Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria
Secció d'Enginyeria Informàtica

Visualización estéreo

Memoria de proyecto de fin de carrera de Ingeniería Informática

Presentada por Alejandro López Polo y dirigida por Enric Martí

Bellaterra, 13 de febrero del 2009

RESUMEN

En este proyecto se muestran las posibilidades de la visión estéreo para la visualización en monitores tanto de objetos simples como de grandes escenarios, así como su aplicación en juegos o en otros ámbitos como el cine, la geología e incluso la medicina.

Para el desarrollo se ha usado una tarjeta con soporte 3d como la Nvidia 7600GT y una pantalla con una tasa de frecuencia alta como una ACER 19 pulgadas a 100Hz.

Los resultados sobre la visualización han sido extraídos de las opiniones de un grupo de 20 personas, de diversas profesiones, no relacionadas con los gráficos por ordenador.

RESUM

En aquest projecte es mostren les possibilitats de la visió estèreo en monitors per a la visualització tant d'objectes com de grans escenaris, així com la seva aplicació en jocs o en altres ambits com el cinema, la geologia i fins i tot la medicina.

Per el seu desenvolupament s'ha fet ús d'una targeta amb suport 3d com l'Nvidia 7600GT i una pantalla amb una alta tassa de freqüència com una Acer 19 polzades a 100hz.

Els resultats de la visualització s'han extret de les opinions d'un grup de 20 persones, de diverses professions, no relacionades com els gràfics per computador.

SUMMARY

This project shows the capabilities of stereo vision in both monitors to display simple objects as large scenarios, and their application in computer or other like areas film, geology and medicine applications.

For development we used a graphic card with 3d support as Nvidia 7600GT and a screen with high refresh rate as Acer 19 inch 100hz.

The results on the display have been drawn from the opinions of a group of 20 people from various professions, not related to computer graphics.

Agradecimientos

A mi familia por el apoyo mostrado todos estos años.

A Enric Martí por sus consejos, sabiduría y paciencia infinita.

A todos aquellos que me han ayudado, y aportado nuevas ideas.

Índice

Introducción

La visión humana se diferencia de otros animales por la situación de los ojos en la parte frontal del cráneo. Esto nos permite ver los objetos que tenemos delante de nosotros con ambos ojos. A esto se le llama visión binocular. En el caso particular de los humanos, la separación entre los dos ojos nos capacita para ver elementos desde dos puntos de vista diferentes, envolviendo nuestra vista al objeto. La información captada por los ojos es enviada al cerebro, el cual procesa dicha información decidiendo qué objeto está delante y cual está detrás.

Los gráficos por computador han ido ganando en importancia en los sistemas de computación. En su creación como un periférico más a principios de los 70, la tarjeta gráfica estaba pensada para enviar a la pantalla las 25 líneas de texto visibles por aquel entonces en blanco y negro.

Ahora, año tras año, salen miles de juegos, con altos requerimientos de computación que han hecho que las tarjetas gráficas vayan evolucionando hasta límites de procesamiento inimaginables y resoluciones impensables como la tan famosa Alta Definición [1].

Actualmente no basta con tener procesadores de doble núcleo o cuádruple si no que resulta normal en equipos pensados para jugadores empedernidos la colocación de varias tarjetas gráficas para duplicar su potencia (mediante conexiones SLI en Nvidia[2] y CrossFire en ATI-AMD[3]) y también su consumo. No hay que olvidar que el consumo de la gama alta de estas tarjetas se sitúa sobre los 300W [4] cuando hasta ahora el total de un ordenador nunca a supuesto la necesidad de colocar un transformador de más de 350W. Así que un equipo dotado de la última tecnología con, por ejemplo, 3 gráficas, a pleno rendimiento jugando, con programas de diseño o comprimiendo video, sobrepasa claramente los 1200W [5].

Resúmenes de trabajos sobre Estéreo

En los 70 con la aparición de nuevos entornos gráficos, no basados en texto, y los primeros juegos, surge la idea de explotar la capacidad visual del ser humano. Y con ellos los primeros intentos en representar figuras en 3D en una computadora.

Los primeros estudios no se hicieron esperar. En ellos se buscaba la forma de representar los objetos en la pantalla para que el cerebro interpretara su situación tridimensional tanto fuera como dentro de la pantalla, simulando que el objeto podía tanto salir, como entrar en el monitor [6].

Más entrado en los 90 se intenta simular la visión humana en robots, dotados con dos cámaras simulando los ojos y programando los algoritmos necesarios para analizar las imágenes y saber con exactitud la distancia a la que se puede encontrar de ciertos objetos [7].

Algunas videoconsolas incorporan cierta tecnología de visión, como la PlayStation2 con su cámara EyeToy, para ver los movimientos de los jugadores. Esta cámara no presenta visión binocular, con dos lentes, así que para averiguar si un objeto se aleja o se acerca, hacen detección de objetos y según su tamaño al moverse (si crece, se acerca, si se hace pequeño, se aleja) determinan su movimiento.

Reproducir visión estéreo por ordenador

El mayor problema para poder simular un escenario en 3 dimensiones con la pantalla del ordenador es que ambos ojos están viendo lo mismo. Por lo tanto no se puede recrear un espacio tridimensional.

Hay formas de poder engañar al cerebro mediante dibujos realistas como los trampantojos o las ilusiones ópticas jugando con formas y colores. Los trampantojos [8] no dejan de ser dibujos pintados en paredes o suelos, recreando un escenario o lugar de forma muy realista, que mirándolos desde cierto punto de vista puede hacer creer que existe una cierta profundidad, tal como se muestra en las figuras 1a y 1b. Las ilusiones ópticas [9] solo sirven para engañarnos acerca de colores y formas de diferentes objetos, dependiendo siempre de las características visuales de la persona que mira el dibujo.

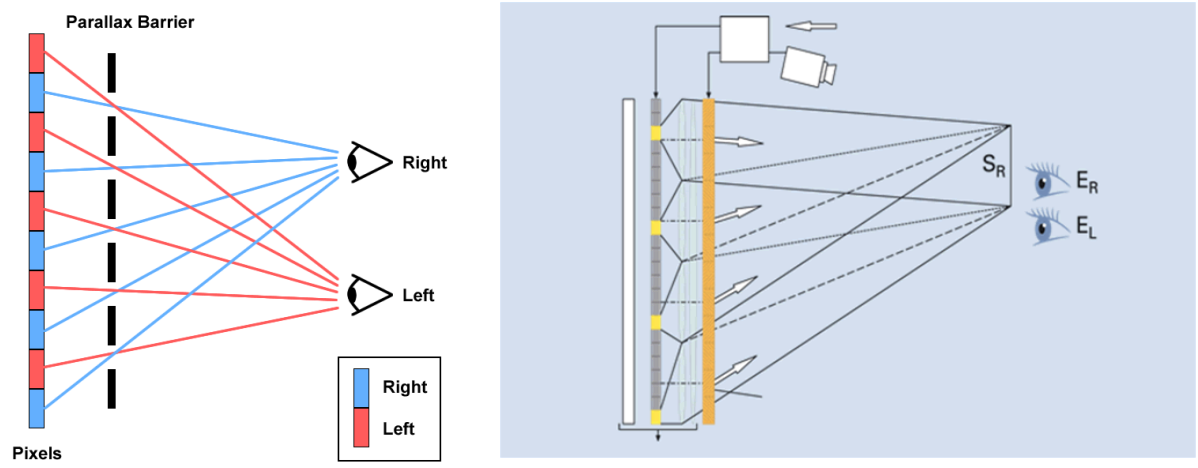
a b

Figura 1: (a): Entrada de un parking dibujado sobre la puerta del mismo.(b): Muro sobre el cual se ha pintado un camino.

Centrándonos en el problema real, para obtener una visión estéreo hay que mostrar a cada ojo una imagen diferente. Hay varias tecnologías que permiten hacer esto, cada una con sus ventajas e inconvenientes.

Como método más primitivo nos encontramos el estereoscopio [10]. Se trata de unas gafas en los cristales de las cuales se sitúan diferentes imágenes, como diapositivas, correspondientes a lo que vería cada ojo en la realidad, es decir, si no llevara gafas vería diferentes objetos desde diferente punto de vista con cada ojo. Esto ha evolucionado hasta el punto de llegar a gafas donde, en lugar de fotografías estáticas, nos encontramos con dos pequeñas pantallas de 22 milímetros de diagonal, una para cada ojo, de resoluciones superiores a 1280x1024 píxeles, que nos permiten mostrar video a gran calidad, como en la figura 4a. O también las gafas de realidad aumentada [11], que además de ver objetos programados por ordenador, se puede ver la realidad, ya se mediante cámaras en cada lente o lentes semitransparentes que mezclen las dos visiones. El gran inconveniente de esta tecnología es el alto precio de las gafas virtuales, que oscilan entre 800€ las más sencillas, hasta los 10.000€ las más sofisticadas.

Otro método para mostrar a cada ojo imágenes diferentes es a través de monitores auto-estéreo [12]. Estos monitores emiten 2 imágenes a la vez, cada una de las dos imágenes perceptible solo desde un solo ojo, según se muestra en la figura 2a. La facilidad de implementación es una de las ventajas, ya que incluso encontramos en el mercado, portátiles cuya pantalla usa esta tecnología o móviles de la marca Sharp como en la figura 4b. Uno de los principales problemas es el coste de estos monitores ya que estos son excesivamente caros. Y otro inconveniente es el hecho de que la información es enviada a cada ojo, es decir, el usuario que esté mirando el monitor no puede moverse, puesto que esto implicará que los ojos no podrán ver la imagen y se perderá el efecto estéreo. El sistema actual para que el visor pueda moverse, es una pequeña cámara encima de la pantalla, que capte la situación de la persona, la posición de los ojos, y según el movimiento, se mueva la pantalla, para que el observador no pierda la visión, figura 2b. Además no está pensado para un público masivo, solo puede verse por unas pocas personas.

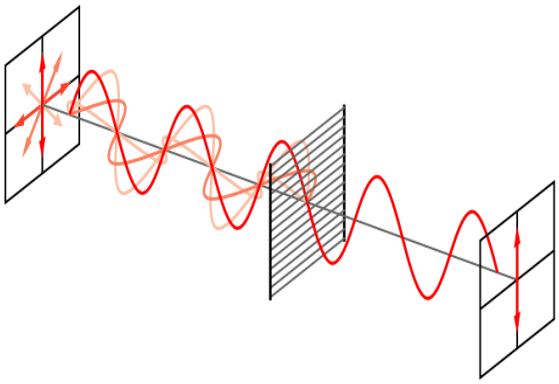


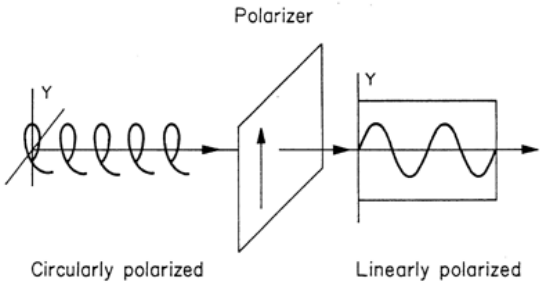
a b

Figura 2: (a): Monitores auto-estéreo. La rejilla anterior evita que un ojo vea la información del otro. Varias personas pueden ver lo mismo, si estos se colocan bien. (b): Si se mueven las personas, la cámara superior a la pantalla, hace mover ésta, hasta la posición de correcta visualización.

Un sistema parecido al anterior pero usando gafas sería el polarizado. En este caso tratamos con gafas de polarización pasiva. La luz es una onda emitida en varias direcciones y este sistema eliminaría direcciones sobrantes. Las figuras 3a y 3b contienen filtros polarizadores que dejan pasar ondas verticales. Por lo tanto, con un monitor adecuado se podrían emitir dos señales simultáneas, cada una de estas dos señales en direcciones diferentes, y que se filtrasen a través de las gafas, dejando pasar una señal concreta a cada ojo. La resolución es muy buena, y el costo no es excesivo. Solo requeriría de un monitor con estas características. Este sistema es el usado actualmente en los cines IMAX en España. Un ejemplo de monitor polarizado sería el de la figura 4c.

Las tecnologías mostradas hasta ahora requieren hardware concreto, monitores especiales o gafas con mini-pantallas. No todo el mundo puede permitirse los altos costos que representan estos métodos de visión estéreo, y tampoco es necesario para poder aprovechar al máximo nuestra visión. Existen métodos más rudimentarios y menos costosos que nos darán las mismas posibilidades o incluso más de las anteriormente descritas.





3a 3b

Figura 3: (a) y (b): El polarizado filtra las direcciones de ondas que forman la luz, dejándolas pasar en una dirección.

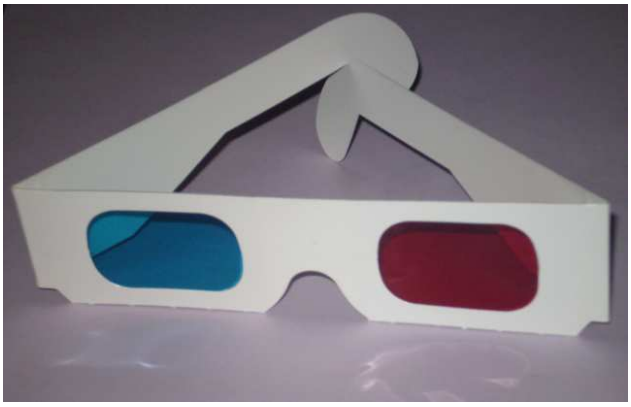
El sistema anaglifo [13] se basa en el uso de gafas, cuyas dos lentes están tintadas de colores opuestos, es decir, una lente roja y la otra cyan o lentes verde-magenta, figura 5a. No hace falta ningún monitor especial, sólo tratar las imágenes de cada ojo con el color necesario para crear el efecto. Las gafas pueden ser de cristal o incluso desechables, como la figura 5a, hechas de forma casera [14]. El único inconveniente que se encuentra es la falta de apreciación de los colores que no es buena, aunque el resultado final no es malo. Lo ideal para este método sería mostrar por la pantalla imágenes en escala de grises.

a b c

Figura 4: (a): Gafas virtuales. (b): Móvil Sharp con tecnología auto-estéreo. (c): Pantalla polarizada para gafas de polarización pasiva.

Aparte del polarizado pasivo encontramos lo que se llamaría el polarizado activo. Se trata también de gafas de cristal líquido, la figura 5b es un ejemplo, que dejan pasar la luz o no, dependiendo de la imagen mostrada por la pantalla. Si me muestra la imagen del ojo derecho, el cristal del ojo izquierdo se oscurecerá, negando la visión de tal ojo. Al instante siguiente, se oscurecerá el cristal del ojo derecho la pantalla mostrará la imagen el ojo izquierdo. Así se juega con la capacidad de retención de imágenes por parte del cerebro. El intercambio de imágenes tiene que suceder de forma muy rápida, ya que si no el cerebro notaría cortes entre una imagen y otra, lo que tendría que ver con el teorema del muestreo de Nyquist [15], para lo cual, las imágenes tendrían que ser expuestas al ojo el doble de rápido que el muestreo que el ojo realiza. Las televisiones actuales tienen una frecuencia de 60Hz, pero, se necesita de un monitor con una tasa de refresco superior a 120Hz, para que cada ojo tenga la sensación de

estar recibiendo la información a 60Hz como si de una televisión o monitor se tratara. El coste para disfrutar de esta tecnología supondría comprar un monitor de alta tasa de refresco y unas gafas polarizadas, cuyo precio es muy asequible.



5a

5b

Figura 5: (a): Gafas para dibujos anaglifos, hechas de papel. (b): Gafas de polarizado activo, con cristales líquidos.

Como se puede ver, la visión estéreo no es una tecnología de nueva aparición, lleva muchos años en el mercado y además van saliendo cosas nuevas, está en continua evolución. Hemos pasado de un estereoscopio hecho con diapositivas a unas gafas de cristal líquido, y próximamente gafas de realidad aumentada.

Podríamos dejar de banda en la comparativa a sistema anaglifo, dada la peor coloración que sufren los objetos vistos a través de unas gafas. Pero no olvidemos que actualmente, sería el único sistema que no necesitaría de un hardware adicional para ver la televisión, por ejemplo, y que solo dependería de que el emisor enviara la señal en este sistema.

Si quisiéramos ser algo más exigentes, como por ejemplo un videojuego de última generación, lo recomendable serían unas gafas de cristal líquido, o monitores de polarización pasiva. Con ellos puedes moverte libremente, evitando perder la percepción 3D ante un pequeño sobresalto. Sin perder de vista juegos como los de la consola Nintendo Wii, en los que puede ser necesario estar de pie e ir de un lado a otro de una habitación. Esto sería impensable con monitores auto-estéreo, más enfocados a una visión placida en el sofá de casa.

No podemos obviar las gafas de realidad aumentada, más enfocadas de añadir información en nuestro campo de visión [16].

