del estéreo activo se debe cuidar el arreglo de las cámaras, por ejemplo que la cámara derecha e izquierda siempre tengan una distancia aproximada de 6.5 cm entre ellas; a este hecho de separación le llamaremos separación interaxial (Interaxial Separation) y está dado por el hecho de que nuestros ojos tienen esta separación. El cuidado de estos permitirá una visión correcta de nuestro objeto.

También se debe tener en cuenta que el tipo de pantalla afecta nuestra visión 3D. Es importante saber en dónde se va a proyectar el objeto, pues es diferente para cada sistema de visualización.

Otro aspecto a resaltar es el paralaje, pues cuando éste tiende al infinito nos da la separación de las imágenes y no debe excederse en la separación porque se crea una divergencia entre las imágenes, dando lugar a sombras lo cual es molesto al espectador que ve el objeto representado en 3D.

### 3.7 Renderizado Con Mental Ray

La renderización es la parte final de la producción para poder pasar a la postproducción es decir, realizar el video adicionando sonidos y efectos al mismo. En nuestro caso la renderización se hizo con Mental Ray, debido a que es eficiente, rápido y proporciona una alta calidad en cada imagen además de que es perfectamente compatible con el hardware utilizado.

Existen varios tipos de motores de renderizado; por ejemplo, está Renderman, de Pixar Animation Studios, que posee grandes cualidades de mejoramiento en calidad y tiempo de espera para cada imagen. Sin embargo, no es compatible con el hardware utilizado en este trabajo, además de que el plug-in para maya es costoso. Otro motor de render con el que se experimentó fue Octane Render, el cual posee características muy novedosas debido a que el proceso de renderizado se lleva a cabo únicamente con la GPU ya que no está basado en una arquitectura híbrida; es decir, no usa la CPU y GPU. Es recomendable tener una tarjeta Nvidia de arquitectura Kepler o Fermi, obteniéndose mejores resultados con la primera. Cabe mencionar que el programa Maya debe tener el plug-in para el renderizador que se va a utilizar.

Como un punto importante debemos mencionar que Mental Ray es un buen motor de renderizado, que nos permitió comprender la importancia de un buen manejo de modelación, texturizado, y animación; pues con él pudimos observar que entre más geometría y más complejidad en un modelo puede tardar hasta días en ser

renderizado. Por esta razón se recomienda realizar modelos con el mínimo de geometrías y sin tomar en cuenta tantos vértices.

En el siguiente capítulo se muestra la manera en que se realiza la visualización de los modelos, qué tipo de hardware se usa y los pasos para hacer el video teniendo en cuenta que ya se obtuvieron las imágenes o fotogramas. Posteriormente comentaremos cómo hacer un proceso simple de post producción del video.

# Capítulo 4. El hardware y el Software

En los capítulos anteriores se comentó sobre los tipos de lentes, cómo obtener las dimensiones de la proyección del video s3D y sobre un hardware específico. En este capítulo hablaremos de la tecnología que nosotros usamos.

### 4.1 Visualización

Se usaron lentes tipo activos ya que con ellos la sensación de s3D es muy real, teniendo los filtros para cada ojo perfectamente sincronizados conforme avanza el video; la contra de esta tecnología es que es muy cara pues se requiere de un emisor que será el que sincronice a los lentes y cada lente tendrá un receptor para poder ser sincronizados, además cuentan con un cristal líquido (LCD) que necesita un cuidado especial para ser limpiados ya que se ensucian fácilmente al manipularlos con las manos o al colocarlos en la cara.

Para poder usar este receptor se cuenta con una tarjeta gráfica Nvidia Quadro serie FX 4600 la cual posee una arquitectura unificada, no es tan avanzada como las arquitecturas Kepler o Fermi, pero sirvió para poder hacer los videos s3D.

Todo esto unido junto con un proyector es lo que hizo posible la visualización de nuestros videos s3D, el proyector también posee una característica importante, la frecuencia de refresco vertical de hasta 120 Hz.

### 4.2 Generación De Las Secuencias De Fotogramas

Cuando hablamos de fotogramas a lo que nos estamos refiriendo es a cada imagen que se produce cuando se renderiza nuestro modelo, recordemos que se generan las mismas imágenes pero con diferente cámara y cada imagen va a tener diferente posición del modelo y diferente distribución. Una vez que se tienen todas las imágenes de la animación con la resolución deseada, se puede empezar a compilar los videos e iniciar la post producción.

Hasta este punto solo tenemos cuadros de imágenes y lo que debemos crear la secuencia, es decir, un video con estos cuadros; tales cuadros estarán en formato .bmp (bitmap) para que no se pierda la calidad de la imagen al ser compilado el video. Para la compilación usamos un programa llamado VirtualDub con el cual

creamos videos de formato AVI (siglas en inglés de Audio Video Interleave). Estos videos serán para cada una de las cámaras, es decir, un video izquierdo y un video derecho.

### 4.3 Compilación Del Video Usando Microsoft Encoder y Sony Vegas

Después de haber generado los videos con formato AVI veremos que ocupan una gran cantidad de espacio en el disco duro. Es en este paso cuando se añaden efectos, títulos, sonido, formato al video, etcétera. Para hacer este proceso probamos dos soluciones, la primera fue usando Microsoft Expression Encoder 4, con el cual se trabajaba a cada video por separado dando un códec H.264/ MPG-4 y WMV para poder ser reproducidos de igual manera como un archivo MP4.

Una vez terminado el proceso de codificación para cada video, se realizó una multiplexión para obtener un solo video, este video ya multiplexado se recodificó y se añadieron los títulos, créditos, efectos. Así mismo se añadieron los canales de audio, mediante un códec AC-3 que es el que soporta audio desde el tipo estéreo de 2 canales hasta el de 5.1 canales.

La otra opción que se probó fue Sony Vegas Pro 11 con el cual observamos que todo este proceso se vuelve mucho más eficiente y más sencillo, pues con él podemos incluir los videos que se generaron con VirtualDub en dos canales y hacer el multiplexeo, al mismo tiempo que se agrega el audio, títulos, créditos así como efectos de entrada y salida del video.

## 4.4 Visualización en Amira

El programa Amira 5.3.3 es un software que sirve para visualización en proyectos relacionados con el área de ciencias de la vida así como de manipulación y comprensión de datos biomédicos. Es muy usado en los laboratorios para poder realizar la reconstrucción de modelos a través de tomografías o radiografías. Esta poderosa herramienta no solo puede visualizar imágenes, también posee herramientas para la medición estadística de tejido, densitometría y por supuesto visualización s3D.

Nosotros lo utilizamos para visualizar los modelos y tener una sensación de inmersión en la imagen.

## 4.5 Conversión De Los Modelos Generados En Maya Para Su Visualización En Amira

Nuestros modelos hechos en Autodesk Maya son un conjunto de puntos (vértices) acomodados, que en conjunto nos representan lo que deseamos modelar, por lo que el software Amira solo se encargó de reconstruir esos puntos (vértices) para darles una representación en su ventana.

Recordemos que solo se va a reconstruir la geometría en tiempo real y que con base a ella le vamos a poder manipular a nuestro gusto y solo se podrá retomar cierto tipo de materiales, no se pueden usar los shaders, ni materiales específicos; únicamente se pueden usar materiales base.

A continuación en la Figura (4.5.1) mostramos un ejemplo de la visualización en Amira.

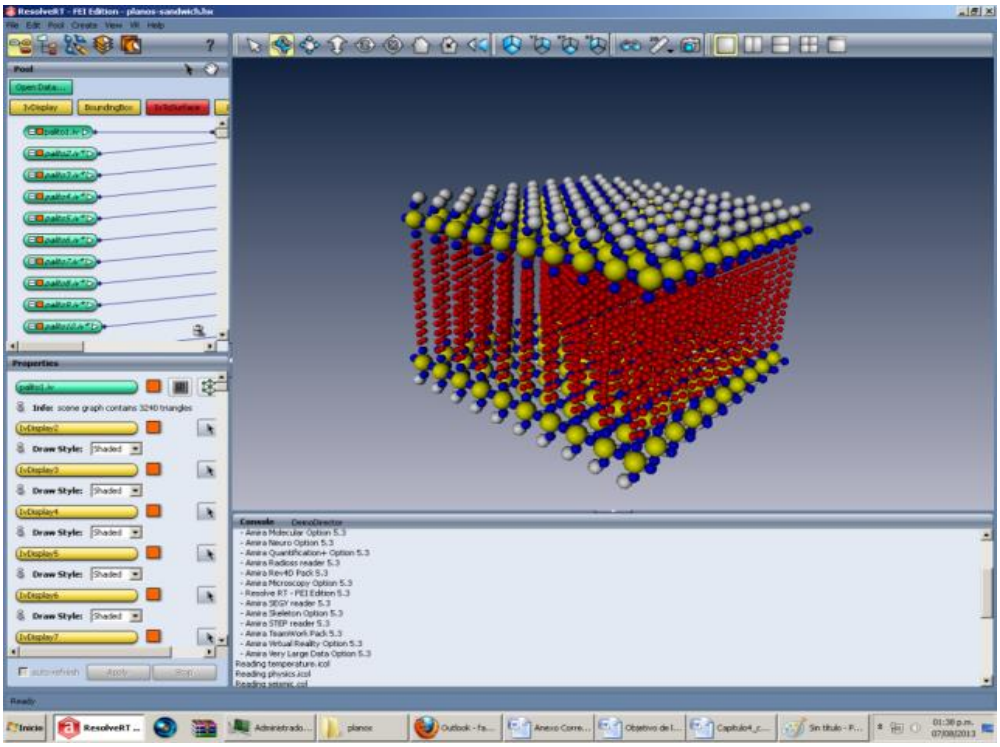


Figura 4.5.1. Plano formado por moléculas de SDS visualizado en Amira

En la imagen se muestra el plano realizado en Autodesk Maya el cual se puede manipular con el puerto de vista de Amira, mejorando los modelos tradicionales.