El **objetivo del proyecto** es la implementación de diferentes algoritmos estéreo dentro de una aplicación gráfica, con el fin de comparar dichos algoritmos en diferentes escenas y apreciar las ventajas de la visión 3D. El proyecto esta compuesto por los siguientes requerimientos y funcionalidades:

- Anaglifos
- Gafas polarizadas
- Entorno API WIN32
- OpenGL
- Desarrollo Visual Studio 2005

- Visor ficheros 3DS
- Lectura de texturas

En el capitulo 1 explicaremos en que consiste la visión 3D, y como podemos conseguir esta visión por ordenador. En el capitulo 2 explicaremos los métodos estéreo implementados en la aplicación creada durante el proyecto. El capítulo 3 se centrará en desarrollo de los algoritmos estéreo, como se ha implementado y que funcionalidades posee la aplicación. El resumen con las pruebas realizadas y sus resultados centraran el capítulo 4.

En los anexos I y II enumeraremos los pasos a seguir para configurar una tarjeta gráfica y detallaremos los cálculos para los algoritmos estéreos, respectivamente. El anexo III servirá como manual de usuario para manejar todas las funciones de la aplicación.

Capítulo 1 Visión 3D

Cuando se habla de visión binocular en los humanos, se esta hablando de captar dos imágenes, cada imagen desde sitios diferentes, y deducir de ambas imágenes un escenario donde los objetos puede estar más lejos o más cerca. El ser humano, a través de sus dos ojos puede captar estas imágenes y gracias al cerebro, juntarlas. Así puede analizar el entorno que le rodea y determinar distancias entre objetos.

Dentro del campo de la visión binocular podemos tratar dos campos: el hecho de reproducir imágenes a personas mediante dispositivos como pantallas o gafas y la adquisición de datos mediante dos cámaras que simulen los ojos humanos.

Este apartado se divide en:

- Métodos de visualización estéreo en computadores
- Adquisición estéreo
- Futuro

1.1 Métodos de visualización estéreo en computadores

A la hora de reproducir imágenes, aparecen en los 70 los primeros sistemas estéreos, entre ellos las gafas anaglíficas y las primeras patentes [16] sobre gafas de cristal líquido. Pero no llega hasta finales de los 90 [17] al mercado unas gafas de estas características. Estas gafas vienen de la mano de Nvidia, hardware del cual es necesario para disfrutar de las ventajas que estas gafas aportan a la hora de visualizar la pantalla, hasta ese momento.

El sistema de gafas anaglíficas consiste en que los cristales son de colores opuestos. Mientras que en un ojo tenemos el color rojo, en el otro su negado, el cyan. También existe la alternativa de cristales azul-amarillo o verde-magenta. Si mirásemos a través del cristal rojo, el color rojo y el blanco no se distinguirían. Por lo tanto, lo rojo sería blanco. Y a través del ojo con filtro cyan pasaría lo mismo. Por tanto, las zonas de un dibujo donde los dos colores coincidieran, sería blanco. Se puede comprobar mirando el siguiente dibujo, figura 6.

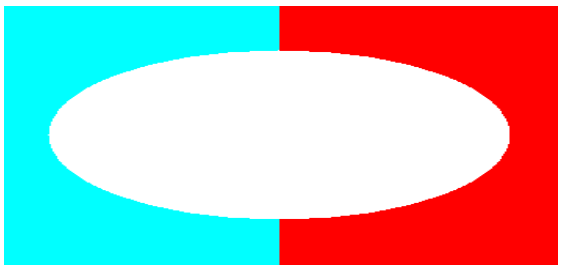
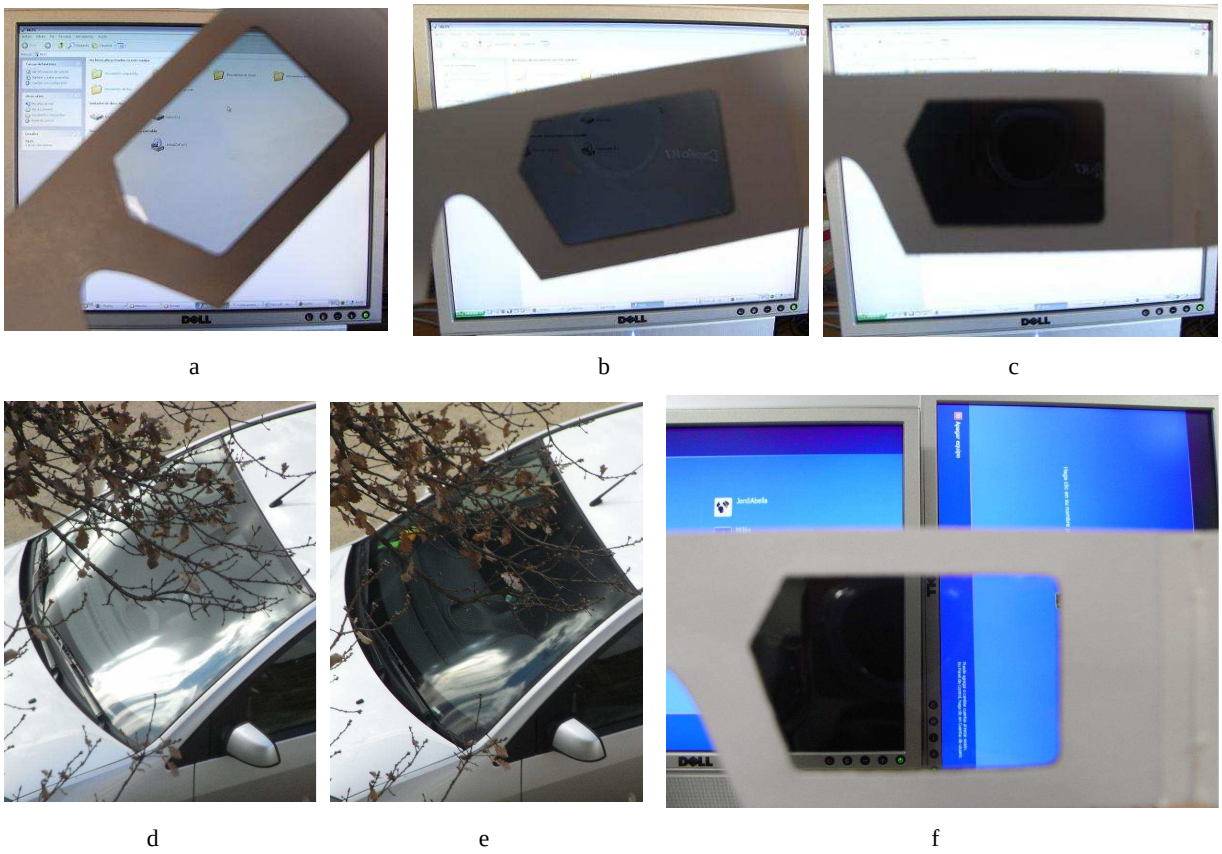


Figura 6: Para comprobar como el color rojo es imperceptible a través de la lente roja, habrá que coger unas gafas anaglíficas, y cerrar el ojo que coincida con el filtro cyan. Así se podrá ver, que el lado derecho de la imagen apenas se distingue del ovalo central

Dentro del grupo de gafas polarizadas, nos encontramos dos modelos. Las gafas de polarizado pasivo son semejantes a las gafas de sol. Consisten en lentes que no dejan pasar las ondas de la luz que van en cierta dirección. Según se vayan inclinando las gafas, dejara pasar unas ondas u otras. Haciendo la prueba con una pantalla LCD, figura 7a, que emite en una sola dirección, se comprueba que poniendo las gafas inclinadas, éstas dejan pasar

completamente la imagen, mientras que si se van poniendo en posición normal van filtrando más ondas, figura 7b, hasta no deja pasar nada, figura 7c. Para polarizar la luz solar, hace falta reflectarla en algún material, como la carrocería de un coche, en la figuras 7d y 7e, es entonces cuando se puede ver el efecto. Existen en el mercado monitores LCD, donde cada línea de píxeles emite en direcciones perpendiculares, por lo tanto, si en cada ojo ponemos filtros también girados perpendicularmente, cada ojo recibiría imágenes diferentes. Sería como inclinar una pantalla LCD, junto a otra no inclinada, al mirar por la misma lente, en un monitor se verían los píxeles y en el otro no, como en la figura 7f.



a b c

d e f

Figura 7: Se puede ver en las diferentes imágenes el efecto del polarizado.

Las gafas de polarizado activo consisten en dos lentes de cristal líquido. Estas lentes reaccionan a una corriente eléctrica, oscureciendo los cristales, filtrando así la luz, no dejándola pasar. Los cristales se irán oscureciendo alternativamente, así que solo se podrá ver por uno de los ojos, ya que el otro se encontrará con el cristal oscurecido. Para ver imágenes diferentes en cada ojo, las gafas tienen que saber que imagen se esta proyectando en ese momento por el monitor. Hay varias formas de conectar ambos elementos. El más primitivo es la conexión TRS o Jack, como la conexión usada en los auriculares, así como la también existente conexión MiniDIN-3. Ambas necesitan cableado.

Otra alternativa en la sincronización por infrarrojos, usada en contados cines IMAX, dado que la gran mayoría usa la polarización pasiva [18]. Para poder conectar las gafas al ordenador a través de estos conectores existen varias posibilidades. Actualmente solo hay algunas tarjetas graficas, como por ejemplo la ATI v7100 [19], que incluyen conectores MiniDin-3 para su soporte Stereo, figura 8a, aunque estos conectores no suelen ser habituales, por lo que se necesita un aparato especial. Este aparato lo llamaremos “dongle”,

tal como aparece en los manuales de instalación. Dicho aparato contiene una entrada de datos, como puede verse en la figura 8b, que se conecta a la salida VGA de la tarjeta gráfica. No importa si la grafica sólo tiene salida DVI, ya que existen adaptadores para pasar de DVI a VGA que no modifican para nada el normal funcionamiento de las gafas. Como salidas contiene una conexión VGA, un conector Jack y otro MiniDIN-3. La idea de cómo funciona la circuitería interna podría resumiste en algo así como que la señal VSync (pin 14 del conector VGA) genera un pico cada vez que se va a refrescar la pantalla. Así, cada vez que refrescamos la pantalla, cambiamos la imagen que mostramos para cada ojo. Otro de los pins decidiría que ojo puede ver y cual no, con un estado 0 o 1, dependiendo si mostramos la imagen del ojo derecho el izquierdo. Aunque a la hora de la verdad esto es algo mas complejo [16].

a b

Figura 8: (a): ATI V7100 con dos salidas DVI y una conexión MiniDin-3. (b): Dongle con conexiones VGA, TRS y MiniDin-3.

Ahora ya sabemos que los ojos van a estar recibiendo información de forma alterna, pero, ¿ha que velocidad?

En el mercado tenemos gran variedad de monitores, entre los que se encuentran los CRT [20] y los TFT-LCD [21]. Los primeros se basan en unos rayos que son lanzados contra una pared de fósforo provocando la aparición de colores en zonas y por tanto formas. Estos rayos van de izquierda a derecha de la pantalla y de arriba abajo, según se les va mandando información. Por tanto, la tarjeta gráfica es la que marca el actualizado de las imágenes que vayan apareciendo. De los monitores TFT-LCD se podría decir que se tratan de millones de pequeñas bombillas de color, que al recibir una corriente se iluminarían. Para definir las características de ambos tipos se definen varios parámetros: Resolución, profundidad de color, frecuencia de refresco, etc.

La resolución y la profundidad de color no implican problemas a la hora de trabajar con unos monitores u otros en este proyecto, aunque las ventajas de los CRT se hacen notar. Pero la frecuencia de refresco si. En el sistema Europeo, las televisiones actualizan su imagen 60 veces por segundo.

El teorema de Nyquist-Shannon [15] dice que para poder reconstruir una señal, está debía ser muestreada al doble o más de frecuencia de la señal original. Una imagen no deja de ser una

señal, y por tanto, a la hora de emitirla por un medio de comunicación como la televisión, hemos de emitirla también al doble de frecuencia de la que nuestros ojos muestrean. Actualmente los monitores CRT y TFT-LCD trabajan con una velocidad de refresco de 60Hz o superior. El problema llega cuando se necesitan 60Hz por cada ojo. En otras palabras, necesitamos que cada ojo reciba la información a 60 imágenes por segundo, pero entre una imagen y otra el ojo tendrá una pantalla opaca que no le dejará ver el monitor, mientras el otro ojo capta su imagen. Por lo tanto, necesitamos 60+60Hz. Un total de 120Hz de refresco de refresco de pantalla. Aquí es donde los monitores TFT-LCD no pueden competir con los CRT, ya que no alcanzan esta velocidad de refresco.

También la persistencia del color en los TFT-LCD, es decir, el tiempo que tarda en apagarse y encenderse los pixeles es superior, lo cual genera un efecto llamado *ghosting*, estela de luz que no logra irse de la pantalla, cuando el otro ojo está visualizando su imagen. Así, para poder utilizar esta tecnología es preciso usar un CRT con alta frecuencia de refresco. Durante el proyecto se ha usado una pantalla con 100Hz y los resultados son bastante notables.

Una vez tenemos el hardware (tarjeta gráfica con soporte estéreo, dongle y pantalla) necesitamos los drivers correspondientes para soportar dicha funcionalidad.

ATI añade esta funcionalidad dentro de su paquete Catalyst. Nvidia, por su parte, requiere los drivers de la tarjeta gráfica, y otros para añadir la funcionalidad 3D.

Una vez instalados los controladores gráficos, tendremos que configurarlos, ya que cuando ejecutemos un programa no lo veremos automáticamente en 3D. A la hora de configurarlos se elige el método que se desea usar: anaglifo, polarizado, etc. Para eso, el programa tiene que estar adaptado y ejecutándose a pantalla completa. Ejemplos de juegos a los que es posible jugar son Age of Empires, Half-life o Need for Speed [22].

Una vez ejecutemos el programa, no hará falta nada más que colocarse las gafas. Los drivers se encargan de sacar por pantalla los gráficos de tal forma que el espectador no tenga que programar ningún algoritmo específico.

1.2 Adquisición estéreo

La funcionalidad estéreo no es algo que sólo el cerebro pueda interpretar. Hasta ahora la adquisición de datos para análisis se hacía siempre con una cámara.

En el campo del reconocimiento facial, por ejemplo, se extraían las características de unas imágenes patrón mediante un algoritmo PCA [23].

Para jugar a consolas, como la PlayStation2, se utilizaba una cámara EyeToy, figura 9. Ésta comprobaba el tamaño de los objeto para determinar si se acercaban o alejaban.

Técnicas para ayudar a la conducción, mediante una cámara situada en la parte delantera, o retos como el Grand Challenge [24] donde un vehículo autónomo debe recorrer el país sin incidencias, eran pruebas para estas técnicas de visión monocular.

En los últimos tiempos se están utilizando métodos de adquisición de datos estéreos con dos, o más cámaras, para reconocimiento facial [25] o reconocimiento de terreno para guiar vehículos. Incluso hay multitud de software para reconstrucción de escenarios a programas CAD mediante cámaras [26].

Figura 9: Eye toy con consola PlaySatation2.

1.3 Futuro

El área de los gráficos 3D esta muy ligado al mundo de los videojuegos y también del cine. Desde hace más de 20 años se proyectan películas grabadas con dos cámaras. Estas son procesadas y distribuidas en diferentes formatos. En el caso de su distribución por Internet, por ejemplo, y su compresión en diferentes formatos, hay que tener en cuenta que ambos ojos reciben una información muy parecida. Por lo tanto, el tamaño no variaría espectacularmente, pudiendo gozar de la calidad de siempre con el efecto 3D.

Como se ha comentado IMAX usa en sus cines polarizado pasivo y activo, y las películas son posteriormente distribuidas en sistema anaglifo, aunque también existen adaptadores para verlas usando gafas polarizadas activas.

Actualmente nos encontramos con cadenas de televisión que empiezan a hacer sus primeras pruebas en 3D. Los japoneses disponen de canales de emisión en 3D como Nippon BS Broadcasting. Los americanos no andan lejos, hacen series y las emiten en la Superbowl, evento de máxima audiencia, como Chuck. En el caso de los japoneses usando gafas polarizadas, en contra de las anaglíficas de la serie Chuck.

También se ha comentado que para la polarización hacen falta monitores especiales, con una tasa de refresco elevada, en el caso del polarizado activo. Hace unos 6 años se hablaba de televisiones CRT de 100hz. Estos salían al mercado con el gancho de que se reducía la estela que dejaban los objetos moviéndose a alta velocidad en la pantalla. Hasta hace poco más de 1 año no se hablaba de estas velocidades en monitores LCD, pero en un año, la tasa de refresco han pasado de ser 100hz a 200hz en pantalla LCD. Un ejemplo es Samsung, que incluye esta característica en algunos de los modelos de sus series 7, 8 y 9. Otro ejemplo es Sony, que acaba de anunciar el lanzamiento al mercado de monitores de 200hz, y próximamente de 240hz.

Es curioso el hecho de que la empresa creadora de la saga de video-consolas PlayStation, con su última creación PlayStation3 a la cabeza, anuncie este tipo de monitores cuando también planean sacar al mercado juegos en 3D, y teniendo en cuenta los requerimientos de éstos, no sería de extrañar que fueran el reclamo perfecto para que su salida al mercado aumentase.

Desde Sony no se han pronunciado sobre que sistema usaran, descartando anaglifo, aunque las primeras demostraciones se hicieron con gafas de polarizado pasivo.

Capítulo 2 Desarrollo de algoritmos estéreo

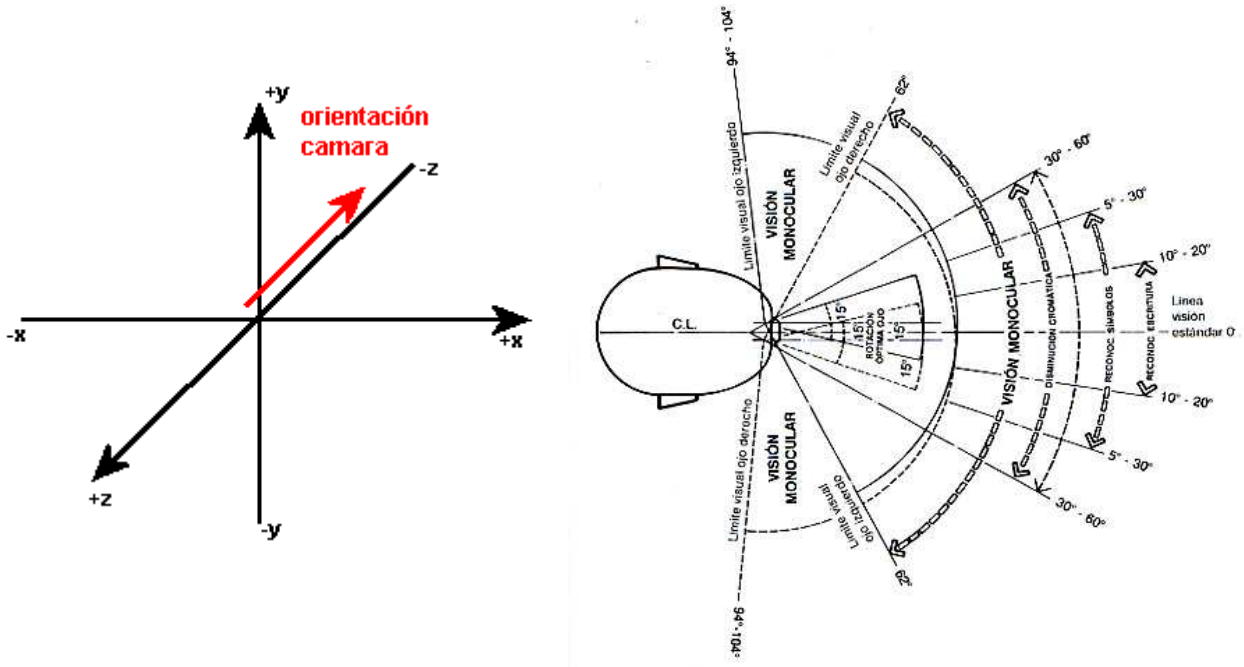
Partiendo del hecho que no todas las gráficas tienen soporte 3D, si se quisiera poder ver imágenes estéreo, haría falta programar el mismo algoritmo que hay en los controladores de la tarjeta gráfica. Los dos algoritmos estéreos que hay son el Toe-in y el Off-axis. Ambos algoritmos sacados del trabajo de Paul Bourke sobre el cálculo de pares estéreos [27].

Estos algoritmos indican como cada ojo recibirá la información sin importar el tipo de gafas o método usado. Es decir, cada algoritmo se puede implementar para unas gafas polarizadas, anaglíficas o monitores auto-estéreo.

Su funcionamiento se basa en doblar los objetos en pantalla. Cada uno de los objetos irá dedicado a un ojo concreto. Los ojos tienen una separación aproximada de 6 centímetros, pero cuando miramos un objeto, miramos con ambos ojos al mismo punto de ese objeto.

A la hora de pasar esto a un programa tenemos que tener en cuenta como emular que en una pantalla se pueda mostrar un objeto como si estuviéramos mirándolo desde dos sitios diferentes. Primero necesitamos crear una “caja” de visualización, es decir, un definir un espacio en coordenadas mundo (coordenadas cartesianas de 3 dimensiones) donde podamos ver todo lo que hay dentro. En la figura 10a tenemos una muestra de coordenadas cartesianas.

Un ojo crea un volumen de visualización en forma de cono, debido a la forma redonda de nuestra cornea, con un ángulo de visión grande, de casi 180º por ojo, figura 10b. La imagen llega al cristalino y se proyecta en la retina. Todo lo que haya fuera de ese cono no se ve.



a b

Figura 10: (a): Los ejes que se toman como punto de partida en un espacio de visualizado creado con OpenGL. (b): Grados alcanzado por el ojo humano.

Seguidamente explicaremos el funcionamiento de los 3 algoritmos estéreos implementados en la aplicación:

- Toe-in
- Off-axis
- Pulfrich

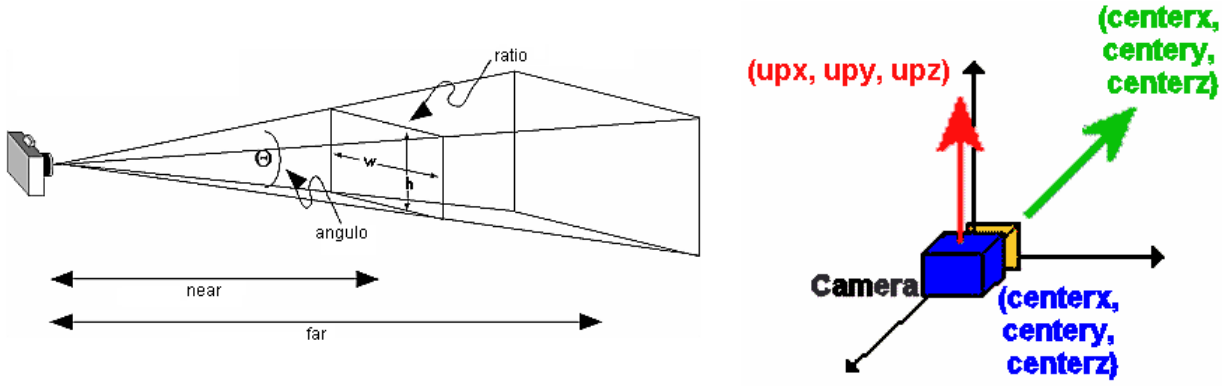
2.1 Toe-in

Para crear un espacio de visualización se utiliza la función gluPerspective [28]. Para definir ese espacio hace falta un ángulo de visión, generalmente 45 o 60 grados, ya que estamos delante de un monitor, similar a mirar a través de una ventana, y no es necesario visualizar objetos que estén muy a los lados.

También hace falta un ratio para proporcionar la imagen, al tamaño de la ventana. Como sabemos, hay diversos tipos de pantallas según la longitud de sus lados. Últimamente es común ver pantallas panorámicas, cuyos ratio es de 16/9, contra las primeras pantallas y más usuales que eran de 4/3. Ahora hay definida una pirámide de visualización que parte de un punto todavía no definido, pero tampoco hay definido un tope para dicha pirámide. Quedan dos parámetros, que servirán para cortar la pirámide, así no tendrá una longitud infinita donde introducir objetos. La figura 11a muestra como queda la pirámide de visualización.

Una vez tenemos que zona podremos ver, nos hará falta definir, dentro del nuestro mundo donde colocar tal zona y hacia donde tiene que ir orientada. Para esto se tienen que definir dos puntos de vista que serán nuestros ojos, y un punto final que es donde miraran.

En OpenGL se pueden definir tantos puntos de vista como se deseen. Para ello se usa la función gluLookAt, figura 11b. Esta requiere 3 parámetros, que definen la situación del punto de vista, u ojo, que son: el punto dentro del mundo, es decir, la coordenada cartesiana donde se situará nuestro ojo, el punto hacia donde mirará nuestro ojo, normalmente el origen de los ejes, y un vector, que indicaría la inclinación de nuestra cabeza, figura 12a.

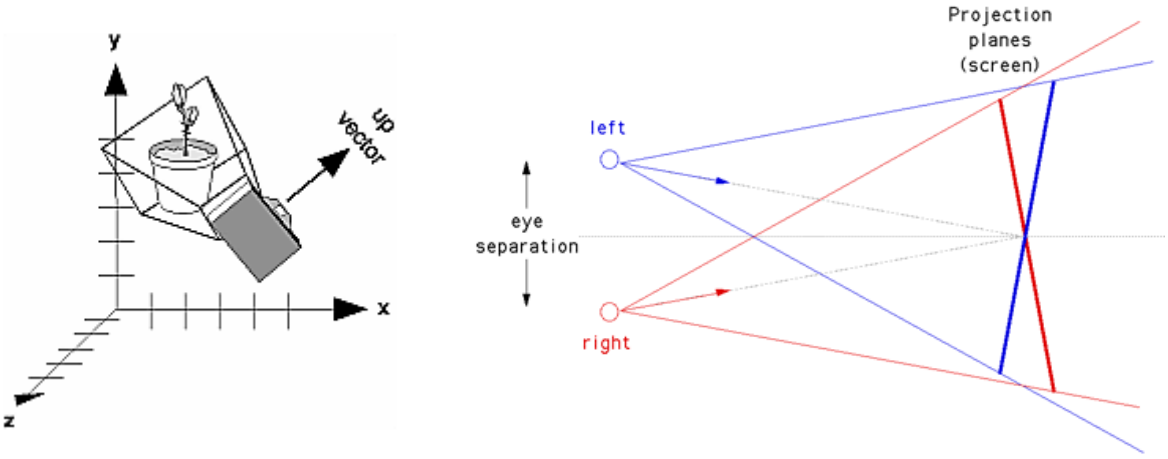


a b

Figura 11: (a): Creación de la pirámide según los 4 parámetros de la función gluPerspective. (b): Son necesarios 3 puntos para definir la posición y orientación de la cámara.

Los dos ojos se situarían a una distancia concreta y apuntarían al mismo punto, figura 12b, definiendo por igual el espacio de visualizado para cada ojo. Después de definir cada espacio y punto de vista, se tendría que visualizar el objeto en cada uno de los espacios.

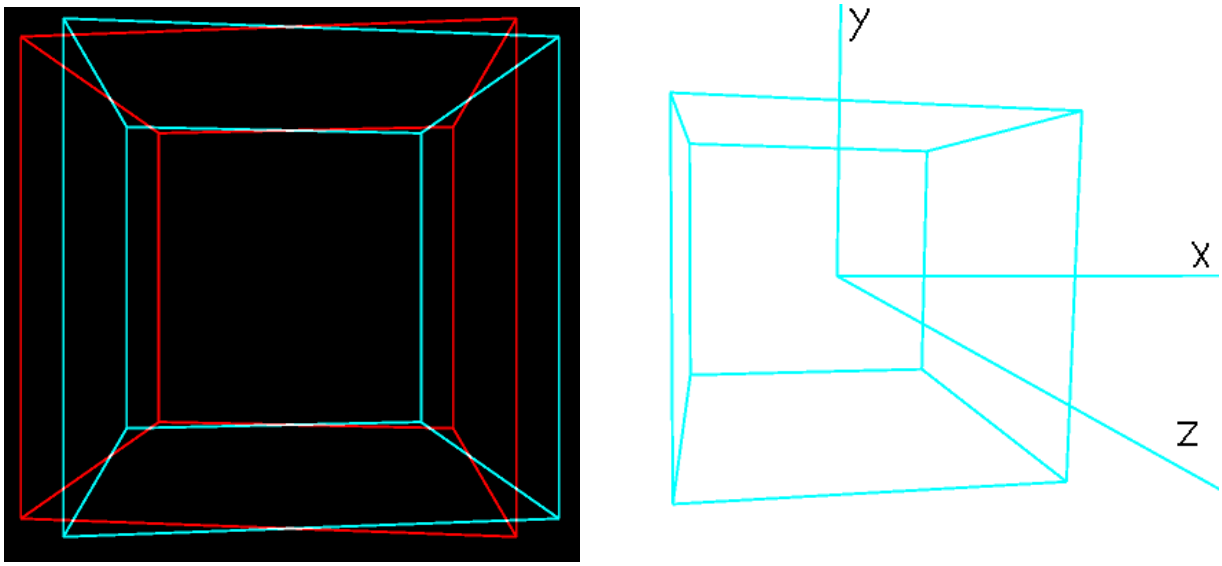
En el caso de gafas anaglíficas, cada objeto aparecería con el color que le toca a cada ojo. En el caso del formato americano, ojo izquierdo color rojo y ojo derecho color cyan. El formato europeo al revés. Si este método se utilizara bajo un polarizado, los objetos no tendrían que mostrarse bajo ningún filtro de color, por lo que se tendría que mostrar dos veces el mismo objeto.



a b

Figura 12: (a): Ejemplo de un vector director no paralelo al eje Y. La imagen aparecerá girada hacia la izquierda. (b): Los dos apuntan al mismo sitio, con una separación entre ellos.

El problema de este algoritmo, es que en la vida real los objetos no están perpendiculares a cada uno de nuestros ojos. Si estuviéramos mirando un cubo, como en la figura 13a, la cara del cubo que estuviéramos viendo sería perpendicular a cada ojo. En la figura 13b puede verse el efecto con un solo filtro. En la vida real eso no es así. Sólo existe un cubo, y como ambos ojos tienen que apuntar o mirar al mismo punto de ese cubo, es imposible que la línea que una ese punto con cada ojo sea perpendicular a la cara a la vez.



a b

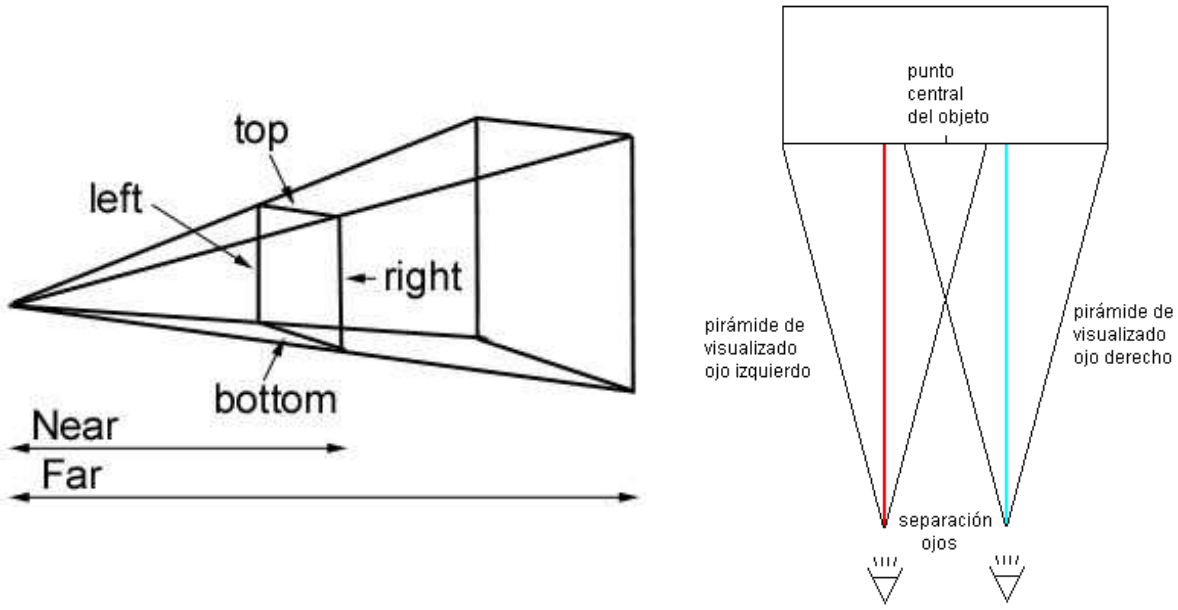
Figura 13: (a): Siguiendo el sistema anaglifo, el ojo derecho verá el cubo azul, y su superficie será perpendicular a la recta formada por el ojo y el punto de referencia, o centro de ejes. (b): Al acercarnos al objeto, éste tiende a girarse mucho, por lo

que la percepción 3D se pierde. El eje sirve como referencia para ver que nuestro ojo derecho no esta tan a la derecha y por tanto la imagen final sería errónea.

2.2 Off-axis

Esto no se puede conseguir con la función gluPerspective. OpenGL proporciona otra función con características similares, glFrustum[28]. Mediante los parámetros de esta función, podemos crear dos pirámides de visualizado, cuyo eje sea paralelo, y por tanto, ambas pirámides sean perpendiculares al objeto, como la figura 14a.