A continuación mostramos el código fuente del proyecto en Amira. Se realizó manualmente usando un objeto de molécula SDS y multiplicándolo con código ya que se reduce considerablemente el tiempo de carga en memoria del archivo.

```
# Amira Script

remove -all

remove sDs1.iv IvDisplay


# Create viewers

viewer setVertical 0
```

```
viewer 0 setBackgroundMode 1

viewer 0 setBackgroundColor 0.06 0.13 0.24

viewer 0 setBackgroundColor2 0.72 0.72 0.78

viewer 0 setTransparencyType 5

viewer 0 setAutoRedraw 0

viewer 0 show

mainWindow show


set hideNewModules 0

[ load ${SCRIPTDIR}/sDs.iv ] setLabel sDs1.iv

sDs1.iv setIconPosition 20 10

sDs1.iv fire

sDs1.iv setViewerMask 16383

sDs1.iv select


set hideNewModules 0

create HxIvDisplay {IvDisplay}

IvDisplay setIconPosition 266 10

IvDisplay data connect sDs1.iv

IvDisplay fire

IvDisplay drawStyle setIndex 0 0

IvDisplay fire

IvDisplay setViewerMask 16383

IvDisplay setShadowStyle 0

IvDisplay setPickable 1
```

```
set hideNewModules 0

viewer 0 setCameraOrientation 1 0 0 3.14159

viewer 0 setCameraPosition -0.491638 -1.89665 -96.2068

viewer 0 setCameraFocalDistance 96.0673

viewer 0 setCameraNearDistance 89.4632

viewer 0 setCameraFarDistance 102.684

viewer 0 setCameraType perspective

viewer 0 setCameraHeightAngle 44.9023

viewer 0 setAutoRedraw 1

viewer 0 redraw


echo "PRIMER PLANO"


for {set i 1} {$i < 10} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0;
create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select;
IvDisplay[expr $i+1] fire  }

sDs1.iv duplicate ; sDs11.iv translate 0 0 10;create HxIvDisplay;IvDisplay11 data connect
sDs11.iv;IvDisplay11 fire

for {set i 11} {$i < 20} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0;
create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select;
IvDisplay[expr $i+1] fire  }

sDs11.iv duplicate ; sDs21.iv translate 0 0 10;create HxIvDisplay;IvDisplay21 data connect
sDs21.iv;IvDisplay21 fire

for {set i 21} {$i < 30} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0;
create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select;
IvDisplay[expr $i+1] fire  }

sDs21.iv duplicate ; sDs31.iv translate 0 0 10;create HxIvDisplay;IvDisplay31 data connect
sDs31.iv;IvDisplay31 fire
```

for {set i 31} {$i < 40} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire }

sDs31.iv duplicate ; sDs41.iv translate 0 0 10;create HxIvDisplay;IvDisplay41 data connect sDs41.iv;IvDisplay41 fire

for {set i 41} {$i < 50} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire }

sDs41.iv duplicate ; sDs51.iv translate 0 0 10;create HxIvDisplay;IvDisplay51 data connect sDs51.iv;IvDisplay51 fire

for {set i 51} {$i < 60} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire }

sDs51.iv duplicate ; sDs61.iv translate 0 0 10;create HxIvDisplay;IvDisplay61 data connect sDs61.iv;IvDisplay61 fire

for {set i 61} {$i < 70} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire }

sDs61.iv duplicate ; sDs71.iv translate 0 0 10;create HxIvDisplay;IvDisplay71 data connect sDs71.iv;IvDisplay71 fire

for {set i 71} {$i < 80} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire }

sDs71.iv duplicate ; sDs81.iv translate 0 0 10;create HxIvDisplay;IvDisplay81 data connect sDs81.iv;IvDisplay81 fire

for {set i 81} {$i < 90} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire }

sDs81.iv duplicate ; sDs91.iv translate 0 0 10;create HxIvDisplay;IvDisplay91 data connect sDs91.iv;IvDisplay91 fire

for {set i 91} {$i < 100} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire }

```
echo "SEGUNDO PLANO"


sDs1.iv duplicate ; sDs101.iv translate 0 -40 00;create HxIvDisplay;IvDisplay101 data connect sDs101.iv;IvDisplay101 fire; sDs101.iv rotate -z 180; sDs101.iv rotate -y 180


for {set i 101} {$i < 110} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire  }

sDs101.iv duplicate ; sDs111.iv translate 0 0 -10;create HxIvDisplay;IvDisplay111 data connect sDs111.iv;IvDisplay111 fire

for {set i 111} {$i < 120} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire  }

sDs111.iv duplicate ; sDs121.iv translate 0 0 -10;create HxIvDisplay;IvDisplay121 data connect sDs121.iv;IvDisplay121 fire

for {set i 121} {$i < 130} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire  }

sDs121.iv duplicate ; sDs131.iv translate 0 0 -10;create HxIvDisplay;IvDisplay131 data connect sDs131.iv;IvDisplay131 fire

for {set i 131} {$i < 140} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire  }

sDs131.iv duplicate ; sDs141.iv translate 0 0 -10;create HxIvDisplay;IvDisplay141 data connect sDs141.iv;IvDisplay141 fire

for {set i 141} {$i < 150} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire  }

sDs141.iv duplicate ; sDs151.iv translate 0 0 -10;create HxIvDisplay;IvDisplay151 data connect sDs151.iv;IvDisplay151 fire
```

for {set i 151} {$i < 160} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire  }

sDs151.iv duplicate ; sDs161.iv translate 0 0 -10;create HxIvDisplay;IvDisplay161 data connect sDs161.iv;IvDisplay161 fire

for {set i 161} {$i < 170} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire  }

sDs161.iv duplicate ; sDs171.iv translate 0 0 -10;create HxIvDisplay;IvDisplay171 data connect sDs171.iv;IvDisplay171 fire

for {set i 171} {$i < 180} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire  }

sDs171.iv duplicate ; sDs181.iv translate 0 0 -10;create HxIvDisplay;IvDisplay181 data connect sDs181.iv;IvDisplay181 fire

for {set i 181} {$i < 190} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire  }

sDs181.iv duplicate ; sDs191.iv translate 0 0 -10;create HxIvDisplay;IvDisplay191 data connect sDs191.iv;IvDisplay191 fire

for {set i 191} {$i < 200} {set i [expr $i+1]} {sDs$i.iv duplicate;sDs[expr $i+1].iv translate -local 10 0 0; create HxIvDisplay; IvDisplay[expr $i+1] data connect sDs[expr $i+1].iv; IvDisplay[expr $i+1] select; IvDisplay[expr $i+1] fire  }

load molecula.iv;molecula.iv setLabel molecula1.iv;molecula1.iv translate 0 20 0;create HxIvDisplay IvDisplayM1;IvDisplayM1 data connect molecula1.iv; IvDisplayM1 fire;

for {set i 1} {$i < 10} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula1.iv duplicate ; molecula11.iv translate 0 0 10;create HxIvDisplay IvDisplayM11;IvDisplayM11 data connect molecula11.iv;IvDisplayM11 fire

for {set i 11} {$i < 20} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula11.iv duplicate ; molecula21.iv translate 0 0 10;create HxIvDisplay IvDisplayM21;IvDisplayM21 data connect molecula21.iv;IvDisplayM21 fire

for {set i 21} {$i < 30} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula21.iv duplicate ; molecula31.iv translate 0 0 10;create HxIvDisplay IvDisplayM31;IvDisplayM31 data connect molecula31.iv;IvDisplayM31 fire

for {set i 31} {$i < 40} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula31.iv duplicate ; molecula41.iv translate 0 0 10;create HxIvDisplay IvDisplayM41;IvDisplayM41 data connect molecula41.iv;IvDisplayM41 fire

for {set i 41} {$i < 50} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula41.iv duplicate ; molecula51.iv translate 0 0 10;create HxIvDisplay IvDisplayM51;IvDisplayM51 data connect molecula51.iv;IvDisplayM51 fire

for {set i 51} {$i < 60} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula51.iv duplicate ; molecula61.iv translate 0 0 10;create HxIvDisplay IvDisplayM61;IvDisplayM61 data connect molecula61.iv;IvDisplayM61 fire

for {set i 61} {$i < 70} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create

HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula61.iv duplicate ; molecula71.iv translate 0 0 10;create HxIvDisplay IvDisplayM71;IvDisplayM71 data connect molecula71.iv;IvDisplayM71 fire

for {set i 71} {$i < 80} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula71.iv duplicate ; molecula81.iv translate 0 0 10;create HxIvDisplay IvDisplayM81;IvDisplayM81 data connect molecula81.iv;IvDisplayM81 fire

for {set i 81} {$i < 90} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula81.iv duplicate ; molecula91.iv translate 0 0 10;create HxIvDisplay IvDisplayM91;IvDisplayM91 data connect molecula91.iv;IvDisplayM91 fire

for {set i 91} {$i < 100} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula1.iv duplicate ; molecula101.iv translate 0 -80 00;create HxIvDisplay IvDisplayM101;IvDisplayM101 data connect molecula101.iv;IvDisplayM101 fire; molecula101.iv rotate -z 180; molecula101.iv rotate -y 180

for {set i 101} {$i < 110} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula101.iv duplicate ; molecula111.iv translate 0 0 -10;create HxIvDisplay IvDisplayM111;IvDisplayM111 data connect molecula111.iv;IvDisplayM111 fire

for {set i 111} {$i < 120} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create

HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula111.iv duplicate ; molecula121.iv translate 0 0 -10;create HxIvDisplay IvDisplayM121;IvDisplayM121 data connect molecula121.iv;IvDisplayM121 fire

for {set i 121} {$i < 130} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula121.iv duplicate ; molecula131.iv translate 0 0 -10;create HxIvDisplay IvDisplayM131;IvDisplayM131 data connect molecula131.iv;IvDisplayM131 fire

for {set i 131} {$i < 140} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula131.iv duplicate ; molecula141.iv translate 0 0 -10;create HxIvDisplay IvDisplayM141;IvDisplayM141 data connect molecula141.iv;IvDisplayM141 fire

for {set i 141} {$i < 150} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula141.iv duplicate ; molecula151.iv translate 0 0 -10;create HxIvDisplay IvDisplayM151;IvDisplayM151 data connect molecula151.iv;IvDisplayM151 fire

for {set i 151} {$i < 160} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula151.iv duplicate ; molecula161.iv translate 0 0 -10;create HxIvDisplay IvDisplayM161;IvDisplayM161 data connect molecula161.iv;IvDisplayM161 fire

for {set i 161} {$i < 170} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula161.iv duplicate ; molecula171.iv translate 0 0 -10;create HxIvDisplay IvDisplayM171;IvDisplayM171 data connect molecula171.iv;IvDisplayM171 fire

```
for {set i 171} {$i < 180} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula171.iv duplicate ; molecula181.iv translate 0 0 -10;create HxIvDisplay IvDisplayM181;IvDisplayM181 data connect molecula181.iv;IvDisplayM181 fire

for {set i 181} {$i < 190} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }

molecula181.iv duplicate ; molecula191.iv translate 0 0 -10;create HxIvDisplay IvDisplayM191;IvDisplayM191 data connect molecula191.iv;IvDisplayM191 fire

for {set i 191} {$i < 200} {set i [expr $i+1]} {molecula$i.iv duplicate;molecula[expr $i+1].iv translate [expr rand()*15] 0 [expr rand()*5];molecula[expr $i+1].iv rotate -xz [expr rand()*180]; create HxIvDisplay IvDisplayM[expr $i+1]; IvDisplayM[expr $i+1] data connect molecula[expr $i+1].iv; IvDisplayM[expr $i+1] select; IvDisplayM[expr $i+1] fire  }


create HxSound {Sound}

Sound setFilename "${SCRIPTDIR}/Isao Tomita   Arabesque No1.wav"

Sound setLoop 1

Sound play
```

### 4.6 Resultados

A continuación se presentan los resultados obtenidos a lo largo de esta tesis; entre las cuales se incluyen las imágenes anaglíficas de las micelas de SDS, tubos de micelas de SDS, micelas de SDS con Óxido de Silicio, así como las bicapas de SDS con y sin Óxido de Silicio.

Primeramente se presentan los resultados obtenidos con el software Amira

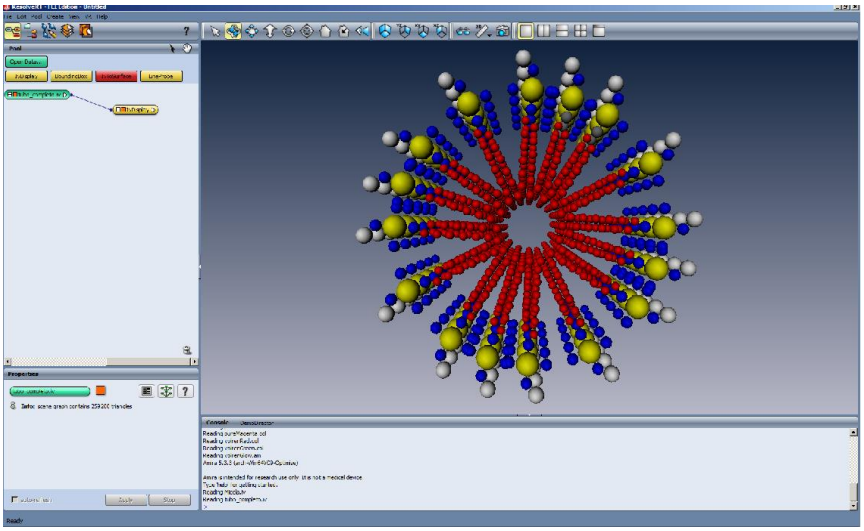


Figura 4.6.2. Tubo formado por Micelas de SDS

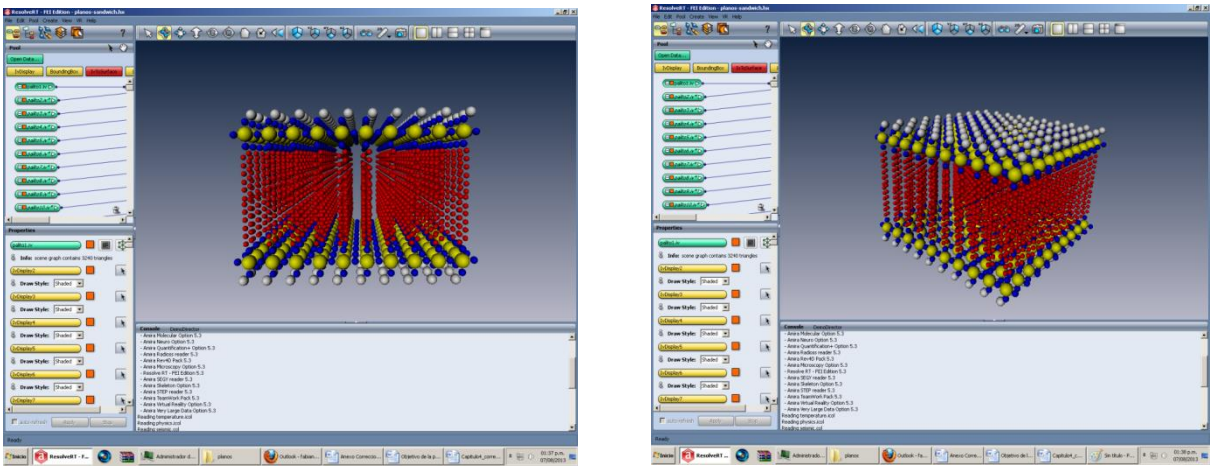


Figura 4.6.3. Plano formado por micelas de SDS.

A continuación se muestran las vistas para ser observadas con lentes anaglíficos

Figura 4.6.8. Micela

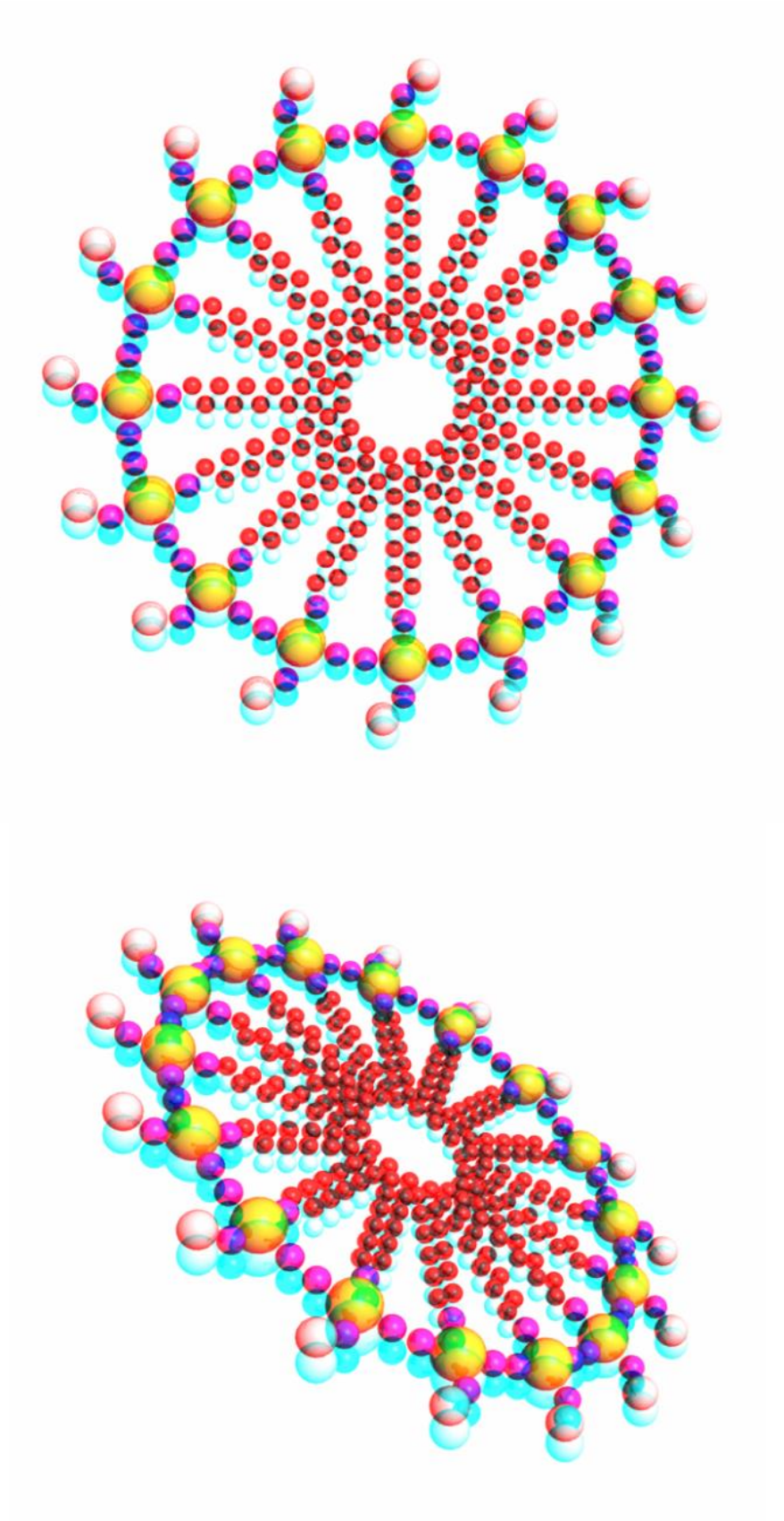


Figura 4.6.9. Micela con Óxido de Silicio

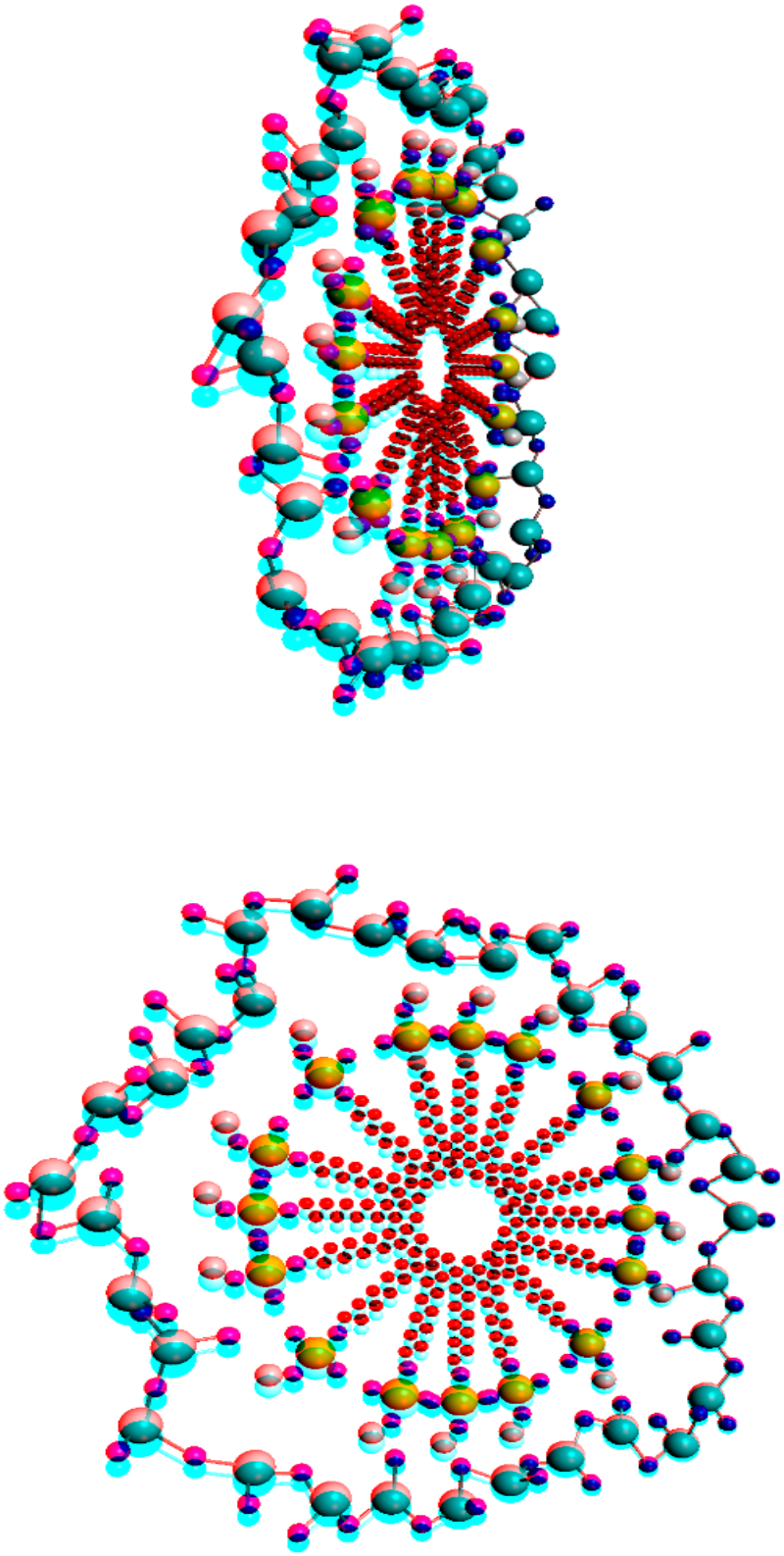


Figura 4.6.10. Plano de micelas de SDS.

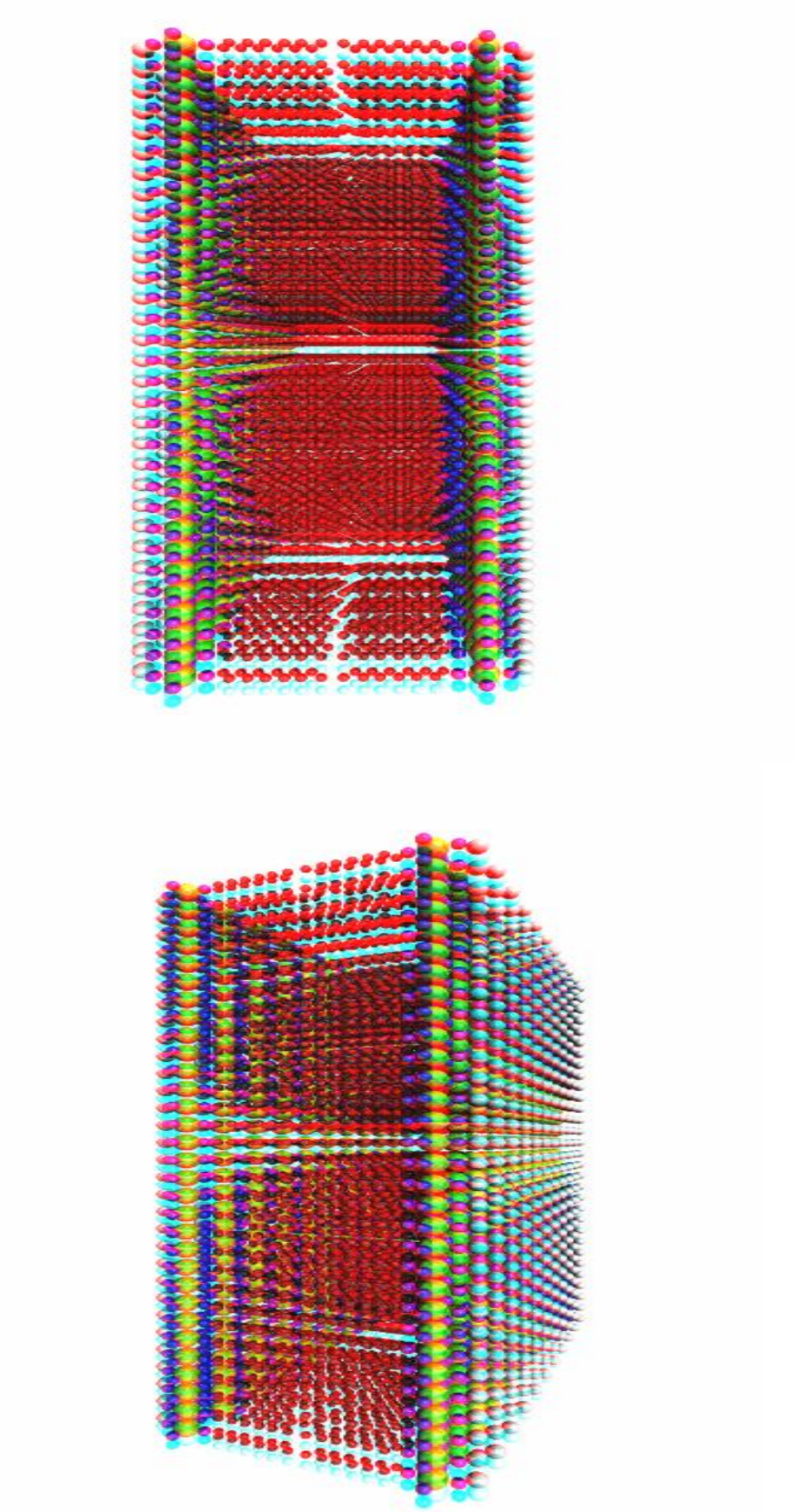


Figura 4.6.11. Plano de micelas de SDS con Oxido de Silicio.