GlFrustum tiene 6 parámetros. Los 2 primeros, servirían para desplazar las áreas de visualizado hacia la derecha e izquierda. Los otros 4 parámetros servirán para crear la pirámide de visualizado, indicando altura y profundidad de la misma, como se muestra en la figura 14b.

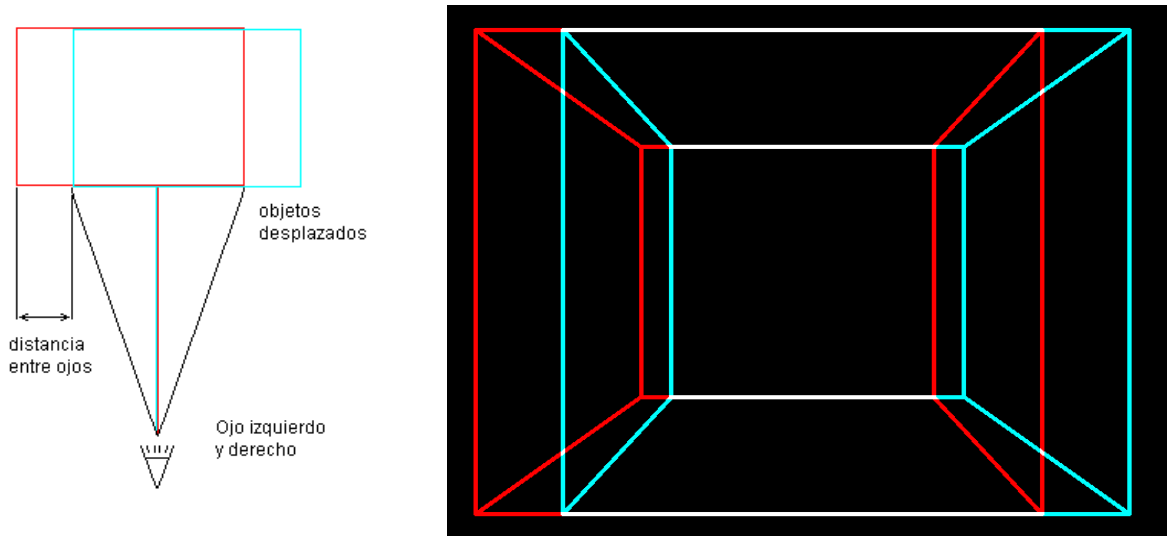


a b

Figura 14: (a): Los 6 parámetros necesarios para crear una pirámide de visualizado. (b): Dos pirámides paralelas, apuntando a diferentes partes del objeto.

Una vez creadas las áreas de visualizado, se tendrán que crear los puntos de vista. Usaremos la función gluLookAt como en el algoritmo anterior.

Si dejáramos los parámetros de esta función como antes, veríamos los objetos duplicados, y separados. ¿Por qué?. Porque el punto de referencia hacia donde apunta el ojo sigue el eje de la pirámide, por lo tanto, los dos ojos no estarían mirando al mismo punto, sino que mirarían en paralelo a puntos tan distantes como la distancia entre ojos. Como solución, sólo hay que desplazar lateralmente las pirámides, figura 6a. Así cada ojo apuntará al mismo punto del objeto, y la visualización será correcta, como la figura 6b.



a b

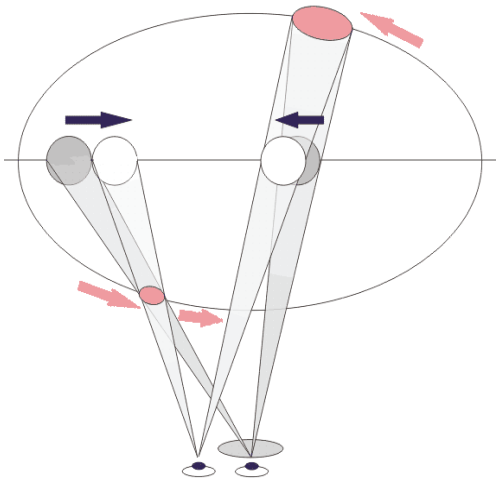
Figura 15: (a): Pirámides desplazadas la distancia entre ojos. (b): La visualización es correcta, contrasta con la figura 13a que se ve incorrectamente.

2.3 Pulfrich

Este método no tiene nada que ver con los otros dos. Las gafas usadas contienen dos lentes, una transparente y la otra posee una tonalidad más oscura.

Los objetos que queremos ver en 3D tienen que estar en movimiento. Así, dentro de una escena, los objetos que estén fijos no tendrán ningún efecto tridimensional. Los objetos que queremos ver en 3D tienen que estar en movimiento. Así, dentro de una escena, los objetos que estén fijos no tendrán ningún efecto tridimensional.

La luz reflejada por un objeto llegará más tarde al ojo que tenga el filtro oscuro. Así cuando un objeto viaje de izquierda a derecha, el ojo con el filtro oscuro verá el objeto más a la izquierda que el ojo con el filtro transparente, como muestra la figura 16a. Si el filtro oscuro está en el ojo derecho, como las gafas utilizadas durante el proyecto, dará la sensación de que objeto sobresalga de la pantalla. En el caso de que el objeto fuera en sentido opuesto, será la sensación de lejanía la que tengamos.

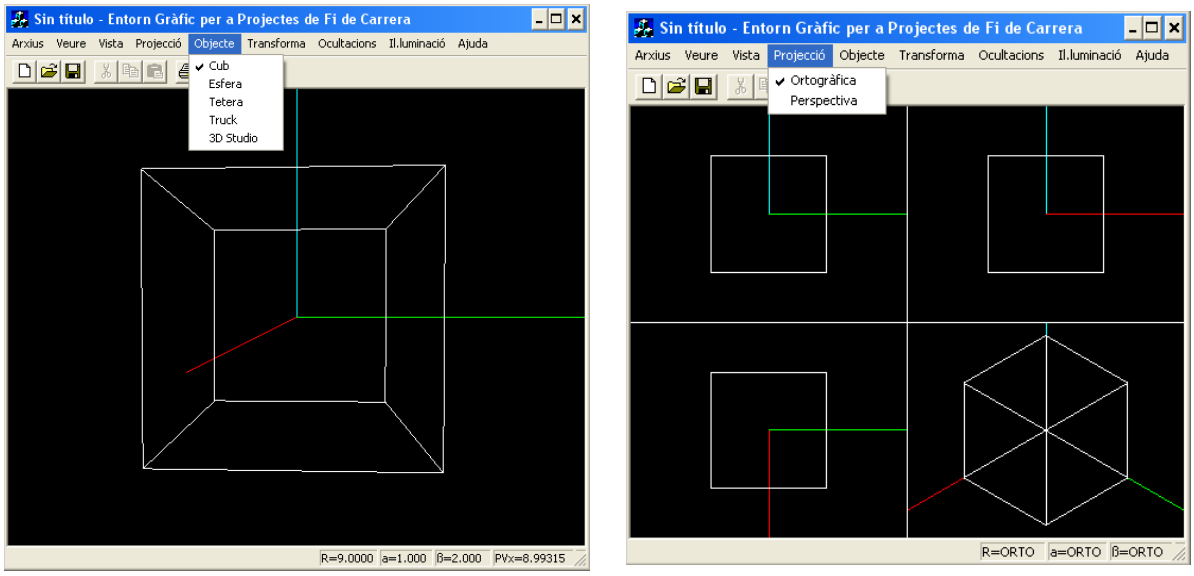


a

Figura 16: (a): Efecto tridimensional de un objeto dependiendo del movimiento.

2.4 Desarrollo del entorno

La idea principal de este proyecto era modificar el entorno gráfico usado en la asignatura de "Gràfics per Computador". Éste contiene un menú superior, donde aparecen las los objetos a escoger, figura 17a, y las vistas, figura 17b, entre más opciones. Pero después de varios intentos, no se pudo encontrar una compatibilidad con los controladores estéreos de la tarjeta gráfica. Por tanto se decidió hacer un entorno nuevo, que debía contar con las mismas funcionalidades que el entorno usado en la asignatura.



a b

Figura 17: (a): Muestra un cubo y los diferentes objetos predeterminados disponibles. (b): Vista Ortogonal de un cubo, y el menú con los dos tipos de visualización.

Los puntos en los que se divide el desarrollo del entorno de visualización estéreo se compone de los siguientes puntos:

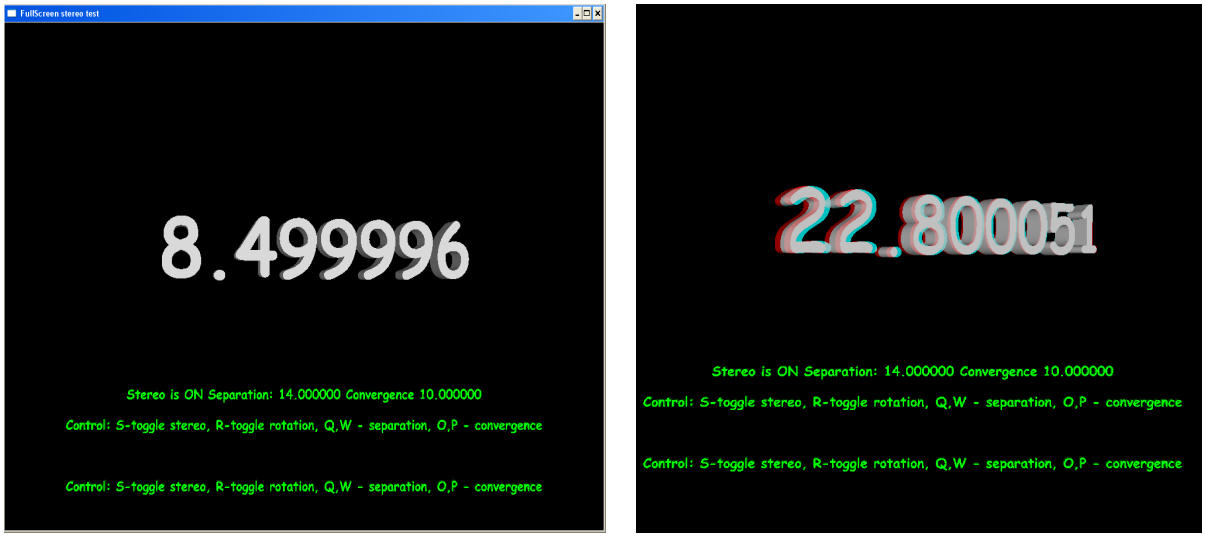
- Entorno de programación
- API gráfica
- Programación del entorno gráfico

2.4.1 Entorno de programación

Buscando información por la red, encontramos un entorno extremadamente sencillo, que destacaba por tener implementada la compatibilidad con la opción estéreo y la funcionalidad de pantalla completa, necesaria para que se pueda activar la funcionalidad 3D de la tarjeta gráfica, ya que un programa que no pueda ejecutarse a pantalla completa no podrá usarla.

Una vez ejecutado el programa, éste mostraba por pantalla un contador en 3D, figuras 2a y 2b, que giraba sobre si mismo. No tenía implementado movimiento de cámara, ni menús. Estaba programado con el Entorno de Desarrollo Visual Studio 2003 .NET. Como el entorno de la asignatura está programado en Visual Studio 2005, se tuvo que adaptar a este nuevo

entorno. De todas formas el entorno no acababa de funcionar bien. A la hora de finalizar el programa daba error, y algunos mensajes de alerta bloqueaban el programa.



a b

Figura 18: (a): Como el programa no se ejecuta a pantalla completa, no se ve el efecto tridimensional. (b): Programa ejecutándose a pantalla completa.

2.4.2 API gráfica

Para desarrollar programas con entornos gráficos hay varias API's. Las dos más usadas son OpenGL [29] y Direct3D [30]. Esta última, creada por Microsoft para el desarrollo de aplicaciones multimedia junto a otras API's [31], y generalmente usada para la programación de videojuegos. Para este proyecto se usó OpenGL dado que el entorno de la asignatura, del cual se toman funcionalidades, está programado en OpenGL, y también permite a los alumnos usar sistemas operativos diferentes de Microsoft Windows, dado que OpenGL es multiplataforma, al contrario que Direct3D, por lo que el código escrito durante el proyecto, podrá ser usado para proyectos futuro en GNU/Linux y MAC.

2.4.3 Programación del entorno gráfico

Igual que en el entorno de la asignatura, se crearon diferentes proyecciones para ver los objetos. En el entorno original estaban las proyecciones: ortográfica y perspectiva. En la primera se ve el objeto desde diferentes puntos, lo que en dibujo se conoce como vista de alzado, perfil y planta, además de la isométrica. La pantalla se divide en 4 partes iguales. Con el fin de poder girar un objeto y verse desde los 4 puntos de vista. Como añadido a la proyección perspectiva, se decidió que según se pulsaran las flechas del teclado se moviera el objeto dibujado. Estos movimientos también son funcionales en la proyección perspectiva, aunque usando tal proyección se puede mover el ratón, pulsando el botón izquierdo, para mover la cámara. Para poder rotar la cámara alrededor de los objetos se definieron dos variables según nos fuéramos moviendo hacia la derecha o izquierda y otra si queríamos subir o bajar, para ver el objeto desde abajo. Por lo tanto, trabajábamos bajo coordenadas esféricas [32]. Nos falta la longitud del radio, la distancia a la que se encontrarán nuestros ojos del centro de los objetos a visualizar, como se puede ver en la figura 19.

Cada vez que se moviera el ratón con el botón izquierdo apretado se calcularía la nueva posición de los ojos, el objeto permanecería inmóvil, aunque parecería que se moviera. Con el botón derecho del ratón se controlaría el zoom, si la cámara se acerca al objeto o se aleja.

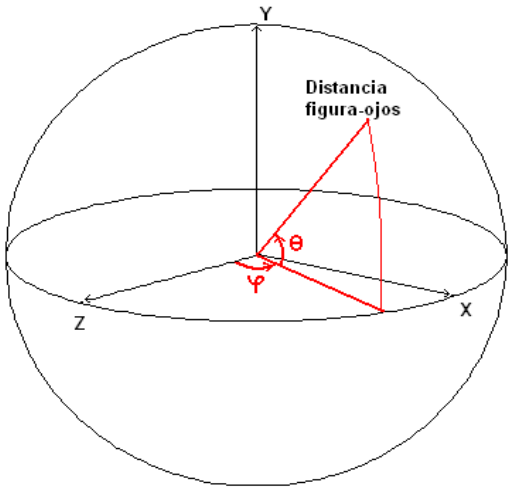


Figura 19: Coordenadas esféricas, donde a través de los dos ángulos y una distancia se puede posicionar la cámara en cualquier punto

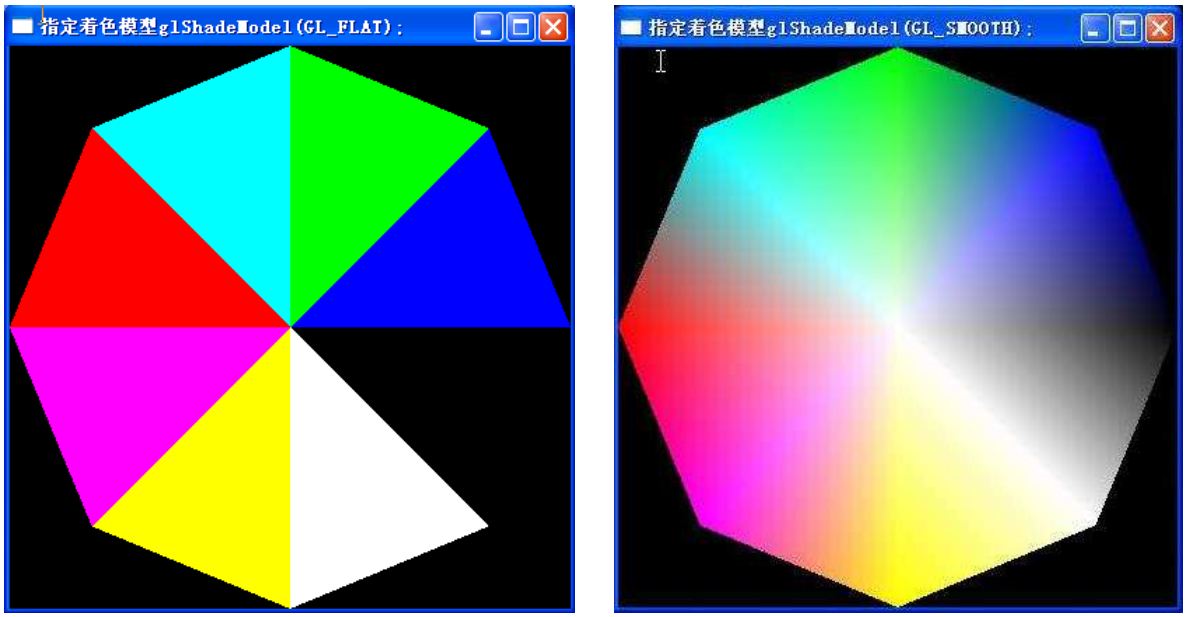
Para cambiar de objetos se utiliza el teclado, en el que cada número representa un objeto. El entorno original también puede leer archivos 3Ds, creados por el programa Autodesk 3D Studio MAX [33], desarrollado por la empresa Autodesk Inc. La principal idea era aprovechar el código del entorno de la asignatura, pero éste se basaba en librerías MFC [34], incompatibles con el entorno Win32 [35] que se estaba programando. Así que hubo que buscar unas funciones que no dependieran de librerías MFC. Se encontró un proyecto que usaba unos archivos similares a los que se buscaban para leer archivos 3Ds [36].