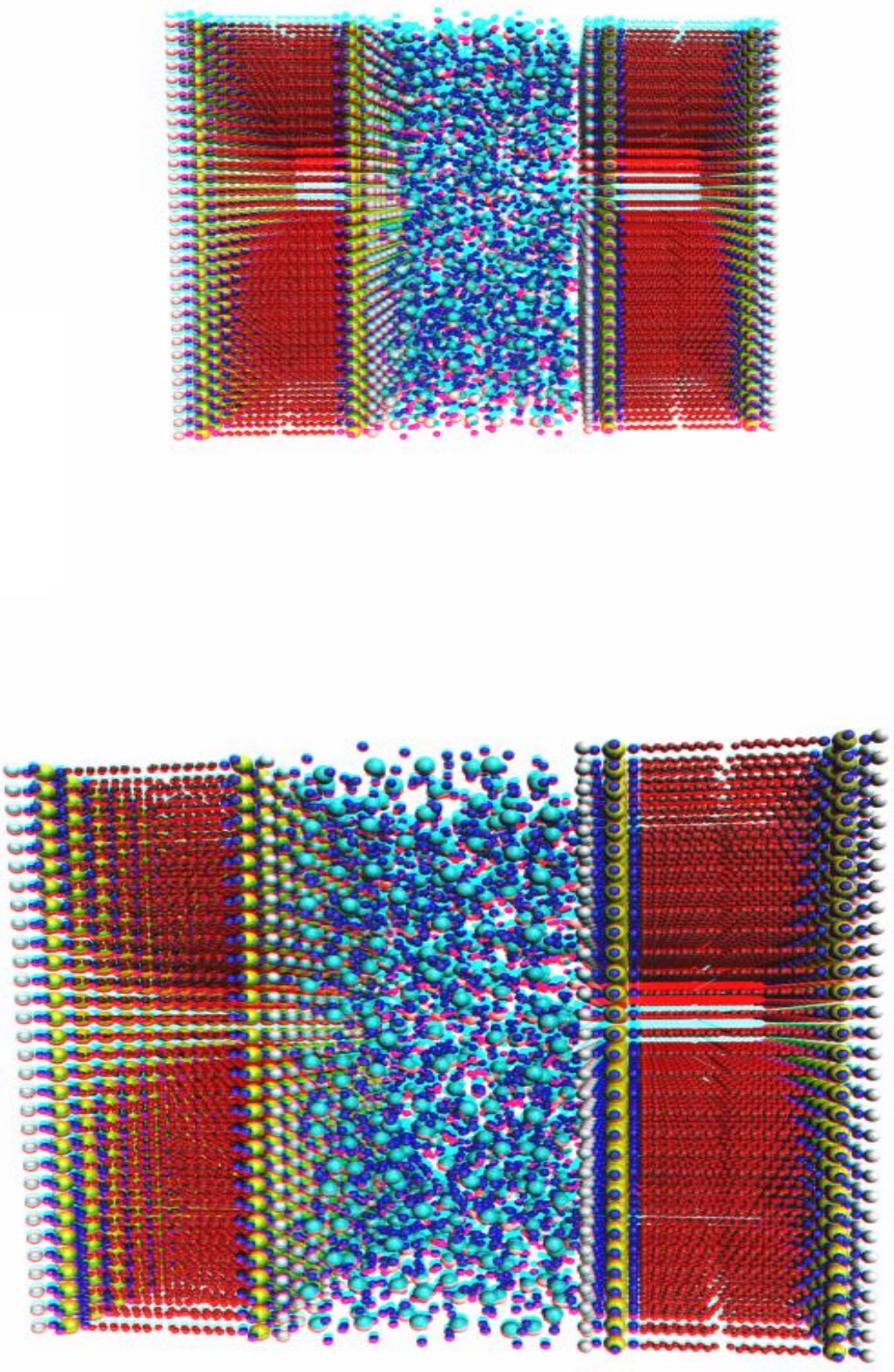


Figura 4.6.12. Tubo con Óxido de Silicio.

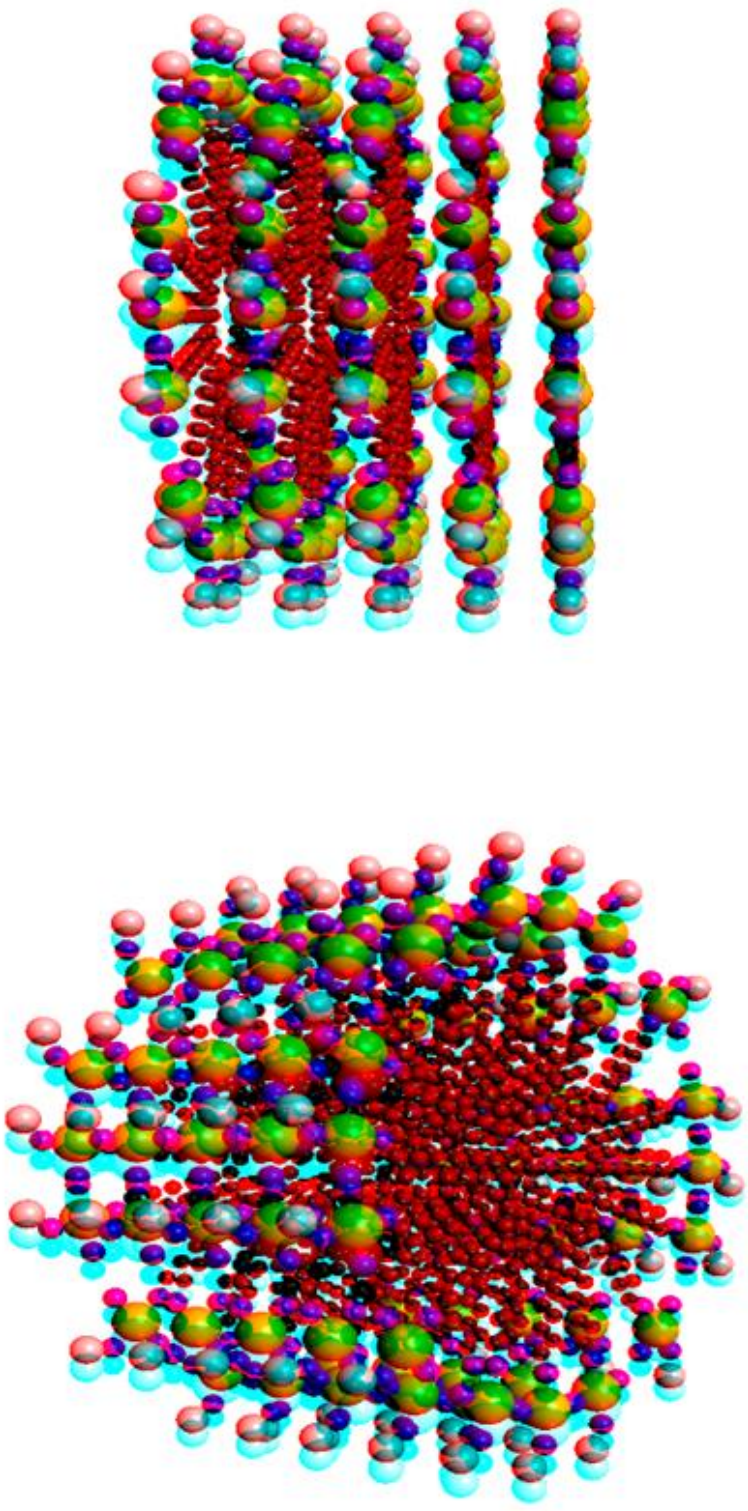
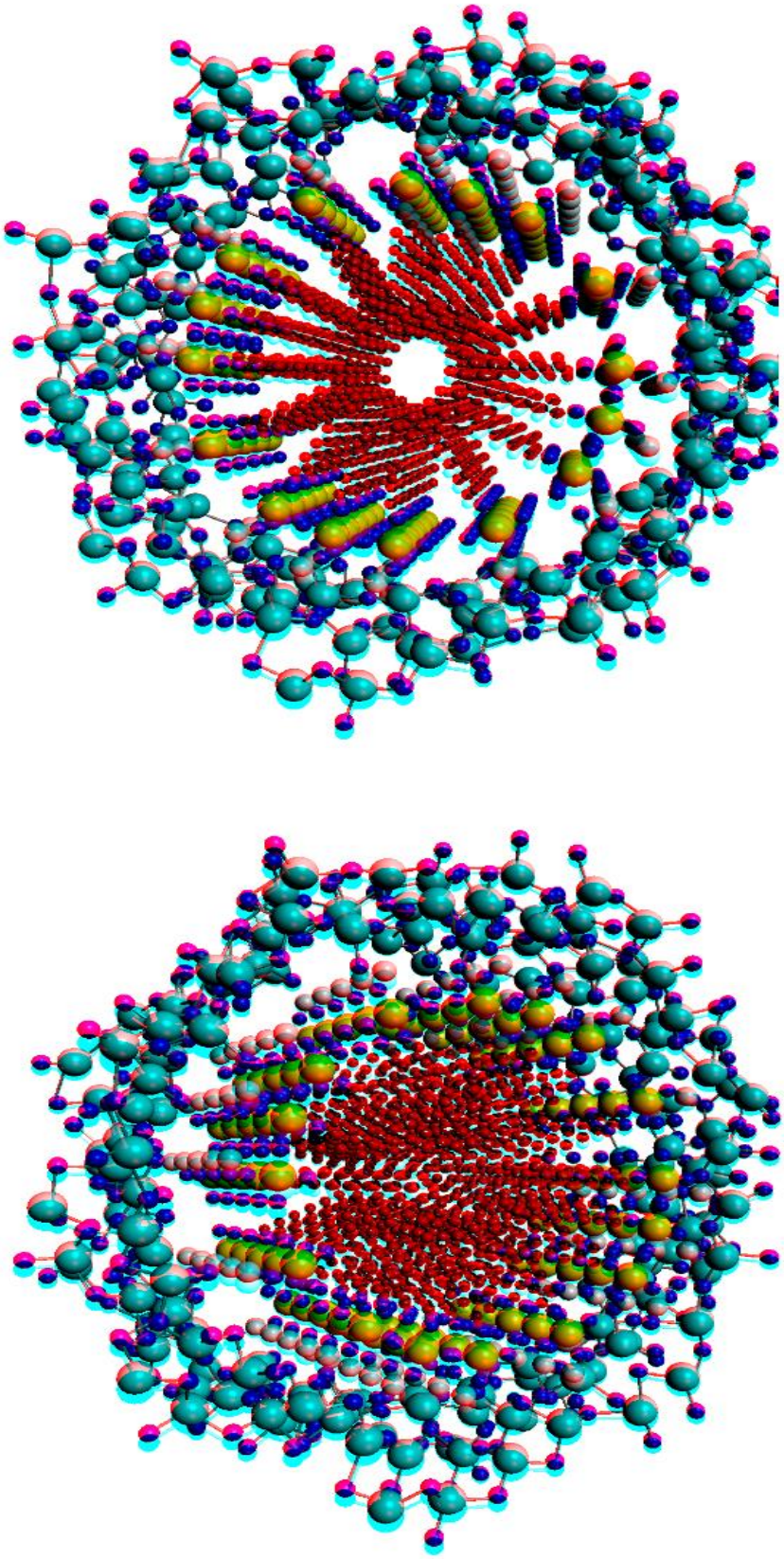