Los archivos 3Ds contienen uno o varios objetos a tamaño real, así que puede que al leer un archivo, este no quepa en el área de visualización. Como se ha comentado antes, con el botón derecho del ratón nos podemos alejar o acercar, pero también se ha implementado una función para ajustar el objeto a la pantalla. Además se puede tratar de un escenario muy grande o simplemente que el objeto almacenado en el archivo no este centrado, esta función la centraría.

También hay varios modelos de dibujado, esto se refiere a la forma que un objeto se dibuja en pantalla. Un objeto se puede dibujar mostrando solo las aristas que conforman su estructura, los vértices que las unen o las caras que forman las aristas. A esto hay que añadirle el tipo de iluminación de las caras. Estas pueden iluminarse teniendo en cuenta la normal de la cara, por lo que en una cara solo habría un color, o teniendo en cuenta la normal de los vértices, interpolando así las intensidades y colores de cada vértice. Una iluminación basada en vértices sería la correspondiente a la figura 20a, y otra basada en interpolación la figura 20b.

Volviendo al movimiento de la cámara, ésta no sólo debe moverse alrededor del objeto, alejándose o acercándose, sino que, como si de un circuito se tratase, poder ir a derecha e izquierda, arriba o abajo. Esto nos da la posibilidad de poder recorrer el escenario creado para un juego, por ejemplo.

Poder elegir entre un dibujo y otro, diferentes modos de visualizado e iluminación por teclado no deja de ser algo engorroso, si uno no esta familiarizado. En el entorno original la rueda del ratón hace la misma función que el botón derecho. Se decidió, que con tal de dar la misma funcionalidad al teclado que al ratón se usara este botón para hacer aparecer un menú.



a b

Figura 20: (a): Para dar color a los diferentes triángulos se coge el color de uno de los vértices. (b): Cada píxel de los triángulos son coloreados interpolando las intensidades de los vértices [38]

Como en el programa de la asignatura, en este menú se podría escoger entre las diferentes figuras, visualizado, etc.

Como las pantallas se suelen utilizar de forma horizontal, y cada vez son más usuales las pantallas panorámicas, se optó por crear un menú lateral, para tener más área de visualizado, ya que un menú superior dejaría una zona muy estrecha para ver objetos. Figura 21.

El lateral derecho se dividió en 10 zonas, cada una para un objeto diferente, objetos cuya equivalencia con el teclado es del número 1 al 0. Empezando por unos simples ejes, pasando por un cubo, conos, icosaedro, hasta unas figuras más complejas, como un nudo o un atractor de Lorenz [38].

Para poder elegir diferentes vistas, se usó el lateral izquierdo. También se dividió en 10 zonas. Aunque ambos menús compartan tamaño y variables para simplificar, estos pueden ser independientes muy fácilmente sin complicar el código para nada.

La primera opción es la vista perspectiva. Es la opción por defecto, igual que el dibujado de ejes, que será la primera figura que aparezca al ejecutar el programa. Debajo está la visualización ortográfica. Como los menús se han hecho transparentes, se puede ver parte del objeto debajo de los menús. Figura 22. Es la vista en la que más se puede ver esta característica. De todas formas, en caso de no gustar se puede quitar dicha transparencia o incluso pulsando el botón central del ratón (o rueda), hacer desaparecer el menú, para aprovechar todo el área del monitor.

Debajo de la opción ortográfica se puede elegir entre cualquiera de los dos algoritmos estéreo implementados. Estos, al ser independientes de la tarjeta gráfica, no necesitan que el programa esté ejecutándose a pantalla completa para poder apreciar el efecto 3D.

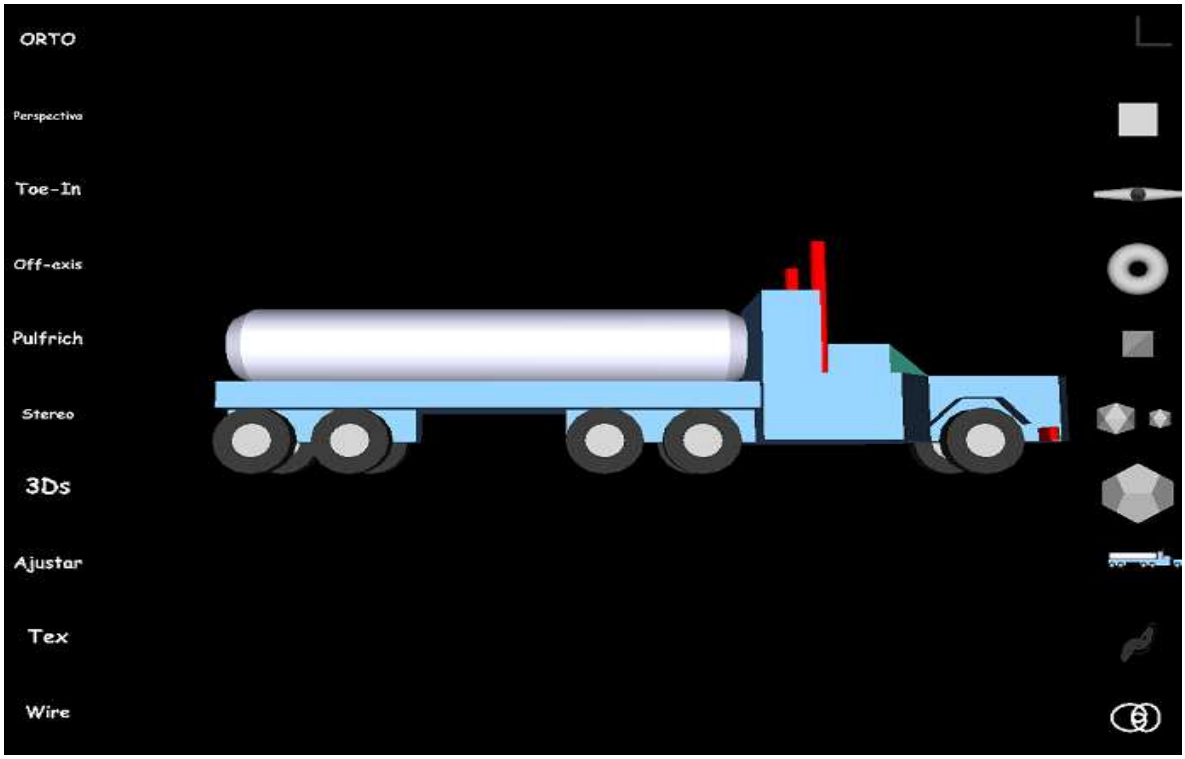


Figura 21: Muestra la aplicación ejecutándose a pantalla completa. A la izquierda diferentes formas de visualización. A la derecha objetos diversos.

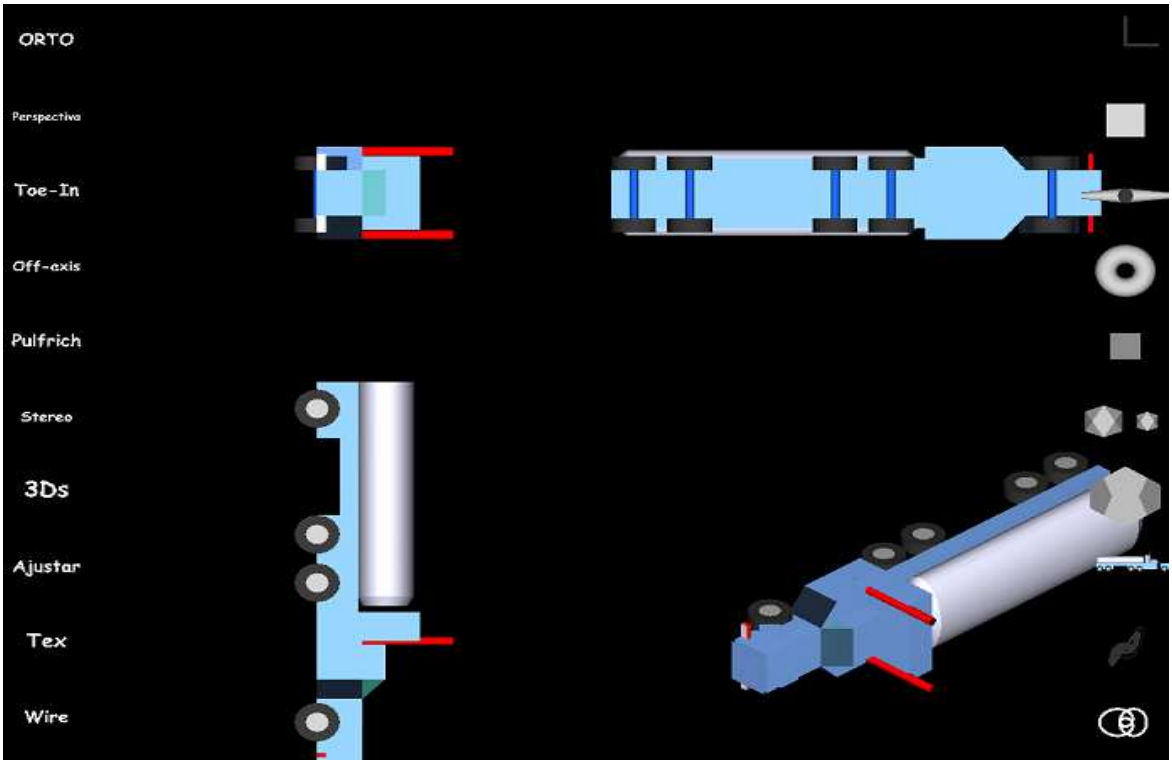


Figura 22: Proyección ortográfica. Se puede observar como la cisterna del camión queda parcialmente tapada por un dodecaedro.

Como muestran las figuras anteriores, debajo de la opción Off-axis, está la vista “Pulflich”. Para ver el efecto 3D con esta opción habrá que cambiar de gafas. Es un efecto que solo es perceptible con escenas en movimiento. El objeto se empequeñecerá y empezara a moverse de un lado a otro de la pantalla.

También podemos deshabilitar la opción estéreo, pero sólo cuando esta puesta la primera opción, es decir, estéreo a través de gráfica.

El siguiente botón abre un cuadro de diálogo, donde se podrá elegir un archivo 3Ds para ser visualizado. Cada vez que se pulse este botón se puede elegir visualizar un objeto diferente. Si se cancela, continuará el objeto elegido anteriormente. La siguiente opción servirá para ajustarlo al tamaño de la ventana, tanto si esta a pantalla completa como si no.

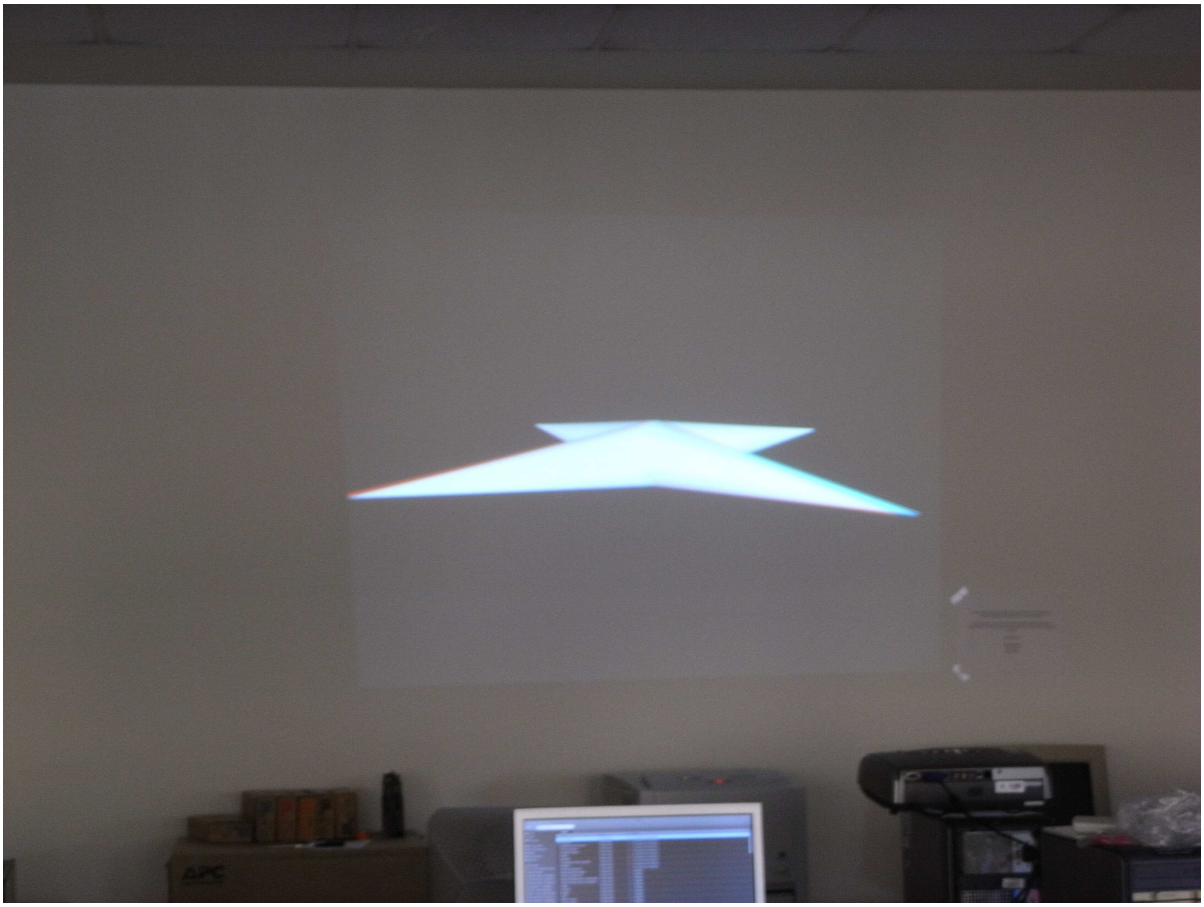
Mediante el botón de debajo se accede otro cuadro de diálogo para elegir una textura. Esta tiene que ser un archivo en formato de mapa de bits. Así un simple cubo puede parecer un trozo de madera o metal.

El siguiente botón selecciona el modo de iluminación: aristas, caras planas o color según interpolación de vértices, como se ha comentado anteriormente.

Otras funciones que no aparecen en los menús pero que son accesibles por teclado son por ejemplo la opción de cambiar el color de fondo, en lugar del negro por defecto, pasar a blanco, donde también se denota el efecto 3D. Todas estas funcionalidades quedan reflejadas en el manual de usuario, anexo III.

Capítulo 3 Resumen

Para evaluar los algoritmos implementados se optó por hacer un juego de pruebas. La prueba consistía en mirar una pared blanca, figura 23, con gafas anaglíficas, donde se proyectaban diferentes objetos. La amplitud del proyector rondaría los 3 metros de diagonal, o unas 118 pulgadas, como la mostrada en la figura x, y la pared se visualizaría desde varias distancias. Posteriormente se hicieron las mismas pruebas en una pantalla CRT de 19 pulgadas.



23

Seguidamente enumeraremos las partes en las que se divide este apartado:

- Complejidad
- Medios de visualización
- Pruebas
- Incidencias

3.1 Complejidad

Una de las partes fundamentales del proyecto, era comparar dos algoritmos de visualización estéreo. Uno, a priori, más correcto y complejo (Off-axis), y otro que presenta peor visión (Toe-in), pero más sencillo computacionalmente.

El hecho de calcular los dos puntos de referencia, en lugar de uno, hacía pensar más requerimientos computacionales, pero a la hora de la verdad no es así. Haciendo cálculos ambos presentan muchas similitudes, y solo se diferencian de unas pocas operaciones, las diferencias de las cuales se detallan a continuación:

Toe-in: 1 división + 2 multiplicaciones

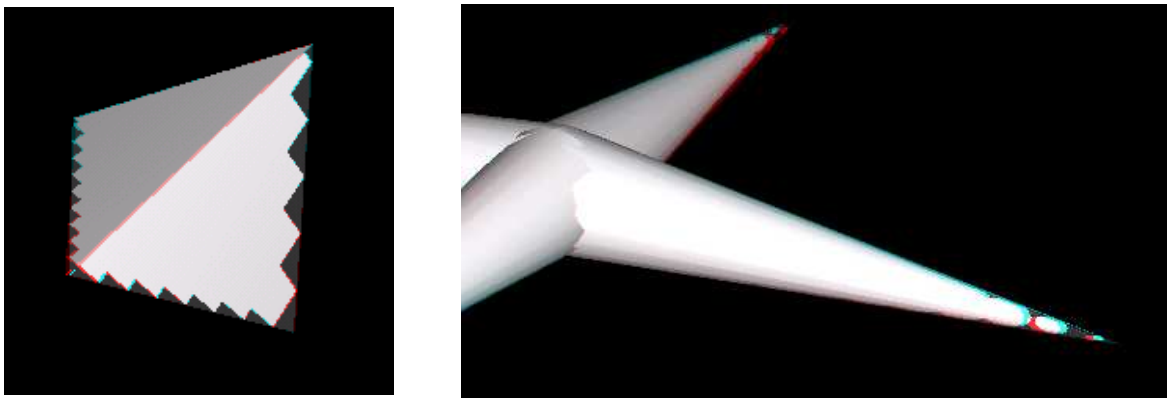
Off-axis: 2 cosenos + 2 senos + 2 sumas + 1 resta

El coste de las operaciones trigonométricas es mayor que las aritméticas, solo que no tanto como para disgregar estos algoritmos en dos grupos computacionalmente alejados.

3.2 Medios de visualización

Al observar detalladamente, uno se ve correctamente y el otro no, eso es verdad, pero, ¿en qué situaciones? Consultando con el grupo empleado para testear la aplicación, no notaron ninguna diferencia. Esto es debido a que estas diferencias se notan ligeramente en los laterales de los objetos, y van aumentando según nos acercamos a ellos o si tuviéramos una pantalla de tamaño considerable, donde unos pocos milímetros en una pantalla de 19 pulgadas se convirtieran en centímetros.

Aun estando cerca, el error apenas es visible. También hay que tener en cuenta que el tipo de proyector usado [39] no tenía una capacidad lumínica destacable y la no muy escasa iluminación del lugar también evitaba un contraste blanco-negro que daría más realismo. En un lugar completamente cerrado, una pantalla gigante y con un proyector más potente tal vez podría distinguiste algún defecto. Cambiando el color de fondo a blanco se perdía bastante la percepción 3D, sobretodo proyectado en la pared.



a b

Figura 24: (a): Los bordes del tetraedro no se ven correctamente, muestra la cara opuesta, cuando esta tendría que estar oculta. (b): Las puntas del cuádruple cono, y la intersección de los 4 conos sufren problemas similares.

Usando otro proyector [40] sin tanta profundidad de color, 16 bits, nos dimos cuenta que la vista 3D anaglífica mediante el driver de Nvidia contiene defectos, mientras que los algoritmos implementados, no. Una vez visto mediante el proyector, se probó en la pantalla,

bajando también la profundidad a 16 bits y se comprobó que ocurría lo mismo. Estos defectos tienen como base el cálculo de profundidad, y afectan las esquinas e intersección de los objetos, como se puede ver en las figuras 24a y 24b.

3.3 Pruebas

Los objetos mostrados en las pruebas consistían en diferentes figuras creadas a partir de directivas OpenGL, como cubos, conos o icosaedros. Estos permanecerían quietos, rotando sobre si mismos, o con una trayectoria que sobresaliera de la pantalla y volviera hacia atrás. No se expuso a nadie más de 20 minutos con las gafas anaglíficas, eso si, necesitaron unos segundos después de quitárselas, para percibir bien los colores. La muestra fue tomada por 10 personas, las opiniones de las cuales eran muy similares.

También se hizo el mismo tipo de pruebas en un monitor de ordenador de 19 pulgadas, pero la espectacularidad se pierde, ya que al no ser tan grande la pantalla el objeto no sobresale tanto, ni se ve tan grande, por el contrario mejora el contraste.

La única pega que sacaron del sistema anaglifo era que el color azul es ligeramente perceptible con el filtro rojo, por lo que, a pesar de no tenerse que ver, continua viéndose creando una molesta sombra. Otro inconveniente ocurría cuando se hacía sobresalir mucho un objeto de la pantalla. Esto en el ordenador representa una separación excesiva de dos objetos, que el cerebro no interpreta correctamente, como se puede ver en la figura 25. Esta separación significa que el objeto sobresale más de la pantalla de lo que nuestros ojos son capaces de ver, por lo que no logra interpretarlo.

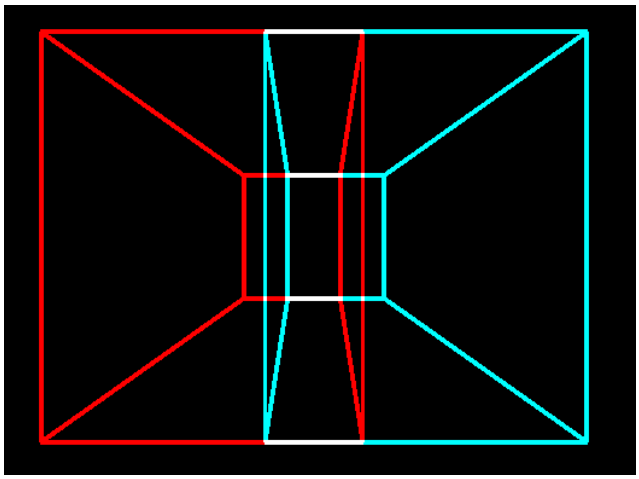


Figura 25: Separación excesiva entre canales de color.

Con las pruebas realizadas se puede lograr una salida de pantalla de entre un 60 y un 70% de la distancia entre la pantalla y el jugador, sin que éste pierda la percepción 3D. En cambio, usando las gafas polarizadas se conseguía salir más de la pantalla.

Dado que también se implementó el método ”pulfrich”, se hizo un test para probar si realmente se obtiene una percepción 3D. La mayoría de las personas aseguraron no ver nada, excepto algunas que veían cierta curvatura hacía el fondo de la pantalla, sin acabar de ver la trayectoria ovalada. Como no se ha encontrado ninguna especificación sobre la velocidad a la que tiene que moverse el objeto, se probaron varias velocidades, consiguiendo, con las más

lentas, mejor percepción. Esto puede ser debido al efecto de ghosting que creaba el objeto, dado que se trataba de una figura blanca bajo un fondo negro.

Independientemente del algoritmo usado, y centrándonos en el método de visualizado, todas las personas coincidían en que las gafas polarizadas sincronizadas con el monitor, daban una calidad de visión mucho mejor. Incluso después de estar un tiempo con ellas y quitárselas, no experimentaban la misma sensación de perdida de color temporal que con el sistema anaglifo.

El resto de tecnologías como la auto-estéreo o el polarizado pasivo no se han podido probar dado que es material no disponible por la universidad.

3.4 Incidencias

Los principales problemas encontrados a según avanzaba el proyecto quedan recogidos a continuación:

- Uno de los principales problemas a la hora de empezar este proyecto fue la incompatibilidad de la aplicación MFC de la asignatura “Grafics per computador” para incorporar el efecto 3D a través de la gráfica. Una de las cosas que había que tener en cuenta era que el fenómeno 3D solo ocurre cuando la aplicación está a pantalla completa. Como esto no se puedo conseguir, se optó por hacer otro entorno con la consecuente tarea de programar las mismas funciones que el otro programa tenía.
- Se encontró un entorno en la web: www.gali-3D.com, dedicada a reproducción de imágenes 3D a través de monitores y gafas. El entorno estaba editado en Visual Studio 2003 .NET y daba problemas a la hora de cerrar el programa, debido a diferentes liberados de memoria que se producían, igual que al cambiar de pantalla completa a ventana. Se tuvieron que solventar estos problemas previamente antes de continuar con el proyecto en si. El entorno de desarrollo tenía que ser VS2005, como el de la asignatura.
- Cuando se juntaban las dos imágenes de cada ojo, doblando el número de objetos a mostrar, el test de profundidad actuaba y uno de los dos objetos no se podía ver bien. Para esto había que limpiar el buffer de profundidad utilizando la función glClear(GL_DEPTH_BUFFER_BIT).
- Dentro del entorno de la asignatura, había la posibilidad de leer archivos 3Ds. Estos contiene escenarios u objetos creados con una herramienta de CAD, como 3DStudio MAX. Los utilizados por el entorno original daban errores al compilar. Se encontró una página web donde estaba el código de los dos archivos. Al incluirlos al proyecto daban errores de link, o en según que circunstancias no se podía compilar, debido a unas dependencias con librerías relacionadas con dispositivos móviles. Se tuvieron que incluir mediante la directiva “#include” pero sin añadirlos dentro del explorador de soluciones.
- Otro problema surgió al crear el menú. Las diferentes partes o botones se generaban como si de una vista ortográfica se tratara. Empezando por abajo y subiendo a la hora de ir generando opciones del menú. Cada opción estaba acotada a una porción de pantalla, un 10% vertical y horizontal, para que se redimensionase automáticamente al pasar a ventana. Y estas porciones estaban por encima de los objetos de pantalla. El problema era que la última opción no dejaba ver un el 10% superior de la pantalla, como la figura 26. Para quitar este defecto, se creo una opción más fuera de la pantalla, con lo que si se ocultaba algo, esta quedaría fuera de nuestra área de visión. Una vez creado el menú de la derecha esto se solventó. Con lo que ya se pudo quitar el “parche”.

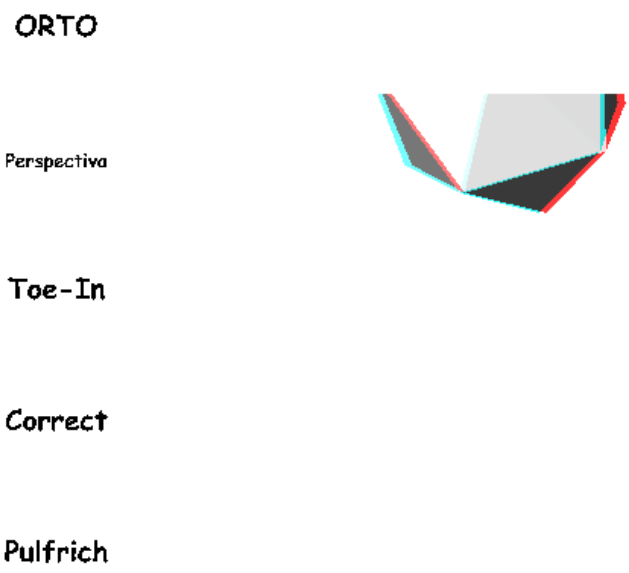


Figura 26: La opción “ORTO” tapa la parte superior del icosaedro.

- Se quería mejorar la forma de guardar imágenes extraídas del programa con el botón “PrintScreen” creando una ventana donde escribir el nombre del archivo. Figura 27. Debido al diseño del algoritmo usado para extraer tales imágenes no se puedo hacer, ya que en la pantalla quedaban restos del cuadro de dialogo. También un contador que fuera renombrando archivos, pero se creaban varios iguales, debido al tiempo de pulsado del botón. Así que los archivos se guardarán con el nombre de “PantallaCompleta.bmp”. El código con cuadro de dialogo está comentado para posible línea de continuación.

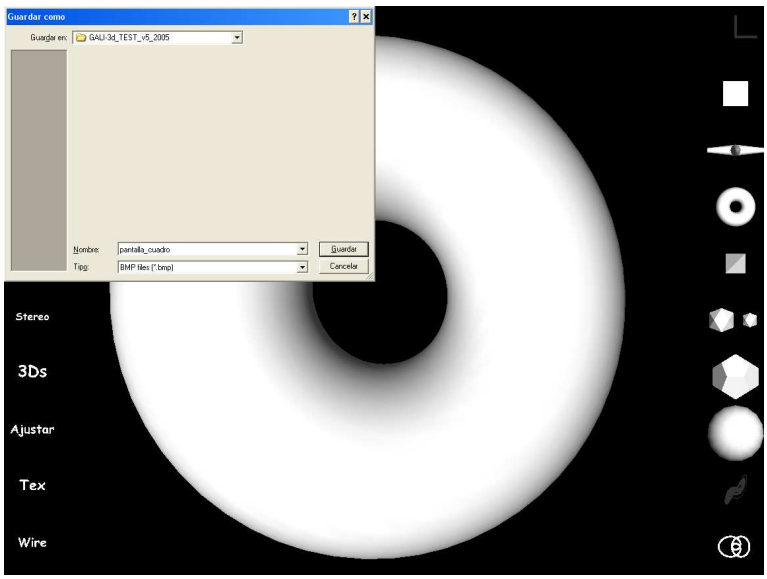


Figura 27: Al presionar el botón guardar, se guarda todo lo que hay en pantalla, incluyendo el cuadro de diálogo.

Como mejoras se podrían destacar algunas mejoras y líneas de continuación:

- Se intento programar un juego similar al Pong [41] usando la funcionalidad 3D, pero no hubo tiempo. Hay hecho un escenario creado con una bola rebotando. También se logró saber las coordenadas de la pelota en todo momento y el ángulo con el que saldría al ser golpeada. Faltaría la inteligencia artificial de la máquina.
- Se podrían colocar diferentes luces para el menú y la escena principal, dado que al movernos, también se mueve el punto de luz, por lo que los objetos del menú se oscurece, siguiendo el haz de luz. Si nos movemos mucho, los menús pasarán de estar blancos a un gris oscuro,
- Cambiar las funcionalidades de las teclas por similitud, ya que la letra ‘N’ no se relaciona fácilmente con Toe-in, ni ‘V’ con Pulfrich. Así como la letra ‘Q’ con iluminación Gouraud[37].
- Averiguar como funcionan las salidas de la tarjeta gráfica para sincronizarse con el dongle. Así se podrían usar estos dos algoritmos bajo cualquiera de los métodos mencionados, como gafas de cristal líquido o monitores auto-estéreos.
- Mejorar la eficiencia del código, ya que se repiten muchas operaciones innecesariamente

Capitulo 4 Conclusiones

Como conclusiones más destacables podemos nombrar las siguientes:

- Se ha adaptado un entorno gráfico añadiendo las funcionalidades que tiene la aplicación usada en la asignatura.
- Se ha creado un menú con las funcionalidades básicas y fácilmente ampliable.
- Se ha probado la ejecución a pantalla completa, pudiendo elegir resolución, profundidad de color y frecuencia de refresco de pantalla.
- Se ha hecho una prueba con 10 personas con resultados notables.
- El entorno es funcional y robusto, por lo que se puede usar como base para nuevas aplicaciones.

Las incidencias más importantes se comentan a continuación:

- El retraso producido por tener que subsanar los errores de programación de un entorno gráfico ajeno.
- Los problemas de link y dependencias de diferente código fuente hallado para leer archivos 3Ds.
- El ocultamiento de parte de la pantalla al generar el menú lateral.

Como mejoras y líneas futuras destacaremos las siguientes:

- Eliminar operaciones innecesarias.
- Programación de un juego o aplicación que aproveche la visión estéreo.
- Mejorar la forma de interactuar con el teclado, más intuitivo.

Anexo I Configuración tarjeta gráfica

Como se ha comentado anteriormente, para este proyecto se ha usado una tarjeta gráfica Nvidia 7600GT. Para poder visualizar programas en 3D no solo es necesario instalar los drivers correspondientes, si no que también configurarlos.

En este anexo haremos un breve manual para configurar la tarjeta gráfica y veremos algunas de las opciones más relevantes.

Dados los problemas insalvables encontrados durante el desarrollo del proyecto con los drivers de nuestra tarjeta gráfica, escogeremos unos controladores concretos que han funcionado a la perfección. Hay que tener en cuenta los diferentes componentes de cada uno, y seguir las indicaciones dependiendo de estos.

Primero hay que ir a la página Web de Nvidia, sección “Controladores archivados y beta”. Aparecerán unos desplegables para que elijamos la tarjeta gráfica y el sistema operativo que se usará. En este caso, Nvidia GeForce serie 7, familia 7600GT y sistema operativo Windows XP.

Debajo aparecerá una lista de diferentes controladores, como en la figura 28. Para este proyecto se ha elegido la versión 94.24, que, actualmente, es la recomendada por Nvidia, a pesar de ser unos drivers de hace ya casi 2 años. Así que se recomienda descargar, en caso de disponer de otro hardware, la versión recomendada en caso que los controladores que se posean no funcionen correctamente.

Figura 28

La instalación es totalmente automática, no pide instalar nada, ni ninguna configuración previa. Solo pide desactivar el antivirus.

En el caso de Nvidia también es necesario descargarse los controladores 3D. Para ello, iremos a la sección “Descarga los controladores”. Abajo del todo aparecen varias opciones entre la que se encuentra “GeForce 3D Stereo Controladores”.