Figura 4.6.13. Tubo de micelas con oxido de silicio.



## CONCLUSIONES

En este trabajo abordamos el problema del desarrollo de materiales multimedia para ilustrar y ejemplificar los modelos teóricos y prácticos de experimentos que se desarrollan en el laboratorio de Fotónica de Geles del Instituto de Física en la U.N.A.M.

Para desarrollar este tipo de materiales nos ayudamos de la rama de la computación gráfica, involucrándonos en el ámbito de la visualización en 3D. En 1838, cuando se inventó el estereógrafo y se usaron gafas de colores como filtros con el objetivo de mejorar esta visualización, se desarrollaron varias técnicas para poder desplegar imágenes en 3D que dieran una mejor sensación de profundidad. Posteriormente se desarrollaron nuevas gafas y se mejoraron los filtros como son los lentes pasivos tanto anaglíficos como polarizados, hasta llegar a las gafas activas de cristal líquido que se usan actualmente

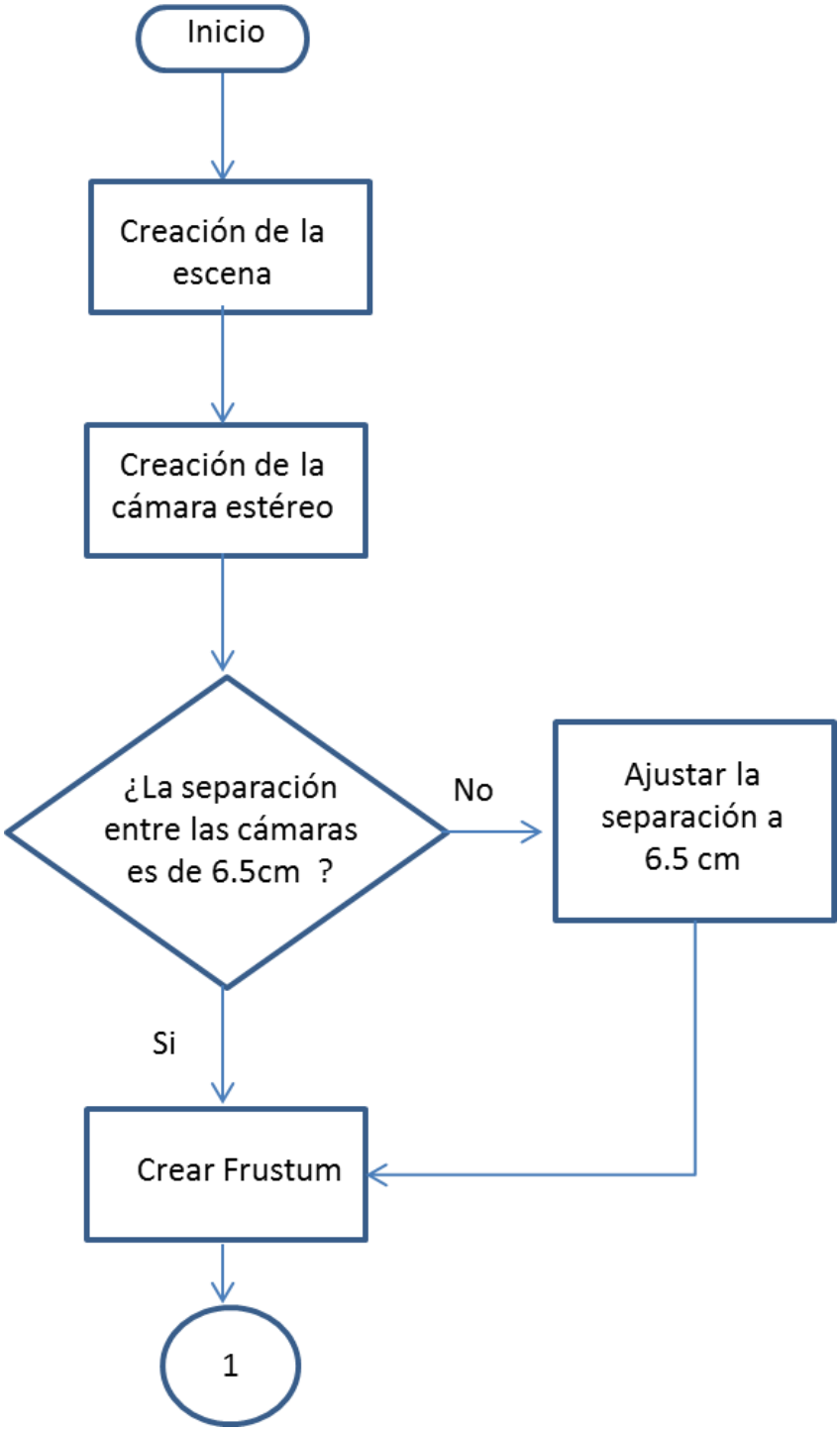
La investigación que se realizó en este trabajo involucró estudiar las tecnologías que actualmente se utilizan tanto en software como en hardware. Para lograr los efectos deseados se involucraron ambas partes, el hardware para poder obtener y visualizar los resultados esperados de los diseños realizados por software, además de poder mejorar su calidad a través de software de postproducción.

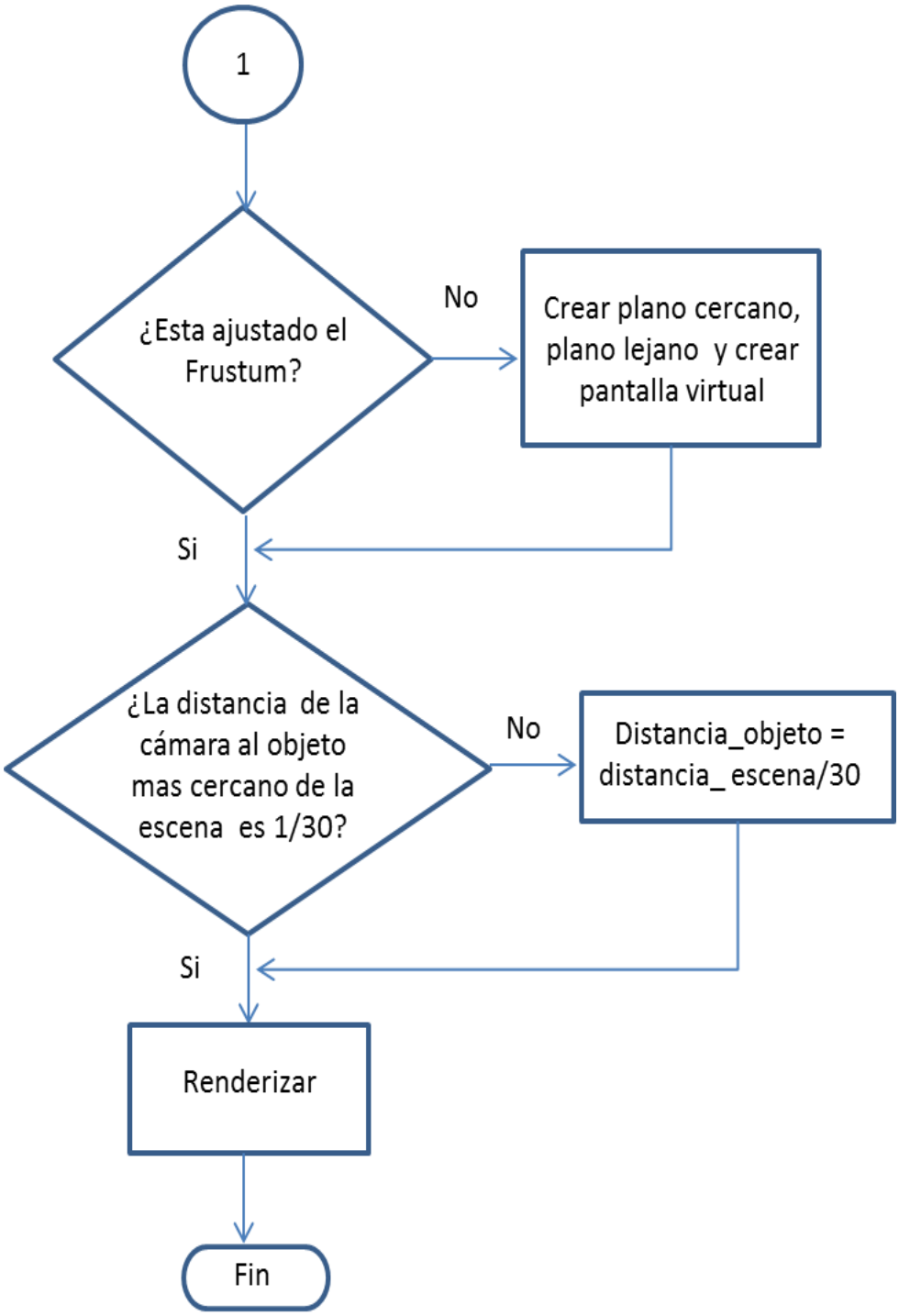
En este trabajo se desarrolló una metodología basada en investigación de software y hardware para generar un contenido multimedia y de imágenes 3D de última generación. Se compararon lentes activos y anaglíficos, para determinar sus diferencias en cuanto a calidad de visualización. Igualmente se compararon varios programas para obtener imágenes óptimas tanto en proyección activa como pasiva.

En el caso de la producción de contenido estereoscópico anaglífico; ya sea para gráficos por computadora o material impreso, se observó que una visualización apropiada depende en gran medida de la calidad del material de los lentes anaglíficos, ya que las pequeñas variaciones de la concentración del color en la superficie del filtro pueden afectar en gran medida la eficacia del mismo.

La producción de contenido multimedia estereoscópico se desarrolló a partir de pasos secuenciales que pueden describirse brevemente en el siguiente diagrama de flujo.

Diagrama de flujo para la obtención de una imagen s3D




1
¿Esta ajustado el Frustum?
No
Crear plano cercano, plano lejano y crear pantalla virtual
Si
¿La distancia de la cámara al objeto mas cercano de la escena es 1/30?
No
Distancia_objeto = distancia_ escena/30
Si
Renderizar
Fin

## ANEXO

En esta tesis se presentan imágenes estereoscópicas impresas por lo cual es necesario poder visualizarlas, esta visualización se hará empleando lentes anaglíficos. Como sabemos, estos lentes tienen dos colores que actúan como filtro de los colores de las imágenes que vemos; por lo que a continuación mostramos un experimento para definir con que tonalidad de colores deben de estar impresas esas imágenes para los lentes anaglíficos.

En este experimento se comparó la absorbancia entre unos lentes construidos con celofán y unos comerciales anaglíficos. Se realizaron mediciones de absorbancia a los filtros de ambos lentes. De esta forma se midió la saturación de los colores que se emplearon en este trabajo. Recordemos que el espectro de luz visible abarca las longitudes de onda de los 380 nm hasta los 750 nm como se muestra en la Figura A1.

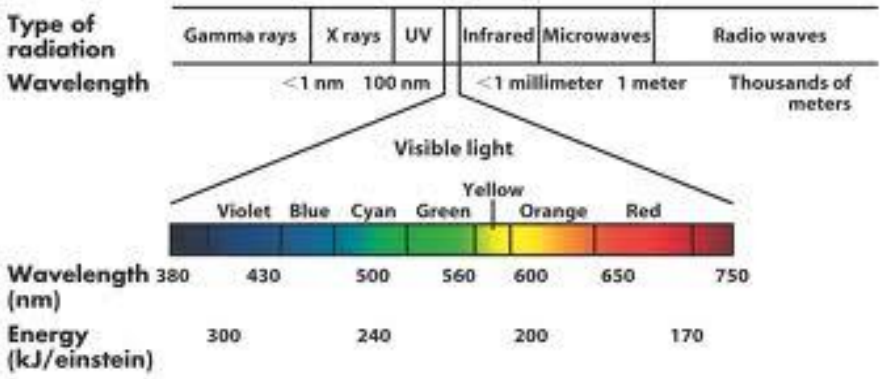
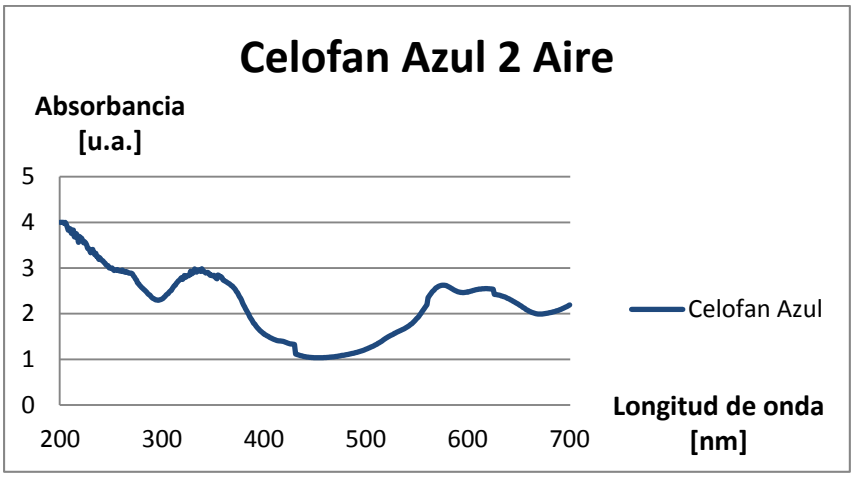


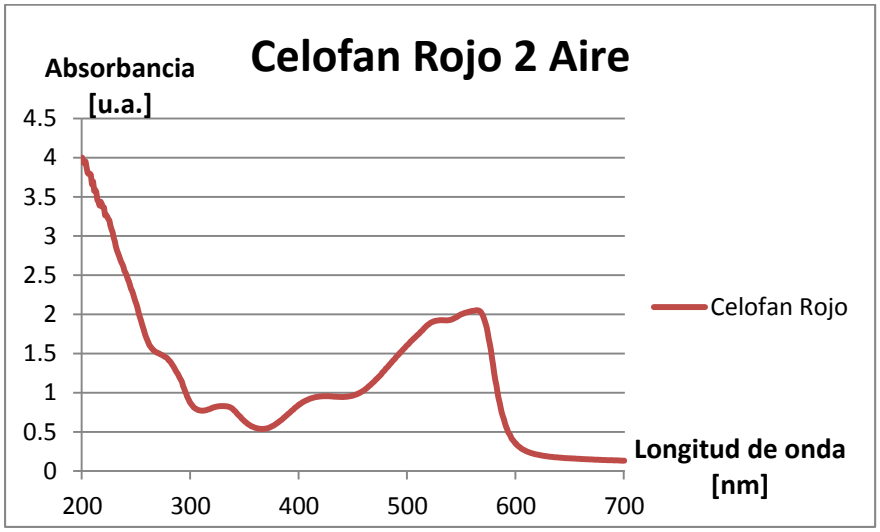

Figura (A1). Espectro de luz visible.

### Experimento de los filtros anaglíficos

La absorbancia se midió con el espectrofotómetro Genesis 2 de Themospectronic. Los resultados, tomando como referencia el aire son las siguientes gráficas (A1 y A2).