En esta página web se pueden ver una serie de especificaciones y una lista de juegos compatibles, figura 29.

Más abajo se encuentra el controlador. En este caso se ha elegido la versión 91.31, con más de dos años de antigüedad, dado su buen funcionamiento. Su instalación tampoco tiene más complicación que tener que reiniciar para que los cambios surjan efecto.

Figura 29

Una vez tenemos todo instalado, sólo habrá que configurar su funcionamiento. Si pulsamos el botón derecho en el escritorio, nos aparecerá una nueva entrada llamada “Pantalla de NVIDIA” dentro de la cual aparece el nombre de nuestro monitor. Si pulsamos iremos a parar a la configuración de la tarjeta gráfica, tal como se muestra en la figura 30.

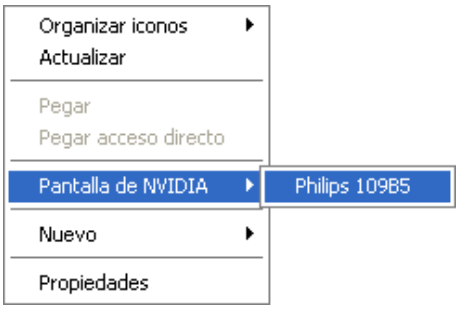


Figura 30

Una vez dentro puede aparecer un botón que ponga “Iniciar panel de control” con lo que nos saldría en la figura 31.

En el árbol de la izquierda escogemos la opción “Stereo Properties”. A la derecha nos aparecerá algo similar a lo que hay en el dibujo. La primera opción serviría para activar o desactivar la visión estéreo, o incluso activarlo por teclado. Y el desplegable de debajo para

escoger el sistema estéreo que deseemos utilizar. Haremos un breve resumen de las opciones que nos da Nvidia a la hora de conectar diferentes aparatos (monitores, gafas, etc.) en sus tarjetas gráficas.

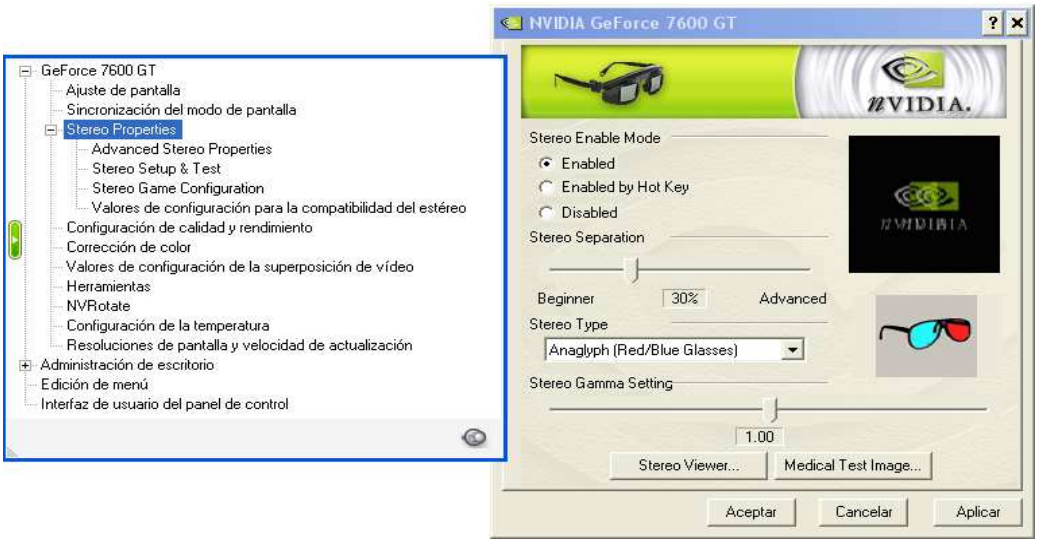


Figura 31

DDC VGA glasses: Gafas de polarizado activo. Se sincronizan a través de un dongle con la conexión Jack de la gafas. Precio aproximado no superior a 60€. Requiere un monitor CRT con una tasa de refresco elevada.

Dual VGA Output: La tarjeta gráfica tiene dos salidas diferentes. Cada una sacará la imagen de cada ojo. No necesita un hardware concreto, y el precio dependerá del monitor que se coloque en cada salida.

Vertical Interleaved: Monitores 3D auto-estéreo. Se conectan a la gráfica directamente. Sólo requiere que la persona que mire la pantalla no se mueva. Existen algunos con cámara que detecta la posición de la persona y mueve el monitor según el movimiento de la persona. Normalmente sólo para uso individual, pero existen variantes más caras para varias personas. Precio oscilante, dependiendo de las variantes, entre 3.000 y 25.000€.

Anaglyph: Gafas anaglíficas, generalmente rojo-cyan. Se pueden configurar los colores, según las gafas que se disponga en ese momento (verde-magenta). Existen gafas desechables, cuyo precio no supera los 2€.

Sharp 3D Display: Monitor 3D auto-estéreo. LCD de un solo visor. Ronda los 1.000€.

Dual VGA with Mirror up/down: Monitor de polarizado pasivo. Dos monitores colocados perpendicularmente, uno encima del otro, cada uno con un polarizado diferente. Un espejo transparente refleja el monitor superior dejando ver también el inferior. Las gafas seleccionan que ojo ve cada imagen, según el polarizado. El precio varia del tamaño de la pantalla. Aproximadamente 3.000€ si es de 19”, y 10.000€ una pantalla de 42”. Requieren gafas polarizadas, también existen desechables.

Planar mirror: Polarizado pasivo similar al anterior.

NeurOK front/back: Sistema parecido al anterior pero sin necesidad de reflectar una imagen. El propio monitor ya emite con dos polarizados. Solo se necesitan gafas de polarizado pasivo. El precio del monitor es realmente asequible, unos 300 € aproximadamente, pantalla de 22”.

Philips 3D: Monitor auto-estéreo. El precio varia según tamaño.

Direct 3-Pin DIN VESA Connect: Gafas de polarizado activo. Se conectan a través de dongle con conector 3-Pin Din o directamente a la gráfica si lo soporta. Precio y requerimientos igual que la primera opción.

Siguiendo con las opciones de configurado, en el panel de la izquierda nos encontramos "Advanced Stereo Properties" donde se puede configurar el activado estéreo por teclado y la separación de los ojos, para dar más sensación de acercamiento y alejamiento de los objetos que salgan por pantalla. Figura 32.

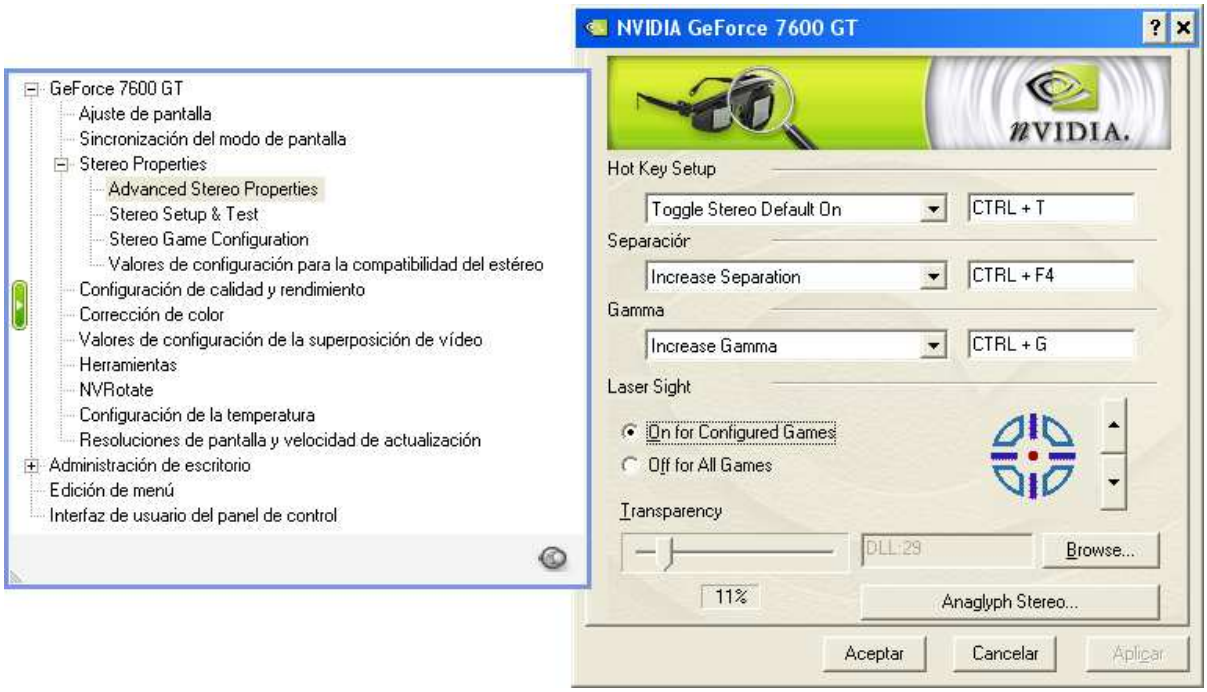


Figura 32

En la opción "Stereo setup & test" nos encontramos que configuración tendrá la pantalla al entrar en una aplicación que active la vista 3D, como por ejemplo un juego a pantalla completa. Cabe recordar que si se usan unas gafas de polarizado activo, se recomienda usar una tasa de refresco superior a 100hz. Pulsando en el botón "Launch Test Application…" se ejecutará una pequeña aplicación, donde puede verse un pequeño túnel y el logotipo y letras de Nvidia alejándose y acercándose, para comprobar el correcto funcionamiento del estéreo activado. Figura 33.

Siguiendo con el menú, figura 34 encontramos "Stereo Game Configuration" donde aparecen unos valores indicando la compatibilidad o mejor dicho, jugabilidad de diferentes juegos en un entorno 3D. También se pueden guardar ajustes para cada juego, según el usuario los vaya probando.

La última opción solo advierte de los requerimientos de memoria al usar la configuración estéreo, la figura 35 es una muestra. Recordemos que el 3D dobla los requerimientos tanto de búfer como de cálculos, ya que muestra los objetos por duplicado.