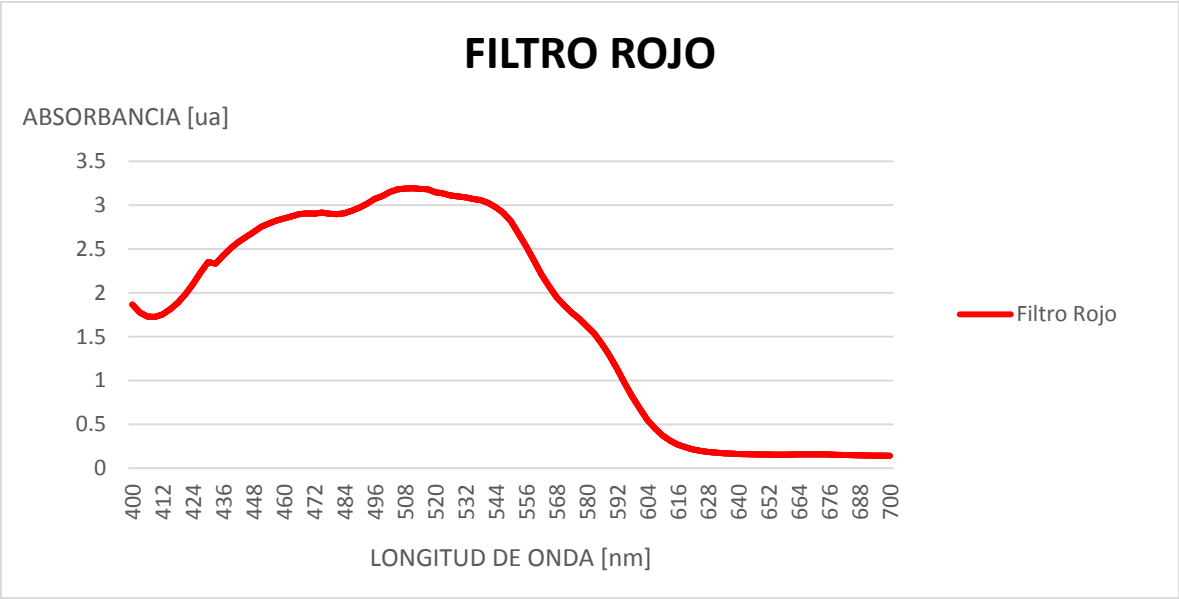



Gráfica A1. Celofán Azul

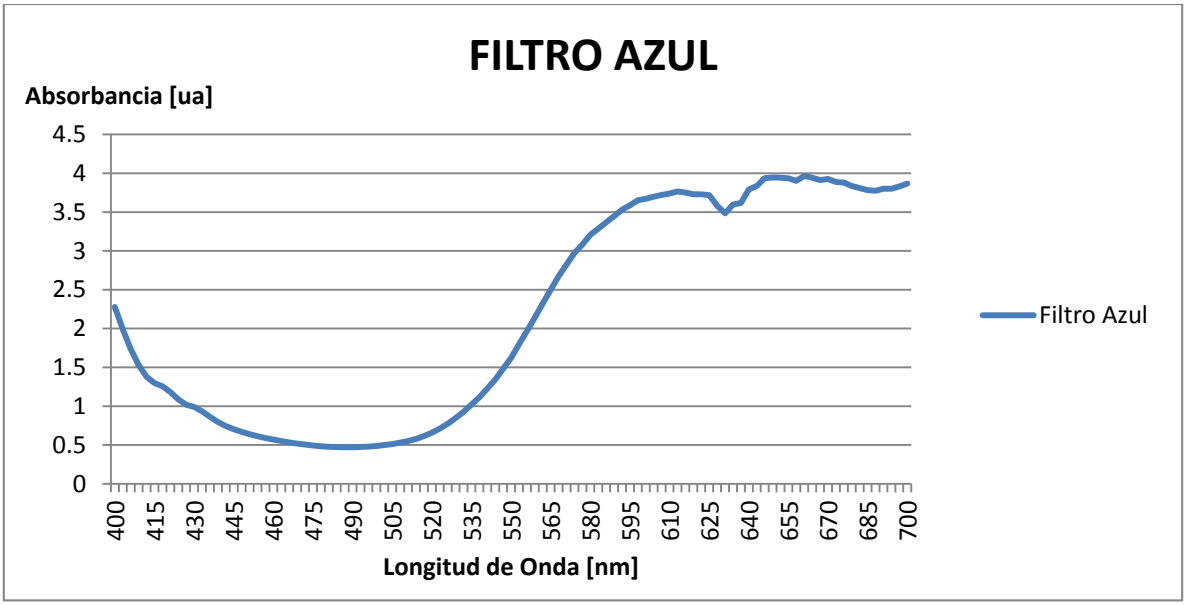



Gráfica A2. Celofán Rojo

Como vemos, los celofanes presentan bastante irregularidad en lugar de una curva suave, por lo que el efecto 3D no se podrá apreciar al usarse en los lentes 3D, posteriormente se midió la absorbancia a 2 filtros de unos lentes anáglifos y se observó que su distribución del color es homogénea y no como en los celofanes. Se obtuvieron sus espectros de absorción óptica presentados en las Gráficas A3 y A4.



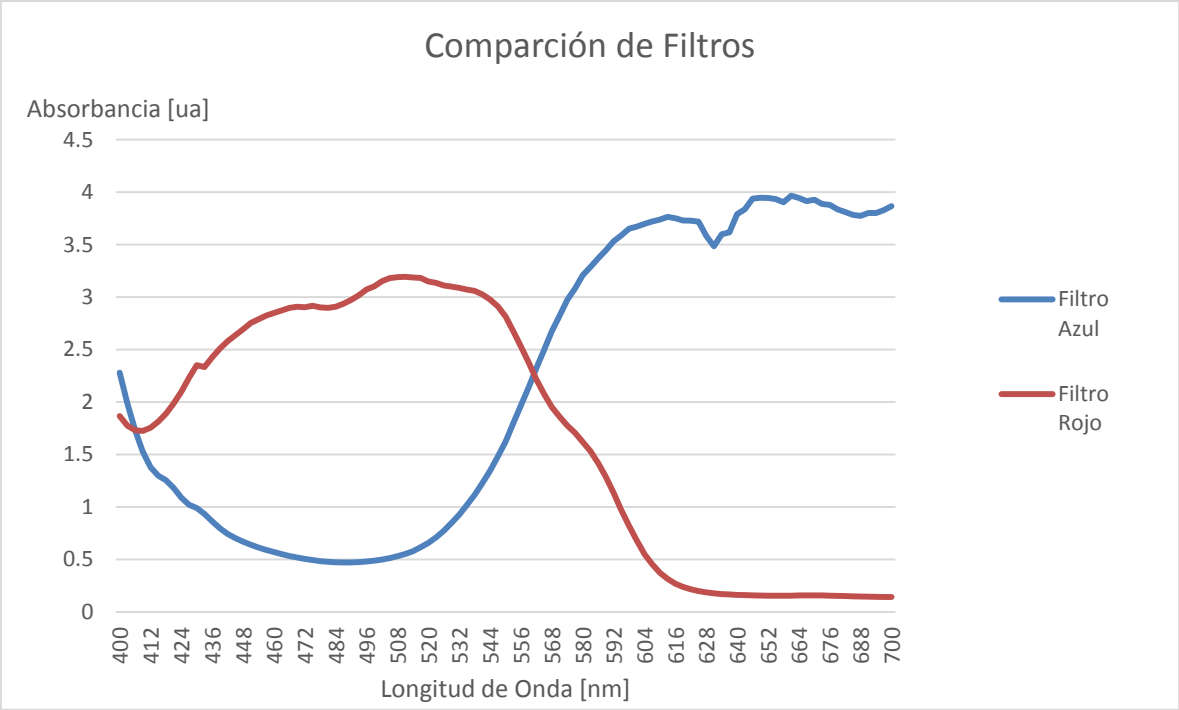

Gráfica A3. Filtro Rojo




Gráfica A4. Filtro Azul

Como podemos observar en estos filtros es más suave la absorbancia, y cuando observamos las gráficas podemos ver exactamente como los filtros absorben de los 410 [nm] a los 550 [nm] (Filtro Rojo) y de los 550[nm] a los 700[nm] (Filtro Azul) esto quiere decir que nuestros filtros absorben del color violeta hasta el color verde y del color amarillo hasta el color rojo respectivamente.

A continuación mostraremos la unión de las gráficas para que podamos apreciar los cortes de los filtros. Gráfica A5.




Gráfica A5. Unión de las gráficas de los filtros

**BIBLIOGRAFÍA**


1. Lenny Lipton., (1982*) "Foundations of the Stereoscopic Cinema, A Study in Depth"*, Ed. New York, N.Y., Van Nostrand Reinhold Company.

2. Scott Spencer, Eric Keller and Paul Gaboury., (2010) *"Zbrush Digital Sculpting, Digital Anatomy"*, Ed. Indianapolis, Inidana, Wiley Publishing Inc.

3. Ray Zone., (2007) *"Stereoscopic Cinema and the Origins of 3-D Film, 1838-1952"*, Ed. Lexington, Kentucky., The University Press of Kentucky.

4. Daniel Minoli., (2010) *"3DTV Content Capture, Encoding and Transmission, Building the Transport Infraestructure for Commercial Services"*, Ed. Hoboken, New Jersey., Jonh Wiley & Sons Inc.

5. Eddie Simmons, Form. Andrew Woods, (1992) *"The World of 3D Movies"*, Ed. Curtin University Technology.

6. Nick Holliman, Dr., (2005) *"3D Displays Systems"*, Ed. South Road, Durham, Department of Computer Science, University of Durham.

7. Fore June, (2011) *"An Introduction to 3D Computer Graphics, Stereoscopic Image, and Animation in OpenGL and C/C++"*.

8. Bernand Mendiburu, Yves Pupulin and Steve Schklair, (2012) *"3D TV and 3D Cinema, Tools and Processes for Creative Stereoscopy"*, Ed. Waltham, MA, USA., Elsevier Inc.

9. David S. Ebert ET. ALL., (2003) *"Texturing and Modeling, A Procedural Approach"*, Ed. San Francisco, CA., Morgan Kaufman Publishers.

10. William Vaughan., (2012) *"Digital Modeling"*, Ed. Berkley, CA., Pearson Education, Inc.

11. Bernard Mendiburu., (2009) *"3D Movie Making, Stereoscopic Digital Cinema from Script to Screen"*, Ed. Burlington, MA., Elsevier, Inc.

## MESOGRAFÍA


12. http://3dvision-blog.com/tag/quad-buffer-opengl-stereo/
13. http://3dvision-blog.com/tag/s3d-programming-guide/
14. http://www.visumotion.com/
15. http://www.stereoscopic.org/2012/program.html
16. http://developer.nvidia.com/siggraph-2011-stereoscopy-course
17. http://www.3dvisionlive.com/3dapps
18. http://www.stereoscopy.com/3d-books/photo-books.html#mounting
19. http://www.becot.info/stereo/anglais/3Dtheory.htm
20. http://drt3d.blogspot.mx/2008/02/what-is-best-stereo-base.html
21. http://nzphoto.tripod.com/stereo/3dtake/fbmarzio.htm#marziomacro
22. http://www.dashwood3d.com/blog/beginners-guide-to-shooting-stereoscopic-3d/
23. http://diloengrafico.wikispaces.com/El+mundo+gr%C3%A1fico+en+3D
24. http://www.stereo-3d-info.de/historie1-eng.html
25. http://www.depthcharge3d.com/3dphotography.shtml

26. http://3dvision-blog.com/2159-brief-stereoscopic-history-from-its-beginning-until-today/

27. http://docencia.izt.uam.mx/japg/Sites/Bioquimica/files/Agua/Micelas_en_agua.pdf