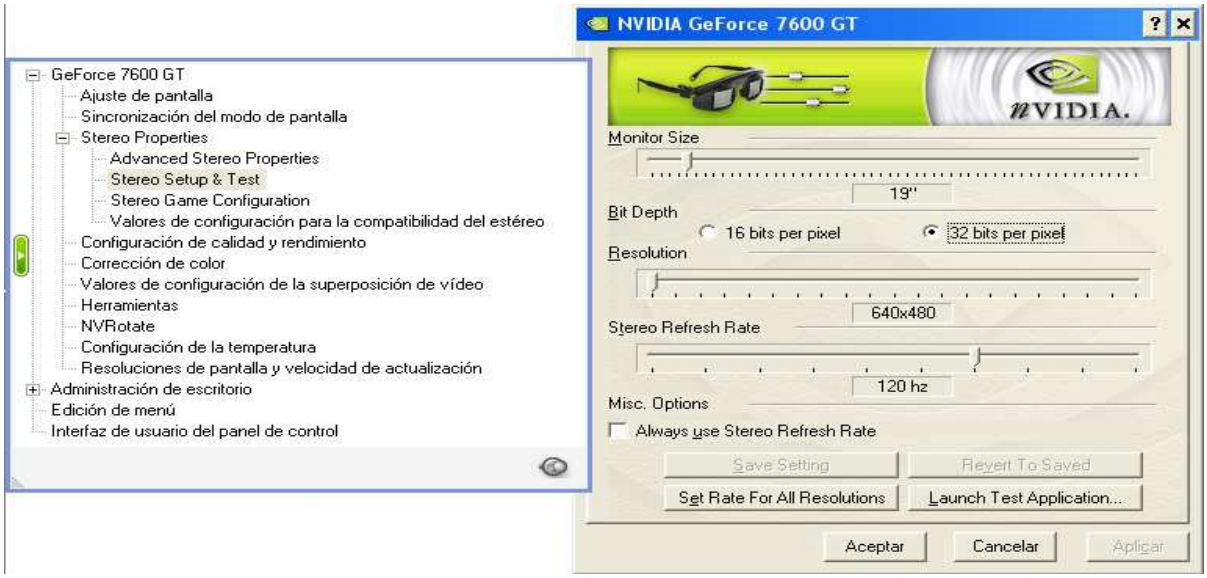


Figura 33

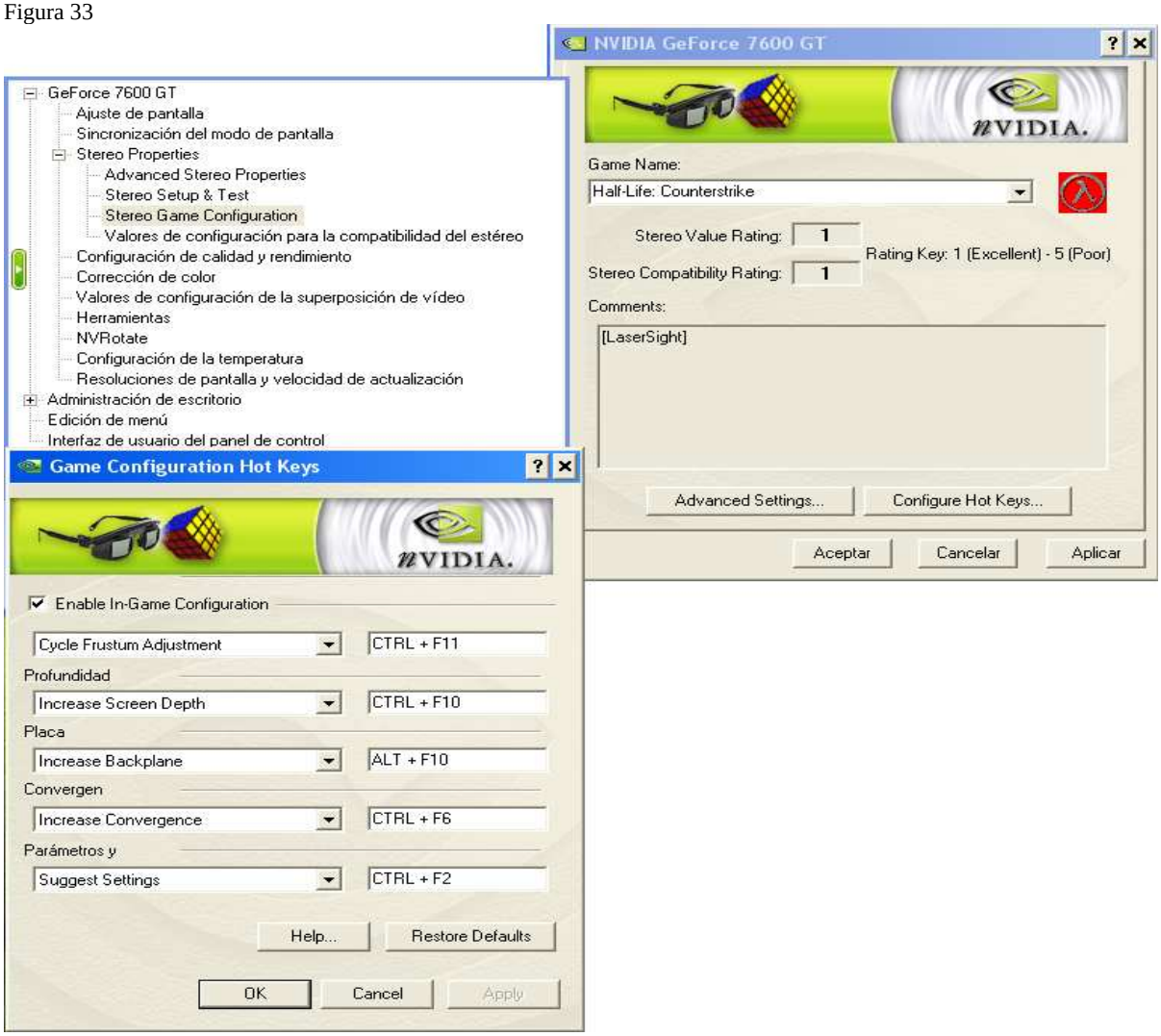


Figura 34

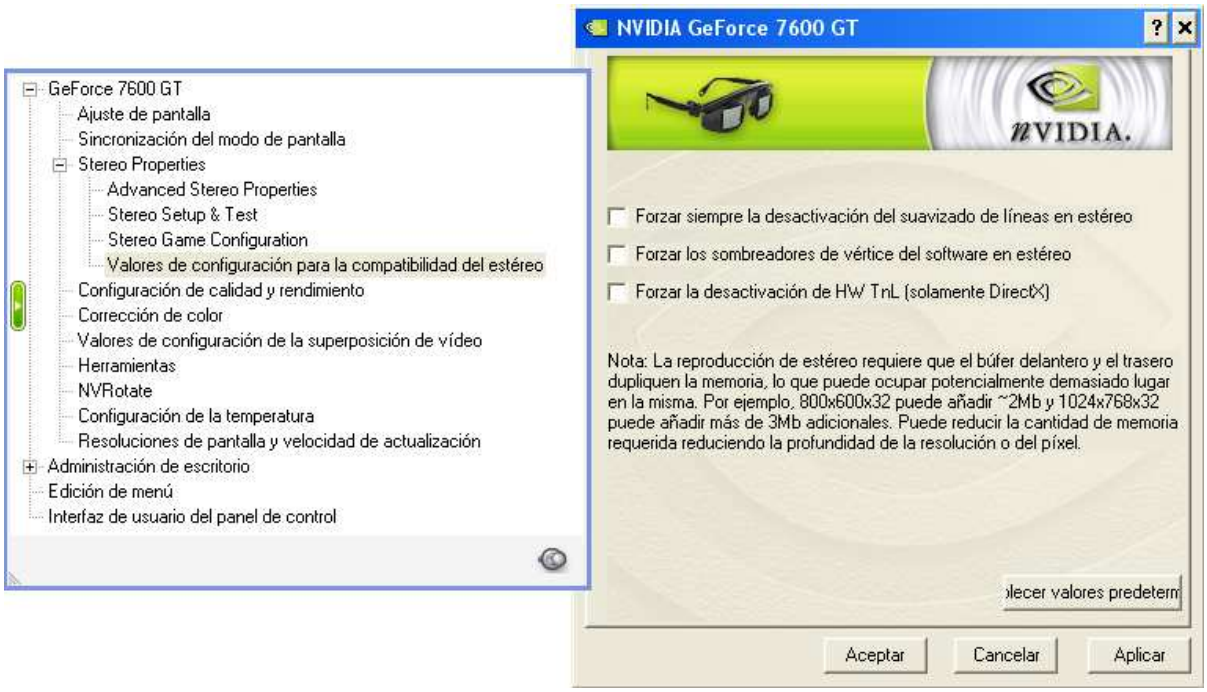

GeForce 7600 GT
Ajuste de pantalla
Sincronización del modo de pantalla
Stereo Properties
Advanced Stereo Properties
Stereo Setup & Test
Stereo Game Configuration
Valores de configuración para la compatibilidad del estéreo
Configuración de calidad y rendimiento
Corrección de color
Valores de configuración de la superposición de vídeo
Herramientas
NVRotate
Configuración de la temperatura
Resoluciones de pantalla y velocidad de actualización
Administración de escritorio
Edición de menú
Interfaz de usuario del panel de control
NVIDIA GeForce 7600 GT
NVIDIA.
Forzar siempre la desactivación del suavizado de líneas en estéreo
Forzar los sombreadores de vértice del software en estéreo
Forzar la desactivación de HW TnL (solamente DirectX)
Nota: La reproducción de estéreo requiere que el búfer delantero y el trasero dupliquen la memoria, lo que puede ocupar potencialmente demasiado lugar en la misma. Por ejemplo, 800x600x32 puede añadir ~2Mb y 1024x768x32 puede añadir más de 3Mb adicionales. Puede reducir la cantidad de memoria requerida reduciendo la profundidad de la resolución o del píxel.
lecer valores predeterm
Aceptar
Cancelar
Aplicar

Figura 35

Anexo II Cálculos para crear pares estéreos

Las operaciones necesarias para calcular la localización de las cámaras dentro del área de visualización no difieren mucho entre los dos algoritmos, Toe-in y Off-axis, ya que ambos comparten cálculos en común.

Estas operaciones serán las necesarias para hallar los parámetros de la función gluLookAt. Primero de todo haremos los cálculos como si de una sola cámara se tratase. Incluyendo en dicho cálculo el vector director. Luego haremos el desplazamiento para cada ojo y el punto de referencia.

Para mover la cámara se utilizan dos ángulos, rotax para el movimiento horizontal, y rotay, para el moviendo vertical.

El vector director se calcula cada vez que se modifica la posición de la cámara, pero no cuando se hace un zoom. Este vector equivaldría a la inclinación de nuestra cabeza en el mundo real. Así, en la posición inicial, nuestra cabeza permanecería recta, siguiendo el vector (0, 1, 0), y los valores de rotax y rotay serían 0.

```
director.x=-sin(rotax)*sin(rotay);
director.y=cos(rotay);
director.z=-cos(rotax)*sin(rotay);
```

El cálculo de la posición del ojo si que se calcula tanto si movemos la cámara como si hacemos zoom. La posición inicial del ojo sera el (0, 0, 10), por tanto el ojo estará sobre el eje Z. La variable zoom inicialmente valdrá 1, y se incrementará según nos alejemos y se decrementará según nos acerquemos al centro de nuestra caja de visualización.

```
ojo.x=(zoom+9)*cos(rotay)*sin(rotax);
ojo.y=(zoom+9)*sin(rotay);
ojo.z=(zoom+9)*cos(rotay)*cos(rotax);
```

En el caso del algoritmo Toe-in, el punto de referencia será el vector punto hacía donde mira nuestra cámara. La variable focal, será la distancia que habrá siempre entre el punto de referencia y nuestro ojo. Por tanto, estos cálculos servirán para mantener esa distancia.

```
punto_ref_st.x=(zoom*(focal-dist))*(-ojo.x/dist);
punto_ref_st.y=(zoom*(focal-dist))*(-ojo.y/dist);
punto_ref_st.z=(zoom*(focal-dist))*(-ojo.z/dist);
```

Según nos vayamos alejando o acercando, el punto de referencia seguirá a la misma distancia de nuestros ojos.

Una vez tenemos todos estos vectores, solo nos faltará crear los dos ojos a partir del ojo inicialmente creado. La coordenada Y que representa la altura, tendría que ser la misma para ambos ojos, ya que nuestro visor no podra inclinar la cabeza hacia los lados.

```
ojoi.x=(ojo.x*zoom)-distancia_ojos*cos(rotax);
ojoi.y=ojo.y*zoom;
ojoi.z=(ojo.z*zoom)+distancia_ojos*sin(rotax);


ojod.x=(ojo.x*zoom)+distancia_ojos*cos(rotax);
ojod.y=ojo.y*zoom;
```

```
ojod.z=(ojo.z*zoom)-distancia_ojos*sin(rotax);
```

Para crear la pirámide de visualización se usará para los dos ojos la misma función:

```
gluPerspective(45.0f,(GLfloat)m_w/(GLfloat)m_h,0.1f,550.0f);
```

En el caso del método Off-axis, el cálculo de los ojo es el mismo, igual que el vector director. Este método necesita el cálculo de dos puntos de referencia, y una función para el pirámide de visualización diferente.

Los puntos de referencia se centrarán en el centro, con la misma separación entre ellos que los ojos. Esto creará dos líneas paralelas entre los ojos y sus puntos de referencia respectivos.

```
punto_ref_st_i.x=-distancia_ojos*cos(rotax);
punto_ref_st_i.y=0;
punto_ref_st_i.z=+distancia_ojos*sin(rotax);


punto_ref_st_d.x=+distancia_ojos*cos(rotax);
punto_ref_st_d.y=0;
punto_ref_st_d.z=-distancia_ojos*sin(rotax);
```

Para conseguir las dos pirámides de visualización se ha hecho uso de la funcio glFrustum, con la cual se pueden conseguir piramides con el eje central paralelo. Restandoles un valor conseguimos desplazar lateralmente las dos áreas de visualizado.

```
glFrustum(left+0.001314,right+0.001314,bottom,top,0.1f,550.0f);
glFrustum(left-0.001314,right-0.001314,bottom,top,0.1f,550.0f);
```

Anexo III Manual de Usuario

Como se ha comentado anteriormente, a través del ratón se pueden acceder a varias funciones de la aplicación. Pulsando el botón central del ratón se accede a dicho menú. Todas esas funciones también son accesibles desde teclado.

Los números corresponderían a cada una de las figuras que aparecen en el menú. La tecla número 1, serían los ejes, que están en la parte superior del menú. La tecla 2, el cubo, y así sucesivamente hasta llegar al nudo, que es la tecla 0.

Siguiendo el menú, especificaremos las letras para acceder a las diferentes proyecciones, que son las siguientes:

P: Estéreo a través de tarjeta Nvidia.

N: Toe-in

B: Off-Axis

V: Pulfrich

M: Proyección ortográfica desde 4 puntos

O: Proyección ortográfica, vista alzado, toda la pantalla

Para quitar el efecto estéreo de la tarjeta se usa la tecla ‘S’. En el entorno original, el contador existente rotaba sobre si mismo, esta funcionalidad se ha dejado, pudiéndose activar con la tecla ‘R’. Si queremos cambiar el sentido de la rotación, pulsando la tecla ‘D’ lo cambiaremos.

Con la tecla ‘F’ abriremos un archivo 3Ds, mientras que con la tecla ‘X’ se abrirá el cuadro de diálogo para elegir una textura. Cabe decir que la textura se aplicara al objeto que tengamos en pantalla. La tecla ‘A’ servirá para ajustar el archivo 3Ds a la pantalla.

Si queremos cambiar el tipo de iluminación, iluminación Gouraud, filamentos o caras planas, usaremos las teclas ‘Q’,’W’ y ’E’ respectivamente.

Para poder extraer imágenes de figuras en 3D fácilmente del programa, también se añadió un botón del teclado para salvar el estado de la pantalla. Así si alguien está visualizando algún objeto y quiere guardarlo lo podrá hacer fácilmente, sin tener que recurrir a programas de dibujo, ni tener que salir de la pantalla completa a través de teclado pulsando F1. Mediante la tecla ‘Print Screen’ se salvará a un archivo llamado “PantallaCompleta.bmp” lo que aparezca en el monitor en ese momento.

Mediante la tecla tabulador y shift, se puede cambiar la tonalidad del fondo de la pantalla. En algunas figuras, como el cuádruple cono o figura 3, también se observa un efecto 3D muy notable, mientras que con unos simples ejes o el cubo, no se distingue, ya que prácticamente todo es blanco. Para aclarar el fondo, pulsar ‘TAB’ y para oscurecerlo, Shift+’TAB’. Para que aparezcan los ejes en cualquier figura, pulsar la tecla ‘C’

Las flechas nos servirán para movernos como si de un juego se tratase, hacia delante, o hacia atrás, y hacia los lados. Para subir la tecla ‘Re pag.’ Y para bajar ‘Av Pag.’. También se puede mover el objeto mediante el teclado numérico, sin necesidad de mover la cámara. Las

teclas 7 y 9 harán girar el objeto sobre el eje Y mientras que las teclas 4 y 6 lo harán sobre el eje Z. Para girarlo sobre el eje X, se usaran las teclas 8 y 2.

Si pulsamos la tecla ‘I’, veremos la información en la pantalla de todas las variables usadas, ya sean para posicionar la cámara, o para ver que opciones están activadas, como la figura mostrada, la proyección o el color de fondo a través de sus 3 componentes RGB.

Para poder evaluar de forma más eficiente los algoritmos, se programó una serie de deformaciones de objetos, en las direcciones de los 3 ejes. Para ensanchar un objeto, no será necesario más que pulsar la tecla ‘Control’ y mover el ratón hacia un lado. Para hacerlo más alto, tecla ‘Control’ y dirigir el ratón hacia arriba. Si se desea hacer más profundo, se utilizara la tecla ‘Shift’, más el movimiento lateral del ratón.

Finalmente, la tecla ‘T’ reseteará los cambios realizados sobre el objeto tales como giros, deformaciones, puntos de vista de la cámara, el zoom y el color.

Referencias

[1] http://pronos.galeon.com/galeria_trampantojos.htm. Galeria con trampantojos

[2] http://es.slizone.com/page/slizone_learn.html. Página web de Nvidia, donde se informa de la tecnología SLI.

[3] http://ati.amd.com/technology/crossfire/demos/es/index.html. Información de ATI-AMD sobre CrossFire.

[4] http://www.tomshardware.com/reviews/nvidia-geforce-9800-gx2-review,1792-14.html. Información de consumo de diferentes gráficas de alto rendimiento en el mercado.

[5] http://www.codinghorror.com/blog/archives/000662.html. Consumo de varios componentes de un ordenador.

[6] http://bdigital.ulpgc.es/digital/visualizar/propiedad.php?accion=monografias&id=1232&vol=no#. Historia de la computación gráfica

[7] Motilal Agrawal, Kurt Konolige and Robert C. Bolles. Localization and Mapping for Autonomous Navigation in Outdoor Terrains : A Stereo Vision Approach. SRI International, Menlo Park, CA 94025

[8] http://sussy.wordpress.com/2007/09/03/trampantojos. Galeria de trampantojos

[9] http://es.wikipedia.org/wiki/Ilusi%C3%B3n_%C3%B3ptica. http://www.portalmix.com/efectos/dosenuno, Paginas dedicadas a las ilusiones ópticas

[10] http://en.wikipedia.org/wiki/Stereoscopy. Estereoscopio

[11] http://en.wikipedia.org/wiki/Augmented_reality. Realidad aumentada

[12] http://www.sid.org/chapters/uki/presentations/qinetiq.pdf, http://www.stereo3d.com/displays.htm#auto. http://www.microimages.com/documentation/cplates/71StereoMonitor.pdf. Tecnología auto-estereo

[13] http://en.wikipedia.org/wiki/Anaglyph_image. Imágenes anaglíficas

[14] http://www.galeon.com/stereochannel/html/body_anaglasses.html. Como fabricar unas gafas anaglíficas

[15] http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem. Teorema de Nyquist-Shannon.

[16] http://www.youtube.com/watch?v=P9KPJlA5yds. Realidad aumentada

[17] http://www.freepatentsonline.com/6529175.html. Patente sobre el funcionamiento del dongle

[18] http://www.asseenontv.com/prod-pages/x3d_extreme_3d_system.html. Requerimientos para usar las gafas.

[19] http://www.cinesonido.com/modules.php?name=Content&pa=showpage&pid=11. Funcionamiento de los cines IMAX

[20] http://miaex.com/index.php?main_page=product_info&cPath=7_8&products_id=19. Características ATI V7100.

[21] http://en.wikipedia.org/wiki/Cathode_ray_tube. Tubo de rayos catódicos.

[22] http://en.wikipedia.org/wiki/TFT_LCD. TFT-LCD.

[23] http://www.nvidia.com/object/GeForce_3D_Vision_3D_Games.html. Listado de juegos compatibles con funcionalidad 3D.

[24] http://www.face-rec.org/algorithms. Algoritmos para reconocimiento facial.

[25] http://www.darpa.mil/grandchallenge/index.asp. Grand Challenge.

[26] Naohide Uchida, Takuma Shibahara, Takafumi Aoki, 3D Face Recognition Using Passive Stereo Vision Graduate School of Information Sciences, Tohoku University Sendai-shi, Japan

Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel, Expression-Invariant 3D Face Recognition Technion – Israel Institute of Technology Department of Electrical Engineering, Haifa 32000, Israel

[27] Zach Christopher , Klaus Andreas, Reitinger Bernhard, Karner Konrad, Optimized Stereo Reconstruction Using 3D Graphics Hardware. Workshop of Vision, Modelling, and Visualization. 119-126, 2003 (VMV 2003)

[28] http://local.wasp.uwa.edu.au/~pbourke/miscellaneous/stereorender. Calculating Stereo Pairs, Paul Bourke

[29] D. Shreiner, M. Woo, J. Neider, T. Davis, OpenGL Programming Guide, 4th edition, Addison-Wesley Developers Press, 2004.

[30] http://www.opengl.org. Página web oficial de OpenGL.

[31] http://en.wikipedia.org/wiki/Direct3D. Información sobre Direct3D.

[32] http://msdn.microsoft.com/es-es/directx/default(en-us).aspx. Sitio de soporte para programación usando DirectX

[33] D. Gil, E. Martí, Visualització interactiva, asignatura Gràfics per computador, UAB, 2002.

[34] www.autodesk.es. Página web de Autodesk Inc.

[35] http://en.wikipedia.org/wiki/Microsoft_Foundation_Class_Library. Información sobre MFC

[36] http://en.wikipedia.org/wiki/Windows_API. Información sobre la API Win32

[37] http://www.geocities.com/valcoey/opengl_3ds/3ds.html. Página web con lector de ficheros 3Ds

[38] E. Martí. Tema 4 Il·luminació, asignatura Gràfics per computador, UAB, 2004. Método de Gouraud para intensidad en píxeles.

[39] http://ozviz.wasp.uwa.edu.au/~pbourke/texture_colour/anaglyph. Atractor de Lorenz y nudo, realizado por Paul Bourke

[40] http://www.panasonic.es/html/es_ES/348007/index.html#anker_348007. Monitor Panasonic PT-LC80E

[41] http://www.epson.es/cgi-bin/epson-es/showproductspecs.cgi?typo=web&search=V11H177040LA, Monitor Epson EMP-X3

[42] http://es.wikipedia.org/wiki/Pong, El juego Pong